



Borcelle
PIZZA RESTAURANT

SQL
PROJECT

Sales Data Analysis

PIZZA HUT

CHECK IT NOW >>





Introduction

Hello, my name is Laiba Hameed, and I am a BS IT student. I have learned SQL and MySQL for practice and to enhance my understanding of database management. In this project, I have utilized SQL queries to solve various questions related to PizzaHut sales, including analyzing order details, pizza categories, and revenue data. This project helped me apply my SQL skills in a real-world context and improve my ability to work with relational databases.



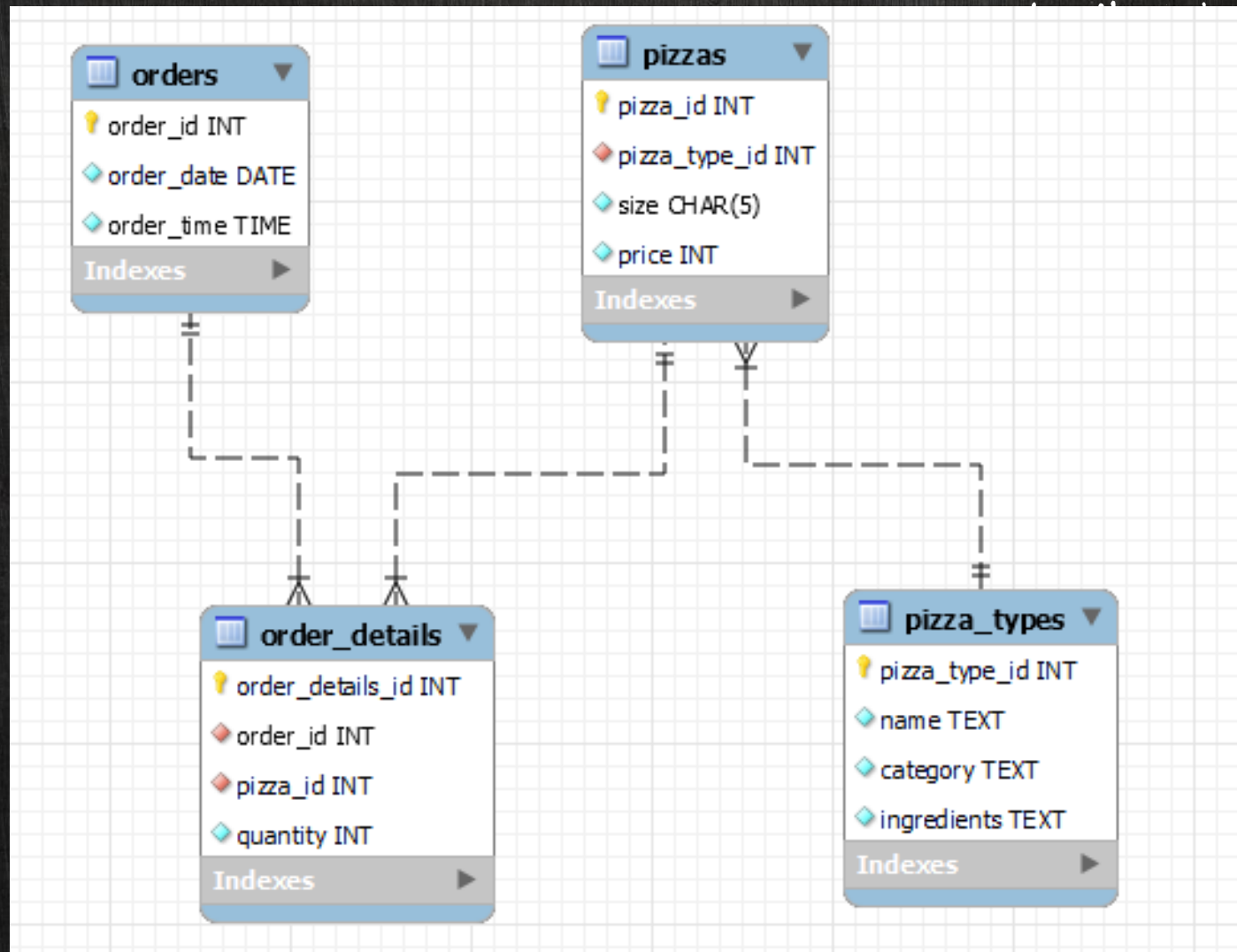
Overview of Pizza Hut Database

The PizzaHut database consists of four main tables to manage pizzas, pizza types, customer orders, and order details:

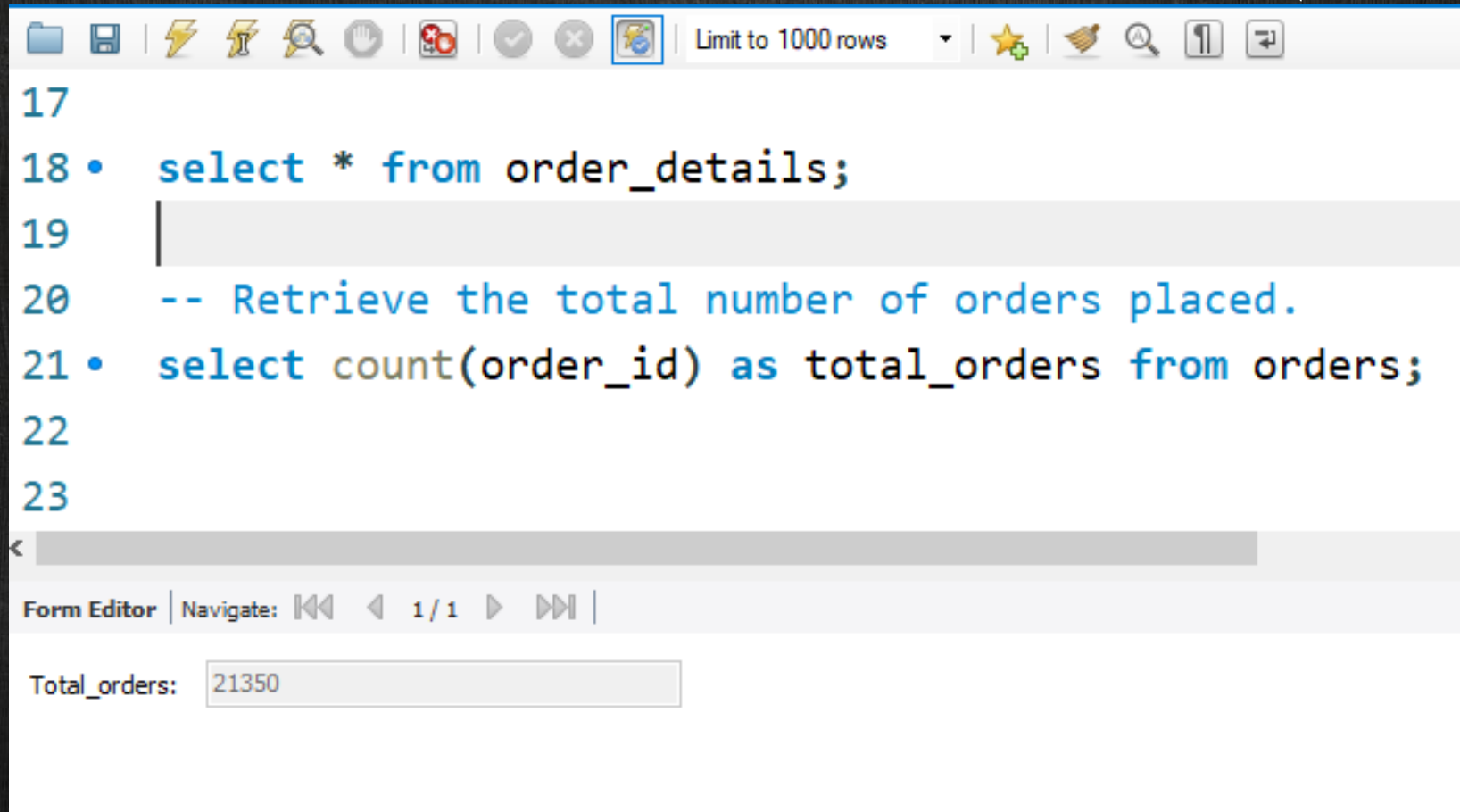
1. **pizzas**: Stores information about available pizzas.
2. **pizza_types**: Defines the categories and ingredients for each type of pizza.
3. **orders**: Manages customer orders and their details.
4. **order_details**: Stores the relationship between orders and pizzas, including quantities.



Entity-Relationship Diagram (ERD)



Q1: Retrieve the total number of orders placed.



The screenshot shows a SQL IDE window with a toolbar at the top containing icons for file operations, execution, and search. A dropdown menu indicates 'Limit to 1000 rows'. The main editor area contains the following SQL code:

```
17
18 • select * from order_details;
19
20 -- Retrieve the total number of orders placed.
21 • select count(order_id) as total_orders from orders;
22
23
```

Below the code editor is a 'Form Editor' section with a 'Navigate' bar showing '1 / 1'. At the bottom, a label 'Total_orders:' is followed by a text input field containing the value '21350'.



Q2: Calculate the total revenue generated from pizza sales.



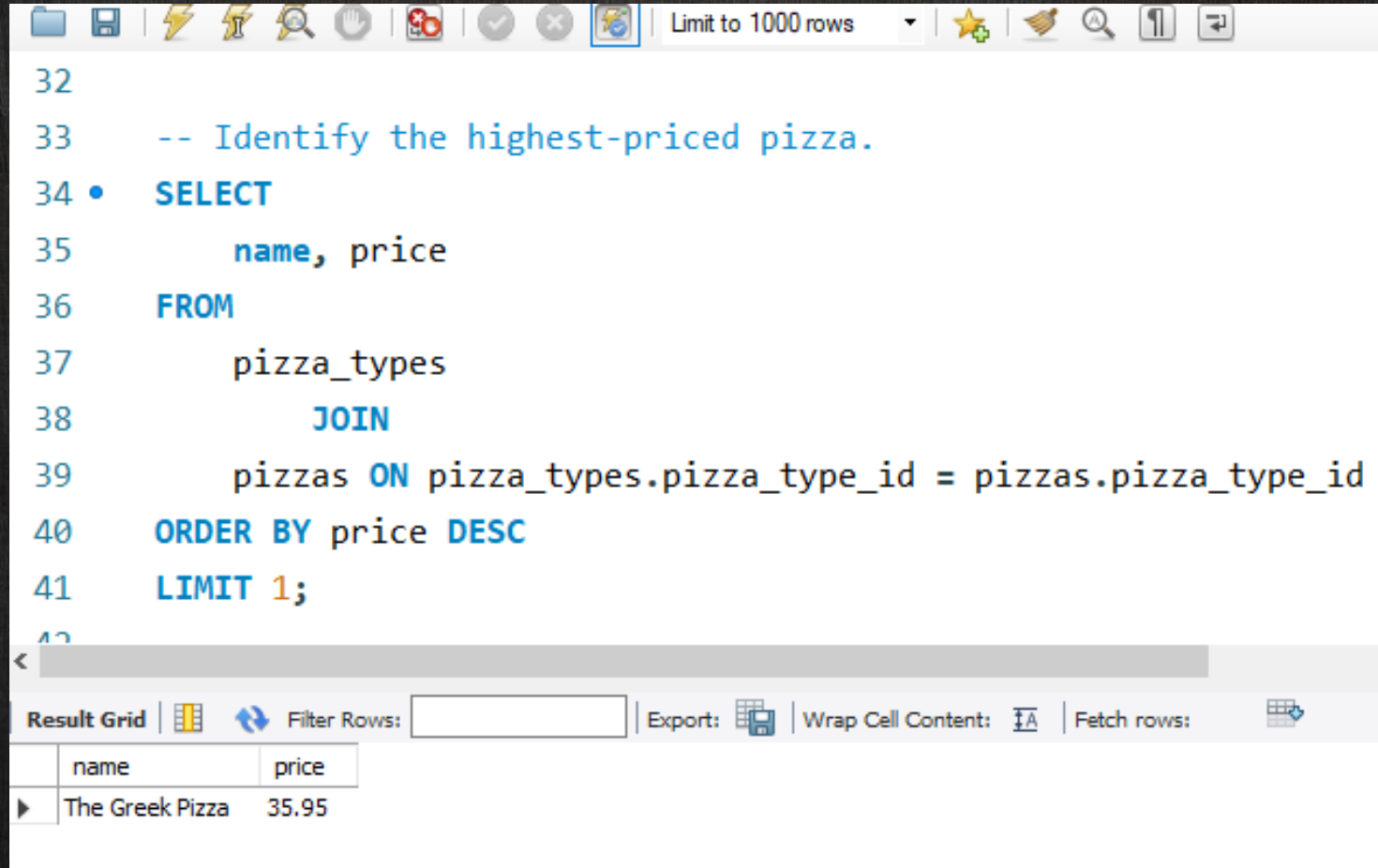
```
24
25  -- Calculate the total revenue generated from pizza sales.
26 • SELECT
27     ROUND(SUM(quantity * price), 2) AS total_revenue
28 FROM
29     order_details
30     JOIN
31     pizzas ON pizzas.pizza_id = order_details.pizza_id;
32
```

Result Grid | | Filter Rows: | Export: | Wrap Cell Content:

	total_revenue
▶	817860.05



Q3: Identify the highest-priced pizza



The screenshot shows a SQL query editor with a toolbar at the top containing icons for file operations, search, and execution. The query text is as follows:

```
32
33  -- Identify the highest-priced pizza.
34 • SELECT
35     name, price
36 FROM
37     pizza_types
38     JOIN
39     pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
40 ORDER BY price DESC
41 LIMIT 1;
```

Below the query editor, there is a 'Result Grid' section with a toolbar for filtering, exporting, and fetching rows. The result grid displays the following data:

	name	price
▶	The Greek Pizza	35.95

Q4: Identify the most common pizza size ordered.

PizzaHut Project

Limit to 1000 rows

```
44  -- Identify the most common pizza size ordered.
45  • SELECT
46      size, COUNT(order_details_id) AS order_count
47  FROM
48      pizzas
49      JOIN
50      order_details ON pizzas.pizza_id = order_details.pizza_id
51  GROUP BY size
52  ORDER BY order_count DESC
53  LIMIT 1;
```

Result Grid

	size	order_count
▶	L	18526



Q5: List the top 5 most ordered pizza types along with their quantities.



```
56  -- List the top 5 most ordered pizza types along with their quantities.
57  • SELECT
58      name, SUM(quantity) AS type_qunatity
59  FROM
60      pizza_types
61      JOIN
62      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
63      JOIN
64      order_details ON order_details.pizza_id = pizzas.pizza_id
65  GROUP BY name
66  ORDER BY type_qunatity DESC
67  LIMIT 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

	name	type_qunatity
▶	The Classic Deluxe Pizza	2453
	The Barbecue Chicken Pizza	2432
	The Hawaiian Pizza	2422
	The Pepperoni Pizza	2418
	The Thai Chicken Pizza	2371



Q6: find the total quantity of each pizza category ordered.

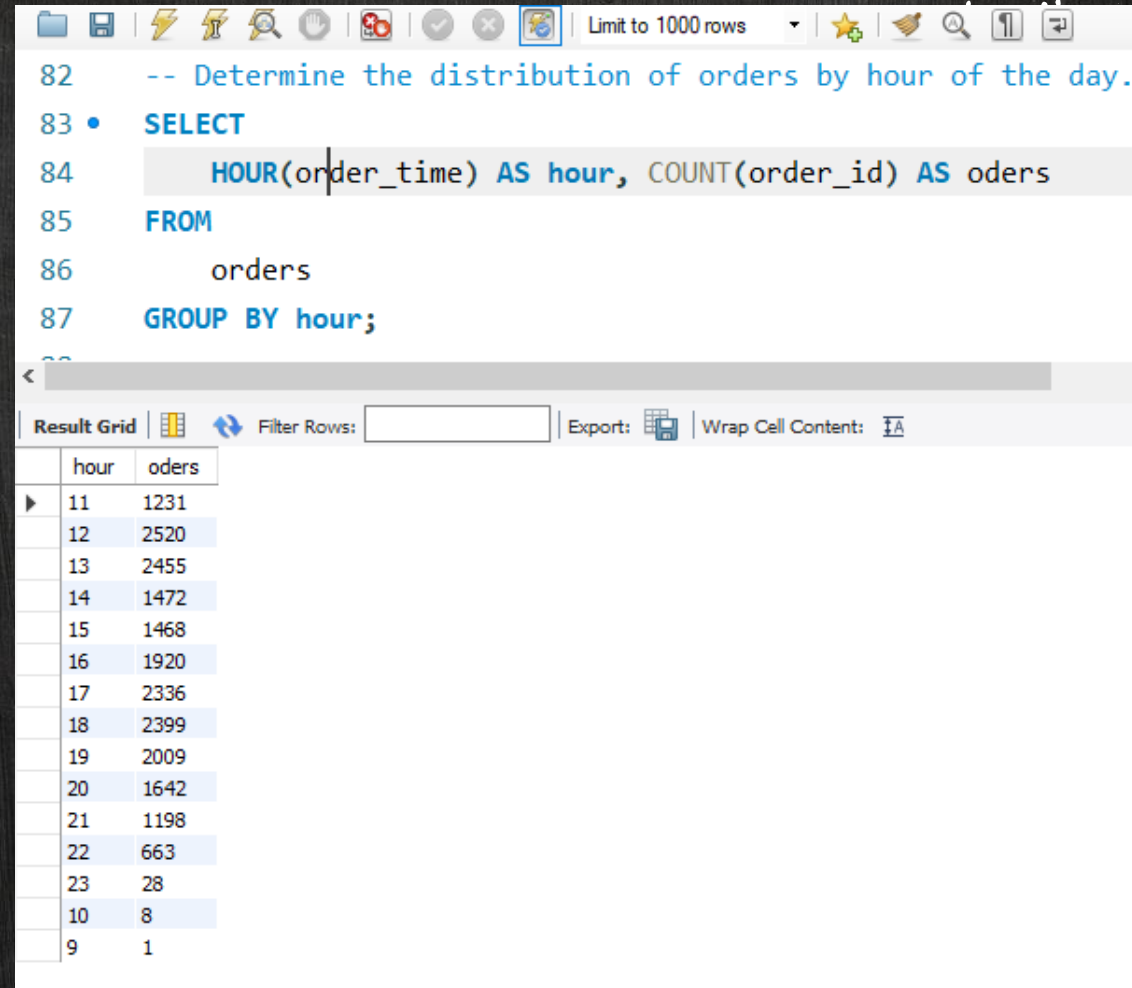
```
70  -- Join the necessary tables to find the total quantity of each pizza category ordered.
71  • SELECT
72      category, SUM(quantity) AS total_quantity
73  FROM
74      pizza_types
75      JOIN
76      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
77      JOIN
78      order_details ON order_details.pizza_id = pizzas.pizza_id
79  GROUP BY category
80  ORDER BY total_quantity DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	category	total_quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050



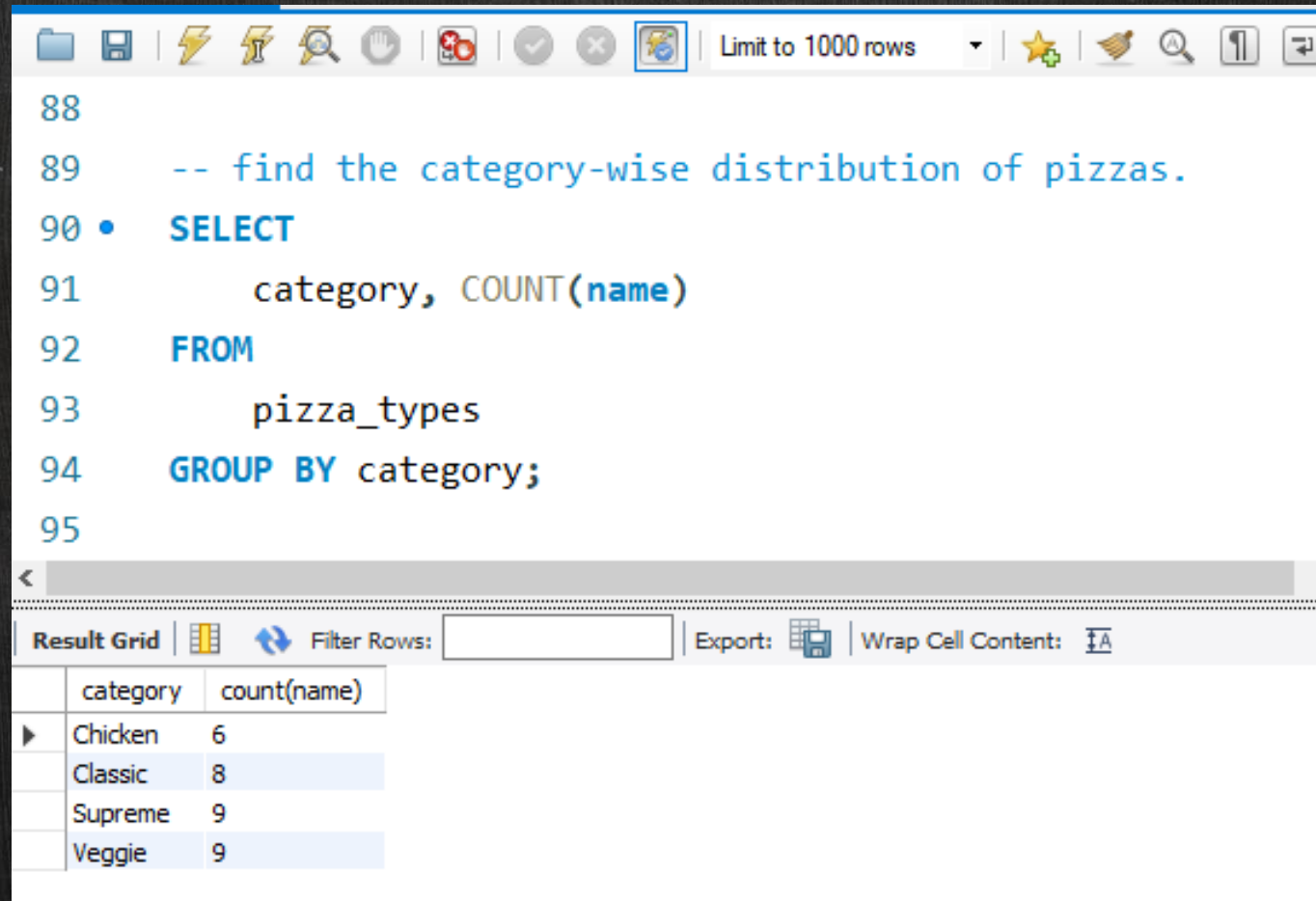
Q7: Determine the distribution of orders by hour of the day.



```
82  -- Determine the distribution of orders by hour of the day.
83  •  SELECT
84      HOUR(order_time) AS hour, COUNT(order_id) AS orders
85  FROM
86      orders
87  GROUP BY hour;
```

	hour	orders
▶	11	1231
	12	2520
	13	2455
	14	1472
	15	1468
	16	1920
	17	2336
	18	2399
	19	2009
	20	1642
	21	1198
	22	663
	23	28
	10	8
	9	1

Q8: find the category-wise distribution of pizzas.



```
88
89  -- find the category-wise distribution of pizzas.
90 • SELECT
91     category, COUNT(name)
92 FROM
93     pizza_types
94 GROUP BY category;
95
```

category	count(name)
Chicken	6
Classic	8
Supreme	9
Veggie	9

Q9: Group the orders by date and calculate the average number of pizzas ordered per day.

```
97  -- Group the orders by date and calculate the average number of pizzas ordered per day.
98  • SELECT
99      ROUND(AVG(quantity), 0) AS avergae_pizzas_ordered_per_day
100 FROM
101     (SELECT
102         order_date, SUM(quantity) AS quantity
103     FROM
104         orders
105     JOIN order_details ON orders.order_id = order_details.order_id
106     GROUP BY order_date) AS oders_per_day;
107
```

Result Grid |  Filter Rows: | Export:  | Wrap Cell Content: 

	avergae_pizzas_ordered_per_day
--	--------------------------------

▶	138
---	-----



Q10: Determine the top 3 most ordered pizza types based on revenue. i.e.ve the total number of orders placed.



```
109  -- Determine the top 3 most ordered pizza types based on revenue.
110 •  SELECT
111      name, SUM(quantity * price) AS revenue
112  FROM
113      pizza_types
114      JOIN
115      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
116      JOIN
117      order_details ON pizzas.pizza_id = order_details.pizza_id
118  GROUP BY name
119  ORDER BY revenue DESC
120  LIMIT 3;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

	name	revenue
▶	The Thai Chicken Pizza	43434.25
	The Barbecue Chicken Pizza	42768
	The California Chicken Pizza	41409.5





Q11: Calculate the percentage contribution of each pizza type to total revenue.



```
124  -- Calculate the percentage contribution of each pizza type to total revenue.
125  •  SELECT
126      category,
127      ROUND((SUM(quantity * price) / (SELECT
128          SUM(quantity * price) FROM order_details
129          JOIN
130          pizzas ON pizzas.pizza_id = order_details.pizza_id)) * 100, 2) AS revenue
131  FROM pizza_types
132      JOIN
133      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
134      JOIN
135      order_details ON pizzas.pizza_id = order_details.pizza_id
136  GROUP BY category
137  ORDER BY revenue DESC;
```

< **Result Grid** | | Filter Rows: | Export: | Wrap Cell Content:

	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68





Q12: Analyze the cumulative revenue generated over time.



The screenshot shows a SQL IDE window with a query editor and a result grid. The query is as follows:

```
139  -- Analyze the cumulative revenue generated over time.
140  •  SELECT
141      order_date, SUM(revenue) over(order by order_date) AS cumulative_rev
142  FROM
143      (SELECT
144          order_date, ROUND(SUM(quantity * price), 2) AS revenue
145      FROM
146          order_details
147      JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id
148      JOIN orders ON order_details.order_id = orders.order_id
149      GROUP BY order_date) AS sales;
```

The result grid shows the following data:

order_date	cumulative_rev
2015-01-01	2713.85
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6
2015-01-05	11929.55
2015-01-06	14358.5
2015-01-07	16560.7
2015-01-08	19399.05

Result 37 x



Q13: Determine the top 3 most ordered pizza types based on revenue for each pizza category.



```
154 • SELECT name, revenue
155 FROM (
156     SELECT pizza_name AS name, category, revenue,
157            RANK() OVER (PARTITION BY category ORDER BY revenue DESC) AS rn
158 FROM (
159     SELECT pizza_types.category, pizza_types.name AS pizza_name,
160            SUM(order_details.quantity * pizzas.price) AS revenue
161 FROM pizza_types
162 JOIN pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
163 JOIN order_details ON pizzas.pizza_id = order_details.pizza_id
164 GROUP BY pizza_types.category, pizza_types.name
165 ) AS a
166 ) AS b
167 WHERE rn <= 3;
```

Result Grid

name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5
The Classic Deluxe Pizza	38180.5
The Hawaiian Pizza	32273.25
The Pepperoni Pizza	30161.75
The Spicy Italian Pizza	34831.25
The Italian Supreme Pizza	33476.75
The Sicilian Pizza	30940.5
The Four Cheese Pizza	32265.700000000065
The Mexicana Pizza	26780.75
The Five Cheese Pizza	26066.5

Result 42 x

THE END

