# Pizza Sales analysis using MySQL Queries

# Queries Used to Analyze

1. Total Revenue
2. Highest Priced Pizza
3. Most common pizza size ordered
4. Top 5 ordered pizza along with their quantities
5. Total quantity of each pizza category
6. Distribution by hour of day
7. Category wise distribution
8. Top 3 most ordered pizza
9. % contribution of each pizza type
10. The average number of pizzas ordered per day
11. Cumulative income generated
12. Top 3 ordered pizza type on the basis of revenue for each category

# Calculate the total revenue generated from pizza sales

**Input query:**

```sql
-- Calculate the total revenue generated from pizza sales.

select
round(sum(order_details.quantity * pizzas.price),2) as total_revenue
from order_details join pizzas
on pizzas.pizza_id = order_details.pizza_id;
```

**Output:**

Result Grid

| total_revenue |
| --- |
| 817860.05 |

# Identify the highest-priced pizza.

Input query:

```
1   -- Identify the highest-priced pizza.
2
3   select pizza_types.name, pizzas.price
4   from pizza_types join pizzas
5   on pizza_types.pizza_type_id=pizzas.pizza_type_id
6   order by pizzas.price desc limit 1;
```

Execute the selected portion of the script or everything, if there is no selection

Output:

Result Grid | Filter R

| | name | price |
|---|---|---|
| ▶ | The Greek Pizza | 35.95 |

# Identify the most common pizza size ordered.

Input query :

```
3 •   select pizzas.size, count(order_details.order_detailsID) as order_count
4     from pizzas join order_details
5     on pizzas.pizza_id = order_details.pizza_id
6     group by pizzas.size order by order_count desc;
```

Output:

Result Grid | Filter

| 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.

Input query :

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

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 desc limit 5;
```

Output:

Result Grid | Filter Rows:

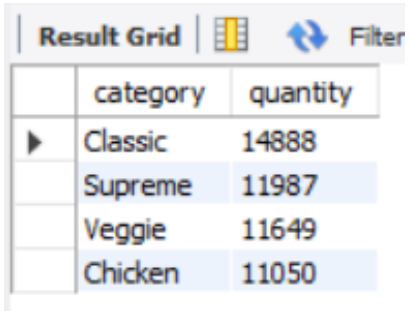
| name | quantity |
|---|---|
| The Classic Deluxe Pizza | 2453 |
| The Barbecue Chicken Pizza | 2432 |
| The Hawaiian Pizza | 2422 |
| The Pepperoni Pizza | 2418 |
| The Thai Chicken Pizza | 2371 |

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

Input query :

```
1    -- Join the necessary tables to find the total quantity of each pizza category.
2
3    select pizza_types.category,
4    sum(order_details.quantity) as quantity
5    from pizza_types join pizzas
6    on pizza_types.pizza_type_id=pizzas.pizza_type_id
7    join order_details
8    on order_details.pizza_id=pizzas.pizza_id
9    group by category order by quantity desc;
```

Output:



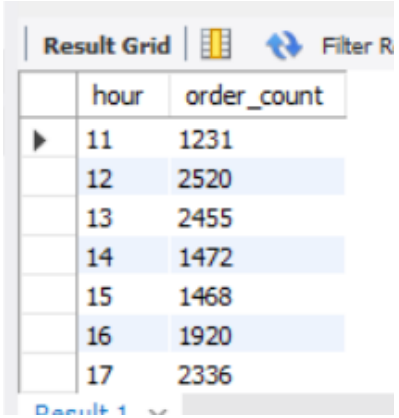

| category | quantity |
|---|---|
| Classic | 14888 |
| Supreme | 11987 |
| Veggie | 11649 |
| Chicken | 11050 |

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

Input query :

```
1    -- Determine the distribution of orders by hour of the day.
2
3    select hour (order_time) as hour, count(order_id) as order_count from orders
4    group by hour (order_time);
```

Output:



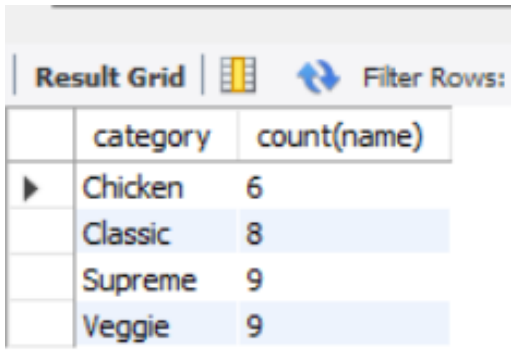

| hour | order_count |
|---|---|
| 11 | 1231 |
| 12 | 2520 |
| 13 | 2455 |
| 14 | 1472 |
| 15 | 1468 |
| 16 | 1920 |
| 17 | 2336 |

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

Input query :

```
1    -- Join relevant tables to find the category-wise distribution of pizzas.
2
3 •  select category, count(name) from pizza_types
4    group by category;
```

Output:




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

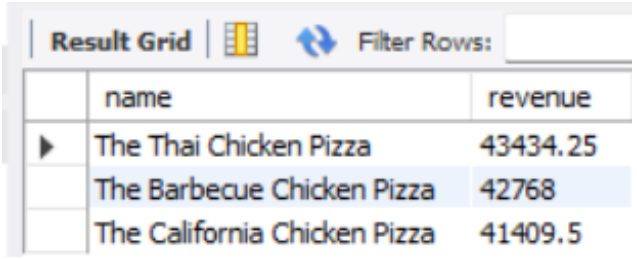
# Determine the top 3 most ordered pizza types based on revenue.

Input query:

```sql
-- 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;
```

Output:




| name | revenue |
|---|---|
| The Thai Chicken Pizza | 43434.25 |
| The Barbecue Chicken Pizza | 42768 |
| The California Chicken Pizza | 41409.5 |

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

Input query :

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

select pizza_types.category, round(sum(order_details.quantity*pizzas.price) / (select round(sum(order_details.quantity*pizzas.price),2) as total_sales
from order_details join pizzas
on pizzas.pizza_id=order_details.pizza_id)*100,2) 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 order by revenue desc;
```

Output:

Result Grid | Filte

| category | revenue |
|---|---|
| Classic | 26.91 |
| Supreme | 25.46 |
| Chicken | 23.96 |
| Veggie | 23.68 |

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

Input query:

```
1    -- Group the orders by date and calculate the average number of pizzas ordered per day
2
3    select round(avg(quantity),0) as avg_pizza_ordered_per_day from
4    (Select orders.order_date, sum(order_details.quantity) as quantity
5    from orders join order_details
6    on orders.order_id = order_details.order_id
7    group by orders.order_date) as order_quantity
```

Output:

Result Grid | Filter Rows:

| avg_pizza_ordered_per_day |
| --- |
| 138 |

# Analyze the cumulative revenue generated over time.

Input query:

```sql
-- 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;
```

Output:

Result Grid | Filter Rows:

| order_date | cum_revenue |
|---|---|
| 2015-01-01 00:00:00 | 2713.8500000000004 |
| 2015-01-02 00:00:00 | 5445.75 |
| 2015-01-03 00:00:00 | 8108.15 |
| 2015-01-04 00:00:00 | 9863.6 |
| 2015-01-05 00:00:00 | 11929.55 |
| 2015-01-06 00:00:00 | 14358.5 |
| 2015-01-07 00:00:00 | 16560.7 |

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

Input query :

```
-- 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;
```

Output:

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 |

Result 1