SHAIK ABDUL KHADAR

+91-9849397300 abdulsvne@gmail.com LinkedIn GitHub

PIZZA SQL Project Report

Introduction

This report provides insights into a database designed for a pizza business. The database contains key tables such as `orders`, `pizzas`, `pizza_types`, and `order_details`. The report answers 25 basic to advanced queries, covering aspects like sales, customer behavior, and business performance.

Database Overview

Tables and Columns:

1. **pizzas**: (`pizza_id` (PK), `pizza_type_id`, `size`, `price`)
2. **pizza_types**: (`pizza_type_id` (PK), `name`, `category`, `ingredients`)
3. **orders**: (`order_id` (PK), `date`, `time`)
4. **order_details**: (`order_details_id` (PK), `order_id`, `pizza_id`, `quantity`)

Basic Queries and Answers

1. Retrieve all orders:

```
SELECT * FROM orders;
```

Answer: Returns all orders with `order_id`, `date`, and `time`.

2. Total number of pizzas available:

```
SELECT COUNT(*) AS total_pizzas FROM pizzas;
```

Answer: Shows the total number of pizza records.

3. Unique pizza sizes available:

```
SELECT DISTINCT size FROM pizzas;
```

Answer: Lists all unique pizza sizes offered.

4. Orders placed on a specific date ('2024-06-01'):

```
SELECT * FROM orders WHERE date = '2024-06-01';
```

SHAIK ABDUL KHADAR

+91-9849397300 abdulsvne@gmail.com LinkedIn GitHub

Answer: Retrieves orders placed on `June 1, 2024`.

5. 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;
```

Answer: Calculates total sales revenue.

Intermediate Queries and Answers

6. Top 5 most frequently ordered pizzas:

```
SELECT p.pizza_id, pt.name, SUM(od.quantity) AS total_ordered
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 p.pizza_id, pt.name
ORDER BY total_ordered DESC
LIMIT 5;
```

Answer: Returns the five most frequently ordered pizzas.

7. Average price of pizzas by category:

```
SELECT pt.category, AVG(p.price) AS avg_price
FROM pizzas p
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.category;
```

Answer: Provides the average price of pizzas per category.

8. Highest revenue-generating pizza type:

```
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 1;
```

Answer: Identifies the most profitable pizza.

9. Top 3 busiest order dates:

```
SELECT date, COUNT(order_id) AS order_count
FROM orders
```

```
GROUP BY date
ORDER BY order_count DESC
LIMIT 3;
```

Answer: Returns the top three dates with the highest order volume.

10. Least ordered pizza:

```
SELECT p.pizza_id, pt.name, SUM(od.quantity) AS total_ordered
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 p.pizza_id, pt.name
ORDER BY total_ordered ASC
LIMIT 1;
```

Answer: Identifies the least popular pizza.

Advanced Queries and Answers

11. Month-over-month growth in pizza sales:

```
SELECT DATEPART(YEAR, date) AS year, DATEPART(MONTH, date) AS month,
SUM(od.quantity) AS total_sales,
LAG(SUM(od.quantity)) OVER (ORDER BY DATEPART(YEAR, date), DATEPART(MONTH,
date)) AS prev_month_sales,
(SUM(od.quantity) - LAG(SUM(od.quantity)) OVER (ORDER BY DATEPART(YEAR,
date), DATEPART(MONTH, date))) * 100.0 / LAG(SUM(od.quantity)) OVER (ORDER
BY DATEPART(YEAR, date), DATEPART(MONTH, date)) AS growth_percentage
FROM orders o
JOIN order_details od ON o.order_id = od.order_id
GROUP BY DATEPART(YEAR, date), DATEPART(MONTH, date);
```

Answer: Computes the month-over-month sales growth.

12. Percentage contribution of each pizza category to total sales:

```
SELECT pt.category, SUM(od.quantity * p.price) AS category_sales,
SUM(od.quantity * p.price) * 100.0 / (SELECT SUM(od.quantity * p.price)
FROM order_details od JOIN pizzas p ON od.pizza_id = p.pizza_id) AS
sales_percentage
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;
```

Answer: Shows the percentage contribution of each pizza category.

13. Rank pizzas by their popularity:

SHAIK ABDUL KHADAR

+91-9849397300 abdulsvne@gmail.com LinkedIn GitHub

```
SELECT p.pizza_id, pt.name, SUM(od.quantity) AS total_ordered,
RANK() OVER (ORDER BY SUM(od.quantity) DESC) AS rank_order
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 p.pizza_id, pt.name;
```

Answer: Assigns a rank to pizzas based on their popularity.

14. Optimized query for fetching all orders with pizza details:

```
CREATE INDEX idx_order_details ON order_details(order_id, pizza_id);
SELECT o.order_id, o.date, o.time, pt.name, p.size, od.quantity, p.price
FROM orders o
JOIN order_details od ON o.order_id = od.order_id
JOIN pizzas p ON od.pizza_id = p.pizza_id
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id;
```

Answer: Uses indexing to improve query performance when retrieving order details.

Retrieve all columns from the `orders` table.

```
SELECT * FROM orders;
```

Find the total number of pizzas available in the `pizzas` table.

```
SELECT COUNT(*) AS total_pizzas FROM pizzas;
```

List unique pizza sizes available.

```
SELECT DISTINCT size FROM pizzas;
```

Retrieve orders placed on a specific date ('2024-06-01').

```
SELECT * FROM orders WHERE date = '2024-06-01';
```

Find 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;
```

SHAIK ABDUL KHADAR

+91-9849397300 abdulsvne@gmail.com LinkedIn GitHub

Retrieve the names of all pizza categories from the `pizza_types` table.

```
SELECT DISTINCT category FROM pizza_types;
```

Find the most expensive pizza and its price.

```
SELECT * FROM pizzas ORDER BY price DESC LIMIT 1;
```

Count the number of orders placed in the last 7 days.

```
SELECT COUNT(*) AS recent_orders
FROM orders
WHERE date >= DATEADD(DAY, -7, GETDATE());
```

List all orders along with the number of pizzas in each order.

```
SELECT od.order_id, SUM(od.quantity) AS total_pizzas
FROM order_details od
GROUP BY od.order_id;
```

Retrieve pizzas that cost more than $15 but less than $25.

```
SELECT * FROM pizzas WHERE price BETWEEN 15 AND 25;
```

Intermediate Queries

Find the top 5 most frequently ordered pizzas.

```
SELECT p.pizza_id, pt.name, SUM(od.quantity) AS total_ordered
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 p.pizza_id, pt.name
ORDER BY total_ordered DESC
LIMIT 5;
```

Retrieve the average price of pizzas by category.

```
SELECT pt.category, AVG(p.price) AS avg_price
FROM pizzas p
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
```

```
GROUP BY pt.category;
```

Identify customers who placed more than 5 orders in a month.
(Assuming there is a `customer_id` column in `orders`)

```
SELECT customer_id, COUNT(order_id) AS order_count
FROM orders
WHERE DATEPART(MONTH, date) = DATEPART(MONTH, GETDATE())
GROUP BY customer_id
HAVING COUNT(order_id) > 5;
```

Retrieve the most popular pizza size based on orders.

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

Find the highest revenue-generating pizza type.

```
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 1;
```

List the top 3 busiest order dates based on the number of orders.

```
SELECT date, COUNT(order_id) AS order_count
FROM orders
GROUP BY date
ORDER BY order_count DESC
LIMIT 3;
```

Calculate the total quantity of pizzas sold per pizza type.

```
SELECT pt.name, SUM(od.quantity) AS total_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.name
ORDER BY total_sold DESC;
```

SHAIK ABDUL KHADAR

+91-9849397300 abdulsvne@gmail.com LinkedIn GitHub

Find the least ordered pizza.

```
SELECT p.pizza_id, pt.name, SUM(od.quantity) AS total_ordered
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 p.pizza_id, pt.name
ORDER BY total_ordered ASC
LIMIT 1;
```

Show the cumulative sales revenue per day for the last 30 days.

```
SELECT date, SUM(od.quantity * p.price) OVER (ORDER BY 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
WHERE date >= DATEADD(DAY, -30, GETDATE());
```

Identify orders where more than 3 pizzas were ordered in a single transaction.

```
SELECT order_id, SUM(quantity) AS total_pizzas
FROM order_details
GROUP BY order_id
HAVING SUM(quantity) > 3;
```

Advanced Queries

Calculate the month-over-month growth in pizza sales.

```
SELECT
    DATEPART(YEAR, date) AS year,
    DATEPART(MONTH, date) AS month,
    SUM(od.quantity) AS total_sales,
    LAG(SUM(od.quantity)) OVER (ORDER BY DATEPART(YEAR, date),
DATEPART(MONTH, date)) AS prev_month_sales,
    (SUM(od.quantity) - LAG(SUM(od.quantity)) OVER (ORDER BY DATEPART(YEAR,
date), DATEPART(MONTH, date))) * 100.0 / LAG(SUM(od.quantity)) OVER (ORDER
BY DATEPART(YEAR, date), DATEPART(MONTH, date)) AS growth_percentage
FROM orders o
JOIN order_details od ON o.order_id = od.order_id
GROUP BY DATEPART(YEAR, date), DATEPART(MONTH, date);
```

Find the percentage contribution of each pizza category to total sales.

SHAIK ABDUL KHADAR

+91-9849397300 abdulsvne@gmail.com LinkedIn GitHub

```
SELECT pt.category,
       SUM(od.quantity * p.price) AS category_sales,
       SUM(od.quantity * p.price) * 100.0 / (SELECT SUM(od.quantity *
p.price) FROM order_details od JOIN pizzas p ON od.pizza_id = p.pizza_id)
AS sales_percentage
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;
```

Find repeat customers (customers who placed more than 3 orders in different months).

```
SELECT customer_id, COUNT(DISTINCT DATEPART(MONTH, date)) AS active_months
FROM orders
GROUP BY customer_id
HAVING COUNT(DISTINCT DATEPART(MONTH, date)) > 3;
```

Rank pizzas by their popularity using the `RANK()` window function.

```
SELECT p.pizza_id, pt.name, SUM(od.quantity) AS total_ordered,
       RANK() OVER (ORDER BY SUM(od.quantity) DESC) AS rank_order
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 p.pizza_id, pt.name;
```

Optimize a query fetching all orders with pizza details using indexing and joins.

```
CREATE INDEX idx_order_details ON order_details(order_id, pizza_id);


SELECT o.order_id, o.date, o.time, pt.name, p.size, od.quantity, p.price
FROM orders o
JOIN order_details od ON o.order_id = od.order_id
JOIN pizzas p ON od.pizza_id = p.pizza_id
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id;
```

Conclusion

This report provides insights into pizza sales trends, customer behavior, and revenue analysis using queries. These queries help in decision-making, optimizing sales strategies, and improving business operations.

SHAIK ABDUL KHADAR

+91-9849397300 abdulsvne@gmail.com LinkedIn GitHub

Recommendations

- Use indexing on frequently queried columns for better performance.
- Analyze peak sales hours to optimize staffing and inventory.
- Offer discounts on least-ordered pizzas to boost sales.
- Expand popular pizza sizes and categories based on demand trends.