SQL Part 3

Q1. Write a query to calculate the balance for each account on a daily Basis, consolidating multiple transactions on the same date for the same account. Assume the balance starts at $0 and that: Credit adds to the balance. Debit subtracts from the balance.

```
 CREATE TABLE bank_txn (
     TransactionID INT PRIMARY KEY,
     AccountID INT NOT NULL,
     TransactionDate DATE NOT NULL,
     Amount DECIMAL(10, 2) NOT NULL,
     TransactionType VARCHAR(10) CHECK (TransactionType IN ('Credit', 'Debit')));

  INSERT INTO bank_txn (TransactionID, AccountID, TransactionDate, Amount, TransactionType) VALUES
   (1, 401, '2024-11-01', 500.00, 'Credit'),
   (2, 402, '2024-11-01', 300.00, 'Debit'),
   (3, 401, '2024-11-02', 200.00, 'Debit'),
   (4, 403, '2024-11-02', 700.00, 'Credit'),
   (5, 401, '2024-11-03', 300.00, 'Credit'),
   (6, 402, '2024-11-03', 100.00, 'Debit'),
   (7, 404, '2024-11-03', 400.00, 'Credit'),
   (8, 401, '2024-11-04', 100.00, 'Debit'),
   (9, 402, '2024-11-04', 500.00, 'Credit'),
   (10, 403, '2024-11-05', 300.00, 'Debit'),
   (11, 404, '2024-11-05', 200.00, 'Credit'),
   (12, 402, '2024-11-06', 300.00, 'Debit'),
   (13, 401, '2024-11-06', 400.00, 'Credit'),
   (14, 401, '2024-11-06', 200.00, 'Debit'),
   (15, 402, '2024-11-06', 100.00, 'Credit'),
   (16, 403, '2024-11-06', 500.00, 'Credit'),
   (17, 404, '2024-11-06', 300.00, 'Debit'),
   (18, 403, '2024-11-06', 200.00, 'Debit');

select * from bank_txn

with cte as (
select *,
case when TransactionType = 'Credit' then Amount else - Amount end as New_Amount
from bank_txn),
cte2 as (
select AccountID,TransactionDate,
SUM(New_Amount) as Amt
from cte
group by AccountID,TransactionDate),
cte3 as (
select AccountID,TransactionDate,
SUM(Amt) over(partition by AccountID order by TransactionDate) as Running_Total
from cte2)
select TransactionDate,
SUM(case when AccountID = 401 then Running_Total else null end ) as '401',
SUM(case when AccountID = 402 then Running_Total else null end ) as '402',
SUM(case when AccountID = 403 then Running_Total else null end ) as '403',
SUM(case when AccountID = 404 then Running_Total else null end ) as '404'
from cte3
group by TransactionDate
```

Q2.How would you find the longest sequence of consecutive days a customer placed orders?

```
CREATE TABLE Customer_Order (
   CustomerID INT,
   OrderDate DATE,
   OrderID INT PRIMARY KEY
);


INSERT INTO Customer_Order (CustomerID, OrderDate, OrderID)
VALUES
(1, '2025-01-01', 101),
(1, '2025-01-02', 102),
(1, '2025-01-04', 103),
(1, '2025-01-05', 104),
(1, '2025-01-07', 105),
(2, '2025-01-01', 106),
(2, '2025-01-02', 107),
(2, '2025-01-03', 108),
(3, '2025-01-01', 109),
(3, '2025-01-03', 110);


select * from Customer_Order


with cte as (
select
CustomerId,
orderdate,
orderid,
dateadd(DAY,-1 * ROW_NUMBER() over(partition by customerid order by orderdate ) ,orderdate) as Window
from Customer_Order ),
cte2 as(
select
Customerid,
Window,
count(CustomerID) as No_Of_Days
from cte
group by Customerid, Window)
select
CustomerID,
MAX(No_Of_Days) as Max_Consecutive_days
from cte2
group by CustomerID
```

Q3. Write a query to display the records for which more than 100 (inclusive) people are visiting the stadium for 3 or more consecutive days ?

```
create table stadium(
id int,
visit_date date,
people int
)

insert into stadium(id,visit_date,people)
values(1,'2022-01-01',10),
(2,'2022-01-02',109),
(3,'2022-01-03',150),
(4,'2022-01-04',99),
(5,'2022-01-05',150),
(6,'2022-01-06',145),
(7,'2022-01-07',199),
(8,'2022-01-08',188),
(9,'2022-01-09',99),
(10,'2022-01-10',109),
(11,'2022-01-11',150),
(12,'2022-01-12',100),
(13,'2022-01-13',89),
(14,'2022-01-14',121),
(15,'2022-01-15',145)

select * from Customer_Order

with Cnt_records as(
select
id,
visit_date,
people,
ROW_NUMBER() over(order by visit_date asc) as rn,
(id - ROW_NUMBER() over(order by visit_date asc) ) as New_Id
from stadium
where people>=100),
consecutive as(
select
visit_date,
people,
COUNT(New_Id) over(partition by New_Id) as Cnt
from Cnt_records)
select
visit_date,
people
from consecutive
where Cnt>=3
```

Q4. Find the employees who are earning a salary greater than and less than the average department salary ?
Without using JOIN and WINDOW functions

```
create table emp_salary(
id int,
name varchar(50),
salary int,
departmentid int
)

insert into emp_salary(Id,Name,Salary,DepartmentId)
values
(1,'Joe',85000,1),
(2,'Henry',80000,2),
(3,'Sam',60000,2),
(4,'Max',90000,1),
(5,'Janet',69000,1),
(6,'Randy',85000,1),
(7,'Will',70000,1),
(8,'Chris',65000,3),
(9,'cathy',75000,3),
(10,'louis',80000,3)


select * from emp_salary


with cte as (
select
id,
name,
salary,
departmentid,
(select AVG(salary) as Avg_Dept  from emp_salary e2
 where  e2.departmentid =e1.departmentid ) as Avg_dept_Salary
from emp_salary e1 )
select
id,
name,
salary,
departmentid,
case when salary>Avg_dept_Salary then 'Higher'
    when salary<Avg_dept_Salary then 'Lower'
    end as Sal_ide
from cte
```

Q5. Write a SQL query to find all the employees from employee table who are also managers

```
CREATE TABLE Employee (
 EMPLOYEE_ID INT IDENTITY(1,1) PRIMARY KEY NOT NULL,
 FIRST_NAME CHAR(25),
 LAST_NAME CHAR(25),
 SALARY INT,
 JOINING_DATE DATE,
 DEPARTMENT CHAR(25),
   MANAGER_ID INT
);


INSERT INTO Employee
(FIRST_NAME, LAST_NAME, SALARY, JOINING_DATE, DEPARTMENT,MANAGER_ID)
 VALUES
  ('James', 'Smith', 100000, '2020-02-02', 'HR', 002),
  ('Jessica', 'Kohl', 80000, '2011-06-17', 'Admin', 005),
  ('Alex', 'Garner', 300000, '2020-02-17', 'HR', 011),
  ('Pratik', 'Pandey', 500000, '2020-02-17', 'Admin', 020),
  ('Christine', 'Robinson', 500000, '2011-06-17', 'Admin', 007),
  ('Deepak', 'Gupta', 200000, '2011-06-17', 'Account', 015),
  ('Jennifer', 'Paul', 75000, '2020-01-17', 'Account', 012),
  ('Deepika', 'Sharma', 90000, '2011-04-17', 'Admin', 017);

select * from Employee

select
CONCAT(RTRIM(e1.FIRST_NAME),' ', RTRIM(e1.LAST_NAME)) as Emp_Name,
CONCAT(RTRIM(e2.FIRST_NAME),' ', RTRIM(e2.LAST_NAME)) as Mngr_Name
from Employee e1
join
Employee e2
on
e1.EMPLOYEE_ID = e2.MANAGER_ID
```

Q6. Write a query to find out the customers who booked both air and train tickets.

```
CREATE TABLE Customer (
 Customer_id INTEGER,
 Booking_Id date NOT NULL,
 Travel_Type varchar(100)
);

INSERT INTO Customer VALUES (152, '2023-01-12', 'train');
INSERT INTO Customer VALUES (353, '2023-01-13', 'air');
INSERT INTO Customer VALUES (152, '2023-01-19', 'air');
INSERT INTO Customer VALUES (777, '2023-01-15', 'air');
INSERT INTO Customer VALUES (352, '2023-01-16', 'train');
INSERT INTO Customer VALUES (353, '2023-01-13', 'train');
INSERT INTO Customer VALUES (353, '2023-01-14', 'train');

select * from Customer



select
distinct c1.Customer_id,
c1.Travel_Type
from Customer c1
join
Customer c2
on
c1.Customer_id = c2.Customer_id
and
c1.Travel_Type <>c2.Travel_Type
order by 1
```

Q7.From Customer table, we need to Find domain from email id.

```
create table customer_tbl
(Id int,
email varchar(50)
)

INSERT INTO customer_tbl (id, email) VALUES
(1, 'abc@gmail.com'),
(2, 'xyz@hotmail.com'),
(3, 'pqr@outlook.com'),
(4, 'john.doe@yahoo.com'),
(5, 'emma@icloud.com'),
(6, 'william@protonmail.com'),
(7, 'sophia@zoho.com'),
(8, 'liam@live.com'),
(9, 'olivia@gmx.com'),
(10, 'michael@aol.com'),
(11, 'charlotte@rediffmail.com'),
(12, 'james@mail.com'),
(13, 'amelia@fastmail.com'),
(14, 'benjamin@tutanota.com'),
(15, 'harper@yandex.com');

select * from customer_tbl
```

Method 1

```
select
id,
email,
SUBSTRING(email, CHARINDEX('@',email)+1,len(email)) as Domain
from customer_tbl
```

Method 2

```
select
id,
email,
RIGHT(email,len(email) - charindex('@',email)) as Domain
from customer_tbl
```

Q8. Find data of all the employees that are hired in last two months , given data of employee name and their joining date

```
CREATE TABLE Employees (
  emp_id INT IDENTITY(1,1) PRIMARY KEY,
  emp_name VARCHAR(100),
  joining_date DATE
);

INSERT INTO Employees (emp_name, joining_date)
VALUES

('John', '2023-10-15'),
('Alice', '2023-12-20'),
('Bob', '2024-01-19'),
('Emily', '2024-02-05'),
('David', '2024-02-20'),
('Sophia', '2024-03-01'),
('Michael', '2024-03-10') ,
('Olivia', '2024-03-15'),
('William', '2024-03-19'),
('Lucas', '2025-01-19'),
('Emma', '2025-02-05'),
('Daniel', '2025-02-20'),
('Isabella', '2025-03-01'),
('Ethan', '2025-03-10'),
('Sophia', '2025-03-15'),
('Noah', '2025-03-19');


select * from Employees


select
emp_id,
emp_name,
joining_date
from Employees
where joining_date >=DATEADD(MM,-3,getdate())
```

Q9. Write a SQL query to find employee (first name, last name, department and bonus with highest bonus.

```
CREATE TABLE Employee (
 EMPLOYEE_ID INT IDENTITY(1,1) PRIMARY KEY NOT NULL,
 FIRST_NAME CHAR(25),
 LAST_NAME CHAR(25),
 SALARY INT,
 JOINING_DATE DATE,
 DEPARTMENT CHAR(25),
   MANAGER_ID INT
);

INSERT INTO Employee
(FIRST_NAME, LAST_NAME, SALARY, JOINING_DATE, DEPARTMENT,MANAGER_ID) VALUES
  ('James', 'Smith', 100000, '2020-02-02', 'HR', 002),
  ('Jessica', 'Kohl', 80000, '2011-06-17', 'Admin', 005),
  ('Alex', 'Garner', 300000, '2020-02-17', 'HR', 011),
  ('Pratik', 'Pandey', 500000, '2020-02-17', 'Admin', 020),
  ('Christine', 'Robinson', 500000, '2011-06-17', 'Admin', 007),
  ('Deepak', 'Gupta', 200000, '2011-06-17', 'Account', 015),
  ('Jennifer', 'Paul', 75000, '2020-01-17', 'Account', 012),
  ('Deepika', 'Sharma', 90000, '2011-04-17', 'Admin', 017);

CREATE TABLE Bonus (
EMPLOYEE_REF_ID INT FOREIGN KEY REFERENCES employee(employee_id),
BONUS_AMOUNT INT,
BONUS_DATE DATE
)

INSERT INTO Bonus
(EMPLOYEE_REF_ID, BONUS_AMOUNT, BONUS_DATE) VALUES
(001, 5000, '2020-02-18'),
(002, 3000, '2018-06-11'),
(003, 4000, '2018-02-20'),
(001, 4500, '2018-02-20'),
(002, 3500, '2018-06-11');

select * from Employee
select * from Bonus

select FIRST_NAME,LAST_NAME,DEPARTMENT,Total_Bonus
from
Employee c
join(
select Top 1 EMPLOYEE_ID, SUM(BONUS_AMOUNT) as Total_Bonus
from Employee e
join bonus b
on e.EMPLOYEE_ID=b.EMPLOYEE_REF_ID
group by EMPLOYEE_ID) d
on c.EMPLOYEE_ID = d.EMPLOYEE_ID
```

Q10. We are having a log table created for storing all logs of execution of jobs. It consists of job id, job start time and job status. Return list of jobs which have consecutive failure 3 times.

```
-- Create LogTable
CREATE TABLE LogTable (
   JobID INT,
   JobStartTime DATE,
   JobStatus VARCHAR(20)
);


-- Insert sample data into LogTable
INSERT INTO LogTable (JobID, JobStartTime, JobStatus) VALUES (1, '2024-03-01', 'Success');
INSERT INTO LogTable (JobID, JobStartTime, JobStatus) VALUES (1, '2024-03-02', 'Success');
INSERT INTO LogTable (JobID, JobStartTime, JobStatus) VALUES (1, '2024-03-03', 'Failure');
INSERT INTO LogTable (JobID, JobStartTime, JobStatus) VALUES (1, '2024-03-04', 'Failure');
INSERT INTO LogTable (JobID, JobStartTime, JobStatus) VALUES (1, '2024-03-05', 'Failure');
INSERT INTO LogTable (JobID, JobStartTime, JobStatus) VALUES (2, '2024-03-01', 'Failure');
INSERT INTO LogTable (JobID, JobStartTime, JobStatus) VALUES (2, '2024-03-02', 'Failure');
INSERT INTO LogTable (JobID, JobStartTime, JobStatus) VALUES (2, '2024-03-03', 'Failure');
INSERT INTO LogTable (JobID, JobStartTime, JobStatus) VALUES (2, '2024-03-04', 'Success');
INSERT INTO LogTable (JobID, JobStartTime, JobStatus) VALUES (2, '2024-03-05', 'Success');
INSERT INTO LogTable (JobID, JobStartTime, JobStatus) VALUES (3, '2024-03-01', 'Failure');
INSERT INTO LogTable (JobID, JobStartTime, JobStatus) VALUES (3, '2024-03-03', 'Failure');
INSERT INTO LogTable (JobID, JobStartTime, JobStatus) VALUES (3, '2024-03-04', 'Failure');
INSERT INTO LogTable (JobID, JobStartTime, JobStatus) VALUES (4, '2024-03-01', 'Failure');

select * from LogTable


with cte as (
select *,
LAG(JobStatus,1)  over(partition by jobid order by jobstarttime asc) as prev1,
LAG(JobStatus,1)  over(partition by jobid order by jobstarttime asc) as prev2
from LogTable)
select  distinct JobID
from cte
where JobStatus = 'Failure'
and prev1 = 'Failure'
and prev2 = 'Failure'
```

Q11. For the product table, we need to Find aggregated cost of product.

```
CREATE TABLE prd_tbl (
   dt DATE,
   brand VARCHAR(50),
   model VARCHAR(50),
   production_cost INT
);

-- Insert records
INSERT INTO prd_tbl (dt, brand, model, production_cost)
VALUES
   ('2023-12-01', 'A', 'A1', 1000),
   ('2023-12-01', 'A', 'A2', 1300),
   ('2023-12-01', 'B', 'B1', 800),
   ('2023-12-02', 'A', 'A1', 1800),
   ('2023-12-02', 'B', 'B1', 900),
   ('2023-12-10', 'A', 'A1', 1400),
   ('2023-12-10', 'A', 'A1', 1200),
   ('2023-12-10', 'C', 'C1', 2500);


select * from prd_tbl


select
dt,
brand,
model,
production_cost,
SUM(production_cost) over(partition by brand,dt order by dt) as Aggregated_Cost
from prd_tbl
```