# PIZZA SALES USING SQL

# SQL PROJECT ON PIZZA SALES

SQL, or Structured Query Language, is a standardized programming language specifically designed for managing and manipulating relational databases. It allows users to perform various operations on data stored in databases, such as querying, updating, inserting, and deleting records. Here are the key components and features of SQL:

A typical pizza sales database might consist of multiple tables, each representing different aspects of the business:

Using SQL for pizza sales data enables businesses to extract valuable insights about revenue, customer preferences, and sales trends. By leveraging SQL queries, businesses can effectively manage their operations, improve decision-making, and enhance customer satisfaction. SQL's ability to handle complex queries through joins, aggregations, and subqueries makes it an indispensable tool in analyzing pizza sales data and driving business strategies.

# RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

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

| | total_orders |
|---|---|
| ▶ | 21350 |

# CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES

```sql
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;
```

Result Grid

| | total_sales |
|---|---|
| ▶ | 817860.05 |

# IDENTIFY THE HIGHEST-PRICED PIZZA

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

Result Grid | Filter Rows:

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

# IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED

```
select quantity , count(order_details_id)
 from order_details group by quantity;
```

| | quantity | count(order_details_id) |
|---|---|---|
| ▶ | 1 | 47693 |
| | 2 | 903 |
| | 3 | 21 |
| | 4 | 3 |

# IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED

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

| size | COUNT(order_details.order_details_id) |
|---|---|
| L | 18526 |
| M | 15385 |
| S | 14137 |
| XL | 544 |
| XXL | 28 |

# LIST THE TOP 5 MOST ORDERED PIZZA TYPE ALONG WITH THEIR QUANTITIES

```sql
SELECT pizza_types.name, SUM(order_details.quantity) AS total_quantity
FROM order_details
JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id
JOIN pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id
GROUP BY pizza_types.name
ORDER BY total_quantity DESC
LIMIT 5;
```

| name | total_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

```
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

```
select hour (order_time), count(order_id) as order_count
from orders
group by hour(order_time);
```

| | hour (order_time) | order_count |
|---|---|---|
| | 9 | 1 |
| | 10 | 8 |
| | 23 | 28 |
| | 22 | 663 |
| | 21 | 1198 |
| | 11 | 1231 |
| | 15 | 1468 |
| | 14 | 1472 |
| | 20 | 1642 |
| | 16 | 1920 |
| | 19 | 2009 |
| | 17 | 2336 |
| | 18 | 2399 |
| ▶ | 13 | 2455 |
| | 12 | 2520 |

# JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS

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

| | 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.

```
select avg(quantity ) from
(select orders.order_date,sum(order_details.quantity)
from orders join order_details
on orders.order_id =order_details_id
group by orders.order_date) as order_quantity;

SELECT AVG(order_quantity.total_quantity)
FROM (
    SELECT orders.order_date, SUM(order_details.quantity) AS total_quantity
    FROM orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY orders.order_date
) AS order_quantity
LIMIT 0, 1000;
```

| AVG(order_quantity.total_quantity) |
| --- |
| 138.4749 |

# 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 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.

```sql
SELECT pizza_types.category,
     round( SUM(order_details.quantity * pizzas.price) ,3) AS revenue
FROM pizza_types
JOIN pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizza_types.category
ORDER BY revenue DESC
LIMIT 0, 1000;
```

| category | revenue |
|---|---|
| Classic | 220053.1 |
| Supreme | 208197 |
| Chicken | 195919.5 |
| Veggie | 193690.45 |

# ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```sql
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;
```

| order_date | cum_revenue |
|---|---|
| 2015-01-01 | 2713.8500000000004 |
| 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 | 23990.350000000002 |
| 2015-01-11 | 25862.65 |
| 2015-01-12 | 27781.7 |
| 2015-01-13 | 29831.300000000003 |
| 2015-01-14 | 32358.700000000004 |
| 2015-01-15 | 34343.50000000001 |
| 2015-01-16 | 36937.65000000001 |
| 2015-01-17 | 39001.75000000001 |
| 2015-01-18 | 40978.600000000006 |
| 2015-01-19 | 43365.75000000001 |
| 2015-01-20 | 45763.65000000001 |
| 2015-01-21 | 47804.20000000001 |
| 2015-01-22 | 50300.90000000001 |

# THANK YOU!