

SQL FOR DATA ANALYSIS

PORTFOLIO PROJECT

PIZZA STORE





HELLO!!

My name is Sathwik Shetty, in
this project I have used SQL
Queries to solve few problem
statements given by the pizza
store



BASIC QUERIES

1. Retrieve the total number of orders placed.

```
6
7 • select count(order_id) from orders
8
9
10
11
```

< [REDACTED]

Result Grid | Filter Rows: [REDACTED] Export: [REDACTED] Wrap Cell Content: [REDACTED]

count(order_id)
21350

2.Calculate the total revenue generated from pizza sales.

The screenshot shows a MySQL Workbench interface with a query editor window. The query is:

```
6 • SELECT
7   ROUND(SUM(order_details.quantity * pizzas.price),
8         2) AS Revenue
9 FROM
10    order_details
11   JOIN
12      pizzas ON order_details.pizza_id = pizzas.pizza_id
```

The result grid shows one row with the column 'Revenue' containing the value '817860.05'.

Revenue
817860.05

3.Identify the highest-priced pizza.

The screenshot shows a MySQL Workbench interface with a query editor and a results grid.

Query Editor:

```
4 • select * from pizzas;
5 • SELECT
6     pizza_types.name, pizzas.price
7 FROM
8     pizza_types
9     JOIN
10    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
11   ORDER BY pizzas.price DESC
12 LIMIT 1;
```

Results Grid:

	name	price
▶	The Greek Pizza	35.95

Identify the most common pizza size ordered.

```
6 • SELECT
7     pizzas.size,
8     COUNT(order_details.order_details_id) AS o_count
9 FROM
10    pizzas
11      JOIN
12          order_details ON pizzas.pizza_id = order_details.pizza_id
13 GROUP BY pizzas.size
14 ORDER BY o_count DESC
15 LIMIT 1
```

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

	size	o_count
▶	L	18526

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

The screenshot shows a MySQL Workbench interface with a query editor and a results grid. The query editor contains the following SQL code:

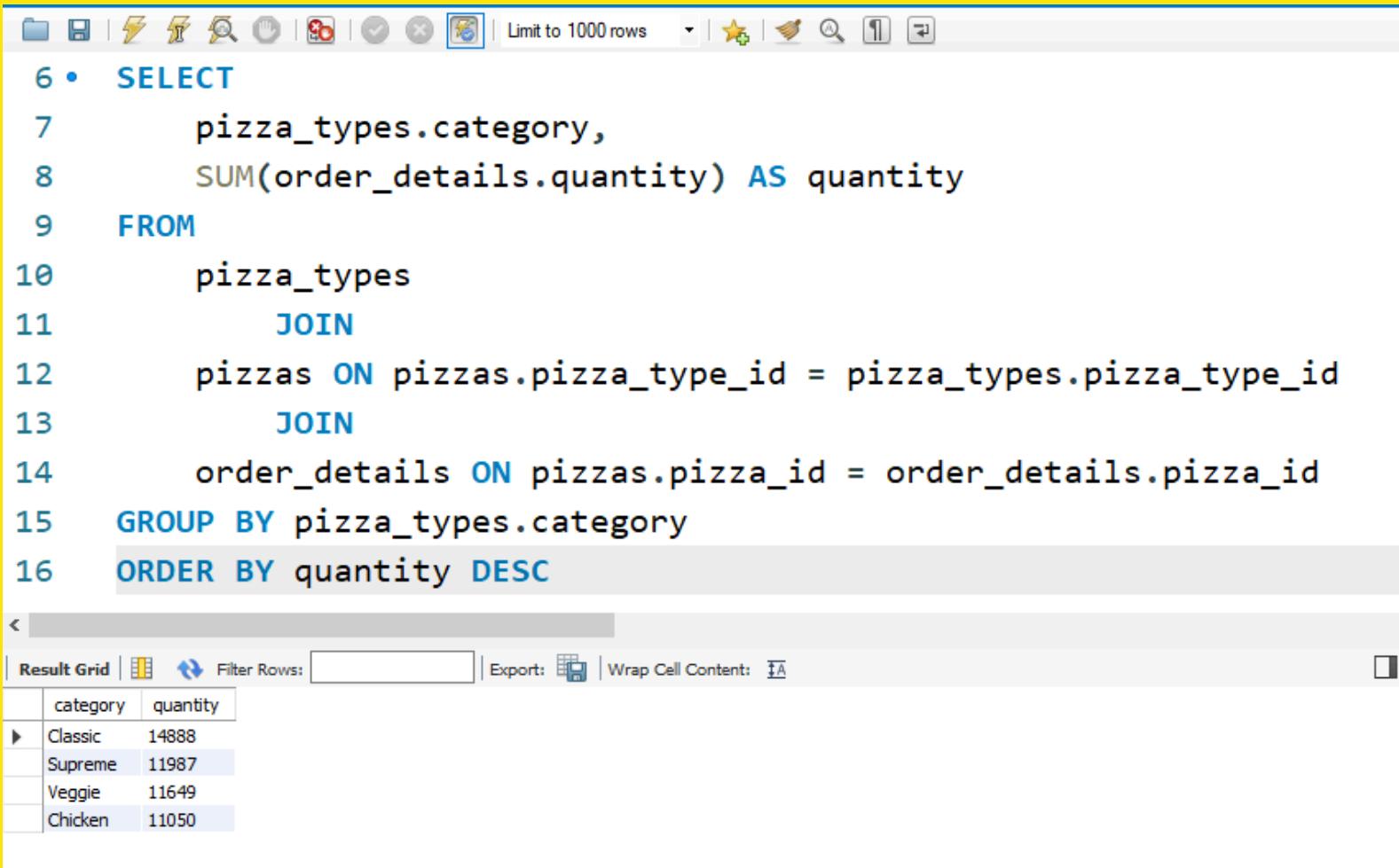
```
Query 1 SQL File 2* x
6 •  SELECT
7      pizza_types.name, SUM(order_details.quantity) AS Quantity
8  FROM
9      pizza_types
10     JOIN
11      pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
12     JOIN
13      order_details ON order_details.pizza_id = pizzas.pizza_id
14 GROUP BY pizza_types.name
15 ORDER BY Quantity DESC
16 LIMIT 5;
```

The results grid displays the following data:

	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

Intermediate:

- 1. Join the necessary tables to find the total quantity of each pizza category ordered.



The screenshot shows a MySQL Workbench interface with a query editor and a result grid.

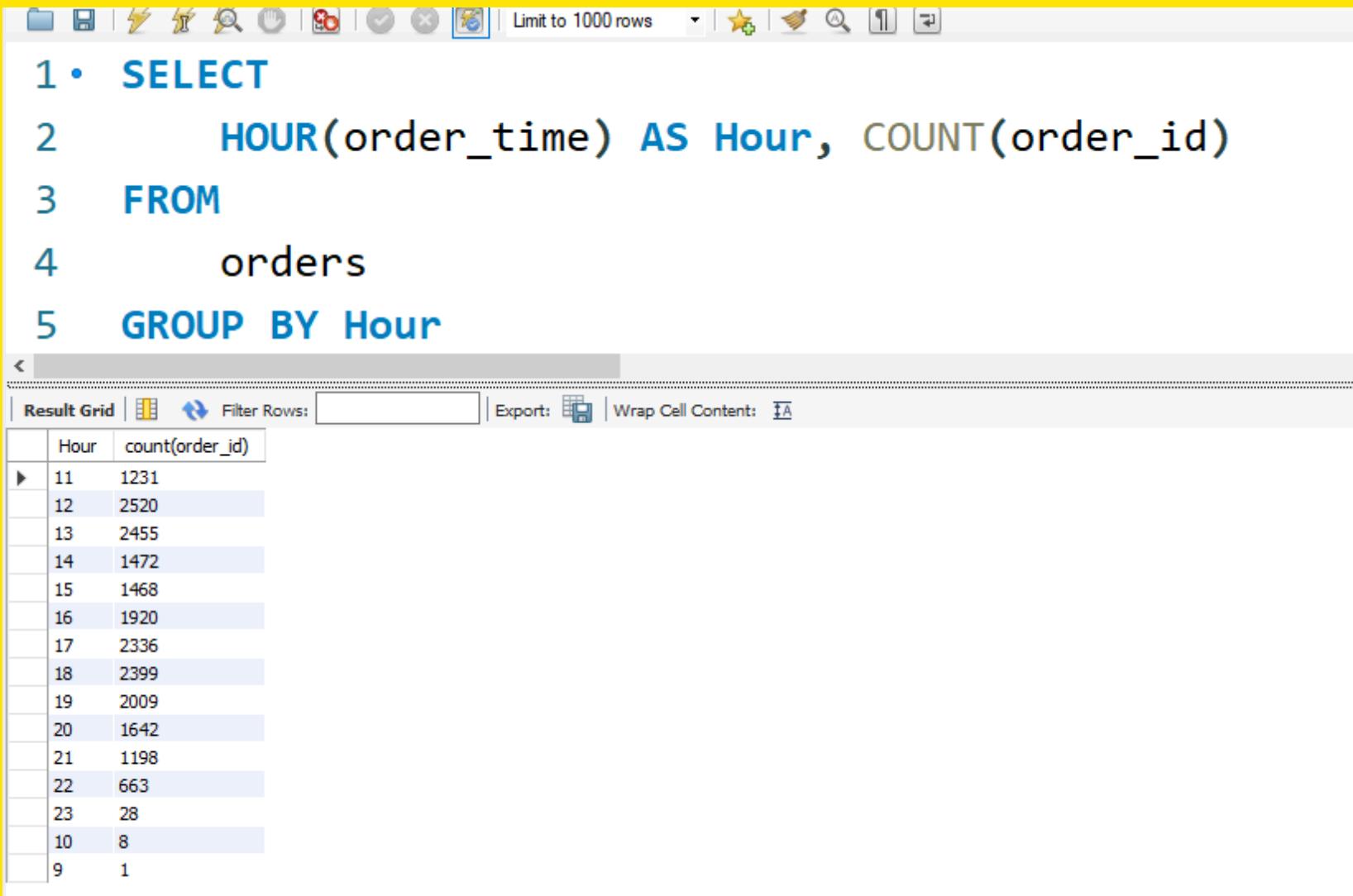
Query Editor:

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

Result Grid:

	category	quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

2.Determine the distribution of orders by hour of the day.



The screenshot shows a MySQL Workbench interface with a query editor and a results grid. The query is:

```
1 • SELECT  
2     HOUR(order_time) AS Hour, COUNT(order_id)  
3 FROM  
4     orders  
5 GROUP BY Hour
```

The results grid displays the count of orders for each hour of the day:

Hour	count(order_id)
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

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

The screenshot shows a MySQL Workbench interface. The top part displays a SQL query:

```
5 • SELECT
6     pizza_types.category,
7     COUNT(order_details.order_id) AS Contribution
8 FROM
9     pizza_types
10    JOIN
11         pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
12    JOIN
13        order_details ON pizzas.pizza_id = order_details.pizza_id
14 GROUP BY pizza_types.category
```

The bottom part shows the results in a grid:

category	Contribution
Classic	14579
Veggie	11449
Supreme	11777
Chicken	10815

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

The screenshot shows a MySQL Workbench interface with the following details:

- Query Editor:** The main window displays a SQL query:

```
5 •   SELECT
6       ROUND(AVG(qu), 0) AS Average_Value
7   FROM
8     (SELECT
9         orders.order_date, SUM(order_details.quantity) AS qu
10    FROM
11      orders
12    JOIN order_details ON orders.order_id = order_details.order_id
13    GROUP BY orders.order_date) AS sum;
```

- Toolbar:** Standard MySQL Workbench toolbar with various icons for file operations, search, and connection management.
- Status Bar:** Shows "Limit to 1000 rows".
- Result Grid:** Below the editor, the result grid shows one row of data:

Average_Value
138
- Buttons:** Buttons for "Result Grid", "Filter Rows:", "Export:", and "Wrap Cell Content:".

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

The screenshot shows a MySQL Workbench interface with a query editor and a result grid.

Query Editor:

```
7
8 •  SELECT
9      pizza_types.name,
10     SUM(order_details.quantity * pizzas.price) AS REVENUE
11   FROM
12     pizza_types
13       JOIN
14     pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
15       JOIN
16     order_details ON order_details.pizza_id = pizzas.pizza_id
17   GROUP BY pizza_types.name
18   ORDER BY revenue DESC
19   LIMIT 3
```

Result Grid:

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

Advanced:

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

The screenshot shows a MySQL query editor interface with the following details:

- Toolbar:** Includes icons for file operations (New, Save, Import, Export), search, and various database management functions.
- Query Editor:** Displays the SQL code:

```
6 •  SELECT
7      pizza_types.category,
8      ROUND(SUM(order_details.quantity * pizzas.price) / (SELECT
9          ROUND(SUM(order_details.quantity * pizzas.price),
10         2) AS total_sales
11     FROM
12       order_details
13     JOIN
14       pizzas ON pizzas.pizza_id = order_details.pizza_id) * 100,
15         2) AS PERCENTAGE_REVENUE
16   FROM
17     pizza_types
18     JOIN
19       pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
20     JOIN
21       order_details ON order_details.pizza_id = pizzas.pizza_id
22   GROUP BY pizza_types.category
23   ORDER BY PERCENTAGE_REVENUE DESC
```
- Result Grid:** Shows the output of the query in a tabular format:

category	PERCENTAGE_REVENUE
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68
- Bottom Bar:** Includes buttons for "Result Grid" (selected), "Filter Rows:", "Export", and "Wrap Cell Content".

Analyze the cumulative revenue generated over time.

The screenshot shows a database query interface with the following details:

Query Editor:

```
1 • select order_date,sum(revenue) over(order by order_date) As Cumulative_Revenue
2   from
3     (select orders.order_date,round(sum(order_details.quantity*pizzas.price),2)as revenue
4       from order_details join pizzas
5         on order_details.pizza_id=pizzas.pizza_id
6       join orders on orders.order_id=order_details.order_id
7     group by orders.order_date)as sales
```

Result Grid:

	order_date	Cumulative_Revenue
▶	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
	2015-01-09	21526.399999999998
	2015-01-10	23990.35
	2015-01-11	25862.649999999998

Status Bar: Result 3 ×

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

```
1 •  select name,revenue from
2   (select category,name,revenue,
3    rank() over( partition by category order by revenue desc) as RN
4   FROM
5   (select pizza_types.category,pizza_types.name,sum(order_details.quantity*pizzas.price)
6    AS revenue
7    from pizza_types join pizzas
8    on pizza_types.pizza_type_id=pizzas.pizza_type_id
9    join order_details on order_details.pizza_id=pizzas.pizza_id
10   group by pizza_types.category,pizza_types.name) as a) as b
11  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	32265.70000000065
	The Mexicana Pizza	26780.75

Thank
you!