Duong Hai Ly

Data analytics in cloud data warehousing: case company

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's thesis

28 February 2019

Helsinki Metropolia University of Applied Sciences

PREFACE

Being a part-time student at Metropolia University of Applied Sciences for 3 years was quite challenging for me to pursue both studying and full-time working simultaneously, especially in the engineering field, which usually requires students to dedicate more time to learning and practice. I did not have much time to finish the degree courses nor the project like a full-time student. However, by putting a little effort every day after work, piece by piece, I finished the final year project, and I am satisfied with the results. I consider this project a great opportunity not only to learn about cloud data warehousing in theory but also to learn how to create and run a cloud data warehouse in practice to solve business problems.

In truth, I could not have finished the project without a huge support from many people. First of all, I would like to say kindest words to teacher Antti Piironen and my supervisor teacher Peter Hjort, who have given me a great deal of support in the project. In the company, my thanks go to our CEO and other product team members. And especially, I really appreciate support from our COO and Product Director who gave me a chance to learn about cloud data warehousing and data analysis. Last but not least, I owe a thank you message for my wife, who helped me believe in the project and always encouraged me to finish the project on time.

Espoo, 28 February 2019

Ly Duong Hai

Author Title Number of Pages Date	Duong Hai Ly Data analytics in cloud data warehousing: case company 51 pages + 1 appendices 28 February 2019
Degree	Bachelor of Engineering
Degree Program	Information Technology
Professional Major	Software Engineering
Instructors	Antti Piironen, Principal Lecturer Peter Hjort, Principal Lecturer

The objective of the final year project was to discover a solution for gathering and analyzing digital marketing service data from different sources such as Google ads, Facebook ads and internal database of case company. By viewing well-presented analyzed data, business owners, sales and campaign managers could easily achieve better and accurate planning and forecasting. As a result, this would improve case company business strategies.

To reach the mentioned objective above, the cloud data warehousing solution was required to tackle two problems. Firstly, required data are scattered among the internet which often makes the analysis processes more complex and time consuming. Secondly, data have different models and analyzing such a large quantities of data demands a high-performance data processing platform.

The project has successfully created a cloud data warehouse in Snowflake and scheduled different analytics processes on the given data imported from various sources. In addition, one of the most important outcomes of the project is that it provides various report files that contain analytics data.

Key words	cloud data warehouse, Snowflake, data analytics

Table of Contents

List of Figures

List of Tables

Appendices

Appendix 1. Snowflake CLI configuration setting file, configured mostly by project manager, modified by the author.

List of Abbreviations

API	Application programming interface
AWS	Amazon Web Services
BI	Business Intelligence
CI	Continuous integration
CMS	Content management system
CSV	Comma separated values
CRM	Customer Relationship Management
Cron job	Time-based job scheduler in Unix-like computer operating systems
DB	Data base
DDL	Data Definition Language
DML	Data Manipulation Language
DOM	Document object model
DW	Data warehouse
EC2	Elastic compute Cloud from Amazon Web Services
EDI	Electronic data Interchange
EDW	Enterprise data warehouse
ETL	Extract, Transform and Load
ERP	Enterprise resource planning

GL	General ledger
HTML	Hypertext Markup Language
I/O	Input/ Output
JS	JavaScript
KPI	Key performance index
MERN	Combination of MongoDB, Express.js, React.js and Node.js
MPP	Massive Parallel Processing
MVC	Model View Control
OLAP	Online analytical processing
PSA	Persistent staging area
PostgreSQL	Powerful, open source object-relational database system
SnowSQL	Snowflake command line interface
SQL	Structured Query Language
SSH	Secure Shell
TCP/IP	Transmission Control Protocol/ Internet Protocol
tty	Teletypewriter
UI	User interface
UX	User experience
WWW	World wide web

1 Introduction

Database theory has become widespread, but in the beginning, there was neither an enterprise prevalent vision nor a more global vision to solve a variety of requests of the organization. In 1980s, subsequently a more sophisticated notion of database emerged, which was defined as Data warehouse by IBM researchers Barry Devlin and Paul Murphy. Data warehouse definition provides a broader picture of operational system in supporting dealing with a wide range of requests and queries. (Rouse, 2018). For business executives, data warehouse brings significant competitive benefits for their enterprises. On the other hand, data warehouse has helped managers and many other end users to overcome traditional roadblocks with more specific business information.

Nowadays, technology has grown with data warehouse correspondingly. With the rapid increase in data use together with the transition to big data era, data warehouse architecture has significantly taken a huge shift from traditional on-site warehouses towards cloud-based data warehouses. A crucial factor affecting the evolution of modern data warehousing is the cloud service. A large portion of companies use the cloud-service data warehouse as an effective way to a have a low-cost storage, easy scale up/scale down capability, flexibility, loss prevention, sustainability and more.

The objective of this thesis was to present a fresh perspective view of modern data warehouse and an idea of utilizing the advantages of cloud-base data warehouse so as to solve case company internet advertisement problem. To be more specific, the case company was looking for a solution to track the delivery process and goals of services that the company provides. The solution would improve business decision making and subsequently return more successful business outcomes. (Because of business secret, the author cannot disclose the company’s real name.)

2 Traditional data warehouse

This section defines in detail the concept of data warehouse and describe the characteristics of a data warehouse. Moreover, it also provides important information on data source layer, staging area, and presentation layer.

2.1 Data warehouse definition

A data warehouse is a large analytical database which gathers and controls heterogeneous data from a variety of operational systems in a single, consistent database. Specifically, it is structured for easy querying, reporting and analyzing. Data warehouse brings back the new concept, according to Ponniah (2002), who believed in making use of the huge volumes of existing data and in transforming it into forms suitable for retrieval, fast and accurate information. As a result, this means excluding the fresh data generation characteristic from data warehouse. Besides relational databases, a data warehouse system can contain data mining capabilities, Extract-Transform-Load (ETL), online analytical processing (OLAP), various customer analysis tools and applications that is responsible for gathering and delivering a vast amount of data. (Lane, 2005)

In short, data warehousing is known as a collection of techniques, tools and methods used to implement data analysis. Finally, this will be used for supporting decision-making processes as well as improving information resources. (Golfarelli and Rizzi, 2009)

2.2 Characteristics of a data warehouse

According to Inmon (1994) who invented the data warehouse term in 1990, data warehouse was defined as a collection of data with following features: subject-oriented, integrated and consistent, nonvolatile, time-variant.

The data Warehouse is a very large database (VLDB) and subject-oriented; therefore, it is subject to enterprise-specific conceptions, for instance, clients, sales, products and orders. It does not concentrate on data requirements of the enterprise department as traditional operating system does but goes beyond

traditional views by focusing on enterprise-wide subjects in order to illustrate an overall picture. (Humphries, et al., 1999)

The data warehouse is read-only and geared towards stability, integration and consistency. The data warehouse retrieves information from a variety of sources and cleans it, finally puts it into its structure. The data warehouse is very large; hence, it can be easily exploited by every user-specific department. The data warehouse itself must be specialized, divided into logical topics which is named as data marts. As a consequence, data warehouse provides an integrated view of numerous data and a full scope of its relationship in data system. Generally speaking, building data warehouse system does not require that new data constantly be added, preferably, exiting data should be in rearrangement.

Moreover, the data warehouse should have non-volatility. Once transaction is completed, information cannot be changed or fixed. In other words, historical data in a data warehouse should never be amended. This can cause some difficulties; however, it is reasonable with the purpose of the data warehouse to enhance decision makers' ability to analyze what has occurred. (Bidgoli, 2005)

Finally, the data warehouse is also time variant as it shows the evolution over time and not just the most recent data as the operational system tends to do. Data warehousing focuses on time-varying changes, with historical data that can be traced to business trends.

2.3 Data warehouse architecture

The data warehouse is a mix of technologies identifying the component parts, the relationship among them, their characteristic and more. The data warehouse can generally be categorized as traditional, hybrid and modern. However, nowadays generally a data warehouses adopts core components such as source system, staging area, data warehouse and data access tools.

The figure below apparently illustrates the architecture for building the data warehouse. In order to build the informational database, four crucial steps are prerequisite to be accomplished.

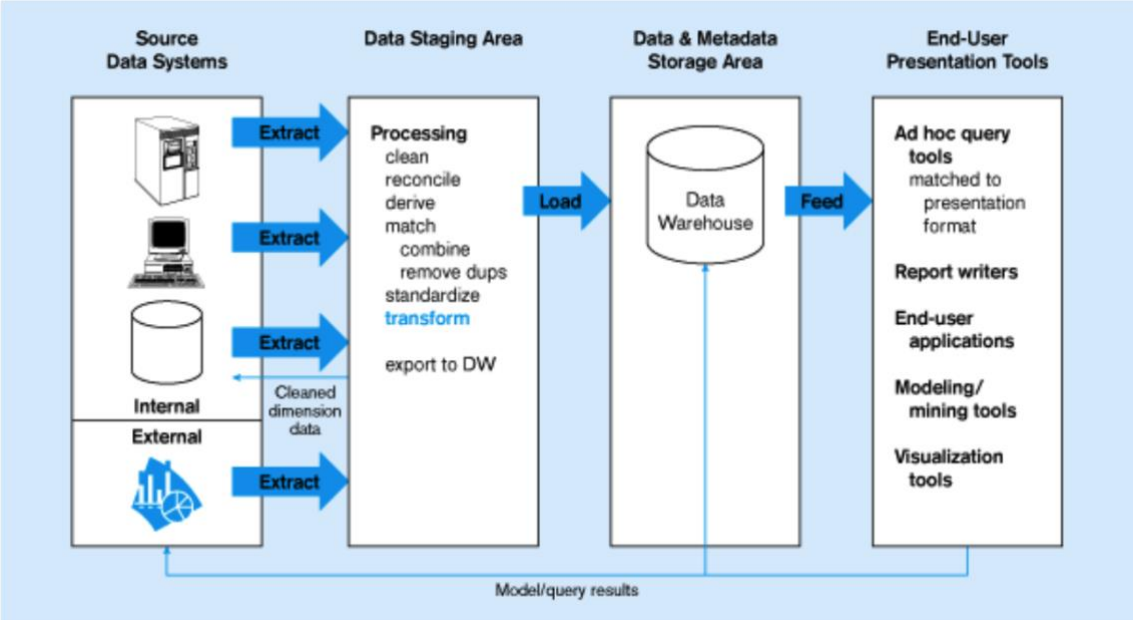

Figure 1. Architecture for building the data warehouse (Bassil, 2012)

As illustrated in Figure 1, different technologies are consequently essential to support these functions. And the data warehouse acts as repositories for data. However, the final result is the formation of a new computing environment for the purpose of providing the strategic information which are significant for every enterprise. A well-defined, properly functioning data warehouse can become a valuable competitive tool in business. Followings are crucial layers building: data warehouse architecture.

2.3.1 Data sources layer

The data sources layer contains all the defined data which is essential for extracting, transforming and loading information into data warehouse. The data can either be stored inside the system (internal data sources) or outside (external data sources). Moreover, information can also exist in various formats such as database, EDI, CMS, Big data.

2.3.2 Data acquisition and integration layer (staging area)

The data acquisition and integration layer (staging area) is the middle area in data warehouse system which is in charge of acquiring data from a variety of internal and external data sources. This part insists of the landing and staging area and data Integration (ETL).

The list below briefly describes each component:

- Landing and staging Area: The staging area stores data temporarily in a homogeneous database with the purpose of profiling and analyzing it before migrating it through to the next EDW layer. Occasionally, a big data ecosystem is also used together if unstructured data source is compulsory for landing. The staging area is essential due to the reason that most of the time, the data sources are not stored in the same server as the data warehouse. And it is always preferred to preserve data integrity. (Coté, et al., 2018)
- Data integration: This process often acknowledged as Extract, Transform and Load tools (ETL) and is in charge of drawing data from the system resources, transforming them to match the data warehouse schema and finally loading them in the target place. (Vassiliadis and Simitsis, 2009)

The following paragraphs explain three main processes in the data integration mentioned above.

The process of retrieving data into the data warehouse environment begins with extraction, which means reading and understanding information and copying the information needed into the staging area for further manipulation.

The next process, transformation, handles the conversion of the extracted data in order that another database can easily import it (Gour Vishal, Anand Sharma, 2010). Transformation is achievable by using lookup tables or rules or by making data combination. Numerous potential transformations are carried out for

instance cleaning the data, associate data from numerous places, copy data, and finally appoint data warehouse keys.

Data are written to the target database in the loading process. The three above-mentioned process steps are all forerunners for the data warehouse area. ETL plays an important role in the data warehouse process as it transforms dissimilar data sources into a uniform structure. Afterwards, people derive meaningful insights and reports from this data. The following diagram provides different phases of the ETL process.

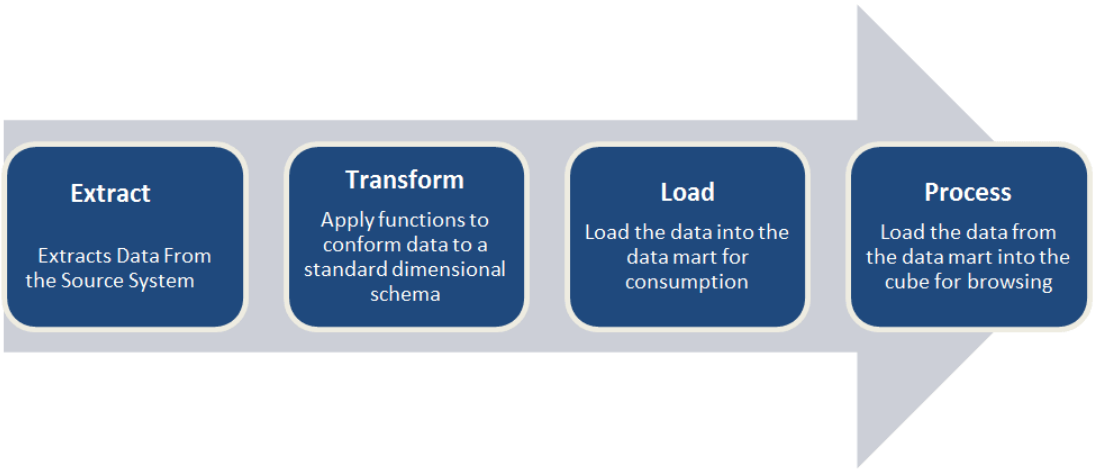

Figure 2. ETL process (Guru99, n.d)

In a nutshell, ETL is generally a complicated consolidation of procedure and technology that requires a considerable data warehouse development efforts and diversified skills of business staff. (Themistocleous and Morabito, 2017)

2.3.3 Data warehouse, enterprise data warehouse (EDW) and data mart

A data warehouse is constructed by combining data from numerous different sources with the purpose of analyzing data for business ideas. This step is known as the most crucial part for any data warehouse operation. Data warehouse is seen as a repository to store extracted information which is necessary for business solutions.

An enterprise data warehouse often refers to a data warehouse for a company, typically it is a place for all company data or at least majority of that. An EDW

makes data more solid from multiple sources and also accessible for different responsible departments in a company. In other words, data is normalized and standardized in order to meet various reporting purposes in business operation. (Adams, 2018). A traditional platform for an EDW or plain data warehouse has been on premise servers. Besides, operational database such as general ledger (GL) and other intra company services are traditionally hosted on-premise.

A data mart is a miniature, subdivision of the data warehouse system that concentrates on a particular business unit such as marketing, sales, logistics or a decision support requisition is intended to meet an immediate requirement. To build a working data mart model, firstly, it is crucially important to build a set of clean consistent dimension tables. Secondly, data mart designers need to ensure that fact tables created from joining up dimension tables do not have dimensional data, adequately, just the facts (Standen, 2008). Dimension tables mentioned here mean tables that store the objects involved in a business intelligent effort whereas fact tables store the data corresponding to a specific business process. Each row in fact tables contains the measurement data associated with a single event occurred within the business process. (Chapple, 2018). A data mart may or may not be dependent on these other data marts in an organization and is also targeted to meet an immediate requirement. If data marts are using the same dimensions and facts, they will be linked together (SIlvers, 2008).

2.3.4 Data presentation layer

The data presentation layer generates required data to end users in many formats depending on their demands. For instance, this layer may provide product or service insight data querying possibility and even support developing automated or ad-hoc reports. (Wainstein, 2018). Usually Business Intelligence and analytics tools are used in this layer.

Business intelligence (BI) is the ecosystem of skills, technologies, analytics and human expertise to enables an organization to get insight into its critical operations through reporting and analysis tools. BI applications may consist of a

wide range of components dashboard, scorecard, drill down, slicing and dicing data, spreadsheets, tabular reports, and charts.

Analytics is the practice of supporting decision-making through number-crunching which takes BI to the higher level. It assists process for instance customer segmentation, department spending. Furthermore, in order to support effective analytics, some tools and techniques are often used such as data mining, regression modelling or statistical analysis (Haertzen, 2012).

Back to 2005, tools such as Google Analytics (GA), were some of the first to offer mainstream access to analytics which plays an important role in obtaining rich insights about website traffic for enterprise. The reports and dashboards within GA provide comprehensive information to various questions that some website owners did not even consider possibility to make a beneficial contribution to their enterprise. Figure 3 below illustrates multiple benefits of BI and analytics to enterprise.

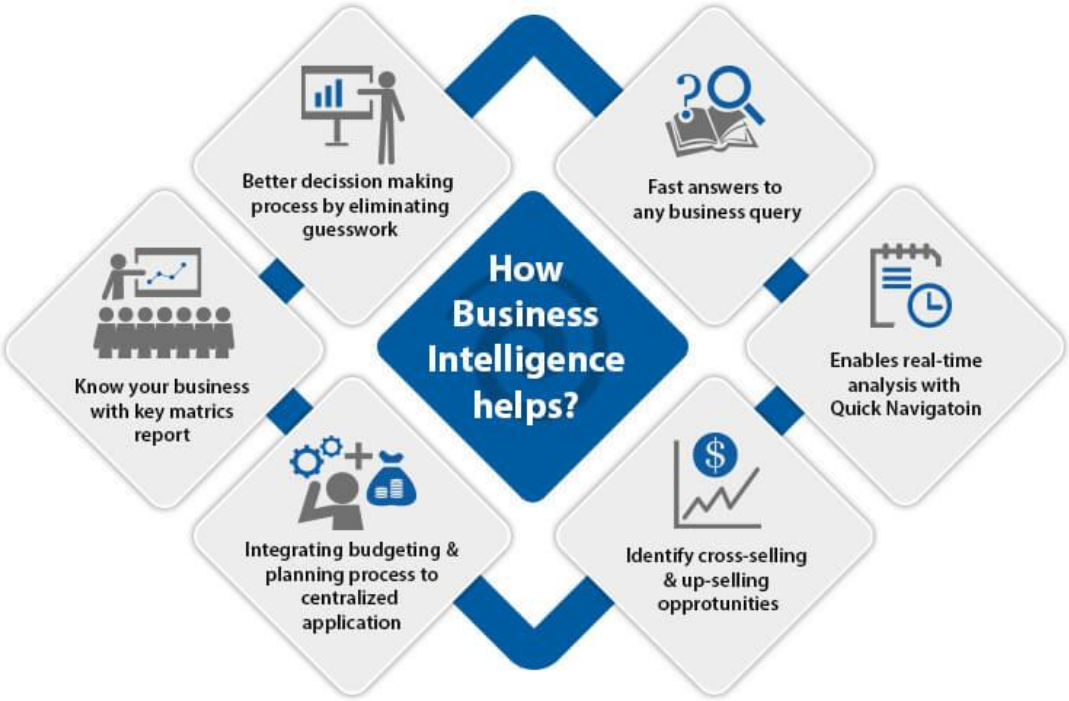

Figure 3. Benefits of business intelligence and analytics (Hoppe, 2016)

As illustrated in Figure 3, business intelligence and analytics turn data into insights and actions. Indeed, instead of relying on a great deal of guesswork, business managers can utilize insights to accelerate the decision-making process. In addition, analytics data plays an important role in creating faster reports, analysis or plans. Reports generated from BI tools contain key metrics and can be used to answer any business query. Thanks to real-time analysis with quick navigation provided in some BI tools, strategic decision making is no longer blocked, and business managers react to market changes promptly and more accurately.

2.4 Operational model and data warehouse model

An operational database model is generally known to stock and administer data in real time for an enterprise. They have a critical impact to data warehouse system as well as business operations due to the fact that they contribute as the essential source for a data warehouse. Those databases are usually in form of SQL or NoSQL-based. The crucial characteristic of operational model is their orientation toward real-time transaction. In this database, records can be added, deleted and adjusted in real-time. The comparison between operational database and data warehouse is shown in Table 1.

Table 1. Operational database and data warehouse (Ricardo and Urban, 2017)

Operational databases	**Data warehouses**
For data retrieval, updating and management	For data analysis and decision making.
Focus on data in	Focus on data out
Stored with a functional or process orientation	Stored with a subject orientation
Represent current transaction	Read historical data
Generally, update regularly	Non-volatile
Complex data structure (relational database)	Multi-dimensional data structure or relational format
Use OLTP (online transaction processing system)	Analytical software such as data mining tools, reporting tools and OLAP (online analytical processing)

Support a limited area within the organization	Provide view of entire organization
Sources from operational domain	Combine from multiple sources (include operational system)

As listed in Table 1 above, operational databases were designed for data retrieval, updating and management while data warehouses are used in data analysis and supports decision making. Operational databases focus on data in and they use relational, object-oriented to store data. On the other hand, data warehouses focus on data out, often dimensional data model and afterward, BI tools turn these data into insights and actions. Moreover, operation databases provide support for a great deal of transactions, theoretically, they are developed to serve in real-time the information needs of end users. Hence, these databases support online transaction processing (OLTP). In contrast, data warehouses support online analytical processing (OLAP). Data in the data warehouse can be imported from multiple various operational databases and they often known as historical data, standing from the perspective of a company, data may come from internal or external databases. (Ricardo and Urban, 2017)To preserve historical characteristic of data in data warehouses, after data are stored, they are fixed over time (Kauffman, 2008).

3 Cloud data warehouse

One of the most remarkable shifts in data warehousing recently has been the emergence of the cloud data warehouse nowadays. This section will introduce more details about cloud data warehousing including its evolution, architecture, benefits and its comparison with traditional on-premise data warehouse.

3.1 Data warehouse evolution

Accompanied with the prompt development of the digital age, more data often opens more opportunities associated with equivalent huge challenges for enterprises. However, the data warehouse system in the past strained under the pressure of great collection of relevant data. Companies appeared to be more

tolerant of longer analytics cycles. They constantly needed to wait at least 24 hours or even more during the process of data warehousing before it was available for analyzing. This fact occasionally resulted in hanging or crashing system and subsequently long downtime or delays. In the 2000s, the majority of the data sources lied in some on-premise system such ERPs, CRMs and GLs. Starting from the early 2000s and digitalization of services and the era of SaaS services, companies started outsourcing their business-critical data to the cloud.

Nowadays, companies even have more extremely large, diverse data sets than ever before including several data sources as well as cloud-based applications. In order to be able to handle and analyze a huge range of data efficiently, companies essentially seek for an effective way that can stock data in variant forms and offer advantageous approach to it. Especially, in digital business, companies' business data sources are mostly scattered among the internet. This sets new requirements for data warehousing, as it is impossible or at least challenging to import cloud data sources to an on-premise data warehousing system.

Fortunately, times has changed, and technology has grown correspondingly rapidly together with the society. Advances in technology are here to help companies overcome those challenges and sometimes even exceed their expectations. (Kurunji, et al., 2014). Cloud data warehousing services, such as Snowflake and Amazon redshift have stepped in, to solve traditional data warehouse problems. The following Figure 4 represents the evolution of data platforms from the early age with only relational database until nowadays.

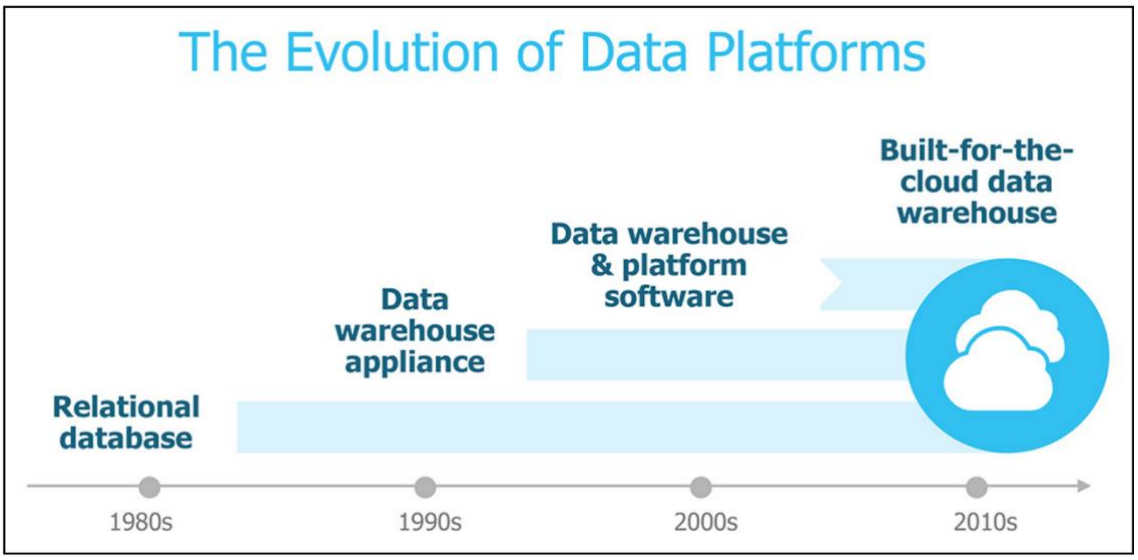

Figure 4. The evolution of data platforms (Kraynak, 2017)

As Figure 4 depicts, data warehousing has existed for a long time; however, it has not been able to manage the quantity and complication of today's data as well as the increased demand for data access and analytics. Eventually cloud data warehousing has emerged from the convergence of those major trends. Cloud data warehousing is recognized as an efficient solution for enterprises to make use of the latest technology without enormous investment and many other innovations.

3.2 On premises vs cloud data warehouse

As Figure 4 above depicts, in the past, data were relatively small and predictable. They can be stored and controlled in an enterprise's data system called on-premise. Advanced technology has helped enterprise in handling the volume, variety and flexibility of today's data. The cloud service is seen as the evolution of modern data warehouse and gradually, it has been a rapid adoption in our business world adapting the exponential increase of data requirement. Table 2 below presents some key considerations which can be used to compare a traditional on-premise data warehouse with a cloud data warehouse.

Table 2. Comparison between on-premise and cloud data warehouse

on-premise data warehouse	**cloud data warehouse**

Need at least one year to deploy a conventional data warehouse	3-6 months can be up and running
Expensive storage, computing and all-time cost	Much cheaper, can be one-tenth of a similar on-premise system
Not flexible sizing, balancing and tuning	Elastic cloud data, flexible, scalable
May occur delay and down time	No delay or downtime
High security, protective, discovery cost	Much lower cost

As can be seen in Table 2, the first feature to compare between on-premise and cloud data warehouse is about the implementation time. In order to deploy a traditional data warehouse, the company needs at least 1 year before extracting insights which can help executives in their decision making and business support. Such a long time can even expose the project to enterprise downturn in our exponential digital age. Whereas a cloud data warehouse can be planned and implemented for six months or even less so as to become a crucial part in supporting company's development.

Secondly, on-premise data warehouses are expensive in terms of additional or IT costs such as software, hardware and administration costs. These costs include servers, additional storage devices, a high-speed network charges, software licensing fees, and salaries for information technology personnel. Not only that, traditional on-premise data warehouses need to pay a huge cost for security and potential security breaches. A poorly implemented security and protection system can easily lead to lost business. On-premise data warehouse regularly needs to pay careful attention to various details such as security protocols, data encryption, firewall protection, monitoring and adapting to emerging security threat. Also, on-premise data warehouse regularly needs to backup data constantly in case of preventing data loss due to equipment failure, power outages, theft or disaster. Those processes also require a great deal of time, huge cost and strong human resources. (Kraynak, 2017)

Whereas, the cloud data warehousing contributes an excellent solution for data security, protection and recovery. Its services are targeted for a huge number of

customers. Naturally, it stores data off premises. However, the cloud data warehousing services will also offer built-in disaster rehabilitation in case data centers are isolated. As a result, cloud data warehousing supplier can offer high security at a much lower cost than on-premise data warehouse. (Nath, et al., 2017)

On the other hand, the third key consideration would be about elasticity, flexibility, resilience and concurrency. For best achievement, enterprises compulsorily tune on-premise data warehouse during peak usage, which may represent only a small period of the year. In order to implement it, companies often need a huge investment together with administration expenses.

Meanwhile, the cloud data warehousing brings key advantages, for example, it provides virtually unlimited storage/compute and scalability in users and workload. They can scale up or down storage and compute easily in order to adapt changing needs. Besides, it can add users and workload easily without affecting performance, additionally distributing virtual data across separate compute clusters which means concurrency. (Zhu, et al., 2014)

Moreover, enterprises using on-premises solutions frequently have two main problems. They often need to wait for a long time before data is available for analyzing. At the same time, for complex query it is also in the same situation. Occasionally, this can lead to hanging or crashing system and finally downtime or delays due to various and concurrent processes. Notwithstanding, cloud data warehouse can deliberate over downtime and delays due to the fact that it keeps virtually unlimited storage, elastic architecture with easily scale up or down and an efficient pipeline to make queries running more effectively.

In a nutshell, cloud-based data warehouses are a large step forward from traditional architectures. Building a properly configured data warehouse is a crucial step in order to maintain an outstanding operation. A well-designed data warehouse can help business executives conducting lightning fast queries on data that’s being processed at near real-time speeds.

3.3 Cloud data warehouse architecture

Technology operations provide customers' access to their data warehouse deployment through their subscription or usage-based model. However, the following cloud approaches provide considerably various product abilities including PaaS (platform as a service), IaaS (infrastructure as a service) and SaaS (software as a service). The following Figure illustrates some remarkable differences among various cloud data warehouse approaches.

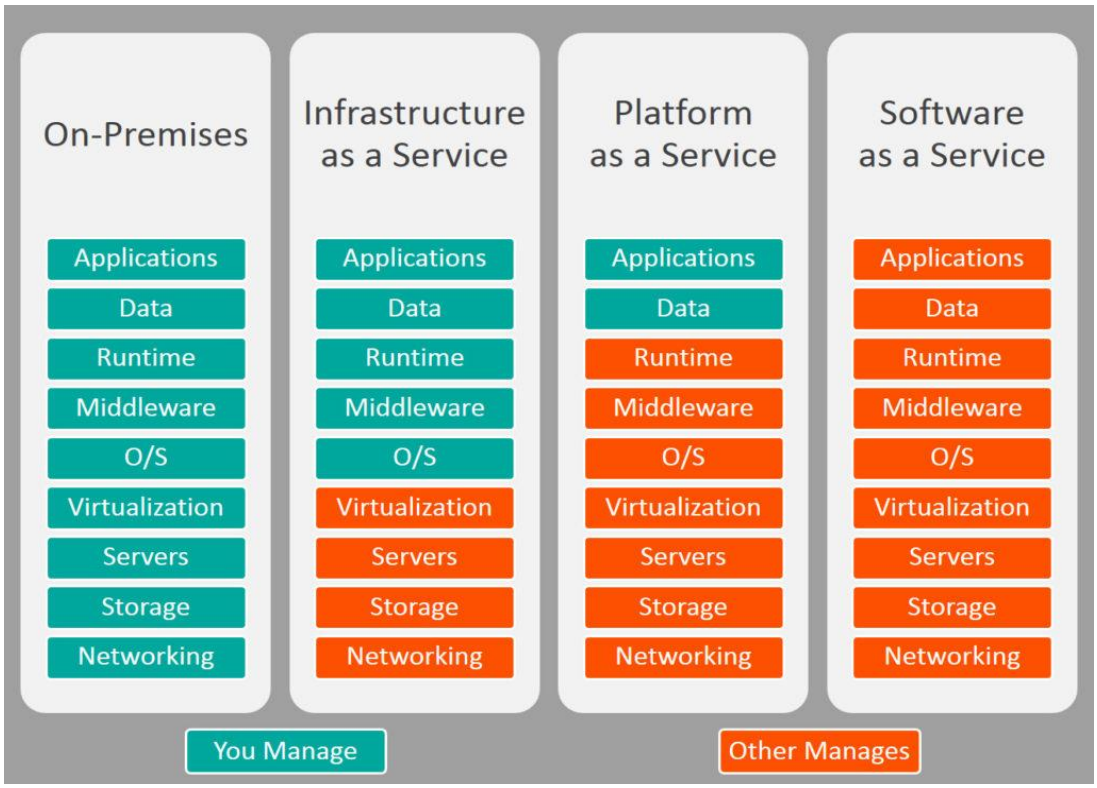

Figure 5. Summary of key differences of On-premises, IaaS, PaaS, SaaS infrastructure (Watts, 2017)

As shown in Figure 5, IaaS provides virtual space, storage and resources through a server helping to create a place called "infinite scalability''. Operating system and applications are installed and updated by customers. They manage all aspects of data warehouse hardware and software. This gives them more flexibility in using resources for what purpose. With this type of cloud approach, businesses can easily leverage level up or down infrastructure to meet demand at an effective price without building internal servers for storage and support. IaaS is on the increase due to the explosion of artificial intelligence (AI), Business

Intelligence (BI), Internet of Things IoT and cloud-based products - all of which require large amounts of storage space and computing power.

PaaS provides hardware as well as software as a cloud service. The provider manages all hardware deployment, software installation such as operating systems, databases, web servers, and programming environments. However, customer is responsible for software management and utilization. In addition, it allows customers to focus on specific applications, terminal services rather than wasting time on the operating system.

SaaS gives opportunity for customers accessing to cloud-based software without managing the infrastructure and the platform it is running on. The data warehouse company supplies all software and hardware as well as all aspects of managing them. (Antonopoulos and Gillam, 2017)

Generally, an enterprise has a proper selection based on the benefits and drawbacks of the different offerings based on usage, security and availability. The architecture of a modern data warehouse nowadays varies greatly depending on their target to the market. Currently there are four popular technology vendors providing these services consisting of Amazon redshift, Microsoft azure, BigQuery and Snowflake.

Amazon Redshift was officially published in 2012 contributing to the larger cloud-computing platform Amazon Web Services. It is truly a virtual version of a traditional data warehouse and is used for large scale database with business intelligence tools. Redshift was built based on the massive parallel processing (MPP) ParAccel by Actian so as to manage tremendous scale database. It means that columnar storage technology is used for parallelizing and distributing queries across various nodes to take full advantage of all accessible sources. (Thelwel, 2015)

Big Query is a RESTful web service that is capable of interactive analysis of extensively large datasets in connection with Google Storage. The heart of Big

Query is a novel query engine built on Google's Dremel project. Google call Big Query as an external version of Dremel query service software which is conducted to query billions of rows of data in just a few seconds. The other consideration that sets Big Query apart is that the whole process of provisioning, assigning, and maintaining of resources, is fully taken care of automatically by Google. This makes Big Query ideal for small organizations or teams that prioritize ease of use over maximum performance. (Inc, Google, 2018)

Microsoft Azure SQL Data Warehouse is an elastic distributed and enterprise-level data warehouse in responsible for handling a vast amount of relational and nonrelational data leveraging the broad ecosystem of SQL Server. This model is known as the first cloud data warehouse of Microsoft offering SQL capabilities. Microsoft's MPP (massive parallel processing) architecture is also used to conduct highly required on-premise data warehouse systems. (Anon., 2018)

In between the customizability of Redshift and the ease of Big Query there is Snowflake. In essence, Snowflake is a custom query engine and data storage format built on top of AWS architecture: Storage is handled by S3, while computing is taken care of by EC2. On the other hand, a new SQL database is also designed with an extraordinary architecture. And this true SaaS service had been chosen as the tool for case company case so it will be explained more detail in the section below.

3.4 Snowflake computing

In this section, snowflake computing will be described more clearly with respect to its characteristics, components, various platforms and different tools which are used during its implementation.

3.4.1 Snowflake's innovations

Snowflake is a cloud data warehouse founded by three data warehousing experts: Thierry Cruanes, Benoit Dageville and Marcin Zukowski in 2012 and finally introduced publicly by Bob Muglia in 2014. At first, it was built based on the

Amazon Web Services (AWS) and afterward in Microsoft azure platform for preview. Their goal is to fully support Snowflake on both AWS and Azure, while taking advantage of the unique capabilities that each platform can provide customers. Snowflake is a true SaaS offering which means Zero management or no hardware, software to implement. All ongoing management is administered by Snowflake.

Snowflake platform in in charge of bringing together all users, all data and all workloads in a single cloud service and enabling businesses to access data from any location. In addition, Snowflake runs completely on cloud infrastructure; hence, virtual compute instances are often used for its compute needs.

Overall, Snowflake's innovations break down the business and technology barriers that company still experience with other data warehouse enterprises. Below is the table highlighting snowflake task versus the manual efforts typically required with ordinary data warehouses.

Table 3. Comparison of Data Warehouse Management Efforts. (Nixon, 2018)

✓ = Effort required (actual system tasks may vary)

	Task	Ordinary On-premises Data Warehouse	Ordinary Cloud Data Warehouse	Snowflake, Data Warehouse-as-a service
Infrastructure	Hardware acquisition/config.	✓	✓	Built-in
	Software license	✓	✓	Not required
	Cluster node setup/ management	✓	✓	Built-in
	Software provisioning	✓	✓	Not required
	Data pipeline from source(s)	✓	✓	✓
	Compute scaling administration	✓	✓	**Snowflake Data Warehouse -as-a-Service**
Data & Service Protection	Data protection	Platform	Cloud Platform	
	Data retention	✓	Cloud Platform	
	Hardware/node failure protection	✓	✓	
	Availability/Disaster recovery	✓	✓	
	Service monitoring & alerting	✓	✓	
Security	Physical security	✓	Cloud Platform	
	Deployment security	✓	✓	
	Security monitoring	✓	✓	
Database Management & Tuning	Index management	✓	✓	
	Data partitioning	✓	✓	
	Workload balancing	✓	✓	
	Metadata & statistics mgmt.	✓	✓	
	Query optimization	✓	✓	

As described in Table 3, Snowflake particularly brings to customers all convenience and flexibility since its service created with built-in cloud infrastructure and database management capabilities, as a result, offering zero management. (Nixon, 2018). In conclusion, Snowflake connects data warehousing strength, big data platforms flexibility and cloud infrastructure elasticity with a much lower cost comparing to traditional way. Its service is automatically engineered with built-in cloud infrastructure and database management capabilities. In the listings below, Snowflake provides 4 important key benefits to users:

Firstly, Snowflake uses standard SQL query language. This language is popular in every corner of companies nowadays so it would be a huge advantage for organization who decide to use it.

Furthermore, Snowflake supports popular data formats for instance Avro, ORC, JSON, Parquet and XML. Snowflake creates a single source to easily store, integrate and extract critical insight from petabytes of structured and semi-structured data. This will help to integrate all heterogeneous data types into a single data warehouse. This can ultimately accelerate more value on the database.

Moreover, Snowflake has a unique architecture with a subtler approach by separating data processing, data storage and data consumption. Their storage is completely separated with compute. it can scale up or down easily depend on user demand and they only pay for what they need.

Last but not least, Snowflake offers two separate user experiences for both data engineers and data analysts. Data engineers are responsible for running and managing the system, they frequently work from the application side, whereas data analysts extract business insights from the data which is available after the process of running system by a data engineer. Two separate process together with instant scalability help them to handle concurrency bottlenecks during high demand conveniently.

3.4.2 Snowflake architecture

Snowflake data warehouse is built up with three main components. At first, the architecture of Snowflake is seen as a combination of shared-nothing architectures (SN) and hybrid of traditional shared-disk architectures (SD). In shared-nothing architecture, every node is autonomous and self-responsible. More specifically, none of the nodes have to share disk storage, memory nor responsibility with others. As a result, data is completely segregated, with each node having full autonomy over its distinct subset. In contrast, shared disk is absolutely contrary to shared-nothing. All data except main memory is accessible

from all cluster nodes through the interconnection network. Any machine can adjust data if it's essential. (Stopford, 2009)

As a combination of SD and SN database architecture, snowflake inherits their key advantages for both data management simplicity, free accessibility, good extensibility and availability. Similar to shared-nothing architectures, it uses MPP (massively parallel processing) compute clusters in which each node stocks a small fragment of a whole data set. Notwithstanding, they use a central data repository which is approachable from all compute nodes in the database based on shared-disk model. Figure 6 below describes three main components that create snowflake architecture.

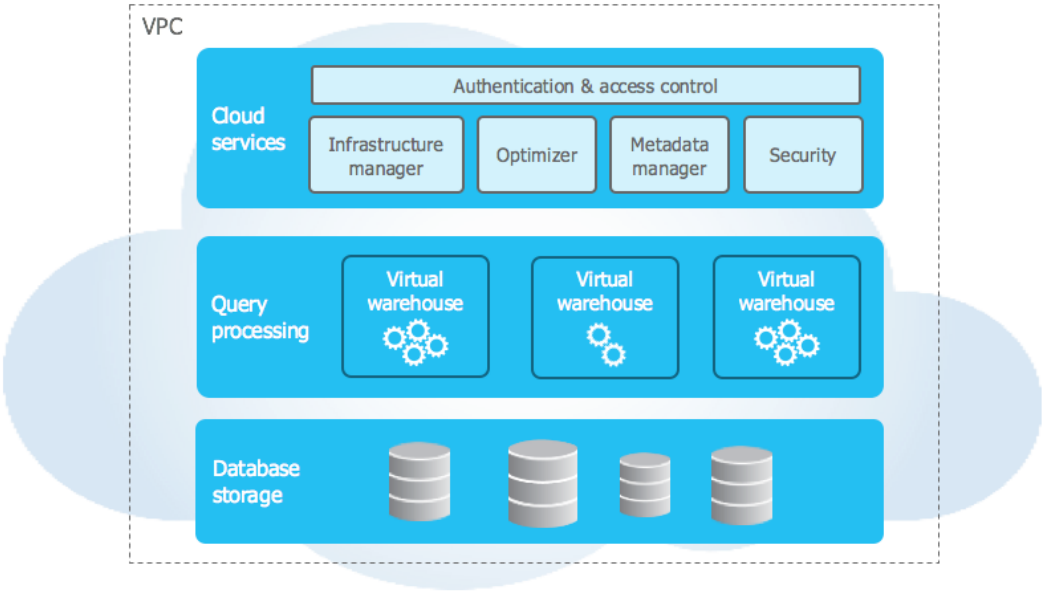

Figure 6. Snowflake data warehouse architecture (Snowflake, 2017)

As Figure 6 describes, Snowflake composes three main layers including database storage, query processing and cloud services. When data is loaded into database storage of Snowflake, it is responsible for reorganizing that data into its internal optimized format. Its key characteristic is its native support for both structured and semi-structured data without a complex ETL pipeline or preprocessing. Snowflake handles the storage of database such as file size, structure, compression, statistics. Staff can only access data objects through SQL query operations by using Snowflake.

In query process, Snowflake uses “virtual warehouses”. Each virtual warehouse is an MPP compute cluster composed of numerous compute nodes allocated by Snowflake. As discussed above, each virtual warehouse is irrelevant with other virtual warehouses dues to the fact that they are all independent.

Ultimately, the cloud services layer is a mixture of services such as authentication, infrastructure and metadata management, security, optimization, and access control. Those services connect all of the divergent components of Snowflake with the target of ensuring all smooth activities through snowflake and processing user requests, during the procedure from login to query dispatch.

3.4.3 Snowflake data lifecycle

All database in Snowflake is coherently performed as tables so as to be easily modified. Figure 7 below illustrates the snowflake data lifecycle in more detail.

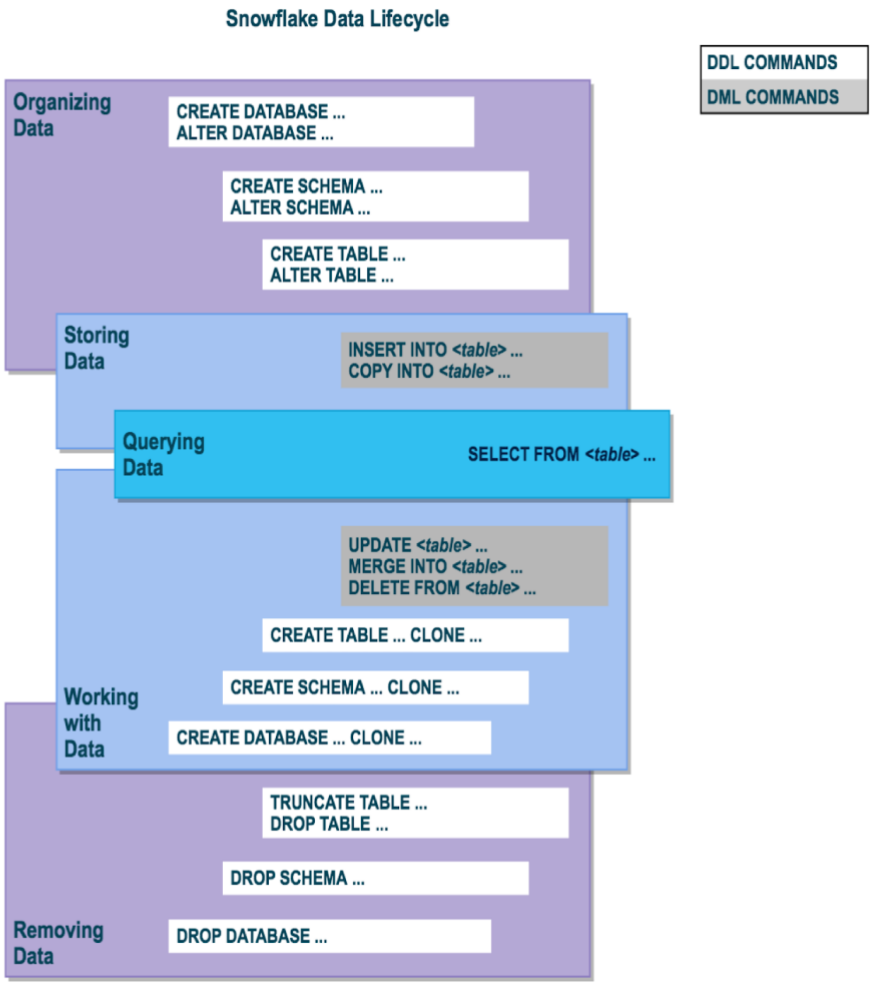

Figure 7 Snowflake data lifecycle (Snowflake, 2017)

As illustrated in Figure 7, at the first stage of data life cycle, data can be organized in databases, schemas, or tables and there is no limitation for the number of databases, schemas or tables users wish. In the storing stage, data can be inserted directly into tables. In the querying data stage, SELECT statement is often used to query data. Once the data are stored in tables, users can issue DML operations on them or even perform DDL actions such as cloning entire database, schema and tables. At the last stage, performing truncation or dropping entire tables, schemas and databases is achievable through DML command for example DELETE. (Snowflake, 2017)

4 Case company

4.1 Business concept

Case company is a job board and recruiting media company that gathers and displays all the job postings in one place. Job seekers come to company not only for job searches but also for many useful pieces of career advice, news about working life and recruitment. Not only that, the case company helps multiple enterprises attract active and passive job candidates through bolstering their brands via video, news on social media as well abundant external website advertisements.

Case company is using internet display advertising network to drive traffic to job advertisements hosted in the case company main website. As a result, job advertisements get attention from readers/viewers, and a portion of those will potentially apply for the job and they become applicants, naturally some of these get employed in the end of the recruiting process. Therefore, one of the strategic goals of the case company is to produce relevant and suitable applicants to the customer companies. This raises the question of managing the job advertisements' delivery process efficiently, results in the recruitment of tracking nearly real-time delivery process and target guarantee for each job advertisement.

4.2 Current situation

This section will introduce the case company's current advertisement management situation, scattered data and data analysis challenges and solution for that.

4.2.1 Current internet advertisement management

At case company, campaign team has to manage several thousand internet advertisements daily and currently the team is using only excel sheets to manage the advertisement data. These data have become gradually larger as case company enjoys its higher demand of services caused by the increase of B2B

clients and end users. It means data flow is painfully slow and there is a lack of automated analytics reports. To be more specific, manual tasks include checking advertisement spends and performing status, documenting them into excel sheets, necessarily, implementing some corresponding charts/ graph based on the documented data. To avoid these time-consuming tasks and improve the KPI, creating automated reports that present advertisement spends and performing status plays an important role in campaign management.

4.2.2 Internal data analytics burden

The internal data in case company is already huge, due to the fact that several thousand job advertisements come to the database weekly, as averagely about a thousand job advertisements daily. Additionally, internal database contains not only job entry data but also other data with both structure or semi-structure types. If analytics procedure implements in the current internal database, it will unexpectedly create an unavoidable burden in internal database. While the tech team always gives higher priority to optimize the back-end performance, cloud data warehousing candidate potentially offers great service for analysis, as it was designed to do analytics work.

A few studies have proved that when using cloud data warehouse, analytics processing is separated from the main internal transactional database, leaving the transactional database free to focus only on transaction. According to Inmon (2002), databases have divided into two categories, classified by the needs of their users. While serving operational needs such as transaction processing is the focus of the first category, serving informational or analytics needs is achievable in the other category. Inmon (2002) also pointed out that the split occurred for several reasons, and one of those reasons is the data serving operational needs is physically different from the data serving in-formational or analytics needs.

Thanks to the scalability and BI tools integration of cloud data warehouse, case company does not have to worry about the size of data storage as well as how to

analyze and visualize data more accurately and efficiently. Figure 8 below depicts current data situation in case company.

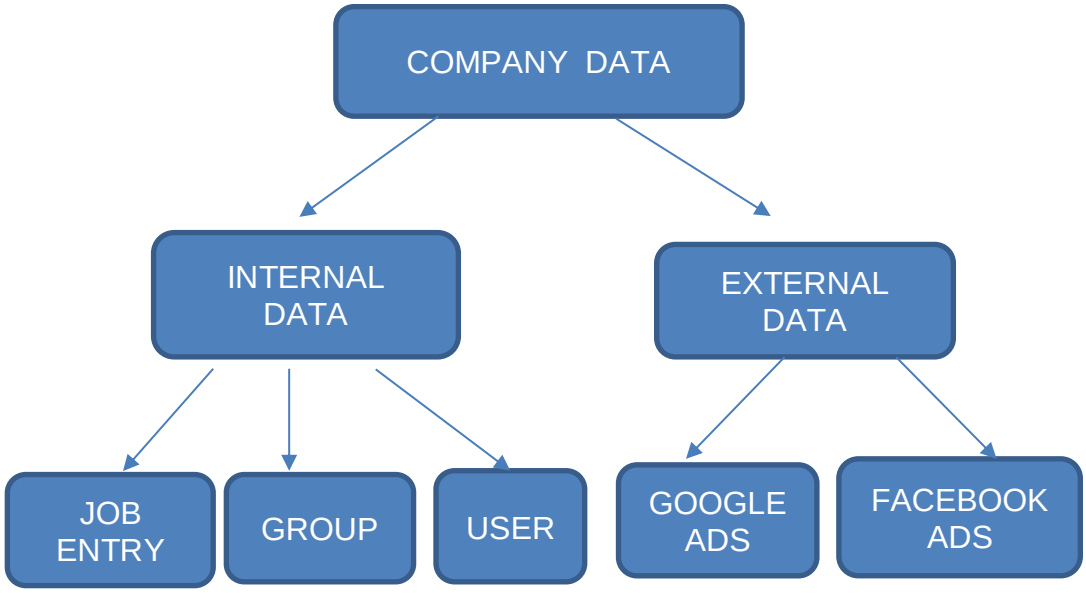

Figure 8. Case company X database scenario

As can be seen in Figure 8, case company has 2 main sources of data flow, namely, internal and external data. Internal data contains data for example job entry, user and group tables while external data are obtained from Facebook and Google ads services. The central issue addressed here is the integration of two data sources together as they have different schemas and data models. Practically, moving external database into internal database does not seem to be a good solution because it has been noted earlier as creating more burden to already heavy transactional internal database.

4.2.3 Solution for analytics data

In this section, a solution for case company's data analytics issue is introduced. To answer to the burden analytics data question described previously, project leader and business managers in company have discovered a straight forward answer, which is moving all necessary data into cloud data warehouses. Figure

9 below describes how data flow from internal and external database into cloud data warehouse – Snowflake.

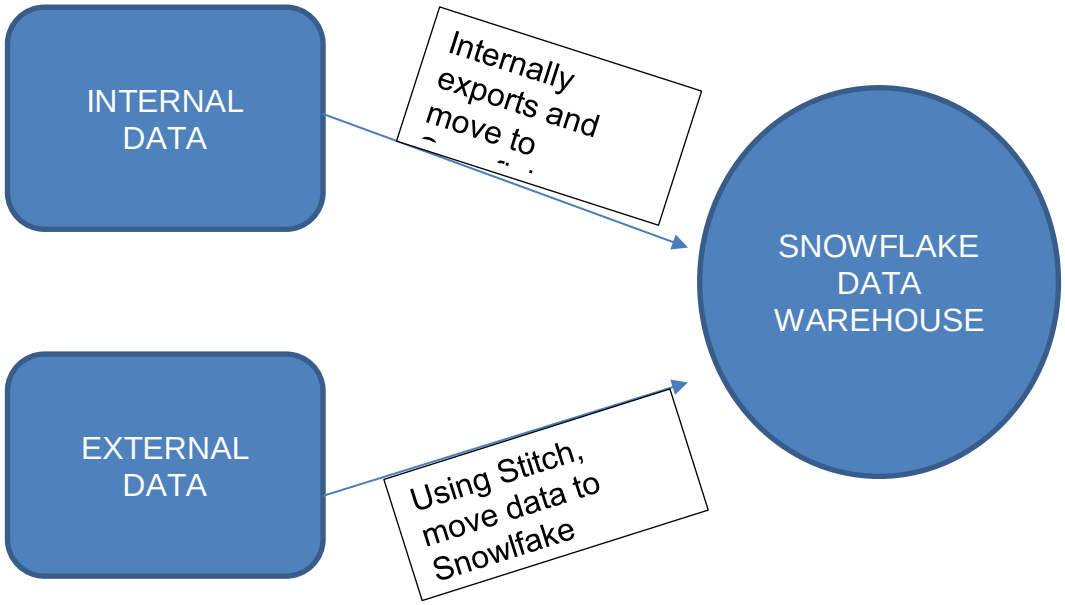

Figure 9. Solution for case company analytics data.

As shown in Figure 9, moving data to Snowflake will mainly contain two stages which are designed to handle internal and external data. The first stage consists of exporting internal data and performing ETL process of that data into Snowflake. This stage ‘s main tasks include setting up SSH tunneling connection to secured internal database, performing exporting internal data into CSV files by using SQL codes and uploading CSV data files into Snowflake. ETL process is then triggered after newly created CSV files uploaded into Snowflake. The thesis covers details of internal data ETL process in section 4.5.7.

The second stage, in the later chapter of the thesis, will be called as “stitching” external data stage, is a process that comprises various tasks. These tasks include setting up Stitch, a third-party service that helps moving data into Snowflake, creating stitch database in Snowflake, connecting Stich to Snowflake, running ETL process with Stitch for Google and Facebook ads.

4.3 Build a cloud data warehouse in snowflake

This section will go through several steps of preparing a cloud data warehouse: creating databases, tables and virtual data warehouses.

4.3.1 Log in to Snowflake

There are two ways to login into snowflake, either using web interface or with snowSQL, a command line interface that was recommended by Snowflake. Log in via snowSQL can be done with the following codes:

```
$ snowsql -a <account_name> -u <user_login_name>
```

Because of high frequency login into snowflake using snowSQL, project leader suggested placing all necessary configurations for snowflake into snowflake config file which can be found from Snowflake config file (Appendix 1). Therefore, the next time login shall be:

```
$ snowsql -c <connection_name> <sql_execuable code>
```

After success authentication in Snowflake, to build a cloud data warehouse, the project followed three steps, namely, creating databases, creating tables (define also schema) and creating virtual data warehouses.

4.3.2 Create databases

The following code creates a new database name X_STAGING_DEV_DB in Snowflake:

```
create or replace database X_STAGING_DEV_DB;
```

Sometimes, listing databases in terminal helps the author quickly view recently created databases. The author uses the following codes to display database visualization through snowSQL:

```
select current_database(), current_schema();
```

The case company has 5 databases, they are created for different purposes, as described in Table 4:

s

Table 4. Different databases and their purposes.

Databases	**Purposes**
X_MANUAL_DEV_DB	For all manual developments and hacks
X_STAGING_DEV_DB	Schema: XDB Only for testing manual imports such as postgreSQL import and export from X DB
X_STAGING_PROD_DB	Production imports from X DB Production imports from Stitch Schemas: XDB, STITCH_FB_ADS, STITCH_GA
X_ANALYTICS_DEV_DB	Data warehouse for development Schema: DW
X_ANALYTICS_PROD_DB	Data warehouse for production Schema: DW

Manual database is a perfect fit for developers when implementing some handy hacks or running random tests in database. Additionally, there are two different kinds of databases, namely, staging and analytics. Staging databases or staging areas are used to load data from various sources, data can then be modified, cleansed before the final load into the data warehouse. Thanks to staging area, should the ETL process be unsuccessful at some points, there is no need to impact sources to extract the data again. Analytics databases contain schema DW and all complex querying scripts run there.

4.3.3 Create tables

After creating databases, different tables area created in Snowflake to match those tables from sources. Each database contains different tables, taking staging development database as an example, this database includes the following tables listed in Table 5.

Table 5. tables and schemas in staging development database.

Tables	**Schemas**
REPORT	STITCH_GOOGLE_ANALYTICS
CAMPAIGNS	STITCH_FACEBOOK_ADS
ADS_INSIGHTS	STITCH_FACEBOOK_ADS
ADS_SET	STITCH_FACEBOOK_ADS
ADS	STITCH_FACEBOOK_ADS
USER_GROUP	XDB
USER	XDB
JOBENTRY	XDB

Table 5 illustrates 8 different tables and their corresponding schemas, despite different schemas, creating a table in Snowflake follows the same syntax as shown in Figure 10.

```
create or replace table XDB.user (
  id integer,
  username string,
  first_name string,
  last_name string,
  email string,
  is_active boolean,
  last_login timestamp,
  _extracttime timestamp
);
```

Figure 10. Create table User (schema XDB)

As can be seen in Figure 10, in staging development database, the codes first declare prefix schema “XDB” before table name so that new created table follows the schema. According to Snowflake documentation, the number of columns in the tables, as well as their positions, data types must be correspondingly matched the fields in the CSV data files that are staged.

4.3.4 Create virtual data warehouse

To create a virtual warehouse, snowflake provides 2 options: using web-based worksheet or snowSQL CLI. The author selected the second option, codes that create 2 virtual warehouses described in Figure 11.

```
CREATE WAREHOUSE X_COMPUTE_WH
WITH WAREHOUSE_SIZE = 'XSMALL'
WAREHOUSE_TYPE = 'STANDARD'
AUTO_SUSPEND = 600
AUTO_RESUME = TRUE
MIN_CLUSTER_COUNT = 1
MAX_CUSTER_COUNT = 2
SCALING_POLICY = 'STANDARD'
COMMENT = ''
CREATE WAREHOUSE X_MANUAL_WH
WITH WAREHOUSE_SIZE = 'XSMALL'
AUT0_SUSPEND = 300
AUTO_RESUME = TRUE
MIN_CLUSTER_COUNT = 1
MAX_CUSTER_COUNT = 2
SCALING_POLICY = 'STANDARD'
COMMENT = ''
```

Figure 11. Creating virtual warehouse with snowSQL.

Figure 11 illustrates how to create two new data warehouses; noticeably, manual warehouse uses smaller “XSMALL” size and set up with smaller auto - suspend time than compute warehouse does. Auto-suspend time is used to define a period of time when there is no activity and the warehouse can be suspended to avoid consuming unexpectedly wanted credits. Auto-resume sets to True to enable warehouse to automatically resume when new queries are submitted.

4.4 From operational data to analytics data - ETL architecture

This section introduces generally ETL architecture which is implemented to solve case company's data analytics problem mentioned earlier. The architecture is illustrated in the Figure below.

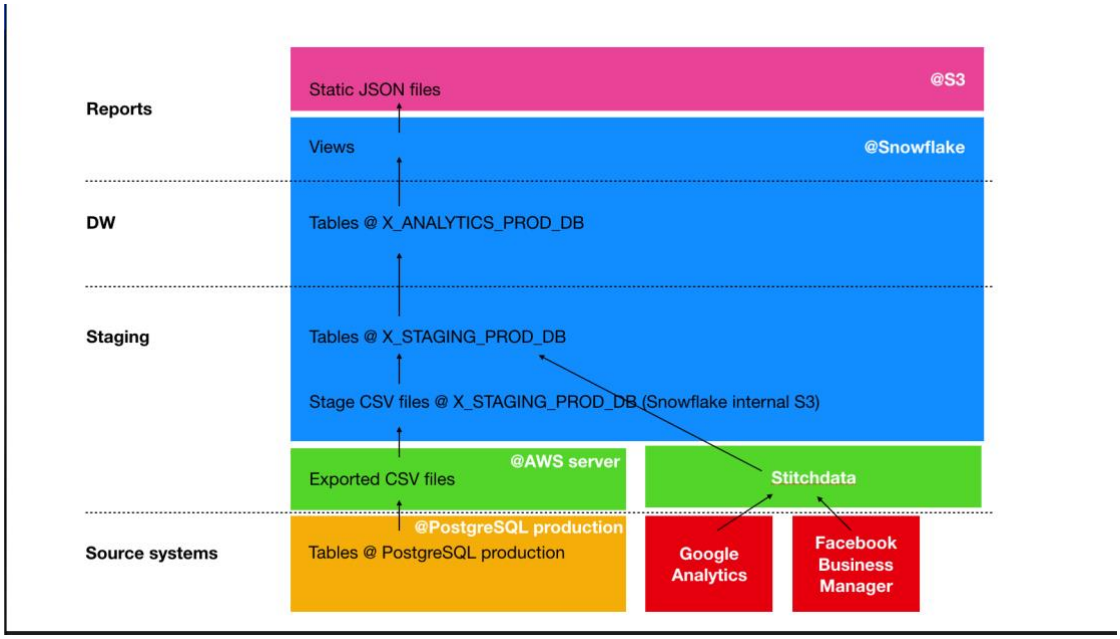

Figure 12. Data flow and ETL architecture

In Figure 12, there are three small tier architectures: Staging, DW and reports. Staging is the place where the data from external or internal sources are gathered, it stands for E (Extraction) of ETL. In addition, DW (data warehouse) is where the data are loaded (L) and the transform (T) happens between staging and DW. Finally, reports essentially refer to data marts for end-user reports. However, in the project, reports have been initially considered as light-weight data marts which means they are practically static JSONs although data marts are usually created using tables (not JSONs). To explain more clearly the mentioned architecture, the following sections below go through every layer of the data flow, from exporting data to creating analytics data that is populated into various JSONs.

4.5 Exporting internal data

This section demonstrates how to export data from internal sources. Internal data are first selected and copied into different CSV files, then they are ready for uploading into Snowflake staging area. This exporting task runs daily and hourly automatically by scheduling bash script in one AWS server that connects to internal database server. Moreover, external data come from Google ads and Facebook ads are exported, staged, extracted, transformed and loaded into Snowflake staging area by a third-party service called “Stitch data”.

4.5.1 Connecting to internal database server

To export data from internal source, the first step is to establish a secure connection (SSH) between one AWS server and internal database server. At the case company, internal database server also runs in an AWS instance which has PostgresSQL installed.

SSH is a well-known software for tunneling all of the traffic from a machine A (client) to machine B (server) where user of machine A has account on. SSH ensures that traffic data is securely encrypted and protected from modification between two machines. For example, SSH encrypts the data sent from machine A to machine B and automatically decrypts it when the data reaches machine B, and the reason behinds secure connection is that SSH uses modern, secure encryption algorithm. (Daniel and Richard, 2001). SSH tunneling or often called SSH port forwarding reroutes a TCP/IP connection to pass through an SSH connection. In order to establish a secure connection from a local machine to postgresSQL database server of case company, the project implements SSH tunneling.

```
ssh -M -S x-control-socket -fnNT -L xxxxx:localhost:yyyy X-db
```

The command above tells SSH to forward connections from local port “xxxxx” to local port “yyyy”, default chosen port that postgreSQL server listens to. In detail, every time a tunnel is created, the script establishes a new SSH connection and gets a shell which seems to be very unnecessary. To avoid this, using tag “nNT”

will tell SSH not to allocate a new tty every time and focus only on port forwarding. Additionally, there are tag “M” which places SSH client into master mode for connection sharing and the tag S which specifies the location of a control socket for connection sharing.

From the command, there is “x-control-socket” argument which handles the socket control. To be more specific, control socket tells SSH enable the sharing of multiple sessions over a single network connection under controlled by socket and this actually improves connection speed. Indeed, any additional sessions will try to reuse the master instance’s network connection rather than initiating a new one and sessions will fall back to normal connection if the control socket does not exist or is not listening. Moreover, to control an active connection multiplexing master process, the command below uses tag “O” to check that master process is running.

```
ssh -S X-control-socket -O X-db
```

As soon as a successful SSH tunneling connection to database server is established, from local machine, running any PostgreSQL commands is simple as described in the following code:

```
psql -h locahost -p xxxxx -u [user_X] [database_X]
```

The command above tells PostgresSQL client connect to PostgreSQL server database named “database_X” at port “xxxxx” under user name “user_X”.

4.5.2 Exporting internal data

Exporting data is implemented by using PostgreSQL which selects and copies different fields from tables into CSV files. These CSV files then get uploaded immediately to Snowflake staging area. One example of exporting user table from the database server is shown in the following codes:

```
COPY (select id, username, first_name, last_name, email, current_timestamp as
_extracttime from user_table) TO ‘FILE_PATH’ DELIMITER u&'\\0001' CSV HEADER;
```

The codes above extract necessary fields from “user_table” into CSV file located in “FILE_PATH” with header and comma delimiter. Afterward, the master process is terminated as described in the following command, tag O is used again with parameter “exit” to request the master process to exit.

```
ssh -S X-control-socket -O exit db.
```

Tag S is used again to tell SSH control the closing of connection and hence, the socket file “X-control-socket” is also eliminated every time exporting database processes finish.

4.6 Stage exported data files

This section covers process of cleaning stage at Snowflake staging area and uploading CSV files into snowflake. Firstly, before new files are put/uploaded into Snowflake internal stage, it is important to remove old staged files. The command below explains how to remove old staged files directly by calling remove function on staging area name.

```
snowsql -c [connection_alias_name] -q "use database [staging_database]; use
schema [schema_name]; remove @[staging_name]"
```

Essentially, there is a slight difference between “staging_database” and “staging_name”. While “staging_database” can be either “X_STAGING_DEV_DB” or “X_STAGING_PRO_DB”, “staging_name” can be either an internal stage (stores data files internally within Snowflake, can be either permanent or temporary) or external stage (references data files externally stored in AWS S3 for example).

Secondly, once the cleaning stage process finishes, recently created CSV files will be uploaded to Snowflake, the command line below depicts uploading process of table “user” exported CSV file into Snowflake.

```
snowsql -c [connection_alias_name] -q "use database [staging_database]; use
schema [schema_name]; put file://[exported_user_file_path] @[staging_name]
auto_compress=true"
```

The codes above tell snowsql to put/upload exported CSV file “exported_user_file_path” with corresponding “staging_database” and “schema” into “staging_name” in Snowflake.

Last but not least, checking done work after each step of a whole process is one useful habit of many programmers. Showing a list of staged CSV files determines staging process finished flawlessly.

```
list @[staging_name]
```

To summarize, at this stage, the project already achieved following milestones:

- Create a cloud data warehouse.
- Set up database, table, schema, staging area.
- Export internal data and stage data files into snowflake internal staging location.
- Export and stage external data using “Stitch data”.

The next milestone of the project includes one of extremely important processes in cloud data warehouse – ETL process.

4.7 Extract – Transform – Load data from external sources using Stitch data

This section introduces a Third-party service – “Stitch data”, which was recently integrated into talend.com in November 2018. Stitch data provides ETL solution for external data from Google ads and Facebook ads.

“Stitch data” is a quite famous name leader in the fast-growing, self-service data integration market which provides simple, extensible ETL build for data teams and fully supports modern cloud data warehouse platforms, including Snowflake. Because of the scope of this thesis, the author will not document in detail how ETL process running in Stitch, but rather explains what kinds of tables are created in Snowflake databases, and also their corresponding schemas. There is an

example of “Stitch data” tables created in database X_STAGING_DEV_DB illustrated in the following figure:

Table Name	Schema
REPORT	STITCH_GOOGLE_ANALYTICS
CAMPAIGNS	STITCH_FACEBOOK_ADS
ADS_INSIGHTS	STITCH_FACEBOOK_ADS
ADSETS	STITCH_FACEBOOK_ADS
ADS	STITCH_FACEBOOK_ADS

Figure 13. Stitch schema tables (screenshot from Snowflake)

As described in Figure 13, there is only one table with schema “STITCH_GOOGLE_ANALYTICS” used for storing fact1 and fact2 values (fact values will be explained in the following chapters) fetched from GA service whereas this number is four in Stitch Facebook schemas. More specifically, Facebook campaign structure needs three parts to run, namely, Campaign, Ad set and Ad. A campaign may contain one or more ad sets and ads while Ad sets contain one or more ads in which a user define targeting, budget, schedule, bidding and placement at the ad set level (to create audience for user’s ad). Finally, ad is what a user’s customers or audience will see, this usually involve setting up images, videos, texts which makes up ad’s creatives (inc, 2019). The ADS_INSIGHTS table contains data of ads reporting and analytics which can be different metric values of visitor numbers and click event count.

4.8 Extract – Transform – Loading (ETL) process for internal data

To automate ETL processes, Cron jobs (schedule bash scripts) have been set up in one AWS server which has privileged access to the case company database server. These jobs are set to run daily and hourly and programmed to report also logs, bugs in case something happens in the ETL process.

Each ETL process once triggered, is designed to go through 2 or 3 steps, depending on different types of ETL process. In the first step (pre-process), common practices include specifying database and schema to be used, listing staging area, running a set of selecting all entries from the relevant tables. The second step is loading staging area or data warehouse. The final step (post-process, only in staging area) contains listing staging areas and tables.

4.8.1 ETL into staging area

ETL process for staging area is described in Figure 14 below.

```
/* Preprocess */

use database &{STAGING_DATABASE};
use schema [schema_name];

list @[staging_area_name];

select count(*) from [table_name];


/* Loading staging area */

create or replace temporary table temp_[table_name] like [table_name];

copy into temp_[table_name]
  from @[staging_area_name]
  pattern='.*_[table_name]-.*'
  on_error = 'skip_file';


delete from [table_name] where id in (select id from temp_[table_name]);
insert into [table_name]
  select * from temp_[table_name];

remove @[staging_area_name] pattern=".*_[table_name]-.*";

/* Postprocess */

list @[staging_area_name];
select count(*) from [table_name];
```

Figure 14 ETL into staging area.

As can be seen in Figure 14, this ETL process goes through 3 steps, namely, pre-process, loading staging area and post-process. The second step is the most important one due to the fact that the codes from this step copy entries from CSV

files and map them into the correct data models / tables in staging database. To be more specific, a temporary table which is similar to the current table in staging database is created or replaced. After that, this temporary table is loaded with new fetched data from the relevant CSV file, then entries from the current table, the ids of which are presented in the temporary table are eliminated. All entries from temporary table are inserted into the current table and CSV file is removed from staging area.

4.8.2 ETL into data warehouse

As soon as data have been loaded flawlessly into staging area, ETL processes continue in data warehouse. It extracts essential data from staging database, the extraction may not include all the field of the data models, but only some necessary fields of certain data models to create fact tables.

The first step of the ETL process indicates that “ANALYTICS_DATABASE” and “dw” schema will be used, as shown in the following codes:

```
/* Preprocess */
set s_database='&{STAGING_DATABASE}';
use database &{ANALYTICS_DATABASE};
use schema dw;
```

The following Figures 15, 16 and17 describe the second step of ETL process into data warehouse. They contain codes that create job entry dimension and fact tables, Facebook ad dimension and fact tables, Google Ads dimension and fact tables, user dimension table, respectively.

```
/* jobentry dimension table */

set s_table=concat($s_database, '.xdb.jobentry');
create or replace table d_jobentry as
  select
    id,
    heading,
    address,
    company_id,
    company_name,
    campaign_person_id,
    _extracttime,
    current_timestamp() as _loadtime
  from table($s_table);

/* jobentry fact table */

set s_table=concat($s_database, '.xdb.jobentry');
create or replace table f_dt_jobentry as
  select
    id,
    fact1,
    fact1_target,
    fact2,
    fact2_target,
    _extracttime,
    current_timestamp() as _loadtime
  from table($s_table);
```

Figure 15. Creating dimension and fact job entry tables.

As shown in Figure 15, firstly, the codes execute creation or replacement of dimension table “d_jobentry”, all fields of this table are derived deliberately from table “s_table” of which prefix “s” stands for staging area (project own naming convention). Amongst listing fields of the fact table, there is a field “campaign_person”, which indicates responsible person of the job advertisement campaign, is a foreign key of table “user”. Secondly, the codes create/replace current fact table “job entry” by selecting only “fact” data, including fact 1 and fact 2 from corresponding table in staging database. The following Figure contains codes that create Facebook ad dimension and fact tables.

```
/* facebook ad dimension table */

set s_table=concat($s_database, '.stitch_facebook_ads.campaigns');

create or replace table d_facebook_campaign as
  select
    id as campaign_id,
    name as campaign_name,
    regexp_substr(name, '[0-9][0-9][0-9][0-9][0-9][0-9]*') as jobentry_id,
    _sdc_received_at as _extracttime,
    current_timestamp() as _loadtime
    from table($s_table) order by type desc, name;

/* facebook ad fact table */

set s_table=concat($s_database, '.stitch_facebook_ads.ads_insights');

create or replace table f_facebook_campaign as
  select
      campaign_id,
      adset_id,
      ad_id,
      clicks,
      date_start as date,
      year(date_start) as year,
      weekiso(date_start) as isoweek,
      impressions,
      inline_link_clicks,
      reach,
      spend,
      unique_actions,
      unique_clicks,
      unique_inline_link_clicks,
      current_timestamp() as _loadtime,
      _sdc_received_at as _extracttime
    from
      table($s_table);
```

Figure 16. Creating Facebook ad dimension and fact tables.

As seen in Figure 16, Facebook ad dimension table “d_facebook_campaign” contains data such as campaign id, campaign name, job entry id, timestamp indicating when Stitch extracted the record from the source (“_sdc_received_at”), current timestamp from corresponding staging table. On the other hand, relevant fields such as “campaign_id”, “adset_id”, “ad_id”, “clicks”, “impressions”, “spend” fill up fields of fact table “f_facebook_campaign”. Creating fact tables from Google ad report is described in the Figure below.

```
/* Google ad fact table */

set s_table=concat($s_database, '.stitch_google_analytics.report');

create or replace table f_gdn_campaign as
  select
      campaign as campaign_name,
      adclicks as clicks,
      adcost as spend,
      start_date as date,
      year(start_date) as year,
      weekiso(start_date) as isoweek,
      sessions,
      users,
      impressions,
      pageviews,
      regexp_substr(addestinationurl, '[0-9][0-9][0-9][0-9][0-9][0-9]*') as jobentry_id,
      current_timestamp() as _loadtime,
      _sdc_received_at as _extracttime
    from
      table($s_table);
```

Figure 17. Creating Google ad fact table.

The codes in Figure 17 create or replace fact table “f_gdn_campaign” by selecting many fields of records from data obtained by Stitch (in staging area). Amongst them, important fields are “campaign”, “adclicks” (number of click events), “adcost” (cost per click), “pageviews”, “sdc_received_at”.

Because each job entry has one person who is responsible for running corresponding campaign, loading user table into data warehouse is an essential need. Also, since there is integrity relation between user and group data models, table “group” is also loaded into data warehouse, depicted in Figure 18.

```
set s_table=concat($s_database, '.X.user_group');

/* campaign group table contains campaign team member data */
create or replace temporary table campaign_group as
  select user_id, group_name from table($s_table);

set s_table=concat($s_database, '.X.user');

/* Dimensional table user */
create or replace table d_user as
  select
    a.id,
    a.email,
    group_name,
    a._extracttime,
    current_timestamp() as _loadtime
  from
    table($s_table) a left join campaign_group b on a.id = b.user_id;
```

Figure 18 Creating dimension table user.

As shown in Figure 18, the codes first create temporary table “campaign_group” and left-join it with later created dimension table “d_user”.

4.9 Extract - Transform – Load (ETL) process for report data

This section explains the generating process of daily report of running campaigns as mentioned earlier in the project objective with the outcome is various JSON files which are uploaded to AWS S3 bucket. There are totally 5 JSON files at the time this thesis is written, depending on the coming on-demand features, there will be more JSON files generated for different purposes. JSON files can be accessed securely only from company network or in some personal computers that have exceptional declared in AWS inbound and outbound rule.

To provide good data visualization of job advertisement campaign for stake holders, there is a small company internal dashboard implemented along with the project in which campaign report data can be easily sorted by week, team or campaign member. Although the project has been using Stitch data service to gather ad spend from Facebook and Google ads, the project has currently

excluded the ad spend reporting and focused only on the campaign success tracking.

4.9.1 Pre-process

Unlike ETL process for staging area, ETL process for report data contains through two sub processes: pre-process and creating view, exporting JSON process. In preprocess, report data uses database “ANALYTICS_DATABASE” and schema “report” as shown in the code below:

```
/* Preprocess */
use database &{ANALYTICS_DATABASE};
use schema report;
```

All the query executes later in this ETL process will take place only at defined schema and database. It is worth mentioning again that “ANALYTICS_DATABASE” is the main analytics database which already contains job entry dimension table “d_jobentry”, job entry fact table “f_dt_jobentry”, and user dimension table “d_user”.

4.9.2 Creating different views for success rate

Success rate is used to measure campaign performance This section will cover two smaller sections, firstly introducing 5 views of success rate and lastly explaining codes that construct these views in Snowflake. In the following Figure 19 to Figure 24, the author will go through SQL codes of creating 5 different type of success rate view in Snowflake. The views are described in Figure 19.

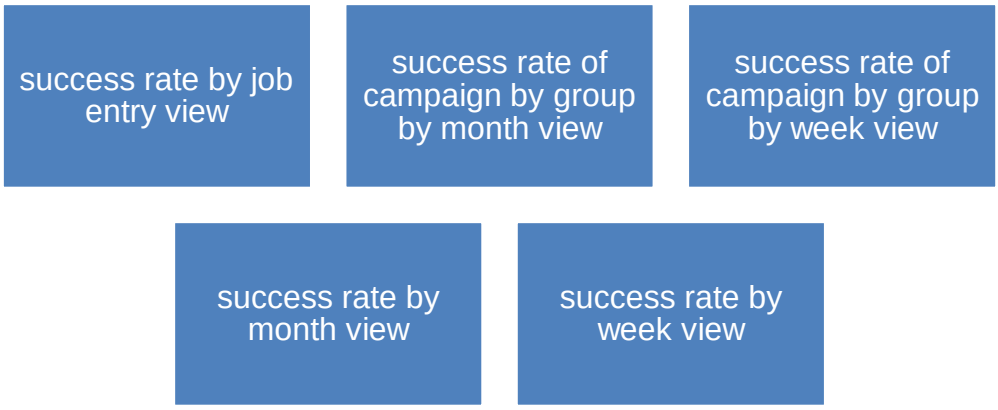

Figure 19. 5 different views generated during ETL process for report.

Figure 19 represents different views based on success rate which are later extracted into various JSON files and get copied into “export_staging” area and finally get uploaded to the case company’s AWS S3 bucket. The following codes in Figure 20 describes how the project manager created success rate by job entry view.

```
create or replace view success_rates_by_jobentry as
  select
    a.id as jobentry_id,
    b.heading as jobentry_heading,
    b.company_name,
    b.campaign_person_id,
    c.name as campaign_person_name,
    c.group_name as campaign_group_name,
    a.fact1,
    a.fact1_target,
    iff(a.fact1 < a.fact1_target, 0, 1) as fact1_performing,
    a.fact2,
    a.fact2_target,
    iff(a.fact2 < a.fact2_target, 0, 1) as fact2_performing,
  from
    dw.f_dt_jobentry a inner join
    (select * from dw.d_jobentry where year(date_ends) >= 2018 and campaign_person_id is not null) b
    on a.id = b.id
    left join
    dw.d_user c
    on b.campaign_person_id = c.id
  ;
```

Figure 20. SQL codes to create success rate by job entry view.

As shown in Figure 20, there are 3 tables that are involved in the success rate view function call:

- dw.f_dt_jobentry: job entry fact table from schema “dw”
- dw.d_jobentry: job entry table from dw schema “dw”
- dw.d_user: user table from schema “dw”.

Job entry fact table takes an inner join with job entry dimension table where only job entries that have campaign person are considered in calculating success rate. Then, the job entry table performs left join with user dimension table.

To define whether a job entry campaign perform well or not, Boolean values were used to measure the success of fact1 and fact2 data.

```
iff(a.fact1 < a.fact1 _target, 0, 1) as fact1_performing,
iff(a.fact2 < a.fact2 _target, 0, 1) as fact2_performing,
```

The codes calculate success rate by group weekly are shown in Figure 21.

```
/* SR by campagin group by weekly */

create or replace view success_rates_by_week_camapaign_group as
  select
    c.group_name as responsible_group_name,
    year(b.end_date) as year_ends,
    weekiso(b.end_date) as isoweek_ends,
    count(*) as campaigns,
    sum(iff(a.fact1 < a.fact1_target, 0, 1))/campaigns as fact1_performing,
    sum(iff(a.fact2 < a.fact2_target, 0, 1))/campaigns as fact2_performing,
    sum(iff(a.fact1 < a.fact1_target or a.fact2 < a.fact2_target, 0, 1))/campaigns as performing
  from
    dw.f_dt_jobentry a inner join
    (select * from dw.d_jobentry where year(end_date) >= 2018 and campaign_person_id is not null) b
    on a.id = b.id
    left join
    dw.d_user c
    on b.campaign_person_id = c.id
    group by responsible_group_name, year_ends, isoweek_ends
  ;
```

Figure 21. SQL codes to create success rate by group weekly view.

As can be seen in Figure 21, although this view has similar inner join and left join pattern as in success rate by job entry view snippet codes, there are few extra

fields presented in the codes that provide calculating success rate by group but not by job entry; these are group name, number of campaigns, fact1 and fact2 performing, overall performing, respectively. To be more specific, in order to group all campaign job entries by group, for those job entries that have campaign (or have campaign person), SQL codes select name of the corresponding group (later order selected fields or results by group name).

For success rate by week, because there is no need to group by group name, the same codes as previous success rate by group weekly would achieve result when group name query is omitted. For remaining success rate by group monthly and success rate by month codes, because there are only slightly different from the mention codes above, the author does not want to repeat similar codes here.

4.9.3 Creating different success rate JSON files

After creating view process finishes, there are 5 different JSON files created as shown in Figure 22 which are also copied into “export staging” area later.

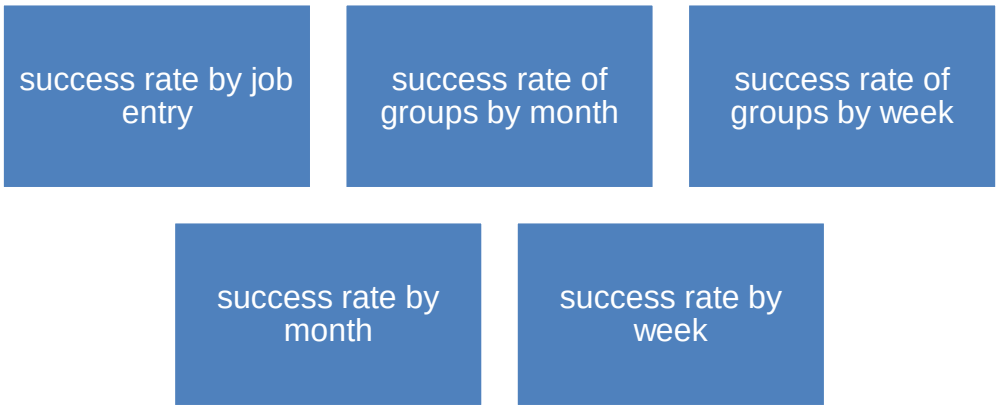

Figure 22. 5 different JSON files generated after ETL process for report done.

Besides 5 JSON files, there is one important JSON file which contains status of ETL process implementation each time It gets triggered. Project leader named it “status.json”, codes described in Figure 23.

```
copy into @export_stage/status.json
  from (select object_construct(
    'report_updated_at', current_timestamp,
    'last_extracttime', (select max(_extracttime) from dw.d_jobentry),
    'last_loadtime', (select max(_loadtime) from dw.d_jobentry),
    'jobentries', (select count(*) from dw.d_jobentry)
  ))
  overwrite=true single=true;
```

Figure 23. status JSON file.

As can be seen in Figure 23, this JSON file contains many useful fields that provide general view and debug log of the report. The report updated time, recent extract and load time and a number of job entries involved in the process get logged respectively.

The following codes explain how a new success rate by job entry JSON file is generated by extracting view in Snowflake. Because of data model secret, only one example of success rates by job entry will be presented.

```
/* create VIEW of SR by job entry */
copy into @export_stage/SR_by_jobentry.json
from (
  select object_construct(
    'jobentry_id', jobentry_id,
    'jobentry_heading', jobentry_heading,
    'company_name', company_name,
    'person_name', person_name,
    'group_name', group_name,
    'fact1', fact1,
    'fact1_target', fact1_target,
    'fact1_performing', fact1_performing,
    'fact2', fact2,
    'fact2_target', fact2_target,
    'fact2_performing', fact2_performing,
  ) from report.success_rates_by_jobentry
) overwrite=true single=true;
```

Figure 24. Generating success rate by job entry JSON file.

“success_rates_by_jobentry” JSON file is created by constructing a new object from corresponding view as explained in section 4.8.2. The object contains

various important fields, but worth mentioning are job entry id, page fact1 fact2 performing as well as company and campaign person that belong to the job entry.

5 Project results

This section summarizes the results of the project and reviews some challenges while implementing tasks under project's milestones. In addition, future improvement on cloud costs usage and extra AI model initialization are also mentioned.

5.1 Project outcomes

The first result of the project is a cloud data warehouse built in Snowflake. Its simplicity, scalability and flexibility offer great solution for the increased volume of data for case company in the future. Additionally, the project has set up automated cron jobs in AWS server for implementing important processes (exporting data and ETL) and logging all the details and errors of these processes. Whenever exporting and ETL processes finish or fail, slack messages will be broadcasted to case company slack channel.

The second result worth mentioning is various JSON files located in Amazon Web Service S3 storage bucket. These JSONs contain analyzed data from internal and external data and are refreshed daily or even in nearly real time if there is demand for that. These files are currently used in another project called internal dashboard for fetching success rates of various marketing campaigns.

As for the author, he has acquired basic to advance knowledge in data and cloud data warehousing in general and the capability of constructing data models and analyzing data stored in the cloud. Besides, he has a certain understanding on how to design ETL process and schedule it with advance bash scripts. Moreover, he starts to be aware of the importance of storing data in cloud and great potential of applying machine learning on data collected from different sources.

5.2 Future improvements

At the very first stage of the project, costs effectiveness was not focused with high priority. However, when the cloud data warehouse is up and running daily or more frequently, this raises costs optimization question. This improvement is achievable by re-examining the data model, measuring the costs of current ETL processes and coming up with better solution after checking up all aspects of the processes.

The codes under the hood work as the team's expectation, although several bugs were found and treated promptly. However, maintaining the clean codes plays an important role in code readability, understandability and further project documentation. Therefore, project leader has set up a plan for refactoring the whole project code base. The project will continue with the other goal, that is ad spend reporting. Since this reporting is important as the success rate tracking.

5.3 AI models for job advertisement spend estimation

As data keeps growing larger, big data is no longer a threat, but a chance for uncovering hidden patterns, correlations and other insights. It has been said that the new benefits that big data analytics brings to the table are speed and efficiency. For that reason, creating cloud data warehouse AI models that focus on tracking ad spend and provide real-time optimization (ad expense and content recommendation) is worth considering.

6 Conclusion

The incredible pace of change in the data center today is making companies more challenging. They are struggling to get a business model which is able to take advantage of all advanced technologies with the expectation of rapid accessibility to service. IT has experienced the same phenomenon but at a much faster pace. As a consequence, cloud computing was born as a solution for this thriving market. It has created new paradigms that align with other trends such as Big Data, Virtualization or Security.

This thesis has provided a comprehensive review about traditional data warehouse as well as cloud-based data warehouse and various key considerations between them in theory. Besides, the author also introduces the implementation and management of snowflake computing for the case company in order to get insights of job advertisement campaigns. This implementation will hopefully be a general tutorial for everyone who wants to create a cloud computing data warehouse in Snowflake from scratch since it covers outright detail from creating a new cloud data warehouse to implementing relevant ETL processes along with the given data which are scattered among the internet. Furthermore, although the project seldom mentions about logging report, setting up bash scripts that automates log report broadcasting plays an important role in monitoring the whole ETL process.

Overall, Snowflake is an attractive proposition as a Cloud Data Warehousing solution for case company. It has provided some distinct advantages over legacy technologies as outlined above. More specifically, Snowflake has proven to be a prospective platform for starting a production-ready data warehouse from day one with almost zero extra overhead for configuring the platform itself. Moreover, thanks to Snowflake native support for processing JSON (a de factor for the most internet-based APIs out there), the service itself is easy to integrate to different SaaS services such as Google Analytics and Facebook Business Manager. However, for creating and maintaining a larger cloud data warehousing solution, a visual ETL tool would bring back a great deal of advantages such as an overview of the whole process and bug reporting. Since Snowflake is a cloud database, it does not offer an ETL tool.

Reference

Adams, K., 2018. *key2consulting.* [Online]
Available at: https://key2consulting.com/concepts-of-an-enterprise-data-warehouse-edw-what-is-an-edw/
[Accessed 17 January 2019].

Anon., 2018. *System Properties Comparison Amazon Redshift vs. Microsoft Azure SQL Data Warehouse.* [Online]
Available at: https://db-engines.com/en/system/Amazon+Redshift%3BMicrosoft+Azure+SQL+Data+Warehouse
[Accessed November 2018].

Antonopoulos, N. and Gillam, L., 2017. *Cloud Computing, Principles, Systems and Applications.* 2nd Edition ed. UK: Springer International Publishing.

Bassil, Y., 2012. A Data Warehouse Design for A Typical University Information System. *Journal of Computer Science & Research (JCSCR) - ISSN 2227-328X,* I(A Data Warehouse Design for A Typical University Information System), p. 14.

Bidgoli, H., 2005. *The Internet Encyclopedia.* New Jersey: John Wiley & Sons, Inc.

Bourg, D. M. and Seemann, G., 2004. *AI for Game Developers.* Sebastopol(California): O'Reilly Media.

Busitelce, 2015. *busitelce.* [Online]
Available at: http://www.busitelce.com/data-warehousing/13-etl-process-in-data-warehouse
[Accessed 21 February 2019].

Chapple, M., 2018. *Facts vs Dimensions.* [Online]
Available at: https://www.lifewire.com/facts-vs-dimensions-1019646
[Accessed December 2018].

Coté, C., Gutzait, M. and Ciaburro, G., 2018. *Hands-on Data warehousing with Azure Data Factory.* Birmingham: Packt Publishing.

Barrett, D.J. and Silverman, R.E., 2001. *SSH, The Secure Shell: The Definitive Guide.* 1st Edition ed. Massachusetts: O'Reilly.

Golfarelli, M. and Rizzi, S., 2009. *Data Warehouse Design: Modern Principles and Methodologie.* Bologna: The McGraw-Hill companies, Srl-Publishing Group Italia.

Gour, V, Sarangdevot, S.S., Tanwar, G.S. and Sharma, A., 2010. Improve Performance of Extract, Transform and Load (ETL) in Data Warehouse. *Internation Journal on Comptuter Science and Engineering,* Volume 02, p. 786.

Haertzen, D., 2012. *The Analytical Puzzle, Profitable Data warehousing, Business Intelligence and Analytics.* New Jersey: Technics Publications, LLC.

Hoppe, G., 2016. *The Top 14 Business Intelligence LinkedIn and Facebook Groups.* [Online]
Available at: https://blog.capterra.com/the-top-14-business-intelligence-linkedin-and-facebook-groups/
[Accessed November 2018].

Humphries, M.W., Hawkins, M.W. and Dy, M.C. 1999. *Data warehousing: Architecture and Implementation.* New Jersey: Prentice Hall PTR.

Google Inc. 2018. *bigquery.* [Online]
Available at: https://cloud.google.com/bigquery/
[Accessed November 2018].

Facebook Inc. 2019. *About the structure of Facebook ads.* [Online]
Available at:
https://www.facebook.com/business/help/706063442820839?helpref=page_content
[Accessed Febrary 2019].

Inmon,W.H., 2002. *Building the Data Warehouse.* 3rd Edition ed. New York: Wiley Computer Publishing.

Inmon, W. H. and Hackathorn, R. D., 1994. *Using the Data Warehouse.* 1st Edition ed. New york: Wiley.

Kauffman, R., 2008. Data warehouses and GIS. In: S. Shekhar and H. Xiong, eds. *Encyclopedia of GIS.* New York: SpringerSicence+ Business Media, p. 221.

Kraynak, J., 2017. *Cloud Data Warehousing.* New Jersey: John Wiley & Sons, Inc.

Kurunji, S. et al., 2014. Optimizing Aggregate Query Processing in Cloud Data Warehouses. In: A. Hameurlain, T. K. Dang and F. Morvan, eds. *Data Management in Cloud, Grid and P2P Systems.* Switzerland: Springer International Publishing, pp. 1-2

Lane, P., 2005. *Oracle Database Data Warehousing Guide.* 2nd Edition ed. Redwood: Oracle Corporation.

Nath, S., Stackowiak, R. and Romano, C., 2017. *Architecting the industrial internet.* Birmingham: Packt Publishing.

Nixon, M., 2018. *Breaking Free From Complexity Saves Time And Money.* [Online]
Available at: https://www.snowflake.com/blog/breaking-free-from-complexity-saves-time-and-money/
[Accessed December 2018].

Ponniah, P., 2002. The Compelling Need for Data Warehousing. In: *Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals.* New Jersey: John Wiley and Sons, Inc., pp. 5-6.

Ricardo, C. M. and Urban, S. D., 2017. *Databases Illuminated.* Massachusetts: Jones and Barlett Learning.

Rouse, M., 2018. *data warehouse.* [Online]
Available at: https://searchdatamanagement.techtarget.com/definition/data-warehouse
[Accessed 10 Decemeber 2018].

SIlvers, F., 2008. *Building and Maintaining a Data Warehouse.* 1st Edition ed. Florida: CRC Press.

Snowflake, 2017. *Key Concepts & Architecture.* [Online]
Available at: https://docs.snowflake.net/manuals/user-guide/intro-key-concepts.html
[Accessed December 2018].

Snowflake, 2017. *snowflake.* [Online]
Available at: https://docs.snowflake.net/manuals/user-guide/data-lifecycle.html
[Accessed December 2018].

Standen, J., 2008. *Dimensional Tables and Fact Tables.* [Online]
Available at: http://www.datamartist.com/dimensional-tables-and-fact-tables
[Accessed December 2018].

Stopford, B., 2009. *Analysis and Opinion.* [Online]
Available at: http://www.benstopford.com/2009/11/24/understanding-the-shared-nothing-architecture/
[Accessed December 2018].

Thelwel, R., 2015. *What is Amazon Redshift? AWS's "fastest growing service" unboxed.* [Online]
Available at: https://www.matillion.com/blog/redshift/boost-amazon-redshift-performance-with-these-schema-design-best-practices/
[Accessed December 2018].

Themistocleous, M. and Morabito, V. (., 2017. *Information Systems.* Springer International Publishing AG ed. Switzerland: Springer International Publishing.

Guru99, n.d.. ETL Testing or Data Warehouse Testing Tutorial [Online]
Available at: https://www.guru99.com/utlimate-guide-etl-datawarehouse-testing.html
[Accessed 23 October 2018].

Wainstein, L., 2018. *techburst.io.* [Online]
Available at: https://techburst.io/data-warehouse-architecture-an-overview-2b89287b6071
[Accessed 20 February 2019].

Vassiliadis, P. and Simitsis, A., 2009. Near Real TIme ETL. In: i. S. Kozielski and R. Wrembel, eds. *New Trends in Data Warehousing and Data Analysis.* New york: Springer, p. 21.

Watts, S., 2017. *SaaS vs PaaS vs IaaS: What's The Difference and How To Choose.* [Online]
Available at: https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/
[Accessed November 2018].

Zhu, W-D., Gupta, M., Kumar, V., Perepa, S., Sathi, A and Statchuk, C., 2014. *Building Big Data And Analytics Solutions in the Cloud.* New York: International Business Machine Corporation.

Snowflake configuration bash file

```
[connections.Xtestdb]

 accountname = ***
 username = ***
 password = ***
 #dbname = X_TESTBED_DB
 schemaname = PUBLIC
 warehousename = X_MANUAL_WH
 rolename = SYSADMIN
 #Can be used in SnowSql as #connect example
 [connections.Xanalytics]

 accountname = ***
 username = ***
 password = ***
 #dbname = X_ANALYTICS_DB
 schemaname = PUBLIC
 warehousename = X_MANUAL_WH
 rolename = SYSADMIN
 [variables]
 #Loads these variables on startup
 #Can be used in SnowSql as select $example_variable

 example_variable=27

 [options]
 # If set to false auto-completion will not occur interactive
mode.
 auto_completion = True

 # main log file location. The file includes the log from
SnowSQL main
 # executable.
 log_file = ~/.snowsql/log

 # Timing of sql statments and table rendering.
 timing = True

 # Table format. Possible values: psql, plain, simple, grid,
fancy_grid, pipe,
 # orgtbl, rst, mediawiki, html, latex, latex_booktabs, tsv.
```