

SQL PROJECT ON PIZZA SALES



Hello!!
My name is Asif Shaikh
In this project
I have utilized SQL queries to solve the questions that
were related to Pizza Sales.



RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

The screenshot shows a SQL IDE interface. On the left, the 'Navigator' pane displays the 'pizzhut' database schema with tables: order_details, orders, pizza_types, and pizzas. The main editor, titled 'SQL File 4*', contains the following SQL code:

```
1 • create database pizzhut;  
2 • use pizzhut;  
3 • select count(order_id) as total_orders from orders;
```

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

total_orders
21350

CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

The screenshot shows a database management interface. On the left, the 'SCHEMAS' panel displays a tree view for the 'pizzhut' database, including tables like 'order_details', 'orders', 'pizza_types', and 'pizzas'. The main area contains a SQL query editor with the following code:

```
1 SELECT
2 ROUND (SUM(order_details.quantity * pizzas.price),
3 2) AS total_sales
4 FROM
5 order_details
6 JOIN
7 pizzas ON pizzas.pizza_id = order_details.pizza_id;
```

Below the query editor, the 'Result Grid' is visible, showing the output of the query:

total_sales
817860.05

IDENTIFY THE HIGHEST-PRICED PIZZA.

SCHEMAS

Filter objects

pizzhut

- Tables
 - order_details
 - orders
 - pizza_types
 - pizzas
- Views
- Stored Procedures
- Functions

sakila

sys

world

```
1 • SELECT
2     pizza_types.name, pizzas.price
3 FROM
4     pizza_types
5     JOIN
6     pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
7 ORDER BY pizzas.price DESC;
```

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

	name	price
▶	The Greek Pizza	35.95
	The Greek Pizza	25.5
	The Brie Carre Pizza	23.65
	The Italian Vegetables Pizza	21
	The Spinach Supreme Pizza	20.75
	The Barbecue Chicken Pizza	20.75
	The California Chicken Pizza	20.75
	The Spicy Italian Pizza	20.75
	The Chicken Alfredo Pizza	20.75
	The Chicken Pesto Pizza	20.75
	The Italian Supreme Pizza	20.75
	The Southwest Chicken Pizza	20.75
	The Prosciutto and Arugula...	20.75
	The Pepper Salami Pizza	20.75
	The Thai Chicken Pizza	20.75
	The Soppressata Pizza	20.75
	The Spinach Pesto Pizza	20.75

Administration Schemas Information

IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

Navigator: SCHEMAS

Filter objects

pizzhut

- Tables
 - order_details
 - orders
 - pizza_types
 - pizzas
- Views
- Stored Procedures
- Functions

sakila

sys

SQL File 4* SQL File 5* SQL File 6* SQL File 7* x

Limit to 500 rows

```
1 SELECT
2     pizzas.size,
3     COUNT(order_details.order_details_id) AS order_count
4 FROM
5     pizzas
6     JOIN
7     order_details ON pizzas.pizza_id = order_details.pizza_id
8 GROUP BY pizzas.size
9 ORDER BY order_count DESC;
```

Result Grid

Filter Rows:

Export: Wrap Cell Content: IA

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

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

SCHEMAS

Filter objects

pizzhut

- Tables
 - order_details
 - orders
 - pizza_types
 - pizzas
- Views
- Stored Procedures
- Functions

sakila

sys

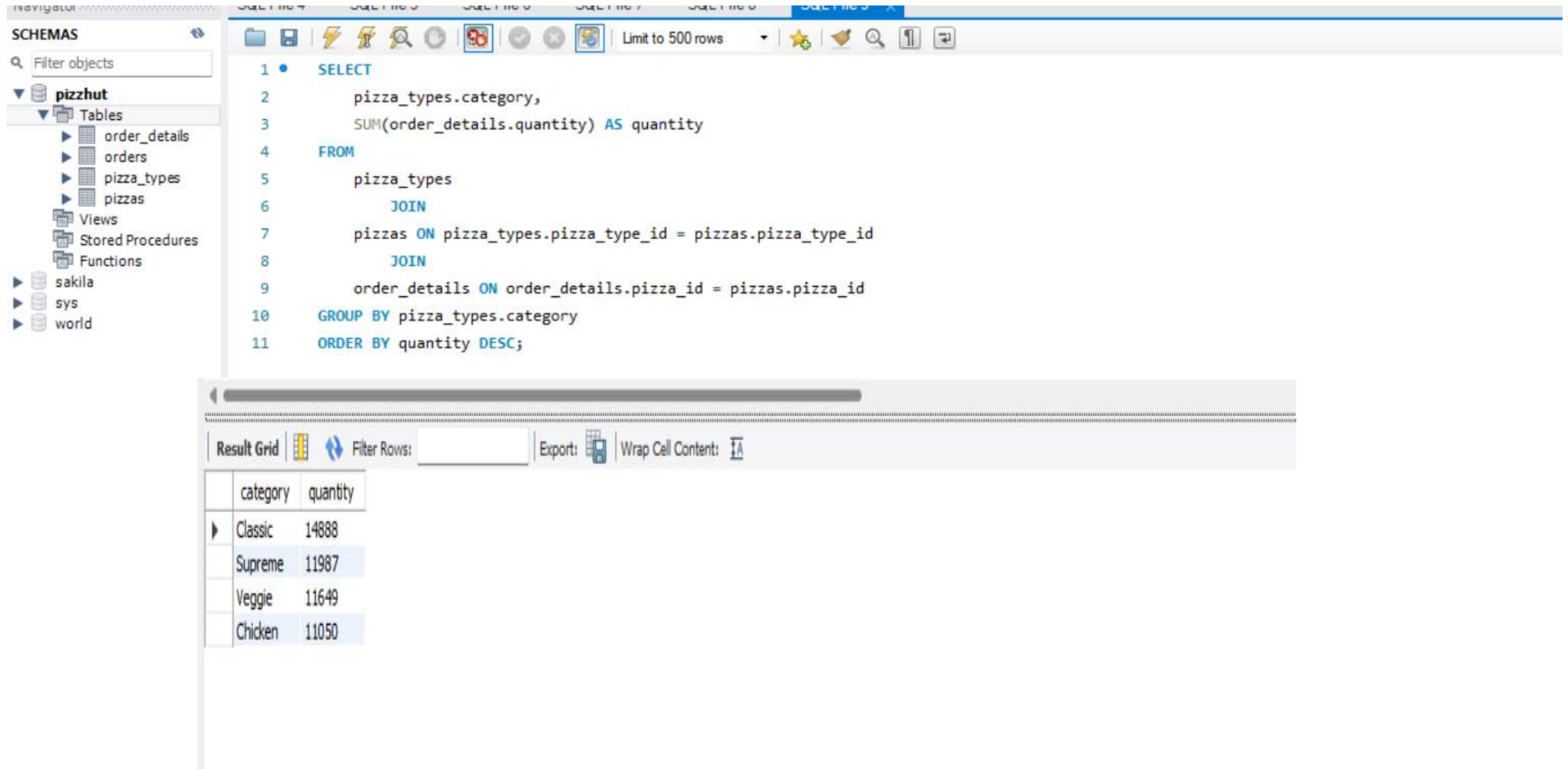
world

```
1 • SELECT
2     pizza_types.name, SUM(order_details.quantity) AS quantity
3 FROM
4     pizza_types
5     JOIN
6     pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
7     JOIN
8     order_details ON order_details.pizza_id = pizzas.pizza_id
9 GROUP BY pizza_types.name
10 ORDER BY quantity DESC
11 LIMIT 5;
```

Result Grid

	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

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



The screenshot displays a SQL IDE interface. On the left, the 'SCHEMAS' pane shows a tree view with 'pizzhut' expanded, listing tables: 'order_details', 'orders', 'pizza_types', and 'pizzas'. The main editor shows a SQL query:

```
1 SELECT
2     pizza_types.category,
3     SUM(order_details.quantity) AS quantity
4 FROM
5     pizza_types
6     JOIN
7     pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
8     JOIN
9     order_details ON order_details.pizza_id = pizzas.pizza_id
10 GROUP BY pizza_types.category
11 ORDER BY quantity DESC;
```

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

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

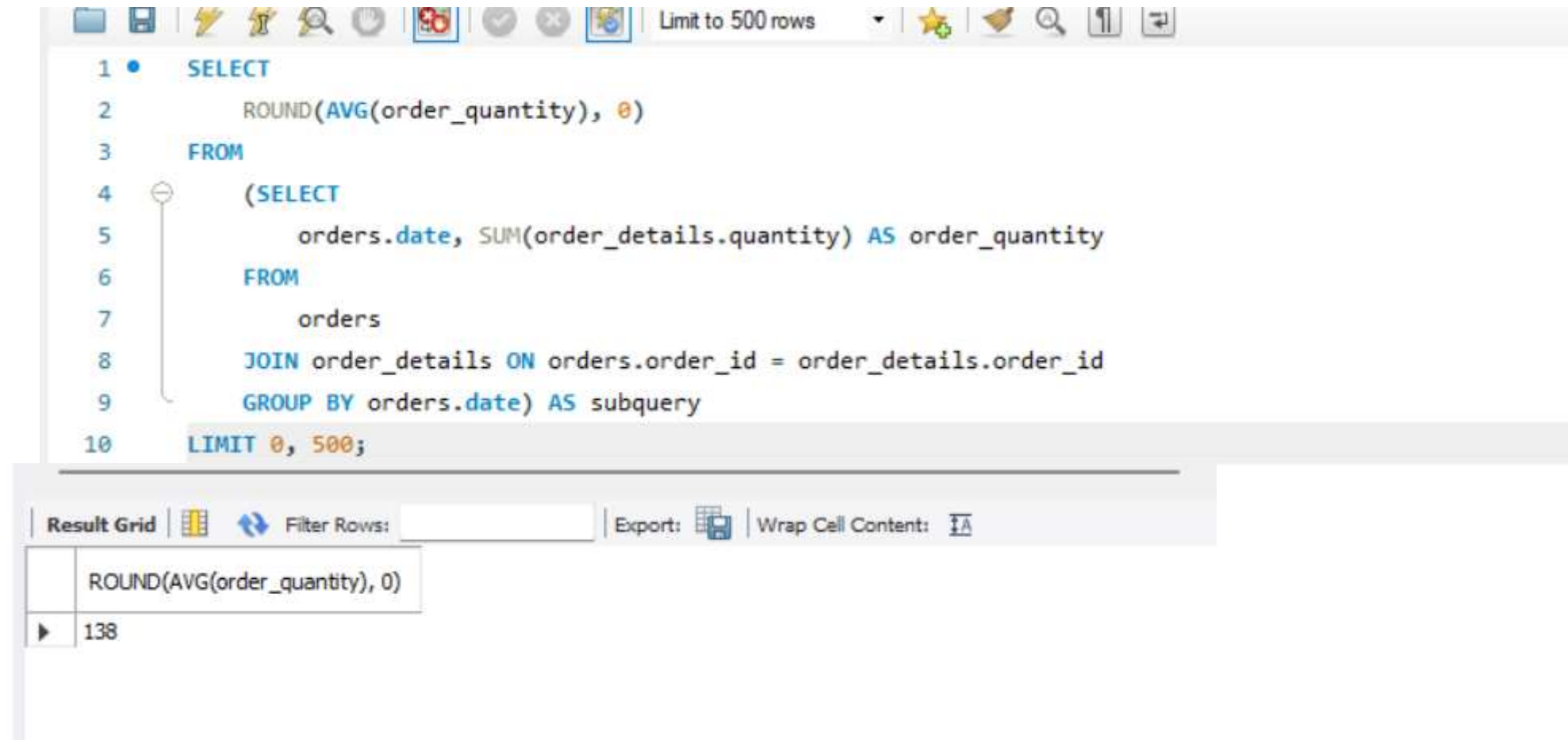
The screenshot shows a SQL IDE interface. On the left, the 'SCHEMAS' pane shows a database named 'pizzhut' with tables 'order_details', 'orders', 'pizza_types', and 'pizzas'. The main editor shows a SQL query in 'SQL File 11':

```
1 use pizzhut;  
2 SELECT  
3     category, COUNT(name)  
4 FROM  
5     pizza_types  
6 GROUP BY category;
```

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

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

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



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 500 rows' dropdown. The SQL editor contains the following query:

```
1 • SELECT
2     ROUND(AVG(order_quantity), 0)
3 FROM
4     (SELECT
5         orders.date, SUM(order_details.quantity) AS order_quantity
6     FROM
7         orders
8     JOIN order_details ON orders.order_id = order_details.order_id
9     GROUP BY orders.date) AS subquery
10 LIMIT 0, 500;
```

Below the editor is the 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The result grid displays the following data:

	ROUND(AVG(order_quantity), 0)
▶	138

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

SQL File 4* SQL File 5* SQL File 6* SQL File 7* SQL File 8* SQL File 9* SQL File 11* SQL File 12* SQL File 13* x



```
1 • SELECT pizza_types.name,  
2       SUM(order_details.quantity * pizzas.price) AS revenue  
3 FROM pizza_types  
4 JOIN pizzas  
5     ON pizzas.pizza_type_id = pizza_types.pizza_type_id -- Specify the table for pizza_type_id  
6 JOIN order_details  
7     ON order_details.pizza_id = pizzas.pizza_id  
8 GROUP BY pizza_types.name  
9 ORDER BY revenue DESC  
10 LIMIT 3;  
11
```

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		