SQL Coding Question

Orders Table

order_id	customer_id	product	quantity	order_date
1	101	Phone	2	2020-01-01
2	102	Laptop	1	2020-02-01
3	102	Mouse	2	2020-03-01
4	103	Headphone	3	2020-04-01

1. How would you select all columns from the orders table?

Answer-

```sql
SELECT * FROM orders;
```

2. How would you select all columns from the orders table but only the rows where the quantity is greater than 1?

Answer-

```sql
SELECT * FROM orders WHERE quantity > 1;
```

3. How would you select the product and quantity columns from the orders table where the order_date is between '2020-02-01' and '2020-03-31'?

Answer-

```sql
SELECT product, quantity FROM orders
WHERE order_date BETWEEN '2020-02-01' AND '2020-03-31';
```

4. How would you count the number of orders in the orders table?

Answer-

```sql
SELECT COUNT(*) FROM orders;
```

5. How would you select the unique customer_id values from the orders table?

Answer-

```sql
SELECT DISTINCT customer_id FROM orders;
```

6. How would you calculate the total quantity of items ordered in the orders table?

Answer-

```sql
SELECT SUM(quantity) FROM orders;
```

7. How would you find the average quantity of items ordered per order in the orders table?
Answer-

```
SELECT AVG(quantity) FROM orders;
```

8. How would you select the product, quantity, and order_date columns from the orders table and order the results by the order_date in ascending order?
Answer-

```
SELECT product, quantity, order_date FROM orders
ORDER BY order_date;
```

9. How would you update the quantity to 4 for the order with an order_id of 2 in the orders table?
Answer-

```
UPDATE orders SET quantity = 4 WHERE order_id = 2;
```

10. How would you delete the order with an order_id of 3 in the orders table?
Answer-

```
DELETE FROM orders WHERE order_id = 3;
```

11. How would you insert a new order into the orders table with order_id 5, customer_id 104, product 'Keyboard', quantity 3, and order_date '2020-05-01'?
Answer-

```
INSERT INTO orders (order_id, customer_id, product, quantity,
order_date)
VALUES (5, 104, 'Keyboard', 3, '2020-05-01');
```

12. How would you select the product and order_date columns from the orders table where the customer_id is 102?
Answer-

```
SELECT product, order_date FROM orders
WHERE customer_id = 102;
```

13. How would you find the maximum quantity ordered in the orders table?
Answer-

```
SELECT MAX(quantity) FROM orders;
```

14. How would you find the minimum quantity ordered in the orders table?
Answer-

```
SELECT MIN(quantity) FROM orders;
```

15. How would you select the product and quantity columns from the orders table, but only for orders with a quantity greater than the average quantity of all orders in the table?
Answer-

```
SELECT product, quantity FROM orders
WHERE quantity > (SELECT AVG(quantity) FROM orders);
```

16. How would you find the most frequently ordered product in the orders table?
Answer-

```
SELECT product, COUNT(product) FROM orders
GROUP BY product
ORDER BY COUNT(product) DESC
LIMIT 1;
```

17. How would you join the orders table with a customers table to get all the customer information for each order in the orders table?
Answer-

```
CREATE TABLE customers (
  customer_id INT PRIMARY KEY,
  name VARCHAR(50),
  address VARCHAR(100)
);

SELECT * FROM orders
JOIN customers
ON orders.customer_id = customers.customer_id;
```

18. How would you find the total quantity of each product ordered for each customer in the orders table?
Answer-

```
SELECT customers.name, orders.product, SUM(orders.quantity)
FROM orders
JOIN customers
ON orders.customer_id = customers.customer_id
GROUP BY customers.name, orders.product;
```

19. Write a SQL query to find the total number of orders for each customer.
Dataset:

orderid	customer	date
1	A	1/1/2022
2	A	2/2/2022
3	B	3/3/2022
4	B	4/4/2022
5	C	5/5/2022

Answer-

```
SELECT customer, count(*) as total_orders
FROM orders
GROUP BY customer;
```

20. Write a SQL query to find the total sales for each month.
Dataset:

orderid	customer	date	amount
1	A	1/1/2022	100
2	B	2/2/2022	200
3	C	3/3/2022	300
4	D	4/4/2022	400
5	E	5/5/2022	500

Answer-

```
SELECT MONTH(date) as month, SUM(amount) as total_sales
FROM orders
GROUP BY MONTH(date);
```

21. Write a SQL query to find the customers who have made more than 2 orders.
Dataset:

orderid	customer	date
1	A	1/1/2022
2	A	2/2/2022
3	B	3/3/2022
4	B	4/4/2022
5	C	5/5/2022

Answer-

```
SELECT customer, count(*) as total_orders
FROM orders
GROUP BY customer
HAVING count(*) > 2;
```

22. Write a SQL query to find the second highest salary of all employees in a company.
Dataset:

empid	name	salary
1	John	5000
2	Sarah	6000
3	Michael	7000
4	David	8000
5	Alice	9000

Answer-

```
SELECT MAX(salary)
FROM (
  SELECT salary
  FROM employees
  ORDER BY salary DESC
  LIMIT 2
) AS second_highest_salary;
```

23. Write a SQL query to find the products that have been sold in all stores.
Dataset:

storeid	product	sold
1	P1	10
2	P1	15
3	P1	20
1	P2	5
2	P2	10
3	P3	20

Answer-

```
SELECT product
FROM sales
GROUP BY product
HAVING COUNT(DISTINCT storeid) = (SELECT COUNT(DISTINCT storeid)
FROM sales);
```

24. Write a SQL query to find the customers who have made the maximum number of orders in the past 30 days.
Dataset:

orderid	customer	date
1	A	1/1/2022
2	B	2/2/2022
3	A	3/3/2022
4	C	4/4/2022
5	B	5/5/2022

Answer-

```
SELECT customer, count(*) as total_orders
FROM orders
WHERE date >= DATE_SUB(NOW(), INTERVAL 30 DAY)
GROUP BY customer
ORDER BY total_orders DESC
LIMIT 1;
```

25. Write a SQL query to find the products that have never been sold.
Dataset:

storeid	product	sold
1	P1	10
2	P1	15
3	P2	20
1	P3	5
2	P4	10

Answer-

```
SELECT product
FROM products
WHERE product NOT IN (SELECT product FROM sales);
```

26. Write a SQL query to find the average salary of all employees in each department, ordered by the average salary in descending order.
Dataset:

empid	name	salary	deptid
1	John	5000	1
2	Sarah	6000	2
3	Michael	7000	1
4	David	8000	2
5	Alice	9000	1

Answer-

```
SELECT deptid, AVG(salary) as average_salary
FROM employees
GROUP BY deptid
ORDER BY average_salary DESC;
```

27. Write a SQL query to find the total number of orders for each customer in the past year.
Dataset:

orderid	customer	date
1	A	1/1/2022
2	B	2/2/2022
3	A	3/3/2022
4	C	4/4/2022
5	B	5/5/2022

Answer-

```
SELECT customer, count(*) as total_orders
FROM orders
WHERE date >= DATE_SUB(NOW(), INTERVAL 1 YEAR)
GROUP BY customer;
```

28. Write a SQL query to find the total sales for each store in the past month.
Dataset:

storeid	product	sold
1	P1	10
2	P1	15
3	P2	20
1	P3	5
2	P4	10

Answer-

```
SELECT storeid, SUM(sold) as total_sales
FROM sales
WHERE date >= DATE_SUB(NOW(), INTERVAL 1 MONTH)
GROUP BY storeid;
```

29.Write a SQL query to find the names of the employees who have a salary greater than the average salary of all employees.
Dataset:

empid	name	salary
1	John	5000
2	Sarah	6000
3	Michael	7000
4	David	8000
5	Alice	9000

Answer-

```
SELECT name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);
```

30. Write a SQL query to find the average salary of employees who joined the company in the past 6 months.
Dataset:

empid	name	salary	date
1	John	5000	1/1/2022
2	Sarah	6000	2/2/2022
3	Michael	7000	3/3/2022
4	David	8000	4/4/2022
5	Alice	9000	5/5/2022

Answer-

```
SELECT AVG(salary) as average_salary
FROM employees
WHERE date >= DATE_SUB(NOW(), INTERVAL 6 MONTH);
```

31. Write a SQL query to find the names and departments of employees who have the same salary as the highest-paid employee.
Dataset:

empid	name	salary	deptid
1	John	5000	1
2	Sarah	6000	2
3	Michael	7000	1
4	David	8000	2
5	Alice	9000	1

Answer-

```
SELECT name, department
FROM employees
JOIN departments
ON employees.deptid = departments.deptid
WHERE salary = (SELECT MAX(salary) FROM employees);
```

32. Write a SQL query to find the most popular product among customers who have made more than 5 orders in the past year.
Dataset:

orderid	customer	product
1	A	P1
2	B	P2
3	A	P1
4	C	P3
5	B	P2

Answer-

```
SELECT product, COUNT(product) as product_count
FROM orders
WHERE customer IN (SELECT customer
                   FROM orders
                   GROUP BY customer
                   HAVING COUNT(*) > 5)
AND date >= DATE_SUB(NOW(), INTERVAL 1 YEAR)
GROUP BY product
ORDER BY product_count DESC
LIMIT 1;
```

33. Write a SQL query to find the names of employees who have not been assigned to any projects.
Dataset:

empid	name	project
1	John	P1
2	Sarah	P2
3	Michael	P3
4	David	P2
5	Alice	P1

Answer-

```
SELECT name
FROM employees
WHERE empid NOT IN (SELECT empid FROM projects);
```

34. Write a SQL query to find the average salary of employees in each department.
Dataset:

empid	name	salary	deptid
1	John	5000	1
2	Sarah	6000	2
3	Michael	7000	1
4	David	8000	2
5	Alice	9000	1

Answer-

```
SELECT departments.deptname, AVG(salary) as avg_salary
FROM employees
JOIN departments
ON employees.deptid = departments.deptid
GROUP BY deptname;
```

35. Write a SQL query to find the most frequently ordered product for each customer.
Dataset:

orderid	customer	product
1	A	P1
2	B	P2
3	A	P1
4	C	P3
5	B	P2

Answer-

```
SELECT customer, product, COUNT(product) as product_count
FROM orders
GROUP BY customer, product
ORDER BY customer, product_count DESC;
```

36. Write a SQL query to find the second highest salary of employees.
Dataset:

empid	name	salary
1	John	5000
2	Sarah	6000
3	Michael	7000
4	David	8000
5	Alice	9000

Answer-

```
SELECT MAX(salary) as second_highest_salary
FROM employees
WHERE salary NOT IN (SELECT MAX(salary) from employees);
```

37. Write a SQL query to find the number of orders made by each customer in the past year.
Dataset:

orderid	customer	product	date
1	A	P1	2022-01
2	B	P2	2022-02
3	A	P1	2022-03
4	C	P3	2022-04
5	B	P2	2022-05

Answer-

```
SELECT customer, COUNT(orderid) as order
```

38. Write a query to retrieve the name, salary, and job title of all employees, along with the average salary for each job title.
Answer-

```
CREATE TABLE employees (
  id INT PRIMARY KEY,
  name VARCHAR(255),
  job_title VARCHAR(255),
  salary DECIMAL(10,2)
);

INSERT INTO employees (id, name, job_title, salary)
VALUES
  (1, 'John Doe', 'Manager', 75000),
  (2, 'Jane Doe', 'Developer', 65000),
  (3, 'Jim Smith', 'Manager', 80000),
  (4, 'Sarah Johnson', 'Analyst', 70000);
```

```
WITH avg_salary AS (
  SELECT job_title, AVG(salary) avg_salary
  FROM employees
  GROUP BY job_title
)
```

```
SELECT e.name, e.salary, e.job_title, a.avg_salary
FROM employees e
JOIN avg_salary a ON e.job_title = a.job_
```

39. Write a query to retrieve the name and salary of all employees, along with the total salary of all employees.

Answer-

```
WITH total_salary AS (
  SELECT SUM(salary) total_salary
  FROM employees
)
SELECT name, salary, (SELECT total_salary FROM total_salary) AS
total_salary
FROM employees;
```

40. Write a query to retrieve the name and salary of all employees, as well as a column indicating whether the salary is above or below the average salary for all employees.

Answer-

```
WITH avg_salary AS (
  SELECT AVG(salary) avg_salary
  FROM employees
)
SELECT name, salary,
       CASE
         WHEN salary > (SELECT avg_salary FROM avg_salary) THEN
'Above Average'
         ELSE 'Below Average'
       END AS salary_status
FROM employees;
```

41. Write a query to retrieve the name and salary of the highest-paid employee in each job title.

Answer-

```
SELECT job_title, name, salary
FROM employees
WHERE salary = (SELECT MAX(salary) FROM employees WHERE job_title
= employees.job_title);
```

42. Write a query to retrieve the number of employees in each job title.
Answer-

```
SELECT job_title, COUNT(*)
FROM employees
GROUP BY job_title;
```

43.
Dataset
https://docs.google.com/spreadsheets/d/1O5HzZoktA-TKypdfUZTRLO3jWBCkQLQQj1zrWgtYEs0/edit?usp=sharing

Write a query to find the name (first_name, last_name) and the salary of the employees who have a higher salary than the employee whose last_name='Bull'.
Answer-

```
SELECT FIRST_NAME, LAST_NAME, SALARY
FROM employees
WHERE SALARY >
(SELECT salary FROM employees WHERE last_name = 'Bull');
```

44. Write a query to find the name (first_name, last_name) of all employees who works in the IT department
Answer-

```
SELECT first_name, last_name
FROM employees
WHERE department_id
IN (SELECT department_id FROM departments WHERE
department_name='IT');
```

45. Write a query to get the details of the employees where the length of the first name greater than or equal to 8.
Answer-

```
SELECT *
FROM employees
WHERE LENGTH(first_name) >= 8;
```

46. Write a query to display the last name of employees having 'e' as the third character.
Answer-

```
SELECT last_name FROM employees WHERE last_name LIKE '__e%';
```

47. Write a query to select all record from employees where last name in 'BLAKE', 'SCOTT', 'KING' and 'FORD'.
Answer-

```
SELECT *
FROM employees
WHERE last_name IN('JONES', 'BLAKE', 'SCOTT', 'KING',
'FORD');
```

48. Write a query to get the maximum salary of an employee working as a Programmer.
Answer-

```
SELECT MAX(salary)
FROM employees
WHERE job_id = 'IT_PROG';
```

49. Write a query to get the average salary and number of employees working the department 90.
Answer-

```
SELECT AVG(salary),count(*)
FROM employees
WHERE department_id = 90;
```

50. Write a query to get the difference between the highest and lowest salaries.
Answer-

```
SELECT MAX(salary) - MIN(salary) DIFFERENCE
FROM employees;
```

51. Write a query to get the average salary for all departments employing more than 10 employees.
Answer-

```
SELECT department_id, AVG(salary), COUNT(*)
FROM employees
GROUP BY department_id
HAVING COUNT(*) > 10;
```

52.
Dataset: Products Table
https://docs.google.com/spreadsheets/d/1MGeI6aJkeLnRmYECTVCC5hs-lH1qUwX6M3bb6wuNOPM/edit?usp=sharing

Write a query to count current and discontinued products.
Answer-

```
SELECT Count(ProductName)
FROM Products
GROUP BY Discontinued;
```

53. Write a query to get Product list (name, units on order , units in stock) of stock is less than the quantity on order.
Answer-

```
SELECT ProductName,  UnitsOnOrder , UnitsInStock
FROM Products
WHERE (((Discontinued)=False) AND
((UnitsInStock)<UnitsOnOrder));
```

54.Write a query to get Product list (name, unit price) of ten most expensive products.

Answer-

```
SELECT DISTINCT ProductName as
Twenty_Most_Expensive_Products, UnitPrice
FROM Products AS a
WHERE 20 >= (SELECT COUNT(DISTINCT UnitPrice)
                    FROM Products AS b
                    WHERE b.UnitPrice >= a.UnitPrice)
ORDER BY UnitPrice desc;
```

55.Write a query to get Product list (id, name, unit price) where products cost between $15 and $25.

Answer-

```
SELECT ProductName, UnitPrice
FROM Products
WHERE (((UnitPrice)>=15 And (UnitPrice)<=25)
AND ((Products.Discontinued)=False))
ORDER BY Products.UnitPrice DESC;
```

56.Write a query to get Product list (id, name, unit price) where current products cost less than $20.

Answer-

```
SELECT ProductID, ProductName, UnitPrice
FROM Products
WHERE (((UnitPrice)<20) AND ((Discontinued)=False))
ORDER BY UnitPrice DESC;
```

57. Write a query to get Product list (name, units on order , units in stock) of stock is less than the quantity on order.

Answer-

```
SELECT ProductName,  UnitsOnOrder , UnitsInStock
FROM Products
WHERE (((Discontinued)=False) AND
((UnitsInStock)<UnitsOnOrder));
```

58. Write a query to get Product name and quantity/unit.

Answer-

```
SELECT ProductName, QuantityPerUnit
FROM Products;
```

59. **Weekly_sales Table**

week_date	region	platform	segment	customer_type	transactions	sales
31/8/20	ASIA	Retail	C3	New	120631	3656163
31/8/20	ASIA	Retail	F1	New	31574	996575
31/8/20	USA	Retail	null	Guest	529151	16509610
31/8/20	EUROPE	Retail	C1	New	4517	141942
31/8/20	AFRICA	Retail	C2	New	58046	1758388
31/8/20	CANADA	Shopify	F2	Existing	1336	243878
31/8/20	AFRICA	Shopify	F3	Existing	2514	519502
31/8/20	ASIA	Shopify	F1	Existing	2158	371417
31/8/20	AFRICA	Shopify	F2	New	318	49557
31/8/20	AFRICA	Retail	C3	New	111032	3888162

What day of the week is used for each week_date value?

Answer-

```
SELECT
     DISTINCT date_part('dow', week_day)::int AS day_of_week,
     to_char(week_day, 'Day') AS day_of_week_name
FROM clean_weekly_sales;
```

60. What range of week numbers are missing from the dataset?

Answer-

```
WITH RECURSIVE week_count AS (
      SELECT 1 AS week_num
      UNION ALL
      SELECT week_num | 1
      FROM week_count
      WHERE week_num < 52
)
SELECT week_num AS missing_weeks
FROM week_count
WHERE week_num NOT IN (
            SELECT DISTINCT week_number
            FROM clean_weekly_sales
      );
```

61. How many total transactions were there for each year in the dataset?

Answer-

```
SELECT calendar_year,
      sum(transactions) AS total_transactions
FROM clean_weekly_sales
GROUP BY calendar_year
ORDER BY calendar_year;
```

62. What is the total sales for each region for each month?

Answer-

```
SELECT region,
      calendar_year,
      month_number,
      sum(sales) AS total_sales
FROM clean_weekly_sales
GROUP BY region,
      calendar_year,
      month_number
ORDER BY calendar_year,
      month_number,
      region;
```

63. What is the total count of transactions for each platform?
Answer-

```sql
SELECT platform,
      sum(transactions) AS total_transactions
FROM clean_weekly_sales
GROUP BY platform;
```

64. What is the percentage of sales for Retail vs Shopify for each month?
Answer-

```sql
SELECT calendar_year,
      month_number,
      round(
            100 * sum(
                  CASE
                        WHEN platform = 'Retail' THEN total_sales
                        ELSE 0
                  END
            ) / sum(total_sales),
            2
      ) AS retail_perc,
      round(
            100 * sum(
                  CASE
                        WHEN platform = 'Shopify' THEN total_sales
                        ELSE 0
                  END
            ) / sum(total_sales),
            2
      ) AS shopify_perc
from (
            SELECT platform,
                  calendar_year,
                  month_number,
                  sum(sales) AS total_sales
            FROM clean_weekly_sales
            GROUP BY platform,
                  calendar_year,
                  month_number
            ORDER BY calendar_year,
                  month_number,
                  platform
      ) AS tmp
GROUP BY calendar_year,
      month_number;
```

65. What is the percentage of sales by demographic for each year in the dataset?
Answer-

```
SELECT calendar_year,
      demographics,
      sum(sales) AS sales_per_demographic,
            round(
            100 * sum(sales) / sum(sum(sales)) OVER (PARTITION BY
calendar_year),
            2
      ) AS percentage
FROM clean_weekly_sales
GROUP BY demographics,
      calendar_year
ORDER BY calendar_year,
      demographics;
```

66. Which age_band and demographic values contribute the most to Retail sales?
Answer-

```
WITH get_total_sales_from_all AS (
      SELECT
            demographics,
            age_band,
            sum(sales) AS total_sales,
            rank() OVER (ORDER BY sum(sales) desc) AS rnk,
            round(100 * sum(sales) / sum(sum(sales)) over (), 2) AS
percentage
      FROM
            clean_weekly_sales
      WHERE
            platform = 'Retail'
      AND
            age_band <> 'unknown'
      GROUP BY
            demographics,
            age_band
)
SELECT
      demographics,
      age_band,
      total_sales,
      percentage
from
      get_total_sales_from_all
WHERE rnk = 1;
```

67. Can we use the avg_transaction column to find the average transaction size for each year for Retail vs Shopify? If not - how would you calculate it instead?
Answer-

```
SELECT calendar_year,
      platform,
      (sum(sales) / sum(transactions)) AS avg_transaction_size
FROM clean_weekly_sales
GROUP BY calendar_year,
      platform
ORDER BY calendar_year,
      platform;
```

68. What is the total sales for the 4 weeks before and after 2020-06-15?
Answer-

```
SELECT CASE
            WHEN week_number BETWEEN 21 AND 24 THEN 'Before'
            WHEN week_number BETWEEN 25 AND 28 THEN 'After'
            ELSE null
      END AS time_period,
      sum(sales) AS total_sales
FROM clean_weekly_sales
WHERE calendar_year = '2020'
GROUP BY time_period -- Remove null values from time_period
HAVING (
            CASE
                  WHEN week_number BETWEEN 21 AND 24 THEN 'Before'
                  WHEN week_number BETWEEN 25 AND 28 THEN 'After'
                  ELSE null
            END
      ) IS NOT NULL
```

```
ORDER BY time_period DESC;
```

69.

```
CREATE TABLE fresh_segments.interest_map (
  "id" INTEGER,
  "interest_name" TEXT,
  "interest_summary" TEXT,
  "created_at" TIMESTAMP,
  "last_modified" TIMESTAMP
);

CREATE TABLE fresh_segments.interest_metrics (
  "_month" VARCHAR(4),
  "_year" VARCHAR(4),
  "month_year" VARCHAR(7),
  "interest_id" VARCHAR(5),
  "composition" FLOAT,
  "index_value" FLOAT,
  "ranking" INTEGER,
  "percentile_ranking" FLOAT
);
```

Update the fresh_segments.interest_metrics table by modifying the month_year column to be a date data type with the start of the month?

Answer-

```
SELECT *
FROM fresh_segments.interest_metrics
ORDER BY ranking
LIMIT 5;
```

70. What is count of records in the fresh_segments.interest_metrics for each month_year value sorted in chronological order (earliest to latest) with the null values appearing first?

Answer-

```
SELECT month_year,
      count(*) as month_year_count
FROM fresh_segments.interest_metrics
GROUP BY month_year
```

```sql
ORDER BY month_year ASC NULLS FIRST;
```

71. Which interests have been present in all month_year dates in our dataset?
Answer-

```sql
WITH persistent_interests AS (
      SELECT interest_id
      FROM fresh_segments.interest_metrics
      GROUP BY interest_id
      HAVING count(DISTINCT month_year) = (
                  SELECT count(DISTINCT month_year)
                  FROM fresh_segments.interest_metrics
            )
)
SELECT count(*) AS n_interests
FROM persistent_interests;
```

72. Using this same total_months measure - calculate the cumulative percentage of all records starting at 14 months - which total_months value passes the 90% cumulative percentage value?
Answer-

```sql
WITH cte_total_months AS (
      SELECT interest_id,
            count(DISTINCT month_year) AS total_months
      FROM fresh_segments.interest_metrics
      GROUP BY interest_id
)
SELECT total_months,
      count(*) AS n_ids
FROM cte_total_months
GROUP BY total_months
ORDER BY total_months DESC;
```

73. If we were to remove all interest_id values which are lower than the total_months value we found in the previous question - how many total data points would we be removing?

Answer-

```
WITH cte_total_months AS (
      SELECT interest_id,
            count(DISTINCT month_year) AS total_months
      FROM fresh_segments.interest_metrics
      GROUP BY interest_id
      HAVING count(DISTINCT month_year) < 6
) -- Select results that are < 90%
SELECT count(*) rows_removed
FROM fresh_segments.interest_metrics
WHERE exists(
            SELECT interest_id
            FROM cte_total_months
            WHERE cte_total_months.interest_id =
fresh_segments.interest_metrics.interest_id
      );
```

74. After removing these interests - how many unique interests are there for each month?

Answer-

```
WITH cte_total_months AS (
      SELECT interest_id,
            count(DISTINCT month_year) AS total_months
      FROM fresh_segments.interest_metrics
      GROUP BY interest_id
      HAVING count(DISTINCT month_year) >= 6
)
SELECT month_year,
      count(interest_id) AS n_interests
FROM fresh_segments.interest_metrics
WHERE interest_id IN (
            SELECT interest_id
            FROM cte_total_months
      )
GROUP BY month_year
ORDER BY month_year;
```

75. Which 5 interests had the lowest average ranking value?
Answer-

```
WITH get_lowest_avgs AS (
	SELECT ip.interest_name,
		round(avg(ranking)::numeric, 2) AS avg_ranking,
		rank() OVER (
			ORDER BY avg(ranking) desc
		) AS rnk
	FROM filtered_data
		JOIN fresh_segments.interest_map AS ip ON
interest_id::numeric = ip.id
	GROUP BY ip.interest_name
)
SELECT *
FROM get_lowest_avgs
WHERE rnk <= 5;
```

76. Which 5 interests had the largest standard deviation in their percentile_ranking value?
Answer-

```
WITH get_std_dev AS (
	SELECT ip.interest_name,
		round(stddev(percentile_ranking)::numeric, 2) AS
std_dev,
		rank() OVER (
			ORDER BY stddev(percentile_ranking) desc
		) AS rnk
	FROM filtered_data
		JOIN fresh_segments.interest_map AS ip ON
interest_id::numeric = ip.id
	GROUP BY ip.interest_name
)
SELECT *
FROM get_std_dev
WHERE rnk <= 5;
```

77.

```sql
CREATE TABLE plans (
  plan_id INTEGER,
  plan_name VARCHAR(13),
  price DECIMAL(5,2)
);

CREATE TABLE subscriptions (
  customer_id INTEGER,
  plan_id INTEGER,
  start_date DATE
);

DROP TABLE IF EXISTS subs_plans;
CREATE TEMP TABLE subs_plans AS (
      SELECT s.customer_id,
             s.plan_id,
             p.plan_name,
             p.price,
             s.start_date
      FROM subscriptions AS s
             JOIN PLANS AS p ON p.plan_id = s.plan_id
);
```

Insert Data by your own.

How many customers has Foodie-Fi ever had?

Answer-

```sql
SELECT count(DISTINCT customer_id) AS n_customers
FROM subs_plans;
```

78. What is the monthly distribution of trial plan start_date values for our dataset - use the start of the month as the group by value.

Answer-

```
SELECT to_char(start_date, 'Month') AS trial_month,
      count(*) AS n_trials
FROM subs_plans
WHERE plan_id = 0
GROUP BY trial_month
ORDER BY to_date(to_char(start_date, 'Month'), 'Month');
```

79. What plan start_date values occur after the year 2020 for our dataset? Show the breakdown by count of events for each plan_name?

Answer-

```
SELECT count(plan_name) AS n_plans,
      plan_name
FROM subs_plans
WHERE start_date >= '2020-01-01'
GROUP BY plan_name;
```

80.What is the customer count and percentage of customers who have churned rounded to 1 decimal place?

Answer-

```
DROP TABLE IF EXISTS churn_count;
CREATE TEMP TABLE churn_count AS (
      SELECT count(DISTINCT customer_id) AS n_churn
      FROM subs_plans
      WHERE plan_name = 'churn'
);
DROP TABLE IF EXISTS cust_count;
CREATE TEMP TABLE cust_count AS (
      SELECT count(DISTINCT customer_id) AS n_customers
      FROM subs_plans
);
SELECT n_customers,
      n_churn,
      round((n_churn::numeric / n_customers::numeric) * 100, 1) AS
churn_perc
FROM cust_count,
      churn_count;
```

81. What is the number and percentage of customer plans after their initial free trial?
Answer-

```
SELECT plan_name,
      count(plan_name) AS plan_count,
      round(
            (count(plan_name)::numeric / n_customers::numeric) *
100,
            2
      ) AS plan_perc
from (
            SELECT DISTINCT customer_id,
                  plan_name,
                  plan_id,
                  row_number() OVER (
                        PARTITION BY customer_id
                        ORDER BY plan_id
                  ) AS rn
            FROM subs_plans
            ORDER BY customer_id,
                  plan_id
      ) AS a,
      cust_count
WHERE rn = 2
GROUP BY plan_name,
      n_customers;
```

82. What is the customer count and percentage breakdown of all 5 plan_name values at 2020-12-31?
Answer-

```
SELECT count(customer_id) AS customer_count,
      plan_name,
      plan_id,
      round(
            (count(plan_name)::numeric / n_customers::numeric) *
100,
            2
      ) AS plan_perc
from (
            SELECT DISTINCT customer_id,
                  plan_name,
                  plan_id,
                  start_date,
                  row_number() OVER (
                       PARTITION BY customer_id
                       ORDER BY plan_id desc
                  ) AS rn
            FROM subs_plans
            WHERE start_date <= '2020-12-31' -- Must add this
condition to capture trial period
                  OR start_date BETWEEN '2020-12-25' AND
'2020-12-31'
            GROUP BY customer_id,
                  plan_name,
                  plan_id,
                  start_date
      ) AS tmp,
      cust_count
WHERE rn = 1
GROUP BY n_customers,
      plan_name,
      plan_id
ORDER BY plan_id;
```

83. How many customers have upgraded to an annual plan in 2020?

Answer-

```
SELECT count(customer_id) AS customer_count
from (
           SELECT customer_id,
                plan_id,
                row_number() OVER (
                     PARTITION BY customer_id
                     ORDER BY plan_id
                ) AS rn
           FROM subs_plans
           WHERE extract(
                     YEAR
                     FROM start_date
                ) = '2020'
      ) AS tmp
WHERE rn != 1
      AND plan_id = 3;
```

84. How many days on average does it take for a customer to an annual plan from the day they join Foodie-Fi?

Answer-

```
DROP TABLE IF EXISTS get_join_date;
CREATE TEMP TABLE get_join_date AS (
      SELECT DISTINCT customer_id,
            min(start_date) AS join_date
      FROM subs_plans
      GROUP BY customer_id
      ORDER BY customer_id
);
DROP TABLE IF EXISTS get_aplan_date;
CREATE TEMP TABLE get_aplan_date AS (
      SELECT DISTINCT customer_id,
            max(start_date) AS aplan_date
      FROM subs_plans
      WHERE plan_id = 3
      GROUP BY customer_id
      ORDER BY customer_id
);
SELECT round(avg(ad.aplan_date - jd.join_date), 2) AS avg_days
FROM get_join_date AS jd
      JOIN get_aplan_date AS ad ON jd.customer_id = ad.customer_id;
```

85. How many customers downgraded from a pro monthly to a basic monthly plan in 2020?

Answer-

```
SELECT count(customer_id) AS cust_downgrade_count
from (
            SELECT customer_id,
                  plan_name,
                  lead(plan_name) OVER (
                        PARTITION BY customer_id
                        ORDER BY start_date
                  ) AS downgrade
            FROM subs_plans
            WHERE extract(
                        YEAR
                        FROM start_date
                  ) = '2020'
      ) AS tmp
WHERE plan_name = 'pro monthly'
```

```
      AND downgrade = 'basic monthly';
```

86.

```
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');
```

```sql
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');
```

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

Answer-

```sql
SELECT s.customer_id AS c_id,
      SUM(m.price) AS total_spent
FROM sales AS s
      JOIN menu AS m ON s.product_id = m.product_id
GROUP BY c_id
ORDER BY total_spent DESC;
```

87. How many days has each customer visited the restaurant?
Answer-

```sql
SELECT customer_id AS c_id,
      COUNT(DISTINCT order_date) AS n_days
FROM sales
GROUP BY customer_id
ORDER BY n_days DESC;
```

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

1. Create a CTE and join the sales and menu tables.
2. Use the row_number window function to give a unique row number to every item purchased by the customer.
3. Order the items by the order_date
4. Select customer_id and product_name for every item where the row_number is '1'

Answer-

```
WITH cte_first_order AS (
      SELECT s.customer_id AS c_id,
            m.product_name,
            ROW_NUMBER() OVER (
                  PARTITION BY s.customer_id
                  ORDER BY s.order_date,
                        s.product_id
            ) AS rn
      FROM sales AS s
            JOIN menu AS m ON s.product_id = m.product_id
)
SELECT c_id,
      product_name
FROM cte_first_order
WHERE rn = 1
```

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

Answer-

```
SELECT m.product_name,
      COUNT(s.product_id) AS n_purchased
FROM menu AS m
      JOIN sales AS s ON m.product_id = s.product_id
GROUP BY m.product_name
ORDER BY n_purchased DESC
LIMIT 1;
```

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

1. Create a CTE and join the sales and menu tables.
2. Use the rank window function to rank every item purchased by the customer.
3. Order the items by the numbers or times purchase in descending order (highest to lowest).
4. Select 'everything' for every item where the rank is '1'.

Answer-

```
WITH cte_most_popular AS (
     SELECT s.customer_id AS c_id,
            m.product_name AS p_name,
            RANK() OVER (
                 PARTITION BY customer_id
                 ORDER BY COUNT(m.product_id) DESC
            ) AS rnk
     FROM sales AS s
            JOIN menu AS m ON s.product_id = m.product_id
     GROUP BY c_id,
            p_name
)
SELECT *
FROM cte_most_popular
WHERE rnk = 1;
```

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

1. Create a CTE and join the sales and menu tables to the members table.
2. Use the rank window function to rank every item purchased by the customer.
3. Order the items by the numbers or times purchase in ascending order (lowest to highest).
4. Filter the results to orders made after the join date.
5. Select customer and product where rank = '1'.

Answer-

```
WITH cte_first_member_purchase AS (
     SELECT m.customer_id AS p,
            m2.product_name AS product,
            RANK() OVER (
                 PARTITION BY m.customer_id
                 ORDER BY s.order_date
            ) AS rnk
     FROM members AS m
            JOIN sales AS s ON s.customer_id = m.customer_id
            JOIN menu AS m2 ON s.product_id = m2.product_id
      WHERE s.order_date >= m.join_date
)
SELECT customer,
      product
FROM cte_first_member_purchase
WHERE rnk = 1;
```

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

1. Create a CTE and join the sales and menu tables to the members table.
2. Use the rank window function to rank every item purchased by the customer.
3. Order the items by the numbers or times purchase in descending order (highest to lowest).
4. Filter the results to orders made before the join date.
5. Select customer and product where rank = '1'.

Answer-

```
WITH cte_last_nonmember_purchase AS (
      SELECT m.customer_id AS customer,
            m2.product_name AS product,
            RANK() OVER (
                  PARTITION BY m.customer_id
                  ORDER BY s.order_date DESC
            ) AS rnk
      FROM members AS m
            JOIN sales AS s ON s.customer_id = m.customer_id
            JOIN menu AS m2 ON s.product_id = m2.product_id
      WHERE s.order_date < m.join_date
)
SELECT customer,
      product
FROM cte_last_nonmember_purchase
WHERE rnk = 1;
```

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

1. Create a CTE and join the sales and menu tables to the members table.
2. Get the customer_id, total number of items and the total amount spent.
3. Filter the results to orders made before the join date.
4. Group by the customer id.

Answer-

```
WITH cte_total_nonmember_purchase AS (
      SELECT m.customer_id AS customer,
            COUNT(m2.product_id) AS total_items,
            SUM(m2.price) AS total_spent
      FROM members AS m
            JOIN sales AS s ON s.customer_id = m.customer_id
            JOIN menu AS m2 ON s.product_id = m2.product_id
      WHERE s.order_date < m.join_date
      GROUP BY customer
)
SELECT *
FROM cte_total_nonmember_purchase
ORDER BY customer;
```

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

1. Create a CTE and join the sales and menu tables to the members table.
2. Use a case statement inside of the sum function to calculate total points including 2x multiplier.
3. Filter the results to orders made before the join date.
4. Group by the customer id.

Answer-

```
WITH cte_total_member_points AS (
      SELECT m.customer_id AS customer,
            SUM(
                  CASE
                        WHEN m2.product_name = 'sushi' THEN (m2.price
* 20)
                        ELSE (m2.price * 10)
                  END
            ) AS member_points
      FROM members AS m
```

```
            JOIN sales AS s ON s.customer_id = m.customer_id
            JOIN menu AS m2 ON s.product_id = m2.product_id
      GROUP BY customer
)
SELECT *
FROM cte_total_member_points
```

95. 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?

1. Create a CTE and join the sales and menu tables to the members table.
2. Use a case statement inside of the sum function to calculate total points including 2x multiplier.
3. If the order is after membship or within the 6 days after membership then use the 2x multiplier on all items. Else, only on sushi.
4. Filter the results to orders made in Jan 2021.
5. Group by the customer id.

Answer-

```
WITH cte_jan_member_points AS (
      SELECT m.customer_id AS customer,
            SUM(
                  CASE
                        WHEN s.order_date < m.join_date THEN
                              CASE
                                    WHEN m2.product_name = 'sushi'
THEN (m2.price * 20)
                                    ELSE (m2.price * 10)
                              END
                        WHEN s.order_date > (m.join_date + 6) THEN
                              CASE
                                    WHEN m2.product_name = 'sushi'
THEN (m2.price * 20)
                                    ELSE (m2.price * 10)
                              END
                        ELSE (m2.price * 20)
                  END
            ) AS member_points
      FROM members AS m
```

```
           JOIN sales AS s ON s.customer_id = m.customer_id
           JOIN menu AS m2 ON s.product_id = m2.product_id
      WHERE s.order_date <= '2021-01-31'
      GROUP BY customer
)
SELECT *
FROM cte_jan_member_points
ORDER BY customer;
```

96.

```
 CREATE TABLE regions (
  region_id INTEGER,
  region_name VARCHAR(9)
);

CREATE TABLE customer_nodes (
  customer_id INTEGER,
  region_id INTEGER,
  node_id INTEGER,
  start_date DATE,
  end_date DATE
);

CREATE TABLE customer_transactions (
  customer_id INTEGER,
  txn_date DATE,
  txn_type VARCHAR(10),
  txn_amount INTEGER
);
```

How many unique nodes are there on the Data Bank system?

Answer-

```
WITH region_node_count AS (
      SELECT
            region_id,
            count(DISTINCT node_id) AS n_nodes
      FROM
            customer_nodes
      GROUP BY
            region_id
)
SELECT
      sum(n_nodes) AS total_nodes
FROM
      region_node_count;
```

97. What is the number of nodes per region?
Answer-

```
SELECT
      r.region_name,
      count(DISTINCT cn.node_id) AS node_count
FROM customer_nodes AS cn
JOIN regions AS r ON r.region_id = cn.region_id
GROUP BY r.region_name;
```

98. How many customers are allocated to each region?
Answer-

```
SELECT
      r.region_name,
      count(DISTINCT cn.customer_id) AS customer_count
FROM customer_nodes AS cn
JOIN regions AS r ON r.region_id = cn.region_id
GROUP BY r.region_name;
```

99. How many days on average are customers reallocated to a different node?

- Note that we will exlude data from any record with 9999 end date.
- Note that we will NOT count when the node does not change from one start date to another.

Answer-

```
WITH get_start_and_end_dates as (
	SELECT
		customer_id,
		node_id,
		start_date,
		end_date,
		LAG(node_id) OVER (PARTITION BY customer_id ORDER BY 
start_date) AS prev_node
	FROM
		customer_nodes
	WHERE
		EXTRACT(YEAR FROM end_date) != '9999'
	ORDER BY
		customer_id,
		start_date
)
SELECT
	floor(avg(end_date - start_date)) AS rounded_down,
	round(avg(end_date - start_date), 1) AS avg_days,
	CEIL(avg(end_date - start_date)) AS rounded_up
FROM
	get_start_and_end_dates
WHERE
	prev_node != node_id;
```

100. What is the median, 80th and 95th percentile for this same reallocation days metric for each region?

Answer-

```
WITH get_all_days AS (
      SELECT
            r.region_name,
            cn.customer_id,
            cn.node_id,
            cn.start_date,
            cn.end_date,
            LAG(cn.node_id) OVER (PARTITION BY cn.customer_id ORDER
BY cn.start_date) AS prev_node
      FROM
            customer_nodes AS cn
      JOIN regions AS r
      ON r.region_id = cn.region_id
      WHERE
            EXTRACT(YEAR FROM cn.end_date) != '9999'
      ORDER BY
            cn.customer_id,
            cn.start_date
),
perc_reallocation AS (
SELECT
      region_name,
      PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY end_date -
start_date) AS "50th_perc",
      PERCENTILE_CONT(0.8) WITHIN GROUP(ORDER BY end_date -
start_date) AS "80th_perc",
      PERCENTILE_CONT(0.95) WITHIN GROUP(ORDER BY end_date -
start_date) AS "95th_perc"
FROM
      get_all_days
WHERE
      prev_node != node_id
GROUP BY
      region_name
)
SELECT
      region_name,
      CEIL("50th_perc") AS median,
      CEIL("80th_perc") AS "80th_percentile",
      CEIL("95th_perc") AS "95th_percentile"
FROM
      perc_reallocation;
```