Sales Data Analysis using Python and SQL

Introduction

In this project, I analyzed sales data using **SQL and Python** to extract valuable insights related to revenue, profit, customer trends, and product performance. The dataset contained information on orders, including order details, shipping methods, product categories, sales amount, and profit.

Objectives

- Perform **data extraction and transformation** using SQL.
- Conduct **advanced SQL queries** for business insights.
- Identify **trends in sales, profit margins, and discounts**.
- Create meaningful **data visualizations**.
- Optimize queries for **better performance**.

Technology Stack

- **Database:** PostgreSQL
- **Programming Language:** Python
- **Libraries Used:** Pandas, SQLAlchemy

Dataset Overview

The dataset contains the following columns:

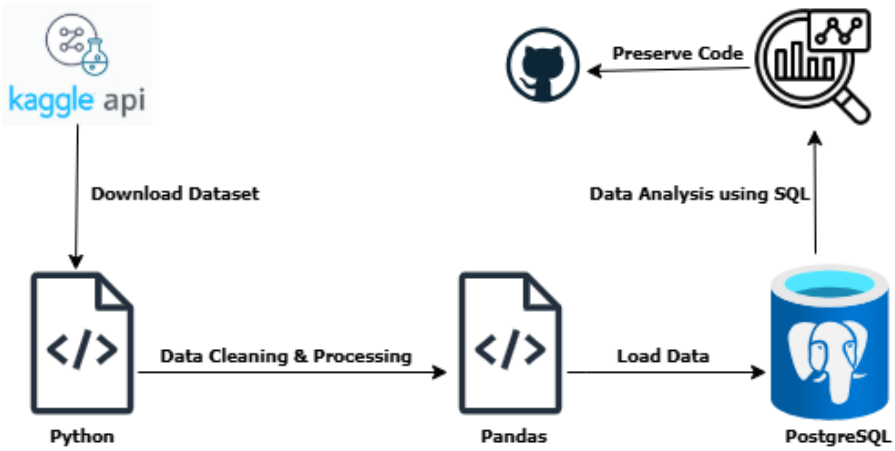
- **order_id** – Unique identifier for each order.
- **order_date** – Date of the order.
- **ship_mode** – Mode of shipment.
- **segment** – Customer segment.
- **country, city, state, region** – Geographic data.
- **category, sub_category** – Product category information.
- **product_id** – Unique product identifier.
- **quantity** – Number of units sold.
- **discount** – Discount applied to the order.
- **sale_price** – Total sales amount.
- **profit** – Profit earned per order.

Methodology

The project followed a structured approach:

1. **Data Collection:** Extracted sales data from a Kaggle using Kaggle API.
2. **Data Cleaning and Transformation:** Processed data using SQL and Python to ensure accuracy.
3. **Exploratory Data Analysis (EDA):** Performed SQL queries to identify trends and patterns.
4. **Business Insights:** Generated key metrics related to revenue, discounts, and customer behavior.
5. **Visualization and Reporting:** Summarized findings through visualizations.

Below is a flowchart outlining the workflow of this project:



SQL Queries and Analysis

Beginner Level Analysis:

1. Retrieve all records from the dataset.

```
SELECT * FROM df_orders;
```

2. Find the total number of orders placed.

```
SELECT COUNT(*) AS total_orders FROM df_orders;
```

3. List unique ship modes available in the dataset.

```
SELECT DISTINCT(ship_mode) FROM df_orders;
```

4. Retrieve all orders where the quantity ordered is more than 5.

```
SELECT * FROM df_orders WHERE quantity > 5;
```

5. Find the earliest and latest order dates.

SELECT MIN(order_date) AS first_order, MAX(order_date) AS last_order FROM df_orders;

6. What is the **total sales amount** for all orders?

SELECT SUM(sale_price) AS total_sales FROM df_orders;

7. What is the **total profit amount** for all orders?

SELECT SUM(profit) AS total_profit FROM df_orders;

8. Calculate the **average discount** offered.

SELECT AVG(discount) AS avg_discount FROM df_orders;

9. Find the **total quantity sold per sub-category** and rank them.

SELECT sub_category, SUM(quantity) AS total_quantity, RANK() OVER(ORDER BY SUM(quantity) DESC) AS rank FROM df_orders GROUP BY sub_category;

10. Determine the **average profit margin (profit/sale_price) for each category**, rounded to 2 decimal places.

SELECT category, ROUND(SUM(profit) / SUM(sale_price) * 100, 2) AS profit_margin FROM df_orders GROUP BY category;

Intermediate Level Analysis:

11. Find the **total sales and profit per category**.

SELECT category, SUM(sale_price) AS total_sales, SUM(profit) AS total_profit FROM df_orders GROUP BY category;

12. List the **top 5 most profitable products**.

SELECT product_id, SUM(profit) AS total_profit FROM df_orders GROUP BY product_id ORDER BY total_profit DESC LIMIT 5;

13. Retrieve **monthly sales trends**.

SELECT EXTRACT(MONTH FROM order_date) AS month, SUM(sale_price) AS total_sales FROM df_orders GROUP BY month;

14. Find **total orders placed in each region** and sort them in descending order.

SELECT region, COUNT(*) AS total_orders FROM df_orders GROUP BY region ORDER BY total_orders DESC;

15. Identify the **top 3 cities with the highest total sales**.

SELECT city, SUM(sale_price) AS total_sales FROM df_orders GROUP BY city ORDER BY total_sales DESC LIMIT 3;

16. Calculate the **average discount offered per category**.

SELECT category, AVG(discount) AS avg_discount FROM df_orders GROUP BY category;

17. Find the **percentage of orders that received a discount**.

SELECT ROUND(COUNT(CASE WHEN discount > 0 THEN 1 END) * 100.0 / COUNT(*), 2) AS discount_percentage FROM df_orders;

18. Find the **highest-selling product in each region**.

WITH cte AS (SELECT region, product_id, SUM(sale_price) AS sales FROM df_orders GROUP BY region, product_id) SELECT * FROM (SELECT *, ROW_NUMBER() OVER(PARTITION BY region ORDER BY sales DESC) AS rn FROM cte) A WHERE rn <= 5;

19. Find **yearly sales growth** by comparing total sales year-over-year.

WITH cte AS (SELECT EXTRACT(YEAR FROM order_date) AS order_year, SUM(sale_price) AS sales FROM df_orders GROUP BY order_year) SELECT order_year, sales, ROUND((sales - LAG(sales) OVER(ORDER BY order_year)) * 100.0 / LAG(sales) OVER(ORDER BY order_year), 2) AS sales_growth FROM cte;

20. Determine the **most profitable shipping mode**.

SELECT ship_mode, SUM(profit) AS total_profit FROM df_orders GROUP BY ship_mode ORDER BY total_profit DESC;

Advanced Level Analysis:

21. **Identify loss-making orders where profit is negative and analyze which categories have the most losses.**

SELECT * FROM df_orders

WHERE profit < 0;

SELECT category, SUM(profit) AS Profit

FROM df_orders

WHERE profit < 0

GROUP BY category

ORDER BY Profit DESC

LIMIT 1;

22. **Determine the profit margin (profit/sale_price) for each product and rank them.**

```
SELECT category,
    ROUND((SUM(profit) / SUM(sale_price) * 100)::numeric, 2) AS
profit_margin_percentage,
    RANK() OVER (ORDER BY SUM(profit) / SUM(sale_price) DESC) AS rank
FROM df_orders
GROUP BY category;
```

23. **Find the highest-selling product in each category.**

```
SELECT category, product_id, SUM(quantity) AS total_quantity
FROM df_orders
GROUP BY category, product_id
HAVING SUM(quantity) = (
  SELECT MAX(total_quantity)
  FROM (SELECT category, product_id, SUM(quantity) AS total_quantity
      FROM df_orders
      GROUP BY category, product_id) subquery
  WHERE subquery.category = df_orders.category
);
```

24. **Analyze which region has the highest discount rates and its impact on profit.**

```
SELECT region,
    ROUND(AVG(discount)::numeric, 2) AS avg_discount,
    ROUND(SUM(profit)::numeric, 2) AS total_profit
FROM df_orders
GROUP BY region
ORDER BY avg_discount DESC;
```

25. **Find month-over-month growth comparison for 2022 and 2023 sales (e.g., Jan 2022 vs. Jan 2023).**

```
WITH cte AS (
  SELECT
```

```
    EXTRACT(YEAR FROM order_date) AS order_year,
    EXTRACT(MONTH FROM order_date) AS order_month,
    SUM(sale_price) AS sales
  FROM df_orders
  GROUP BY EXTRACT(YEAR FROM order_date), EXTRACT(MONTH FROM order_date)
)
SELECT
  order_month,
  SUM(CASE WHEN order_year = 2022 THEN sales ELSE 0 END) AS sales_2022,
  SUM(CASE WHEN order_year = 2023 THEN sales ELSE 0 END) AS sales_2023
FROM cte
GROUP BY order_month
ORDER BY order_month;
```

26. **Find yearly sales growth by comparing total sales year-over-year.**

```
SELECT
  EXTRACT(YEAR FROM order_date) AS year,
  SUM(sale_price) AS total_sales,
  LAG(SUM(sale_price)) OVER (ORDER BY EXTRACT(YEAR FROM order_date)) AS previous_year_sales,
  ROUND(((SUM(sale_price) - LAG(SUM(sale_price)) OVER (ORDER BY EXTRACT(YEAR FROM order_date))) /
    NULLIF(LAG(SUM(sale_price)) OVER (ORDER BY EXTRACT(YEAR FROM order_date)), 0) * 100)::NUMERIC, 2)
    AS sales_growth_percentage
FROM df_orders
GROUP BY EXTRACT(YEAR FROM order_date)
ORDER BY year;
```

27. **Find top 5 highest-selling products in each region.**

```
WITH cte AS (
```

```
    SELECT region, product_id, SUM(sale_price) AS sales
    FROM df_orders
    GROUP BY region, product_id
)
SELECT * FROM (
    SELECT *, ROW_NUMBER() OVER(PARTITION BY region ORDER BY sales DESC) AS rn
    FROM cte
) A
WHERE rn <= 5;
```

28. **Find the average number of orders placed per customer per month (if customer data is available).**

29. **Identify the most profitable shipping mode based on total profit.**

```
SELECT ship_mode,
    SUM(profit) AS total_profit
FROM df_orders
GROUP BY ship_mode
ORDER BY total_profit DESC
LIMIT 1;
```

30. **Determine the sales contribution of each category as a percentage of total sales.**

```
SELECT category,
    ROUND((SUM(sale_price) / (SELECT SUM(sale_price) FROM df_orders) * 100)::numeric,
2) AS sales_contribution_percentage
FROM df_orders
GROUP BY category
ORDER BY sales_contribution_percentage DESC;
```

Key Insights

- The **most profitable product categories** were identified.
- **Regions with high discounts** and their impact on profits were analyzed.

- **Monthly sales trends** revealed peak seasons for sales.
- High-quantity orders were examined to understand bulk purchase behavior.

Conclusion

This project provided hands-on experience in **data extraction, transformation, and analysis** using SQL and Python. The insights derived from the sales data can help businesses make **data-driven decisions** to optimize sales strategies and maximize profitability.

Acknowledgment: Special thanks to **Ankit Bansal** for his valuable guidance in this project.

Next Steps

- Integrate Power BI or Tableau for enhanced **visual analytics**.
- Use **machine learning models** to predict sales trends.
- Automate data processing using Python scripts.