DBMS_ALL PRACS

PRACTICAL 1
(Design a database for hospital with a set of patients and a set of medical doctors. Associate with each patient a log of various tests and examination conducted.)

DATABASE CREATION

```
use Olive;
create TABLE DOCTOR
(
SerialNumber int IDENTITY (1,1) PRIMARY KEY,
DOCTORId int NOT NULL UNIQUE,
LastNAME varchar(50) NOT NULL,
FirstNAME varchar(50) NOT NULL,
PHONE varchar(50) NULL,
Specialization varchar(50) NOT NULL,
visits_only_on varchar(9) NULL,
);
select* from DOCTOR;
drop table DOCTOR;

create TABLE PATIENT
(
SerialNumber int IDENTITY(1,1) PRIMARY KEY,
PatientID varchar(50) NOT NULL UNIQUE,
FirstNAME varchar(50) NOT NULL,
LastNAME varchar(50) NOT NULL,
Area varchar(50) NOT NULL,
Test_performed varchar(50) NOT NULL,
DATE_of_examination DATE
);
select* from PATIENT;
drop table PATIENT;

Insert into DOCTOR
(DOCTORId,LastNAME,FirstNAME,PHONE,Specialization,visits_only_on)
values
('123','Mendez','Shawn','1234567890','Pediatrician','Monday'),
```

('456','Derulo', 'Jason', '0987654321', 'Gynaecologist','Tuesday'),
('234','Ora','Rita', '7812365490','Cardiologist',''),
('867','B','Cardi','3456789213','Onthropologist',''),
('654','Gomez','Selena','7654321890','Pediatrician','Wednesday');

Insert into PATIENT
(PATIENTId,FirstNAME,LastNAME,Area,Test_performed,DATE_of_examination)
values
('P_123','Brad','Pitt','Turbhe','Blood test','2024-03-04'),
('P_567','Tom','Cruise','Vashi', 'BP measurement','2010-05-19'),
('P_345','Sofia','Vergara','Bhandup','Biopsy','2022-07-01'),
('P_789','Angelina','Jolie','Vashi','Blood test','2010-05-08'),
('P_687','Will','Smith','Mulund','Endoscopy','1999-12-31');

	SerialNumber	DOCTORId	LastNAME	FirstNAME	PHONE	Specialization	visits_only_on
1	1	123	Mendez	Shawn	1234567890	Pediatrician	Monday
2	2	456	Derulo	Jason	0987654321	Gynaecologist	Tuesday
3	3	234	Ora	Rita	7812365490	Cardiologist	
4	4	867	B	Cardi	3456789213	Onthropologist	
5	5	654	Gomez	Selena	7654321890	Pediatrician	Wednesday

	SerialNumber	PatientID	FirstNAME	LastNAME	Area	Test_performed	DATE_of_examination
1	6	P_123	Brad	Pitt	Turbhe	Blood test	2024-03-04
2	7	P_567	Tom	Cruise	Vashi	BP measurement	2010-05-19
3	8	P_345	Sofia	Vergara	Bhandup	Biopsy	2022-07-01
4	9	P_789	Angelina	Jolie	Vashi	Blood test	2010-05-08
5	10	P_687	Will	Smith	Mulund	Endoscopy	1999-12-31

i. find the set of patients who live in “Vashi” and were examined on 8.05.10
--> select* from PATIENT where Area='Vashi' and DATE_of_examination= '2010-05-08'

	SerialNumber	PatientID	FirstNAME	LastNAME	Area	Test_performed	DATE_of_examination
1	9	P_789	Angelina	Jolie	Vashi	Blood test	2010-05-08

ii. List the various tests and examination conducted on each patient
--> select Test_performed,PATIENTId,FirstNAME from PATIENT

Results Messages

	Test_performed	PATIENTId	FirstNAME
1	Blood test	P_123	Brad
2	BP measurement	P_567	Tom
3	Biopsy	P_345	Sofia
4	Blood test	P_789	Angelina
5	Endoscopy	P_687	Will

iii. Find the name of the doctors who visit only on Tuesday

--> select* from DOCTOR where visits_only_on='Tuesday'

	SerialNumber	DOCTORId	LastNAME	FirstNAME	PHONE	Specialization	visits_only_on
1	2	456	Derulo	Jason	0987654321	Gynaecologist	Tuesday

iv. Write any one trigger After Insert

--> create Trigger trig1

on PATIENT After insert

AS

BEGIN

PRINT 'A new value inserted in table'

END;

```
Insert into PATIENT (PATIENTId,FirstNAME,LastNAME,Area,Test_performed,DATE_of_examination)
values
('P_564','Kate','Winslet','Nahur','Biopsy','2002-10-24');
```

Messages

```
A new value inserted in table

(1 row affected)
```

v. Write any one procedure or function to count no of patient from vashi

--> Select Area, COUNT(*) as TotalPatients

From PATIENT

WHERE Area='Vashi'

Group by Area

Results | Messages

	Area	TotalPatients
1	Vashi	2

PRACTICAL 2

(Design a database of a university registrar's office which maintains data about the course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom.)

DATABASE CREATION

use prac1;

create table COURSE

(

CourseID INT PRIMARY KEY,

YEAR int,

```
Semester VARCHAR(50),
SectionNumber int,
InstructorID int,
Timings VARCHAR(100),
ClassID int,
CourseName varchar(50) NOT NULL,
FOREIGN KEY (ClassID) REFERENCES CLASSROOM(ClassroomID),
FOREIGN KEY(InstructorID) REFERENCES Instructor(InstructorID)
);
select* from COURSE;
drop TABLE COURSE;

CREATE TABLE INSTRUCTOR
(
InstructorID int PRIMARY KEY,
InstructorName varchar(50)
);
select* from INSTRUCTOR;
drop TABLE INSTRUCTOR;

CREATE TABLE CLASSROOM
(
ClassroomID INT PRIMARY KEY,
RoomNumber varchar(50),
Building varchar(50)
);
select* from CLASSROOM;
drop TABLE CLASSROOM;


INSERT INTO Instructor (InstructorID, InstructorName) VALUES
(1, 'Jain'),
(2, 'Smith'),
(3, 'Doe');

INSERT INTO Classroom (ClassroomID, RoomNumber, Building) VALUES
(1, '101', 'Main Building'),
(2, '201', 'Science Building');

INSERT INTO COURSE (CourseID, Year, Semester, SectionNumber, InstructorID,
Timings,ClassID,CourseName) VALUES
```

(101, 2009, '3', 1, 1, '9:00 AM - 10:30 AM',1 ,'dbms'),
(102, 2010, '3', 2, 1, '10:00 AM - 11:30 AM',1,'java'),
(103, 2009, '4', 1, 2, '1:00 PM - 2:30 PM',2,'ds'),
(104, 2010, '4', 2, 3, '3:00 PM - 4:30 PM',2,'sbl');

Results | Messages

	CourseID	YEAR	Semester	SectionNumber	InstructorID	Timings	ClassID	CourseName
1	101	2009	3	1	1	9:00 AM - 10:30 AM	1	dbms
2	102	2010	3	2	1	10:00 AM - 11:30 AM	1	java
3	103	2009	4	1	2	1:00 PM - 2:30 PM	2	ds
4	104	2010	4	2	3	3:00 PM - 4:30 PM	2	sbl

	InstructorID	InstructorName
1	1	Jain
2	2	Smith
3	3	Doe

	ClassroomID	RoomNumber	Building
1	1	101	Main Building
2	2	201	Science Building

i. Retrieve the name of all courses taught by professor Jain in 2009 and 2010.

```
select CourseName from COURSE c
where InstructorID in
(select InstructorID from INSTRUCTOR i where InstructorName='Jain' and c.year
in(2009,2010))
```

	CourseName
1	dbms
2	java

ii. List all the course numbers taught by each instructor.

```
select c.CourseName, c.InstructorID from COURSE C
inner join INSTRUCTOR i
on i.InstructorID=c.InstructorID
```

Results | Messages

	CourseName	InstructorID
1	dbms	1
2	java	1
3	ds	2
4	sbl	3

iii. Update a course of your choice and delete the course which you don't like.

```
UPDATE COURSE
Set Timings= '8:00 AM - 9:30 AM'
WHERE CourseNumber = 101;
```

	CourseID	YEAR	Semester	SectionNumber	InstructorID	Timings	ClassID	CourseName
1	101	2009	3	1	1	8:00 AM - 9:30 AM	1	dbms
2	102	2010	3	2	1	10:00 AM - 11:30 AM	1	java
3	103	2009	4	1	2	1:00 PM - 2:30 PM	2	ds
4	104	2010	4	2	3	3:00 PM - 4:30 PM	2	sbl

```
DELETE FROM Course
WHERE CourseNumber = 103;
```

Results | Messages

	CourseID	YEAR	Semester	SectionNumber	InstructorID	Timings	ClassID	CourseName
1	101	2009	3	1	1	8:00 AM - 9:30 AM	1	dbms
2	102	2010	3	2	1	10:00 AM - 11:30 AM	1	java
3	104	2010	4	2	3	3:00 PM - 4:30 PM	2	sbl

iv. Write any one trigger

```
CREATE TRIGGER trig2
ON Course AFTER INSERT
AS
BEGIN
PRINT'A NEW COURSE HAS BEEN ENTERED'
END;
```

v. Write any one procedure or function

```
CREATE PROCEDURE GetCourseDetails(@CourseNumber INT)
AS
BEGIN
SELECT *
FROM Course
WHERE CourseNumber = @CourseNumber;
END;
EXEC GetCourseDetails @CourseNumber = 101;
```

PRACTICAL 11

(A database is being constructed to keep track of the teams and games of a sports league. A team has several players, not all of whom participated in each game. It is desired to keep track of the players participating in each game for each team, the positions they played in that game, and the result of the game. Design a relational database for this application, stating any assumptions you make. Choose your favorite sport (e.g., soccer, football, baseball).

a. List all the games played in the sports league

b. Create a view showing detail of each player

c. Find the total number of teams participating in the sports league.

d. Write any one trigger

e. Write any one procedure or function)

DATABASE

```
create database exp11;
use exp11;

CREATE TABLE Team (
   TeamID INT PRIMARY KEY,
   TeamName VARCHAR(100)
);

CREATE TABLE Player (
   PlayerID INT PRIMARY KEY,
   TeamID INT,
   PlayerName VARCHAR(100)
);

CREATE TABLE Game (
   GameID INT PRIMARY KEY,
   DatePlayed DATE,
   Location VARCHAR(100),
   Result VARCHAR(10)
);
```

```sql
CREATE TABLE GamePlayer (
   GameID INT,
   PlayerID INT,
   Position VARCHAR(50),
   PRIMARY KEY (GameID, PlayerID),
   FOREIGN KEY (GameID) REFERENCES Game(GameID),
   FOREIGN KEY (PlayerID) REFERENCES Player(PlayerID)
);

-- Insert sample teams
INSERT INTO Team (TeamID, TeamName) VALUES
(1, 'Team A'),
(2, 'Team B'),
(3, 'Team C');

-- Insert sample players
INSERT INTO Player (PlayerID, TeamID, PlayerName) VALUES
(101, 1, 'Player 1'),
(102, 1, 'Player 2'),
(103, 2, 'Player 3'),
(104, 2, 'Player 4'),
(105, 3, 'Player 5');

-- Insert sample games
INSERT INTO Game (GameID, DatePlayed, Location, Result) VALUES
(201, '2024-04-01', 'Stadium A', 'Win'),
(202, '2024-04-05', 'Stadium B', 'Draw'),
(203, '2024-04-10', 'Stadium C', 'Loss');

-- Insert sample game-player mappings
INSERT INTO GamePlayer (GameID, PlayerID, Position) VALUES
(201, 101, 'Forward'),
(201, 102, 'Midfielder'),
(202, 103, 'Defender'),
(202, 104, 'Goalkeeper'),
(203, 105, 'Midfielder');

select* from Team
select* from Player
select* from Game
select* from GamePlayer;
```

Results | Messages

1	1	Team A
2	2	Team B
3	3	Team C

	PlayerID	TeamID	PlayerName
1	101	1	Player 1
2	102	1	Player 2
3	103	2	Player 3
4	104	2	Player 4
5	105	3	Player 5

	GameID	DatePlayed	Location	Result
1	201	2024-04-01	Stadium A	Win
2	202	2024-04-05	Stadium B	Draw
3	203	2024-04-10	Stadium C	Loss

	GameID	PlayerID	Position
1	201	101	Forward
2	201	102	Midfielder
3	202	103	Defender
4	202	104	Goalkee...
5	203	105	Midfielder

```
CREATE VIEW AllGames AS
SELECT g.GameID, g.DatePlayed, g.Location, t.TeamName, g.Result
FROM Game g
JOIN GamePlayer gp ON g.GameID = gp.GameID
JOIN Player p ON gp.PlayerID = p.PlayerID
JOIN Team t ON p.TeamID = t.TeamID;
```

select* from AllGames;

Results | Messages

	GameID	DatePlayed	Location	TeamName	Result
1	201	2024-04-01	Stadium A	Team A	Win
2	201	2024-04-01	Stadium A	Team A	Win
3	202	2024-04-05	Stadium B	Team B	Draw
4	202	2024-04-05	Stadium B	Team B	Draw
5	203	2024-04-10	Stadium C	Team C	Loss

```
CREATE VIEW PlayerDetail AS
SELECT p.PlayerID, p.PlayerName, t.TeamName, gp.Position
FROM Player p
```

```
JOIN Team t ON p.TeamID = t.TeamID
JOIN GamePlayer gp ON p.PlayerID = gp.PlayerID;

select* from PlayerDetail;
```

Results | Messages

	PlayerID	PlayerName	TeamName	Position
1	101	Player 1	Team A	Forward
2	102	Player 2	Team A	Midfielder
3	103	Player 3	Team B	Defender
4	104	Player 4	Team B	Goalkeeper
5	105	Player 5	Team C	Midfielder

```
CREATE PROCEDURE GetTotalTeams
AS
BEGIN
  SELECT COUNT(DISTINCT TeamID) AS TotalTeams FROM Team;
END;
EXEC GetTotalTeams;
```

	TotalTeams
1	3

```
CREATE TRIGGER UpdateGameResult
ON Game
AFTER INSERT, UPDATE
AS
BEGIN
  UPDATE Game
  SET Result = 'Draw'
  WHERE EXISTS (
    SELECT * FROM inserted WHERE Result IS NULL
  );
END;

CREATE PROCEDURE GetGamePlayers
  @GameID INT
AS
BEGIN
  SELECT p.PlayerName, gp.Position
  FROM GamePlayer gp
  JOIN Player p ON gp.PlayerID = p.PlayerID
  WHERE gp.GameID = @GameID;
```

END;

EXEC GetGamePlayers @GameID=201;

	PlayerName	Position
1	Player 1	Forward
2	Player 2	Midfielder

EXPERIMENT 12
(Design a database to keep track of information for an art museum. The museum has a collection of ART_ OBJECTS. Each ART_ OBJECTS has a unique IdNo, an Artist (if known), a Year (when it was created, if known), a Title, and a Description.
a. Create a view which gives description on artist their ART_ OBJECTS.
b. Give description of all the ART_OBJECTS which were created in the year 2001
c. List all the female artists whose name starts with “Re”
d. Write any one trigger
e. Write any one procedure or function)

```
CREATE TABLE ArtObject (
   IdNo INT PRIMARY KEY,
   Artist VARCHAR(100),
   YearCreated INT,
   Title VARCHAR(100),
   Description VARCHAR(255)
);
-- Insert sample art objects
INSERT INTO ArtObject (IdNo, Artist, YearCreated, Title, Description) VALUES
(1, 'Leonardo da Vinci', 1503, 'Mona Lisa', 'Portrait painting of Lisa Gherardini'),
(2, 'Vincent van Gogh', 1889, 'The Starry Night', 'Oil painting of the night sky'),
(3, 'Pablo Picasso', 1937, 'Guernica', 'Depiction of the bombing of Guernica during the Spanish Civil War'),
(4, 'Rembrandt', 1642, 'The Night Watch', 'Group portrait of a militia company'),
(5, 'Rene Magritte', 1928, 'The Treachery of Images', 'Surrealist painting of a pipe with the text "This is not a pipe"'),
(6, 'Remedios Varo', 1955, 'Creation of the Birds', 'Surrealist painting depicting the creation of birds');

select* from ArtObject;
```

	IdNo	Artist	YearCreated	Title	Description
1	1	Leonardo da Vinci	1503	Mona Lisa	Portrait painting of Lisa Gherardini
2	2	Vincent van Gogh	1889	The Starry Night	Oil painting of the night sky
3	3	Pablo Picasso	1937	Guernica	Depiction of the bombing of Guernica during the ...
4	4	Rembrandt	1642	The Night Watch	Group portrait of a militia company
5	5	Rene Magritte	1928	The Treachery of Images	Surrealist painting of a pipe with the text "This is n...
6	6	Remedios Varo	1955	Creation of the Birds	Surrealist painting depicting the creation of birds

```
CREATE VIEW ArtistArtObjects AS
SELECT Artist, Title, Description
FROM ArtObject;
SELECT* from ArtistArtObjects;
```

	Artist	Title	Description
1	Leonardo da Vinci	Mona Lisa	Portrait painting of Lisa Gherardini
2	Vincent van Gogh	The Starry Night	Oil painting of the night sky
3	Pablo Picasso	Guernica	Depiction of the bombing of Guernica during the ...
4	Rembrandt	The Night Watch	Group portrait of a militia company
5	Rene Magritte	The Treachery of Images	Surrealist painting of a pipe with the text "This is n...
6	Remedios Varo	Creation of the Birds	Surrealist painting depicting the creation of birds

```
SELECT Title, Description
FROM ArtObject
WHERE YearCreated = 1937
```

	Title	Description
1	Guernica	Depiction of the bombing of Guernica during the Spanish Civil War

```
SELECT Artist
FROM ArtObject
WHERE Artist LIKE 'Re%'
```

Results | Message

	Artist
1	Rembrandt
2	Rene Magritte
3	Remedios Varo

```
CREATE PROCEDURE GetArtObjectById
  @IdNo INT
AS
BEGIN
  SELECT Artist, YearCreated, Title, Description
  FROM ArtObject
  WHERE IdNo = @IdNo;
END;
```

EXEC GetArtObjectById @IdNo=4

Results | Messages

	Artist	YearCreated	Title	Description
1	Rembrandt	1642	The Night Watch	Group portrait of a militia company

PRACTICAL 13

(Design a database for a small private airport database that is used to keep track of airplanes, their owners, airport employees, and pilots
a. List all the pilots of the plane “Kingfisher
b. Retrieve the list of all employees who were appointed between 01.01.2017 and 01.01.2019.
c. Update the database of your choice and delete which you don’t like.
d. Write any one trigger
e. Write any one procedure or function)

```
create database exp13;
use exp13;

CREATE TABLE Airplane (
  RegistrationNo VARCHAR(20) PRIMARY KEY,
  Model VARCHAR(100),
  OwnerID INT,
  FOREIGN KEY (OwnerID) REFERENCES Owner(OwnerID)
);

CREATE TABLE Owner (
  OwnerID INT PRIMARY KEY,
  Name VARCHAR(100),
  ContactNumber VARCHAR(20)
);

CREATE TABLE Employee (
  EmployeeID INT PRIMARY KEY,
  Name VARCHAR(100),
  HireDate DATE
);
```

```
CREATE TABLE Pilot (
   PilotID INT PRIMARY KEY,
   Name VARCHAR(100),
   LicenseNumber VARCHAR(20)
);

select* from Airplane
select* from Owner
select* from Pilot
select* from Employee;
```

Results | Messages

	RegistrationNo	Model	OwnerID

	OwnerID	Name	ContactNumber
1	101	John Doe	123-456-7890
2	102	Jane Smith	987-654-3210
3	103	Alice Johnson	555-555-5555

	PilotID	Name	LicenseNumber
1	301	Robert Johnson	PLT12345
2	302	Jessica Lee	PLT67890
3	303	David Miller	PLT54321

	EmployeeID	Name	HireDate
1	201	Michael Brown	2016-05-15
2	202	Emily Davis	2018-02-20
3	203	William Wilson	2019-11-10

```
SELECT p.Name AS PilotName, p.LicenseNumber
FROM Pilot p
JOIN Airplane a ON p.PilotID = a.PilotID
WHERE a.Model = 'Kingfisher';
(errorrr)

SELECT *
FROM Employee
WHERE HireDate BETWEEN '2017-01-01' AND '2019-01-01';
```

anagement

	yeeID	Name	HireDate
1	202	Emily Davis	2018-02-20

```
CREATE TRIGGER PreventAirplaneDeletion
ON Airplane
INSTEAD OF DELETE
AS
BEGIN
  IF EXISTS (SELECT * FROM deleted d JOIN Pilot p ON d.PilotID = p.PilotID)
  BEGIN
    RAISERROR ('Cannot delete airplane with associated pilot', 16, 1);
  END
  ELSE
  BEGIN
    DELETE FROM Airplane WHERE RegistrationNo IN (SELECT RegistrationNo FROM
deleted);
  END
END;
```

```
EXEC GetEmployeeById @EmployeeID= 203
```

	Name	HireDate
1	William Wilson	2019-11-10

EXPERIMENT 14

(Consider the following relations for a database that keeps track of business trips of salespersons in a sales office: SALESPERSON(Sid, Name, Start_Year, Dept_No) TRIP(Sid, From_City, To_City, Departure_Date, Return_Date, Trip_ID) Expense(Trip_ID, Account, Amount)

a. Give the details (all attributes of Trip relation) for trips that exceeded Rs. 2000 in expenses

b. Print the ENO of salesman who took trip to ‘Honolulu’

c. Print the total trip expenses incurred by the salesman with ENO = ‘234-56- 7890’

d. Write any one trigger

e. Write any one procedure or function)

```
CREATE TABLE Expense (
  Trip_ID INT,
  Account VARCHAR(100),
  Amount DECIMAL(10, 2),
  FOREIGN KEY (Trip_ID) REFERENCES TRIP(Trip_ID)
);
```

```
Select* from Expense
Select* from TRIP
Select* from SALESPERSON;

-- Insert sample salespersons data
INSERT INTO SALESPERSON (Sid, Name, Start_Year, Dept_No) VALUES
(101, 'John Doe', 2015, 1),
(102, 'Jane Smith', 2017, 2),
(103, 'Alice Johnson', 2018, 1);

-- Insert sample trips data
INSERT INTO TRIP (Trip_ID, Sid, From_City, To_City, Departure_Date, Return_Date)
VALUES
(1, 101, 'New York', 'Los Angeles', '2024-04-01', '2024-04-05'),
(2, 102, 'Chicago', 'Honolulu', '2024-05-10', '2024-05-15'),
(3, 103, 'Houston', 'Miami', '2024-06-20', '2024-06-25');

-- Insert sample expenses data
INSERT INTO Expense (Trip_ID, Account, Amount) VALUES
(1, 'Travel', 1500),
(1, 'Accommodation', 800),
(2, 'Travel', 2500),
(2, 'Meals', 500),
(3, 'Travel', 1800),
(3, 'Meals', 300);
```

	Trip_ID	Account	Amount
1	1	Travel	1500.00
2	1	Accommodation	800.00
3	2	Travel	2500.00
4	2	Meals	500.00
5	3	Travel	1800.00
6	3	Meals	300.00

	Trip_ID	Sid	From_City	To_City	Departure_Date	Return_Date
1	1	101	New York	Los Angeles	2024-04-01	2024-04-05
2	2	102	Chicago	Honolulu	2024-05-10	2024-05-15
3	3	103	Houston	Miami	2024-06-20	2024-06-25

	Sid	Name	Start_Year	Dept_No
1	101	John Doe	2015	1
2	102	Jane Smith	2017	2
3	103	Alice Johnson	2018	1

```
SELECT *
FROM TRIP t
WHERE EXISTS (
   SELECT 1
   FROM Expense e
   WHERE e.Trip_ID = t.Trip_ID
   GROUP BY e.Trip_ID
   HAVING SUM(e.Amount) > 2000
);
```

	Trip_ID	Sid	From_City	To_City	Departure_Date	Return_Date
1	1	101	New York	Los Angeles	2024-04-01	2024-04-05
2	2	102	Chicago	Honolulu	2024-05-10	2024-05-15
3	3	103	Houston	Miami	2024-06-20	2024-06-25

```
SELECT Sid
FROM TRIP
WHERE To_City = 'Honolulu';
```

Results | Messages

	Sid
1	102

```
SELECT SUM(e.Amount) AS TotalExpenses
FROM Expense e
JOIN TRIP t ON e.Trip_ID = t.Trip_ID
JOIN SALESPERSON s ON t.Sid = s.Sid
WHERE s.Sid = 101;
```

	TotalExpenses
1	2300.00

```
CREATE TRIGGER PreventExcessiveExpense
ON Expense
INSTEAD OF INSERT
AS
BEGIN
  IF EXISTS (SELECT * FROM inserted WHERE Amount > 5000)
  BEGIN
    RAISERROR ('Expense amount exceeds limit', 16, 1);
  END
  ELSE
  BEGIN
    INSERT INTO Expense (Trip_ID, Account, Amount)
    SELECT Trip_ID, Account, Amount FROM inserted;
  END
END;

CREATE PROCEDURE GetTripDetailsById
  @Trip_ID INT
AS
BEGIN
  SELECT *
  FROM TRIP
  WHERE Trip_ID = @Trip_ID;
END;
EXEC GetTripDetailsById @Trip_ID= 2
```

EXPERIMENT 15 (Consider the following relations for the database that keeps track of student enrollment in courses and the books opted for each course: Student(StuNo, Name, Major, Bdate) Course(Course-id, Cname, Dept) Enroll(StudNo, Course-id,

Quarter, Grade) Book_Adoption(Course-id,Quarter,Book_ISBN) Text(Book_ISBN, Book_Title, Publisher, Author)

a. List number of courses taken by aa students named 'Ravi Shankar' in Winter 2018

b. Produce a list of books for courses offered by the 'CSE' department that have used more than two books,

. List any department that has all its adopted books published by 'TMH Publishing'.

d. Write any one trigger

e. Write any one procedure or function)

```
create database exp15;
use exp15;

CREATE TABLE Student (
   StuNo INT PRIMARY KEY,
   Name VARCHAR(100),
   Major VARCHAR(100),
   Bdate DATE
);
CREATE TABLE Course (
   Course_id INT PRIMARY KEY,
   Cname VARCHAR(100),
   Dept VARCHAR(100)
);
CREATE TABLE Enroll (
   StudNo INT,
   Course_id INT,
   Quarter VARCHAR(100),
   Grade VARCHAR(2),
   PRIMARY KEY (StudNo, Course_id),
   FOREIGN KEY (StudNo) REFERENCES Student(StuNo),
   FOREIGN KEY (Course_id) REFERENCES Course(Course_id)
);
CREATE TABLE Book_Adoption (
   Course_id INT,
   Quarter VARCHAR(100),
   Book_ISBN VARCHAR(20),
   PRIMARY KEY (Course_id, Quarter, Book_ISBN),
   FOREIGN KEY (Course_id) REFERENCES Course(Course_id),
   FOREIGN KEY (Book_ISBN) REFERENCES Text(Book_ISBN)
);
```

```sql
CREATE TABLE Text (
   Book_ISBN VARCHAR(20) PRIMARY KEY,
   Book_Title VARCHAR(100),
   Publisher VARCHAR(100),
   Author VARCHAR(100)
);
select* from Student
select* from Course
select* from Enroll
select* from Book_Adoption
select* from Text;

-- Insert sample student data
INSERT INTO Student (StuNo, Name, Major, Bdate) VALUES
(1, 'Ravi Shankar', 'Computer Science', '2000-01-01'),
(2, 'John Doe', 'Engineering', '1999-05-15'),
(3, 'Alice Johnson', 'Mathematics', '2001-07-20');

-- Insert sample course data
INSERT INTO Course (Course_id, Cname, Dept) VALUES
(101, 'Introduction to Programming', 'CSE'),
(102, 'Database Management Systems', 'CSE'),
(103, 'Calculus', 'Math'),
(104, 'Introduction to Engineering', 'Engineering');

-- Insert sample enrollment data
INSERT INTO Enroll (StudNo, Course_id, Quarter, Grade) VALUES
(1, 101, 'Winter 2018', 'A'),
(1, 102, 'Winter 2018', 'B'),
(1, 103, 'Winter 2018', 'A'),
(2, 101, 'Winter 2018', 'B'),
(2, 102, 'Winter 2018', 'A'),
(3, 101, 'Winter 2018', 'A');

-- Insert sample book adoption data
INSERT INTO Book_Adoption (Course_id, Quarter, Book_ISBN) VALUES
(101, 'Winter 2018', '9780134494166'),
(102, 'Winter 2018', '9780134477367'),
(103, 'Winter 2018', '9780134494166'),
(103, 'Winter 2018', '9780134477367');
```

```
-- Insert sample textbook data
INSERT INTO Text (Book_ISBN, Book_Title, Publisher, Author) VALUES
('9780134494166', 'Introduction to Java Programming', 'Pearson', 'Y. Daniel Liang'),
('9780134477367', 'Database Systems: The Complete Book', 'Pearson', 'Hector Garcia-Molina');
```

Results | Messages

	StuNo	Name	Major	Bdate
1	1	Ravi Shankar	Computer Science	2000-01-01
2	2	John Doe	Engineering	1999-05-15
3	3	Alice Johnson	Mathematics	2001-07-20

	Course_id	Cname	Dept
1	101	Introduction to Programming	CSE
2	102	Database Management Systems	CSE
3	103	Calculus	Math
4	104	Introduction to Engineering	Engineering

	StudNo	Course_id	Quarter	Grade
1	1	101	Winter 2018	A
2	1	102	Winter 2018	B
3	1	103	Winter 2018	A
4	2	101	Winter 2018	B
5	2	102	Winter 2018	A
6	3	101	Winter 2018	A

	Course_id	Quarter	Book_ISBN

	Book_ISBN	Book_Title	Publisher	Author
1	9780134477367	Database Systems: The Complete Book	Pearson	Hector Garcia-Molina
2	9780134494166	Introduction to Java Programming	Pearson	Y. Daniel Liang

```
SELECT COUNT(*) AS NumCourses
FROM Enroll e
JOIN Student s ON e.StudNo = s.StuNo
WHERE s.Name = 'Ravi Shankar' AND e.Quarter = 'Winter 2018';
```

	NumCourses
1	3

```
SELECT ba.Course_id, ba.Quarter, t.Book_Title
FROM Book_Adoption ba
JOIN Course c ON ba.Course_id = c.Course_id
JOIN Text t ON ba.Book_ISBN = t.Book_ISBN
WHERE c.Dept = 'CSE'
GROUP BY ba.Course_id, ba.Quarter
HAVING COUNT(*) > 2;
```

(no soln found)

```
SELECT c.Dept
FROM Course c
LEFT JOIN Book_Adoption ba ON c.Course_id = ba.Course_id
LEFT JOIN Text t ON ba.Book_ISBN = t.Book_ISBN
GROUP BY c.Dept
HAVING COUNT(*) = SUM(CASE WHEN t.Publisher = 'TMH Publishing' THEN 1 ELSE 0
END);
```

(no soln)

```
CREATE TRIGGER PreventInvalidEnrollment
ON Enroll
INSTEAD OF INSERT
AS
BEGIN
  IF EXISTS (SELECT * FROM inserted i LEFT JOIN Student s ON i.StudNo = s.StuNo
WHERE s.StuNo IS NULL)
  BEGIN
    RAISERROR ('Invalid student', 16, 1);
  END
  ELSE IF EXISTS (SELECT * FROM inserted i LEFT JOIN Course c ON i.Course_id =
c.Course_id WHERE c.Course_id IS NULL)
  BEGIN
    RAISERROR ('Course not offered', 16, 1);
  END
  ELSE
  BEGIN
    INSERT INTO Enroll (StudNo, Course_id, Quarter, Grade)
    SELECT StudNo, Course_id, Quarter, Grade FROM inserted;
  END
END;
```

```
CREATE PROCEDURE GetTextbookByISBN
  @ISBN VARCHAR(20)
AS
BEGIN
  SELECT *
  FROM Text
  WHERE Book_ISBN = @ISBN;
END;
```

EXEC GetTextbookByISBN @ISBN= 9780134477367

	Book_ISBN	Book_Title	Publisher	Author
1	9780134477367	Database Systems: The Complete Book	Pearson	Hector Garcia-Molina

EXPERIMENT 16 (Consider the following relations for a database that keeps track of auto sales in a car dealership Car(Serial_No, Model, Manufacturer, Price) Options(serial-No, Option-Name, Price) Sales(Salesperson-id, Serial-No. Date, Sale-Price) Salesperson(Salesperson-id,Name,Phone)

a. List the serial and Model of cars that have no options.

b. For the salesperson named 'Jane Doe', list the following information for all the cars she sold

c. List the name of the salesperson who have no phone.

d. Write any one trigger

e. Write any one procedure or function)

```
create database exp16;
use exp16;

CREATE TABLE Car (
   Serial_No INT PRIMARY KEY,
   Model VARCHAR(100),
   Manufacturer VARCHAR(100),
   Price DECIMAL(10, 2)
);
CREATE TABLE Options (
   Serial_No INT,
   Option_Name VARCHAR(100),
   Price DECIMAL(10, 2),
   FOREIGN KEY (Serial_No) REFERENCES Car(Serial_No)
);
CREATE TABLE Sales (
   Salesperson_id INT,
   Serial_No INT,
   Date DATE,
   Sale_Price DECIMAL(10, 2),
   FOREIGN KEY (Salesperson_id) REFERENCES Salesperson(Salesperson_id),
```

```
   FOREIGN KEY (Serial_No) REFERENCES Car(Serial_No)
);
drop TABLE Sales;
CREATE TABLE Salesperson (
   Salesperson_id INT PRIMARY KEY,
   Name VARCHAR(100),
   Phone VARCHAR(20)
);
-- Insert sample car data
INSERT INTO Car (Serial_No, Model, Manufacturer, Price) VALUES
(1, 'Corolla', 'Toyota', 20000.00),
(2, 'Civic', 'Honda', 22000.00),
(3, 'Camry', 'Toyota', 25000.00);

-- Insert sample options data
INSERT INTO Options (Serial_No, Option_Name, Price) VALUES
(1, 'Sunroof', 1000.00),
(1, 'Leather Seats', 1500.00),
(3, 'Navigation System', 1200.00);

-- Insert sample salesperson data
INSERT INTO Salesperson (Salesperson_id, Name, Phone) VALUES
(101, 'John Smith', '123-456-7890'),
(102, 'Jane Doe', NULL),
(103, 'Alice Johnson', '987-654-3210');

-- Insert sample sales data
INSERT INTO Sales (Salesperson_id, Serial_No, Date, Sale_Price) VALUES
(101, 1, '2024-04-01', 21000.00),
(102, 2, '2024-04-05', 23000.00),
(102, 3, '2024-04-10', 26000.00);

select* from Car
select* from Options
select* from Salesperson
select* from Sales;
```

150 %

Results | Messages

	Serial_No	Model	Manufacturer	Price
1	1	Corolla	Toyota	20000.00
2	2	Civic	Honda	22000.00
3	3	Camry	Toyota	25000.00

	Serial_No	Option_Name	Price
1	1	Sunroof	1000.00
2	1	Leather Seats	1500.00
3	3	Navigation System	1200.00

	Salesperson_id	Name	Phone
1	101	John Smith	123-456-7890
2	102	Jane Doe	NULL
3	103	Alice Johnson	987-654-3210

	Salesperson_id	Serial_No	Date	Sale_Price
1	101	1	2024-04-01	21000.00
2	102	2	2024-04-05	23000.00
3	102	3	2024-04-10	26000.00

```
SELECT Serial_No, Model
FROM Car
WHERE Serial_No NOT IN (SELECT Serial_No FROM Options);
```

Results | Messages

	Serial_No	Model
1	2	Civic

```
SELECT c.Serial_No, c.Model, c.Manufacturer, s.Date, s.Sale_Price
FROM Sales s
JOIN Car c ON s.Serial_No = c.Serial_No
JOIN Salesperson sp ON s.Salesperson_id = sp.Salesperson_id
WHERE sp.Name = 'Jane Doe';
```

Results | Messages

	Serial_No	Model	Manufacturer	Date	Sale_Price
1	2	Civic	Honda	2024-04-05	23000.00
2	3	Camry	Toyota	2024-04-10	26000.00

```
SELECT Name
FROM Salesperson
```

```
WHERE Phone IS NULL;
```

Results | Messages

	Name
1	Jane Doe

```
CREATE TRIGGER PreventLowSalePrice
ON Sales
INSTEAD OF INSERT
AS
BEGIN
  IF EXISTS (SELECT * FROM inserted i JOIN Car c ON i.Serial_No = c.Serial_No WHERE
i.Sale_Price < c.Price)
  BEGIN
    RAISERROR ('Sale price cannot be lower than car price', 16, 1);
  END
  ELSE
  BEGIN
    INSERT INTO Sales (Salesperson_id, Serial_No, Date, Sale_Price)
    SELECT Salesperson_id, Serial_No, Date, Sale_Price FROM inserted;
  END
END;


CREATE PROCEDURE GetSalespersonById
  @Salesperson_id INT
AS
BEGIN
  SELECT *
  FROM Salesperson
  WHERE Salesperson_id = @Salesperson_id;
END;

EXEC GetSalespersonbyID @Salesperson_id= 101
```

150 %

Results | Messages

	Salesperson_id	Name	Phone
1	101	John Smith	123-456-7890

EXPERIMENT 17

(Consider the following six relations for an order-processing database application in a company: CUSTOMER(Cust#, Cname, City) ORDER(Order, Odate,Cust#, Ord_Amt) ORDER_ITEM(Order, Item, Qty) ITEM(Item, Unit_Price) SHIPMENT(Order, Warehouse#, Ship_date) WAREHOUSE(Warehouse#, City)

a. List the Order and Ship_date for all orders shipped from Warehouse number 'W122'

b. List the orders that were not shipped within 30 days of ordering

c. List the Order for orders that were shipped from all warehouses that the company has in New York

. d. Write any one trigger

e. Write any one procedure or function)

```
create database exp17;
use exp17;

CREATE TABLE CUSTOMER (
   CustID INT PRIMARY KEY,
   Cname VARCHAR(100),
   City VARCHAR(100)
);
drop table CUSTOMER;

CREATE TABLE DORDER (
   OrderID INT PRIMARY KEY,
   Odate DATE,
   CustID INT,
   Ord_Amt DECIMAL(10, 2),
   FOREIGN KEY (CustID) REFERENCES CUSTOMER(CustID)
);
DROP TABLE DORDER;
CREATE TABLE ORDER_ITEM (
   OrderID INT,
   Item INT,
   Qty INT,
   PRIMARY KEY (OrderID, Item),
   FOREIGN KEY (OrderID) REFERENCES DORDER(OrderID)
);
DROP TABLE ORDER_ITEM;
```

```sql
CREATE TABLE ITEM (
   Item INT PRIMARY KEY,
   Unit_Price DECIMAL(10, 2)
);
DROP TABLE ITEM;
CREATE TABLE SHIPMENT (
   OrderID INT,
   WarehouseID VARCHAR(20),
   Ship_date DATE,
   PRIMARY KEY (OrderID),
   FOREIGN KEY (OrderID) REFERENCES DORDER(OrderID)
);
DROP TABLE SHIPMENT;
CREATE TABLE WAREHOUSE (
   WarehouseID VARCHAR(20) PRIMARY KEY,
   City VARCHAR(100)
);
DROP TABLE WAREHOUSE;

-- Insert sample customer data
INSERT INTO CUSTOMER (CustID, Cname, City) VALUES
(1, 'ABC Inc.', 'New York'),
(2, 'XYZ Corp.', 'Los Angeles'),
(3, 'DEF Ltd.', 'New York');

-- Insert sample order data
INSERT INTO DORDER (OrderID, Odate, CustID, Ord_Amt) VALUES
(101, '2024-04-01', 1, 500.00),
(102, '2024-04-05', 2, 700.00),
(103, '2024-04-10', 3, 1000.00);

-- Insert sample order item data
INSERT INTO ORDER_ITEM (OrderID, Item, Qty) VALUES
(101, 1, 2),
(101, 2, 3),
(102, 1, 1),
(102, 3, 2),
(103, 2, 4),
(103, 3, 3);

-- Insert sample item data
```

```
INSERT INTO ITEM (Item, Unit_Price) VALUES
(1, 50.00),
(2, 30.00),
(3, 40.00);

-- Insert sample shipment data
INSERT INTO SHIPMENT (OrderID, WarehouseID, Ship_date) VALUES
(101, 'W122', '2024-04-05'),
(102, 'W123', '2024-04-12'),
(103, 'W122', '2024-04-15');

-- Insert sample warehouse data
INSERT INTO WAREHOUSE (WarehouseID, City) VALUES
('W122', 'New York'),
('W123', 'Los Angeles');

Select* from CUSTOMER
select* from DORDER
select* from ORDER_ITEM
select* from ITEM
select* from SHIPMENT
select* from WAREHOUSE;
```

Results | Messages

	CustID	Cname	City
1	1	ABC Inc.	New York
2	2	XYZ Corp.	Los Angeles
3	3	DEF Ltd.	New York

	OrderID	Odate	CustID	Ord_Amt
1	101	2024-04-01	1	500.00
2	102	2024-04-05	2	700.00
3	103	2024-04-10	3	1000.00

	OrderID	Item	Qty
1	101	1	2
2	101	2	3
3	102	1	1
4	102	3	2
5	103	2	4
6	103	3	3

	Item	Unit_Price
1	1	50.00
2	2	30.00
3	3	40.00

	OrderID	WarehouseID	Ship_date
1	101	W122	2024-04-05
2	102	W123	2024-04-12
3	103	W122	2024-04-15

	WarehouseID	City
1	W122	New York
2	W123	Los Angeles

```
SELECT OrderID, Ship_date
FROM SHIPMENT
WHERE WarehouseID = 'W122';
```

50 %

Results | Messages

	OrderID	Ship_date
1	101	2024-04-05
2	103	2024-04-15

```
SELECT o.OrderID
FROM DORDER o
LEFT JOIN SHIPMENT s ON o.OrderID = s.OrderID
WHERE DATEDIFF(DAY, o.Odate, s.Ship_date) > 30 OR s.Ship_date IS NULL;
```

(no result)

```
SELECT OrderID
FROM DORDER o
WHERE NOT EXISTS (
   SELECT w.WarehouseID
   FROM WAREHOUSE w
   WHERE w.City = 'New York'
   EXCEPT
   SELECT s.WarehouseID
   FROM SHIPMENT s
   WHERE s.OrderID = o.OrderID
);
```

	OrderID
1	101
2	103

```
CREATE TRIGGER UpdateOrderAmount
ON ORDER_ITEM
AFTER UPDATE
AS
BEGIN
   UPDATE o
   SET Ord_Amt = (SELECT SUM(Qty * i.Unit_Price) FROM ORDER_ITEM oi JOIN ITEM i
ON oi.Item = i.Item WHERE oi.OrderID = o.OrderID)
   FROM DORDER o
   JOIN inserted i ON o.OrderID = i.OrderID;
END;
```


```
CREATE TRIGGER UpdateOrderAmount
ON ORDER_ITEM
AFTER UPDATE
AS
BEGIN
   UPDATE o
   SET Ord_Amt = (SELECT SUM(Qty * i.Unit_Price) FROM ORDER_ITEM oi JOIN ITEM i
ON oi.Item = i.Item WHERE oi.OrderID = o.OrderID)
   FROM DORDER o
   JOIN inserted i ON o.OrderID = i.OrderID;
END;
```

EXPERIMENT 18

(Design a COMPANY database which is specified as below Employee(EID, Name, Bdate, Address, Salary, DeptID) Department(DeptID, Dname, Office, Mng-EID) Project(Code, Name, Budget, DeptID) Join(EID, Pcode, StartDate) Emp-Dependent(EID, Dependent-Name, Bdate, Relationship)

a. Find the name of the employee who joined in ‘Green Green” project

. b. Find the name of the employees who have no dependents

c. Find the name of the employees who work for both project number 1 and project number 2

. d. Write any one trigger

e. Write any one procedure or function)

```
create database exp18;
use exp18;
 CREATE TABLE Employee (
   EID INT PRIMARY KEY,
   Name VARCHAR(100),
   Bdate DATE,
   Address VARCHAR(255),
   Salary DECIMAL(10, 2),
   DeptID INT
);
CREATE TABLE Department (
   DeptID INT PRIMARY KEY,
   Dname VARCHAR(100),
   Office VARCHAR(100),
   Mng_EID INT,
   FOREIGN KEY (Mng_EID) REFERENCES Employee(EID)
);
CREATE TABLE Project (
   Code INT PRIMARY KEY,
   Name VARCHAR(100),
   Budget DECIMAL(12, 2),
   DeptID INT,
   FOREIGN KEY (DeptID) REFERENCES Department(DeptID)
);
```

```sql
CREATE TABLE Eve (
   EID INT,
   Pcode INT,
   StartDate DATE,
   PRIMARY KEY (EID, Pcode),
   FOREIGN KEY (EID) REFERENCES Employee(EID),
   FOREIGN KEY (Pcode) REFERENCES Project(Code)
);

CREATE TABLE Emp_Dependent (
   EID INT,
   Dependent_Name VARCHAR(100),
   Bdate DATE,
   Relationship VARCHAR(100),
   PRIMARY KEY (EID, Dependent_Name),
   FOREIGN KEY (EID) REFERENCES Employee(EID)
);

-- Insert sample employee data
INSERT INTO Employee (EID, Name, Bdate, Address, Salary, DeptID) VALUES
(1, 'John Doe', '1990-05-15', '123 Main St', 50000.00, 1),
(2, 'Jane Smith', '1988-10-20', '456 Elm St', 60000.00, 2),
(3, 'Alice Johnson', '1995-03-25', '789 Oak St', 55000.00, 1);

-- Insert sample department data
INSERT INTO Department (DeptID, Dname, Office, Mng_EID) VALUES
(1, 'HR', 'Building A', 1),
(2, 'Finance', 'Building B', 2);

-- Insert sample project data
INSERT INTO Project (Code, Name, Budget, DeptID) VALUES
(1, 'Green Green', 100000.00, 1),
(2, 'Blue Blue', 150000.00, 2);

-- Insert sample employee-project assignments
INSERT INTO Eve (EID, Pcode, StartDate) VALUES
(1, 1, '2024-01-01'),
(2, 2, '2024-02-01'),
(3, 1, '2024-03-01'),
(3, 2, '2024-04-01');
```

-- Insert sample employee dependents data
INSERT INTO Emp_Dependent (EID, Dependent_Name, Bdate, Relationship) VALUES
(1, 'Emily Doe', '2010-02-10', 'Child'),
(2, 'Michael Smith', '2012-05-20', 'Child');

150 %

Results | Messages

	EID	Name	Bdate	Address	Salary	DeptID
1	1	John Doe	1990-05-15	123 Main St	50000.00	1
2	2	Jane Smith	1988-10-20	456 Elm St	60000.00	2
3	3	Alice Johnson	1995-03-25	789 Oak St	55000.00	1

	DeptID	Dname	Office	Mng_EID
1	1	HR	Building A	1
2	2	Finance	Building B	2

	Code	Name	Budget	DeptID
1	1	Green Green	100000.00	1
2	2	Blue Blue	150000.00	2

	EID	Pcode	StartDate
1	1	1	2024-01-01
2	2	2	2024-02-01
3	3	1	2024-03-01
4	3	2	2024-04-01

	EID	Dependent_Name	Bdate	Relationship
1	1	Emily Doe	2010-02-10	Child
2	2	Michael Smith	2012-05-20	Child

SELECT e.Name
FROM Employee e
JOIN Eve j ON e.EID = j.EID
JOIN Project p ON j.Pcode = p.Code
WHERE p.Name = 'Green Green';

	Name
1	John Doe
2	Alice Johnson

SELECT Name
FROM Employee
WHERE EID NOT IN (SELECT DISTINCT EID FROM Emp_Dependent);

	Name
1	Alice Johnson

```
SELECT e.Name
FROM Employee e
JOIN Eve j ON e.EID = j.EID
WHERE j.Pcode IN (1, 2)
GROUP BY e.Name
HAVING COUNT(DISTINCT j.Pcode) = 2;
```

	Name
1	Alice Johnson

```
CREATE PROCEDURE GetEmployeeInfoById
   @EID INT
AS
BEGIN
   SELECT *
   FROM Employee
   WHERE EID = @EID;
END;
```

PRACTICAL 19

(Design the library database which is used to keep track of books, borrowers, and book loans

a. Retrieve the names of all borrowers who do not have any books checked out

b. How many copies of the book titled The Lost Tribe are owned by each library branch?

c. Retrieve the names, address, and number of books checked out for all borrowers who have more than five books checked out.

d. Write any one trigger

e. Write any one procedure or function)

```
create database exp19;
use exp19;

CREATE TABLE Books (
   BookID INT PRIMARY KEY,
   Title VARCHAR(255),
   Author VARCHAR(255),
   BranchID INT,
```

```
  FOREIGN KEY (BranchID) REFERENCES LibraryBranch(BranchID)
);
CREATE TABLE Borrowers (
  BorrowerID INT PRIMARY KEY,
  Name VARCHAR(100),
  Address VARCHAR(255)
);
CREATE TABLE BookLoans (
  LoanID INT PRIMARY KEY,
  BookID INT,
  BorrowerID INT,
  CheckoutDate DATE,
  ReturnDate DATE,
  FOREIGN KEY (BookID) REFERENCES Books(BookID),
  FOREIGN KEY (BorrowerID) REFERENCES Borrowers(BorrowerID)
);
CREATE TABLE LibraryBranch (
  BranchID INT PRIMARY KEY,
  Name VARCHAR(100),
  Address VARCHAR(255)
);
-- Insert sample library branch data
INSERT INTO LibraryBranch (BranchID, Name, Address) VALUES
(1, 'Central Library', '123 Main St'),
(2, 'Westside Branch', '456 Elm St');

-- Insert sample book data
INSERT INTO Books (BookID, Title, Author, BranchID) VALUES
(101, 'The Lost Tribe', 'Jane Doe', 1),
(102, 'The Lost Tribe', 'Jane Doe', 2),
(103, 'The Great Gatsby', 'F. Scott Fitzgerald', 1),
(104, 'To Kill a Mockingbird', 'Harper Lee', 2);

-- Insert sample borrower data
INSERT INTO Borrowers (BorrowerID, Name, Address) VALUES
(201, 'John Smith', '789 Oak St'),
(202, 'Alice Johnson', '101 Pine St'),
(203, 'Bob Jones', '222 Maple St');

-- Insert sample book loans data
```

```
INSERT INTO BookLoans (LoanID, BookID, BorrowerID, CheckoutDate, ReturnDate)
VALUES
(301, 101, 201, '2024-04-01', NULL),
(302, 102, 202, '2024-04-02', NULL),
(303, 101, 203, '2024-04-03', NULL),
(304, 103, 201, '2024-04-04', NULL),
(305, 104, 202, '2024-04-05', NULL),
(306, 101, 202, '2024-04-06', NULL);

-- Insert additional book loans for testing
INSERT INTO BookLoans (LoanID, BookID, BorrowerID, CheckoutDate, ReturnDate)
VALUES
(307, 102, 201, '2024-04-07', NULL),
(308, 103, 202, '2024-04-08', NULL),
(309, 104, 203, '2024-04-09', NULL),
(310, 101, 201, '2024-04-10', NULL);

select* from LibraryBranch
select* from Books
select* from Borrowers
select* from BookLoans;
```

Results | Messages

	BranchID	Name	Address
1	1	Central Library	123 Main St
2	2	Westside Branch	456 Elm St

	BookID	Title	Author	BranchID
1	101	The Lost Tribe	Jane Doe	1
2	102	The Lost Tribe	Jane Doe	2
3	103	The Great Gatsby	F. Scott Fitzgerald	1
4	104	To Kill a Mockingbird	Harper Lee	2

	BorrowerID	Name	Address
1	201	John Smith	789 Oak St
2	202	Alice Johnson	101 Pine St
3	203	Bob Jones	222 Maple St

	LoanID	BookID	BorrowerID	CheckoutDate	ReturnDate
1	301	101	201	2024-04-01	NULL
2	302	102	202	2024-04-02	NULL
3	303	101	203	2024-04-03	NULL
4	304	103	201	2024-04-04	NULL

```
SELECT b.Name
```

```
FROM Borrowers b
LEFT JOIN BookLoans bl ON b.BorrowerID = bl.BorrowerID
WHERE bl.LoanID IS NULL;

SELECT lb.Name AS BranchName, COUNT(b.BookID) AS NumCopies
FROM LibraryBranch lb
JOIN Books b ON lb.BranchID = b.BranchID
WHERE b.Title = 'The Lost Tribe'
GROUP BY lb.Name;
```

	BranchName	NumCopies
1	Central Library	1
2	Westside Branch	1

```
SELECT b.Name, b.Address, COUNT(bl.LoanID) AS NumBooksCheckedOut
FROM Borrowers b
JOIN BookLoans bl ON b.BorrowerID = bl.BorrowerID
GROUP BY b.Name, b.Address
HAVING COUNT(bl.LoanID) > 5;

CREATE TRIGGER UpdateBookStatus
ON BookLoans
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
   -- Update book status based on loan status
   -- (e.g., 'Checked Out' if loan exists, 'Available' if no loan exists)
   UPDATE Books
   SET Status = CASE
            WHEN EXISTS (SELECT * FROM BookLoans WHERE BookID = inserted.BookID
AND ReturnDate IS NULL)
               THEN 'Checked Out'
            ELSE 'Available'
         END
   FROM inserted
   WHERE Books.BookID = inserted.BookID;
END;

CREATE PROCEDURE GetBookInfoByTitle
   @Title VARCHAR(255)
AS
```

```
BEGIN
   SELECT *
   FROM Books
   WHERE Title = @Title;
END;
```

EXPERIMENT 20!!!

(Design the library database which is used to keep track of books, borrowers, and book loans

a. How many copies of the book titled The Lost Tribe are owned by the library branch whose name is 'Sharps town'.

b. For each library branch, retrieve the branch name and total number of books loaned out from that branch

c. Update the database four new books.

d. Write any one trigger

e. Write any one procedure or function)

```
use exp20;
CREATE TABLE Books (
   BookID INT PRIMARY KEY,
   Title VARCHAR(255),
   Author VARCHAR(255),
   BranchID INT,
   FOREIGN KEY (BranchID) REFERENCES LibraryBranch(BranchID)
);
CREATE TABLE Borrowers (
   BorrowerID INT PRIMARY KEY,
   Name VARCHAR(100),
   Address VARCHAR(255)
);
CREATE TABLE BookLoans (
   LoanID INT PRIMARY KEY,
   BookID INT,
   BorrowerID INT,
   CheckoutDate DATE,
   ReturnDate DATE,
   FOREIGN KEY (BookID) REFERENCES Books(BookID),
   FOREIGN KEY (BorrowerID) REFERENCES Borrowers(BorrowerID)
```

```
);
CREATE TABLE LibraryBranch (
   BranchID INT PRIMARY KEY,
   Name VARCHAR(100),
   Address VARCHAR(255)
);
-- Insert sample library branch data
INSERT INTO LibraryBranch (BranchID, Name, Address) VALUES
(1, 'Central Library', '123 Main St'),
(2, 'Sharps town', '456 Elm St');

-- Insert sample book data
INSERT INTO Books (BookID, Title, Author, BranchID) VALUES
(101, 'The Lost Tribe', 'Jane Doe', 1),
(102, 'The Lost Tribe', 'Jane Doe', 2),
(103, 'The Great Gatsby', 'F. Scott Fitzgerald', 1),
(104, 'To Kill a Mockingbird', 'Harper Lee', 2);

-- Insert sample borrower data
INSERT INTO Borrowers (BorrowerID, Name, Address) VALUES
(201, 'John Smith', '789 Oak St'),
(202, 'Alice Johnson', '101 Pine St'),
(203, 'Bob Jones', '222 Maple St');

-- Insert sample book loans data
INSERT INTO BookLoans (LoanID, BookID, BorrowerID, CheckoutDate, ReturnDate)
VALUES
(301, 101, 201, '2024-04-01', NULL),
(302, 102, 202, '2024-04-02', NULL),
(303, 101, 203, '2024-04-03', NULL),
(304, 103, 201, '2024-04-04', NULL),
(305, 104, 202, '2024-04-05', NULL),
(306, 101, 202, '2024-04-06', NULL);

select* from LibraryBranch
select* from Books
select* from Borrowers
select* from BookLoans;
```

150 %

Results | Messages

	BranchID	Name	Address
1	1	Central Library	123 Main St
2	2	Sharps town	456 Elm St

	BookID	Title	Author	BranchID
1	101	The Lost Tribe	Jane Doe	1
2	102	The Lost Tribe	Jane Doe	2
3	103	The Great Gatsby	F. Scott Fitzgerald	1
4	104	To Kill a Mockingbird	Harper Lee	2

	BorrowerID	Name	Address
1	201	John Smith	789 Oak St
2	202	Alice Johnson	101 Pine St
3	203	Bob Jones	222 Maple St

	LoanID	BookID	BorrowerID	CheckoutDate	ReturnDate
1	301	101	201	2024-04-01	NULL
2	302	102	202	2024-04-02	NULL
3	303	101	203	2024-04-03	NULL
4	304	102	201	2024-04-04	NULL

Query executed succes... | OLIVEDANIEL\MSSQLSERVER01 (... | OLIVEDANIEL\o

```
SELECT COUNT(BookID) AS NumCopies
FROM Books
WHERE Title = 'The Lost Tribe' AND BranchID = (SELECT BranchID FROM LibraryBranch
WHERE Name = 'Sharps town');
```

Results | Messages

	NumCopies
1	1

```
SELECT lb.Name AS BranchName, COUNT(bl.BookID) AS NumBooksLoanedOut
FROM LibraryBranch lb
LEFT JOIN Books b ON lb.BranchID = b.BranchID
LEFT JOIN BookLoans bl ON b.BookID = bl.BookID
GROUP BY lb.Name;
```

	BranchName	NumBooksLoanedOut
1	Central Library	4
2	Sharps town	2

```
CREATE TRIGGER UpdateBookStatus
```

```
ON BookLoans
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
   -- Update book status based on loan status
   -- (e.g., 'Checked Out' if loan exists, 'Available' if no loan exists)
   UPDATE Books
   SET Status = CASE
              WHEN EXISTS (SELECT * FROM BookLoans WHERE BookID = inserted.BookID
AND ReturnDate IS NULL)
                 THEN 'Checked Out'
              ELSE 'Available'
            END
   FROM inserted
   WHERE Books.BookID = inserted.BookID;
END;


CREATE TRIGGER UpdateBookStatus
ON BookLoans
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
   -- Update book status based on loan status
   -- (e.g., 'Checked Out' if loan exists, 'Available' if no loan exists)
   UPDATE Books
   SET Status = CASE
              WHEN EXISTS (SELECT * FROM BookLoans WHERE BookID = inserted.BookID
AND ReturnDate IS NULL)
                 THEN 'Checked Out'
              ELSE 'Available'
            END
   FROM inserted
   WHERE Books.BookID = inserted.BookID;
END;
```