**Danny’s Diner SQL Analysis**

**Project Overview: Danny’s Diner SQL Analysis**

**Project Name:**

**Danny’s Diner SQL Analysis**

**Project Description:**

**This project involves analyzing sales data from Danny’s Diner to answer specific business questions using SQL queries. The analysis covers various aspects such as customer spending, visit frequency, popular items, and membership benefits.**

**Objectives:**

1. **Understand Customer Behavior: Analyze how much each customer spends, how often they visit, and what items they purchase.**
2. **Identify Popular Items: Determine the most popular items on the menu and the most frequently purchased items by each customer.**
3. **Evaluate Membership Impact: Assess the impact of the membership program on customer purchases and calculate points earned by members.**
4. **Generate Insights: Provide actionable insights that can help Danny’s Diner improve its menu offerings, customer engagement, and membership program.**

**Key Components:**

1. **Database and Table Creation:**
   * **Create a database named dannys\_diner.**
   * **Create tables for sales, menu, and members.**
   * **Insert sample data into these tables.**

**SQL Queries:**

* **Total Amount Spent: Calculate the total amount each customer spent.**
* **Days Visited: Count the number of days each customer visited the restaurant.**
* **First Item Purchased: Identify the first item purchased by each customer.**
* **Most Purchased Item: Determine the most purchased item on the menu.**
* **Popular Item for Each Customer: Find the most popular item for each customer.**
* **First Purchase After Membership: Identify the first item purchased after becoming a member.**
* **Last Purchase Before Membership: Find the last item purchased before becoming a member.**
* **Total Items and Amount Spent Before Membership: Calculate the total items and amount spent before becoming a member.**
* **Points Calculation:** Calculate points earned by customers based on their purchases.
* **Points Calculation for First Week After Membership:** Calculate points for the first week after membership.
* **Recreate Table Output:** Recreate the table output with membership status.
* **Ranking Items:** Rank items purchased by each customer after becoming a member.

**Expected Outcomes:**

* **Customer Spending Patterns: Insights into how much customers spend and their purchasing behavior.**
* **Menu Popularity: Identification of the most and least popular items on the menu.**
* **Membership Benefits: Understanding the impact of the membership program on customer purchases and loyalty.**
* **Actionable Insights: Recommendations for improving menu offerings, customer engagement, and membership benefits.**

**Tools and Technologies:**

* **SQL: For querying and analyzing the data.**
* **Database Management System (DBMS): To create and manage the database and tables.**

**Project Timeline:**

* **Week 1: Database and table creation, data insertion.**
* **Week 2: Writing and testing SQL queries.**
* **Week 3: Analyzing results and generating insights.**
* **Week 4: Compiling the final report and recommendations.**

**Team Members and Roles:**

* **Project Manager: Oversees the project and ensures timely completion.**
* **Data Analyst: Writes and tests SQL queries, analyzes data.**
* **Database Administrator: Manages the database and ensures data integrity.**
* **Report Writer: Compiles the final report and recommendations.**

**This project aims to provide a comprehensive analysis of Danny’s Diner’s sales data, offering valuable insights that can help improve business operations and customer satisfaction.**

**1. Database and Table Creation**

We start by creating a database named dannys\_diner and then switch to using it:

CREATE DATABASE dannys\_diner;

USE dannys\_diner;

Next, we create three tables: sales, menu, and members. The sales table records each sale with the customer ID, order date, and product ID. The menu table lists the products available with their prices. The members table keeps track of customers who have joined the membership program along with their join dates.

CREATE TABLE sales(

customer\_id VARCHAR(1),

order\_date DATE,

product\_id INTEGER

);

INSERT INTO sales

(customer\_id, order\_date, product\_id)

VALUES

('A', '2021-01-01', 1),

('A', '2021-01-01', 2),

('A', '2021-01-07', 2),

('A', '2021-01-10', 3),

('A', '2021-01-11', 3),

('A', '2021-01-11', 3),

('B', '2021-01-01', 2),

('B', '2021-01-02', 2),

('B', '2021-01-04', 1),

('B', '2021-01-11', 1),

('B', '2021-01-16', 3),

('B', '2021-02-01', 3),

('C', '2021-01-01', 3),

('C', '2021-01-01', 3),

('C', '2021-01-07', 3);

CREATE TABLE menu(

product\_id INTEGER,

product\_name VARCHAR(5),

price INTEGER

);

INSERT INTO menu

(product\_id, product\_name, price)

VALUES

(1, 'sushi', 10),

(2, 'curry', 15),

(3, 'ramen', 12);

CREATE TABLE members(

customer\_id VARCHAR(1),

join\_date DATE

);

INSERT INTO members

(customer\_id, join\_date)

VALUES

('A', '2021-01-07'),

('B', '2021-01-09');

**2. Questions and SQL Queries**

1. **Total Amount Spent by Each Customer:** This query calculates the total amount each customer spent at the restaurant by summing the prices of the items they purchased.

SELECT s.customer\_id, SUM(m.price) AS total\_spent

FROM sales s

JOIN menu m ON s.product\_id = m.product\_id

GROUP BY s.customer\_id;

1. **Number of Days Each Customer Visited:** This query counts the number of distinct days each customer visited the restaurant.

**SELECT s.customer\_id, COUNT(DISTINCT s.order\_date) AS days\_visited**

**FROM sales s**

**GROUP BY s.customer\_id;**

1. First Item Purchased by Each Customer: This query identifies the first item each customer purchased by finding the earliest order date and matching it with the product.

**WITH customer\_first\_purchase AS (**

**SELECT s.customer\_id, MIN(s.order\_date) AS first\_purchase\_date**

**FROM sales s**

**GROUP BY s.customer\_id**

**)**

**SELECT cfp.customer\_id, cfp.first\_purchase\_date, m.product\_name**

**FROM customer\_first\_purchase cfp**

**JOIN sales s ON s.customer\_id = cfp.customer\_id AND cfp.first\_purchase\_date = s.order\_date**

**JOIN menu m ON m.product\_id = s.product\_id;**

1. Most Purchased Item on the Menu: This query determines the most purchased items on the menu by counting the number of times each item was ordered.

**SELECT TOP 3 m.product\_name, COUNT(\*) AS total\_purchased**

**FROM sales s**

**JOIN menu m ON s.product\_id = m.product\_id**

**GROUP BY m.product\_name**

**ORDER BY total\_purchased DESC;**

1. Most Popular Item for Each Customer: This query finds the most popular item for each customer by counting the number of times each item was purchased and ranking them.

WITH customer\_popularity AS (

SELECT s.customer\_id, m.product\_name, COUNT(\*) AS purchase\_count,

DENSE\_RANK() OVER(PARTITION BY s.customer\_id ORDER BY COUNT(\*) DESC) AS rank

FROM sales s

JOIN menu m ON s.product\_id = m.product\_id

GROUP BY s.customer\_id, m.product\_name

)

SELECT cp.customer\_id, cp.product\_name, cp.purchase\_count

FROM customer\_popularity cp

WHERE rank = 1;

1. **First Item Purchased After Becoming a Member:** This query identifies the first item each customer purchased after becoming a member.

WITH first\_purchase\_after\_membership AS (

SELECT s.customer\_id, MIN(s.order\_date) AS first\_purchase\_date

FROM sales s

JOIN members mb ON s.customer\_id = mb.customer\_id

WHERE s.order\_date >= mb.join\_date

GROUP BY s.customer\_id

)

SELECT fpam.customer\_id, m.product\_name

FROM first\_purchase\_after\_membership fpam

JOIN sales s ON s.customer\_id = fpam.customer\_id AND fpam.first\_purchase\_date = s.order\_date

JOIN menu m ON s.product\_id = m.product\_id;

1. **Item Purchased Just Before Becoming a Member:** This query finds the last item each customer purchased before becoming a member.

WITH last\_purchase\_before\_membership AS (

SELECT s.customer\_id, MAX(s.order\_date) AS last\_purchase\_date

FROM sales s

JOIN members mb ON s.customer\_id = mb.customer\_id

WHERE s.order\_date < mb.join\_date

GROUP BY s.customer\_id

)

SELECT lpbm.customer\_id, m.product\_name

FROM last\_purchase\_before\_membership lpbm

JOIN sales s ON lpbm.customer\_id = s.customer\_id AND lpbm.last\_purchase\_date = s.order\_date

JOIN menu m ON s.product\_id = m.product\_id;

1. **Total Items and Amount Spent Before Becoming a Member:** This query calculates the total number of items and the total amount spent by each customer before they became a member.

SELECT s.customer\_id, COUNT(\*) AS total\_items, SUM(m.price) AS total\_spent

FROM sales s

JOIN menu m ON s.product\_id = m.product\_id

JOIN members mb ON s.customer\_id = mb.customer\_id

WHERE s.order\_date < mb.join\_date

GROUP BY s.customer\_id;

1. **Points Calculation:** This query calculates the total points each customer has, with a special multiplier for sushi

SELECT s.customer\_id, SUM(

CASE

WHEN m.product\_name = 'sushi' THEN m.price \* 20

ELSE m.price \* 10

END) AS total\_points

FROM sales s

JOIN menu m ON s.product\_id = m.product\_id

GROUP BY s.customer\_id;

1. **Points Calculation for First Week After Membership:** This query calculates the total points for customers A and B, considering a special multiplier for the first week after they join the membership program.

SELECT s.customer\_id,

SUM(

CASE

WHEN s.order\_date BETWEEN mb.join\_date AND DATEADD(day, 7, mb.join\_date) THEN m.price \* 20

WHEN m.product\_name = 'sushi' THEN m.price \* 20

ELSE m.price \* 10

END

) AS total\_points

FROM sales s

JOIN menu m ON s.product\_id = m.product\_id

LEFT JOIN members mb ON s.customer\_id = mb.customer\_id

WHERE s.customer\_id IN ('A', 'B')

AND s.order\_date <= '2021-01-31'

GROUP BY s.customer\_id;

1. **Recreate Table Output:** This query recreates the table output with an additional column indicating whether the customer was a member at the time of purchase.

SELECT s.customer\_id, s.order\_date, m.product\_name, m.price,

CASE WHEN s.order\_date >= mb.join\_date THEN 'Y'

ELSE 'N' END AS member

FROM sales s

JOIN menu m ON s.product\_id = m.product\_id

LEFT JOIN members mb ON s.customer\_id = mb.customer\_id

ORDER BY s.customer\_id, s.order\_date;

1. **Ranking Items:** This query ranks the items purchased by each customer after they became a member.

WITH customers\_data AS (

SELECT s.customer\_id, s.order\_date, m.product\_name, m.price,

CASE

WHEN s.order\_date < mb.join\_date THEN 'N'

WHEN s.order\_date >= mb.join\_date THEN 'Y'

ELSE 'N' END AS member

FROM sales s

LEFT JOIN members mb ON s.customer\_id = mb.customer\_id

JOIN menu m ON s.product\_id = m.product\_id

)

SELECT \*,

CASE WHEN member = 'N' THEN NULL

ELSE RANK