

TASK 6: SALES TREND ANALYSIS USING AGGREGATIONS

1. OBJECTIVE

The goal of this project is to analyze **monthly revenue** and **order volume** from the given orders.csv dataset.

- **Revenue** is calculated as:

$$Revenue = (List\ Price * Quantity) * (1 - \frac{Discount\ Percent}{100})$$

- **Order Volume** is calculated as the **number of unique orders per month**.

We use **PostgreSQL** to implement queries that summarize this information.

2. DATA PREPARATION

2.1 Create Table

We created a table called orders to store the dataset.

Column names were kept in quotes because the dataset contained **spaces and capital letters**.

```
CREATE TABLE orders (  
  "Order Id" INT,  
  "Order Date" DATE,  
  "Ship Mode" VARCHAR(50),  
  "Segment" VARCHAR(50),  
  "Country" VARCHAR(100),  
  "City" VARCHAR(100),  
  "State" VARCHAR(100),  
  "Postal Code" INT,  
  "Region" VARCHAR(50),  
  "Category" VARCHAR(50),  
  "Sub Category" VARCHAR(50),  
  "Product Id" VARCHAR(50),  
  "cost price" INT,
```

"List Price" INT,
"Quantity" INT,
"Discount Percent" INT
);

Explanation:

- The table schema matches the dataset structure.
- "Order Date" is stored as DATE for easier filtering.
- "List Price", "Quantity", and "Discount Percent" are numeric because we need them in calculations.

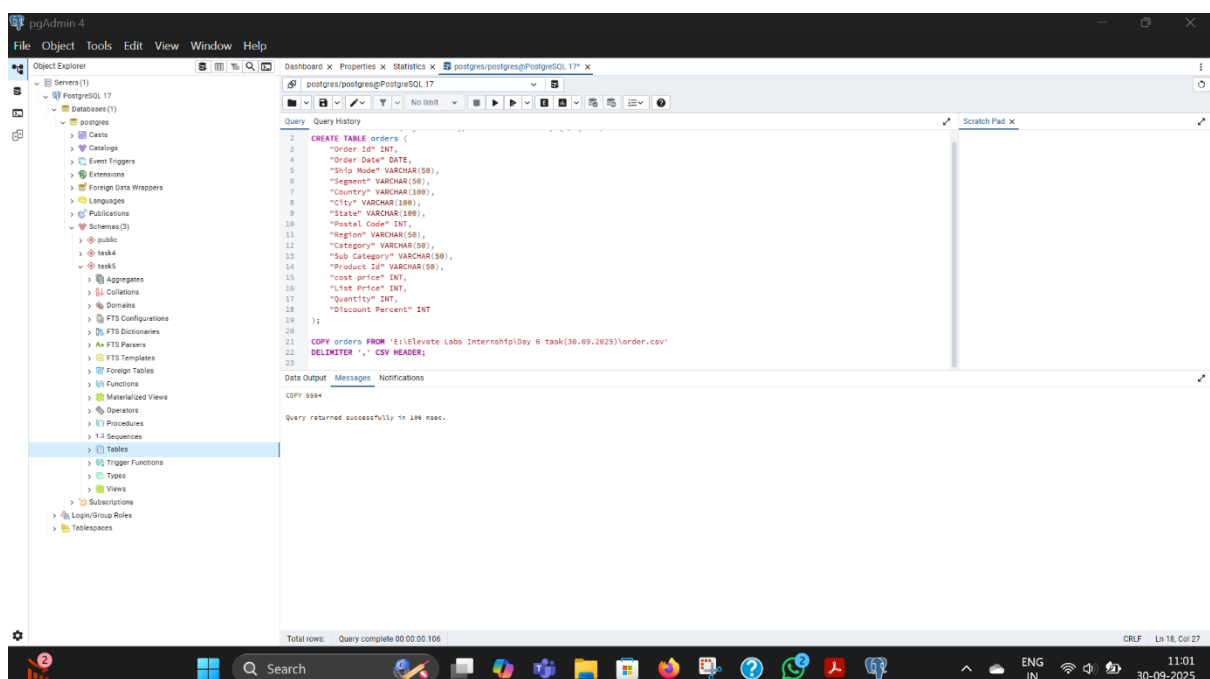
2.2 Load Data

- Depending on the database, different commands are used:

`COPY orders FROM '/path/to/orders.csv' DELIMITER ',' CSV HEADER;`

Explanation:

- These commands load the CSV into the table.
- `IGNORE 1 ROWS / CSV HEADER` skips the column headers.



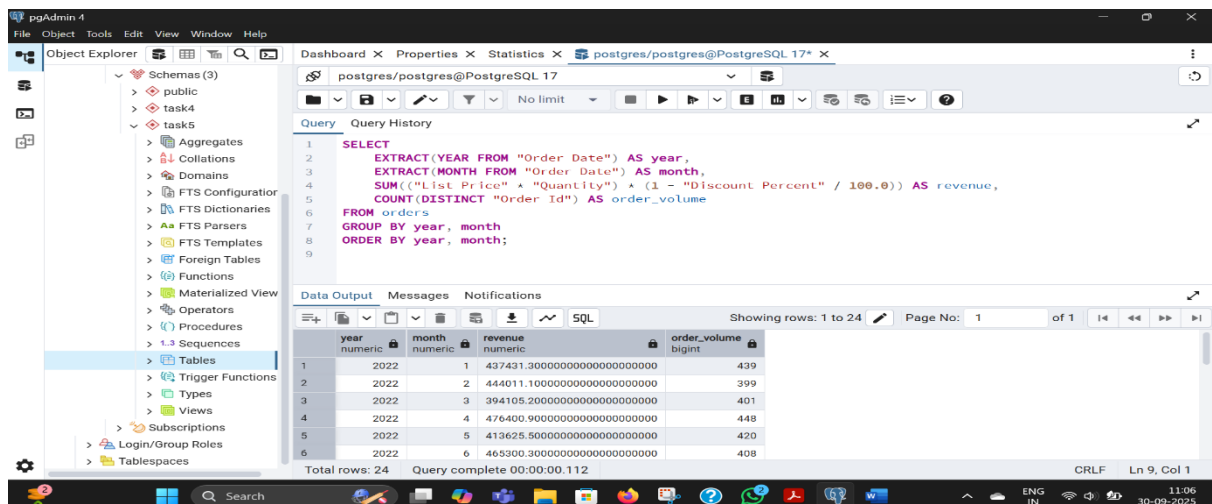
3.QUERYING THE DATA

3.1 Calculate Monthly Revenue & Order Volume

```
SELECT
    EXTRACT(YEAR FROM "Order Date") AS year,
    EXTRACT(MONTH FROM "Order Date") AS month,
    SUM(("List Price" * "Quantity") * (1 - "Discount Percent" / 100.0)) AS
revenue,
COUNT(DISTINCT "Order Id") AS order_volume
FROM orders
GROUP BY year, month
ORDER BY year, month;
```

Explanation:

- EXTRACT(YEAR FROM "Order Date") and EXTRACT(MONTH FROM "Order Date") split the date into year & month.
- SUM(("List Price" * "Quantity") * (1 - "Discount Percent" / 100.0)) calculates the monthly revenue after applying discounts.
- COUNT(DISTINCT "Order Id") counts the number of unique orders per month.
- GROUP BY year, month aggregates results month by month.
- ORDER BY year, month sorts results chronologically.



The screenshot shows the pgAdmin 4 interface. On the left is the Object Explorer showing the database structure. The main pane displays a SQL query and its results. The query is the same as the one in the previous block. The results are shown in a table with 5 columns: year, month, revenue, and order_volume. The data is grouped by year and month, showing 6 rows of results for the year 2022.

year	month	revenue	order_volume
2022	1	437431.30000000000000000000000000	439
2022	2	444011.10000000000000000000000000	399
2022	3	394105.20000000000000000000000000	401
2022	4	476400.90000000000000000000000000	448
2022	5	413625.50000000000000000000000000	420
2022	6	468300.30000000000000000000000000	408

Total rows: 24 Query complete 00:00:00.112

3.2 Limit to a Specific Time Period (Example: Year 2023)

SELECT

EXTRACT(YEAR FROM "Order Date") AS year,

EXTRACT(MONTH FROM "Order Date") AS month,

SUM(("List Price" * "Quantity") * (1 - "Discount Percent" / 100.0)) AS
revenue,

COUNT(DISTINCT "Order Id") AS order_volume

FROM orders

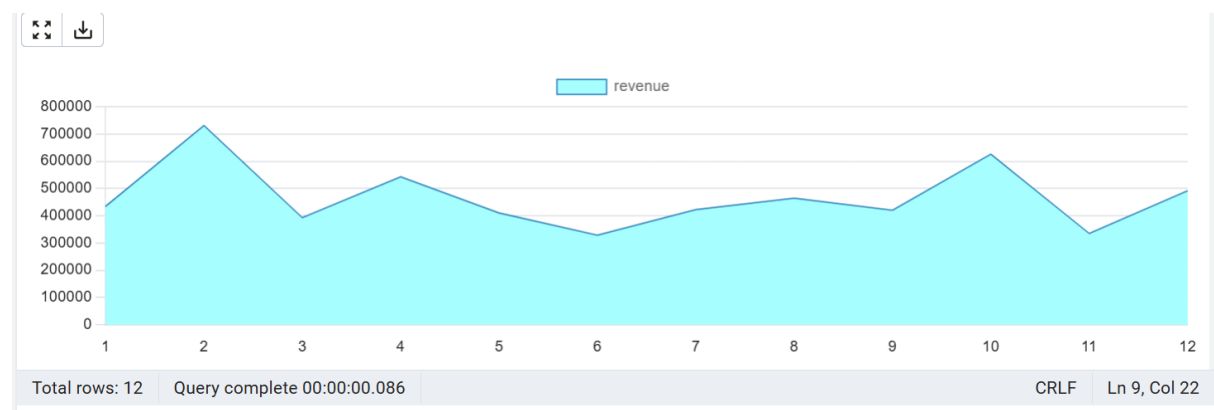
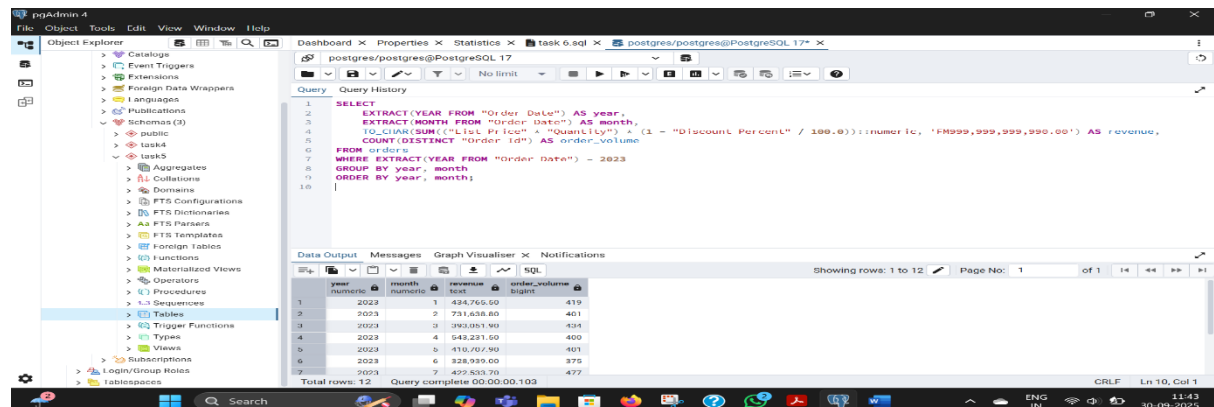
WHERE EXTRACT(YEAR FROM "Order Date") = 2023

GROUP BY year, month

ORDER BY year, month;

Explanation:

- The WHERE clause limits results to **2023 only**.
- This allows us to analyze a specific period rather than the full dataset.



4. RESULTS (ILLUSTRATION)

Running the above queries will give a table like:

"year""month"	"revenue"	"order_volume"
2023 1	"434,765.50"	419
2023 2	"731,638.80"	401
2023 3	"393,051.90"	434
2023 4	"543,231.50"	400
2023 5	"410,707.90"	401
2023 6	"328,939.00"	375
2023 7	"422,533.70"	477
2023 8	"465,010.30"	420
2023 9	"420,620.50"	353
2023 10	"626,498.30"	418
2023 11	"334,940.60"	413
2023 12	"491,848.90"	446