Data Base Management System-Lab Mannual

230701167

Ex. No.: 1

CREATION OF BASE TABLE AND DML OPERATIONS

1.

```
CREATE TABLE MY_EMPLOYEE (
   ID NUMBER(4) NOT NULL,
   Last_name VARCHAR2(25),
   First_name VARCHAR2(25),
   Userid VARCHAR2(25),
   Salary NUMBER(9,2),
   CONSTRAINT pk_employee PRIMARY KEY (ID)
);
```

2.

```
INSERT INTO MY_EMPLOYEE (ID, Last_name, First_name, Userid, Salary)
VALUES (1, 'Patel', 'Ralph', 'rpatel', 895);
INSERT INTO MY_EMPLOYEE (ID, Last_name, First_name, Userid, Salary)
VALUES (2, 'Dancs', 'Betty', 'bdancs', 860);
```

3.

```
SELECT * FROM MY_EMPLOYEE;
```

4.

```
INSERT INTO MY_EMPLOYEE (ID, Last_name, First_name, Userid, Salary)
VALUES (3, 'Biri', 'Ben', NULL, 1100);
INSERT INTO MY_EMPLOYEE (ID, Last_name, First_name, Userid, Salary)
VALUES (4, 'Newman', 'Chad', NULL, 750);

UPDATE MY_EMPLOYEE
SET Userid = LOWER(CONCAT(SUBSTR(First_name, 1, 1), SUBSTR(Last_name, 1, 7)))
WHERE ID = 3 OR ID = 4;
```

5.

```
DELETE FROM MY_EMPLOYEE
WHERE First_name = 'Betty' AND Last_name = 'Dancs';
```

6.

```
UPDATE MY_EMPLOYEE
SET Last_name = NULL, First_name = NULL, Userid = NULL, Salary = NULL
WHERE ID = 4;
```

7.

```
COMMIT;
```

8.

```
UPDATE MY_EMPLOYEE
SET Last_name = 'Drexler'
WHERE ID = 3;
```

9.

```
UPDATE MY_EMPLOYEE
SET Salary = 1000
WHERE Salary < 900;
```

Ex. No.: 2

DATA MANIPULATIONS

A.

Initial:

```
CREATE TABLE EMPLOYEES (
   Employee_id NUMBER(6) NOT NULL,
   First_Name VARCHAR2(20),
   Last_Name VARCHAR2(25) NOT NULL,
   Email VARCHAR2(25) NOT NULL,
   Phone_Number VARCHAR2(20),
   Hire_date DATE  NOT  NULL,
   Job_id VARCHAR2(10) NOT NULL,
   Salary NUMBER(8,2),
   Commission_pct NUMBER(2,2),
   Manager_id NUMBER(6),
   Department_id NUMBER(4),
   CONSTRAINT pk_employee_id PRIMARY KEY (Employee_id)
);
```

```
INSERT INTO EMPLOYEES
VALUES (101, 'John', 'Doe', 'jdoe@example.com', '1234567890', TO_DATE('2022-06-15', 'YYYY-MM-DD'), 'IT_PROG', 5000, NULL, 100, 60);
```

```
INSERT INTO EMPLOYEES
VALUES (102, 'Jane', 'Austin', 'jaustin@example.com', '0987654321', TO_DATE('2022-08-20', 'YYYY-MM-DD'), 'HR_MAN', 4800, NULL, 101, 70);
```

```
INSERT INTO EMPLOYEES
VALUES (103, 'Mark', 'Smith', 'msmith@example.com', '1230984567', TO_DATE('2023-01-10', 'YYYY-MM-DD'), 'SA_REP', 4600, 0.10, 100, 80);
```

```
INSERT INTO EMPLOYEES
```

```
VALUES (104, 'Chad', 'Newman', 'cnewman@example.com', '7896541230',
TO_DATE('2021-11-03', 'YYYY-MM-DD'), 'FI_MGR', 6000, NULL, 102, 60);


INSERT INTO EMPLOYEES

VALUES (105, 'Betty', 'Austin', 'baustin@example.com', '9874563210', TO_DATE('2020-
12-25', 'YYYY-MM-DD'), 'HR_CLERK', 3900, NULL, 101, 70);
```

1.

```
SELECT Employee_id, First_Name, Last_Name, Salary

FROM EMPLOYEES;
```

2.

```
SELECT Employee_id, First_Name, Last_Name

FROM EMPLOYEES

WHERE Manager_id = 100;
```

3.

```
SELECT First_Name, Last_Name

FROM EMPLOYEES

WHERE Salary >= 4800;
```

4.

```
SELECT First_Name, Last_Name

FROM EMPLOYEES

WHERE Last_Name = 'AUSTIN';
```

5.

```
SELECT First_Name, Last_Name

FROM EMPLOYEES

WHERE Department_id IN (60, 70, 80);
```

6.

```
SELECT DISTINCT Manager_id

FROM EMPLOYEES;
```

B.

Initial:

```
CREATE TABLE EMP (
   EmpNo NUMBER(6),
   EmpName VARCHAR2(25),
   Job VARCHAR2(20),
   Basic NUMBER(8,2),
   DA NUMBER(8,2),
   HRA NUMBER(8,2),
   PF NUMBER(8,2),
   GrossPay NUMBER(8,2),
   NetPay NUMBER(8,2),
   Department_id NUMBER(4)
);
```

1.

```
INSERT INTO EMP (EmpNo, EmpName, Job, Basic, Department_id)
VALUES (1, 'John Doe', 'Manager', 5000, 60);
INSERT INTO EMP (EmpNo, EmpName, Job, Basic, Department_id)
VALUES (2, 'Jane Austin', 'Clerk', 4000, 70);
INSERT INTO EMP (EmpNo, EmpName, Job, Basic, Department_id)
VALUES (3, 'Mark Smith', 'Sales', 3500, 80);
INSERT INTO EMP (EmpNo, EmpName, Job, Basic, Department_id)
VALUES (4, 'Chad Newman', 'Manager', 6000, 60);
INSERT INTO EMP (EmpNo, EmpName, Job, Basic, Department_id)
VALUES (5, 'Betty Austin', 'HR', 3900, 70);

UPDATE EMP
SET
   DA = 0.30 * Basic,
   HRA = 0.40 * Basic,
   PF = 0.12 * Basic;
```

```
UPDATE EMP
SET
   GrossPay = Basic + DA + HRA;

UPDATE EMP
SET
   NetPay = GrossPay - PF;
```

2.

```
SELECT *
FROM EMP e
WHERE Basic = (
   SELECT MIN(Basic)
   FROM EMP
   WHERE Department_id = e.Department_id
);
```

3.

```
SELECT EmpName, NetPay
FROM EMP
WHERE NetPay < 7500;
```

C.

1.

```
CREATE TABLE DEPT (
   ID NUMBER(7),
   NAME VARCHAR2(25),
   CONSTRAINT pk_dept PRIMARY KEY (ID)
);
```

2.

```
CREATE TABLE EMP (
  ID NUMBER(7),
  LAST_NAME VARCHAR2(25),
  FIRST_NAME VARCHAR2(25),
  DEPT_ID NUMBER(7),
  CONSTRAINT pk_emp PRIMARY KEY (ID)
);
```

3.

```
ALTER TABLE EMP
MODIFY LAST_NAME VARCHAR2(50);
```

4.

```
CREATE TABLE EMPLOYEES2 AS
SELECT Employee_id AS Id, First_Name, Last_Name, Salary, Department_id AS Dept_id
FROM EMPLOYEES;
```

5.

```
DROP TABLE EMP;
```

6.

```
ALTER TABLE EMPLOYEES2
RENAME TO EMP;
```

7.

```
COMMENT ON TABLE DEPT IS 'Department Table';
COMMENT ON TABLE EMP IS 'Employees Table';
DESC DEPT;
DESC EMP;
```

8.

```
ALTER TABLE EMP
DROP COLUMN First_Name;
DESC EMP;
```

Ex. No.: 3

WRITING BASIC SQL SELECT STATEMENTS

Initial:

```
CREATE TABLE departments (
   dept_id NUMBER(4) PRIMARY KEY,
   dept_name VARCHAR2(30),
   manager_id NUMBER(6),
   location_id NUMBER(4)
);

INSERT INTO departments (dept_id, dept_name, manager_id, location_id)
VALUES (10, 'HR', 101, 1001);
INSERT INTO departments (dept_id, dept_name, manager_id, location_id)
VALUES (20, 'Sales', 102, 1002);
INSERT INTO departments (dept_id, dept_name, manager_id, location_id)
VALUES (30, 'IT', 103, 1003);

CREATE TABLE EMPLOYEES (
   Employee_id NUMBER(6) NOT NULL,
   First_Name VARCHAR2(20),
   Last_Name VARCHAR2(25) NOT NULL,
   Email VARCHAR2(25) NOT NULL,
   Phone_Number VARCHAR2(20),
   Hire_date DATE  NOT  NULL,
   Job_id VARCHAR2(10) NOT NULL,
   Salary NUMBER(8,2),
   Commission_pct NUMBER(2,2),
   Manager_id NUMBER(6),
   Department_id NUMBER(4),
   CONSTRAINT pk_employee_id PRIMARY KEY (Employee_id)
);
```

```
INSERT INTO EMPLOYEES
VALUES (101, 'John', 'Doe', 'jdoe@example.com', '1234567890', TO_DATE('2022-06-15', 'YYYY-MM-DD'), 'IT_PROG', 5000, NULL, 100, 60);

INSERT INTO EMPLOYEES
VALUES (102, 'Jane', 'Austin', 'jaustin@example.com', '0987654321', TO_DATE('2022-08-20', 'YYYY-MM-DD'), 'HR_MAN', 4800, NULL, 101, 70);

INSERT INTO EMPLOYEES
VALUES (103, 'Mark', 'Smith', 'msmith@example.com', '1230984567', TO_DATE('2023-01-10', 'YYYY-MM-DD'), 'SA_REP', 4600, 0.10, 100, 80);

INSERT INTO EMPLOYEES
VALUES (104, 'Chad', 'Newman', 'cnewman@example.com', '7896541230', TO_DATE('2021-11-03', 'YYYY-MM-DD'), 'FI_MGR', 6000, NULL, 102, 60);

INSERT INTO EMPLOYEES
VALUES (105, 'Betty', 'Austin', 'baustin@example.com', '9874563210', TO_DATE('2020-12-25', 'YYYY-MM-DD'), 'HR_CLERK', 3900, NULL, 101, 70);
```

1.

```
SELECT Employee_id, Last_Name, Salary * 12 AS "ANNUAL SALARY"
FROM EMPLOYEES;
```

2.

```
DESC departments;
SELECT * FROM departments;
```

3.

```
SELECT employee_id, last_name, job_id, hire_date
FROM employees;
```

4.

```
SELECT employee_id, last_name, job_id, hire_date AS "STARTDATE"
FROM employees;
```

5.

```
SELECT DISTINCT job_id
FROM employees;
```

6.

```
SELECT last_name || ', ' || job_id AS "EMPLOYEE and TITLE"
FROM employees;
```

7.

```
SELECT employee_id || ', ' || first_name || ', ' || last_name || ', ' || email || ', ' ||
phone_number || ', ' || hire_date || ', ' || job_id || ', ' || salary || ', ' || commission_pct || ', ' ||
manager_id || ', ' || department_id AS "THE_OUTPUT"
FROM employees;
```

Ex. No.: 4

WORKING WITH CONSTRAINTS

Initial:

```
CREATE TABLE departments (
   dept_id NUMBER(4),
   dept_name VARCHAR2(30),
   manager_id NUMBER(6),
   location_id NUMBER(4)
);

INSERT INTO departments (dept_id, dept_name, manager_id, location_id)
VALUES (10, 'HR', 101, 1001);
INSERT INTO departments (dept_id, dept_name, manager_id, location_id)
VALUES (20, 'Sales', 102, 1002);
INSERT INTO departments (dept_id, dept_name, manager_id, location_id)
VALUES (30, 'IT', 103, 1003);

CREATE TABLE EMP (
   Employee_id NUMBER(6) NOT NULL,
   First_Name VARCHAR2(20),
   Last_Name VARCHAR2(25) NOT NULL,
   Email VARCHAR2(25) NOT NULL,
   Phone_Number VARCHAR2(20),
   Hire_date DATE  NOT  NULL,
   Job_id VARCHAR2(10) NOT NULL,
   Salary NUMBER(8,2),
   Commission_pct NUMBER(2,2),
   Manager_id NUMBER(6),
   Department_id NUMBER(4)
);

INSERT INTO EMP
```

VALUES (101, 'John', 'Doe', 'jdoe@example.com', '1234567890', TO_DATE('2022-06-15', 'YYYY-MM-DD'), 'IT_PROG', 5000, NULL, 100, 60);

INSERT INTO EMP

VALUES (102, 'Jane', 'Austin', 'jaustin@example.com', '0987654321', TO_DATE('2022-08-20', 'YYYY-MM-DD'), 'HR_MAN', 4800, NULL, 101, 70);

INSERT INTO EMP

VALUES (103, 'Mark', 'Smith', 'msmith@example.com', '1230984567', TO_DATE('2023-01-10', 'YYYY-MM-DD'), 'SA_REP', 4600, 0.10, 100, 80);

INSERT INTO EMP

VALUES (104, 'Chad', 'Newman', 'cnewman@example.com', '7896541230', TO_DATE('2021-11-03', 'YYYY-MM-DD'), 'FI_MGR', 6000, NULL, 102, 60);

INSERT INTO EMP

VALUES (105, 'Betty', 'Austin', 'baustin@example.com', '9874563210', TO_DATE('2020-12-25', 'YYYY-MM-DD'), 'HR_CLERK', 3900, NULL, 101, 70);

1.

ALTER TABLE EMP

ADD CONSTRAINT my_emp_id_pk PRIMARY KEY (Employee_id);

2.

ALTER TABLE DEPARTMENTS

ADD CONSTRAINT my_dept_id_pk PRIMARY KEY (dept_id);

3.

ALTER TABLE EMP

ADD DEPT_ID NUMBER(4);

ALTER TABLE EMP

ADD CONSTRAINT my_emp_dept_id_fk FOREIGN KEY (DEPT_ID)

REFERENCES DEPARTMENTS(dept_id);

4.

```
ALTER TABLE EMP
ADD COMMISSION NUMBER(2,2);

ALTER TABLE EMP
ADD CONSTRAINT chk_commission_gt_zero CHECK (COMMISSION > 0);
```

Ex. No.: 5

CREATING VIEWS

Initial:

```
CREATE TABLE JOB_GRADE (
   Grade_level VARCHAR2(2),
   Lowest_sal NUMBER,
   Highest_sal NUMBER
);

INSERT INTO JOB_GRADE (Grade_level, Lowest_sal, Highest_sal)
VALUES ('A', 3000, 4999);
INSERT INTO JOB_GRADE (Grade_level, Lowest_sal, Highest_sal)
VALUES ('B', 5000, 6999);
INSERT INTO JOB_GRADE (Grade_level, Lowest_sal, Highest_sal)
VALUES ('C', 7000, 9999);

CREATE TABLE DEPARTMENTS (
   dept_id NUMBER(4) PRIMARY KEY,
   dept_name VARCHAR2(30),
   manager_id NUMBER(6),
   location_id NUMBER(4)
);

INSERT INTO DEPARTMENTS (dept_id, dept_name, manager_id, location_id)
VALUES (80, 'HR', 101, 1001);
INSERT INTO DEPARTMENTS (dept_id, dept_name, manager_id, location_id)
VALUES (20, 'Sales', 102, 1002);
INSERT INTO DEPARTMENTS (dept_id, dept_name, manager_id, location_id)
VALUES (30, 'IT', 103, 1003);
INSERT INTO DEPARTMENTS (dept_id, dept_name, manager_id, location_id)
VALUES (50, 'Support', 104, 1004);
```

```
CREATE TABLE EMPLOYEES (
   Employee_id NUMBER(6) NOT NULL,
   First_Name VARCHAR2(20),
   Last_Name VARCHAR2(25) NOT NULL,
   Email VARCHAR2(25) NOT NULL,
   Phone_Number VARCHAR2(20),
   Hire_date DATE  NOT  NULL,
   Job_id VARCHAR2(10) NOT NULL,
   Salary NUMBER(8,2),
   Commission NUMBER(2,2),
   Manager_id NUMBER(6),
   Dept_ID NUMBER(4),
   CONSTRAINT pk_employee_id PRIMARY KEY (Employee_id),
   CONSTRAINT fk_department FOREIGN KEY (Dept_ID) REFERENCES
DEPARTMENTS(dept_id)
);


INSERT INTO EMPLOYEES (Employee_id, First_Name, Last_Name, Email,
Phone_Number, Hire_date, Job_id, Salary, Commission, Manager_id, Dept_ID)
VALUES (101, 'John', 'Doe', 'jdoe@example.com', '1234567890',
TO_DATE('2022-06-15', 'YYYY-MM-DD'), 'IT_PROG', 5000, 0.05, 100, 80);


INSERT INTO EMPLOYEES (Employee_id, First_Name, Last_Name, Email,
Phone_Number, Hire_date, Job_id, Salary, Commission, Manager_id, Dept_ID)
VALUES (102, 'Jane', 'Austin', 'jaustin@example.com', '0987654321',
TO_DATE('2022-08-20', 'YYYY-MM-DD'), 'HR_MAN', 4800, NULL, 101, 50);


INSERT INTO EMPLOYEES (Employee_id, First_Name, Last_Name, Email,
Phone_Number, Hire_date, Job_id, Salary, Commission, Manager_id, Dept_ID)
VALUES (103, 'Mark', 'Smith', 'msmith@example.com', '1230984567',
TO_DATE('2023-01-10', 'YYYY-MM-DD'), 'SA_REP', 4600, 0.10, 100, 30);


INSERT INTO EMPLOYEES (Employee_id, First_Name, Last_Name, Email,
Phone_Number, Hire_date, Job_id, Salary, Commission, Manager_id, Dept_ID)
VALUES (104, 'Chad', 'Matos', 'cnewman@example.com', '7896541230',
TO_DATE('2021-11-03', 'YYYY-MM-DD'), 'FI_MGR', 6000, NULL, 102, 50);
```

```
INSERT INTO EMPLOYEES (Employee_id, First_Name, Last_Name, Email,
Phone_Number, Hire_date, Job_id, Salary, Commission, Manager_id, Dept_ID)

VALUES (105, 'Betty', 'Austin', 'baustin@example.com', '9874563210',
TO_DATE('2020-12-25', 'YYYY-MM-DD'), 'HR_CLERK', 3900, NULL, 101, 20);
```

1.

```
CREATE VIEW EMPLOYEE_VU AS

SELECT Employee_id,

    First_Name || ' ' || Last_Name AS EMPLOYEE,

    Dept_ID

FROM EMPLOYEES;
```

2.

```
SELECT * FROM EMPLOYEE_VU;
```

3.

```
SELECT VIEW_NAME, TEXT

FROM USER_VIEWS

WHERE VIEW_NAME = 'EMPLOYEE_VU';
```

4.

```
SELECT EMPLOYEE, Dept_ID

FROM EMPLOYEE_VU;
```

5.

```
CREATE VIEW DEPT50 AS

SELECT Employee_id AS EMPNO,

    Last_Name AS EMPLOYEE,

    Dept_ID AS DEPTNO

FROM EMPLOYEES

WHERE Dept_ID = 50;
```

6.

```
DESC DEPT50;

SELECT * FROM DEPT50;
```

7.

```
UPDATE EMPLOYEES
SET Dept_ID = 80
WHERE Last_Name = 'Matos';
```

8.

```
CREATE VIEW SALARY_VU AS
SELECT E.Last_Name AS Employee,
    D.dept_name AS Department,
    E.Salary AS  Salary,
    J.Grade_level AS Grade
FROM EMPLOYEES E
JOIN DEPARTMENTS D ON E.Dept_ID = D.dept_id
JOIN JOB_GRADE J ON E.Salary BETWEEN J.Lowest_sal AND J.Highest_sal;
```

Ex. No.: 6

RESTRICTING AND SORTING DATA

Initial:

```
CREATE TABLE EMPLOYEES (
   Employee_id NUMBER(6) NOT NULL,
   Last_Name VARCHAR2(25) NOT NULL,
   First_Name VARCHAR2(20),
   Email VARCHAR2(25) NOT NULL,
   Phone_Number VARCHAR2(20),
   Hire_date DATE  NOT  NULL,
   Job_id VARCHAR2(10) NOT NULL,
   Salary NUMBER(8,2),
   Commission_pct NUMBER(2,2),
   Manager_id NUMBER(6),
   Department_id NUMBER(4),
   CONSTRAINT pk_employee_id PRIMARY KEY (Employee_id)
);
```

```
INSERT INTO EMPLOYEES
VALUES (176, 'Smith', 'John', 'jsmith@example.com', '555-1234', TO_DATE('1994-07-15', 'YYYY-MM-DD'), 'SA_REP', 13000, 0.10, NULL, 30);
```

```
INSERT INTO EMPLOYEES
VALUES (177, 'Doe', 'Jane', 'jdoe@example.com', '555-5678', TO_DATE('1998-03-25', 'YYYY-MM-DD'), 'IT_PROG', 11000, NULL, 176, 20);
```

```
INSERT INTO EMPLOYEES
VALUES (178, 'Johnson', 'Emily', 'ejohnson@example.com', '555-8765', TO_DATE('1995-11-30', 'YYYY-MM-DD'), 'ST_CLERK', 2500, NULL, 176, 50);
```

```
INSERT INTO EMPLOYEES
VALUES (179, 'Miller', 'Tom', 'tmiller@example.com', '555-4321', TO_DATE('1996-09-10', 'YYYY-MM-DD'), 'SA_REP', 8000, 0.15, 176, 20);
```

```
INSERT INTO EMPLOYEES
VALUES (180, 'Matos', 'Daniel', 'dmatos@example.com', '555-7890', TO_DATE('1994-05-23', 'YYYY-MM-DD'), 'HR_CLERK', 3000, NULL, NULL, 50);

INSERT INTO EMPLOYEES
VALUES (196, 'Sharukesh', 'John', 'jsharuk@example.com', '555-1274', TO_DATE('1999-07-15', 'YYYY-MM-DD'), 'SA_REP', 16000, 0.10, NULL, 60);
```

1.

```
SELECT Last_Name, Salary
FROM EMPLOYEES
WHERE Salary > 12000;
```

2.

```
SELECT Last_Name, Department_id
FROM EMPLOYEES
WHERE Employee_id = 176;
```

3.

```
SELECT Last_Name, Salary
FROM EMPLOYEES
WHERE Salary NOT BETWEEN 5000 AND 12000;
```

4.

```
SELECT Last_Name, Job_id, Hire_date
FROM EMPLOYEES
WHERE Hire_date BETWEEN TO_DATE('1998-02-20', 'YYYY-MM-DD') AND
TO_DATE('1998-05-01', 'YYYY-MM-DD')
ORDER BY Hire_date;
```

5.

```
SELECT Last_Name, Department_id
FROM EMPLOYEES
WHERE Department_id IN (20, 50)
ORDER BY Last_Name;
```

6.

```
SELECT Last_Name AS EMPLOYEE, Salary AS "MONTHLY SALARY"
FROM EMPLOYEES
WHERE Salary BETWEEN 5000 AND 12000
AND Department_id IN (20, 50)
ORDER BY Last_Name;
```

7.

```
SELECT Last_Name, Hire_date
FROM EMPLOYEES
WHERE TO_CHAR(Hire_date, 'YYYY') = '1994';
```

8.

```
SELECT Last_Name, Job_id
FROM EMPLOYEES
WHERE Manager_id IS NULL;
```

9.

```
SELECT Last_Name, Salary, Commission_pct
FROM EMPLOYEES
WHERE Commission_pct IS NOT NULL
ORDER BY Salary DESC, Commission_pct DESC;
```

10.

```
SELECT Last_Name
FROM EMPLOYEES
WHERE Last_Name LIKE '_a%';
```

11.

```
SELECT Last_Name
FROM EMPLOYEES
WHERE Last_Name LIKE '%a%' AND Last_Name LIKE '%e%';
```

12.

```
SELECT Last_Name, Job_id, Salary
FROM EMPLOYEES
WHERE Job_id IN ('SA_REP', 'ST_CLERK')
AND Salary NOT IN (2500, 3500, 7000);
```

Ex. No.: 7

USING SET OPERATORS

Initial:

```
CREATE TABLE EMPLOYEES (
   employee_id NUMBER PRIMARY KEY,
   last_name VARCHAR2(50),
   job_id VARCHAR2(10),
   department_id NUMBER,
   hire_date DATE
);
CREATE TABLE DEPARTMENTS (
   department_id NUMBER PRIMARY KEY,
   department_name VARCHAR2(50),
   country_id VARCHAR2(10)
);

CREATE TABLE JOB_HISTORY (
   employee_id NUMBER,
   job_id VARCHAR2(10) PRIMARY KEY,
   hire_date DATE
);

CREATE TABLE COUNTRIES (
   country_id VARCHAR2(10) PRIMARY KEY,
   country_name VARCHAR2(50)
);

INSERT INTO EMPLOYEES VALUES
(101, 'Smith', 'ST_CLERK', 10, TO_DATE('2015-06-01', 'YYYY-MM-DD'));
INSERT INTO EMPLOYEES VALUES
(102, 'Johnson', 'SA_MAN', 50, TO_DATE('2018-03-12', 'YYYY-MM-DD'));
```

```
INSERT INTO EMPLOYEES VALUES
(103, 'Williams', 'ST_CLERK', 20, TO_DATE('2019-07-14', 'YYYY-MM-DD'));
INSERT INTO EMPLOYEES VALUES
(104, 'Brown', 'IT_PROG', 30, TO_DATE('2017-11-25', 'YYYY-MM-DD'));
INSERT INTO EMPLOYEES VALUES
(105, 'Jones', 'HR_REP', 40, TO_DATE('2020-01-03', 'YYYY-MM-DD'));
INSERT INTO EMPLOYEES VALUES
(106, 'Garcia', 'ST_CLERK', 50, TO_DATE('2015-04-19', 'YYYY-MM-DD'));
INSERT INTO EMPLOYEES VALUES
(107, 'Davis', 'IT_PROG', 20, TO_DATE('2019-01-01', 'YYYY-MM-DD'));
INSERT INTO EMPLOYEES VALUES
(108, 'Taylor', 'SA_MAN', 10, TO_DATE('2021-09-12', 'YYYY-MM-DD'));
INSERT INTO EMPLOYEES VALUES
(109, 'Clark', 'IT_PROG', 30, TO_DATE('2018-08-01', 'YYYY-MM-DD'));

INSERT INTO DEPARTMENTS
VALUES (10, 'Administration', 'US');
INSERT INTO DEPARTMENTS
VALUES (20, 'Marketing', 'US');
INSERT INTO DEPARTMENTS
VALUES (30, 'IT', 'UK');
INSERT INTO DEPARTMENTS
VALUES (40, 'HR', 'FR');
INSERT INTO DEPARTMENTS
VALUES (50, 'Sales', 'DE');
INSERT INTO DEPARTMENTS
VALUES (60, 'Finance', 'IN');

INSERT INTO JOB_HISTORY
VALUES (101, 'ST_CLERK', TO_DATE('2015-06-01', 'YYYY-MM-DD'));
INSERT INTO JOB_HISTORY
VALUES (102, 'SA_MAN', TO_DATE('2018-03-12', 'YYYY-MM-DD'));
```

```
INSERT INTO JOB_HISTORY
VALUES (107, 'IT_PROG',TO_DATE('2019-01-01', 'YYYY-MM-DD'));

INSERT INTO COUNTRIES
VALUES ('US', 'United States');
INSERT INTO COUNTRIES
VALUES ('UK', 'United Kingdom');
INSERT INTO COUNTRIES
VALUES ('FR', 'France');
INSERT INTO COUNTRIES
VALUES ('DE', 'Germany');
INSERT INTO COUNTRIES
VALUES ('IN', 'India');
INSERT INTO COUNTRIES
VALUES ('JP', 'Japan');
```

1.

```
SELECT department_id
FROM  DEPARTMENTS
MINUS
SELECT department_id
FROM EMPLOYEES
WHERE job_id = 'ST_CLERK';
```

2.

```
SELECT country_id, country_name
FROM COUNTRIES
WHERE country_id IN (
   SELECT country_id FROM COUNTRIES
   MINUS
   SELECT DISTINCT country_id FROM DEPARTMENTS
   WHERE department_name='HR'
);
```

3.

```
SELECT job_id, department_id
FROM EMPLOYEES
WHERE department_id = 10
UNION ALL
SELECT job_id, department_id
FROM EMPLOYEES
WHERE department_id = 50
UNION ALL
SELECT job_id, department_id
FROM EMPLOYEES
WHERE department_id = 20;
```

4.

```
SELECT employee_id, job_id, hire_date
FROM EMPLOYEES
INTERSECT
SELECT employee_id, job_id, hire_date
FROM JOB_HISTORY
ORDER BY hire_date ASC;
```

5.

```
SELECT last_name, department_id, NULL AS department_name
FROM EMPLOYEES
UNION
SELECT NULL AS last_name, department_id, department_name
FROM DEPARTMENTS;
```

Ex. No.: 8

WORKING WITH MULTIPLE TABLES

Initial:

```
CREATE TABLE EMPLOYEES (
   EMPLOYEE_ID NUMBER(6) PRIMARY KEY,
   FIRST_NAME  VARCHAR2(20),
   LAST_NAME VARCHAR2(25) NOT NULL,
   EMAIL VARCHAR2(50) UNIQUE NOT NULL,
   PHONE_NUMBER VARCHAR2(20),
   HIRE_DATE DATE NOT NULL,
   JOB_ID VARCHAR2(10) NOT NULL,
   SALARY NUMBER(8,2),
   COMMISSION_PCT NUMBER(2,2),
   MANAGER_ID NUMBER(6),
   DEPARTMENT_ID NUMBER(4)
);

CREATE TABLE  DEPARTMENTS  (
   DEPARTMENT_ID NUMBER(4) PRIMARY KEY,
   DEPARTMENT_NAME VARCHAR2(30) NOT NULL,
   MANAGER_ID NUMBER(6),
   LOCATION_ID NUMBER(4)
);

CREATE TABLE JOBS (
   JOB_ID VARCHAR2(10) PRIMARY KEY,
   JOB_TITLE VARCHAR2(35) NOT NULL,
   MIN_SALARY NUMBER(8,2),
   MAX_SALARY NUMBER(8,2)
);

CREATE TABLE LOCATIONS (
```

```
   LOCATION_ID NUMBER(4) PRIMARY KEY,
   STREET_ADDRESS VARCHAR2(40),
   POSTAL_CODE VARCHAR2(12),
   CITY VARCHAR2(30) NOT NULL,
   COUNTRY VARCHAR2(25),
   COUNTRY_CODE VARCHAR2(20)
);

CREATE TABLE JOB_GRADES (
   GRADE_LEVEL CHAR(1) PRIMARY KEY,
   LOW_SALARY NUMBER(8,2),
   HIGH_SALARY NUMBER(8,2)
);

INSERT INTO LOCATIONS VALUES
(1000, '123 Main St', '560001', 'Toronto', 'Ontario', 'CA');
INSERT INTO LOCATIONS VALUES
(1001, '456 Park Ave', '110020', 'New York', 'New York', 'US');
INSERT INTO LOCATIONS VALUES
(1002, '789 King Rd', '700008', 'London', 'England', 'UK');
INSERT INTO LOCATIONS VALUES
(1003, '696 VOC Rd', '600098', 'Chennai', 'India', 'IND');

INSERT INTO DEPARTMENTS VALUES
(10, 'Administration', NULL, 1001);
INSERT INTO DEPARTMENTS VALUES
(20, 'Marketing', 101, 1002);
INSERT INTO DEPARTMENTS VALUES
(30, 'IT', 102, 1001);
INSERT INTO DEPARTMENTS VALUES
(40, 'HR', 103, 1000);
INSERT INTO DEPARTMENTS VALUES
(50, 'Sales', 104, 1000);
```

```
INSERT INTO DEPARTMENTS VALUES
(80, 'Finance', 105, 1003);

INSERT INTO JOBS VALUES
('AD_PRES', 'President', 20000, 40000);
INSERT INTO JOBS VALUES
('MK_MAN', 'Marketing Manager', 10000, 20000);
INSERT INTO JOBS VALUES
('IT_PROG', 'Programmer', 5000, 15000);
INSERT INTO JOBS VALUES
('HR_REP', 'HR Representative', 6000, 12000);
INSERT INTO JOBS VALUES
('FI_MGR', 'Finance Manager', 12000, 25000);
INSERT INTO JOBS VALUES
('SA_REP', 'Sales Representative', 5000, 10000);

INSERT INTO JOB_GRADES VALUES
('A', 5000, 7000);
INSERT INTO JOB_GRADES VALUES
('B', 7001, 12000);
INSERT INTO JOB_GRADES VALUES
('C', 12001, 15000);
INSERT INTO JOB_GRADES VALUES
('D', 15001, 20000);
INSERT INTO JOB_GRADES VALUES
('E', 20001, 40000);

INSERT INTO EMPLOYEES VALUES
(101, 'John', 'King', 'JKing@example.com', '1234567890', TO_DATE('2010-01-01', 'YYYY-MM-DD'), 'AD_PRES', 30000, NULL, NULL, 10);
INSERT INTO EMPLOYEES VALUES
(102, 'Sara', 'Davies', 'SDavies@example.com', '2234567890', TO_DATE('2013-05-10', 'YYYY-MM-DD'), 'MK_MAN', 15000, NULL, 101, 20);
```

```
INSERT INTO EMPLOYEES VALUES

(103, 'Mike', 'Smith', 'MSmith@example.com', '3234567890', TO_DATE('2012-03-15',
'YYYY-MM-DD'), 'IT_PROG', 9000, NULL, 102, 80);

INSERT INTO EMPLOYEES VALUES

(104, 'Anna', 'Brown', 'ABrown@example.com', '4234567890', TO_DATE('2013-09-20',
'YYYY-MM-DD'), 'HR_REP', 7000, 0.10, 102, 40);

INSERT INTO EMPLOYEES VALUES

(105, 'James', 'Wilson', 'JWilson@example.com', '5234567890', TO_DATE('2014-07-23',
'YYYY-MM-DD'), 'FI_MGR', 18000, NULL, 101, 80);

INSERT INTO EMPLOYEES VALUES

(106, 'Sophia', 'Johnson', 'SJohnson@example.com', '6234567890', TO_DATE('2015-11-
05', 'YYYY-MM-DD'), 'SA_REP', 8000, 0.15, 103, 50);

INSERT INTO EMPLOYEES VALUES

(107, 'Emily', 'Taylor', 'ETaylor@example.com', '7234567890', TO_DATE('2016-04-18',
'YYYY-MM-DD'), 'SA_REP', 8500, 0.12, 104, 50);
```

1.

```
SELECT e.LAST_NAME, e.DEPARTMENT_ID, d.DEPARTMENT_NAME

FROM EMPLOYEES e

JOIN DEPARTMENTS d ON e.DEPARTMENT_ID = d.DEPARTMENT_ID;
```

2.

```
SELECT DISTINCT e.JOB_ID, d.LOCATION_ID, l.COUNTRY

FROM EMPLOYEES e

JOIN DEPARTMENTS d ON e.DEPARTMENT_ID = d.DEPARTMENT_ID

JOIN LOCATIONS l ON d.LOCATION_ID=l.LOCATION_ID

WHERE e.DEPARTMENT_ID = 80;
```

3.

```
SELECT e.LAST_NAME, d.DEPARTMENT_NAME, d.LOCATION_ID, l.CITY

FROM EMPLOYEES e

JOIN DEPARTMENTS d ON e.DEPARTMENT_ID = d.DEPARTMENT_ID

JOIN LOCATIONS l ON d.LOCATION_ID = l.LOCATION_ID

WHERE e.COMMISSION_PCT IS NOT NULL;
```

4.

```
SELECT e.LAST_NAME, d.DEPARTMENT_NAME
FROM EMPLOYEES e
JOIN DEPARTMENTS d ON e.DEPARTMENT_ID = d.DEPARTMENT_ID
WHERE LOWER(e.LAST_NAME) LIKE '%a%';
```

5.

```
SELECT e.LAST_NAME, e.JOB_ID, e.DEPARTMENT_ID, d.DEPARTMENT_NAME
FROM EMPLOYEES e
JOIN DEPARTMENTS d ON e.DEPARTMENT_ID = d.DEPARTMENT_ID
JOIN LOCATIONS l ON d.LOCATION_ID = l.LOCATION_ID
WHERE l.CITY = 'Toronto';
```

6.

```
SELECT e.LAST_NAME AS "Employee", e.EMPLOYEE_ID AS "Emp#",
    m.LAST_NAME AS "Manager", m.EMPLOYEE_ID AS "Mgr#"
FROM EMPLOYEES e
JOIN EMPLOYEES m ON e.MANAGER_ID = m.EMPLOYEE_ID;
```

7.

```
SELECT e.LAST_NAME AS "Employee", e.EMPLOYEE_ID AS "Emp#",
    m.LAST_NAME AS "Manager", m.EMPLOYEE_ID AS "Mgr#"
FROM EMPLOYEES e
LEFT JOIN EMPLOYEES m ON e.MANAGER_ID = m.EMPLOYEE_ID
ORDER BY e.EMPLOYEE_ID;
```

8.

```
SELECT e1.LAST_NAME AS "Employee", e1.DEPARTMENT_ID, e2.LAST_NAME AS "Co-
Workers"
FROM EMPLOYEES e1
JOIN EMPLOYEES e2 ON e1.DEPARTMENT_ID = e2.DEPARTMENT_ID
WHERE e1.EMPLOYEE_ID = 106 AND e1.EMPLOYEE_ID <> e2.EMPLOYEE_ID;
```

9.

```
DESCRIBE JOB_GRADES;

SELECT e.LAST_NAME, e.JOB_ID, d.DEPARTMENT_NAME, e.SALARY, jg.GRADE_LEVEL
FROM EMPLOYEES e
JOIN DEPARTMENTS d ON e.DEPARTMENT_ID = d.DEPARTMENT_ID
JOIN JOB_GRADES jg ON e.SALARY BETWEEN jg.LOW_SALARY AND jg.HIGH_SALARY;
```

10.

```
SELECT e.LAST_NAME AS "Employee", e.HIRE_DATE AS "Hire Date"
FROM EMPLOYEES e
JOIN EMPLOYEES r ON r.LAST_NAME = 'Davies'
WHERE e.HIRE_DATE > r.HIRE_DATE;
```

11.

```
SELECT e.LAST_NAME AS "Employee", e.HIRE_DATE AS "Emp Hired",
    m.LAST_NAME AS "Manager", m.HIRE_DATE AS "Mgr Hired"
FROM EMPLOYEES e
JOIN EMPLOYEES m ON e.MANAGER_ID = m.EMPLOYEE_ID
WHERE e.HIRE_DATE < m.HIRE_DATE AND e.EMPLOYEE_ID <> m.EMPLOYEE_ID;
```

Ex. No.: 9

SUB QUERIES

Initial:

```
CREATE TABLE departments (
   department_id NUMBER PRIMARY KEY,
   department_name VARCHAR2(100),
   location_id NUMBER
);

CREATE TABLE employees (
   employee_id NUMBER PRIMARY KEY,
   last_name VARCHAR2(100),
   first_name VARCHAR2(100),
   hire_date DATE,
   salary NUMBER(10, 2),
   department_id NUMBER,
   job_id VARCHAR2(10),
   manager_id NUMBER,
   FOREIGN KEY (department_id) REFERENCES departments(department_id)
);

INSERT INTO departments VALUES (10, 'Executive', 1700);
INSERT INTO departments VALUES (20, 'HR', 1800);
INSERT INTO departments VALUES (30, 'IT', 1700);
INSERT INTO departments VALUES (40, 'Finance', 1600);

INSERT INTO employees
VALUES (1, 'King', 'John', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 10000, 10, 'CEO',
NULL);
INSERT INTO employees
VALUES (2, 'Zlotkey', 'Jane', TO_DATE('2001-02-15', 'YYYY-MM-DD'), 8000, 10, 'VP', 1);
```

```
INSERT INTO employees

VALUES (3, 'Smith', 'Anna', TO_DATE('2005-03-10', 'YYYY-MM-DD'), 8500, 10,
'Manager', 1);

INSERT INTO employees

VALUES (4, 'Green', 'Tom', TO_DATE('2010-05-20', 'YYYY-MM-DD'), 4500, 20, 'HR Rep',
2);

INSERT INTO employees

VALUES (5, 'Brown', 'Lily', TO_DATE('2011-06-22', 'YYYY-MM-DD'), 4200, 20, 'HR Rep',
2);

INSERT INTO employees

VALUES (6, 'Turner', 'Michael', TO_DATE('2012-07-13', 'YYYY-MM-DD'), 5000, 30,
'Developer', 3);

INSERT INTO employees

VALUES (7, 'Miller', 'Sandra', TO_DATE('2014-08-25', 'YYYY-MM-DD'), 5500, 30,
'Developer', 3);

INSERT INTO employees

VALUES (8, 'Jones', 'Peter', TO_DATE('2018-09-15', 'YYYY-MM-DD'), 6000, 40,
'Accountant', 1);

INSERT INTO employees

VALUES (9, 'Austin', 'James', TO_DATE('2014-06-13', 'YYYY-MM-DD'), 7500, 30,
'Developer', 1);
```

1.

```
SELECT last_name, hire_date

FROM employees

WHERE department_id = (

   SELECT department_id FROM employees

   WHERE last_name = 'Zlotkey'

)

AND last_name != 'Zlotkey';
```

2.

```
SELECT employee_id, last_name, salary
FROM employees
WHERE salary > (
   SELECT AVG(salary) FROM employees
)
ORDER BY salary;
```

3.

```
SELECT employee_id, last_name
FROM employees
WHERE department_id IN (
   SELECT department_id FROM employees
   WHERE last_name LIKE '%u%'
);
```

4.

```
SELECT last_name, department_id, job_id
FROM employees
WHERE department_id IN (
   SELECT department_id FROM departments
   WHERE location_id=1700
);
```

5.

```
SELECT last_name, salary
FROM employees e
WHERE EXISTS(
   SELECT last_name FROM employees m
   WHERE e.manager_id = m.employee_id
   AND m.last_name='King'
);
```

6.

```
SELECT department_id, last_name, job_id
FROM employees
WHERE department_id = (
   SELECT department_id
   FROM departments
   WHERE department_name = 'Executive'
);
```

7.

```
SELECT e.employee_id, e.last_name, e.salary
FROM employees e
WHERE e.salary > (SELECT AVG(salary) FROM employees)
AND EXISTS (
   SELECT *
   FROM employees e2
   WHERE e.department_id = e2.department_id
   AND e2.last_name LIKE '%u%'
);
```

Ex. No.: 10

AGGREGATING DATA USING GROUP FUNCTIONS

Initial:

```
CREATE TABLE departments (
   department_id NUMBER PRIMARY KEY,
   department_name VARCHAR2(100),
   location_id NUMBER
);

CREATE TABLE employees (
   employee_id NUMBER PRIMARY KEY,
   last_name VARCHAR2(100),
   first_name VARCHAR2(100),
   hire_date DATE,
   salary NUMBER(10, 2),
   department_id NUMBER,
   job_id VARCHAR2(10),
   manager_id NUMBER,
   FOREIGN KEY (department_id) REFERENCES departments(department_id)
);

INSERT INTO departments VALUES (10, 'Executive', 1700);
INSERT INTO departments VALUES (20, 'HR', 1800);
INSERT INTO departments VALUES (30, 'IT', 1700);
INSERT INTO departments VALUES (40, 'Finance', 1600);

INSERT INTO employees
VALUES (1, 'King', 'John', TO_DATE('1998-01-01', 'YYYY-MM-DD'), 10000, 10, 'CEO',
NULL);
INSERT INTO employees
VALUES (2, 'Zlotkey', 'Jane', TO_DATE('1995-02-15', 'YYYY-MM-DD'), 8000, 10, 'VP', 1);
```

```
INSERT INTO employees

VALUES (3, 'Smith', 'Anna', TO_DATE('1996-03-10', 'YYYY-MM-DD'), 8500, 10,
'Manager', 1);

INSERT INTO employees

VALUES (4, 'Green', 'Tom', TO_DATE('1998-05-20', 'YYYY-MM-DD'), 7500, 20, 'HR Rep',
2);

INSERT INTO employees

VALUES (5, 'Brown', 'Lily', TO_DATE('1997-06-22', 'YYYY-MM-DD'), 7200, 20, 'HR Rep',
2);

INSERT INTO employees

VALUES (6, 'Turner', 'Michael', TO_DATE('1995-07-13', 'YYYY-MM-DD'), 5000, 30,
'Developer', 3);

INSERT INTO employees

VALUES (7, 'Miller', 'Sandra', TO_DATE('1992-08-25', 'YYYY-MM-DD'), 5500, 30,
'Developer', 3);

INSERT INTO employees

VALUES (8, 'Jones', 'Peter', TO_DATE('1997-09-15', 'YYYY-MM-DD'), 6500, 40,
'Accountant', 1);

INSERT INTO employees

VALUES (9, 'Austin', 'James', TO_DATE('1996-06-13', 'YYYY-MM-DD'), 7500, 30,
'Developer', 1);
```

1. TRUE
2. FALSE
3. TRUE

4.

```
SELECT

   ROUND(MAX(salary)) AS Maximum,

   ROUND(MIN(salary)) AS Minimum,

   ROUND(SUM(salary)) AS Sum,

   ROUND(AVG(salary)) AS Average

FROM employees;
```

5.

```
SELECT
   job_id,
   ROUND(MIN(salary)) AS Minimum,
   ROUND(MAX(salary)) AS Maximum,
   ROUND(SUM(salary)) AS Sum,
   ROUND(AVG(salary)) AS Average
FROM employees
GROUP BY job_id;
```

6.

```
SELECT
   job_id,
   COUNT(*) AS Number_of_People
FROM employees
WHERE job_id = 'Developer'
GROUP BY job_id;
```

7.

```
SELECT
   COUNT(DISTINCT manager_id) AS Number_of_Managers
FROM employees
WHERE manager_id IS NOT NULL;
```

8.

```
SELECT
   ROUND(MAX(salary) - MIN(salary)) AS DIFFERENCE
FROM employees;
```

9.

```
SELECT
   manager_id,
   MIN(salary) AS Lowest_Salary
FROM employees
WHERE manager_id IS NOT NULL
GROUP BY manager_id
HAVING MIN(salary) > 6000
ORDER BY Lowest_Salary DESC;
```

10.

```
SELECT
   COUNT(*) AS Total_Employees,
   SUM(CASE WHEN EXTRACT(YEAR FROM hire_date) = 1995 THEN 1 ELSE 0 END) AS Employees_1995,
   SUM(CASE WHEN EXTRACT(YEAR FROM hire_date) = 1996 THEN 1 ELSE 0 END) AS Employees_1996,
   SUM(CASE WHEN EXTRACT(YEAR FROM hire_date) = 1997 THEN 1 ELSE 0 END) AS Employees_1997,
   SUM(CASE WHEN EXTRACT(YEAR FROM hire_date) = 1998 THEN 1 ELSE 0 END) AS Employees_1998
FROM employees;
```

11.

```
SELECT
   job_id,
   department_id,
   SUM(salary) AS Total_Salary,
   AVG(salary) AS Average_Salary
FROM employees
WHERE department_id IN (20, 50, 80, 90)
GROUP BY job_id, department_id
ORDER BY department_id, job_id;
```

12.

```
SELECT
   d.department_name AS "Name-Location",
   d.location_id AS Location,
   COUNT(e.employee_id) AS "Number of People",
   ROUND(AVG(e.salary), 2) AS Salary
FROM departments d
LEFT JOIN employees e ON d.department_id = e.department_id
GROUP BY d.department_name, d.location_id;
```

Ex. No.: 11

PL SQL PROGRAMS

Initial:

```
CREATE TABLE employees (
   employee_id NUMBER PRIMARY KEY,
   first_name  VARCHAR2(50),
   last_name VARCHAR2(50),
   job_id VARCHAR2(10),
   salary NUMBER(8, 2),
   hire_date DATE,
   department_id NUMBER
);

CREATE TABLE departments (
   department_id NUMBER PRIMARY KEY,
   department_name VARCHAR2(50),
   manager_id NUMBER
);

CREATE TABLE jobs (
   job_id VARCHAR2(10) PRIMARY KEY,
   job_title VARCHAR2(50),
   min_salary NUMBER(8, 2), max_salary NUMBER(8, 2)
);

CREATE TABLE job_history (
   employee_id  NUMBER,
   start_date DATE, end_date DATE,
   job_id VARCHAR2(10),
   department_id NUMBER
);
```

```
BEGIN
    INSERT INTO employees VALUES (110, 'John', 'Doe', 'IT_PROG', 60000,
    TO_DATE('2020-01-15', 'YYYY-MM-DD'), 50);
    INSERT INTO employees VALUES (122, 'Jane', 'Smith', 'SA_REP', 55000,
    TO_DATE('2019-07-10', 'YYYY-MM-DD'), 80);

    INSERT INTO departments VALUES (50, 'IT', 110);
    INSERT INTO departments VALUES (80, 'Sales', 122);
    INSERT INTO jobs VALUES ('IT_PROG', 'Programmer', 40000, 80000);
    INSERT INTO jobs VALUES ('SA_REP', 'Sales Representative', 30000, 60000);

    INSERT INTO job_history VALUES (110, TO_DATE('2018-05-01', 'YYYY-MM-
    DD'), TO_DATE('2020-01-14', 'YYYY-MM-DD'), 'HR_REP', 60);
    INSERT INTO job_history VALUES (122, TO_DATE('2017-03-01', 'YYYY-MM-
    DD'), TO_DATE('2019-07-09', 'YYYY-MM-DD'), 'SA_REP', 80);
END;/
```

1.

```
DECLARE
    emp_salary employees.salary%TYPE;
     incentive NUMBER(8,2);
BEGIN
    SELECT salary INTO emp_salary FROM employees WHERE employee_id = 110;
    incentive := emp_salary * 0.1;
    DBMS_OUTPUT.PUT_LINE('Incentive for Employee ID 110: ' || incentive);
END; /
```

2.

```
DECLARE
    "EmployeeID" NUMBER := 110;
BEGIN
    DBMS_OUTPUT.PUT_LINE(EmployeeID);
END; /
```

3.

```
BEGIN
    UPDATE employees SET salary = salary + 5000 WHERE employee_id = 122;
    DBMS_OUTPUT.PUT_LINE('Salary adjusted for Employee ID 122');
END; /
```

4.

```
CREATE OR REPLACE PROCEDURE CheckNullAndOperator IS
    value1 BOOLEAN := TRUE;
    value2 BOOLEAN := TRUE;
BEGIN
    IF value1 IS NOT NULL AND value2 IS NOT NULL AND value1 AND value2 THEN
        DBMS_OUTPUT.PUT_LINE('Both conditions are TRUE');
    ELSE
        DBMS_OUTPUT.PUT_LINE('One or both conditions are FALSE');
    END IF;
END; /
```

5.

```
DECLARE
    emp_name employees.first_name%TYPE;
BEGIN
    FOR rec IN (SELECT first_name FROM employees WHERE first_name LIKE 'J%')
    LOOP
        DBMS_OUTPUT.PUT_LINE('Employee name starting with J: ' ||
        rec.first_name);
    END LOOP;
END; /
```

6.

```
DECLARE
    num1 NUMBER := 10;
    num2 NUMBER := 5;
    num_small NUMBER;
    num_large NUMBER;
BEGIN
    IF num1 < num2 THEN
        num_small := num1;
        num_large := num2;
    ELSE
        num_small := num2;
        num_large := num1;
    END IF;
    DBMS_OUTPUT.PUT_LINE('Small Number: ' || num_small || ', Large Number: ' ||
    num_large);
END; /
```

7.

```
CREATE OR REPLACE PROCEDURE UpdateIncentive IS
    target NUMBER := 100000;
    sales NUMBER := 120000;
    incentive NUMBER;
BEGIN
    IF sales >= target THEN
        incentive := sales * 0.1;
        DBMS_OUTPUT.PUT_LINE('Incentive updated to ' || incentive);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Target not met. No incentive.');
    END IF;
END; /
```

8.

```
CREATE OR REPLACE PROCEDURE CalculateIncentive(sales_limit IN NUMBER) IS
    incentive NUMBER;
BEGIN
    IF sales_limit > 50000 THEN
        incentive := sales_limit * 0.15;
    ELSE
        incentive := sales_limit * 0.1;
    END IF;
    DBMS_OUTPUT.PUT_LINE('Incentive: ' || incentive);
END; /
```

9.

```
DECLARE
    emp_count NUMBER;
    vacancies NUMBER := 45;
BEGIN
    SELECT COUNT(*) INTO emp_count FROM employees WHERE department_id =
    50;
    IF emp_count < vacancies THEN
        DBMS_OUTPUT.PUT_LINE('Vacancies available: ' || (vacancies -
        emp_count));
    ELSE
        DBMS_OUTPUT.PUT_LINE('No vacancies');
    END IF;
END; /
```

10.

```
DECLARE
    emp_count NUMBER;
    dept_id NUMBER := 80;
    vacancies NUMBER := 45;
BEGIN
    SELECT COUNT(*) INTO emp_count FROM employees WHERE department_id =
    dept_id;
    IF emp_count < vacancies THEN
        DBMS_OUTPUT.PUT_LINE('Vacancies in Department ' || dept_id || ': ' ||
        (vacancies - emp_count));
    ELSE
        DBMS_OUTPUT.PUT_LINE('No vacancies');
    END IF;
END; /
```

11.

```
DECLARE
    CURSOR emp_cursor IS
        SELECT employee_id, first_name, job_id, hire_date, salary FROM
        employees;
BEGIN
    FOR emp IN emp_cursor LOOP
        DBMS_OUTPUT.PUT_LINE('ID: ' || emp.employee_id || ', Name: ' ||
        emp.first_name || ', Job: ' || emp.job_id || ', Hire Date: ' || emp.hire_date ||
        ', Salary: ' || emp.salary);
    END LOOP;
END; /
```

12.

```
DECLARE
    CURSOR emp_dept_cursor IS
        SELECT e.employee_id, e.first_name, d.department_name
        FROM employees e
        JOIN departments d ON e.department_id = d.department_id;
BEGIN
    FOR emp IN emp_dept_cursor LOOP
    DBMS_OUTPUT.PUT_LINE('ID: ' || emp.employee_id || ', Name: ' ||
    emp.first_name || ', Dept: ' || emp.department_name);
    END LOOP;
END; /
```

13.

```
DECLARE
    CURSOR job_cursor IS
        SELECT job_id, job_title, min_salary FROM jobs;
BEGIN
    FOR job IN job_cursor LOOP
        DBMS_OUTPUT.PUT_LINE('Job ID: ' || job.job_id || ', Title: ' || job.job_title
        || ', Min Salary: ' || job.min_salary);
    END LOOP;
END; /
```

14.

```
DECLARE
    CURSOR job_hist_cursor IS
        SELECT employee_id, start_date FROM job_history;
BEGIN
    FOR job_hist IN job_hist_cursor LOOP
        DBMS_OUTPUT.PUT_LINE('Employee ID: ' || job_hist.employee_id || ',
        Start Date: ' || job_hist.start_date);
    END LOOP;
END; /
```

15.

```
DECLARE
        CURSOR job_hist_cursor IS
                SELECT employee_id, end_date FROM job_history;
BEGIN
        FOR job_hist IN job_hist_cursor LOOP
                DBMS_OUTPUT.PUT_LINE('Employee ID: ' || job_hist.employee_id || ',
                End Date: ' || job_hist.end_date);
        END LOOP;
END; /
```

Ex. No.: 12

WORKING WITH CURSOR, PROCEDURES AND FUNCTIONS

1.

```
CREATE OR REPLACE FUNCTION factorial(n NUMBER) RETURN NUMBER IS
        result NUMBER := 1;
BEGIN
        IF n < 0 THEN
                RETURN NULL;
        ELSIF n = 0 THEN
                RETURN 1;
        ELSE
                FOR i IN 1..n LOOP
                        result := result * i;
                END LOOP;
        END IF;
        RETURN result;
END factorial; /

DECLARE
        num NUMBER := 5;
        fact NUMBER;
BEGIN
        fact := factorial(num);
        DBMS_OUTPUT.PUT_LINE('Factorial of ' || num || ' is: ' || fact);
END; /
```

2. Initial:

```
CREATE TABLE books (
   book_id NUMBER PRIMARY KEY,
   title VARCHAR2(100),
   author VARCHAR2(100),
   genre VARCHAR2(50),
   publication_year NUMBER
); /


BEGIN
        INSERT INTO books VALUES (1, '1984', 'George Orwell', 'Dystopian', 1949);
        INSERT INTO books VALUES (2, 'To Kill a Mockingbird', 'Harper Lee', 'Fiction',
        1960);
        INSERT INTO books VALUES (3, 'The Great Gatsby', 'F. Scott Fitzgerald', 'Classic',
        1925);
        INSERT INTO books VALUES (4, 'Moby-Dick', 'Herman Melville', 'Adventure',
        1851);
        INSERT INTO books VALUES (5, 'Pride and Prejudice', 'Jane Austen', 'Romance',
        1813);
END; /
```

2.

```
CREATE OR REPLACE PROCEDURE get_book_info (
   p_book_id IN NUMBER,
   p_title IN OUT VARCHAR2,
   p_author OUT VARCHAR2,
   p_genre OUT VARCHAR2,
   p_publication_year OUT NUMBER
) IS
BEGIN
        SELECT title, author, genre, publication_year
        INTO p_title, p_author, p_genre, p_publication_year
        FROM books
        WHERE book_id = p_book_id;
```

```
EXCEPTION
        WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No book found with ID: ' || p_book_id);
END get_book_info; /

DECLARE
        book_id NUMBER := 3;
        title VARCHAR2(100) := 'Default Title';
        author VARCHAR2(100);
        genre VARCHAR2(50);
        publication_year NUMBER;
BEGIN
        get_book_info(book_id, title, author, genre, publication_year);
        DBMS_OUTPUT.PUT_LINE('Title: ' || title);
        DBMS_OUTPUT.PUT_LINE('Author: ' || author);
        DBMS_OUTPUT.PUT_LINE('Genre: ' || genre);
        DBMS_OUTPUT.PUT_LINE('Publication Year: ' || publication_year);
END; /
```

Ex. No.: 13

WORKING WITH TRIGGER

Initial:

```
CREATE TABLE orders (
   order_id NUMBER PRIMARY KEY,
   item_id NUMBER,
   quantity NUMBER,
   order_date DATE,
   running_total NUMBER,
   user_id NUMBER,
   FOREIGN KEY (item_id) REFERENCES items(item_id)
);

INSERT INTO orders (order_id, item_id, quantity, order_date, running_total, user_id)
VALUES (1, 1, 20, SYSDATE, 20, 101);
INSERT INTO orders (order_id, item_id, quantity, order_date, running_total, user_id)
VALUES (2, 2, 30, SYSDATE, 50, 102);

CREATE TABLE items (
   item_id NUMBER PRIMARY KEY,
   item_name VARCHAR2(50),
   stock_level NUMBER,
   pending_orders NUMBER DEFAULT 0
);

INSERT INTO items (item_id, item_name, stock_level, pending_orders)
VALUES (1, 'Item A', 100, 0);
INSERT INTO items (item_id, item_name, stock_level, pending_orders)
VALUES (2, 'Item B', 50, 0);
INSERT INTO items (item_id, item_name, stock_level, pending_orders)
VALUES (3, 'Item C', 150, 0);
```

```
CREATE  TABLE  audit_log (
   log_id NUMBER PRIMARY KEY,
   table_name VARCHAR2(50),
   operation VARCHAR2(10),
   change_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   user_id  NUMBER,
   details VARCHAR2(200)
);

CREATE SEQUENCE audit_log_seq
START WITH 1
INCREMENT BY 1;
```

1.

```
CREATE OR REPLACE TRIGGER prevent_parent_delete
BEFORE DELETE ON items
FOR EACH ROW
DECLARE
        child_count NUMBER;
BEGIN
        SELECT COUNT(*) INTO child_count FROM orders
        WHERE item_id = :OLD.item_id;

        IF child_count > 0 THEN
                RAISE_APPLICATION_ERROR(-20001, 'Cannot delete item; dependent
                orders exist.');
        END IF;
END; /
```

2.

```
CREATE OR REPLACE TRIGGER check_for_duplicates
BEFORE INSERT OR UPDATE ON orders
FOR EACH ROW
DECLARE
    duplicate_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO duplicate_count FROM orders
    WHERE item_id = :NEW.item_id AND order_id != :NEW.order_id;

    IF duplicate_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Duplicate item entry found in
        orders.');
    END IF;
END; /
```

3.

```
CREATE OR REPLACE TRIGGER restrict_insertion
BEFORE INSERT ON orders
FOR EACH ROW
DECLARE
    total_quantity NUMBER;
BEGIN
    SELECT SUM(quantity) INTO total_quantity FROM orders;
    IF (total_quantity + :NEW.quantity) > 500 THEN
        RAISE_APPLICATION_ERROR(-20003, 'Cannot insert order; total
        quantity exceeds threshold.');
    END IF;
END; /
```

4.

```
CREATE OR REPLACE TRIGGER log_changes

AFTER UPDATE ON orders

FOR EACH ROW

BEGIN

    INSERT INTO audit_log (log_id, table_name, operation, user_id, details) VALUES
    (audit_log_seq.NEXTVAL, 'orders', 'UPDATE', :NEW.user_id, 'Order ' ||
    :NEW.order_id || ' changed from ' || :OLD.quantity || ' to ' || :NEW.quantity );

END; /
```

5.

```
CREATE OR REPLACE TRIGGER log_user_activity

AFTER INSERT OR DELETE OR UPDATE ON orders

FOR EACH ROW

BEGIN

    INSERT INTO audit_log (log_id, table_name, operation, user_id, details) VALUES
    (audit_log_seq.NEXTVAL, 'orders',

        CASE

            WHEN INSERTING THEN 'INSERT'

            WHEN UPDATING THEN 'UPDATE'

            WHEN DELETING THEN 'DELETE'

        END,

    NVL(:NEW.user_id, :OLD.user_id), 'User action recorded on order ' ||
    NVL(:NEW.order_id, :OLD.order_id));

END; /
```

7.

```
CREATE OR REPLACE TRIGGER update_running_total

AFTER INSERT ON orders

FOR EACH ROW

BEGIN

    UPDATE orders SET running_total = (SELECT SUM(quantity) FROM orders)
    WHERE order_id = :NEW.order_id;

END; /
```

8.

```
CREATE OR REPLACE TRIGGER validate_item_availability
BEFORE INSERT ON orders
FOR EACH ROW
DECLARE
    available_stock NUMBER;
BEGIN
    SELECT stock_level - pending_orders INTO available_stock FROM items
    WHERE item_id = :NEW.item_id;

    IF :NEW.quantity > available_stock THEN
        RAISE_APPLICATION_ERROR(-20004, 'Insufficient stock available for the
        order.');
    END IF;

    UPDATE items SET pending_orders = pending_orders + :NEW.quantity
    WHERE item_id = :NEW.item_id;
END; /
```

Week 14:

MongoDB

Part 1 - Restaurants:

1.
```
db.restaurants.find(
  {
    $or: [
      { cuisine: { $nin: ["American", "Chinese"] } },
      { name: /^Wil/ }
    ]
  },
  { restaurant_id: 1, name: 1, borough: 1, cuisine: 1 }
)
```

2.
```
db.restaurants.find(
  {
    grades: {
      $elemMatch: {
        grade: "A",
        score: 11,
        date: ISODate("2014-08-11T00:00:00Z")
      }
    }
  },
  { restaurant_id: 1, name: 1, grades: 1 }
)
```

3.
```
db.restaurants.find(
  {
    "grades.1.grade": "A",
    "grades.1.score": 9,
    "grades.1.date": ISODate("2014-08-11T00:00:00Z")
  },
  { restaurant_id: 1, name: 1, grades: 1 }
)
```

4.
```
db.restaurants.find(
  { "address.coord.1": { $gt: 42, $lte: 52 } },
  { restaurant_id: 1, name: 1, address: 1, "address.coord": 1 }
)
```

5.
```
db.restaurants.find().sort({ name: 1 })
```

6.
```
db.restaurants.find().sort({ name: -1 })
```

7.
```
db.restaurants.find().sort({ cuisine: 1, borough: -1 })
```

8.
```
db.restaurants.find({ "address.street": { $exists: true } })
```

9.
```
db.restaurants.find({ "address.coord": { $type: "double" } })
```

10.
```
db.restaurants.find(
  { "grades.score": { $mod: [7, 0] } },
  { restaurant_id: 1, name: 1, grades: 1 }
)
```

11.
```
db.restaurants.find(
  { name: /mon/i },
  { name: 1, borough: 1, "address.coord": 1, cuisine: 1 }
)
```

12.
```
db.restaurants.find(
  { name: /^Mad/ },
  { name: 1, borough: 1, "address.coord": 1, cuisine: 1 }
)
```

13.
```
db.restaurants.find({ "grades.score": { $lt: 5 } })
```

14.
```
db.restaurants.find({ "grades.score": { $lt: 5 }, borough: "Manhattan" })
```

15.
```
db.restaurants.find({ "grades.score": { $lt: 5 }, borough: { $in: ["Manhattan", "Brooklyn"] } })
```

16.
```
db.restaurants.find(
  { "grades.score": { $lt: 5 }, borough: { $in: ["Manhattan", "Brooklyn"] }, cuisine: { $ne:
"American" } }
)
```

17.
```
db.restaurants.find(
   { "grades.score": { $lt: 5 }, borough: { $in: ["Manhattan", "Brooklyn"] }, cuisine: { $nin:
["American", "Chinese"] } }
)
```

18.
```
db.restaurants.find({
    grades: {
        $all: [
            { $elemMatch: { score: 2 } },
            { $elemMatch: { score: 6 } }
        ]
    }
})
```

19.
```
db.restaurants.find({
    grades: {
        $all: [
            { $elemMatch: { score: 2 } },
            { $elemMatch: { score: 6 } }
        ]
    },
    borough: "Manhattan"
})
```

20.
```
db.restaurants.find({
    grades: {
        $all: [
            { $elemMatch: { score: 2 } },
            { $elemMatch: { score: 6 } }
        ]
    },
    borough: { $in: ["Manhattan", "Brooklyn"] }
})
```

21.
```
db.restaurants.find({
    grades: {
        $all: [
            { $elemMatch: { score: 2 } },
            { $elemMatch: { score: 6 } }
        ]
    },
    borough: { $in: ["Manhattan", "Brooklyn"] },
    cuisine: { $ne: "American" }
})
```

22.
```
db.restaurants.find({
    grades: {
        $all: [
            { $elemMatch: { score: 2 } },
            { $elemMatch: { score: 6 } }
        ]
    },
    borough: { $in: ["Manhattan", "Brooklyn"] },
    cuisine: { $nin: ["American", "Chinese"] }
})
```

23.
```
.db.restaurants.find({
    grades: { $elemMatch: { score: { $in: [2, 6] } } }
})
```

Part 2 - Movies:

1.
```
db.movies.find({ year: 1893 })
```

2.
```
db.movies.find({ runtime: { $gt: 120 } })
```

3.
```
db.movies.find({ genres: "Short" })
```

4.
```
db.movies.find({ directors: "William K.L. Dickson" })
```

5.
```
db.movies.find({ countries: "USA" })
```

6.
```
db.movies.find({ rated: "UNRATED" })
```

7.
```
db.movies.find({ "imdb.votes": { $gt: 1000 } })
```

8.
```
db.movies.find({ "imdb.rating": { $gt: 7 } })
```

9.
```
db.movies.find({ "tomatoes.viewer.rating": { $gt: 4 } })
```

10.
```
db.movies.find({ "awards.wins": { $gt: 0 } })
```

11.
```
db.movies.find({
    "awards": { $exists: true, $ne: null }
})
```

12.
```
db.movies.find({
    "awards.nominations": { $gte: 1 }
}, {
    title: 1,
    languages: 1,
    released: 1,
    directors: 1,
    writers: 1,
    awards: 1,
    year: 1,
    genres: 1,
    runtime: 1,
    cast: 1,
    countries: 1
})
```

13.

```
db.movies.find({
    cast: "Charles Kayser"
}, {
    title: 1,
    languages: 1,
    released: 1,
    directors: 1,
    writers: 1,
    awards: 1,
    year: 1,
    genres: 1,
    runtime: 1,
    cast: 1,
    countries: 1
})
```

14.

```
db.movies.find({
    released: new Date("1893-05-09")
}, {
    title: 1,
    languages: 1,
    released: 1,
    directors: 1,
    writers: 1,
    countries: 1
})
```

15.

```
db.movies.find({
    title: /scene/i
}, {
    title: 1,
    languages: 1,
    released: 1,
    directors: 1,
    writers: 1,
    countries: 1
})
```

Week 15:

OTHER DATABASE OBJECTS

1.

```
CREATE SEQUENCE DEPT_ID_SEQ
INCREMENT BY 10
START WITH 200
MAXVALUE 1000
NOCYCLE;
```

2.

```
SELECT sequence_name, max_value, increment_by, last_number
FROM user_sequences;
```

3.

```
INSERT INTO DEPT (ID, DEPARTMENT_NAME)
VALUES (DEPT_ID_SEQ.NEXTVAL, 'Education');

INSERT INTO DEPT (ID, DEPARTMENT_NAME)
VALUES (DEPT_ID_SEQ.NEXTVAL, 'Administration');

SELECT * FROM DEPT;
```

4.

```
CREATE INDEX emp_dept_id_idx
ON EMP(DEPT_ID);
```

5.

```
SELECT ic.index_name, ic.column_name, ic.column_position AS col_pos, ix.uniqueness
FROM user_indexes ix
JOIN user_ind_columns ic ON ic.index_name = ix.index_name
WHERE ic.table_name = 'EMP';
```

Week 16:

CONTROLLING USER ACCESS

1.
The user should be given the `CREATE SESSION` privilege. This is a **system privilege**.

2.
The user should be given the `CREATE TABLE` privilege.
3.
Only the owner of the table (the user who created the table) can pass along privileges to other users on that table.

4.
You should create a **role** with the necessary privileges and then grant this role to each user.

5.
ALTER USER username IDENTIFIED BY new_password;

6.
GRANT SELECT ON departments TO other_user;

GRANT SELECT ON departments TO original_user;

7.
SELECT * FROM departments;

8.
INSERT INTO departments (department_id, department_name) VALUES (500, 'Education');

INSERT INTO departments (department_id, department_name) VALUES (510, 'Human Resources');

9.
SELECT * FROM other_team_user.departments;

10.
REVOKE SELECT ON departments FROM other_team_user;

11.
DELETE FROM departments WHERE department_id = 500;
COMMIT;

DELETE FROM departments WHERE department_id = 510;
COMMIT;