Practical 1

Aim: Implementation of DDL commands of SQL with suitable examples

a) Create table b) Alter table c) Truncate table d) Drop table

Implementation of DML commands of SQL with suitable examples

a) Select b) Insert c) Update d) Delete

Solution:

CREATE, ALTER, DROP, TRUNCATE

(i) Create following tables according to the following definition.

DEPOSIT: Actno, Cname, Bname, Amount, Adate
BRANCH: Bname,City
CUSTOMERS: Cname, City
BORROW: Loanno, Cname, Bname, Amount

DEPOSIT	
Columnname	**Size**
Actno	5
Cname	20
Bname	20
Amount	8,2
Adate	

BORROW	
Columnname	**Size**
Loanno	5
CNAME	20
BNAME	20
AMOUNT	8,2

CUSTOMERS	
Columnname	**Size**
Cname	20
City	20

BRANCH	
Columnname	**Size**
Bname	20
City	20

Query: For DEPOSIT table - CREATE TABLE DEPOSIT (
Actno INT,
Cname VARCHAR(20),
Bname VARCHAR(20),
Amount DECIMAL(8,2),
Adate DATE
);

For BRANCH table - CREATE TABLE BRANCH (
Bname VARCHAR(20),
City VARCHAR(20)
);

For CUSTOMERS table - CREATE TABLE CUSTOMERS (
Cname VARCHAR(20),
City VARCHAR(20)
);

For BORROW table - CREATE TABLE BORROW (
Loanno INT,
Cname VARCHAR(20),
Bname VARCHAR(20),
Amount DECIMAL(8,2)
);

1. Add new column DOB into the CUSTOMER table.
Query: ALTER TABLE customers ADD dob DATE;

2. Add LOCATION column in BRANCH table.
Query: ALTER TABLE branch ADD location VARCHAR(20);

3. Modify customer name space to 30 characters.
Query: ALTER TABLE customers MODIFY Cname VARCHAR (30);
ALTER TABLE deposit MODIFY Cname VARCHAR (30);
ALTER TABLE borrow MODIFY Cname VARCHAR (30);

4. Change datatype of Loan number column to numeric type.
Query: ALTER TABLE borrow MODIFY loanno NUMERIC;

5. Delete LOCATION column.
Query: ALTER TABLE branch DROP COLUMN location;

6. Create DEPOSIT_TMP table that contains all information of DEPOSIT table.
Query: CREATE TABLE deposit_tmp AS SELECT * FROM deposit;

7. Create BRANCH_TMP table that contains only structure same as BRANCH table.
Query: CREATE TABLE branch_tmp AS SELECT * FROM branch WHERE false;

8. Remove all rows from DEPOSIT_TMP table.
Query: TRUNCATE TABLE deposit_tmp;

9. Delete DEPOSIT_TMP.
Query: DROP TABLE deposit_tmp;

(ii) Insert the data as shown below.

DEPOSIT

ACTNO	CNAME	BNAME	AMOUNT	ADATE
1	Abhinav	STATE BANK OF INDIA	1000.00	1-MAR-95
2	Aditya	KOTAK MAHINDRA BANK	5000.00	4-JAN-96
3	Minal	ANDHRA BANK	3500.00	17-NOV-95
4	Shrushti	YES BANK	1200.00	17-DEC-95
5	Radha	DENA BANK	3000.00	27-MAR-96
6	Akash	UCO	2000.00	31-MAR-96
7	Aniket	CITY	1000.00	5-SEP-95
8	Chandan	UNION	5000.00	2-JUL-95
9	Dushant	NAGPUR	7000.00	10-AUG-95

Query: INSERT INTO deposit (Actno, Cname, Bname, Amount, Adate)

VALUES (1, 'Abhinav', 'STATE BANK OF INDIA', 1000.00, '03-01-1995')

Query: INSERT ALL

INTO deposit VALUES (2, 'Aditya', 'KOTAK MAHINDRA BANK', 5000.00, '01-04-1996')

INTO deposit VALUES (3, 'Minal', 'ANDHRA BANK', 3500.00, '11-17-1995')

INTO deposit VALUES (4, 'Shrushti', 'YES BANK', 1200.00, '12-17-1995')

INTO deposit VALUES (5, 'Radha', 'DENA BANK', 3000.00, '03-27-1996')

INTO deposit VALUES (6, 'Akash', 'UCO', 2000.00, '03-31-1996')

INTO deposit VALUES (7, 'Aniket', 'CITY', 1000.00, '09-05-1995')

INTO deposit VALUES (8, 'Chandan', 'UNION', 5000.00, '07-02-1995')

INTO deposit VALUES (9, 'Dushant', 'NAGPUR', 7000.00, '08-10-1995')

SELECT * FROM DUAL;

BRANCH

BNAME	CITY
STATE BANK OF INDIA	NAGPUR
KOTAK MAHINDRA BANK	NAGPUR
ANDHRA BANK	DELHI
YES BANK	DELHI
DENA BANK	NAGPUR
UCO	BANGLORE
CITY	BOMBAY
UNION	BOMBAY
STATE BANK OF INDIA	DELHI
KOTAK MAHINDRA BANK	BOMBAY

Query: INSERT ALL

```
INTO branch VALUES ('STATE BANK OF INDIA','NAGPUR')
INTO branch VALUES ('KOTAK MAHINDRA BANK', 'NAGPUR')
INTO branch VALUES ('ANDHRA BANK', 'DELHI')
INTO branch VALUES ('YES BANK', 'DELHI')
INTO branch VALUES ('DENA BANK', 'NAGPUR')
INTO branch VALUES ('UCO', 'BANGLORE')
INTO branch VALUES ('CITY', 'BOMBAY')
INTO branch VALUES ('UNION', 'BOMBAY')
INTO branch VALUES ('STATE BANK OF INDIA', 'DELHI')
INTO branch VALUES ('KOTAK MAHINDRA BANK', 'BOMBAY')
SELECT * FROM DUAL;
```

CUSTOMERS

CNAME	CITY
Abhinav	CALCUTTA
Aditya	DELHI
Mrinal	
Shrushti	PATNA
Radha	NAGPUR
Akash	NAGPUR
Aniket	SURAT
Chandan	BOMBAY
Dushant	BOMBAY

Query: INSERT INTO customers (cname, city) VALUES ('Abhinav', 'CALCUTTA');

INSERT INTO customers (cname, city) VALUES ('Aditya', 'DELHI');

Query: INSERT INTO customers VALUES ('Mrinal', null);

Query: INSERT ALL

INTO customers VALUES ('Shrushti', 'PATNA')

INTO customers VALUES ('Radha', 'NAGPUR')

INTO customers VALUES ('Akash', 'NAGPUR')

INTO customers VALUES ('Aniket', 'SURAT')

INTO customers VALUES ('Chandan', 'BOMBAY')

INTO customers VALUES ('Dushant', 'BOMBAY')

SELECT * FROM DUAL;

BORROW

LOANNO	CNAME	BNAME	AMOUNT
10	Abhinav	STATE BANK OF INDIA	1000.00
20	Aditya	KOTAK MAHINDRA BANK	5000.00
30	Minal	ANDHRA BANK	3000.00
40	Shrushti	YES BANK	2000.00
50	Radha	DENA BANK	8000.00
60	Akash	UCO	3000.00
70	Aniket	CITY	4000.00

Query: INSERT ALL

```
INTO borrow VALUES (10,'Abhinav','STATE BANK OF INDIA', 1000.00)
INTO borrow VALUES (20,'Aditya','KOTAK MAHINDRA BANK', 5000.00)
INTO borrow VALUES (30,'Minal','ANDHRA BANK', 3000.00)
INTO borrow VALUES (40,'Shrushti','YES BANK', 2000.00)
INTO borrow VALUES (50,'Radha','DENA BANK', 8000.00)
INTO borrow VALUES (60,'Akash','UCO', 3000.00)
INTO borrow VALUES (70,'Aniket','CITY', 4000.00)
SELECT * FROM DUAL;
```

(iii) From the above given tables perform the following queries:

(1) Describe deposit, branch.
Query: DESC deposit;
DESC branch;

(2) Describe borrow, customers.
Query: DESC borrow;
DESC customers;

(3) List all data from table DEPOSIT.
Query: SELECT * FROM deposit;

(4) Give account no and amount of depositors.
Query: SELECT actno AS AccountNumber, amount FROM deposit;

(5) Give name of depositors having amount greater than 4000.
Query: SELECT cname FROM deposit WHERE amount > 4000;

(6) List all data from table BORROW.
Query: SELECT * FROM borrow;

(7) Give customer name and loan amount of borrowers.
Query: SELECT cname AS customername, amount FROM borrow;

(8) Give name of borrows from YES BANK branch and loan amount greater than 3000.
Query: SELECT cname WHERE bname='YES BANK' AND amount >3000;

(9) List all data from table CUSTOMERS.
Query: SELECT * FROM customers;

(10) List name of customers from BOMBAY.
Query: SELECT cname FROM customers WHERE city='BOMBAY';

(11) List all data from table BRANCH.
Query: SELECT * FROM branch;

(12) Give name of customers who opened account after date '1-12-96'.
Query: SELECT cname FROM deposit WHERE adate > '12-01-96';

(iv) Change the name of borrow to loan (rename borrow to loan).
Query: ALTER TABLE borrow RENAME TO loan;

(v) Give the SQL solutions for following queries

(1) Select branch names from the loan relation
Query: SELECT bname FROM loan;

(2) List all the tuples & attributes from the loan relation
Query: SELECT * FROM loan;

(3) Select branch_name & loan no. From the loan relation.
Query: SELECT bname, loanno FROM loan;

(4) Select all the loan numbers & amount from the loan relation.
Query: SELECT loanno, amount FORM loan;

(5) Find all loan no. & amount of the DENA BANK branch where the amount is Greater than 1300
Query: SELECT loanno, amount FROM loan WHERE bname='DENA BANK' AND amount > 1300;

(6) Give 10% interest to all depositors.
Query: UPDATE deposit SET amount = amount * 1.10;

(7) Give 10% interest to all depositors having branch UCO
Query: UPDATE deposit SET amount = amount * 1.10 WHERE bname='UCO';

(8) Delete deposit of VIJAY
Query: DELETE FROM deposit WHERE cname='VIJAY';

(9) Delete customers from BOMBAY city
Query: DELETE FROM customers WHERE city='BOMBAY';

(10) Delete depositors if the branch is ANDHRA BANK and depositor name is Aditya Karansingh Kachhawah.
Query: DELETE FROM deposit WHERE bname='ANDHRA BANK' AND cname='Aditya Karansingh Kachhawah';

(11) Delete depositors having deposit less than 500
Query: DELETE FROM deposit WHERE amount <500;

(12) Delete borrower having loan more than 10000
Query: DELETE FROM loan WHERE amount >10000;

(13) Delete borrower having loan more than 1000 and branch KOTAK MAHINDRA BANK
Query: DELETE FROM loan WHERE amount >1000 AND bname='KOTAK MAHINDRA BANK';

(14) Delete borrower having branch name UCO
Query: DELETE FROM loan WHERE bname='UCO'

Practical 2

Aim: Study and implementation of different types of constraints.

Create the tables described below

PRODUCT_MASTER

Column Name	Datatype	Size	Constraints
Product_No	Char	6	Primary Key
Description	Varchar2	15	Not Null
ProfitPercent	Number	4,2	Not Null
UnitMeasure	Varchar2	10	Not Null
SellPrice	Number	8,2	Not Null
CostPrice	Number	8,2	Not Null

SELLPRICE	Between 10000 to 80000
COSTPRICE	>1000

SALESMAN_MASTER

Column Name	Datatype	Size	Constraints
Salesman_No	Char	6	Primary Key
SalesmanName	Varchar2	20	Not Null
Address	Varchar2	30	Not Null
City	Varchar2	20	
PinCode	Number	6	
State	Varchar2	20	
SalAmt	Number	8,2	Not Null

CITY	CAN'T BE NAGPUR
PINCODE	UNIQUE

Client_Master

Column Name	Datatype	Size	Constraints
Client_No	Char	6	Primary Key
Name	Varchar2	20	Not Null
Address	Varchar2	100	
City	Varchar2	15	
Pincode	Number	6	
State	Varchar2	20	
Bal_Due	Number	8,2	

Sales_Order

Column Name	Datatype	Size	Default	Constraints
Order_No	Char	6		Primary Key
Client_No	Char	6		Foreign Key References Client_No of Client_Master Table
OrderDate	Date			Not Null
Salesman_No	Char	6		Foreign Key References Salesman_No of Salesman_Master Table

Solution Query:

A. For PRODUCT_MASTER Table:

```
CREATE TABLE PRODUCT_MASTER (
    Product_No CHAR(6) PRIMARY KEY,
    Description VARCHAR2(15) NOT NULL,
    ProfitPercent NUMBER(4,2) NOT NULL,
    UnitMeasure VARCHAR2(10) NOT NULL,
    SellPrice NUMBER(8,2) NOT NULL CHECK (SellPrice BETWEEN 10000 AND
     80000),
    CostPrice NUMBER(8,2) NOT NULL CHECK (CostPrice > 1000)
);
```

B. For SALESMAN_MASTER Table:

```
CREATE TABLE SALESMAN_MASTER (
    Salesman_No CHAR(6) PRIMARY KEY,
    SalesmanName VARCHAR2(20) NOT NULL,
    Address VARCHAR2(30) NOT NULL,
    City VARCHAR2(20),
    PinCode NUMBER(6),
    State VARCHAR2(20),
    SalAmt NUMBER(8,2) NOT NULL,
    CONSTRAINT pincode_unique UNIQUE (PinCode),
    CONSTRAINT city_not_nagpur CHECK (City <> 'NAGPUR')
);
```

C. For CLIENT_MASTER Table:

```
CREATE TABLE CLIENT_MASTER (
    Client_No CHAR(6) PRIMARY KEY,
    Name VARCHAR2(20) NOT NULL,
    Address VARCHAR2(100),
    City VARCHAR2(15),
    Pincode NUMBER(6),
    State VARCHAR2(20),
    Bal_Due NUMBER(8,2)
);
```

D. For SALES_ORDER Table:

```
CREATE TABLE SALES_ORDER (
   Order_No CHAR(6) PRIMARY KEY,
   Client_No CHAR(6),
   OrderDate DATE NOT NULL,
   Salesman_No CHAR(6),
   FOREIGN KEY (Client_No) REFERENCES CLIENT_MASTER(Client_No),
   FOREIGN KEY (Salesman_No) REFERENCES
    SALESMAN_MASTER(Salesman_No)
);
```

Practical 3

Aim: Study and implementation of different types of constraints.

Create the tables described below

PRODUCT_MASTER

Column Name	Datatype	Size	Constraints
Product_No	Char	6	Primary Key
Description	Varchar2	15	Not Null
ProfitPercent	Number	4,2	Not Null
UnitMeasure	Varchar2	10	Not Null
SellPrice	Number	8,2	Not Null
CostPrice	Number	8,2	Not Null

SELLPRICE	Between 10000 to 80000
COSTPRICE	>1000

SALESMAN_MASTER

Column Name	Datatype	Size	Constraints
Salesman_No	Char	6	Primary Key
SalesmanName	Varchar2	20	Not Null
Address	Varchar2	30	Not Null
City	Varchar2	20	
PinCode	Number	6	
State	Varchar2	20	
SalAmt	Number	8,2	Not Null

CITY	CAN'T BE NAGPUR
PINCODE	UNIQUE

Client_Master

Column Name	Datatype	Size	Constraints
Client_No	Char	6	Primary Key
Name	Varchar2	20	Not Null
Address	Varchar2	100	
City	Varchar2	15	
Pincode	Number	6	
State	Varchar2	20	
Bal_Due	Number	8,2	

Sales_Order

Column Name	Datatype	Size	Default	Constraints
Order_No	Char	6		Primary Key
Client_No	Char	6		Foreign Key References Client_No of Client_Master Table
OrderDate	Date			Not Null
Salesman_No	Char	6		Foreign Key References Salesman_No of Salesman_Master Table

Solution Query:

A. For PRODUCT_MASTER Table:

```
CREATE TABLE PRODUCT_MASTER (
   Product_No CHAR(6) PRIMARY KEY,
   Description VARCHAR2(15) NOT NULL,
   ProfitPercent NUMBER(4,2) NOT NULL,
   UnitMeasure VARCHAR2(10) NOT NULL,
   SellPrice NUMBER(8,2) NOT NULL CHECK (SellPrice BETWEEN 10000 AND
    80000),
   CostPrice NUMBER(8,2) NOT NULL CHECK (CostPrice > 1000)
);
```

B. For SALESMAN_MASTER Table:

```
CREATE TABLE SALESMAN_MASTER (
   Salesman_No CHAR(6) PRIMARY KEY,
   SalesmanName VARCHAR2(20) NOT NULL,
   Address VARCHAR2(30) NOT NULL,
   City VARCHAR2(20),
   PinCode NUMBER(6),
   State VARCHAR2(20),
   SalAmt NUMBER(8,2) NOT NULL,
   CONSTRAINT pincode_unique UNIQUE (PinCode),
   CONSTRAINT city_not_nagpur CHECK (City <> 'NAGPUR')
);
```

C. For CLIENT_MASTER Table:

```
CREATE TABLE CLIENT_MASTER (
   Client_No CHAR(6) PRIMARY KEY,
   Name VARCHAR2(20) NOT NULL,
   Address VARCHAR2(100),
   City VARCHAR2(15),
   Pincode NUMBER(6),
   State VARCHAR2(20),
   Bal_Due NUMBER(8,2)
);
```

D. For SALES_ORDER Table:

```
CREATE TABLE SALES_ORDER (
   Order_No CHAR(6) PRIMARY KEY,
   Client_No CHAR(6),
   OrderDate DATE NOT NULL,
   Salesman_No CHAR(6),
   FOREIGN KEY (Client_No) REFERENCES CLIENT_MASTER(Client_No),
   FOREIGN KEY (Salesman_No) REFERENCES
    SALESMAN_MASTER(Salesman_No)
);
```

Practical 4

Aim: Study and Implementation of

a) Aggregate functions
b) Group By & Having clause
c) Order by clause

Table: Furniture

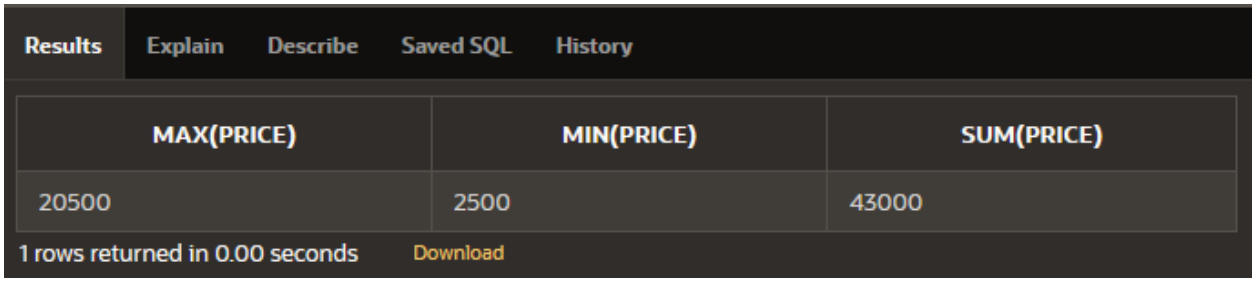
FCODE	NAME	MATERIAL	PRICE	MANUFDATE	WCODE
1002	Coffee Chair	Wood	3000	19-NOV-2017	W01
1003	Dining table	Wood	20500	12-JAN-2019	W02
1004	Coffee Table	Glass	5000	06-JAN-2019	W02
1005	Chair	Wood	2500	07-MAY-2017	W01
1006	Recliner	Fibre	12000	31-MAR-2018	W03

Solution:

1. Find the maximum, minimum, and total price.

Query: SELECT MAX(price), MIN(price), SUM(price) FROM Furniture;

Output:



2. Find the total number of furniture.

Query: SELECT COUNT(fcode) FROM Furniture;

Output:

Results | Explain | Describe | Saved SQL | History

COUNT(FCODE)
5

1 rows returned in 0.00 seconds Download

3. List the total price for each material.

Query: SELECT material, SUM(price) FROM Furniture GROUP BY (material);

Output:

Results | Explain | Describe | Saved SQL | History

MATERIAL	SUM(PRICE)
Glass	5000
Wood	26000
Fibre	12000

3 rows returned in 0.01 seconds Download

4. Give maximum price for wood.

Query: SELECT material, MAX(price) FROM Furniture GROUP BY (material)
HAVING material='Wood';

Output:

Results | Explain | Describe | Saved SQL | History

MATERIAL	MAX(PRICE)
Wood	20500

1 rows returned in 0.01 seconds Download

5. Display the average price of each material.

Query: SELECT material, AVG(price) AS "Average Price" FROM Furniture
GROUP BY (material);

Output:

Results | Explain | Describe | Saved SQL | History

MATERIAL	Average Price
Glass	5000
Wood	8666.666666666666666666666666666667
Fibre	12000

3 rows returned in 0.01 seconds Download

6. List fcode and the amount of all furniture in ascending order of price.

Query: SELECT fcode, price FROM Furniture ORDER BY price ASC;

Output:

Results | Explain | Describe | Saved SQL | History

FCODE	PRICE
1005	2500
1002	3000
1004	5000
1006	12000
1003	20500

5 rows returned in 0.01 seconds Download

7. List the materials having a sum of price more than 10000.

Query: SELECT material, SUM(price) FROM Furniture GROUP BY (material)
HAVING SUM(price)>10000;

Output:

Results | Explain | Describe | Saved SQL | History

MATERIAL	SUM(PRICE)
Wood	26000
Fibre	12000

2 rows returned in 0.01 seconds Download

8. Give the name of furniture whose number of wcode is less than 2.

Query: SELECT name FROM Furniture
WHERE wcode IN (
SELECT wcode FROM Furniture GROUP BY wcode
HAVING COUNT(wcode) < 2
);

Output:

Results | Explain | Describe | Saved SQL | History

NAME
Recliner

1 rows returned in 0.00 seconds Download

9. Give the name of materials whose average price is greater than 15000.

Query: SELECT material, AVG(price) FROM Furniture GROUP BY (material)
HAVING AVG(price)>15000;

Output: No data found.

10. Give the name of materials whose total price is less than 6000.

Query: SELECT material, SUM(price) FROM Furniture GROUP BY (material)
HAVING SUM(price)<6000;

Output:

Results | Explain | Describe | Saved SQL | History

MATERIAL	SUM(PRICE)
Glass	5000

1 rows returned in 0.01 seconds Download

11. Display the name of the furnitures in descending order of manufacturing date.

Query: SELECT name FROM Furniture ORDER BY manufdate DESC;

Output:

Results | Explain | Describe | Saved SQL | History

NAME
Dining table
Coffee Table
Recliner
Coffee Chair
Chair

5 rows returned in 0.01 seconds Download

Practical 5

Aim: Study and Implementation of different types of joins like cross join, natural join, inner join, and outer joins.

Solution:

1. Find the details of customer Aditya.

Query: SELECT * FROM customers NATURAL JOIN deposit WHERE Cname='Aditya'

2. Find the names of customers who are borrowers and depositors and having living city Nagpur.

Query: SELECT * FROM customers,loan,deposit
WHERE customers.cname=loan.cname AND customers.cname=deposit.cname
AND customers.city='NAGPUR'

3. Find the name, department number, and department name of all employees.

Query: SELECT ename,emp.deptno,dname FROM emp,dept
WHERE dept.deptno=emp.deptno

4. Display all jobs that are in department 30. Include the location of the department in the output.

Query: SELECT job,loc FROM emp,dept
WHERE dept.deptno=emp.deptno AND emp.deptno=30

5. Find the employee name, department number, and department name for all employees who work in NEW YORK.

Query: SELECT ename,emp.deptno,dname FROM emp,dept
WHERE emp.deptno = dept.deptno AND dept.loc='NEW YORK'

6. Find the name, account no, amount and city of customer ANIL.

Query: SELECT customers.cname,actno,amount,city FROM customers,deposit
WHERE customers.cname=deposit.cname AND customers.cname='ANIL'

7. Find the names of customers who are borrowers and depositors.

Query: SELECT deposit.cname FROM deposit,loan WHERE deposit.cname=loan.cname

8. Find the names of customers who are borrowers and depositors and having living city Nagpur and branch city mumbai.

Query: SELECT * FROM customers,loan,deposit,branch
WHERE customers.cname=loan.cname AND customers.cname=deposit.cname
AND deposit.bname=branch.bname AND customers.city='NAGPUR'
AND branch.city='MUMBAI'

9. Find the name, department number, and department name of all employees having salary 5000

Query: SELECT ename,emp.deptno,dname FROM emp,dept
WHERE emp.deptno = dept.deptno AND emp.sal >5 000

10. Retrieve all employees and their department names, including those without a department.

Query: SELECT * FROM emp LEFT OUTER JOIN dept ON emp.deptno = dept.deptno

11. Retrieve all departments and their employees, including departments without employees.

Query: SELECT * FROM emp RIGHT OUTER JOIN dept ON emp.deptno = dept.deptno

12. Retrieve all employees and departments.

Query: SELECT * FROM emp FULL OUTER JOIN dept ON emp.deptno = dept.deptno

13. Retrieve employees and their department names, sorting the result by department name in ascending order.

Query: SELECT * FROM emp, dept WHERE emp.deptno = dept.deptno
ORDER BY dept.dname ASC

14. Retrieve all customers and their deposit amounts, including those without deposits.

Query: SELECT customers.cname,deposit.amount

FROM customers LEFT OUTER JOIN deposit

ON customers.cname = deposit.cname

15. Retrieve all deposits and the corresponding customer information, including deposits without associated customers.

Query: SELECT * FROM customers RIGHT OUTER JOIN deposit

ON customers.cname = deposit.cname

16. Retrieve all customers and deposits, including those without matches on either side.

Query: SELECT * FROM customers FULL OUTER JOIN deposit

ON customers.cname = deposit.cname

Practical 6

Aim: Study and Implementation of Sub queries

Solution:

Employee (emp_no, emp_name, emp_sal, emp_comm, dept_no, Job_id, location)

Job (job_id, job_title, min_sal, max_sal)

department (dept_no, dept_name, location_id).

employee_manager (emp_no, emp_hiredate, mng_no,mng_name).

Query:
```
CREATE TABLE Employee (
  emp_no INT PRIMARY KEY,
  emp_name VARCHAR(100),
  emp_sal DECIMAL(10, 2),
  emp_comm DECIMAL(10, 2),
  dept_no INT, Job_id INT,
  location VARCHAR(100)
);

CREATE TABLE Job (
  job_id INT PRIMARY KEY,
  job_title VARCHAR(100),
  min_sal DECIMAL(10, 2),
  max_sal DECIMAL(10, 2)
);

CREATE TABLE Department (
  dept_no INT PRIMARY KEY,
  dept_name VARCHAR(100),
  location_id INT
);

CREATE TABLE Employee_Manager (
  emp_no INT PRIMARY KEY,
  emp_hiredate DATE, mng_no INT,
  mng_name VARCHAR(100)
);
```

17. Create a report that displays the employee number, name, and salary of all employees who earn more than the average salary. Sort the results in order of ascending salary

Query: SELECT emp_no, emp_name, emp_sal FROM Employee
WHERE emp_sal > (SELECT AVG(emp_sal) FROM Employee)
ORDER BY emp_sal ASC;

18. The HR department needs a report that displays the name, department number, and job ID of all employees whose department location ID is 1700.

Query: SELECT e.emp_name, d.dept_no, e.job_id FROM Employee e
JOIN department d ON e.dept_no = d.dept_no
WHERE d.location_id = 1700;

19. Create a report for HR that displays the department number, name, and job ID for every employee in the Executive department

Query: SELECT d.dept_no, d.dept_name, e.job_id
FROM Employee e JOIN department d ON e.dept_no = d.dept_no
WHERE d.dept_name = 'Executive';

20. Create a report that displays a list of all employees whose salary is more than the salary of any employee from department 60..

Query: SELECT emp_no, emp_name, emp_sal FROM Employee
WHERE emp_sal > ANY (SELECT emp_sal FROM Employee
WHERE dept_no = 60
);

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

Table: Customers

order_id	amount	customer_id
1	200	4
2	500	1
3	300	3
4	800	1
5	150	2

Table: Orders

```
CREATE TABLE Customer ( Customer_id INT PRIMARY KEY, First_name
                        VARCHAR(50), Last_name VARCHAR(50), Age INT,
                        Country VARCHAR(50)
);
INSERT INTO Customer (Customer_id, First_name, Last_name, Age, Country)
VALUES(1, 'John', 'Doe', 31,'USA');
INSERT INTO Customer (Customer_id, First_name, Last_name, Age, Country)
VALUES(2, 'Robert', 'Luna', 22, 'USA');
INSERT INTO Customer (Customer_id, First_name, Last_name, Age, Country)
VALUES(3, 'David', 'Robinson', 22, 'UK');
INSERT INTO Customer (Customer_id, First_name, Last_name, Age, Country)
VALUES(4, 'John', 'Reinhardt', 25, 'UK');
INSERT INTO Customer (Customer_id, First_name, Last_name, Age, Country)
VALUES(5, 'Betty', 'Doe', 28, 'UAE');
```

Results | Explain | Describe | Saved SQL | History

CUSTOMER_ID	FIRST_NAME	LAST_NAME	AGE	COUNTRY
1	John	Doe	31	USA
4	John	Reinhardt	25	UK
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
5	Betty	Doe	28	UAE

5 rows returned in 0.01 seconds Download

```
CREATE TABLE Orders (
    Order_id INT PRIMARY KEY,
    Amount INT,
    Customer_id INT,
    FOREIGN KEY (Customer_id) REFERENCES Customer(Customer_id)
);
INSERT INTO Orders (Order_id, Amount, Customer_id) VALUES(1, 200, 4);
INSERT INTO Orders (Order_id, Amount, Customer_id) VALUES(2, 500, 1);
INSERT INTO Orders (Order_id, Amount, Customer_id) VALUES(3, 300, 3);
INSERT INTO Orders (Order_id, Amount, Customer_id) VALUES(4, 800, 1);
INSERT INTO Orders (Order_id, Amount, Customer_id) VALUES(5, 150, 2);
```

Results | Explain | Describe | Saved SQL | History

ORDER_ID	AMOUNT	CUSTOMER_ID
1	200	4
3	300	3
4	800	1
2	500	1
5	150	2

5 rows returned in 0.01 seconds Download

1. Find the details of customers who have placed an order.

Query: SELECT c.* FROM Customer c

WHERE EXISTS (SELECT 1 FROM Orders o

WHERE o.Customer_id = c.Customer_id

);

Results | Explain | Describe | Saved SQL | History

CUSTOMER_ID	FIRST_NAME	LAST_NAME	AGE	COUNTRY
4	John	Reinhardt	25	UK
3	David	Robinson	22	UK
1	John	Doe	31	USA
2	Robert	Luna	22	USA

4 rows returned in 0.02 seconds Download

2. Find all customers who do not have order more than $500.

Query: SELECT c.* FROM Customer c

WHERE NOT EXISTS (SELECT 1 FROM Orders o

WHERE o.Customer_id = c.Customer_id

AND o.Amount > 500);

Results | Explain | Describe | Saved SQL | History

CUSTOMER_ID	FIRST_NAME	LAST_NAME	AGE	COUNTRY
2	Robert	Luna	22	USA
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE
3	David	Robinson	22	UK

4 rows returned in 0.01 seconds Download

3. Find the customers who have placed highest order.

Query: SELECT c.* FROM Customer c JOIN Orders o ON c.Customer_id = o.Customer_id
WHERE o.Amount = (SELECT MAX(Amount) FROM Orders);

Results | Explain | Describe | Saved SQL | History

CUSTOMER_ID	FIRST_NAME	LAST_NAME	AGE	COUNTRY
1	John	Doe	31	USA

1 rows returned in 0.01 seconds Download

4. Find all customers whose age is greater than the average age of all customers.

Query: SELECT * FROM Customer
WHERE Age > (SELECT AVG(Age) FROM Customer)

Results | Explain | Describe | Saved SQL | History

CUSTOMER_ID	FIRST_NAME	LAST_NAME	AGE	COUNTRY
1	John	Doe	31	USA
5	Betty	Doe	28	UAE

2 rows returned in 0.01 seconds Download

5. Find the name of customers whose age is equal to age of John.

Query: SELECT First_name, Last_name FROM Customer
WHERE Age = (SELECT MAX(Age) FROM Customer
WHERE First_name = 'John');

Results | Explain | Describe | Saved SQL | History

FIRST_NAME	LAST_NAME
John	Doe

1 rows returned in 0.01 seconds Download