PIZZA SALES SQL ANALYTICS PROJECT

Business Insights & Revenue Analysis using MySQL

Neeraj Gupta
Aspiring Data Analyst

Tools: MySQL, MySQL Workbench

BUSINESS PROBLEM:

A pizza store wants to understand:

- How much revenue it is generating
- Which pizzas and categories perform best
- Peak order times and sales trends
- Product performance by size and category

OBJECTIVE:

Use SQL to convert raw transactional data into actionable business insights.

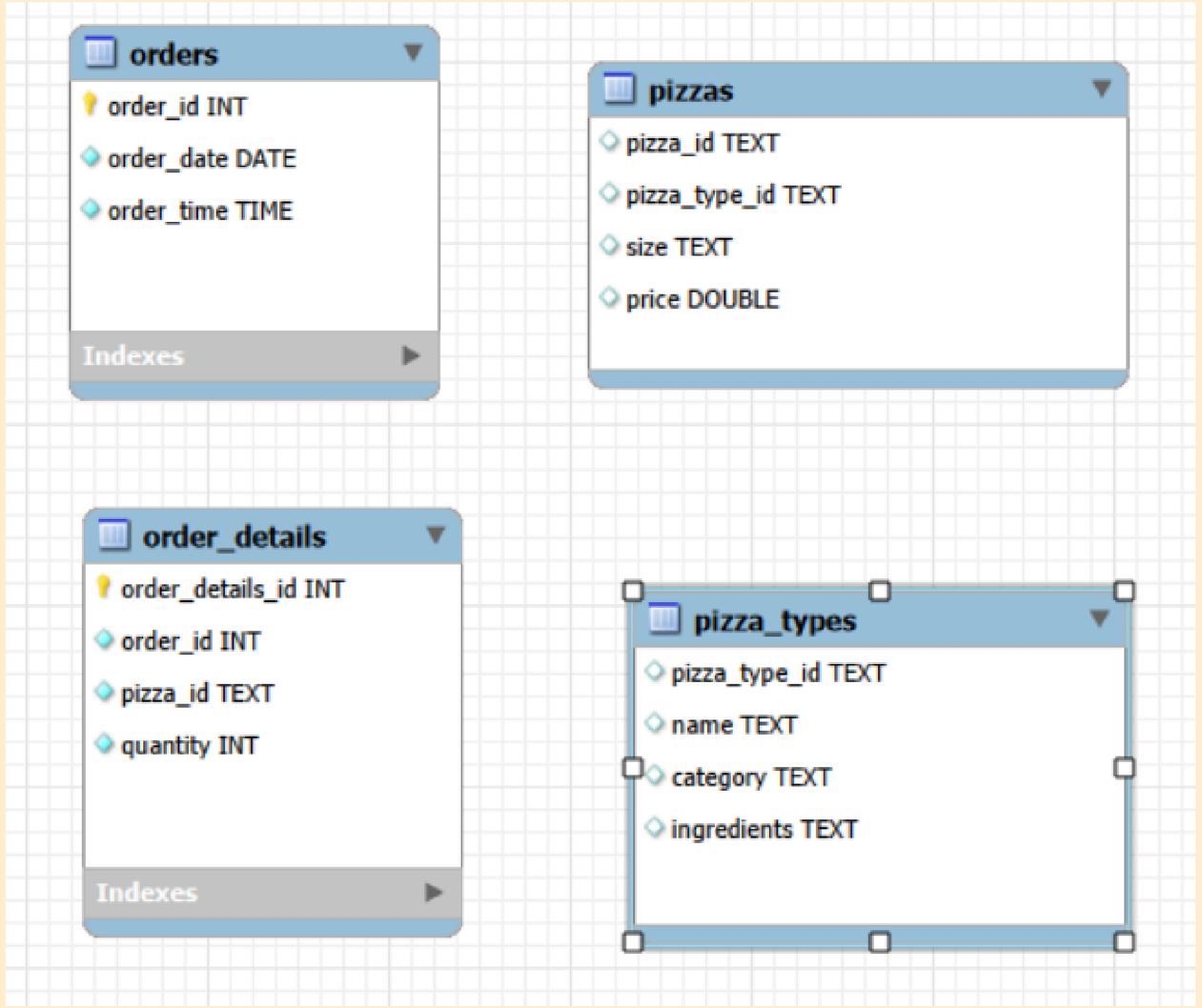
DATASET OVERVIEW:

- Realistic pizza sales transactional dataset
- Covers 1 full year of order data
- ~50,000+ order records
- Designed for sales and revenue analysis

DATABASE DESIGN:

- Relational schema with logical table relationships
- Tables connected using common identifiers (order_id, pizza_id, pizza_type_id)
- Relationships handled via SQL JOINs for analytical querying
- Optimized for reporting and business analysis

EER DIAGRAM :



CALCULATE THE TOTAL NUMBER OF PIZZAS SOLD (SUM OF QUANTITIES ACROSS ALL ORDERS).

```
SELECT
    SUM(quantity) AS total_pizzas_sold
FROM
    order_details;
```

Result Grid | Filter Ro

total_pizzas_sold
49574

DETERMINE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

```sql
SELECT
    ROUND(SUM(od.quantity * p.price), 2) AS total_revenue
FROM
    order_details od
        JOIN
    pizzas p ON od.pizza_id = p.pizza_id;
```

Result Grid | Filter Rows:

total_revenue
817860.05

IDENTIFY THE NUMBER OF PIZZAS SOLD BY SIZE (SMALL, MEDIUM, LARGE, EXTRA LARGE).

```sql
SELECT
    p.size, SUM(od.quantity) AS pizza_sold
FROM
    pizzas p
        JOIN
    order_details od ON p.pizza_id = od.pizza_id
GROUP BY p.size;
```

Result Grid | Filter Rows: | Ex

size	pizza_sold
M	15635
L	18956
S	14403
XL	552
XXL	28

CALCULATE THE TOTAL NUMBER OF PIZZAS SOLD PER CATEGORY (E.G., CLASSIC, CHICKEN, VEGGIE, SUPREME).

```sql
SELECT
    pt.category, SUM(od.quantity) AS pizza_sold
FROM
    order_details od
        JOIN
    pizzas p ON od.pizza_id = p.pizza_id
        JOIN
    pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.category
ORDER BY pizza_sold DESC;
```

Result Grid | Filter Rows: | Export

category	pizza_sold
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

IDENTIFY THE HIGHEST-PRICED PIZZA.

```sql
SELECT
    p.pizza_id, pt.name, p.size, p.price
FROM
    pizzas p
        JOIN
    pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
WHERE
    p.price = (SELECT
            MAX(price)
        FROM
            pizzas);
```

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

pizza_id	name	size	price
the_greek_xxl	The Greek Pizza	XXL	35.95

CALCULATE THE TOTAL REVENUE GENERATED FOR EACH DAY.

```sql
SELECT
    o.order_date,
    ROUND(SUM(od.quantity * p.price), 2) AS day_total_revenue
FROM
    orders o
        JOIN
    order_details od ON o.order_id = od.order_id
        JOIN
    pizzas p ON od.pizza_id = p.pizza_id
GROUP BY o.order_date
ORDER BY o.order_date;
```

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

order_date	day_total_revenue
2015-01-01	2713.85
2015-01-02	2731.9
2015-01-03	2662.4
2015-01-04	1755.45
2015-01-05	2065.95

IDENTIFY THE TOP 5 BEST-SELLING PIZZAS BASED ON TOTAL QUANTITY SOLD.

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

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

pizza_type_id	name	category	total_quantity_sold
classic_dlx	The Classic Deluxe Pizza	Classic	2453
bbq_ckn	The Barbecue Chicken Pizza	Chicken	2432
hawaiian	The Hawaiian Pizza	Classic	2422
pepperoni	The Pepperoni Pizza	Classic	2418
thai_ckn	The Thai Chicken Pizza	Chicken	2371

DETERMINE THE HOUR OF THE DAY WITH THE HIGHEST NUMBER OF ORDERS.

```sql
SELECT HOUR(order_time) AS order_hour,
       COUNT(*) AS total_orders
FROM orders
GROUP BY HOUR(order_time)
ORDER BY total_orders DESC;
```

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

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

IDENTIFY PIZZAS THAT HAVE NEVER BEEN ORDERED.

```sql
SELECT
    pt.pizza_type_id,
    pt.name,
    pt.category,
    pt.ingredients
FROM pizza_types pt
LEFT JOIN pizzas p
    ON pt.pizza_type_id = p.pizza_type_id
LEFT JOIN order_details od
    ON p.pizza_id = od.pizza_id
WHERE od.order_id IS NULL;
```

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

pizza_type_id	name	category	ingredients
big_meat	The Big Meat Pizza	Classic	Bacon, Pepperoni, Italian Sausage, Chorizo Sau...
big_meat	The Big Meat Pizza	Classic	Bacon, Pepperoni, Italian Sausage, Chorizo Sau...
five_cheese	The Five Cheese Pizza	Veggie	Mozzarella Cheese, Provolone Cheese, Smoked ...
five_cheese	The Five Cheese Pizza	Veggie	Mozzarella Cheese, Provolone Cheese, Smoked ...
four_cheese	The Four Cheese Pizza	Veggie	Ricotta Cheese, Gorgonzola Piccante Cheese, ...

FIND PIZZA TYPES THAT WERE ORDERED, BUT ONLY IN ONE SIZE (S OR M OR L), NEVER IN MULTIPLE SIZES.

```sql
SELECT
    pt.pizza_type_id,
    pt.name,
    COUNT(DISTINCT p.size) AS size_count
FROM pizza_types pt
JOIN pizzas p
    ON pt.pizza_type_id = p.pizza_type_id
JOIN order_details od
    ON p.pizza_id = od.pizza_id
GROUP BY
    pt.pizza_type_id, pt.name
HAVING COUNT(DISTINCT p.size) = 1;
```

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

pizza_type_id	name	size_count
big_meat	The Big Meat Pizza	1
brie_carre	The Brie Carre Pizza	1
five_cheese	The Five Cheese Pizza	1

FIND THE TOP 3 REVENUE-GENERATING PIZZAS IN EACH CATEGORY.

```sql
WITH pizza_revenue AS (
    SELECT
        pt.pizza_type_id,
        pt.name AS pizza_name,
        pt.category,
        SUM(od.quantity * p.price) AS total_revenue
    FROM pizza_types pt
    JOIN pizzas p
        ON pt.pizza_type_id = p.pizza_type_id
    JOIN order_details od
        ON p.pizza_id = od.pizza_id
    GROUP BY pt.pizza_type_id, pt.name, pt.category
),
ranked_pizzas AS (
    SELECT
        pizza_name,
        category,
        total_revenue,
        RANK() OVER (PARTITION BY category ORDER BY total_revenue DESC) AS category_rank
    FROM pizza_revenue
)
SELECT *
FROM ranked_pizzas
WHERE category_rank <= 3
ORDER BY category, category_rank;
```

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

pizza_name	category	total_revenue	category_rank
The Thai Chicken Pizza	Chicken	43434.25	1
The Barbecue Chicken Pizza	Chicken	42768	2
The California Chicken Pizza	Chicken	41409.5	3
The Classic Deluxe Pizza	Classic	38180.5	1
The Hawaiian Pizza	Classic	32273.25	2
The Pepperoni Pizza	Classic	30161.75	3
The Spicy Italian Pizza	Supreme	34831.25	1
The Italian Supreme Pizza	Supreme	33476.75	2
The Sicilian Pizza	Supreme	30940.5	3
The Four Cheese Pizza	Veggie	32265.70000000065	1
The Mexicana Pizza	Veggie	26780.75	2
The Five Cheese Pizza	Veggie	26066.5	3

CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

```sql
SELECT
    pt.name AS pizza_name,
    SUM(od.quantity * p.price) AS pizza_revenue,
    ROUND(
        SUM(od.quantity * p.price)
        / SUM(SUM(od.quantity * p.price)) OVER () * 100, 2
    ) AS pct_of_total_revenue
FROM pizza_types pt
JOIN pizzas p
    ON pt.pizza_type_id = p.pizza_type_id
JOIN order_details od
    ON p.pizza_id = od.pizza_id
GROUP BY pt.name
ORDER BY pct_of_total_revenue DESC;
```

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

pizza_name	pizza_revenue	pct_of_total_revenue
The Thai Chicken Pizza	43434.25	5.31
The Barbecue Chicken Pizza	42768	5.23
The California Chicken Pizza	41409.5	5.06
The Classic Deluxe Pizza	38180.5	4.67
The Spicy Italian Pizza	34831.25	4.26
The Southwest Chicken Pizza	34705.75	4.24
The Italian Supreme Pizza	33476.75	4.09

ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```sql
SELECT
    o.order_date AS order_date,
    SUM(od.quantity * p.price) AS daily_revenue,
    SUM(SUM(od.quantity * p.price)) OVER (ORDER BY o.order_date) AS cumulative_revenue
FROM orders o
JOIN order_details od
    ON o.order_id = od.order_id
JOIN pizzas p
    ON od.pizza_id = p.pizza_id
GROUP BY o.order_date
ORDER BY o.order_date;
```

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

order_date	daily_revenue	cumulative_revenue
2015-01-01	2713.8500000000004	2713.8500000000004
2015-01-02	2731.8999999999996	5445.75
2015-01-03	2662.3999999999996	8108.15
2015-01-04	1755.4500000000003	9863.6
2015-01-05	2065.95	11929.55
2015-01-06	2428.95	14358.5
2015-01-07	2202.2000000000003	16560.7
2015-01-08	2838.3499999999995	19399.05
2015-01-09	2127.3500000000004	21526.4

9863.6

KEY BUSINESS INSIGHTS:

- **Peak Ordering Times:**

The busiest hours occur between 12 PM - 1 PM and 5 PM - 6 PM, indicating lunch and dinner rushes.

- **Top Performing Categories:**

Classic pizzas lead in total sales volume, while Chicken pizzas generate higher average revenue per order.

- **Best-Selling Pizza:**

The Thai Chicken Pizza ranks as the highest revenue-generating item on the menu.

- **Weekly Sales Trends:**

Fridays and Saturdays account for approximately 35% of the weekly total revenue, highlighting key days for targeted promotions.

BUSINESS RECOMMENDATIONS:

Recommendations:

- Promote top revenue pizzas more aggressively
- Introduce combo offers during peak hours
- Remove or redesign low-performing pizzas
- Optimize inventory based on size demand

LET'S CONNECT

Neeraj Gupta

Aspiring Data Analyst

🔗 LinkedIn:
www.linkedin.com/in/neeraj-tech

💻 GitHub:
https://github.com/Neeraj007235

📧 Email:
guptaneeraj2811@gmail.com


THANK YOU