DOMINOS PIZZA

CLASSIC

35.95

SUPREME

23.65

CHIKEN

20.75

VEGGIES

21.00

hardikr727@gmail.com 91+ 8238904191 Ahmedabad

DOMINOS SALES ANALYSIS

THIS PROJECT REVOLVES AROUND ANALYZING PIZZA SALES DATA STORED IN A STRUCTURED DATABASE. THE DATABASE INCLUDES TABLES FOR ORDERS, ORDER DETAILS, PIZZAS, AND PIZZA TYPES. BELOW ARE THE KEY OBJECTIVES AND INSIGHTS GENERATED THROUGH SQL QUERIES:

DATABASE DESIGN AND TABLES:

1. DATABASE NAME: DOMINOS

```
CREATE DATABASE Domenis;
SELECT * FROM  Dominos.pizzas;
```

KEY TABLES CREATED:

- ORDERS: STORES METADATA FOR ORDERS INCLUDING ORDER ID, DATE, AND TIME.

```
CREATE TABLE orders (
    order_id INT NOT NULL,
    order_date DATE NOT NULL,
    order_time TIME NOT NULL,
    PRIMARY KEY (order_id)
);
```

- ORDER_DETAILS: CONTAINS DETAILS OF EACH ORDER, SUCH AS PIZZA ID, QUANTITY, AND A UNIQUE ORDER DETAIL ID.

```
CREATE TABLE order_details (
    order_details_id INT NOT NULL,
    order_id INT NOT NULL,
    pizza_id TEXT NOT NULL,
    quantity INT NOT NULL,
    PRIMARY KEY (order_details_id)
);
```

- PIZZAS: LISTS PIZZA ATTRIBUTES INCLUDING PRICE, SIZE, AND THEIR CORRESPONDING TYPES.

- PIZZA_TYPES: CATEGORIZES PIZZAS INTO DIFFERENT TYPES AND CATEGORIES, SUCH AS VEGETARIAN OR NON-VEGETARIAN.

KEY ANALYSES AND QUERIES:

1. **TOTAL ORDERS:** COUNTED THE TOTAL NUMBER OF ORDERS PLACED (COUNT(ORDER_ID)).

```sql
SELECT
    *
FROM
    domenis.orders;
SELECT
    COUNT(order_id)
FROM
    domenis.orders;
```

Result Grid | Filter Rows:

COUNT(order_id)
21350

2. **TOTAL REVENUE:** CALCULATED REVENUE FROM PIZZA SALES USING QUANTITIES AND PRICES (SUM(QUANTITY * PRICE)).

```sql
SELECT
    SUM(order_details.quantity * pizzas.price) AS total_sales
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id;
```

Result Grid | Filt

total_sales
817860.049999993

3. **HIGHEST-PRICED PIZZA:** IDENTIFIED THE MOST EXPENSIVE PIZZA IN THE DATASET.

```sql
SELECT
    pizzas.size,
    COUNT(order_details.order_details_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
LIMIT 5;
```

Result Grid | Filter Rows:

size	order_Count
L	18526
M	15385
S	14137
XL	544
XXL	28

4. POPULAR PIZZA SIZES: FOUND THE MOST COMMONLY ORDERED PIZZA SIZES.

```sql
SELECT
    pizzas.size,
    COUNT(order_details.order_details_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
LIMIT 5;
```

Result Grid | Filter Rows:

size	order_Count
L	18526
M	15385
S	14137
XL	544
XXL	28

5. TOP 5 PIZZA TYPES BY QUANTITY: RANKED THE MOST FREQUENTLY ORDERED PIZZA TYPES WITH THEIR QUANTITIES.

```sql
SELECT
    pizza_types.name,
    SUM(order_details.quantity) AS high_quantity
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.name
ORDER BY high_quantity DESC
LIMIT 5;
```

Result Grid | Filter Rows:

name	high_quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

6. PIZZA CATEGORY-WISE ORDERS: ANALYZED TOTAL QUANTITIES OF PIZZAS ORDERED BY CATEGORY.

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

Result Grid | Filter Rows:

category	high_quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

7. ORDER DISTRIBUTION BY HOUR: EXPLORED ORDER PATTERNS BY HOUR OF THE DAY.

```sql
SELECT
    HOUR(order_time) AS hour, COUNT(order_id) AS order_count
FROM
    orders
GROUP BY hour
```

Result Grid | Filter Rows:

hour	order_count
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

8. CATEGORY-WISE DISTRIBUTION: INVESTIGATED THE DISTRIBUTION OF PIZZAS BY CATEGORY

.

```sql
SELECT
    pizza_types.category AS category,
    COUNT(pizza_types.name) AS name
FROm
    pizza_types
GROUP BY category
ORDER BY name DESC;
```

Result Grid | Filter Rov

category	name
Supreme	9
Veggie	9
Classic	8
Chicken	6

9. DAILY AVERAGES: GROUPED ORDERS BY DATE AND COMPUTED THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.

```sql
SELECT
    AVG(quantity)
FROM
    (SELECT
        orders.order_time AS hour,
            SUM(order_details.quantity) AS quantity
    FROM
        orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY hour) AS order_quantity;
```

Result Grid | Filter

AVG(quantity)
3.0261

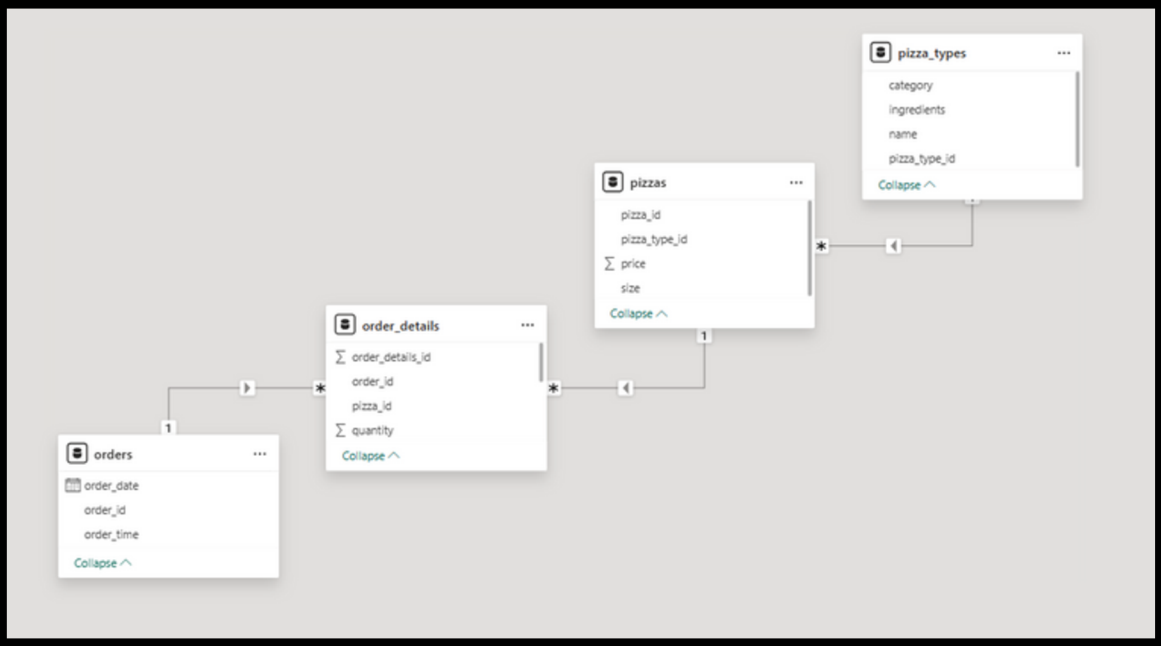
10. TOP 3 PIZZA TYPES BY REVENUE: RANKED PIZZA TYPES GENERATING THE HIGHEST REVENUE.

```sql
SELECT
    pizza_types.name AS name,
    ROUND(SUM(order_details.quantity * pizzas.price),
            2) AS quantity
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 name
ORDER BY quantity DESC
LIMIT 3;
```

Result Grid | Filter Rows:

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

INTER-CONNECTIONS IN THE DOMINOS SALES DATABASE SCHEMA



Summary of Relationships:

Parent Table	Child Table	Relationship Type	Primary Key	Foreign Key
`orders`	`order_details`	One-to-Many	`order_id`	`order_id`
`pizzas`	`order_details`	One-to-Many	`pizza_id`	`pizza_id`
`pizza_types`	`pizzas`	One-to-Many	`pizza_type_id`	`pizza_type_id`

Why This Structure?

- **Normalization**: Reduces data redundancy by breaking it into separate tables.
- **Query Efficiency**: Helps in generating insights like total revenue, popular pizzas, and order trends.
- **Scalability**: Can accommodate more data without duplication.

BUSINESS IMPLICATIONS

INVENTORY AND RESOURCE MANAGEMENT: KNOWING THE BEST-SELLING PIZZAS AND PEAK ORDER TIMES CAN OPTIMIZE STOCK LEVELS AND REDUCE WASTE.

MENU OPTIMIZATION: POPULAR PIZZA TYPES CAN BE PROMOTED FURTHER, WHILE UNDERPERFORMING ONES CAN BE RECONSIDERED.

REVENUE MAXIMIZATION: FOCUSING ON HIGH-REVENUE PIZZAS AND CUSTOMER PREFERENCES CAN IMPROVE PROFITABILITY.

OPERATIONAL EFFICIENCY: UNDERSTANDING ORDER PATTERNS ALLOWS FOR BETTER STAFF SCHEDULING AND KITCHEN WORKFLOW IMPROVEMENTS.

FINAL THOUGHTS

THE ANALYSIS PROVIDES A DATA-DRIVEN APPROACH TO ENHANCING BUSINESS OPERATIONS, MARKETING STRATEGIES, AND OVERALL SALES PERFORMANCE. BY LEVERAGING THESE INSIGHTS, A PIZZA BUSINESS CAN IMPROVE EFFICIENCY, CUSTOMER SATISFACTION, AND PROFITABILITY.