Q1. We are trying to find paired products that are often purchased together by the same user, such as chips and soft drinks, milk and curd etc.. Find the top paired products names.
(Tables: Orderdetails, Products)

```sql
SELECT *
FROM (
  SELECT a.ProductID as original_ID, b.ProductID as bought_with,count (*) AS num
  FROM OrderDetails A
  INNER join OrderDetails B
  ON a.orderid = b.orderid  AND a.ProductID > b.ProductID
  GROUP BY a.ProductID,b.ProductID) c
ORDER BY num DESC
```

115 %

Results | Messages | Client Statistics

	original_ID	bought_with	num
1	15616	4211	2
2	15353	4949	2
3	19433	10303	2
4	20504	4735	2
5	17780	4735	2
6	12110	4735	2
7	12863	4735	2
8	25986	2476	2
9	21900	21605	2
10	25852	23808	2
11	20420	19333	2
12	4735	706	2
13	4735	732	2
14	10624	2792	2
15	4735	881	2
16	21996	4387	1
17	9256	6135	1
18	12893	5793	1
19	12101	2690	1
20	26098	22639	1

Query executed successfully. BYOD-6

Q2. We want to understand the impact of running a campaign during July'21-Oct'21 what was the total sales generated for the categories "Beauty & Hygiene" and "Bevarages" by entire customer base. (Tables: Orders, Orderdetails, Products, Category)

```sql
CREATE VIEW Z AS (
SELECT A.OrderID,D.CategoryID,(B.Quantity * A.Total_order_amount) AS sales
FROM Orders A
JOIN OrderDetails B ON A.OrderID=B.OrderID
JOIN Products C ON B.ProductID=C.ProductID
JOIN Category D ON C.Category_ID=D.CategoryID
WHERE (MONTH(A.OrderDate) BETWEEN '07' AND '10') AND (YEAR(A.OrderDate)='2021') AND
D.CategoryName in ('Beauty & Hygiene' ,'Beverages'));

SELECT * FROM Z
```

115 %

Results | Messages | Client Statistics

	OrderID	CategoryID	sales
1	7657527	5008	340921.84765625
2	7657530	5008	127752.802734375
3	7657530	5008	63876.4013671875
4	7657531	5008	24003
5	7657532	5008	564
6	7657533	5008	297904
7	7657534	5008	1931051.640625
8	7657534	5008	508171.484375
9	7657534	5008	2032685.9375
10	7657535	5008	16933
11	7657536	5008	163126.702148438
12	7657536	5009	192786.102539063
13	7657536	5008	118637.6015625
14	7657537	5009	192762.744140625
15	7657537	5008	154210.1953125
16	7657537	5008	244166.142578125
17	7657539	5008	33723
18	7657544	5008	917844
19	7657545	5008	155326.5
20	7657545	5008	172585
21	7657545	5008	34517
22	7657546	5008	62992.16015625
23	7657546	5009	157480.400390625

Q3. Create a YOY analysis for the count of customers enrolled with the company each month. (Tables: Customers)

```sql
select Months, sum([2020]) AS Year_2020,sum([2021]) as Year_2021
from
(Select month(DateEntered) As Months , [2020] , [2021]
from (Select * , Year(DateEntered) As year_  From Customers) A
Pivot
(Count(year_) for year_ in ([2020], [2021]) ) As pivt) B
group by B.Months
```

115 %

Results | Messages

	Months	Year_2020	Year_2021
1	1	10	40
2	2	23	41
3	3	30	39
4	4	30	25
5	5	21	23
6	6	20	23
7	7	10	20
8	8	15	22
9	9	22	20
10	10	18	23
11	11	10	20
12	12	10	10

Q4. FInd out the difference between the last two order dates for each of the customers and categorize the customers in two categories such that if the difference is less than 5 days tag the customer as “Frequent Buyer” else tag it as “Infrequent”.
(Tables: Orders)

```sql
CREATE VIEW abc
AS (
SELECT * FROM ( SELECT CustomerID,OrderID,OrderDate,Rank_
FROM (
 SELECT *,RANK() OVER (PARTITION BY CustomerID ORDER BY OrderDate DESC) Rank_ FROM Orders ) C
GROUP BY CustomerID,OrderID,OrderDate,Rank_ ) D
 WHERE Rank_= 1 OR Rank_=2 );

SELECT A.CustomerID, A.OrderDate AS second_last_ordr, B.OrderDate AS last_order,
DATEDIFF(DAY,A.OrderDate,B.OrderDate) AS days_diff,
CASE WHEN DATEDIFF(DAY,A.OrderDate,B.OrderDate)<5 THEN 'Frequent Buyer'
ELSE 'Infrequent' END AS Category_
FROM abc AS A
JOIN abc AS B ON A.CustomerID=B.CustomerID
AND A.OrderDate<B.OrderDate
ORDER BY Category_,days_diff
```

115 %

Results | Messages | Client Statistics

	CustomerID	second_last_ordr	last_order	days_diff	Category_
1	57089	2021-11-25	2021-11-26	1	Frequent Buyer
2	57135	2021-10-25	2021-10-26	1	Frequent Buyer
3	57138	2021-11-09	2021-11-10	1	Frequent Buyer
4	57171	2020-11-14	2020-11-15	1	Frequent Buyer
5	57190	2021-12-27	2021-12-28	1	Frequent Buyer
6	57193	2021-10-25	2021-10-26	1	Frequent Buyer
7	57241	2021-12-24	2021-12-25	1	Frequent Buyer
8	57269	2021-12-25	2021-12-26	1	Frequent Buyer
9	57276	2021-10-19	2021-10-20	1	Frequent Buyer
10	57282	2021-11-22	2021-11-23	1	Frequent Buyer
11	57313	2021-11-26	2021-11-27	1	Frequent Buyer
12	57341	2021-10-24	2021-10-25	1	Frequent Buyer
13	57381	2021-10-12	2021-10-13	1	Frequent Buyer
14	57393	2021-12-19	2021-12-20	1	Frequent Buyer

Query executed successfully.

Q5. Create a Procedure to filter the orders from Orders table where total purchase amount in a each order id < @x and purchase of year is @Y where @x and @y are the inputs provided by the user.
(Tables: Orders)

```sql
CREATE PROCEDURE sp_proc_name(@x FLOAT,@Y DATE)
AS
BEGIN
      SELECT *
      FROM
          Orders
      WHERE
          total_order_amount<@x AND
          orderdate = @Y
END;
EXEC sp_proc_name 522230,'2022-09-22';
```

115 %

Messages | Client Statistics

```
Commands completed successfully.

Completion time: 2022-03-11T23:27:03.5481015+05:30
```

Q6. Find the top 3 Shipper companies in terms of average delivery time for each category for the latest year(Tables: Orders, Shippers)

```sql
--a.Average delivery time for each category for the latest year
SELECT * FROM (
SELECT  B.CompanyName,D.Category_ID,B.shipperID, AVG(DATEDIFF(DAY,OrderDate,DeliveryDate)) avg_delivery_time,
DENSE_RANK() OVER(PARTITION BY Category_ID ORDER BY  AVG(DATEDIFF(DAY,OrderDate,DeliveryDate))) AS _rank_
FROM Orders A
LEFT JOIN Shippers B ON A.ShipperID=B.ShipperID
LEFT JOIN OrderDetails C ON A.OrderID=C.OrderID
LEFT JOIN Products D ON C.ProductID=D.ProductID
LEFT JOIN Category E ON D.Category_ID=E.CategoryID
WHERE YEAR(OrderDate)=2021
GROUP BY B.CompanyName,D.Category_ID,B.shipperID
) abc
WHERE _rank_ <=3
ORDER BY avg_delivery_time
```

115 %

Results | Messages | Client Statistics

	CompanyName	Category_ID	shipperID	avg_delivery_time	_rank_
1	Fed Ex	5005	4	14	1
2	ONE Ocean Network Express	5007	8	14	1
3	Fed Ex	5007	4	14	1
4	COSCO China Ocean Shipping Company	5011	6	14	1
5	Delhivery	5011	5	15	2
6	Hapag Lloyd	5011	7	15	2
7	Lufthansa Cargo	5011	3	15	2
8	Delhivery	5007	5	15	2
9	Lufthansa Cargo	5007	3	15	2
10	Fed Ex	5010	4	15	1
11	ONE Ocean Network Express	5010	8	15	1
12	COSCO China Ocean Shipping Company	5010	6	15	1
13	Delhivery	5005	5	15	2
14	COSCO China Ocean Shipping Company	5006	6	15	1
15	Fed Ex	5008	4	15	1
16	Delhivery	5008	5	15	1
17	COSCO China Ocean Shipping Company	5008	6	15	1
18	ONE Ocean Network Express	5008	8	15	1

Query executed successfully.

Q7. Write a SQL query to find the salespeople who deal with a single customer. Return salesman_id, name, city and commission. (Tables: Salesman, Customer)
Answer-

Approach 1:

```sql
SELECT * FROM Salesman WHERE NOT EXISTS (
SELECT COUNT(*) FROM Customer AS B
WHERE Salesman.salesman_id=B.salesman_id
GROUP BY B.salesman_id  HAVING COUNT(*)>1)

SELECT * FROM Salesman WHERE EXISTS (
SELECT COUNT(*) FROM Customer AS B
WHERE Salesman.salesman_id=B.salesman_id
GROUP BY B.salesman_id  HAVING COUNT(*)=1)
```

125 %

Results | Messages

	salesman_id	name	city	commission
1	5005	Pit Alex	London	0.11
2	5006	Mc Lyon	Paris	0.14
3	5007	Paul Adam	Rome	0.13
4	5003	Lauson Hen	San Jose	0.12

	salesman_id	name	city	commission
1	5005	Pit Alex	London	0.11
2	5006	Mc Lyon	Paris	0.14
3	5007	Paul Adam	Rome	0.13
4	5003	Lauson Hen	San Jose	0.12

Approach 2:

(Tables: Salesman, Customer)

```sql
SELECT *
FROM salesman
WHERE salesman_id IN (
    SELECT DISTINCT salesman_id
    FROM customer a
    WHERE NOT EXISTS (
        SELECT * FROM customer b
        WHERE a.salesman_id=b.salesman_id
        AND a.cust name <> b.cust name));
```

Results | Messages

salesman_id	name	city	commission
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	0.12

Q8. Find the cumulative sum using Partition by and self join.
(Tables: Employees, Departments)

DPP11.sql - BYOD-6...1\Keerthi Das (59))*

```sql
--USING PARTITION
SELECT A.ID,A.Dept_ID,A.Job_Role,B.Departments,
SUM(A.Salary) OVER(PARTITION BY A.Dept_ID ORDER BY A.ID ASC) AS Cum_sum
FROM Employee A
LEFT JOIN Departments B ON A.Dept_ID=B.ID

---USING SELF JOIN
SELECT A.ID,A.Name,A.Salary,A.Dept_ID,SUM(B.Salary) AS Cum_sum
FROM Employee A
JOIN Employee B ON A.ID>=B.ID
 GROUP BY A.ID,A.Name,A.Salary,A.Dept_ID
 ORDER BY A.ID
```

115 %

Results | Messages

	ID	Dept_ID	Job_Role	Departments	Cum_sum
1	103	1	Data Analyst	IT	62000
2	106	1	SDE	IT	136000
3	115	1	SDE	IT	223000
4	120	1	Data Scientist	IT	314000
5	102	2	Business Analyst	Management	70000
6	111	2	Business Analyst	Management	143000
7	116	2	President	Management	488000
8	117	2	VP	Management	675000
9	110	3	Salesman	Sales	32000

	ID	Name	Salary	Dept_ID	Cum_sum
1	102	Amit	70000	2	70000
2	103	Rahul	62000	1	132000
3	105	Rina	60000	5	192000
4	106	Dilip	74000	1	266000
5	107	Aman	81000	5	347000
6	108	Neha	50000	4	397000
7	110	Bhu...	32000	3	429000
8	111	Ashi...	73000	2	502000
9	112	Pooja	45000	4	547000

Query executed successfully. BYOD-607291\KEERTHI (15.0 RTM) BYOD-607291\Keerthi Da

Q9. Find all the owners whose pet’s name does not begin from letters ‘D’ to ‘S’
(Tables: Owners, Pets)

```sql
--14.   Find all the owners whose pet’s name does not begin  from letters d to s.
SELECT A.Name FROM owners A
WHERE EXISTS
  (SELECT B.Name FROM Pets B WHERE NOT EXISTS
    (SELECT B.Name FROM Pets WHERE UPPER(B.Name) BETWEEN 'D%' AND 'S%')
    )
```

115 %

Results | Messages

	Name
1	Jessica
2	Rosa
3	Susan
4	Benjamin
5	Charles
6	Joe
7	Jason
8	Joseph
9	Carolyn
10	Doris
11	Jeffrey
12	Christopher
13	William
14	Robert
15	Luisa
16	Wm
17	John
18	Anne
19	Bruce
20	John
21	Travis
22	Paul
23	Ed