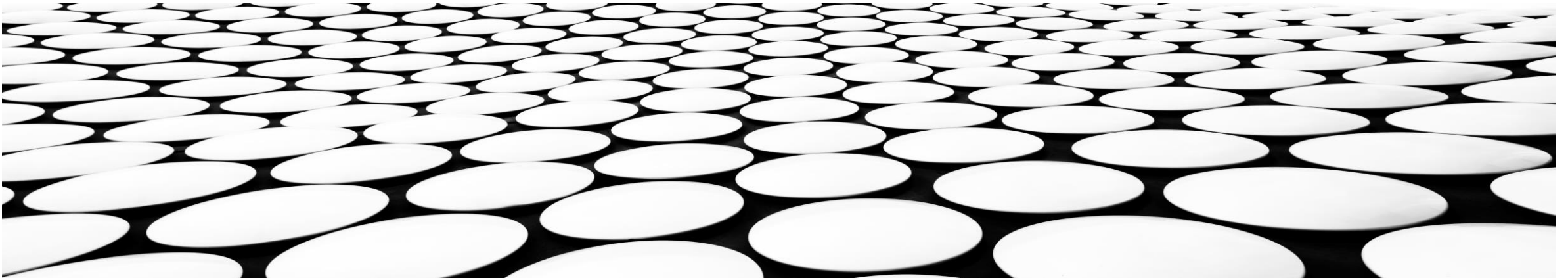


---

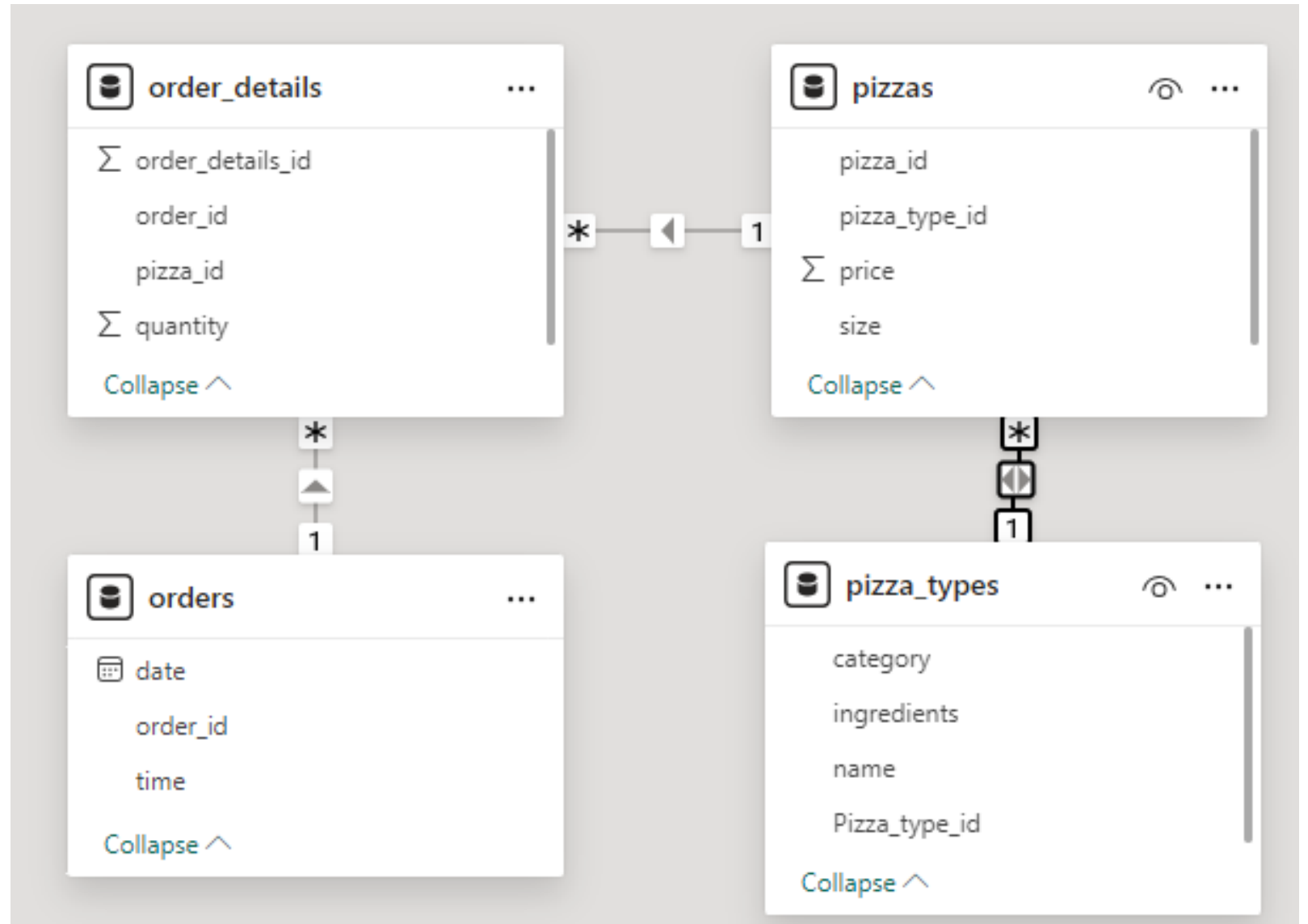
# HELLO

MY NAME IS JASNEET KAUR.

IN THIS PROJECT, I HAVE UTILIZED SQL QUERIES TO SOLVE QUESTIONS THAT WERE RELATED TO PIZZA SALES.



# SQL SCHEMA



# LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES

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

```
SELECT TOP 5 pizza_types.name, SUM(order_details.quantity) 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 name
ORDER BY quantity DESC;
```

177 %

Results Messages

	name	quantity
1	The Classic Deluxe Pizza	2453
2	The Barbecue Chicken Pizza	2432
3	The Hawaiian Pizza	2422
4	The Pepperoni Pizza	2418
5	The Thai Chicken Pizza	2371

# IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

--Identify the most common pizza size ordered.

```
SELECT top 1 pizzas.size, COUNT(order_details.order_details_id) AS common_pizza
FROM pizzas
JOIN order_details
ON pizzas.pizza_id=order_details.pizza_id
GROUP BY pizzas.size
ORDER BY common_pizza DESC;
```

177 %

Results Messages

	size	common_pizza
1	L	18526

# IDENTIFY THE HIGHEST-PRICED PIZZA.

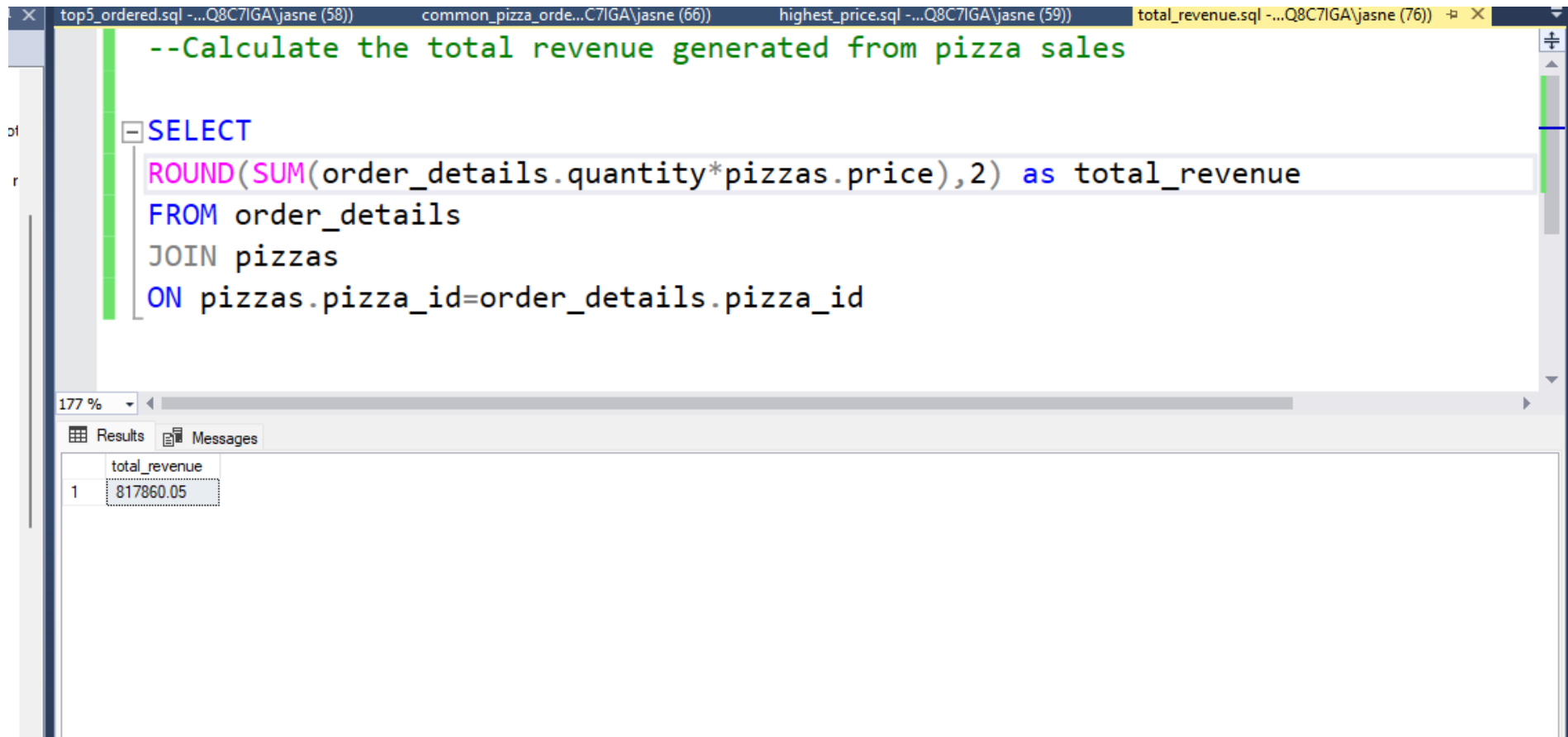
```
--Identify the highest-priced pizza
SELECT TOP 1 pizza_types.name, ROUND(pizzas.price, 2) AS price
FROM pizza_types
JOIN pizzas
ON pizza_types.pizza_type_id=pizzas.pizza_type_id
ORDER BY pizzas.price DESC;
```

177 %

Results Messages

	name	price
1	The Greek Pizza	35.95

# CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.



The screenshot shows a SQL IDE with four tabs: 'top5\_ordered.sql', 'common\_pizza\_orde...', 'highest\_price.sql', and 'total\_revenue.sql'. The 'total\_revenue.sql' tab is active and contains the following SQL query:

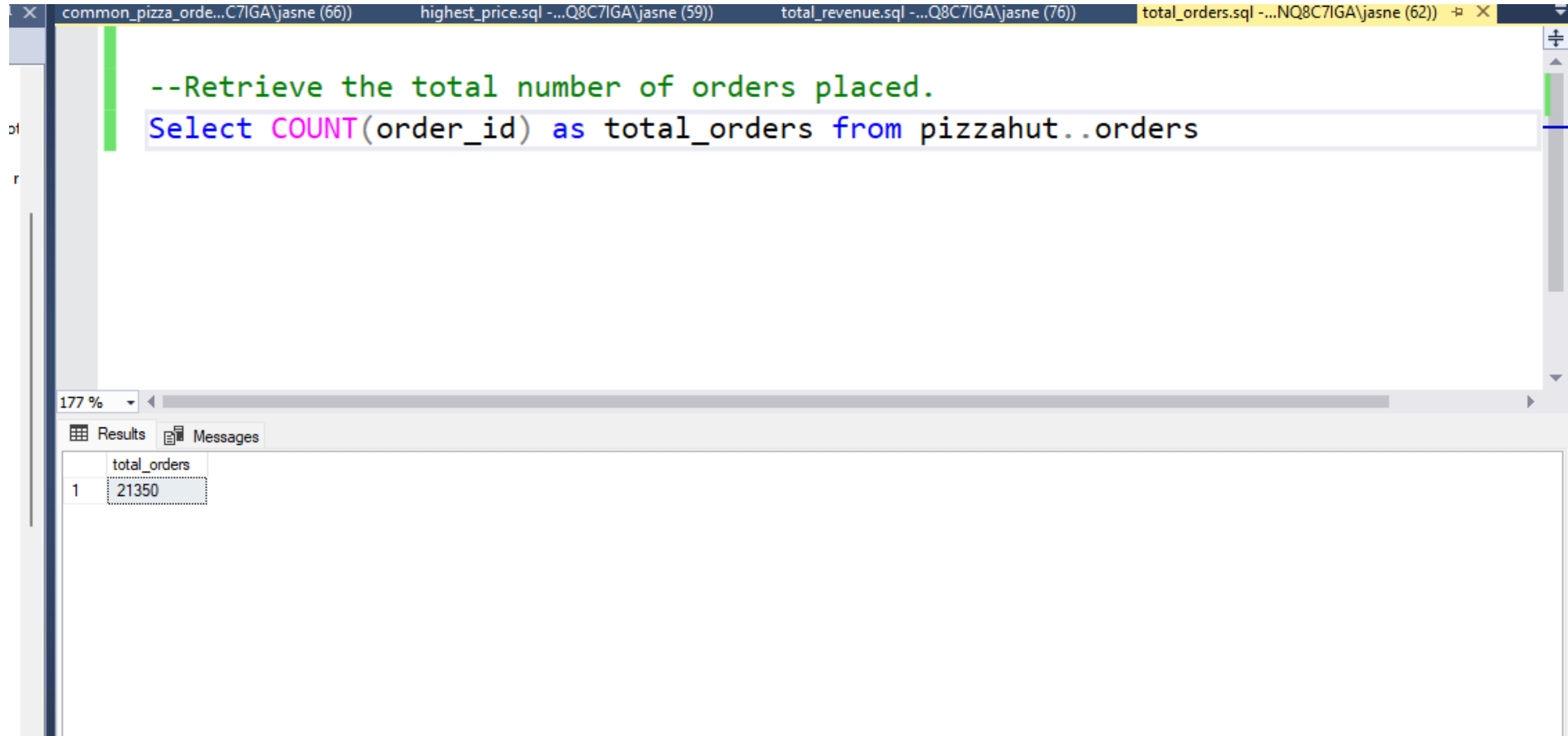
```
--Calculate the total revenue generated from pizza sales

SELECT
ROUND(SUM(order_details.quantity*pizzas.price),2) as total_revenue
FROM order_details
JOIN pizzas
ON pizzas.pizza_id=order_details.pizza_id
```

Below the query editor, the 'Results' pane shows a single row of data:

	total_revenue
1	817860.05

# RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.



The screenshot shows a SQL IDE with multiple tabs. The active tab is titled "total\_orders.sql - ...NQ8C7IGA\jasne (62)". The query editor contains the following SQL code:

```
--Retrieve the total number of orders placed.  
Select COUNT(order_id) as total_orders from pizzahut..orders
```

Below the query editor, the "Results" tab is selected, displaying a single row of data:

	total_orders
1	21350

# JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.

```
--Join the necessary tables to find the total quantity
--of each pizza category ordered.
SELECT pizza_types.category, SUM(order_details.quantity) AS quantity
FROM pizzahut..pizza_types
JOIN pizzahut..pizzas
ON pizza_types.pizza_type_id=pizzas.pizza_type_id
JOIN pizzahut..order_details
ON order_details.pizza_id=pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY quantity DESC;
```

177 %

Results Messages

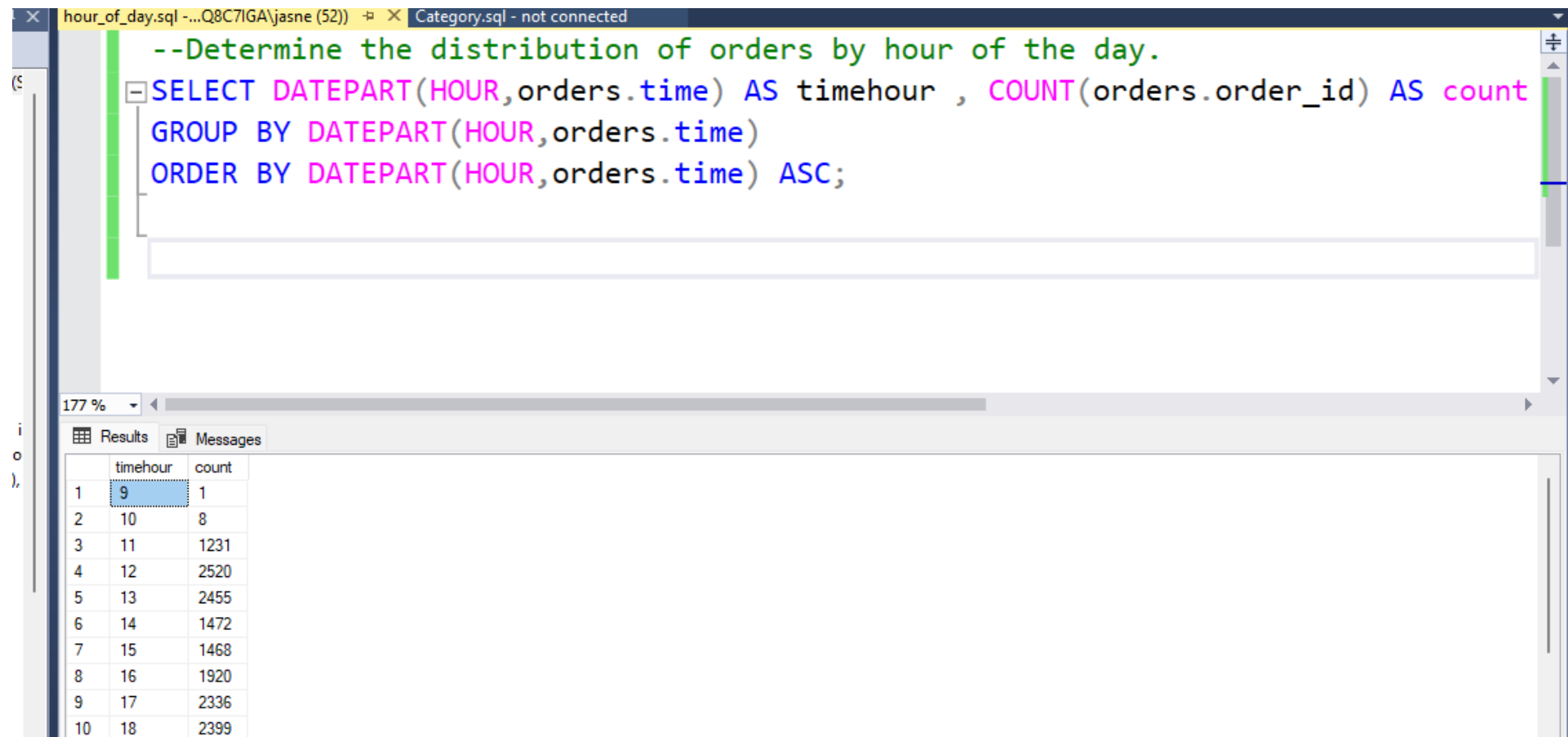
	category	quantity
1	Classic	14888
2	Supreme	11987
3	Veggie	11649
4	Chicken	11050

Disconnected.

Ln 11 Col 1 Ch 1 INS



# DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.



The screenshot shows a SQL IDE window with two tabs: "hour\_of\_day.sql - ...Q8C7IGA\jasne (52)" and "Category.sql - not connected". The active tab contains the following SQL query:

```
--Determine the distribution of orders by hour of the day.  
SELECT DATEPART(HOUR,orders.time) AS timehour , COUNT(orders.order_id) AS count  
GROUP BY DATEPART(HOUR,orders.time)  
ORDER BY DATEPART(HOUR,orders.time) ASC;
```

Below the query editor, the "Results" tab is active, displaying a table with the following data:

	timehour	count
1	9	1
2	10	8
3	11	1231
4	12	2520
5	13	2455
6	14	1472
7	15	1468
8	16	1920
9	17	2336
10	18	2399

# GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.

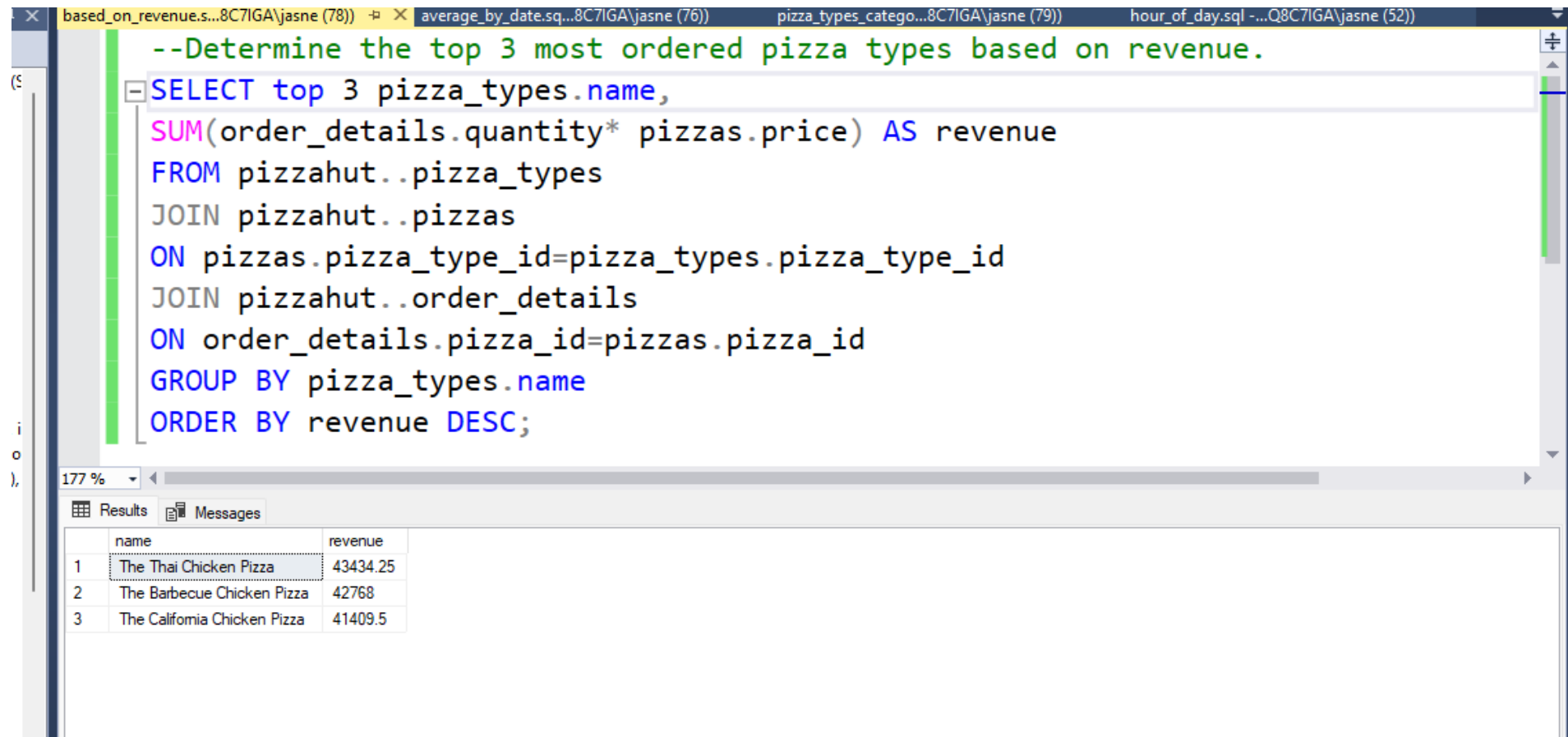
The screenshot shows a SQL IDE with a query editor and a results pane. The query editor contains the following SQL code:

```
--Group the orders by date and  
--calculate the average number of pizzas ordered per day.  
SELECT AVG(total) AS average from  
(SELECT orders.date, SUM(order_details.quantity ) AS total  
FROM pizzahut..orders  
JOIN pizzahut..order_details  
ON orders.order_id=order_details.order_id  
GROUP BY date) AS order_quantity;
```

The results pane shows a single row with the average value of 138.

average
138

# DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.



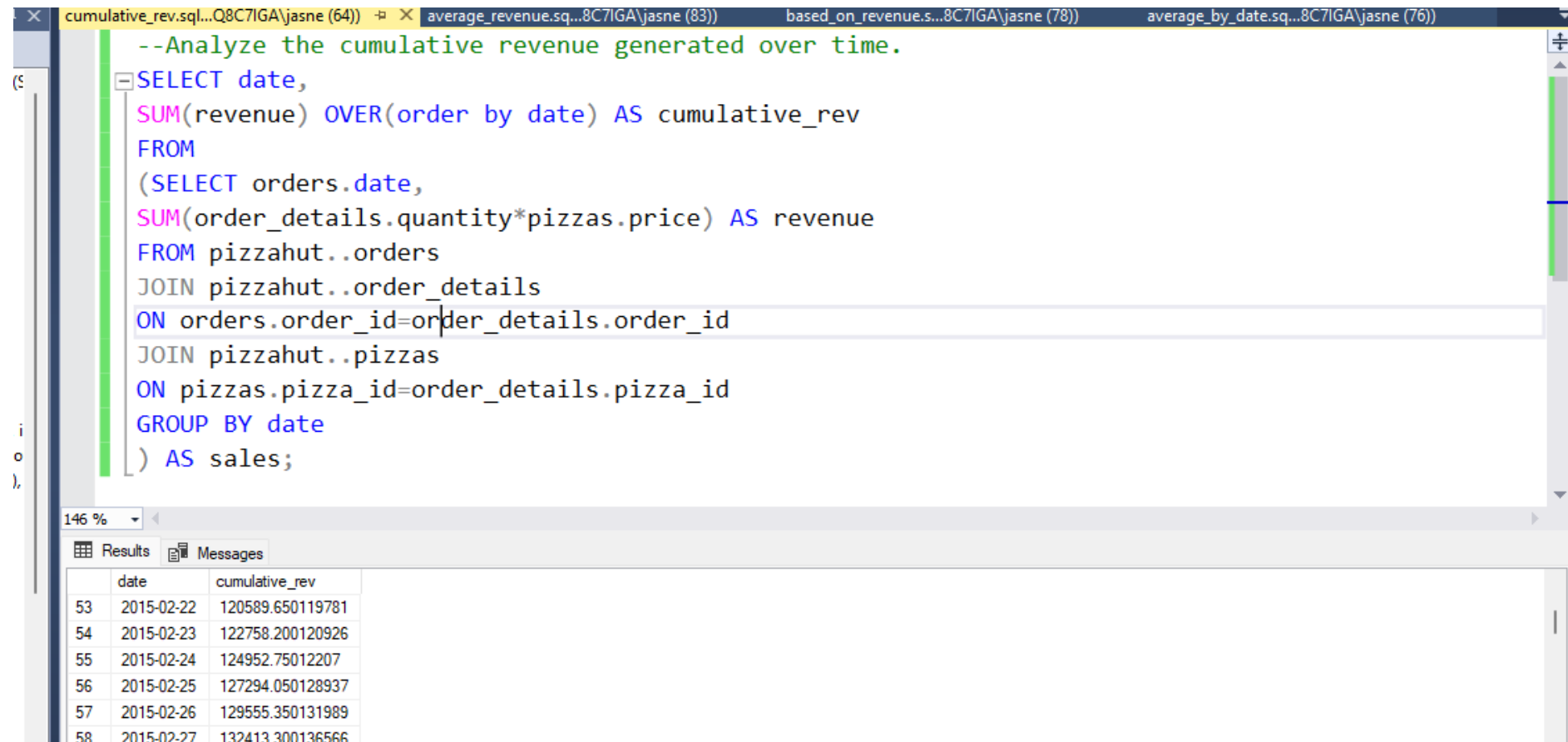
The screenshot shows a SQL IDE with multiple tabs. The active tab is titled "based\_on\_revenue.s...8C7IGA\jasne (78)". The SQL editor contains the following query:

```
--Determine the top 3 most ordered pizza types based on revenue.  
SELECT top 3 pizza_types.name,  
SUM(order_details.quantity* pizzas.price) AS revenue  
FROM pizzahut..pizza_types  
JOIN pizzahut..pizzas  
ON pizzas.pizza_type_id=pizza_types.pizza_type_id  
JOIN pizzahut..order_details  
ON order_details.pizza_id=pizzas.pizza_id  
GROUP BY pizza_types.name  
ORDER BY revenue DESC;
```

Below the editor, the "Results" tab is active, displaying a table with the following data:

	name	revenue
1	The Thai Chicken Pizza	43434.25
2	The Barbecue Chicken Pizza	42768
3	The California Chicken Pizza	41409.5

# ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.



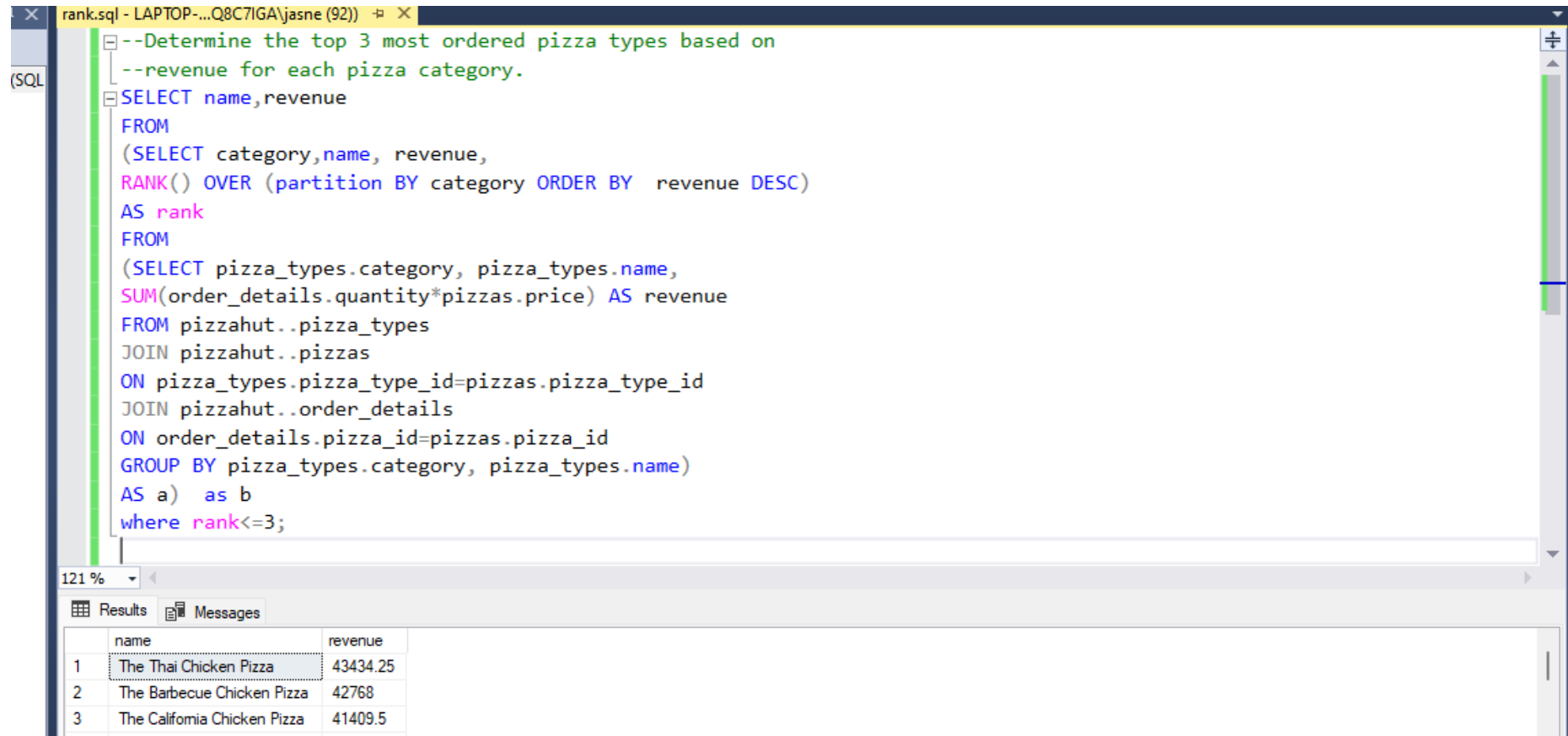
The screenshot shows a SQL IDE with four tabs: `cumulative_rev.sql...Q8C7IGA\jasne (64))`, `average_revenue.sql...8C7IGA\jasne (83))`, `based_on_revenue.s...8C7IGA\jasne (78))`, and `average_by_date.sql...8C7IGA\jasne (76))`. The active tab contains the following SQL query:

```
--Analyze the cumulative revenue generated over time.  
SELECT date,  
       SUM(revenue) OVER(order by date) AS cumulative_rev  
FROM  
  (SELECT orders.date,  
   SUM(order_details.quantity*pizzas.price) AS revenue  
   FROM pizzahut..orders  
   JOIN pizzahut..order_details  
   ON orders.order_id=order_details.order_id  
   JOIN pizzahut..pizzas  
   ON pizzas.pizza_id=order_details.pizza_id  
   GROUP BY date  
  ) AS sales;
```

The query results are displayed in a table with two columns: `date` and `cumulative_rev`. The results show the cumulative revenue for each date from 2015-02-22 to 2015-02-27.

	date	cumulative_rev
53	2015-02-22	120589.650119781
54	2015-02-23	122758.200120926
55	2015-02-24	124952.750122207
56	2015-02-25	127294.050128937
57	2015-02-26	129555.350131989
58	2015-02-27	132413.300136566

# DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.



```
--Determine the top 3 most ordered pizza types based on
--revenue for each pizza category.
SELECT name, revenue
FROM
  (SELECT category, name, revenue,
    RANK() OVER (partition BY category ORDER BY revenue DESC)
    AS rank
  FROM
    (SELECT pizza_types.category, pizza_types.name,
      SUM(order_details.quantity*pizzas.price) AS revenue
    FROM pizzahut..pizza_types
    JOIN pizzahut..pizzas
    ON pizza_types.pizza_type_id=pizzas.pizza_type_id
    JOIN pizzahut..order_details
    ON order_details.pizza_id=pizzas.pizza_id
    GROUP BY pizza_types.category, pizza_types.name)
  AS a) as b
where rank<=3;
```

Results

	name	revenue
1	The Thai Chicken Pizza	43434.25
2	The Barbecue Chicken Pizza	42768
3	The California Chicken Pizza	41409.5



**THANK YOU**