Schema- Farmers Market Database
Link-https://www.interviewbit.com/problems/exploring-the-farmer-s-market/

From customer_purchases Table:

1. Calculate the total revenue (SUM(quantity * cost_to_customer_per_qty)) generated for each customer.

```
select cost_to_customer_per_qty, customer_id,

(sum(quantity*cost_to_customer_per_qty))as 'Total Revenue'

from customer_purchases

group by customer_id
```

2. Find the maximum cost_to_customer_per_qty for each product.

```
select cost_to_customer_per_qty, product_id,

max(cost_to_customer_per_qty) as 'max_quantity'

from customer_purchases group by product_id
```

3. Determine the average quantity purchased for each market date.

```
select quantity, market_date,

avg(quantity)

from customer_purchases group by market_date
```

4. Identify the minimum quantity purchased across all transactions.

```
select quantity, transaction_time,

min(quantity)

from customer_purchases
```

5. List the total sales (SUM(quantity * cost_to_customer_per_qty)) for each vendor.

```
select cost_to_customer_per_qty, vendor_id, product_id,

(sum(quantity*cost_to_customer_per_qty))as 'Total Sales'

from customer_purchases
```

6. Find the number of transactions where the total cost_to_customer_per_qty was greater than $50.

```
select count(transaction_time), sum(cost_to_customer_per_qty)

from customer_purchases

where cost_to_customer_per_qty >5
```

7. Calculate the total quantity sold for product_id = 2 using the SUM function.

```
select product_id,

sum(quantity)

from customer_purchases

where product_id = 2
```

8. List the top 5 products based on the highest quantity sold. Use ORDER BY and LIMIT.

```
select product_id, quantity

from customer_purchases

order by quantity desc limit 5
```

9. Find the average cost_to_customer_per_qty for vendor_id = 3.

```
select avg(cost_to_customer_per_qty)

from customer_purchases

where vendor_id = 3
```

10. Identify the total quantity sold across all market dates for each product. Use the HAVING clause to filter products with total sales greater than 100.

```
select sum(quantity),market_date, product_id

from customer_purchases

group by product_id

having (sum(quantity*cost_to_customer_per_qty)) > 100
```

From vendor Table:

11. Retrieve the list of vendors whose vendor_owner_last_name starts with 'S'. Use the WHERE clause.

```
select vendor_owner_last_name

from vendor

where vendor_owner_last_name like 'S%'
```

12. List all vendors ordered by vendor_name in descending order. Use ORDER BY.

```
select vendor_id, vendor_name

from vendor

order by vendor_name desc
```

13. Find the vendor whose vendor_name contains the word 'Sustainable'.

```
select vendor_id, vendor_name

from vendor

where vendor_name like '%Sustainable%'
```

14. Retrieve the vendor names that have a vendor_type as 'Fresh Variety'. Use the WHERE clause.

```
select vendor_name, vendor_type

from vendor

where vendor_type like '%Fresh Variety%'
```

15. List the first 3 vendors based on vendor_id. Use LIMIT.

```
select vendor_id, vendor_name

from vendor

order by vendor_id limit 3
```

16. Identify vendors whose vendor_owner_last_name ends with 'n'. Use the LIKE operator.

```
select vendor_id, vendor_owner_last_name, vendor_name

from vendor

where vendor_owner_last_name like '%n'
```

17. Retrieve vendor details where vendor_type contains the word 'Eggs'. Use WHERE with LIKE.

```
select vendor_id, vendor_name, vendor_type

from vendor

where vendor_type like '%Eggs%'
```

18. Find the total number of vendors whose vendor_owner_last_name starts with 'H'.

```
select count(vendor_id), vendor_name, vendor_owner_last_name

from vendor

where vendor_owner_last_name like 'H%'
```

19. Retrieve all vendors with vendor_id greater than 5. Use the WHERE clause.

```
select vendor_name,vendor_id

from vendor

where vendor_id > 5
```

20. Find the vendor whose vendor_name starts with 'M'. Use the LIKE operator.

```
select vendor_name,vendor_id

from vendor

where vendor_name like "M%"
```

From product Table:

21. List all products where product_qty_type is 'lbs'. Use WHERE.

```
select product_name, product_id, product_qty_type

from product

where product_qty_type like "%lbs%"
```

22. Retrieve the product names where product_name contains the word 'Organic'. Use WHERE.

```
select product_name, product_id

from product

where product_name like "%Organic%"
```

23. Find the minimum product size for all products.

```
select min(product_size)

from product

# output not coming
```

24. List products where the product_size is 'medium'. Use WHERE.

```
select product_name, product_size

from product

where product_size like "%medium%"
```

25. Retrieve the first 5 products ordered by product_id in ascending order. Use ORDER BY and LIMIT.

```
select product_name, product_id

from product

order by product_id limit 5
```

26. Find the maximum product size for products in the 'Fresh Fruits & Vegetables' category.

    —--> have to discuss this.

27. List products where product_qty_type is 'unit'. Use WHERE.

```
select product_name, product_id, product_qty_type

from product

where product_qty_type like "%unit%"
```

28. Identify all products with product_name starting with 'H'. Use WHERE with LIKE.

```
select product_name, product_id

from product

where product_name like "H%"
```

29. Find the total number of products in the 'Packaged Pantry Goods' category.

— —>> **Have to discuss**

30. List all products in ascending order of product_name. Use ORDER BY.

```
select product_id, product_name

from product

order by product_name
```

From vendor_inventory Table:

31. Calculate the total quantity available for vendor_id = 4 using the SUM function.

```
select sum(quantity), vendor_id

from vendor_inventory

where vendor_id = 4
```

32. Find the average original price for products sold by vendor_id = 6.

```
select avg(original_price)

from vendor_inventory

where vendor_id = 6
```

33. Identify the maximum original price for product_id = 1.

```
select max(original_price), product_id

from vendor_inventory

where product_id = 1
```

34. List all products with an original price greater than $10. Use the WHERE clause.

```
select product_id, original_price

from vendor_inventory

where original_price > 10

group by product_id
```

35. Retrieve the first 3 products with the highest original_price. Use ORDER BY and LIMIT.

```
select product_id, original_price

from vendor_inventory

order by original_price desc

limit 3
```

36. Calculate the total quantity sold for each vendor. Use GROUP BY and SUM.

```


select sum(quantity), vendor_id

from vendor_inventory

group by vendor_id
```

37. Find the minimum original price for products sold on market_date = '2019-07-03'.

```
select min(original_price), product_id, market_date

from vendor_inventory

where market_date = '2019-07-03'
```

```
select min(original_price), product_id, market_date

from vendor_inventory

where market_date like "%2019-07-03%"
```

38. Identify the average original price for all products using the AVG function.

```
select avg(original_price), product_id

from vendor_inventory

group by product_id
```

39. Find the total number of products sold by vendor_id = 7. Use WHERE and COUNT.

```
select count(product_id), vendor_id

from vendor_inventory

where vendor_id = 7
```

40. Retrieve the list of products where original_price is less than $5. Use WHERE.

```
select product_id, original_price

from vendor_inventory

where original_price < 5

group by product_id
```

From market_date_info Table:

41. List all market dates where the market_rain_flag is 1 (rainy days). Use WHERE.

```
select market_date, market_rain_flag

from market_date_info

where market_rain_flag = 1
```

42. Find the maximum market_max_temp for all market dates.

```
select market_date, max(market_max_temp)

from market_date_info

group by market_date
```

43. Retrieve all market dates where market_snow_flag is 1. Use WHERE.

```
select market_date, market_snow_flag

from market_date_info

where market_snow_flag = 1

group by market_date
```

44. List the first 5 market dates ordered by market_min_temp in descending order. Use ORDER BY and LIMIT.

```
select market_date,market_min_temp

from market_date_info

order by market_min_temp desc limit 5
```

45. Find the average market_max_temp for all market dates.

```


select market_date, avg(market_max_temp)

from market_date_info

group by market_date
```

46. List all market dates where market_max_temp is greater than 40 degrees. Use WHERE.

```
select market_date, market_max_temp

from market_date_info

where market_max_temp > 40
```

47. Identify the minimum market_min_temp for each market_season. Use GROUP BY and MIN.

```
select min(market_min_temp), market_season

from market_date_info

group by market_season
```

48. Find the total number of market dates where the market_day is 'Saturday'. Use WHERE.

```
select count(market_date), market_day

from market_date_info

where market_day = 'Saturday'
```

49. List the market dates where the market_start_time is '8:00 AM'. Use WHERE.

```
select market_date, market_start_time

from market_date_info

where market_start_time = '8:00 AM'
```

50. Retrieve all market dates ordered by market_year in ascending order. Use ORDER BY.

```
select market_date, market_year

from market_date_info

order by market_year
```

From customer_purchases Table:

1. **Use a CASE statement to categorize transactions where the total cost (quantity * cost_to_customer_per_qty) is greater than $50 as 'High' and others as 'Low'.**
   Table: customer_purchases

```
select transaction_time, (quantity* cost_to_customer_per_qty) as
'total sales',

case when (quantity* cost_to_customer_per_qty) > 25 then 'high'

else 'low'

end as 'rating'

from customer_purchases
```

2. **Write a query that uses a CASE statement to categorize products as 'Large Quantity' if the quantity sold is greater than 5, otherwise 'Small Quantity'.**
   Table: customer_purchases

```
select quantity, product_id,

case when quantity > 5 then 'Large Quantity'

else 'Small Quantity'

end as 'large/small'

from customer_purchases
```

3. **Use a CASE statement to classify vendors into 'High Sales' or 'Low Sales' based on whether their total sales (quantity * cost_to_customer_per_qty) exceed $1000.**
   Table: customer_purchases

```
select vendor_id, (quantity* cost_to_customer_per_qty) as 'total
sales',

case when (quantity* cost_to_customer_per_qty) > 1000 then 'high
sales'

else 'low sales'

end as 'high/Low'

from customer_purchases
```

have to discuss

4. **Use a CASE statement to assign 'High', 'Medium', or 'Low' based on the cost_to_customer_per_qty, where > $10 is 'High', between $5 and $10 is 'Medium', and below $5 is 'Low'.**
   Table: customer_purchases

```
select vendor_id, cost_to_customer_per_qty,

case when cost_to_customer_per_qty > 10 then 'High'

when cost_to_customer_per_qty >=5 and cost_to_customer_per_qty
<10 then 'Medium'

else 'low'

end as 'c_t_c'

from customer_purchases
```

5. **Write a query using a CASE statement to return 'Weekend' if the market_date falls on a Saturday or Sunday, and 'Weekday' otherwise.**
   Table: customer_purchases

```
select market_date,

case when market_date = 'Saturday' or market_date = 'Sunday' then
'Weekend'

else 'Weekday'

end as 'type of day'

from customer_purchases
```

From vendor_inventory Table:

6. **Use a CASE statement to classify products as 'Expensive' if the original price is greater than $15, otherwise 'Affordable'.**
   Table: vendor_inventory

```
select product_id, original_price,

case when original_price > 15 then 'Expensive'

else 'Affordable'

end as 'type of price'

from vendor_inventory

group by product_id
```

7. **Write a query that uses a CASE statement to classify vendors as 'Large Stock' or 'Small Stock' based on whether the total quantity available exceeds 20 units.**
   Table: vendor_inventory

```
select vendor_id, quantity,

case when quantity > 20 then 'Large Stock'

else 'Small Stock'

end as 'type of stock'

from vendor_inventory
```

8. **Use a CASE statement to return 'Hot Day' if the market_max_temp is greater than 35°C, otherwise return 'Cool Day'.**
   Table: market_date_info

```
select market_date, market_day, market_max_temp,

case when market_max_temp > 35 then 'Hot Day'

else 'Cool Day'

end as 'type of day'

from market_date_info
```

9. **Write a query using a CASE statement to flag products with prices higher than $20 as 'Premium', between $10 and $20 as 'Standard', and below $10 as 'Budget'.**
   Table: vendor_inventory

```
select product_id, original_price,

case when original_price > 20 then 'Premium'

when original_price >= 10 and original_price <= 20 then
'Standard'

else 'Budget'

end as 'type of price'

from vendor_inventory
```

10. **Use a CASE statement to categorize dates as 'Rainy' if market_rain_flag = 1, 'Snowy' if market_snow_flag = 1, and 'Clear' otherwise.**
    Table: market_date_info

```sql
select market_date, market_rain_flag, market_snow_flag,

case when market_rain_flag = 1 then 'Rainy'

when market_snow_flag = 1 then 'Snowy'

else 'Clear'

end as 'type of day'

from market_date_info
```