



WELCOME

WELCOME TO PIZZA SALES DATABASE ANALYSIS!

My name is Sakshi Rajesh Bhavsar
student of B.E Computer Science & Engineering
batch (2021-25)

I'm excited to share my project as presentation on
analyzing pizza sales data. This project aims to uncover
valuable insights from a database using SQL queries in
MYSQL Workbench.

So, let's see what insights we can uncover from the
pizza sales database!

PROJECT INDEX

- INTRODUCTION
- OBJECTIVE
- DATA MODEL
- ENTITY RELATIONSHIP
- SQL QUERIES FOR DATA EXPLORATION
- RESULT
- CONCLUSION



INTRODUCTION

PIZZA SALES PROJECT

USING SQL QUERIES

This project aims to provide valuable insights into the dynamics of pizza sales by analyzing comprehensive data on orders, customers, and operations.

By leveraging SQL queries, we will uncover trends and other pizza sales.

PROJECT OBJECTIVES

- Identify top-selling pizza types and toppings
- Analyze customer purchasing behavior and demographics
- Optimize inventory management and logistics
- Improve marketing strategies and target high-value customers

DATA MODEL

Table:orders

<u>order_id</u>	int PK
order_date	date
order_time	time

Table:pizzas

pizza_id	text
pizza_type_id	text
size	text
price	double

DATA MODEL

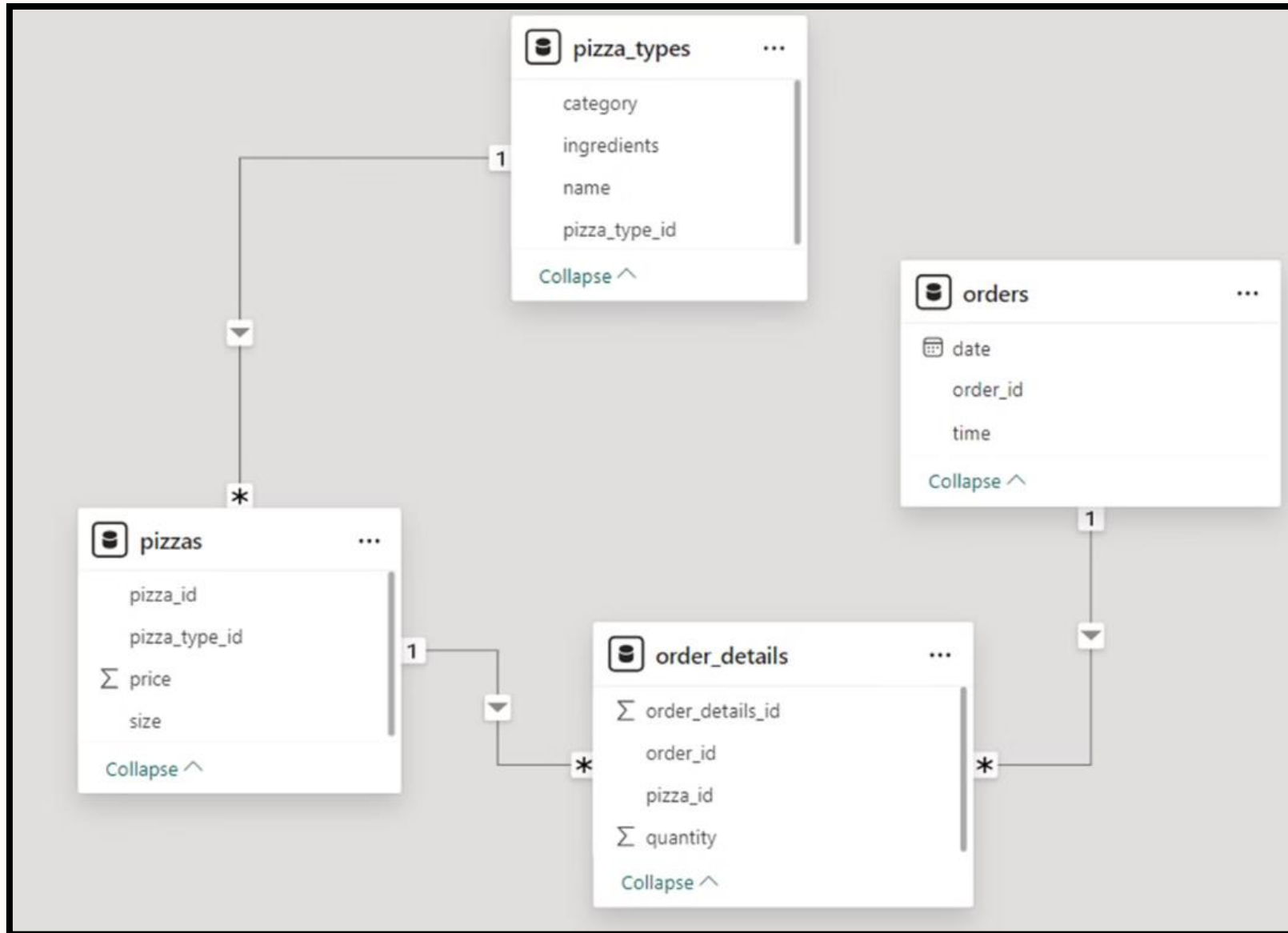
Table: **pizza_types**

pizza_type_id	text
name	text
category	text
ingredients	text

Table: **orders_details**

<u>order_details_id</u>	int PK
order_id	int
pizza_id	text
quantity	int

ENTITY RELATIONSHIPS



SQL QUERIES FOR DATA EXPLORATION

1 Top-Selling Pizzas

Identify the most popular pizza types and toppings by analyzing order data.

3 Inventory Insights

Monitor pizza quantity, order by hour, identify pizza sales, and revenue.

2 Customer Segmentation

Classify customers based on factors like order frequency, average order value, and demographic data.

4 Operational Efficiency

Analyze order processing times, quantity, size, price and pizza category.

Query : Retrieve the total number of orders placed.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with the 'pizzahut' database selected. The 'orders' table is highlighted under the 'Tables' section. The main editor window shows a SQL query: `-- Retrieve the total number of orders placed.` followed by `select count(order_id) as total_orders from orders;`. The 'Result Grid' at the bottom shows a single row with the value 21350 for the column 'total_orders'. The interface includes a menu bar (File, Edit, View, Query, Database, Server, Tools, Scripting, Help) and a toolbar with various icons for file operations, query execution, and result viewing.

total_orders
21350

Query : Calculate the total revenue generated from pizza sales.

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows the 'pizzahut' database selected, with tables 'orders', 'orders_details', 'pizza_types', and 'pizzas' visible. The main editor window displays a SQL query to calculate the total revenue from pizza sales. The query is as follows:

```
1  -- Calculate the total revenue generated from pizza sales.
2
3  •  SELECT
4      ROUND(SUM(orders_details.quantity * pizzas.price),
5             2) AS total_revenue_sales
6  FROM
7      orders_details
8      JOIN
9      pizzas ON pizzas.pizza_id = orders_details.pizza_id;
```

Below the query editor, the 'Result Grid' tab is active, showing the result of the query:

total_revenue_sales
817860.05

The bottom of the interface shows the 'Administration' and 'Schemas' tabs, and the 'Information' pane.

Query : Identify the highest-priced pizza

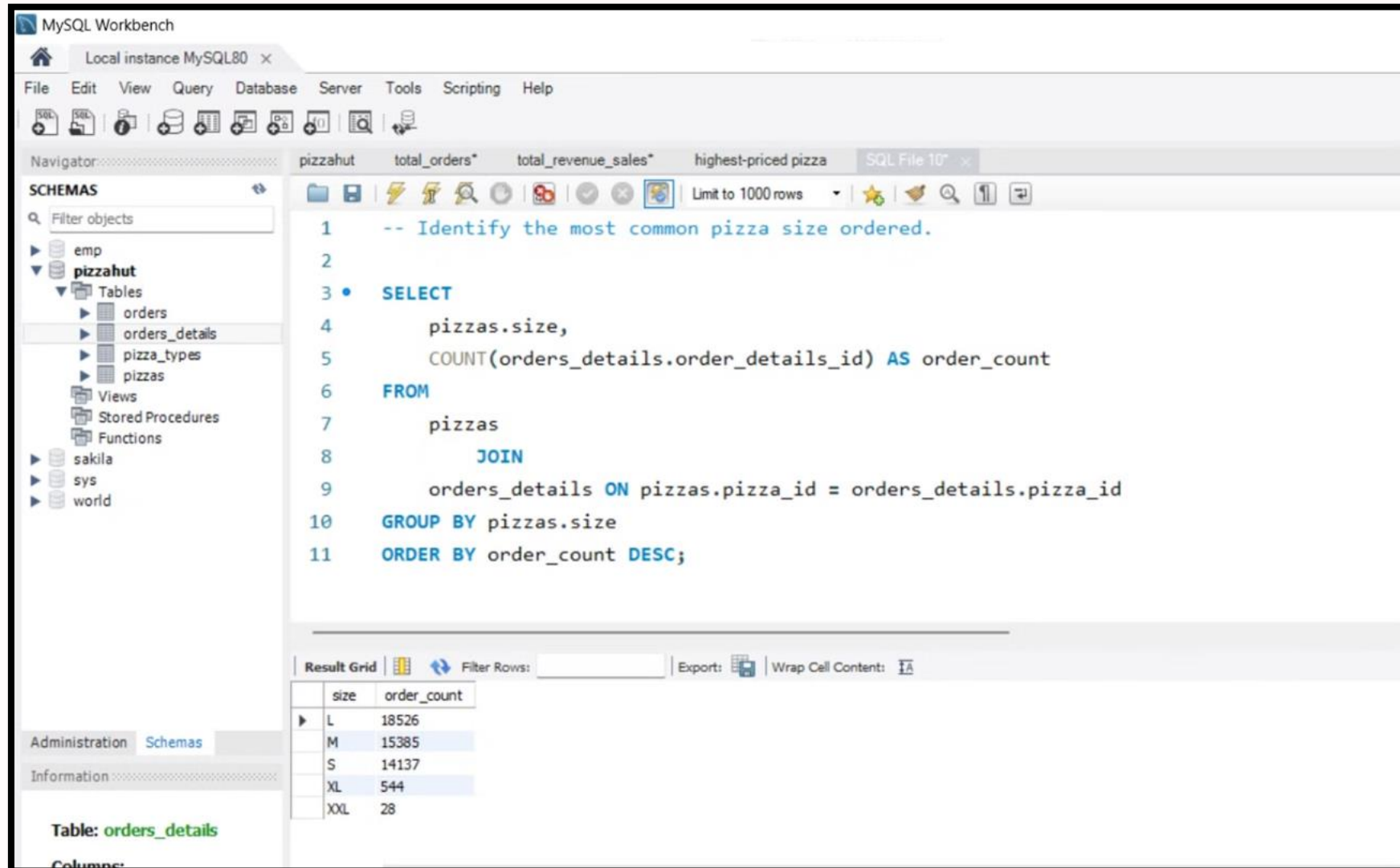
The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view with databases: emp, pizzahut, sakila, sys, and world. The 'pizzahut' database is expanded, showing tables: orders, orders_details, pizza_types, and pizzas. The main editor window has a tab titled 'highest-priced pizza' and contains the following SQL query:

```
1  -- Identify the highest-priced pizza.
2
3  SELECT
4      pizzas.price, pizza_types.name
5  FROM
6      pizzas
7      JOIN
8      pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id
9  ORDER BY pizzas.price DESC
10 LIMIT 1;
```

Below the query editor, the 'Result Grid' tab is active, displaying the results of the query in a table:

price	name
35.95	The Greek Pizza

Query : Identify the most common pizza size ordered.



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'pizzahut' selected. The main editor shows a SQL query to identify the most common pizza size ordered. The query is as follows:

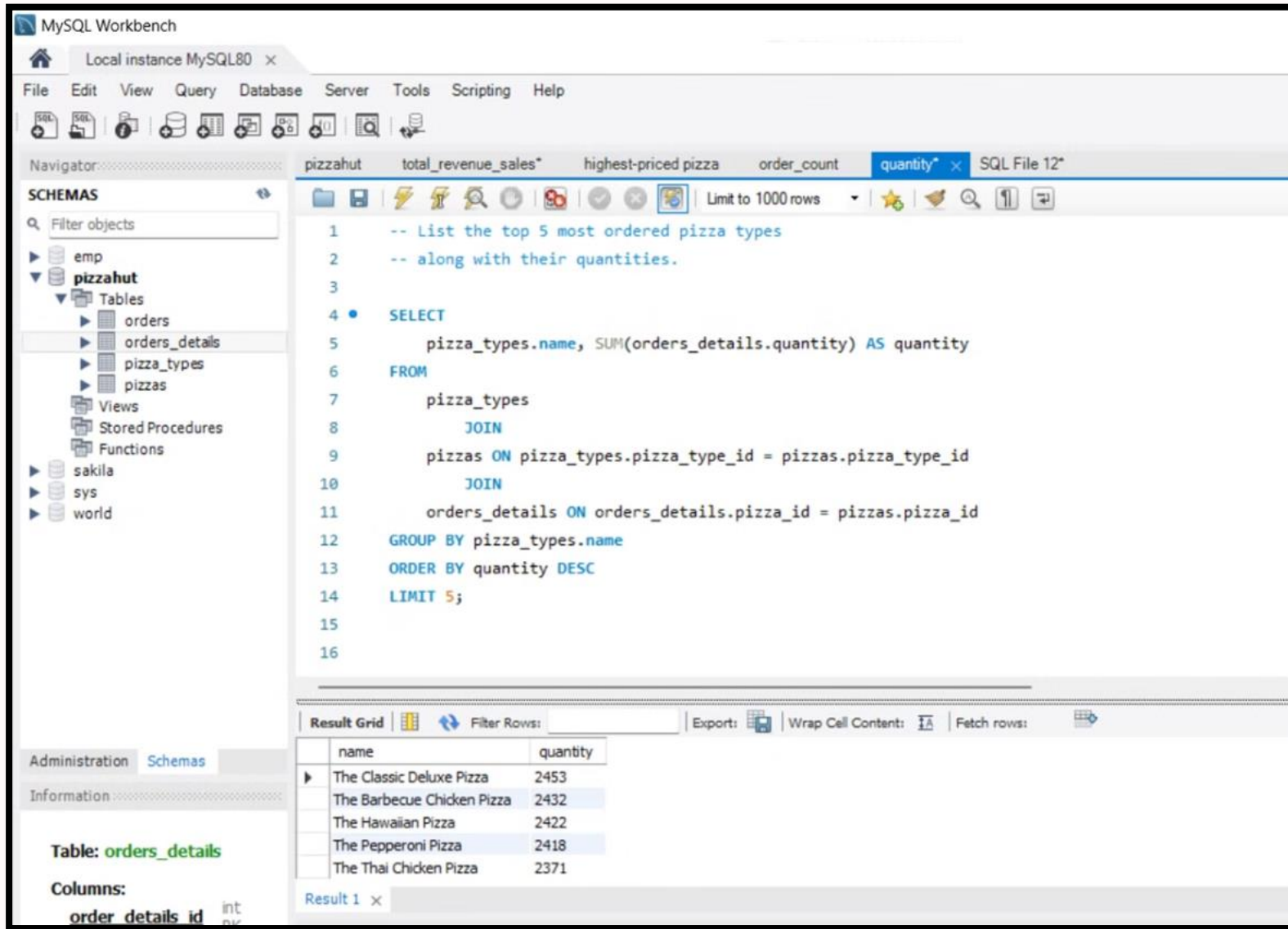
```
1  -- Identify the most common pizza size ordered.
2
3  •  SELECT
4      pizzas.size,
5      COUNT(orders_details.order_details_id) AS order_count
6  FROM
7      pizzas
8      JOIN
9      orders_details ON pizzas.pizza_id = orders_details.pizza_id
10 GROUP BY pizzas.size
11 ORDER BY order_count DESC;
```

The 'Result Grid' at the bottom shows the following data:

size	order_count
L	18526
M	15385
S	14137
XL	544
XXL	28

The interface also shows the 'Administration' and 'Schemas' tabs at the bottom left, and the 'Table: orders_details' and 'Columns:' labels at the bottom.

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

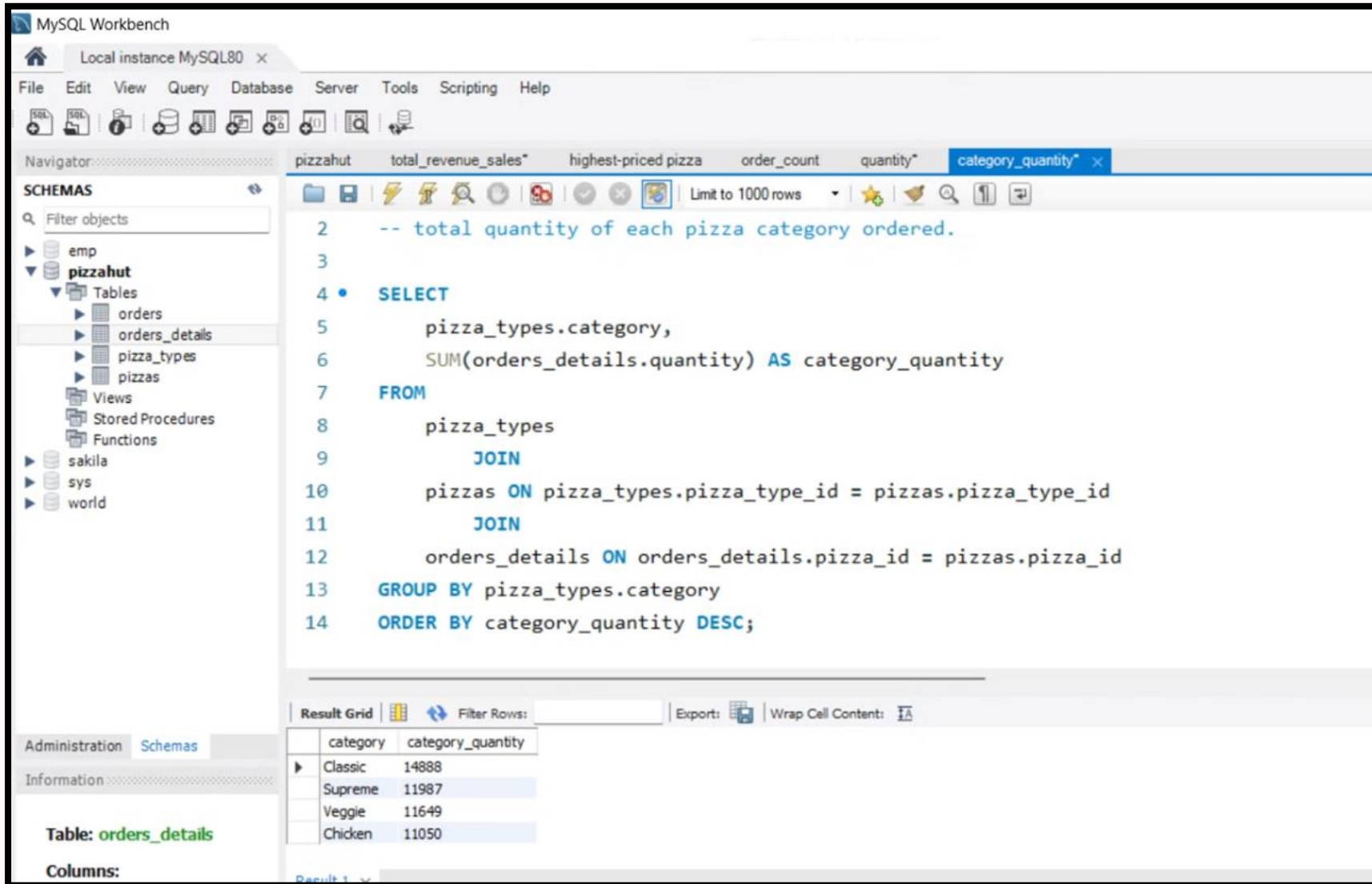


The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of databases, with 'pizzahut' expanded to show tables like 'orders', 'orders_details', 'pizza_types', and 'pizzas'. The main editor window contains a SQL query to list the top 5 most ordered pizza types. Below the query, the 'Result Grid' shows the output of the query, displaying the pizza names and their corresponding quantities.

```
1  -- List the top 5 most ordered pizza types
2  -- along with their quantities.
3
4  •  SELECT
5      pizza_types.name, SUM(orders_details.quantity) AS quantity
6  FROM
7      pizza_types
8      JOIN
9      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
10     JOIN
11     orders_details ON orders_details.pizza_id = pizzas.pizza_id
12 GROUP BY pizza_types.name
13 ORDER BY quantity DESC
14 LIMIT 5;
```

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

Query : Join the necessary tables to find the total quantity of each pizza category ordered.



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with the 'pizzahut' database selected. The main editor window contains a SQL query to find the total quantity of each pizza category ordered. The query is as follows:

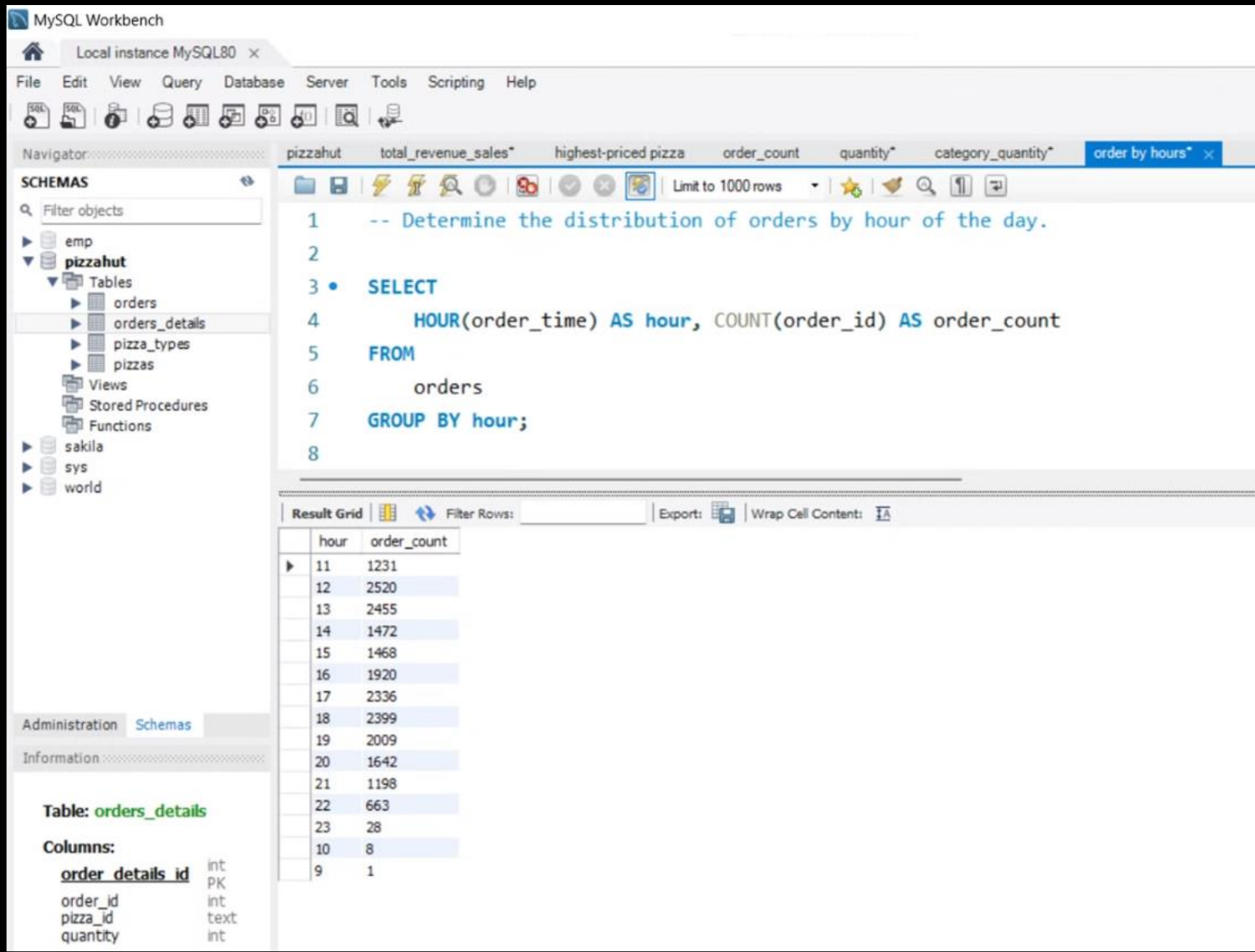
```
-- total quantity of each pizza category ordered.

SELECT
    pizza_types.category,
    SUM(orders_details.quantity) AS category_quantity
FROM
    pizza_types
JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY category_quantity DESC;
```

Below the query editor, the 'Result Grid' tab is active, showing the results of the query. The results are displayed in a table with two columns: 'category' and 'category_quantity'.

category	category_quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

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



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with the 'pizzahut' database selected. The main editor window contains a SQL query to determine the distribution of orders by hour of the day. The query is as follows:

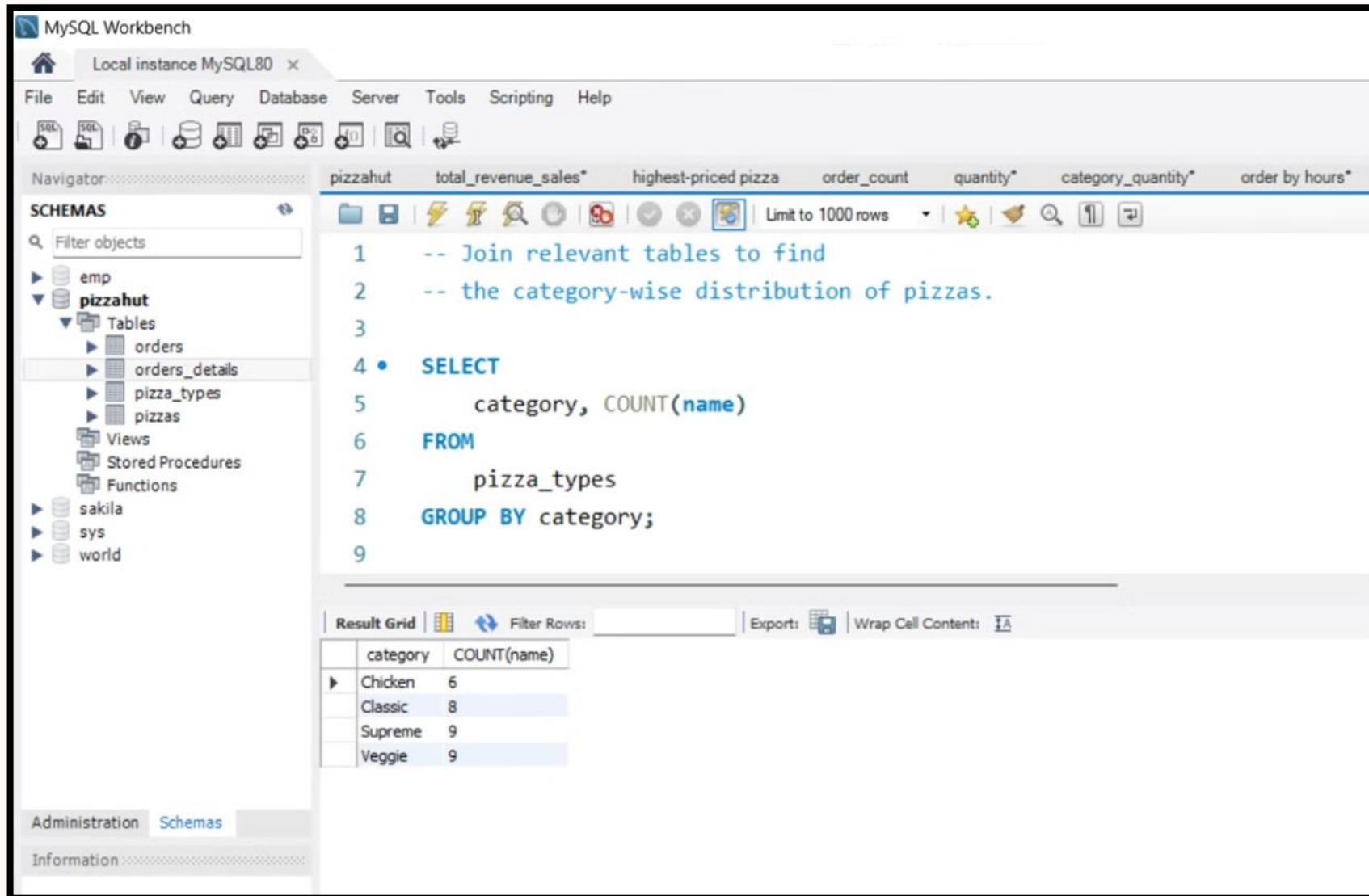
```
1  -- Determine the distribution of orders by hour of the day.
2
3  • SELECT
4      HOUR(order_time) AS hour, COUNT(order_id) AS order_count
5  FROM
6      orders
7  GROUP BY hour;
```

The 'Result Grid' at the bottom shows the output of the query, displaying the hour and the corresponding order count. The results are as follows:

hour	order_count
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

The bottom left of the interface shows the 'Table: orders_details' with its columns: order_details_id (int, PK), order_id (int), pizza_id (text), and quantity (int).

Query : Join relevant tables to find the category-wise distribution of pizzas.



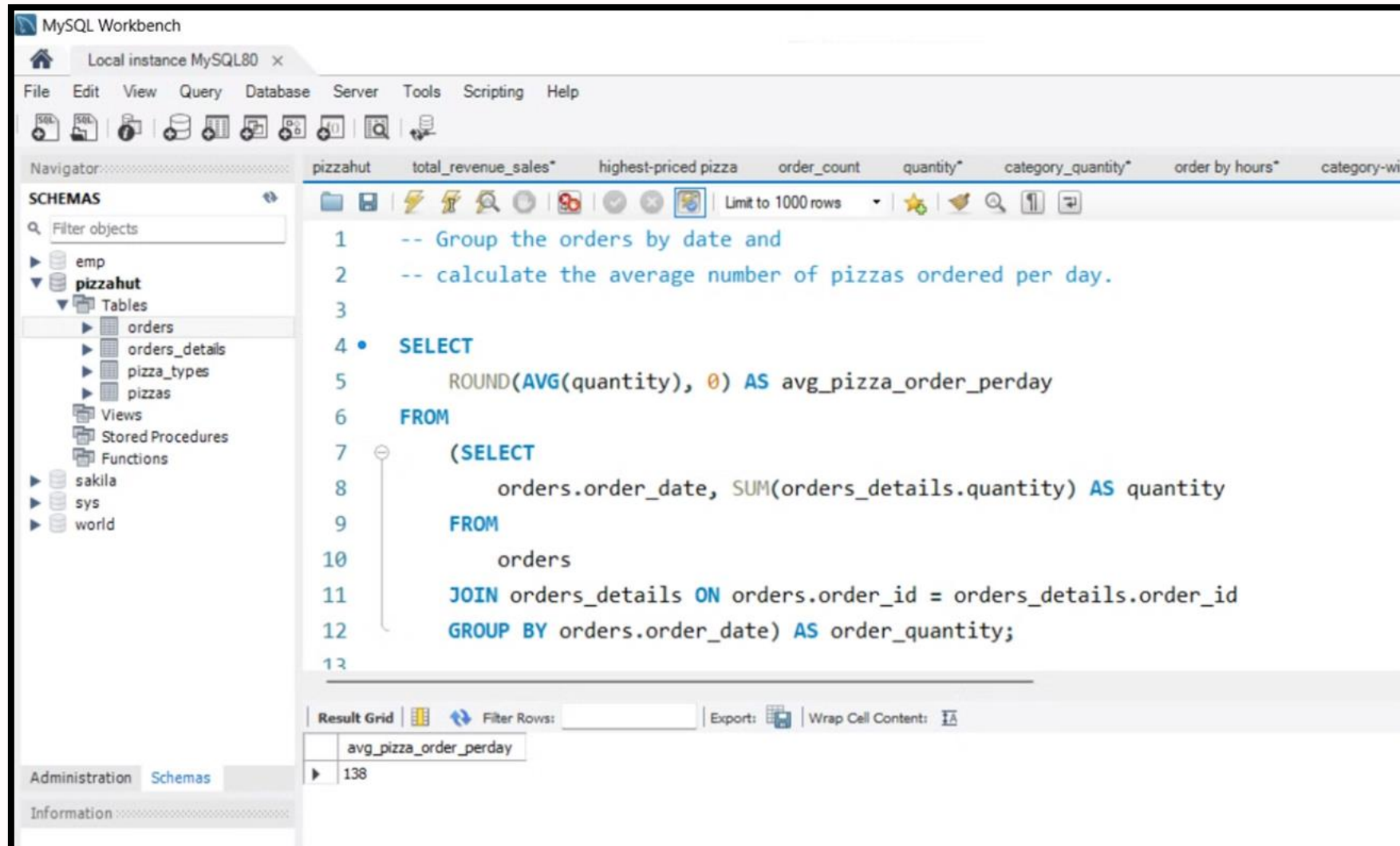
The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of databases, with 'pizzahut' selected and its tables ('orders', 'orders_details', 'pizza_types', 'pizzas') expanded. The main editor displays a SQL query:

```
1  -- Join relevant tables to find
2  -- the category-wise distribution of pizzas.
3
4  •  SELECT
5      category, COUNT(name)
6  FROM
7      pizza_types
8  GROUP BY category;
```

Below the query editor, the 'Result Grid' shows the output of the query:

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

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



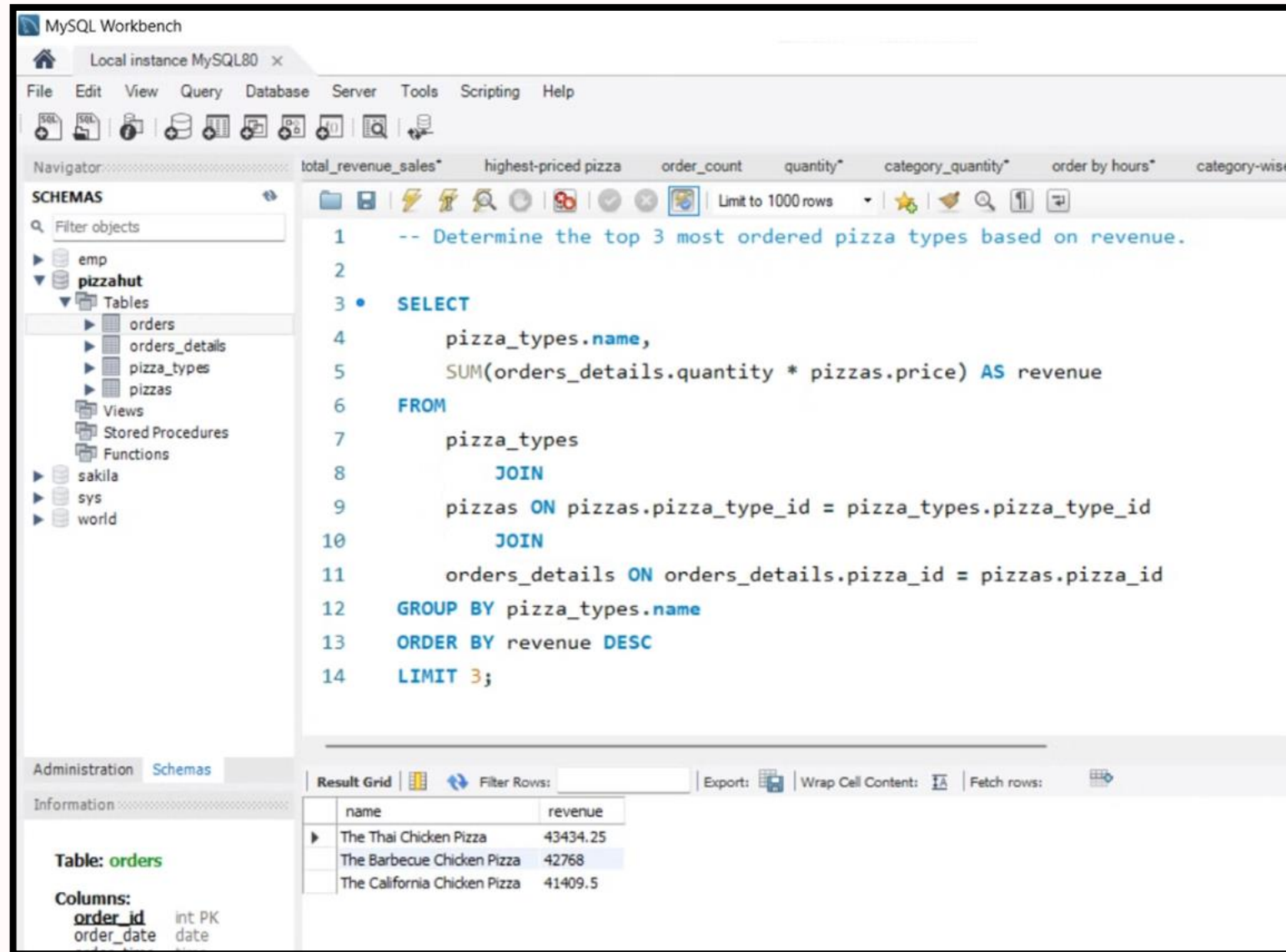
The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows the 'pizzahut' database selected, with tables like 'orders', 'orders_details', 'pizza_types', and 'pizzas' listed. The main editor displays a SQL query to calculate the average number of pizzas ordered per day. The query uses a subquery to first group orders by date and sum their quantities, then calculates the average of these sums.

```
1  -- Group the orders by date and
2  -- calculate the average number of pizzas ordered per day.
3
4  •  SELECT
5      ROUND(AVG(quantity), 0) AS avg_pizza_order_perday
6  FROM
7      (SELECT
8          orders.order_date, SUM(orders_details.quantity) AS quantity
9      FROM
10         orders
11      JOIN orders_details ON orders.order_id = orders_details.order_id
12      GROUP BY orders.order_date) AS order_quantity;
```

The 'Result Grid' at the bottom shows a single row with the value 138 for the column 'avg_pizza_order_perday'.

avg_pizza_order_perday
138

Query : Determine the top 3 most ordered pizza types based on revenue.

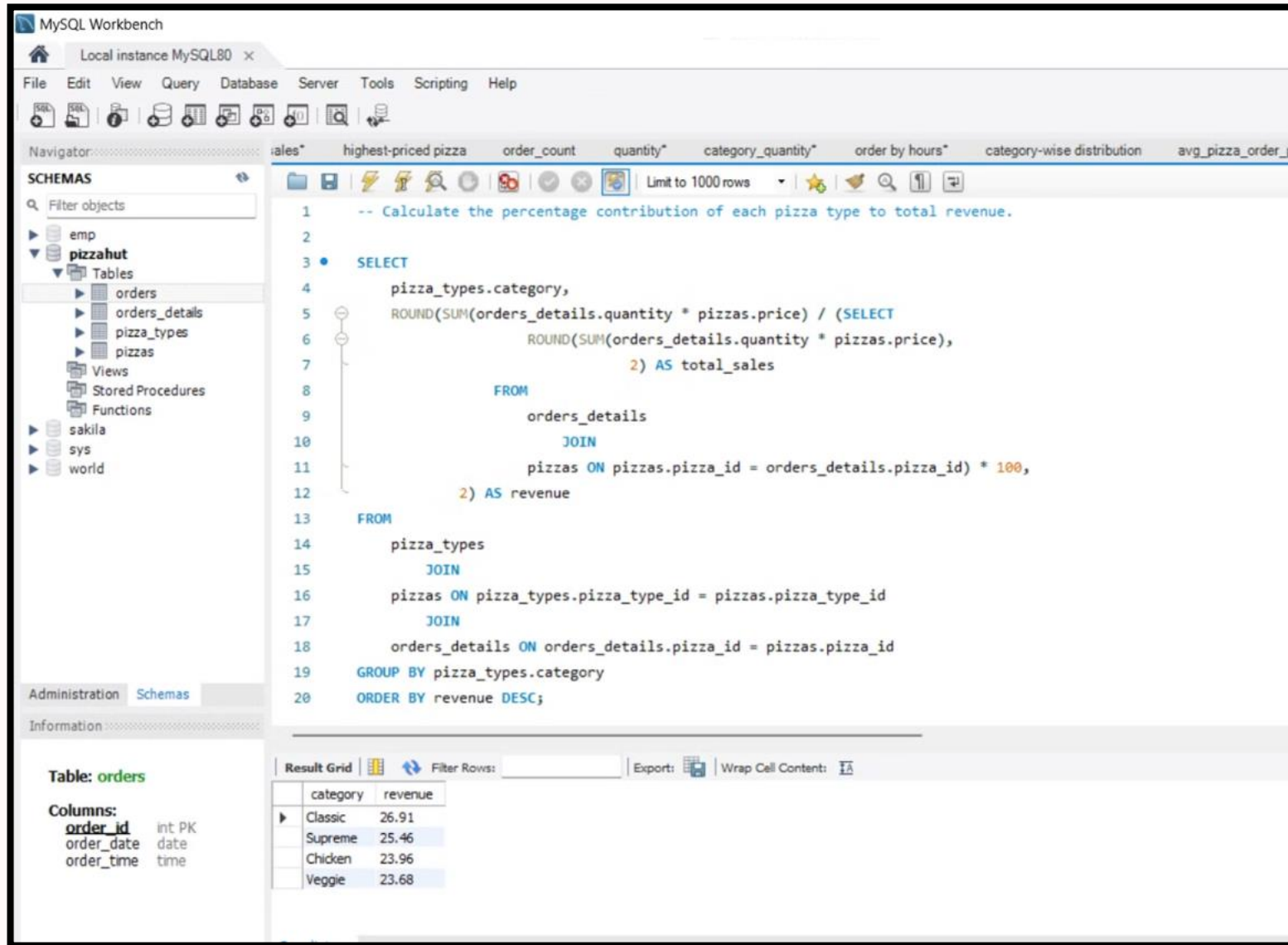


The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of databases, with 'pizzahut' expanded to show tables like 'orders', 'orders_details', 'pizza_types', and 'pizzas'. The main editor window contains a SQL query to find the top 3 most ordered pizza types based on revenue. The query uses a SELECT statement with pizza_types.name, SUM of orders_details.quantity multiplied by pizzas.price as revenue, joined with pizza_types, pizzas, and orders_details. It is grouped by pizza_types.name, ordered by revenue in descending order, and limited to 3 rows. The bottom pane shows the 'Result Grid' with the following data:

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

Below the result grid, the 'Table: orders' information is displayed, showing columns: order_id (int PK) and order_date (date).

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



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with the 'pizzahut' database selected. The main editor window contains a SQL query to calculate the percentage contribution of each pizza type to total revenue. The query is as follows:

```
-- Calculate the percentage contribution of each pizza type to total revenue.

SELECT
  pizza_types.category,
  ROUND(SUM(orders_details.quantity * pizzas.price) / (SELECT
    ROUND(SUM(orders_details.quantity * pizzas.price),
      2) AS total_sales
    FROM
      orders_details
      JOIN
        pizzas ON pizzas.pizza_id = orders_details.pizza_id) * 100,
    2) AS revenue
FROM
  pizza_types
  JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
  JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY revenue DESC;
```

The bottom right of the interface shows the 'Result Grid' with the following data:

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

Query : Analyze the cumulative revenue generated over time.

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view with databases 'emp', 'pizzahut', 'sakila', 'sys', and 'world'. The 'pizzahut' database is expanded, showing tables 'orders', 'orders_details', 'pizza_types', and 'pizzas'. Below this, the 'Table: orders' is selected, showing its columns: 'order_id' (int PK), 'order_date' (date), and 'order_time' (time). The main editor displays a SQL query to calculate cumulative revenue. The query is as follows:

```
1  -- Analyze the cumulative revenue generated over time.
2
3  * select order_date,
4     sum(revenue)
5     over(order by order_date) as cum_revenue
6  from
7     (select orders.order_date,
8        sum(orders_details.quantity*pizzas.price) as revenue
9     from orders_details join pizzas
10    on orders_details.pizza_id=pizzas.pizza_id
11    join orders
12    on orders.order_id=orders_details.order_id
13    group by orders.order_date) as sales;
```

Below the query editor, the 'Result Grid' tab is active, showing the results of the query. The grid has two columns: 'order_date' and 'cum_revenue'. The results are as follows:

order_date	cum_revenue
2015-01-01	2713.8500000000004
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
2015-01-09	21526.4
2015-01-10	23990.350000000002
2015-01-11	25862.65

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

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays the database structure, including the 'pizzahut' database with tables like 'orders', 'orders_details', 'pizza_types', and 'pizzas'. The main editor displays a SQL query to find the top 3 most ordered pizza types by revenue for each category. The query uses a subquery with a window function 'rank()' to order pizzas by revenue within each category, then selects the top 3 from each group.

```
1  -- Determine the top 3 most ordered pizza types
2  -- based on revenue for each pizza category.
3
4  * select name, revenue from
5  (select category, name, revenue,
6   rank() over(partition by category order by revenue desc) as rn
7   from
8   (select pizza_types.category, pizza_types.name,
9    sum((orders_details.quantity)*pizzas.price) as revenue
10   from pizza_types join pizzas
11   on pizza_types.pizza_type_id=pizzas.pizza_type_id
12   join orders_details
13   on orders_details.pizza_id=pizzas.pizza_id
14   group by pizza_types.category, pizza_types.name) as a) as b
15  where rn<=3;
```

The 'Result Grid' at the bottom shows the output of the query, listing the top 3 pizzas by revenue for each category. The results are as follows:

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

RESULT - ANALYZING PIZZA SALES DATA

- Total number of orders placed (21350)
- Total revenue generated from pizza sales (817860.05)
- Identification of the highest-priced and most commonly ordered pizza types (pizza name- The Greek Pizza, Price- 35.95)
- The most common pizza size ordered (size of Large with order count of 18526)
- The most ordering pizza type quantities (name- The classis deluxe pizza with quantity of 2453)
- Average number of pizzas ordered per day (138)
- Determination of the top pizza types by revenue (The Thai chicken pizza of revenue- 4343.25)
- Evaluation of the percentage contribution of each pizza type to total revenue.
- Examination of the cumulative revenue over time.

CONCLUSION

By analyzing this data,
we have gain valuable insights to inform business decisions
and optimize our pizza operations.

Follow my GitHub for complete SQL project and it's used raw database:

<https://github.com/bsakshi2019/Pizzahut-sales-sql-project.git>

Reference:

[pizza-sales---SQL/Questions.txt at main · Ayushi0214/pizza-sales---SQL · GitHub](#)



Thank
You