GLOBAL MART SALES DATA ANALYSIS

1. Introduction & Motivation

• In the era of data-driven business, retail businesses generate large volumes of order data that, once analyzed, can provide actionable insights about sales trends, customer behavior, product performance, and operational efficiency.

• The goal of this project is to analyze the **Retail Orders** dataset to discover patterns, key metrics (KPIs), and insights that can help decision-makers — e.g. what products to promote, how discounting affects profit, which regions are top performing, seasonality, etc.

• This kind of analysis is valuable for retail managers, business analysts, supply chain planners, and marketing teams.

2. Dataset Description

Source:

- The dataset is available on Kaggle under the name *Retail Orders* by ankitbansal06. kaggle.com
- It contains a single file orders.csv (approx. 1.35 MB) with order-level records.

Columns / Features

Typical fields (after inspecting the dataset and other projects) include:

- order_id
- order_date
- ship_mode
- segment
- country, city, state, postal_code, region
- category, sub_category
- product_id
- cost_price
- list_price
- quantity
- discount_percent

From these, I derived the additional metrics like:

- discount_amount
- sale_price
- Profit = (sale_price – cost_price)

And dropped the unused raw columns.

3. Data Cleaning & Preprocessing

First, I downloaded the dataset from Kaggle using API.

```
pip install kaggle
```

```
import kaggle
```

```
# download dataset using kaggle api
!kaggle datasets download ankitbansal06/retail-orders -f orders.csv
```

```
Dataset URL: https://www.kaggle.com/datasets/ankitbansal06/retail-orders
License(s): CC0-1.0
Downloading orders.csv to c:\Users\gauta\Desktop\Projects\pyhton+sql project


  0%|          | 0.00/200k [00:00<?, ?B/s]
100%|██████████| 200k/200k [00:00<00:00, 199MB/s]
```

Then, I Unzipped the file

```
# if file is Zip then this is how we can unzip
import zipfile
zip_ref = zipfile.ZipFile('orders.csv.zip')
zip_ref.extractall() #exctract file to same location
zip_ref.close() #close file
```

Data Cleaning:

- Handel Null values

```
# read data from the file and handle null values
import pandas as pd
df = pd.read_csv('orders.csv',na_values=['Not Available','unknown'])
df.head(20)
df['Ship Mode'].unique()
df.columns = df.columns.str.strip().str.replace(' ','_').str.lower()
df.columns
```

```
Index(['order_id', 'order_date', 'ship_mode', 'segment', 'country', 'city',
       'state', 'postal_code', 'region', 'category', 'sub_category',
       'product_id', 'cost_price', 'list_price', 'quantity',
       'discount_percent'],
      dtype='object')
```

- Derive new columns discount amount, sale price, profit

```
# derive new columns discount, sale price and profit
df['discount'] = df['list_price']*df['discount_percent']*.01
df.head(10)
df['sale_price'] = df['list_price']-df['discount']
df.head(5)
df['profit'] = df['sale_price']-df['cost_price']
df.head(5)
```

- Type conversion order_date (object to datetime)

```
# convert order date from object data type to datetime
df['order_date'] = pd.to_datetime(df['order_date'])
df.dtypes
df['order_date']
```

- Drop unused columns like cost price, list price, discount percent

```
# drop cost price list price and discount percent columns
df.drop(columns= ['list_price','cost_price', 'discount_percent'], inplace=True)
```

4. Data Storage & Pipeline (ETL)

Once cleaned, I loaded the data into relation database -> PostgreSQL

Used Python (pandas + SQLAlchemy) to push the data to the SQL database.

```python
from sqlalchemy import create_engine
```

```python
# Load the data into sql server
username = "postgres"
password = "admin1234"
host = "localhost"
port = "5432"
database = "orders_data"

engine = create_engine(f"postgresql+psycopg2://{username}:{password}@{host}:{port}/{database}")

# load dataframe into postgresSQL

table_name = "orders"
df.to_sql(table_name,engine,if_exists="append",index=False)
print(f"Data Successfully loaded into table'{table_name}' in database '{database}'.")
```

```
Data Successfully loaded into table'orders' in database 'orders_data'.
```

Defined an appropriate table schema with proper data types (e.g. date, varchar, int, decimal)

```sql
create table df_orders (
      order_id int primary key,
      order_date date,
      ship_mode varchar(20),
      segment varchar(20),
      country varchar(20),
      city varchar(20),
      state varchar(20),
      postal_code varchar(20),
      region varchar(20),
      category varchar(20),
      sub_category varchar(20),
      product_id varchar(50),
      quantity int,
      discount decimal(7,2),
      sale_price decimal(7,2),
      profit decimnal(7,2)
)
```

5. Key Analytical Questions / Insights

1. Total Number of Orders:

```
SELECT COUNT(DISTINCT(order_id)) AS total_orders
FROM orders
```

	total_orders bigint
1	9994

2. Total sales, profit and quantity

```
SELECT ROUND(SUM(sale_price * quantity),2) as Total_Sales,
       ROUND(SUM(profit),2) as Total_Profit,
       SUM(quantity) as Total_Quantity
FROM orders
```

	total_sales numeric	total_profit numeric	total_quantity bigint
1	11079328.20	205168.70	37873

3. Sales per Region

```
SELECT region as Regions,
ROUND(SUM(sale_price * quantity),2) as Total_Sales
FROM orders
GROUP BY region
ORDER BY Total_sales DESC
```

	regions character varying (20)	total_sales numeric
1	West	3467409.60
2	East	3257983.80
3	Central	2387881.20
4	South	1966053.60

5. Top 10 products by total sales

```
SELECT DISTINCT(product_id) AS Products, SUM(sale_price * quantity) as Total_Sales
FROM orders
GROUP BY Products
ORDER BY Total_Sales DESC LIMIT 10
```

	products character varying (50)	total_sales numeric
1	TEC-CO-10004722	245056.00
2	OFF-BI-10000545	163777.70
3	TEC-MA-10002412	130406.40
4	FUR-CH-10002024	120090.70
5	TEC-PH-10001459	113041.90
6	TEC-CO-10001449	107388.00
7	OFF-BI-10003527	97082.90
8	TEC-MA-10000822	89622.30
9	FUR-BO-10002213	84014.80
10	TEC-MA-10001047	81549.00

5. Orders Per Year

```
SELECT EXTRACT(YEAR FROM order_date) AS Order_Year, COUNT(order_id) AS Total_Orders
FROM orders
GROUP BY Order_Year
ORDER BY Total_Orders DESC
```

	order_year numeric	total_orders bigint
1	2022	5037
2	2023	4957

6. Monthly Sales and Profit Trend

```
SELECT EXTRACT(MONTH FROM order_date) AS Order_Month,
ROUND(SUM(sale_price * quantity),2) AS Total_Sales,
ROUND(SUM(profit),2) AS Total_Profit
FROM orders
GROUP BY Order_Month
ORDER BY Order_Month
```

	order_month numeric	total_sales numeric	total_profit numeric
1	1	872196.80	17365.10
2	2	1175649.90	20655.20
3	3	787157.10	14718.30
4	4	1019632.40	18830.20
5	5	824333.40	15276.20
6	6	794239.30	14487.00
7	7	797812.10	15586.00
8	8	999572.70	20121.60
9	9	854507.50	13570.80
10	10	1228206.10	22254.20
11	11	786750.20	14288.10
12	12	939270.70	18016.00

7. Average Discount and Profit by category

```sql
SELECT category, ROUND(AVG(discount),2) AS Average_Discount,
ROUND(AVG(profit),2) AS Average_Profit
FROM orders
GROUP BY category
ORDER BY Average_Profit DESC
```

	category character varying (20)	average_discount numeric	average_profit numeric
1	Technology	15.89	41.38
2	Furniture	12.28	31.34
3	Office Supplies	4.14	10.33

8. Top 5 sub-categories by profit in each region

```sql
WITH ranked_profit AS (
    SELECT
        region,
        sub_category,
        ROUND(SUM(profit), 2) AS total_profit,
        ROW_NUMBER() OVER (PARTITION BY region ORDER BY SUM(profit) DESC) AS rank
    FROM orders
    GROUP BY region, sub_category
)
SELECT
    region,
    sub_category,
    total_profit
FROM ranked_profit
WHERE rank <= 5
ORDER BY region, total_profit DESC;
```

	region character varying (20)	sub_category character varying (20)	total_profit numeric
1	Central	Chairs	8191.70
2	Central	Phones	6168.60
3	Central	Binders	5122.90
4	Central	Copiers	4348.40
5	Central	Storage	4141.70
6	East	Phones	8904.80
7	East	Chairs	8652.60
8	East	Storage	6587.60
9	East	Machines	5655.20
10	East	Copiers	5008.00
11	South	Machines	6151.00
12	South	Phones	4985.60
13	South	Tables	4331.30
14	South	Binders	4279.40
15	South	Chairs	4261.70
16	West	Chairs	8709.10
17	West	Phones	8309.30
18	West	Tables	7343.60
19	West	Binders	5772.50
20	West	Accessories	5579.40

9. Sales contribution by customer segment

```sql
SELECT segment, ROUND(SUM(sale_price * quantity),2) AS Total_Sales,
ROUND(100 * SUM(sale_price * quantity)/(SELECT SUM(sale_price * quantity) FROM orders),2) AS pct_contribution
FROM orders
GROUP BY segment
ORDER BY Total_Sales
```

	segment character varying (20)	total_sales numeric	pct_contribution numeric
1	Home Office	2076552.00	18.74
2	Corporate	3394916.10	30.64
3	Consumer	5607860.10	50.62

10. Discount vs Profit correlation data

```sql
SELECT discount, ROUND(AVG(profit), 2) AS avg_profit
FROM orders
GROUP BY discount
ORDER BY discount DESC;
```

	discount numeric (7,2)	avg_profit numeric
1	905.60	3624.40
2	560.00	420.00
3	525.00	1855.00
4	494.50	985.50
5	409.50	410.50
6	326.40	983.60
7	273.00	1187.00
8	260.00	420.00
9	254.00	766.00
10	252.00	838.00
11	245.00	295.00
12	224.00	676.00
13	218.50	521.50

6. Dashboard in Power BI