Table of Content

EX: 1 Write a program to create DDL and DML commands.

EX 1.1 Write a program to create DDL

AIM:

Create a program for DDL and DML command.

PROCEDURE:

- Open an Oracle then click SQL command.
- Create a table.
- Alter can be adding attributes to an existing relation.
- Drop command deletes all information about the dropped relation from the database.
- Truncate command removes all the records from the table.
- Rename command is used to rename the objects.

PROGRAM:

CREATE TABLE:

```
CREATE TABLE EMPLOYEES (EMPNO NUMBER (5), NAME VARCHAR2(15),
DESIGNATION VARCHAR2(10), SALARY NUMBER (20));
```

INSERT VALUES:

1. INSERT INTO EMPLOYEES VALUES(111,'ANBU','PROFESSOR',10000);
2. INSERT INTO EMPLOYEES VALUES(112,'KUMAR','OA',7000);
3. INSERT INTO EMPLOYEES VALUES(113,'SAKTHI','VAO',12000);

SELECT*FROM EMPLOYEES;

ALTER TABLE:

```
ALTER TABLE EMPLOYEES MODIFY DESIGNATION VARCHAR2(20);
DESC EMPLOYEES;
ALTER TABLE EMPLOYEES ADD QUALIFICATION VARCHAR2(20);
ALTER TABLE STAFFS DROP(QUALIFICATION);
SELECT*FROM STAFFS;
TRUNCATE TABLE STAFFS;
SELECT*FROM STAFFS;
```

OUTPUT

EMPNO	NAME	DESIGNATION	SALARY
111	ANBU	PROFESSOR	10000
112	KUMAR	OA	7000
113	SAKTHI	VAO	12000

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Comment
EMPLOYEES	EMPNO	Number	-	5	0	-	✓	-
	NAME	Varchar2	15	-	-	-	✓	-
	DESIGNATION	Varchar2	20	-	-	-	✓	-
	SALARY	Number	-	20	0	-	✓	-

1 - 4

EMPNO	NAME	DESIGNATION	SALARY	QUALIFICATION
111	ANBU	PROFESSOR	10000	-
112	KUMAR	OA	7000	-
113	SAKTHI	VAO	12000	-

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
STAFFS	EMPNO	Number	-	5	0	-	✓	-	-
	NAME	Varchar2	15	-	-	-	✓	-	-
	DESIGNATION	Varchar2	20	-	-	-	✓	-	-
	SALARY	Number	-	20	0	-	✓	-	-

1 - 4

EMPNO	NAME	DESIGNATION	SALARY
111	ANBU	PROFESSOR	10000
112	KUMAR	OA	7000
113	SAKTHI	VAO	12000

Results Explain Describe Saved SQL History

no data found

RESULT:

Thus the program has been executed Successfully.

EX 1.2 Write a program to create DML commands.

AIM:

Create a program for DML command.

PROCEDURE:

- Open an Oracle then click SQL command.
- Create a table.
- Update the table records.
- Delete the records.
- Merge the records.

PROGRAM:

CREATE TABLE STUDENTS (STUDENTID INT, NAME VARCHAR (20), AGE INT, COURSE VARCHAR (50));

INSERT INTO Students VALUES (1, 'Alice Johnson', 20, 'Computer Science');

INSERT INTO Students VALUES (2, 'Bob Smith', 22, 'Mathematics')

INSERT INTO Students VALUES (3, 'Charlie Brown', 21, 'Physics')

SELECT * FROM STUDENTS;

Update age of a students

UPDATE Students SET Age = 23 WHERE StudentID = 2;

Update multiple columns

UPDATE Students SET Age = 24, Course = 'Data Science' WHERE Name = 'Alice Johnson';

Delete one record

DELETE FROM Students WHERE StudentID = 3;

Delete multiple records

DELETE FROM Students WHERE Age > 22;

Delete all records

DELETE FROM Students;

OUTPUT:

STUDENT ID	NAME	AGE	COURSE
1	Alice Johnson	20	Computer Science

2	Bob Smith	22	Mathematics
3	Charlie Brown	21	Physics
STUDENT ID	**NAME**	**AGE**	**COURSE**
1	Alice Johnson	20	Computer Science
2	Bob Smith	23	Mathematics

STUDENT ID	NAME	AGE	COURSE
1	Alice Johnson	20	Computer Science
2	Bob Smith	23	Mathematics
3	Charlie Brown	21	Physics

STUDENT ID	NAME	AGE	COURSE
1	Alice Johnson	20	Computer Science
2	Bob Smith	24	Data Science

RESULT:

Thus the program has been executed successfully.

EX 2 Write a program for inserting and retrieving the data from database.

AIM

To create a program for inserting and retrieving the data from database.

PROCEDURE

PROGRAM

```
CREATE TABLE employees (employee_id int, first_name VARCHAR2(50),last_name VARCHAR2(50),salary int);
INSERT INTO employees VALUES (101, 'John', 'Doe', 50000.00);
INSERT INTO employees VALUES (102, 'tom', 'jeery', 60000.00);
INSERT INTO employees VALUES (103, 'flix', 'dd', 50000.00);
select * from employees;
```

Retrieving specific columns from a table:

```
SELECT first_name, last_name, salary FROM employees;
```

Retrieving data with a condition:

```
SELECT * FROM employees WHERE salary < 50000;
```

Retrieving distinct values:

```
SELECT DISTINCT last_name FROM employees;
select * from employees;
```

OUTPUT

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
101	John	Doe	50000
102	tom	jeery	60000
103	flix	dd	50000

FIRST_NAME	LAST_NAME	SALARY
John	Doe	50000
tom	jeery	60000
flix	dd	50000

Results Explain Describe Saved SQL History

no data found

LAST_NAME
Doe
jeery
dd

RESULT

Thus the program has been executed successfully.

Ex 3 Write a program to create and manage multiple tables using primary key

AIM

To create a create and manage multiple tables using primary key

PROCEDURE

- Open SQL in oracle
- Create 3 Tables books, members and borrow
- Insert the table values using multiple primary keys.
- Then view the table in oracle.

PROGRAM

```
CREATE TABLE Books ( book_id INT PRIMARY KEY, title VARCHAR(100), author VARCHAR(100));
CREATE TABLE Members ( member_id INT PRIMARY KEY, name VARCHAR(100), phone VARCHAR(15));
CREATE TABLE Borrow (borrow_id INT PRIMARY KEY, member_id INT, book_id INT, borrow_date DATE, FOREIGN KEY (member_id) REFERENCES Members(member_id),FOREIGN KEY (book_id) REFERENCES Books(book_id));
INSERT INTO Books (book_id, title, author) VALUES (1, 'Harry Potter', 'J.K. Rowling');
INSERT INTO Books (book_id, title, author) VALUES(2, 'Wings of Fire', 'A.P.J. Abdul Kalam');
select * from books;
INSERT INTO Members (member_id, name, phone) VALUES (101, 'Karthik', '9876543210');
INSERT INTO Members (member_id, name, phone) VALUES(102, 'Divya', '9123456780');
select * FROM MEMBERS;
INSERT INTO Borrow (borrow_id, member_id, book_id, borrow_date) VALUES (1, 101, 1, '20-aug-2025');
INSERT INTO Borrow (borrow_id, member_id, book_id, borrow_date) VALUES(2, 102, 2, '21-aug-2025');
SELECT m.name, b.title, br.borrow_date FROM Borrow br JOIN Members m ON br.member_id = m.member_id JOIN Books b ON br.book_id = b.book_id;
```

OUTPUT

BOOK_ID	TITLE	AUTHOR
1	Harry Potter	J.K. Rowling
2	Wings of Fire	A.P.J. Abdul Kalam

MEMBER_ID	NAME	PHONE
101	Karthik	9876543210
102	Divya	9123456780

NAME	TITLE	BORROW_DATE
Karthik	Harry Potter	20-AUG-25
Divya	Wings of Fire	21-AUG-25

RESULT

Thus the program, has been executed successfully.

EX 4 Write a program to design a relational database using primary key, Foreign Key, Candidate Key, Super Key.

AIM

To write a program to design a relational database using primary key, Foreign Key, Candidate Key, Super Key.

PROCEDURE

Primary Key

- student_id in participant
- course_id in Circullam
- enrollment_id in Enroll

Candidate Keys

- email in participant
- course_code in Circullam
- (student_id, course_id) in Enroll

Super Keys

Any superset of candidate keys (e.g., {student_id, email} in participant).

Foreign Keys

- student_id in Enroll → participant(student_id)
- course_id in Enroll → Circullam (course_id)

PROGRAM

```
CREATE TABLE participant (
   student_id NUMBER PRIMARY KEY,
   email VARCHAR2(100) UNIQUE,
   name VARCHAR2(100),
   dob DATE
);
INSERT INTO participant (student_id, email, name, dob) VALUES
(1, 'alice@example.com', 'Alice Johnson','20-aug-2025');
INSERT INTO participant (student_id, email, name, dob) VALUES
(2, 'bob@example.com', 'Bob Smith','22-aug-1999');
INSERT INTO participant (student_id, email, name, dob) VALUES
(3, 'carol@example.com', 'Carol White', '30-jan-2001');
select * from participant;
```

```
CREATE TABLE Circullam (
   course_id NUMBER PRIMARY KEY,
   course_code VARCHAR2(20) UNIQUE,
   course_name VARCHAR2(100)
);
INSERT INTO Circullam (course_id, course_code, course_name) VALUES
(101, 'CS101', 'Introduction to Computer Science');

INSERT INTO Circullam (course_id, course_code, course_name) VALUES
(102, 'MATH201', 'Discrete Mathematics');
INSERT INTO Circullam (course_id, course_code, course_name) VALUES
(103, 'ENG150', 'English Literature');
select * from Circullam;
CREATE TABLE Enroll (
   enrollment_id NUMBER PRIMARY KEY,
   student_id NUMBER,
   course_id NUMBER,
   grade CHAR(2),
      CONSTRAINT fk_participant FOREIGN KEY (student_id)
      REFERENCES participant(student_id),
   CONSTRAINT fk_Circullam FOREIGN KEY (course_id)
      REFERENCES Circullam(course_id),
      CONSTRAINT uq_enroll UNIQUE (student_id, course_id)
);
INSERT INTO Enroll (enrollment_id, student_id, course_id, grade) VALUES
(1001, 1, 101, 'A');
INSERT INTO Enroll (enrollment_id, student_id, course_id, grade) VALUES
(1002, 1, 102, 'B');
INSERT INTO Enroll (enrollment_id, student_id, course_id, grade) VALUES
(1003, 2, 101, 'B');
INSERT INTO Enroll(enrollment_id, student_id, course_id, grade) VALUES
(1004, 3, 103, 'A');
select * from Enroll;
SELECT s.student_id, s.name AS student_name, c.course_code, c.course_name, e.grade
FROM Enroll e
JOIN participant s ON e.student_id = s.student_id
JOIN Circullam c ON e.course_id = c.course_id;
SELECT c.course_code, c.course_name, e.grade
FROM Enroll e
JOIN Circullam c ON e.course_id = c.course_id
JOIN participant s ON e.student_id = s.student_id
WHERE s.email = 'alice@example.com';
SELECT s.name, c.course_name, e.grade
FROM Enroll e
JOIN participant s ON e.student_id = s.student_id
JOIN Circullam c ON e.course_id = c.course_id
WHERE e.grade = 'A';
```

OUTPUT

STUDENT_ID	EMAIL	NAME	DOB
1	alice@example.com	Alice Johnson	20-AUG-25
2	bob@example.com	Bob Smith	22-AUG-99
3	carol@example.com	Carol White	30-JAN-01

COURSE_ID	COURSE_CODE	COURSE_NAME
101	CS101	Introduction to Computer Science
102	MATH201	Discrete Mathematics
103	ENG150	English Literature

ENROLLMENT_ID	STUDENT_ID	COURSE_ID	GRADE
1001	1	101	A
1002	1	102	B
1003	2	101	B
1004	3	103	A

STUDENT_ID	STUDENT_NAME	COURSE_CODE	COURSE_NAME	GRADE
2	Bob Smith	CS101	Introduction to Computer Science	B
1	Alice Johnson	CS101	Introduction to Computer Science	A
1	Alice Johnson	MATH201	Discrete Mathematics	B
3	Carol White	ENG150	English Literature	A

COURSE_CODE	COURSE_NAME	GRADE
CS101	Introduction to Computer Science	A
MATH201	Discrete Mathematics	B

NAME	COURSE_NAME	GRADE
Alice Johnson	Introduction to Computer Science	A
Carol White	English Literature	A

RESULT

Thus the program, has been executed successfully.

EX 5 Write a program to convert the unnormalized table to third normal form (3NF).

AIM

Create a program to convert the unnormalized table to third normal form (3NF).

PROCEDURE

- Create an Unnormalized Table (UNF)
- Convert to First Normal Form (1NF)
  - Remove repeating groups so that each column contains atomic values.
- Convert to Second Normal Form (2NF)
  - Eliminate partial dependencies by dividing the data into separate tables.
- Convert to Third Normal Form (3NF):
  - Remove transitive dependencies so that non-key attributes depend only on the primary key.
- Verify and Execute SQL Commands.

PROGRAM

```
CREATE TABLE Student (
   StudentID INT PRIMARY KEY,
   StudentName VARCHAR(100),
   AdvisorName VARCHAR(100)
);
INSERT INTO Student VALUES (1, 'Alice Brown', 'Dr. Smith');
INSERT INTO Student VALUES(2, 'Bob Martin', 'Prof. Johnson');
INSERT INTO Student VALUES(3, 'Charlie Davis', 'Dr. Williams');
INSERT INTO Student VALUES(4, 'Diana Clark', 'Dr. Smith');
select * from Student;
CREATE TABLE Course (
   StudentID INT,
   Course VARCHAR(100),
   PRIMARY KEY (StudentID, Course),
   FOREIGN KEY (StudentID) REFERENCES Student(StudentID)
);
```

```
INSERT INTO Course VALUES(1, 'Mathematics');
INSERT INTO Course VALUES(1, 'Computer Science');
INSERT INTO Course VALUES(2, 'History');
INSERT INTO Course VALUES(2, 'Political Science');
INSERT INTO Course VALUES(3, 'Physics');
INSERT INTO Course VALUES(3, 'Chemistry');
INSERT INTO Course VALUES(4, 'Mathematics');
INSERT INTO Course VALUES(4, 'English Literature');
select * from Course;
CREATE TABLE Advisor (
    AdvisorName VARCHAR(100) PRIMARY KEY,
    AdvisorOffice VARCHAR(50)
);
INSERT INTO Advisor VALUES ('Dr. Smith', 'Room A101');
INSERT INTO Advisor VALUES('Prof. Johnson', 'Room B202');
INSERT INTO Advisor VALUES('Dr. Williams', 'Room C303');
select * from Advisor;
SELECT
    s.StudentID,
    s.StudentName,
    sc.Course,
    s.AdvisorName,
    a.AdvisorOffice
FROM Student s
JOIN Course sc ON s.StudentID = sc.StudentID
JOIN Advisor a ON s.AdvisorName = a.AdvisorName;
```

OUTPUT

STUDENTID	STUDENTNAME	COURSE	ADVISORNAME	ADVISOROFFICE
1	Alice Brown	Computer Science	Dr. Smith	Room A101
1	Alice Brown	Mathematics	Dr. Smith	Room A101
3	Charlie Davis	Chemistry	Dr. Williams	Room C303
3	Charlie Davis	Physics	Dr. Williams	Room C303
2	Bob Martin	History	Prof. Johnson	Room B202
2	Bob Martin	Political Science	Prof. Johnson	Room B202
4	Diana Clark	English Literature	Dr. Smith	Room A101
4	Diana Clark	Mathematics	Dr. Smith	Room A101

STUDENTID	STUDENTNAME	ADVISORNAME
1	Alice Brown	Dr. Smith
3	Charlie Davis	Dr. Williams
2	Bob Martin	Prof. Johnson
4	Diana Clark	Dr. Smith

ADVISORNAME	ADVISOROFFICE
Dr. Smith	Room A101
Prof. Johnson	Room B202
Dr. Williams	Room C303

STUDENTID	COURSE
1	Mathematics
1	Computer Science
2	History
2	Political Science
3	Physics
3	Chemistry
4	Mathematics
4	English Literature

RESULT

Thus the program has been executed successfully

EX: 6 Create a program to design an ER Model for Library Management System

Aim:

To create a library database and execute the SQL queries.

Procedure:

- Go to database Home page.
- Create a table with necessary entities.
- Insert values in the existing table.
- Execute the retrieve, selection, delete and view query on the appropriate table.
- Exit.

Entity-Relationship Diagram:

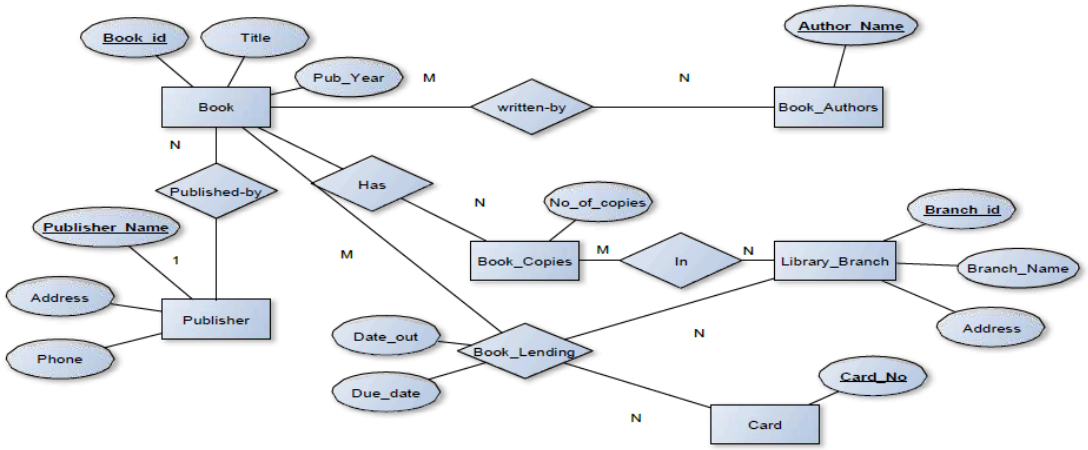

Table Creation:

CREATE TABLE PUBLISHER (NAME VARCHAR2 (20) PRIMARY KEY, PHONE INTEGER, ADDRESS VARCHAR2 (20));

CREATE TABLE BOOK(BOOK_ID INTEGER PRIMARY KEY, TITLE VARCHAR2 (20), PUB_YEAR VARCHAR2 (20), PUBLISHER_NAME REFERENCES PUBLISHER (NAME) ON DELETE CASCADE);

CREATE TABLEBOOK_AUTHORS (AUTHOR_NAME VARCHAR2(20), BOOK_ID REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE, PRIMARY KEY (BOOK_ID, AUTHOR_NAME));

CREATE TABLE LIBRARY_BRANCH (BRANCH_ID INTEGER PRIMARY KEY, BRANCH_NAME VARCHAR2 (50), ADDRESS VARCHAR2 (50));

CREATE TABLE BOOK_COPIES (NO_OF_COPIES INTEGER, BOOK_ID REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE,BRANCH_ID REFERENCES LIBRARY_BRANCH (BRANCH_ID) ON DELETE CASCADE,PRIMARY KEY (BOOK_ID, BRANCH_ID));

CREATE TABLE CARD (CARD_NO INTEGER PRIMARY KEY);

CREATE TABLE BOOK_LENDING (DATE_OUT DATE,DUE_DATE DATE,BOOK_ID REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE,BRANCH_ID REFERENCES LIBRARY_BRANCH (BRANCH_ID) ON DELETE CASCADE, CARD_NO REFERENCES CARD (CARD_NO) ON DELETE CASCADE, PRIMARY KEY (BOOK_ID, BRANCH_ID, CARD_NO));

Insertion of Values to Tables:

PUBLISHER TABLE:

INSERT INTO PUBLISHER VALUES (MCGRAW-HILL, 9989076587, BANGALORE);
INSERT INTO PUBLISHER VALUES (PEARSON, 9889076565, NEWDELHI);
INSERT INTO PUBLISHER VALUES (RANDOM HOUSE, 7455679345,HYDRABAD);
INSERT INTO PUBLISHER VALUES (HACHETTE LIVRE,8970862340, CHENAI);
INSERT INTO PUBLISHER VALUES (GRUPO PLANETA, 7756120238, BANGALORE);

BOOK TABLE:

INSERT INTO BOOK VALUES (1,'DBMS','JAN-2017', 'MCGRAW-HILL');
INSERT INTO BOOK VALUES (2,'ADBMS,'JUN-2016','MCGRAWHILL');
INSERT INTO BOOK VALUES (3,'CN','SEP-2016', 'PEARSON');
INSERT INTO BOOK VALUES (4,'CG','SEP-2015', 'GRUPOPLANETA');
INSERT INTO BOOK VALUES (5,'OS','MAY-2016','PEARSON');

BOOK_AUTHORS TABLE:

INSERT INTO BOOK_AUTHORS VALUES ('NAVATHE', 1);
INSERT INTO BOOK_AUTHORS VALUES ('NAVATHE', 2);
INSERT INTO BOOK_AUTHORS VALUES ('TANENBAUM', 3);
INSERT INTO BOOK_AUTHORS VALUES ('EDWARD ANGEL',4);
INSERT INTO BOOK_AUTHORS VALUES ('GALVIN', 5);

LIBRARY TABLE:

INSERT INTO LIBRARY_BRANCH VALUES (10,'RR NAGAR, 'BANGALORE');

INSERT INTO LIBRARY_BRANCH VALUES (11,'RNSIT, 'BANGALORE');
INSERT INTO LIBRARY_BRANCH VALUES (12,'RAJAJI NAGAR','BANGALORE');
INSERT INTO LIBRARY_BRANCH VALUES (13,'NITTE,'MANGALORE');
INSERT INTO LIBRARY_BRANCH VALUES (14,'MANIPAL', 'UDUPI');

BOOK_COPIES TABLE:

INSERT INTO BOOK_COPIES VALUES (10, 1, 10);
INSERT INTO BOOK_COPIES VALUES (5, 1, 11);
INSERT INTO BOOK_COPIES VALUES (2, 2, 12);
INSERT INTO BOOK_COPIES VALUES (5, 2, 13);
INSERT INTO BOOK_COPIES VALUES (7, 3, 14);
INSERT INTO BOOK_COPIES VALUES (1, 5, 10);
INSERT INTO BOOK_COPIES VALUES (3, 4, 11);

CARD TABLE:

INSERT INTO CARD VALUES (100);
INSERT INTO CARD VALUES (101);
INSERT INTO CARD VALUES (102);
INSERT INTO CARD VALUES (103);
INSERT INTO CARD VALUES (104);

INSERT INTO BOOK_LENDING VALUES ('01-JAN-17','01-JUN-17', 1, 10, 101);
INSERT INTO BOOK_LENDING VALUES ('11-JAN-17','11-MAR-17', 3, 14, 101);
INSERT INTO BOOK_LENDING VALUES ('21-FEB-17','21-APR-17', 2, 13, 101);
INSERT INTO BOOK_LENDING VALUES ('15-MAR-17','15-JUL-17', 4, 11, 101);
INSERT INTO BOOK_LENDING VALUES ('12-APR-17','12-MAY-17', 1, 11, 104);
SELECT * FROM PUBLISHER;

Result:

Thus the library databases with SQL queries were executed successfully.

Ex7 : Create a database for student management with constraints

Aim :

To write a program for creating a student management using constraints.

Procedure:

- Go to database Home page.
- Create a table for department and details.
- Insert values in the existing table.
- Execute the constraints used in the appropriate table.
- Exit.

Program:

```
CREATE TABLE Department ( deptid INT PRIMARY KEY ,deptname VARCHAR(100) NOT NULL UNIQUE, location VARCHAR(100) );

INSERT INTO Department  VALUES  (101,'Computer Science', 'Building A');

INSERT INTO Department VALUES(102,'Electrical Engineering', 'Building B');

INSERT INTO Department  VALUES(103,'Mechanical Engineering', 'Building C');

INSERT INTO Department  VALUES(104,'Business Administration', 'Building D');

select * from Department;

CREATE TABLE details( firstname VARCHAR2(50) NOT NULL, lastname VARCHAR2(50) NOT NULL,gender CHAR(1) CHECK (gender IN ('M','F')),dob DATE NOT NULL,email VARCHAR2(100) UNIQUE,phone VARCHAR2(15) UNIQUE,deptid NUMBER NOT NULL,admissiondate DATE DEFAULT SYSDATE,CONSTRAINT fkstudentdept FOREIGN KEY (deptid)REFERENCES Department(deptid)ON DELETE CASCADE );

INSERT INTO details VALUES ('Alice','Johnson','F',TO_DATE('2002-05-14'),'alice.johnson@example.com','9876543210',101,TO_DATE('2021-08-15'));
```

INSERT INTO details VALUES ('Bob', 'Smith', 'M', TO_DATE('2001-03-22', 'YYYY-MM-DD'), 'bob.smith@example.com', '9876543211', 102, TO_DATE('2020-08-10', 'YYYY-MM-DD'));

INSERT INTO details VALUES ('Charlie', 'Brown', 'M', TO_DATE('2003-07-10', 'YYYY-MM-DD'), 'charlie.brown@example.com', '9876543212', 103, TO_DATE('2022-08-20', 'YYYY-MM-DD'));

select * from details;

Output :

FIRSTNAME	LASTNAME	GENDER	DOB	EMAIL	PHONE	DEPTID	ADMISSIONDATE
Alice	Johnson	F	14-MAY-02	alice.johnson@example.com	9876543210	101	15-AUG-21
Bob	Smith	M	22-MAR-01	bob.smith@example.com	9876543211	102	10-AUG-20
Charlie	Brown	M	10-JUL-03	charlie.brown@example.com	9876543212	103	20-AUG-22

DEPTID	DEPTNAME	LOCATION
101	Computer Science	Building A
102	Electrical Engineering	Building B
103	Mechanical Engineering	Building C
104	Business Administration	Building D

Result

Thus the program has been executed successfully.

Ex 8 Create a program for INNER JOIN, LEFT JOIN and RIGHT JOIN to display the student details.

Aim :

To write a program for INNER JOIN, LEFT JOIN and RIGHT JOIN to display the student details.

Procedure:

- Go to database Home page.
- Create a table for faculty and learner.
- Insert values in the existing table.
- Execute the join queries
    - INNER JOIN: Show learners with a faculty.
    - LEFT JOIN: Show all learners, including those without a faculty.
    - RIGHT JOIN: Show all faculties, including those without learners.
    - FULL OUTER JOIN: Show all learners and all faculties.
- Exit.

Program :

```
CREATE TABLE Faculty (
   facultyid NUMBER PRIMARY KEY,
   facultyname VARCHAR2(100) NOT NULL
);
INSERT INTO Faculty (facultyid, facultyname) VALUES (201, 'Computer Science');
INSERT INTO Faculty (facultyid, facultyname) VALUES (202, 'Electrical Engineering');
INSERT INTO Faculty (facultyid, facultyname) VALUES (203, 'Mechanical Engineering');
INSERT INTO Faculty (facultyid, facultyname) VALUES (204, 'Business Administration');
```

```
SELECT * FROM FACULTY;
CREATE TABLE Learner (
   learnerid NUMBER PRIMARY KEY,
   firstname VARCHAR2(50) NOT NULL,
   lastname VARCHAR2(50) NOT NULL,
   gender CHAR(1) CHECK (gender IN ('M','F')),
   dob DATE NOT NULL,
   email VARCHAR2(100) UNIQUE,
   phone VARCHAR2(15) UNIQUE,
   facultyid NUMBER,
   admissiondate DATE DEFAULT SYSDATE,
   CONSTRAINT fk_learner_faculty FOREIGN KEY (facultyid)
      REFERENCES Faculty(facultyid)
      ON DELETE CASCADE
);
INSERT INTO Learner
VALUES (1, 'Alice', 'Johnson', 'F', TO_DATE('2002-05-14','YYYY-MM-DD'),
                                    'alice.johnson@example.com',      '9876543210',      201,
TO_DATE('2021-08-15','YYYY-MM-DD'));
INSERT INTO Learner
VALUES (2, 'Bob', 'Smith', 'M', TO_DATE('2001-03-22','YYYY-MM-DD'),
                                      'bob.smith@example.com',      '9876543211',      202,
TO_DATE('2020-08-10','YYYY-MM-DD'));
INSERT INTO Learner
VALUES (3, 'Charlie', 'Brown', 'M', TO_DATE('2003-07-10','YYYY-MM-DD'),
                                    'charlie.brown@example.com',      '9876543212',      201,
TO_DATE('2022-08-20','YYYY-MM-DD'));
INSERT INTO Learner
VALUES (4, 'Diana', 'Evans', 'F', TO_DATE('2002-01-05','YYYY-MM-DD'),
                                      'diana.evans@example.com',      '9876543213',      203,
TO_DATE('2021-08-15','YYYY-MM-DD'));
INSERT INTO Learner
VALUES (5, 'Ethan', 'Wright', 'M', TO_DATE('2000-09-12','YYYY-MM-DD'),
```

'ethan.wright@example.com', '9876543214', NULL, TO_DATE('2019-08-12','YYYY-MM-DD'));

COMMIT;

```
SELECT l.learnerid, l.firstname, l.lastname, f.facultyname
FROM Learner l
INNER JOIN Faculty f
ON l.facultyid = f.facultyid;
SELECT l.learnerid, l.firstname, l.lastname, f.facultyname
FROM Learner l
LEFT JOIN Faculty f
ON l.facultyid = f.facultyid;
SELECT l.learnerid, l.firstname, l.lastname, f.facultyname
FROM Learner l
RIGHT JOIN Faculty f
ON l.facultyid = f.facultyid;
SELECT
   l.learnerid,
   l.firstname,
   l.lastname,
   f.facultyid,
   f.facultyname
FROM Learner l
FULL OUTER JOIN Faculty f
ON l.facultyid = f.facultyid;
```

Output :

FACULTYID	FACULTYNAME
201	Computer Science
202	Electrical Engineering
203	Mechanical Engineering
204	Business Administration

LEARNERID	FIRSTNAME	LASTNAME	GENDER	DOB	EMAIL	PHONE	FACULTYID	ADMISSIONDATE
1	Alice	Johnson	F	14-MAY-02	alice.johnson@example.com	9876543210	201	15-AUG-21
2	Bob	Smith	M	22-MAR-01	bob.smith@example.com	9876543211	202	10-AUG-20
3	Charlie	Brown	M	10-JUL-03	charlie.brown@example.com	9876543212	201	20-AUG-22
4	Diana	Evans	F	05-JAN-02	diana.evans@example.com	9876543213	203	15-AUG-21
5	Ethan	Wright	M	12-SEP-00	ethan.wright@example.com	9876543214	-	12-AUG-19

Inner join

LEARNERID	FIRSTNAME	LASTNAME	FACULTYNAME
1	Alice	Johnson	Computer Science
2	Bob	Smith	Electrical Engineering
3	Charlie	Brown	Computer Science
4	Diana	Evans	Mechanical Engineering

Left join

LEARNERID	FIRSTNAME	LASTNAME	FACULTYNAME
3	Charlie	Brown	Computer Science
1	Alice	Johnson	Computer Science
2	Bob	Smith	Electrical Engineering
4	Diana	Evans	Mechanical Engineering
5	Ethan	Wright	-

Right join

LEARNERID	FIRSTNAME	LASTNAME	FACULTYNAME
1	Alice	Johnson	Computer Science
2	Bob	Smith	Electrical Engineering
3	Charlie	Brown	Computer Science
4	Diana	Evans	Mechanical Engineering
-	-	-	Business Administration

Full outer join

LEARNERID	FIRSTNAME	LASTNAME	FACULTYID	FACULTYNAME
3	Charlie	Brown	201	Computer Science
1	Alice	Johnson	201	Computer Science
2	Bob	Smith	202	Electrical Engineering
4	Diana	Evans	203	Mechanical Engineering
5	Ethan	Wright	-	-
-	-	-	204	Business Administration

Result :

Thus the program has been executed successfully.

Ex 9 Create a program to create a trigger for automatically updates the records.

Aim:

To create a trigger that automatically updates records when data is modified.

Procedure

- Open Oracle SQL Command
- Create `employee` table with columns: `emp_id (PK)`, `emp_name`, `salary`, `last_update`.
- Write and compile a `BEFORE UPDATE` row-level trigger `trg_update_date` on `employee`.
- In the trigger body set `:NEW.last_update := SYSDATE;`.
- Insert a sample employee row.
- Update the employee's salary.
- Select from `employee` to observe `last_update` change.

Program

```
CREATE TABLE employee (
   emp_id NUMBER PRIMARY KEY,
   emp_name VARCHAR2(50),
   salary NUMBER,
   last_update DATE
```

```
);
INSERT INTO employee VALUES (1, 'Alice', 30000, SYSDATE);
INSERT INTO employee VALUES (2, 'john', 20000, SYSDATE);
CREATE OR REPLACE TRIGGER trg_update_date
BEFORE UPDATE ON employee
FOR EACH ROW
BEGIN
   :NEW.last_update := SYSDATE;
END;
UPDATE employee SET salary = 35000 WHERE emp_id = 2;
SELECT * FROM employee;
```

Output :

EMP_ID	EMP_NAME	SALARY	LAST_UPDATE
1	Alice	35000	16-OCT-25
2	john	35000	16-OCT-25

Result

Thus the program has been executed successfully.

Ex 10 Create a program for banking system where funds transfer using BEGIN, COMMIT and ROLLBACK.

Aim

To perform a funds transfer between two accounts using transaction control and ensure atomicity with COMMIT/ROLLBACK.

Procedure

- Open Oracle SQL Command.
- Create accounts table with columns: acc_no (PK), holder_name, balance.
- Insert two sample accounts with initial balances and COMMIT.
- Start a PL/SQL block (BEGIN ... END;).
- In the block, subtract transfer amount from source account: UPDATE accounts SET balance = balance - amount WHERE acc_no = src.
- Add transfer amount to destination account: UPDATE accounts SET balance = balance + amount WHERE acc_no = dest.
- Check the source account balance; if negative, execute ROLLBACK, else COMMIT.
- Query accounts table to verify balances after the transaction.

Program

```
CREATE TABLE accounts (
   acc_no NUMBER PRIMARY KEY,
   holder_name VARCHAR2(50),
   balance NUMBER
);
```

```
INSERT INTO accounts VALUES (101, 'John', 5000);
INSERT INTO accounts VALUES (102, 'Mary', 4000);
BEGIN
   UPDATE accounts SET balance = balance - 1000 WHERE acc_no = 101;
   UPDATE accounts SET balance = balance + 1000 WHERE acc_no = 102
 IF (SELECT balance FROM accounts WHERE acc_no = 101) < 0 THEN
      ROLLBACK;
   ELSE
      COMMIT;
   END IF;
END;
SELECT * FROM accounts;
```

Output

ACC_NO	HOLDER_NAME	BALANCE
101	John	4000
102	Mary	5000

Result

Thus the program has been executed successfully.