

(1) Retrieve the total number of orders placed.

The screenshot shows a SQL IDE interface with a tab labeled "MY\_PROJECT.sql". The connection is set to "mohan/postgres@PostgreSQL 17". The query editor contains the following SQL code:

```
1
2  --(1) Retrieve the total number of orders placed.
3
4  select count(order_id) as total_orders from order_details
5
```

The "Data Output" tab is active, displaying the results of the query. The results are shown in a table with one column, "total\_orders", and one row with the value 48621.

	total_orders
1	48621

(2) Calculate the total revenue generated from pizza sales.

The screenshot shows a PostgreSQL query editor interface. At the top, a tab labeled 'MY\_PROJECT.sql' is active. Below it, the connection is set to 'mohan/postgres@PostgreSQL 17'. A toolbar contains various icons for file operations, query execution, and settings. The 'Query' tab is selected, displaying the following SQL code:

```
6  -- (2) Calculate the total revenue generated from pizza sales.  
7  
8  SELECT SUM(CAST(order_details.quantity AS numeric) * pizzas.price) AS total_sum  
9  FROM order_details  
10 JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id  
11
```

Below the query editor, the 'Data Output' tab is active, showing the result of the query. The result is a single row with the value 817860.049999993 for the column 'total\_sum'.

	total_sum double precision
1	817860.049999993

### (3) Identify the highest-priced pizza.

The screenshot shows a PostgreSQL query editor interface. At the top, the connection is set to 'mohan/postgres@PostgreSQL 17'. Below the connection bar is a toolbar with icons for file operations, query execution, and settings. The 'Query' tab is active, displaying the following SQL code:

```
11
12 -- (3) Identify the highest-priced pizza.
13
14 SELECT pizza_types.pizza_name, pizzas.price
15 from pizza_types join pizzas on pizza_types.pizza_type_id = pizzas.pizza_type_id
16 order by pizzas.price desc limit 1
17
```

Below the query editor, the 'Data Output' tab is active, showing the results of the query in a table format. The table has two columns: 'pizza\_name' (character varying (255)) and 'price' (double precision). The first row shows 'The Soppresata Pizza' with a price of 20.75.

	pizza_name character varying (255)	price double precision
1	The Soppresata Pizza	20.75

(4) Identify the most common pizza size ordered.

MY\_PROJECT.sql x

mohan/postgres@PostgreSQL 17

No limit

Query Query History

```
17
18 -- (4) Identify the most common pizza size ordered.
19
20 select pizzas.size, count(order_details.order_details_id) as common_pizza from pizzas
21 join order_details on order_details.pizza_id = pizzas.pizza_id
22 group by pizzas.size
23 order by common_pizza desc
24
```

Data Output Messages Notifications

	size character varying (30)	common_pizza bigint
1	L	18526
2	M	15385
3	S	14137
4	XL	544
5	XXL	28

(5) List the top 5 most ordered pizza types along with their quantities.

The screenshot shows a PostgreSQL IDE interface. At the top, there's a tab labeled 'MY\_PROJECT.sql'. Below it, the connection name 'mohan/postgres@PostgreSQL 17' is displayed. A toolbar with various icons for file operations, query execution, and settings is visible. The main area is divided into 'Query' and 'Query History' tabs. The 'Query' tab contains the following SQL code:

```
24
25 -- (5) List the top 5 most ordered pizza types along with their quantities.
26
27 select pizza_types.pizza_name, SUM(CAST(order_details.quantity AS numeric)) as quantity from pizza_types
28 join pizzas on pizza_types.pizza_type_id = pizzas.pizza_type_id
29 join order_details on order_details.pizza_id = pizzas.pizza_id
30 Group by pizza_types.pizza_name
31 order by quantity desc limit 5
32
```

Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with the results of the query. The table has two columns: 'pizza\_name' (character varying (255)) and 'quantity' (numeric). The results are as follows:

	pizza_name character varying (255)	quantity numeric
1	The Hawaiian Pizza	2422
2	The Pepperoni Pizza	2418
3	The Sicilian Pizza	1938
4	The Mexicana Pizza	1484
5	The Napolitana Pizza	1464

(6) Join the necessary tables to find the total quantity of each pizza category ordered.



The screenshot shows a PostgreSQL IDE interface. The top bar indicates the connection is 'mohan/postgres@PostgreSQL 17'. Below the connection bar is a toolbar with various icons for file operations, query execution, and settings. The main query editor displays the following SQL code:

```
-- (6) Join the necessary tables to find the total quantity of each pizza category ordered.  
  
select pizza_types.category, SUM(CAST(order_details.quantity AS numeric)) as quantity from pizza_types  
join pizzas on pizza_types.pizza_type_id = pizzas.pizza_type_id  
join order_details on order_details.pizza_id = pizzas.pizza_id  
Group by pizza_types.category  
order by quantity desc
```

Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing the results of the query in a table format:

	category character varying (255)	quantity numeric
1	Classic	6304
2	Supreme	3836
3	Veggie	2418

(7) Determine the distribution of orders by hour of the day.

MY\_PROJECT.sql x

mohan/postgres@PostgreSQL 17

Query Query History

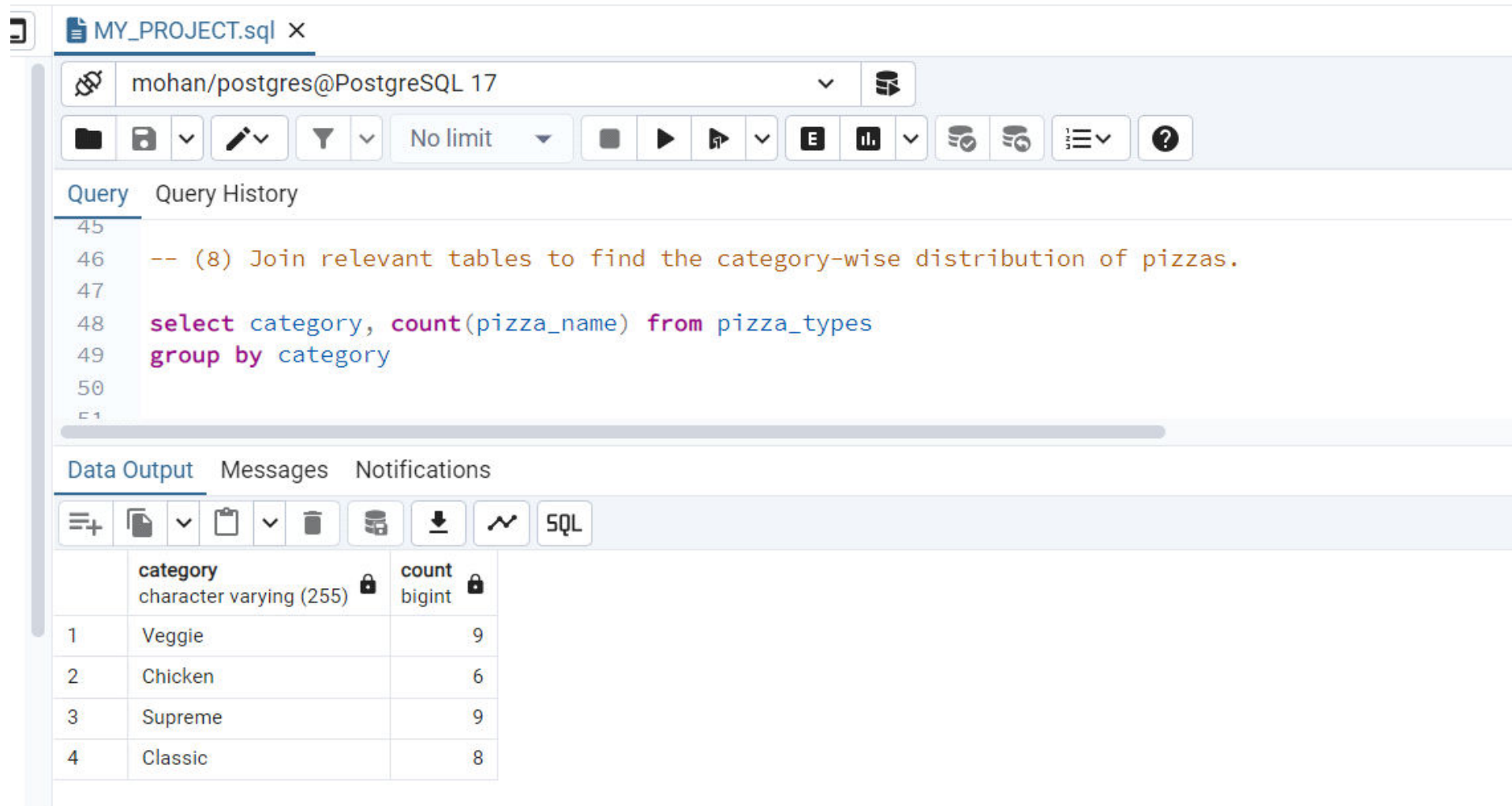
```
40
41 -- (7) Determine the distribution of orders by hour of the day.
42
43 select EXTRACT(hour from time) as hours, count(order_id) as order_count from orders
44 group by hours
45
```

Data Output Messages Notifications

	hours numeric	order_count bigint
1	11	1231
2	23	28
3	18	2399
4	19	2009
5	15	1468
6	9	1
7	21	1198
8	17	2336
9	20	1642
10	13	2455
11	10	8
12	16	1920
13	22	663
14	12	2520
15	14	1472



(8) Join relevant tables to find the category-wise distribution of pizzas.



The screenshot shows a PostgreSQL IDE interface. At the top, a tab labeled 'MY\_PROJECT.sql' is open. Below it, the connection name 'mohan/postgres@PostgreSQL 17' is displayed. A toolbar with various icons for file operations, query execution, and settings is visible. The 'Query' tab is active, showing a SQL query that counts the number of pizzas for each category. The query is as follows:

```
45
46 -- (8) Join relevant tables to find the category-wise distribution of pizzas.
47
48 select category, count(pizza_name) from pizza_types
49 group by category
50
```

Below the query editor, the 'Data Output' tab is active, displaying the results of the query in a table format. The table has two columns: 'category' (character varying (255)) and 'count' (bigint). The results show four categories: Veggie (9), Chicken (6), Supreme (9), and Classic (8).

	category character varying (255)	count bigint
1	Veggie	9
2	Chicken	6
3	Supreme	9
4	Classic	8



(9) Group the orders by date and calculate the average number of pizzas ordered per day.

The screenshot shows a PostgreSQL IDE interface. At the top, a tab labeled 'MY\_PROJECT.sql' is open. Below it, the connection string 'mohan/postgres@PostgreSQL 17' is displayed. A toolbar contains various icons for file operations, query execution, and settings. The main area is divided into 'Query' and 'Query History' tabs. The 'Query' tab shows a SQL query starting at line 50 and ending at line 58. The query is a comment followed by a SELECT statement that calculates the average number of pizzas ordered per day. Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with two columns: 'round' and 'numeric'. The first row of data shows the value '138' under the 'numeric' column.

```
50
51
52 -- (9) Group the orders by date and calculate the average number of pizzas ordered per day.
53
54 select round(avg(quantity),0) from
55 (select orders.date, SUM(CAST(order_details.quantity AS numeric)) as quantity
56 from orders join order_details on order_details.order_id = orders.order_id
57 group by orders.date)
58
```

	round	numeric
1		138

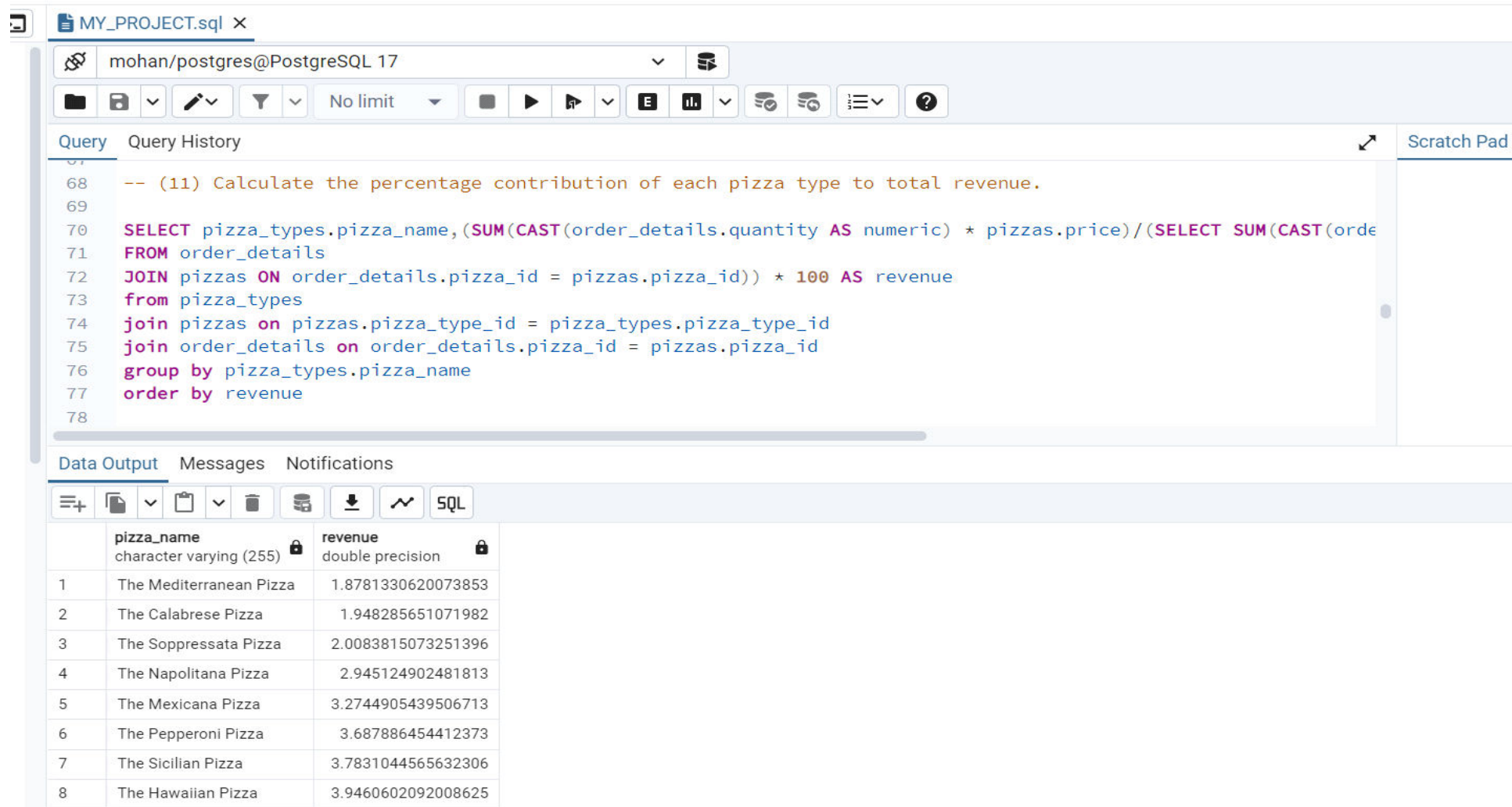
(10) Determine the top 3 most ordered pizza types based on revenue.

The screenshot shows a PostgreSQL IDE interface. At the top, a tab labeled 'MY\_PROJECT.sql' is open. Below it, a toolbar contains various icons for file operations, query execution, and settings. The main area displays a SQL query with line numbers 59 through 67. The query is a SELECT statement that calculates the total revenue for each pizza type by joining the 'pizza\_types', 'pizzas', and 'order\_details' tables. It uses 'SUM' to aggregate the revenue and 'ORDER BY' to sort the results in descending order, limiting the output to the top 3 results. Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with two columns: 'pizza\_name' and 'revenue'. The table contains three rows of data, representing the top 3 most ordered pizza types based on revenue.

```
59  -- (10) Determine the top 3 most ordered pizza types based on revenue.
60
61  SELECT pizza_types.pizza_name, SUM(CAST(order_details.quantity AS numeric) * pizzas.price) AS revenue
62  FROM pizza_types
63  JOIN pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
64  JOIN order_details ON order_details.pizza_id = pizzas.pizza_id
65  GROUP BY pizza_types.pizza_name
66  ORDER BY revenue DESC LIMIT 3
67
```

	pizza_name character varying (255)	revenue double precision
1	The Hawaiian Pizza	32273.25
2	The Sicilian Pizza	30940.5
3	The Pepperoni Pizza	30161.75

(11) Calculate the percentage contribution of each pizza type to total revenue.



The screenshot shows a PostgreSQL IDE interface. The top toolbar includes icons for file operations, query execution, and settings. The main editor displays a SQL query for calculating the percentage contribution of each pizza type to total revenue. The query uses a subquery to calculate the total revenue and then joins it with the pizza\_types table to calculate the percentage for each type. The results are shown in a table with columns for pizza\_name and revenue.

```
68 -- (11) Calculate the percentage contribution of each pizza type to total revenue.
69
70 SELECT pizza_types.pizza_name, (SUM(CAST(order_details.quantity AS numeric) * pizzas.price) / (SELECT SUM(CAST(orde
71 FROM order_details
72 JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id)) * 100 AS revenue
73 FROM pizza_types
74 JOIN pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
75 JOIN order_details ON order_details.pizza_id = pizzas.pizza_id
76 GROUP BY pizza_types.pizza_name
77 ORDER BY revenue
78
```

	pizza_name character varying (255)	revenue double precision
1	The Mediterranean Pizza	1.8781330620073853
2	The Calabrese Pizza	1.948285651071982
3	The Soppressata Pizza	2.0083815073251396
4	The Napolitana Pizza	2.945124902481813
5	The Mexicana Pizza	3.2744905439506713
6	The Pepperoni Pizza	3.687886454412373
7	The Sicilian Pizza	3.7831044565632306
8	The Hawaiian Pizza	3.9460602092008625

(12) Analyze the cumulative revenue generated over time.

The screenshot shows a PostgreSQL query editor interface. The top bar indicates the user is 'mohan/postgres@PostgreSQL 17'. Below the toolbar, the 'Query' tab is active, displaying a SQL query that calculates cumulative revenue over time. The query uses a window function `SUM() over(order by date)` to calculate the cumulative revenue from a subquery that joins `order_details` and `pizzas` tables. The 'Data Output' tab is also visible, showing a table with two columns: `date` and `cum_revenue`. The table contains 13 rows of data, showing the cumulative revenue increasing over time from 2015-01-01 to 2015-01-13.

```
78
79  -- (12) Analyze the cumulative revenue generated over time.
80
81  select date, SUM(revenue) over(order by date) as cum_revenue from
82  (select orders.date, SUM(CAST(order_details.quantity AS numeric) * pizzas.price) AS revenue
83  from order_details JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id
84  join orders on order_details.order_id = orders.order_id
85  group by orders.date)
86
```

	date date	cum_revenue double precision
1	2015-01-01	2713.85000000000004
2	2015-01-02	5445.75
3	2015-01-03	8108.15
4	2015-01-04	9863.6
5	2015-01-05	11929.55
6	2015-01-06	14358.5
7	2015-01-07	16560.7
8	2015-01-08	19399.05
9	2015-01-09	21526.4
10	2015-01-10	23990.35000000000002
11	2015-01-11	25862.65
12	2015-01-12	27781.7
13	2015-01-13	29831.30000000000003

**(13) Determine the top 3 most ordered pizza types based on revenue for each pizza category.**

MY\_PROJECT.sql

mohan/postgres@PostgreSQL 17

No limit

E

Query

Query History

```
86
87 --(13) Determine the top 3 most ordered pizza types based on revenue for each pizza category.
88
89
90 select pizza_name, revenue from
91 (select pizza_name, category, revenue,
92 rank() over(partition by category order by revenue desc) as rn from
93 (SELECT pizza_types.pizza_name, pizza_types.category,
94 SUM(CAST(order_details.quantity AS numeric) * pizzas.price) AS revenue
95 from pizza_types join pizzas on pizza_types.pizza_type_id = pizzas.pizza_type_id
96 join order_details on order_details.pizza_id = pizzas.pizza_id
97 group by pizza_types.pizza_name, pizza_types.category) as a) as b
98 where rn <= 3;
```

Data Output

Messages

Notifications

SQL

	pizza_name character varying (255)	revenue double precision
1	The Hawaiian Pizza	32273.25
2	The Pepperoni Pizza	30161.75
3	The Napolitana Pizza	24087
4	The Sicilian Pizza	30940.5
5	The Soppressata Pizza	16425.75
6	The Calabrese Pizza	15934.25
7	The Mexicana Pizza	26780.75
8	The Mediterranean Pizza	15360.5

