

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

Query:

```
select customer_id, sum(price) as total_amount
from (
    select s.customer_id, m.price
    from dannys_diner.sales s
    left join dannys_diner.menu m
    on s.product_id = m.product_id) sm
group by customer_id
order by customer_id asc;
```

Result:

customer_id character varying (1)	total_amount bigint
A	76
B	74
C	36

Insight: Out of 3 customers, customer A spent the most with total of \$76, then followed by customer B of \$74, and customer C of \$36.

2. How many days has each customer visited the restaurant?

Query:

```
select customer_id, count(distinct order_date) as total_visit
from dannys_diner.sales
group by customer_id
order by customer_id asc;
```

Result:

customer_id character varying (1)	total_visit bigint
A	4
B	6
C	2

Insight: Out of 3 customers, customer B came to the restaurant the most, 6 times. Then followed by customer A 4 times and customer twice.

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

Query:

```
select customer_id, product_name
from(
    select s.customer_id, s.order_date, m.product_name,
    dense_rank() over(partition by s.customer_id
                        order by s.order_date) as rank
    from dannys_diner.sales s
    left join dannys_diner.menu m
    on s.product_id = m.product_id) slmn
where rank = 1
group by customer_id, product_name;
```

Result:

customer_id character varying (1)	product_name character varying (5)
A	curry
A	sushi
B	curry
C	ramen

Insight: From the result above, we can see that customer A bought two items: curry & sushi. While customer B ordered only a curry, and customer C purchased only a ramen.

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

Query:

```
select m.product_name, count(s.product_id) as total_purchased
from dannys_diner.sales s
left join dannys_diner.menu m
on s.product_id = m.product_id
group by m.product_name
order by total_purchased desc
limit 1;
```

Result:

product_name	total_purchased
character varying (5)	bigint
ramen	8

Insight: The most purchased item is ramen with a total order of 8.

5. Which item was the most popular for each customer?

Query:

```
select customer_id, product_name as most_purchased, total_product as total_purchased
from(
    select s.customer_id, m.product_name, count(m.product_name) as total_product,
    dense_rank() over(partition by customer_id
                        order by count(m.product_name) desc) as rank
    from dannys_diner.sales s
    left join dannys_diner.menu m
    on s.product_id = m.product_id
    group by s.customer_id,m.product_name
    order by s.customer_id, count(m.product_name) ASC) sm
where rank = 1;
```

Result:

customer_id	most_purchased	total_purchased
character varying (1)	character varying (5)	bigint
A	ramen	3
B	sushi	2
B	curry	2
B	ramen	2
C	ramen	3

Insight: Ramen is the most picked item by customers A & C with a total purchase of 3 While customer B likes all menus with the same total order for each menu of 2.

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

Query:

```
select customer_id, order_date, product_name
from(select customer_id, order_date, product_name,
         dense_rank() over( partition by customer_id
                           order by order_date asc) as order_rank
 from(select vta.customer_id, vta.order_date, vta.join_date, mn.product_name
        from(select s.customer_id, s.order_date, m.join_date, s.product_id
              from dannys_diner.sales s
              inner join dannys_diner.members m
              on s.customer_id = m.customer_id) vta
        left join dannys_diner.menu mn
        on vta.product_id = mn.product_id
        where order_date >= join_date
        order by customer_id asc) vtb ) vtc
where order_rank = 1;
```

Result:

customer_id character varying (1)	order_date date	product_name character varying (5)
A	2021-01-07	curry
B	2021-01-11	sushi

Insight: Out of 3 customers, customers A & B registered for being a member. Customer A ordered a curry right after he became a member. While customer B ordered sushi.

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

Query:

```
select customer_id, order_date, product_name
from(select customer_id, order_date, join_date, product_name,
         dense_rank() over( partition by customer_id
                           order by order_date desc) as order_rank
 from(select vta.customer_id, vta.order_date, vta.join_date, mn.product_name
        from(select s.customer_id, s.order_date, m.join_date, s.product_id
              from dannys_diner.sales s
              inner join dannys_diner.members m
              on s.customer_id = m.customer_id) vta
        left join dannys_diner.menu mn
        on vta.product_id = mn.product_id
        where order_date < join_date
        order by customer_id asc) vtb ) vtc
where order_rank = 1;
```

Result:

customer_id character varying (1)	order_date date	product_name character varying (5)
A	2021-01-01	sushi
A	2021-01-01	curry
B	2021-01-04	sushi

Insight: Customer A ordered sushi and curry, while customer B ordered a sushi.

8. What is the total items and amount spent for each member before they became a member?

Query:

```
select vtd.customer_id, count(vtd.product_name) as total_item, sum(mnu.price) as amount_spent
from (select customer_id, order_date, product_name
      from (select customer_id, order_date, join_date, product_name,
                  dense_rank() over (partition by customer_id
                                     order by order_date desc) as order_rank
            from (select vta.customer_id, vta.order_date, vta.join_date, mn.product_name
                  from (select s.customer_id, s.order_date, m.join_date, s.product_id
                        from dannys_diner.sales s
                        inner join dannys_diner.members m
                        on s.customer_id = m.customer_id) vta
                  left join dannys_diner.menu mn
                  on vta.product_id = mn.product_id
                  where order_date < join_date
                  order by customer_id asc) vtb ) vtc
      where order_rank = 1) vtd
left join dannys_diner.menu mnu
on vtd.product_name = mnu.product_name
group by vtd.customer_id;
```

Result:

customer_id character varying (1)	total_item bigint	amount_spent bigint
A	2	25
B	1	10

Insight: Customer A spent \$25 for 2 items and customer B spent \$10 for 1 item.

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

Query:

```
select customer_id, sum(points) as total_points
from (select customer_id, product_name, sum(price) as total_price,
      case when product_name = 'sushi' then sum(price*20)
      else sum(price*10)
      end as points
      from (select s.customer_id, m.product_name, m.price
            from dannys_diner.sales s
            left join dannys_diner.menu m
            on s.product_id = m.product_id) vta
      group by customer_id, product_name) vtb
group by customer_id
order by customer_id asc;
```

Result:

customer_id character varying (1)	total_points numeric
A	860
B	940
C	360

Insight: Customer B earns the highest points of 940, followed by customer A with 860 points, and customer C with 360 points.

10. 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?

Query:

```
select customer_id, sum(points) as total_points
from (select customer_id, category, product_name,
      case when category = 'before' and product_name = 'sushi' then price*20
      when category = 'before' and product_name <> 'sushi' then price*10
      when category = '1st week' then price*20
      when category = 'after 1st week' and product_name = 'sushi' then price*20
      else price*10
      end as points
      from (select vtb.customer_id, vtb.category, vtb.product_id, m.price, m.product_name
            from (select customer_id, order_date, join_date,
                  case when order_date < join_date then 'before'
                  when order_date >= join_date and order_date <= (join_date + INTERVAL '6 DAY') then '1st week'
                  else 'after 1st week'
                  end as category,
                  product_id
                  from (select s.customer_id, s.order_date, m.join_date, s.product_id
                        from dannys_diner.sales s
                        inner join dannys_diner.members m
                        on s.customer_id = m.customer_id
                        where s.order_date <= (date_trunc('month', to_timestamp('01 Jan 2021', 'DD Mon YYYY')) + interval '1 month - 1 day')) vta) vtb
            left join dannys_diner.menu m
            on vtb.product_id = m.product_id) vtc) vtd
group by customer_id;
```

Result:

customer_id character varying (1) 🔒	total_points bigint 🔒
A	1370
B	940

Insight: Customer A collects 1370 point and customer B collects 940 in the end of January.

11. Bonus Question – Join All The Things: The following questions are related creating basic data tables that Danny and his team can use to quickly derive insights without needing to join the underlying tables using SQL. Recreate the following table output using the available data:

customer_id	order_date	product_name	price	member
A	2021-01-01	curry	15	N
A	2021-01-01	sushi	10	N
A	2021-01-07	curry	15	Y
A	2021-01-10	ramen	12	Y
A	2021-01-11	ramen	12	Y
A	2021-01-11	ramen	12	Y
B	2021-01-01	curry	15	N
B	2021-01-02	curry	15	N
B	2021-01-04	sushi	10	N
B	2021-01-11	sushi	10	Y
B	2021-01-16	ramen	12	Y
B	2021-02-01	ramen	12	Y
C	2021-01-01	ramen	12	N
C	2021-01-01	ramen	12	N
C	2021-01-07	ramen	12	N

Query:







```
select customer_id, order_date, product_name, price,
       case when join_date isnull then 'N'
            when order_date < join_date then 'N'
            else 'Y'
       end as member
from(select s.customer_id, s.order_date, mn.product_name, mn.price, me.join_date
     from dannys_diner.sales s
     left join dannys_diner.menu mn
     on s.product_id = mn.product_id
     left join dannys_diner.members me
     on s.customer_id = me.customer_id) vta
order by customer_id, order_date, product_name asc;
```

12. 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.

Query:

```
select *,
       case when member = 'Y' then rank() over(partition by customer_id, member
                                               order by order_date asc)
            else NULL
       end as ranking
from(select customer_id, order_date, product_name, price,
       case when join_date isnull then 'N'
            when order_date < join_date then 'N'
            else 'Y'
       end as member
     from(select s.customer_id, s.order_date, mn.product_name, mn.price, me.join_date
          from dannys_diner.sales s
          left join dannys_diner.menu mn
          on s.product_id = mn.product_id
          left join dannys_diner.members me
          on s.customer_id = me.customer_id) vta
     order by customer_id, order_date, product_name asc) vtb;
```


Result:

customer_id character varying (1) 	order_date date 	product_name character varying (5) 	price integer 	member text 	ranking bigint 
A	2021-01-01	curry	15	N	[null]
A	2021-01-01	sushi	10	N	[null]
A	2021-01-07	curry	15	Y	1
A	2021-01-10	ramen	12	Y	2
A	2021-01-11	ramen	12	Y	3
A	2021-01-11	ramen	12	Y	3
B	2021-01-01	curry	15	N	[null]
B	2021-01-02	curry	15	N	[null]
B	2021-01-04	sushi	10	N	[null]
B	2021-01-11	sushi	10	Y	1
B	2021-01-16	ramen	12	Y	2
B	2021-02-01	ramen	12	Y	3
C	2021-01-01	ramen	12	N	[null]
C	2021-01-01	ramen	12	N	[null]
C	2021-01-07	ramen	12	N	[null]