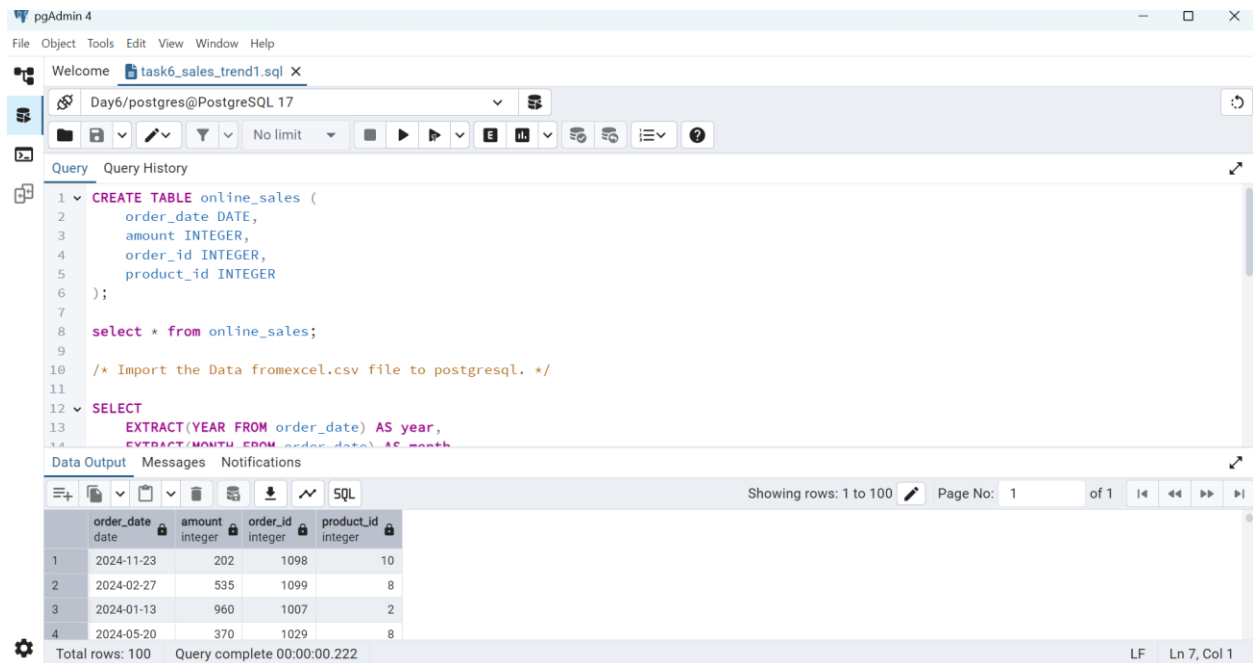


# Sales Trend Analysis Using Aggregations

Step 1: -



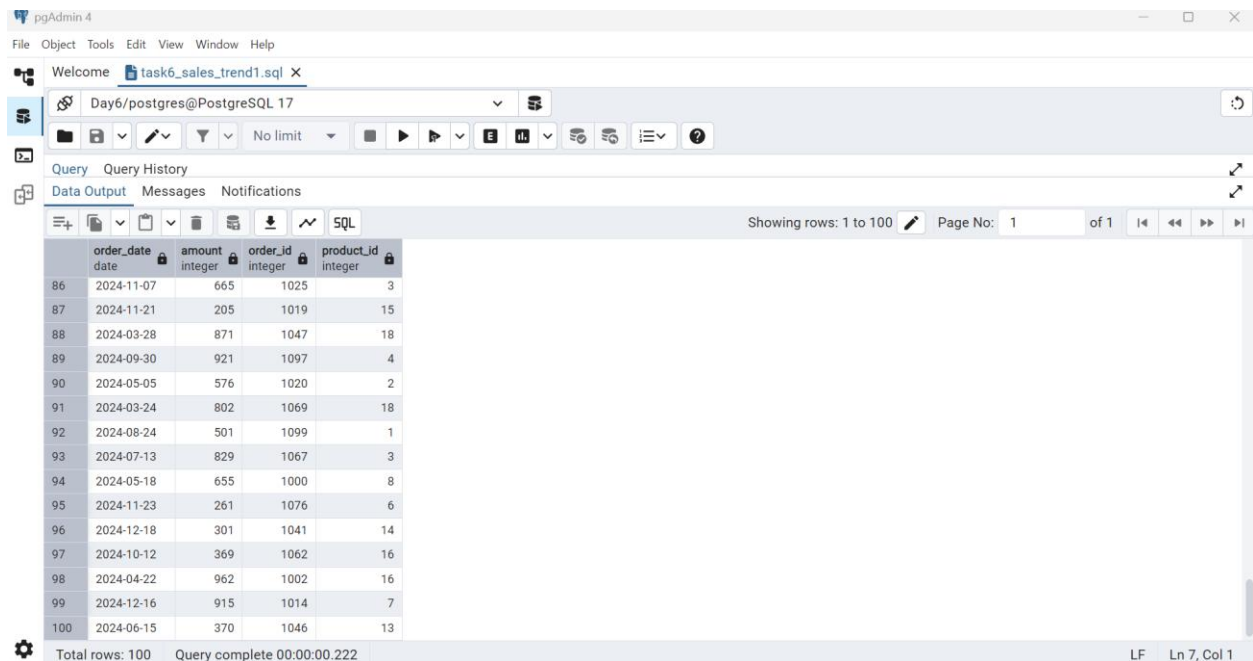
The screenshot shows the pgAdmin 4 interface with a SQL query executed. The query creates a table named 'online\_sales' and imports data from a CSV file. The data output is displayed in a table with 4 columns: order\_date, amount, order\_id, and product\_id. The status bar indicates 'Total rows: 100' and 'Query complete 00:00:00.222'.

```
1 CREATE TABLE online_sales (  
2   order_date DATE,  
3   amount INTEGER,  
4   order_id INTEGER,  
5   product_id INTEGER  
6 );  
7  
8 select * from online_sales;  
9  
10 /* Import the Data from excel.csv file to postgresql. */  
11  
12 SELECT  
13   EXTRACT(YEAR FROM order_date) AS year,  
14   EXTRACT(MONTH FROM order_date) AS month,
```

	order_date	amount	order_id	product_id
1	2024-11-23	202	1098	10
2	2024-02-27	535	1099	8
3	2024-01-13	960	1007	2
4	2024-05-20	370	1029	8

Total rows: 100 Query complete 00:00:00.222

Step 2: -



The screenshot shows the pgAdmin 4 interface with a SQL query executed. The data output is displayed in a table with 4 columns: order\_date, amount, order\_id, and product\_id. The status bar indicates 'Total rows: 100' and 'Query complete 00:00:00.222'.

	order_date	amount	order_id	product_id
86	2024-11-07	665	1025	3
87	2024-11-21	205	1019	15
88	2024-03-28	871	1047	18
89	2024-09-30	921	1097	4
90	2024-05-05	576	1020	2
91	2024-03-24	802	1069	18
92	2024-08-24	501	1099	1
93	2024-07-13	829	1067	3
94	2024-05-18	655	1000	8
95	2024-11-23	261	1076	6
96	2024-12-18	301	1041	14
97	2024-10-12	369	1062	16
98	2024-04-22	962	1002	16
99	2024-12-16	915	1014	7
100	2024-06-15	370	1046	13

Total rows: 100 Query complete 00:00:00.222

## Step 3:

The screenshot shows the pgAdmin 4 interface with a SQL query executed. The query is a SELECT statement that extracts the year and month from the order\_date, sums the amount to get monthly\_revenue, and counts the distinct order\_ids to get order\_volume. The results are grouped by year and month, and ordered by year and month. The data output shows 12 rows, with the first 5 rows displayed.

```
SELECT
  EXTRACT(YEAR FROM order_date) AS year,
  EXTRACT(MONTH FROM order_date) AS month,
  SUM(amount) AS monthly_revenue,
  COUNT(DISTINCT order_id) AS order_volume
FROM
  online_sales
GROUP BY
  EXTRACT(YEAR FROM order_date), EXTRACT(MONTH FROM order_date)
ORDER BY
  year, month;
```

	year numeric	month numeric	monthly_revenue bigint	order_volume bigint
1	2024	1	4415	7
2	2024	2	6744	10
3	2024	3	5085	7
4	2024	4	5914	11
5	2024	5	6331	10

Total rows: 12    Query complete 00:00:00.225    LF    Ln 22, Col 17

## Step 4:

The screenshot shows the pgAdmin 4 interface with the same SQL query executed. The data output shows 12 rows, with the last 5 rows displayed (rows 8 to 12).

```
SELECT
  EXTRACT(YEAR FROM order_date) AS year,
  EXTRACT(MONTH FROM order_date) AS month,
  SUM(amount) AS monthly_revenue,
  COUNT(DISTINCT order_id) AS order_volume
FROM
  online_sales
GROUP BY
  EXTRACT(YEAR FROM order_date), EXTRACT(MONTH FROM order_date)
ORDER BY
  year, month;
```

	year numeric	month numeric	monthly_revenue bigint	order_volume bigint
8	2024	8	3611	7
9	2024	9	1684	2
10	2024	10	3394	9
11	2024	11	4313	9
12	2024	12	4880	10

Total rows: 12    Query complete 00:00:00.225    LF    Ln 22, Col 17

## Step 5:

pgAdmin 4

File Object Tools Edit View Window Help

Welcome task6\_sales\_trend1.sql X

Day6/postgres@PostgreSQL 17

No limit

Query Query History

```
25
26 SELECT
27     EXTRACT(YEAR FROM order_date) AS year,
28     EXTRACT(MONTH FROM order_date) AS month,
29     SUM(amount) AS monthly_revenue
30 FROM
31     online_sales
32 GROUP BY
33     EXTRACT(YEAR FROM order_date), EXTRACT(MONTH FROM order_date)
34 ORDER BY
35     monthly_revenue DESC
36 LIMIT 3;
```

Data Output Messages Notifications

Showing rows: 1 to 3 Page No: 1 of 1

	year numeric	month numeric	monthly_revenue bigint
1	2024	2	6744
2	2024	5	6331
3	2024	4	5914

Total rows: 3 Query complete 00:00:00.134 LF Ln 37, Col 1