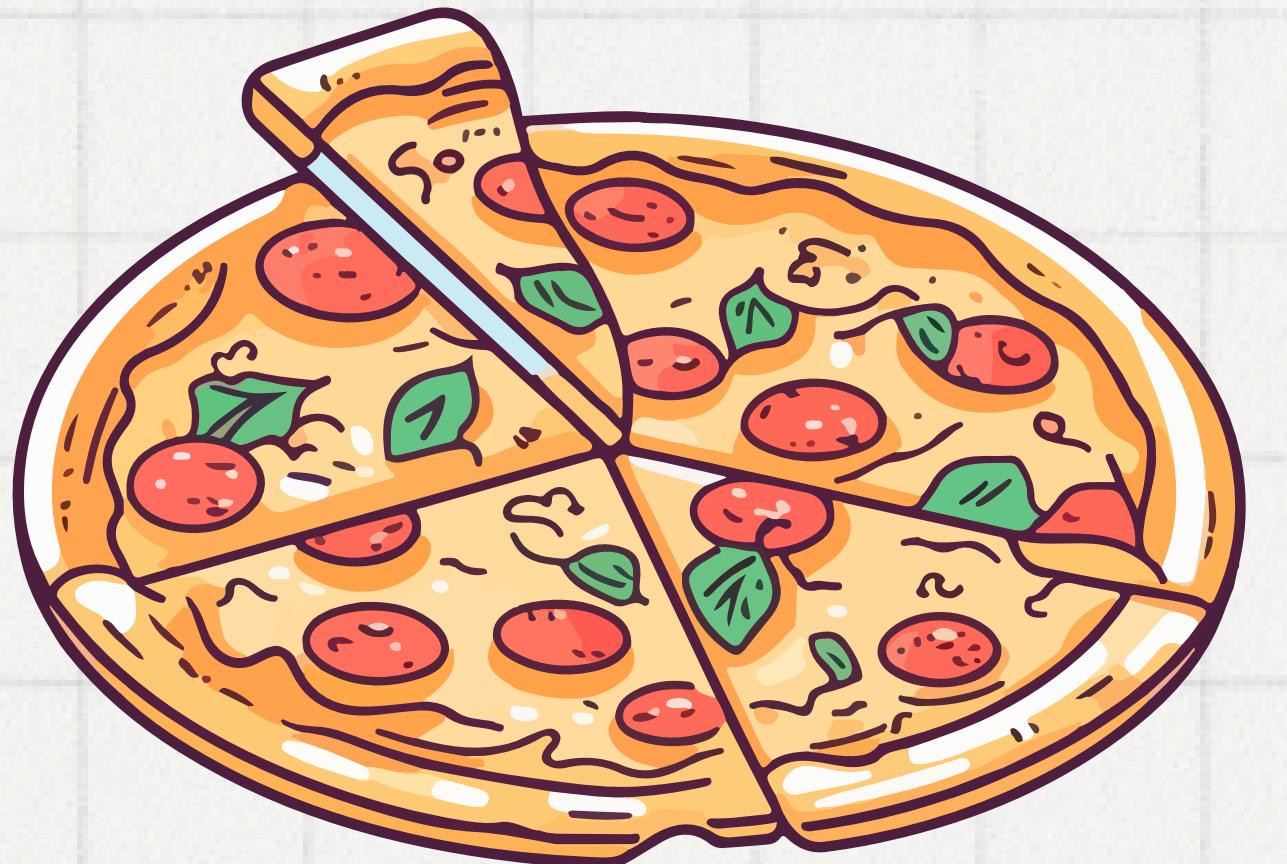


SQL PROJECT

Presented by PRAGYA SINGH

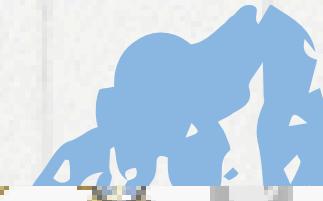
PIZZA SALES



This Pizza Sales Project uses SQL to analyze sales data for a pizza restaurant. It tracks customer orders, inventory, and sales trends in a database. The project focuses on finding out which menu items are popular, managing stock levels, and improving marketing efforts. By looking at the data, the restaurant can make better decisions to increase profits .



Retrieve the total number of orders placed.



```
1      -- retrieve the total numbers of orders placed.  
2      -- retreive the total number of orders placed.  
3  
4 • select count(idorders) as total_orders from orders;
```

5

6

7

8

result Grid |



Filter Rows:

Export:



Wrap Cell Content:

total_orders
21350

Calculate the total revenue generated from pizza sales.



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

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

Total_sales
817860.05

Identify the highest-priced pizza.

```
1 -- identify the highest-priced pizza.  
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;  
11
```

Result Grid |



Filter Rows:

Export:



Wrap Cell

	name	price
▶	The Greek Pizza	35.95

Identify the most common pizza size ordered.

```
-- identify the most common pizza size ordered.  
• SELECT  
    pizzas.size,  
    COUNT(order_details.idorder_details) AS order_count  
FROM  
    pizzas  
        JOIN  
    order_details ON pizzas.pizza_id = order_details.pizza_id  
GROUP BY pizzas.size  
ORDER BY order_count DESC;
```

| Result Grid | Filter Rows: | Export: |

	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.

```
1 -- list the tp 5 most ordered pizza .
2 -- types along with their quantities.
3
4 • select pizza_types.name,
5     SUM(order_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        order_details ON order_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

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

```
-- join the necessary tables to find the
-- total quantity of each pizza category ordered.

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

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

Determine the distribution of orders by hour of the day.

-- determine the distribution of orders by hour of the day.

- **SELECT**

```
HOUR(orderscol1) AS hour, COUNT(idorders) AS order_count  
FROM  
    orders  
GROUP BY HOUR(orderscol1);
```

Grid



Filter Rows:

Export:



Wrap Cell Content:



hour	order_count
1231	
2520	
2455	
1472	
1468	
1920	
2336	
2399	
2009	
1642	
1198	
663	
28	
8	

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

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

Result Grid



Filter Rows:

Export:



Wrap Cell Content:



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

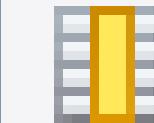
Group the orders by date and calculate the average number of pizzas ordered per day.

```
-- group the orders by date and calculate the average  
-- number of pizzas ordered per day.
```

- **SELECT**

```
    ROUND(AVG(quantity), 0)  
FROM  
(SELECT  
    orders.orderscol, SUM(order_details.quantity) AS quantity  
FROM  
    orders  
JOIN order_details ON orders.idorders = order_details.order_id  
GROUP BY orders.orderscol) AS order_quantity;
```

Result Grid



Filter Rows:

ROUND(AVG(quantity), 0)
138

138

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

```
-- determine the top 3 most ordered pizza types based on revenue.
```

- ```
SELECT pizza_types.name,
sum(order_details.quantity *pizzas.price) as revenue
FROM
pizza_types
JOIN
pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
JOIN
order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3;
```

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

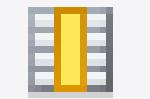
- Result Grid:** The results of the SQL query are presented in a grid format.
- Columns:** The grid has two columns: "name" and "revenue".
- Data:** The results are as follows:

|   | name                         | revenue  |
|---|------------------------------|----------|
| ▶ | The Thai Chicken Pizza       | 43434.25 |
|   | The Barbecue Chicken Pizza   | 42768    |
|   | The California Chicken Pizza | 41409.5  |
- UI Elements:** The interface includes a toolbar with icons for Result Grid, Filter Rows, and Export, along with a search bar and other standard database navigation buttons.

## 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 (SUM(order_details.quantity * pizzas.price) / (SELECT
7 ROUND(SUM(order_details.quantity * pizzas.price),
8 2) AS total_sales
9
10 FROM
11 order_details
12 JOIN
13 pizzas ON pizzas.pizza_id = order_details.pizza_id)) * 100 AS revenue
14
15 FROM
16 pizza_types
17 JOIN
18 pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
19 JOIN
20 order_details ON order_details.pizza_id = pizzas.pizza_id
21 GROUP BY pizza_types.category
22 ORDER BY revenue DESC;
```

Result Grid



Filter Rows:

E

|   | category | revenue            |
|---|----------|--------------------|
| ▶ | Classic  | 26.90596025566967  |
|   | Supreme  | 25.45631126009862  |
|   | Chicken  | 23.955137556847287 |
|   | Veggie   | 23.682590927384577 |

## Analyze the cumulative revenue generated over time.

```
-- analyse the cumulative revenue generated over time.
```

```
+ select orderscol ,sum(revenue) over(order by orderscol) as cum_revenue
from
 (select orders.orderscol,
 sum(order_details.quantity * pizzas.price) as revenue
 from order_details join pizzas
 on order_details.pizza_id = pizzas.pizza_id
 join orders
 on orders.idorders = order_details.order_id
 group by orders.orderscol) as sales;
```

Result Grid



Filter Rows:

|     | orderscol  | cum_revenue        |
|-----|------------|--------------------|
| 1   | 2015-01-01 | 2713.8500000000004 |
| 2   | 2015-01-02 | 5445.75            |
| 3   | 2015-01-03 | 8108.15            |
| 4   | 2015-01-04 | 9863.6             |
| 5   | 2015-01-05 | 11929.55           |
| 6   | 2015-01-06 | 14358.5            |
| 7   | 2015-01-07 | 16560.7            |
| 8   | 2015-01-08 | 19399.05           |
| 9   | 2015-01-09 | 21526.4            |
| 10  | 2015-01-10 | 23990.35000000002  |
| 11  | 2015-01-11 | 25862.65           |
| 12  | 2015-01-12 | 27781.7            |
| 13  | 2015-01-13 | 29831.30000000003  |
| 14  | 2015-01-14 | 32358.70000000004  |
| 15  | 2015-01-15 | 34343.50000000001  |
| 16  | 2015-01-16 | 36937.65000000001  |
| 17  | 2015-01-17 | 39001.75000000001  |
| ... | ...        | -----              |

Result 1

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

```
1 -- determine the top 3 most ordered pizza types
2 -- based on revenue for each pizza category.
3 • select name, revenue from
4 (select category ,name,revenue,
5 rank() over (partition by category order by revenue desc)as rn
6 from
7
8 (select pizza_types.category,pizza_types.name,
9 sum((order_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 order_details
13 on order_details.pizza_id = pizzas.pizza_id
14 group by pizza_types.category, pizza_types.name) as a) as b
15 where rn <= 3;
```

Result Grid | Filter Rows:  |

|   | 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          |
|   | The Five Cheese Pizza        | 26066.5           |

**Thank you  
very much!**