**1.** Assume you are given the following tables on Walmart transactions and products. Find the number of **unique** product combinations that are purchased in the same transaction.

For example, if there are 2 transactions where apples and bananas are bought, and another transaction where bananas and soy milk are bought, my output would be 2 to represent the 2 **unique** combinations.

Assumptions:

* For each transaction, a maximum of 2 products is purchased.
* You may or may not need to use the products table.

*P.S. Solution is updated as of 1 Feb 2023.*

**transactions Table:**

| **Column Name** | **Type** |
| --- | --- |
| transaction\_id | integer |
| product\_id | integer |
| user\_id | integer |
| transaction\_date | datetime |

**transactions Example Input:**

| **transaction\_id** | **product\_id** | **user\_id** | **transaction\_date** |
| --- | --- | --- | --- |
| 231574 | 111 | 234 | 03/01/2022 12:00:00 |
| 231574 | 444 | 234 | 03/01/2022 12:00:00 |
| 231574 | 222 | 234 | 03/01/2022 12:00:00 |
| 137124 | 111 | 125 | 03/05/2022 12:00:00 |
| 137124 | 444 | 125 | 03/05/2022 12:00:00 |

**products Table:**

| **Column Name** | **Type** |
| --- | --- |
| product\_id | integer |
| product\_name | string |

**products Example Input:**

| **product\_id** | **product\_name** |
| --- | --- |
| 111 | apple |
| 222 | soy milk |
| 333 | instant oatmeal |
| 444 | banana |
| 555 | chia seed |

Sol:- SELECT count(distinct (t1.product\_id, t2.product\_id)) as

unique\_product

FROM transactions t1

JOIN transactions t2

ON t1.transaction\_id = t2.transaction\_id

AND t1.product\_id < t2.product\_id; ##t1.product\_id < t2.product\_id ensures that the product IDs are unique pairs.

**2.** A Microsoft Azure Supercloud customer is a company which buys at least 1 product from each product category.

Write a query to report the company ID which is a Supercloud customer.

As of 5 Dec 2022, data in the *customer\_contracts* and *products* tables were updated.

### customer\_contracts Table:

| **Column Name** | **Type** |
| --- | --- |
| customer\_id | integer |
| product\_id | integer |
| amount | integer |

### customer\_contracts Example Input:

| **customer\_id** | **product\_id** | **amount** |
| --- | --- | --- |
| 1 | 1 | 1000 |
| 1 | 3 | 2000 |
| 1 | 5 | 1500 |
| 2 | 2 | 3000 |
| 2 | 6 | 2000 |

### products Table:

| **Column Name** | **Type** |
| --- | --- |
| product\_id | integer |
| product\_category | string |
| product\_name | string |

### products Example Input:

| **product\_id** | **product\_category** | **product\_name** |
| --- | --- | --- |
| 1 | Analytics | Azure Databricks |
| 2 | Analytics | Azure Stream Analytics |
| 4 | Containers | Azure Kubernetes Service |
| 5 | Containers | Azure Service Fabric |
| 6 | Compute | Virtual Machines |
| 7 | Compute | Azure Functions |

### Example Output:

| **customer\_id** |
| --- |
| 1 |

Sol:-

WITH supercloud AS (

SELECT

customers.customer\_id,

COUNT(DISTINCT products.product\_category) as unique\_count

FROM customer\_contracts AS customers

LEFT JOIN products

ON customers.product\_id = products.product\_id

GROUP BY customers.customer\_id

)

SELECT customer\_id,unique\_count

FROM supercloud

WHERE unique\_count = (

SELECT COUNT(DISTINCT product\_category)

FROM products)

ORDER BY customer\_id;

**3.** Assume you are given the table containing measurement values obtained from a Google sensor over several days. Measurements are taken several times within a given day.

Write a query to obtain the sum of the odd-numbered and even-numbered measurements on a particular day, in two different columns. Refer to the Example Output below for the output format.

Definition:

* 1st, 3rd, and 5th measurements taken **within a day** are considered odd-numbered measurements and the 2nd, 4th, and 6th measurements are even-numbered measurements.

**measurements Table:**

| **Column Name** | **Type** |
| --- | --- |
| measurement\_id | integer |
| measurement\_value | decimal |
| measurement\_time | datetime |

**measurements Example Input:**

| **measurement\_id** | **measurement\_value** | **measurement\_time** |
| --- | --- | --- |
| 131233 | 1109.51 | 07/10/2022 09:00:00 |
| 135211 | 1662.74 | 07/10/2022 11:00:00 |
| 523542 | 1246.24 | 07/10/2022 13:15:00 |
| 143562 | 1124.50 | 07/11/2022 15:00:00 |
| 346462 | 1234.14 | 07/11/2022 16:45:00 |

Sol’n :- with cte as(SELECT

measurement\_time,

CAST(measurement\_time AS DATE) AS measurement\_day,

measurement\_value,

row\_number() OVER(PARTITION BY CAST(measurement\_time AS DATE )

ORDER BY measurement\_time) as measurement\_num

from measurements)

select measurement\_day,

SUM(

CASE when measurement\_num %2 !=0 THEN measurement\_value ELSE 0

END) as odd\_sum,

SUM(

CASE WHEN measurement\_num %2 =0 THEN measurement\_value ELSE 0

END) as even\_sum

from cte

GROUP BY measurement\_day

**4.** Assume you are given the table on Walmart user transactions. Based on a user's most recent transaction date, write a query to obtain the users and the number of products bought.

Output the user's most recent transaction date, user ID and the number of products sorted by the transaction date in chronological order.

P.S. As of 10 Nov 2022, the official solution was changed from output of the transaction date, number of users and number of products to the current output.

### user\_transactions Table:

| **Column Name** | **Type** |
| --- | --- |
| product\_id | integer |
| user\_id | integer |
| spend | decimal |
| transaction\_date | timestamp |

### user\_transactions Example Input:

| **product\_id** | **user\_id** | **spend** | **transaction\_date** |
| --- | --- | --- | --- |
| 3673 | 123 | 68.90 | 07/08/2022 12:00:00 |
| 9623 | 123 | 274.10 | 07/08/2022 12:00:00 |
| 1467 | 115 | 19.90 | 07/08/2022 12:00:00 |
| 2513 | 159 | 25.00 | 07/08/2022 12:00:00 |
| 1452 | 159 | 74.50 | 07/10/2022 12:00:00 |

Sol’n - with cte as ( SELECT user\_id,product\_id,transaction\_date,

RANK() OVER( PARTITION BY user\_id order by transaction\_date DESC) as days\_rank

from user\_transactions)

select transaction\_date,user\_id, COUNT(product\_id) as purchase\_count

from cte where days\_rank =1

group by user\_id,transaction\_date

ORDER BY transaction\_date

**5.** Assume you are given the table on Walmart user transactions. Based on a user's most recent transaction date, write a query to obtain the users and the number of products bought.

Output the user's most recent transaction date, user ID and the number of products sorted by the transaction date in chronological order.

P.S. As of 10 Nov 2022, the official solution was changed from output of the transaction date, number of users and number of products to the current output.

### user\_transactions Table:

| **Column Name** | **Type** |
| --- | --- |
| product\_id | integer |
| user\_id | integer |
| spend | decimal |
| transaction\_date | timestamp |

### user\_transactions Example Input:

| **product\_id** | **user\_id** | **spend** | **transaction\_date** |
| --- | --- | --- | --- |
| 3673 | 123 | 68.90 | 07/08/2022 12:00:00 |
| 9623 | 123 | 274.10 | 07/08/2022 12:00:00 |
| 1467 | 115 | 19.90 | 07/08/2022 12:00:00 |
| 2513 | 159 | 25.00 | 07/08/2022 12:00:00 |
| 1452 | 159 | 74.50 | 07/10/2022 12:00:00 |

Sol’n :-

with cte as ( SELECT user\_id,product\_id,transaction\_date,

RANK() OVER( PARTITION BY user\_id order by transaction\_date DESC) as days\_rank

from user\_transactions)

select transaction\_date,user\_id, COUNT(product\_id) as purchase\_count

from cte where days\_rank =1

group by user\_id,transaction\_date

ORDER BY transaction\_date

**6.** You are trying to find the most common (aka the mode) number of items bought per order on Alibaba.

However, instead of doing analytics on all Alibaba orders, you have access to a summary table, which describes how many items were in an order (item\_count), and the number of orders that had that many items (order\_occurrences).

In case of multiple item counts, display the item\_counts in ascending order.

### items\_per\_order Table:

| **Column Name** | **Type** |
| --- | --- |
| item\_count | integer |
| order\_occurrences | integer |

### items\_per\_order Example Input:

| **item\_count** | **order\_occurrences** |
| --- | --- |
| 1 | 500 |
| 2 | 1000 |
| 3 | 800 |
| 4 | 1000 |

### Example Output:

| **mode** |
| --- |
| 2 |
| 4 |

Sol’n :- SELECT item\_count

FROM items\_per\_order

WHERE order\_occurrences =

(SELECT MAX(order\_occurrences) FROM items\_per\_order)

ORDER BY item\_count

**7.** Your team at JPMorgan Chase is soon launching a new credit card. You are asked to estimate how many cards you'll issue in the first month.

Before you can answer this question, you want to first get some perspective on how well new credit card launches typically do in their first month.

Write a query that outputs the name of the credit card, and how many cards were issued in its launch month. The launch month is the earliest record in the monthly\_cards\_issued table for a given card. Order the results starting from the biggest issued amount.

### monthly\_cards\_issued Table:

| **Column Name** | **Type** |
| --- | --- |
| issue\_month | integer |
| issue\_year | integer |
| card\_name | string |
| issued\_amount | integer |

### monthly\_cards\_issued Example Input:

| **issue\_month** | **issue\_year** | **card\_name** | **issued\_amount** |
| --- | --- | --- | --- |
| 1 | 2021 | Chase Sapphire Reserve | 170000 |
| 2 | 2021 | Chase Sapphire Reserve | 175000 |
| 3 | 2021 | Chase Sapphire Reserve | 180000 |
| 3 | 2021 | Chase Freedom Flex | 65000 |
| 4 | 2021 | Chase Freedom Flex | 70000 |

### Example Output:

| **card\_name** | **issued\_amount** |
| --- | --- |
| Chase Sapphire Reserve | 170000 |
| Chase Freedom Flex | 65000 |

Sol’n :- with card\_launch as(select card\_name,issued\_amount,make\_date(issue\_year, issue\_month, 1) as issue\_date,

MIN(make\_date(issue\_year,issue\_month,1)) OVER (PARTITION BY card\_name) as launch\_date

from monthly\_cards\_issued)

select card\_name,issued\_amount from card\_launch where launch\_date = issue\_date order by issued\_amount desc