

SQL PROJECT : ANALYSIS ON PIZZA SALES

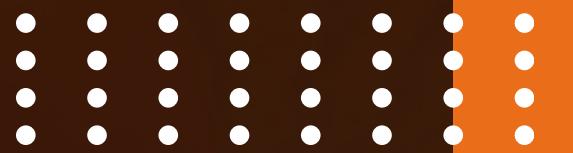


HELLO !



Hi, I am Urvashi Shasne. I utilized SQL queries to analyze and solve problems related to pizza sales in this project.





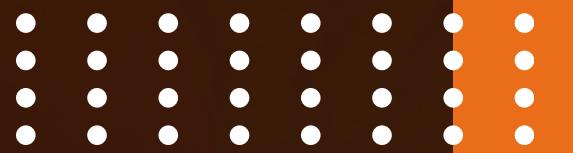
Retrieve the total number of orders placed.

- `SELECT
 COUNT(order_id) AS number_orders
FROM
 orders;`

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

- Result Grid:** The results are displayed in a grid format.
- Columns:** There is one column named "number_orders".
- Rows:** One row is present, containing the value "204".
- Toolbar:** The toolbar includes icons for "Result Grid" (selected), "Filter Rows:", and "Export".

	number_orders
▶	204



Calculate Total revenue generated from pizza sales and round it to 2 decimal places

```
SELECT  
    ROUND(SUM(order_details.quantity * pizzas.price),  
        2) AS total_sales  
FROM  
    order_details  
    JOIN  
    pizzas ON order_details.pizza_id = pizzas.pizza_id;
```

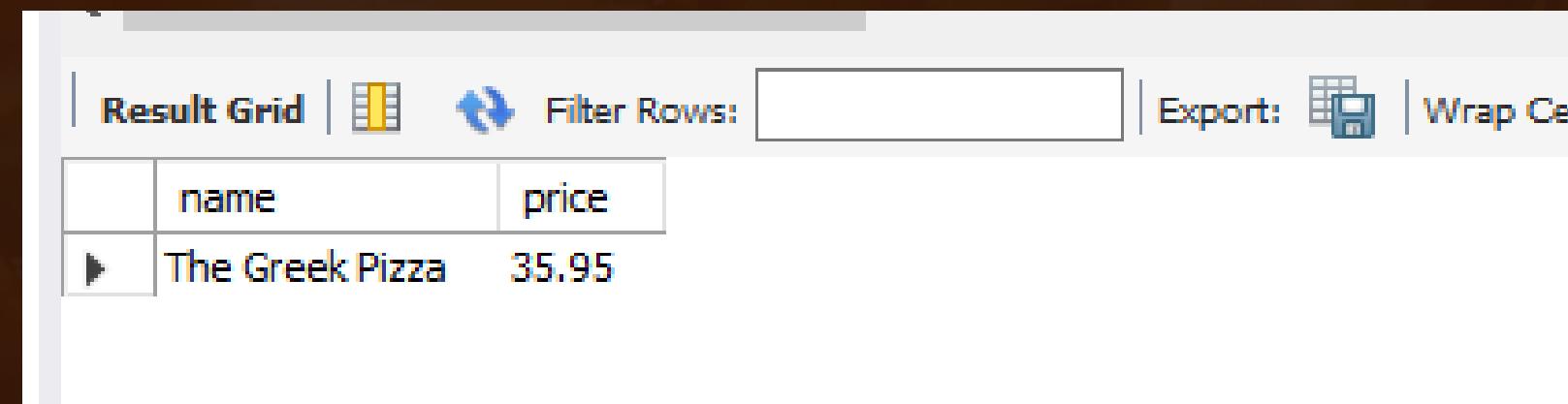
The screenshot shows the MySQL Workbench interface with the following details:

- Result Grid:** A table displaying the results of the executed SQL query.
- Columns:** The table has one column named "total_sales".
- Data:** There is one row containing the value "379190.9".
- Toolbar:** At the top, there are buttons for "Result Grid" (highlighted in blue), "Filter Rows:", and "Export:".

total_sales
379190.9

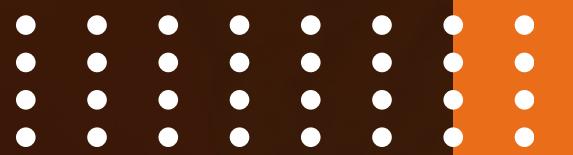
Identify the highest priced pizza.

```
SELECT
    pizza_types.name, pizzas.price
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY price DESC
LIMIT 1;
```



The screenshot shows a MySQL Workbench interface with a result grid. The grid has two columns: 'name' and 'price'. A single row is displayed, representing the highest-priced pizza found by the query. The row contains the name 'The Greek Pizza' and the price '35.95'.

	name	price
▶	The Greek Pizza	35.95



Identify the most commonly ordered pizza size

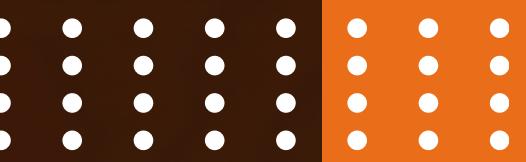
```
SELECT
    pizzas.size,
    COUNT(order_details.order_details_id) AS total_order
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY total_order DESC;
```

	size	total_order
▶	L	8623
	M	7113
	S	6500
	XL	269
	XXL	15

pizzas which is ordered most along with their quantity

```
• SELECT
    pizza_types.name, 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.name
  ORDER BY quantity
  LIMIT 5;
```

Result Grid		
	name	quantity
▶	The Brie Carre Pizza	221
	The Mediterranean Pizza	420
	The Calabrese Pizza	428
	The Spinach Pesto Pizza	441
	The Chicken Pesto Pizza	442



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

- **SELECT**

```
    pizza_types.category,  
    SUM(order_details.quantity) AS total_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;
```

Result Grid | Filter Rows:

	category	total_quantity
▶	Classic	6838
	Veggie	5483
	Supreme	5568
	Chicken	5070

Determine the distribution of orders by hour of the day.

- **SELECT**

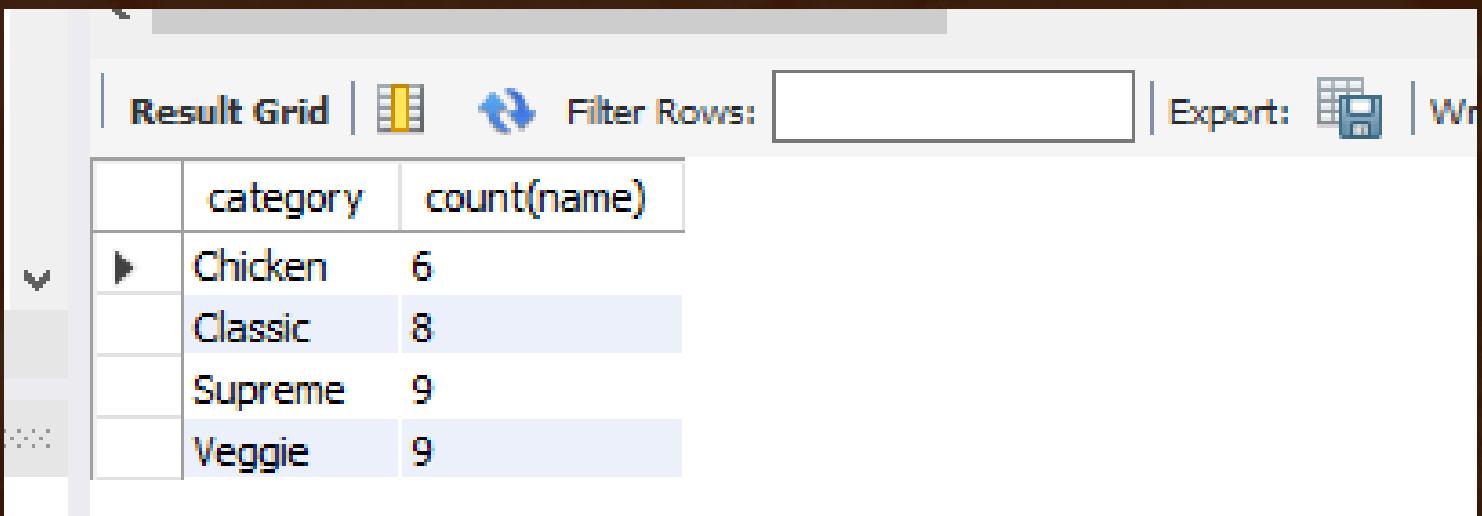
```
HOUR(order_time) AS hour, COUNT(order_id) AS no_orders  
FROM  
orders  
GROUP BY HOUR(order_time);
```

Result Grid | Filter Rows:

	hour	no_orders
▶	11	11
	12	18
	13	18
	14	15
	15	20
	16	15
	17	24
	18	27
	19	17
	20	16
	21	12
	22	11

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

- `select * from pizza_types;`
- `select category , count(name) from pizza_types
group by category;`

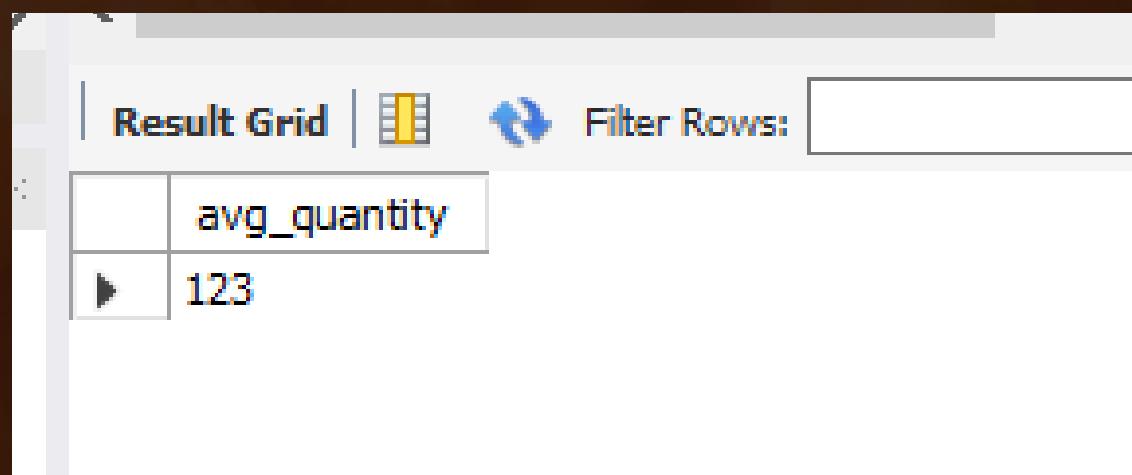


The screenshot shows a MySQL Workbench interface with a result grid. The grid has three columns: 'category' (with values 'Chicken', 'Classic', 'Supreme', and 'Veggie'), 'count(name)' (with values 6, 8, 9, and 9 respectively), and an unnamed column header. The 'category' and 'count(name)' columns are bolded. The 'Classic' row is highlighted with a light blue background.

	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.

```
32 •   SELECT
33     ROUND(AVG(quantity), 0) AS avg_quantity
34   FROM
35     (SELECT
36       SUM(order_details.quantity) AS quantity, orders.order_date
37     FROM
38       orders
39     JOIN order_details ON orders.order_id = order_details.order_id
40     GROUP BY orders.order_date) AS A;
```



The screenshot shows the MySQL Workbench interface with the results of the executed SQL query. The results are displayed in a grid with one row and two columns. The first column is empty, and the second column is labeled 'avg_quantity' with the value '123'.

	avg_quantity
▶	123

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

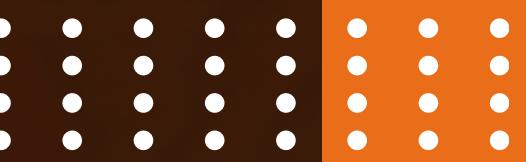
```
SELECT  
    pizza_types.name,  
    SUM(order_details.quantity * pizzas.price) AS revenue  
FROM  
    pizza_types  
        JOIN  
    pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id  
        JOIN  
    order_details ON order_details.pizza_id = pizzas.pizza_id  
GROUP BY pizza_types.name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

	name	revenue
▶	The Barbecue Chicken Pizza	20693.25
▶	The Thai Chicken Pizza	19242.5
▶	The California Chicken Pizza	18932.5

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

```
# total_revenue = sum(revenue)
• SELECT
    pizza_types.category,
    (SUM(order_details.quantity * pizzas.price) / (SELECT
        SUM(order_details.quantity * pizzas.price)
    FROM
        order_details
        JOIN
            pizzas ON order_details.pizza_id = pizzas.pizza_id) * 100) AS total_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.category
ORDER BY total_revenue DESC;
```

	category	total_revenue
▶	Classic	26.688799757588743
	Supreme	25.53467132254451
	Veggie	24.04112282230342
	Chicken	23.735406097561615



Analyze the cumulative revenue generated over time.

```
select order_date,sum(revenue) over (order by order_date) as cum_revenue
from
  (select orders.order_date, 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.order_id= order_details.order_id
   group by orders.order_date) as sales;
```

Result Grid | Filter Rows: | Export

	order_date	cum_revenue
▶	2015-01-01	2713.8500000000004
	2015-01-02	5445.75
	2015-01-03	8108.15
	2015-01-04	8195.9



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((order_details.quantity)*pizzas.price) as revenue
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, pizza_types.name) as a)as b
where rn<=3;
```

	name	revenue
▶	The Barbecue Chicken Pizza	20693.25
	The Thai Chicken Pizza	19242.5
	The California Chicken Pizza	18932.5
	The Classic Deluxe Pizza	17038
	The Hawaiian Pizza	14793.25
	The Pepperoni Pizza	14011.5
	The Spicy Italian Pizza	16121.5
	The Italian Supreme Pizza	15907.75
	The Sicilian Pizza	14155
	The Four Cheese Pizza	15481.500000000151
	The Five Cheese Pizza	12395
	The Mexicana Pizza	12196.5