



ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CESENA CAMPUS
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING - DISI
SECOND CYCLE DEGREE IN DIGITAL TRANSFORMATION MANAGEMENT
Class: LM-91

BIG DATA ANALYTICS IN THE TRAVEL ARRANGEMENT INDUSTRY

Graduation thesis in
Big Data and Cloud Platform

Supervisor
Prof. Matteo Golfarelli

Candidate
Mohamed Alie Kamara

Academic Year 2022 - 2023

Acknowledgement

I would like to express my deepest appreciation to Professor Matteo Golfarelli who made this thesis paper possible and to my family and friends whose love and support kept me going. You always have a special place in my heart.

Abstract

The Easy Market Company based in Rimini deals with online sale of flights and hotels. Online searches generate up to one billion transactions daily. Easy Market is creating a Big Data Analytics system with Google technology to extract value from this data. The company aims to design and implement a system which is expected to perform both batch and real-time analysis to optimize operational processes.

Table of Contents

Acknowledge.....	2
Abstract.....	3
List of figures.....	5
List of Tables.....	7
Abbreviations.....	8
1. Introduction.....	10
Proposal.....	10
Structure of the thesis.....	11
2. The flight optimization problem.....	13
2.1 Easy Market Business Model.....	13
2.2 The optimization problem in-depth.....	16
2.3 Architecture of the data flow in Easy Market system.....	19
2.4 What Easy Market want to optimize.....	20
3. Literature on Big Data and Google Cloud Platform Technology.....	22
3.1 What is Big Data.....	23
3.2 Big Data Technology.....	31
3.3 Apache Spark.....	40
3.4 Google cloud platform (GCP).....	43
3.5 Running Hadoop and Spark on GCP's DataProc.....	46
4. Automating Look-To-Book Computation.....	48
4.1 Structure of the raw data (sematic).....	49
4.2 Extract Transform Load (ETL) process.....	49
4.3 Cost of the computation.....	59
4.4 Proposed Solution.....	59
4.5 Average monthly cost of computation.....	60
5. Conclusion.....	62
6. References.....	64
7. Appendix.....	69

List of figures

Figure 1. Market share by search count.....	14
Figure 2. Easy Market Business Model Canvas.....	15
Figure 3. Booking on lol.travel.....	16
Figure 4. Easy Market booking system.....	17
Figure 5. Filter engines.....	18
Figure 6. Architecture of data flow in Easy Market’s ecosystem.....	20
Figure 7. Volume of data created in zettabytes.....	22
Figure 8. Size of the big data analytics market worldwide.....	23
Figure 9. Interest for the term Big Data.....	24
Figure 10. 3Vs of big data.....	27
Figure 11. Human and machine-generated data.....	28
Figure 12. Html code.....	29
Figure 13. Big Data lifecycle.....	29
Figure 14. Big Data architectural strategy.....	31
Figure 15. HDFS Architecture by ASF.....	32
Figure 16. MapReduce Architecture.....	33
Figure 17. Hadoop architecture comparison 1.0 and 2.0.....	34
Figure 18. NIST Big Data Reference Architecture.....	35
Figure 19. Components of a big data architecture.....	35
Figure 20. Batch processing.....	36
Figure 21. Real-time processing.....	37
Figure 22. Lambda architecture.....	38
Figure 23. Kappa architecture.....	38
Figure 24. Hadoop ecosystem.....	39
Figure 25. The Apache Spark Stack.....	40
Figure 26. Streaming, batches of input data	41
Figure 27. Service-oriented architecture.....	43
Figure 28. Easy Market GCP ecosystem.....	43
Figure 29. ETL process.....	50
Figure 30. Flights Star model	51
Figure 31. ETL warehouse schema	52

Figure 32. Python transformation script.....56
Figure 33. Metrics computation script.....58
Figure 34. Cloud Proposal architecture.....59

List of Tables

Table 1. Differences in the attributes of big data and RDBMS.....27

Abbreviations

ASF	=Apache Software Foundations
API	=Application Programming Interface
B2B	=Business to Business
B2C	=Business to Customer
BD	=Big Data
BI	=Business Intelligence
BigQ	=BigQuery
COVID-19	=Coronavirus 2019
DAG	=Directed acyclic graph
ELT	=Extract Load Transform
ETL	=Extract Transform Load
GCP	=Google Cloud Platform
HDFS	=Hadoop Distributed File System
IoT	=Internet of Things
JSON	=JavaScript Object Notation
L2B	=Look-to-Book Ratio
MSEs	=Metasearch Engines
NoSQL	=Not only SQL
OSS	=Operations Support System
OSP	=Open Source Project
OTAs	=Online Travel Agencies
PII	=Personal Identifiable Information
RDBMS	=Relational Database Management System
RDDs	=Resilient Distributed Datasets
SGI	=Silicon Graphics
XML	=Extensible Markup Language
YARN	=Yet Another Resource Negotiator

1. Introduction

At the heart of this paper is Easy Market a major player in Italy's travel arrangements industry since the early 2000s, it offers various travelling packages, some of which include hotels, car-rentals and flights at suitable rate from its suppliers. The advent of massive digitalization has made access to booking platforms such as lol.travel, and their competitive low cost flight packages easy to purchase, this has led to a boom in tourism and all the activities perform by customers/bots before, during and after the booking process has been generating massive amount of data that has spark the interest of Easy Market to use advanced data mining and business intelligence techniques in other to tap into this data and gain valuable insights about their customers' behaviour so they can better tailor their services to suit both customers and business needs. In this paper I will study big data and it effect on Easy Market's daily operation while attempting to propose a solution that will perform both batch and streaming analytics to help solve some of the problems Easy Market is facing in this era of big data.

The activities performed on Easy Market's system by ordinary customers (b2c), by bots or by business (b2b) generates data, and at the core of its system is an optimization problem that is too many queries are sent into the system for which results of flights are return and no booking is made, eventually leading to a high search low booking (low look-to-book ratio/ conversion rate). Easy Market aims to develop a system that will be able to calculate the look-to-book ratio, and automatically block unnecessary queries that might not lead to a booking.

Proposal

The proposal entails two solutions, one has to do with adjusting Easy Market business model while the other is developing a technical system. The first proposal is adjusting the business model by applying a limit/quota on how many shopping requests can be made for a specific route at a particular time, this will help mitigate the costs, but the downside is by putting a limit to the number of requests there is risk of losing some sales, so this might not be the most optimal solution. Another adjustment that can be made on the business model is by asking users to pay for excess usage of Easy Market's API, this can incentivize all users to be more responsible and be aware that they pay more by sending unnecessary queries. This can improve the performance of the system and help reduce the costs associated with high look-to-book ratios. Pricing models such as Pay-as-you-go, Subscription-based and Transaction-based can also be implemented.

The second proposal which is technical has to do with the real design of a cloud architecture on GCP this solution is just an addition to what Easy Market already has, this involves just adding vertex AI workbench and AlloyDB to the suite of products. This solution is discussed in detail in chapter 4 in the section “Proposed solution”. A snapshot of another technical solution for Hadoop and Spark savvy users which can be implemented using GCP Dataproc, can also be found on the appendix introduction, proposal.

The thesis is structured in seven different sections the first chapter which is the introduction, contains a high level overview of Easy Market’s optimization problem. The second chapter looks at the business model, value proposition, competitive advantages and position in the Italian travel industry and worldwide. This chapter also gives an in-depth explanation of the optimization problem with a visual representation of the “filter engines,” and a thorough look at the underlining logic of these tables. Chapter three covers the literature of big data concepts, evolution, characteristics, sources, type and life cycle of big data. This chapter also looks at one of the most popular big data technologies Apache Hadoop and its core components. Apache Spark a tool that can be used for various BD tasks including ML. Furthermore, I briefly discussed some of GCP’s products Easy Market is using and the ones mentioned in the proposal. Chapter four continues with the computational process, explaining details of the data semantics and architecture workflow of the python & SQL program that I wrote to perform the ETL and look-to-book-ratio. An estimate of the cost value in euros for performing the experimental computation is given in this chapter. Chapter 6 and 7 are references and appendix respectively, while in Chapter 5 an average monthly cost and profit of Easy Market from its activities in flight bookings. I finally concluded by giving my final thoughts on the project and Easy Market competences to further its research and development.

2. The flight optimization problem

Before the explosion of big data Easy Market was doing most of its operations on premises, one of the first problem it was trying to solve is filtering the huge number of transactions coming daily into its system through automation. By performing filtration, it can select which transactions for a particular route should be sent to GDs/suppliers which will cut cost and increase server performance. The other problem is the management and storage of the data from all these transactions coming into the system, the petabytes of data stored in GCP BigQuery. Easy Market strongly believe that by solving these problems, its system will be able to process high traffic speed, customer booking experience and reduce infrastructure cost.

2.1 Easy Market Business Model

The company was born in 1999 as one of the first distributor of holiday packages for travel agencies. Innovation is at the core of Easy Market as it strives to provides quality products and services to its customers. It operates as a B2B or B2C, some of its products and services includes hotels from different places around the world, excursions, flights at negotiated rates, car-rental, activities and insurance, group and transfers. These products and services are offered through its platform lol.travel and via APIs it sells as a service to other third-party agencies and booking platforms such as Skyscanner, viaggiogratias.com etc. According to report on data from 01-12-23 Skyscanner makes up approximately 53% of searches coming into the system, making it one of the largest shares of market count, google makes up approximately 32%, as depicted in figure 1. Easy Market is constantly developing it system to keep up with customer demands, one of its most recent innovation products is **REVOLUTION**, a B2B system made for agencies. **REVOLUTION** as the name implies is said to change the game in the travelling industry for agencies, with its advance and new easy to use features including graphical user interface, simple filters for search refinement, standardized purchasing and more. **REVOLUTION** is free, allows flexible payment methods and provides quick solutions with rich robust database of competitive offer prices and commissions (Easy Market, 2024). Beside the wonderful services and products Easy Market is offering to its customers, it has also been forging partnerships in its industry and beyond. In 2010 it became a member of Hotelbeds a major player in the global B2B bed banks whose parent company HBX Group. In academia the company partnered with the university of Bologna for research and development in it IT-sector and scouting for talented young minds to employ through its internship program with the department of DISI (Business Intelligence Group) which made this paper possible, and

department of economics & tourism. Easy Market also collaborates with HUWARE a cloud consulting company based in Milan, one of its key partners in its digital transformation journey. It also takes the time to participate in sustainability activities with Fondazione Cetacea ONLUS to save marine life. Other great feature which position Easy Market to compete are most of its workforce especially in the marketing department are young graduates with sound minds with understand of the travelling industry. As of 24 April 2024, Easy Market boast of having work with over +700 flight companies, +1million hotels, +500 car rental companies, +18000 activities in more than 223 countries and has flown over 800,000 passengers. The figure 2 below tries to map Easy Market’s business model in a canvas to get a full picture of its value, propositions, customer segments & relationships, channel of value creation, key partners, resources and activities.

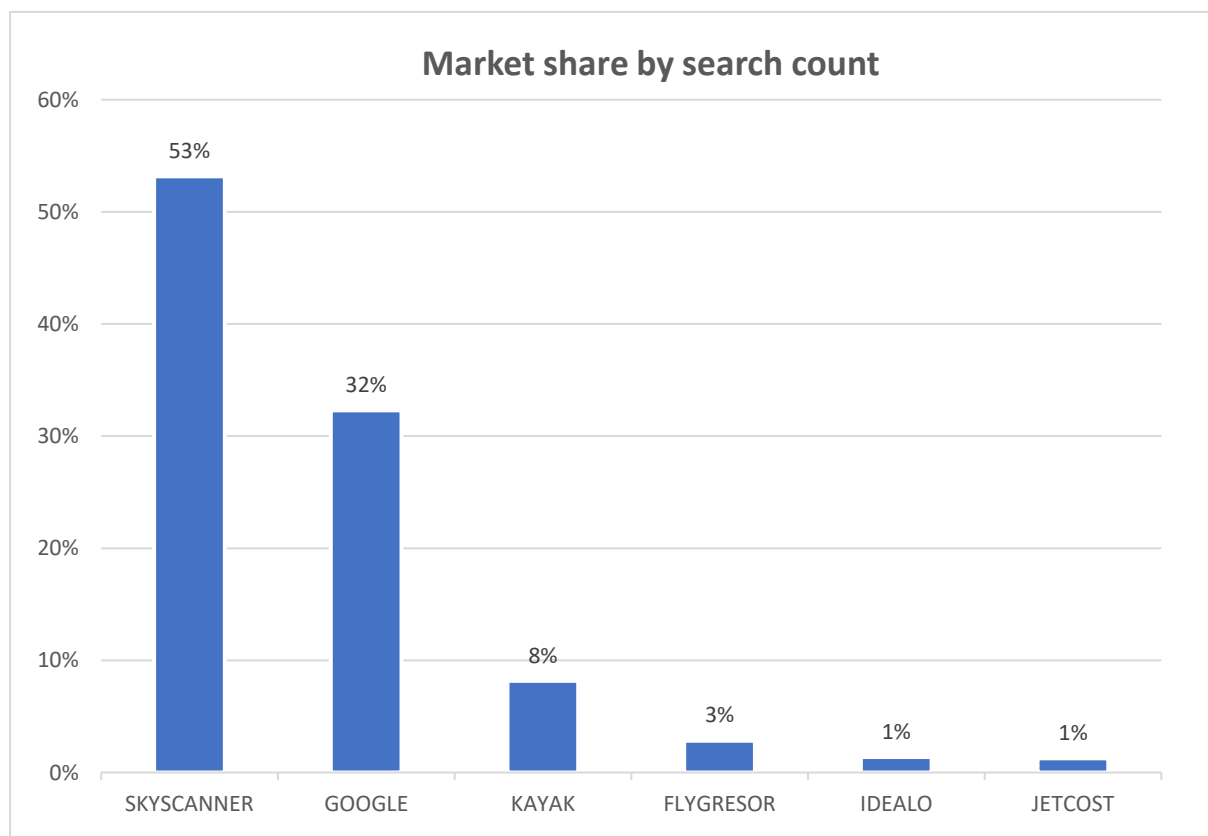


Figure 1. Market share by search count (self-contribution, 2023)










 <h1>Business Model Canvas</h1>			
Key Partners  <ul style="list-style-type: none"> HBX Group (hotel beds) Hotels MyBank HUWARE Car rental companies 	Key Activities  <ul style="list-style-type: none"> Safety measurement of its employees and customer. Research & Development of its products and services. Partnering with airlines, B2B TravelTech leaders. Employee recreational events 	Value Proposition  <ul style="list-style-type: none"> Negotiated price of flights, hotels, car-rental reservations Simplicity and easy of booking through its website lol.travel. APIs service to third-party travel agencies and booking platforms (Skyscanner). Platform as a Service through its software (Revolution). 	Customer Relationships  <ul style="list-style-type: none"> 24/7 Customer service Social media (Instagram) Exclusive promotional offers Public event (running)
Key Resources  <ul style="list-style-type: none"> Brand Well trained staff (business & IT professionals). Office building , computer infrastructure (on-premise & cloud). 	Channels  <ul style="list-style-type: none"> Physical offices Website(lol.travel) Social media (LinkedIn, Instagram etc.) Public fares 	Revenue Streams  <ul style="list-style-type: none"> Sales of API as a service Platform as a Service (Revolution) Flight, Hotels, and Car-rental reservation 	Customer Segments  <ul style="list-style-type: none"> B2B: <ul style="list-style-type: none"> Travel agencies Booking platforms B2C: <ul style="list-style-type: none"> Families Business class Excursion groups Ordinary travellers
Cost structure <ul style="list-style-type: none"> Marketing costs Employee salaries Rent for office building GDs subscription fees General partnership fees Research & Development 			

Figure 2. Easy Market Business Model Canvas (self-contribution, 2024)

2.2 The optimization problem in-depth

At the core of the problem is the “PROFILES_ENGINES_AVAILABILITY” a relational table sort of database in which the first stage of transaction filtering is done. But before diving deeper into the problem it is necessary to give a high level overview of the system from a software engineering perspective. The figure 3 shows how the booking process of a flight takes place on Easy Market system. Note this paper does not cover the underlining logic of the hotel and car-rental, so it will be wrong to assume the same logic holds for the other two. Whenever someone tries to book a flight let say directly from lol.travel, an API call is made to the system for a travelling route and a result is return by the system or not depending on if that destination is not in a BLACKLIST/Temporary blocked. The result could be one or many solutions with different pricing the customer then choose the most suitable choice based on her preference. In the process of making this API call a series of operation processes are performed in the backend and my focus here is on the “PROFILES_ENGINES_AVAILABILITY.” The figure 3 shows an example of results return from search performed for the route Bologna – Barcelona.

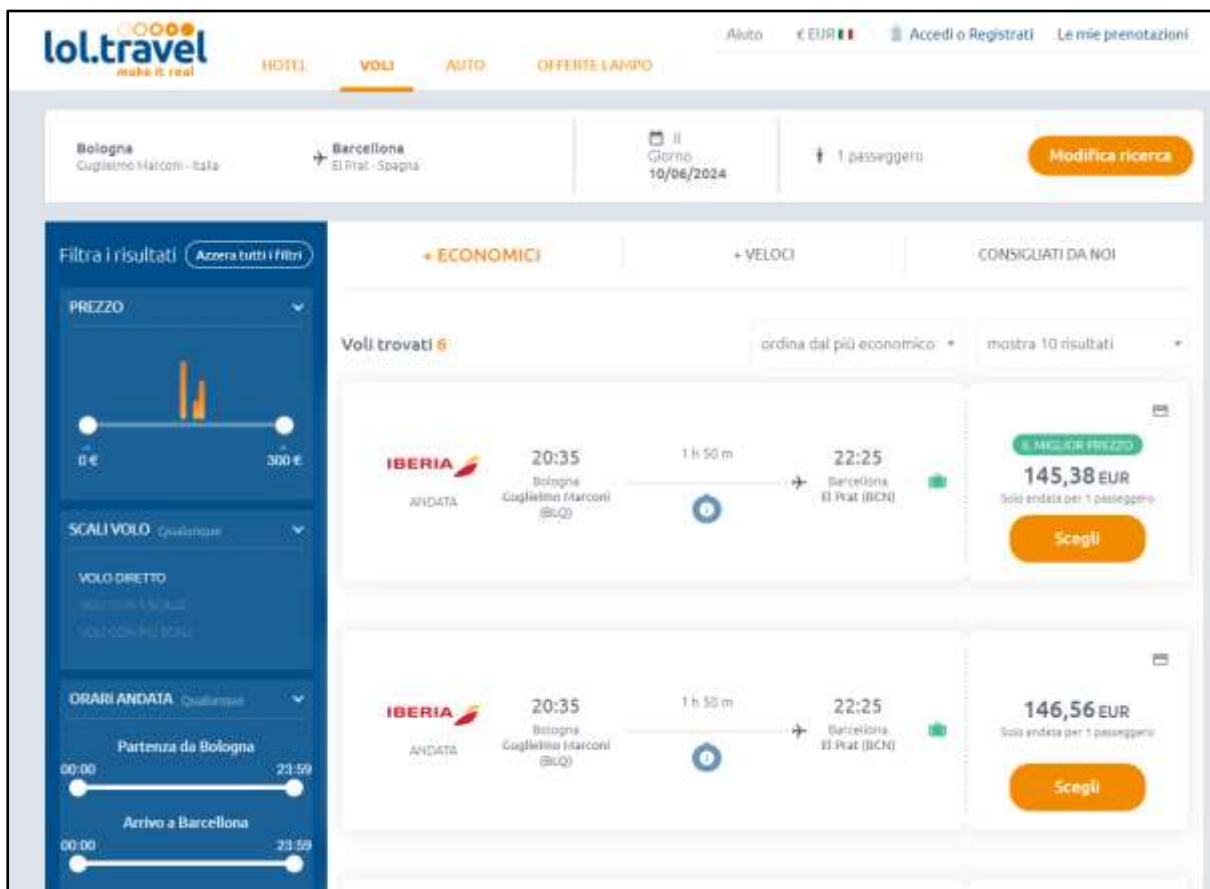


Figure 3. Booking on lol.travel (self-contribution, 2024)

The figure 4 shows the underlining logic of the booking process, a concise explanation of this diagram is let say a traveller who wants to travel to a certain destination (Bologna-Barcelona) comes to lol.travel and enters the departure and arrival airport including other details about the journey, when the traveller makes this search on lol.travel a query is made via an application programming interface (API), a software program which allows two or more software to talk to each other. So, in this scenario the traveller interact with the website (front-end) and the API allows the communication to the backend (Easy Market resources), when a query is made the request is process by Easy Market system and sent to various airlines and computerized network systems known as GDS that enable transactions between travel industry service providers in real-time to facilitate the booking process for travel arrangements companies and online booking engines. If the query made by the traveller for a particular route is available on Easy Market system then it will further the request to it suppliers and GDS, a result is then return to the traveller and when a click is made to further the process to a booking Easy Market then sell and make profit. Another scenario is Easy Market sells by making it resources available to third parties via API, these third parties include small travelling agencies and large players in the online travelling arrangements industry such as Skyscanner, Google flights, Kayak etc. As seen in the business canvas Easy Market sell API as a service but also offers platform as a service for free (REVOLUTION), a B2B system designed exclusively for Travel agencies with the aim to simplify travel agents.

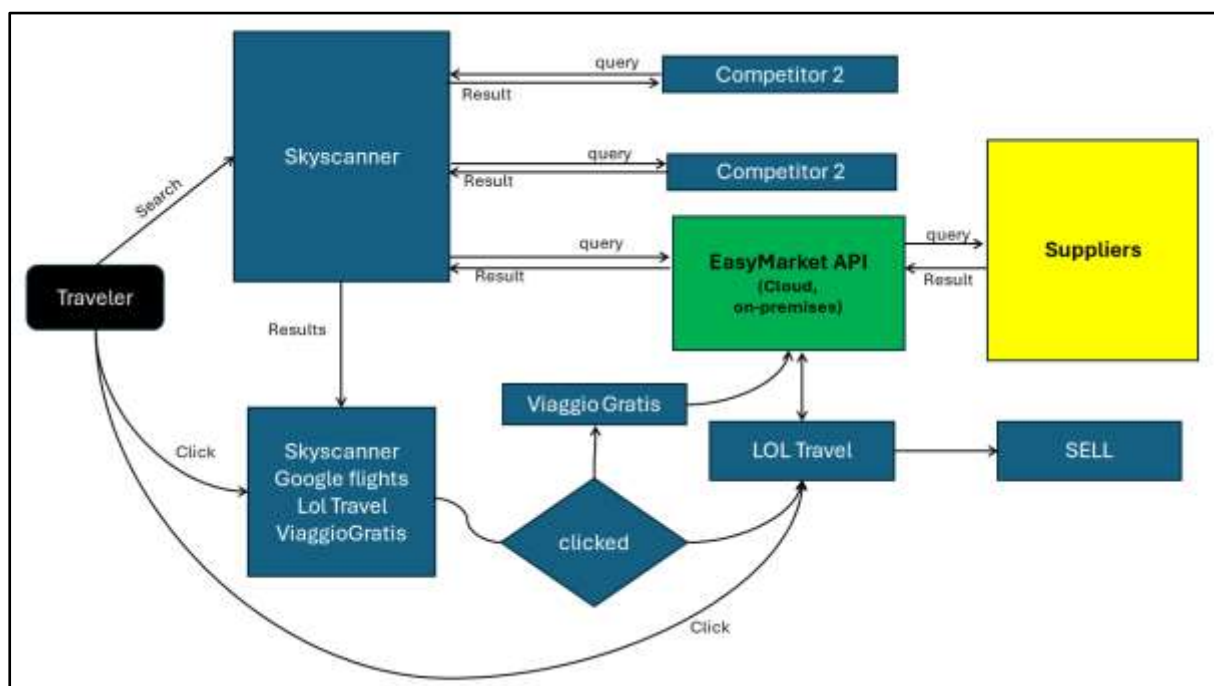


Figure 4. Easy Market booking system (IT Team & self-contribution, 2023)

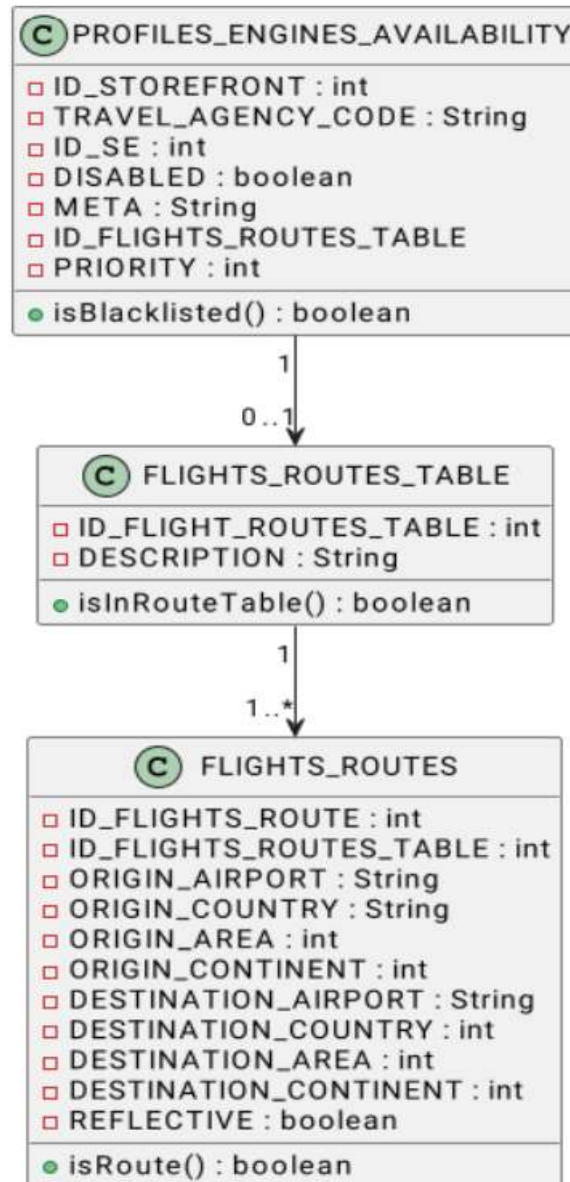


Figure 5. Filter engines (IT Team, 2023)

Behind the API made to third parties by Easy Market is its backend which is made up of both cloud and on-premises resources including storages in the form of a data lake and relational database. The first filter which is the PROFILES_ENGINES_AVAILABILITY is managed on-premises by Easy Market and serves as the key point of the system performance, because of the role it plays in the company's daily operation (BLACKLISTING and UNBLOCKING). The logic behind this filter is, all records in it are ordered in descending order by priority and if two or more records are having the same priority the first record which was loaded into the list is considered the top. The first record that satisfies all the criteria for selection will be used to determine if the GDS is blacklisted or not.

The selection criteria are made up of the values store in the columns of the PROFILES_ENGINES_AVAILABILITY, they are id_storefront, travel_agency_code, id_se, disabled, meta, id_flights_routes_table, priority.

The id_storefront is an int data type that holds information of the various platforms Easy Market is selling, it is made up of both b2b and b2c. Example of some b2b/b2c are shown in the business model canvas and some of the ids of these b2b/b2c are: B2B (2 = Revolution, 54 = BOL), B2C (49 = ViaggioGratis, 54 = LOL) and -1 = A wildcard for all storefronts.

The travel_agency_code is a string data type that holds information of the agencies/metasearch, while the id_se represents the ids of the GDSs which the queries will be sent to. Another important criterion is the id_flights_routes_table which checks if the route is acceptable, a value of -1 means all routes are acceptable value different from -1 it for route is dealt with by the next table/second filter which is out of the scope of this thesis scope.

2.3 Architecture of the data flow in Easy Market system

Easy Market's data flow architecture has four phases, the process might look like the booking system, but figure 4 (booking system) is mostly from a software engineering perspective while the data flow diagram is from a cloud architect/ big data engineer perspective. The first phase is where data generation takes place, which is from its website lol.travel, Skyscanner, metasearch, and other agencies it sells API as a service. In the second phase data is sent to Easy Market's GCP infrastructure, where Pub/Sub is used as a data ingestion tool from the various sources and sent BigQuery in its raw form. The on-premises system which contains Easy Market's softwares and engines (relational databases) to process these queries before sending them to the suppliers which are third and fourth phase, respectively. In the third phase which is on premise engines has three relational databases namely availability, pre-search filter, and post-search filter as I explained in the previous section. The availability engine checks whether a certain flight route for example let say Milan to Rome is available, the other engines try to check for other attributes or information included in the query before it is sent to the suppliers and results is given back to Easy Market. The results from suppliers are stored in BigQuery and sent to the customer simultaneously with various pricings, and if the customer decide to make any booking, the same process applies.

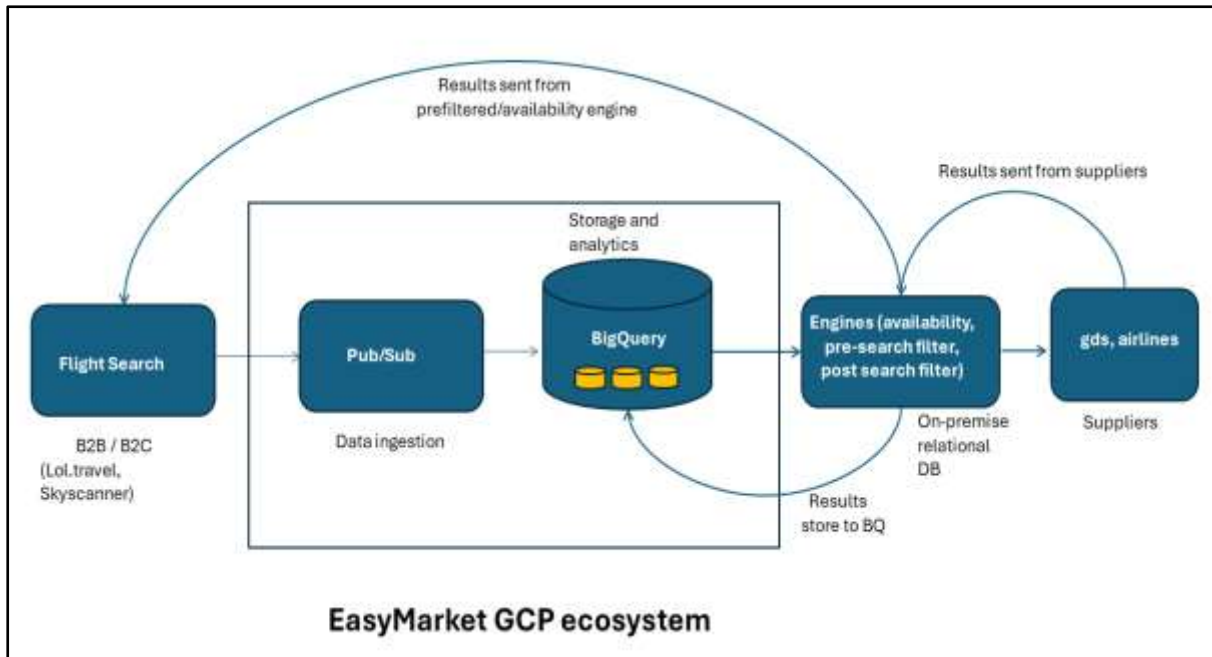


Figure 6. Architecture Data flow in Easy Market's ecosystem (self-contribution, 2024)

2.4 What Easy Market want to optimize

Scalability to manage the petabytes of data is no longer a problem. However, Easy Market aims to develop a unified autonomous system that get data from its various departments (marketing, flight operation, IT etc.), perform complex analysis on these petabytes of data in real-time with less or no human intervention in term of taking the decision on which route to block or not through a series of pipeline based on the engine availability logic. The system should also be able to perform other statistical computation such as look-to-book ratio (conversion rate) for the various route. The benefit of such system is it will save time and money eventually reducing infrastructure, increase look-to-book ratio (conversion rate).

3. Literature on Big Data and Google Cloud Platform Technology

In the era of IoT, pervasive computing and massive digitalization it is hard to say that every single person with a computing device (mobile, computer, smart fridge, Alexa) is not generating or contributing to the explosion of what has been coined Big Data (BD). As of 14th March 2024, report from Statista estimated that the ‘total amount of data created, captured, and copied and consumed globally is forecast to increase rapidly reaching 64.2 zettabytes in 2020 and is projected to surpass >180 zettabytes in 2025 (Taylor, 2023) as show in figure 7.

Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025 (in zettabytes)

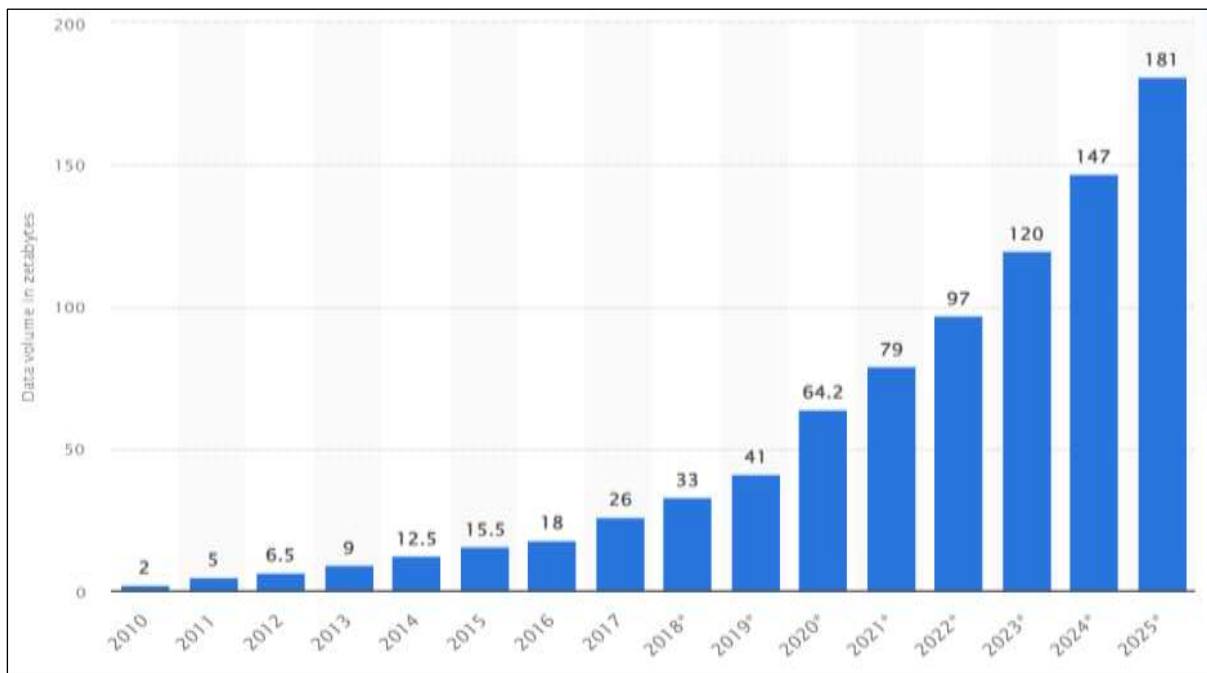


Figure 7. Volume of data created in zettabytes (Taylor, 2023)

Big data has been on the public limelight for years now and the term has spark interest from people in academia and cases such as BD been used by researchers at Oxford university to help clinicians detect cancer earlier and treat it better (Oxford research, 2023). Another popular use case of BD is the ImageNet database invented by Prof. Fei-Fei nicknamed, ‘The Godmother of AI’ (Downey, 2024) because of her contributions to the advancement of Artificial Intelligence. The ImageNet database has been used by several researchers in the field of computer vision

and deep learning for developing image recognition software. “It is available for free to researchers for non-commercial use” (ImageNet, 2021).

Finance has been one of the most data-intensive industry with investors and other finance professionals having to keep up with real-time stock markets insights, customer analytics, risk management and fraud detection which are only possible with the development of complex machine and deep learning algorithms that learn from massive amounts of data (CFI Team, 2024). According to Taylor (2024), “the global big data analytics market is expected to see significant growth over the coming years, with a forecasted market value of over 650 billion dollars by 2029”. The following is shown in figure 8.

Size of the big data analytics market worldwide from 2021 to 2029 (in billion U.S. dollars)

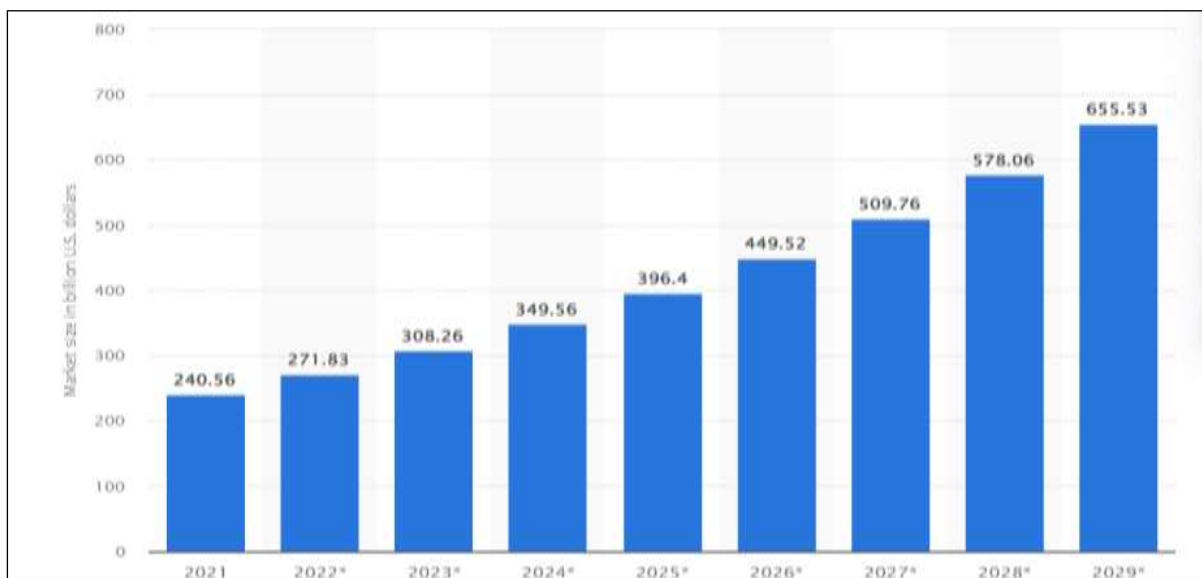


Figure 8. Size of the big data analytics market worldwide (Taylor, 2024, [statista.com](https://www.statista.com))

In politics BD has gain notoriety with regards to Cambridge Analytica, a company that offers IT services to political parties and businesses who want to change customer or audience perceptions and behaviour. Cambridge Analytica engaged in Facebook’s data breach of 50million profiles (Osborne, 2018). Figure 9 shows the public interest in BD for the past 16years.

3.1 What is Big Data

Big data refers to data that is so large, fast, or complex that it is difficult or impossible to process using traditional methods (SAS). Doug Laney in the early 2000s defined BD in three dimensions namely, volume, velocity, and variety (Balusamy et al., 2021, p.2). But according

to Dr. Enrico Gallinucci, professor of big data at the University of Bologna he sees BD in two flavours, that is as a noun: ‘we have big data’ and an adjective: we use big data tools.

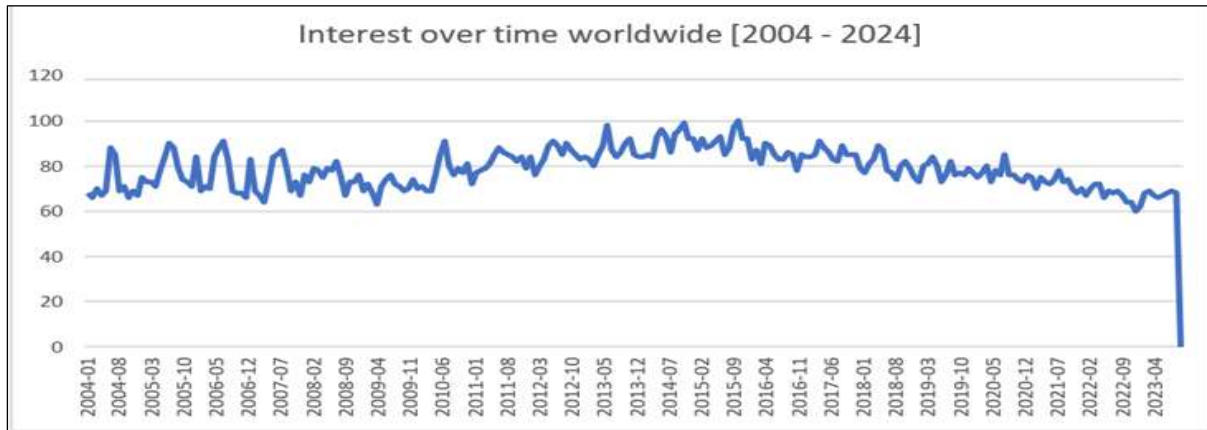


Figure 9. Interest for the term Big Data, (Google Trends, 2024)

He further elaborated on BD as a noun using McKinsey’s definition, “BD refers to data sets whose size is beyond the ability of typical database software tools to capture, store, manage and analyse.” Like Laney’s view of 3Vs about BD, Gallinucci added a fourth-dimension veracity which was coined by IBM (Gandomi & Haider, 2015), and pointed out the new Vs arising (value, viscosity, virality, and visualization) as shown in the appendix A of the literature review section. BD as an adjective becomes more specific that is architecture (lambda, kappa), tools (e.g. Apache Spark) and paradigm (e.g. MapReduce), we will explore all these concepts in the Hadoop section of BD technology.

Even though there was massive digitalization in the early 2000, BD did not started there, the act of gathering and storing large amounts of data dates back to the early 1950s when the first commercial mainframe computers were introduced (Lee, 2017). The 1950s saw the development of the first data centers and relational database (Tiao, 2024), most data at the time were structured to support operational and transactional information system (Lee, 2017). The invention of the world wide web (www) by computer scientist Tim Bernes Lee in 1989 ([web foundation](#)) led to the explosive growth of data and the development of big data and data analytics which have evolved through major stages from big data 1.0 to 3.0 (Lee, 2017). Lee (2017) describes BD 1.0 during the time of the dot.com bubble and e-commerce in 1994, BD 2.0 powered by social media and web 2.0 allowed users to create their own content, Big Data 3.0 includes both 1.0, 2.0 and IoT applications that generate unstructured data (images, audios, and videos).

The credits of where the term BD came from and when it was first used was given to [Dr John Mashey](#) a computer scientist who worked at Bell labs from 1973-83 and was an early contributor to the UNIX operating system, 20years later he became the VP & chief scientist at Silicon Graphics (SGI) from 1992-2000, SGI is producer of special-effects computer graphics for Hollywood and video surveillance spy agencies (Lohr, 2013). According to Lohr (2013) in an article of the New York Times titled “The Origins of Big Data: An Etymology Detective Story”, he reached out to Mr. Mashey about the claim and Dr. Mashey replied, “I was using one label for a range of issues, and I wanted the simplest, shortest phrase to convey that the boundaries of computing keep advancing”. There was no academic record or journal to prove giving in the late 1990’s, one of the presentation slides can be found on USENIX titled “Big Data and the Next Wave of Infrastrass” (Lohr, 2013).

However, in a 1997 academic conference later published by IEEE titled “Application-controlled demand paging for out-of-core visualization”, NASA scientists Michael Cox and David Ellsworth put forward a problem faced by engineers and scientists in visualizing large datasets mostly in ‘Computation Fluid Dynamics (CFD)’ (Morales, 2020). Cox and Ellsworth were both from MRJ/NASA Ames Research Center, in their publication gave the first academic explanation saying, “data sets are quite large taxing the capacities of the main memory, local disk and even remote disk, so they called it the problem of big data” (Balusamy et al., 2021, p.2). At the time the two projected that dataset will surpass one hundred gigabytes but as seen in figure 7, Statista forecast that we would reach 181 zettabytes in 2025 which is 28years since the term first appeared in academic journals.

Since 1970 when the relational model was first put forward by Edgar F. Codd a former computer scientist at IBM, RDMS has been the go to storage medium by most organisations but in 2005 people began to realize just how much data users were generating through social media platform like Facebook and video streaming e.g YouTube and other services (Tiao, 2024) which poses a lot of drawbacks for traditional databases. Balusamy et al. in their book titled “Big Data – Concepts, Technology and Architecture” gave the following points as limitations faced by traditional database in the era of big data:

- i. Volume: the exponential increase in data volume, which scales in terabytes and petabytes
- ii. Cost: to keep up with the volume, increment in RDBMS processors and memory units are made eventually leading to increase in cost.

- iii. Format: IoT data such as audio, videos and sensors from computing machinery just caused more problems for RDBMS since they are semi-structured and unstructured.
- iv. Speed: RDBMS were created for transactional operations and not to capture data coming in at high velocity.

Volume, velocity and variety mentioned above are what has been called the 3Vs or characteristics of BD in most academic literature, which is discussed in the next section.

Table 1 shows the difference in the attributes of big data and RDBMS (Balusamy, 2021, p.4)

ATTRIBUTES	RDBMS	BIG DATA
Data volume	gigabytes to terabytes	petabytes to zettabytes
Organization	centralized	distributed
Data type	structured	unstructured and semi-structured
Hardware type	high-end model	commodity hardware
Updates	read/write many times	write once, read many times
Schema	static	dynamic

As the name implies big data, size has been the first dimension often exaggerated, commodity hardware has made storage easier and therefore the steady growth in size is referred by the term “volume” in big data technology. The large volume of data generated come in all distinct types, be it structured, unstructured, or sometimes even a mix of both, this heterogeneity is the second dimension known as “variety.” The third is “velocity” the rate at which all the data generated (Balusamy et al., 2021, pp.5-6).

Volume: The major sources of BD are social media, online banking, point of sales (POS) transactions, GPS, and vehicles sensors. BD volume measure from terabytes to zettabytes (1024 GB = 1 terabyte; 1024 TB = 1 petabyte; 1024 PB = 1 exabyte; 1024 EB = 1 zettabyte; 1024 ZB = 1 yottabyte) (Balusamy et al.,pp-5-6). BD systems store data in a distributed computing by replicating across multiple nodes, in commodity hardware (Achari, 2015, p.2).

Velocity: Big data velocity entails both the speed at which data is generated and analysed. Massive data arrive so fast posing difficulties in capturing and analysing. IN 2009 Yahoo created a record-breaking result by sorting petabyte of data in just 16.25 hours, and 62 seconds for terabyte (Achari, 2015, p.3).

Variety: big data support all three types of data, structured, semi-structured, and unstructured. RDBMS and excel spreadsheets are examples of structured data which are mostly in tabular form, XML and JSON are semi-structured, because they contain tags to organize fields within the data and do not fit the formal data model associated with traditional database (Balusamy et al., 2021, p.6)

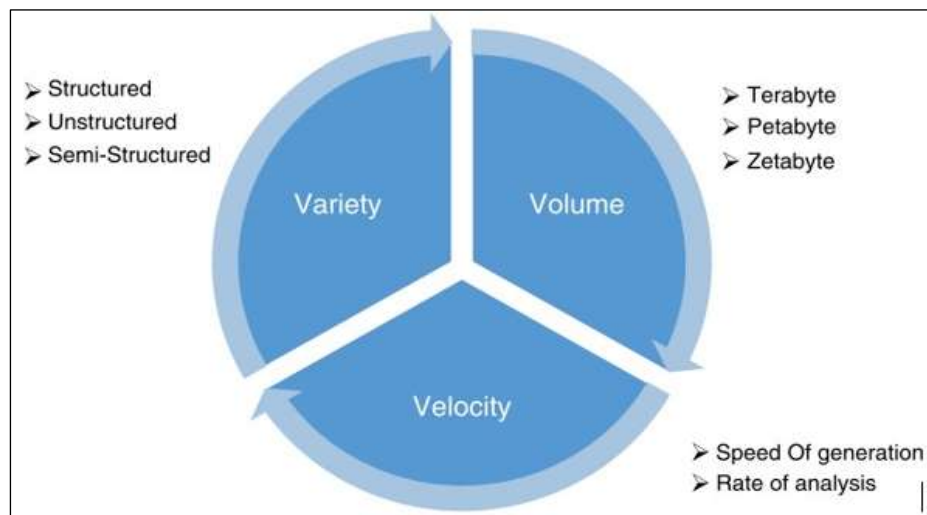


Figure 10. 3Vs of big data (Balusamy et al., 2021, p.5)

The huge growth of data from various sources is due to the digitalization of anything and everything in the globe (Balusamy et al., 2021, p.7), e-payment technology such as fintech, blockchain, social media, email, and most recently user generated data by interacting with AI applications e.g. ChatGPT (DOMO, 2023). Sensor data which is data from accelerometer installed in mobile, medical, and vehicles to sense the vibrations and other movements contribute to large volume of big data. Other sources include black box data generated by planes often record flight activities and performance (Balusamy et al., 2021, p.7). Medical record take from patient is also contributing to big data and most of these records are stored and shared by medical professionals to advance the field of medicine as in the case of Oxford research cancer big data. The example above is not an exhaustive list of all the dataset contributing to the big data that comes in distinct types which is discussed in the next section.

In our connected and tech driven world both machines and humans generated data every single minute. Human-generated data refers to data generated because of people interacting with machines examples of these are emails, documents, and social media posts (Balusamy et al.,

2021, pp.8-11). Computer applications, sensors, disaster warning systems and satellite data are common examples of machine-generated data with no active human intervention, as shown in figure 11. The data can be structured, unstructured, or semi-structured in format.

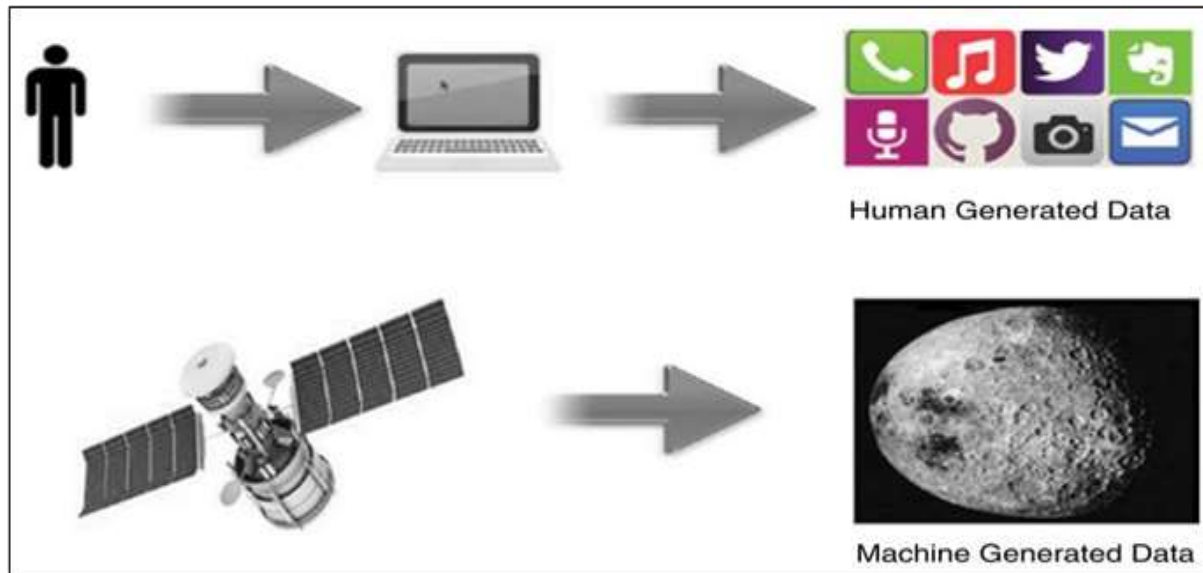


Figure 11. Human and machine-generated data (Balusamy et al., 2021, p.9)

Structured Data: data stored in relational databases and other spreadsheet applications in tabular format with rows and columns are called structured data. These datasets can be processed and queried using specific identifiers refer to as primary key. Sales, employees’ details table with specific tuple identifiers as key are example of structured data.

Unstructured Data: According to Hambert (2021) in an article for MIT SLOAN titled “Tapping the power of unstructured data”, the author mentioned that unstructured data (images, audio, videos, emails, etc) make up 80-90% of big data and there is a huge potential for companies to gain competitive advantage once they tap on it. Unstructured data are mostly stored in binary or text files, and do not conform to the traditional relational model.

Semi-structured Data: JSON, XML and other markup languages also do not conform to traditional relational model, the data is normally organized through tags or nested in curly braces. There are several benefits of semi-structured data, it is portable and storable compared to unstructured data but can be challenging to query if prior knowledge of the dataset is unknown. Most NoSQL databases example MongoDB can store semi-structure data. Figure 12 shows a simple example of a semi-structured data.

```
<html>
  <head>
    <title>DTM master's thesis</title>
  </head>
  <body>
    <h1>Big data Thesis</h1>
    <p>HTML is an example of semi-structured data</p>
    <p> Semi-structured datasets are stored in NoSQL databases e.g. MongoDB</p>
  </body>
</html>
```

Figure 12. Html code (self-contribution)

Big data is not just about size and speed, there is more to it. Data goes through various stage known as the big data life cycle. According to Professor Elisa Bertino et al. (2011) at Purdue university in their paper titled “Challenges and Opportunities with Big Data” listed the following stages as the life cycle of BD from it acquisition to consumption by final users. The process (acquisition, extraction, integration, analysis, interpretation, and decision-making) is shown in figure 13. The following stages of the BD lifecycle as layout are explained below:

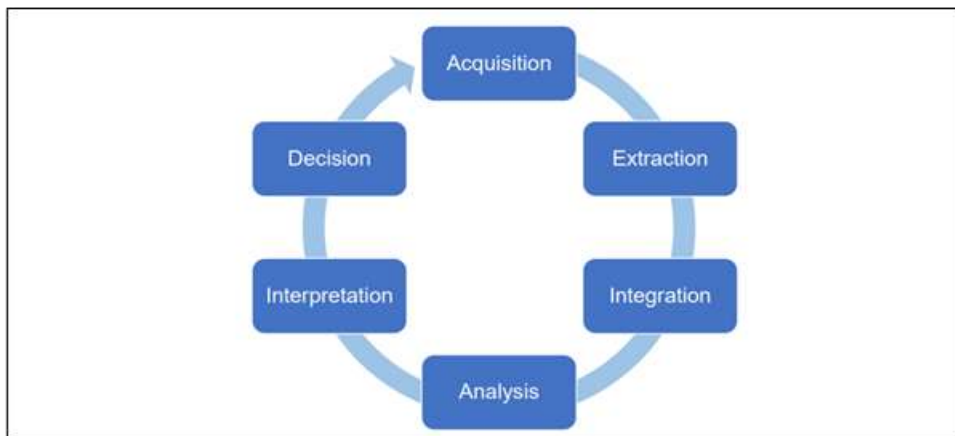


Figure 13. Big Data lifecycle (Gallinucci, p.19)

Acquisition: This is often the first step in the BD lifecycle and consist of sub-processes such as selection, filtering, compression, and metadata collection. The selection process has to do with mostly not only choosing which sources of data to be considered but also which data is valuable for analysis, before filtering & compression can be done. Another important thing about this stage of the BD lifecycle is collecting information about the data itself normally referred to as metadata.

Extraction: After the acquisition of the data, processes such as transformation, normalization cleaning and error handling can be performed. Transformation and normalization are often the process of expressing data in a structured form suitable for analysis, while handling inaccurate, biased opinion and obsolete data has to do with cleaning and error handling.

Integration: This stage has to do with bringing data from multiple sources, discovering the relationship between the datasets, entity resolution, conflict management and standardization.

Analysis: The analysis of data is one of the most interesting parts of any data project, activities such as exploration, analytics and delivery are performed. In the exploratory phase statistical techniques and data visualization tools are used to uncover hidden characteristics and patterns of the dataset. Other analytical techniques include diagnostics, predictive and prescriptive are normally performed for in-depth insight useful for advanced modelling e.g. BI star schema and machine learning. The various types of data analysis put forward by Gartner are shown in the appendix E of the literature review section.

Interpretation: Extracting valuable knowledge from big data is more difficult than collecting and storing it, so caution must be taken before final conclusions can be made. Expertise such as domain knowledge and data lineage are of high importance in this stage because an accepted standard in one field of study/context is considered low for another. An example is the R-squared, a 0.5 threshold in social science is considered good, while 0.7 and above is considered high level in finance (Fernando et al., 2023). All assumptions must be tested using artificial/synthetic or sub-sample of the dataset to verify expectations, discover hidden patterns and correlations.

Decision: Data ready for business and other use normally requires strong managerial skills in the decision making process. The business strategy evolves based on previous knowledge and the impact are verified on the new data, then feedback is provided for continuous improvement (Gallinucci, 2023, p.19). Retailers such as Wal-Mart is using big data to know which two or more products together can lead to improve sales (Jeble et al., 2018). Data from aircraft black-box can be used to anticipating natural disaster an example is the Cesena flooding in 2023 when everyone got a high alert warning about the flood. More examples of big data in making decisions can be found in the appendix F of the literature review section.

3.2 Big Data Technology

No big data paper is complete without discussing the underlining technologies which has led to the success in the advancement of research in various field of academia, an example is Baranowski et al. (2019) at CERN IT using “Hadoop Platform and Ecosystem for High Energy Physics” a substitute of workload that are hard to scale with traditional database. Another use case of BD technologies is for-profit making companies such as Nokia “Nokia: Using Big Data to Bridge the Virtual & Physical Worlds” by analysing terabyte of data using Cloudera’s distribution including Apache Hadoop. Many technologies are now arising to manage the massive data generated every single minute. According to Shiva Achari author of “Hadoop Essentials” expert in big data and former employee of Oracle, Teradata and AWS, mentioned that technologies that can solve BD problem should use distributed computing system, massive parallel processing (MPP), NoSQL and Analytical database as shown in figure 14. But the most popular one so far has been Hadoop, which is briefly discussed in the next section but for in-depth knowledge on every core component of Hadoop ecosystem readers are strongly encouraged to check the documentation on Apache Hadoop’s official website THE APACHE SOFTWARE FOUNDATION.

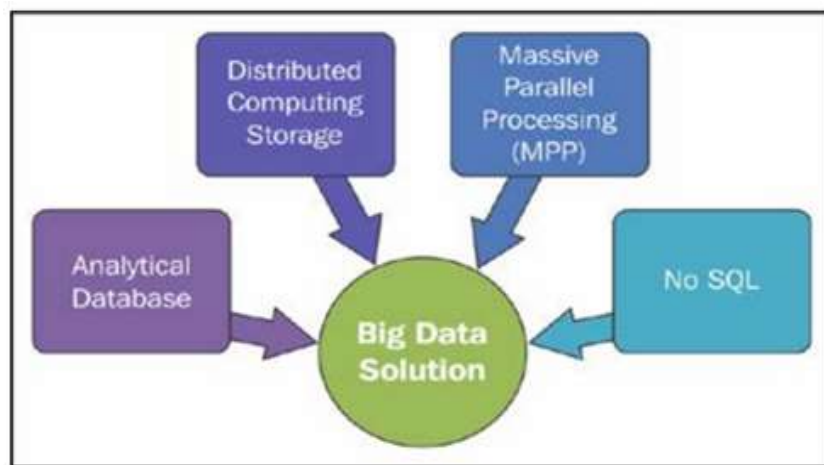


Figure 14. Big Data architectural strategy (Achari, 2015, p.4)

Apache Hadoop is an open-source software framework written in Java programming language, it can be used for storage and large-scale processing of datasets with streaming access on clusters of commodity hardware. Hadoop was created by Doug Cutting and Mike Cafarella in 2005, it was originally developed to support distribution of the “Nutch Search Engine Project.” Hadoop was named after Doug’s son’s toy elephant, Hadoop. All the modules in Hadoop are designed with the fundamental assumption that hardware fails, or a component of the hardware can fail at some point in time. Hadoop was originally derived from Google’s MapReduce and

file system and has evolved ever since its inception (Natasha Balac, Ph.D., 2024).

The Hadoop 1.0 architecture is made up of two layers, the storage layers which is HDFS and computational layer previously MapReduce. HDFS is designed to store large amount of data on commodity hardware, it is also suitable for datasets with streaming access pattern and not for application with low latency access data. HDFS architecture consist of the NameNode and DataNode, it adopts the master/slave architecture where one machine acts as the master and the others the slaves. It was originally developed for resilience, scalability, portability and application, locality. The NameNode stores metadata about all the files in the system and knows which DataNodes to query upon request by a client for a particular information. Block creation, replication and deletion are also performed by DataNodes upon instructions from the NameNode. The figure 15 shows the architecture of HDFS.

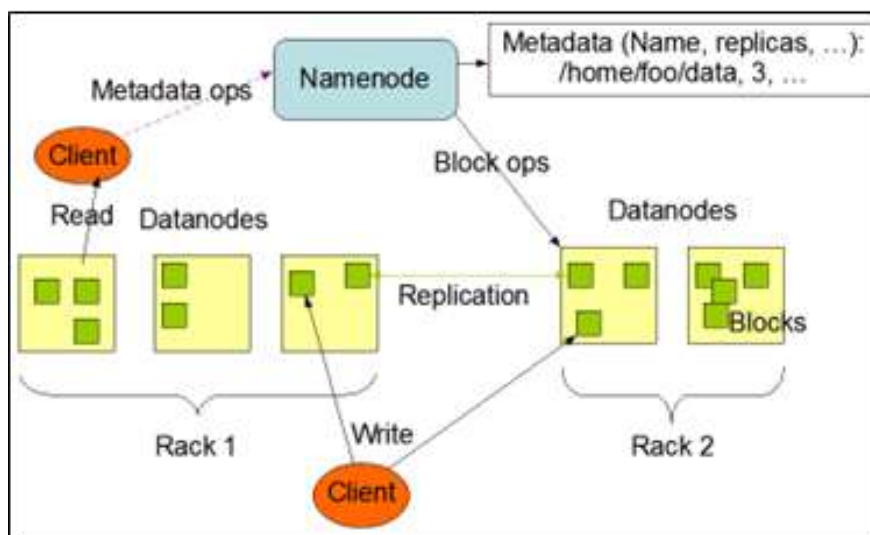


Figure 15. HDFS Architecture by ASF

MapReduce is a programming paradigm for processing data in batch, which was inspired by functional programming's map and reduce functions (databricks, 2024). It is dependable, fault tolerant, and highly scalable. Divide-and-conquer is its main principle when processing any data format. MapReduce job splits data into chunks, map tasks process data chunks, framework sorts of map output, reduce tasks use sorted map data as input. Even though MapReduce was so successful in Hadoop 1.0, there were still some limitations posed by various computational activities. It was not designed to process smaller datasets, the metadata of large chunks of smaller datasets cannot be stored in the NameNode due to memory space wastage. One of the greatest problems in Hadoop 1.0 is that NameNode can serve as a single point of failure, when

the NameNode goes down the cluster becomes unavailable making the whole process inefficient. Reduce phase of a MapReduce job cannot start until the map task is complete. Two other points mentioned by Mahidhar Tatineni, Ph.D. leader of San Diego Supercomputer Center (SDSC) are iterative data exploration and processing. An example of a simple MapReduce program of “wordcount” is shown in appendix G of the literature review.

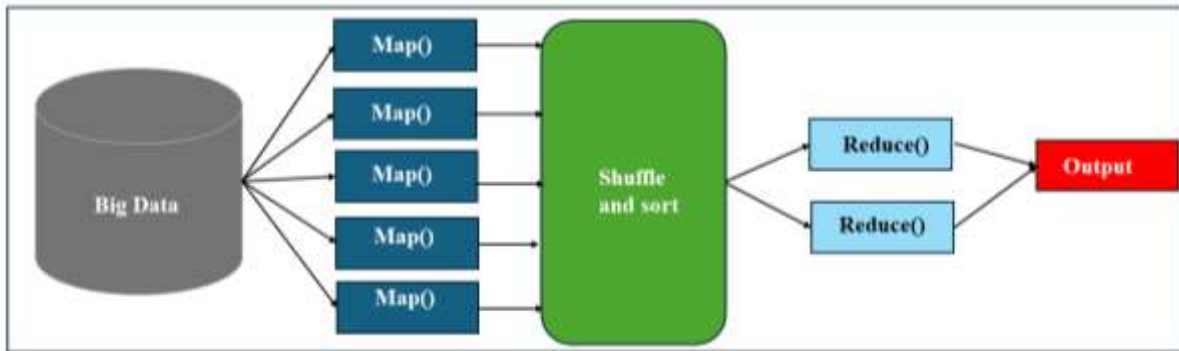


Figure 16. MapReduce Architecture

Hadoop 2.0 was developed to solve the existing problem of single point of failure, by having two NameNodes running on the same cluster when the active NameNode fails the standby/secondary NameNode acts. One of Hadoop 2.0 greatest feature is the YARN architecture that splits the responsibilities of JobTracker into a global ResourceManager and per-application ApplicationMaster. Resource management is taken care by the ResourceManager while job scheduling and monitoring is done by the ApplicationMaster. YARN's core components are: ResourceManager, ApplicationMaster and NodeMaster. YARN supports most of the frameworks in the Hadoop 2.0 ecosystem, but Spark can also run directly on HDFS without YARN (Tatineni M. Ph.D., 2024).

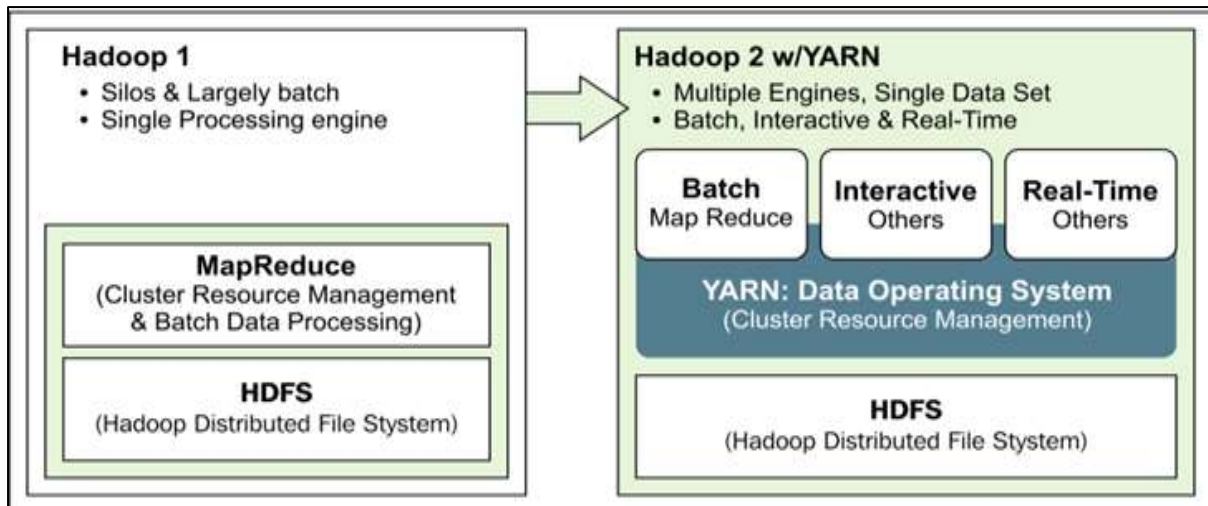


Figure 17. Hadoop architecture comparison 1.0 and 2.0 (hortonswork)

Even though YARN has managed to eliminate the master-slave architecture bottlenecks (single-point-of-failure), researchers from KTH Royal Institute of Technology pointed out that there are still some opportunities for optimizations Hadoop/MapReduce which are group in three main categories: performance issues, programming model extensions and usability enhancements (Kalavri et al., 2013, p.2).

Since the thesis is centred around designing a system that performs both batch and streaming analytics it is important that I dedicate this section to explaining about BD architecture that has been proposed and their drawbacks before looking at Hadoop components that was developed to handle such data processing. According to Microsoft (2024) BD architecture are designed to handle the ingestion, processing and analysis of data that is too large or complex for traditional database systems, and the threshold at which organizations called dataset big is based on their existing capabilities and tools (technology) to store and process all the incoming dataset. The NIST architecture is one of the most referenced architectures in BD. Its system comprised of various components that make up the infrastructure integration layer and information or data transformation flow (Demchenko, 2017, p.32).

Demchenko et. al (2017) described the following components that are involved in BD production, processing, delivery and consuming. The Data Provider which produces/supplies data related to specific process or even, Data Consumer utilize the processed/ready-to-use data. BD Application Provider consist of all services related to the analysis and transformation, BD Framework Provider is mostly about the infrastructure and its components while System Orchestrator is a separately defined functional role that may include both internal workflow of

the Big Data Application Provider and external workflow of the Big Data value added services. The logical component of BD are data sources, data storage, batch processing, real-time message ingestion, stream processing, machine learning, analytical data store analysis & reporting, and orchestration. Apache Hadoop support all the following and can be integrated with most cloud vendor such as google, amazon etc.

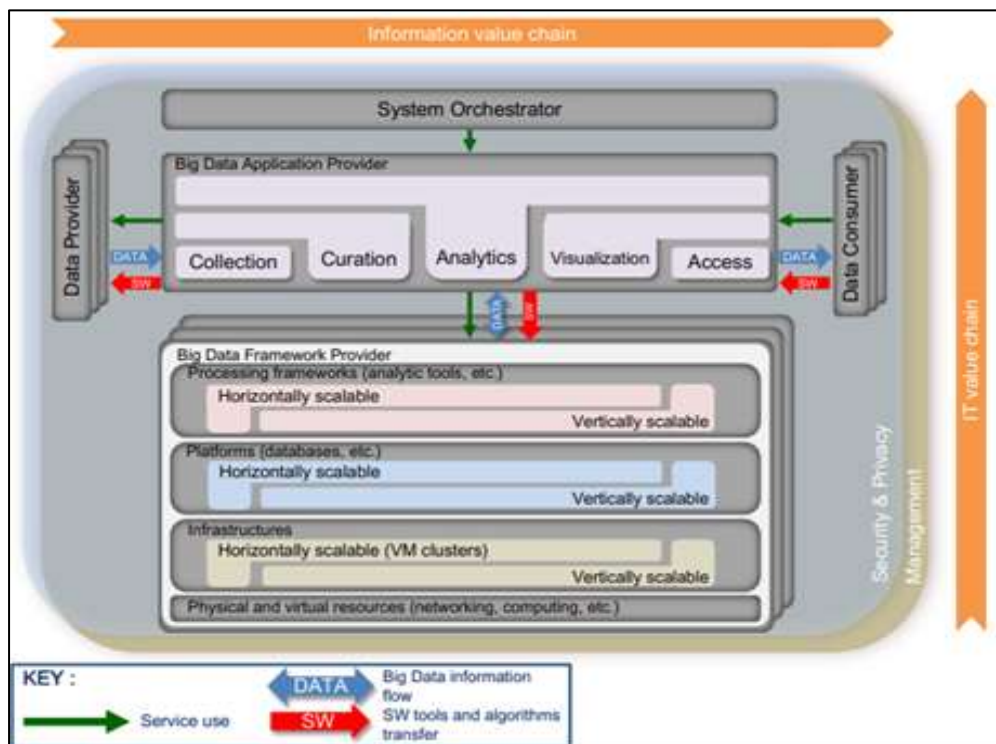


Figure 18. NIST Big Data Reference Architecture (Demchenko et. al., 2017, p.9)

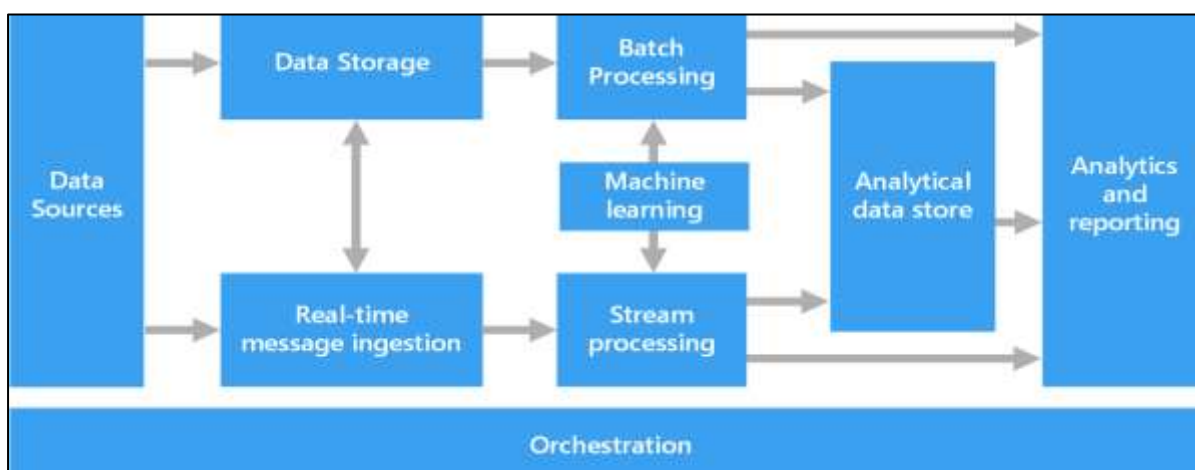


Figure 19. Components of a big data architecture by Microsoft

Data sources: In the previous section “Where does Big Data come from,” we explore the

various sources of data, and in two modes namely real-time mode and batch mode (Yaseen & Obaid, 2020, p.3).

Data storage: data ingested from various sources in batch and/or real-time are stored in a distributed file system such as HDFS, NoSQL or RDBMS.

Batch processing: this data processing is mostly suitable for application with terabytes or petabytes of data where response time is not particularly important (Balusamy et al., 2021, p.88). In batch series of jobs are logically connected and executed sequentially or sometime even parallel, meaning the output of individual jobs put together can give a final output (Balusamy et al., 2021, p.88). Hadoop MapReduce was developed purposely for such job/task. An example of batch processing is shown below.

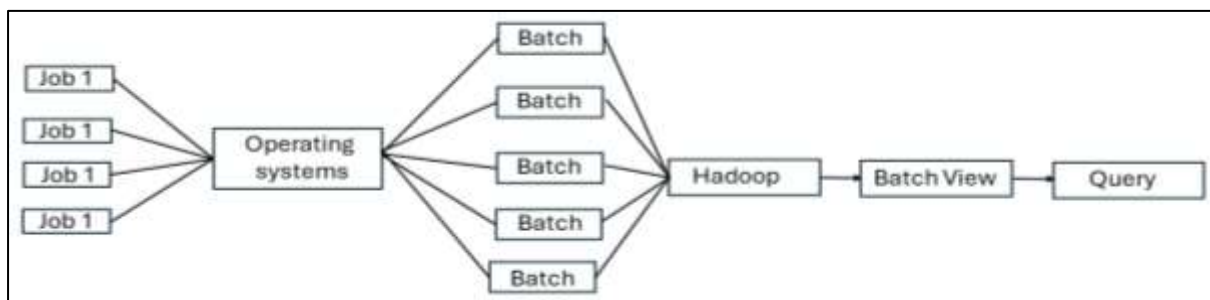


Figure 20. Batch processing (Balusamy et al., 2021, p.88)

Real-time message ingestion: message/data ingestion is normally stored by streaming buffering tools/message queuing semantics like Azure Event Hubs, Kafka, Pub/Sub, before they are sent to streaming tools, for further process or Lakehouse for storage and other purpose.

Stream processing: real-time messages received are processed by filtering, aggregating before written in an output sink. Apache Flink, Storm and Spark, GCP (Pub/Sub, Datastream, Dataflow) are all great tools for handling data with streaming access pattern. An example of real-time processing is shown in figure 21.

Analytical data store: Analytical data store can be queried using data analysis tools. Data is sometimes well prepared and already in a structured format. It can be a Kimball-style relational data warehouse (Microsoft, 2024). Examples of analytical store are Apache HBase, Hive and GCP (BigQuery).

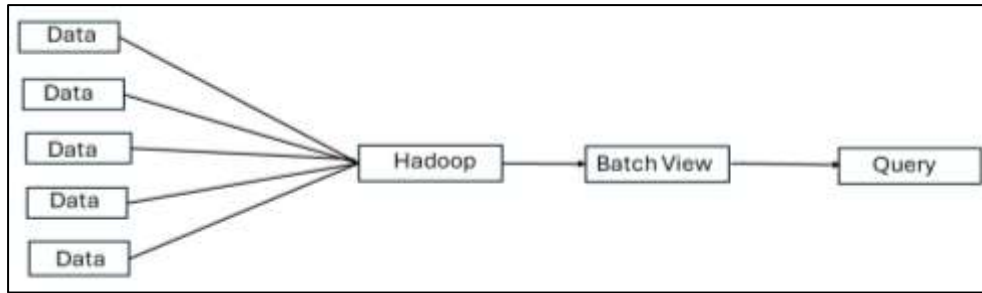


Figure 21. Real-time processing (Balusamy et al., 2021, p.89)

Analysis and reporting: the end goal of most big data solution is to empower users to analyse the data. Data modelling layer included in the architecture for multi-dimensional OLAP cube and support of self-service BI tools for visualization such as Excel.

Orchestration: it is extremely useful to automate repeated task, in a workflow. The process of data transformation and movement from multiple sources and sinks before they are loaded into an analytical data store or push of results to a dashboard can be orchestrated with Apache Oozie and Sqoop, GCP (Cloud Composer) and Azure Data Factory.

Two of the most popular data processing architecture over the year are the lambda and kappa.

1) The lambda architecture was proposed in 2015 by Nathan Marz to address the problem of batch and real-time analytics. Data processing in a lambda system flow through two paths:

A batch layer also known as the cold path, stores all incoming data in it raw form before performing any batch process and stores the result as a batch view in the serving layer. Data in the batch layer are more accurate but less timely.

The speed layer or hot path is designed for more timely data (low latency), data flowing through this path is analyse in real time but there is a potential of less accuracy.

However, results from both paths can be access at the analytics client application where they converge, as shown in figure 22. This architecture is good for fraudulent claims system and rapid clients feedback (Kalipe et al., 2019 ,p.3)

There are great benefits of the lambda architecture, it is a fault tolerant, scalable and serverless system, no server software installation or update and maintenance is required in it management and in the result of a system crash all historical data are managed by the batch layer (snowflake, 2024). There are great benefits of lambda architecture but according to snowflake (2024) some of its drawbacks are logic duplication that is maintaining separate code bases for batch and

streaming layers, batch processing inefficiencies and complexity. These drawbacks gave rise to the Kappa architecture proposed in 2014 by Jay Kreps on a blog post titled “Questioning the Lambda Architecture”.

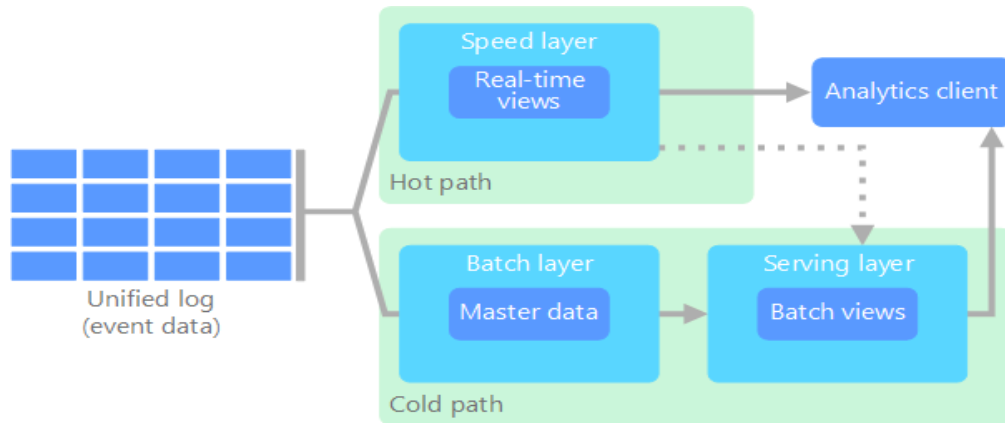


Figure 22. Lambda architecture (Tejada, 2024, [Microsoft.com](https://www.microsoft.com))

2) The Kappa architecture has the same goals as lambda architecture, but the difference lies in the data flow. Data in Kappa flow through a simple path, using a stream processing (Tejada, 2024, [Microsoft.com](https://www.microsoft.com)). Since Kappa architecture focuses on speed layer, it is suitable for real-time monitoring of fraud detection and IoT data processing etc. Kalipe et al., 2019 ,p.4)

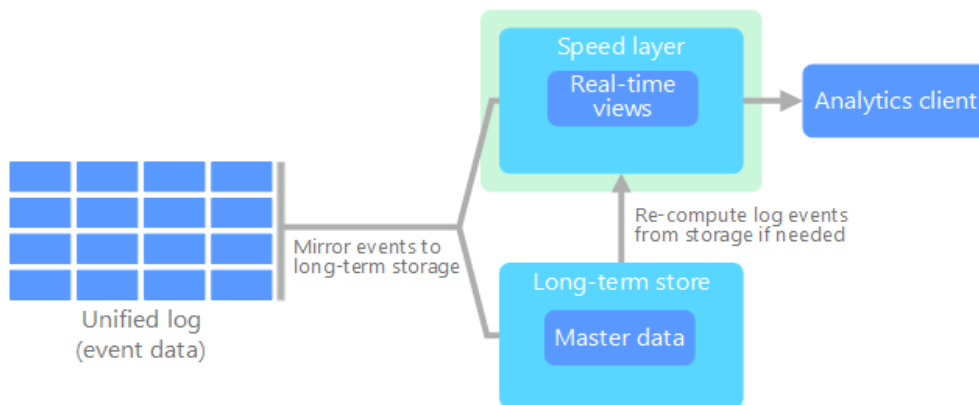


Figure 23. Kappa architecture (Tejada, 2024, [Microsoft.com](https://www.microsoft.com))

Kappa architecture has been adopted by companies like LinkedIn and Uber. However, some of its drawbacks are it is impossible to perform transactional operations, also streams with long Time to live (TTL) are not supported by native cloud services (Kalipe & Behera, 2019 ,p.4). Other limitations put forward by Roshan Naik (2019) at Uber Engineering Meetup in a presentation titled “*Kappa+ Architecture using Apache Flink*” are high cost of storing data in

Kafka compared to HDFS, are infeasibility of data retention beyond few days. Other BD architecture are Microservices and IoT (Kalipe & Behera, 2019, pp.4-7).

The main pillars of Hadoop are HDFS, YARN and MapReduce. Other tools can be grouped into the four different layers, the data storage which consist of HDFS and HBASE, data processing is made up of YARN and MapReduce, the data access consist of (HIVE, Pig, Mahout, Avro, SQOOP) and finally the data management layer (Oozie, Chukwa, Flume, Zookeeper). APACHE STORM and Spark and streaming tools which are part of Hadoop ecosystem but not core component. Apache Tez can serve as an alternative to Hadoop MapReduce, it allows the creation of complex direct acyclic graph (DAG) of task for data processing (Amazon aws, 2024). In the next section a brief overview of Apache Spark is given since it is gaining popularity. Figure 24 gives a high-level overview of Hadoop’s ecosystem.

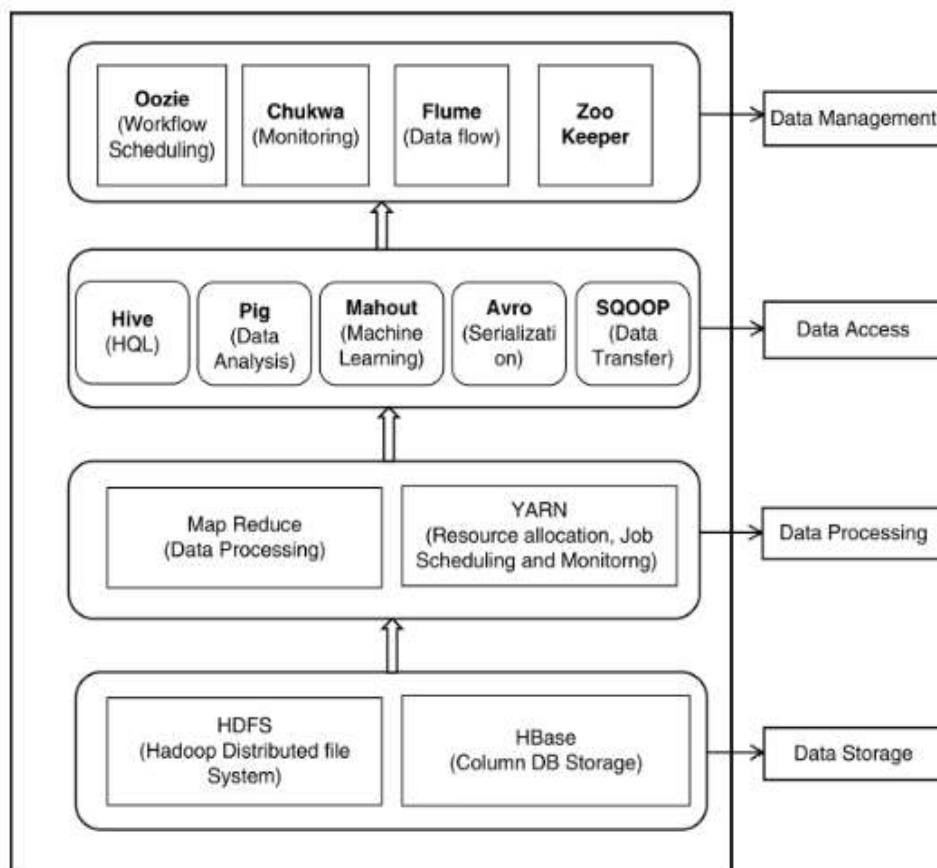


Figure 24. Hadoop ecosystem (Balusamy et al., 2021, p.113)

3.3 Apache Spark

Apache Spark was developed at UC Berkeley’s AMPLab in 2009 (IBM, 2009) as a research project focused on data-intensive application domains and is now managed by the ASF (Amazon aws, 2024). Spark is designed for streaming data analytics, graph analytics, fast interactive queries, and machine learning (Achari, 2015, p.152). Spark is compared an alternative to MapReduce but not a total replacement, and one of its key features is it run 100 times faster than MapReduce when running in-memory by using RDDs (Veith & Assuncao, 2018, p.1), and 10 times faster when running on disk (Achari, 2015, p.152). Spark was written in Scala, which makes Scala the most suitable language for doing big data analytics, but the framework includes APIs that support multiple programming languages such as Java, Python, SQL and R, making it popular among data professionals (analyst, scientist and engineers). Spark’s core computing engine can distribute task across cluster, schedules and monitor the process in real-time. It can do cluster management through “standalone scheduler,” Apache YARN, or Apache Mesos (Veith & Assuncao, 2018, p.1). Figure 25 shows Spark’s core framework.

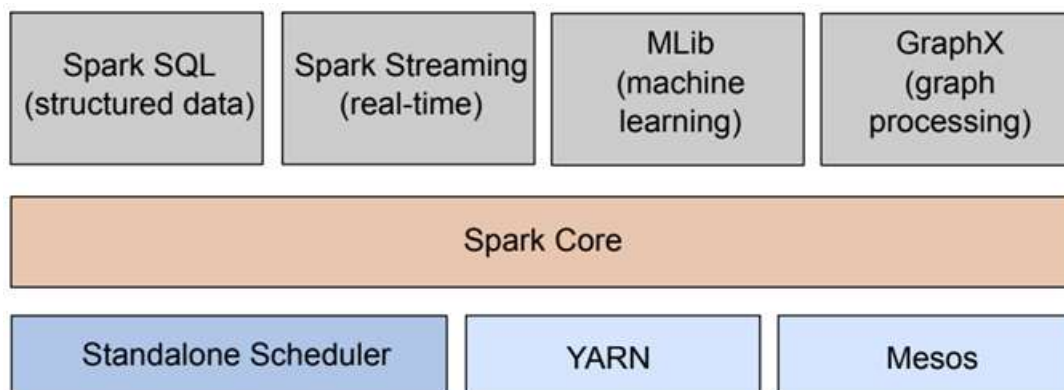


Figure 25. The Apache Spark Stack (Veith & Assuncao, 2018, p.2)

An overview of Spark’s core framework is explained below for in-depth knowledge on each component, check the official ASF documentation [Apache spark](#).

Spark SQL: a high level wrapper that transforms SQL queries into Spark jobs to produce desired results. It can work with variety of files such as JSON, Parquet, Hive tables, [Datasets](#) and [DataFrames](#) (Achari, 2015, p.154).

GraphX: designed for solving complex graph problems through graph-based algorithms and

integrates with graph databases. It built on the abstraction that extends RDD for graphs and graphs-parallel computation (IBM, 2024). Some of its applications are PageRank, Label propagation, and triangle count.

Mlib: a scalable machine learning library that works on top of Spark. It contains functionality such as learning algorithms for regression, classification, clustering and recommendation based collaborative filtering, and featurisation including selection, feature extraction, transformation and dimensionality reduction (Veith & Assuncao, 2018, p.4)

Spark Streaming: Spark’s streaming library enables scalable, fault-tolerant, high throughput processing of streaming data in real time. It can be well integrated with Mlib and GraphX to process their algorithms in streaming data. It is compatible with various streaming ingestion technologies such as Kafka, Flume, HDFS/S3, Kinesis and Twitter (Achari, 2015, p.154). The ingested streaming data is break into small batches and stored as an internal dataset (RDD) for processing, as shown in figure 26.

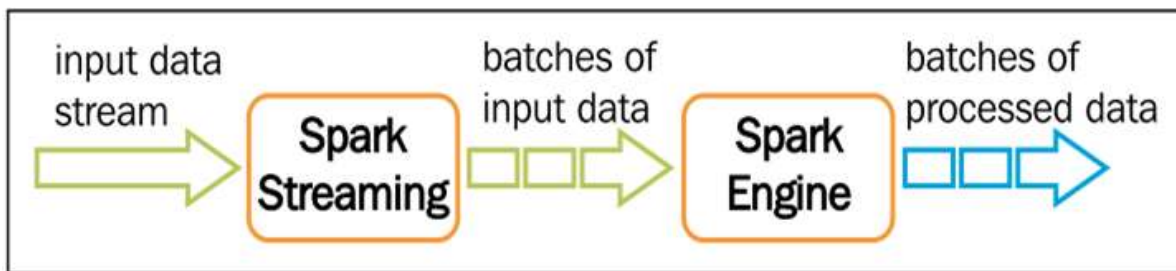


Figure 26. Streaming, batches of input data (Achari, 2015, p.154)

Big Data and Cloud are two of the evolving paradigm driving the revolution in various field of computing, while BD promotes the development of e-commerce, e-finance, telematics, smart cities and more, cloud computing has change the ways of storing, accessing, and manipulating the data by adopting new concepts of storage and moving computing and data closer through data locality (Balusamy et al., 2021, pp.93-95). According to Mell & Grance from NIST (2011), “Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.” Cloud computing has made storing and analysing BD cheaper and more cost-effective for organizations in terms of operation and

maintenance of in house or on-premises infrastructures, but there are drawbacks in terms of privacy and security (Balusamy et al., 2021, pp.93-95). Some major cloud service providers are Amazon, Microsoft, Google, Oracle and so on. This paper only focuses on Google Cloud. Cloud computing is classified into three types based on infrastructure (Balusamy et al., 2021, pp.93-95), they are public, private and hybrid. But Mell & Grance (2011) added a fourth type called “community cloud”. The **public cloud** is a service provided by third-party vendors over the internet (Balusamy et al., 2021, pp.93-95). It is open to use by the public on a pay-as-you-go model which significantly reduces the cost, since the cloud provider manages the maintenance of both hardware and software in their data centers. Examples of activities on a public cloud are saving documents to google drive. **Private cloud** or corporate cloud is meant only for the exclusive use by a single organization, controlling and maintaining its own datacentre, there is data security, but limitations associated with traditional IT environments. Private cloud can also be externally hosted with full guarantee of privacy. **Hybrid cloud** is a combination of public and private clouds, it has at least one public and one private cloud therefore resources are managed both in-house and external sources. **Community cloud** defined by NIST as a service “provisioned exclusively by a specific community of consumers from organizations that have shared concerns. It can be operated by one or more organizations in the community.”

Mell & Grance (2011) there are three different cloud service models namely software as a service SaaS, platform as a Service, and infrastructure as a service (IaaS), other cloud services arising are function as a service (FaaS) and anything as a Service (XaaS).

IaaS: in this model infrastructure such as servers, network, virtualization and storage are provided and managed the cloud provided and made available to the consumer through an API (RedHat, 2022). IaaS provides IT departments and developers with the highest level of flexibility and management control over your IT resources like what they are familiar with (aws, 2024).

PaaS: in PaaS remove the hassle of having to manage the underlying infrastructure (hardware and operating systems) allowing user to focus on deployment and management of application running on top of provided infrastructure (aws, 2024), it is primarily for developers and programmers (RedHat, 2024).

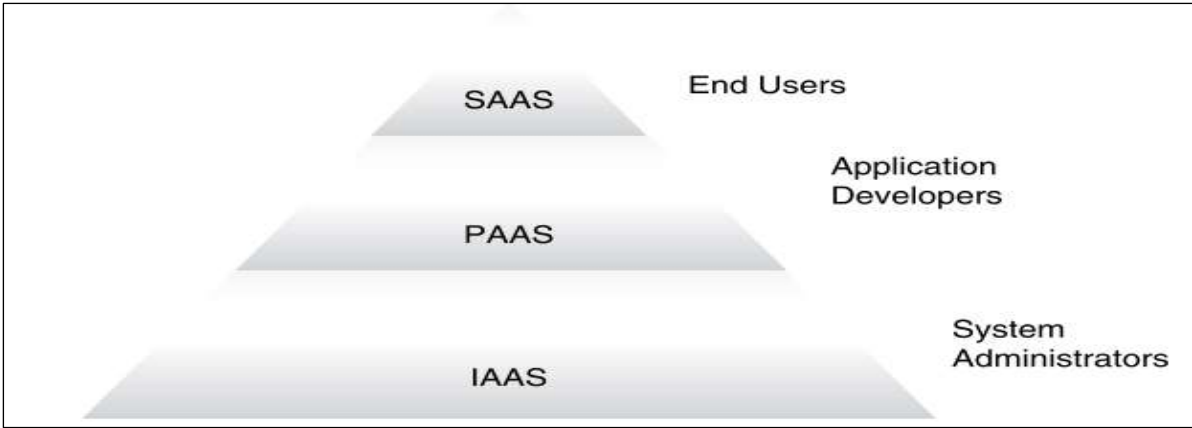


Figure 27. Service-oriented architecture (Balusamy et. al., 2021, p.96)

SaaS: this model is mostly referring to as end-user software application, it provides consumer with a complete product, nor worries about maintaining the underlying infrastructure (aws, 2024) or installation of application locally on individual user’s computer (RedHat, 2024).

3.4 Google Cloud Platform (GCP)

GCP is a suite of cloud computing services that provides a series of modular cloud services including data storage, computing, data analytics, AI and machine learning (wikipedia, 2024). Google cloud is scalable, reliable and cost-effective with data centers around the globe, making it easier to locate resources closer to clients and reduced latency (Google cloud, 2024). The suite of application that I used are Pub/Sub, BigQuery, Vertex AI, and Colab research which is an independent online jupyter notebook that can be integrated with GCP. Other GCP products that I have included are Dataflow, DataProc, Data Fusion, AlloyDB and cloud composer, which are useful in designing and automating any batch or streaming analytics system on GCP.

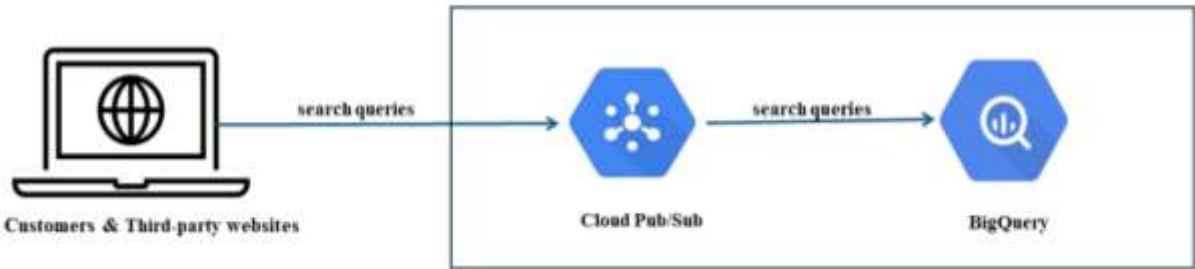


Figure 28. Easy Market GCP ecosystem (self-contribution, 2024)

Below are Google’s own very definition of Pub/Sub, BigQuery, Colab, Vertex AI, Dataflow, DataProc, Data Fusion and cloud composer on Google Cloud Tech YouTube channel.

Pub/Sub: According to Google Cloud (2024) Pub/Sub is an asynchronous messaging service with latencies of about 100 milliseconds that helps your tools to send, receive, and filter events or data streams. It has a durable message storage scalable in-order message delivery consistently high availability and performance at any scale. It runs on any Google cloud in the world. Pub/Sub does not need to be provisioned because it scales global data delivery automatically from zero to millions of messages per second. Pub/Sub can be set up between services and applications by defining a topic/s and then subscriptions which allow services to receive the messages published on those topics, making one-to-many communications gets much simpler enabling the spread of batch image analysis over multiple workers or send logs from your security system to archiving, processing, and analytics services or stream data into BigQuery or Dataflow for intelligent processing. Pub/Sub is said to be very handy for notification when something bad happens example system failure or service downtime. Other than streaming analytics pub/sub can be a middleware for integration of simple communication medium for modern microservices. In big data analytics Pub/Sub is often integrated with GCP orchestration tool like Composer, and Application integration (iPaaS) which provides trigger to start integrations. Pub/Sub is GCP's Kafka. At the time of the internship Easy Market uses Pub/Sub to ingest data into BigQuery from its website (lol.travel) and third-party customers to which it delivers API as a service.

BigQuery: A major part of my time was spent doing analysis with BigQuery and Colab research. BigQuery is an enterprise data warehouse designed for ingestion, storage, analysis and visualization of data with ease. Data can be uploaded to BigQuery in batch or by streaming data directly, enabling real-time insights. It is a fully managed data warehouse, eliminating the burden of having to take care of the infrastructure, while focusing on analysing up to petabyte scale of data. It supports standard SQL dialect that is ANSI compliant, with a nice Cloud Console UI for interacting with data and running complex analysis. It also supports API libraries like pandas-gbq with python which you can use to pull data, perform complex wrangling and analysis, then load back to BigQuery a process which data engineer called ETL. It is quite easy to integrate BI tools for turning complex data into compelling stories. The pricing model is quite simple customers pay for data storage, streaming inserts, and querying data, while loading and exporting data are free of charge. Storage costs are based on the amount of data stored, and you can decide to pay for queries or a flat rate for dedicated resources. At the time of the writing, Easy Market is spending an average monthly cost of 200euros for

storage and 750euros for query as said by the Dr. Manuele Bastianelli head of IT department. BigQuery can also be used for training and deploying ML models with ease as it provides features for novice by using their preexisting SQL skills. Another important feature is analysing geographic data through GIS for critical business decisions that revolves around location.

Colab Research: a cloud-based platform specifically designed for machine learning research and education. It provides a jupyter notebook environment that you access directly from your web browser, eliminating the need for local setup. The key features are free access to computing resources, including GPUs and TPUs, collaboration, pre-installed libraries. It can be integrated with most GCP suite products making it suitable for enterprise deployment (Bard AI, 2024). The data exploratory phase of Easy Market was done using Colab.

Vertex AI: an end-to-end platform that helps both developers and data scientist accelerate the delivery of ML models and applications to production. It enables more people to innovate with ML with low code tools and advanced capabilities for custom development on fully managed infrastructure. It has all the features in Colab Research and can be used as an alternative for jupyter notebook on GCP. It gives access to task specific APIs and AutoML foundation models from Google Research, and a variety of third-party models. After initial development of the code on Colab Research, Vertex AI workbench was used for deployment.

Dataflow: a unified programming model, serverless fast, and cost-effective data-processing service for stream and batch data which removes operational overhead by automating the infrastructure provisioning, and auto-scaling as your data grows. The process of dataflow is easy, data can be read from a source like Pub/Sub, transform and written back into a sink. There's portability with processing pipeline created using open-source Apache Beam libraries in the developers' language of choice and applying it as dataflow job. It offers Cloud Console UI, APIs, prebuilt or custom templates, SQL statements to develop pipelines right from BigQuery UI, notebooks. All data running through pipelines are well encrypted.

AlloyDB: a fully managed PostgreSQL-compatible database service offered by Google Cloud. Some of its key features are high availability, high performance and scalability and security.

DataProc: a managed service for any OSS jobs that support big data processing including ETL and machine learning. DataProc provides support for the most popular open-source software

(Hadoop, Spark, Flink, Jupyter, Presto, Pig, Hive). On-premises data cluster can be migrated using DataProc to maximize efficiency and enable scale or use it with Cloud AI Notebook or BigQuery to build an end-to-end data science environment. It can auto-scale cluster in just 90 seconds, with management of creation and monitoring and job orchestration.

Data Fusion: A fully managed point-and-click enterprise data integration and ingestion tool, which helps customers build data marts, data warehouses and data lakes fast. It provides users a code-free environment to deploy ETL/ELT data pipelines. It is powered by CDAP OSP.

Cloud Composer: A fully managed workflow orchestration service built on Apache Airflow. It helps you create, schedule, monitor and manage workflows. Composer workflows can connect data processing and services on GCP, public clouds and on-premises environments. Cloud Composer enables users to create pipeline workflows using DAGs.

3.5 Running Hadoop and Spark on GCP's Dataproc

While Hadoop used to be the go-to for BD problems mostly on-premise, cloud vendors like Google and Amazon have provided alternative solutions but have not totally abandoned Hadoop. GCP supports running Hadoop in the cloud and has made it quite easy to migrate from on-premise to the cloud. Technologies such as HBase can be migrated on GCP's Bigtable, Hadoop Jobs and Spark clusters can be run using Dataproc. Some of the benefits include, scaling and efficiency, modernizing data processing pipelines, managed hardware & configuration, simplified version management, and flexible job configuration (Google Cloud, 2024).

4. Automating Look-To-Book Computation

One of Easy Market's main goal is to increase customer conversion rate by increasing its L2B ratio of flights bookings on its website (lol.travel) and third-party website it is powering. Lufthansa Group (2020) defined L2B ratio as a figure that shows the percentage of people who visit a travel website compared to those who make a purchase. This service can be so costly that most companies providing APIs to third-party have laid some restriction on how many times an agent can verify the availability of flights in their system because sending too many queries at once consumes the server bandwidth, increase cost and response time. For example, in 2014 TAP Portugal allows a look-to-book of 400:1 (TAP Portugal, 2014), Hervé Couturier Head of R&D Amadeus IT Group also gave an estimate of 1000:1 ratio in a blog titled "Solving the challenge of ever increasing flight search volumes" detailing the exponential growth of customers searching for airline products that have led to the massive increase in volume of shopping transactions that their system has to deal with. Amadeus IT Group with help of GCP developed the Amadeus Airline Cloud Availability, "piloted by Lufthansa via GCP IaaS, enabling instances of all the revenue management and airline data logic to be deployed in the cloud to serve local demand on all continents (Couturier, 2015)." In hotel hospitality industry L2B ratio depends on the type of property, for example High-end luxury hotels might have a lower L2B because some visitors might just be browsing out of curiosity, not necessarily looking to book, while Mid-tier hotels typically have a L2B ratio between 2-7% (Bard AI, 2024). Report from IATA (2019) after interviewing industry professional identified these challenges, which are cost, IT infrastructure and responses time. In the report IATA gave the following solutions that have been implemented by industry players:

- a. Provision of relevant and consistent information to travellers at the right time
- b. Working with MSEs/OTAs to optimize L2B
- c. Filtering out robots and limiting the deployment of result-based caches
- d. Implementing Airline Profiles

Easy Market is constantly working on its system and providing the best optimal solution to customer but some of its processes are still done manually by its engineers and want to automate and save time. So, in this section of the thesis, I will go through a list of processes that I tried to develop to perform series of operations in ETL and L2B computation on over 70million rows of data ingested to GCP BigQuery daily from their website, travel agencies and other third-party customers. There was a lot of coding that I will explain in detail, so this section is arranged

in the following order: structure of the raw data (semantic), ETL process, computation of values, cost of computation, propose solution.

4.1 Structure of the raw data (sematic)

Data ingested into BigQ from Pub/Sub has no prior analytics, and therefore reach in form. The main table which I will be referring to as flight-lake for the purpose of this writing because all the data is raw messy and untouched. There is no PII in the table and it consist of 26 main fields and nested fields as type “record” in the “gdsSearchConditions.” To put it straight a “record” or “struct” is way to represent complex data structures efficiently (Bard AI, 2024). The first field is “created” which is a TIMESTAMP that store the exact datetime including seconds of when an “itemOperation” is generated. The itemOperation are search, booking and price all type String and abbreviated as SRC, BRC, PRC, respectively. SRC=searchTransactionId, BRC=bookingTransactionId, PRC=pricingTransactionId holds a crucial information about the whole L2b (SRC to BRC) computation. Each itemOperation has a unique ID that stores a unique transaction, and duplicate SRC can be found when complex queries are performed to see the results sent by various engines available. The storefrontId is an INTEGER data type that holds information about the market type of Easy Market, they are b2b [2 = Revolution.travel, 63 = BOL] and b2c [49 = VIAGGIOGRATIS, 54 = LOL], the next field is the agencyCode which self-explanatory and STRING in type, the market field stores names of various travelling agencies e.g. AMICA, UVET and GEOTRAVEL etc. Some BOOLEAN variables are directFlightsOnly, allowTOFares, and offersWithFreeBagsOnly. The gdsSearchConditions is of type RECORD storing information from the Global Distribution System about serviceEngineId, flights, and solutions or response from the suppliers. This field is further nested into sub structures showing the hierarchy of which data is stored and providing a comprehensive logic of complex system operation in the cloud.

4.2 Extract Transform Load (ETL) process

There are two processes in data transformation, ETL which has been around since the 1970s and ELT the new norm (Amazon aws, 2024), both can be used for complex analytics and one of the main differences is that ETL performs transformation before loading the data into target Datawarehouse, lake, table. Other differences and benefits of when to use one instead of the other in certain situation are in terms of cost ELT is cheaper compared to ETL because it requires fewer systems to be built, which also reduce the cost of maintenance. In terms of speed ELT have proven to be faster and uses cloud infrastructure such as processing power and

parallelization for real/near-time data transformation, ETL in the other hand slows down the system as data size increases. In this paper ETL was adopted because of its suitability for experimenting and research purpose. Easy Market uses BigQ as a storage and for performing complex analytics with SQL. The ETL system built consist of three phases; the first is “extraction” which involves getting the data from BigQ to Vertex AI workbench using python and SQL, the second is “transformation” using set of files as dependencies and python scripts including over 200 lines of code to perform cleaning, wrangling, and more. The final phase is populating the cleaned structured dataset to the ETL warehouse. A high-level overview of the ETL process is shown in figure 29. The next section takes a deep dive into the schema of the target table(ETL warehouse), python & SQL code for extraction, supporting files and python scripts for “TRANSFORMATION” and load.

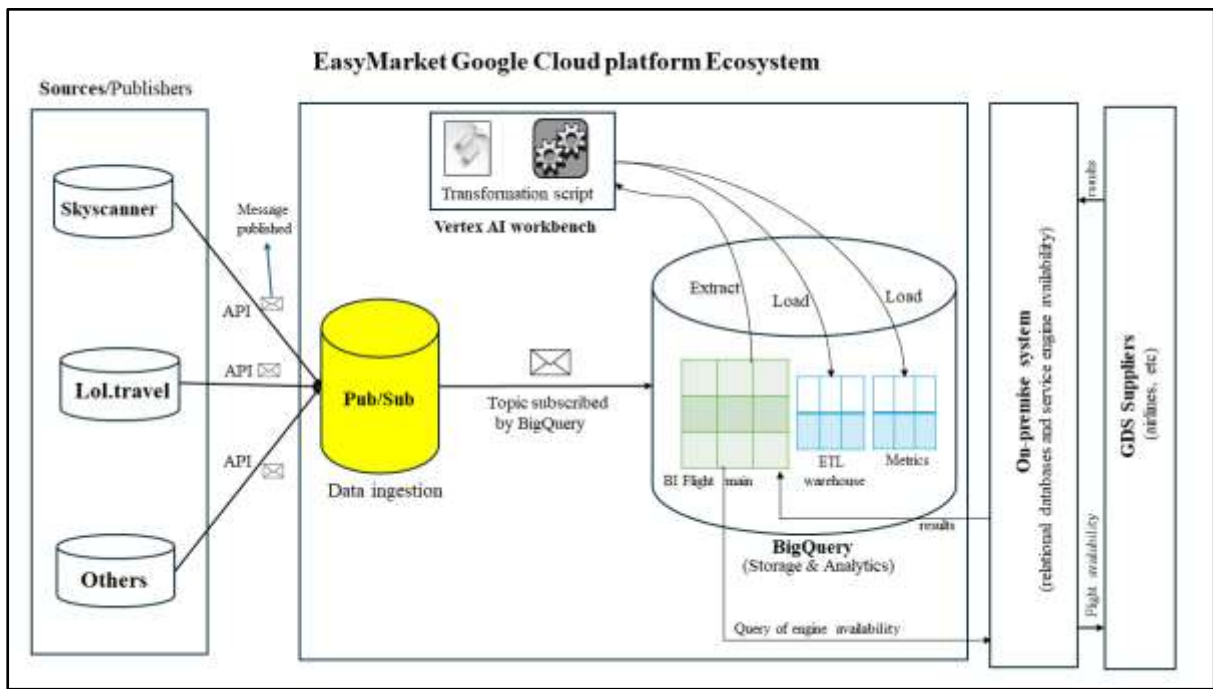


Figure 29. ETL process (self-contribution, 2024)

The ETL warehouse schema was design as requested by the IT business requirement analysis. This involves creating a date dimension with year, month, day all separated in different column. Most traditional BI analysis involve modelling a multi-dimensional data model to organize data in a data warehouses. A start schema is one of the most popular multi-dimensional models in the BI world and has therefore been adopted for the purpose of this writing. A point to note is that Easy Market’s BigQ storage is not relational database and the kind of analytics they are performing does not require the use of star schema. So, breaking down columns from the main

BigQ table into smaller tables will require joins to query and unnecessary storage due to data duplication, which may lead to increment in cost. Another point to mention is that traditional BI softwares like excel can only handle 1,048,576 million rows per spreadsheet even though power query is an ETL tool for big data and does not have a hard row limit. The ease of representing data in one big table like spreadsheet makes performing complex queries and analytics faster because of less joins, beside the rows of data ingested into BigQ each day is over 70million, therefore visualizing the look-to-book ratio for each route is cumbersome and Easy Market was not interest in seeing dashboards. However, the multidimensional star schema model has laid the very foundation of the one big final table is easy to query and perform complex analytics with less joins. The model created from the main source table has a fact table name flights which stores the information of measures and dimension tables (date/created, storefrontId, itemOperation, gdsSearchConditions) which adds description the to the fact table. The created/date dimension allows to perform more effective analysis across different time periods, while the itemOperation dimension provides information about the kind of operation perform it can be a search, pricing or booking. The other two dimensions gdsSearchCondition and storefrontId provides information about the result from the suppliers, and the platform/website from which the flight transaction came from it could be a b2b or b2c example of storefront are Revolution, LOL, BOL and ViaggioGratis. The figure 30 depicts the model.

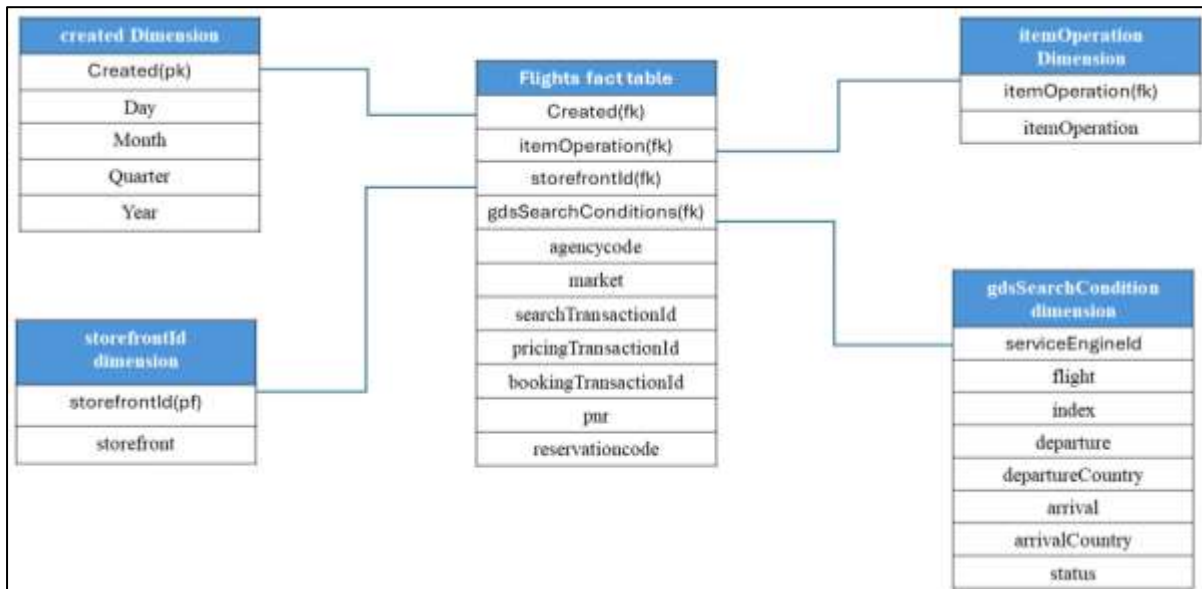


Figure 30. Flights Star model (self-contribution, 2024)

The figure 31 shows the practical implementation of the warehouse schema on GCP.

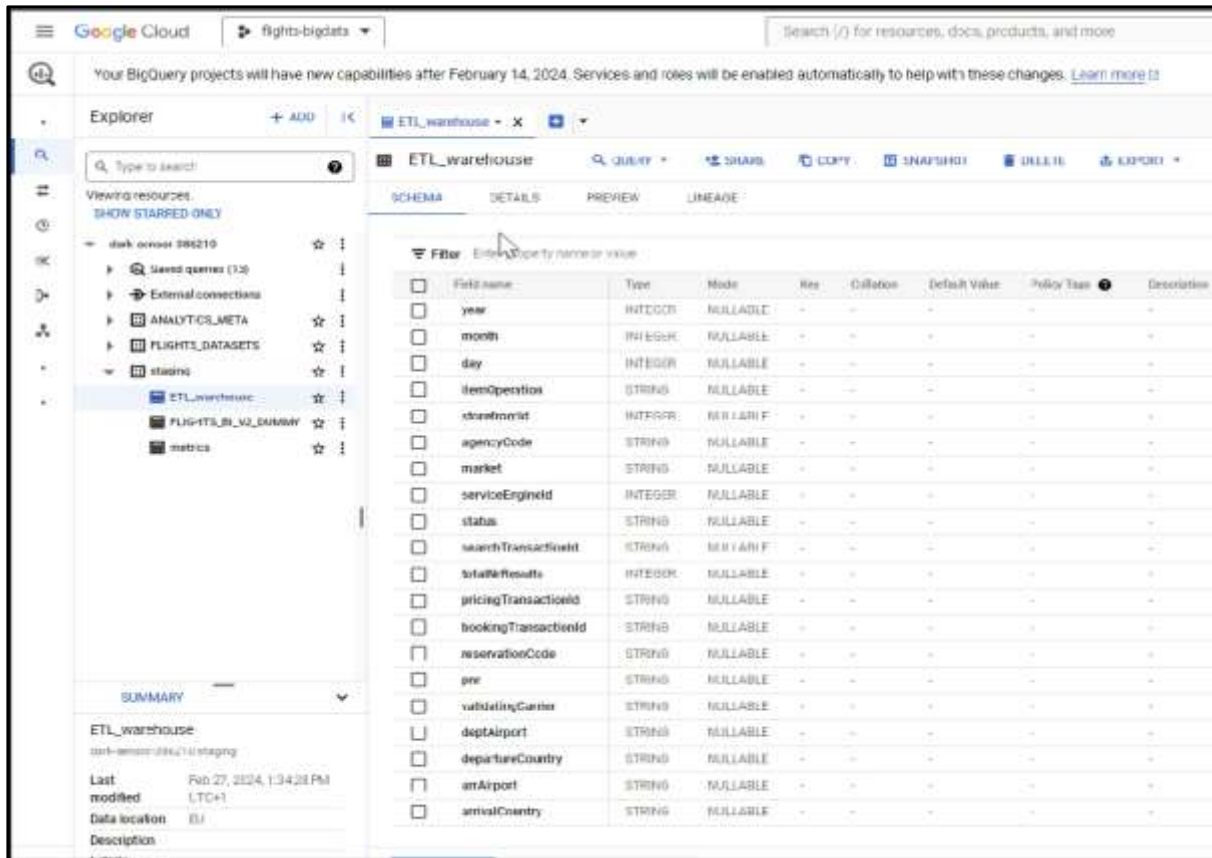


Figure 31. ETL warehouse schema (self-contribution, 2024)

After designing the model and the ETL warehouse schema, the next step of was populating it with data, in other to do such operation data needs to be extracted from the main storage which in this situation is from BigQ to Vertex AI workbench. The following are the steps in the ETL:

Step 1. Getting the data

The extraction process is done with a combination of Python & SQL, the sql code was written in a text file and read by python as normal string then saved in a variable called 'sql.' The BigQuery API client library google-cloud-bigquery was used to query data into Vertex AI workbench (Jupyter notebook) and convert into a data frame, a temporary copy was created just in case if there is an error during the computation, the copy will then be used to avoid rerunning the sql. The query can be run on Vertex AI workbench with no authentication requirement, however the same does not apply for Colab and local host which requires specific authentication requirements such as project name, location where the data is store in the case of Easy Market it is europe, users might also be required to use their Gmail to verify their access to a specific table in a project. Below is a snapshot of the code snippet of extracting the data from BigQ to Vertex AI using Python & Sql.

The code snippet is a sql query for a single day in the month of February 2024.

```
SELECT
    created,
    itemOperation,
    storefrontId,
    agencyCode,
    market,
    searchTransactionId,
    pricingTransactionId,
    bookingTransactionId,
    reservationCode,
    pnr,
    supp.serviceEngineId,
    supp.status as status,
    bi.totalNrResults,
    supp.solutions[SAFE_OFFSET(0)].validatingCarrier as validatingCarrier,
    supp.flights[SAFE_OFFSET(0)].departure as deptAirport,
    supp.flights[SAFE_OFFSET(0)].departureCountry as departureCountry,
    supp.flights[SAFE_OFFSET(0)].arrival as arrAirport,
    supp.flights[SAFE_OFFSET(0)].arrivalCountry as arrivalCountry,
FROM `dark-sensor-386210.FLIGHTS_DATASETS.FLIGHTS_BI_V2` bi,
UNNEST(gdsSearchConditions) as supp
WHERE TIMESTAMP_TRUNC(created, DAY) = TIMESTAMP("2024-02-21")
and supp.flights[SAFE_OFFSET(0)].departure is not null
and supp.flights[SAFE_OFFSET(0)].arrival is not null
group by
    created,
    itemOperation,
    storefrontId,
    agencyCode,
    market,
    pnr,
    supp.status,
    supp.serviceEngineId,
    searchTransactionId,
    bookingTransactionId,
    reservationCode,
    pricingTransactionId,
    supp.status,
    totalNrResults,
    supp.solutions[SAFE_OFFSET(0)].validatingCarrier,
    supp.flights[SAFE_OFFSET(0)].departure,
    supp.flights[SAFE_OFFSET(0)].departureCountry,
    supp.flights[SAFE_OFFSET(0)].arrival,
    supp.flights[SAFE_OFFSET(0)].arrivalCountry
```

The code snippet is a python program to read the sql code from a text file and perform a query from BigQ.

```
def query(filepath):
    # the method reads query from a text file
    with open(filepath, 'r') as q:
        query = q.read()
    return query

sql = query(files[files.index('query.txt')])
# --- Extract Transform Load module---
# --- Authenticate :----
# @title Setup
# Project ID inserted based on the query results selected to explore
project = 'dark-sensor-386210'
location = 'EU' # Location inserted based on the query results selected to
explore
client = bigquery.Client(project=project, location=location)
data_table.enable_dataframe_formatter()
auth.authenticate_user()
# --- read data from big query ---
query_job = client.query(sql)
df1 = pd.DataFrame(query_job.result().to_dataframe())
# create a copy of the dataframe
df = df1.copy()
```

Step 2. Transformation

The next phase after importing data into Vertex AI workbench in a pandas dataframe format is to perform the transformation. The transformation code is organised in a module and contains four main functions “datedim,” fillMissingCountryDep, fillMissingCountryArr, and wrangle:

a. datedim

The “datedim” which is the first function takes a data frame as an input, import pandas and splits the ‘created’ column into year, month and day, this is done to perform complex analysis and having the ease to select a specific year, month or day overtime a specific time. A reindexing is performed to keep the data frame in order with the year, month, day, itemOperation and storefrontId as the first five columns followed by the remaining 14 columns. This order is done according to the IT department’s requirement of how they want to see the table so that they can easily query it for results. The data frame with date separated is then sent to the stage which is data imputation for filling arrival and departure of missing country.

The imputation of missing values in departure and arrival is done with two functions `fillMissingCountryDep` and `fillMissingCountryArr`. They both have the same underlining processing logic, so only one of them will be explained in this paper.

b. `fillMissingCountryDep`

The `fillMissingCountryDep` function will be the one to consider because whenever someone is travelling from one point to another the departure always comes first before the arrival. The `fillMissingCountryDep` function takes in two parameters as inputs, the first parameter is the dataframe from the previous stage which is the output of the `datedim` function and the second is the IATA data containing airports and countries. The IATA data has two columns the abbreviation of departure airport and country, these columns are then split and converted into two separate list called `city` and `country`. The python `zip` function is then used to create a tuple of each city and country example `((MIL,IT), (NAP,IT),(ROM,IT))` then a dict function is applied to create key-value pairs. The next step is a new empty list “`newDC`” is created to hold the new values for both present and missing values imputed. A for loop is applied to the data frame departure airport column to checks the value’s key and if it is equal to ‘NA’ the function appends “NA” because Namibia in the IATA country is represented as “NA.” One thing to note is that in most programming languages “NA” is normally refers to as a missing value so an escape character can be used to handle such problem. If the key differs from “NA” then append the key/country IATA code. The function also uses a brute force approach to fill in missing value if an airport has not been represent in the IATA code, which could be a possibility that the airport is out of operation or it is a new airport, the value to impute in departure country column is an arbitrary one “`checkDeptairport`”, meaning if anyone is using the dataset they can find the name of the departure country by using the name of the departure airport. The function might not be the best or most optimal code, but the logical process and practical implementation worked well in imputing all the missing values and took few seconds to impute all missing values for departure and arrival country in over 70million rows of data. The use of python dictionary `{key: value}` data structure to solve such complex task makes the process fast.

c. `fillMissingCountryArr`

The same logic of imputing the missing departure country holds for arrival country, therefore the explanation of the `fillMissingCountryArr` is the same as `fillMissingCountryDep` with only arrival airports and country to consider.

d. `wrangle`

The last function in the python transformation script is `wrangle` function which takes the data frame from the previous steps and IATA code as inputs, there is a little bit of repetition of the

fillMissingCountryDep and fillMissingCountryArr logic this is due to fact of making the code robust. However, the difference in the wrangle function is that all missing departure and arrival row tuples are subset before an imputation is made. That is set “checkDeptairport” for missing value in departure country and “checkarrAirport” for missing value in the arrival country. Other data wrangling process includes filling “status” column as “OK” whenever the tuple is blank for this column. Note in most literatures and data related book a missing value is said to be problematic but in Easy Market’s system in some column missing values are ok, this might sound contradicting but that is how their design work. Blank or missing values in the “reservationCode” means a particular itemOperation does not lead to reservation so the value is set to “noRes” short for no reservation, as for “validatingCarrier” it stores all the names and codes for a specific airline flight, and all missing values in the “market” column are set to the arbitrary string “market” or “REvolution" when storefrontId == 2. Imputing “REvolution" for storefrontId == 2 is done to give a higher preference to such market because it is one of the most recent innovations of Easy Market. After this sequence of processes are performed the final cleaned dataframe is return and written into the ETL warehouse. The figure 32 shows a high level overview of the transformation process.

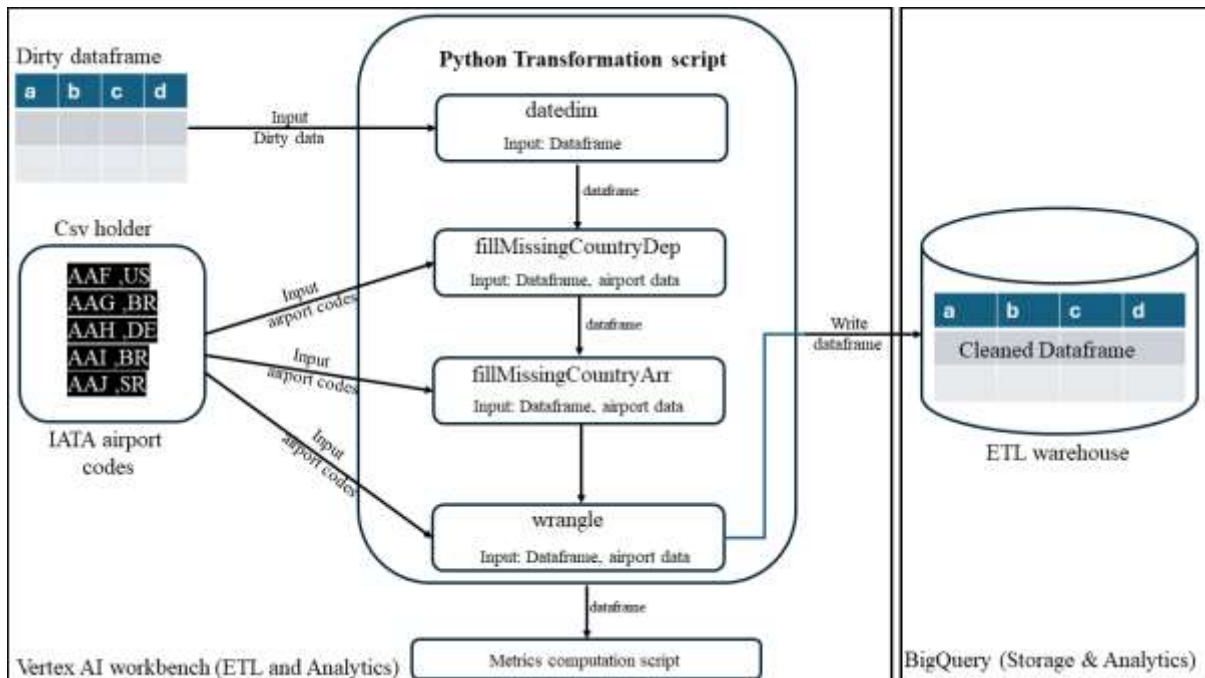


Figure 32. Python transformation script (self-contribution, 2024)

The above diagram summarizes the whole process that is explained above one thing to note is that the script is designed to perform computation for a specific day, but adjustments can be

made for the program to be able to perform weekly, monthly, quarterly or annually according to how the business dynamics changes overtime. The IATA codes can also be updated directly by editing it directly since airport and country code are store in a csv file format with two separate columns, respectively. Once the data has gone through all the function the module it can be written directly to the ETL warehouse for storing and performing analytics for later use. It can also be passed on to another script called the “metrics computation script” for calculating the look-to-book ratio and the search-to-price ratio. The process of doing this metrics computation is discussed in detail in the next section.

Step 3. Computation of metric values

The search-to-price and look-to-book are some of the most important ratios in the airline industry. The look-to-book (conversion rate) is often more widely discuss because of its importance. However, in the computational process and in this paper both are considered.

While the look-to-book has to do with the “customer conversion rate” the search-to-price ratio shows if there was a pricing for a specific route, and pricing can only be return when a route is not in BLACKLIST (temporarily blocked). Like most business Easy Market spend a lot of time doing complex analysis to compute it look-to-book ratio for some of it most profitable routes, and drills down into its data storage on BigQ for new emerging travelling destinations and previously blocked unprofitable ones, this is what makes this section really important.

The computation process can be done in two ways, one by writing a SQL script to query data directly from the ETL warehouse which contains cleaned dataframe or passing by cleaned dataset from the transformation script to the metrics computation script directly. The second process was adopted in the computation process to save time and energy of having to write different SQL script that one to query the main storage of raw data the other to ETL warehouse. As the cleaned dataset is passed on to the metrics computation scripts series of process are perform using NumPy, pandas and datetime module. The first step of these process is a sanity check for missing values in the airport columns that is in both departure/ arrival names of cities and countries, then a string concatenation is perform on these names to create a new column for the airport route called “routeAirport” and country route “routeCountry” example “MIL-ROM”, “IT-IT” respectively. A “.groupby” function is then applied to a subset of columns including “routeAirport” before chaining “.count” pandas’ function to a count of the number of ‘searchTransactionId,’ ‘pricingTransactionId’, ‘bookingTransactionId’. The result from the previous step is then chained with the “.sort_values” to order the dataframe by

'bookingTransactionId' in descending order. Two new columns df['S2P'] and df['L2B'] are then created from the count of the 'searchTransactionId', 'pricingTransactionId', 'bookingTransactionId' by applying the ".map(str)" function. The datetime is recomputed to fit the exact row and avoid mismatch, lastly the columns are manually reindexed for populating into the BigQ metrics table for later analysis.

A high level overview of the metrics computation process and snapshot of the code snippet:

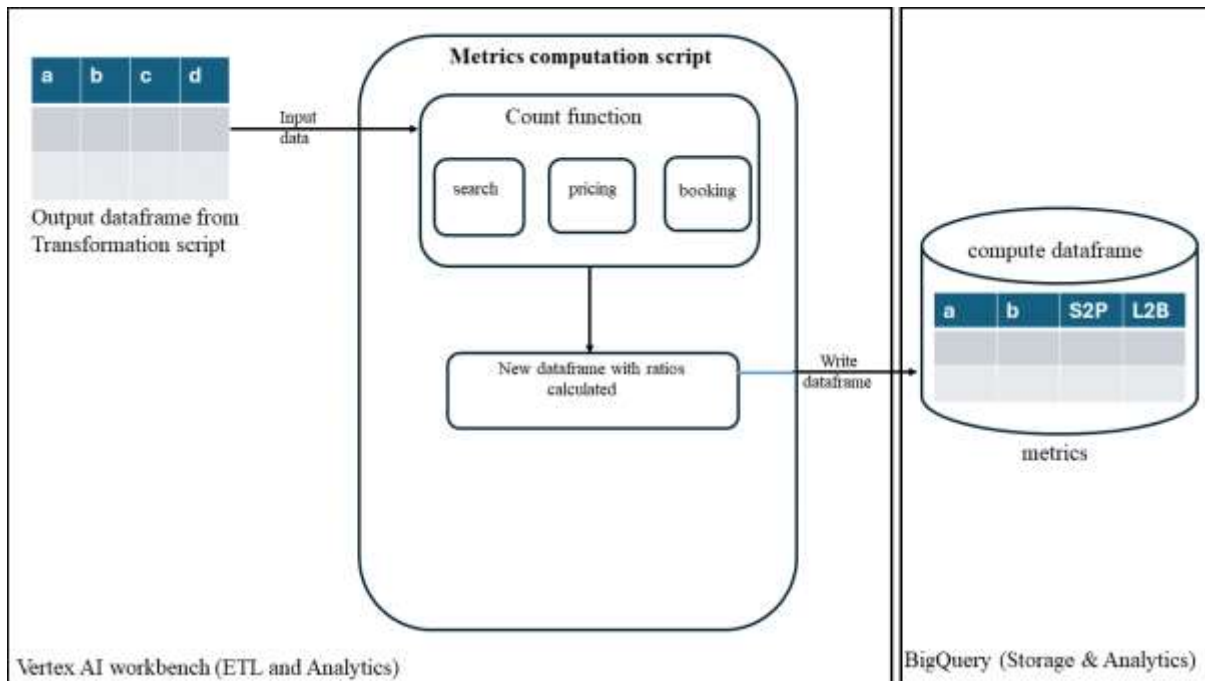


Figure 33. Metrics computation script (self-contribution, 2024)

code snippet:

```

# initialize original data dataframe
airPortRoute = dataframe.groupby(['routeId', 'airport', 'agencyCode', 'market', 'serviceId', 'totalResults'])
    .agg({'searchTransactionId': 'count', 'pricingTransactionId': 'count', 'bookingTransactionId': 'count'})
# sort by bookingTransactionId in descending order
airPortRoute = airPortRoute.sort_values(by='bookingTransactionId', ascending=False)
# join the count of searchTransactionId and pricingTransactionId to form a ratio
airPortRoute['S2P'] = airPortRoute['searchTransactionId'].map(
    str) + "/" + airPortRoute['pricingTransactionId'].map(
    str)
airPortRoute['L2B'] = airPortRoute['searchTransactionId'].map(
    str) + "/" + airPortRoute['bookingTransactionId'].map(
    str)
# sort dataframe by a column and save
airPortRoute = airPortRoute.sort_values(
    by='bookingTransactionId', ascending=False)
# initialize an empty dataframe
airPortRoute = airPortRoute.reset_index()

# add the datetime
# import datetime module
from datetime import date

# assign to year, month, day
year = dataframe.iloc[0, 0]
month = dataframe.iloc[0, 1]
day = dataframe.iloc[0, 2]
# form date = datetime
data_subject = date(year, month, day)
empty = []
# iterate through the len of the computed dataframe (length range) and append just the first date of the original dataframe
for i in range(airPortRoute.shape[0]):
    empty.append(data_subject)

# assign a new column
airPortRoute['created'] = empty
# initialize computation by datetime
airPortRoute['created'] = pd.to_datetime(airPortRoute['created'])
# split the data column to year, month and day
airPortRoute['year'] = airPortRoute['created'].map(lambda x: x.year)
airPortRoute['month'] = airPortRoute['created'].map(lambda x: x.month)
airPortRoute['day'] = airPortRoute['created'].map(lambda x: x.day)

```

4.3 Cost of the computation

The aim of any business is to reduce cost and increase profit, the same applies for Easy Market throughout its research and development in the IT department. Decision about how many monthly queries to run on BigQ or what GCP product to add to its suite of applications already in use is carefully taken. Easy Market already spends a monthly average €950 on BigQ with €750 for queries and €200 for storage.

The cost of carrying out the computational experiment is €11.28, all scripts were tested and ran on Colab research Pro. The queries were made directly from the main storage for three days performing a computation of over 210 million rows of Easy Market flight data. After a successful deployment of the code with Colab, GCP Vertex AI workbench with a basic free plan was used for final deployment on the subset of the main data in the staging area set for research and development.

4.4 Proposed Solution

The diagram below is my solution for Easy Market to handle its system's optimization problem.

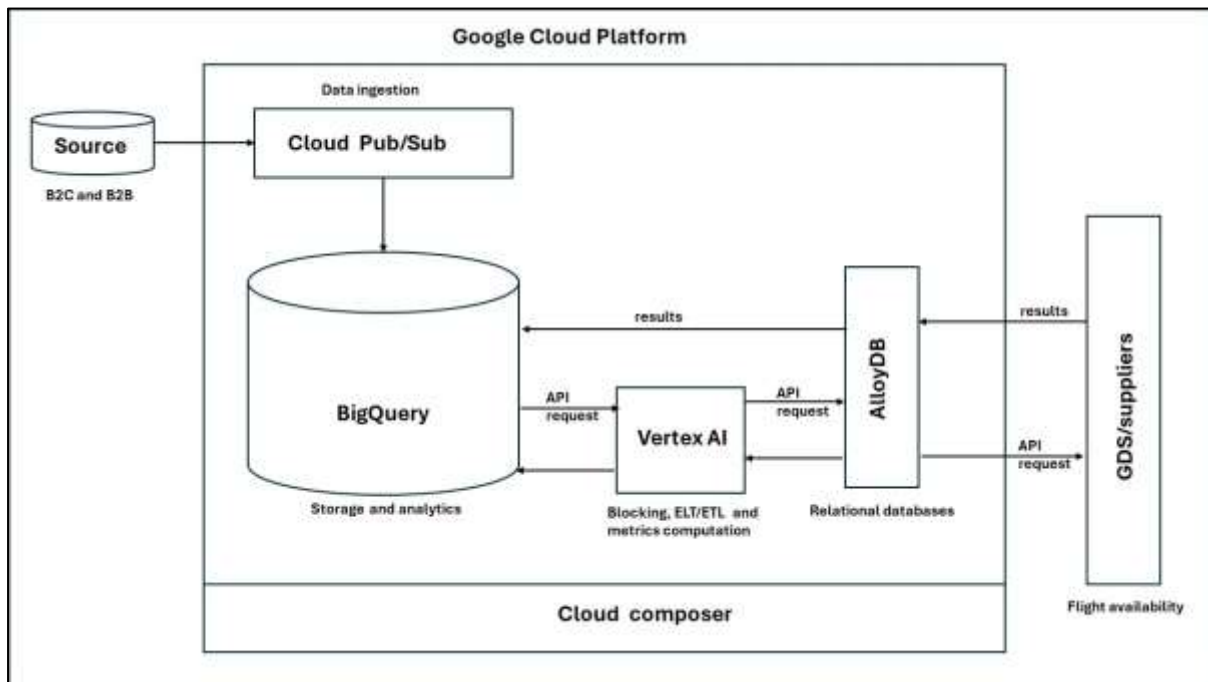


Figure 34. Cloud Proposal architecture (self-contribution, 2024)

This system will automate the data engineer's task through series of pipeline. In the design, Cloud Pub/sub will be ingesting data from all sources into BigQuery then a series of pipeline will be created to pull/store data from BigQuery into Vertex AI where ETL/ELT and metrics

computation will be performed, Easy Market should migrate their on-premises tables (filter engines) which they are using to filter transactions on the cloud to AlloyDB. All transactions which are to be denied will be done on by the (profiles_engines_availability) on AlloyDB before they are sent to the gds/suppliers. The last component of this design is cloud composer which will perform the orchestration of the whole process.

4.5 Average monthly cost of computation:

The computational cost is based on information provided by IT manager and the number of bookings was derived from the analysis made on bigquery. A monthly average of €200 for storage, €750 for queries, €5000 for computation and a “bidding cost” of ‘X’ because no response was given upon the request of this value from the company. Average booking is 200 a day making €50 from each.

The profit equation will be:

$$\text{Average profit} = \text{Total revenue} - \text{Total cost}$$

$$\text{total cost} = \text{GDS API} + \text{Google Cloud Platform (BigQ query and storage)} + \text{Bidding Cost}$$

$$\text{Average total revenue} = \text{average number of total bookings} \times \text{sale price}$$

$$\text{Average profit} = (200 \times 30 \times 50) - (5000 + 750 + 200) - x = (\text{€}294,050 - x) \text{ per month}$$

5. Conclusion

After studying and doing a lot of research on the problem that Easy Market is trying to solve I will say it is very promising if they keep the research and development going since the digital space is constantly evolving and with this come a problem of obsolescence. As you have seen in this paper a system that had worked very well in the past, became obsolete with scalability issues as people's travelling habit changes. The design and implementation of real-time analytics systems to handle big data is no easy task, it cost money, time and expertise. The IT team hopes to create this analytics system that will automate the computation processes and BLACKLISTING, I am not saying this is not possible soon but with such systems comes a great responsibility. In early stages of the data exploratory, I found out that the number of searches overweight the booking so when developing a system to automate task like computation of look-to-book and blocking/unblocking. One should note that there are specific route that has more traveller than other and therefore what is the threshold ratio to set to tell the system when to block, Easy Market was only blocking out of previous experience and there was no threshold ratio.

The other point has to do with human behaviour, mostly people do a lot of searches on various platforms to find the fastest/cheapest flight to a specific destination with no intention of buying anything. Therefore, modelling human behaviour is not an easy task and travelling taste of people change over time, a destination which will be a hot travelling spot in a certain period will not be appetizing in the future. Never mind assumptions are to be tested and if the IT department aims to pursue this research further, this paper has laid a foundation and question to asked, by showing what is possible like the ETL and metrics computation, what can be done in-house or outsourced.

My proposal in chapter 1 should serve as a foundation of the utilization of their cloud ecosystem but google offers range of products to perform the same task. On average Easy Market will be earning €294,050 a month just on flight bookings. The bookings per month is not fixed, on seasonal holidays like summer, Christmas and new year people tends to move around more increasing the number of searches and bookings. The future of Easy Market is great, and I believe with all the resources they have in hand they'll will reach their goal.

6. References

- Achary S. (2015) - Hadoop Essentials. Chapter 1: Introduction to Big Data and Hadoop, V's Vs of big data (p.3). Big Data Architectural strategy (p.4). MapReduce example (p.48). An introduction to Spark (p.152). Features of Spark (p.152)
- Apache Software Foundation (11/03/2024) - YARN Architecture
<https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YARN.html>
- TEZ (11/04/2024). Introduction, <https://tez.apache.org/>
- Amazon (aws) (11/04/2024) – Apache Tez, <https://shorturl.at/dCHL9>
- What is Apache Spark (17/04/2014) <https://aws.amazon.com/what-is/apache-spark/>
- Types of Cloud Computing (17/04/2024) <https://shorturl.at/wxzMU>
- What's the difference between ETL and ELT? <https://shorturl.at/bowX2>
- Balusamy B., Kadry N., Gandomi A. (2021). Big Data: Concepts, Technology, and Architecture. Evolution of Big Data (p.2). Differences in the attributes of big data and RDBMS (p.4). The 3Vs of Big Data (pp.5-6). Sources of Big data (p.7). Different Types of Data (pp.8-11). Batch processing (p.88). Real-time processing (p.89). Hadoop ecosystem (p.113). Introduction to Cloud (pp.93-103)
- Baranowski Z. (2019) - Evolution of Hadoop Platform and Ecosystem for High Energy Physics. https://cds.cern.ch/record/2700232/files/10.1051_epjconf_201921404058.pdf
- Bard AI (18/04/2024) – Google Colab. Look-to-book ratio. Record/STRUCT data type
- Bertino E. (2011) - Challenges and Opportunities with Big Data
<https://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=1000&context=cctech>
- Building stream processing pipelines with Dataflow, <https://rb.gy/gti0v4>
- Cadwalladr C. – The great British Brexit robbery: how our democracy was hijacked
<https://shorturl.at/efNO8>
- CDAP (2024) – GitHub, data fusion. <https://github.com/cdapio/cdap>
- Cloudera – Nokia: Using Big Data to Bridge the Virtual & Physical Worlds
<https://shorturl.at/IKS19>
- Cloud Composer in a minute, <https://www.youtube.com/watch?v=pSjD5nHkXLU>
- Cloud Pub/Sub, <https://www.youtube.com/watch?v=jLI-84UjZLE&t=6s>
- Columbia Southern University – Emerging Trends in Information Technology for 2023
<https://shorturl.at/imOQT>

Confessore N. - The New York Times Cambridge Analytica and Facebook: The scandal and the Fallout So Far. <https://shorturl.at/jkvC7>

Cox M., Ellsworth D. (1997) IEEE Xplore: Application-controlled demand paging for <https://ieeexplore.ieee.org/document/663888>

Crabtree M., Nehme A. – Datacamp, what is Data Analysis? An Expert Guide with Examples <https://www.datacamp.com/blog/what-is-data-analysis-expert-guide>

Databricks – MapReduce, History of MapReduce. <https://shorturl.at/nqvB4>

Data Compute Unit billing. <https://cloud.google.com/dataflow/pricing>

Dataflow in a minute, <https://www.youtube.com/watch?v=XdsuDOQ9nkU&t=3s>

Demchenko Y., Turkmen F., Laa d. C., Hsu C.H., Blanchet C., Loomis C. – ScienceDirect (2017) Chapter 2 – Cloud Computing Infrastructure for Data Intensive Applications. Big Data architecture components (p.32), Big Data Reference Architecture (p.32). <https://www.sciencedirect.com/science/article/pii/B9780128093931000027>

Downey C. S. – Princeton University ALUMNI, The Godmother of AI: How Fei-Fei Li'99 is safeguarding the future of human and artificial intelligence. <https://alumni.princeton.edu/stories/fei-fei-li-woodrow-wilson-award>

Ellsworth D. (2021) – NASA, Accelerating Demand Paging for Local and Remote Out-of-Core Visualization. <https://www.nas.nasa.gov/assets/nas/pdf/techreports/2001/nas-01-004.pdf>

Esmailian P. - LinkedIn, Big Data: The 8Vs of Big Data. <https://shorturl.at/jVZ78>

Flink Forward Moving from Lambda and Kappa Architectures to Kappa+ at Uber, <https://www.youtube.com/watch?v=ExU7fJFw4Bg>

Gallinucci E. (2024) – DTM, Big Data Course, Introduction: Big data definition, big data lifecycle (p.19).

Google Cloud (2024) – Google Cloud overview. <https://cloud.google.com/docs/overview>
Cloud Pub/Sub. <https://shorturl.at/ioyzC>

Google Cloud Platform. https://en.wikipedia.org/wiki/Google_Cloud_Platform
Google Cloud Tech, YouTube - Cloud Pub/Sub Overview, <https://rb.gy/ral7nn>

Harbert T. - MIT SLOAN School of Management, Tapping the power of unstructured data, <https://mitsloan.mit.edu/ideas-made-to-matter/tapping-power-unstructured-data>

IBM (2024) – What is Apache Spark. <https://www.ibm.com/topics/apache-spark>

IMAGENET (2024) – Homepage: <https://www.image-net.org/index.php>

Imperial College London (2018) - Big Data: Yesterday, Today and Tomorrow <https://www.imperial.ac.uk/events/98132/big-data-yesterday-today-and-tomorrow/>

Jeble S., Kumari S., Patil Y. (2018) – Role of Big Data in Decision Making (pp.5-6).
<https://shorturl.at/gprSZ>.

Kalipe K. G., Behera K. R. (2019) – *IJITEE*, Big Data Architectures: A Detailed and Application Oriented Analysis. <https://shorturl.at/fxPRT>

Kalavri V., Vlassov V. (2013) – KTH Royal Institute of Technology, MapReduce: Limitations, Optimizations and Open Issues (p.2).

Kreps J. (2014) – O'Reilly, Radar Insight, Analysis, and Research About Emerging Technologies, Questioning the Lambda Architecture. <https://shorturl.at/hjnoN>

Lee I. (2017) - ScienceDirect, Big data: Dimensions, evolution, impacts and challenges
<https://www.sciencedirect.com/science/article/abs/pii/S0007681317300046>

Lohr S. (2013) – New York Times, The Origins of ‘Big Data’: An Etymological Detective Story. <https://shorturl.at/hnrT7>

LUFTHANSA GROUP (2020) – Continuous Pricing Version 1. <https://shorturl.at/ahq08>

Mashey J. (2024) – LinkedIn: Official John Mashey profile picture
<https://www.linkedin.com/in/johnmashey/>

McLean J. (2024) – Data Never Sleeps Hits Double Digits!
<https://www.domo.com/blog/data-never-sleeps-hits-double-digits/>

Mell P., Grance T. (2011) – The NIST Definition of Cloud Computing
<https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf>

Microsoft (2024) – Big data architectures. <https://shorturl.at/otP57>

Migrating Hadoop and Spark Clusters to Google Cloud, <https://rb.gy/hsg2i4>

Monge Tomas (2015) – Amadeus, Amadeus Airline Cloud Availability solves the challenge of ever increasing flight search volumes. <https://shorturl.at/jlrL3>

NASA (2024) – Ames Research Center
<https://www.nasa.gov/ames/about-ames/>

NATIONAL AIR AND SPACE MUSEUM | Smithsonian (2024) - The Wright Brothers
<https://airandspace.si.edu/explore/stories/wright-brothers>

OXFORD CANCER (2024) – Research, Cancer Big Data
<https://www.cancer.ox.ac.uk/research/research-themes/cancer-big-data>

Osborne H. (2018) – The Guardian, what is Cambridge Analytica? The firm at the centre of
<https://shorturl.at/vBJ17>

Reichental J. Dr. (2020) – Medium, Cloud Computing Unlocks Innovation Opportunities For City CIOs. <https://rb.gy/15vfgn>

RedHat (2022) – Types of cloud computing, <https://shorturl.at/aikZ7>

Roshan Naik (2019) – YouTube, Uber Engineering: [Uber Seattle Introduction to Kappa+ Architecture using Apache Flink. <https://www.youtube.com/watch?v=4qSlsYogALo>
 Run Spark and Hadoop faster with DataProc, <https://rb.gy/g8aoh6>

SAS (2024) – Big Data, what it is and why it matters. <https://shorturl.at/iuESW>

Sharton B. R. (2021) – Ransomware Attacks Are Spiking. Is Your Company Prepared? <https://hbr.org/2021/05/ransomware-attacks-are-spiking-is-your-company-prepared>

Simplilearn - MapReduce Architecture, Hadoop Ecosystem Explained: Learn the Fundamental Tools and Frameworks. <https://www.simplilearn.com/tutorials/hadoop-tutorial/hadoop-ecosystem>

Snowflake – LAMBDA ARCHITECTURE, BENEFITS OF LAMBDA ARCHITECTURE’S DATA SERVING. <https://shorturl.at/xBDJU>

Stanford University (1998) – Big Data and the Next Wave of InfraStress <https://web.stanford.edu/class/ee380/9798win/lect08.html>

TAP Portugal (DEC 2014) – “Responsible Booking Policies”, Booking Procedures and Consequent Penalties

Tatineni M. Ph.D. (2024) – Coursera, Hadoop Platform and Application Framework, YARN, and Spark. <https://www.coursera.org/learn/hadoop>

Taylor P. - Statista, Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025 (2023). (in zettabytes). <https://shorturl.at/esyIN>, Size of the big data analytics market worldwide from 2021 to 2029 (in billion U.S. dollars). <https://www.statista.com/statistics/1336002/big-data-analytics-market-size/>

Tejada Z. (2024) – Microsoft, Big data architectures. <https://shorturl.at/dkIL4>

Tiao S. (2024) – Oracle, What Is Big Data? <https://www.oracle.com/big-data/what-is-big-data/>

Travelocity. <https://en.wikipedia.org/wiki/Travelocity>

Veith A., Assuncao M. (2018) – Apache Spark <https://www.researchgate.net/publication/323447097>, The Apache Spark Stack (p.2)

Wikipedia (2024) - Lambda architecture https://en.wikipedia.org/wiki/Lambda_architecture#cite_note-marz-blog-4

What is BigQuery, https://www.youtube.com/watch?v=d3MDxC_iuaw

World Wide Web Foundation (2024) - History of the web

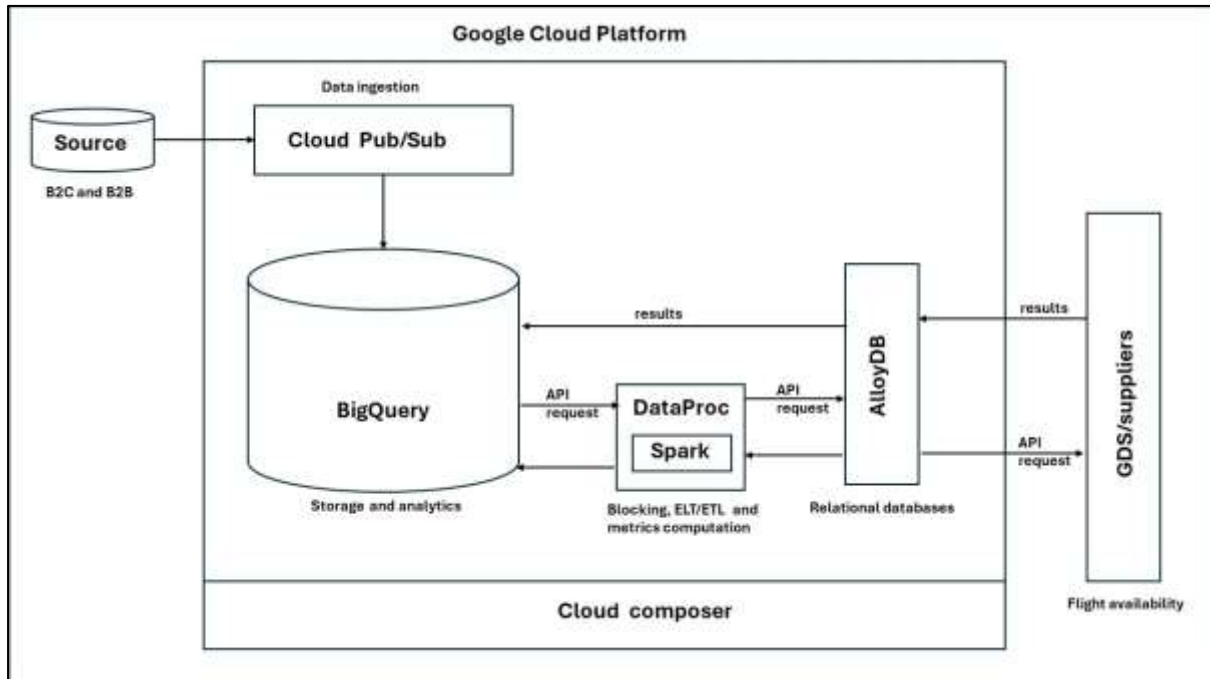
Yaseen K. H., Obaid M. A. (2020) - Joiv, INTERNATIONAL JOURNAL ON

INFORMATICS. Big Data: Definition, Architecture & Applications (p.3).

<https://shorturl.at/eJNQV>

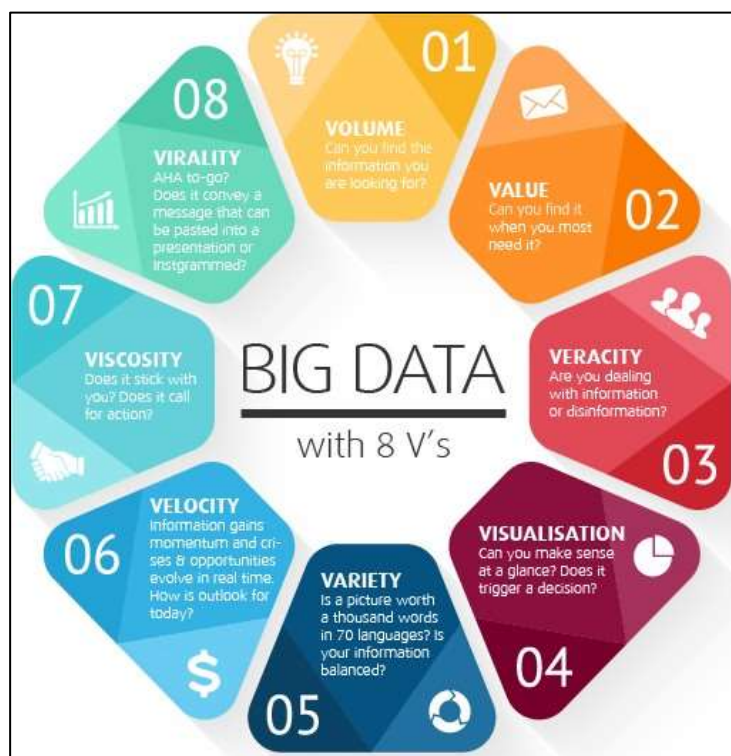
7. Appendix – Introduction, Proposal

A. Apache spark solution for real-time analytics and metrics computation



7. Appendix – Literature Review

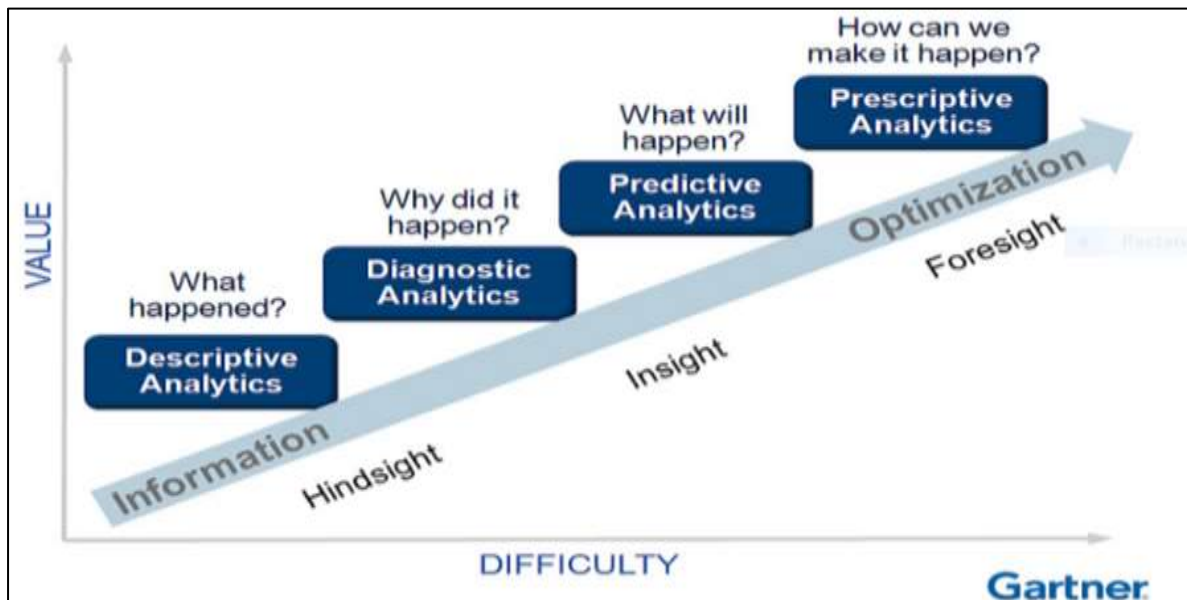
A. Big Data: The 8 Vs of Big Data, Enrico Gallinucci: Big Data lecture introduction p.8



B. Infographic data never sleeps 11.0, data generate every 1 minute of the day (DOMO, 2023)



C. Types of data analysis

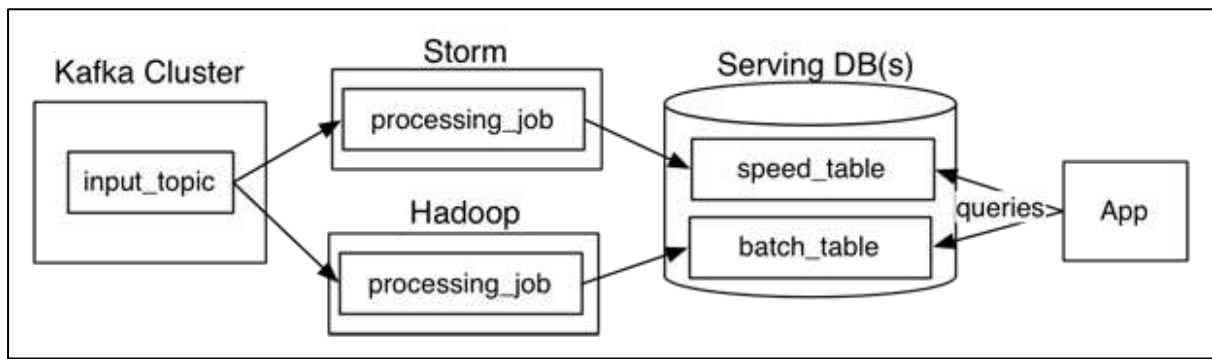


source: objective

D. Role of big data in making decisions by Shirish et al., pp.6-7.

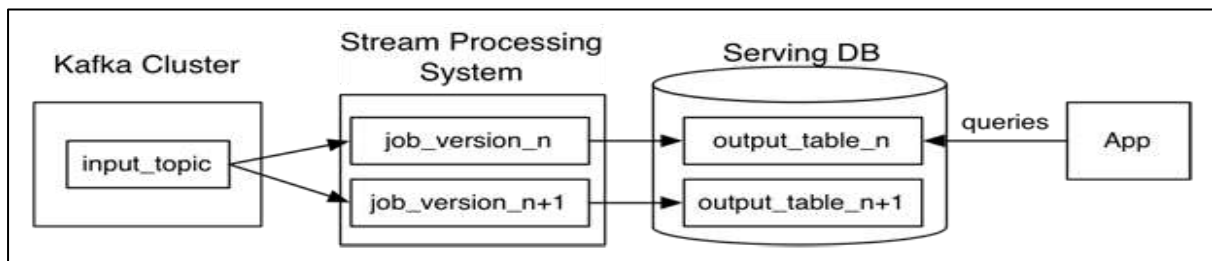
Big data source	Big data driven insights	Actionable decisions	Reference
Google search for a product or brand	<ul style="list-style-type: none"> <input type="checkbox"/> Customer intention to buy a particular product <input type="checkbox"/> Identify customer preference for a particular brand 	Predicting demand for product	
Google search by specific key words	What information citizens are looking for or concerned about	Predict spread of flu by geography by regions	Mayer-Schönberger & Cukier, 2013
Amazon search	Customer intention to buy a particular product	Reminder to customer next time she/he visits the site leading to chances of sale	Amazon.com website
Amazon Purchase history	Using association rules mined from billions of records, identify which different products are bought by customers	Product recommendation (customer who bought this also bought)	Amazon.com website
Walmart POS data	<ul style="list-style-type: none"> <input type="checkbox"/> Using association rules mined from billions of records, identify which products customers buy together (market basket analysis) <input type="checkbox"/> Facing disaster such as hurricanes people buy some unusual things like pop-tarts etc. 	<ul style="list-style-type: none"> <input type="checkbox"/> Store layouts redesign to place such products together <input type="checkbox"/> Inventory planning based on buying patterns prior to disasters such as hurricanes 	<ul style="list-style-type: none"> <input type="checkbox"/> Waller & Fawcett, 2013 <input type="checkbox"/> Dyché, 2014

E. Lambda architecture using Hadoop ecosystem by Jay Kreps



source: [Oreilly Radar](#)

Kappa architecture using Hadoop ecosystem by Jay Kreps.



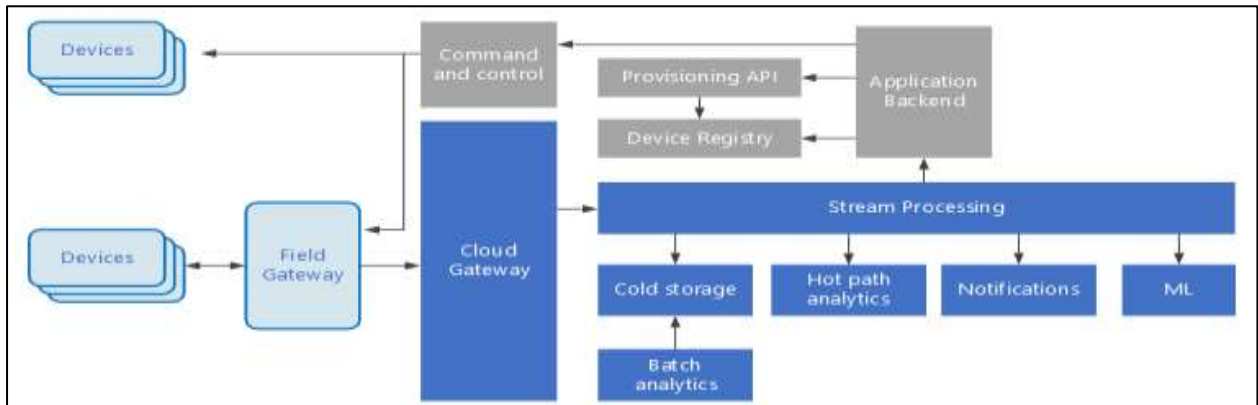
source: [Oreilly Radar](#)

Zeta Architecture: Hexagon is the new circle by Jim Scott



source: [Oreilly Radar](#)

Internet of Things (IoT) specialized subset big data solution by Microsoft



source: [Microsoft](#)

F. Batch processing vs stream processing: A tabular comparison

Criteria	Batch Processing	Stream Processing
Nature of Data	Processed in chunks or batches.	Processed continuously, one event at a time.
Latency	High latency: insights are obtained after the entire batch is processed.	Low latency: insights are available almost immediately or in near-real-time.
Processing Time	Scheduled (e.g., daily, weekly).	Continuous.
Infrastructure Needs	Significant resources might be required but can be provisioned less frequently.	Requires systems to be always on and resilient.
Throughput	High: can handle vast amounts of data at once.	Varies: optimized for real-time but might handle less data volume at a given time.
Complexity	Relatively simpler as it deals with finite data chunks.	More complex due to continuous data flow and potential order or consistency issues.
Ideal Use Cases	Data backups, ETL jobs, monthly reports.	Real-time analytics, fraud detection, live dashboards.
Error Handling	Detected after processing the batch; might need to re-process data.	Needs immediate error-handling mechanisms; might also involve later corrections.
Consistency & Completeness	Data is typically complete and consistent when processed.	Potential for out-of-order data or missing data points.
Tools & Technologies	Hadoop, Apache Hive, batch-oriented Apache Spark.	Apache Kafka, Apache Flink, Apache Storm.

source: [atlan](#)

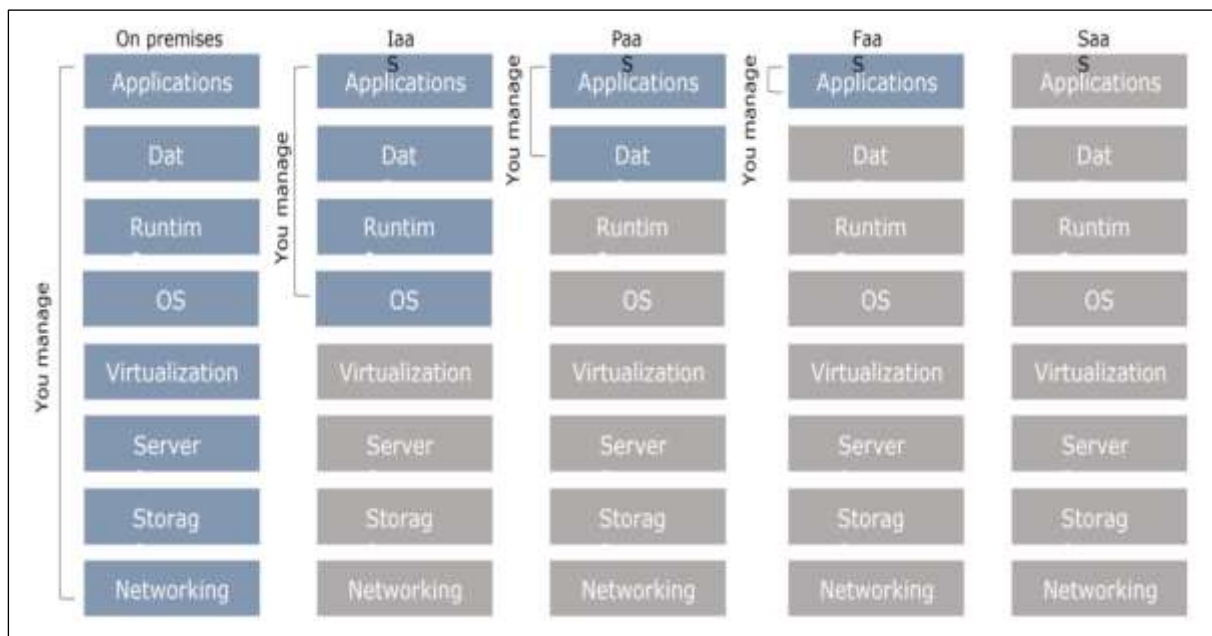
G. Spark Streaming Ingestion by Shiva Achari, “Hadoop Essential, 2015, (p.154)”



H. Pricing of the GCP products:

- (i). [Pub/Sub](#) (ii). [BigQuery](#) (iii). [Colab Research](#) (iv). [Vertex AI](#) (v). [Dataflow](#) (vi). [DataProc](#) (vii). [Data Fusion](#) (viii). [Cloud Composer](#)

I. XaaS (anything as a service) by Prof. Matteo Francia (2023)



source: DTM cloud computing lecture slides

7. Appendix - Automating Look-to-book computation

A. Key differences between: ETL vs ELT

Differences	ETL	ELT
Data compatibility	Suitable for structured data	Handles all types of data
Speed	Slows system as data size increases	Faster than ETL
Costs	costly	Cheaper as compared to ETL
Security	easy to implement	More difficult to implement

B. Python Transformation script

```
# create a year, month, day
```

```
def datedim(df):
```

```
    import pandas as pd
```

```
    # Sort the dataframe by datetime
```

```
    df = df.copy()
```

```
    df['created'] = pd.to_datetime(df['created'])
```

```
    df = df.sort_values(by='created', ascending=True)
```

```
    # date dimension function
```

```
    # Assuming df is your DataFrame and 'created' is the date column
```

```
    # extract year, month, day
```

```
    df["year"] = df['created'].map(lambda x: x.year)
```

```
    df["month"] = df['created'].map(lambda x: x.month)
```

```
    df["day"] = df['created'].map(lambda x: x.day)
```

```
    # columns of the dataframe
```

```
    cols = ['year', 'month', 'day', 'itemOperation', 'storefrontId', 'agencyCode', 'market',  
'serviceEngineId', 'status', 'searchTransactionId', 'totalNrResults', 'pricingTransactionId',  
'bookingTransactionId', 'reservationCode', 'pnr', 'validatingCarrier', 'deptAirport',  
'departureCountry', 'arrAirport',]
```

```
    # reindex to get year-month-day
```

```
    df = df.reindex(columns=cols)
```

```
    # return df
```

```

return df

# This function fills in fillMissingCountryDep values.
def fillMissingCountryDep(df, airports):
    """
    This function fills in missing values.
    """
    # city country to list
    city = airports.iloc[:, 0].tolist()
    country = airports.iloc[:, 1].tolist()
    # create key value pairs of city-country
    pair = dict(zip(city, country))
    # departure key value pair
    newDp = []
    newDc = []
    for dep in df['deptAirport']:
        if dep in pair:
            # newDp.append(dep)
            if pair[dep] == 'NA':
                newDc.append("NA")
            else:
                newDc.append(pair[dep])
        else:
            # newDp.append(dep)
            newDc.append('checkDeptairport')
    df["departureCountry"] = newDc
    return df

# This function fills in fillMissingCountryArr values.
def fillMissingCountryArr(df, airports):
    """

```

This function fills in missing values.

```
"""
# city country to list
city = airports.iloc[:, 0].tolist()
country = airports.iloc[:, 1].tolist()
# create key value pairs of city-country
pair = dict(zip(city, country))
# departure key value pair
newAp = []
newAc = []
for dep in df['arrAirport']:
    if dep in pair:
        # newDp.append(dep)
        if pair[dep] == 'NA':
            newAc.append("NA")
        else:
            newAc.append(pair[dep])
    else:
        # newDp.append(dep)
        newAc.append('checkArrairport')
df["arrivalCountry"] = newAc
return df

# The wrangle function takes in the dataframe and cleans up all missing values
# replace meaningful names
def wrangle(df, airports):
    """
    Cleans the dirty dataset
    """
    import pandas as pd
    # load airport-country pair dataset
    # replacing the null values
    df['status'] = df['status'].fillna('OK')
```

```

df['pnr'] = df['pnr'].fillna('noPnr')
df['reservationCode'] = df['reservationCode'].fillna('noRes')
df['validatingCarrier'] = df['validatingCarrier'].fillna('noValCarr')
df['totalNrResults'] = df['totalNrResults'].fillna(-1)
# market REvolution for storefrontId == 2
df.loc[df['storefrontId'] == 2, 'market'] = 'REvolution'
# if market is missing fill as market
df['market'] = df['market'].fillna('market')
# searchTransactionId, pricingTransactionId, bookingTransactionId
df['searchTransactionId'] = df['searchTransactionId'].fillna('PoB')
df['pricingTransactionId'] = df['pricingTransactionId'].fillna('SoB')
df['bookingTransactionId'] = df['bookingTransactionId'].fillna('SoP')
# city country to list
city = airports.iloc[:, 0].tolist()
country = airports.iloc[:, 1].tolist()
# create key value pairs of city-country
pair = dict(zip(city, country))
# get index of missing dept and arr countries
import numpy as np
nullDC = np.array(df[df['departureCountry'].isnull()].index.to_list())
nullAC = np.array(df[df['arrivalCountry'].isnull()].index.to_list())
# replace missing departure country
try:
    for i in nullDC:
        df.loc[i, 'departureCountry'] = pair[df.loc[i, 'deptAirport']]
except KeyError:
    df.loc[i, 'departureCountry'] = 'checkdeptAirport'
# replace missing arrival country
try:
    for e in nullAC:
        df.loc[e, 'arrivalCountry'] = pair[df.loc[e, 'arrAirport']]
except KeyError:
    df.loc[e, 'arrivalCountry'] = 'checkarrAirport'

```

```

# check for replacing values
# get index of missing dept and arr countries
nullC = np.array(df[df['departureCountry'] ==
                    'checkdeptAirport'].index.to_list())
nullA = np.array(
    df[df['arrivalCountry'] == 'checkarrAirport'].index.to_list())
# replace missing departure country
try:
    for i in nullC:
        df.loc[i, 'departureCountry'] = pair[df.loc[i, 'deptAirport']]
except KeyError:
    df.loc[i, 'departureCountry'] = 'checkdeptAirport'
# replace missing arrival country
try:
    for e in nullA:
        df.loc[e, 'arrivalCountry'] = pair[df.loc[e, 'arrAirport']]
except KeyError:
    df.loc[e, 'arrivalCountry'] = 'checkarrAirport'

# check namibia
# fix namibia mising values again
namibia = {'WDH': 'NAM', 'WVB': 'NAM', 'ERS': 'NAM', 'LUD': 'NAM'}
for k in namibia:
    df.loc[df['arrAirport'] == k, 'arrivalCountry'] = '\NA\'
    df.loc[df['deptAirport'] == k, 'departureCountry'] = '\NA\'

# explicit datatype conversion
cols = df.select_dtypes(include='object').columns.to_list()
# numeric coln
coln = df._get_numeric_data().columns.to_list()
df[cols] = df[cols].astype(str)
df[coln] = df[coln].astype('int64')
# missing values not previously captured
df['arrivalCountry'] = df['arrivalCountry'].replace(

```

```
    ['None', 'checkarrAirport')
df['departureCountry'] = df['departureCountry'].replace(
    ['None', 'checkdeptAirport')
# return final dataset
return df
```


C. Cloud deployment automation code

```
# Wrangle: Import wrangle module and other extension dependencies

# key value pair helper file
airports = pd.read_csv('iata_airport.csv')
# create year, month, day
df = datedim(df)
# fill-in dept country
df = fillMissingCountryDep(df, airports)
# fill-in arrival country
df = fillMissingCountryArr(df, airports)
# wrangle other fields
df = wrangle(df, airports)
# manually reindex date columns:
cols = ['year', 'month', 'day', 'itemOperation', 'storefrontId', 'agencyCode', 'market',
        'serviceEngineId', 'status', 'searchTransactionId', 'totalNrResults', 'pricingTransactionId',
        'bookingTransactionId', 'reservationCode', 'pnr', 'validatingCarrier', 'deptAirport',
        'departureCountry', 'arrAirport', 'arrivalCountry']
df = df.reindex(columns=cols)

# ----- Writing to staging.ETL_warehouse -----
table_id = 'staging.ETL_warehouse'
location = 'EU'
project = "dark-sensor-386210"
```

```

# Insert the DataFrame into the table
try:
    # write to big query "append data on existing ETL_warehouse table"
    df.to_gbq(table_id, project_id=project, if_exists='append')
    print(f'Successfully written to {table_id} !!!')
except Exception as e:
    print(e)
# ----- write to the metrics table ----- :
# import metrics_module
df_met = metrics(df)

# write to table
table_id = 'staging.metrics'
# Insert the DataFrame into the table
try:
    # write to big query "append data on existing ETL_warehouse table"
    df_met.to_gbq(table_id, project_id=project, if_exists='append')
    print(f'Successfully written to {table_id} !!!')
except Exception as e:
    print(e)
print("File successfully executed!!!")

```

D. Colab subscription plan by Google

Pay As You Go	Recommended Colab Pro	Colab Pro+	Colab Enterprise
<p>1,28 € for 100 compute units</p> <p>1,34 € for 600 compute units</p> <p>You currently have 0 compute units. Compute units expire after 90 days. Purchase more as you need them.</p> <ul style="list-style-type: none"> ✓ No subscription required. Only pay for what you use. ✓ Faster GPUs. Upgrade to more powerful GPUs. 	<p>1,28 € per month</p> <ul style="list-style-type: none"> ✓ 100 compute units per month. Compute units expire after 90 days. Purchase more as you need them. ✓ Faster GPUs. Upgrade to more powerful GPUs. ✓ More memory. Access our highest memory machines. ✓ Terminal. Ability to use a terminal with the connected VM. <p>Selected countries and 18+ only:</p> <ul style="list-style-type: none"> ✓ AI-enabled autocompletion. Intelligent multi-line suggestions automatically rendered while you type. ✓ Code generation. Generate code with natural language, including an integrated chatbot. 	<p>1,54 € per month</p> <p>All of the benefits of Pro, plus:</p> <ul style="list-style-type: none"> ✓ An additional 400 compute units for a total of 500 per month. Compute units expire after 90 days. Purchase more as you need them. ✓ Faster GPUs. Priority access to upgrade to more powerful premium GPUs. ✓ Background execution. With compute units, your actively running notebook will continue running for up to 24 hours, even if you close your browser. 	<p>Pay for what you use.</p> <ul style="list-style-type: none"> ✓ Integrated. Tightly integrated with Google Cloud services like BigQuery and Vertex AI. ✓ Enterprise notebook storage. Replace your usage of Google Drive notebooks with GCP notebooks, stored and shared within your cloud console. ✓ Productive. Generative AI powered code completion and generation.