TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

Business Intelligence & Database Management Project

Paper written by *Mouna Saadaoui & Houssine Khlif*

Major: Marketing/BA - **Minor:** IT

Table of contents

1. **Introduction and requirement gathering**
2. **Data and resources gathering**
3. **Multidimensional modeling**
4. **ETL process design and development**
5. **Data warehouse creation and data storage**
6. **Data visualization and analysis**

1. Introduction and requirement gathering

1.1 About the project

This project focuses on optimizing the performance of Amazon's operations by creating a data warehouse (DWH) and leveraging analytical tools to derive insights. The primary goal is to enhance decision-making processes, improve resource allocation, and support strategic planning

1.2 Requirements gathering

Key questions identified for this project:

1. Which product categories and subcategories perform best?
2. What regions exhibit the highest sales and customer engagement?
3. How can marketing campaigns be optimized for better ROI?
4. What are the seasonal trends and customer purchasing behaviors?
5. Which operational inefficiencies can be mitigated to improve performance?

2. Data and resources gathering

2.1 Data sources

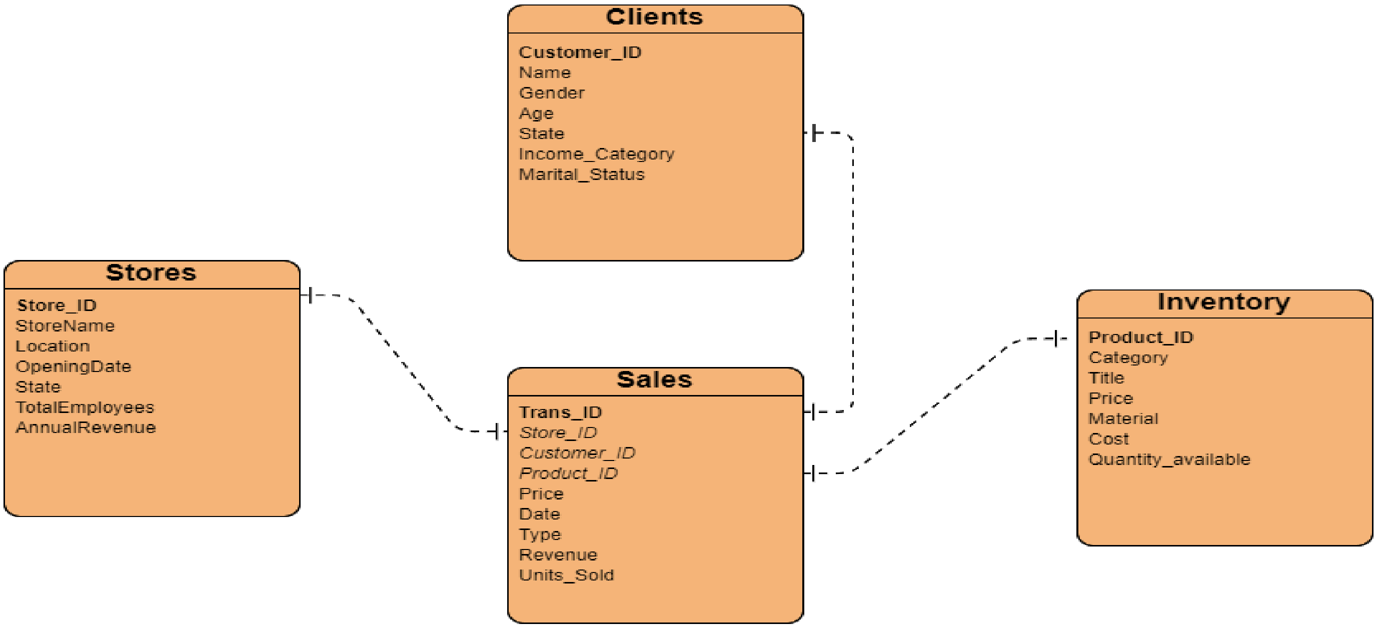
- ***Internal database:***

The company's internal database is a well-organized relational system designed to store and manage critical business information. It serves as the backbone for key operations such as inventory management, sales monitoring, and client relationship management.

This database, named **Company_Performance_DWH**, consists of four interconnected tables: **SALES**, **INVENTORY**, **CLIENTS**, and **STORES**. These tables are linked through defined relationships, including Primary and Foreign Keys. Below is an overview of its

Relational Schema to demonstrate the database structure:

Internal Database Relational Schema



➢ *The data in this database originates from multiple sources. The **CLIENTS** and **SALES** datasets were sourced from the Kaggle website, containing information aligned with the attributes outlined in the schema. On the other hand, the **STORES** and **INVENTORY** datasets were manually generated using Microsoft Excel, with simulated and random data tailored to fit the structure of the company's database. Finally, these four datasets were combined to build the SQL database in SQL Workbench, where the files were imported, and their relationships were established.*

- ***External data sources:***

Amazon's core competency lies in efficient logistics, customer service, and offering a vast range of products, which includes managing its inventory budget strategically. To optimize operations, the company requires up-to-date information about consumer behavior and market trends, as well as historical data on regional and demographic insights. This information helps Amazon allocate its resources effectively across its supply chain and inventory management.

To support this, the following external data is utilized, though it has been simulated for analytical purposes:

1. **Geographical data**:
    - **File Type**: CSV
    - Data representing various regions, including population density, infrastructure, and market potential. This data is used to help optimize warehouse locations and delivery routes.
2. **Demographic data**:
    - **File Type**: CSV
    - Data provides insights into customer segments, such as age, income, and preferences. This information enables Amazon to tailor its marketing strategies and product offerings for different markets.

This external data, combined with internal databases, ensures that Amazon can analyze trends, strategize effectively, and maintain its competitive edge while managing resources efficiently.

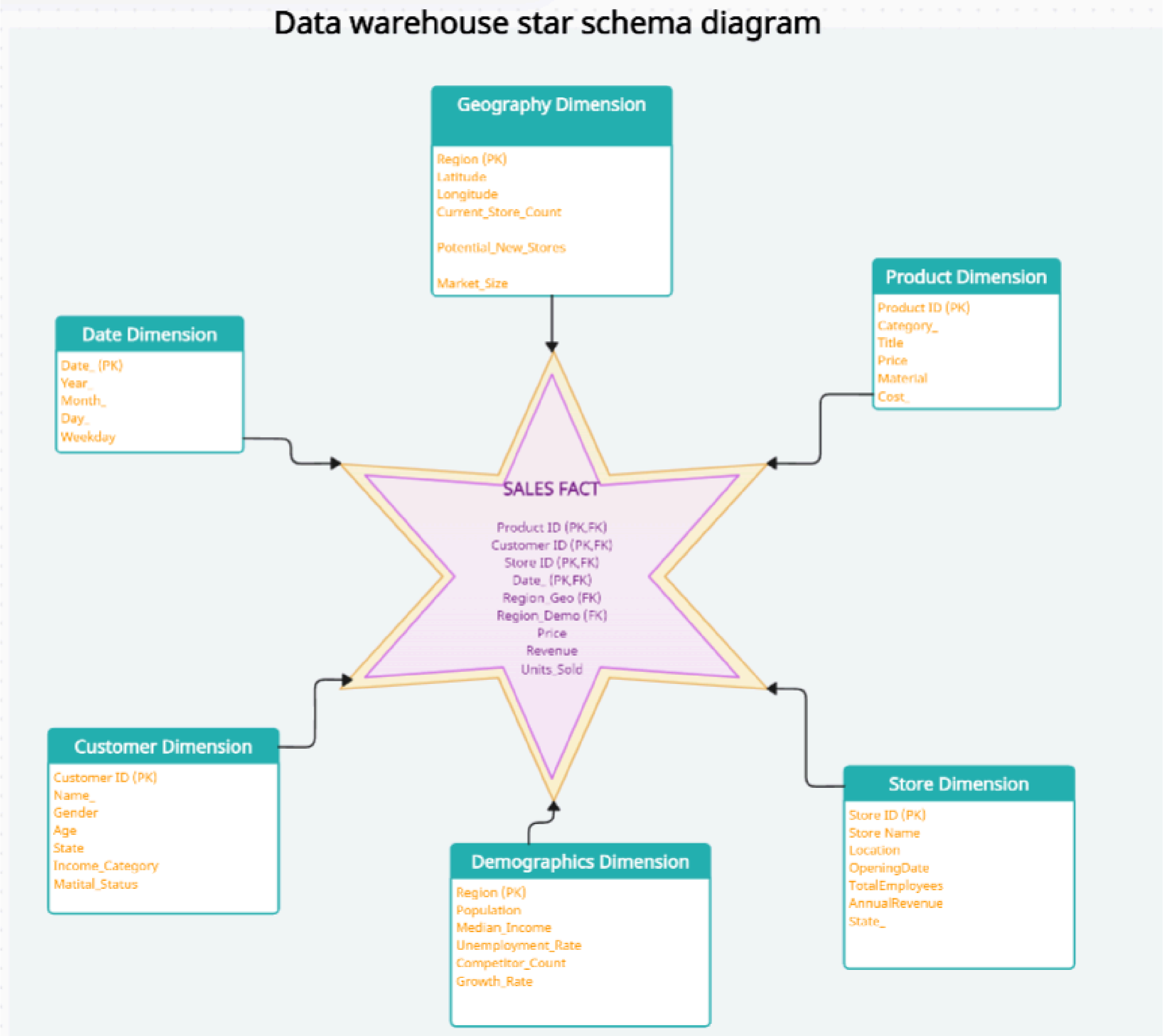
- ***Throughout the project, we will be working with 2 different data sources types: SQL database and CSV file.***

---

3. Multidimensional modeling

As the Data Warehouse is yet to be established, it is more practical to design and develop its dimensional model before initiating the ETL process. The transformation step in ETL involves adapting the extracted data to align with the target system's structure, which cannot proceed without a clearly defined model.

A dimensional model serves as a blueprint for a particular business process. In line with the managers' requirements, this project will concentrate on analyzing and addressing the **SALES BUSINESS PROCESS**.



03.01 FACT/DIMENSION IDENTIFICATION

This dimensional model, combined with these files, lays the groundwork for constructing the Data Warehouse and performing the ETL process effectively.

Fact table: sales fact table

- **Attributes:**
    - Product_ID
    - Customer_ID
    - StoreID

   - Date
   - Price
   - Revenue
   - Units_Sold
   - Region-Geo
   - Region_Demo

Dimensions and their attributes

1. **Store dimension**
   - StoreID
   - StoreName
   - Location
   - OpeningDate
   - TotalEmployees
   - AnnualRevenue
   - State
2. **Product dimension**
   - Product_ID
   - Category
   - Title
   - Price
   - Material
   - Cost
3. **Geography dimension**
   - Region
   - Latitude
   - Longitude
   - Current_Store_Count
   - Potential_New_Stores
   - Market_Size
4. **Demographics dimension**
   - Region
   - Population
   - Median_Income
   - Unemployment_Rate
   - Competitor_Count
   - Growth_Rate
5. **Date dimension**
   - Date

   - Year
   - Month
   - Day
   - Weekday

6. **Customer dimension**
   - Customer_ID
   - Name
   - Gender
   - Age
   - State
   - Income_Category
   - Marital_Status

3.2 Star schema

Due to the straightforward structure of the dimensions and their direct connection to the fact table, we chose to use a star schema. Furthermore, a star schema is generally preferred over a snowflake schema for OLAP because it offers simpler, more intuitive query structures, leading to faster query performance and easier optimization. The denormalized design of the star schema minimizes the need for complex joins, making it more efficient for large-scale aggregations and analytical queries. Which is our main objective in this project.

---

4. ETL Process design and development

04.01 Data extraction

Before constructing the Data Warehouse, we first carried out the ETL process using **Python**. This phase involved extracting data from various available sources, both internal and external.

External files extraction

Using Python, we extracted data from external files and ensured its quality by applying cleaning procedures. The extraction process included two main types:

1. ***Logical extraction**:*
   A full extraction was performed, where all the data from source files was retrieved completely.

2. ***Physical extraction:***
    - **Offline extraction**: Data from external and internal sources was copied into CSV files for offline processing and fetched for further transformations.

The cleaning was performed using a custom Python function, which:

- Removed duplicates.
- Handled missing values.
- Reformatted column names (example: replacing spaces with underscores).
- Ensured numeric columns were properly typed.
- Added columns that were necessary for the functionality of the star schema.

04.02 Data transformation

The transformation phase was executed in Python to ensure the raw data met the quality, consistency, and compatibility standards required for the star schema. Each table underwent specific transformations, described below:

1. **Date dimension**:
    - Extracted unique dates from the Sales dataset.
    - Added columns for year, month, day, and weekday using Python's datetime module.
2. **Product dimension**:
    - Created by extracting product-related attributes (Product_ID, Category, Title, Price, etc.) from the Inventory dataset.
3. **Customer dimension**:
    - Derived from the Clients dataset by extracting customer-related fields (Customer_ID, Name, Gender, etc.).
    - Duplicates were removed based on Customer_ID.
4. **Store dimension**:
    - Created using store-related data (StoreID, StoreName, Location, etc.) from the Stores dataset.
5. **Geography dimension**:
    - Cleaned and processed data from the Geographical_Data.csv file, keeping relevant columns like Region, Latitude, and Market_Size.
6. **Demographics dimension**:
    - Cleaned and processed data from the Demographic_Data.csv file, focusing on fields such as Population and Median_Income.
7. **Sales fact table**:
    - Extracted from the Sales dataset.

  - Renamed and linked Region to both Geography and Demographics dimensions as Region-Geo and Region-Demo.

Data cleaning example

- **State data transformation**: Underscores in state names (e.g., New_Jersey) were replaced with spaces (New Jersey) using a Python replace function.
- **Column consistency**: Column names were reformatted to avoid spaces and ensure consistency.

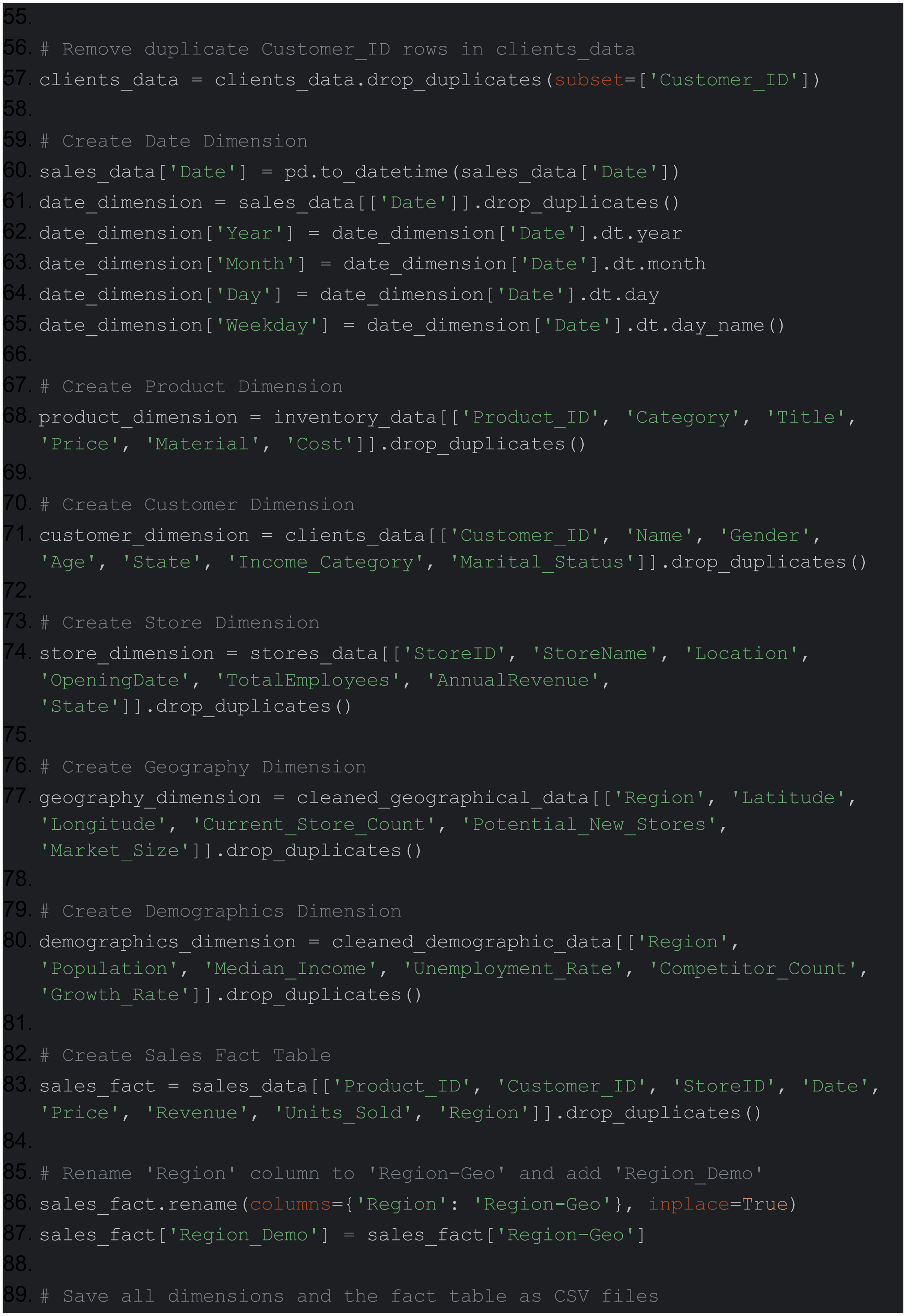
04.03 Schema creation

After transformations, the star schema was created with Python:

- Each dimension and the fact table were saved as separate CSV files.
- SQL files were generated for each table, including CREATE TABLE statements and INSERT INTO commands for loading data into the database.

Python code

```python
import pandas as pd

def clean_data(file_path):
    # Load the data
    data = pd.read_csv(file_path)

    # Drop duplicate rows
    data = data.drop_duplicates()

    # Handle missing values (e.g., fill with median for numeric columns)
    for column in data.columns:
        if data[column].dtype in ['int64', 'float64']:
            data[column] = data[column].fillna(data[column].median())
        else:
            data[column] = data[column].fillna('Unknown')

    # Strip whitespace from column names
    data.columns = data.columns.str.strip()

    # Rename columns for consistency (optional: customize as needed)
    data.rename(columns=lambda x: x.strip().replace(" ", "_"), inplace=True)

    # Ensure numeric columns are correctly typed
    for column in data.select_dtypes(include=['object']).columns:
        try:
            data[column] = pd.to_numeric(data[column])
        except ValueError:
            pass

    # Return the cleaned DataFrame
    return data

# File paths
file_path_geographical = 'C:/Users/21650/Downloads/New_script/Geographical_Data.csv'
file_path_demographic = 'C:/Users/21650/Downloads/New_script/Demographic_Data.csv'

# Clean the datasets
cleaned_geographical_data = clean_data(file_path_geographical)
cleaned_demographic_data = clean_data(file_path_demographic)

# Save cleaned data to new files
cleaned_geographical_data.to_csv('C:/Users/21650/Downloads/New_script_output/Cleaned_Geographical_Data.csv', index=False)
cleaned_demographic_data.to_csv('C:/Users/21650/Downloads/New_script_output/Cleaned_Demographic_Data.csv', index=False)

print("Datasets have been cleaned and saved.")

# Star schema creation
# Load datasets
sales_data = pd.read_csv("C:/Users/21650/Downloads/New_script/Sales.csv")
inventory_data = pd.read_csv("C:/Users/21650/Downloads/New_script/Inventory.csv")
clients_data = pd.read_csv("C:/Users/21650/Downloads/New_script/Clients.csv")
stores_data = pd.read_csv("C:/Users/21650/Downloads/New_script/Stores.csv")
cleaned_geographical_data = pd.read_csv("C:/Users/21650/Downloads/New_script_output/Cleaned_Geographical_Data.csv")
cleaned_demographic_data = pd.read_csv("C:/Users/21650/Downloads/New_script_output/Cleaned_Demographic_Data.csv")
```



```

# Remove duplicate Customer_ID rows in clients_data
clients_data = clients_data.drop_duplicates(subset=['Customer_ID'])

# Create Date Dimension
sales_data['Date'] = pd.to_datetime(sales_data['Date'])
date_dimension = sales_data[['Date']].drop_duplicates()
date_dimension['Year'] = date_dimension['Date'].dt.year
date_dimension['Month'] = date_dimension['Date'].dt.month
date_dimension['Day'] = date_dimension['Date'].dt.day
date_dimension['Weekday'] = date_dimension['Date'].dt.day_name()

# Create Product Dimension
product_dimension = inventory_data[['Product_ID', 'Category', 'Title',
'Price', 'Material', 'Cost']].drop_duplicates()

# Create Customer Dimension
customer_dimension = clients_data[['Customer_ID', 'Name', 'Gender',
'Age', 'State', 'Income_Category', 'Marital_Status']].drop_duplicates()

# Create Store Dimension
store_dimension = stores_data[['StoreID', 'StoreName', 'Location',
'OpeningDate', 'TotalEmployees', 'AnnualRevenue',
'State']].drop_duplicates()

# Create Geography Dimension
geography_dimension = cleaned_geographical_data[['Region', 'Latitude',
'Longitude', 'Current_Store_Count', 'Potential_New_Stores',
'Market_Size']].drop_duplicates()

# Create Demographics Dimension
demographics_dimension = cleaned_demographic_data[['Region',
'Population', 'Median_Income', 'Unemployment_Rate', 'Competitor_Count',
'Growth_Rate']].drop_duplicates()

# Create Sales Fact Table
sales_fact = sales_data[['Product_ID', 'Customer_ID', 'StoreID', 'Date',
'Price', 'Revenue', 'Units_Sold', 'Region']].drop_duplicates()

# Rename 'Region' column to 'Region-Geo' and add 'Region_Demo'
sales_fact.rename(columns={'Region': 'Region-Geo'}, inplace=True)
sales_fact['Region_Demo'] = sales_fact['Region-Geo']

# Save all dimensions and the fact table as CSV files
```

```
output_paths = {
    "Date Dimension": "C:/Users/21650/Downloads/New_script_output/Date_Dimension.csv",
    "Product Dimension": "C:/Users/21650/Downloads/New_script_output/Product_Dimension.csv",
    "Customer Dimension": "C:/Users/21650/Downloads/New_script_output/Customer_Dimension.csv",
    "Store Dimension": "C:/Users/21650/Downloads/New_script_output/Store_Dimension.csv",
    "Geography Dimension": "C:/Users/21650/Downloads/New_script_output/Geography_Dimension.csv",
    "Demographics Dimension": "C:/Users/21650/Downloads/New_script_output/Demographics_Dimension.csv",
    "Sales Fact Table": "C:/Users/21650/Downloads/New_script_output/Sales_Fact_Table.csv"
}

    date_dimension.to_csv(output_paths["Date Dimension"], index=False)
    product_dimension.to_csv(output_paths["Product Dimension"], index=False)
    customer_dimension.to_csv(output_paths["Customer Dimension"], index=False)
    store_dimension.to_csv(output_paths["Store Dimension"], index=False)
    geography_dimension.to_csv(output_paths["Geography Dimension"], index=False)
    demographics_dimension.to_csv(output_paths["Demographics Dimension"], index=False)
    sales_fact.to_csv(output_paths["Sales Fact Table"], index=False)

    print("Star schema created and saved:")
    for name, path in output_paths.items():
        print(f"- {name}: {path}")

    # Save each table as an SQL file
    output_paths_sql = {
        "Date_Dimension.sql": date_dimension,
        "Product_Dimension.sql": product_dimension,
        "Customer_Dimension.sql": customer_dimension,
        "Store_Dimension.sql": store_dimension,
        "Geography_Dimension.sql": geography_dimension,
        "Demographics_Dimension.sql": demographics_dimension,
        "Sales_Fact_Table.sql": sales_fact
    }

```

```
for filename, dataframe in output_paths_sql.items():
    table_name = filename.split('.')[0]
    sql_file_path = f"C:/Users/21650/Downloads/New_script/{filename}"

    with open(sql_file_path, 'w', encoding='utf-8') as sql_file:
        # Write the CREATE TABLE statement
        columns = ', '.join([f'[{col}] TEXT' for col in dataframe.columns])  # Simplified as TEXT
        sql_file.write(f"CREATE TABLE {table_name} ({columns});\n")

        # Write the INSERT INTO statements
        for _, row in dataframe.iterrows():
            values = []
            for value in row.values:
                if isinstance(value, str):
                    value = value.replace("'", "''")  # Escape single quotes for SQL
                    values.append(f"'{value}'")  # Wrap strings in single quotes
                elif pd.isna(value):
                    values.append("NULL")  # Handle missing values
                else:
                    values.append(str(value))  # Convert other values to string

            values_str = ', '.join(values)
            sql_file.write(f"INSERT INTO {table_name} VALUES ({values_str});\n")
```

4.3 Loading

After transforming internal and external raw data into a structured and stable format, we load it into the RDBMS in CSV format:

- Loading type : Full Loading

---

5. Data warehouse creation and data storage

After completing the ETL process, we proceeded to create and populate the data warehouse, a critical step in transforming raw data into a format that supports analytical queries and decision-making. For this purpose, we utilized SQL Developer as our RDBMS, leveraging its robust set of features to streamline the process and optimize performance.

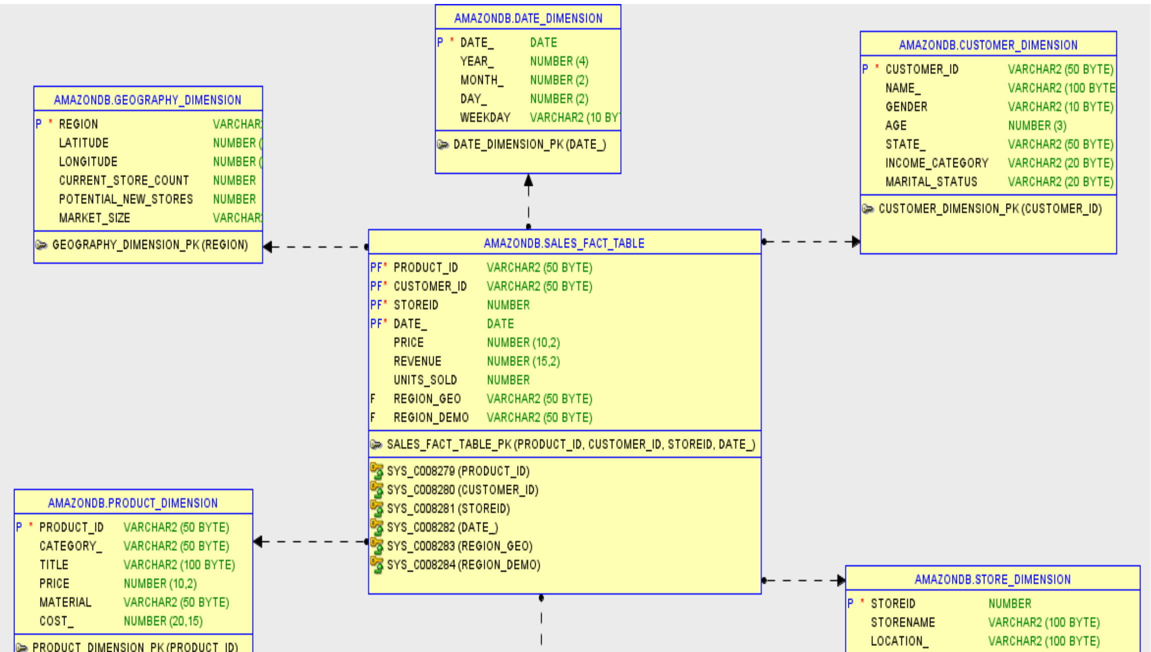
The data warehouse design was based on the **star schema**, a widely recognized and effective model for organizing data. This design consisted of dimension tables and a central fact table, all of which we loaded into the database using CSV files. The use of the star schema ensured a simple yet powerful structure, enabling us to efficiently store data while facilitating intuitive querying and reporting.

We chose SQL Developer for several key reasons. First, it supports **multidimensional modeling**, acting as a ROLAP (Relational Online Analytical Processing) server. This allowed us to build a relational foundation for analyzing large datasets without compromising flexibility. One of the standout features we utilized was its support for **materialized views**, which played a crucial role in enhancing query performance. By precomputing and storing complex aggregations, materialized views significantly reduced the time required to run analytical queries, enabling faster insights.

Throughout this process, we saved and documented our progress to maintain transparency and traceability. The ERD (Entity-Relationship Diagram) of the data model served as a visual representation of the relationships between our tables, providing a clear blueprint for the underlying structure of the data warehouse. Having this documentation not only supported our implementation but also provided a useful reference for future development or troubleshooting.

What made this stage particularly interesting was the tangible shift from handling raw, unorganized data to seeing it transformed into a well-structured system capable of delivering insights. The structured, stable nature of the star schema, combined with the power of SQL Developer, brought our data warehouse to life.

In addition to the technical benefits, working through this process also gave us a deeper appreciation of the role a well-designed data warehouse plays in the overall data pipeline.



AMAZONDB.DATE_DIMENSION

P *	DATE_	DATE
	YEAR_	NUMBER (4)
	MONTH_	NUMBER (2)
	DAY_	NUMBER (2)
	WEEKDAY	VARCHAR2 (10 BY

DATE_DIMENSION_PK (DATE_)

AMAZONDB.CUSTOMER_DIMENSION

P *	CUSTOMER_ID	VARCHAR2 (50 BYTE)
	NAME_	VARCHAR2 (100 BYTE
	GENDER	VARCHAR2 (10 BYTE)
	AGE	NUMBER (3)
	STATE_	VARCHAR2 (50 BYTE)
	INCOME_CATEGORY	VARCHAR2 (20 BYTE)
	MARITAL_STATUS	VARCHAR2 (20 BYTE)

CUSTOMER_DIMENSION_PK (CUSTOMER_ID)

AMAZONDB.GEOGRAPHY_DIMENSION

P *	REGION	VARCHAR
	LATITUDE	NUMBER (
	LONGITUDE	NUMBER (
	CURRENT_STORE_COUNT	NUMBER
	POTENTIAL_NEW_STORES	NUMBER
	MARKET_SIZE	VARCHAR

GEOGRAPHY_DIMENSION_PK (REGION)

AMAZONDB.SALES_FACT_TABLE

PF*	PRODUCT_ID	VARCHAR2 (50 BYTE)
PF*	CUSTOMER_ID	VARCHAR2 (50 BYTE)
PF*	STOREID	NUMBER
PF*	DATE_	DATE
	PRICE	NUMBER (10,2)
	REVENUE	NUMBER (15,2)
	UNITS_SOLD	NUMBER
F	REGION_GEO	VARCHAR2 (50 BYTE)
F	REGION_DEMO	VARCHAR2 (50 BYTE)

SALES_FACT_TABLE_PK (PRODUCT_ID, CUSTOMER_ID, STOREID, DATE_)

SYS_C008279 (PRODUCT_ID)
SYS_C008280 (CUSTOMER_ID)
SYS_C008281 (STOREID)
SYS_C008282 (DATE_)
SYS_C008283 (REGION_GEO)
SYS_C008284 (REGION_DEMO)

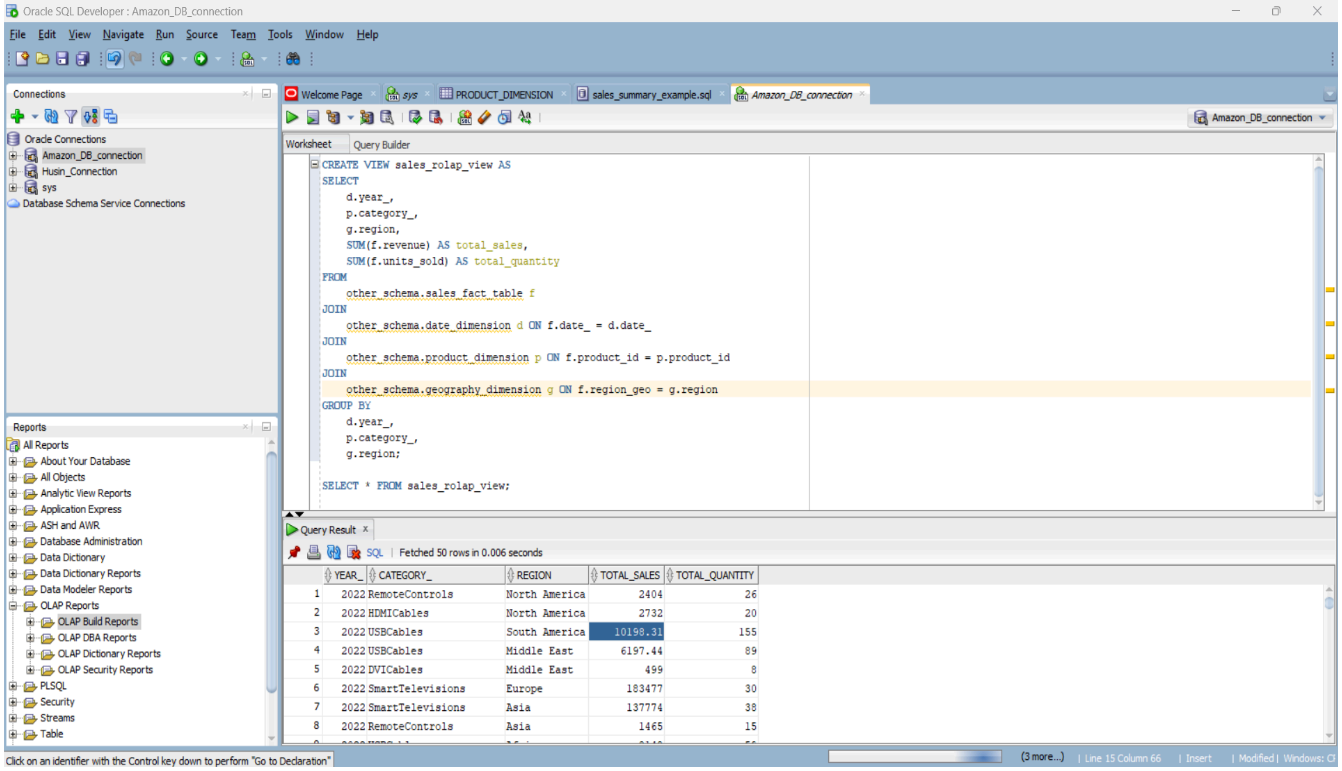
AMAZONDB.PRODUCT_DIMENSION

P *	PRODUCT_ID	VARCHAR2 (50 BYTE)
	CATEGORY_	VARCHAR2 (50 BYTE)
	TITLE	VARCHAR2 (100 BYTE)
	PRICE	NUMBER (10,2)
	MATERIAL	VARCHAR2 (50 BYTE)
	COST_	NUMBER (20,15)

PRODUCT_DIMENSION_PK (PRODUCT_ID)

AMAZONDB.STORE_DIMENSION

P *	STOREID	NUMBER
	STORENAME	VARCHAR2 (100 BYTE)
	LOCATION_	VARCHAR2 (100 BYTE)



```
CREATE VIEW sales_rolap_view AS
SELECT
    d.year_,
    p.category_,
    g.region,
    SUM(f.revenue) AS total_sales,
    SUM(f.units_sold) AS total_quantity
FROM
    other_schema.sales_fact_table f
JOIN
    other_schema.date_dimension d ON f.date_ = d.date_
JOIN
    other_schema.product_dimension p ON f.product_id = p.product_id
JOIN
    other_schema.geography_dimension g ON f.region_geo = g.region
GROUP BY
    d.year_,
    p.category_,
    g.region;

SELECT * FROM sales_rolap_view;
```

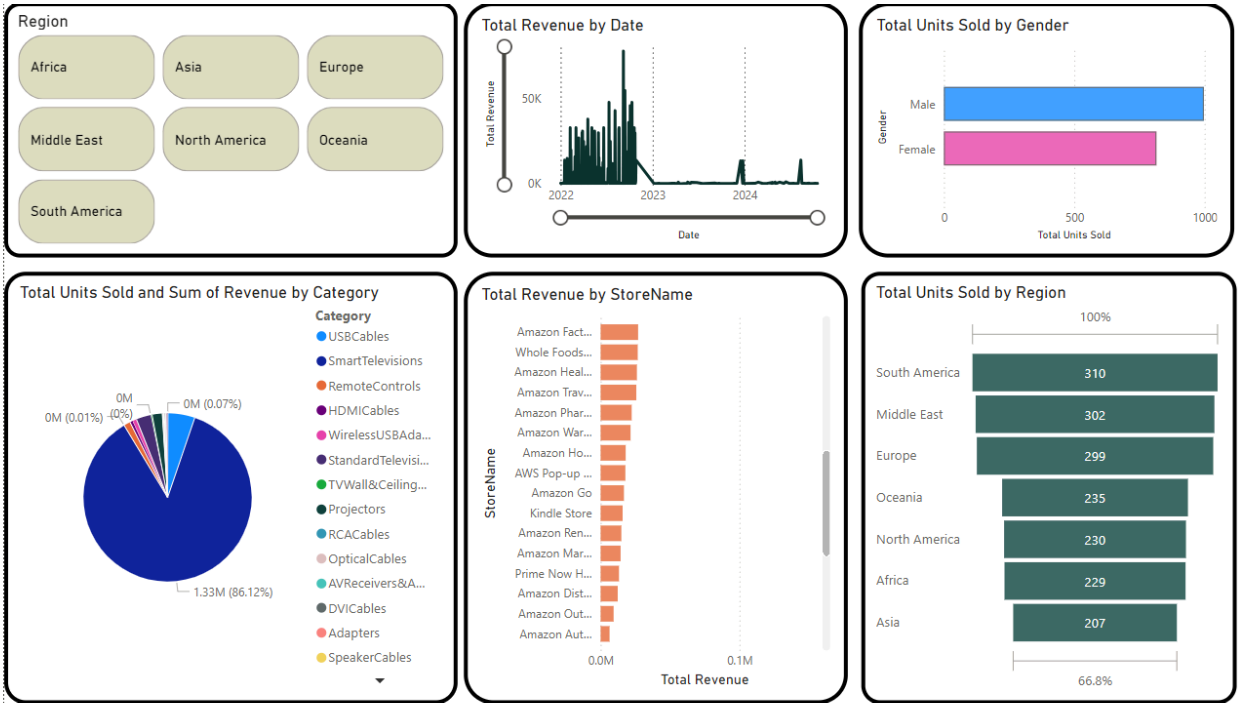
Fetched 50 rows in 0.006 seconds

	YEAR_	CATEGORY_	REGION	TOTAL_SALES	TOTAL_QUANTITY
1	2022	RemoteControls	North America	2404	26
2	2022	HDMICables	North America	2732	20
3	2022	USBCables	South America	10198.31	155
4	2022	USBCables	Middle East	6197.44	89
5	2022	DVICables	Middle East	499	8
6	2022	SmartTelevisions	Europe	183477	30
7	2022	SmartTelevisions	Asia	137774	38
8	2022	RemoteControls	Asia	1465	15

Click on an identifier with the Control key down to perform "Go to Declaration"

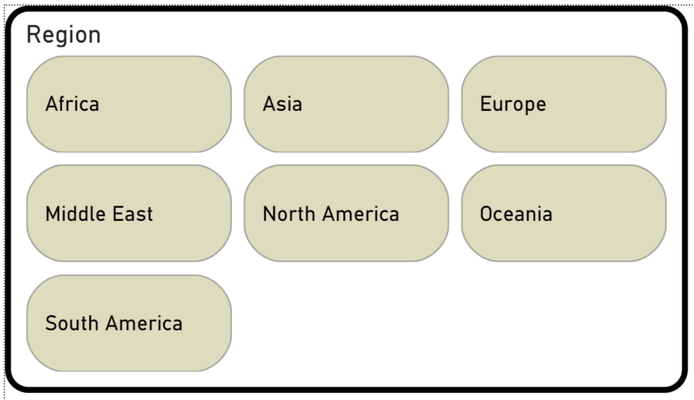
6. Data visualization and analysis

Power BI was employed for creating visualizations and dashboards.



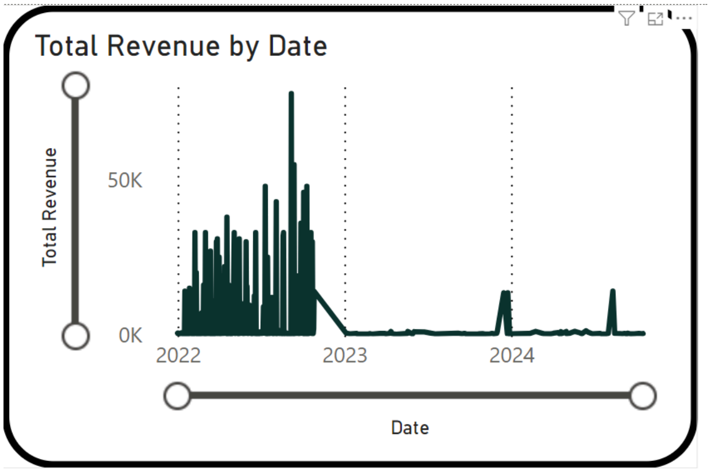
1. Regions overview :

- The dashboard has clickable regions (Africa, Asia, Europe, Middle East, North America, Oceania, and South America).
- This likely serves as a filter to view data specific to each region.



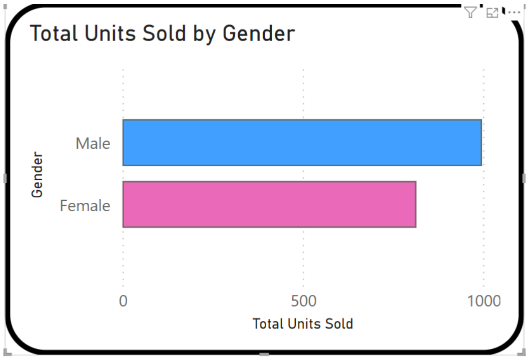
2. Total revenue by date :

- A time-series graph shows revenue trends over time (2022 to 2024).
- There is a notable spike in revenue during certain periods, indicating high-performing months or campaigns.
- There are also dips in revenue, possibly related to seasonality or operational issues.



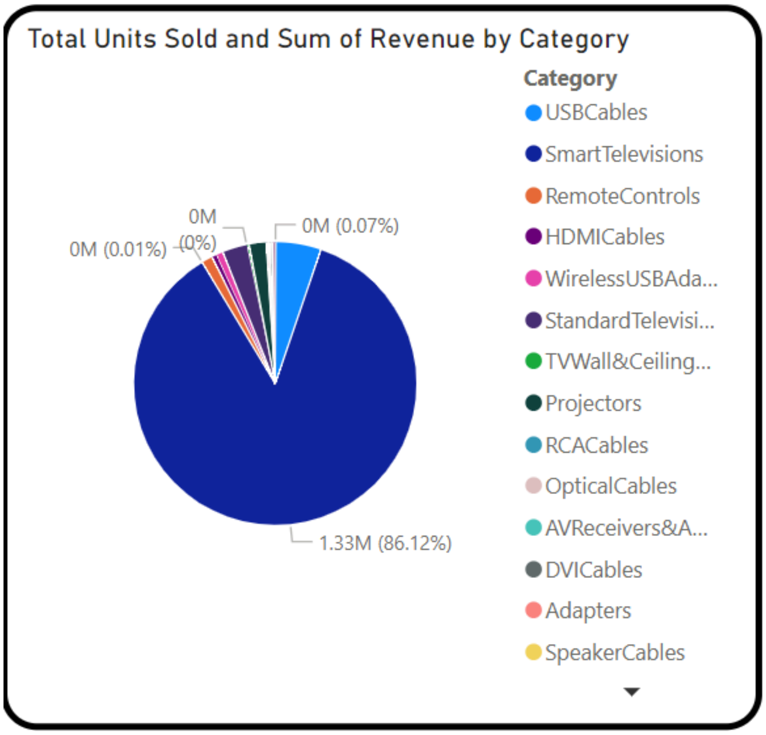
3. Total units sold by gender :

- Males appear to have purchased more units compared to females, indicating higher participation or demand from male customers.



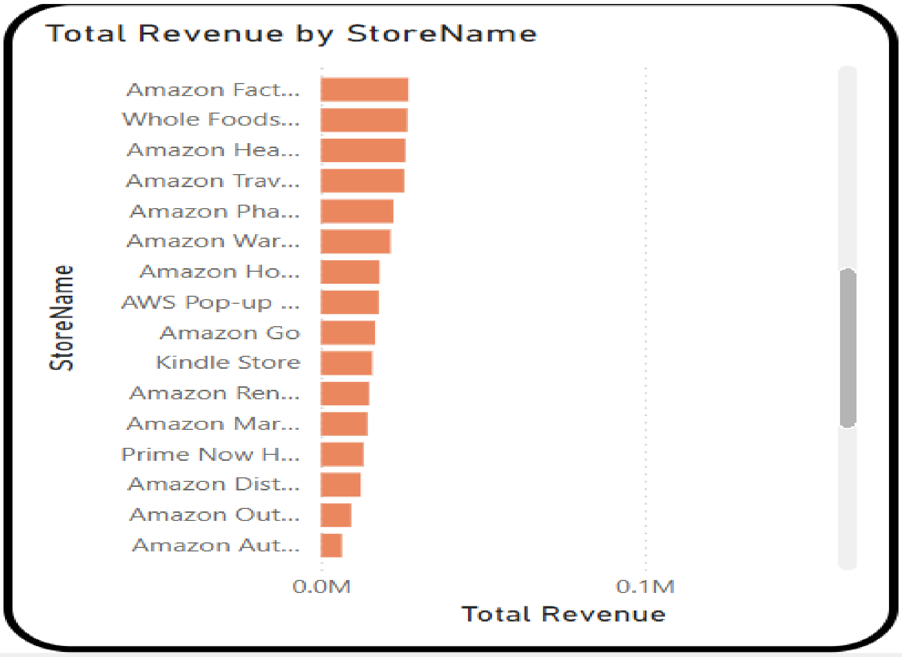
4. Total units sold and revenue by category of products:

- USB cables dominate the sales and revenue share with 86.12% of units sold.
- Other categories contribute minimally (e.g., HDMI cables, remote controls, etc.), suggesting product portfolio imbalance.



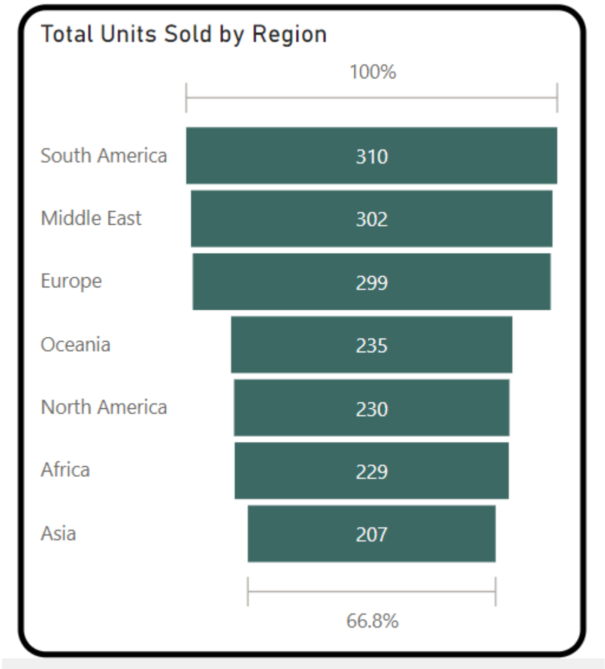
5. Total revenue by store name :

- Amazon Pet Store generates the highest revenue, followed by Amazon Logistics and Amazon Publications.
- Revenue seems diverse across stores, though there is a clear top performer.



6. Total units sold by region :

- South America has the highest units sold (310), followed by the Middle East (302) and Europe (299).
- Asia has the lowest units sold (207), possibly highlighting a lack of market penetration or lower demand.



➢ Key insights:

1. **Top product**: USB cables are a strong driver of both sales and revenue, suggesting their critical importance in the business.
2. **High-performing region**: South America leads in units sold, which could indicate a strong market presence or demand there.
3. **Demographic insights**: Male customers purchase more, indicating potential for targeting female customers through campaigns.
4. **Revenue concentration**: A small number of store names contribute significantly to total revenue; diversification or optimization in lower-performing stores could help.
5. **Seasonality or event impact**: The time-series chart highlights spikes in revenue, which could align with events, promotions, or holidays.

References

1. GitHub repository: https://github.com/houssinekhlif/Amazon_DWH_Project
2. External sources:

https://www.kaggle.com/code/mehakiftikhar/amazon-sales-dataset-eda
https://www.youtube.com/watch?v=0BKlUySopU4&list=PLwIcJx1aSL1SeTJgPbFgf1V-5CfsV4l1l
https://youtu.be/I7DZP4rVQOU?list=PLTsu3dft3CWhOUPyXdLw8DGy_1l2oK1yy

---

Emails:

mouna.saadaoui202@gmail.com

houssine.khlif.7@gmail.com

Thank you for reading