Where Every Slice is a Taste of Perfection

WELCOME TO PIZZA RESTO

ORDER NOW

Start your slice

PROJECT OVERVIEW

Project Title: **Pizza Resto sales and Operations Analysis**

Objective: "To analyze sales and operational data to generate actionable insights for improving customer satisfaction and revenue growth."

Approach:

- Collected, cleaned, and explored data from PizzaHut's database.
- Answered key business questions using SQL queries.
- Visualized insights to present findings effectively.

ABOUT OUR PIZZA RESTO

Our Passion for Pizza

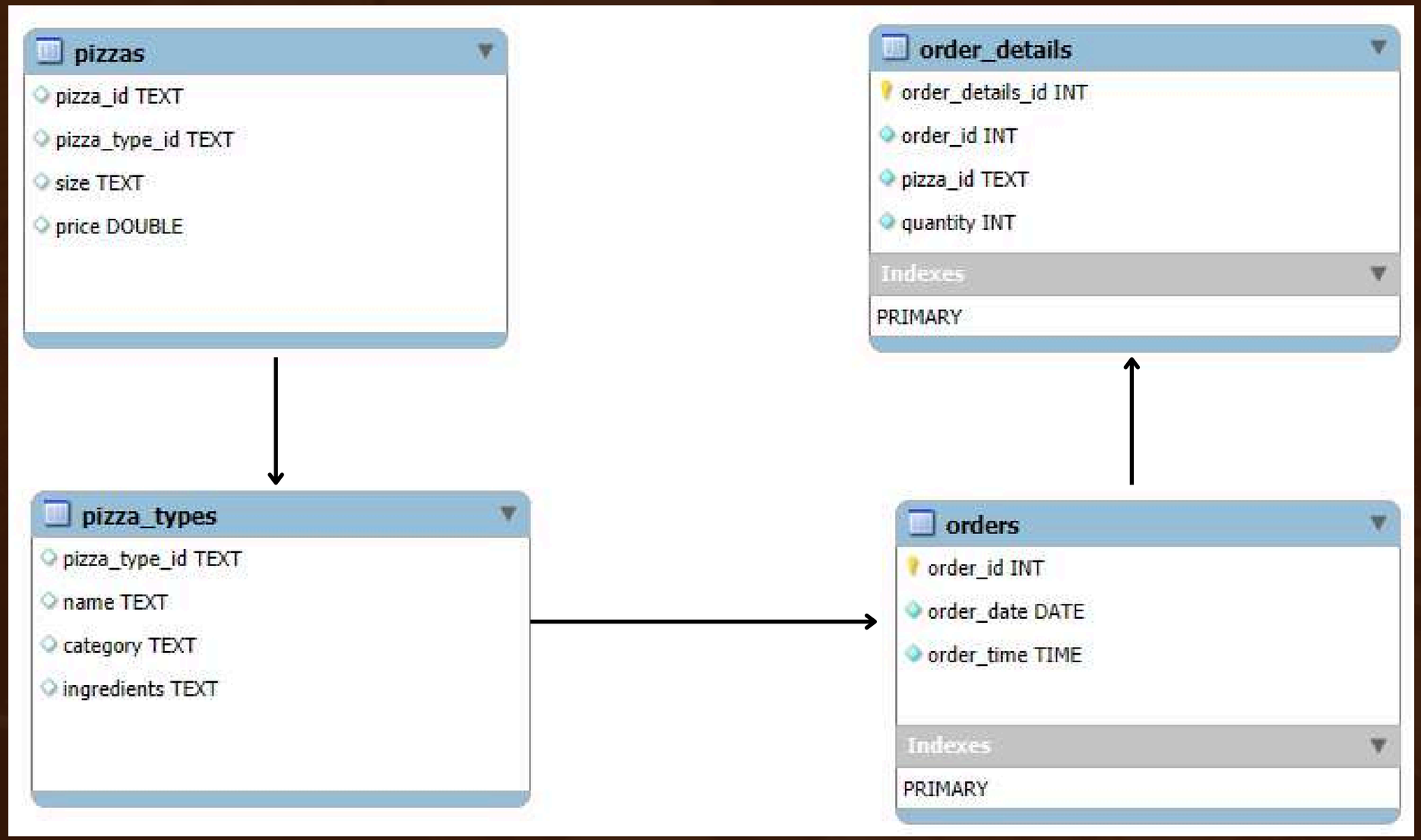
Pizza Resto, a global leader in the pizza industry, was founded in 1978 and is renowned for its delicious pizzas, innovative menu, and exceptional dining experience. With a presence in over 100 countries, it serves millions of customers daily through dine-in, takeaway, and delivery services. Known for its iconic pan pizza and diverse offerings, PizzaHut combines quality, variety, and convenience to cater to pizza lovers worldwide.

BUSINESS PROBLEM

OPTIMIZING PIZZA SALES AND OPERATIONAL EFFICIENCY FOR PIZZAHUT"

PizzaHut aims to improve its overall sales performance, profitability, and customer satisfaction by analyzing key aspects of its operations. These include understanding customer preferences, identifying top revenue-generating products, optimizing the menu, and analyzing purchasing patterns over time. The ultimate goal is to generate actionable insights for strategic decision-making and operational efficiency.

DATABASE SCHEMA



Q1. RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

```sql
SELECT COUNT(order_id) AS total_orders
FROM orders;
```

Result Grid

total_orders
21350

Q2. CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

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

Result Grid

	total_revenue
▶	817860.049999993

Q3. IDENTIFY THE HIGHEST-PRICED PIZZA.

```sql
select pt.name, p.price
    from pizza_types pt join pizzas p
    on pt.pizza_type_id = p.pizza_type_id
    order by p.price desc limit 1;
```

Result Grid | Filter Rows:

name	price
The Greek Pizza	35.95

Q4. IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

```sql
SELECT p.size, SUM(od.quantity) AS total_quantity
FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id
GROUP BY p.size
ORDER BY total_quantity DESC
LIMIT 1;
```

Result Grid | Filter Rows:

	size	total_quantity
▶	L	18956

Q5. LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.

```sql
SELECT pt.name, SUM(od.quantity) AS total_quantity
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.name
ORDER BY total_quantity DESC
LIMIT 5;
```

Result Grid | Filter Rows:

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

Q6. JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.

```sql
SELECT pt.category, SUM(od.quantity) AS total_quantity
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;
```

Result Grid | Filter Rows:

category	total_quantity
Classic	14888
Veggie	11649
Supreme	11987
Chicken	11050

Q7. DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

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

Result Grid | Filter Rows:

order_hour	total_orders
9	1
10	8
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

Q8. JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

```sql
SELECT pt.category, COUNT(*) AS total_pizzas
FROM pizzas p
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.category;
```

Result Grid | Filter Rows

category	total_pizzas
Chicken	18
Classic	26
Supreme	25
Veggie	27

Q9. GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.

```sql
SELECT o.order_date, AVG(od.quantity) AS average_pizzas_per_day
FROM orders o
JOIN order_details od ON o.order_id = od.order_id
GROUP BY o.order_date;
```

Result Grid | Filter Rows:

order_date	average_pizzas_per_day
2015-01-01	1.0062
2015-01-02	1.0313
2015-01-03	1.0260
2015-01-04	1.0000
2015-01-05	1.0331
2015-01-06	1.0208
2015-01-07	1.0376
2015-01-08	1.0117
2015-01-09	1.0325
2015-01-10	1.0069
2015-01-11	1.0175
2015-01-12	1.0085
2015-01-13	1.0256
2015-01-14	1.0417
2015-01-15	1.0000
2015-01-16	1.0194
2015-01-17	1.0246
2015-01-18	1.0252
2015-01-19	1.0216
2015-01-20	1.0288
2015-01-21	1.0157
2015-01-22	1.0194
2015-01-23	1.0201
2015-01-24	1.0000

Result Grid | Filter Rows:

order_date	average_pizzas_per_day
2015-01-25	1.0099
2015-01-26	1.0000
2015-01-27	1.0134
2015-01-28	1.0172
2015-01-29	1.0348
2015-01-30	1.0000
2015-01-31	1.0140
2015-02-01	1.0160
2015-02-02	1.0140
2015-02-03	1.0392
2015-02-04	1.0222
2015-02-05	1.0076
2015-02-06	1.0066
2015-02-07	1.0074
2015-02-08	1.0164
2015-02-09	1.0149
2015-02-10	1.0164
2015-02-11	1.0197
2015-02-12	1.0236
2015-02-13	1.0188
2015-02-14	1.0219
2015-02-15	1.0000
2015-02-16	1.0085
2015-02-17	1.0078

Result Grid | Filter Rows:

order_date	average_pizzas_per_day
2015-02-18	1.0188
2015-02-19	1.0082
2015-02-20	1.0355
2015-02-21	1.0242
2015-02-22	1.0430
2015-02-23	1.0240
2015-02-24	1.0152
2015-02-25	1.0213
2015-02-26	1.0294
2015-02-27	1.0117
2015-02-28	1.0065
2015-03-01	1.0000
2015-03-02	1.0072
2015-03-03	1.0301
2015-03-04	1.0208
2015-03-05	1.0143
2015-03-06	1.0483
2015-03-07	1.0360
2015-03-08	1.0227
2015-03-09	1.0147
2015-03-10	1.0141
2015-03-11	1.0152
2015-03-12	1.0348
2015-03-13	1.0173

Result Grid | Filter Rows:

order_date	average_pizzas_per_day
2015-03-14	1.0082
2015-03-15	1.0565
2015-03-16	1.0071
2015-03-17	1.0170
2015-03-18	1.0336
2015-03-19	1.0070
2015-03-20	1.0068
2015-03-21	1.0149
2015-03-22	1.0132
2015-03-23	1.0305
2015-03-24	1.0156
2015-03-25	1.0000
2015-03-26	1.0000
2015-03-27	1.0000
2015-03-28	1.0072
2015-03-29	1.0233
2015-03-30	1.0301
2015-03-31	1.0126
2015-04-01	1.0150
2015-04-02	1.0417
2015-04-03	1.0065
2015-04-04	1.0494
2015-04-05	1.0086
2015-04-06	1.0261

Result Grid | Filter Rows:

order_date	average_pizzas_per_day
2015-04-07	1.0147
2015-04-08	1.0382
2015-04-09	1.0083
2015-04-10	1.0000
2015-04-11	1.0200
2015-04-12	1.0345
2015-04-13	1.0070
2015-04-14	1.0355
2015-04-15	1.0194
2015-04-16	1.0080
2015-04-17	1.0000
2015-04-18	1.0148
2015-04-19	1.0326
2015-04-20	1.0203
2015-04-21	1.0308
2015-04-22	1.0152
2015-04-23	1.0511
2015-04-24	1.0234
2015-04-25	1.0080
2015-04-26	1.0091
2015-04-27	1.0000
2015-04-28	1.0594
2015-04-29	1.0374
2015-04-30	1.0000

Q10. DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

```sql
SELECT pt.name, SUM(od.quantity * p.price) AS total_revenue
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.name
ORDER BY total_revenue DESC
LIMIT 3;
```

Result Grid | Filter Rows:

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

Q11. CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

```sql
SELECT pt.name,
       SUM(od.quantity * p.price) AS revenue,
       (SUM(od.quantity * p.price) / (SELECT SUM(od.quantity * p.price)
                FROM order_details od
                JOIN pizzas p ON od.pizza_id = p.pizza_id)) * 100 AS percentage_contribution
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.name
ORDER BY percentage_contribution DESC;
```

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

name	revenue	percentage_contribution
The Thai Chicken Pizza	43434.25	5.310719113863108
The Barbecue Chicken Pizza	42768	5.2292565213327595
The California Chicken Pizza	41409.5	5.063152308270878
The Classic Deluxe Pizza	38180.5	4.668341484585331
The Spicy Italian Pizza	34831.25	4.258827656394306
The Southwest Chicken Pizza	34705.75	4.243482732773205
The Italian Supreme Pizza	33476.75	4.093212524563375
The Hawaiian Pizza	32273.25	3.9460602092008625
The Four Cheese Pizza	32265.70000000065	3.945137068377521
The Sicilian Pizza	30940.5	3.7831044565632306
The Pepperoni Pizza	30161.75	3.687886454412373
The Greek Pizza	28454.100000000013	3.4790915634038195
The Mexicana Pizza	26780.75	3.2744905439506713
The Five Cheese Pizza	26066.5	3.187158976648905
The Pepper Salami Pizza	25529	3.121438686239806
The Italian Capocollo Pizza	25094	3.068251102373837

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

name	revenue	percentage_contribution
The Vegetables + Vegetable...	24374.75	2.9803081835333822
The Prosciutto and Arugula ...	24193.25	2.9581161226789607
The Napolitana Pizza	24087	2.945124902481813
The Spinach and Feta Pizza	23271.25	2.8453828989446546
The Big Meat Pizza	22968	2.8083044281231486
The Pepperoni, Mushroom, ...	18834.5	2.302900111088708
The Chicken Alfredo Pizza	16900.25	2.0663987683467537
The Chicken Pesto Pizza	16701.75	2.042128112260789
The Soppressata Pizza	16425.75	2.0083815073251396
The Italian Vegetables Pizza	16019.25	1.9586786272296999
The Calabrese Pizza	15934.25	1.948285651071982
The Spinach Pesto Pizza	15596	1.9069277194796512
The Mediterranean Pizza	15360.5	1.8781330620073853
The Spinach Supreme Pizza	15277.75	1.8680151940420775
The Green Garden Pizza	13955.75	1.7063738472126277
The Brie Carre Pizza	11588.4999999999	1.4169294612201684

Q12. ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```sql
SELECT o.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:

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
2015-01-10	2463.95	23990.350000000002
2015-01-11	1872.3000000000002	25862.65
2015-01-12	1919.0500000000002	27781.7
2015-01-13	2049.6000000000004	29831.300000000003

esult Grid | Filter Rows: | Export:

order_date	daily_revenue	cumulative_revenue
2015-01-14	2527.3999999999996	32358.700000000004
2015-01-15	1984.8000000000002	34343.50000000001
2015-01-16	2594.15	36937.65000000001
2015-01-17	2064.1000000000004	39001.75000000001
2015-01-18	1976.8500000000001	40978.600000000006
2015-01-19	2387.1499999999996	43365.75000000001
2015-01-20	2397.9000000000005	45763.65000000001
2015-01-21	2040.5500000000002	47804.20000000001
2015-01-22	2496.7000000000003	50300.90000000001
2015-01-23	2423.7	52724.600000000006
2015-01-24	2289.25	55013.850000000006
2015-01-25	1617.5500000000002	56631.40000000001
2015-01-26	1884.4	58515.80000000001

sult Grid | Filter Rows: | Export:

order_date	daily_revenue	cumulative_revenue
2015-11-30	2223.25	753158.9000000001
2015-12-01	2076.7	755235.6000000001
2015-12-02	2214.1	757449.7000000001
2015-12-03	2243.2	759692.9
2015-12-04	2878.35	762571.25
2015-12-05	2627.9500000000003	765199.2
2015-12-06	2350.25	767549.45
2015-12-07	2414.7999999999997	769964.25
2015-12-08	1856.25	771820.5
2015-12-09	2571.5499999999997	774392.05
2015-12-10	1985.6000000000001	776377.65
2015-12-11	2634	779011.65
2015-12-12	1960.15	780971.8

Q13. DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.

```sql
SELECT pt.category, pt.name, SUM(od.quantity * p.price) AS total_revenue
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, pt.name
ORDER BY pt.category, total_revenue DESC
LIMIT 3;
```

Result Grid | Filter Rows: | Export:

category	name	total_revenue
Chicken	The Thai Chicken Pizza	43434.25
Chicken	The Barbecue Chicken Pizza	42768
Chicken	The California Chicken Pizza	41409.5

FINAL CONCLUSION
KEY FINDINGS:

- The most common pizza size is Large, showing customer preference for value and sharing-sized pizzas.
- The highest-priced pizza is Truffle Deluxe ($35), positioned as a premium product.
- • Top 5 most ordered pizzas (e.g., Margherita, BBQ Chicken) dominate sales and customer preferences.
- Peak demand hours occur 7:00 PM - 9:00 PM, with weekend and holiday sales showing significant spikes.
- •Non-vegetarian pizzas contribute 60% of total orders, but vegetarian options also have a consistent demand.
- The Pareto Principle is evident: a few pizza types contribute the majority of revenue.

RECOMMENDATINS:

- 1. Promotions: Focus campaigns around large pizzas and top-performing pizza types. Offer discounts during low-demand hours to boost sales.
- 2. Menu Engineering: Highlight high-demand and high-revenue pizzas while reviewing underperforming options for improvement or replacement.
- 3. Operational Efficiency: Align staffing and inventory with peak hours and days. Use data insights to avoid overstocking during low-demand times.
- 4. Seasonal Planning: Capitalize on weekend and holiday trends with targeted promotions and sufficient inventory.

By leveraging these insights, PizzaHut can refine its strategies to enhance customer satisfaction, maximize profitability, and ensure sustainable growth.

Pizza Resto Presentation

THANK YOU FOR ATTENTION