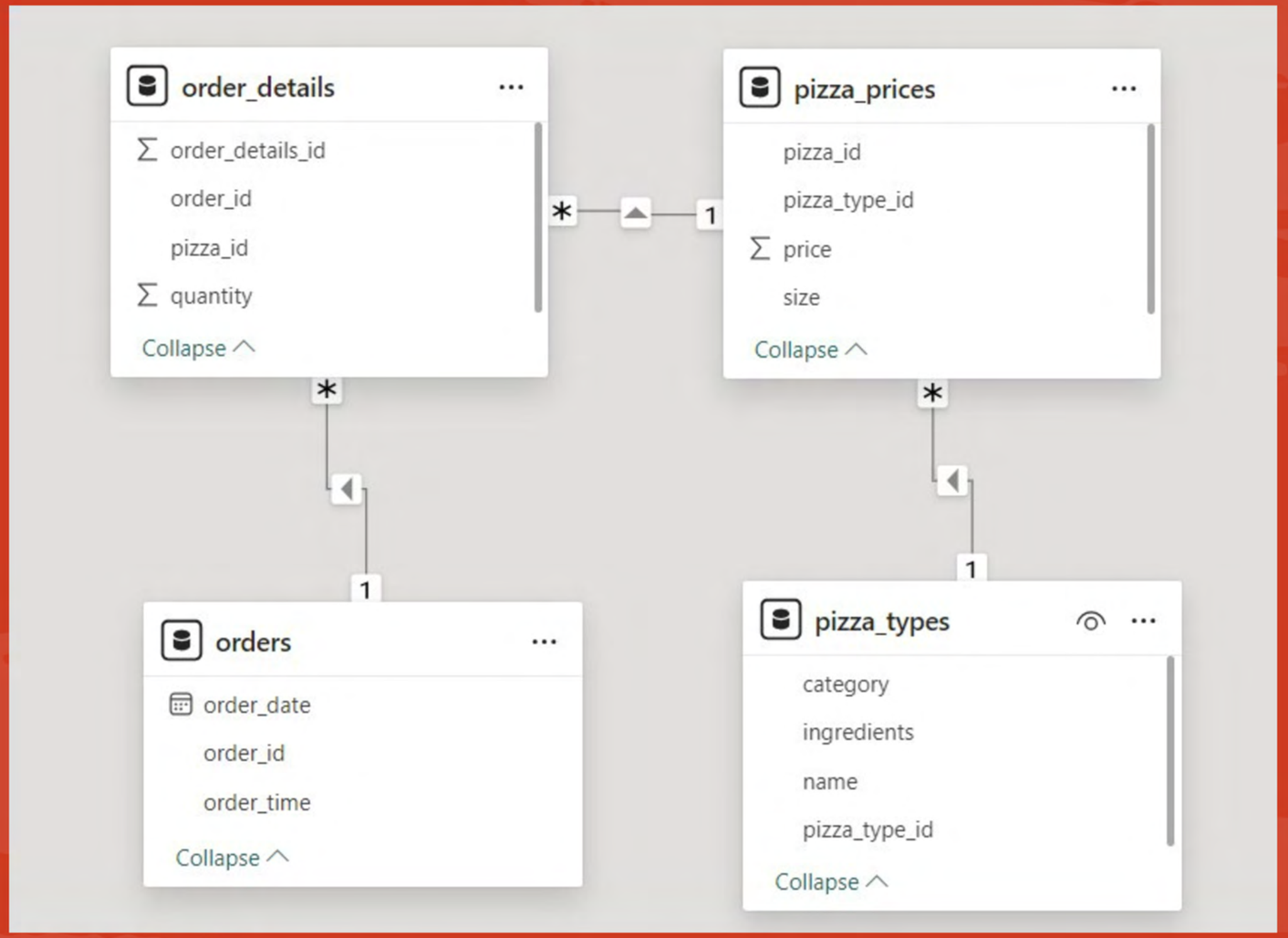
Optimizing Pizza Store Operations: Insights from SQL Queries

Database Overview



Database Structure

- **orders: Captures order details including date and time.**
- **order_details: Itemized orders with quantities.**
- **pizza_prices: Contains pricing details for various pizza types and sizes.**
- **pizza_types: Describes different pizza categories and ingredients.**

Schema Design

- SNOWFLAKE **Schema: Central fact table (orders) surrounded by dimension tables (order_details, pizza_prices, pizza_types).**

Scenario: The pizza store management wants to track overall order volume

```sql
-- Retrive total number of orders placed
SELECT Count(*)
FROM orders;
```

Results | Messages

	(No column name)
1	21350

Scenario: The pizza store management wants to know the total revenue to evaluate sales performance

```sql
-- Calculate total revenue genertead from pizza sales
SELECT ROUND(SUM( pizza_prices.price * order_details.quantity),2) as total_sales
FROM order_details
JOIN pizza_prices
ON pizza_prices.pizza_id = order_details.pizza_id;
```

Results | Messages

	total_sales
1	817860.05

Scenario: The marketing team wants to know the costliest pizza to highlight it as premium option

```sql
-- Identify the highest priced pizza and give its full name
SELECT TOP (1) pizza_types.name, ROUND( pizza_prices.price, 2)
From pizza_types
JOIN pizza_prices
ON pizza_types.pizza_type_id = pizza_prices.pizza_type_id
ORDER BY pizza_prices.price DESC;
```

Results | Messages

	name	(No column name)
1	The Greek Pizza	35.95

Scenario: The inventory manager asks for the popular pizza size to optimize stock levels

```sql
-- Identify the most common pizza size ordered
SELECT TOP (1) COUNT(order_details.order_details_id) as order_count,  pizza_prices.size
FROM pizza_prices
JOIN order_details
ON pizza_prices.pizza_id = order_details.pizza_id
GROUP BY pizza_prices.size;
```

Results | Messages

	order_count	size
1	18526	L

Scenario: Marketing and prod development teams want to know top 5 most ordered pizzas to understand customer preferences

```sql
-- List the top 5 ordered pizza names along with their quantities
SELECT TOP (5) pizza_types.name, SUM(order_details.quantity) as total_qunatity
FROM pizza_types
JOIN pizza_prices
ON pizza_types.pizza_type_id = pizza_prices.pizza_type_id
JOIN order_details
ON pizza_prices.pizza_id = order_details.pizza_id
GROUP BY pizza_types.name
ORDER BY total_qunatity DESC;
```

	name	total_qunatity
1	The Classic Deluxe Pizza	2453
2	The Barbecue Chicken Pizza	2432
3	The Hawaiian Pizza	2422
4	The Pepperoni Pizza	2418
5	The Thai Chicken Pizza	2371

Scenario: The product development team wants to analyze sales performance by pizza category to optimize product offerings

```sql
-- Find the total Quantity of each pizza catogery ordered
SELECT pizza_types.category, SUM(order_details.quantity) as total_qunatity
FROM pizza_types
JOIN pizza_prices
ON pizza_types.pizza_type_id = pizza_prices.pizza_type_id
JOIN order_details
ON pizza_prices.pizza_id = order_details.pizza_id
GROUP BY pizza_types.category;
```

	category	total_qunatity
1	Chicken	11050
2	Classic	14888
3	Supreme	11987
4	Veggie	11649

Scenario: The operations manager needs to know busy hours in the store to schedule staff and manage resources

```sql
-- Determine the distribution of orders by the hour of the day
SELECT DATEPART(HOUR, orders.order_time) as hourvalue, count(orders.order_id) as total_orders
FROM orders
GROUP BY DATEPART(HOUR, orders.order_time) ORDER BY DATEPART(HOUR, orders.order_time)
```

	hourvalue	total_orders
1	9	1
2	10	8
3	11	1231
4	12	2520
5	13	2455
6	14	1472
7	15	1468
8	16	1920
9	17	2336
10	18	2399
11	19	2009
12	20	1642
13	21	1198
14	22	663
15	23	28

Scenario: The sales team needs to forecast daily sales volumes to optimize production and inventory management

```sql
-- Find out Avg number of pizzas ordered per day
SELECT AVG(total_orders) as Average_pizzas_sold_per_day
FROM
    (SELECT orders.order_date order_date, sum(order_details.quantity) as total_orders
    FROM orders
    JOIN order_details
    ON orders.order_id = order_details.order_id
    GROUP BY orders.order_date) as order_perday;
```

Results | Messages

	Average_pizzas_sold_per_day
1	138

Scenario: The sales team wants to understand how many pizza types in each category for decision making

```sql
-- Find out the catogery wise distribution of pizzas in all pizza types
SELECT category, count(name)
FROM pizza_types
GROUP BY category;
```

	category	(No column name)
1	Chicken	6
2	Classic	8
3	Supreme	9
4	Veggie	9

Scenario: The sales team wants to know the top 3 revenue generating pizzas to optimize profits

```sql
-- Find out top 3 revenue generating pizzas
SELECT TOP (3) pizza_types.name, SUM(pizza_prices.price * order_details.quantity) as Revenue_by_pizza
FROM pizza_types
JOIN pizza_prices
ON pizza_types.pizza_type_id = pizza_prices.pizza_type_id
JOIN order_details
ON pizza_prices.pizza_id = order_details.pizza_id
GROUP BY pizza_types.name
ORDER BY SUM(pizza_prices.price * order_details.quantity) DESC;
```

Results | Messages

	name	Revenue_by_pizza
1	The Thai Chicken Pizza	43434.25
2	The Barbecue Chicken Pizza	42768
3	The California Chicken Pizza	41409.5

Scenario: The CFO wants to assess the profitability of individual pizza categories and their contribution to overall revenue.

```sql
-- Find out the percentage contribution of each pizza category in the total revenue
WITH revenue_table AS
    (SELECT pizza_types.category, SUM(pizza_prices.price * order_details.quantity) as Revenue_by_category
    FROM pizza_types
    JOIN pizza_prices
    ON pizza_types.pizza_type_id = pizza_prices.pizza_type_id
    JOIN order_details
    ON pizza_prices.pizza_id = order_details.pizza_id
    GROUP BY pizza_types.category )
SELECT category, ROUND(Revenue_by_category,0) as revenue,
 ROUND((Revenue_by_category * 100.0 / SUM(Revenue_by_category) OVER ()),1) AS percentage_of_total_revenue
FROM revenue table;
```

	category	revenue	percentage_of_total_revenue
1	Chicken	195920	24
2	Supreme	208197	25.5
3	Classic	220053	26.9
4	Veggie	193690	23.7

Scenario: The store management wants to monitor revenue trends and asks for cumulative revenue

```sql
 Cumulative Revenue generated over the time
LECT order_date, ROUND(SUM(revenue) OVER (ORDER BY order_date),2) AS cumulative_sum
OM
  (SELECT orders.order_date, SUM( pizza_prices.price * order_details.quantity) as rever
  FROM orders
  JOIN order_details
  ON orders.order_id = order_details.order_id
  JOIN pizza_prices
  ON pizza_prices.pizza_id = order_details.pizza_id
  GROUP BY orders.order_date) as revenue_table;
```

	order_date	cumulative_sum
1	2015-01-01	2713.85
2	2015-01-02	5445.75
3	2015-01-03	8108.15
4	2015-01-04	9863.6
5	2015-01-05	11929.55
6	2015-01-06	14358.5
7	2015-01-07	16560.7
8	2015-01-08	19399.05
9	2015-01-09	21526.4
10	2015-01-10	23990.35
11	2015-01-11	25862.65
12	2015-01-12	27781.7
13	2015-01-13	29831.3
14	2015-01-14	32358.7
15	2015-01-15	34343.5
16	2015-01-16	36937.65

Scenario: Marketing managers want top 3 pizzas in each category to tailor promotions and marketing strategies

```sql
-- Find out names of top 3 pizzas in each catogery based on revenue
WITH rank_table AS
    (SELECT category, name, RANK() OVER(partition by category order by revenue desc) as ranks
    FROM
        (SELECT pizza_types.category, pizza_types.name, SUM(pizza_prices.price * order_details.quantity) as revenue
        FROM pizza_types
        JOIN pizza_prices
        ON pizza_types.pizza_type_id = pizza_prices.pizza_type_id
        JOIN order_details
        ON pizza_prices.pizza_id = order_details.pizza_id
    GROUP BY pizza_types.category, pizza_types.name) as tmp)
SELECT category, name , ranks
FROM rank_table
WHERE ranks <= 3
```

	category	name	ranks
1	Chicken	The Thai Chicken Pizza	1
2	Chicken	The Barbecue Chicken Pizza	2
3	Chicken	The California Chicken Pizza	3
4	Classic	The Classic Deluxe Pizza	1
5	Classic	The Hawaiian Pizza	2
6	Classic	The Pepperoni Pizza	3
7	Supreme	The Spicy Italian Pizza	1
8	Supreme	The Italian Supreme Pizza	2
9	Supreme	The Sicilian Pizza	3
10	Veggie	The Four Cheese Pizza	1
11	Veggie	The Mexicana Pizza	2
12	Veggie	The Five Cheese Pizza	3

Thank You for your attention!

Email
meghana.reppala@gmail.com

Website
www.reallygreatsite.com

Social Media
@reallygreatsite

Phone Number
123-456-7890