Case study 1- Danny’s Dinner

Q1. What is the total amount each customer spent at the restaurant?

|  |  |
| --- | --- |
| customer\_id | total\_sales |
| A | 76 |
| B | 74 |
| C | 36 |

Ans- Use sum ,join ,& group by to solve this query

**Q2. How many days has each customer visited the restaurant?**

**Ans-** Use count & group by to solve this query

|  |  |
| --- | --- |
| customer\_id | days\_visited |
| A | 4 |
| B | 6 |
| C | 2 |

**Q3. What was the first item from the menu purchased by each customer?**

Ans-

* Use dense\_rank(), group by, order by and with clause.
* row\_number- row number is a function that assigns a sequential integer to each row within the partition.

Instead of row\_number, we will use dense\_rank

|  |  |
| --- | --- |
| **Customer\_id** | **Product\_name** |
| **A** | sushi |
| **A** | Curry |
| **B** | Curry |
| **C** | Ramen |

Q4. What is the most purchased item on the menu and how many times was it purchased by all customers?

Ans- Use count, & join, group by & order by to solve the query

|  |  |
| --- | --- |
| **product\_name** | **most\_purchased** |
| **ramen** | 8 |
| **curry** | 4 |
| **sushi** | 3 |

**Q5. Which item was the most popular for each customer?**

**Ans**:

* Use with clause for creating a sub &
* Other functions- join, group by & order by

|  |  |  |
| --- | --- | --- |
| **customer\_id** | **product\_name** | **order\_count** |
| **A** | ramen | 3 |
| **B** | curry | 2 |
| **B** | sushi | 2 |
| **B** | ramen | 2 |
| **C** | ramen | 3 |

**Q6. Which item was purchased first by the customer after they became a member?**

**Ans-** Use dense\_rank(), join, & with clause to solve the query

|  |  |  |
| --- | --- | --- |
| customer\_id | product\_id | order\_id |
| B | sushi | 2021-01-11 |
| A | curry | 2021-01-07 |

**Q7. Which item was purchased just before the customer became a member?**

**Ans-** Use dense\_rank(), join, & with clause to solve the query

|  |  |  |
| --- | --- | --- |
| customer\_id | product\_name | order\_date |
| A | sushi | 2021-01-01 |
| B | sushi | 2021-01-04 |
| A | curry | 2021-01-01 |

**Q8. What is the total items and amount spent for each member before they become a member?**

**Ans**- Use join, group by, count, sum & logical operators to solve the query

|  |  |  |
| --- | --- | --- |
| customer\_id | total\_items | total\_price |
| B | 3 | 40 |
| A | 2 | 25 |

**Q9. If each $1 spent equates to 10 points and sushi has a 2x points multiplier- how many points would each customer have?**

**Ans**- Use with clause, case-when, join, sum & group by to solve the query

**Case-when** is used for conditional statement.

|  |  |
| --- | --- |
| customer\_id | total\_points |
| A | 860 |
| B | 940 |
| C | 360 |

**Q10. In the first week after a customer joins the program (including their join date) they earn 2x points on all items, not just sushi - how many points do customer A and B have at the end of January?**

**Ans-** Use date, case-when, between and with clause , sum, and logical operators to solve the query.

|  |  |
| --- | --- |
| customer\_id | total\_points |
| B | 820 |
| A | 1010 |

**Bonus Questions-**

1. **Join All The Things - Recreate the table with: customer\_id, order\_date, product\_name, price, member (Y/N)**

Use **case-when, order by,** & **left join,** to solve the squery

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| customer\_id | order\_date | price | product\_name | members |
| A | 2021-01-01 | 10 | sushi | N |
| A | 2021-01-01 | 10 | sushi | N |
| A | 2021-01-01 | 15 | curry | N |
| A | 2021-01-07 | 15 | curry | Y |
| A | 2021-01-10 | 12 | ramen | Y |
| A | 2021-01-11 | 12 | ramen | Y |
| A | 2021-01-11 | 12 | ramen | Y |
| B | 2021-01-01 | 15 | curry | N |
| B | 2021-01-02 | 15 | curry | N |
| B | 2021-01-04 | 10 | sushi | N |
| B | 2021-01-11 | 10 | sushi | Y |
| B | 2021-01-16 | 12 | ramen | Y |
| B | 2021-02-01 | 12 | ramen | Y |
| C | 2021-01-01 | 12 | ramen | N |
| C | 2021-01-01 | 12 | ramen | N |
| C | 2021-01-07 | 12 | ramen | N |

1. **Rank All The Things - Danny also requires further information about the ranking of customer products, but he purposely does not need the ranking for non-member purchases so he expects null ranking values for the records when customers are not yet part of the loyalty program.**

Use **dense\_rank, with clause, case-when, left join,** & **order by** to solve the query

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| customer\_id | order\_date | price | product\_name | members | Ranking |
| A | 2021-01-01 | 10 | sushi | N | Null |
| A | 2021-01-01 | 10 | sushi | N | Null |
| A | 2021-01-01 | 15 | curry | N | 1 |
| A | 2021-01-07 | 15 | curry | Y | 2 |
| A | 2021-01-10 | 12 | ramen | Y | 3 |
| A | 2021-01-11 | 12 | ramen | Y | 3 |
| A | 2021-01-11 | 12 | ramen | Y | Null |
| B | 2021-01-01 | 15 | curry | N | Null |
| B | 2021-01-02 | 15 | curry | N | Null |
| B | 2021-01-04 | 10 | sushi | N | 1 |
| B | 2021-01-11 | 10 | sushi | Y | 2 |
| B | 2021-01-16 | 12 | ramen | Y | 3 |
| B | 2021-02-01 | 12 | ramen | Y | Null |
| C | 2021-01-01 | 12 | ramen | N | Null |
| C | 2021-01-01 | 12 | ramen | N | Null |
| C | 2021-01-07 | 12 | ramen | N | Null |