

PIZZA SALES ANALYSIS

SQL

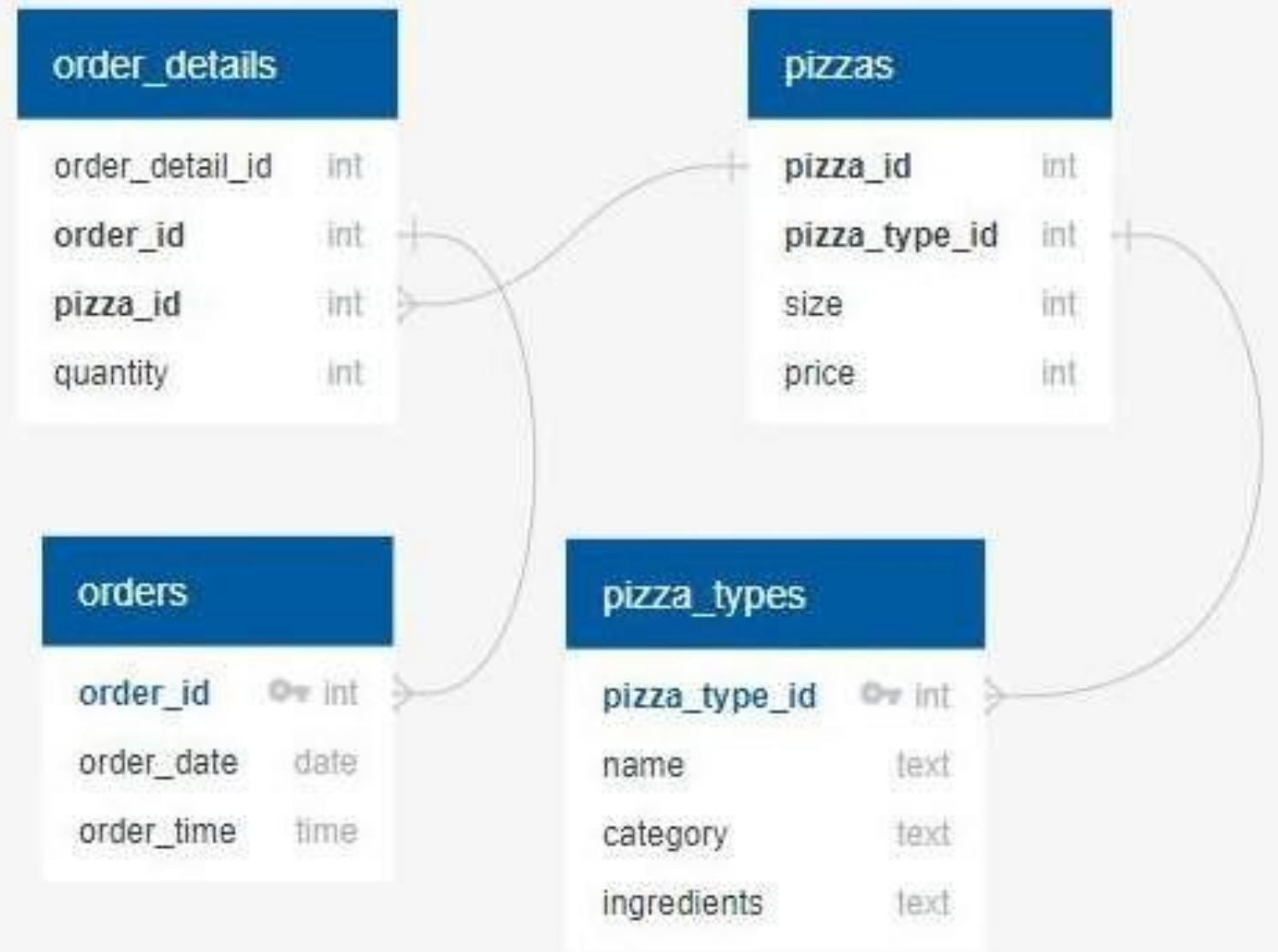


OVERVIEW

Welcome to my Pizza Sales Analysis project! In this project, I used SQL to explore and analyze pizza sales data to uncover meaningful insights. Through various SQL queries, I examined sales trends, customer preferences, and revenue patterns across different pizza types, sizes, and categories. This project demonstrates my skills in data analysis, SQL querying, and drawing actionable insights from complex datasets



RELATIONAL TABLE



TOTAL ORDERS

SQL File 3* Administration - Dashboard

Limit to 1000 rows

```
1 -- Retrieve the total number of orders placed.  
2  
3 • select count(order_id) As total_orders  
4 from pizzadb.orders;
```

Result Grid

	total_orders
▶	21350

Export: Wrap Cell Content:

Result Grid



TOTAL SALES

SQL File 3* SQL File 4* x SQL File 5

Limit to 1000 rows

```
1 -- Calculate the total revenue generated from pizza sales.
2 SELECT
3     SUM(ROUND(order_detail_id.quantity * pizzas.price)) AS total_sales
4 FROM
5     pizzadb.order_detail_id
6     JOIN
7     pizzas ON order_detail_id.pizza_id = pizzas.pizza_id
8
9
```

total_sales

815306

Filter Rows: Export: Wrap Cell Content:



HIGHEST-PRICED PIZZA

SQL File 3* SQL File 4* SQL File 5* x

Limit to 1000 rows

```
1  -- Identify the highest-priced pizza--
2  SELECT
3      pizza_types.name, pizzas.price
4  FROM
5      pizzadb.pizza_types
6      JOIN
7      pizzadb.pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
8  ORDER BY price DESC
9  LIMIT 1
10
```

	name	price
▶	The Greek Pizza	35.95

Export: Wrap Cell Content: Fetch rows:



MOST ORDERED PIZZA SIZE

SQL File 3* SQL File 4* SQL File 5* SQL File 6* x

Limit to 1000 rows

```
1 -- Identify the most common pizza size ordered
2 SELECT
3     pizzas.size,
4     COUNT(order_details.order_details_id) AS order_count
5 FROM
6     pizzas
7     JOIN
8     order_details ON pizzas.pizza_id = order_details.pizza_id
9 GROUP BY pizzas.size
10 ORDER BY order_count DESC;
```

	size	order_count
▶	L	18526
	M	15385
	S	14137
	XL	544
	XXL	28

Export: Wrap Cell Content:



TOP 5 MOST ORDERED PIZZA

SQL File 4* SQL File 5* SQL File 6* SQL File 7* SQL File 8* x

Limit to 1000 rows

```
1  -- List the top 5 most ordered pizza types along with their quantities--
2  SELECT
3      pizza_types.name AS names,
4      SUM(order_details.quantity) AS quantity
5  FROM
6      order_details
7      JOIN
8      pizzas ON order_details.pizza_id = pizzas.pizza_id
9      JOIN
10     pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
11  GROUP BY names
12  ORDER BY quantity DESC
```

Result Grid

names	quantity
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371
The California Chicken Pizza	2370
The Sicilian Pizza	1938
The Spicy Italian Pizza	1924



QUANTITY ORDERED BY CATEGORY

SQL File 4* SQL File 5* SQL File 6* SQL File 7* SQL File 9* x

Limit to 1000 rows

```
1  -- Join the necessary tables to find the total quantity of each pizza category ordered.--
2  SELECT
3      pizza_types.category AS pizza_category,
4      SUM(order_details.quantity) AS order_quantity
5  FROM
6      pizza_types
7      JOIN
8      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
9      JOIN
10     order_details ON pizzas.pizza_id = order_details.pizza_id
11  GROUP BY pizza_category
12  ORDER BY order_quantity DESC
```

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

pizza_category	order_quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050



DISTRIBUTION OF ORDERS BY HOUR

SQL File 4* SQL File 5* SQL File 6* SQL File 7* SQL File 10* x

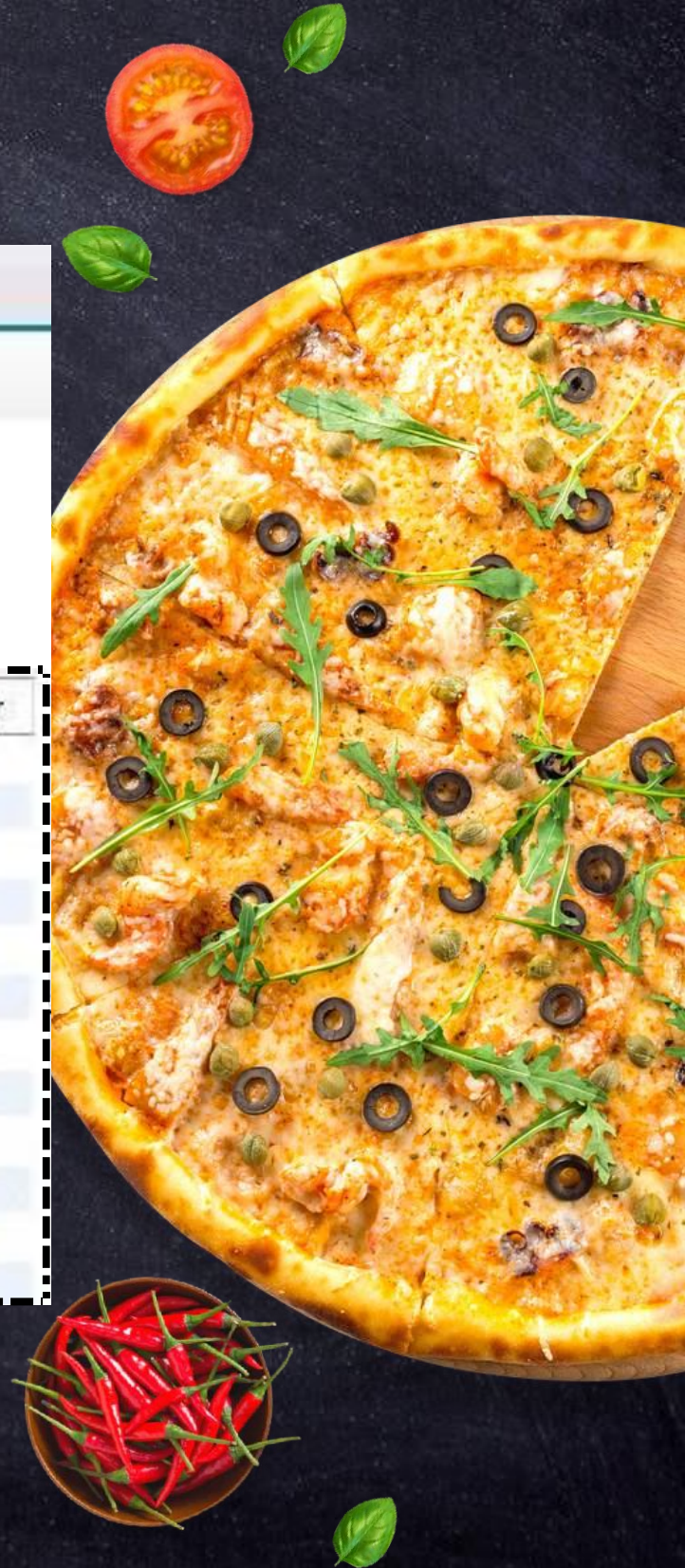
Limit to 1000 rows

```
1 -- Determine the distribution of orders by hour of the day.
2 SELECT
3     HOUR(order_time) AS hour, COUNT(order_id) AS total_order
4 FROM
5     pizzadb.orders
6 GROUP BY hour
```

Result Grid

	hour	total_order
▶	11	1231
	12	2520
	13	2455

	hour	total_order
▶	11	1231
	12	2520
	13	2455
	14	1472
	15	1468
	16	1920
	17	2336
	18	2399
	19	2009
	20	1642
	21	1198
	22	663





DISTRIBUTION BY CATEGORY

SQL File 4* SQL File 5* SQL File 6* SQL File 7* SQL File 11* x

Limit to 1000 rows

```
1 -- Join relevant tables to find the category-wise distribution of pizzas.
2 SELECT
3     category, COUNT(category) AS count_of_pizzas
4 FROM
5     pizzadb.pizza_types
6 GROUP BY category
```

	category	count_of_pizzas
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9

Export:  Wrap Cell Content: 



AVERAGE PIZZA ORDERED PER DAY

SQL File 4* SQL File 5* SQL File 6* SQL File 7* SQL File 12* x

Limit to 1000 rows

```
1  -- Group the orders by date and calculate the average number of pizzas ordered per day.
2  SELECT round(avg(total_quantity)) AS avg_pizzas_per_day
3  FROM (
4      SELECT
5          order_date AS date,
6          SUM(order_details.quantity) AS total_quantity
7      FROM
8          pizzadb.order_details
9      JOIN
10         pizzadb.orders
11      ON
12         order_details.order_id = orders.order_id
13      GROUP BY
14         order_date
15  ) AS order_quantity;
```

	avg_pizzas_per_day
▶	138



TOP 3 PIZZA ORDERED BY REVENUE

SQL File 4* SQL File 5* SQL File 6* SQL File 7* SQL File 13* x

Limit to 1000 rows

```
1 -- Determine the top 3 most ordered pizza types based on revenue.
2 SELECT
3     name, ROUND(SUM(quantity * price), 1) AS revenue
4 FROM
5     pizzas
6     JOIN
7     pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id
8     JOIN
9     order_details ON pizzas.pizza_id = order_details.pizza_id
10 GROUP BY name
11 ORDER BY revenue DESC
```

Result Grid

name	revenue
The Thai Chicken Pizza	43434.2
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5



CONTRIBUTION BY PERCENTAGE

SQL File 4* SQL File 5* SQL File 6* SQL File 7* SQL File 15* x SQL File 17* SQL File 18*

Limit to 1000 rows

```
1 -- Calculate the percentage contribution of each pizza type to total revenue.
2 • SELECT
3     category,
4     CONCAT(ROUND((SUM(quantity * price) / (SELECT
12 FROM
13     pizzadb.order_details
14     JOIN
15     pizzadb.pizzas ON order_details.pizza_id = pizzas.pizza_id
16     JOIN
17     pizzadb.pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
18 GROUP BY category
19 ORDER BY percentage_contribution DESC;
```


Result Grid

category	percentage_contribution
Classic	26.91%
Supreme	25.46%
Chicken	23.96%
Veggie	23.68%






CUMULATIVE REVENUE BY DATE

SQL File 4* SQL File 5* SQL File 6* SQL File 7* SQL File 15* SQL File 17* x SQL File 18*

Limit to 1000 rows

```
1  -- Analyze the cumulative revenue generated over time
2  SELECT
3      order_date,
4      ROUND(SUM(revenue) OVER (ORDER BY order_date), 1) AS cum_revenue
5  FROM
6      (SELECT
7          orders.order_date,
8          SUM(order_details.quantity * pizzas.price) AS revenue
9      FROM order_details
10     JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id
11     JOIN orders ON orders.order_id = order_details.order_id
12     GROUP BY orders.order_date) AS sales;
```

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

	order_date	cum_revenue
▶	2015-01-01	2713.9

	order_date	cum_revenue
▶	2015-01-01	2713.9
	2015-01-02	5445.8
	2015-01-03	8108.2
	2015-01-04	9863.6
	2015-01-05	11929.6



TOP 3 PIZZA TYPE BY REVENUE FOR EACH CATEGORY

```
SQL File 4*  SQL File 5*  SQL File 6*  SQL File 7*  SQL File 15*  SQL File 17*  SQL File 18* x
Limit to 1000 rows
1  -- Determine the top 3 most ordered pizza types based on revenue for each pizza category.
2  SELECT
3      name, revenue
4  FROM (SELECT
5      category, name, revenue,
6      RANK() OVER (PARTITION BY category ORDER BY revenue DESC) AS rn
7  FROM (SELECT
8      pizza_types.category, pizza_types.name,
9      SUM(order_details.quantity * pizzas.price) AS revenue
10     FROM pizza_types
11     JOIN pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
12     JOIN order_details ON order_details.pizza_id = pizzas.pizza_id
13     GROUP BY pizza_types.category, pizza_types.name) AS a ) AS b
14 WHERE rn <= 3;
```

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

