Online_Retail Analysis

Introduction:

This project is tailored to assist a UK-based and registered non-store online retail by developing a data-driven solution that enables insightful analysis and informed decision-making. It aims to provide valuable insights into customer purchasing behavior, product performance, and sales trends across different regions and time periods.

Business Needs:

- Deciding in which country to become a brick and mortar
- Deciding which products to market more

Goals :

- **Data Warehouse Implementation :** Create a robust data warehouse that facilitates efficient storage, retrieval, and analysis of transactional data
- **Enhanced Decision-Making:** Provide actionable insights to stakeholders by analyzing historical sales data.
- **Performance Monitoring:** Track sales performance across various dimensions such as products, customers, and time.

Delivrables:

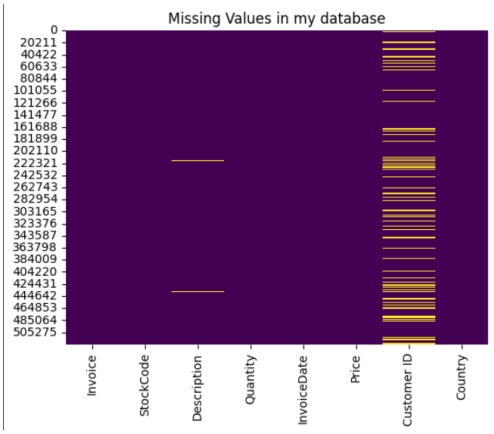
- A comprehensive data model
- Implementation of an ETL process to populate the data warehouse.
- Interactive dashboards using Power BI for visualization.

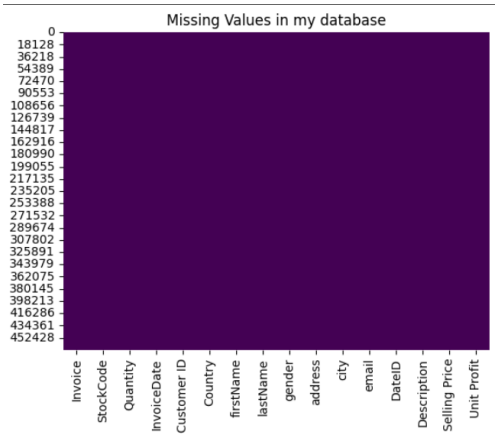
Implemented Phases

1. Data collection and Preprocessing

- Collected raw transactional data from two sources, they include invoices, product details, and customer information.
- Conducted data cleaning by removing inconsistencies, handling missing values, and standardizing formats.

exemple:





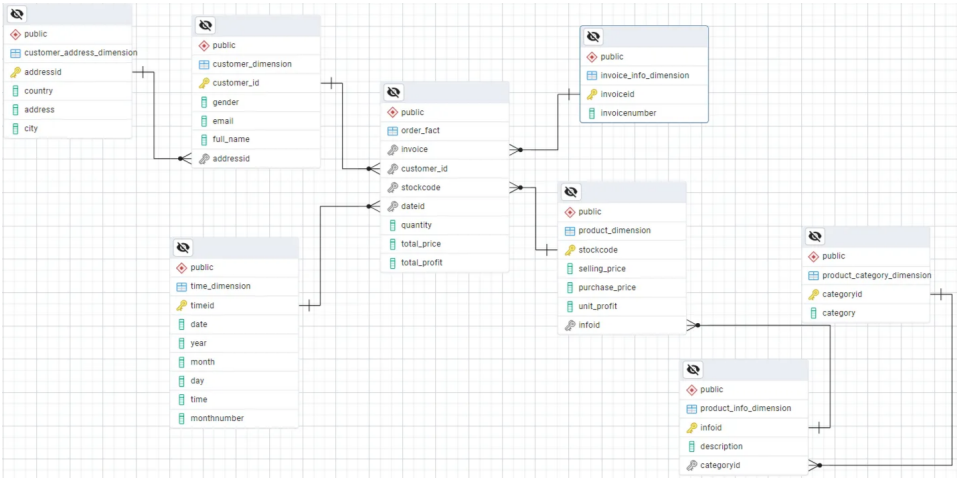
→ As we can see from the heatmap the data is missing values espacially for customer ID . This problem was solved by filling them with unique ID's .

2.ETL Process Implementation:

- ETL process was implemented using python on Colab . There was done:
  - **Extraction:** extracted data from online_retail and customer excel files .
  - **Transormation:** the extracted data was then subjected to the necessery transformations such as data type tranformation , spliting column into multiple column like the date column that was divided into a day , month ,and year column , and data derivation like calculating the total revenue from unit price and quantity
  - **Loading :** loading was done manually by doawnloading the transformed data into a csv file and importing it to postgresql

3.Data Modeling:

- Designed a Snowflake schema consisting of :
  - A central fact table (order_fact)
  - Dimension tables (cusstomer_dim, customer_address_dim, time_dim,invoice_info_dim,product_dim,product_category_dim,product_info_dim).
- Identified key relationships to ensure efficient querying and reporting.



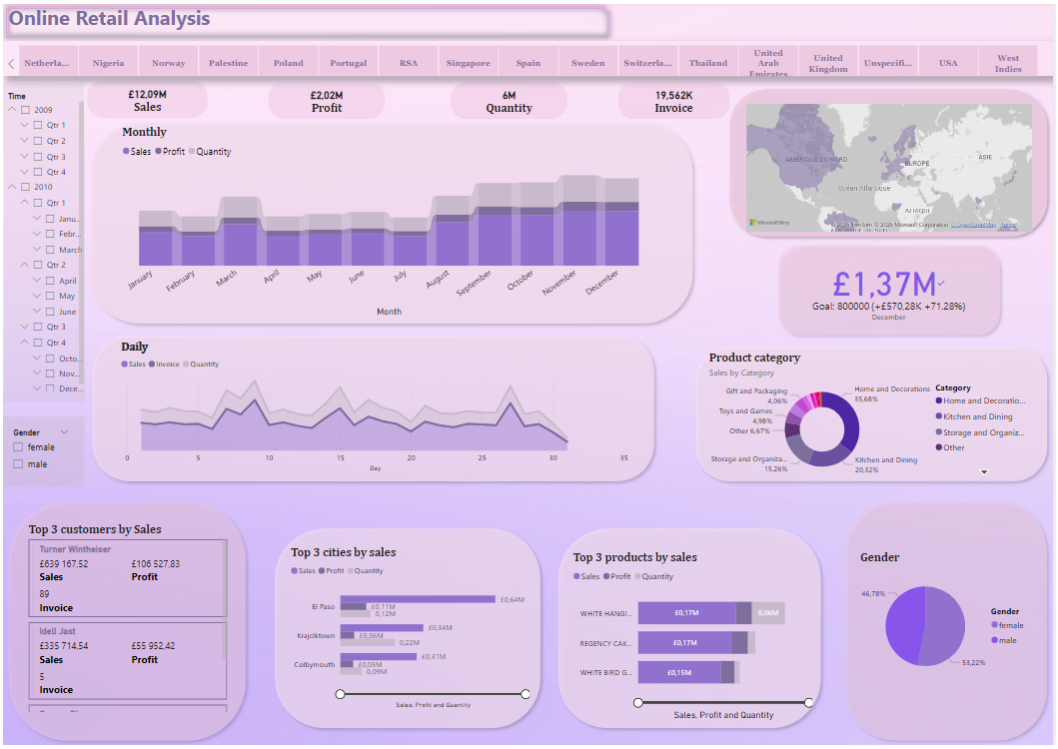
→ We adopted the snowflake schema since there are some dimension tables that can be normalized into several related table such as customer_dimension table.

4.OLAP Implementation:

OLAP (Online Analytical Processing) was implemented using **Power BI** with data sourced from **PostgreSQL**. The chosen approach was **Import Mode**, which follows a MOLAP (Multidimensional OLAP) model by preloading data into Power BI's in-memory engine

5.Dashboard Development:

- Built Power BI dashboards to visualize key performance indicators such as total sales, top-selling products, and top-buying customers.
- Implemented drill-down such for time where we can drill-down to months and to days or rollup to year and filtering capabilities such for gender and country for deeper insights.



Conclusion:

Throughout the project, several challenges were encountered and addressed:

- **Data Quality Issues:** Handling missing and inconsistent data required extensive preprocessing and validation.

Possible Enhancements:

- Automating the ETL process for better scheduling and monitoring.
- Implementing HOLAP (Hybrid OLAP) for a balance between storage efficiency and query performance.
- Enhancing dashboards with predictive analytics using machine learning models.
- Expanding the data warehouse to include social media and customer feedback data for sentiment analysis.

Project code:

https://colab.research.google.com/drive/1s2vekRgxX7fO0korWblz_E0-HfrcDpHp?usp=sharing

Student name : Sabrine Ayadi

(in case link didn’t open here is the code)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import files
uploaded = files.upload()
#Extraction
customeer = pd.read_excel('customers.xlsx')
online_retail = pd.read_excel('onlineretail - Copie.xlsx')
```

```
#Transformation
sns.heatmap(customeer.isnull(), cbar=False, cmap='viridis')
plt.title("Missing Values in my database")
plt.show()
empty_cols_customeer = [col for col in customeer.columns if customeer[col].isnull()
customeer = customeer.drop(columns=empty_cols_customeer)
print(customeer.columns)
sns.heatmap(online_retail.isnull(), cbar=False, cmap='viridis')
plt.title("Missing Values in my database")
plt.show()
# Detecting duplicate rows for customer
customer_duplicates = customeer.duplicated()

#Displaying the duplicate rows
duplicate_rows = customeer[customer_duplicates]
print("Duplicate rows in 'customer' (exact duplicates):\n", duplicate_rows)

# Counting the number of duplicates
num_duplicates = customer_duplicates.sum()
print("\nNumber of duplicate rows:", num_duplicates)
#removing duplicates
customer_no_duplicates = customeer.drop_duplicates()
 # Checking for duplicates in the cleaned DataFrame
customer_check_duplicates = customer_no_duplicates.duplicated()
num_remaining_duplicates = customer_check_duplicates.sum()
print("Number of remaining exact duplicate rows in 'customer':", num_remaining_dupl
#joining customer dataset with online_retail
joined_data = pd.merge(online_retail, customer_no_duplicates, on='Customer ID', how
# Filtering for Negative Quantities
returns_data = joined_data[joined_data['Quantity'] < 0]

#Printing the new dataframe
print(returns_data.head())
non_negative_quantity_mask = joined_data['Quantity'] >= 0

# Filtering the DataFrame using the mask
joined_data = joined_data[non_negative_quantity_mask]

#  Checking to make sure the negative quantities are gone
print(joined_data[joined_data['Quantity'] < 0])
# Selecting object columns
object_cols = joined_data.select_dtypes(include=['object']).columns

# Converting object columns to string
joined_data[object_cols] = joined_data[object_cols].astype('string')

#  Checking the data types
print(joined_data.dtypes)
joined_data['Description'] = joined_data['Description'].str.replace('&', 'and')

# making customer dimension
customer_dimension = joined_data[[
```

```
    'Customer ID', 'Country', 'firstName', 'lastName', 'gender', 'address', 'city',
]]

# Remove=ing duplicates
customer_dimension = customer_dimension.drop_duplicates(subset=['Customer ID'])
customer_dimension.head()
#making all column names start with an uppercase
customer_dimension.columns = customer_dimension.columns.str.capitalize()
#merging first name and last name into a column named full name
customer_dimension['Full Name'] = customer_dimension['Firstname'] + ' ' + customer_
#droping first name and last name
customer_dimension = customer_dimension.drop(columns=['Firstname', 'Lastname'])
customer_dimension.head()
#creating a customer address dimension
customer_address_dimension = customer_dimension[[ 'Country', 'Address', 'City']]
# making an address id column
customer_address_dimension['AddressID'] = range(1, len(customer_address_dimension)
#searching for duplicates
duplicates = customer_address_dimension[customer_address_dimension.duplicated(subse
print(duplicates)
# droping the country adress and city from customer dimension and adding a adressid
customer_dimension = pd.merge(customer_dimension, customer_address_dimension[['Addr
customer_dimension = customer_dimension.drop(columns=['Country', 'Address', 'City']
customer_dimension.head()
customer_address_dimension.head()
#making addressid the first column
customer_address_dimension = customer_address_dimension[['AddressID'] + [col for co
customer_address_dimension.head()
customer_dimension.head()
# searching for email containing ""
email_with_quotes = customer_dimension[customer_dimension['Email'].str.contains('"'
print(email_with_quotes)
# deleting "" and firstname from email
customer_dimension['Email'] = customer_dimension['Email'].str.replace('"', '')
customer_dimension['Email'] = customer_dimension['Email'].str.replace('firstName',
#load dimension to csv file
customer_dimension.to_csv('customer_dimension.csv', index=False)
customer_address_dimension.to_csv('customer_address_dimension.csv', index=False)
# making time dimension
invoice_dates = joined_data['InvoiceDate'].unique()
# Creating a DataFrame with the InvoiceDates as index
time_dimension = pd.DataFrame(index=pd.to_datetime(invoice_dates))

# Adding time-related columns
time_dimension['Year'] = time_dimension.index.year
time_dimension['Month'] = time_dimension.index.month
time_dimension['Day'] = time_dimension.index.day
time_dimension['Time'] = time_dimension.index.time
# Reseting index to have a 'Date' column
time_dimension = time_dimension.reset_index().rename(columns={'index': 'Date'})
# Creating an ID column using a range
time_dimension['TimeID'] = range(1, len(time_dimension) + 1)
```

```
# Reordering columns
time_dimension = time_dimension[['TimeID'] + [col for col in time_dimension.columns
import calendar
time_dimension['Month'] = time_dimension['Month'].apply(lambda x: calendar.month_na
time_dimension.head()
# changing data type of Time column to time
time_dimension['Time'] = pd.to_datetime(time_dimension['Time'], format='%H:%M:%S').
# creating column month number
time_dimension['Month Number'] = time_dimension['Month'].apply(lambda x: list(calen
time_dimension.head()
#loading time dimension to a csv file
time_dimension.to_csv('time_dimension.csv', index=False)
 # making product dimension
product_dimension = joined_data[[
    'StockCode', 'Description', 'Price'
]]
#renaming price to selling price
product_dimension = product_dimension.rename(columns={'Price': 'Selling Price'})

#defining a markup percentage
markup_percentage = 0.20

# Calculating the purchase price using reverse markup
product_dimension['Purchase Price'] = product_dimension['Selling Price'] / (1 + mar
# Calculating the unit profit
product_dimension['Unit Profit'] = product_dimension['Selling Price'] - product_dim
# removing double quotes from description in product dimension

product_dimension['Description'] = product_dimension['Description'].str.replace('"'
# transforming all the letters in stockcode to uppercase in product dimension
product_dimension['StockCode'] = product_dimension['StockCode'].str.upper()
# replacing , in description with nothing
product_dimension['Description'] = product_dimension['Description'].str.replace(','
#replacing & and + with and also replace - with a space
product_dimension['Description'] = product_dimension['Description'].str.replace('&'
product_dimension['Description'] = product_dimension['Description'].str.replace('+'
product_dimension['Description'] = product_dimension['Description'].str.replace('-'

# searching for empty stockcode
empty_stockcode = product_dimension[product_dimension['StockCode'].isnull()]
print(empty_stockcode)
#counting number of duplicates
duplicates = product_dimension[product_dimension.duplicated(subset=['Description'],
print(duplicates)
# Removing duplicates (if any)
product_dimension = product_dimension.drop_duplicates(subset=['Description'])
product_dimension
# searching for duplicates in product dimension in stockcode
duplicates = product_dimension[product_dimension.duplicated(subset=['StockCode'], k
# displaying duplicates ordered by stockcode
duplicates.sort_values(by=['StockCode'])
# removing duplicates in stockcode
```

```
product_dimension = product_dimension.drop_duplicates(subset=['StockCode'])
# Checking for duplicates again
duplicates = product_dimension[product_dimension.duplicated(subset=['StockCode'], k
print(duplicates)
# rounding selling price , purchase price and unit profit to 2 decimals
product_dimension['Selling Price'] = product_dimension['Selling Price'].round(2)
product_dimension['Purchase Price'] = product_dimension['Purchase Price'].round(2)
product_dimension['Unit Profit'] = product_dimension['Unit Profit'].round(2)
product_dimension
# searching for empty values
empty_stockcode = product_dimension[product_dimension['StockCode'].isnull()]
print(empty_stockcode)
# creating a function that will assign the category to the products based on keywor
def categorize_product(description):
    description = str(description).lower()

    if any(keyword in description for keyword in ["gift", "card", "wrap","ribbon"])
        return "Gift and Packaging"
    elif any(keyword in description for keyword in ["decoration","decorative","chan
        return "Home and Decorations"
    elif any(keyword in description for keyword in ["mug","collander","porcelain","
        return "Kitchen and Dining"
    elif any(keyword in description for keyword in ["toy","dominoes","dinosaur","so
        return "Toys and Games"
    elif any(keyword in description for keyword in ["jewelry","necklac", "lariat","
        return "Jewelry"
    elif any(keyword in description for keyword in ["paper", "calculator","calendar
        return "Stationery and Offine supplies"
    elif any(keyword in description for keyword in ["bag","shelving","egg house","b
        return "Storage and Organization"
    elif any(keyword in description for keyword in ["furniture","pouffe", "chair",
        return "Furniture"
    elif any(keyword in description for keyword in ["cat", "dog", "pet", "bird","an
        return "Pet Supplies"
    elif any(keyword in description for keyword in ["gloves","sombrero","bow tie","
        return "Clothes"
    elif any(keyword in description for keyword in ["potting","garden tools","arbor
        return "Gardening"
    elif any(keyword in description for keyword in ["party","bells","stocking","eas
        return "Party and Holidays"
    elif any(keyword in description for keyword in ["bathroom","toilet","tub","bath
        return "Bathroom related"
    elif any (keyword in description for keyword in ["paiting","sewing","your own",
        return "Arts and Crafts"
    elif any(keyword in description for keyword in ["passport","luggage","travel"])
       return "Travel"
    elif any(keyword in description for keyword in ["phone","mobile","thermometer",
      return "Gadgets and accessories"
    elif any(keyword in description for keyword in ["hair","clips"]):
      return "Beauty"
    elif any(keyword in description for keyword in ["essence","oils","matches","inc
      return "Essences"
```

```
    else:
        return "Other"
product_dimension['Category'] = product_dimension['Description'].apply(categorize_p
#making product_infoID in product dimension
product_dimension['infoID'] = range(1, len(product_dimension) + 1)
product_dimension
# Renaming columns for merging
time_dimension = time_dimension.rename(columns={'Date': 'InvoiceDate'})

# Merging to get DateID into joined_data
sales_fact = pd.merge(joined_data, time_dimension[['InvoiceDate', 'TimeID']], on='I
sales_fact = sales_fact.rename(columns={'TimeID': 'DateID'})
# replacing information about product such price and description in sales fact with
sales_fact = pd.merge(sales_fact, product_dimension[['StockCode', 'Description', 'S

sales_fact
# removing description x and price x
sales_fact = sales_fact.drop(columns=['Description_x', 'Price'], errors='ignore')
# renaming description y and price y to description and price
sales_fact = sales_fact.rename(columns={'Description_y': 'Description'})
sales_fact
# searching for empty descritptions
empty_description = sales_fact[sales_fact['Description'].isnull()]
#deleting duplicates
sales_fact = sales_fact.dropna(subset=['Description'])
sales_fact
# heat map
sns.heatmap(sales_fact.isnull(), cbar=False, cmap='viridis')
plt.title("Missing Values in my database")
plt.show()
#making a product_info_dimension with columns info id , description from product di
product_info_dimension = product_dimension[['infoID', 'Description',"Category"]]
# droping description from product dimension
product_dimension = product_dimension.drop(columns=['Description'])
# making a product_category_dimension with categoryID ,  category name from product
product_category_dimension = product_dimension[['Category']]
#making categoryID for product category dimension
product_category_dimension['CategoryID'] = range(1, len(product_category_dimension)
# reordering columns
product_category_dimension = product_category_dimension[['CategoryID', 'Category']]
#removing duplicates from product category dimension based on category
product_category_dimension = product_category_dimension.drop_duplicates(subset=['Ca
# droping category from product dimension
product_dimension = product_dimension.drop(columns=['Category'])
# merging product info dimension and product category dimension
product_info_dimension = pd.merge(product_info_dimension, product_category_dimensio
#droping category
product_info_dimension = product_info_dimension.drop(columns=['Category'])
product_info_dimension
#loading to csv
product_dimension.to_csv('product_dimension.csv', index=False)
product_info_dimension.to_csv('product_info_dimension.csv', index=False)
```

```
product_category_dimension.to_csv('product_category_dimension.csv', index=False)
# making a total sales column
sales_fact['Total_Price'] = sales_fact['Quantity'] * sales_fact['Selling Price']
#making total profit column
sales_fact['Total_Profit'] = sales_fact['Quantity'] * sales_fact['Unit Profit']
#changing invoice type to integer
sales_fact['Invoice'] = sales_fact['Invoice'].astype(int)
sales_fact = sales_fact[['Invoice', 'Customer ID', 'StockCode', 'DateID', 'Quantity
sales_fact.head()
sales_fact.info()
#changing type of invoice to integer
sales_fact['Invoice'] = pd.to_numeric(sales_fact['Invoice'], errors='coerce')
sales_fact.info()
#rounding the totalprice to 2 decimals and total profit
sales_fact['Total_Price'] = sales_fact['Total_Price'].round(2)
sales_fact['Total_Profit'] = sales_fact['Total_Profit'].round(2)

# renaming sales_fact to order_fact
order_fact = sales_fact
order_fact
#loading to csv
order_fact.to_csv('order_fact.csv', index=False)
# making an invoice info dimension by getting the invoice from oder fact
invoice_info_dimension = order_fact[['Invoice']]
#renaming invoice to invoiceid
invoice_info_dimension = invoice_info_dimension.rename(columns={'Invoice': 'Invoice
#removing duplicates
invoice_info_dimension = invoice_info_dimension.drop_duplicates(subset=['InvoiceID'
# making an invoice number column
invoice_info_dimension['InvoiceNumber'] = range(1, len(invoice_info_dimension) + 1)
invoice_info_dimension
#loading to csv
invoice_info_dimension.to_csv('invoice_info_dimension.csv', index=False)
```