🔹 Problem 1: Employee Hierarchy & Salary Aggregation

```
CREATE TABLE Employees (
    emp_id INT PRIMARY KEY,
    emp_name VARCHAR(50),
    manager_id INT,
    salary INT
);

INSERT INTO Employees VALUES
(1, 'Alice', NULL, 100000),
(2, 'Bob', 1, 80000),
(3, 'Charlie', 1, 75000),
(4, 'David', 2, 50000),
(5, 'Eva', 2, 45000),
(6, 'Frank', 3, 40000);
```

Task: Find the **total salary under each manager**, including indirect reports.

Ans-

```
With recursive cte as (
Select EMP_id,Manager_id,Salary,EMP_id as N,emp_name from Employees
Where EMp_id  in (Select manager_id from employees)
union all
Select e.EMP_id,E.Manager_id,E.Salary,Cte.N,cte.emp_name from Employees as E
Join Cte on Cte.EMP_ID=E.Manager_id),Cte1 as(
Select N as EMPId,EMp_id ,EMP_name,Salary from Cte )
Select EMPID,EMP_name,Sum(Salary) as Tot_salary from Cte1
Group by EMPID,EMP_name
order by EMPID;
```

🔹 Problem 2: Fraudulent Transactions (Impossible Travel)

```sql
CREATE TABLE Transactions (
    txn_id INT PRIMARY KEY,
    user_id INT,
    amount DECIMAL(10,2),
    txn_time TIMESTAMP,
    location VARCHAR(50)
);

INSERT INTO Transactions VALUES
(1, 101, 250.00, '2025-09-01 09:00:00', 'India'),
(2, 101, 300.00, '2025-09-01 09:20:00', 'USA'),
(3, 102, 150.00, '2025-09-01 10:00:00', 'India'),
(4, 102, 200.00, '2025-09-01 12:00:00', 'India'),
(5, 103, 500.00, '2025-09-01 14:00:00', 'Germany'),
(6, 103, 600.00, '2025-09-01 14:25:00', 'France');
```

Task: Find users who transacted in **two different countries within 30 minutes.**

Ans-

```sql
With cte as (
Select *,Lead(Txn_time ) Over(partition by User_id order by TXn_time)
as TM from transactions1),cte1 as (
Select *,timestampdiff(minute,Txn_time,TM) as Dif from cte),Cte2 as (
Select *,
Case when Dif<=30 then 1
when Lag(Dif,1,0) Over(partition by User_id order by Txn_time)<=30 then 1 else 0
 end as TM1 from cte1)
 Select User_id,Count(distinct Location) as CT from cte2
 where TM1=1
 Group by User_id
 having Ct=2;
```

Problem 3: Daily Login Streaks

```sql
CREATE TABLE Logins (
    user_id INT,
    login_date DATE
);

INSERT INTO Logins VALUES
(201, '2025-09-01'),
(201, '2025-09-02'),
(201, '2025-09-03'),
(201, '2025-09-05'),
(202, '2025-09-01'),
(202, '2025-09-03'),
(202, '2025-09-04'),
(202, '2025-09-05');
```

Task: Find the **longest consecutive login streak** per user.

Ans-

```sql
With cte as (
Select *,row_number() Over(partition by User_id order by Login_date)
 as RNK from Logins)
,Cte1 as (
Select *,date_sub(Login_date,interval RNk Day)
 as GP from Cte),Cte2 as (
Select User_id,GP,Count(0) as CT,Max(Count(0))
Over(partition by user_id) as MX from Cte1
group by User_id,GP)
Select User_id,MX from cte2
Where CT=MX;
```

Problem 4: Anomalous Orders

```sql
CREATE TABLE Orders (
    order_id INT,
    customer_id INT,
    order_date DATE,
    amount DECIMAL(10,2)
);

INSERT INTO Orders VALUES
(1, 301, '2025-08-01', 200),
(2, 301, '2025-08-10', 220),
(3, 301, '2025-08-20', 2500),
(4, 302, '2025-08-05', 300),
(5, 302, '2025-08-15', 320),
(6, 303, '2025-08-07', 1000);
```

Task: Find customers whose order is **3x higher than their 6-month average**.

```sql
INSERT INTO Orders VALUES
(7, 301, '2025-08-25', 10000),  -- anomalous: much higher than previous average
(8, 302, '2025-08-20', 1200),   -- anomalous: previous avg ~310, 1200 > 3*310
(9, 304, '2025-08-10', 100),
(10, 304, '2025-08-15', 110),
(11, 304, '2025-08-20', 400),   -- anomalous: previous avg 105, 3*105 = 315
(12, 305, '2025-08-18', 50),
(13, 305, '2025-08-22', 60),
(14, 305, '2025-08-25', 55);
```

Ans-

```sql
with cte as (
Select *,Avg(Amount) Over(partition by customer_id order by Order_date
rows between unbounded preceding and 1 preceding
)*3 as Av from Order1
order by Customer_id)
select * from Cte
where AV is not null and AV<Amount;
```

Problem 5: Loyal Customers by Category

```sql
CREATE TABLE Products (
    prod_id INT,
    category VARCHAR(50)
);

CREATE TABLE Sales (
    sale_id INT,
    customer_id INT,
    prod_id INT,
    sale_date DATE
);
```

```sql
INSERT INTO Products VALUES
(1, 'Electronics'),
(2, 'Electronics'),
(3, 'Books'),
(4, 'Books');

INSERT INTO Sales VALUES
(1, 401, 1, '2025-08-01'),
(2, 401, 2, '2025-08-15'),
(3, 402, 3, '2025-08-02'),
(4, 402, 3, '2025-09-02'),
(5, 403, 4, '2025-08-10'),
(6, 403, 4, '2025-09-12');
```

Task: For each category, find the **most loyal customer** (who bought most months).

Ans-

```sql
with cte as (
Select S.*,P.Category from Sales as S
join
Products as p
on S.Prod_id=P.Prod_id),cte1 as(
Select *,date_format(sale_date,'%b-%Y') as Ym from cte)
Select Category,Customer_id,Count(distinct YM) from cte1
group by Category,Customer_id;
```

🔹 Problem 6: Salary Gaps

```sql
CREATE TABLE DeptEmployees (
    emp_id INT,
    department VARCHAR(50),
    salary INT
);

INSERT INTO DeptEmployees VALUES
(1, 'HR', 50000),
(2, 'HR', 80000),
(3, 'HR', 200000),
(4, 'IT', 60000),
(5, 'IT', 62000),
(6, 'IT', 63000);
```

Task: Find departments where **2nd highest salary > 2x median salary**.

Ans-

```sql
with cte as (
Select *,dense_rank() Over (partition by
Department order by Salary Desc) as RNK,dense_rank() Over (partition by
Department order by Salary) as RNK1,
Count(0) Over(partition by Department) as CT
 from DeptEmployees),Cte1 as (
 Select *,
 Case when CT%2=0 then CT/2
 else (CT+1)/2
 End  as R1,
 Case when CT%2=0 then (Case when
 CT%2=0 then  CT/2  else (CT+1)/2 End)+1 else 0 end
 as MX from Cte),cte2 as (
 Select Department,Avg(Salary) as AG from Cte1
 where RNK=R1 or RNK=MX
 Group by Department)
 Select Cte.Department,Cte.Salary from Cte
 join Cte2 on Cte.Department=Cte2.Department
 where Cte.RNK=2
 and Cte.Salary> (2*Cte2.AG);
```

Problem 7: Running Competition Leaderboard

```sql
CREATE TABLE Races (
    race_id INT,
    user_id INT,
    finish_time INT,   -- seconds
    race_date DATE
);

INSERT INTO Races VALUES
(1, 501, 350, '2025-08-01'),
(2, 502, 340, '2025-08-01'),
(3, 501, 360, '2025-08-15'),
(4, 502, 330, '2025-08-15'),
(5, 503, 370, '2025-08-15');
```

Task: Show **rank progression of each user across races.**

Ans-

```sql
With cte as (
Select User_id,Race_Date,dense_rank()
Over(partition by Race_date order by Finish_time ) as RNK from races
order by user_id)
Select *,Lag(RNK,1,null) over(partition by user_id order by Race_date)
 as PreRank,RNK-Lag(RNK,1,null) over(partition by user_id) as RankChange
 from Cte;
```

Problem 8: Stock Price Spike

```sql
CREATE TABLE StockPrices (
    stock_id INT,
    price DECIMAL(10,2),
    date DATE
);

INSERT INTO StockPrices VALUES
(1, 100, '2025-08-01'),
(1, 115, '2025-08-02'),
(1, 130, '2025-08-03'),
(1, 150, '2025-08-04'),
(2, 200, '2025-08-01'),
(2, 202, '2025-08-02'),
(2, 205, '2025-08-03');
```

Task: Find stocks with a **20% increase within 3 consecutive days.**

Ans-

```sql
With cte as (
Select *,Lag(Date,2,0) over(partition by stock_id order by Date)
 as L  from StockPrices ),Cte1 as (
Select A.Stock_id,A.Date,A.Price,B.Price as PC  from cte as A
join Cte as B on
A.Stock_id=B.Stock_id and A.Date=B.L)
Select *,round((Pc/Price-1)*100,2) as Per from Cte1
where round((Pc/Price-1)*100,2)>20
order by Stock_id,date;
```

🔹 Problem 9: Shopping Cart Abandonment

```sql
CREATE TABLE CartEvents (
    user_id INT,
    cart_id INT,
    event_type VARCHAR(20),  -- add, remove, checkout
    event_time TIMESTAMP
);

INSERT INTO CartEvents VALUES
(601, 1, 'add', '2025-09-01 10:00:00'),
(601, 1, 'add', '2025-09-01 10:05:00'),
(601, 1, 'remove', '2025-09-01 10:10:00'),
(602, 2, 'add', '2025-09-01 11:00:00'),
(602, 2, 'add', '2025-09-01 11:05:00'),
(603, 3, 'add', '2025-09-02 09:00:00'),
(603, 3, 'checkout', '2025-09-03 12:00:00'),
(604, 4, 'add', '2025-09-03 14:00:00'),
(604, 4, 'add', '2025-09-03 14:05:00'),
(605, 5, 'add', '2025-09-04 08:00:00'),
(605, 5, 'remove', '2025-09-04 08:10:00'),
(606, 6, 'add', '2025-09-05 10:00:00'),
(606, 6, 'checkout', '2025-09-12 11:00:00'),
(607, 7, 'add', '2025-09-06 09:00:00'),
(607, 7, 'add', '2025-09-06 09:05:00'),
(608, 8, 'add', '2025-09-07 15:00:00'),
(608, 8, 'checkout', '2025-09-08 16:00:00'),
(609, 9, 'add', '2025-09-08 10:00:00'),
(609, 9, 'remove', '2025-09-08 10:10:00'),
(610, 10, 'add', '2025-09-09 12:00:00');
```

##Task- Find users who added items but didn’t checkout within 7 days.

Ans-

```
with cte as (
Select *,Min(
Case when Event_type='add' then Event_time else null end
) Over(partition by cart_id)
 as MN,Date_add(Min(Case when Event_type='add' then Event_time else null end)
Over(partition by cart_id),interval 7 day) as DD from cartevents),Cte1 as (
select  User_id,
Case when Event_time<=DD and Event_type='checkout' then 1 else 0 end  as Stat
 from cte)
 Select User_id from Cte1
 group by User_id having Sum(stat)=0;
```

◆ Problem 10: Overlapping Bookings

```
CREATE TABLE Bookings (
    booking_id INT,
    room_id INT,
    start_time TIMESTAMP,
    end_time TIMESTAMP
);
```

```sql
INSERT INTO Bookings VALUES
-- Room 701: partial overlap
(1, 701, '2025-09-01 10:00:00', '2025-09-01 12:00:00'),
(2, 701, '2025-09-01 11:30:00', '2025-09-01 13:00:00'),
(3, 701, '2025-09-01 12:30:00', '2025-09-01 14:00:00'),

-- Room 702: consecutive bookings, no overlap
(4, 702, '2025-09-01 14:00:00', '2025-09-01 16:00:00'),
(5, 702, '2025-09-01 16:00:00', '2025-09-01 18:00:00'),

-- Room 703: one booking completely inside another
(6, 703, '2025-09-02 09:00:00', '2025-09-02 12:00:00'),
(7, 703, '2025-09-02 10:00:00', '2025-09-02 11:00:00'),

-- Room 704: multiple overlapping bookings
(8, 704, '2025-09-02 13:00:00', '2025-09-02 15:00:00'),
(9, 704, '2025-09-02 14:00:00', '2025-09-02 16:00:00'),
(10, 704, '2025-09-02 14:30:00', '2025-09-02 15:30:00'),
```

```sql
-- Room 705: single booking, no overlap
(11, 705, '2025-09-03 08:00:00', '2025-09-03 10:00:00'),

-- Room 706: partial overlap with consecutive booking
(12, 706, '2025-09-03 12:00:00', '2025-09-03 14:00:00'),
(13, 706, '2025-09-03 13:30:00', '2025-09-03 15:30:00'),

-- Room 707: multiple bookings with gaps, some overlap
(14, 707, '2025-09-04 09:00:00', '2025-09-04 11:00:00'),
(15, 707, '2025-09-04 10:30:00', '2025-09-04 12:00:00'),
(16, 707, '2025-09-04 12:30:00', '2025-09-04 14:00:00');
```

Ans-

```sql
with recursive Cte as (
Select Booking_id,Room_id,Start_time,End_time from (
Select *,min(start_time) Over(partition by Room_id ) as Mn from bookings)
as t1 where Mn=Start_time
union all
Select t2.Booking_id,t3.Room_id,t2.Start_time,t2.End_time from bookings as T2
join Cte as t3 on T2.Start_time>=T3.End_time and T2.Room_id=T3.room_id)
select *,'Available' as Room_status from cte
Union
Select *,'Not Available' as ST from bookings
where booking_id not in (Select booking_id from Cte)
order by room_id,start_time;
```