Practical 1 **Date: 11/07/2023**

Aim: To Practice DDL commands.

Code: (Commands used are **CREATE, ALTER, DROP, TRUNCATE and RENAME)**

```
CREATE TABLE Employee (
   id INT PRIMARY KEY,
   name VARCHAR(50),
   age INT
);
```

OUTPUT:

Employee

id	name	age
empty		

```
ALTER TABLE Employee
ADD email VARCHAR(100);
```

OUTPUT:

Employee

id	name	age	email
empty			

```
DROP TABLE Employee;

TRUNCATE TABLE Employee;

RENAME TABLE Employees TO Employee29;
```

Practical 2 **Date: 18/07/2023**

Aim: To Practice DML commands.

Code: (Commands used are **CREATE TABLE , INSERT INTO, SELECT, WHERE, UPDATE, DELETE, COUNT, SUM** and **AVG)**

```
CREATE TABLE Students (
   StudentID int,
   FirstName varchar (100),
   LastName varchar(100),
   Marks int,
   Course varchar(100)
);
INSERT INTO Students (StudentID, Firstname, Lastname, Marks, Course)
VALUES ('1529', 'Barbie', 'Kumari', '100', 'BTech(CSE)');
INSERT INTO Students (StudentID, Firstname, Lastname, Marks, Course)
VALUES ('1530', 'Barbadi', 'Kumari', '100', 'BTech(CSE)');
INSERT INTO Students (StudentID, Firstname, Lastname, Marks, Course)
VALUES ('1531', 'Bushiya', 'Kumari', '100', 'BTech(CSE)');
INSERT INTO Students (StudentID, Firstname, Lastname, Marks, Course)
VALUES ('1532', 'Buchhi', 'Kumari', '100', 'BTech(CSE)');
INSERT INTO Students (StudentID, Firstname, Lastname, Marks, Course)
VALUES ('1533', 'Bebo', 'Kumari', '100', 'BTech(CSE)');
```

OUTPUT:

STUDENTID	FIRSTNAME	LASTNAME	MARKS	COURSE
1530	Barbadi	Kumari	100	BTech(CSE)
1532	Buchhi	Kumari	100	BTech(CSE)
1533	Bebo	Kumari	100	BTech(CSE)
1529	Barbie	Kumari	100	BTech(CSE)
1531	Bushiya	Kumari	100	BTech(CSE)

5 rows returned in 0.00 seconds CSV Export

```
SELECT StudentID, Firstname, Course FROM Students
WHERE StudentID = '1529';
```

OUTPUT:

STUDENTID	FIRSTNAME	COURSE
1529	Barbie	BTech(CSE)

1 rows returned in 0.00 seconds CSV Export

```
UPDATE Students SET Lastname= 'Jha' ,Marks = '20' WHERE StudentID= '1529';
SELECT * FROM Students;
```

OUTPUT:

STUDENTID	FIRSTNAME	LASTNAME	MARKS	COURSE
1530	Barbadi	Kumari	100	BTech(CSE)
1532	Buchhi	Kumari	100	BTech(CSE)
1533	Bebo	Kumari	100	BTech(CSE)
1529	Barbie	Jha	20	BTech(CSE)
1531	Bushiya	Kumari	100	BTech(CSE)

5 rows returned in 0.00 seconds CSV Export

DELETE FROM Students WHERE Firstname = 'Bebo';

OUTPUT:

STUDENTID	FIRSTNAME	LASTNAME	MARKS	COURSE
1530	Barbadi	Kumari	100	BTech(CSE)
1532	Buchhi	Kumari	100	BTech(CSE)
1529	Barbie	Kumari	100	BTech(CSE)
1531	Bushiya	Kumari	100	BTech(CSE)

4 rows returned in 0.00 seconds CSV Export

```
SELECT COUNT(Firstname)
FROM Students
WHERE Lastname = 'Kumari';
SELECT AVG(Marks)
FROM Students
WHERE Course = 'BTech(CSE)';
SELECT SUM(Marks)
FROM Students
WHERE Lastname = 'Kumari';
```

OUTPUT:

COUNT(FIRSTNAME)
4

1 rows returned in 0.02 seconds

AVG(MARKS)
100

1 rows returned in 0.01 seconds

SUM(MARKS)
400

1 rows returned in 0.00 seconds

Practical 3 **Date: 25/07/2023**

Aim: To Practice SQL constraints commands.

Code: (Commands used are **CREATE TABLE, CONSTRAINTS, NOT NULL, PRIMARY KEY, FOREIGN KEY, ALTER, DROP** and **INSERT INTO)**

- CREATE TABLE Persons (
  ID int NOT NULL,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255),
  Age int,
  UNIQUE (ID)
  );

OUTPUT: Object Type TABLE Object PERSONSS

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PERSONSS	ID	Number	-	-	0	-	-	-	-
	LASTNAME	Varchar2	255	-	-	-	-	-	-
	FIRSTNAME	Varchar2	255	-	-	-	-	-	-
	AGE	Number	-	-	0	-	✓	-	-

1 - 4

- INSERT INTO Person1
  VALUES('1529','Jha','',0);

OUTPUT:

ID	LASTNAME	FIRSTNAME	AGE
1529	jha	-	0

1 rows returned in 0.02 seconds CSV Export

- SELECT * FROM Person1;
- ALTER TABLE Person1
  ADD CONSTRAINT UC_Person1 UNIQUE (FirstName,LastName);
- ALTER TABLE Person1
  DROP CONSTRAINT UC_Person1;
- ALTER TABLE Persons
  ADD PRIMARY KEY (ID);

- CREATE TABLE Order1(
  OrderID int NOT NULL,
  OrderNumber int NOT NULL,
  ID int,
  PRIMARY KEY (OrderID),
  FOREIGN KEY (ID)
  REFERENCES Person1(ID)
  );
- INSERT INTO Order1
  VALUES('112', '23', '1529');

OUTPUT:

ORDERID	ORDERNUMBER	ID	ID	LASTNAME	FIRSTNAME	AGE
112	23	1529	1529	jha	-	0

1 rows returned in 0.32 seconds CSV Export

Practical 4 **Date: 01/08/2023**

Aim: To Practice different types of joins.

Code: (Left Join. Right Join, Full Join, Cross Join, Inner Join, Left Outer Join, Right Outer Join, Full Outer Join)

- CREATE TABLE Employee1660(
  EmployeeID int NOT NULL PRIMARY KEY,
  FirstName varchar(255) NOT NULL,
  LastName varchar(255),
  Salary int
  );

Object Type **TABLE** Object **EMPLOYEE1660**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMPLOYEE1660	EMPLOYEEID	Number	-	-	0	1	-	-	-
	FIRSTNAME	Varchar2	255	-	-	-	-	-	-
	LASTNAME	Varchar2	255	-	-	-	✓	-	-
	SALARY	Number	-	-	0	-	✓	-	-
									1 - 4

- CREATE TABLE Dept1660(
  Dept_ID int NOT NULL PRIMARY KEY,
  Dept_Name varchar(255) NOT NULL,
  EID int
  );

Object Type **TABLE** Object **DEPT1660**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPT1660	DEPT_ID	Number	-	-	0	1	-	-	-
	DEPT_NAME	Varchar2	255	-	-	-	-	-	-
	EID	Number	-	-	0	-	✓	-	-
									1 - 3

- INSERT INTO Employee1660(EMPLOYEEID, FirstName, LastName, Salary) VALUES(1, "Shambhavi","Mishra", "10000000");
  .....

EMPLOYEEID	FIRSTNAME	LASTNAME	SALARY
2	Jigyasa	Jha	9999999
1	Shambhavi	Mishra	10000000
3	Khushi	Sharma	9400000
4	Saumya	Singh	9700000
5	Pratyaksha	Sharma	9300000

- INSERT INTO Dept1660(Dept_Id, Dept_Name, EID)
  VALUES(32, 'Finance', 1);
  .....

DEPT_ID	DEPT_NAME	EID
62	Marketing	5
51	HR	3
32	Finance	1
27	Software	4
19	Management	2

❖ **Left Outer Join**

➢ SELECT e.EmployeeID, e.FirstName, e.LastName, e.Salary, d.Dept_ID, d.Dept_Name FROM Employee1660 e LEFT OUTER JOIN Dept1660 d ON e.EmployeeID = d.EID;

EMPLOYEEID	FIRSTNAME	LASTNAME	SALARY	DEPT_ID	DEPT_NAME
5	Pratyaksha	Sharma	9300000	62	Marketing
3	Khushi	Sharma	9400000	51	HR
1	Shambhavi	Mishra	10000000	32	Finance
4	Saumya	Singh	9700000	27	Software
2	Jigyasa	Jha	9999999	19	Management

❖ **Right Outer Join**

SELECT e.EmployeeID, e.FirstName, e.LastName, e.Salary, d.Dept_ID, d.Dept_Name FROM Employee1660 e RIGHT OUTER JOIN Dept1660 d ON e.EmployeeID = d.EID;

EMPLOYEEID	FIRSTNAME	LASTNAME	SALARY	DEPT_ID	DEPT_NAME
2	Jigyasa	Jha	9999999	19	Management
1	Shambhavi	Mishra	10000000	32	Finance
3	Khushi	Sharma	9400000	51	HR
4	Saumya	Singh	9700000	27	Software
5	Pratyaksha	Sharma	9300000	62	Marketing

❖ **Full Outer Join**

➢ SELECT e.EmployeeID, e.FirstName, e.LastName, e.Salary, d.Dept_ID, d.Dept_Name FROM Dept1660 d LEFT OUTER JOIN Employee1660 e ON e.EmployeeID = d.EID UNION SELECT e.EmployeeID, e.FirstName, e.LastName, e.Salary, d.Dept_ID, d.Dept_Name FROM Dept1660 d RIGHT OUTER JOIN Employee1660 e ON e.EmployeeID = d.EID;

EMPLOYEEID	FIRSTNAME	LASTNAME	SALARY	DEPT_ID	DEPT_NAME
1	Shambhavi	Mishra	10000000	32	Finance
2	Jigyasa	Jha	9999999	19	Management
3	Khushi	Sharma	9400000	51	HR
4	Saumya	Singh	9700000	27	Software
5	Pratyaksha	Sharma	9300000	62	Marketing

❖ **Inner Join**

➢ SELECT EmployeeID, FirstName, LastName, Salary
FROM Employee1660 e, Dept1660 d
WHERE e.EmployeeID =d.EID;

EMPLOYEEID	FIRSTNAME	LASTNAME	SALARY
5	Pratyaksha	Sharma	9300000
3	Khushi	Sharma	9400000
1	Shambhavi	Mishra	10000000
4	Saumya	Singh	9700000
2	Jigyasa	Jha	9999999

❖ **Cross Join**

➢ SELECT *FROM Employee1660 CROSS JOIN Dept1660;

EMPLOYEEID	FIRSTNAME	LASTNAME	SALARY	DEPT_ID	DEPT_NAME	EID
2	Jigyasa	Jha	9999999	62	Marketing	5
2	Jigyasa	Jha	9999999	51	HR	3
2	Jigyasa	Jha	9999999	32	Finance	1
2	Jigyasa	Jha	9999999	27	Software	4
2	Jigyasa	Jha	9999999	19	Management	2
1	Shambhavi	Mishra	10000000	62	Marketing	5
1	Shambhavi	Mishra	10000000	51	HR	3
1	Shambhavi	Mishra	10000000	32	Finance	1
1	Shambhavi	Mishra	10000000	27	Software	4
1	Shambhavi	Mishra	10000000	19	Management	2
More than 10 rows available. Increase rows selector to view more rows.						

❖ **Left Join**

➢ SELECT e.EmployeeID, e.FirstName, e.LastName, e.Salary, d.Dept_ID, d.Dept_Name FROM Employee1660 e LEFT JOIN Dept1660 d ON e.EmployeeID = d.EID;

EMPLOYEEID	FIRSTNAME	LASTNAME	SALARY	DEPT_ID	DEPT_NAME
5	Pratyaksha	Sharma	9300000	62	Marketing
3	Khushi	Sharma	9400000	51	HR
1	Shambhavi	Mishra	10000000	32	Finance
4	Saumya	Singh	9700000	27	Software
2	Jigyasa	Jha	9999999	19	Management

❖ **Right Join**

➢ SELECT e.EmployeeID, e.FirstName, e.LastName, e.Salary, d.Dept_ID, d.Dept_Name FROM Employee1660 e RIGHT JOIN Dept1660 d ON e.EmployeeID = d.EID;

EMPLOYEEID	FIRSTNAME	LASTNAME	SALARY	DEPT_ID	DEPT_NAME
2	Jigyasa	Jha	9999999	19	Management
1	Shambhavi	Mishra	10000000	32	Finance
3	Khushi	Sharma	9400000	51	HR
4	Saumya	Singh	9700000	27	Software
5	Pratyaksha	Sharma	9300000	62	Marketing

❖ **Full Join**

➢ SELECT *
FROM Employee1660 e
FULL OUTER JOIN Dept1660 d ON e.EmployeeID = d.EID;

EMPLOYEEID	FIRSTNAME	LASTNAME	SALARY	DEPT_ID	DEPT_NAME	EID
5	Pratyaksha	Sharma	9300000	62	Marketing	5
3	Khushi	Sharma	9400000	51	HR	3
1	Shambhavi	Mishra	10000000	32	Finance	1
4	Saumya	Singh	9700000	27	Software	4
2	Jigyasa	Jha	9999999	19	Management	2

EXPERIMENT -5

Aim: To complete **LAB ASSIGNMENT 1.**

Code: To Create the following five tables **(Course, Course_fee, Student, Installments, Course_Taken)**

```
--Create Course table--
CREATE TABLE Course (
course_no CHAR(4) PRIMARY KEY,
course_name VARCHAR(20)
);
--Insert Five Values into the Course Table—
INSERT INTO Course (course_no, course_name)
VALUES
('C001', 'Introduction to Programming'),
('C002', 'Database Management'),
('C003', 'Web Development'),
('C004', 'Data Structures'),
('C005', 'Machine Learning');
```

Output:

Course

course_no	course_name
C001	Oracle
C002	Database Management
C003	UNIX
C004	Data Structures
C005	Machine Learning

```
--Create Course_Fee table--
CREATE TABLE Course_fee (
course_no CHAR(4) PRIMARY KEY,
full_part CHAR(1) CHECK (full_part IN ('F', 'P')),
fees NUMBER(10),
CONSTRAINT fk_course_fee_course FOREIGN KEY (course_no) REFERENCES Course (course_no)
);
--Insert Five Values into the Course_Fee Table--
INSERT INTO Course_fee (course_no, full_part, fees)
VALUES
('C001', 'F', 1000),
('C002', 'P', 800),
('C003', 'F', 1200),
('C004', 'P', 900),
('C005', 'F', 1500);
```

Output:

Course_fee

course_no	full_part	fees
C001	F	1000
C002	P	800
C003	F	1200
C004	P	900
C005	F	1500

-- Create Student table--

```
CREATE TABLE Student (
prospectus_no NUMBER(10) PRIMARY KEY,
name VARCHAR(20),
address VARCHAR(30),
phone_no NUMBER(11),
D_O_B DATE,
total_amt NUMBER(10, 2),
amt_paid NUMBER(10, 2),
installment CHAR(1) CHECK (installment IN ('I', 'F'))
);
```

-- Insert into Student table--

```
INSERT INTO Student (prospectus_no, name, address, phone_no, D_O_B, total_amt, amt_paid, installment)
VALUES
(1001, 'JIggi Fegusa', '125 Amity University', 1234567890, '2000-05-15', 1500.00, 500.00, 'I'),
(1002, 'Jannat Jeez', '71 Noida', 9876543210, '1999-08-22', 1800.00, 1000.00, 'F'),
(1003, 'Viraj Sharma', 'Central Park', 5556667777, '2001-03-10', 1200.00, 800.00, 'I'),
(1004, 'Shambhavi Mishra', '567 Noida City', 4443332222, '2002-01-20', 1000.00, 200.00, 'I'),
(1005, 'David Wiley', '890 Maple St', 1112223333, '2000-12-05', 1400.00, 600.00, 'F');
```

Output:

Student

prospectus_no	name	address	phone_no	D_O_B	total_amt	amt_paid	installment
1001	JIggi Fegusa	125 Amity University	1234567890	2000-05-15	1500	500	I
1002	Jannat Jeez	71 Noida	9876543210	1999-08-22	1800	1000	F
1003	Viraj Sharma	Central Park	5556667777	2001-03-10	1200	800	I
1004	Shambhavi Mishra	567 Noida City	4443332222	2002-01-20	1000	200	I
1005	Alfredo Singh	890 Maple St	1112223333	2000-12-05	1400	1400	F

-- Create Installment table--

```
CREATE TABLE Installment (
prospectus_no NUMBER(10),
installment_amt NUMBER(10, 2),
due_dt DATE,
paid CHAR(1) CHECK (paid IN ('P', 'U')),
PRIMARY KEY (prospectus_no, due_dt),
CONSTRAINT fk_installment_student FOREIGN KEY (prospectus_no) REFERENCES Student
(prospectus_no) ON DELETE CASCADE
);
```

-- Insert into Installment table--

```
INSERT INTO Installment (prospectus_no, installment_amt, due_dt, paid)
VALUES
(1001, 200.00, '2023-08-20', 'U'),
(1002, 400.00, '2023-08-18', 'P'),
(1003, 100.00, '2023-08-25', 'U'),
(1004, 300.00, '2023-08-22', 'U'),
(1005, 200.00, '2023-08-30', 'P');
```

Output:

Installment

prospectus_no	installment_amt	due_dt	paid
1001	200	2023-08-20	U
1002	400	2023-08-18	P
1003	100	2023-08-25	U
1004	300	2023-08-22	U
1005	200	2023-08-30	P

-- Create Course_taken table--

```
CREATE TABLE Course_taken (
prospectus_no NUMBER(10),
course_no CHAR(4),
start_dt DATE,
full_part CHAR(1) CHECK (full_part IN ('F', 'P')),
time_slot CHAR(2),
performance VARCHAR(20),
CONSTRAINT fk_course_taken_student FOREIGN KEY (prospectus_no) REFERENCES Student
(prospectus_no),
CONSTRAINT fk_course_taken_course FOREIGN KEY (course_no) REFERENCES Course (course_no)
 );
```

-- Insert into Course_taken table--

```
INSERT INTO Course_taken (prospectus_no, course_no, start_dt, full_part, time_slot, performance)
VALUES
(1001, 'C001', '2023-08-15', 'F', 'M1', 'Good'),
(1002, 'C003', '2023-08-16', 'P', 'E1', 'Excellent'),
(1003, 'C002', '2023-08-14', 'F', 'A2', 'Average'),
(1004, 'C004', '2023-08-17', 'P', 'M2', 'Good'),
(1005, 'C005', '2023-08-13', 'F', 'E2', 'Very Good');
```

Output:

Course_taken

prospectus_no	course_no	start_dt	full_part	time_slot	performance
1001	C001	2023-08-15	F	M1	Good
1002	C003	2023-08-16	P	E1	Excellent
1003	C002	2023-08-14	F	A2	Average
1004	C004	2023-08-17	P	M2	Good
1005	C005	2023-08-13	F	E2	Very Good

Write the following SQL Queries:

1.) Retrieve name and course no of all the students.

- SELECT s.name, ct.course_no
  FROM Student s
  JOIN Course_taken ct ON s.prospectus_no = ct.prospectus_no;

name	course_no
JIggi Fegusa	C001
Jannat Jeez	C003
Viraj Sharma	C002
Shambhavi Mishra	C004
David Wiley	C005

2.) List the names of students who have paid the full amount at the time of admission.

- SELECT name
  FROM Student
  WHERE total_amt = amt_paid;

name
Alfredo Singh

3.) Find the names of students starting with A.

- SELECT name
  FROM Student
  WHERE name LIKE 'A%';

name
Alfredo Singh

4.) Print the names of students whose total amount is not equal to the amount due.

- SELECT name
  FROM Student
  WHERE total_amt <> amt_paid;

name
JIggi Fegusa
Jannat Jeez
Viraj Sharma
Shambhavi Mishra

5.) Count the number of students who have joined in the current year, current month.

- SELECT COUNT(*)
  FROM Student
  WHERE EXTRACT(YEAR FROM D_O_B) = EXTRACT(YEAR FROM CURRENT_DATE)
  AND EXTRACT(MONTH FROM D_O_B) = EXTRACT(MONTH FROM CURRENT_DATE);

6.) Determine the maximum and minimum course fees.

- SELECT MAX(fees) AS max_fee, MIN(fees) AS min_fee
  FROM Course_fee;

max_fee	min_fee
1500	800

7.) Increase the fee of oracle by 50%.

- UPDATE Course_fee
  SET fees = fees * 1.5
  WHERE course_no = 'C001';

8.) Print the details of courses whose fees are between 5000 and 10000.

- SELECT *
  FROM Course_fee
  WHERE fees BETWEEN 5000 AND 10000;

9.) Display the admission date in Date, Month, Year format.

- SELECT TO_CHAR(D_O_B, 'DD MONTH YYYY') AS admission_date
  FROM Student;

10.) Find out in which course the maximum number of students have taken admission.

```
SELECT course_no
FROM (
SELECT course_no, RANK() OVER (ORDER BY COUNT(*) DESC) AS rank
FROM Course_taken
GROUP BY course_no
) ranked
WHERE rank = 1;
```

EXPERIMENT-6

Aim: To complete **LAB ASSIGNMENT 2.**

Code: To Create the following five tables **(SUPPLIER, PART, PROJECTS and SPJ)**

```
-- Create SUPPLIER table --
  CREATE TABLE SUPPLIER (
  SNO CHAR(4) PRIMARY KEY,
  SNAME VARCHAR(50),
  STATUS INT,
  CITY VARCHAR(50)
  );
 --Insert Five Values into the SUPPLIER Table --
  INSERT INTO SUPPLIER (SNO, SNAME, STATUS, CITY)
  VALUES
  ('S1', 'Supplier A', 1, 'New York'),
  ('S2', 'Supplier B', 2, 'Los Angeles'),
  ('S3', 'Supplier C', 3, 'Chicago'),
  ('S4', 'Supplier D', 1, 'Houston'),
  ('S5', 'Supplier E', 2, 'San Francisco');
```

OUTPUT:

SUPPLIER

SNO	SNAME	STATUS	CITY
S1	Supplier A	1	New York
S2	Supplier B	2	Los Angeles
S3	Supplier C	3	Chicago
S4	Supplier D	1	Houston
S5	Supplier E	2	San Francisco

```
--Create PARTS table --
  CREATE TABLE PARTS (
  PNO CHAR(4) PRIMARY KEY,
  PNAME VARCHAR(50),
  COLOR VARCHAR(20),
  WEIGHT DECIMAL(10, 2),
  CITY VARCHAR(50)
  );
--Insert data into PARTS table --
  INSERT INTO PARTS (PNO, PNAME, COLOR, WEIGHT, CITY)
  VALUES
  ('P1', 'Part X', 'Red', 10.5, 'New York'),
  ('P2', 'Part Y', 'Blue', 5.2, 'Los Angeles'),
  ('P3', 'Part Z', 'Green', 8.7, 'Chicago'),
  ('P4', 'Part W', 'Red', 12.3, 'Houston'),
  ('P5', 'Part V', 'Yellow', 7.0, 'San Francisco');
```

OUTPUT: PARTS

PNO	PNAME	COLOR	WEIGHT	CITY
P1	Part X	Red	10.5	New York
P2	Part Y	Blue	5.2	Los Angeles
P3	Part Z	Green	8.7	Chicago
P4	Part W	Red	12.3	Houston
P5	Part V	Yellow	7	San Francisco

```
--Create PROJECT table--
 CREATE TABLE PROJECT (
 JNO CHAR(4) PRIMARY KEY,
 JNAME VARCHAR(50),
 CITY VARCHAR(50)
 );
```

-- Insert data into PROJECT table --

```
INSERT INTO PROJECT (JNO, JNAME, CITY)
VALUES
('J1', 'Project Alpha', 'New York'),
('J2', 'Project Beta', 'Los Angeles'),
('J3', 'Project Gamma', 'Chicago'),
('J4', 'Project Delta', 'Houston'),
('J5', 'Project Epsilon', 'San Francisco');
```

OUTPUT: PROJECT

JNO	JNAME	CITY
J1	Project Alpha	New York
J2	Project Beta	Los Angeles
J3	Project Gamma	Chicago
J4	Project Delta	Houston
J5	Project Epsilon	San Francisco

-- Create SPJ table --

```
CREATE TABLE SPJ (
SNO CHAR(4),
PNO CHAR(4),
JNO  CHAR(4),
QTY INT,
PRIMARY KEY (SNO, PNO, JNO),
FOREIGN KEY (SNO) REFERENCES SUPPLIER(SNO),
FOREIGN KEY (PNO) REFERENCES PARTS(PNO),
FOREIGN KEY (JNO) REFERENCES PROJECT(JNO)
);
```

-- Insert data into SPJ table --

```
INSERT INTO SPJ (SNO, PNO, JNO, QTY)
VALUES
('S1', 'P1', 'J1', 100),
('S1', 'P2', 'J2', 200),
```

('S2', 'P3', 'J2', 150),
('S2', 'P4', 'J3', 300),
('S3', 'P1', 'J3', 75),
('S3', 'P3', 'J3', 50),
('S4', 'P5', 'J4', 250),
('S5', 'P2', 'J5', 100),
('S5', 'P4', 'J5', 50);

OUTPUT:

SPJ

SNO	PNO	JNO	QTY
S1	P1	J1	100
S1	P2	J2	200
S2	P3	J2	150
S2	P4	J3	300
S3	P1	J3	75
S3	P3	J3	50
S4	P5	J4	250
S5	P2	J5	100
S5	P4	J5	50

Write the following SQL Queries:

1.) Get sno values for suppliers who supply project j1.

- SELECT SNO
  FROM SPJ
  WHERE JNO = 'J1';

SNO
S1

2.) Get sno values for suppliers who supply project j1 with part p1.

- SELECT SNO
  FROM SPJ
  WHERE JNO = 'J1' AND PNO = 'P1';

SNO
S1

3.) Get JNAME values for projects supplied by supplier S1.

- SELECT DISTINCT PROJECT.JNAME
  FROM PROJECT
  JOIN SPJ ON PROJECT.JNO = SPJ.JNO
  WHERE SPJ.SNO = 'S1';

JNAME
Project Alpha
Project Beta

4.) Get COLOR values for parts supplied by supplier S1.

- SELECT DISTINCT PARTS.COLOR
  FROM PARTS
  JOIN SPJ ON PARTS.PNO = SPJ.PNO
  WHERE SPJ.SNO = 'S1';

COLOR
Red
Blue

5.) Get PNO values for parts supplied to any project in New York.

- SELECT DISTINCT SPJ.PNO
  FROM SPJ
  JOIN PROJECT ON SPJ.JNO = PROJECT.JNO
  WHERE PROJECT.CITY = 'New York';

PNO
P1

6.) Get SNO values for suppliers who supply project J1 with a red part.

- SELECT DISTINCT SPJ.SNO
  FROM SPJ
  JOIN PARTS ON SPJ.PNO = PARTS.PNO
  WHERE SPJ.JNO = 'J1' AND PARTS.COLOR = 'Red';

SNO
S1

7.) Get SNO values for suppliers who supply a New York or Los Angeles project with a red part.

- SELECT DISTINCT SPJ.SNO
  FROM SPJ
  JOIN PARTS ON SPJ.PNO = PARTS.PNO
  JOIN PROJECT ON SPJ.JNO = PROJECT.JNO
  WHERE PARTS.COLOR = 'Red' AND (PROJECT.CITY = 'New York' OR PROJECT.CITY = 'Los Angeles');

SNO
S1

8.) Get PNO values for parts supplied to any project by a supplier in the same city.

- SELECT DISTINCT SPJ.PNO
  FROM SPJ
  JOIN SUPPLIER ON SPJ.SNO = SUPPLIER.SNO
  JOIN PROJECT ON SPJ.JNO = PROJECT.JNO
  WHERE SUPPLIER.CITY = PROJECT.CITY;

PNO
P1
P3
P5
P2
P4

9.) Get PNO values for parts supplied to any project in New York by a supplier in New York.

- SELECT DISTINCT SPJ.PNO
  FROM SPJ
  JOIN SUPPLIER ON SPJ.SNO = SUPPLIER.SNO
  JOIN PROJECT ON SPJ.JNO = PROJECT.JNO
  WHERE SUPPLIER.CITY = 'New York' AND PROJECT.CITY = 'New York';

PNO
P1

10.) Get JNO values for projects supplied by at least one supplier not in the same city.

- SELECT DISTINCT SPJ.JNO
  FROM SPJ
  JOIN SUPPLIER ON SPJ.SNO = SUPPLIER.SNO
  JOIN PROJECT ON SPJ.JNO = PROJECT.JNO
  WHERE SUPPLIER.CITY <> PROJECT.CITY;

JNO
J2
J3

11.) Get all pairs of CITY values such that a supplier in the first CITY supplies a project in the second CITY.

- SELECT DISTINCT SUPPLIER.CITY AS CITY1, PROJECT.CITY AS CITY2
  FROM SPJ
  JOIN SUPPLIER ON SPJ.SNO = SUPPLIER.SNO
  JOIN PROJECT ON SPJ.JNO = PROJECT.JNO;

CITY1	CITY2
New York	New York
New York	Los Angeles
Los Angeles	Los Angeles
Los Angeles	Chicago
Chicago	Chicago
Houston	Houston
San Francisco	San Francisco

12.) Get SNO values for suppliers who supply the same part to all projects.

- SELECT DISTINCT SPJ.SNO
  FROM SPJ
  WHERE SPJ.PNO IN (
  SELECT PNO
  FROM SPJ
  GROUP BY PNO
  HAVING COUNT(DISTINCT JNO) = (SELECT COUNT(*) FROM PROJECT)
  );

13.) Get PNO values for parts supplied to all projects in New York.

- SELECT DISTINCT SPJ.PNO
  FROM SPJ
  JOIN PROJECT ON SPJ.JNO = PROJECT.JNO

```
WHERE PROJECT.CITY = 'New York'
AND SPJ.PNO NOT IN (
   SELECT PNO
   FROM SPJ
   JOIN PROJECT ON SPJ.JNO = PROJECT.JNO
   WHERE PROJECT.CITY != 'New York'
);
```

14.) Get SNAME values for suppliers who supply at least one red part to any project.

- ```
  SELECT DISTINCT SUPPLIER.SNAME
  FROM SPJ
  JOIN SUPPLIER ON SPJ.SNO = SUPPLIER.SNO
  JOIN PARTS ON SPJ.PNO = PARTS.PNO
  WHERE PARTS.COLOR = 'Red';
  ```

SNAME
Supplier A
Supplier B
Supplier C
Supplier E

15.) Get total quantity of part P1 supplied by supplier S1.

- ```
  SELECT SUM(QTY) AS total_quantity
  FROM SPJ
  WHERE SNO = 'S1' AND PNO = 'P1';
  ```

total_quantity
100

16.) Get the total number of projects supplied by supplier S3.

- ```
  SELECT COUNT(DISTINCT JNO) AS total_projects
  FROM SPJ
  WHERE SNO = 'S3';
  ```

total_projects
1

17.) Change COLOR of all red parts to orange.

- ```
  UPDATE PARTS
  SET COLOR = 'Orange'
  WHERE COLOR = 'Red';
  ```

18.) Get SNAME values for suppliers who supply to both projects J1 and J2.

- ```
  SELECT DISTINCT SUPPLIER.SNAME
  FROM SPJ
  JOIN SUPPLIER ON SPJ.SNO = SUPPLIER.SNO
  WHERE JNO = 'J1'
  AND SUPPLIER.SNO IN (
     SELECT SNO
     FROM SPJ
     WHERE JNO = 'J2'
  ```
  ```

);

SNAME
Supplier A

19.)Get all CITY, PNO, CITY triples such that a supplier in the first CITY supplies the specified part to a project in the second CITY.

- SELECT DISTINCT SUPPLIER.CITY AS CITY1, SPJ.PNO, PROJECT.CITY AS CITY2
  FROM SPJ
  JOIN SUPPLIER ON SPJ.SNO = SUPPLIER.SNO
  JOIN PROJECT ON SPJ.JNO = PROJECT.JNO;

CITY1	PNO	CITY2
New York	P1	New York
New York	P2	Los Angeles
Los Angeles	P3	Los Angeles
Los Angeles	P4	Chicago
Chicago	P1	Chicago
Chicago	P3	Chicago
Houston	P5	Houston
San Francisco	P2	San Francisco
San Francisco	P4	San Francisco

20.) Get JNAMEs for projects which are supplied by supplier XYZ.

- SELECT DISTINCT PROJECT.JNAME
  FROM SPJ
  JOIN PROJECT ON SPJ.JNO = PROJECT.JNO
  JOIN SUPPLIER ON SPJ.SNO = SUPPLIER.SNO
  WHERE SUPPLIER.SNAME = 'XYZ';

EXPERIMENT -7

Aim: To complete **LAB ASSIGNMENT 3.**

Code: To Create the following two tables **(Employee and department)**

Create DEPARTMENT table

```
CREATE TABLE Departments (
DepartmentID INT PRIMARY KEY,
DepartmentName VARCHAR(255)
);
```

Insert Values into the DEPARTMENT Table

```
INSERT INTO Departments (DepartmentID, DepartmentName)
VALUES
(1, 'Systems'),
(2, 'Marketing'),
(3, 'Sales');
```

OUTPUT:

Departments

DepartmentID	DepartmentName
1	Systems
2	Marketing
3	Sales

Create EMPLOYEE table

```
CREATE TABLE Employees (
EmployeeID INT PRIMARY KEY,
EmployeeName VARCHAR(255),
DepartmentID INT,
HireDate DATE,
Salary DECIMAL(10, 2),
JobType VARCHAR(50),
ManagerID INT
);
```

Insert data into EMPLOYEE table

```
INSERT INTO Employees (EmployeeID, EmployeeName, DepartmentID, HireDate, Salary, JobType, ManagerID)
VALUES
(1, 'Apple singh', 1, '2020-01-15', 55000, 'Manager', NULL),
(2, 'Jamun jha', 1, '2021-03-10', 48000, 'Engineer', 1),
(3, 'Pineapple mishra', 1, '2022-05-20', 52000, 'Engineer', 1),
(4, 'Banana wilson', 2, '2019-11-02', 60000, 'Manager', NULL),
(5, 'Mango kaur', 2, '2020-07-18', 55000, 'Marketing Specialist', 4),
(6, 'Peach dhawan', 2, '2021-09-05', 48000, 'Marketing Specialist', 4),
(7, 'orange roy', 3, '2022-01-30', 62000, 'Salesperson', NULL),
(8, 'raspberry pie', 3, '2022-04-15', 58000, 'Salesperson', NULL);
```

OUTPUT:

Employees

DepartmentID	HireDate	Salary	JobType	Salary	JobType	ManagerID
1	2020-01-15	55000	Manager	55000	Manager	
1	2021-03-10	48000	Engineer	48000	Engineer	1
1	2022-05-20	52000	Engineer	52000	Engineer	1
2	2019-11-02	60000	Manager	60000	Manager	
2	2020-07-18	55000	Marketing Specialist	55000	Marketing Specialist	4
2	2021-09-05	48000	Marketing Specialist	48000	Marketing Specialist	4
3	2022-01-30	62000	Salesperson	62000	Salesperson	
3	2022-04-15	58000	Salesperson	58000	Salesperson	

Write the following SQL Queries:

1.) Display each employee name and hiredate of the Systems department.

- SELECT EmployeeName, HireDate
  FROM Employees
  WHERE DepartmentID = 1;SELECT SNO
  FROM SPJ
  WHERE JNO = 'J1';

EmployeeName	HireDate
Apple singh	2020-01-15
Jamun jha	2021-03-10
Pineapple mishra	2022-05-20

2.) Calculate length of service of each employee

- SELECT EmployeeName, DATEDIFF(CURDATE(), HireDate) AS LengthOfService
  FROM Employees;

3.) Find the second maximum salary of all employee

- SELECT MAX(Salary) AS SecondMaxSalary
  FROM Employees
  WHERE Salary < (SELECT MAX(Salary) FROM Employees);

SecondMaxSalary
60000

4.) Display all employee names and department names in department name order.

- SELECT E.EmployeeName, D.DepartmentName
  FROM Employees E
  JOIN Departments D ON E.DepartmentID = D.DepartmentID
  ORDER BY D.DepartmentName;

EmployeeName	DepartmentName
Banana wilson	Marketing
Mango kaur	Marketing
Peach dhawan	Marketing
orange roy	Sales
raspberry pie	Sales
Apple singh	Systems
Jamun jha	Systems
Pineapple mishra	Systems

5.) Find the name of the lowest paid employee for each manager.

- SELECT M.EmployeeName AS ManagerName, E.EmployeeName AS LowestPaidEmployee
  FROM Employees E
  LEFT JOIN Employees M ON E.ManagerID = M.EmployeeID
  WHERE E.Salary = (
  SELECT MIN(Salary)
  FROM Employees
  WHERE ManagerID = E.ManagerID
  );

ManagerName	LowestPaidEmployee
Apple singh	Jamun jha
Banana wilson	Peach dhawan

6.) Display the department that has no employee.

- SELECT DepartmentName
  FROM Departments
  WHERE DepartmentID NOT IN (SELECT DISTINCT DepartmentID FROM Employees);

7.) Find the employees who earn the maximum salary in each job type. Sort in descending order of salary.

- SELECT JobType, EmployeeName, Salary
  FROM Employees E1
  WHERE Salary = (
  SELECT MAX(Salary)
  FROM Employees E2
  WHERE E1.JobType = E2.JobType
)
  ORDER BY Salary DESC, JobType;

JobType	EmployeeName	Salary
Salesperson	orange roy	62000
Manager	Banana wilson	60000
Marketing Specialist	Mango kaur	55000
Engineer	Pineapple mishra	52000

8.) In which year did most people join the company? Display the year and number of employees.

- SELECT YEAR(HireDate) AS JoinYear, COUNT(*) AS NumberOfEmployees

```
FROM Employees
GROUP BY JoinYear
ORDER BY NumberOfEmployees DESC
LIMIT 1;
```

9.) Display the details of those employees who earn greater than average of their department.

- ```
  SELECT E.EmployeeName, E.DepartmentID, E.Salary
  FROM Employees E
  JOIN (
  SELECT DepartmentID, AVG(Salary) AS AvgSalary
  FROM Employees
  GROUP BY DepartmentID
  ) AS AvgSalaries ON E.DepartmentID = AvgSalaries.DepartmentID
  WHERE E.Salary > AvgSalaries.AvgSalary;
  ```

EmployeeName	DepartmentID	Salary
Apple singh	1	55000
Pineapple mishra	1	52000
Banana wilson	2	60000
Mango kaur	2	55000
orange roy	3	62000

10.) List the employees having salary between 10000 and 20000.

- ```
  SELECT EmployeeName, Salary
  FROM Employees
  WHERE Salary BETWEEN 10000 AND 20000;
  ```

11.)Display all employees hired during 1983 and who earn greater than average of their department

- ```
  SELECT E.EmployeeName, E.HireDate, E.Salary
  FROM Employees E
  WHERE YEAR(HireDate) = 1983
  AND E.Salary > (
  SELECT AVG(Salary)
  FROM Employees
  WHERE DepartmentID = E.DepartmentID
  );
  ```

12.) Update the salaries of all employees in marketing department & hike it by 15%.

- ```
  UPDATE Employees
  SET Salary = Salary * 1.15
  WHERE DepartmentID = 2;
  ```

13.) Get the gross salaries of all the employees.

- ```
  SELECT EmployeeName, Salary, Salary * 12 AS GrossSalary
  FROM Employees;
  ```

EmployeeName	Salary	GrossSalary
Apple singh	55000	660000
Jamun jha	48000	576000
Pineapple mishra	52000	624000
Banana wilson	69000	828000
  ```

14.) Get the names of employees and their managers' names.

- SELECT E.EmployeeName, M.EmployeeName AS ManagerName
  FROM Employees E
  LEFT JOIN Employees M ON E.ManagerID = M.EmployeeID;

EmployeeName	ManagerName
Apple singh	
Jamun jha	Apple singh
Pineapple mishra	Apple singh
Banana wilson	

15.) Display the name, location, and department name of all the employees earning more than 1500.

- SELECT E.EmployeeName, D.DepartmentName, E.Salary
  FROM Employees E
  JOIN Departments D ON E.DepartmentID = D.DepartmentID
  WHERE E.Salary > 1500;

EmployeeName	DepartmentName	Salary
Apple singh	Systems	55000
Jamun jha	Systems	48000
Pineapple mishra	Systems	52000
Banana wilson	Marketing	69000
Manao kaur	Marketing	63249.99999999999

16.) Show all the employees in Dallas.

- SELECT EmployeeName
  FROM Employees E
  JOIN Departments D ON E.DepartmentID = D.DepartmentID
  WHERE D.DepartmentName = 'Dallas';

17.) List the employees' name, job, salary, grade, and department for employees in the company except clerks. Sort on employee names.

- SELECT EmployeeName, JobType, Salary, Grade, DepartmentName
  FROM Employees E
  JOIN Departments D ON E.DepartmentID = D.DepartmentID
  WHERE E.JobType != 'Clerk'
  ORDER BY EmployeeName;

18.) Find the employees who earn the minimum salary for their job. Sort in descending order of salary.

- SELECT JobType, EmployeeName, Salary
  FROM Employees E1
  WHERE Salary = (
  SELECT MIN(Salary)
  FROM Employees E2
  WHERE E1.JobType = E2.JobType
)
  ORDER BY Salary DESC, JobType;

JobType	EmployeeName	Salary
Salesperson	raspberry pie	58000
Marketing Specialist	Peach dhawan	55199.99999999999
Manager	Apple singh	55000
Engineer	Jamun jha	48000

19.) Find the most recently hired employees in department order by hire date.

- SELECT EmployeeName, DepartmentName, HireDate
  FROM Employees E
  JOIN Departments D ON E.DepartmentID = D.DepartmentID
  ORDER BY DepartmentName, HireDate DESC;

EmployeeName	DepartmentName	HireDate
Peach dhawan	Marketing	2021-09-05
Mango kaur	Marketing	2020-07-18
Banana wilson	Marketing	2019-11-02
raspberry pie	Sales	2022-04-15
orange roy	Sales	2022-01-30

20.) Find out the difference between the highest and lowest salaries.

- SELECT MAX(Salary) - MIN(Salary) AS SalaryDifference
  FROM Employees;

SalaryDifference
21000

Practical 8 **Date: 03/10/2023**

Aim: To Practice View Command.

Code: (Create, Update and Delete)

- CREATE TABLE Employee1660(
  EmployeeID int NOT NULL PRIMARY KEY,
  FirstName varchar(255) NOT NULL,
  LastName varchar(255),
  Salary int
  );

Employee1660

EmployeeID	FirstName	LastName	Salary
1	Shambhavi	Mishra	10000000
2	Jigyasa	Jha	9900000
3	Khushi	Sharma	9700000
4	Saumya	Singh	9500000
5	Pratyaksha	Sharma	9300000

- CREATE TABLE Dept1660(
  Dept_ID int NOT NULL PRIMARY KEY,
  Dept_Name varchar(255) NOT NULL,
  EID int
  );

Dept1660

Dept_ID	Dept_Name	EmployeeID
32	Finance	1
62	Marketing	5
51	HR	3
27	Software	4
19	Management	2

❖ **Creating a View**

▪ **From a Single Table**

```
CREATE VIEW details AS
SELECT EmployeeID, FirstName
FROM Employee1660
WHERE Salary >= '9100000';

SELECT *
FROM details;
```

EmployeeID	FirstName
1	Shambhavi
2	Jigyasa
3	Khushi
4	Saumya
5	Pratyaksha

- **From Multiple Tables**

```
CREATE VIEW details2 AS
SELECT Employee1660.FirstName, Employee1660.Salary, Dept1660.Dept_Name
FROM Employee1660, Dept1660
WHERE Employee1660.EmployeeID = Dept1660.EmployeeID;

SELECT *
FROM details2;
```

FirstName	Salary	Dept_Name
Shambhavi	10000000	Finance
Pratyaksha	9300000	Marketing
Khushi	9700000	HR
Saumya	9500000	Software
Jigyasa	9900000	Management

❖ Updating a View

```
CREATE OR REPLACE VIEW details AS
SELECT Employee1660.FirstName, Employee1660.Salary, Dept1660.Dept_Name, Dept1660.Dept_ID
FROM Employee1660, Dept1660
WHERE Employee1660.EmployeeID = Dept1660.EmployeeID;
```

FirstName	Salary	Dept_Name	Dept_ID
Shambhavi	10000000	Finance	32
Pratyaksha	9300000	Marketing	62
Khushi	9700000	HR	51
Saumya	9500000	Software	27
Jigyasa	9900000	Management	19

❖ Deleting a View

```
DROP VIEW details;
```

Output

SQL query successfully executed. However, the result set is empty.

Practical 9 **Date: 03/10/2023**

Aim: To Practice PL/SQL Commands.

Code: Basics: Syntax, Comments, Variable Attributes, Conditionals: IF-THEN-ELSE, Case, Loops – For, While

- **Syntax**

```
DECLARE
  message  varchar2(20):= 'Hello, World! From Shambhavi and Jigyasa';
BEGIN
  dbms_output.put_line(message);
END;
/
```

```
Hello World From Shambhavi and Jigyasa

PL/SQL procedure successfully completed.
```

- **Comments**

```
DECLARE
  -- variable declaration
  message  varchar2(20):= 'Hello, World!';
BEGIN
  /*
  *  PL/SQL executable statement(s)
  */
  dbms_output.put_line(message);
END;
/
```

```
Hello World

PL/SQL procedure successfully completed.
```

- **Example**

```
DECLARE
  a integer := 30;
  b integer := 40;
  c integer;
  f real;
BEGIN
  c := a + b;
  dbms_output.put_line('Value of c: ' || c);
  f := 100.0/3.0;
  dbms_output.put_line('Value of f: ' || f);
END;
```

```
Value of c: 70
Value of f: 33.333333333333333333

PL/SQL procedure successfully completed.
```

❖ **Variable Attributes**

% TYPE

```
DECLARE
SALARY EMP.SAL % TYPE;
ECODE EMP.empno % TYPE;
BEGIN
Ecode :=&Ecode;
Select SAL into SALARY from EMP where EMPNO = ECODE;
dbms_output.put_line('Salary of ' || ECODE || 'is = || salary');
END;
```

```
Enter value for ecode : 7499
Salary of 7499 is = 1600
PL/SQL procedure successfully completed.
```

%ROWTYPE

```
DECLARE
EMPLOYEE EMP. % ROW TYPE;
BEGIN
EMPLOYEE.EMPNO := 2092;
5   EMPLOYEE.ENAME := 'Sanju';
Insert into EMP where (EMPNO, ENAME) Values (employee.empno, employee.ename);
dbms_output.put_line('Row Inserted');
END;
```

```
Row Inserted
PL/SQL procedure successfully completed.
```

❖ **Conditionals**

1) IF -THEN-ELSE

```
DECLARE
  a number(3) := 500;
BEGIN
  -- check the boolean condition using if statement
  IF( a < 20 ) THEN
    -- if condition is true then print the following
    dbms_output.put_line('a is less than 20 ' );
  ELSE
    dbms_output.put_line('a is not less than 20 ' );
  END IF;
  dbms_output.put_line('value of a is : ' || a);
END;
```

```
a is not less than 20
value of a is : 500
PL/SQL procedure successfully completed.
```

2) CASE

```
DECLARE
  grade char(1) := 'A';
BEGIN
  CASE grade
    when 'A' then dbms_output.put_line('Excellent');
    when 'B' then dbms_output.put_line('Very good');
    when 'C' then dbms_output.put_line('Good');
    when 'D' then dbms_output.put_line('Average');
    when 'F' then dbms_output.put_line('Passed with Grace');
    else dbms_output.put_line('Failed');
  END CASE;
END;
```

```
Excellent
PL/SQL procedure successfully completed.
```

❖ **Loop**

1) FOR

```
DECLARE
VAR1 NUMBER;
BEGIN
VAR1:=10;
FOR VAR2 IN 1..10
LOOP
DBMS_OUTPUT.PUT_LINE (VAR1*VAR2);
END LOOP;
END;
```

```
10
20
30
40
50
60
70
80
90
100
```

2) WHILE

```
DECLARE
VAR1 NUMBER;
VAR2 NUMBER;
BEGIN
VAR1:=200;
VAR2:=1;
WHILE (VAR2<=10)
LOOP
DBMS_OUTPUT.PUT_LINE (VAR1*VAR2);
VAR2:=VAR2+1;
END LOOP;
END;
```

```
200
400
600
800
1000
1200
1400
1600
1800
2000
```

CREATE TABLE Student (ID int, Name varchar(30),Age Number(3), Class Number(4),Marks Number(3));

insert into Student values (22, 'ABC',8,2,9);

Select * from Student;

Results Explain Describe Saved SQL History

ID	NAME	AGE	CLASS	MARKS
22	ABC	8	2	9
24	KST	8	2	6
20	XYZ	8	2	9
21	PQR	8	2	9
23	UVW	8	2	7

5 rows returned in 0.00 seconds CSV Export

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **STUDENT**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
STUDENT	ID	Number	-	-	0	-	✓	-	-
	NAME	Varchar2	50	-	-	-	✓	-	-
	AGE	Number	-	3	0	-	✓	-	-
	CLASS	Number	-	2	0	-	✓	-	-
	MARKS	Number	-	3	0	-	✓	-	-

1 - 5

CREATE TABLE IceCream (CustomerNo Number(10), Flavour varchar(30),Totalorders Number(3), Price Number (10));

insert into IceCream values (22, 'ABC',8,2,9);

insert into IceCream values (2,'Choco Bliss',1,50);

insert into IceCream values (5,'Butter Scotch',1,40);

Select * from IceCream;

Results Explain Describe Saved SQL History

CUSTOMERNO	FLAVOUR	TOTALORDERS	PRICE
4	Choco Bliss	1	50
5	Kufi	1	70
1	Butter Scotch	2	60
2	Chocolate	1	40
3	Cashew Pista	1	60

5 rows returned in 0.01 seconds CSV Export

desc IceCream;

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **ICECREAM**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ICECREAM	CUSTOMERNO	Number	-	10	0	-	✓	-	-
	FLAVOUR	Varchar2	30	-	-	-	✓	-	-
	TOTALORDERS	Number	-	3	0	-	✓	-	-
	PRICE	Number	-	10	0	-	✓	-	-

1 - 4

SELECT

SELECT Flavour,Price FROM IceCream

FLAVOUR	PRICE
Kulfi	70
Cashew Pista	100
Rasbery Mango	30
Butter Scotch	40
Choco Bliss	200

SELECT * FROM IceCream WHERE Price=100

CUSTOMERNO	FLAVOUR	TOTALORDERS	PRICE
3	Cashew Pista	2	100

INSERT

INSERT INTO IceCream(CustomerNo, Flavour,TotalOrders,Price)VALUES(6,'American Nuts',1,90);

CUSTOMERNO	FLAVOUR	TOTALORDERS	PRICE
1	Kulfi	1	70
3	Cashew Pista	2	100
4	Rasbery Mango	2	30
5	Butter Scotch	1	40
2	Choco Bliss	4	200
6	American Nuts	1	90

UPDATE

UPDATE IceCream SET Price=80 Where Customerno='6';

CUSTOMERNO	FLAVOUR	TOTALORDERS	PRICE
1	Kulfi	1	70
3	Cashew Pista	2	100
4	Rasbery Mango	2	30
5	Butter Scotch	1	40
2	Choco Bliss	4	200
6	American Nuts	1	80

DELETE

Delete from IceCream where CustomerNo=6;

CUSTOMERNO	FLAVOUR	TOTALORDERS	PRICE
1	Kulfi	1	70
3	Cashew Pista	2	100
4	Rasbery Mango	2	30
5	Butter Scotch	1	40
2	Choco Bliss	4	200

AVERAGE

select AVG(price) from IceCream

AVG(PRICE)
88

SUM

select SUM (PRICE) FRom IceCream;

SUM(PRICE)
440

COUNT

Select Count (CustomerNo) From IceCream;

COUNT(CUSTOMERNO)
5

ORDER

Select * from IceCream order by Customerno;

CUSTOMERNO	FLAVOUR	TOTALORDERS	PRICE
1	Kulfi	1	70
2	Choco Bliss	4	200
3	Cashew Pista	2	100
4	Rasbery Mango	2	30
5	Butter Scotch	1	40

NOT NULL

Alter Table IceCream

Modify Flavour varchar(30) NOT NULL;

desc IceCream;

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **ICECREAM**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ICECREAM	CUSTOMERNO	Number	-	10	0	-	✓	-	-
	FLAVOUR	Varchar2	30	-	-	-	-	-	-
	TOTALORDERS	Number	-	3	0	-	✓	-	-
	PRICE	Number	-	10	0	-	✓	-	-

1 - 4

UNIQUE

Alter Table IceCream

Add Unique (CustomerNo);

insert into IceCream values (2,'Choco Bliss',1,50);

```
ORA-00001: unique constraint (AIB4.SYS_C0016753) violated
```

DROP

Alter Table IceCream

Drop (CustomerNo);

insert into IceCream values (2,'Choco Bliss',1,50);

FLAVOUR	TOTALORDERS	PRICE
Choco Bliss	1	50
Kufi	1	70
Butter Scotch	2	60
Chocolate	1	40
Cashew Pista	1	60

Alter Table IceCream

Add Primary Key (Flavour);

CREATE TABLE Customer (Customer_Name varchar(30),Customer_No Number(10), Flavour Varchar(30),PRIMARY KEY(Customer_Name), FOREIGN KEY (Flavour)REFERENCES IceCream (Flavour));

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **CUSTOMER**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CUSTOMER	CUSTOMER_NAME	Varchar2	30	-	-	1	-	-	-
	CUSTOMER_NO	Number	-	10	0	-	✓	-	-
	FLAVOUR	Varchar2	30	-	-	-	✓	-	-

1 - 3

EXPERIMENT-8

AIM: To Perform view commands.

CREATE VIEW

```
Create view icecream_view as
select Flavour,price
from icecream;
select *from icecream_view;
```

FLAVOUR	PRICE
Choco Bliss	50
Kufi	70
Butter Scotch	60
Chocolate	40
Cashew Pista	60

WITH CHECK

```
Create view icecream_view1 as
select Flavour,price
from icecream
where flavour is not null
with check option;
select *from icecream_view1;
```

FLAVOUR	PRICE
Choco Bliss	50
Kufi	70
Butter Scotch	60
Chocolate	40
Cashew Pista	60

UPDATE

```
update icecream_view
set price=70
where flavour = 'Butter Scotch';
```

FLAVOUR	PRICE
Choco Bliss	50
Kufi	70
Butter Scotch	70
Chocolate	40
Cashew Pista	60

INSERT

INSERT INTO IceCream_view(Flavour,Price)VALUES('American Nuts',90);

select *from icecream_view;

FLAVOUR	PRICE
Choco Bliss	50
Kufi	70
American Nuts	90
Butter Scotch	70
Chocolate	40
Cashew Pista	60

DELETE

delete from icecream_view where price =90;

FLAVOUR	PRICE
Choco Bliss	50
Kufi	70
Butter Scotch	70
Chocolate	40
Cashew Pista	60

DROPPING VIEW

Drop view icecream_view;

```
View dropped.

0.41 seconds
```

OPEN ENDED EXPERIMENT 1

Program :

Implementation of triggers in SQL

Input:

CREATE TABLE Employee

(

)

Id INT PRIMARY KEY,

Name VARCHAR(45),

Salary INT,

Gender VARCHAR(12),

DepartmentId INT

Inserting some record in the table

INSERT INTO Employee VALUES (1,'Steffan', 82000, 'Male', 3),

(2,'Amelie', 52000, 'Female', 2),

(3,'Antonio', 25000, 'male', 1),

(4,'Marco', 47000, 'Male', 2),

(5,'Eliana', 46000, 'Female', 3)

SELECT * FROM Employee;

Id	Name	Salary	Gender	DepartmentId
1	Steffan	82000	Male	3
2	Amelie	52000	Female	2
3	Antonio	25000	male	1
4	Marco	47000	Male	2
5	Eliana	46000	Female	3
6	Peter	62000	Male	3

We will also create another table named 'Employee_Audit_Test' to automatically store transaction records of each operation, such as INSERT, UPDATE, or DELETE on the Employee table

```
CREATE TABLE Employee_Audit_Test
(
Id int IDENTITY,
Audit_Action text
)
```

Now, creating a trigger that stores transaction records of each insert operation on the Employee table into the Employee_Audit_Test table.

```
CREATE TRIGGER trInsertEmployee
ON Employee
FOR INSERT
AS
BEGIN
Declare @Id int
SELECT @Id = Id from inserted
INSERT INTO Employee_Audit_Test
VALUES ('New employee with Id = ' + CAST(@Id AS VARCHAR(10)) + ' is added at ' + C
 AST(Getdate() AS VARCHAR(22)))
END
INSERT INTO Employee VALUES (6,'Peter', 62000, 'Male', 3)
```

Id	Audit_Action
1	New employee with Id = 6 is added at Mar 24 2021 2:08PM

```
CREATE TRIGGER trDeleteEmployee
ON Employee
FOR DELETE
AS
BEGIN
Declare @Id int
SELECT @Id = Id from deleted
INSERT INTO Employee_Audit_Test
VALUES ('An existing employee with Id = ' + CAST(@Id AS VARCHAR(10)) + ' is deleted
```

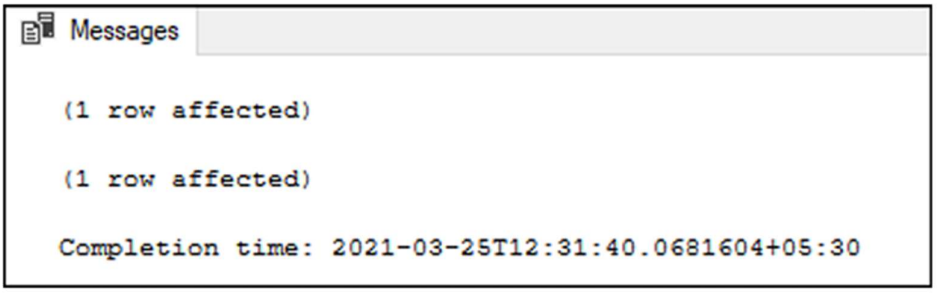
at ' + CAST(Getdate() AS VARCHAR(22)))

END

After creating a trigger, we will delete a record from the Employee table:

DELETE FROM Employee WHERE Id = 2;

If no error is found, it gives the message as below:



Finally, execute the SELECT statement to check the audit records:

Result:

Id	Audit_Action
1	New employee with Id = 6 is added at Mar 24 2021 2:08PM
2	An existing employee with Id = 2 is deleted at Mar 25 2021 12:26PM

OPEN ENDED EXPERIMENT 2

Program :

Implementation of cursor in SQL

Input:

```
CREATE TABLE customer (
id int PRIMARY KEY,
c_name nvarchar(45) NOT NULL,
email nvarchar(45) NOT NULL,
city nvarchar(25) NOT NULL
);
```

Next, we will insert values into the table

```
INSERT INTO customer (id, c_name, email, city)
VALUES (1,'Steffen', 'stephen@javatpoint.com', 'Texas'),
(2, 'Joseph', 'Joseph@javatpoint.com', 'Alaska'),
(3, 'Peter', 'Peter@javatpoint.com', 'California'),
(4,'Donald', 'donald@javatpoint.com', 'New York'),
(5, 'Kevin', 'kevin@javatpoint.com', 'Florida'),
(6, 'Marielia', 'Marielia@javatpoint.com', 'Arizona'),
(7,'Antonio', 'Antonio@javatpoint.com', 'New York'),
(8, 'Diego', 'Diego@javatpoint.com', 'California');
```

We can verify the data by executing the SELECT statement:

```
SELECT * FROM customer;
```

After executing the query,

id	c_name	email	city
1	Steffen	stephen@javatpoint.com	Texas
2	Joseph	Joseph@javatpoint.com	Alaska
3	Peter	Peter@javatpoint.com	California
4	Donald	donald@javatpoint.com	New York
5	Kevin	kevin@javatpoint.com	Florida
6	Marielia	Marielia@javatpoint.com	Arizona
7	Antonio	Antonio@javatpoint.com	New York
8	Diego	Diego@javatpoint.com	California

Now, we will create a cursor to display the customer records. --Declare the variables for holding data.

```
DECLARE @id INT, @c_name NVARCHAR(50), @city NVARCHAR(50)   --Declare and set counter.
DECLARE @Counter INT
SET @Counter = 1   --Declare a cursor
DECLARE PrintCustomers CURSOR
FOR
SELECT id, c_name, city FROM customer   --Open cursor
OPEN PrintCustomers   --Fetch the record into the variables.
FETCH NEXT FROM PrintCustomers INTO
@id, @c_name, @city   --LOOP UNTIL RECORDS ARE AVAILABLE.
WHILE @@FETCH_STATUS = 0
BEGIN
IF @Counter = 1
BEGIN
PRINT 'id' + CHAR(9) + 'c_name' + CHAR(9) + CHAR(9) + 'city'
PRINT '-------------------------'
END   --Print the current record
PRINT CAST(@id AS NVARCHAR(10)) + CHAR(9) + @c_name + CHAR(9) + CHAR(9
 ) + @city   --Increment the counter variable
SET @Counter = @Counter + 1   --Fetch the next record into the variables.
FETCH NEXT FROM PrintCustomers INTO
@id, @c_name, @city
END   --Close the cursor
CLOSE PrintCustomers   --Deallocate the cursor
DEALLOCATE PrintCustomers
```

After executing a cursor, we will get

Result: