### **Mini Project: Pizza Sales Analysis with SQL**

**Problem Statement:**This SQL project involves analyzing a pizza sales dataset to gain insights about sales patterns, order distributions, and revenue. The dataset contains details about customer orders, pizzas, their categories, and prices. Students will be required to write SQL queries to extract and analyze data based on a series of progressively challenging questions.

**Dataset Link:**[Pizza Sales Dataset](https://drive.google.com/file/d/1wfqpxZyATIZTZFnG_WWEXUBl9BIa08GD/view?usp=sharing)

### **Guidelines for Students:**

1. **Data Understanding:**
   * Understand the structure of the dataset by inspecting the tables and their relationships.
   * Familiarize yourself with the schema, particularly the pizza categories, order details, and sales records.
2. **Data Exploration:**
   * Analyze the dataset by writing queries to retrieve basic information such as the total number of orders, revenue, and frequently ordered items.
3. **Advanced Analysis:**
   * Perform more complex queries involving joins and groupings to calculate metrics like revenue distribution, pizza category sales, and cumulative sales over time.
4. **Optimization and Interpretation:**
   * Ensure that your queries are optimized for performance (e.g., using GROUP BY, JOIN operations, and HAVING clauses).
   * Interpret the results of each query to understand trends and patterns.

### **Project Questions:**

#### **Basic:**

1. **Retrieve the total number of orders placed.** **Objective:** Understand the total volume of orders.

| SELECT COUNT(DISTINCT(order\_id)) AS count\_orders  FROM orders; |
| --- |
|  |

1. **Calculate the total revenue generated from pizza sales.** **Objective:** Calculate the total revenue generated from all pizza orders.

| # Table/s -> pizzas, order\_details  SELECT SUM(od.quantity \* p.price) AS Total\_Revenue  FROM order\_details AS od  JOIN pizzas AS p  ON od.pizza\_id = p.pizza\_id;  SELECT ROUND(SUM(od.quantity \* p.price), 3) AS Total\_Revenue  FROM order\_details AS od  JOIN pizzas AS p  ON od.pizza\_id = p.pizza\_id;  SELECT CEIL(SUM(od.quantity \* p.price)) AS Total\_Revenue  FROM order\_details AS od  JOIN pizzas AS p  ON od.pizza\_id = p.pizza\_id;  SELECT FLOOR(SUM(od.quantity \* p.price)) AS Total\_Revenue  FROM order\_details AS od  JOIN pizzas AS p  ON od.pizza\_id = p.pizza\_id; |
| --- |
|  |

1. **Identify the highest-priced pizza.** **Objective:** Find out which pizza is the most expensive.

| # Table/s -> pizzas, pizza\_types  SELECT \* FROM pizza\_types;  SELECT \* FROM pizzas;  SELECT \* FROM order\_details;  SELECT \* FROM orders;  SELECT pt.name, p.price  FROM pizza\_types AS pt  JOIN pizzas AS p  ON pt.pizza\_type\_id = p.pizza\_type\_id  ORDER BY p.price DESC  LIMIT 1;  SELECT pt.name  FROM pizza\_types AS pt  WHERE pt.pizza\_type\_id = (  SELECT p1.pizza\_type\_id  FROM pizzas AS p1  WHERE p1.price = (  SELECT MAX(p2.price)  FROM pizzas AS p2  )  ); |
| --- |
|  |

1. **Identify the most common pizza size ordered.** **Objective:** Determine which pizza size (e.g., small, medium, large) is ordered the most.

| # Table/s -> pizzas, order\_details  SELECT \* FROM pizzas;  SELECT \* FROM order\_details;  SELECT size  FROM (  SELECT p.size, COUNT(od.order\_details\_id) AS Number\_of\_Orders  FROM pizzas AS p  JOIN order\_details AS od  ON p.pizza\_id = od.pizza\_id  GROUP BY p.size  ORDER BY Number\_of\_Orders DESC  ) AS a  LIMIT 1; |
| --- |
|  |

1. **List the top 5 most ordered pizza types along with their quantities.**

**Objective:** Find out which pizza types are most frequently ordered.

| # Table/s -> pizza\_types, order\_details, pizzas  SELECT p.pizza\_type\_id, SUM(od.quantity) AS Total\_Quantity  FROM pizzas AS p  JOIN Order\_Details AS od  ON od.pizza\_id = p.pizza\_id  GROUP BY p.pizza\_type\_id  ORDER BY Total\_Quantity DESC  LIMIT 5;  /\*  Joins cannot be removed in this case becausein the SELECT clause,  we are taking columns from 2 different tables.  \*/ |
| --- |
|  |

#### **Intermediate:**

1. **Join the necessary tables to find the total quantity of each pizza category ordered.** **Objective:** Explore the relationship between pizza categories and quantities ordered.

| # Table/s -> pizza\_types, order\_details, pizzas  SELECT pt.category, SUM(od.quantity) AS Total\_Quantity  FROM pizza\_types AS pt  JOIN pizzas AS p  ON p.pizza\_type\_id = pt.pizza\_type\_id  JOIN Order\_Details AS od  ON od.pizza\_id = p.pizza\_id  GROUP BY pt.category  ORDER BY Total\_Quantity DESC; |
| --- |
|  |

1. **Determine the distribution of orders by hour of the day.** **Objective:** Analyze how orders are distributed across different times of day.

| # Table/s -> orders  SELECT HOUR(time) AS Hour\_, COUNT(order\_id) AS Number\_of\_Orders  FROM Orders  GROUP BY Hour\_  ORDER BY Hour\_; |
| --- |
|  |

1. **Join relevant tables to find the category-wise distribution of pizzas.** **Objective:** Find out how pizzas from different categories are ordered.

| # Table/s -> pizza\_types, pizza, order\_details  SELECT \* FROM order\_details;  SELECT \* FROM pizza\_types;  SELECT \* FROM pizzas;  SELECT pt.category, COUNT(od.order\_id) AS Number\_of\_Orders  FROM pizza\_types AS pt  JOIN pizzas AS p  ON pt.pizza\_type\_id = p.pizza\_type\_id  JOIN order\_details AS od  ON od.pizza\_id = p.pizza\_id  GROUP BY pt.category  ORDER BY Number\_of\_Orders DESC; |
| --- |
|  |

1. **Group the orders by date and calculate the average number of pizzas ordered per day.** **Objective:** Analyze daily order trends and average quantities.

| # Table/s -> Orders, Order\_Details  SELECT FLOOR(AVG(Quantity)) AS Avg\_Pizzas\_Ordered\_Per\_Day  FROM (  SELECT o.date, SUM(od.quantity) AS Quantity  FROM Orders AS o  JOIN Order\_Details AS od  ON o.order\_id = od.order\_id  GROUP BY o.date  ) AS a; |
| --- |
|  |

1. **Determine the top 3 most ordered pizza types based on revenue.** **Objective:** Identify the pizza types that generated the most revenue.

| # Table/s -> pizza\_types, order\_details, pizza  SELECT pt.name, FLOOR(SUM(od.quantity \* p.price)) AS Total\_Revenue  FROM pizza\_types AS pt  JOIN pizzas AS p  ON p.pizza\_type\_id = pt.pizza\_type\_id  JOIN order\_details AS od  ON od.pizza\_id = p.pizza\_id  GROUP BY pt.name  ORDER BY Total\_Revenue DESC  LIMIT 3; |
| --- |
|  |

#### **Advanced:**

1. **Calculate the percentage contribution of each pizza type to total revenue.** **Objective:** Understand each pizza's contribution to overall sales.

| # Table/s -> order\_details, pizzas, pizza\_types  SELECT pt.category,  ROUND((SUM(od1.quantity \* p1.price) / (  SELECT SUM(od2.quantity \* p2.price)  FROM order\_details AS od2  JOIN pizzas AS p2  ON od2.pizza\_id = p2.pizza\_id)) \* 100, 2) AS Proportion  FROM order\_details AS od1  JOIN pizzas AS p1  ON od1.pizza\_id = p1.pizza\_id  JOIN pizza\_types AS pt  ON p1.pizza\_type\_id = pt.pizza\_type\_id  GROUP BY pt.category; |
| --- |
|  |

1. **Analyze the cumulative revenue generated over time.** **Objective:** Track how revenue accumulates over time.

| # Table/s -> order\_details, pizzas, orders  SELECT date, FLOOR(Revenue) AS Revenue,  FLOOR(SUM(Revenue) OVER (ORDER BY date)) AS Cum\_Revenue  FROM (  SELECT o.date,  SUM(od.quantity \* p.price) AS Revenue  FROM Order\_Details AS od  JOIN Pizzas AS p  ON p.pizza\_id = od.pizza\_id  JOIN Orders AS o  ON o.order\_id = od.order\_id  GROUP BY o.date  ) AS s; |
| --- |
|  |

1. **Determine the top 3 most ordered pizza types based on revenue for each pizza category.** **Objective:** Find the highest-grossing pizzas within each category.

| SELECT name, Revenue  FROM (  SELECT category, name, revenue,  RANK() OVER (PARTITION BY category ORDER BY Revenue DESC) AS rank\_  FROM (  SELECT pt.category, pt.name,  SUM(od.quantity \* p.price) AS Revenue  FROM pizza\_types AS pt  JOIN pizzas AS p  ON pt.pizza\_type\_id = p.pizza\_type\_id  JOIN order\_details AS od  ON od.pizza\_id = p.pizza\_id  GROUP BY pt.category, pt.name  ) AS a  ) AS b  WHERE rank\_ <= 3; |
| --- |
|  |

### **Expected Outcomes:**

* **Basic:** Students will understand how to perform fundamental SQL queries to analyze pizza sales data, including aggregation and filtering.
* **Intermediate:** Students will be able to join tables, group data, and calculate average values or distributions.
* **Advanced:** Students will gain expertise in advanced SQL concepts such as percentage contribution, cumulative analysis, and partitioned ranking.