Enterprise Data Warehousing with Google’s BigQuery, a Cloud Implementation for Analytical Database

M P Darshana Sampath Dahanayake
FGS/MDA/2021/030
University of Kelaniya, Sri lanka
dsdahanayake@yahoo.com

***Abstract*— In today’s world enterprises are moving towards more and more data driven approaches to achieve their business goals. Enterprises data strategy is one of the key components of modern enterprises to thrive their business at a fast phase and Data Warehouse will be the definite thing for their analytical appetite. Conventionally Data warehouses has been used by the data analyst to extract data as a part of their day to day tasks. However, it has been significantly used to create real time dash boards, provide decision making guide lines, predictive analyzing and to make ad-hoc queries. Due to the nature of these dynamic data analytics requirements businesses are moving toward cloud-based data warehouses such as Google BigQuery to leverage data driven capabilities rather than worrying about cost and resource management.**

Keywords*— Google Cloud Platform, BigQuery, Data Warehouse, Enterprise Data Warehouse, ETL, Star Schema

I. Introduction

Modern dynamic business context and availability of millions of bytes of data in milli seconds of time elapses, has provided businesses unimaginable opportunities as well as put them in a tight spot; where they are unable to manage those vast number data with a traditional and conventional database system.

In order to address such conditions, Google-one of the world’s biggest silicon valleys has provided serverless, highly scalable, low cost enterprise data ware house available in its own cloud platform. It is named with BigQuery and capable of managing petabytes and zettabyte of data in fraction time as its name promises. Initially Google used these technologies to manage their own search engine data and later realized the necessity of market their technologies as a product to other companies. Google cloud platform contains dozens of tools and services to serve different business requirements.

BigQuery supports tried-and-true data technologies that are still relevant today. For a example BigQuery is a serverless compute architecture that combined with compute and storage. This enables different layers of architecture to perform and scale independently. Also, it enables data engineers and analyst to have the flexibility in design and deployment of the data warehouse. In addition to that BigQuery supports native machine learning and geospatial analysis. This is one of the unique features available among other cloud solutions. We could leverage BigQuery capabilities with other Google cloud platform services like Cloud Pub/Sub, Cloud Dataflow, Cloud Storage, Cloud Bigtable, Cloud AI Platform and other third-party integrations. BigQuery supports legacy and standard SQL. It proves BigQuery is not limited to modern Data centered Warehouses, it could also manage traditional databases as well.[1]

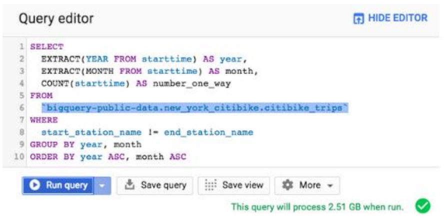
II. Why BigQuery

Let’s discuss some of the key features and benefits, why BigQuery must be use in an enterprise business.

1. Service for interactive analysis for massive datasets.

   BigQuery capable of querying millions of records. Seconds to read and seconds to return. How does it do that? Because OLTP databases are not a good fit for a ad-hoc queries that requires traversal through the entire dataset. Analysis such require excessive traversal need to program in high level language like Java and Python. In 2003 two Google developer were implemented a method to computations to process large amount of raw data. A map function that processed a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merged all intermediate values associated with the same intermediate key.[14] This MapReduce method one of the key advancements in BigQuery. BigQuery uses SQL command to execute the queries which, break the barriers of having a high-level programing knowledge among the data analysts and data engineers. It’s a Serverless and distributed SQL engine. Imagine, If you are to run such query where it requires traversal through entire dataset like MapReduce in an on-prem server and managing a sophisticated infrastructure.

   That is where BigQuery does the magic by being serverless. Without maintaining an expensive hardware infrastructure and maintenance staff. BigQuery actually focuses business on their analytical data strategy. Simple SQL query like following could be easily executed through its console.



```
SELECT
  EXTRACT(YEAR FROM starttime) AS year,
  EXTRACT(MONTH FROM starttime) AS month,
  COUNT(starttime) AS number_one_way
FROM
  `bigquery-public-data.new_york_citibike.citibike_trips`
WHERE
  start_station_name != end_station_name
GROUP BY year, month
ORDER BY year ASC, month ASC
```

This query will process 2.51 GB when run.

Fig 1: Query editor in BigQuery

2. Data security and governance

What is data governance? It is the process enterprises do to ensure their data is secure, private, accurate, available and usable inside BigQuery. Usually governance starts during the data onboarding. First to understand the organizations data, figuring out the data and where they coming from and what form of an information it contains. For an example let's consider transactional data. Secondly, they categorized the data with column marking to specify the sensitive data. In order to achieve this BigQuery has integration with data catalogue, google data discovery and metadata management services. Hence data catalogue automatically tracks metadata and label sensitive columns. After this process it is to configure the access policies. In BigQuery we have IAM or Identity Access Management it will take care of the policy implementation. When it comes to data sharing, you could create separate user groups and provide the access to particular department of users. Example you digital marketing data to marketing department team. Let's say digital marketing team wants to access another dataset which belongs to product development team. Organization could create Authorized views to access product development data for digital marketing analysts with limited view. Further admins could limit the view for record level whether digital marketing could view specific records of data belongs to specific product only. Lastly, BigQuery has ongoing monitoring mechanism to check the data quality. For an example admins can write custom assertions using SQL to check the duplicate entries and automated them whenever new data will be loaded to BigQuery tables.[3]

3. Availability and durability

There are few failure domains that could occur in the Google cloud datacenters and it has categorized as following:

Machine-level: Hardware failures can occur in a single or few machine, but not all within a Google Cloud Zone.

Zonal: Failures that can happen to single Google cloud zone where it will be unavailable due to power failure, building fire, cut off fiber optic cable in a same geographical location.

Regional: Failures can affect entire Google Cloud region that made of multiple zones. Due to natural disaster like earthquake and hurricanes.

When creating a BigQuery dataset it asks to select location which that dataset should store. This location is either a region like (Mumbai-southasia1) or Montreal (northamerica-northeast1), or a multi-region which is a large geographical area.

However, BigQuery will automatically create a copy of your dataset in two different zones in the same selected location.

Situations where, machine-level failure happens, BigQuery will continue running not more than few milliseconds delay.

Events where zonal failure happen, no data losses will be there, but currently running queries may fail and will have to resubmit. If it's a power failure or destroy transformer situation system will resume its running automatically within few minutes.

In the event of a hard-regional failure like natural disaster destroy entire region, then it could cause data losses since BigQuery automatically will not take backup or copy data set to a different geographical region. Enterprises such spread across multi continentals could create cross-regional dataset copy to enhance their disaster recovery strategy.[4]

III. Use Case

For this review I have used an Instacart public dataset which was initially made it public via Kaggle website (https://www.kaggle.com) which is a popular data science resource site.[5]

Instacart is an American company that operates a grocery services in the United Stated and Canada. Company founded in 2021 and headquarters located in San Francisco, California US which made revenue USD 1.5 billion in 2020. It operates over 10,000 employees across 5,500 cities in United States and Canada.[6]

Instacart has released this dataset for data science competition in 2017. Goal is to predict which products will be in users next order. The dataset contains main 4 CSV files which namely aisles, departments, orders and product tables. This dataset is anonymized and contains over 3.4 million grocery orders from more than 200,000 Instacart users. For each user provide between 4 and 100 of their orders with the sequence of products purchased in each order. Also order data table provided the columns with week and hours of day the order was placed and relative measure of the time between orders.[5]

IV. Data Warehouse Design

Designing a data warehouse could be tedious and challenging task. But yet it is a critical task for data engineers. This could accommodate fundamentals of data warehousing concepts such as anticipate data and ad-hoc queries. There are two great schemas out there to make thing easier. Star schemas and the snowflake schemas are the most popular and well adopted two schemas.

For this implementation of Instacart data warehouse in Google BigQuery star schema has been selected to build the data warehouse.

Star schema and Kimball's Bottom-Up-Approach

As Instacart dataset show denormalized table structure, Star schema can be easily incorporated with in the BigQuery. Even though BigQuery does not support primary key and foreign key constrains star schema can be implement with table joints. Star schema is nothing but once fact table and all the dimensions tables connected via fact table.

Ralph Kimball initiated dimension model or bottom-up for data warehouse architecture where data marts are created based on the denormalized data from specific business processes. For an example Marketing, Human resources, Finance can create data marts for each and then link to up to form the organizational data. In this case most of the dimension are slowly changing dimensions.[7]

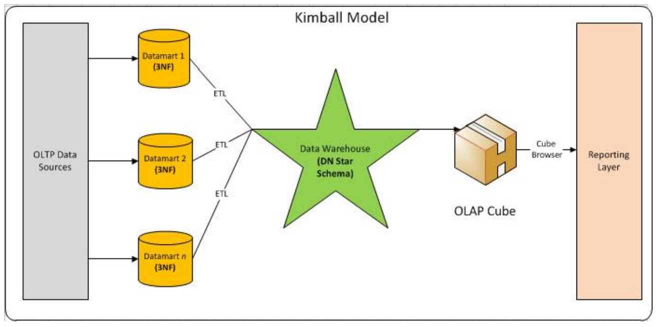


Fig 2: Kimball's Bottom-Up Approach

After reviewing the Instacart dataset following Dimenstion tables were identify to implement in the Data warehouse model

Dimension Name	Description
Dim_aisle	This represent the data of each product categories of lanes, where products are stored in Instacart physical market/warehouse.
Dim_departments	Information related to the department which products are belongs to
Dim_products	Information of products includes product name, aisle and department details related each product.
Dim_date	Information about timeline such as year, month, date, day of the week, week of the month, week number of the year, etc. to keep the granularity level of the date and time.
Dim_orders	Information that relates to orders, which includes user, product, purchase hour and date details.

Table 1: Dimensions Table for Instacart dataset

After defining the fact tables, key step is to determine the Fact table for the selected dimension table. This involves with identifying the grains best fit for Fact table. Surrogate key are introduce to link the Fact table and dimensions in our data warehouse design.

Fact Name	Description
Fact_sales	This represent the data of each order and surrogate keys

Table 2: Attributes of Dimensions

Following surrogate keys are introduced to link the dimension tables to Fact table and those are namely order_SK, product_SK, department_SK, aisle_SK.

Following illustrate the table schema of the dimension tables and the fact table.

Dimension Name	Attributes
Dim_aisle	aisle_SK, aisle_id, aisle
Dim_departments	department_SK, department_id, department
Dim_products	product_SK, product_id, product_name, aisle_id, department_id
Dim_date	DateNum, Date, YearMonthNum, Calendar_Quarter, MonthNum, MonthName, MonthShortName, WeekNum, DayNumOfYear, DayNumOfMonth, DayNumOfWeek, DayName, DayShortName, Quarter, YearQuarterNum, DayNumOfQuarter
Dim_orders	order_SK, order_id, user_id, eval_set, order_number, order_dow, order_hour_of_day, days_since_prior_order, product_id, purchase_date

Fact Name	Attributes
Fact_sales	order_SK, product_SK, department_SK, aisle_SK, order_id, user_id, product_name, department, aisle, order_hour_of_day, Date

Table 3: Attributes of Dimensions and Fact tables

Table diagram of the data ware house

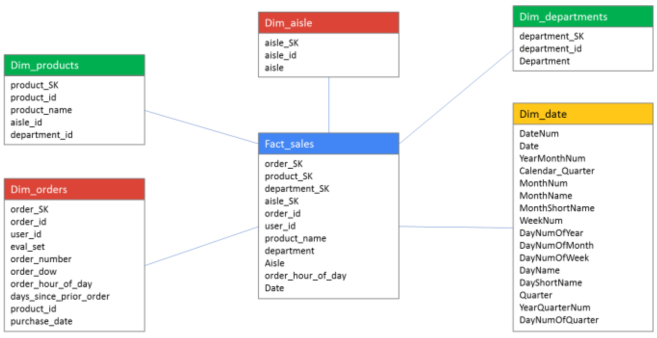
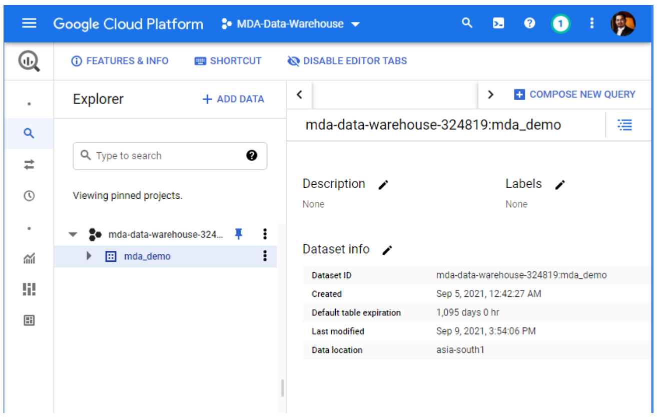


Fig 3: Table diagram of the Instacart data warehouse

V. Table Creation in BigQuery

Since we discussed the data warehouse design schemas, design and model, its time to discuss the technical implementation of the tables, schemas and constrains in BigQuery.

BigQuery provide number of options to create tables and schemas. Usual cloud environment uses project scope to define its product and services. Hence first step to define the project for our data warehouse. After creating the project next step will be creating a dataset. Dataset will be the more abstract or root node of the data warehouse. All the component including tables, queries, aggregated tables and other relevance meta data will be stored under dataset.



mda-data-warehouse-324819:mda_demo

Description
None

Labels
None

Dataset info

Dataset ID	mda-data-warehouse-324819:mda_demo
Created	Sep 5, 2021, 12:42:27 AM
Default table expiration	1,095 days 0 hr
Last modified	Sep 9, 2021, 3:54:06 PM
Data location	asia-south1

Fig 4: Dataset or Data mart window in BigQuery

One important point to consider while creating the dataset is its Data location. Google have several regional locations and based on the organizations to scale and multinational or multi continental scope need to consider. Specially need to consider the supporting disaster recovery strategy of the organization. Also need to consider the cloud storage region or location of selected when creating the storage bucket.

Table creation of the BigQuery also provide multiple convenient options for both none technical and technical staff. This provided enterprise the flexibility of use their own staff to create to data warehouse in the cloud.

Following illustrate the flexible option provide by the BigQuery:

1. Create table by using file upload:
   Using cloud storage or local disk users could easily upload the files which supported CSV, JSON AVRO, PARQUET formats. Any file greater than 100MB need to be uploaded to cloud storage and upload it from the respective storage bucket.

   Table schema will be automatically identified by the BigQuery and assigned suitable data types for the table data.

2. User can create empty table via cloud console GUI selecting the create empty table option

3. Using standard SQL query:
   Users can execute standard SQL commands in BigQuery query editor.

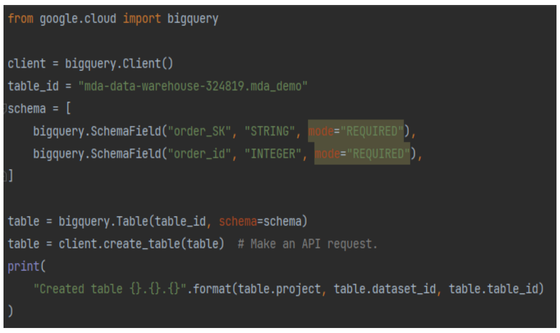
```
“CREATE TABLE `mda-data-warehouse-
324819.mda_demo.Dim_departments` (
    `department_SK` INT64,
    `department_id` INT64,
    `department` STRING
)”
```

4. Using JSON:
   User can use standard JSON code to create the table via cloud console

```
[
  {
    "description": "department_SK",
    "mode": "REQUIRED",
    "name": "department_SK",
    "type": "INT64"
  },
  {
    "description": "department_id",
    "mode": "NULLABLE",
    "name": "department_id",
    "type": "INT64"
  },
  {
    "description": "department",
    "mode": "NULLABLE",
    "name": "department",
    "type": "STRING"
  }
]
```

5. Using BigQuery API users can create table using high-level language like JAVA, C#, Python, Node.js, PHP, Ruby, etc.

   For this review Python have used to demonstrate the sample API call to BigQuery



```
from google.cloud import bigquery

client = bigquery.Client()
table_id = "mda-data-warehouse-324819.mda_demo"
schema = [
    bigquery.SchemaField("order_SK", "STRING", mode="REQUIRED"),
    bigquery.SchemaField("order_id", "INTEGER", mode="REQUIRED"),
]

table = bigquery.Table(table_id, schema=schema)
table = client.create_table(table)  # Make an API request.
print(
    "Created table {}.{}.{}".format(table.project, table.dataset_id, table.table_id)
)
```

Fig 5: Python API call to BigQuery

VI. Extract-Transform-Load Process

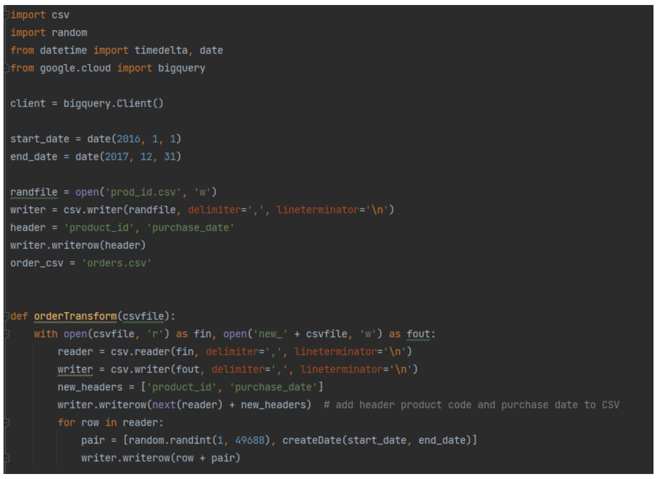
Extract-Transform-Load (ETL) process is one of the key processors when considering the Data warehouse. It collects data from various different data sources and transform the data to be fit for the dimension tables with respective data formats. Then it loads data respective dimension tables for analysts to run they analytical queries with most updated data.

BigQuery provided more flexible and efficient method to accommodate ETL processors more efficient, effective and user-friendly manner.

This review I have developed own ETL tools with google BigQuery API. It provided more flexibility for developers to build their own ELT tools without depend on the 3rd party vendors.

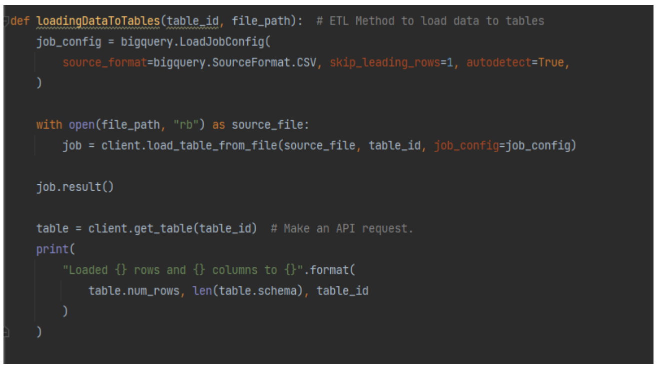
Also, Python provided tones of packages to work with file operations with great flexibility. Packages like numpy, panda, csv great assets for python programmers to make thing much easier.

1. Load data via API using python:



```python
import csv
import random
from datetime import timedelta, date
from google.cloud import bigquery

client = bigquery.Client()

start_date = date(2016, 1, 1)
end_date = date(2017, 12, 31)

randfile = open('prod_id.csv', 'w')
writer = csv.writer(randfile, delimiter=',', lineterminator='\n')
header = 'product_id', 'purchase_date'
writer.writerow(header)
order_csv = 'orders.csv'


def orderTransform(csvfile):
    with open(csvfile, 'r') as fin, open('new_' + csvfile, 'w') as fout:
        reader = csv.reader(fin, delimiter=',', lineterminator='\n')
        writer = csv.writer(fout, delimiter=',', lineterminator='\n')
        new_headers = ['product_id', 'purchase_date']
        writer.writerow(next(reader) + new_headers)  # add header product code and purchase date to CSV
        for row in reader:
            pair = [random.randint(1, 49688), createDate(start_date, end_date)]
            writer.writerow(row + pair)
```

Fig 6: Transform data with python

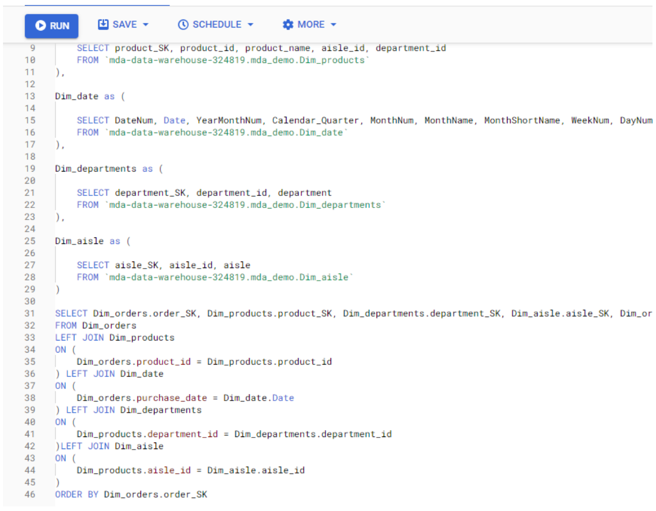


```python
def loadingDataToTables(table_id, file_path):  # ETL Method to load data to tables
    job_config = bigquery.LoadJobConfig(
        source_format=bigquery.SourceFormat.CSV, skip_leading_rows=1, autodetect=True,
    )

    with open(file_path, "rb") as source_file:
        job = client.load_table_from_file(source_file, table_id, job_config=job_config)

    job.result()

    table = client.get_table(table_id)  # Make an API request.
    print(
        "Loaded {} rows and {} columns to {}".format(
            table.num_rows, len(table.schema), table_id
        )
    )
```

Fig 7: Transform data with python

2. Batch loading:
   With batch loading user can load data to BigQuery tables in a single batch operation. Load the data from cloud storage or local files like CSV or other supported files by creating 'load job' option in BigQuery. Then user can use BigQuery Data Transfer Service to automate the data loading.[8]

3. Also, user can schedule a standard SQL query to fetch the data from dimensions tables and overwrite the fact table regular manner.

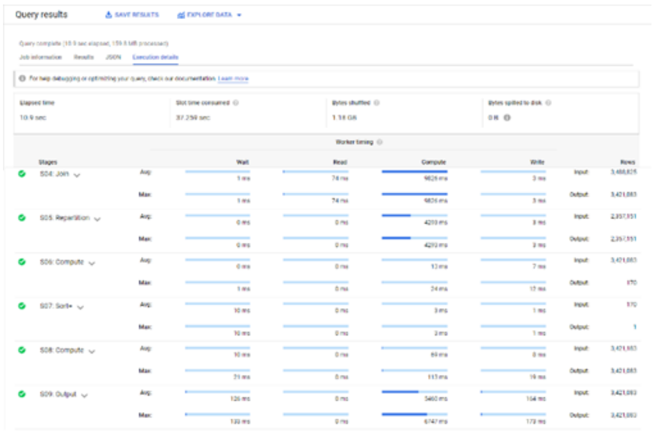


```sql
    SELECT product_SK, product_id, product_name, aisle_id, department_id
    FROM `mda-data-warehouse-324819.mda_demo.Dim_products`
),

Dim_date as (

    SELECT DateNum, Date, YearMonthNum, Calendar_Quarter, MonthNum, MonthName, MonthShortName, WeekNum, DayNum
    FROM `mda-data-warehouse-324819.mda_demo.Dim_date`
),

Dim_departments as (

    SELECT department_SK, department_id, department
    FROM `mda-data-warehouse-324819.mda_demo.Dim_departments`
),

Dim_aisle as (

    SELECT aisle_SK, aisle_id, aisle
    FROM `mda-data-warehouse-324819.mda_demo.Dim_aisle`
)

SELECT Dim_orders.order_SK, Dim_products.product_SK, Dim_departments.department_SK, Dim_aisle.aisle_SK, Dim_or
FROM Dim_orders
LEFT JOIN Dim_products
ON (
    Dim_orders.product_id = Dim_products.product_id
) LEFT JOIN Dim_date
ON (
    Dim_orders.purchase_date = Dim_date.Date
) LEFT JOIN Dim_departments
ON (
    Dim_products.department_id = Dim_departments.department_id
)LEFT JOIN Dim_aisle
ON (
    Dim_products.aisle_id = Dim_aisle.aisle_id
)
ORDER BY Dim_orders.order_SK
```

Fig 8: SQL query to fetch the data to tables

VII. Analysis

BigQuery's biggest strength is to run petabytes of data withing seconds time. This enables analyst to run large dataset queries and diagnose data with sort time and rerun the queries again with necessary adjustments. user can run various queries with number of joint tables and joint tables are greater concern in other conventional data warehouses. It will save money and time for the enterprises. Then enterprises can leverage that time and money for more business impact areas like new product design and marketing campaigns.



Query complete ([illegible] sec elapsed, [illegible] processed)

For help debugging or optimizing your query, check our documentation. Learn more

Elapsed time	Slot time consumed	Bytes shuffled	Bytes spilled to disk
10.9 sec	37.259 sec	1.18 GB	0 B

Worker timing

Stages		Wait	Read	Compute	Write		Rows
S04: Join	Avg:	1 ms	74 ms	9826 ms	3 ms	Input:	3,488,825
	Max:	1 ms	74 ms	9826 ms	3 ms	Output:	3,421,083
S05: Repartition	Avg:	0 ms	0 ms	4293 ms	3 ms	Input:	2,357,551
	Max:	0 ms	0 ms	4293 ms	3 ms	Output:	2,357,551
S06: Compute	Avg:	0 ms	0 ms	13 ms	7 ms	Input:	3,421,083
	Max:	1 ms	0 ms	24 ms	17 ms	Output:	170
S07: Sort+	Avg:	10 ms	0 ms	3 ms	1 ms	Input:	170
	Max:	10 ms	0 ms	3 ms	1 ms	Output:	1
S08: Compute	Avg:	10 ms	0 ms	69 ms	8 ms	Input:	3,421,083
	Max:	71 ms	0 ms	113 ms	19 ms	Output:	3,421,083
S09: Output	Avg:	126 ms	0 ms	5460 ms	164 ms	Input:	3,421,083
	Max:	133 ms	0 ms	6747 ms	173 ms	Output:	3,421,083

Fig 9: Query performance data of BigQuery

Above illustrate the query execution time for analysis report of the Instacart dataset. It shows that the BigQuery has traversal through entire dataset in order to project the expected output. In googles powerful computation and distributed processors enables BigQuery to reversal 3,421,083 records within 10.9 seconds.

Also, BigQuery comes with its own data visualization tool Google Data Studio which provide comprehensive and great features to Data visualization analysts and no need to go for Power BI or Tableau like 3rd party vendor tools and it is freely available without any additional cost. Dashboards could be share via web and no need for additional hardware to setup the Google Data Studio.

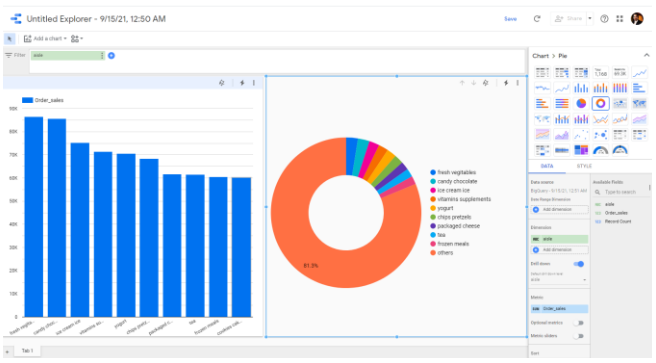


Fig 10: Google Data Studio User Interface

Following illustrate few analyses done with Instacart data set and data visualizations with Google data studio.

1. Order Sales during the hours of the day: Sales pick up from 9.00 am till 4.00 pm daily. More staff and resource could be allocated during that period to further increase the sales and reduce the staff over head during off peak time.

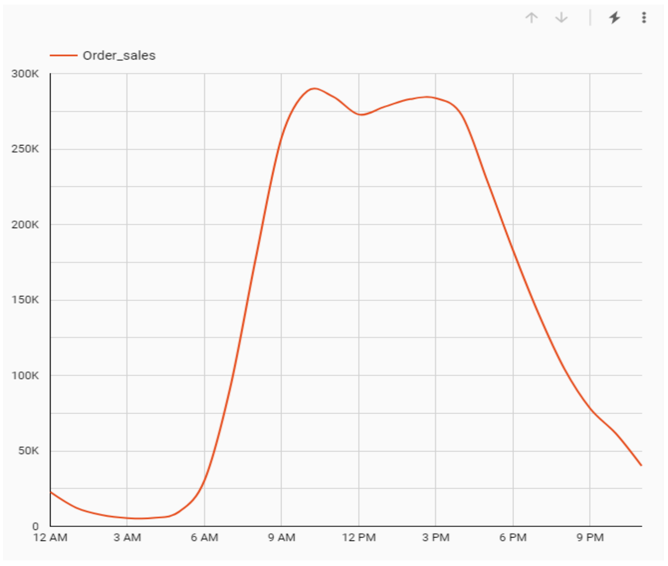


Fig 11: Chart 1 Order vs hour of the day

2. Sales Vs aisle – the greatest number of sales happen on which aisle. Rearrange the aisles lanes according to highest sales aisles together in order to save time while staff selecting products for particular customer order.

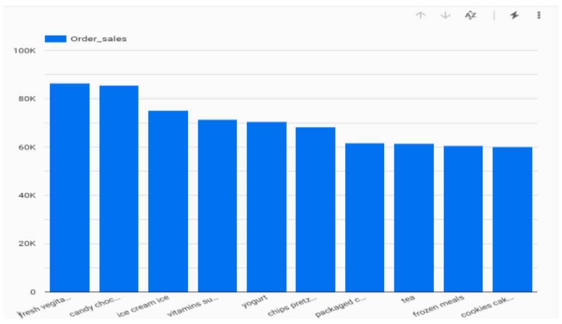


Fig 12: Chart 2 Sales Vs Aisle

VIII. Pros and Cons

Pros:

- Multiple supported format and options for Table creation
- Cloud storage / buckets
- Partitioning and clustering
- Easy data ingesting and ETL options
- API integration
- Automated load jobs/ Batch jobs
- Any Table modification can accommodate while running the queries
- No cost for caching queries
- Inbuilt Google Data Studio for data visualization

Cons:

- No Primary Key, Foreign Key constrains
- Configuration
- Limited data types
  - Only few datatypes and no auto incremental
- Strict data governance

IX. Challenges

Primary key and foreign key constrain not supported in BigQuery was challenging when migrating form, a traditional Data warehouse with relation database system. For a example Instacart dataset used for this review is did not have the proper relations between tables. If its new implementation for BigQuery enterprise will not face much difficulties onboarding a new data warehouse. But for databases does not have much details about their schemes and meta data may face some challenges.

Googles Data governance policies for zones and region may impact for large enterprises own data government policies. Those who are operate in multi continental geo location will apply its countries own data policies and face difficulties when implement googles data policies.

Some enterprises are still very reluctant to go for a cloud solution. Since they feel hosting their data in the cloud may loss organization trust, reputation and control over data among their customers and stakeholders.

Cloud is not a freebie, though it reduces the CapEx (capital expenditure) for enterprises, OpEx cost might go up significantly over the time. Google BigQuery charges pay-as-you-go model. Large corporates may get most out of such models and small enterprises with larger dataset may find it difficult to manage OpEx cost.

X. Conclusion and Future Works

Google's state of the art technology, super computational power, scalable storage facility, physical and online security protocols may give a competitive advantage over the other cloud and on-prem data warehouse solutions. Also, it provides greater analytical and machine learning tools under one roof.

Google cloud platform flexibility and agile nature gives enterprise to reduce their IT infrastructure cost and maintaining resource cost without imposing significant risk to organization data.

However, every product out there in the market has its own pros and cons. Overall BigQuery has build a great reputation in modern data warehouse context and it will further develop gradually with Google's research and development initiatives

Further we can look further improvements in BigQuery support in different datatypes as it currently supports few datatypes compared to other data warehouse solutions out there in the market. Also its real time streaming data capability will further improved.

References

[1] Valliappa Lakshmanan and Jordan Tigani, "Google BigQuery: The Definitive Guide" O'Reilly Media Inc, Oct 2019.

[2] Jeffrey Dean and Sanjay Ghemawat, MapReduce: Simplified Data Processing on Large Cluster, in Google Inc, Feb 2004.

[3] "Introduction to data security and governance," Jan. 08, 2020. Accessed on: Sep. 12, 2021. [Online]. Available:https://cloud.google.com/bigquery/docs/data-governance

[4] "Availability and durability," Jan. 08, 2020. Accessed on:Sep.12,2021.[Online]. Available: https://cloud.google.com/bigquery/docs/availability

[5] "Instacart Market Basket Analysis," Aug 15, 2017. Accessed on:Aug 28,2021.[Online]. Available: https://www.kaggle.com/c/instacart-market-basket-analysis/overview

[6] "Instacart" Jan 05, 2014. Accessed on:Aug 28,2021.[Online]. Available: https://en.wikipedia.org/wiki/Instacart

[7] Christopher Adamson, "Star Schema The Complete Reference" McGraw-Hill, Jul 2010.

[8] "Batch loading data" Jan. 08, 2020. Accessed on:Aug 28,2021.[Online]. Available: https://cloud.google.com/bigquery/docs/batch-loading-data