

Where Every Slice is a Taste of Perfection

WELCOME TO

SQL PROJECT: PIZZA SALES DATA ANALYSIS

"Project
Overview"





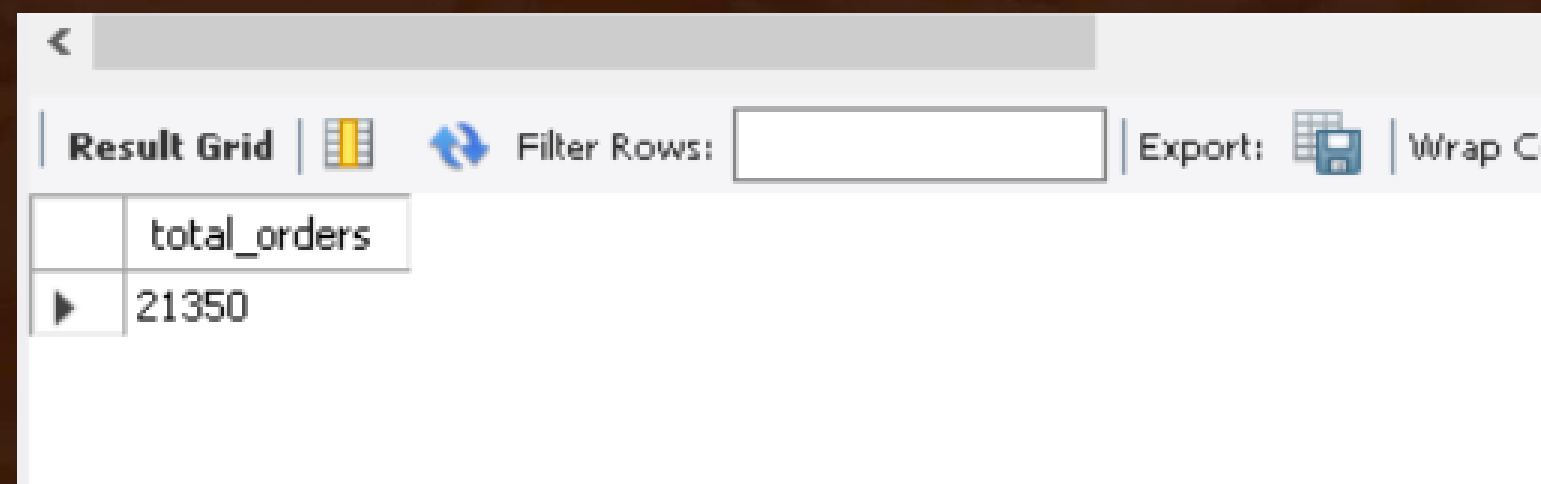
ABOUT OUR PIZZA SALES

Our Passion for Pizza Data 🍕

Hello everyone, my name is "[Mihir Aryan Mishra](#)". I've Created a SQL Analysis on Pizza Sales Data. This project uses SQL to analyze PizzaHut's sales dataset. We explore customer orders, popular pizzas, and revenue patterns using queries from basic SELECT statements to advanced SQL techniques. The goal is to gain insights into sales trends and demonstrate SQL proficiency.

Q1) RETREVE THE TOTAL NUMBER OF ORDERS PLACED.

```
select count(order_id) as total_orders from orders;
```



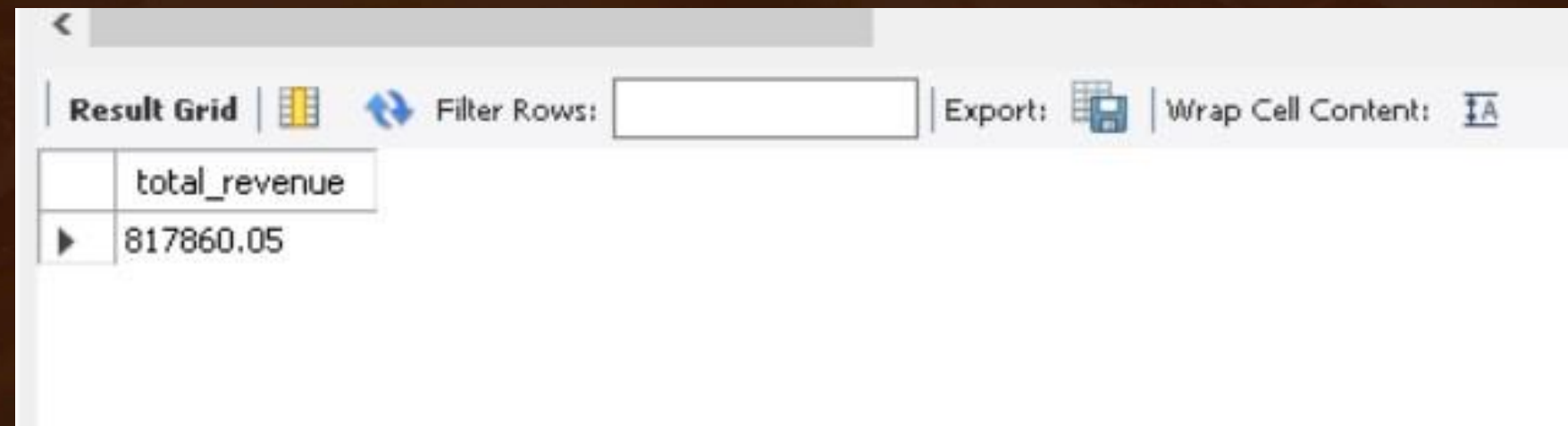
The screenshot shows a database query result interface. At the top, there is a toolbar with a back arrow, a 'Result Grid' button, a 'Filter Rows' button with a dropdown menu, an 'Export' button, and a 'Wrap C' button. Below the toolbar is a table with one column named 'total_orders' and one row containing the value '21350'.

	total_orders
▶	21350

**“Total Orders =
21,350”**

Q2) CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

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

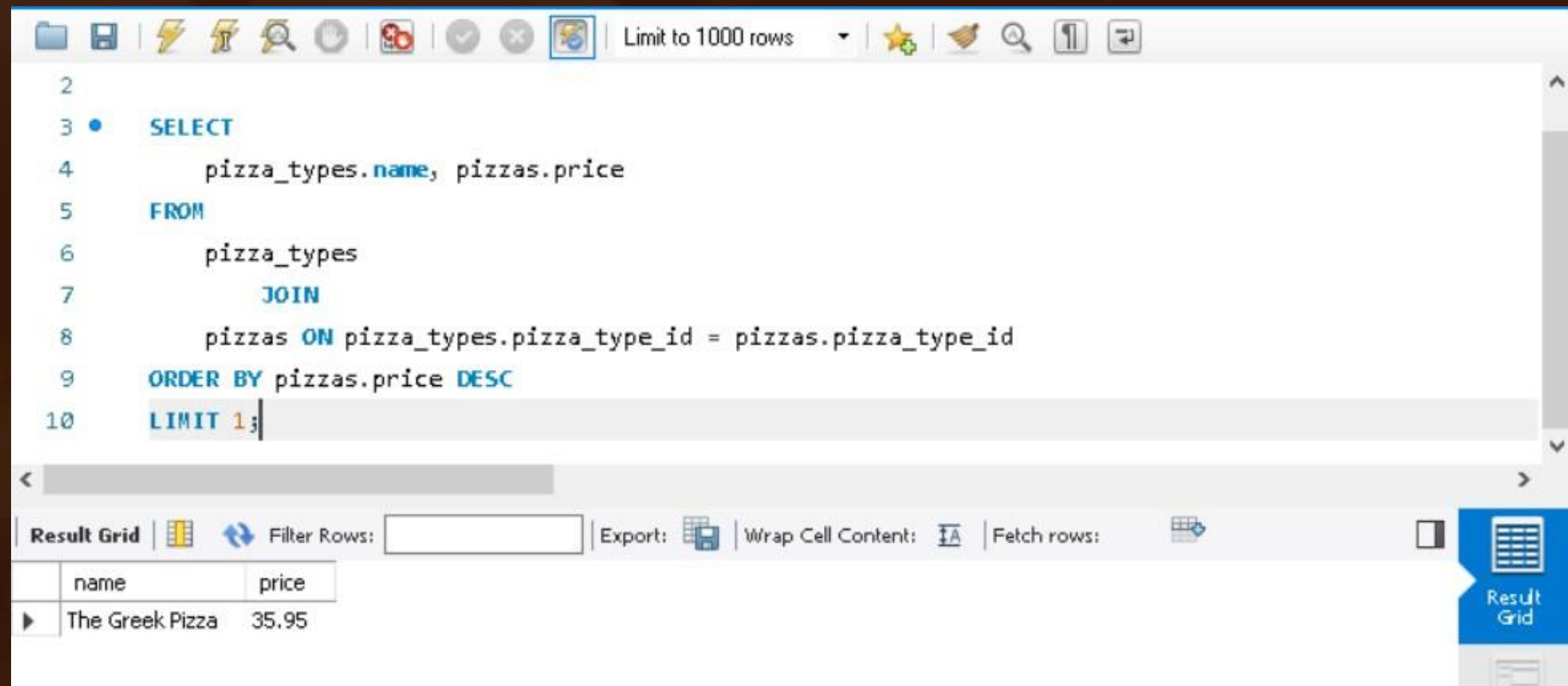


The screenshot shows a database query result interface. At the top, there is a toolbar with icons for 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below the toolbar, a table displays the results of the query. The table has one column named 'total_revenue' and one row with the value '817860.05'.

	total_revenue
▶	817860.05

 **Total Revenue Generated:**
\$817,860

Q3) IDENTIFY THE HIGHEST PRICED PIZZA.



The screenshot shows a SQL query editor with the following query:

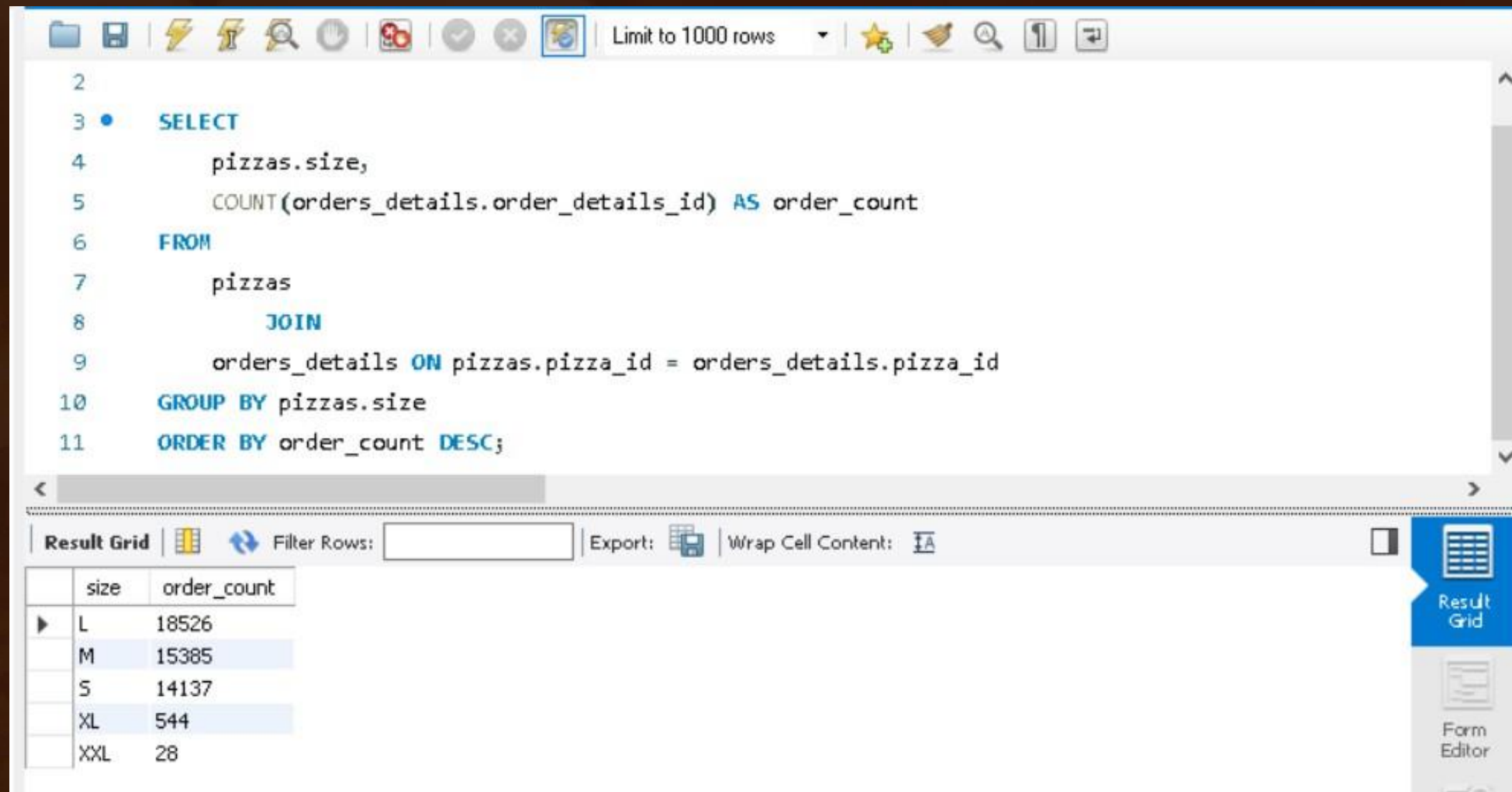
```
2  
3 • SELECT  
4     pizza_types.name, pizzas.price  
5 FROM  
6     pizza_types  
7     JOIN  
8     pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id  
9 ORDER BY pizzas.price DESC  
10 LIMIT 1;
```

Below the query editor, the 'Result Grid' is displayed with the following data:

	name	price
▶	The Greek Pizza	35.95

🍕 Highest-Priced Pizza: "The Greek Pizza"
— \$35.95

Q4) IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

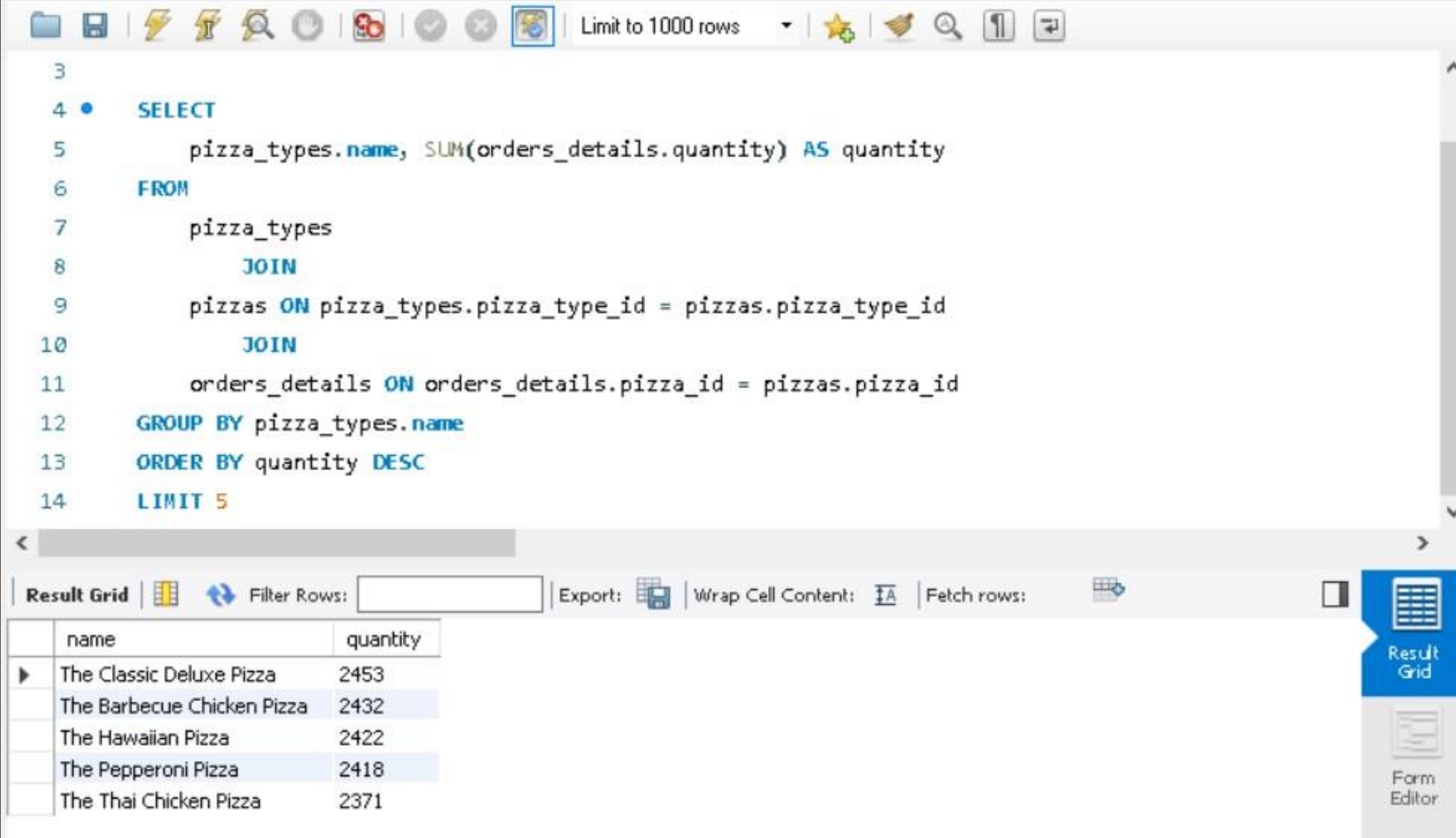
```
2  
3 • SELECT  
4     pizzas.size,  
5     COUNT(orders_details.order_details_id) AS order_count  
6 FROM  
7     pizzas  
8     JOIN  
9     orders_details ON pizzas.pizza_id = orders_details.pizza_id  
10 GROUP BY pizzas.size  
11 ORDER BY order_count DESC;
```

Below the query editor, there is a "Result Grid" section with a table showing the results of the query. The table has two columns: "size" and "order_count". The results are as follows:

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

The "Result Grid" section also includes a "Filter Rows" input field, an "Export" button, and a "Wrap Cell Content" checkbox. On the right side of the "Result Grid" section, there are two buttons: "Result Grid" and "Form Editor".

Q5) LIST THE TOP 5 MOST ORDERED PIZZA TYPES. 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
```

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

	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

Result Grid
Form Editor

Q6) JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.

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

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

	category	quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

Result Grid
Form Editor

Q7) DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

```
3 • SELECT
4     HOUR(order_time) AS hour, COUNT(order_id) AS order_count
5 FROM
6     orders
7 GROUP BY HOUR(order_time);
```

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

	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

Result 1 | Read Only

Result Grid
Form Editor
Field Types
Query Stats

Q8) JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

```
4 • SELECT
5     category, COUNT(name)
6 FROM
7     pizza_types
8 GROUP BY category;
```

< >

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

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

Result Grid

Form Editor

Field Types

⬆ ⬇ ⬆

Q9) GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.

```
4 • SELECT
5     ROUND(AVG(quantity), 0)
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;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:		Result Grid
	ROUND(AVG(quantity), 0)					
▶	138					Form Editor

Q10) DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE

```
2
3 • SELECT
4     pizza_types.name,
5     SUM(orders_details.quantity * pizzas.price) AS revenue
6 FROM
7     pizza_types
8     JOIN
9     pizzas ON pizzas.pizza_type_id = pizza_types.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 revenue DESC
```

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

Result
Grid

Form
Editor

Q11) CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

```
1  -- Calculate the percentage contribution of each
2  -- pizza type to total revenue.
3
4  • SELECT
5      pizza_types.category,
6      ROUND(SUM(orders_details.quantity * pizzas.price) / (SELECT
7          ROUND(SUM(orders_details.quantity * pizzas.price),
8              2) AS total_sales
9          FROM
10             orders_details
11             JOIN
12             pizzas ON pizzas.pizza_id = orders_details.pizza_id) * 100,
13          2) AS revenue
14  FROM
15      pizza_types
16      JOIN
17      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
18      JOIN
19      orders_details ON orders_details.pizza_id = pizzas.pizza_id
20  GROUP BY pizza_types.category
```

Result Grid |   Filter Rows:

	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

Q12) ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```
3 • select order_date,  
4      sum(revenue) over(order by order_date) as cum_revenue  
5      from  
6      (select orders.order_date,  
7         sum(orders_details.quantity * pizzas.price) as revenue  
8         from orders_details join pizzas  
9         on orders_details.pizza_id= pizzas.pizza_id  
10        join orders  
11        on orders.order_id= orders_details.order_id  
12        group by orders.order_date) as sales;
```

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

	order_date	cum_revenue
▶	2015-01-01	2713.85000000000004
	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	23990.3500000000002

Result 2 x

Q13) 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 rn
from
(select pizza_types.category, pizza_types.name,
sum((orders_details.quantity) * pizzas.price) 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, pizza_types.name) as a) as b
where rn <=3;
```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	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			

CONCLUSION



- This project demonstrated how SQL can transform raw sales data into meaningful insights.
- By analyzing PizzaHut's orders, I applied concepts from basic queries to advanced SQL techniques such as joins, aggregations, and ranking functions.
- Through this, I identified the most popular pizzas, revenue drivers, and ordering trends.
- Beyond insights, this project strengthened my problem-solving, data analysis, and database management skills, showcasing the power of SQL in making data-driven business decisions.

Pizza Sales Presentation

**THANKYOU
FOR ATTENTION**

SQL Pizza Sales Project

by

MIHIR A. MISHRA