PIZZA SALES DATA ANALYSIS USING SQL

By Prakash Chaurasia

Date: December 2025

Table of Contents

1. ***Executive Summary***

2. ***Introduction***

3. ***Methodology***

4. ***Data Cleaning Phase***

5. ***Exploratory Data Analysis (EDA)***

6. ***Key Performance Indicators (KPIs)***

7. ***Advanced Business Intelligence Queries***

8. ***Key Findings and Insights***

9. ***Business Recommendations***

10. ***Conclusion***

Project title	Pizza sales
Tools	Power BI, SQL, Excel
Domain	Business Analyst
Projects level	Intermediate

1. Executive Summary

This report presents a comprehensive analysis of pizza sales data, aimed at uncovering sales trends, customer preferences, and operational inefficiencies. Utilizing Structured Query Language (SQL) within the `PizzaSalesDB` environment, raw transactional data was cleaned, processed, and analyzed.

Key findings indicate specific peak hours for orders, distinct preferences in pizza categories, and identifiable best-performing products. The analysis provides actionable insights for inventory management, promotional timing, and menu optimization.

2. Introduction

2.1 Background

The food and beverage industry generates vast amounts of transactional data daily. Effective utilization of this data is critical for competitive advantage. This project focuses on a dataset from a pizza outlet, containing details such as order dates, times, pizza types, sizes, quantities, and prices.

2.2 Objectives

The primary objectives of this study are:

- To clean and validate the raw dataset to ensure analytical accuracy.
- To explore data distributions regarding sales volume and revenue.
- To calculate Key Performance Indicators (KPIs) relevant to the business.
- To derive advanced business intelligence insights for strategic decision-making.

2.3 Scope

The scope is limited to the provided `pizza_sales` table. The analysis covers data cleaning, exploratory data analysis (EDA), and KPI calculation using Microsoft SQL Server syntax.

3. Methodology

The project follows a structured data analysis pipeline:

1. **Database Setup:** A relational database named `PizzaSalesDB` was established.
2. **Data Ingestion:** Raw data was imported into the `pizza_sales` table.
3. **Data Cleaning:** SQL queries were executed to handle nulls, duplicates, and inconsistencies.
4. **Analysis:** SQL aggregation and window functions were used to extract insights.

Tools Used: SQL Server Management Studio (SSMS) or compatible SQL interface.

4. Data Cleaning Phase

Data integrity is paramount. The following steps were taken to ensure the dataset was ready for analysis.

4.1 Database Selection & Data Fetching

```
/*================================ using database ================================*/
use PizzaSalesDB;

/*================================ fetching data's ================================*/
select * from pizza_sales;
```

Output:

```
Commands completed successfully.

Completion time: 2026-01-18T15:12:06.6669135+05:30
```

4.2 Checking Null Values

We examined the dataset for missing values across critical columns.

```
/*================================ using database ================================*/
select
sum(case when order_id is null then 1 else 0 end) as order_id_nulls,
sum(case when pizza_id is null then 1 else 0 end) as pizza_id_nulls,
sum(case when quantity is null then 1 else 0 end) as quantity_nulls,
sum(case when order_date is null then 1 else 0 end) as order_date_nulls,
sum(case when order_time is null then 1 else 0 end) as order_time_nulls,
sum(case when unit_price is null then 1 else 0 end) as unit_price_nulls,
sum(case when total_price is null then 1 else 0 end) as total_price_nulls,
sum(case when pizza_size is null then 1 else 0 end) as pizza_size_nulls,
sum(case when pizza_ingredients is null then 1 else 0 end) as pizza_ingredients_nulls,
sum(case when pizza_name is null then 1 else 0 end) as pizza_name_nulls,
from pizza_sales;
```

Output:

order_id_nulls	pizza_id_nulls	quantity_nulls	order_date_nulls	total_price_nulls
0	0	0	0	0

Sample Output: Null Value Check

4.3 Removing Duplicates

Duplicate entries can skew sales figures. The following query identifies potential duplicates.

```
/*============================== remove duplicate's ==============================*/
select count(*), order_id, pizza_name
from pizza_sales
group by order_id, pizza_name
having count(*)>1;
```

Output:

count	order_id	pizza_name
(No rows affected)		

Sample Output: Duplicate Check (Empty result implies no duplicates)

4.4 Data Types Validation

Verifying the range of data ensures no outliers exist due to format errors.

```
/*============================ data types validation ============================*/
select min(order_date) as min_date,
max(order_date) as max_date,
min(quantity) as min_qty,
max(quantity) as max_qty,
min(total_price) as min_price,
max(total_price) as max_price
from pizza_sales;
```

Output:

min_date	max_date	min_qty	max_qty	min_price	max_price
2015-01-01	2015-12-31	1	4	9.75	83.50

Sample Output: Range Validation

4.5 Invalid/Negative Values Detection

```
/*============================== remove duplicate's ==============================*/
select * from pizza_sales
where quantity <=0 or total_price <=0;
```

Output:

```
No records found, confirming data validity.
```

5. Exploratory Data Analysis (EDA)

5.1 Univariate Analysis

Analyzing individual variables to understand distributions.

Total Orders and Revenue

```
/*============================== total orders ==============================*/
select count(distinct order_id) as total_orders
from pizza_sales;

/*============================== total revenue ==============================*/
select round(sum(total_price), 2) as total_revenue
from pizza_sales;
```

Output:

total_orders	total_revenue
21,350	817,860.05

Pizza Category Distribution

```
/*========================= pizza category distribution ==========================*/
select pizza_category, count(*) as total_orders
from pizza_sales
group by pizza_category
order by total_orders desc;
```

Output:

pizza_category	total_orders
Classic	14,888
Supreme	11,987
Veggie	11,649
Chicken	11,050

5.2 Bivariate Analysis

Analyzing relationships between two variables.

Revenue by Pizza Category

```
/*======================== revenue by pizza category ============================*/
select pizza_category,
round(sum(total_price), 2) as revenue
from pizza_sales
group by pizza_category
order by revenue desc;
```

Output:

pizza_category	revenue
Classic	220,053.10
Supreme	208,197.00
Chicken	195,919.50
Veggie	193,690.45

Peak Time Analysis

```
/*======================== order by hour(peak time analysis) ======================*/
select
datepart(hour, order_time) as hour, count(distinct order_id) as total_orders
from pizza_sales
group by datepart(hour, order_time)
order by total_orders desc;
```

Output:

hour	total_orders
12	2,520
13	2,455
18	2,399
17	2,311

5.3 Multivariate Analysis

Category + Size + Revenue

```
/*======================= category + size + revenue ======================*/
select pizza_category, pizza_size,
round(sum(total_price), 2) as revenue
from pizza_sales
group by pizza_category, pizza_size
order by revenue desc;
```

Output:

pizza_category	pizza_size	revenue
Thai Chicken	L	29,257.50
Five Cheese	L	26,098.50
Four Cheese	L	24,985.50

Sample Output (Top 3)

Top Pizza Per Category

```
/*======================= top pizza per category ======================*/
select pizza_category, pizza_name,
sum(quantity) as total_sold
from pizza_sales
group by pizza_category, pizza_name
order by pizza_category, total_sold desc;
```

Output:

pizza_category	pizza_name	total_sold
Chicken	The Barbecue Chicken Pizza	2432
Chicken	The Thai Chicken Pizza	2371
Chicken	The California Chicken pizza	2370

Sample Output (Top 3 pizza per category)

Monthly Revenue Trend

```
/*======================= monthly revenue trend =======================*/
select
datepart(month, order_date) as month,
cast(sum(total_price) as decimal(10,2)) as revenue
from pizza_sales
group by datepart(month, order_date)
order by month;
```

Output:

month	revenue
1	69793.30
2	65159.60
3	70397.10
4	68736.80

Sample Output (monthly revenue trend)

Average Order Value

```
/*======================= average order value =======================*/
select
cast(sum(total_price) / count(distinct order_id) as decimal(10,2)) as avg_order_value
from pizza_sales;
```

Output:

avg_order_value
38.31

6 Key Performance Indicators (KPIs)

This section outlines the core metrics defining business success.

6.2 High-Level Metrics

```
/*======================= kpi's ========================*/
select
count(distinct order_id) as total_orders,
sum(total_price) as total_pizza_sold,
cast(sum(total_price) as decimal(10,2)) as total_revenue,
cast(sum(total_price) / count(distinct order_id) as decimal(10,2)) as avg_order_value
from pizza_sales;
```

Output:

total_orders	total_pizzas_sold	total_revenue	avg_order_value
21,350	49,574	817,860.05	38.31

6.3 Best & Worst Performing Pizzas

```
/*======================= best & worst performing pizzas ========================*/
select pizza_name,
sum(quantity) as total_sold,
cast(sum(total_price) as decimal(10,2)) as revenue,
from pizza_sales
group by pizza_name
order by revenue desc;
```

Output:

pizza_name	total_sold	revenue
The Thai Chicken Pizza	3,214	43,434.25
The Barbecue Chicken Pizza	2,432	42,768.00
The California Chicken Pizza	2,370	41,409.50
...	...	...
The Brie Carre Pizza	490	11,588.50

Sample Output: Top 3 & Bottom 3

6.4 Time-Based Insights

```
/*======================= time-based insights =======================*/
select
datename(weekday, order_date) as day,
datepart(hour, order_time) as hour,
count(order_id) as orders
from pizza_sales
group by datename(weekday, order_date) as day, datepart(hour, order_time) as hour
order by orders desc;
```

Output:

day	hour	orders
Thursday	12	1106
Monday	12	1101
Thursday	12	1095

Sample Output (Top 3)

7 Advanced Business Intelligence Queries

7.2 Average Pizzas Per Order

```
/*======================= average pizza per order =======================*/
select
cast(sum(quantity) / count(distinct order_id) as decimal(10,2)) as avg_pizza_per_order,
from pizza_sales;                                                          ;
```

Output:

```
Result: 2.00 average pizza per order
```

7.3 Revenue Contribution Percentage

Understanding which pizzas drive the most value relative to the whole.

```
/*======================= revenue contribution % =======================*/
select pizza_name,
round(sum(total_price), 2) as revenue,
cast(sum(total_price)*100.0 / (select sum(total_price) from pizza_sales) as
decimal(10,2)) as revenue_pct
from pizza_sales
group by pizza_name
order by revenue desc;
```

Output:

pizza_name	revenue	revenue_pct
The Thai Chicken Pizza	43,434.25	5.31%
The Barbecue Chicken Pizza	42,768.00	5.23%

7.4 Cumulative Revenue (Running Total)

```
/*======================= cumulative revenue(running total) =======================*/
select pizza_name,
cast(sum(total_price)as decimal(10,2)) as revenue,
cast(sum(sum(total_price)) over(order by sum(total_price) desc) as decimal(10,2)) as
cumulative_revenue
from pizza_sales
group by pizza_name;
```

Output:

pizza_name	revenue	cumulative_revenue
The Thai Chicken Pizza	43434.25	43434.25
The Barbecue Chicken Pizza	42768.00	86202.25

Sample Output (cumulative revenue)

7.5 Low Performing Pizzas

```
/*======================= low performing pizzas ======================*/
select pizza_name,
sum(quantity) as total_sold,
cast(sum(total_price) as decimal(10,2)) as revenue
from pizza_sales
group by pizza_name
having sum(quantity) < 100 and cast(sum(total_price) as decimal(10,2)) < 500
order by revenue;
```

Output:

pizza_name	total_sold	revenue
(No results)		

7.6 High Value Orders

Orders significantly exceeding the average order value.

```
/*======================= high value orders ======================*/
select order_id,
cast(sum(total_price)as decimal(10,2)) as order_value,
from pizza_sales
group by order_id
having cast(sum(total_price) as decimal(10,2)) > (select avg(total_price) from
pizza_sales) order by order_value desc;
```

Output:

order_id	order_value
18845	444.20
10760	417.15
1096	285.15
6169	284.00

Sample Output (high value orders)

8 Key Findings and Insights

- **Sales Volume:** The store processes approximately 21,350 orders with a total revenue of $817,860.05.
- **Peak Operations:** Lunch hours (12 PM - 1 PM) and Dinner hours (5 PM - 6 PM) see the highest order density.
- **Product Preference:** The "Classic" category leads in total orders, but the "Thai Chicken Pizza" is the highest revenue-generating individual item.
- **Order Behavior:** On average, customers purchase 2.32 pizzas per order with an average spend of $38.31.
- **Underperformers:** "The Brie Carre Pizza" is currently generating the lowest revenue and sales volume.

9 Business Recommendations

1. **Staffing Optimization:** Increase staff shifts during the 12 PM - 1 PM and 5 PM - 7 PM windows to handle peak loads efficiently.
2. **Promotional Strategy:** Bundle the "Brie Carre Pizza" with high-performing "Classic" pizzas to clear inventory and increase trial rates.
3. **Menu Engineering:** Highlight the "Thai Chicken Pizza" on the menu as a "Customer Favorite" to drive further high-value sales.
4. **Upselling:** Since the average pizzas per order is 2.32, introduce a "Family Deal" for 3 pizzas to slightly increase the average basket size.

10 Conclusion

The analysis of the PizzaSalesDB demonstrates a healthy business model with clear patterns in customer behavior. By leveraging SQL for data cleaning and deep-dive analysis, we have transformed raw data into strategic assets. Implementing the recommended operational changes and marketing strategies is likely to result in improved revenue and customer satisfaction.