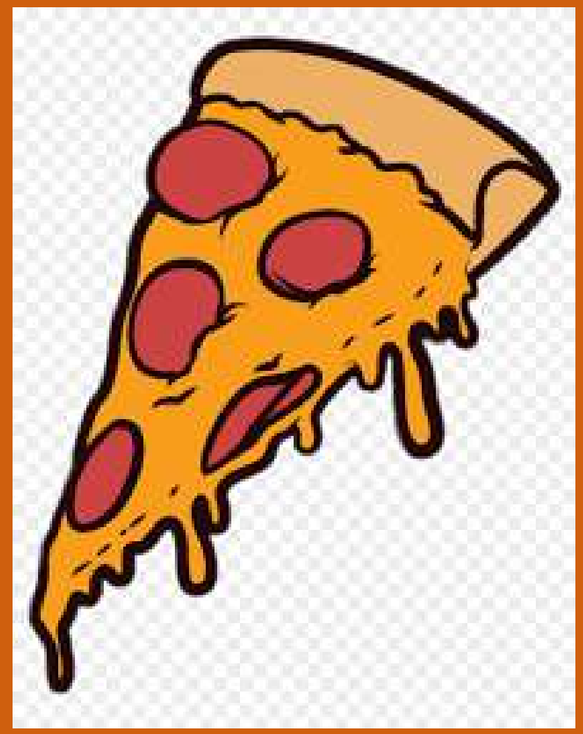
PIZZA SALES REPORT



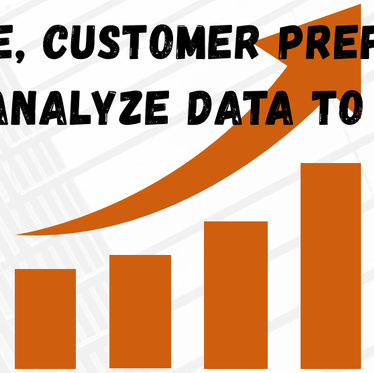
presented by

ARUNABH RANJAN

2024

INTRODUCTION

THIS PROJECT FOCUSES ON ANALYZING PIZZA SALES DATA USING MYSQL TO UNCOVER VALUABLE BUSINESS INSIGHTS. THE GOAL IS TO HELP DECISION-MAKERS UNDERSTAND SALES PERFORMANCE, CUSTOMER PREFERENCES, AND OPERATIONAL EFFICIENCY. BY WRITING SQL QUERIES, WE EXTRACT, CLEAN, AND ANALYZE DATA TO ANSWER KEY QUESTIONS SUCH AS:



WHICH PIZZAS ARE TOP SELLERS?

WHAT ARE PEAK ORDER TIMES?

HOW DO SALES VARY BY CATEGORY OR SIZE?

THIS PROJECT SHOWCASES THE POWER OF SQL IN REAL-WORLD DATA ANALYTICS SCENARIOS.

2024

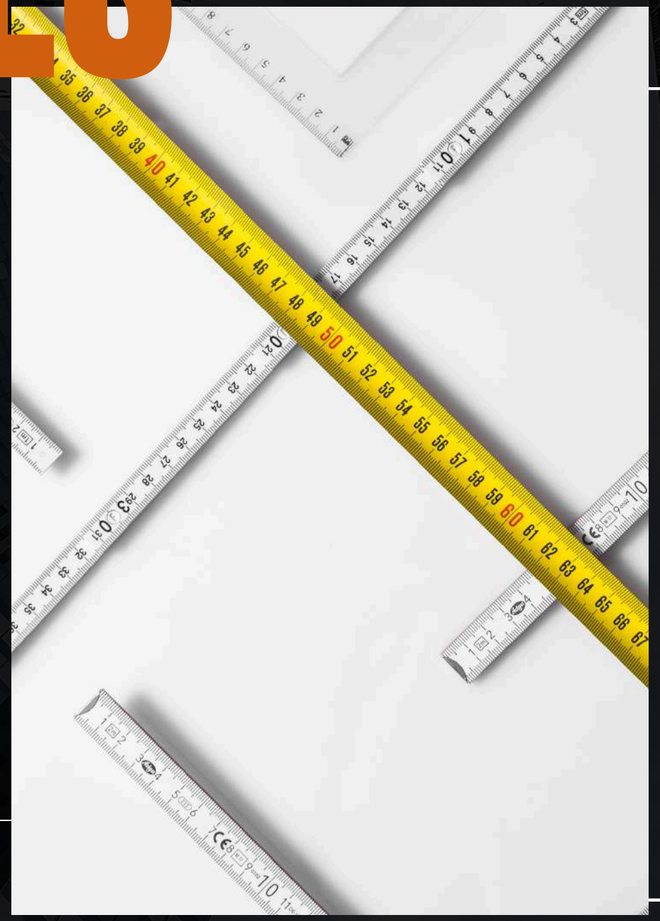
DATASET OVERVIEW

Tables:

- orders – order_id, date, time
- order_details – quantity, pizza_id
- pizzas – pizza_id, size, price
- pizza_types – name, category, ingredients

OBJECTIVES

2024



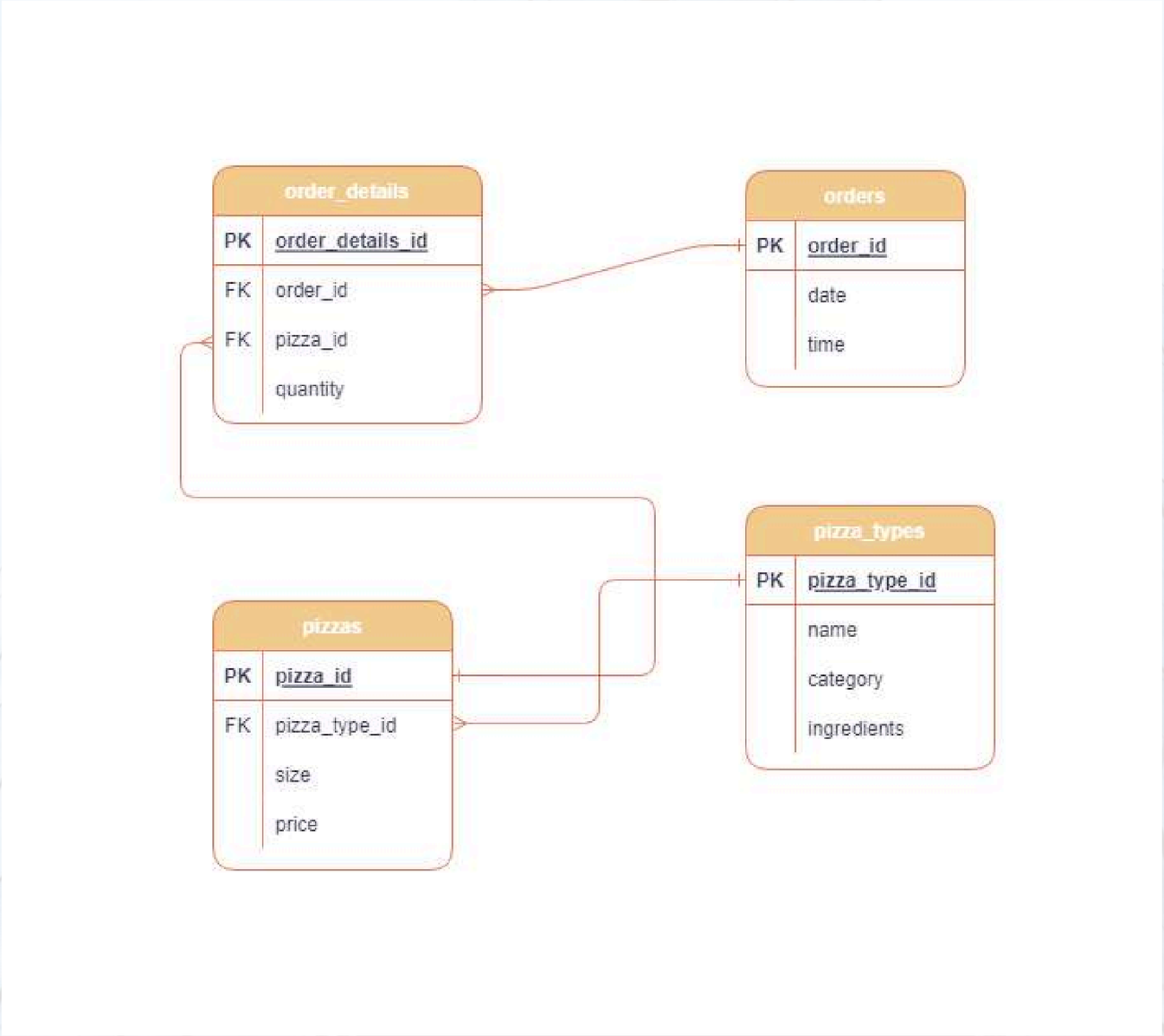
- Understand peak order times and days
- Find top-selling pizza types and categories
- Analyze sales by size and revenue trends
- Identify underperforming items
- Improve inventory and marketing strategy

2024

DATA CLEANING

- Removed duplicates and null values
- Standardized date and time formats
- Fixed inconsistent category and size values
- Ensured proper data types (e.g., DATE, TIME, INT, DECIMAL)

DATA MODELLING USING POWER BI



2024

- Total Revenue: SUM(price * quantity)
- Top 5 Pizzas by Sales
- Most Popular Pizza Sizes
- Sales by Day and Time
- Category-wise Sales Analysis

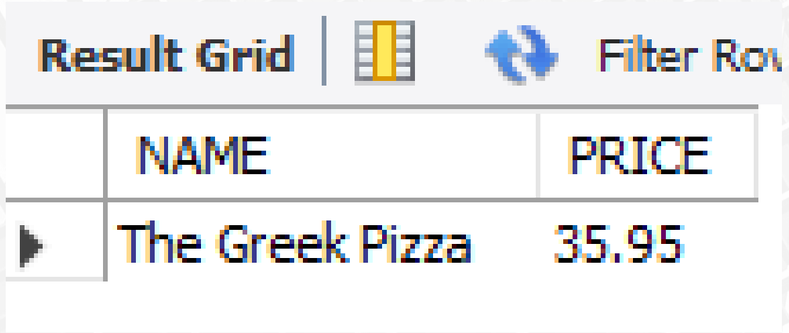
KEY QUERIES

```sql
-- Retrieve the total number of orders placed.

SELECT COUNT(ORDER_ID) AS TOTAL_ORDERS FROM ORDERS;
```

Result Grid

	TOTAL_ORDERS
▶	21350

```sql
-- 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;
```



Result Grid | Filter Rov

	NAME	PRICE
▶	The Greek Pizza	35.95

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

SELECT
ROUND(SUM((ORDER_DETAILS.QUANTITY*PIZZAS.PRICE)),2) AS REVENUE_GENERATED
FROM ORDER_DETAILS JOIN PIZZAS ON ORDER_DETAILS.PIZZA_ID = PIZZAS.PIZZA_ID;
```

Result Grid | Filter R

	REVENUE_GENERATED
▶	817860.05

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

SELECT pizza_types.NAME,SUM(ORDER_DETAILS.QUANTITY) AS C_t_o
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 C_t_o DESC LIMIT 5;
```

NAME	C_t_o
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

```sql
-- Identify the most common pizza size ordered.

SELECT PIZZAS.SIZE,COUNT(ORDER_DETAILS.ORDER_DETAIL_ID)
 AS ORDER_COUNT FROM PIZZAS JOIN order_details
 ON PIZZAS.PIZZA_ID = ORDER_DETAILS.PIZZA_ID
GROUP BY PIZZAS.SIZE ORDER BY ORDER_COUNT DESC;
```

SIZE	ORDER_COUNT
L	18526
M	15385
S	14137
XL	544
XXL	28

```sql
-- Join the necessary tables to find the total quantity of each pizza category ordered.

SELECT pizza_types.CATEGORY,SUM(order_details.QUANTITY) AS T_Q
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 CATEGORY ORDER BY T_Q;
```

CATEGORY	T_Q
Chicken	11050
Veggie	11649
Supreme	11987
Classic	14888

```sql
-- Determine the distribution of orders by hour of the day.

SELECT HOUR(ORDERS.ORDER_TIME), COUNT(ORDER_ID) FROM ORDERS
GROUP BY HOUR(ORDERS.ORDER_TIME);
```

	HOUR(ORDERS.ORDER_TIME)	COUNT(ORDER_ID)
▶	11	1231
	12	2520
	13	2455
	14	1472
	15	1468
	16	1920
	17	2336
	18	2399
	19	2009
	20	1642
	21	1198
	22	663
	23	28
	10	8
	9	1

```sql
-- Join relevant tables to find the category-wise distribution of pizzas.
SELECT
    category AS category_name, COUNT(name) AS pizza_count
FROM
    pizza_types
GROUP BY 1
ORDER BY 2 DESC;
```

category_name	pizza_count
Supreme	9
Veggie	9
Classic	8
Chicken	6

```sql
-- Group the orders by date and calculate the average number of pizzas ordered per day.

SELECT ORDERS.ORDER_DATE , round(AVG(ORDER_DETAILS.QUANTITY),2) FROM ORDERS JOIN ORDER_DETAILS
ON ORDERS.ORDER_ID = ORDER_DETAILS.ORDER_ID
GROUP BY ORDERS.ORDER_DATE;
```

ORDER_DATE	round(AVG(ORDER_DETAILS.QUANTITY),2)
2015-01-01	1.01
2015-01-02	1.03
2015-01-03	1.03
2015-01-04	1.00
2015-01-05	1.03
2015-01-06	1.02
2015-01-07	1.04
2015-01-08	1.01
2015-01-09	1.03
2015-01-10	1.01
2015-01-11	1.02
2015-01-12	1.01
2015-01-13	1.03
2015-01-14	1.04
2015-01-15	1.00
2015-01-16	1.02

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

SELECT PIZZA_TYPE_ID,
ROUND(SUM((ORDER_DETAILS.QUANTITY*PIZZAS.PRICE)),2) AS REVENUE_GENERATED
FROM ORDER_DETAILS JOIN PIZZAS ON ORDER_DETAILS.PIZZA_ID = PIZZAS.PIZZA_ID
GROUP BY PIZZA_TYPE_ID ORDER BY REVENUE_GENERATED DESC LIMIT 3;
```

Result Grid | Filter Rows:

PIZZA_TYPE_ID	REVENUE_GENERATED
thai_ckn	43434.25
bbq_ckn	42768
cali_ckn	41409.5

```sql
WITH total_revenue_cte AS (
    SELECT
        ROUND(SUM(od.quantity * p.price), 2) AS total_revenue
    FROM
        order_details od
        JOIN pizzas p ON p.pizza_id = od.pizza_id
),
category_revenue_cte AS (
    SELECT
        pt.category AS pizza_category,
        SUM(od.quantity * p.price) AS category_revenue
    FROM
        pizza_types pt
        JOIN pizzas p ON p.pizza_type_id = pt.pizza_type_id
        JOIN order_details od ON od.pizza_id = p.pizza_id
    GROUP BY
        pt.category
)
SELECT
    cr.pizza_category,
    ROUND((cr.category_revenue / tr.total_revenue) * 100, 2) AS revenue
FROM
    category_revenue_cte cr CROSS JOIN total_revenue_cte tr
ORDER BY
    revenue DESC;
```

pizza_category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

```sql
-- Analyze the cumulative revenue generated over time.

SELECT order_date, SUM(revenue) OVER(ORDER BY order_date) AS cumulative_revenue
FROM
(SELECT
    orders.order_date,
    round(SUM(order_details.quantity * pizzas.price),2) 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 1) AS sales;
```

Result Grid | Filter Rows:

order_date	cumulative_revenue
2015-01-01	2713.85
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	21526.399999999998
2015-01-10	23990.35

```sql
-- Determine the top 3 most ordered pizza types based on revenue for each pizza c



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 LIMIT 3;
```

Result Grid | Filter Rows:

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

THANK YOU!