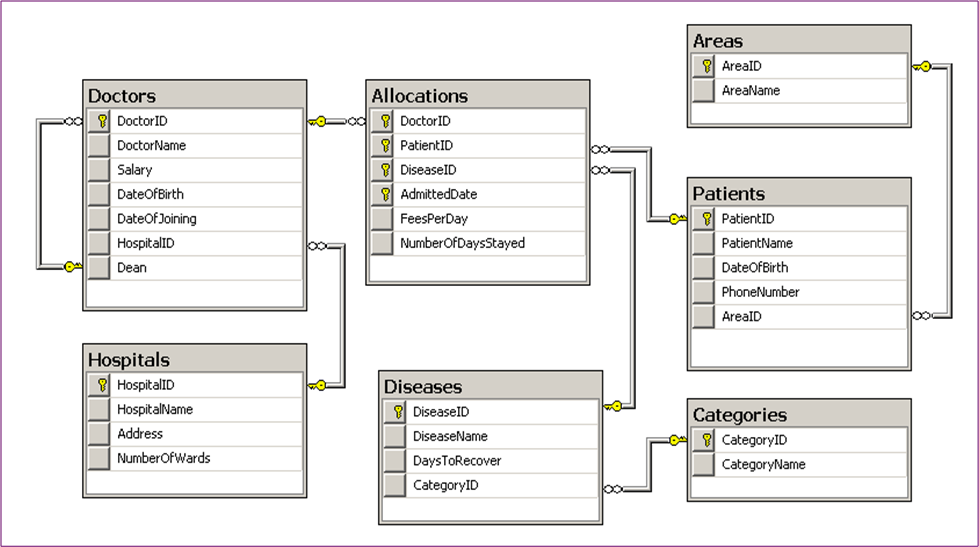
Hospital Management Domain



USE hospitalmanagementdomain;

CREATE TABLE Categories (

CategoryID VARCHAR(10) PRIMARY KEY,

CategoryName VARCHAR(25)

);

CREATE TABLE Areas (

AreaID VARCHAR(10) PRIMARY KEY,

AreaName VARCHAR(25)

);

CREATE TABLE Patients (

PatientID VARCHAR(10) PRIMARY KEY,

PatientName VARCHAR(25),

DOB DATE,

PhoneNo BIGINT,

AreaID VARCHAR(10),

FOREIGN KEY (AreaID) REFERENCES Areas(AreaID) ON DELETE SET NULL

);

CREATE TABLE Diseases (

DiseaseID VARCHAR(10) PRIMARY KEY,

DiseaseName VARCHAR(25),

DaysToRecover INT,

CategoryID VARCHAR(10),

FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID) ON DELETE SET NULL

);

CREATE TABLE Hospitals (

HospitalID VARCHAR(10) PRIMARY KEY,

HospitalName VARCHAR(25),

Address VARCHAR(25),

NoOfWards INT

);

CREATE TABLE Doctors (

DoctorID VARCHAR(10) PRIMARY KEY,

DoctorName VARCHAR(25),

Salary DECIMAL(10,2),

DOB DATE,

DOJ DATE,

Dean VARCHAR(10),

HospitalId VARCHAR(10),

FOREIGN KEY (HospitalId) REFERENCES Hospitals(HospitalId) ON DELETE SET NULL,

FOREIGN KEY (Dean) REFERENCES Doctors(DoctorID) ON DELETE SET NULL

);

CREATE TABLE Allocation (

DoctorID VARCHAR(10),

PatientID VARCHAR(10),

DiseaseID VARCHAR(10),

AdmittedDate DATE,

FeesPerDay DECIMAL(10,2),

NoOfDays INT,

PRIMARY KEY (DoctorID, PatientID, DiseaseID, AdmittedDate),

FOREIGN KEY (DoctorID) REFERENCES Doctors(DoctorID) ON DELETE CASCADE,

FOREIGN KEY (PatientID) REFERENCES Patients(PatientID) ON DELETE CASCADE,

FOREIGN KEY (DiseaseID) REFERENCES Diseases(DiseaseID) ON DELETE CASCADE

);

-- Inserting Data

INSERT INTO Categories VALUES

('C01','FEVER'), ('C02','CANCER'), ('C03','FRACTURES'), ('C04','PSYCHOLOGY'),

('C05','LEPROSY'), ('C06','CORONARY'), ('C07','ENT'), ('C08','MIGRAINE');

INSERT INTO Areas VALUES

('A01','ANNA NAGAR'), ('A02','TAMBARAM'), ('A03','T NAGAR'), ('A04','MYLAPORE'),

('A05','CHROMPET'), ('A06','GUINDY'), ('A07','CHENGALPET'), ('A08','OMR');

INSERT INTO Patients VALUES

('P01','SITA','2011-01-12',9994555123,'A01'),

('P02','GITA','2012-02-23',9794355123,'A01'),

('P03','TINA','2011-01-12',8876055123,'A02'),

('P04','MEENA','2013-03-12',9789555123,'A01'),

('P05','RAJA','2011-01-12',9994555123,'A03'),

('P06','KUMAR','2011-12-02',8894555569,'A01'),

('P07','MANISH','2012-06-24',7794554423,'A08'),

('P08','RAJESH','2010-08-06',9994015123,'A06'),

('P09', 'ARJUN', '1995-07-15', 9876543210, 'A01'),

('P10', 'PRIYA', '1988-10-23', 9867541230, 'A02'),

('P11', 'VIKRAM', '2001-03-12', 9785124369, 'A03'),

('P12', 'ANANYA', '1999-05-27', 9658743215, 'A04'),

('P13', 'SURYA', '1990-12-09', 9987456321, 'A05'),

('P14', 'MEERA', '1985-06-18', 9765432189, 'A06'),

('P15', 'KARTHIK', '1998-04-25', 9856471236, 'A07'),

('P16', 'DIVYA', '1993-11-30', 9658745632, 'A08'),

('P18', 'RAHUL', '1996-09-14', 9786453125, 'A01');

INSERT INTO Diseases VALUES

('DIS01','VIRAL FEVER',3,'C01'),

('DIS02','CHOLERA',27,'C01'),

('DIS03','TUBERCULOSIS',15,'C01'),

('DIS04','SKIN CANCER',25,'C02'),

('DIS05','DEPRESSION',7,'C04'),

('DIS06','CHEST PAIN',23,'C06'),

('DIS07','HIGH FEVER',4,'C01');

INSERT INTO Hospitals VALUES

('H01','APOLLO','ANNA NAGAR',100),

('H02','MEDWIN','T NAGAR',100),

('H03','MAXWELL','CHROMPET',100),

('H04','FORTIS','CHENGALPET',100),

('H05','MIOT','TAMBARAM',100),

('H06','AIIMS','ADYAR',100),

('H07','JIPMER','PONDICHERRY',100);

INSERT INTO Doctors VALUES

('D01','JYOTHI',65000,'1953-06-12','2000-01-15','D01','H01'),

('D02','SARAN',55000,'1967-12-02','2002-03-05','D01','H02'),

('D03','VIVEK',35000,'1957-02-26','2006-11-23','D01','H01'),

('D04','RAMAN',22000,'1957-04-22','2005-08-12','D04','H03'),

('D05','GEETHA',11000,'1971-03-17','2000-07-09','D05','H03'),

('D06','SHREE',99500,'1956-03-12','1996-02-19','D06','H04'),

('D07','RANI',35000,'1967-06-19','2004-01-08','D07','H05'),

('D08','KUMAR',96000,'1960-07-23','2000-04-11','D08','H06');

INSERT INTO Allocation VALUES

('D01','P01','DIS01','2011-04-15',50,1),

('D01','P02','DIS02','2012-12-13',150,27),

('D02','P03','DIS01','2009-08-21',75,10),

('D01','P04','DIS03','2007-06-27',80,12),

('D03','P05','DIS04','2011-06-06',69,1),

('D03','P06','DIS05','2012-04-11',45,4),

('D03','P01','DIS01','2012-05-16',20,8),

('D05','P03','DIS05','2013-01-24',60,9),

('D05','P04','DIS05','2011-11-23',40,10),

('D06','P07','DIS06','2011-10-13',50,20),

('D01','P01','DIS07','2010-06-09',500,27),

('D06','P08','DIS03','2011-12-11',50,20),

('D06','P09','DIS01','2011-02-25',50,12),

('D01','P04','DIS07','2011-05-23',50,19),

('D01','P03','DIS05','2011-12-31',200,7);

-- 1. Generate a report: Patient Name | Doctor Name | Disease Name

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS rpt\_patient\_doctor\_disease;

DELIMITER $$

CREATE PROCEDURE rpt\_patient\_doctor\_disease()

BEGIN

SELECT p.patientname, d.doctorname, ds.diseasename

FROM allocation a

JOIN patients p

ON p.patientid = a.patientid

JOIN doctors d

ON d.doctorid = a.doctorid

JOIN diseases ds

ON ds.diseaseid = a.diseaseid;

END $$

DELIMITER ;

CALL rpt\_patient\_doctor\_disease();

-- Method 2: Using JOIN

SELECT p.patientname, d.doctorname, ds.diseasename

FROM allocation a

JOIN patients p

ON p.patientid = a.patientid

JOIN doctors d

ON d.doctorid = a.doctorid

JOIN diseases ds

ON ds.diseaseid = a.diseaseid;

-- Method 3: Using WITH

WITH ad AS (

SELECT doctorid, patientid, diseaseid

FROM allocation

)

SELECT p.patientname, d.doctorname, ds.diseasename

FROM ad

JOIN patients p

ON p.patientid = ad.patientid

JOIN doctors d

ON d.doctorid = ad.doctorid

JOIN diseases ds

ON ds.diseaseid = ad.diseaseid;

-- Method 4: Using NOT IN

SELECT p.patientname, d.doctorname, ds.diseasename

FROM allocation a

JOIN patients p

ON p.patientid = a.patientid

JOIN doctors d

ON d.doctorid = a.doctorid

JOIN diseases ds

ON ds.diseaseid = a.diseaseid

WHERE a.patientid NOT IN (

SELECT patientid

FROM allocation aa

WHERE aa.patientid IS NULL

);

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT p.patientname, d.doctorname, ds.diseasename

FROM patients p

JOIN allocation a

ON a.patientid = p.patientid

WHERE EXISTS (

SELECT 1

FROM doctors d

WHERE d.doctorid = a.doctorid

)

AND EXISTS (

SELECT 1

FROM diseases ds

WHERE ds.diseaseid = a.diseaseid

);

-- Method 6: Using UNION (SET operators)

SELECT p.patientname, d.doctorname, ds.diseasename

FROM allocation a

JOIN patients p

ON p.patientid = a.patientid

JOIN doctors d

ON d.doctorid = a.doctorid

JOIN diseases ds

ON ds.diseaseid = a.diseaseid

UNION

SELECT p.patientname, d.doctorname, ds.diseasename

FROM allocation a

JOIN doctors d

ON d.doctorid = a.doctorid

JOIN diseases ds

ON ds.diseaseid = a.diseaseid

JOIN patients p

ON p.patientid = a.patientid;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT p.patientname, d.doctorname, ds.diseasename

FROM allocation a

JOIN patients p

ON p.patientid = a.patientid

JOIN doctors d

ON d.doctorid = a.doctorid

JOIN diseases ds

ON ds.diseaseid = a.diseaseid

EXCEPT

SELECT p.patientname, d.doctorname, ds.diseasename

FROM patients p;

-- Method 8: Using INTERSECT (SET operators)

SELECT p.patientname, d.doctorname, ds.diseasename

FROM allocation a

JOIN patients p

ON p.patientid = a.patientid

JOIN doctors d

ON d.doctorid = a.doctorid

JOIN diseases ds

ON ds.diseaseid = a.diseaseid

INTERSECT

SELECT p.patientname, d.doctorname, ds.diseasename

FROM diseases ds

JOIN allocation a

ON ds.diseaseid = a.diseaseid

JOIN patients p

ON p.patientid = a.patientid

JOIN doctors d

ON d.doctorid = a.doctorid;

\*/

-- 2. Generate a report: Hospital Name | Number of doctors

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS rpt\_hospital\_doctor\_count;

DELIMITER $$

CREATE PROCEDURE rpt\_hospital\_doctor\_count()

BEGIN

SELECT h.hospitalname, COUNT(d.doctorid) AS number\_of\_doctors

FROM hospitals h

LEFT JOIN doctors d

ON d.hospitalid = h.hospitalid

GROUP BY h.hospitalid;

END $$

DELIMITER ;

CALL rpt\_hospital\_doctor\_count();

-- Method 2: Using JOIN

SELECT h.hospitalname, COUNT(d.doctorid) AS number\_of\_doctors

FROM hospitals h

LEFT JOIN doctors d

ON d.hospitalid = h.hospitalid

GROUP BY h.hospitalid;

-- Method 3: Using WITH

WITH doc\_per\_hosp AS (

SELECT hospitalid, COUNT(\*) AS number\_of\_doctors

FROM doctors

GROUP BY hospitalid

)

SELECT h.hospitalname, COALESCE(dph.number\_of\_doctors, 0) AS number\_of\_doctors

FROM hospitals h

LEFT JOIN doc\_per\_hosp dph

ON dph.hospitalid = h.hospitalid;

-- Method 4: Using NOT IN

SELECT h.hospitalname, (

SELECT COUNT(d.doctorid)

FROM doctors d

WHERE d.hospitalid = h.hospitalid

) AS number\_of\_doctors

FROM hospitals h

WHERE h.hospitalid NOT IN (

SELECT hospitalid FROM hospitals WHERE hospitalid IS NULL -- dummy NOT IN set

);

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT h.hospitalname, (

SELECT COUNT(d.doctorid)

FROM doctors d

WHERE d.hospitalid = h.hospitalid

) AS number\_of\_doctors

FROM hospitals h

WHERE EXISTS (

SELECT 1

FROM doctors d

WHERE d.hospitalid = h.hospitalid

);

-- Method 6: Using UNION (SET operators)

SELECT h.hospitalname, COUNT(d.doctorid) AS number\_of\_doctors

FROM hospitals h

JOIN doctors d

ON d.hospitalid = h.hospitalid

GROUP BY h.hospitalid

UNION

SELECT h.hospitalname, 0 AS number\_of\_doctors

FROM hospitals h

WHERE h.hospitalid NOT IN (

SELECT DISTINCT hospitalid

FROM doctors

);

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT hospitalname, number\_of\_doctors

FROM (

SELECT h.hospitalname, COUNT(d.doctorid) AS number\_of\_doctors

FROM hospitals h

LEFT JOIN doctors d

ON d.hospitalid = h.hospitalid

GROUP BY h.hospitalid

) q

EXCEPT

SELECT hospitalname, 0

FROM dual;

-- Method 8: Using INTERSECT (SET operators)

SELECT hospitalname, number\_of\_doctors

FROM (

SELECT h.hospitalname, COUNT(d.doctorid) AS number\_of\_doctors

FROM hospitals h

LEFT JOIN doctors d

ON d.hospitalid = h.hospitalid

GROUP BY h.hospitalid

) q

INTERSECT

SELECT hospitalname, number\_of\_doctors

FROM (

SELECT h.hospitalname, COUNT(d.doctorid) AS number\_of\_doctors

FROM doctors d

JOIN hospitals h

ON h.hospitalid = d.hospitalid

GROUP BY h.hospitalid

) q2;

\*/

-- 3. Generate a report: Hospital Name | Doctor Name | Patient Name

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS rpt\_hospital\_doctor\_patient;

DELIMITER $$

CREATE PROCEDURE rpt\_hospital\_doctor\_patient()

BEGIN

SELECT h.hospitalname, d.doctorname, p.patientname

FROM allocation a

JOIN doctors d

ON d.doctorid = a.doctorid

JOIN hospitals h

ON h.hospitalid = d.hospitalid

JOIN patients p

ON p.patientid = a.patientid;

END $$

DELIMITER ;

CALL rpt\_hospital\_doctor\_patient();

-- Method 2: Using JOIN

SELECT h.hospitalname, d.doctorname, p.patientname

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

JOIN doctors d

ON a.doctorid = d.doctorid

JOIN hospitals h

ON d.hospitalid = h.hospitalid;

-- Method 3: Using WITH

WITH doc\_pat AS (

SELECT doctorid, patientid FROM allocation

)

SELECT h.hospitalname, d.doctorname, p.patientname

FROM doc\_pat dp

JOIN doctors d

ON d.doctorid = dp.doctorid

JOIN hospitals h

ON h.hospitalid = d.hospitalid

JOIN patients p

ON p.patientid = dp.patientid;

-- Method 4: Using NOT IN

SELECT h.hospitalname, d.doctorname, p.patientname

FROM allocation a

JOIN doctors d

ON d.doctorid = a.doctorid

JOIN hospitals h

ON h.hospitalid = d.hospitalid

JOIN patients p

ON p.patientid = a.patientid

WHERE a.doctorid NOT IN (

SELECT doctorid FROM doctors WHERE hospitalid IS NULL

);

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT h.hospitalname, d.doctorname, p.patientname

FROM hospitals h

JOIN doctors d

ON d.hospitalid = h.hospitalid

WHERE EXISTS (

SELECT 1 FROM allocation a

JOIN patients p

ON p.patientid = a.patientid

WHERE a.doctorid = d.doctorid

);

-- Method 6: Using UNION (SET operators)

SELECT h.hospitalname, d.doctorname, p.patientname

FROM allocation a

JOIN doctors d

ON d.doctorid = a.doctorid

JOIN hospitals h

ON h.hospitalid = d.hospitalid

JOIN patients p

ON p.patientid = a.patientid

UNION

SELECT h.hospitalname, d.doctorname, p.patientname

FROM doctors d

JOIN hospitals h

ON h.hospitalid = d.hospitalid

JOIN allocation a

ON a.doctorid = d.doctorid

JOIN patients p

ON p.patientid = a.patientid;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT hospitalname, doctorname, patientname

FROM (

SELECT h.hospitalname, d.doctorname, p.patientname

FROM allocation a

JOIN doctors d

ON d.doctorid = a.doctorid

JOIN hospitals h

ON h.hospitalid = d.hospitalid

JOIN patients p

ON p.patientid = a.patientid

) q

EXCEPT

SELECT hospitalname, doctorname, patientname

FROM (

SELECT h.hospitalname, d.doctorname, p.patientname

FROM allocation a

JOIN doctors d

ON d.doctorid = a.doctorid

JOIN hospitals h

ON h.hospitalid = d.hospitalid

JOIN patients p

ON p.patientid = a.patientid

WHERE 1 = 0

) z;

-- Method 8: Using INTERSECT (SET operators)

SELECT hospitalname, doctorname, patientname

FROM (

SELECT h.hospitalname, d.doctorname, p.patientname

FROM allocation a

JOIN doctors d

ON d.doctorid = a.doctorid

JOIN hospitals h

ON h.hospitalid = d.hospitalid

JOIN patients p

ON p.patientid = a.patientid

) q

INTERSECT

SELECT hospitalname, doctorname, patientname

FROM (

SELECT h.hospitalname, d.doctorname, p.patientname

FROM allocation a

JOIN doctors d

ON d.doctorid = a.doctorid

JOIN hospitals h

ON h.hospitalid = d.hospitalid

JOIN patients p

ON p.patientid = a.patientid

) q2;

\*/

-- 4. Using ORDER BY clause: Display doctor details with respect to salary. Doctor Name | Salary

-- The salary column must be sorted in descending order.

-- Method 1: Using RANK Function

WITH doc\_rank AS (

SELECT doctorname, salary,

RANK() OVER (ORDER BY salary DESC) AS sal\_rank

FROM doctors

)

SELECT doctorname, salary

FROM doc\_rank

ORDER BY salary DESC;

-- Method 2: Using NOT IN

SELECT doctorname, salary

FROM doctors d

WHERE salary NOT IN (

SELECT salary

FROM doctors

WHERE salary < d.salary

)

ORDER BY salary DESC;

-- Method 3: Using JOIN

SELECT d1.doctorname, d1.salary

FROM doctors d1

LEFT JOIN doctors d2

ON d2.salary > d1.salary

WHERE d2.doctorid IS NULL

UNION ALL

SELECT d.doctorname, d.salary

FROM doctors d

WHERE d.salary < (

SELECT MAX(salary)

FROM doctors

)

ORDER BY salary DESC;

-- Method 4: Using EXISTS (Correlated Subquery)

SELECT d.doctorname, d.salary

FROM doctors d

WHERE NOT EXISTS (

SELECT 1

FROM doctors d2

WHERE d2.salary > d.salary

)

UNION ALL

SELECT d.doctorname, d.salary

FROM doctors d

WHERE EXISTS (

SELECT 1

FROM doctors

WHERE salary > d.salary

)

ORDER BY salary DESC;

-- Method 5: Using UNION (SET operators)

SELECT doctorname, salary

FROM doctors

WHERE salary >= (

SELECT AVG(salary)

FROM doctors

)

UNION

SELECT doctorname, salary

FROM doctors

WHERE salary < (

SELECT AVG(salary)

FROM doctors

)

ORDER BY salary DESC;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT doctorname, salary

FROM doctors

EXCEPT

SELECT doctorname, salary

FROM doctors

WHERE salary < (

SELECT MAX(salary)

FROM doctors

);

-- Method 7: Using INTERSECT (SET operators)

SELECT doctorname, salary

FROM doctors

INTERSECT

SELECT doctorname, salary

FROM doctors;

\*/

-- 5. Select the patient details who has paid maximum bills so far?

-- Method 1: Using RANK Function

WITH patient\_bill AS (

SELECT p.patientid, p.patientname, SUM(a.feesperday \* a.noofdays) AS total\_bill,

RANK() OVER (ORDER BY SUM(a.feesperday \* a.noofdays) DESC) AS bill\_rank

FROM patients p

JOIN allocation a

ON a.patientid = p.patientid

GROUP BY p.patientid, p.patientname

)

SELECT patientid, patientname, total\_bill

FROM patient\_bill

WHERE bill\_rank = 1;

-- Method 2: Using NOT IN

SELECT pb.patientid, pb.patientname, pb.total\_bill

FROM (

SELECT p.patientid, p.patientname, SUM(a.feesperday \* a.noofdays) AS total\_bill

FROM patients p

JOIN allocation a

ON a.patientid = p.patientid

GROUP BY p.patientid, p.patientname

) pb

WHERE pb.total\_bill NOT IN (

SELECT SUM(a2.feesperday \* a2.noofdays)

FROM allocation a2

GROUP BY a2.patientid

HAVING SUM(a2.feesperday \* a2.noofdays) > pb.total\_bill

);

-- Method 3: Using JOIN

WITH bill\_per\_patient AS (

SELECT patientid, SUM(feesperday \* noofdays) AS total\_bill

FROM allocation

GROUP BY patientid

)

SELECT p.patientid, p.patientname, bpp.total\_bill

FROM bill\_per\_patient bpp

JOIN patients p

ON p.patientid = bpp.patientid

LEFT JOIN bill\_per\_patient bpp2

ON bpp2.total\_bill > bpp.total\_bill

WHERE bpp2.patientid IS NULL;

-- Method 4: Using EXISTS (Correlated Subquery)

WITH bill\_per\_patient AS (

SELECT patientid, SUM(feesperday \* noofdays) AS total\_bill

FROM allocation

GROUP BY patientid

)

SELECT p.patientid, p.patientname, bpp.total\_bill

FROM bill\_per\_patient bpp

JOIN patients p

ON p.patientid = bpp.patientid

WHERE NOT EXISTS (

SELECT 1

FROM bill\_per\_patient x

WHERE x.total\_bill > bpp.total\_bill

);

-- Method 5: Using UNION (SET operators)

SELECT p.patientid, p.patientname, b.total\_bill

FROM (

SELECT patientid, SUM(feesperday \* noofdays) AS total\_bill

FROM allocation

GROUP BY patientid

) b

JOIN patients p

ON p.patientid = b.patientid

WHERE b.total\_bill = (

SELECT MAX(SUM(feesperday \* noofdays))

FROM allocation

GROUP BY patientid

)

UNION

SELECT patientid, patientname, 0 AS total\_bill

FROM patients

WHERE patientid NOT IN (

SELECT DISTINCT patientid

FROM allocation

);

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT patientid, patientname, total\_bill

FROM (

SELECT p.patientid, p.patientname, SUM(a.feesperday \* a.noofdays) AS total\_bill

FROM patients p

JOIN allocation a

ON a.patientid = p.patientid

GROUP BY p.patientid, p.patientname

) q

EXCEPT

SELECT patientid, patientname, total\_bill

FROM (

SELECT p.patientid, p.patientname, SUM(a.feesperday \* a.noofdays) AS total\_bill

FROM patients p

JOIN allocation a

ON a.patientid = p.patientid

GROUP BY p.patientid, p.patientname

HAVING SUM(a.feesperday \* a.noofdays) < (

SELECT MAX(SUM(feesperday \* noofdays))

FROM allocation

GROUP BY patientid)

) z;

-- Method 7: Using INTERSECT (SET operators)

SELECT patientid, patientname, total\_bill

FROM (

SELECT p.patientid, p.patientname, SUM(a.feesperday \* a.noofdays) AS total\_bill

FROM patients p

JOIN allocation a

ON a.patientid = p.patientid

GROUP BY p.patientid, p.patientname

) q

INTERSECT

SELECT patientid, patientname, total\_bill

FROM (

SELECT p.patientid, p.patientname, SUM(a.feesperday \* a.noofdays) AS total\_bill

FROM patients p

JOIN allocation a

ON a.patientid = p.patientid

GROUP BY p.patientid, p.patientname

HAVING SUM(a.feesperday \* a.noofdays) = (

SELECT MAX(SUM(feesperday \* noofdays))

FROM allocation

GROUP BY patientid

)

) z;

\*/

-- 6. Select the patient details who has stayed for maximum number of days in the hospital

-- Method 1: Using RANK Function

WITH patient\_stay AS (

SELECT p.patientid, p.patientname, SUM(a.noofdays) AS total\_days\_stayed,

RANK() OVER (ORDER BY SUM(a.noofdays) DESC) AS stay\_rank

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

GROUP BY p.patientid, p.patientname

)

SELECT patientid, patientname, total\_days\_stayed

FROM patient\_stay

WHERE stay\_rank = 1;

-- Method 2: Using NOT IN

SELECT p.patientid, p.patientname, SUM(a.noofdays) AS total\_days\_stayed

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

GROUP BY p.patientid

HAVING SUM(a.noofdays) NOT IN (

SELECT SUM(a2.noofdays)

FROM allocation a2

GROUP BY a2.patientid

HAVING SUM(a2.noofdays) > ALL (

SELECT SUM(a3.noofdays)

FROM allocation a3

GROUP BY a3.patientid

)

);

-- Method 3: Using JOIN

SELECT p.patientid, p.patientname, t.total\_days\_stayed

FROM (

SELECT patientid, SUM(noofdays) AS total\_days\_stayed

FROM allocation

GROUP BY patientid

) t

JOIN patients p

ON t.patientid = p.patientid

WHERE t.total\_days\_stayed = (

SELECT MAX(sum\_a.total)

FROM (

SELECT patientid, SUM(noofdays) AS total

FROM allocation

GROUP BY patientid

) sum\_a

);

-- Method 4: Using EXISTS (Correlated Subquery)

SELECT p.patientid, p.patientname, SUM(a.noofdays) AS total\_days\_stayed

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

GROUP BY p.patientid

HAVING NOT EXISTS (

SELECT 1

FROM (

SELECT patientid, SUM(noofdays) AS total\_stay

FROM allocation

GROUP BY patientid

) b

WHERE b.total\_stay > SUM(a.noofdays)

);

-- Method 5: Using UNION (SET operators)

SELECT p.patientid, p.patientname, SUM(a.noofdays) AS total\_days\_stayed

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

GROUP BY p.patientid

HAVING SUM(a.noofdays) = (

SELECT MAX(total\_days)

FROM (

SELECT patientid, SUM(noofdays) AS total\_days

FROM allocation

GROUP BY patientid

) t

)

UNION

SELECT NULL, NULL, NULL

HAVING FALSE;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT a.patientid, p.patientname, SUM(a.noofdays) AS total\_days\_stayed

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

GROUP BY a.patientid, p.patientname

EXCEPT

SELECT a1.patientid, p1.patientname, SUM(a1.noofdays)

FROM allocation a1

JOIN patients p1

ON a1.patientid = p1.patientid

GROUP BY a1.patientid, p1.patientname

HAVING SUM(a1.noofdays) < (

SELECT MAX(sum\_a.noofdays\_total)

FROM (

SELECT patientid, SUM(noofdays) AS noofdays\_total

FROM allocation

GROUP BY patientid

) sum\_a

);

-- Method 7: Using INTERSECT (SET operators)

SELECT a.patientid, p.patientname, SUM(a.noofdays) AS total\_days\_stayed

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

GROUP BY a.patientid, p.patientname

INTERSECT

SELECT patientid, NULL, MAX(total)

FROM (

SELECT patientid, SUM(noofdays) AS total

FROM allocation

GROUP BY patientid

) t

GROUP BY patientid;

\*/

-- 7. Generate a report: PatientName | Disease ID | Doctor ID | No of times visited

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS GetPatientVisitReport;

DELIMITER $$

CREATE PROCEDURE GetPatientVisitReport()

BEGIN

SELECT p.patientname, a.diseaseid, a.doctorid, COUNT(\*) AS no\_of\_visits

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

GROUP BY p.patientid, p.patientname, a.diseaseid, a.doctorid;

END $$

DELIMITER ;

CALL GetPatientVisitReport();

-- Method 2: Using JOIN

SELECT p.patientname, a.diseaseid, a.doctorid, COUNT(\*) AS no\_of\_visits

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

GROUP BY p.patientid, p.patientname, a.diseaseid, a.doctorid;

-- Method 3: Using WITH

WITH visit\_counts AS (

SELECT a.patientid, a.diseaseid, a.doctorid, COUNT(\*) AS no\_of\_visits

FROM allocation a

GROUP BY a.patientid, a.diseaseid, a.doctorid

)

SELECT p.patientname, v.diseaseid, v.doctorid, v.no\_of\_visits

FROM visit\_counts v

JOIN patients p

ON v.patientid = p.patientid;

-- Method 4: Using NOT IN

SELECT p.patientname, a.diseaseid, a.doctorid, COUNT(\*) AS no\_of\_visits

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

WHERE a.patientid NOT IN (

SELECT patientid

FROM allocation

WHERE diseaseid IS NULL OR doctorid IS NULL

)

GROUP BY p.patientid, p.patientname, a.diseaseid, a.doctorid;

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT p.patientname, a.diseaseid, a.doctorid, COUNT(\*) AS no\_of\_visits

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

WHERE EXISTS (

SELECT 1

FROM allocation a2

WHERE a2.patientid = a.patientid

AND a2.diseaseid = a.diseaseid

AND a2.doctorid = a.doctorid

)

GROUP BY p.patientid, p.patientname, a.diseaseid, a.doctorid;

-- Method 6: Using UNION (SET operators)

SELECT p.patientname, a.diseaseid, a.doctorid, COUNT(\*) AS no\_of\_visits

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

WHERE a.diseaseid IS NOT NULL

GROUP BY p.patientid, p.patientname, a.diseaseid, a.doctorid

UNION

SELECT p.patientname, a.diseaseid, a.doctorid, COUNT(\*) AS no\_of\_visits

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

WHERE a.doctorid IS NOT NULL

GROUP BY p.patientid, p.patientname, a.diseaseid, a.doctorid;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (ANSI SQL - NOT Supported in MySQL)

SELECT p.patientname, a.diseaseid, a.doctorid, COUNT(\*) AS no\_of\_visits

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

GROUP BY p.patientid, p.patientname, a.diseaseid, a.doctorid

EXCEPT

SELECT p.patientname, a.diseaseid, a.doctorid, COUNT(\*) AS no\_of\_visits

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

WHERE a.diseaseid IS NULL OR a.doctorid IS NULL

GROUP BY p.patientid, p.patientname, a.diseaseid, a.doctorid;

-- Method 8: Using INTERSECT (ANSI SQL - NOT Supported in MySQL)

SELECT p.patientname, a.diseaseid, a.doctorid, COUNT(\*) AS no\_of\_visits

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

GROUP BY p.patientid, p.patientname, a.diseaseid, a.doctorid

INTERSECT

SELECT p.patientname, a.diseaseid, a.doctorid, COUNT(\*) AS no\_of\_visits

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

WHERE a.diseaseid IS NOT NULL AND a.doctorid IS NOT NULL

GROUP BY p.patientid, p.patientname, a.diseaseid, a.doctorid;

\*/

-- 8. Generate a report: Patient ID | No of times visited Report should be sorted with respect to second column.

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS rpt\_patient\_visits;

DELIMITER $$

CREATE PROCEDURE rpt\_patient\_visits()

BEGIN

SELECT patientid, COUNT(\*) AS no\_of\_visits

FROM allocation

GROUP BY patientid

ORDER BY no\_of\_visits DESC;

END $$

DELIMITER ;

CALL rpt\_patient\_visits();

-- Method 2: Using JOIN

SELECT patientid, COUNT(\*) AS NoOfVisits

FROM allocation

GROUP BY patientid

ORDER BY no\_of\_visits DESC;

-- Method 3: Using WITH

WITH patientvisitcounts AS (

SELECT patientid, COUNT(\*) AS noofvisits

FROM allocation

GROUP BY patientid

)

SELECT patientid, noofvisits

FROM patientvisitcounts

ORDER BY noofvisits DESC;

-- Method 4: Using NOT IN

SELECT p.patientid, (

SELECT COUNT(\*)

FROM allocation

WHERE patientid = p.patientid

) AS noofvisits

FROM patient p

WHERE p.patientid NOT IN (

SELECT 'InvalidID'

FROM allocation

WHERE patientid IS NULL

);

-- Method 5: Using EXISTS / NOT EXISTS (Correlated Subquery)

SELECT p.patientid, (

SELECT COUNT(a.patientid)

FROM allocation a

WHERE a.patientid = p.patientid

) AS noofvisits

FROM patients p

WHERE EXISTS (

SELECT 1

FROM allocation al

WHERE al.patientid = p.patientid

)

ORDER BY noofvisits DESC;

-- Method 6 Using UNION (SET operators)

SELECT patientid, COUNT(\*) AS no\_of\_visits

FROM allocation

GROUP BY patientid

HAVING COUNT(\*) = (

SELECT MAX(c)

FROM (

SELECT COUNT(\*) AS c

FROM allocation

GROUP BY patientid

) t

)

UNION ALL

SELECT patientid, COUNT(\*) AS no\_of\_visits

FROM allocation

GROUP BY patientid

HAVING COUNT(\*) < (

SELECT MAX(c)

FROM (

SELECT COUNT(\*) AS c

FROM allocation

GROUP BY patientid

) t

)

ORDER BY no\_of\_visits DESC;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT patientid, COUNT(\*) AS no\_of\_visits

FROM allocation

GROUP BY patientid

EXCEPT

SELECT patientid, COUNT(\*) AS no\_of\_visits

FROM allocation

GROUP BY patientid

HAVING COUNT(\*) < (

SELECT MAX(cnt)

FROM (

SELECT COUNT(\*) AS cnt

FROM allocation

GROUP BY patientid

) z

);

-- Method 8: Using INTERSECT (SET operators)

SELECT patientid, COUNT(\*) AS no\_of\_visits

FROM allocation

GROUP BY patientid

INTERSECT

SELECT patientid, COUNT(\*) AS no\_of\_visits

FROM allocation

GROUP BY patientid

HAVING COUNT(\*) = (

SELECT MAX(cnt)

FROM (

SELECT COUNT(\*) AS cnt

FROM allocation

GROUP BY patientid

) z

);

\*/

-- 9. Display the maximum salary earner for each hospital Hospital Name | Doctor Name | Maximum salary

-- Method 1: Using RANK Function

SELECT hospitalname, doctorname, salary

FROM (

SELECT h.hospitalname, d.doctorname, d.salary,

RANK() OVER (PARTITION BY d.hospitalid ORDER BY d.salary DESC) AS rnk

FROM doctors d

JOIN hospitals h ON d.hospitalid = h.hospitalid

) AS rankeddoctors

WHERE rnk = 1;

-- Method 2: Using NOT IN

SELECT h.hospitalname, d.doctorname, d.salary

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

WHERE (d.hospitalid, d.salary) NOT IN (

SELECT d2.hospitalid, d2.salary

FROM doctors d2

WHERE d2.salary < (

SELECT MAX(d3.salary)

FROM doctors d3

WHERE d3.hospitalid = d2.hospitalid

)

);

-- Method 3: Using JOIN

SELECT h.hospitalname, d.doctorname, d.salary

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

JOIN (

SELECT hospitalid, MAX(salary) AS maxsalary

FROM doctors

GROUP BY hospitalid

) AS maxsal

ON d.hospitalid = maxsal.hospitalid AND d.salary = maxsal.maxsalary;

-- Method 4: Using NOT EXISTS (Correlated Subquery)

SELECT h.hospitalname, d.doctorname, d.salary

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

WHERE NOT EXISTS (

SELECT 1

FROM doctors d2

WHERE d2.hospitalid = d.hospitalid AND d2.salary > d.salary

);

-- Method 5: Using UNION (SET operators)

SELECT h.hospitalname, d.doctorname, d.salary

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

WHERE d.salary = (

SELECT MAX(d2.salary)

FROM doctors d2

WHERE d2.hospitalid = d.hospitalid

)

UNION

SELECT h2.hospitalname, d2.doctorname, d2.salary

FROM doctors d2

JOIN hospitals h2

ON d2.hospitalid = h2.hospitalid

WHERE d2.salary = (

SELECT MAX(d3.salary)

FROM doctors d3

WHERE d3.hospitalid = d2.hospitalid

) AND d2.doctorid NOT IN (

SELECT d4.doctorid

FROM doctors d4

WHERE d4.salary = (

SELECT MAX(d5.salary)

FROM doctors d5

WHERE d5.hospitalid = d4.hospitalid

)

);

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT h.hospitalname, d.doctorname, d.salary

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

EXCEPT

SELECT h.hospitalname, d.doctorname, d.salary

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

WHERE d.salary < (

SELECT MAX(d2.salary)

FROM doctors d2

WHERE d2.hospitalid = d.hospitalid

);

-- Method 7: Using INTERSECT (SET operators)

SELECT h.hospitalname, d.doctorname, d.salary

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

INTERSECT

SELECT h.hospitalname, d.doctorname, maxsalary

FROM hospitals h

JOIN (

SELECT hospitalid, MAX(salary) AS maxsalary

FROM doctors

GROUP BY hospitalid

) AS ms

ON h.hospitalid = ms.hospitalid

JOIN doctors d

ON d.hospitalid = ms.hospitalid AND d.salary = ms.maxsalary;

\*/

-- 10. For each disease display the patients who has visited maximum number of times.

-- Method 1: Using JOIN

SELECT a.diseaseid, p.patientname, COUNT(\*) AS visitcount

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

GROUP BY a.diseaseid, a.patientid

HAVING visitcount = (

SELECT MAX(cnt)

FROM (

SELECT diseaseid, patientid, COUNT(\*) AS cnt

FROM allocation

GROUP BY diseaseid, patientid

HAVING allocation.diseaseid = diseaseid

) AS sub

);

-- Method 2: Using NOT IN

SELECT a.diseaseid, p.patientname, COUNT(\*) AS visitcount

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

GROUP BY a.diseaseid, a.patientid

HAVING (a.diseaseid, COUNT(\*)) NOT IN (

SELECT diseaseid, cnt

FROM (

SELECT diseaseid, MAX(visitcount) AS cnt

FROM (

SELECT diseaseid, patientid, COUNT(\*) AS visitcount

FROM allocation

GROUP BY diseaseid, patientid

) AS counts

GROUP BY diseaseid

) AS max\_visits

);

-- Method 3: Using NOT EXISTS (Correlated Subquery)

SELECT a.diseaseid, p.patientname, COUNT(\*) AS visitcount

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

GROUP BY a.diseaseid, a.patientid

HAVING NOT EXISTS (

SELECT 1

FROM (

SELECT diseaseid, patientid, COUNT(\*) AS cnt

FROM allocation

GROUP BY diseaseid, patientid

) AS others

WHERE others.diseaseid = a.diseaseid AND others.cnt > COUNT(\*)

);

-- Method 4: Using UNION (SET operators)

SELECT a.diseaseid, p.patientname, COUNT(\*) AS visitcount

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

GROUP BY a.diseaseid, a.patientid

HAVING COUNT(\*) = (

SELECT MAX(cnt)

FROM (

SELECT diseaseid, patientid, COUNT(\*) AS cnt

FROM allocation

GROUP BY diseaseid, patientid

HAVING diseaseid = a.diseaseid

) AS max\_visit

)

UNION

SELECT a.diseaseid, p.patientname, COUNT(\*) AS visitcount

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

GROUP BY a.diseaseid, a.patientid

HAVING COUNT(\*) = (

SELECT MAX(cnt)

FROM (

SELECT diseaseid, patientid, COUNT(\*) AS cnt

FROM allocation

GROUP BY diseaseid, patientid

HAVING diseaseid = a.diseaseid

) AS max\_visit

);

/\* Does not work in MySQL

-- Method 5: Using EXCEPT (SET operators)

SELECT a.diseaseid, p.patientname, COUNT(\*) AS visitcount

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

GROUP BY a.diseaseid, a.patientid

EXCEPT

SELECT a.diseaseid, p.patientname, COUNT(\*) AS visitcount

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

GROUP BY a.diseaseid, a.patientid

HAVING COUNT(\*) < (

SELECT MAX(cnt)

FROM (

SELECT diseaseid, patientid, COUNT(\*) AS cnt

FROM allocation

GROUP BY diseaseid, patientid

HAVING diseaseid = a.diseaseid

) AS max\_visit

);

-- Method 6: Using INTERSECT (SET operators)

SELECT a.diseaseid, p.patientname, COUNT(\*) AS visitcount

FROM allocation a

JOIN patients p ON a.patientid = p.patientid

GROUP BY a.diseaseid, a.patientid

INTERSECT

SELECT diseaseid, patientname, maxcnt

FROM (

SELECT a.diseaseid, p.patientname, COUNT(\*) AS maxcnt

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

GROUP BY a.diseaseid, a.patientid

) AS max\_table

WHERE maxcnt = (

SELECT MAX(cnt)

FROM (

SELECT diseaseid, patientid, COUNT(\*) AS cnt

FROM allocation

GROUP BY diseaseid, patientid

HAVING diseaseid = max\_table.diseaseid

) AS max\_sub

);

\*/

-- 11. Generate a report: Doctor Name | Dean name

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS rpt\_doctor\_dean;

DELIMITER $$

CREATE PROCEDURE rpt\_doctor\_dean()

BEGIN

SELECT d1.doctorname AS doctor, d2.doctorname AS dean

FROM doctors d1

LEFT JOIN doctors d2

ON d1.dean = d2.doctorid;

END $$

DELIMITER ;

CALL rpt\_doctor\_dean();

-- Method 2: Using LEFT JOIN

SELECT d1.doctorname AS doctor, d2.doctorname AS dean

FROM doctors d1

LEFT JOIN doctors d2

ON d1.dean = d2.doctorid;

-- Method 3: Using WITH

WITH doctor\_dean AS (

SELECT doctorid, doctorname, dean

FROM doctors

)

SELECT dd.doctorname AS doctor, de.doctorname AS dean

FROM doctor\_dean dd

LEFT JOIN doctor\_dean de

ON dd.dean = de.doctorid;

-- Method 4: Using NOT IN

SELECT d1.doctorname AS doctor, (

SELECT d2.doctorname

FROM doctors d2

WHERE d1.dean = d2.doctorid

) AS dean

FROM doctors d1

WHERE d1.doctorid NOT IN (

SELECT doctorid

FROM doctors

WHERE dean IS NULL

);

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT d1.doctorname AS doctor, (

SELECT d2.doctorname

FROM doctors d2

WHERE d2.doctorid = d1.dean

) AS dean

FROM doctors d1

WHERE d1.dean IS NULL

OR EXISTS (

SELECT 1

FROM doctors d2

WHERE d2.doctorid = d1.dean

);

-- Method 6 Using UNION (SET operators)

SELECT d1.doctorname AS doctor, d2.doctorname AS dean

FROM doctors d1

JOIN doctors d2

ON d1.dean = d2.doctorid

UNION

SELECT doctorname AS doctor, NULL AS dean

FROM doctors

WHERE dean IS NULL;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT d1.doctorname AS doctor, d2.doctorname AS dean

FROM doctors d1

JOIN doctors d2

ON d1.dean = d2.doctorid

EXCEPT

SELECT d1.doctorname, d2.doctorname

FROM doctors d1

JOIN doctors d2

ON d1.doctorid = d2.doctorid;

-- Method 8: Using INTERSECT (SET operators)

SELECT d1.doctorname AS doctor, d2.doctorname AS dean

FROM doctors d1

JOIN doctors d2

ON d1.dean = d2.doctorid

INTERSECT

SELECT doctorname, deanname

FROM (

SELECT d.doctorname, d2.doctorname AS deanname

FROM doctors d

LEFT JOIN doctors d2

ON d.dean = d2.doctorid

) AS sub;

\*/

-- 12. Display the doctors who earn more than his dean

-- Method 1: Using JOIN

SELECT d1.doctorname, d1.salary

FROM doctors d1

JOIN doctors d2

ON d1.dean = d2.doctorid

WHERE d1.salary > d2.salary;

-- Method 2: Using NOT IN

SELECT doctorname, salary

FROM doctors

WHERE dean IS NOT NULL

AND doctorid NOT IN (

SELECT d1.doctorid

FROM doctors d1

JOIN doctors d2

ON d1.dean = d2.doctorid

WHERE d1.salary <= d2.salary

);

-- Method 3: Using EXISTS (Correlated Subquery)

SELECT d1.doctorname, d1.salary

FROM doctors d1

WHERE EXISTS (

SELECT 1

FROM doctors d2

WHERE d1.dean = d2.doctorid AND d1.salary > d2.salary

);

-- Method 4: Using UNION ALL (SET operators)

SELECT d1.doctorname, d1.salary

FROM doctors d1

JOIN doctors d2

ON d1.dean = d2.doctorid

WHERE d1.salary > d2.salary

UNION ALL

SELECT d.doctorname, d.salary

FROM doctors d

WHERE d.salary > (

SELECT AVG(salary)

FROM doctors

);

/\* Does not work in MySQL

-- Method 5: Using EXCEPT (SET operators)

SELECT d1.doctorname, d1.salary

FROM doctors d1

JOIN doctors d2

ON d1.dean = d2.doctorid

EXCEPT

SELECT d1.doctorname, d1.salary

FROM doctors d1

JOIN doctors d2

ON d1.dean = d2.doctorid

WHERE d1.salary <= d2.salary;

-- Method 6: Using INTERSECT (SET operators)

SELECT d1.doctorname, d1.salary

FROM doctors d1

JOIN doctors d2

ON d1.dean = d2.doctorid

INTERSECT

SELECT d1.doctorname, d1.salary

FROM doctors d1

JOIN doctors d2

ON d1.dean = d2.doctorid

WHERE d1.salary > d2.salary;

\*/

-- 13. Display the doctors whose salary is greater than the average salary of the hospital

-- Method 1: Using JOIN

SELECT d.doctorname, d.salary, h.hospitalname

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

WHERE d.salary > (

SELECT AVG(salary)

FROM doctors

WHERE hospitalid = d.hospitalid

);

-- Method 2: Using NOT IN

SELECT d.doctorname, d.salary, h.hospitalname

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

WHERE d.salary NOT IN (

SELECT salary

FROM doctors

WHERE hospitalid = d.hospitalid AND salary <= (

SELECT AVG(salary)

FROM doctors

WHERE hospitalid = d.hospitalid

)

);

-- Method 3: Using EXISTS (Correlated Subquery)

SELECT d.doctorname, d.salary, h.hospitalname

FROM doctors d

JOIN hospitals h ON d.hospitalid = h.hospitalid

WHERE EXISTS (

SELECT 1

FROM doctors d2

WHERE d2.hospitalid = d.hospitalid AND d.salary > (

SELECT AVG(salary)

FROM doctors

WHERE hospitalid = d.hospitalid

)

);

-- Method 4: Using UNION ALL (SET operators)

SELECT d.doctorname, d.salary, h.hospitalname

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

WHERE d.salary > (

SELECT AVG(salary)

FROM doctors

WHERE hospitalid = d.hospitalid

)

UNION ALL

SELECT d.doctorname, d.salary, h.hospitalname

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

WHERE d.salary <= (

SELECT AVG(salary)

FROM doctors

WHERE hospitalid = d.hospitalid

);

/\* Does not work in MySQL

-- Method 5: Using EXCEPT (SET operators)

SELECT d.doctorname, d.salary, h.hospitalname

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

EXCEPT

SELECT d.doctorname, d.salary, h.hospitalname

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

WHERE d.salary <= (

SELECT AVG(salary)

FROM doctors

WHERE hospitalid = d.hospitalid

);

-- Method 6: Using INTERSECT (SET operators)

SELECT d.doctorname, d.salary, h.hospitalname

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

INTERSECT

SELECT d.doctorname, d.salary, h.hospitalname

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

WHERE d.salary > (

SELECT AVG(salary)

FROM doctors

WHERE hospitalid = d.hospitalid

);

\*/

-- 14. Using CASE/DECODE: Patient ID | Number of days stayed | Description

-- If no of days stayed is between 1 and 10 : Description is “MILD”

-- If no of days stayed is between 11 and 20 : Description is “MINOR”

-- If no of days stayed is above 30 : Description is “SEVERE”

-- Method 1: Using CASE

SELECT PatientID, NoOfDays,

CASE

WHEN NoOfDays BETWEEN 1 AND 10 THEN 'MILD'

WHEN NoOfDays BETWEEN 11 AND 20 THEN 'MINOR'

WHEN NoOfDays > 30 THEN 'SEVERE'

ELSE 'NORMAL'

END AS Description

FROM Allocation;

-- 15. Generate a report: Disease ID | No attempts (March 2013) | Description

-- If no of attempts is lesser than 10, Description is “Least bothered”

-- If no of attempts is greater than 10 and lesser than 30 , Description is “To be taken special care”

-- If no of attempts is greater than 30, Description is “Spreading dangerous”

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS rpt\_disease\_attempts;

DELIMITER $$

CREATE PROCEDURE rpt\_disease\_attempts()

BEGIN

SELECT diseaseid, COUNT(\*) AS noofattempts,

CASE

WHEN COUNT(\*) < 10 THEN 'Least Bothered'

WHEN COUNT(\*) BETWEEN 10 AND 30 THEN 'To be taken special care'

WHEN COUNT(\*) > 30 THEN 'Spreading Dangerous'

END AS description

FROM allocation

WHERE MONTH(admitteddate) = 3 AND YEAR(admitteddate) = 2013

GROUP BY diseaseid;

END $$

DELIMITER ;

CALL rpt\_disease\_attempts();

-- Method 2: Using CASE

SELECT DiseaseID, COUNT(\*) AS NoOfAttempts,

CASE

WHEN COUNT(\*) < 10 THEN 'Least Bothered'

WHEN COUNT(\*) BETWEEN 10 AND 30 THEN 'To be taken special care'

WHEN COUNT(\*) > 30 THEN 'Spreading Dangerous'

END AS Description

FROM Allocation

WHERE MONTH(AdmittedDate) = 3 AND YEAR(AdmittedDate) = 2013

GROUP BY DiseaseID;

-- Method 3: Using WITH

WITH disease\_attempts AS (

SELECT diseaseid, COUNT(\*) AS noofattempts

FROM allocation

WHERE MONTH(admitteddate) = 3 AND YEAR(admitteddate) = 2013

GROUP BY diseaseid

)

SELECT diseaseid, noofattempts,

CASE

WHEN noofattempts < 10 THEN 'Least Bothered'

WHEN noofattempts BETWEEN 10 AND 30 THEN 'To be taken special care'

WHEN noofattempts > 30 THEN 'Spreading Dangerous'

END AS description

FROM disease\_attempts;

-- Method 4: Using NOT IN

SELECT diseaseid, COUNT(\*) AS noofattempts,

CASE

WHEN COUNT(\*) < 10 THEN 'Least Bothered'

WHEN COUNT(\*) BETWEEN 10 AND 30 THEN 'To be taken special care'

WHEN COUNT(\*) > 30 THEN 'Spreading Dangerous'

END AS description

FROM allocation

WHERE diseaseid NOT IN (

SELECT diseaseid

FROM allocation

WHERE MONTH(admitteddate) <> 3 OR YEAR(admitteddate) <> 2013

)

AND MONTH(admitteddate) = 3 AND YEAR(admitteddate) = 2013

GROUP BY diseaseid;

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT a.diseaseid, COUNT(\*) AS noofattempts,

CASE

WHEN COUNT(\*) < 10 THEN 'Least Bothered'

WHEN COUNT(\*) BETWEEN 10 AND 30 THEN 'To be taken special care'

WHEN COUNT(\*) > 30 THEN 'Spreading Dangerous'

END AS description

FROM allocation a

WHERE MONTH(a.admitteddate) = 3 AND YEAR(a.admitteddate) = 2013

AND EXISTS (

SELECT 1 FROM allocation b

WHERE b.diseaseid = a.diseaseid

AND MONTH(b.admitteddate) = 3 AND YEAR(b.admitteddate) = 2013

)

GROUP BY a.diseaseid;

-- Method 6 Using UNION ALL (SET operators)

SELECT diseaseid, COUNT(\*) AS noofattempts, 'Least Bothered' AS description

FROM allocation

WHERE MONTH(admitteddate) = 3 AND YEAR(admitteddate) = 2013

GROUP BY diseaseid

HAVING COUNT(\*) < 10

UNION ALL

SELECT diseaseid, COUNT(\*) AS noofattempts, 'To be taken special care' AS description

FROM allocation

WHERE MONTH(admitteddate) = 3 AND YEAR(admitteddate) = 2013

GROUP BY diseaseid

HAVING COUNT(\*) BETWEEN 10 AND 30

UNION ALL

SELECT diseaseid, COUNT(\*) AS noofattempts, 'Spreading Dangerous' AS description

FROM allocation

WHERE MONTH(admitteddate) = 3 AND YEAR(admitteddate) = 2013

GROUP BY diseaseid

HAVING COUNT(\*) > 30;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT diseaseid, COUNT(\*) AS noofattempts,

CASE

WHEN COUNT(\*) < 10 THEN 'Least Bothered'

WHEN COUNT(\*) BETWEEN 10 AND 30 THEN 'To be taken special care'

WHEN COUNT(\*) > 30 THEN 'Spreading Dangerous'

END AS description

FROM allocation

WHERE MONTH(admitteddate) = 3 AND YEAR(admitteddate) = 2013

GROUP BY diseaseid

EXCEPT

SELECT diseaseid, 0, 'No Attempts'

FROM allocation

WHERE 1 = 0;

-- Method 8: Using INTERSECT (SET operators)

SELECT diseaseid, COUNT(\*) AS noofattempts,

CASE

WHEN COUNT(\*) < 10 THEN 'Least Bothered'

WHEN COUNT(\*) BETWEEN 10 AND 30 THEN 'To be taken special care'

WHEN COUNT(\*) > 30 THEN 'Spreading Dangerous'

END AS description

FROM allocation

WHERE MONTH(admitteddate) = 3 AND YEAR(admitteddate) = 2013

GROUP BY diseaseid

INTERSECT

SELECT diseaseid, COUNT(\*) AS noofattempts,

CASE

WHEN COUNT(\*) < 10 THEN 'Least Bothered'

WHEN COUNT(\*) BETWEEN 10 AND 30 THEN 'To be taken special care'

WHEN COUNT(\*) > 30 THEN 'Spreading Dangerous'

END AS description

FROM allocation

WHERE MONTH(admitteddate) = 3 AND YEAR(admitteddate) = 2013

GROUP BY diseaseid;

\*/

-- 16. In which month of 2012, no patient has visited the hospital “Sankara Nethralaya”

-- Method 1: Using JOIN

SELECT MONTHS.month\_no,

CASE

WHEN COALESCE(COUNT(a.patientid), 0) = 0 THEN 'No Visits'

ELSE 'Visited'

END AS visit\_status

FROM (

SELECT 1 AS month\_no

UNION ALL

SELECT 2

UNION ALL

SELECT 3

UNION ALL

SELECT 4

UNION ALL

SELECT 5

UNION ALL

SELECT 6

UNION ALL

SELECT 7

UNION ALL

SELECT 8

UNION ALL

SELECT 9

UNION ALL

SELECT 10

UNION ALL

SELECT 11

UNION ALL

SELECT 12

) AS MONTHS

LEFT JOIN allocation a

ON MONTH(a.admitteddate) = MONTHS.month\_no AND YEAR(a.admitteddate) = 2012

LEFT JOIN doctors d

ON a.doctorid = d.doctorid

LEFT JOIN hospitals h

ON d.hospitalid = h.hospitalid AND h.hospitalname = 'Sankara Nethralaya'

GROUP BY MONTHS.month\_no

HAVING visit\_status = 'No Visits'

ORDER BY MONTHS.month\_no;

/\* Does not work in MySQL

-- Method 2: Using PIVOT

SELECT visit\_status, [1] AS Jan, [2] AS Feb, [3] AS Mar, [4] AS Apr, [5] AS May, [6] AS Jun, [7] AS Jul, [8] AS Aug, [9] AS Sep, [10] AS Oct, [11] AS Nov, [12] AS Dec

FROM (

SELECT

CASE

WHEN COALESCE(COUNT(a.patientid), 0) = 0 THEN 'No Visits'

ELSE 'Visited'

END AS visit\_status,

MONTH(a.admitteddate) AS month\_no

FROM allocation a

JOIN doctors d ON a.doctorid = d.doctorid

JOIN hospitals h ON d.hospitalid = h.hospitalid

WHERE YEAR(a.admitteddate) = 2012 AND h.hospitalname = 'Sankara Nethralaya'

GROUP BY MONTH(a.admitteddate)

) AS source\_table

PIVOT (

COUNT(month\_no)

FOR month\_no IN ([1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12])

) AS pivot\_table;

\*/

-- 17. Prepare a report for the hospital “Madras Medical Mission”: Year | Month | Number of patients visited

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS rpt\_madras\_patients;

DELIMITER $$

CREATE PROCEDURE rpt\_madras\_patients()

BEGIN

SELECT YEAR(a.admitteddate) AS visit\_year, MONTH(a.admitteddate) AS visit\_month, COUNT(DISTINCT a.patientid) AS number\_of\_patients

FROM allocation a

JOIN doctors d

ON a.doctorid = d.doctorid

JOIN hospitals h

ON d.hospitalid = h.hospitalid

WHERE h.hospitalname = 'Madras Medical Mission'

GROUP BY YEAR(a.admitteddate), MONTH(a.admitteddate)

ORDER BY YEAR(a.admitteddate), MONTH(a.admitteddate);

END $$

DELIMITER ;

CALL rpt\_madras\_patients();

-- Method 2: Using CASE

SELECT YEAR(a.admitteddate) AS visit\_year, MONTH(a.admitteddate) AS visit\_month,

SUM(CASE WHEN h.hospitalname = 'Madras Medical Mission' THEN 1 ELSE 0 END) AS number\_of\_patients

FROM allocation a

JOIN doctors d

ON a.doctorid = d.doctorid

JOIN hospitals h

ON d.hospitalid = h.hospitalid

GROUP BY YEAR(a.admitteddate), MONTH(a.admitteddate)

HAVING SUM(CASE WHEN h.hospitalname = 'Madras Medical Mission' THEN 1 ELSE 0 END) > 0

ORDER BY YEAR(a.admitteddate), MONTH(a.admitteddate);

/\* Does not work in MySQL

-- Method 3: Using PIVOT

SELECT visit\_year, [1] AS Jan, [2] AS Feb, [3] AS Mar, [4] AS Apr, [5] AS May, [6] AS Jun, [7] AS Jul, [8] AS Aug, [9] AS Sep, [10] AS Oct, [11] AS Nov, [12] AS Dec

FROM (

SELECT YEAR(a.admitteddate) AS visit\_year, MONTH(a.admitteddate) AS visit\_month, a.patientid

FROM allocation a

JOIN doctors d

ON a.doctorid = d.doctorid

JOIN hospitals h

ON d.hospitalid = h.hospitalid

WHERE h.hospitalname = 'Madras Medical Mission'

) AS source\_table

PIVOT (

COUNT(DISTINCT patientid)

FOR visit\_month IN ([1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12])

) AS pivot\_table

ORDER BY visit\_year;

\*/

-- 18. Generate a report: Doctor Name | Age

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS rpt\_doctor\_age;

DELIMITER $$

CREATE PROCEDURE rpt\_doctor\_age()

BEGIN

SELECT doctorname, TIMESTAMPDIFF(YEAR, dob, CURDATE()) AS age

FROM doctors;

END $$

DELIMITER ;

CALL rpt\_doctor\_age();

-- Method 2: Using TIMESTAMPDIFF

SELECT doctorname, TIMESTAMPDIFF(YEAR, dob, CURDATE()) AS age

FROM doctors;

-- Method 3: Using JOIN

SELECT d.doctorname, TIMESTAMPDIFF(YEAR, d.dob, CURDATE()) AS age

FROM doctors d

JOIN allocation a

ON d.doctorid = a.doctorid;

-- Method 4: Using NOT IN

SELECT doctorname, TIMESTAMPDIFF(YEAR, dob, CURDATE()) AS age

FROM doctors

WHERE doctorid NOT IN (

SELECT doctorid

FROM doctors

WHERE dob IS NULL

);

-- Method 5: Using EXISTS / NOT EXISTS (Correlated Subquery)

SELECT d.doctorname, TIMESTAMPDIFF(YEAR, d.dob, CURDATE()) AS age

FROM doctors d

WHERE EXISTS (

SELECT 1

FROM doctors d2

WHERE d2.doctorid = d.doctorid AND d2.dob IS NOT NULL

);

-- Method 6 Using UNION (SET operators)

SELECT doctorname, TIMESTAMPDIFF(YEAR, dob, CURDATE()) AS age

FROM doctors

WHERE dob IS NOT NULL

UNION

SELECT doctorname, NULL AS age

FROM doctors

WHERE dob IS NULL;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT doctorname, DATEDIFF(YEAR, dob, GETDATE()) AS age

FROM doctors

EXCEPT

SELECT doctorname, DATEDIFF(YEAR, dob, GETDATE()) AS age

FROM doctors

WHERE dob IS NULL;

-- Method 8: Using INTERSECT (SET operators)

SELECT doctorname, DATEDIFF(YEAR, dob, GETDATE()) AS age

FROM doctors

INTERSECT

SELECT doctorname, DATEDIFF(YEAR, dob, GETDATE()) AS age

FROM doctors

WHERE dob IS NOT NULL;

\*/

-- 19. Generate a report: Disease ID | Patient ID | From Date | Discharged date

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS rpt\_disease\_patient\_dates;

DELIMITER $$

CREATE PROCEDURE rpt\_disease\_patient\_dates()

BEGIN

SELECT diseaseid, patientid, admitteddate AS fromdate, DATE\_ADD(admitteddate, INTERVAL noofdays DAY) AS dischargeddate

FROM allocation;

END $$

DELIMITER ;

CALL rpt\_disease\_patient\_dates();

-- Method 2: Using DATE\_ADD

SELECT diseaseid, patientid, admitteddate AS fromdate, DATE\_ADD(admitteddate, INTERVAL noofdays DAY) AS dischargeddate

FROM allocation;

-- Method 3: Using JOIN

SELECT a.diseaseid, a.patientid, a.admitteddate AS fromdate, DATE\_ADD(a.admitteddate, INTERVAL a.noofdays DAY) AS dischargeddate

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid;

-- Method 4: Using NOT IN

SELECT diseaseid, patientid, admitteddate AS fromdate, DATE\_ADD(admitteddate, INTERVAL noofdays DAY) AS dischargeddate

FROM allocation

WHERE patientid NOT IN (

SELECT patientid

FROM allocation

WHERE noofdays IS NULL

);

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT a.diseaseid, a.patientid, a.admitteddate AS fromdate, DATE\_ADD(a.admitteddate, INTERVAL a.noofdays DAY) AS dischargeddate

FROM allocation a

WHERE EXISTS (

SELECT 1

FROM patients p

WHERE p.patientid = a.patientid

);

-- Method 6 Using UNION (SET operators)

SELECT diseaseid, patientid, admitteddate AS fromdate, DATE\_ADD(admitteddate, INTERVAL noofdays DAY) AS dischargeddate

FROM allocation

WHERE noofdays IS NOT NULL

UNION

SELECT diseaseid, patientid, admitteddate AS fromdate, NULL AS dischargeddate

FROM allocation

WHERE noofdays IS NULL;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT diseaseid, patientid, admitteddate AS fromdate, DATEADD(DAY, noofdays, admitteddate) AS dischargeddate

FROM allocation

EXCEPT

SELECT diseaseid, patientid, admitteddate AS fromdate, DATEADD(DAY, noofdays, admitteddate) AS dischargeddate

FROM allocation

WHERE noofdays IS NULL;

-- Method 8: Using INTERSECT (SET operators)

SELECT diseaseid, patientid, admitteddate AS fromdate, DATEADD(DAY, noofdays, admitteddate) AS dischargeddate

FROM allocation

INTERSECT

SELECT diseaseid, patientid, admitteddate AS fromdate, DATEADD(DAY, noofdays, admitteddate) AS dischargeddate

FROM allocation

WHERE noofdays IS NOT NULL;

\*/

-- 20. Generate a report: Disease ID | Patient ID | Doctor ID | No of weeks stayed

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS rpt\_disease\_patient\_doctor\_weeks;

DELIMITER $$

CREATE PROCEDURE rpt\_disease\_patient\_doctor\_weeks()

BEGIN

SELECT diseaseid, patientid, doctorid, CEIL(noofdays / 7) AS noofweeksstayed

FROM allocation;

END $$

DELIMITER ;

CALL rpt\_disease\_patient\_doctor\_weeks();

-- Method 2: Using CEIL

SELECT diseaseid, patientid, doctorid, CEIL(noofdays / 7) AS noofweeksstayed

FROM allocation;

-- Method 3: Using JOIN

SELECT a.diseaseid, a.patientid, a.doctorid, CEIL(a.noofdays / 7) AS noofweeksstayed

FROM allocation a

JOIN patients p

ON a.patientid = p.patientid

JOIN doctors d

ON a.doctorid = d.doctorid;

-- Method 4: Using NOT IN

SELECT diseaseid, patientid, doctorid, CEIL(noofdays / 7) AS noofweeksstayed

FROM allocation

WHERE patientid NOT IN (

SELECT patientid

FROM allocation

WHERE noofdays IS NULL

);

-- Method 5: Using EXISTS(Correlated Subquery)

SELECT a.diseaseid, a.patientid, a.doctorid, CEIL(a.noofdays / 7) AS noofweeksstayed

FROM allocation a

WHERE EXISTS (

SELECT 1

FROM patients p

WHERE p.patientid = a.patientid

)

AND EXISTS (

SELECT 1

FROM doctors d

WHERE d.doctorid = a.doctorid

);

-- Method 6 Using UNION (SET operators)

SELECT diseaseid, patientid, doctorid, CEIL(noofdays / 7) AS noofweeksstayed

FROM allocation

WHERE noofdays IS NOT NULL

UNION

SELECT diseaseid, patientid, doctorid, NULL AS noofweeksstayed

FROM allocation

WHERE noofdays IS NULL;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT diseaseid, patientid, doctorid, CEILING(CAST(noofdays AS FLOAT) / 7) AS noofweeksstayed

FROM allocation

EXCEPT

SELECT diseaseid, patientid, doctorid, CEILING(CAST(noofdays AS FLOAT) / 7) AS noofweeksstayed

FROM allocation

WHERE noofdays IS NULL;

-- Method 8: Using INTERSECT (SET operators)

SELECT diseaseid, patientid, doctorid, CEILING(CAST(noofdays AS FLOAT) / 7) AS noofweeksstayed

FROM allocation

INTERSECT

SELECT diseaseid, patientid, doctorid, CEILING(CAST(noofdays AS FLOAT) / 7) AS noofweeksstayed

FROM allocation

WHERE noofdays IS NOT NULL;

\*/

-- 21. Find TOP 5 salaried doctors

-- Method 1: Using RANK Function

WITH ranked\_doctors AS (

SELECT doctorname, salary,

RANK() OVER (ORDER BY salary DESC) AS salary\_rank

FROM doctors

)

SELECT doctorname, salary

FROM ranked\_doctors

WHERE salary\_rank <= 5

ORDER BY salary DESC;

-- Method 2: Using NOT IN

SELECT doctorname, salary

FROM doctors d1

WHERE salary NOT IN (

SELECT d2.salary

FROM doctors d2

WHERE (

SELECT COUNT(DISTINCT d3.salary)

FROM doctors d3

WHERE d3.salary > d2.salary

) >= 5

)

ORDER BY salary DESC;

-- Method 3: Using JOIN

SELECT d1.doctorname, d1.salary

FROM doctors d1

LEFT JOIN doctors d2

ON d2.salary > d1.salary

GROUP BY d1.doctorid, d1.doctorname, d1.salary

HAVING COUNT(d2.salary) < 5

ORDER BY d1.salary DESC;

-- Method 4: Using EXISTS (Correlated Subquery)

SELECT d1.doctorname, d1.salary

FROM doctors d1

WHERE EXISTS (

SELECT 1

FROM doctors d2

WHERE d2.salary = d1.salary

AND (

SELECT COUNT(DISTINCT d3.salary)

FROM doctors d3

WHERE d3.salary > d2.salary

) < 5

)

ORDER BY d1.salary DESC;

-- Method 5: Using UNION (SET operators)

SELECT doctorname, salary

FROM doctors d1

WHERE (

SELECT COUNT(DISTINCT d2.salary)

FROM doctors d2

WHERE d2.salary > d1.salary

) < 5

UNION

SELECT doctorname, salary

FROM doctors d3

WHERE salary IS NULL

ORDER BY salary DESC;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT doctorname, salary

FROM doctors

EXCEPT

SELECT doctorname, salary

FROM doctors d1

WHERE (

SELECT COUNT(DISTINCT d2.salary)

FROM doctors d2

WHERE d2.salary > d1.salary

) >= 5;

-- Method 7: Using INTERSECT (SET operators)

SELECT doctorname, salary

FROM doctors d1

WHERE (

SELECT COUNT(DISTINCT d2.salary)

FROM doctors d2

WHERE d2.salary > d1.salary

) < 5

INTERSECT

SELECT doctorname, salary

FROM doctors;

\*/

-- 22. List TOP 10 hospitals (with respect to maximum no of patients visiting the hospital)

-- Method 1: Using RANK Function

WITH hospital\_patient\_counts AS (

SELECT h.hospitalid, h.hospitalname, COUNT(DISTINCT a.patientid) AS patient\_count,

RANK() OVER (ORDER BY COUNT(DISTINCT a.patientid) DESC) AS rank\_no

FROM hospitals h

LEFT JOIN doctors d

ON d.hospitalid = h.hospitalid

LEFT JOIN allocation a

ON a.doctorid = d.doctorid

GROUP BY h.hospitalid, h.hospitalname

)

SELECT hospitalname, patient\_count

FROM hospital\_patient\_counts

WHERE rank\_no <= 10

ORDER BY patient\_count DESC;

-- Method 2: Using NOT IN

SELECT c1.hospitalname, c1.patient\_count

FROM (

SELECT h.hospitalid, h.hospitalname, COUNT(DISTINCT a.patientid) AS patient\_count

FROM hospitals h

LEFT JOIN doctors d

ON d.hospitalid = h.hospitalid

LEFT JOIN allocation a

ON a.doctorid = d.doctorid

GROUP BY h.hospitalid, h.hospitalname

) AS c1

WHERE c1.patient\_count NOT IN (

SELECT c2.patient\_count

FROM (

SELECT h2.hospitalid, COUNT(DISTINCT a2.patientid) AS patient\_count

FROM hospitals h2

LEFT JOIN doctors d2

ON d2.hospitalid = h2.hospitalid

LEFT JOIN allocation a2

ON a2.doctorid = d2.doctorid

GROUP BY h2.hospitalid

) AS c2

WHERE (

SELECT COUNT(DISTINCT c3.patient\_count)

FROM (

SELECT h3.hospitalid, COUNT(DISTINCT a3.patientid) AS patient\_count

FROM hospitals h3

LEFT JOIN doctors d3

ON d3.hospitalid = h3.hospitalid

LEFT JOIN allocation a3

ON a3.doctorid = d3.doctorid

GROUP BY h3.hospitalid

) AS c3

WHERE c3.patient\_count > c2.patient\_count

) >= 10

)

ORDER BY c1.patient\_count DESC;

-- Method 3: Using JOIN

SELECT h1.hospitalname, COUNT(DISTINCT a1.patientid) AS patient\_count

FROM hospitals h1

LEFT JOIN doctors d1

ON d1.hospitalid = h1.hospitalid

LEFT JOIN allocation a1

ON a1.doctorid = d1.doctorid

LEFT JOIN hospitals h2

ON h2.hospitalid <> h1.hospitalid

LEFT JOIN doctors d2

ON d2.hospitalid = h2.hospitalid

LEFT JOIN allocation a2

ON a2.doctorid = d2.doctorid

GROUP BY h1.hospitalid, h1.hospitalname

HAVING (

SELECT COUNT(DISTINCT a3.patientid)

FROM hospitals h3

LEFT JOIN doctors d3

ON d3.hospitalid = h3.hospitalid

LEFT JOIN allocation a3

ON a3.doctorid = d3.doctorid

WHERE COUNT(DISTINCT a3.patientid) > COUNT(DISTINCT a1.patientid)

) < 10

ORDER BY patient\_count DESC;

-- Method 4: Using EXISTS / NOT EXISTS (Correlated Subquery)

SELECT c1.hospitalname, c1.patient\_count

FROM counts c1

WHERE NOT EXISTS (

SELECT 1

FROM counts c2

WHERE c2.patient\_count > c1.patient\_count

HAVING COUNT(\*) >= 10

)

ORDER BY c1.patient\_count DESC;

-- Method 5: Using UNION (SET operators)

SELECT c.hospitalname, c.patient\_count

FROM counts c

WHERE (

SELECT COUNT(\*)

FROM counts x

WHERE x.patient\_count > c.patient\_count

) < 10

UNION

SELECT c.hospitalname, c.patient\_count

FROM counts c

WHERE FALSE

ORDER BY patient\_count DESC;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT hospitalname, patient\_count

FROM counts

EXCEPT

SELECT hospitalname, patient\_count

FROM counts c

WHERE (

SELECT COUNT(\*)

FROM counts h

WHERE h.patient\_count > c.patient\_count

) >= 10;

-- Method 7: Using INTERSECT (SET operators)

SELECT hospitalname, patient\_count

FROM counts

INTERSECT

SELECT hospitalname, patient\_count

FROM counts c

WHERE (

SELECT COUNT(\*)

FROM counts h

WHERE h.patient\_count > c.patient\_count

) < 10;

\*/

-- 23. For each hospital display th 3rd maximum salary earner

-- Method 1: Using RANK Function

SELECT hospitalname, doctorname, salary

FROM (

SELECT h.hospitalname, d.doctorname, d.salary,

RANK() OVER (PARTITION BY h.hospitalid ORDER BY d.salary DESC) AS rank\_no

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

) ranked

WHERE rank\_no = 3;

-- Method 2: Using NOT IN

SELECT h.hospitalname, d1.doctorname, d1.salary

FROM doctors d1

JOIN hospitals h

ON d1.hospitalid = h.hospitalid

WHERE d1.salary NOT IN (

SELECT DISTINCT d2.salary

FROM doctors d2

WHERE d2.hospitalid = d1.hospitalid

AND (

SELECT COUNT(DISTINCT d3.salary)

FROM doctors d3

WHERE d3.hospitalid = d2.hospitalid AND d3.salary > d2.salary

) < 2

)

AND (

SELECT COUNT(DISTINCT d4.salary)

FROM doctors d4

WHERE d4.hospitalid = d1.hospitalid AND d4.salary > d1.salary

) = 2;

-- Method 3: Using JOIN

SELECT h.hospitalname, d1.doctorname, d1.salary

FROM doctors d1

JOIN hospitals h

ON d1.hospitalid = h.hospitalid

LEFT JOIN doctors d2

ON d1.hospitalid = d2.hospitalid AND d2.salary > d1.salary

GROUP BY h.hospitalname, d1.doctorname, d1.salary

HAVING COUNT(d2.salary) = 2

ORDER BY h.hospitalname, d1.salary DESC;

-- Method 4: Using EXISTS / NOT EXISTS (Correlated Subquery)

SELECT h.hospitalname, d1.doctorname, d1.salary

FROM doctors d1

JOIN hospitals h

ON h.hospitalid = d1.hospitalid

WHERE EXISTS (

SELECT 1

FROM doctors d2

WHERE d2.hospitalid = d1.hospitalid

AND d2.salary > d1.salary

)

AND NOT EXISTS (

SELECT 1

FROM doctors x

WHERE x.hospitalid = d1.hospitalid

AND x.salary > d1.salary

AND EXISTS (

SELECT 1

FROM doctors y

WHERE y.hospitalid = d1.hospitalid

AND y.salary > x.salary

AND EXISTS (

SELECT 1

FROM doctors z

WHERE z.hospitalid = d1.hospitalid

AND z.salary > y.salary

)

)

)

ORDER BY d1.salary DESC;

-- Method 5: Using UNION (SET operators)

SELECT h.hospitalname, d.doctorname, d.salary

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

WHERE (

SELECT COUNT(DISTINCT d2.salary)

FROM doctors d2

WHERE d2.hospitalid = d.hospitalid AND d2.salary > d.salary

) = 2

UNION

SELECT h.hospitalname, d.doctorname, d.salary

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

WHERE d.salary IS NULL;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT h.hospitalname, d.doctorname, d.salary

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

EXCEPT

SELECT h.hospitalname, d.doctorname, d.salary

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

WHERE (

SELECT COUNT(DISTINCT d2.salary)

FROM doctors d2

WHERE d2.hospitalid = d.hospitalid AND d2.salary > d.salary

) NOT IN (2);

-- Method 7: Using INTERSECT (SET operators)

SELECT h.hospitalname, d.doctorname, d.salary

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid

WHERE (

SELECT COUNT(DISTINCT d2.salary)

FROM doctors d2

WHERE d2.hospitalid = d.hospitalid AND d2.salary > d.salary

) = 2

INTERSECT

SELECT h.hospitalname, d.doctorname, d.salary

FROM doctors d

JOIN hospitals h

ON d.hospitalid = h.hospitalid;

\*/

-- 24. How many diseases are cured by senior most doctor in Fortis Hospital

-- Method 1: Using JOIN

SELECT COUNT(DISTINCT a.diseaseid) AS diseases\_cured

FROM doctors d

JOIN hospitals h

ON h.hospitalid = d.hospitalid

JOIN allocation a

ON a.doctorid = d.doctorid

WHERE h.hospitalname = 'FORTIS'

AND d.doj = (

SELECT MIN(d2.doj)

FROM doctors d2

WHERE d2.hospitalid = h.hospitalid

);

-- Method 2: Using NOT IN

SELECT COUNT(DISTINCT a.diseaseid) AS diseases\_cured

FROM doctors d

JOIN hospitals h

ON h.hospitalid = d.hospitalid

JOIN allocation a

ON a.doctorid = d.doctorid

WHERE h.hospitalname = 'FORTIS'

AND d.doctorid NOT IN (

SELECT d2.doctorid

FROM doctors d2

WHERE d2.hospitalid = h.hospitalid

AND d2.doj > d.doj

);

-- Method 3: Using EXISTS / NOT EXISTS (Correlated Subquery)

SELECT COUNT(DISTINCT a.diseaseid) AS diseases\_cured

FROM doctors d

JOIN hospitals h

ON h.hospitalid = d.hospitalid

JOIN allocation a

ON a.doctorid = d.doctorid

WHERE h.hospitalname = 'FORTIS'

AND NOT EXISTS (

SELECT 1

FROM doctors x

WHERE x.hospitalid = d.hospitalid

AND x.doj < d.doj

);

-- Method 4: Using UNION (SET operators)

SELECT COUNT(DISTINCT diseaseid) AS diseases\_cured

FROM (

SELECT a.diseaseid

FROM doctors d

JOIN hospitals h

ON h.hospitalid = d.hospitalid

JOIN allocation a

ON a.doctorid = d.doctorid

WHERE h.hospitalname = 'FORTIS'

AND d.doj = (

SELECT MIN(d3.doj)

FROM doctors d3

WHERE d3.hospitalid = h.hospitalid

)

UNION

SELECT NULL

WHERE FALSE

) AS u;

/\* Does not work in MySQL

-- Method 5: Using EXCEPT (SET operators)

SELECT COUNT(\*) AS diseases\_cured

FROM (

SELECT a.diseaseid

FROM doctors d

JOIN hospitals h

ON h.hospitalid = d.hospitalid

JOIN allocation a

ON a.doctorid = d.doctorid

WHERE h.hospitalname = 'FORTIS'

) AS all\_fortis

EXCEPT

SELECT a.diseaseid

FROM doctors d

JOIN hospitals h

ON h.hospitalid = d.hospitalid

JOIN allocation a

ON a.doctorid = d.doctorid

WHERE h.hospitalname = 'FORTIS'

AND EXISTS (

SELECT 1

FROM doctors z

WHERE z.hospitalid = d.hospitalid

AND z.doj < d.doj

);

-- Method 6: Using INTERSECT (SET operators)

SELECT COUNT(\*) AS diseases\_cured

FROM (

SELECT a.diseaseid

FROM doctors d

JOIN hospitals h

ON h.hospitalid = d.hospitalid

JOIN allocation a

ON a.doctorid = d.doctorid

WHERE h.hospitalname = 'FORTIS'

)

INTERSECT

SELECT a.diseaseid

FROM doctors d

JOIN hospitals h

ON h.hospitalid = d.hospitalid

JOIN allocation a

ON a.doctorid = d.doctorid

WHERE h.hospitalname = 'FORTIS'

AND NOT EXISTS (

SELECT 1

FROM doctors z

WHERE z.hospitalid = d.hospitalid

AND z.doj < d.doj

);

\*/

-- 25. List out the frequently caused diseases by minimum salary earner

-- Method 1: Using JOIN

SELECT ds.diseasename, c.visit\_cnt

FROM (

SELECT a.diseaseid, COUNT(\*) AS visit\_cnt

FROM allocation a

WHERE a.doctorid IN (

SELECT doctorid

FROM doctors

WHERE salary = (

SELECT MIN(salary)

FROM doctors

)

)

GROUP BY a.diseaseid

) AS c

JOIN diseases ds

ON ds.diseaseid = c.diseaseid

WHERE c.visit\_cnt = (

SELECT MAX(sub.visit\_cnt)

FROM (

SELECT COUNT(\*) AS visit\_cnt

FROM allocation a2

WHERE a2.doctorid IN (

SELECT doctorid

FROM doctors

WHERE salary = (

SELECT MIN(salary)

FROM doctors

)

)

GROUP BY a2.diseaseid

) AS sub

);

-- Method 2: Using NOT IN

SELECT ds.diseasename, cnts.visit\_cnt

FROM (

SELECT a.diseaseid, COUNT(\*) AS visit\_cnt

FROM allocation a

WHERE a.doctorid IN (

SELECT doctorid

FROM doctors

WHERE salary = (

SELECT MIN(salary)

FROM doctors

)

)

GROUP BY a.diseaseid

) AS cnts

JOIN diseases ds

ON ds.diseaseid = cnts.diseaseid

WHERE cnts.visit\_cnt NOT IN (

SELECT c2.visit\_cnt

FROM (

SELECT COUNT(\*) AS visit\_cnt

FROM allocation a2

WHERE a2.doctorid IN (

SELECT doctorid

FROM doctors

WHERE salary = (

SELECT MIN(salary)

FROM doctors

)

)

GROUP BY a2.diseaseid

) AS c2

WHERE c2.visit\_cnt > cnts.visit\_cnt

);

-- Method 3: Using NOT EXISTS (Correlated Subquery)

SELECT ds.diseasename, COUNT(\*) AS visit\_cnt

FROM allocation a

JOIN diseases ds

ON ds.diseaseid = a.diseaseid

WHERE a.doctorid IN (

SELECT doctorid

FROM doctors

WHERE salary = (

SELECT MIN(salary)

FROM doctors

)

)

GROUP BY a.diseaseid, ds.diseasename

HAVING NOT EXISTS (

SELECT 1

FROM (

SELECT a2.diseaseid, COUNT(\*) AS vc

FROM allocation a2

WHERE a2.doctorid IN (

SELECT doctorid

FROM doctors

WHERE salary = (

SELECT MIN(salary)

FROM doctors

)

)

GROUP BY a2.diseaseid

) AS x

WHERE x.vc > COUNT(\*)

);

-- Method 4: Using UNION (SET operators)

SELECT ds.diseasename, c.visit\_cnt

FROM (

SELECT a.diseaseid, COUNT(\*) AS visit\_cnt

FROM allocation a

WHERE a.doctorid IN (

SELECT doctorid

FROM doctors

WHERE salary = (

SELECT MIN(salary)

FROM doctors

)

)

GROUP BY a.diseaseid

) AS c

JOIN diseases ds

ON ds.diseaseid = c.diseaseid

WHERE c.visit\_cnt = (

SELECT MAX(visit\_cnt)

FROM (

SELECT COUNT(\*) AS visit\_cnt

FROM allocation a2

WHERE a2.doctorid IN (

SELECT doctorid

FROM doctors

WHERE salary = (

SELECT MIN(salary)

FROM doctors

)

)

GROUP BY a2.diseaseid

) AS y

)

UNION

SELECT NULL, NULL

WHERE FALSE;

/\* Does not work in MySQL

-- Method 5: Using EXCEPT (SET operators)

SELECT diseasename, visit\_cnt

FROM (

SELECT ds.diseasename, COUNT(\*) AS visit\_cnt

FROM allocation a

JOIN diseases ds

ON ds.diseaseid = a.diseaseid

WHERE a.doctorid IN (

SELECT doctorid

FROM doctors

WHERE salary = (

SELECT MIN(salary)

FROM doctors

)

)

GROUP BY ds.diseasename

) q

EXCEPT

SELECT diseasename, visit\_cnt

FROM (

SELECT ds.diseasename, COUNT(\*) AS visit\_cnt

FROM allocation a

JOIN diseases ds

ON ds.diseaseid = a.diseaseid

WHERE a.doctorid IN (

SELECT doctorid

FROM doctors

WHERE salary = (

SELECT MIN(salary)

FROM doctors

)

)

GROUP BY ds.diseasename

) z

WHERE visit\_cnt < (

SELECT MAX(vc)

FROM (

SELECT COUNT(\*) AS vc

FROM allocation a3

WHERE a3.doctorid IN (

SELECT doctorid

FROM doctors

WHERE salary = (

SELECT MIN(salary)

FROM doctors

)

)

GROUP BY a3.diseaseid

) m

);

-- Method 6: Using INTERSECT (SET operators)

SELECT diseasename, visit\_cnt

FROM (

SELECT ds.diseasename, COUNT(\*) AS visit\_cnt

FROM allocation a

JOIN diseases ds

ON ds.diseaseid = a.diseaseid

WHERE a.doctorid IN (

SELECT doctorid

FROM doctors

WHERE salary = (

SELECT MIN(salary)

FROM doctors

)

)

GROUP BY ds.diseasename

) q

INTERSECT

SELECT diseasename, visit\_cnt

FROM (

SELECT ds.diseasename, COUNT(\*) AS visit\_cnt

FROM allocation a

JOIN diseases ds

ON ds.diseaseid = a.diseaseid

WHERE a.doctorid IN (

SELECT doctorid

FROM doctors

WHERE salary = (

SELECT MIN(salary)

FROM doctors

)

)

GROUP BY ds.diseasename

) q2

WHERE visit\_cnt = (

SELECT MAX(vc)

FROM (

SELECT COUNT(\*) AS vc

FROM allocation a4

WHERE a4.doctorid IN (

SELECT doctorid

FROM doctors

WHERE salary = (

SELECT MIN(salary)

FROM doctors

)

)

GROUP BY a4.diseaseid

) t

);

\*/

-- 26. Generate a report: Doctor ID | Doctor Name | No of Patients cured

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS sp\_doctor\_patients\_cured;

DELIMITER $$

CREATE PROCEDURE sp\_doctor\_patients\_cured()

BEGIN

SELECT d.doctorid, d.doctorname, COUNT(DISTINCT a.patientid) AS no\_of\_patients\_cured

FROM doctors d

LEFT JOIN allocation a

ON d.doctorid = a.doctorid

GROUP BY d.doctorid, d.doctorname

ORDER BY d.doctorid;

END $$

DELIMITER ;

CALL sp\_doctor\_patients\_cured();

-- Method 2: Using JOIN

SELECT d.doctorid, d.doctorname, COUNT(DISTINCT a.patientid) AS no\_of\_patients\_cured

FROM doctors d

LEFT JOIN allocation a

ON d.doctorid = a.doctorid

GROUP BY d.doctorid, d.doctorname

ORDER BY d.doctorid;

-- Method 3: Using WITH

WITH PatientCounts AS (

SELECT doctorid, COUNT(DISTINCT patientid) AS no\_of\_patients\_cured

FROM allocation

GROUP BY doctorid

)

SELECT d.doctorid, d.doctorname, COALESCE(pc.no\_of\_patients\_cured, 0) AS no\_of\_patients\_cured

FROM doctors d

LEFT JOIN PatientCounts pc

ON d.doctorid = pc.doctorid

ORDER BY d.doctorid;

-- Method 4: Using NOT IN

SELECT d.doctorid, d.doctorname, (

SELECT COUNT(DISTINCT a.patientid)

FROM allocation a

WHERE a.doctorid = d.doctorid AND a.patientid NOT IN (

SELECT patientid

FROM allocation a2

WHERE a2.doctorid <> d.doctorid

)

) AS no\_of\_patients\_cured

FROM doctors d

ORDER BY d.doctorid;

-- Method 5: Using EXISTS (Correlated Subquery)

SELECT d.doctorid, d.doctorname, (

SELECT COUNT(DISTINCT a.patientid)

FROM allocation a

WHERE a.doctorid = d.doctorid AND EXISTS (

SELECT 1

FROM allocation a2

WHERE a2.patientid = a.patientid AND a2.doctorid = d.doctorid

)

) AS no\_of\_patients\_cured

FROM doctors d

ORDER BY d.doctorid;

-- Method 6 Using UNION (SET operators)

SELECT doctorid, doctorname, no\_of\_patients\_cured

FROM (

SELECT d.doctorid, d.doctorname, COUNT(DISTINCT a.patientid) AS no\_of\_patients\_cured

FROM doctors d

LEFT JOIN allocation a

ON d.doctorid = a.doctorid

GROUP BY d.doctorid, d.doctorname

UNION

SELECT d2.doctorid, d2.doctorname, 0 AS no\_of\_patients\_cured

FROM doctors d2

WHERE d2.doctorid NOT IN (

SELECT doctorid

FROM allocation

)

) AS combined

ORDER BY doctorid;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT d.doctorid, d.doctorname, COUNT(DISTINCT a.patientid) AS no\_of\_patients\_cured

FROM doctors d

JOIN allocation a

ON d.doctorid = a.doctorid

GROUP BY d.doctorid, d.doctorname

EXCEPT

SELECT d2.doctorid, d2.doctorname, 0 AS no\_of\_patients\_cured

FROM doctors d2

WHERE d2.doctorid NOT IN (

SELECT doctorid

FROM allocation

)

ORDER BY doctorid;

-- Method 8: Using INTERSECT (SET operators)

SELECT d.doctorid, d.doctorname, COUNT(DISTINCT a.patientid) AS no\_of\_patients\_cured

FROM doctors d

JOIN allocation a

ON d.doctorid = a.doctorid

GROUP BY d.doctorid, d.doctorname

INTERSECT

SELECT d2.doctorid, d2.doctorname, 0 AS no\_of\_patients\_cured

FROM doctors d2

WHERE d2.doctorid IN (

SELECT doctorid

FROM allocation

)

ORDER BY doctorid;

\*/

-- 27. Which doctor has cured all the diseaes

-- Method 1: Using JOIN

SELECT d.doctorid, d.doctorname

FROM doctors d

JOIN allocation a

ON d.doctorid = a.doctorid

GROUP BY d.doctorid, d.doctorname

HAVING COUNT(DISTINCT a.diseaseid) = (

SELECT COUNT(\*)

FROM diseases

);

-- Method 2: Using NOT IN

SELECT d.doctorid, d.doctorname

FROM doctors d

WHERE NOT EXISTS (

SELECT diseaseid

FROM diseases

WHERE diseaseid NOT IN (

SELECT diseaseid

FROM allocation a

WHERE a.doctorid = d.doctorid

)

);

-- Method 3: Using NOT EXISTS (Correlated Subquery)

SELECT d.doctorid, d.doctorname

FROM doctors d

WHERE NOT EXISTS (

SELECT 1

FROM diseases dis

WHERE NOT EXISTS (

SELECT 1

FROM allocation a

WHERE a.doctorid = d.doctorid AND a.diseaseid = dis.diseaseid

)

);

-- Method 4: Using UNION (SET operators)

SELECT doctorid, doctorname

FROM doctors d

WHERE doctorid IN (

SELECT doctorid

FROM allocation

)

UNION

SELECT doctorid, doctorname

FROM doctors d

WHERE doctorid NOT IN (

SELECT doctorid

FROM allocation

);

SELECT doctorid, doctorname

FROM (

SELECT d.doctorid, d.doctorname, COUNT(DISTINCT a.diseaseid) AS disease\_count

FROM doctors d

LEFT JOIN allocation a

ON d.doctorid = a.doctorid

GROUP BY d.doctorid, d.doctorname

) AS sub

WHERE disease\_count = (SELECT COUNT(\*) FROM diseases);

/\* Does not work in MySQL

-- Method 5: Using EXCEPT (SET operators)

SELECT doctorid

FROM allocation

GROUP BY doctorid

EXCEPT

SELECT doctorid

FROM allocation a

RIGHT JOIN diseases dis

ON a.diseaseid = dis.diseaseid

WHERE a.doctorid IS NULL;

-- Method 6: Using INTERSECT (SET operators)

SELECT doctorid

FROM allocation

INTERSECT

SELECT doctorid

FROM doctors;

\*/

-- 28. Which patient has not visited hospital so far (ALL POSSIBILITIES) not in or related subquery outer join

-- Method 1: Using LEFT JOIN

SELECT p.patientid, p.patientname

FROM patients p

LEFT JOIN allocation a

ON p.patientid = a.patientid

WHERE a.patientid IS NULL;

-- Method 2: Using NOT IN

SELECT patientid, patientname

FROM patients

WHERE patientid NOT IN (

SELECT DISTINCT patientid

FROM allocation

);

-- Method 3: Using NOT EXISTS (Correlated Subquery)

SELECT p.patientid, p.patientname

FROM patients p

WHERE NOT EXISTS (

SELECT 1

FROM allocation a

WHERE a.patientid = p.patientid

);

/\* Does not work in MySQL

-- Method 4: Using UNION (SET operators)

SELECT patientid, patientname

FROM patients

UNION

SELECT patientid, patientname

FROM patients p

JOIN allocation a

ON p.patientid = a.patientid;

SELECT patientid, patientname

FROM patients

EXCEPT

SELECT patientid, patientname

FROM patients p

JOIN allocation a

ON p.patientid = a.patientid;

-- Method 5: Using EXCEPT (SET operators)

SELECT patientid, patientname

FROM patients

EXCEPT

SELECT patientid, patientname

FROM patients p

JOIN allocation a ON p.patientid = a.patientid;

\*/

-- 29. (Using ROLL UP) Generate a report: Doctor ID | Patient ID | No of visits

-- Method 1: Using ROLLUP

SELECT a.doctorid, a.patientid, COUNT(\*) AS noofvisits

FROM allocation a

GROUP BY a.doctorid, a.patientid WITH ROLLUP;

-- 30. (Using ROLL UP) Generate a report: Patient ID | Disease ID

-- Method 1: Using ROLLUP

SELECT a.patientid, a.diseaseid, COUNT(\*) AS noofcases

FROM allocation a

GROUP BY a.patientid, a.diseaseid WITH ROLLUP;

-- 31. (Using NVL/NVL2) Generate a report: Doctor ID | Patient ID | Disease ID | Comments

-- If number of days stayed is NULL, then the comment must be “Treatment going on”

-- Method 1: Returning one value if not null

SELECT a.doctorid, a.patientid, a.diseaseid, IFNULL(a.noofdays, 'Treatment going on') AS Comments

FROM allocation a;

-- 32. How many patients are undergoing treatment in Apollo hospital for the coronary related diseases.

-- Method 1: Using JOIN

SELECT COUNT(DISTINCT a.patientid) AS NoOfPatients

FROM allocation a

JOIN doctors d ON a.doctorid = d.doctorid

JOIN hospitals h ON d.hospitalid = h.hospitalid

JOIN diseases di ON a.diseaseid = di.diseaseid

WHERE h.hospitalname = 'APOLLO' AND di.categoryid = 'C06';

-- Method 2: Using NOT IN

SELECT COUNT(DISTINCT patientid) AS NoOfPatients

FROM allocation

WHERE doctorid IN (

SELECT doctorid

FROM doctors

WHERE hospitalid = 'H01'

)

AND diseaseid IN (

SELECT diseaseid

FROM diseases

WHERE categoryid = 'C06'

);

-- Method 3: Using Subquery

SELECT COUNT(DISTINCT patientid) AS NoOfPatients

FROM (

SELECT a.patientid

FROM allocation a

JOIN doctors d ON a.doctorid = d.doctorid

JOIN diseases di ON a.diseaseid = di.diseaseid

WHERE d.hospitalid = 'H01'

AND di.categoryid = 'C06'

) AS sub;

-- Method 4: Using EXISTS (Correlated Subquery)

SELECT COUNT(DISTINCT a.patientid) AS NoOfPatients

FROM allocation a

WHERE EXISTS (

SELECT 1

FROM doctors d

WHERE d.doctorid = a.doctorid AND d.hospitalid = 'H01'

)

AND EXISTS (

SELECT 1

FROM diseases di

WHERE di.diseaseid = a.diseaseid AND di.categoryid = 'C06'

);

-- Method 5: Using UNION (SET operators)

SELECT DISTINCT a.patientid

FROM allocation a

JOIN doctors d

ON a.doctorid = d.doctorid

WHERE d.hospitalid = 'H01'

UNION

SELECT DISTINCT a.patientid

FROM allocation a

JOIN diseases di

ON a.diseaseid = di.diseaseid

WHERE di.categoryid = 'C06';

/\* Does not work in MySQL

-- Method 6: Using INTERSECT (SET operators)

SELECT DISTINCT a.patientid

FROM allocation a

JOIN doctors d

ON a.doctorid = d.doctorid

WHERE d.hospitalid = 'H01'

INTERSECT

SELECT DISTINCT a.patientid

FROM allocation a

JOIN diseases di

ON a.diseaseid = di.diseaseid

WHERE di.categoryid = 'C06';

\*/

-- 33. Which category has maximum diseases

-- Method 1: Using RANK Function

WITH cte AS (

SELECT c.categoryid, c.categoryname, COUNT(d.diseaseid) AS dcnt, RANK() OVER (ORDER BY COUNT(d.diseaseid) DESC) AS rnk

FROM categories c

LEFT JOIN diseases d

ON d.categoryid = c.categoryid

GROUP BY c.categoryid, c.categoryname

)

SELECT categoryid, categoryname, dcnt

FROM cte

WHERE rnk = 1;

-- Method 2: Using NOT IN

SELECT c.categoryid, c.categoryname, COUNT(d.diseaseid) AS dcnt

FROM categories c

LEFT JOIN diseases d

ON d.categoryid = c.categoryid

GROUP BY c.categoryid, c.categoryname

HAVING COUNT(d.diseaseid) NOT IN (

SELECT COUNT(d2.diseaseid)

FROM categories c2

LEFT JOIN diseases d2

ON d2.categoryid = c2.categoryid

GROUP BY c2.categoryid

HAVING COUNT(d2.diseaseid) > COUNT(d.diseaseid)

);

-- Method 3: Using JOIN

WITH counts AS (

SELECT categoryid, COUNT(diseaseid) AS dcnt

FROM diseases

GROUP BY categoryid

), mx AS (

SELECT MAX(dcnt) AS maxcnt

FROM counts

)

SELECT c.categoryid, c.categoryname, cnt.dcnt

FROM counts cnt

JOIN mx

ON cnt.dcnt = mx.maxcnt

JOIN categories c

ON c.categoryid = cnt.categoryid;

-- Method 4: Using NOT EXISTS (Correlated Subquery)

SELECT c.categoryid, c.categoryname, COUNT(d.diseaseid) AS dcnt

FROM categories c

LEFT JOIN diseases d

ON d.categoryid = c.categoryid

GROUP BY c.categoryid, c.categoryname

HAVING NOT EXISTS (

SELECT 1

FROM categories x

LEFT JOIN diseases dx

ON dx.categoryid = x.categoryid

GROUP BY x.categoryid

HAVING COUNT(dx.diseaseid) > COUNT(d.diseaseid)

);

-- Method 5: Using UNION (SET operators)

SELECT categoryid, categoryname, dcnt

FROM (

SELECT c.categoryid, c.categoryname, COUNT(d.diseaseid) AS dcnt

FROM categories c

LEFT JOIN diseases d

ON d.categoryid = c.categoryid

GROUP BY c.categoryid, c.categoryname

) q

WHERE dcnt = (

SELECT MAX(dcnt)

FROM (

SELECT categoryid, COUNT(diseaseid) AS dcnt

FROM diseases

GROUP BY categoryid

) z

)

UNION

SELECT NULL, NULL, NULL

WHERE FALSE;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT c.categoryid, c.categoryname, COUNT(d.diseaseid) AS dcnt

FROM categories c

LEFT JOIN diseases d

ON d.categoryid = c.categoryid

GROUP BY c.categoryid, c.categoryname

EXCEPT

SELECT c.categoryid, c.categoryname, COUNT(d.diseaseid) AS dcnt

FROM categories c

LEFT JOIN diseases d

ON d.categoryid = c.categoryid

GROUP BY c.categoryid, c.categoryname

HAVING COUNT(d.diseaseid) < (

SELECT MAX(cnt)

FROM (

SELECT COUNT(diseaseid) AS cnt

FROM diseases

GROUP BY categoryid

) m

);

-- Method 7: Using INTERSECT (SET operators)

SELECT c.categoryid, c.categoryname, COUNT(d.diseaseid) AS dcnt

FROM categories c

LEFT JOIN diseases d

ON d.categoryid = c.categoryid

GROUP BY c.categoryid, c.categoryname

INTERSECT

SELECT c.categoryid, c.categoryname, maxcnt

FROM categories c

JOIN (

SELECT MAX(cnt) AS maxcnt

FROM (

SELECT COUNT(diseaseid) AS cnt

FROM diseases

GROUP BY categoryid

) y

) mx

ON 1=1

JOIN (

SELECT categoryid, COUNT(diseaseid) AS cnt

FROM diseases

GROUP BY categoryid

) z

ON z.categoryid = c.categoryid

AND z.cnt = mx.maxcnt;

\*/

-- 34. Which hospital has minimum doctors

-- Method 1: Using RANK Function

WITH cte AS (

SELECT h.hospitalid, h.hospitalname, COUNT(d.doctorid) AS dcnt, RANK() OVER (ORDER BY COUNT(d.doctorid) ASC) AS rnk

FROM hospitals h

LEFT JOIN doctors d

ON d.hospitalid = h.hospitalid

GROUP BY h.hospitalid, h.hospitalname

)

SELECT hospitalid, hospitalname, dcnt

FROM cte

WHERE rnk = 1;

-- Method 2: Using NOT IN

SELECT h.hospitalid, h.hospitalname, COUNT(d.doctorid) AS dcnt

FROM hospitals h

LEFT JOIN doctors d

ON d.hospitalid = h.hospitalid

GROUP BY h.hospitalid, h.hospitalname

HAVING COUNT(d.doctorid) NOT IN (

SELECT COUNT(d2.doctorid)

FROM hospitals h2

LEFT JOIN doctors d2

ON d2.hospitalid = h2.hospitalid

GROUP BY h2.hospitalid

HAVING COUNT(d2.doctorid) < COUNT(d.doctorid)

);

-- Method 3: Using JOIN

WITH counts AS (

SELECT hospitalid, COUNT(doctorid) AS dcnt

FROM doctors

GROUP BY hospitalid

), mn AS (

SELECT MIN(dcnt) AS mincnt

FROM counts

)

SELECT h.hospitalid, h.hospitalname, c.dcnt

FROM counts c

JOIN mn

ON c.dcnt = mn.mincnt

JOIN hospitals h

ON h.hospitalid = c.hospitalid;

-- Method 4: Using NOT EXISTS (Correlated Subquery)

SELECT h.hospitalid, h.hospitalname, COUNT(d.doctorid) AS dcnt

FROM hospitals h

LEFT JOIN doctors d

ON d.hospitalid = h.hospitalid

GROUP BY h.hospitalid, h.hospitalname

HAVING NOT EXISTS (

SELECT 1

FROM hospitals x

LEFT JOIN doctors dx

ON dx.hospitalid = x.hospitalid

GROUP BY x.hospitalid

HAVING COUNT(dx.doctorid) < COUNT(d.doctorid)

);

-- Method 5: Using UNION (SET operators)

SELECT hospitalid, hospitalname, dcnt

FROM (

SELECT h.hospitalid, h.hospitalname, COUNT(d.doctorid) AS dcnt

FROM hospitals h

LEFT JOIN doctors d

ON d.hospitalid = h.hospitalid

GROUP BY h.hospitalid, h.hospitalname

) q

WHERE dcnt = (

SELECT MIN(dcnt)

FROM (

SELECT hospitalid, COUNT(doctorid) AS dcnt

FROM doctors

GROUP BY hospitalid

) z

)

UNION

SELECT NULL, NULL, NULL

WHERE FALSE;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT h.hospitalid, h.hospitalname, COUNT(d.doctorid) AS dcnt

FROM hospitals h

LEFT JOIN doctors d

ON d.hospitalid = h.hospitalid

GROUP BY h.hospitalid, h.hospitalname

EXCEPT

SELECT h.hospitalid, h.hospitalname, COUNT(d.doctorid) AS dcnt

FROM hospitals h

LEFT JOIN doctors d

ON d.hospitalid = h.hospitalid

GROUP BY h.hospitalid, h.hospitalname

HAVING COUNT(d.doctorid) > (

SELECT MIN(cnt)

FROM (

SELECT COUNT(doctorid) AS cnt

FROM doctors

GROUP BY hospitalid

) m

);

-- Method 7: Using INTERSECT (SET operators)

SELECT h.hospitalid, h.hospitalname, COUNT(d.doctorid) AS dcnt

FROM hospitals h

LEFT JOIN doctors d

ON d.hospitalid = h.hospitalid

GROUP BY h.hospitalid, h.hospitalname

INTERSECT

SELECT h.hospitalid, h.hospitalname, mincnt

FROM hospitals h

JOIN (

SELECT MIN(cnt) AS mincnt

FROM (

SELECT COUNT(doctorid) AS cnt

FROM doctors

GROUP BY hospitalid

) x

) mn

ON 1=1

JOIN (

SELECT hospitalid, COUNT(doctorid) AS cnt

FROM doctors

GROUP BY hospitalid

) c

ON c.hospitalid = h.hospitalid

AND c.cnt = mn.mincnt;

\*/

-- 35. Prepare a hierarchical report: Dean Name | Doctorname

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS rpt\_dean\_doctor\_hierarchy;

DELIMITER $$

CREATE PROCEDURE rpt\_dean\_doctor\_hierarchy()

BEGIN

SELECT d2.doctorname AS dean\_name, d1.doctorname AS doctor\_name

FROM doctors d1

LEFT JOIN doctors d2

ON d1.dean = d2.doctorid

ORDER BY dean\_name, doctor\_name;

END $$

DELIMITER ;

CALL rpt\_dean\_doctor\_hierarchy();

-- Method 2: Using LEFT JOIN

SELECT d2.doctorname AS dean\_name, d1.doctorname AS doctor\_name

FROM doctors d1

LEFT JOIN doctors d2

ON d1.dean = d2.doctorid

ORDER BY dean\_name, doctor\_name;

-- Method 3: Using WITH

WITH dean\_cte AS (

SELECT doctorid, doctorname AS dean\_name

FROM doctors

)

SELECT dc.dean\_name, d.doctorname AS doctor\_name

FROM doctors d

LEFT JOIN dean\_cte dc

ON d.dean = dc.doctorid

ORDER BY dc.dean\_name, d.doctorname;

-- Method 4: Using NOT IN

SELECT (

SELECT d2.doctorname

FROM doctors d2

WHERE d2.doctorid = d1.dean

) AS dean\_name, d1.doctorname AS doctor\_name

FROM doctors d1

WHERE d1.doctorid NOT IN (

SELECT doctorid

FROM doctors

WHERE dean IS NULL

)

OR d1.dean IS NULL

ORDER BY dean\_name, doctor\_name;

-- Method 5: Using EXISTS / NOT EXISTS (Correlated Subquery)

SELECT (

SELECT d2.doctorname

FROM doctors d2

WHERE d2.doctorid = d1.dean

) AS dean\_name, d1.doctorname AS doctor\_name

FROM doctors d1

WHERE EXISTS (

SELECT 1

)

ORDER BY dean\_name, doctor\_name;

-- Method 6 Using UNION ALL (SET operators)

SELECT d2.doctorname AS dean\_name, d1.doctorname AS doctor\_name

FROM doctors d1

JOIN doctors d2

ON d1.dean = d2.doctorid

UNION ALL

SELECT NULL AS dean\_name, doctorname AS doctor\_name

FROM doctors

WHERE dean IS NULL

ORDER BY dean\_name, doctor\_name;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT d2.doctorname, d1.doctorname

FROM doctors d1

JOIN doctors d2

ON d1.dean = d2.doctorid

EXCEPT

SELECT d2.doctorname, d1.doctorname

FROM doctors d1

JOIN doctors d2

ON d1.dean = d2.doctorid

WHERE 1 = 0;

-- Method 8: Using INTERSECT (SET operators)

SELECT d2.doctorname, d1.doctorname

FROM doctors d1

JOIN doctors d2

ON d1.dean = d2.doctorid

INTERSECT

SELECT d2.doctorname, d1.doctorname

FROM doctors d1

JOIN doctors d2

ON d1.dean = d2.doctorid;

\*/

-- 36. How many patients are allocated to the senior most doctors in June 2013?

-- Method 1: Using JOIN

SELECT COUNT(DISTINCT A.patientid) AS patients\_allocated

FROM allocation A

JOIN doctors D

ON A.doctorid = D.doctorid

WHERE D.joiningdate = (

SELECT MIN(joiningdate)

FROM doctors

)

AND MONTH(A.admitteddate) = 6

AND YEAR(A.admitteddate) = 2013;

-- Method 2: Using NOT IN

SELECT COUNT(DISTINCT patientid) AS patients\_allocated

FROM allocation

WHERE doctorid NOT IN (

SELECT doctorid

FROM doctors

WHERE joiningdate > (

SELECT MIN(joiningdate)

FROM doctors

)

)

AND MONTH(admitteddate) = 6

AND YEAR(admitteddate) = 2013;

-- Method 3: Using Subquery

SELECT COUNT(\*) AS patients\_allocated

FROM (

SELECT DISTINCT A.patientid

FROM allocation A

WHERE A.doctorid IN (

SELECT doctorid

FROM doctors

WHERE joiningdate = (

SELECT MIN(joiningdate)

FROM doctors

)

)

AND MONTH(A.admitteddate) = 6

AND YEAR(A.admitteddate) = 2013

) AS senior\_patients;

-- Method 4: Using EXISTS (Correlated Subquery)

SELECT COUNT(DISTINCT A.patientid) AS patients\_allocated

FROM allocation A

WHERE EXISTS (

SELECT 1

FROM doctors D

WHERE D.doctorid = A.doctorid

AND D.joiningdate = (

SELECT MIN(joiningdate)

FROM doctors

)

)

AND MONTH(A.admitteddate) = 6

AND YEAR(A.admitteddate) = 2013;

-- Method 5: Using UNION (SET operators)

SELECT COUNT(\*) AS patients\_allocated

FROM (

SELECT DISTINCT patientid

FROM allocation

WHERE doctorid IN (

SELECT doctorid

FROM doctors

WHERE joiningdate = (

SELECT MIN(joiningdate)

FROM doctors

)

)

AND MONTH(admitteddate) = 6

AND YEAR(admitteddate) = 2013

UNION

SELECT patientid

FROM allocation

WHERE 1 = 0 /\* Ensures only 1st SELECT used \*/

) AS union\_result;

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT COUNT(\*) AS patients\_allocated

FROM (

SELECT DISTINCT A.patientid

FROM allocation A

JOIN doctors D

ON A.doctorid = D.doctorid

WHERE D.joiningdate = (

SELECT MIN(joiningdate)

FROM doctors

)

AND MONTH(A.admitteddate) = 6

AND YEAR(A.admitteddate) = 2013

EXCEPT

SELECT patientid

FROM allocation

WHERE 1 = 0

) AS except\_result;

-- Method 7: Using INTERSECT (SET operators)

SELECT COUNT(\*) AS patients\_allocated

FROM (

SELECT DISTINCT A.patientid

FROM allocation A

JOIN doctors D

ON A.doctorid = D.doctorid

WHERE MONTH(A.admitteddate) = 6

AND YEAR(A.admitteddate) = 2013

INTERSECT

SELECT DISTINCT A.patientid

FROM allocation A

JOIN doctors D

ON A.doctorid = D.doctorid

WHERE D.joiningdate = (

SELECT MIN(joiningdate)

FROM doctors

)

) AS intersect\_result;

\*/

-- 37. List out the senior most patient details who had stayed for more than the expected days for the disease “Jaundice”?

-- Method 1: Using RANK Function

WITH overstay AS (

SELECT p.patientid, p.patientname, p.dob

FROM patients p

JOIN allocation a

ON a.patientid = p.patientid

JOIN diseases d

ON d.diseaseid = a.diseaseid

WHERE d.diseasename = 'JAUNDICE' AND a.noofdays > d.daystorecover

), ranked AS (

SELECT patientid, patientname, dob,

RANK() OVER (ORDER BY dob ASC) AS rnk

FROM overstay

)

SELECT patientid, patientname, dob

FROM ranked

WHERE rnk = 1;

-- Method 2: Using NOT IN

SELECT p.patientid, p.patientname, p.dob

FROM patients p

WHERE p.patientid NOT IN (

SELECT o.patientid

FROM (

SELECT p2.patientid, p2.dob

FROM patients p2

JOIN allocation a2

ON a2.patientid = p2.patientid

JOIN diseases d2

ON d2.diseaseid = a2.diseaseid

WHERE d2.diseasename = 'JAUNDICE'

AND a2.noofdays > d2.daystorecover

) AS o

WHERE o.dob > (

SELECT MIN(p3.dob)

FROM patients p3

JOIN allocation a3

ON a3.patientid = p3.patientid

JOIN diseases d3

ON d3.diseaseid = a3.diseaseid

WHERE d3.diseasename = 'JAUNDICE'

AND a3.noofdays > d3.daystorecover

)

);

-- Method 3: Using JOIN

WITH overstay AS (

SELECT p.patientid, p.patientname, p.dob

FROM patients p

JOIN allocation a

ON a.patientid = p.patientid

JOIN diseases d

ON d.diseaseid = a.diseaseid

WHERE d.diseasename = 'JAUNDICE'

AND a.noofdays > d.daystorecover

), oldest AS (

SELECT MIN(dob) AS mindob

FROM overstay

)

SELECT o.patientid, o.patientname, o.dob

FROM overstay o

JOIN oldest x

ON o.dob = x.mindob;

-- Method 4: Using EXISTS / NOT EXISTS (Correlated Subquery)

SELECT p.patientid, p.patientname, p.dob

FROM patients p

WHERE EXISTS (

SELECT 1

FROM allocation a

JOIN diseases d

ON d.diseaseid = a.diseaseid

WHERE a.patientid = p.patientid

AND d.diseasename = 'JAUNDICE'

AND a.noofdays > d.daystorecover

)

AND NOT EXISTS (

SELECT 1

FROM patients q

JOIN allocation a2

ON a2.patientid = q.patientid

JOIN diseases d2

ON d2.diseaseid = a2.diseaseid

WHERE d2.diseasename = 'JAUNDICE'

AND a2.noofdays > d2.daystorecover

AND q.dob < p.dob

);

-- Method 5: Using UNION (SET operators)

SELECT patientid, patientname, dob

FROM (

SELECT p.patientid, p.patientname, p.dob

FROM patients p

JOIN allocation a

ON a.patientid = p.patientid

JOIN diseases d

ON d.diseaseid = a.diseaseid

WHERE d.diseasename = 'JAUNDICE'

AND a.noofdays > d.daystorecover

UNION ALL

SELECT NULL, NULL, NULL /\* Dummy to satisfy union requirement \*/

WHERE FALSE

) t

WHERE dob = (

SELECT MIN(p2.dob)

FROM patients p2

JOIN allocation a2

ON a2.patientid = p2.patientid

JOIN diseases d2

ON d2.diseaseid = a2.diseaseid

WHERE d2.diseasename = 'JAUNDICE'

AND a2.noofdays > d2.daystorecover

);

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT patientid, patientname, dob

FROM (

SELECT p.patientid, p.patientname, p.dob

FROM patients p

JOIN allocation a

ON a.patientid = p.patientid

JOIN diseases d

ON d.diseaseid = a.diseaseid

WHERE d.diseasename = 'JAUNDICE'

AND a.noofdays > d.daystorecover

) all\_overstay

EXCEPT

SELECT patientid, patientname, dob

FROM (

SELECT p.patientid, p.patientname, p.dob

FROM patients p

JOIN allocation a

ON a.patientid = p.patientid

JOIN diseases d

ON d.diseaseid = a.diseaseid

WHERE d.diseasename = 'JAUNDICE'

AND a.noofdays > d.daystorecover

AND p.dob > (

SELECT MIN(dob)

FROM patients x

JOIN allocation a3

ON a3.patientid = x.patientid

JOIN diseases d3

ON d3.diseaseid = a3.diseaseid

WHERE d3.diseasename = 'JAUNDICE'

AND a3.noofdays > d3.daystorecover

)

) younger;

-- Method 7: Using INTERSECT (SET operators)

SELECT patientid, patientname, dob

FROM (

SELECT p.patientid, p.patientname, p.dob

FROM patients p

JOIN allocation a

ON a.patientid = p.patientid

JOIN diseases d

ON d.diseaseid = a.diseaseid

WHERE d.diseasename = 'JAUNDICE'

AND a.noofdays > d.daystorecover

) pset

INTERSECT

SELECT patientid, patientname, dob

FROM (

SELECT patientid, patientname, MIN(dob) AS dob

FROM patients p

JOIN allocation a

ON a.patientid = p.patientid

JOIN diseases d

ON d.diseaseid = a.diseaseid

WHERE d.diseasename = 'JAUNDICE'

AND a.noofdays > d.daystorecover

GROUP BY patientid, patientname

HAVING dob = (

SELECT MIN(dob)

FROM patients q

JOIN allocation a4

ON a4.patientid = q.patientid

JOIN diseases d4

ON d4.diseaseid = a4.diseaseid

WHERE d4.diseasename = 'JAUNDICE'

AND a4.noofdays > d4.daystorecover

)

) oldest;

\*/

-- 38. Prepare a report in the following format. Patient Name | Doctor Name | Disease Name | Total Amount Paid

-- Sort the report with respect to the Total Amount paid in Desending Order.

-- The total amount paid should be 0 if patient has not paid any amount.

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS rpt\_patient\_doctor\_disease\_amount;

DELIMITER $$

CREATE PROCEDURE rpt\_patient\_doctor\_disease\_amount()

BEGIN

SELECT p.patientname, d.doctorname, di.diseasename, COALESCE(SUM(a.feesperday \* a.noofdays), 0) AS total\_amount\_paid

FROM patients p

LEFT JOIN allocation a

ON a.patientid = p.patientid

LEFT JOIN doctors d

ON d.doctorid = a.doctorid

LEFT JOIN diseases di

ON di.diseaseid = a.diseaseid

GROUP BY p.patientname, d.doctorname, di.diseasename

ORDER BY total\_amount\_paid DESC;

END $$

DELIMITER ;

CALL rpt\_patient\_doctor\_disease\_amount();

-- Method 2: Using LEFT JOIN

SELECT p.patientname, d.doctorname, di.diseasename, COALESCE(SUM(a.feesperday \* a.noofdays), 0) AS total\_amount\_paid

FROM patients p

LEFT JOIN allocation a

ON a.patientid = p.patientid

LEFT JOIN doctors d

ON d.doctorid = a.doctorid

LEFT JOIN diseases di

ON di.diseaseid = a.diseaseid

GROUP BY p.patientname, d.doctorname, di.diseasename

ORDER BY total\_amount\_paid DESC;

-- Method 3: Using WITH

WITH amt AS (

SELECT patientid, doctorid, diseaseid, SUM(feesperday \* noofdays) AS amt\_paid

FROM allocation

GROUP BY patientid, doctorid, diseaseid

)

SELECT p.patientname, d.doctorname, di.diseasename, COALESCE(a.amt\_paid, 0) AS total\_amount\_paid

FROM patients p

LEFT JOIN amt a

ON a.patientid = p.patientid

LEFT JOIN doctors d

ON d.doctorid = a.doctorid

LEFT JOIN diseases di

ON di.diseaseid = a.diseaseid

ORDER BY total\_amount\_paid DESC;

-- Method 4: Using NOT IN

SELECT p.patientname, d.doctorname, di.diseasename, COALESCE(SUM(a.feesperday \* a.noofdays), 0) AS total\_amount\_paid

FROM patients p

LEFT JOIN allocation a

ON a.patientid = p.patientid

AND (p.patientid, a.diseaseid, a.doctorid) NOT IN (

SELECT patientid, diseaseid, doctorid

FROM allocation

WHERE feesperday IS NULL /\* Dummy NOT IN screen \*/

)

LEFT JOIN doctors d

ON d.doctorid = a.doctorid

LEFT JOIN diseases di

ON di.diseaseid = a.diseaseid

GROUP BY p.patientname, d.doctorname, di.diseasename

ORDER BY total\_amount\_paid DESC;

-- Method 5: Using EXISTS / NOT EXISTS (Correlated Subquery)

SELECT p.patientname, d.doctorname, di.diseasename, COALESCE((

SELECT SUM(x.feesperday \* x.noofdays)

FROM allocation x

WHERE x.patientid = p.patientid

AND x.doctorid = d.doctorid

AND x.diseaseid = di.diseaseid

), 0) AS total\_amount\_paid

FROM patients p

LEFT JOIN doctors d

ON EXISTS (SELECT 1)

LEFT JOIN diseases di

ON EXISTS (SELECT 1)

ORDER BY total\_amount\_paid DESC;

-- Method 6 Using UNION (SET operators)

SELECT p.patientname, d.doctorname, di.diseasename, SUM(a.feesperday \* a.noofdays) AS total\_amount\_paid

FROM allocation a

JOIN patients p

ON p.patientid = a.patientid

JOIN doctors d

ON d.doctorid = a.doctorid

JOIN diseases di

ON di.diseaseid = a.diseaseid

GROUP BY p.patientname, d.doctorname, di.diseasename

UNION ALL

SELECT p.patientname, NULL AS doctorname, NULL AS diseasename, 0 AS total\_amount\_paid

FROM patients p

WHERE p.patientid NOT IN (

SELECT DISTINCT patientid

FROM allocation

)

ORDER BY total\_amount\_paid DESC;

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT patientname, doctorname, diseasename, total\_amount\_paid

FROM (

SELECT p.patientname, d.doctorname, di.diseasename, COALESCE(SUM(a.feesperday \* a.noofdays), 0) AS total\_amount\_paid

FROM patients p

LEFT JOIN allocation a

ON a.patientid = p.patientid

LEFT JOIN doctors d

ON d.doctorid = a.doctorid

LEFT JOIN diseases di

ON di.diseaseid = a.diseaseid

GROUP BY p.patientname, d.doctorname, di.diseasename

) r

EXCEPT

SELECT patientname, doctorname, diseasename, total\_amount\_paid

FROM (

SELECT p.patientname, d.doctorname, di.diseasename, COALESCE(SUM(a.feesperday \* a.noofdays), 0) AS total\_amount\_paid

FROM patients p

LEFT JOIN allocation a

ON a.patientid = p.patientid

LEFT JOIN doctors d

ON d.doctorid = a.doctorid

LEFT JOIN diseases di

ON di.diseaseid = a.diseaseid

GROUP BY p.patientname, d.doctorname, di.diseasename

HAVING total\_amount\_paid < (

SELECT MAX(COALESCE(SUM(feesperday \* noofdays), 0))

FROM allocation

)

) y;

-- Method 8: Using INTERSECT (SET operators)

SELECT patientname, doctorname, diseasename, total\_amount\_paid

FROM (

SELECT p.patientname, d.doctorname, di.diseasename, COALESCE(SUM(a.feesperday \* a.noofdays), 0) AS total\_amount\_paid

FROM patients p

LEFT JOIN allocation a

ON a.patientid = p.patientid

LEFT JOIN doctors d

ON d.doctorid = a.doctorid

LEFT JOIN diseases di

ON di.diseaseid = a.diseaseid

GROUP BY p.patientname, d.doctorname, di.diseasename

) w

INTERSECT

SELECT patientname, doctorname, diseasename, total\_amount\_paid

FROM (

SELECT p.patientname, d.doctorname, di.diseasename, COALESCE(SUM(a.feesperday \* a.noofdays), 0) AS total\_amount\_paid

FROM patients p

LEFT JOIN allocation a

ON a.patientid = p.patientid

LEFT JOIN doctors d

ON d.doctorid = a.doctorid

LEFT JOIN diseases di

ON di.diseaseid = a.diseaseid

GROUP BY p.patientname, d.doctorname, di.diseasename

) w2;

\*/

-- 39. Display TOP 10 Patient Details with respect to the Bill Amount paid so far

-- Method 1: Using RANK Function

WITH bill\_summary AS (

SELECT p.patientid, p.patientname, COALESCE(SUM(a.feesperday \* a.noofdays), 0) AS total\_bill,

RANK() OVER (ORDER BY COALESCE(SUM(a.feesperday \* a.noofdays), 0) DESC) AS rnk

FROM patients p

LEFT JOIN allocation a

ON p.patientid = a.patientid

GROUP BY p.patientid, p.patientname

)

SELECT patientid, patientname, total\_bill

FROM bill\_summary

WHERE rnk <= 10

ORDER BY total\_bill DESC;

-- Method 2: Using NOT IN

SELECT p.patientid, p.patientname, COALESCE(SUM(a.feesperday \* a.noofdays), 0) AS total\_bill

FROM patients p

LEFT JOIN allocation a

ON p.patientid = a.patientid

GROUP BY p.patientid, p.patientname

HAVING (

SELECT COUNT(DISTINCT COALESCE(SUM(a2.feesperday \* a2.noofdays), 0))

FROM patients p2

LEFT JOIN allocation a2

ON p2.patientid = a2.patientid

GROUP BY p2.patientid

HAVING COALESCE(SUM(a2.feesperday \* a2.noofdays), 0) > COALESCE(SUM(a.feesperday \* a.noofdays), 0)

) < 10

ORDER BY total\_bill DESC;

-- Method 3: Using JOIN

WITH bill\_summary AS (

SELECT p.patientid, p.patientname, COALESCE(SUM(a.feesperday \* a.noofdays), 0) AS total\_bill

FROM patients p

LEFT JOIN allocation a

ON p.patientid = a.patientid

GROUP BY p.patientid, p.patientname

)

SELECT b1.patientid, b1.patientname, b1.total\_bill

FROM bill\_summary b1

LEFT JOIN bill\_summary b2

ON b2.total\_bill > b1.total\_bill

GROUP BY b1.patientid, b1.patientname, b1.total\_bill

HAVING COUNT(b2.patientid) < 10

ORDER BY b1.total\_bill DESC;

/\* Does not work in MySQL

-- Method 4: Using EXCEPT (SET operators)

WITH BillSummary AS (

SELECT p.PatientID, p.PatientName, COALESCE(SUM(a.FeesPerDay \* a.NoOfDays), 0) AS TotalBill

FROM Patients p

LEFT JOIN Allocation a

ON p.PatientID = a.PatientID

GROUP BY p.PatientID, p.PatientName

),

Top10Bills AS (

SELECT TOP 10 PatientID, PatientName, TotalBill

FROM BillSummary

ORDER BY TotalBill DESC

)

SELECT PatientID, PatientName, TotalBill

FROM BillSummary

EXCEPT

SELECT PatientID, PatientName, TotalBill

FROM BillSummary

WHERE PatientID NOT IN (

SELECT PatientID

FROM Top10Bills

)

ORDER BY TotalBill DESC;

-- Method 5: Using INTERSECT (SET operators)

WITH BillSummary AS (

SELECT p.PatientID, p.PatientName, COALESCE(SUM(a.FeesPerDay \* a.NoOfDays), 0) AS TotalBill

FROM Patients p

LEFT JOIN Allocation a

ON p.PatientID = a.PatientID

GROUP BY p.PatientID, p.PatientName

),

Top10Bills AS (

SELECT TOP 10 PatientID, PatientName, TotalBill

FROM BillSummary

ORDER BY TotalBill DESC

),

HighBills AS (

SELECT PatientID, PatientName, TotalBill

FROM BillSummary

WHERE TotalBill > 5000 -- example threshold

)

SELECT PatientID, PatientName, TotalBill

FROM Top10Bills

INTERSECT

SELECT PatientID, PatientName, TotalBill

FROM HighBills

ORDER BY TotalBill DESC;

\*/

-- 40. Display the doctors who had joined before their dean

-- Method 1: Using JOIN

SELECT d.doctorid, d.doctorname, d.joiningdate, de.joiningdate AS deanjoiningdate

FROM doctors d

JOIN deans de

ON d.deanid = de.deanid

WHERE d.joiningdate < de.joiningdate;

-- Method 2: Using NOT IN

SELECT doctorid, doctorname, joiningdate

FROM doctors

WHERE doctorid NOT IN (

SELECT d.doctorid

FROM doctors d

JOIN deans de

ON d.deanid = de.deanid

WHERE d.joiningdate >= de.joiningdate

);

-- Method 3: Using Subquery

SELECT doctorid, doctorname, joiningdate

FROM doctors d

WHERE joiningdate < (

SELECT joiningdate

FROM deans de

WHERE de.deanid = d.deanid

);

-- Method 4: Using EXISTS (Correlated Subquery)

SELECT doctorid, doctorname, joiningdate

FROM doctors d

WHERE EXISTS (

SELECT 1

FROM deans de

WHERE de.deanid = d.deanid

AND d.joiningdate < de.joiningdate

);

-- Method 5: Using UNION (SET operators)

SELECT doctorid, doctorname, joiningdate

FROM doctors d

WHERE joiningdate < (

SELECT joiningdate

FROM deans de

WHERE de.deanid = d.deanid

)

UNION

SELECT doctorid, doctorname, joiningdate

FROM doctors d

WHERE joiningdate < (

SELECT MIN(joiningdate)

FROM deans

);

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT doctorid, doctorname, joiningdate

FROM doctors

EXCEPT

SELECT d.doctorid, d.doctorname, d.joiningdate

FROM doctors d

JOIN deans de

ON d.deanid = de.deanid

WHERE d.joiningdate >= de.joiningdate;

-- Method 7: Using INTERSECT (SET operators)

SELECT doctorid, doctorname, joiningdate

FROM doctors

INTERSECT

SELECT d.doctorid, d.doctorname, d.joiningdate

FROM doctors d

JOIN deans de

ON d.deanid = de.deanid

WHERE d.joiningdate < de.joiningdate;

\*/

-- 41. Prepare a report in this format

-- Doctor Id | Number of patients attended so far

-- D1 | 5

-- D2 | 3

-- D3 | 0

-- D4 | 7

-- D5 | 0

-- D6 | 23

-- Method 1: Using PROCEDURE

DROP PROCEDURE IF EXISTS report\_doctor\_patient\_count;

DELIMITER $$

CREATE PROCEDURE report\_doctor\_patient\_count()

BEGIN

SELECT d.doctorid,

COUNT(a.patientid) AS number\_of\_patients

FROM doctors d

LEFT JOIN allocation a ON d.doctorid = a.doctorid

GROUP BY d.doctorid

ORDER BY d.doctorid;

END $$

DELIMITER ;

CALL report\_doctor\_patient\_count();

-- Method 2: Using GROUP BY

SELECT d.doctorid, COUNT(a.patientid) AS numberofpatients

FROM doctors d

LEFT JOIN allocation a

ON d.doctorid = a.doctorid

GROUP BY d.doctorid

ORDER BY d.doctorid;

-- Method 3: Using JOIN

SELECT d.doctorid, COUNT(a.patientid) AS numberofpatients

FROM doctors d

LEFT JOIN allocation a

ON d.doctorid = a.doctorid

GROUP BY d.doctorid;

-- Method 4: Using NOT IN

SELECT d.doctorid, (

SELECT COUNT(\*)

FROM allocation a

WHERE a.doctorid = d.doctorid

) AS numberofpatients

FROM doctors d

WHERE d.doctorid NOT IN (

SELECT DISTINCT doctorid

FROM allocation

WHERE doctorid IS NULL

)

ORDER BY d.doctorid;

-- Method 5: Using EXISTS / NOT EXISTS (Correlated Subquery)

SELECT d.doctorid, (

SELECT COUNT(\*)

FROM allocation a

WHERE a.doctorid = d.doctorid

) AS numberofpatients

FROM doctors d

WHERE EXISTS (

SELECT 1

FROM allocation a

WHERE a.doctorid = d.doctorid

)

OR NOT EXISTS (

SELECT 1

FROM allocation a

WHERE a.doctorid = d.doctorid

)

ORDER BY d.doctorid;

-- Method 6 Using UNION (SET operators)

SELECT d.doctorid, COUNT(a.patientid) AS numberofpatients

FROM doctors d

JOIN allocation a

ON d.doctorid = a.doctorid

GROUP BY d.doctorid

UNION

SELECT d.doctorid, 0 AS numberofpatients

FROM doctors d

WHERE d.doctorid NOT IN (SELECT DISTINCT doctorid FROM allocation);

/\* Does not work in MySQL

-- Method 7: Using EXCEPT (SET operators)

SELECT d.doctorid, COUNT(a.patientid) AS numberofpatients

FROM doctors d

JOIN allocation a

ON d.doctorid = a.doctorid

GROUP BY d.doctorid

EXCEPT

SELECT d.doctorid, 0 AS numberofpatients

FROM doctors d

LEFT JOIN allocation a

ON d.doctorid = a.doctorid

GROUP BY d.doctorid

HAVING COUNT(a.patientid) > 0;

-- Method 8: Using INTERSECT (SET operators)

SELECT d.doctorid, COUNT(a.patientid) AS numberofpatients

FROM doctors d

JOIN allocation a

ON d.doctorid = a.doctorid

GROUP BY d.doctorid

INTERSECT

SELECT d.doctorid, COUNT(a.patientid) AS numberofpatients

FROM doctors d

JOIN allocation a

ON d.doctorid = a.doctorid

GROUP BY d.doctorid;

\*/

-- 42. How many patients are allocated to the doctor Ashok during the first quarter of the year 2012 (A Year has four quarters)

-- Method 1: Using JOIN

SELECT COUNT(DISTINCT a.patientid) AS patientsallocated

FROM allocation a

JOIN doctors d

ON a.doctorid = d.doctorid

WHERE d.doctorname = 'Ashok'

AND a.admitteddate >= '2012-01-01'

AND a.admitteddate < '2012-04-01';

-- Method 2: Using NOT IN

SELECT COUNT(DISTINCT patientid) AS patientsallocated

FROM allocation

WHERE doctorid = (

SELECT doctorid

FROM doctors

WHERE doctorname = 'Ashok'

)

AND admitteddate >= '2012-01-01'

AND admitteddate < '2012-04-01'

AND patientid NOT IN (

SELECT patientid

FROM allocation

WHERE admitteddate < '2012-01-01'

);

-- Method 3: Using Subquery

SELECT COUNT(DISTINCT patientid) AS patientsallocated

FROM allocation

WHERE doctorid = (

SELECT doctorid

FROM doctors

WHERE doctorname = 'Ashok'

)

AND admitteddate BETWEEN '2012-01-01' AND '2012-03-31';

-- Method 4: Using EXISTS / NOT EXISTS (Correlated Subquery)

SELECT COUNT(DISTINCT a.patientid) AS patientsallocated

FROM allocation a

WHERE EXISTS (

SELECT 1

FROM doctors d

WHERE d.doctorid = a.doctorid

AND d.doctorname = 'Ashok'

)

AND a.admitteddate BETWEEN '2012-01-01' AND '2012-03-31';

-- Method 5: Using UNION (SET operators)

SELECT patientid FROM allocation

WHERE doctorid = (SELECT doctorid FROM doctors WHERE doctorname = 'Ashok')

AND admitteddate BETWEEN '2012-01-01' AND '2012-03-31'

UNION

SELECT patientid FROM allocation

WHERE doctorid = (SELECT doctorid FROM doctors WHERE doctorname = 'Ashok')

AND admitteddate BETWEEN '2012-01-01' AND '2012-03-31';

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT patientid

FROM allocation

WHERE doctorid = (

SELECT doctorid

FROM doctors

WHERE doctorname = 'Ashok'

)

AND admitteddate BETWEEN '2012-01-01' AND '2012-03-31'

EXCEPT

SELECT patientid

FROM allocation

WHERE doctorid = (

SELECT doctorid

FROM doctors

WHERE doctorname = 'Ashok'

)

AND admitteddate NOT BETWEEN '2012-01-01' AND '2012-03-31';

-- Method 7: Using INTERSECT (SET operators)

SELECT patientid

FROM allocation

WHERE doctorid = (

SELECT doctorid

FROM doctors

WHERE doctorname = 'Ashok'

)

AND admitteddate BETWEEN '2012-01-01' AND '2012-06-30'

INTERSECT

SELECT patientid

FROM allocation

WHERE doctorid = (

SELECT doctorid

FROM doctors

WHERE doctorname = 'Ashok'

)

AND admitteddate BETWEEN '2012-01-01' AND '2012-03-31';

-- Count version:

SELECT COUNT(\*) AS patientsallocated

FROM (

SELECT patientid

FROM allocation

WHERE doctorid = (

SELECT doctorid

FROM doctors

WHERE doctorname = 'Ashok'

)

AND admitteddate BETWEEN '2012-01-01' AND '2012-06-30'

INTERSECT

SELECT patientid

FROM allocation

WHERE doctorid = (

SELECT doctorid

FROM doctors

WHERE doctorname = 'Ashok'

)

AND admitteddate BETWEEN '2012-01-01' AND '2012-03-31'

) AS sub;

\*/

-- 43. How many patients are allocated to the doctor Ashok during the third quarter of the year 2011. (A Year has four quarters)

-- Method 1: Using JOIN

SELECT COUNT(DISTINCT a.PatientID) AS PatientsAllocated

FROM Allocation a

JOIN Doctors d

ON a.DoctorID = d.DoctorID

WHERE d.DoctorName = 'Ashok'

AND a.AdmittedDate >= '2011-07-01'

AND a.AdmittedDate < '2011-10-01';

-- Method 2: Using NOT IN

SELECT COUNT(DISTINCT patientid) AS patientsallocated

FROM allocation

WHERE doctorid = (

SELECT doctorid

FROM doctors

WHERE doctorname = 'Ashok'

)

AND admitteddate >= '2011-07-01'

AND admitteddate < '2011-10-01'

AND patientid NOT IN (

SELECT patientid

FROM allocation

WHERE admitteddate < '2011-07-01'

);

-- Method 3: Using Subquery

SELECT COUNT(DISTINCT patientid) AS patientsallocated

FROM allocation

WHERE doctorid = (

SELECT doctorid

FROM doctors

WHERE doctorname = 'Ashok'

)

AND admitteddate BETWEEN '2011-07-01' AND '2011-09-30';

-- Method 4: Using EXISTS (Correlated Subquery)

SELECT COUNT(DISTINCT a.patientid) AS patientsallocated

FROM allocation a

WHERE EXISTS (

SELECT 1

FROM doctors d

WHERE d.doctorid = a.doctorid

AND d.doctorname = 'Ashok'

)

AND a.admitteddate BETWEEN '2011-07-01' AND '2011-09-30';

-- Method 5: Using UNION (SET operators)

SELECT patientid FROM allocation

WHERE doctorid = (SELECT doctorid FROM doctors WHERE doctorname = 'Ashok')

AND admitteddate BETWEEN '2011-07-01' AND '2011-09-30'

UNION

SELECT patientid FROM allocation

WHERE doctorid = (SELECT doctorid FROM doctors WHERE doctorname = 'Ashok')

AND admitteddate BETWEEN '2011-07-01' AND '2011-09-30';

/\* Does not work in MySQL

-- Method 6: Using EXCEPT (SET operators)

SELECT patientid

FROM allocation

WHERE doctorid = (SELECT doctorid FROM doctors WHERE doctorname = 'Ashok')

AND admitteddate BETWEEN '2011-07-01' AND '2011-09-30'

EXCEPT

SELECT patientid

FROM allocation

WHERE doctorid = (SELECT doctorid FROM doctors WHERE doctorname = 'Ashok')

AND admitteddate NOT BETWEEN '2011-07-01' AND '2011-09-30';

-- Count version:

SELECT COUNT(\*) AS patientsallocated

FROM (

SELECT patientid

FROM allocation

WHERE doctorid = (SELECT doctorid FROM doctors WHERE doctorname = 'Ashok')

AND admitteddate BETWEEN '2011-07-01' AND '2011-09-30'

EXCEPT

SELECT patientid

FROM allocation

WHERE doctorid = (SELECT doctorid FROM doctors WHERE doctorname = 'Ashok')

AND admitteddate NOT BETWEEN '2011-07-01' AND '2011-09-30'

) AS sub;

-- Method 7: Using INTERSECT (SET operators)

SELECT patientid

FROM allocation

WHERE doctorid = (SELECT doctorid FROM doctors WHERE doctorname = 'Ashok')

AND admitteddate BETWEEN '2011-04-01' AND '2011-09-30'

INTERSECT

SELECT patientid

FROM allocation

WHERE doctorid = (SELECT doctorid FROM doctors WHERE doctorname = 'Ashok')

AND admitteddate BETWEEN '2011-07-01' AND '2011-09-30';

-- Count version:

SELECT COUNT(\*) AS patientsallocated

FROM (

SELECT patientid

FROM allocation

WHERE doctorid = (SELECT doctorid FROM doctors WHERE doctorname = 'Ashok')

AND admitteddate BETWEEN '2011-04-01' AND '2011-09-30'

INTERSECT

SELECT patientid

FROM allocation

WHERE doctorid = (SELECT doctorid FROM doctors WHERE doctorname = 'Ashok')

AND admitteddate BETWEEN '2011-07-01' AND '2011-09-30'

) AS sub;

\*/