VIETNAM NATIONAL UNIVERSITY, HANOI

INTERNATIONAL SCHOOL

FINAL PROJECT

DATA WAREHOUSING & BUSINESS ANALYTICS FOR AMAZON'S SALE PERFORMANCE ANALYSIS

Lecturer: **Tran Thi Ngan**

Course Name: **Data Warehousing and Business Analytics**

Course ID: **INS3073_02**

Group: **4**

Hanoi, January 2025

CONTRIBUTION

Full Name	Student ID	Tasks	Contribution
Đào Thị Huế	21070007	- Introduction - Dataset Description - Business Questions - Querying on OLAP - Exploratory Data Analysis	25%
Phạm Hồng Ngọc	21070557	- Business Questions - ELT Architecture design - Dimensional Modeling - Querying on OLAP - Exploratory Data Analysis	25%
Nguyễn Thị Ngọc Giao	20070704	- Business Questions - Airflow DAGs (DAGs Code)	25%
Phạm Thị Xuân Quỳnh	21070388	- ERD Design - Airflow DAGs (SQL File Structure: Create ERD tables, DIMFACT tables) - Conclusion	25%

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

1. Problem Overview

In the modern business environment, data mining and analysis play an extremely important role in optimizing sales strategies and making accurate business decisions. However, with the large volume of data generated from many different sources every day, businesses will encounter many difficulties without an effective data management and analysis system. Therefore, the application of Data Warehouse to sales data management and analysis becomes a key factor in helping businesses grasp trends, optimize performance, and maintain competitive advantages.

Data Warehouse is a centralized system that allows businesses to store, organize, and analyze data from many different sources. Through the integration and management of data in a unified manner, Data Warehouse helps ensure data consistency and support the decision-making process. At the same time, businesses can analyze sales performance in different dimensions such as time, region, product, and customer... This not only helps identify market trends but also helps forecast product demand, plan future development

In addition, Data Warehouse also helps businesses evaluate market trends, as well as have an overview of sales, and purchasing trends, and identify potential customer groups. When having an in-depth view of sales data, businesses can make accurate decisions, thereby maximizing profits and improving operating performance. Applying Data Warehouse to sales data analysis will help businesses not only improve work efficiency but also develop sustainable business strategies.

2. Project Objectives

This project aims to analyze and optimize Amazon's sales performance. The project focuses on using data-driven methods to gain insights, improve strategic decision-making, and improve operational efficiency. Key objectives include:

- *Identify Seasonal Sales Trends:* Analyze monthly and quarterly sales data to identify the busiest sales periods, understand seasonal shopping behaviors, and uncover trends related to events such as holidays or promotional campaigns.
- *Evaluate the Effectiveness of Discount Levels by Product Category:* Assess the impact of various discount levels on the sales performance of different product categories to determine the most effective discount strategies. Use these insights to balance profitability with customer satisfaction by designing discount campaigns tailored to the needs of each category.
- *Forecast Ordering Trends:* Study historical order data to predict future order volumes and identify potential decreases or growth trends. Develop strategies to

address declining trends, such as diversifying products, enhancing customer retention efforts, and launching targeted promotional campaigns.

- *Segment Customers Based on Shopping Behavior:* Apply advanced clustering techniques (e.g., K-Means and PCA) to segment customers into distinct groups. Develop customized engagement strategies to retain VIP customers, activate potential buyers, and reactivate inactive customers, thereby maximizing customer lifetime value.
- *Optimize Sales Strategies and Resource Allocation:* Leverage data-driven analyses to implement strategies for managing discount programs, promotional campaigns, and customer interaction initiatives. Ensure efficient resource allocation by focusing on high-value customer segments.

II. UNDERSTANDING THE DATASET

1. Dataset Description

In this project, we use two main datasets from Kaggle to analyze Amazon's sales performance. Below is the description of each dataset:

Amazon Sale Report.csv: This dataset enables in-depth analysis of sales performance, customer behavior, and operational efficiency across different regions and product categories. It serves as a valuable resource for optimizing marketing strategies, inventory management, and overall profitability in the e-commerce sector.

Table 1. Amazon Sale Report dataset attributes and description.

Attribute	**Type**	**Description**
index	Integer	Index number for the dataset
Order ID	Integer	Unique identifier for each order
Date	Date	Date of the sale
Status	String	Status of the sale
Fulfilment	String	Fulfilment method
Sale Channel	String	Channel through which the sale was made
ship-service-level	String	Shipping service level used
Style	String	Style of the product

SKU	String	Stock Keeping Unit
Category	String	Category of the product
Size	String	Size of the product
ASIN	String	Amazon Standard Identification Number
Courier Status	String	Current status of the courier service
Qty	Integer	Quantity of the product sold
Currency	String	Currency in which the transaction occurred
Amount	Float	Total amount for the transaction
ship-city	String	City of the shipping address
ship-state	String	State of the shipping address
ship-postal	String	Postal code of the shipping address
ship-country	String	Country of the shipping address
promotion-ids	String	IDs of promotions applied to the transaction
B2B	Boolean	Indicates whether the sale is business-to-business
fulfilled-by	String	Entity responsible for order fulfillment

Amazon Sales FY2020-21.csv: Focuses on Amazon's sales performance in fiscal year 2020-21, and provides detailed data on sales, customer demographics, product details, and financial metrics. helps analyze sales performance over time and region. This helps us analyze and forecast product demand, optimize shipping strategies, and maximize profits.

Table 2. Amazon Sale FY2020-21 dataset attributes and description.

Attribute	**Type**	**Description**
order_id	Integer	Unique identifier for each order

order_date	Date	Date the order was placed
status	String	Current status of the order
item_id	Integer	Unique identifier for each item in the order
sku	String	Stock Keeping Unit, a unique identifier for each product
qty_ordered	Integer	Quantity of the item ordered
price	Float	Price per unit of the item ordered.
value	Float	The value of the item in the order (price * qty_ordered)
discount_amount	Float	Discount applied to the item
total	Float	Total amount for the order item (price * qty_ordered - discount_amount)
category	String	Category of the product
payment_method	String	Method of payment used for the transaction
bi_st	String	Business Intelligence status, potentially indicating the order's analysis status
cust_id	Integer	Unique identifier for each customer
year	Integer	The year the order was placed
month	Integer	The month the order was placed
ref_num	String	Reference number associated with the order.
Name Prefix	String	Prefix of the customer’s name
First Name	String	First name of the customer
Middle Initial	String	Middle initial of the customer
Last Name	String	Last name of the customer
Gender	String	Gender of the customer

age	Integer	Age of the customer
full_name	String	Full name of the customer (First Name + Last Name)
E Mail	String	Email address of the customer
Sign in date	Date	Date the customer signed into the platform
Phone No	String	Phone number of the customer.
Place Name	String	Name of the place associated with the customer
Country	String	Country where the order was shipped to
City	String	City of the shipping address
State	String	State or region of the shipping address
Zip	String	Zip code of the shipping address
Region	String	Region associated with the shipping address
User Name	String	Username associated with the customer's account
discount_percent	Float	Percentage discount applied to the order item

These datasets provide an in-depth look at customer shopping behavior, seasonal changes, and factors that influence profits in the e-commerce sector. At the same time, it provides an overview and details of the factors affecting Amazon's sales performance.

Combined insights from both datasets:

- Sales Trends: Analyze sales performance across time (daily, monthly, and yearly trends).
- Customer Insights: Study customer demographics, purchasing behavior, and loyalty.
- Product Analysis: Identify top-performing categories and items.
- Regional Analysis: Evaluate performance across cities and countries.

By integrating these datasets into a data warehouse, Amazon can derive meaningful insights into its operations, enabling strategic decision-making and optimized performance tracking.

2. Business Questions

- Which discount percentage is the most effective in boosting sales for each product category?
- What are the busiest sales months or quarters for the business?
- How will the ordering trend for next month?
- What groups can customers be segmented into based on shopping behavior?

III. DATA WAREHOUSE DESIGN

1. Entity Relationship Diagram Design

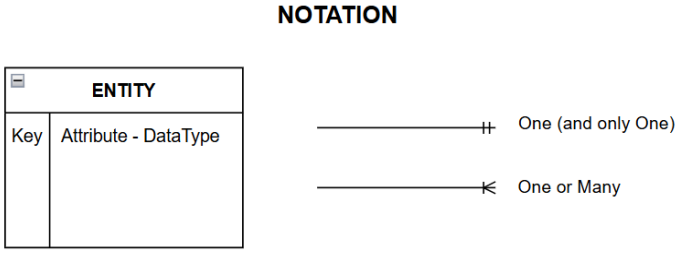


Figure 1: ERD Notation

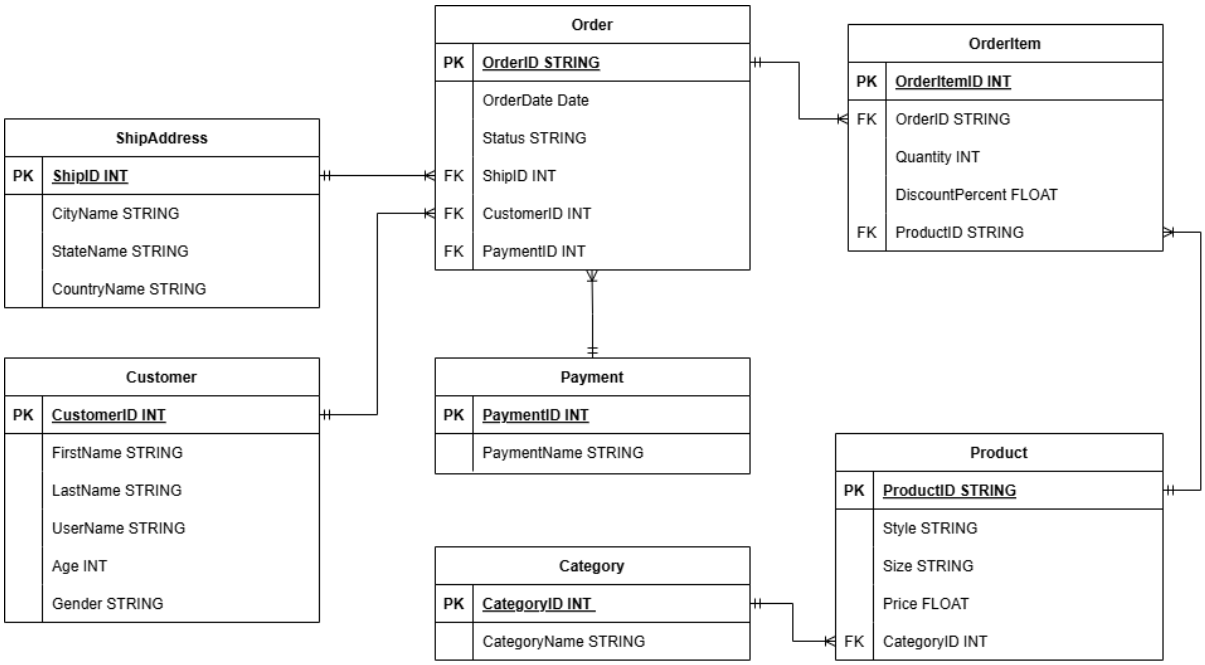


Figure 2: Entity-Relationship Diagram

The ERD (Entity-Relationship Diagram) depicts the integration of Amazon's sales data, modeling key entities and their relationships to streamline sales and operational analytics. This diagram was created based on two datasets: **Amazon Sales**

FY2020-21 and **Amazon Sales Report**. Below is a detailed description of the entities and their relationships:

The *Customer* entity stores customer-related information, identified by a unique *CustomerID*. This includes demographic details such as first name, last name, gender, and age as well as account information like username. The *Customer* entity is connected to the *Order* entity through a one-to-many relationship, as each customer can place multiple orders.

The *Payment* entity stores details of transactions associated with each order. It includes attributes such as *PaymentName*, facilitating insights into customer preferences for payment options. It is linked to the *Order* entity via a one-to-many relationship.

The *ShipAddress* entity contains attributes like shipping city, state, and country. Multiple orders can share the same shipping address, forming a many-to-one relationship with the *Order* entity. This setup supports geographical analysis of customer behavior and logistics efficiency.

The *Order* entity represents transactions made by customers, uniquely identified by *order_id*. It records metadata like the order date, order status, and reference numbers. This entity has a one-to-many relationship with *OrderItem*, where each order may consist of multiple items. It is also linked to *Payment* to record payment details and *ShipAddress* to document shipping details for each order.

The *OrderItem* entity details specific items included in an order. Attributes such as ProductID, quantity, price, discount percent,and OrderID. It serves as a bridge between the *Order* and *Product* entities, enabling detailed analysis of which products are sold in each order, along with their quantities, pricing, and discounts.

The *Product* entity represents individual items for sale, identified by ProductID. Attributes such as style, size, and product-specific details are included. Each product belongs to a specific category, forming a many-to-one relationship with the *Category* entity.

The *Category* entity groups products into higher-level classifications, identified by a unique CategoryID. This structure supports aggregated analysis, such as evaluating sales performance by category or identifying trends in specific product groups.

2. ELT Architecture

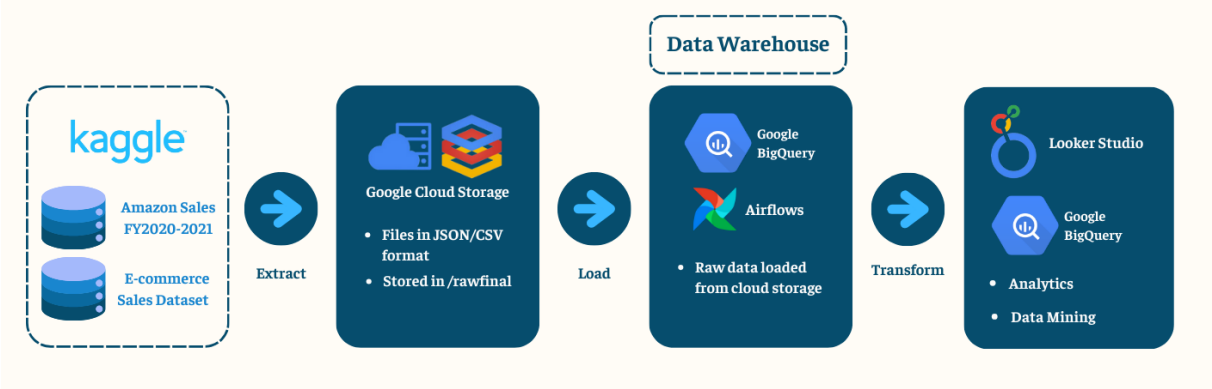


Figure 3: ELT Architecture

We design this ELT architecture to efficiently process, store and analyse large datasets for business insights. The architecture consists of 4 main processes:

- **Extract**: Data is sourced from Kaggle and provided in JSON/CSV format.
- **Load**: The raw data is uploaded and stored in Google Cloud Storage, ensuring all raw data is centralized and ready.
- **Transform**: Data transformation is handled with Google BigQuery, where data is processed, cleaned, and structured to meet business requirements. Automation of these tasks is managed by Apache Airflow, which orchestrates workflows and ensures data pipeline execution
- **Analyse** and **visualize**: Processed data in BigQuery is connected to visualization tools like Looker for analytics and reporting.

3. Dimensional modeling

Step 1: Choose the Business Process

Sales is a critical metric in business as it directly reflects the performance and revenue-generating capability of a company. Our group chose to analyze the sales process as it helps us gain deeper insights into best-selling products, customer purchasing trends, and the effectiveness of payment methods. Additionally, this analysis provides valuable information on revenue across time and geographic regions, enabling the identification of potential or underperforming areas. As a result, businesses can develop appropriate strategies, optimize profits, and enhance customer experiences.

Step 2: Declare the grain

The grain in our project is defined as a single row in the fact table, representing a specific product within an order (order item). Each row includes information about the product sold, the quantity sold, the transaction value, and the associated order through OrderID, and includes key information such as the product sold (along with its

category), the time of the transaction, the customer who made the purchase, the payment method used, and the customer's geographic location. Defining the grain at the transaction level ensures that the data is detailed and suitable for comprehensive analysis.

Step 3: Identify dimensions

Dimensional tables' purpose is to store descriptive attributes or textual context for the facts. It could provide a filter, group, label data, and contain hierarchies. We deploy our dimensional tables as follows:

- **dimProduct** - Contains information about products available on Amazon, including ProductID as primary key, CategoryID linked to category dimensional table to show which category the product belongs to, Style and Size are mentioned as product description.
- **dimCustomer** - Contains information about Amazon customers. While CustomerID records each specific customer, Gender, and Age reflect their demographics, which could be useful for further analysis. dimDate - Contain a detailed date of the order, assist deep analysis by dividing it into Year, Quarter, Month, Week, Day separately.
- **dimDate** - Contains information about time. DateID represents a unique identifier for each date. It enables filtering, grouping, and aggregating data by time periods such as days, months, quarters, or years by containing attributes FullDate, Year, Quarter, Month, Week, Day.
- **dimPayment** - Where stores data of customer payment by PaymentName, which payment method they chose, and the statistical data of transactions.
- Besides, we created **dimCountry**, **dimState**, **dimCity** to easily extract information about geographic aspects, based on orders made by customers on Amazon.

Step 4: Identify the facts

Fact table’s purpose is to store quantitative data including UnitSold, TotalAmount which contains measurable metrics, includes foreign keys linking to dimensional tables, support aggregation and calculations.

- **SaleOrderID**: A unique identifier for each product transaction
- **ProductID**: Identifies the specific product sold in the order item
- **Customer_ID**: A unique identifier for the customer associated with the transaction
- **OrderDateID**: Indicates the date when the transaction occurred
- **PaymentID**: Represent the payment method of customer
- **CityID**: Refers to the city where the customer is located.

- **OrderStatus**: Status of the order
- **TotalAmount**: Total amount of product that customer pay on their order
- **UnitSold**: Units of product sold
- **DiscountedPrice**: Price of product after applying promotions
- **DiscountedPercent**: Percent of promotions that applied for products

The figure below show our dimensional modeling:

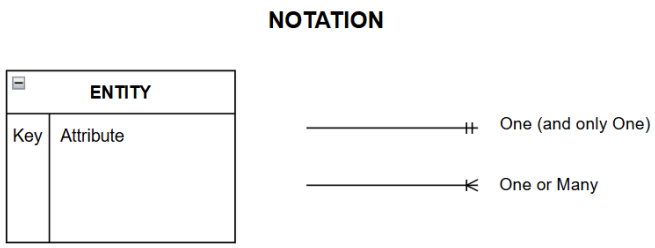


Figure 4: Dim-Fact Notation

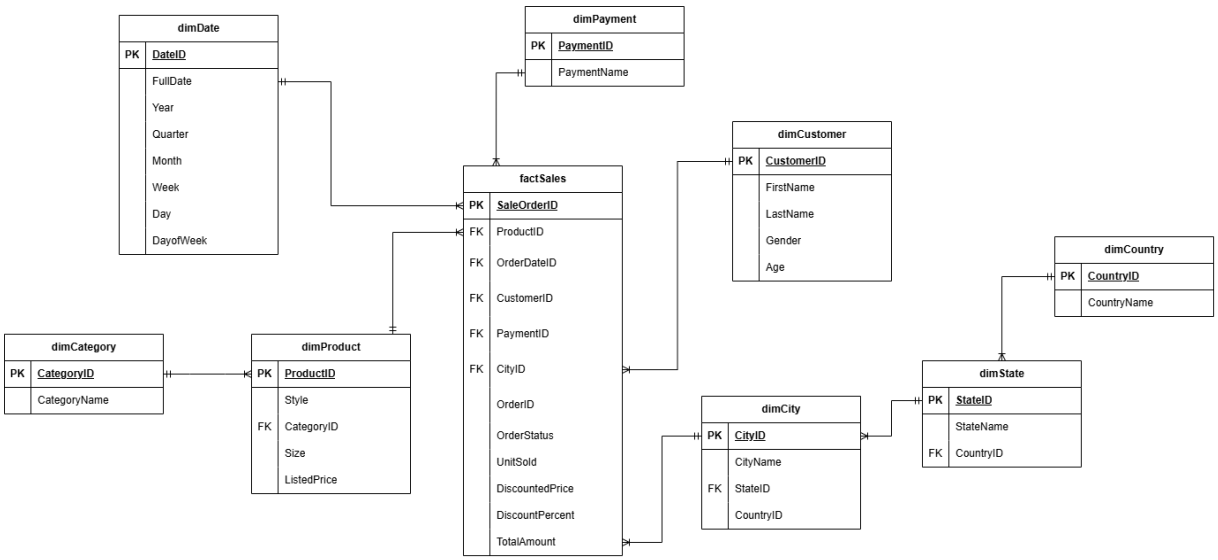


Figure 5: Snowflake schema

This dimensional model enhances query performance with its snowflake schema design, enabling faster aggregations and simplified reporting through clear fact-dimension relationships. It supports time-based analysis using **dimDat**e for trends and provides customer insights through **dimCustomer** for segmentation by demographics or location. Additionally, it facilitates product performance tracking across categories and regions, while geographical hierarchies in **dimCity** and **dimCountry** allow for regional trend analysis.

IV. AIRFLOW

1. Overview

The main purpose of using Airflow to stream data into Google BigQuery is to automate, schedule, manage, and monitor data workflows effectively. This approach guarantees that data is consistently updated and accurate, significantly enhancing the performance and efficiency of data analysis. Apache Airflow is a robust open-source tool designed for orchestrating and tracking complex data workflows. When integrated with Google BigQuery, a powerful analytical data warehouse, Airflow delivers several key benefits:

- **Scheduling and Automation**: Airflow facilitates the scheduling of ELT (Extract, Load, Transform) jobs, enabling seamless data transfers to BigQuery on a regular timetable.
- **Workflow Management**: It empowers users to define and manage intricate workflows efficiently, simplifying the process of consolidating data from diverse sources into BigQuery.
- **Monitoring and Alerting**: Airflow includes a user-friendly graphical interface that allows for real-time monitoring of job progress and sends alerts when issues arise.
- **Scalability and Flexibility**: Airflow is designed to scale effortlessly, accommodating large and complex workflows, while also providing the flexibility to customize jobs to meet specific organizational needs.

2. Airflow DAGs

We designed code that defines an ELT (Extract, Load, Transform) pipeline implemented using Apache Airflow for orchestrating data workflows. The pipeline is designed to process Amazon sales data, load it into a BigQuery staging area, transform it, and load it into dimension and fact tables. The pipeline ensures data reliability, scalability, and repeatability.

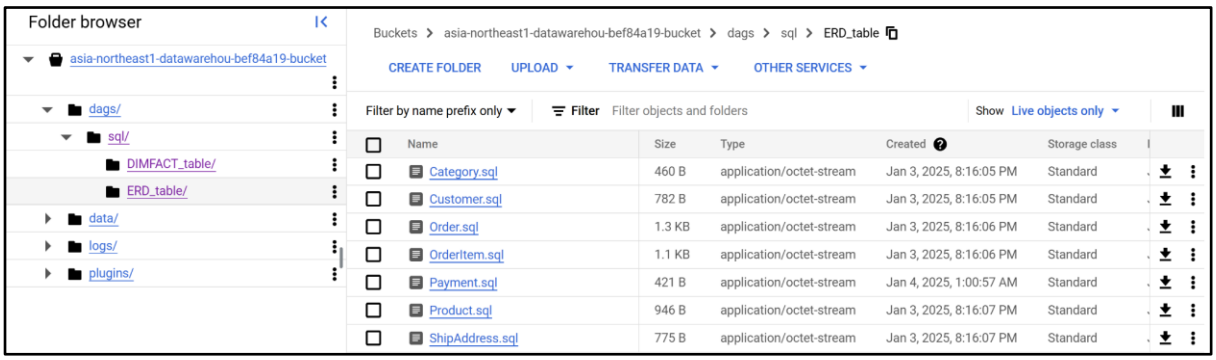
Imports and Constants

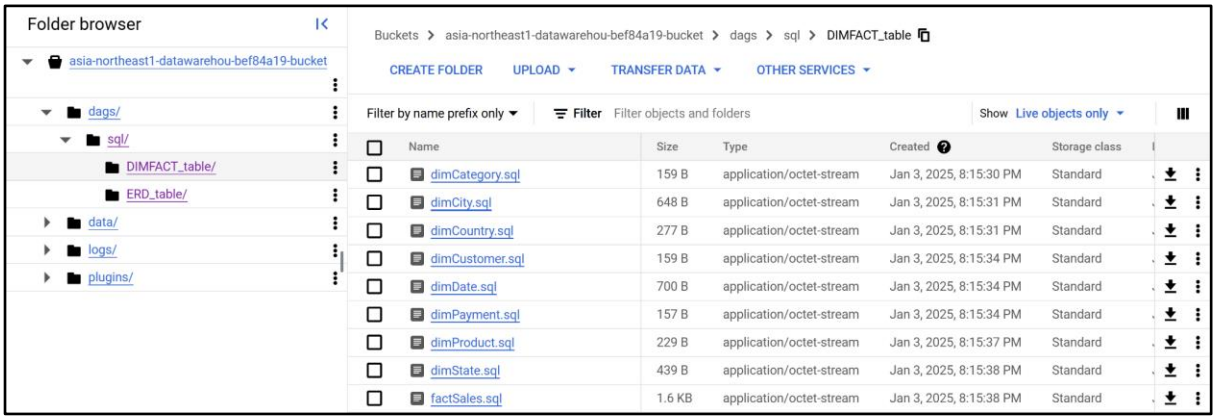
Import necessary modules and set up DAGs structure:

```python
import os
from datetime import timedelta, datetime
from airflow import DAG
from airflow.utils.dates import days_ago
from airflow.operators.empty import EmptyOperator
from airflow.providers.google.cloud.operators.bigquery import BigQueryCheckOperator, BigQueryInsertJobOperator
from airflow.providers.google.cloud.transfers.gcs_to_bigquery import GCSToBigQueryOperator
from airflow.providers.google.cloud.hooks.gcs import GCSHook

# Constants
GOOGLE_CONN_ID = "google_cloud_default"
PROJECT_ID = "datawarehousing-444903"
GS_PATH = "rawdata/"
BUCKET_NAME = 'rawdatafinal'
BUCKET_NAME2 = 'asia-northeast1-datawarehou-bef84a19-bucket'
STAGING_DATASET = "amazon_staging_dataset"
LOCATION = "asia-southeast1"
SQL_FOLDER_PATH_1 = "dags/sql/ERD_table/"
SQL_FOLDER_PATH_2 = "dags/sql/DIMFACT_table/"
```

The Constants section defines reusable parameters for the pipeline, including GOOGLE_CONN_ID for Google Cloud connection, PROJECT_ID for the project ID, BUCKET_NAME and BUCKET_NAME2 define the GCS buckets used for raw data storage and SQL script storage, respectively. The STAGING_DATASET represents the name of the BigQuery staging dataset, which serves as an intermediary storage location for data during transformations. Additionally, SQL_FOLDER_PATH_1 and SQL_FOLDER_PATH_2 provide paths to the directories in GCS where SQL scripts are stored.





Data Loading from CSV

```python
    # Loading Amazon Sale Report CSV
    load_dataset_amazon_sale_report = GCSToBigQueryOperator(
        task_id='load_dataset_amazon_sale_report',
        bucket=BUCKET_NAME,
        source_objects=['rawdata/Amazon Sale Report.csv'],
        destination_project_dataset_table=f'{PROJECT_ID}:{STAGING_DATASET}.amazon_sale_report',
        write_disposition='WRITE_TRUNCATE',
        source_format='CSV',
        allow_quoted_newlines=True,
        skip_leading_rows=1,
        schema_fields=[
            {'name': 'index', 'type': 'INTEGER', 'mode': 'NULLABLE'},
            {'name': 'Order_ID', 'type': 'STRING', 'mode': 'NULLABLE'},
            {'name': 'Date', 'type': 'DATE', 'mode': 'NULLABLE'},
            {'name': 'Status', 'type': 'STRING', 'mode': 'NULLABLE'},
```

```python
# Loading Amazon Sales FY2020-21 CSV
    load_dataset_amazon_sales_fy2020_21 = GCSToBigQueryOperator(
        task_id='load_dataset_amazon_sales_fy2020_21',
        bucket=BUCKET_NAME,
        source_objects=['rawdata/amazon_sales_fy2020_21.csv'],
        destination_project_dataset_table=f'{PROJECT_ID}:{STAGING_DATASET}.amazon_sales_fy2020_21',
        write_disposition='WRITE_TRUNCATE',
        source_format='CSV',
        allow_quoted_newlines=True,
        skip_leading_rows=1,
        schema_fields = [
            {'name': 'order_id', 'type': 'INTEGER', 'mode': 'NULLABLE'},
            {'name': 'order_date', 'type': 'DATE', 'mode': 'NULLABLE'},
            {'name': 'status', 'type': 'STRING', 'mode': 'NULLABLE'},
```

This section includes tasks that load raw data from GCS into BigQuery staging tables. Two GCSToBigQueryOperator tasks, load_dataset_amazon_sale_report and load_dataset_amazon_sales_fy2020_21, import datasets from GCS and define their schema explicitly. The staging tables in BigQuery serve as intermediate storage, where raw data is prepared for subsequent transformations.

Creating ERD Tables

```python
    create_ERD_Tables = EmptyOperator(
        task_id = 'create_ERD_Tables',
        dag = dag
        )


# Create Customer table
    create_customer = BigQueryInsertJobOperator(
        task_id='create_customer',
        configuration={
            "query": {
                "query": customer_sql.format(
                    PROJECT_ID=PROJECT_ID,
                    STAGING_DATASET=STAGING_DATASET
                ),
                "useLegacySql": False
            }
        },
        location='US'
    )


# Create Category Table
    create_category = BigQueryInsertJobOperator(
        task_id='create_category',
```

The Creating ERD Tables section defines the pipeline's logic for creating normalized tables based on Entity-Relationship Diagrams (ERD). Each table creation task, such as create_customer and create_category, uses the BigQueryInsertJobOperator to execute pre-written SQL scripts stored in GCS. These tasks transform and load the staged data into structured formats within the BigQuery staging dataset.

Transforming Dimensional and Fact Tables

```
298     Transform_DIMFACT_tables = EmptyOperator(
299         task_id = 'Transform_DIMFACT_tables',
300         dag = dag
301         )
302
303     # Create dimDate Table
304     dim_date = BigQueryInsertJobOperator(
305         task_id='dim_date',
306         configuration={
307                 "query": {
308                     "query": dimDate_sql.format(
309                         PROJECT_ID=PROJECT_ID,
310                         STAGING_DATASET=STAGING_DATASET
311                     ),
312                     "useLegacySql": False
313                 }
314             },
315             location='US'
316         )
317
318     # Create dimCategory Table
319     dim_category = BigQueryInsertJobOperator(
320         task_id='dim_category',
321         configuration={
322                 "query": {
323                     "query": dimCategory_sql.format(
324                         PROJECT_ID=PROJECT_ID,
325                         STAGING_DATASET=STAGING_DATASET
326                     ),
327                     "useLegacySql": False
328                 }
329             },
330             location='US'
331         )
```

This section is dedicated to building dimensional and fact tables for analytical purposes. SQL scripts for dimensions such as dimCustomer, dimCategory, and dimPayment, as well as the factSales table, are retrieved from GCS and executed using the BigQueryInsertJobOperator. These tasks use the structured data from staging tables to create optimized tables for querying and reporting, following the principles of a star schema.

SQL File Structure

The SQL files play a crucial role in defining the schema and structure of the tables created within the Google BigQuery environment. Each SQL script is designed to execute specific queries that facilitate the transformation of raw data into structured tables suitable for analysis. By separating SQL logic from the main DAG code, the

design promotes modularity, maintainability, and clarity. Below are some examples of the contents of each SQL file used in the ELT pipeline:

dags > sql > ERD_table > Product.sql

```sql
CREATE OR REPLACE TABLE `datawarehousing-444903.amazon_staging_dataset.Product` AS
SELECT DISTINCT
    SKU AS ProductID,
    Style,
    Size,
    ROUND(AVG(Amount), 2) AS Price,
    c.CategoryID AS CategoryID
FROM `datawarehousing-444903.amazon_staging_dataset.amazon_sale_report` a1
INNER JOIN `datawarehousing-444903.amazon_staging_dataset.Category` c
    ON a1.Category = c.CategoryName
WHERE Amount IS NOT NULL AND Amount !=0
GROUP BY SKU, Style, Size, c.CategoryID

UNION DISTINCT

SELECT DISTINCT
    sku AS ProductID,
    CAST(NULL AS STRING) AS Style,
    CAST(NULL AS STRING) AS Size,
    ROUND(AVG(price), 2) AS Price,
    c.CategoryID AS CategoryID
FROM `datawarehousing-444903.amazon_staging_dataset.amazon_sales_fy2020_21` a2
INNER JOIN `datawarehousing-444903.amazon_staging_dataset.Category` c
    ON a2.Category = c.CategoryName
WHERE price IS NOT NULL AND price !=0
GROUP BY sku, c.CategoryID;
```

Figure 6: Creation of Product Table

dags > sql > DIMFACT_table > factSales.sql

```sql
CREATE OR REPLACE TABLE `datawarehousing-444903.amazon_data_warehouse.factSales` AS
SELECT
    oi.OrderItemID AS SaleOrderID,
    o.OrderID,
    oi.ProductID,
    dd.DateID AS OrderDateID,
    cus.CustomerID,
    c.CityID,
    pm.PaymentID,
    o.Status,
    oi.Quantity AS UnitSold,
    CAST(p.ListedPrice * (1 - COALESCE(oi.DiscountPercent, 0) / 100) AS FLOAT64) AS DiscountedPrice,
    oi.DiscountPercent,
    CAST(p.ListedPrice * oi.Quantity * (1 - COALESCE(oi.DiscountPercent, 0) / 100) AS FLOAT64) AS TotalAmount
FROM `datawarehousing-444903.amazon_staging_dataset.OrderItem` oi
INNER JOIN `datawarehousing-444903.amazon_staging_dataset.Order` o
    ON oi.OrderID = o.OrderID
INNER JOIN `datawarehousing-444903.amazon_data_warehouse.dimDate` dd
    ON CAST(o.OrderDate AS DATE) = dd.FullDate
INNER JOIN `datawarehousing-444903.amazon_data_warehouse.dimCustomer` cus
    ON o.CustomerID = cus.CustomerID
INNER JOIN `datawarehousing-444903.amazon_data_warehouse.dimCity` c
    ON o.ShipID = c.CityID
INNER JOIN `datawarehousing-444903.amazon_data_warehouse.dimProduct` p
    ON oi.ProductID = p.ProductID
INNER JOIN `datawarehousing-444903.amazon_data_warehouse.dimPayment` pm
    ON o.PaymentID = pm.PaymentID
WHERE oi.Quantity IS NOT NULL AND oi.Quantity != 0;
```

Figure 7: Creation of factSales Table

Running DAGs

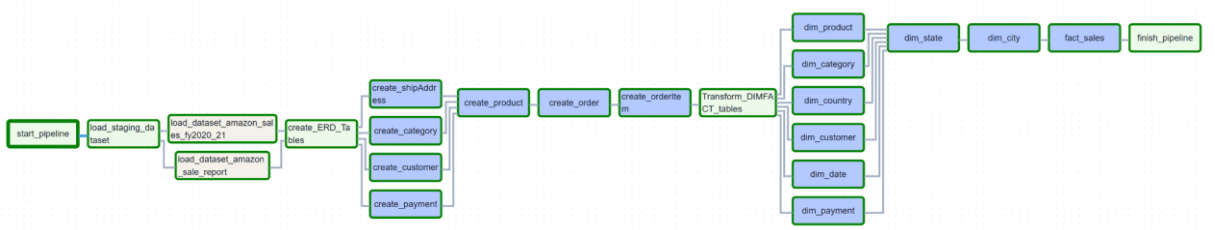


Figure 8: ELT Pipelines with Airflow

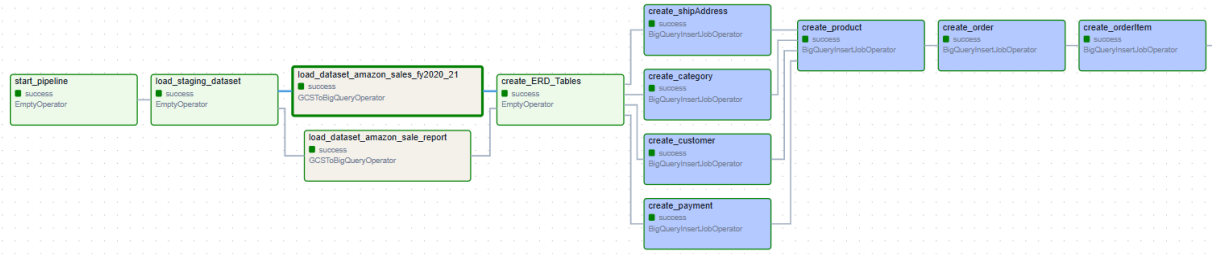


Figure 9: Loading DAGs

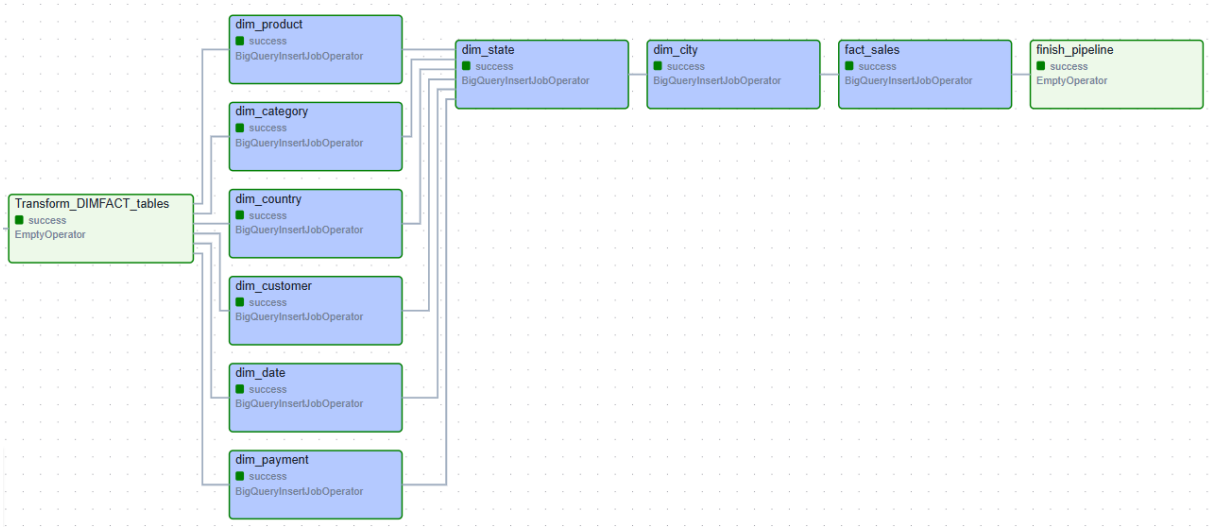
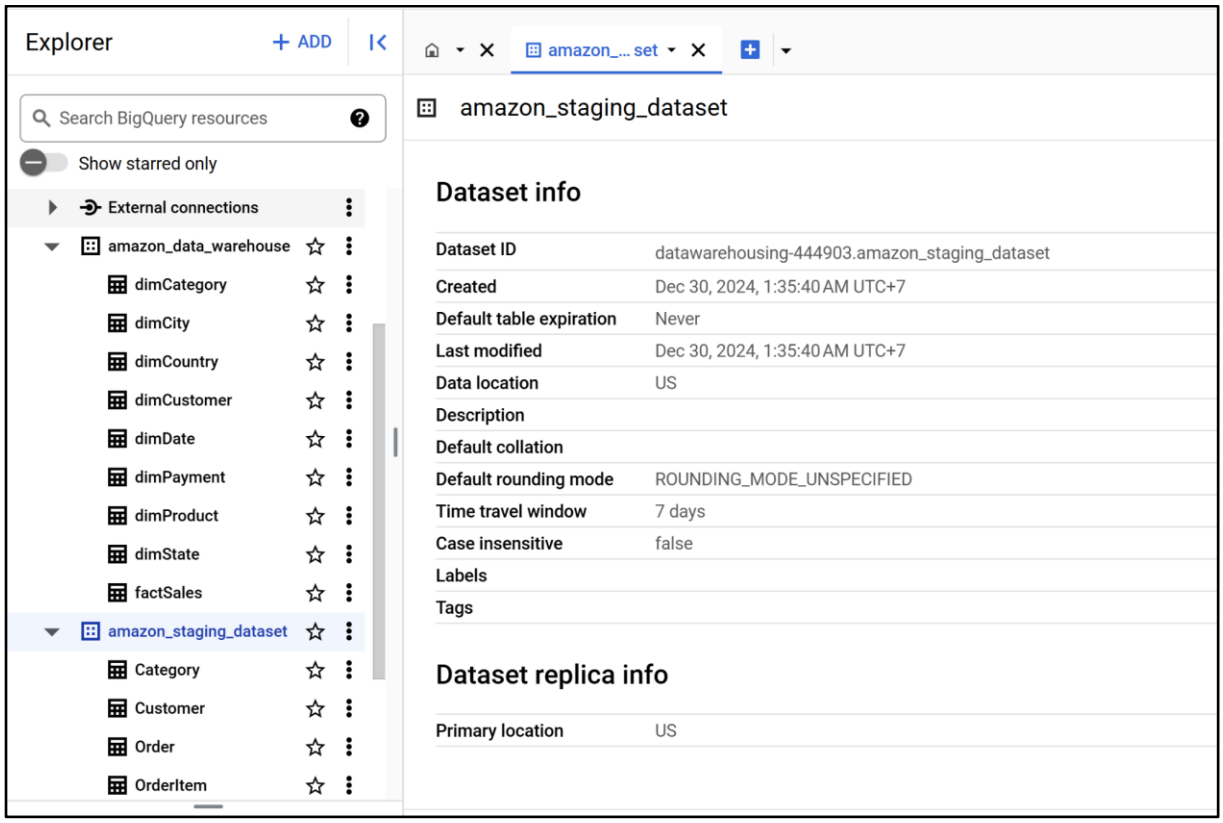


Figure 10: Transform DAGs

The process begins with the `*start_pipeline*` task, which acts as the entry point. Following this, two tasks, `*load_dataset_amazon_sales_fy2020_21*` and `*load_dataset_amazon_sale_report*`, load raw data from GCS into BigQuery staging tables. These tasks ensure that all required datasets are available for subsequent transformations. The next step is the creation of ERD (Entity-Relationship Diagram) tables, where the raw data is normalized and distributed across logically structured tables. Subsequently, the `*Transform_DIMFACT_tables*` task processes the normalized data to create dimensional and fact tables, crucial for analytical queries and reporting.

These transformations organize the data in a star schema format, optimizing it for querying and business intelligence purposes. The pipeline concludes with the *`finish_pipeline`* task, signaling the successful execution of all prior tasks. The dependencies in this pipeline are meticulously defined to ensure that tasks execute in the correct sequence, maintaining the integrity of the data as it moves through extraction, transformation, and loading stages. This structured orchestration ensures efficient data processing and preparation for analytical consumption.



Upon successfully executing the DAGs, we can view all the generated ERD tables, along with dimension and fact tables, complete with their respective values in BigQuery.

V. RESULTS AND IMPLICATIONS

1. Querying on OLAP

This part includes SQL queries used to assist in exploring the data analysis process of the business. Each question is queried and gives back results, then visualized in Looker or BigQuery to give a graphical view.

Q1: What are the sales by each age group?

```sql
---What are the Sales by each age group?
  SELECT
    CASE
      WHEN c.Age BETWEEN 13 AND 19 THEN '13-19'
      WHEN c.Age BETWEEN 20 AND 29 THEN '20-29'
      WHEN c.Age BETWEEN 30 AND 39 THEN '30-39'
      WHEN c.Age BETWEEN 40 AND 49 THEN '40-49'
      WHEN c.Age BETWEEN 50 AND 59 THEN '50-59'
      WHEN c.Age >= 60 THEN '60+'
      ELSE 'Unknown'
    END AS age_group,
    SUM(s.TotalAmount) AS total_sales
  FROM
    `datawarehousing-444903.amazon_data_warehouse.dimCustomer` c
  JOIN
    `datawarehousing-444903.amazon_data_warehouse.factSales` s
  ON
    c.CustomerID = s.CustomerID
  GROUP BY age_group
  ORDER BY age_group;
```

Row	age_group	total_sales
1	13-19	14137521.97778...
2	20-29	78877408.87260...
3	30-39	88309878.75516...
4	40-49	77671014.47007...
5	50-59	75170872.60830...
6	60+	126013594.5992...
7	Unknown	77182910.28997...

Q2: What are the busiest months or quarters for the business?

```sql
---What are the busiest months or quarters for the business?
SELECT
  CONCAT(d.Year,'-',d.Month) AS Month,
  SUM(f.TotalAmount) As Total_Sales,
  SUM(f.UnitSold) AS Product_Sold
FROM
  `datawarehousing-444903.amazon_data_warehouse.factSales` f
JOIN
  `datawarehousing-444903.amazon_data_warehouse.dimDate` d
ON
  f.OrderDateID = d.DateID
GROUP BY Month
ORDER BY Month ASC;

```

Row	Month	Total_Sales	Product_Sold
1	2020-10	10263116.00421...	22518
2	2020-11	20548852.86506...	44752
3	2020-12	122844101.3755...	184619
4	2021-1	13893869.78393...	47797
5	2021-2	9677785.511986...	32472
6	2021-3	46854240.96023...	101372
7	2021-4	65509931.82357...	150952
8	2021-5	18390939.13875...	47591
9	2021-6	55727263.81824...	91882

Q3: Which discount percentage is the most effective in boosting sales for each product category?

```sql
---Which discount percentage is the most effective in boosting sales for each product category?
SELECT
  c.CategoryName,
  CASE
    WHEN f.DiscountPercent = 0 THEN 'No Discount'
    WHEN f.DiscountPercent BETWEEN 1 AND 10 THEN '1-10%'
    WHEN f.DiscountPercent BETWEEN 11 AND 20 THEN '11-20%'
    WHEN f.DiscountPercent BETWEEN 21 AND 30 THEN '21-30%'
    WHEN f.DiscountPercent > 30 THEN 'Above 30%'
    ELSE 'Unknown'
  END AS DiscountRange,
  SUM(f.UnitSold) AS TotalUnitsSold
FROM
  `datawarehousing-444903.amazon_data_warehouse.factSales` f
JOIN
  `datawarehousing-444903.amazon_data_warehouse.dimProduct` p
ON
  f.ProductID = p.ProductID
JOIN
  `datawarehousing-444903.amazon_data_warehouse.dimCategory` c
ON
  p.CategoryID = c.CategoryID
GROUP BY
  c.CategoryName,
  DiscountRange
ORDER BY
  c.CategoryName,
  TotalUnitsSold DESC;
```

Row	CategoryName	DiscountRange	TotalUnitsSold
1	Appliances	No Discount	38451
2	Appliances	11-20%	20627
3	Appliances	21-30%	8772
4	Appliances	1-10%	6344
5	Appliances	Above 30%	4945
6	Appliances	Unknown	3240
7	Beauty & Grooming	No Discount	50211
8	Beauty & Grooming	Above 30%	9103
9	Beauty & Grooming	11-20%	6986
10	Beauty & Grooming	21-30%	2658
11	Beauty & Grooming	1-10%	1566

Q4: How will the ordering trend for next month?

```sql
-- How will the ordering trend for next month?
create or replace model `datawarehousing-444903.amazon_data_warehouse.arima_model`
options(
  model_type='ARIMA_PLUS',
  time_series_timestamp_col='date',
  time_series_data_col='nOrders',
  horizon= 30 * 4
) as
select
  d.FullDate as date,
  count(s.OrderDateID) as nOrders
from
  `datawarehousing-444903.amazon_data_warehouse.factSales` s
  inner join
  `datawarehousing-444903.amazon_data_warehouse.dimDate` d
  on s.OrderDateID = d.DateID
group by d.FullDate

select
  d.FullDate as date,
  count(s.OrderDateID) as nOrders
from
  `datawarehousing-444903.amazon_data_warehouse.factSales` s
  inner join
  `datawarehousing-444903.amazon_data_warehouse.dimDate` d
  on s.OrderDateID = d.DateID
where d.year = 2022
group by d.FullDate
union all
select
  date(forecast_timestamp) as date,
  forecast_value as nOrders
from ml.forecast(
  model `datawarehousing-444903.amazon_data_warehouse.arima_model`,
  struct(120 as horizon)
)
```

Row	date	nOrders
1	2022-04-06	1425.0
2	2022-04-02	1402.0
3	2022-04-01	1311.0
4	2022-06-04	1335.0
5	2022-05-02	1879.0
6	2022-04-08	1516.0
7	2022-06-16	1149.0
8	2022-04-04	1324.0
9	2022-06-09	1398.0
10	2022-06-18	1066.0
11	2022-06-20	1101.0
12	2022-04-11	1383.0
13	2022-06-17	1046.0

Results per page: 50 1 – 50 of 211

Q5: What groups can customers be segmented into based on shopping behavior?

```sql
-- What groups can customers be segmented into based on shopping behavior?
create or replace table `datawarehousing-444903.amazon_data_warehouse.customer_behavior`
as
select
  unix_date(max(d.fulldate)) as recency,
  7 * count(orderid) / ( unix_date(max(d.fulldate)) - unix_date(min(d.fulldate)) + 1) as frequency,
  avg(totalamount) as monetary
from
  `datawarehousing-444903.amazon_data_warehouse.factSales` s
  inner join
  `datawarehousing-444903.amazon_data_warehouse.dimDate` d
  on s.orderdateid = d.dateid
group by customerid;

delete from `datawarehousing-444903.amazon_data_warehouse.customer_behavior`
where frequency > 8000
```

```sql
CREATE OR REPLACE MODEL `datawarehousing-444903.amazon_data_warehouse.pca_model`
OPTIONS(model_type='PCA', num_principal_components=2) AS
SELECT
  recency,
  frequency,
  monetary
FROM `datawarehousing-444903.amazon_data_warehouse.customer_behavior`
```

```sql
create or replace model `datawarehousing-444903.amazon_data_warehouse.kmeans_model`
options(model_type='kmeans', num_clusters=3) as
select
  recency,
  frequency,
  monetary
from `datawarehousing-444903.amazon_data_warehouse.customer_behavior`
```

```sql
with pca as (
  select
    *,
    row_number() over (order by (select null)) as customerid,
  from ML.PREDICT(
    model `datawarehousing-444903.amazon_data_warehouse.pca_model`,
    (
      select
        recency,
        frequency,
        monetary
      from `datawarehousing-444903.amazon_data_warehouse.customer_behavior`
    )
  )
), kmeans as (
  select
    row_number() over (order by (select null)) as customerid,
    centroid_id,
    recency,
    frequency,
    monetary
  from ml.predict(
    model `datawarehousing-444903.amazon_data_warehouse.kmeans_model`,
    (
      select
        recency,
        frequency,
        monetary
      from `datawarehousing-444903.amazon_data_warehouse.customer_behavior`
    )
  )
)
select * from kmeans
inner join pca on kmeans.customerid = pca.customerid
```

Row	customerid	centroid_id	recency	frequency	monetary	principal_component	principal_component
1	5	1	18701	7.0	81.09	-0.06235332716...	-0.63883529704...
2	6	1	18710	7.0	144.0	-0.32044937931...	-0.77495669704...
3	7	1	18746	7.0	200.0	-0.34831247036...	-0.67187339063...
4	8	1	18747	7.0	300.0	-0.62996237593...	0.138384241180...
5	9	1	18747	7.0	250.0	3.724340687480...	1.414830609234...
6	10	1	18747	7.0	400.0	-0.38779494379...	-0.73518516313...
7	11	1	18771	7.0	159.8	-0.32506366707...	-0.66969460521...
8	14	1	18806	7.0	999.0	-0.22145853132...	-0.66260537419...
9	15	1	18856	7.0	3413.0	-0.46808758185...	-0.53710746290...
10	20	1	18758	0.233333333333	5937.940000000	-0.27545104568	-0.63155458855

Results per page: 50 1 – 50 of 64127

2. Exploratory Data Analysis

Q1: What are the busiest sales months or quarters for the business?

Below is a combined chart, reflecting sales revenue from products sold and the number of products sold by month.

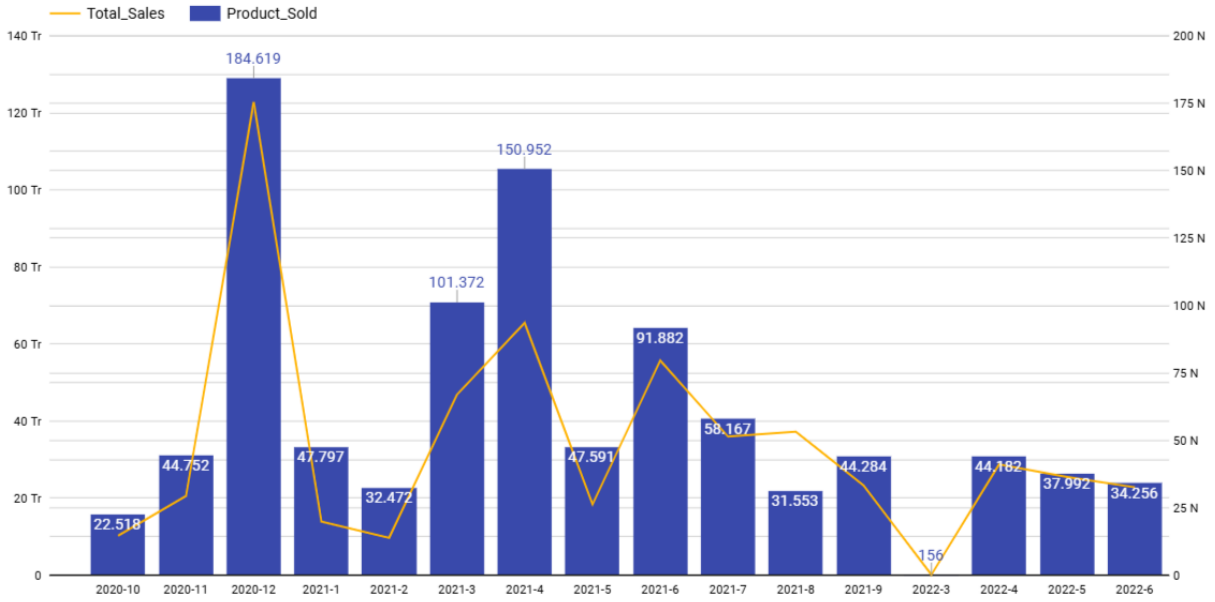


Figure 11: Sale Revenue and Product Sold by Months

From the chart, it is evident that sales and products sold in December at the end of 2020 showed the most promising figures. Placed in the context of late 2020, after the complicated developments of the COVID-19 pandemic, the surge in online shopping platforms makes this trend entirely understandable. Additionally, sales during March and April 2021 were also notable, with an average of 125,000 products sold per month on Amazon. Visualizing monthly sales figures helps to identify seasonal trends and buying behaviors more easily.

The data shows that the final months of the year typically see high sales due to major holidays such as Christmas Eve and New Year's Eve. These are times when shops on e-commerce platforms often offer significant discounts and promotions to attract customers. After the New Year, there is a noticeable downward fluctuation in the number of products sold during January and February. However, by March and April, at the start of the second quarter, customer demand increases again as they begin preparing for the rest of the year.

Additionally, customer shopping behavior is influenced by their age group. According to statistics, the majority of Amazon customers are over 60 years old or fall within the 30-39 age range. These groups are likely to focus their shopping toward the end of the year to prepare for family needs. While those over 60 may not actively participate in peak sales seasons (like Black Friday), they tend to shop steadily throughout the year for essential items.

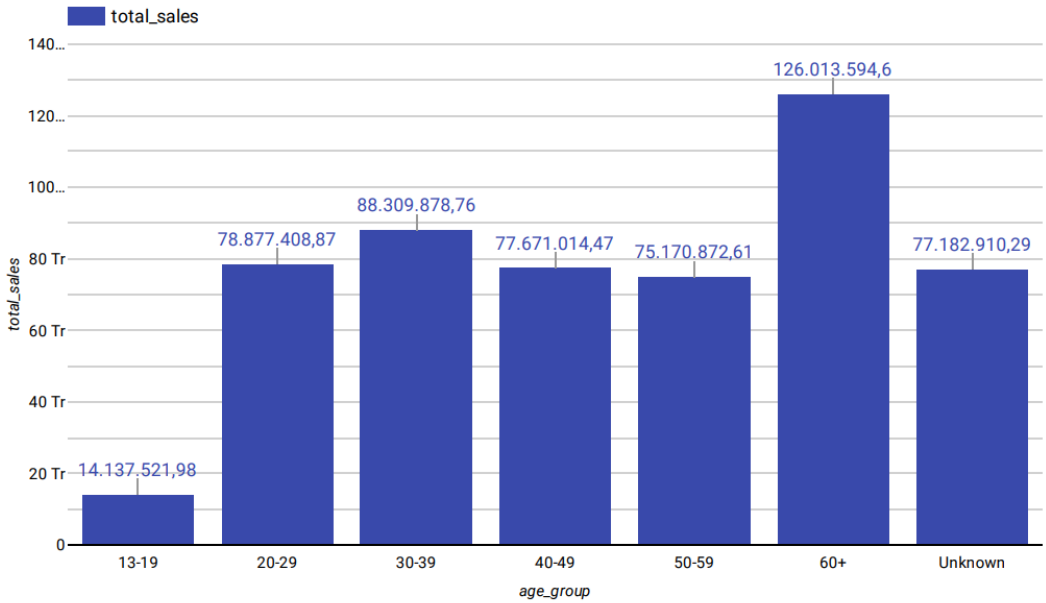


Figure 12: Sale by Age Groups

Identifying seasonal trends across years is crucial to understanding customer shopping behavior. This allows for the strategic use of promotions to boost sales more effectively while saving costs. Based on the analysis, the busiest months of the year often fall in the late fourth quarter, specifically November and December. Moving into the following year, sales can be maximized toward the end of the first quarter and the beginning of the second quarter. In the latter half of the year, sales may rebound again at the start of the winter season.

Q2: How will the ordering trend for next month?

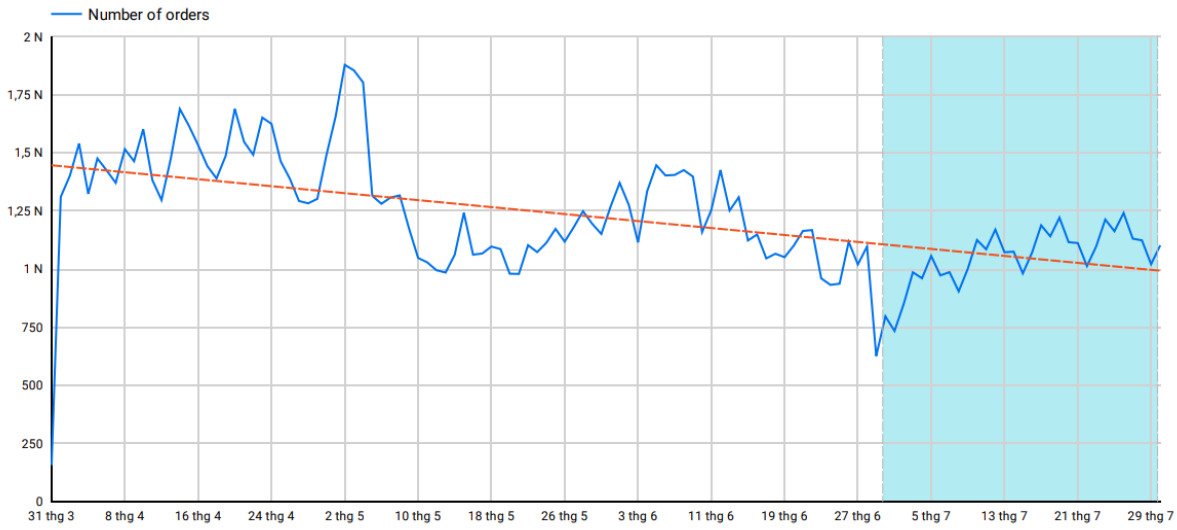


Figure 13: Sale Forecast

The chart shows the number of orders from the end of March to the end of June, along with a forecast for July. The actual data reveals significant fluctuations in the number of orders, with irregular peaks and troughs. The trendline (orange) is sloping downward, indicating a general decline in order volume. The forecast area for July shows more stable order numbers but continues to reflect a slight downward trend.

Insights:

- The data shows a decrease in order volume over the months.
- Between April and early May, there were noticeable spikes in order volume, possibly due to promotional campaigns, special events, or other short-term factors.
- The forecast for July predicts a continued decline in orders but with less fluctuation, indicating the downward trend has not been significantly improved.

Implications for Amazon's Sales Strategy:

- Diversify the product portfolio to meet changing customer demands and reduce reliance on saturated products.
- Build customer retention strategies, such as loyalty programs and tailored incentives for different customer groups, to increase repeat purchases.
- Adjust business strategies to align with current trends:
    + Strengthen communication about product value by emphasizing standout features or superior benefits compared to competitors.
    + Consider pricing adjustments or launching appropriate promotional campaigns to stimulate demand and boost sales.

Q3: Which discount percentage is the most effective in boosting sales for each product category?

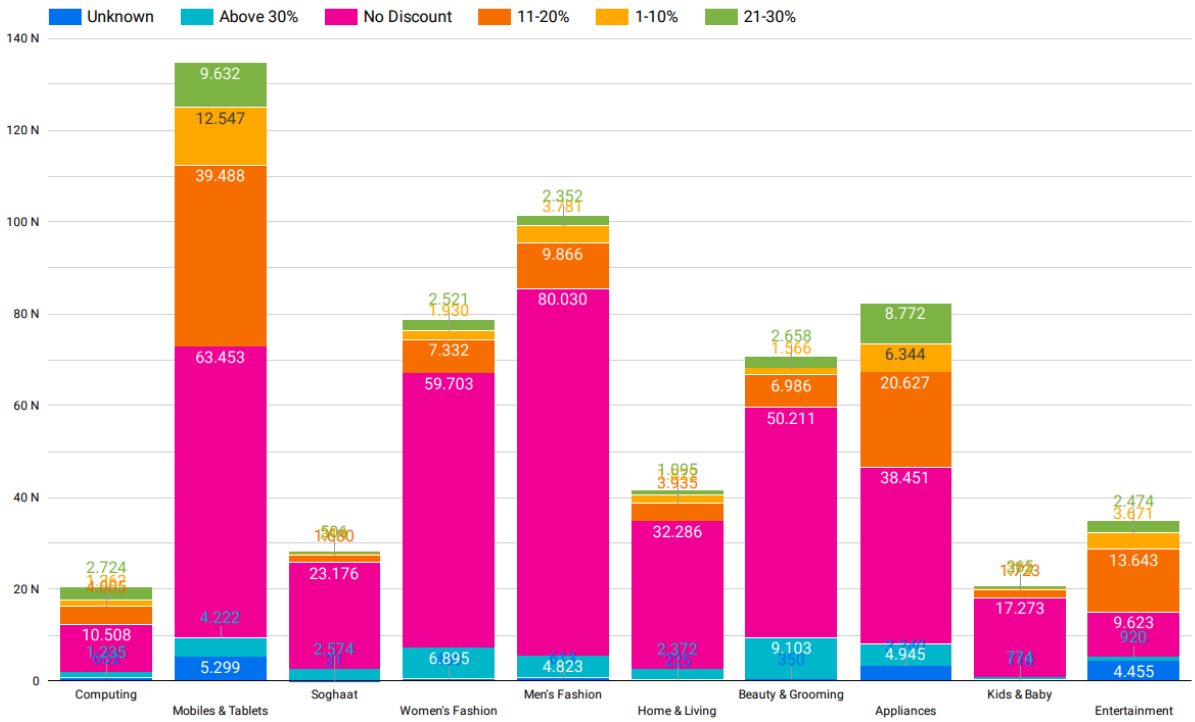


Figure 14: Sales by Discount Percent and Category

Based on the chart, an overview of revenue by discount percentage can be observed as follows:

- **No discount**: Certain categories, such as Mobile & Tablets, Men's & Women's Fashion, achieve the highest sales without any discount. This indicates the essential nature of products in these categories, as a large number of customers are still willing to pay despite no discounts.
- **1-10% discount:** This discount level generates good revenue for categories like Mobile & Tablets, showing that customers can be attracted by minor promotions for popular products. A small discount might make purchasing decisions easier for them.
- **10-20% discount**: This discount level generates the highest revenue for categories such as Home & Entertainment and Appliances, showing that household goods may require moderate discounts to boost sales. These goods are often mid- to high-value items and face significant market competition.
- **21-30% discount:** Products in categories like Beauty & Grooming and Computing perform well under this discount level. Since these items are not frequently replaced, customers often wait for big sales to purchase them.
- **Above 30% discount:** It is evident that Women's Fashion and Beauty & Grooming benefit the most from deep discounts. These categories are highly

price-sensitive, and customers in these groups are easily motivated by large discounts.

Insights:

- Categories responding well to small discounts and having high-profit margins (0 → 10%): Mobile & Tablets, Men's Fashion.
- Categories requiring moderate discounts to stay competitive (11 → 30%): Home & Living, Appliances, Women's Fashion.
- Categories needing deep discounts to drive campaigns (Above 30%): Entertainment, Beauty & Grooming.

Implications for Amazon's Sales Strategy:

- Use small discounts for customers who are less price-sensitive.
- Offer moderate discounts for mid to high-value items to balance promotions and profits.
- Leverage deep discounts for major categories to strongly enhance sales.

By doing so, unnecessary costs can be reduced, and customer purchasing behaviors can be better understood.

Q4: What groups can customers be segmented into based on shopping behavior?

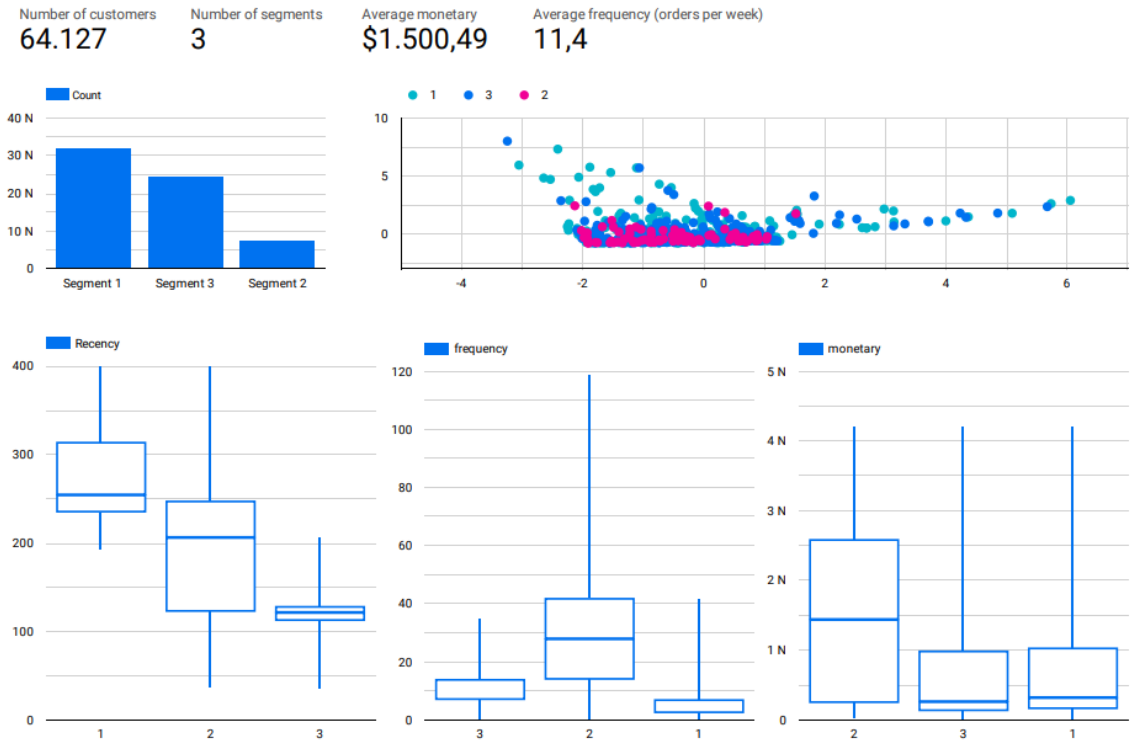


Figure 15: Customer Segmentation

In an effort to segment customers to enhance engagement and optimize revenue, we applied the K-Means clustering method based on three factors: **Recency** (number of

days since the last purchase), **Frequency** (purchase frequency), and **Monetary** (spending value). Additionally, we incorporated the Principal Component Analysis (PCA) technique to reduce data dimensionality. Through this analysis, we identified three customer segments:

- **Segment 1: Inactive**: This group represents the lowest value segment, consisting of customers who have made few recent purchases with low frequency and spending levels. They include individuals who once purchased but no longer maintain engagement with the business. Companies can implement reactivation strategies such as sending special discount offers, reminder emails, or personalized incentives to encourage them to return. However, if reactivation efforts prove ineffective, businesses may consider reducing resource allocation to this group and focusing on other segments with higher value.
- **Segment 2: VIP**: This is the most important customer group, accounting for a small percentage (approximately **10,000 customers**). However, they exhibit high purchase frequency, significant spending, and strong loyalty, contributing significantly to the company's revenue. Businesses should focus on retaining this group by offering exclusive benefits, personalized campaigns (such as rewards or special discounts), and premium services.
- **Segment 3: Active**: This group consists of potential customers with over **20,000 individuals**. They are customers who have recently made purchases, with medium levels of frequency and spending. Although they are not as critical as the VIP group, they remain active and show potential to become loyal customers if nurtured properly. Businesses can implement strategies to encourage them to shop more frequently, such as point accumulation programs, introducing new products, or offering promotional campaigns.

VI. CONCLUSION

In this project, we leveraged Data Warehouse technology to optimize and analyze Amazon's sales performance, focusing on streamlining data management, enhancing business insights, and improving operational efficiency. By integrating two comprehensive datasets and applying a structured approach through ELT pipelines, we successfully transformed raw sales data into actionable insights that can drive better decision-making.

The design of the Data Warehouse, using a snowflake schema for dimensional modeling, allowed for efficient querying, reporting, and performance tracking across different dimensions such as time, geography, and product categories. Through the implementation of the Entity-Relationship Diagram (ERD), we modeled key business

entities and their relationships, ensuring smooth data flow and analysis. The use of Apache Airflow to automate and manage the data pipeline further increased the efficiency and scalability of the process.

By analyzing sales trends, customer behavior, product performance, and regional insights, we identified valuable patterns that could inform strategic decisions such as targeted marketing, inventory optimization, and forecasting future demand. The project also highlighted the importance of clean, reliable data in making data-driven decisions that align with business goals.

The visualizations and insights generated through tools like Looker and BigQuery provide clear, intuitive representations of sales performance, which can empower stakeholders to act swiftly and with confidence. Ultimately, this project demonstrates the power of a well-designed Data Warehouse in enhancing the ability to manage large datasets, make informed decisions, and optimize overall business performance.

Moving forward, the integration of advanced analytics, such as predictive modeling and machine learning, could further enhance the insights provided by the Data Warehouse, driving even greater operational success and sustained growth for businesses like Amazon.

REFERENCES

[1] Amazon Sales FY2020-21. (2022, November 8). Kaggle. https://www.kaggle.com/datasets/earthfromtop/amazon-sales-fy202021?select=Amazon+Sales+FY2020-21.csv

[2] E-Commerce Sales Dataset. (2022, December 3). Kaggle. https://www.kaggle.com/datasets/thedevastator/unlock-profits-with-e-commerce-sales-data

[3] Sheldon, R. (2023, July 20). dimension table. Search Data Management. https://www.techtarget.com/searchdatamanagement/definition/dimension-table

[4] Simmdata. (2022, September 1). data warehousing automation using airflow and big query in gcp [Video]. YouTube. https://www.youtube.com/watch?v=aV9_GzM8qvI