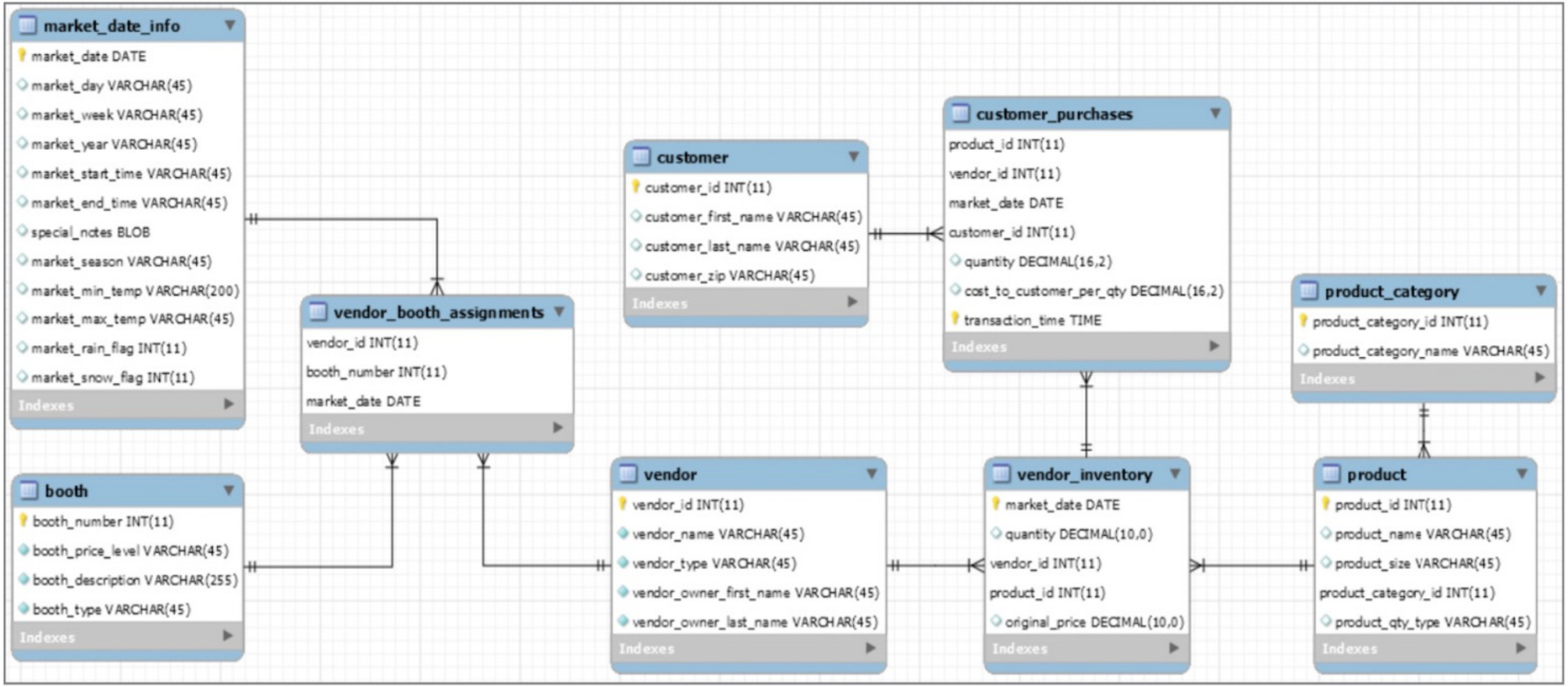
SQL-Group BY

Quiz

- = is equi join and >=, <=, >, <, between are non-equi join?
  - True - correct
  - False
- Table1 contains products the customer has already purchased. Table2 is products table. Which type of join to use to recommend products the customer does not already have
  - Select * from DB.tbl1 = DB.tbl2 on product_id = product_id
  - Select * from DB.tbl1 = DB.tbl2 on product_id > product_id
  - Select * from DB.tbl1 = DB.tbl2 on product_id <> product_id - correct
  - Select * from DB.tbl1 = DB.tbl2 on product_id <= product_id
- As long as the column numbers/datatypes are matching we can union two tables even though the order of columns are different i.e ‘country’, ‘Fname’, ‘qualification’ can be unioned with ‘qualification’, ‘country’, ‘Fname’
  - True - correct
  - False
- UNION_distinct means combine two tables and output only distinct rows. UNION_ALL means combine 2 tables even with duplicated as output?
  - True - correct
  - False
- Union is horizontal growth and Joins is vertical growth?
  - True
  - False – correct
- Table_1.col = [1,2,3], Table_2.col = [2,3]. What is the resultant number of rows because of the this query?

  Select * from tbl_1 join tbl_2 on tbl_1.col1>tbl_2.col1
  - 1 - correct
  - 2
  - 3
  - 0

Question: Get a list of the customer IDs who made purchases on each market date.



market_date_info
market_date DATE
market_day VARCHAR(45)
market_week VARCHAR(45)
market_year VARCHAR(45)
market_start_time VARCHAR(45)
market_end_time VARCHAR(45)
special_notes BLOB
market_season VARCHAR(45)
market_min_temp VARCHAR(200)
market_max_temp VARCHAR(45)
market_rain_flag INT(11)
market_snow_flag INT(11)
Indexes

customer
customer_id INT(11)
customer_first_name VARCHAR(45)
customer_last_name VARCHAR(45)
customer_zip VARCHAR(45)
Indexes

customer_purchases
product_id INT(11)
vendor_id INT(11)
market_date DATE
customer_id INT(11)
quantity DECIMAL(16,2)
cost_to_customer_per_qty DECIMAL(16,2)
transaction_time TIME
Indexes

vendor_booth_assignments
vendor_id INT(11)
booth_number INT(11)
market_date DATE
Indexes

product_category
product_category_id INT(11)
product_category_name VARCHAR(45)
Indexes

booth
booth_number INT(11)
booth_price_level VARCHAR(45)
booth_description VARCHAR(255)
booth_type VARCHAR(45)
Indexes

vendor
vendor_id INT(11)
vendor_name VARCHAR(45)
vendor_type VARCHAR(45)
vendor_owner_first_name VARCHAR(45)
vendor_owner_last_name VARCHAR(45)
Indexes

vendor_inventory
market_date DATE
quantity DECIMAL(10,0)
vendor_id INT(11)
product_id INT(11)
original_price DECIMAL(10,0)
Indexes

product
product_id INT(11)
product_name VARCHAR(45)
product_size VARCHAR(45)
product_category_id INT(11)
product_qty_type VARCHAR(45)
Indexes

Question: Get a list of the customers who made purchases on each market date.

```
SELECT
    market_date,
    customer_id
 FROM farmers_market.customer_purchases
 ORDER BY market_date, customer_id;
```

Question: Count the number of purchases each customer made per market date.

```
SELECT
     market_date,
     customer_id,
     COUNT(*) AS num_purchases
  FROM farmers_market.customer_purchases
  GROUP BY market_date, customer_id
```

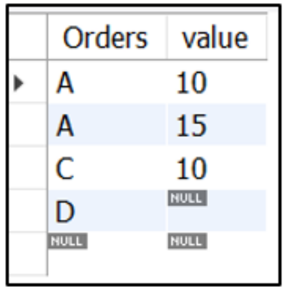
Alternate Question: Calculate the total quantity purchased by each customer per market_date.

```
SELECT
    market_date,
    customer_id,
    SUM(quantity) AS total_qty_purchased
 FROM farmers_market.customer_purchases
 GROUP BY market_date, customer_id
```

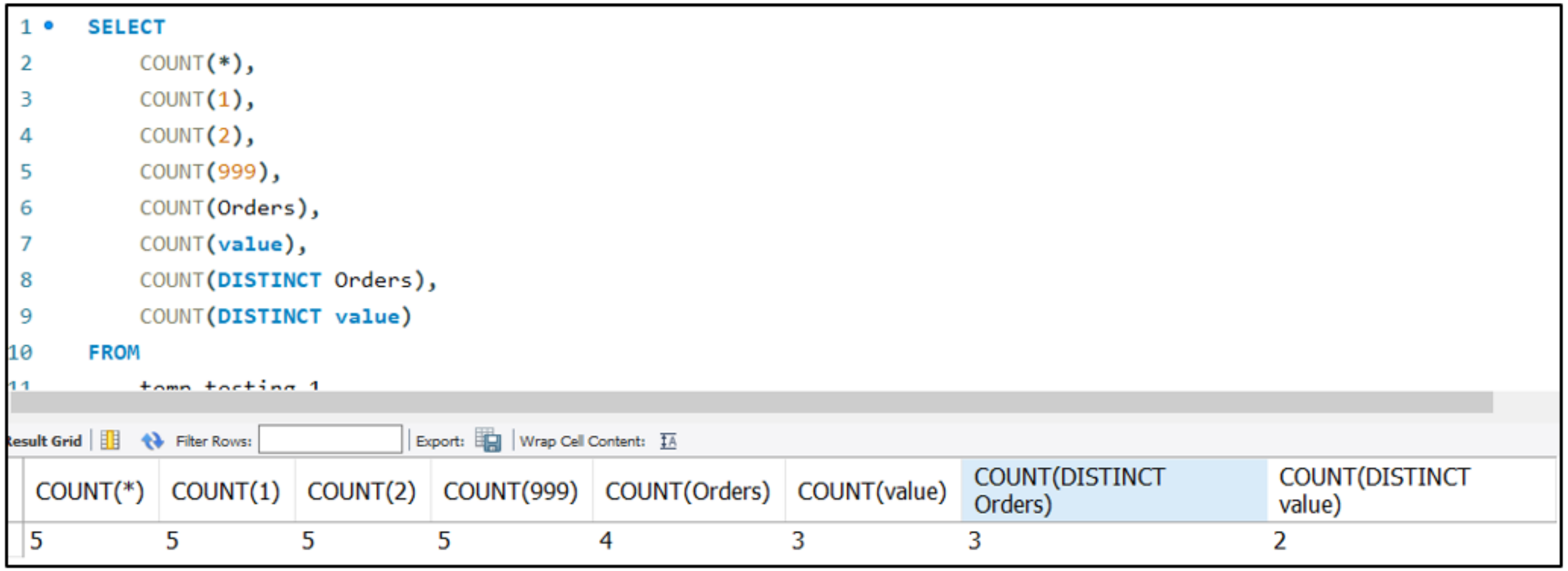
Slightly complex question: how many different kinds of products were purchased by each customer on each market date:

```
SELECT
    market_date,
    customer_id,
    COUNT(DISTINCT product_id) AS
different_products_purchased
FROM farmers_market.customer_purchases
GROUP BY market_date, customer_id
ORDER BY market_date;
```

How count () works



Orders	value
A	10
A	15
C	10
D	NULL
NULL	NULL



```sql
SELECT
    COUNT(*),
    COUNT(1),
    COUNT(2),
    COUNT(999),
    COUNT(Orders),
    COUNT(value),
    COUNT(DISTINCT Orders),
    COUNT(DISTINCT value)
FROM
    temp_testing_1
```

COUNT(*)	COUNT(1)	COUNT(2)	COUNT(999)	COUNT(Orders)	COUNT(value)	COUNT(DISTINCT Orders)	COUNT(DISTINCT value)
5	5	5	5	4	3	3	2

Question: Calculate the total price paid by customer_id 3 per market_date.

```
SELECT
        market_date,
        SUM(quantity *
cost_to_customer_per_qty) AS total_spent
FROM farmers_market.customer_purchases
WHERE
      customer_id = 3
GROUP BY market_date
ORDER BY market_date;
```

Question: What if we wanted to find out how much each customer had spent at each vendor, regardless of date?

```
SELECT
    customer_id,
    vendor_id,
    SUM(quantity * cost_to_customer_per_qty) AS total_spent
FROM farmers_market.customer_purchases
GROUP BY customer_id, vendor_id
ORDER BY customer_id, vendor_id;
```

Let’s add some customer details and vendor details to these results. Customer details are in the customer table and vendor details are in the vendor table.

```
SELECT
    c.customer_first_name,
    c.customer_last_name,
    cp.customer_id,
    v.vendor_name,
    cp.vendor_id,
    ROUND(SUM(quantity * cost_to_customer_per_qty), 2) AS total_spent
 FROM farmers_market.customer c
    LEFT JOIN farmers_market.customer_purchases cp
       ON c.customer_id = cp.customer_id
    LEFT JOIN farmers_market.vendor v
       ON cp.vendor_id = v.vendor_id
 GROUP BY
    cp.customer_id,
    cp.vendor_id
 ORDER BY cp.customer_id, cp.vendor_id;
```

Question: We want to get the most and least expensive items per product category, considering the fact that each vendor sets their own prices and can adjust prices per customer.

```
SELECT
    p.product_category_id,
    MIN(vi.original_price) AS minimum_price,
    MAX(vi.original_price) AS maximum_price
 FROM farmers_market.vendor_inventory AS vi
    INNER JOIN farmers_market.product AS p
       ON vi.product_id = p.product_id
 GROUP BY p.product_category_id;
```

Question: Count how many products(can be duplicates) were for sale on each market date

```
SELECT
    market_date,
    COUNT(product_id) AS product_count
FROM farmers_market.vendor_inventory
GROUP BY market_date
ORDER BY market_date;
```

how many different products each vendor offered. **between** - 2019-05-02 and 2019-05-16?

```
SELECT
    vendor_id,
    COUNT(DISTINCT product_id) AS different_products_offered
FROM farmers_market.vendor_inventory
WHERE
     market_date BETWEEN '2019-05-02' AND '2019-05-16'
GROUP BY vendor_id
ORDER BY vendor_id;
```

Filter out vendors who brought at least 100 items from the farmer’s market over the period - 2019-05-02 and 2022-05-16.

Using subqueries

```
select x.vendor_id, x.products_brought from
(
    select vendor_id, count(product_id) as products_brought
    from `farmers_market.vendor_inventory`
    where market_date between "2019-04-02" and "2022-05-16"
    group by vendor_id
) as x
where x.products_brought>100
```

Using Having clause

```
SELECT
    vendor_id,
    count(product_id) AS products_brought,
 FROM farmers_market.vendor_inventory
 WHERE market_date BETWEEN '2019-04-03' AND '2019-05-16’
        GROUP BY vendor_id
 HAVING products_brought >= 100
 ORDER BY vendor_id;
```

Order of execution

- FROM, including JOINs
- WHERE
- GROUP BY
- HAVING
- WINDOW functions
- SELECT
- DISTINCT
- UNION
- ORDER BY
- OFFSET
- LIMIT