**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