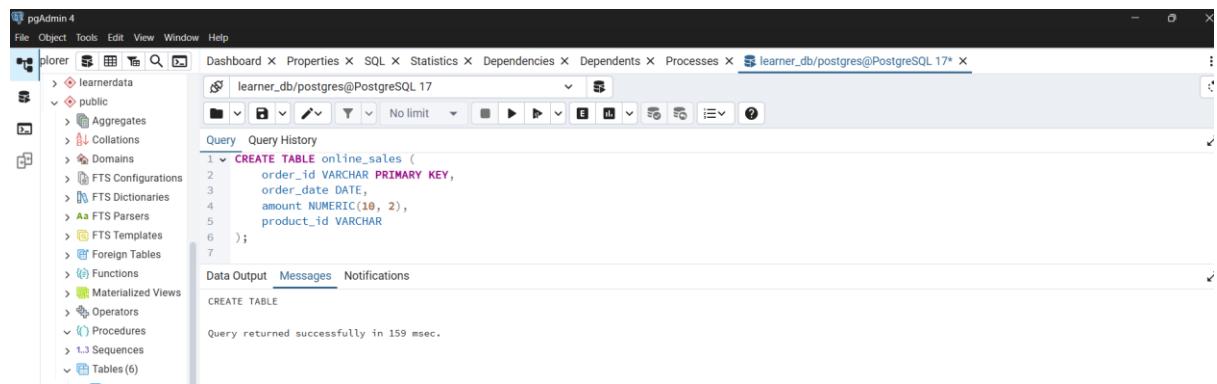


✓ PostgreSQL Table Creation

```
CREATE TABLE online_sales (
    order_id VARCHAR PRIMARY KEY,
    order_date DATE,
    amount NUMERIC(10, 2),
    product_id VARCHAR
);
```



1. View the Dataset

```
SELECT * FROM online_sales;
```

A screenshot of the pgAdmin 4 interface showing the results of the SELECT query. The left sidebar shows the 'learnerdata' schema with various tables like 'customers', 'orders', and 'products'. The main window displays the results of the query in a grid format. The columns are labeled 'order_id', 'order_date', 'amount', and 'product_id'. The data consists of 25 rows, each representing an order with its details. The last row is highlighted.

Total rows: 25 | Query complete 00:00:00.315

CRLF | Ln 1, Col 28

2. Extract Year & Month

SELECT

```
order_id,  
order_date,  
EXTRACT(YEAR FROM order_date) AS year,  
EXTRACT(MONTH FROM order_date) AS month  
FROM online_sales;
```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with tables like learnerdata, customers, opportunities, and online_sales. The main area shows a query editor with the following SQL code:

```
1 SELECT  
2     order_id,  
3     order_date,  
4     EXTRACT(YEAR FROM order_date) AS year,  
5     EXTRACT(MONTH FROM order_date) AS month  
6 FROM online_sales;
```

Below the query, the results are displayed in a table:

	order_id	order_date	year	month
1	O1000	2023-11-24	2023	11
2	O1001	2023-02-27	2023	2
3	O1002	2023-01-13	2023	1
4	O1003	2023-05-21	2023	5
5	O1004	2023-05-06	2023	5
6	O1005	2023-04-25	2023	4
7	O1006	2023-09-13	2023	3
8	O1007	2023-02-22	2023	2
9	O1008	2023-12-13	2023	12
10	O1009	2023-10-07	2023	10
11	O1010	2023-02-14	2023	2
12	O1011	2023-10-30	2023	10
13	O1012	2023-08-05	2023	8
14	O1013	2023-01-17	2023	1

3. Monthly Revenue and Order Volume

SELECT

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

The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:

```

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

```

The results grid displays the following data:

	year	month	total_revenue	order_volume
1	2023	1	1366.64	4
2	2023	2	966.99	4
3	2023	3	364.16	1
4	2023	4	1410.63	4
5	2023	5	467.16	2
6	2023	8	93.52	1
7	2023	9	487.90	1
8	2023	10	995.72	3
9	2023	11	953.33	3
10	2023	12	429.78	2

4. Filtered by Date Range (e.g., July–December 2023)

SELECT

```

EXTRACT(YEAR FROM order_date) AS year,
EXTRACT(MONTH FROM order_date) AS month,
SUM(amount) AS total_revenue,
COUNT(DISTINCT order_id) AS order_volume
FROM online_sales
WHERE order_date BETWEEN '2023-07-01' AND '2023-12-31'
GROUP BY year, month
ORDER BY year, month;

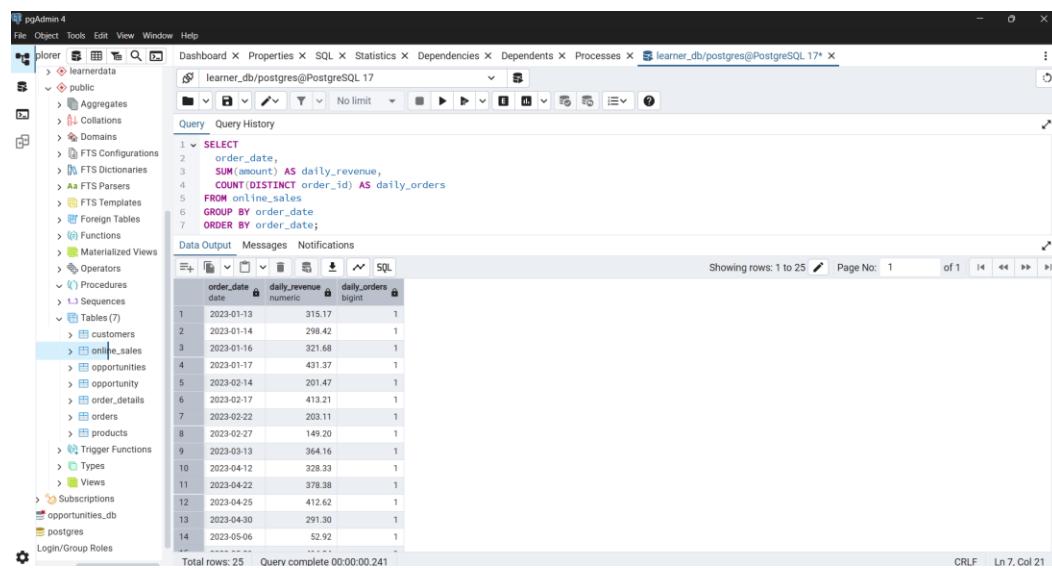
```

The screenshot shows the pgAdmin 4 interface with a query editor window. The query is identical to the one above, but the results grid shows only 5 rows, indicating the WHERE clause has been applied.

	year	month	total_revenue	order_volume
1	2023	8	93.52	1
2	2023	9	487.90	1
3	2023	10	995.72	3
4	2023	11	953.33	3
5	2023	12	429.78	2

5. Daily Revenue Trend (Optional Granularity)

```
SELECT  
    order_date,  
    SUM(amount) AS daily_revenue,  
    COUNT(DISTINCT order_id) AS daily_orders  
FROM online_sales  
GROUP BY order_date  
ORDER BY order_date;
```



The screenshot shows the pgAdmin 4 interface with a query editor window. The query is the same as the one above, selecting daily revenue and order counts from the 'online_sales' table. The results are displayed in a table with three columns: 'order_date', 'daily_revenue', and 'daily_orders'. The data shows 25 rows of daily sales information.

order_date	daily_revenue	daily_orders
2023-01-13	315.17	1
2023-01-14	298.42	1
2023-01-16	321.68	1
2023-01-17	431.37	1
2023-02-14	201.47	1
2023-02-17	413.21	1
2023-02-22	203.11	1
2023-02-27	149.20	1
2023-03-13	364.16	1
2023-04-12	328.33	1
2023-04-22	378.38	1
2023-04-25	412.62	1
2023-04-30	291.30	1
2023-05-06	52.92	1
...
Total rows: 25	Query complete 00:00:00.241	CRLF Ln 7, Col 21

6. Monthly Average Order Value (AOV)

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

```

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

```

The screenshot shows the pgAdmin 4 interface with the results of the above SQL query. The results table has three columns: year, month, and avg_order_value. The data shows the average order value for each month in 2023.

	year	month	avg_order_value
1	2023	1	341.66000000000000000000
2	2023	2	241.74750000000000000000
3	2023	3	364.16000000000000000000
4	2023	4	352.65750000000000000000
5	2023	5	233.58000000000000000000
6	2023	8	93.52000000000000000000
7	2023	9	487.90000000000000000000
8	2023	10	331.9066666666666666667
9	2023	11	317.7766666666666666667
10	2023	12	214.89000000000000000000

Total rows: 10 Query complete 00:00:00.166 CRLF Ln 7, Col 22

7. Top 5 Months by Revenue

```

SELECT
    EXTRACT(YEAR FROM order_date) AS year,
    EXTRACT(MONTH FROM order_date) AS month,
    SUM(amount) AS total_revenue
FROM online_sales
GROUP BY year, month
ORDER BY total_revenue DESC
LIMIT 5;

```

```

SELECT
    EXTRACT(YEAR FROM order_date) AS year,
    EXTRACT(MONTH FROM order_date) AS month,
    SUM(amount) AS total_revenue
FROM online_sales
GROUP BY year, month
ORDER BY total_revenue DESC
LIMIT 5;

```

The screenshot shows the pgAdmin 4 interface with the results of the above SQL query. The results table has three columns: year, month, and total_revenue. The data shows the total revenue for each month in 2023.

	year	month	total_revenue
1	2023	4	1410.63
2	2023	1	1366.64
3	2023	10	995.72
4	2023	2	966.99
5	2023	11	953.33

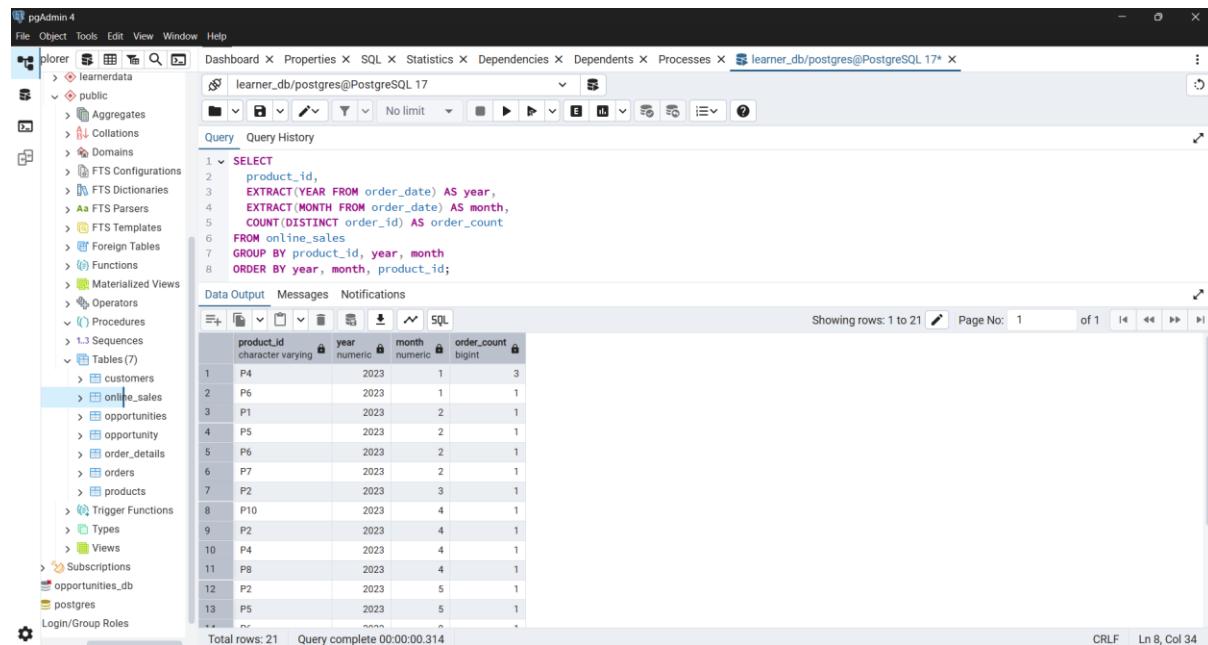
Total rows: 5 Query complete 00:00:00.311 CRLF Ln 8, Col 9

Successfully run. Total query runtime: 311 msec. 5 rows affected.

8. Number of Orders Per Product Per Month

SELECT

```
product_id,  
EXTRACT(YEAR FROM order_date) AS year,  
EXTRACT(MONTH FROM order_date) AS month,  
COUNT(DISTINCT order_id) AS order_count  
  
FROM online_sales  
  
GROUP BY product_id, year, month  
  
ORDER BY year, month, product_id;
```



The screenshot shows the pgAdmin 4 interface with a query editor window. The query is the same as the one above, selecting product_id, year, month, and order_count from the online_sales table, grouped by product_id, year, and month, and ordered by year, month, and product_id. The results are displayed in a table with four columns: product_id, year, month, and order_count. The data shows 13 rows of sales for products P4 through P10 across four months (1, 2, 3, 4) in the year 2023.

	product_id	year	month	order_count
1	P4	2023	1	3
2	P6	2023	1	1
3	P1	2023	2	1
4	P5	2023	2	1
5	P6	2023	2	1
6	P7	2023	2	1
7	P2	2023	3	1
8	P10	2023	4	1
9	P2	2023	4	1
10	P4	2023	4	1
11	P8	2023	4	1
12	P2	2023	5	1
13	P5	2023	5	1

9. Total Revenue Per Product (Yearly Aggregation)

SELECT

```
product_id,  
EXTRACT(YEAR FROM order_date) AS year,  
SUM(amount) AS total_revenue  
  
FROM online_sales  
  
GROUP BY product_id, year  
  
ORDER BY product_id, year;
```

```

SELECT
    product_id,
    EXTRACT(YEAR FROM order_date) AS year,
    SUM(amount) AS total_revenue
FROM online_sales
GROUP BY product_id, year
ORDER BY product_id, year;

```

The screenshot shows the pgAdmin 4 interface with a successful query execution. The results are displayed in a table:

product_id	year	total_revenue
P1	2023	149.20
P10	2023	291.30
P2	2023	1090.68
P3	2023	1002.87
P4	2023	1347.89
P5	2023	1257.23
P6	2023	726.36
P7	2023	640.88
P8	2023	809.08
P9	2023	220.34

Total rows: 10 Query complete 00:00:00.336

10. Revenue Summary by Quarter

```

SELECT
    EXTRACT(YEAR FROM order_date) AS year,
    CEIL(EXTRACT(MONTH FROM order_date) / 3.0) AS quarter,
    SUM(amount) AS total_revenue,
    COUNT(DISTINCT order_id) AS order_volume
FROM online_sales
GROUP BY year, quarter
ORDER BY year, quarter;

```

```

SELECT
    EXTRACT(YEAR FROM order_date) AS year,
    CEIL(EXTRACT(MONTH FROM order_date) / 3.0) AS quarter,
    SUM(amount) AS total_revenue,
    COUNT(DISTINCT order_id) AS order_volume
FROM online_sales
GROUP BY year, quarter
ORDER BY year, quarter;

```

The screenshot shows the pgAdmin 4 interface with a successful query execution. The results are displayed in a table:

year	quarter	total_revenue	order_volume
2023	1	2697.79	9
2023	2	1877.79	6
2023	3	581.42	2
2023	4	2378.83	8

Total rows: 4 Query complete 00:00:00.170