

RAJALAKSHMI ENGINEERING COLLEGE
RAJALAKSHMI NAGAR, THANDALAM – 602 105



RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

**CS23332 DATABASE MANAGEMENT
SYSTEMS LAB**

Laboratory Record Note Book

Name	Aravinthaa S
Year / Branch / Section	II / B.E.CSE / A
University Register No	2116230701032
College Roll No	230701032
Semester	III
Academic Year	2024 - 2025

CS23332 DATABASE MANAGEMENT SYSTEMS

NAME	ARAVINTHAA S
ROLL NO	230701032
DEPT	B.E.CSE
SEC	A

Ex.No.: 1	CREATION OF BASE TABLE AND DML OPERATIONS	
Date:	31 / 07 /24	

1. Create MY_EMPLOYEE table with the following structure

NAME	NULL?	TYPE
ID	Not null	Number(4)
Last_name		Varchar(25)
First_name		Varchar(25)
Userid		Varchar(25)
Salary		Number(9,2)

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

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MY_EMPLOYEE	ID	NUMBER	-	4	0	-	-	-	-
	LAST_NAME	VARCHAR2	25	-	-	-	✓	-	-
	FIRST_NAME	VARCHAR2	25	-	-	-	✓	-	-
	USERID	VARCHAR2	25	-	-	-	✓	-	-
	SALARY	NUMBER	-	9	2	-	✓	-	-

2. Add the first and second rows data to MY_EMPLOYEE table from the following sample data.

ID	Last_name	First_name	Userid	salary
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	Cnewman	750
5	Ropebur	Audrey	aropebur	1550

Begin

```
INSERT INTO MY_EMPLOYEE VALUES (1, 'Patel', 'Ralph', 'rpatel', 895);
INSERT INTO MY_EMPLOYEE VALUES (2, 'Dancs', 'Betty', 'bdancs', 860);
End;
```

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860

3. Display the table with values.

```
Select * from My_Employee;
```

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860

4. Populate the next two rows of data from the sample data. Concatenate the first letter of the first_name with the first seven characters of the last_name to produce Userid.

Begin

```
INSERT INTO MY_EMPLOYEE (ID, Last_name, First_name, Userid, Salary)
VALUES (3, 'Biri', 'Ben', SUBSTR('Biri', 1, 1) || SUBSTR('Biri', 1, 7), 1100);
INSERT INTO MY_EMPLOYEE (ID, Last_name, First_name, Userid, Salary)
VALUES (4, 'Newman', 'Chad', SUBSTR('Newman', 1, 1) || SUBSTR('Newman', 1, 7), 750);
End;
```

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	BBiri	1100
4	Newman	Chad	NNorman	750

5. Delete Betty dancs from MY_EMPLOYEE table.

```
DELETE FROM MY_EMPLOYEE WHERE Last_name = 'Dancs';
```

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
3	Biri	Ben	BBiri	1100
4	Newman	Chad	NNorman	750

6. Empty the fourth row of the emp table.

```
DELETE FROM MY_EMPLOYEE WHERE ID = 4;
```

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
3	Biri	Ben	BBiri	1100

7. Make the data additions permanent.

COMMIT;

Statement processed.

0.01 seconds

8. Change the last name of employee 3 to Drexler.

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

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
3	Drexler	Ben	BBiri	1100

9. Change the salary to 1000 for all the employees with a salary less than 900.

UPDATE MY_EMPLOYEE SET Salary = 1000 WHERE Salary < 900;

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
3	Drexler	Ben	BBiri	1100

Ex.No.: 2		DATA MANIPULATIONS
Date:	02 / 08 / 24	

Create the following tables with the given structure.

EMPLOYEES TABLE

NAME	NULL?	TYPE
Employee_id	Not null	Number(6)
First_Name		Varchar(20)
Last_Name	Not null	Varchar(25)
Email	Not null	Varchar(25)
Phone_Number		Varchar(20)
Hire_date	Not null	Date
Job_id	Not null	Varchar(10)
Salary		Number(8,2)
Commission_pct		Number(2,2)
Manager_id		Number(6)
Department_id		Number(4)

(a) Find out the employee id, names, salaries of all the employees

```
SELECT Employee_id, First_name, Last_name, Salary FROM EMPLOYEES;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
101	John	Doe	6000
102	Jane	Smith	4500
103	Mike	Johnson	7200
104	Emily	Davis	5000
105	Robert	Miller	6200
106	Sophia	Wilson	5600
107	Daniel	Brown	5800
108	Lisa	Taylor	4600
109	Kevin	Anderson	7100
110	Rachel	Thomas	5300

(b) List out the employees who works under manager 100

```
SELECT Employee_id, First_name, Last_name FROM EMPLOYEES  
WHERE Manager_id = 100;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
101	John	Doe

(c) Find the names of the employees who have a salary greater than or equal to 4800

SELECT First_name, Last_name FROM EMPLOYEES WHERE Salary >= 4800;

FIRST_NAME	LAST_NAME
John	Doe
Mike	Johnson
Emily	Davis
Robert	Miller
Sophia	Wilson
Daniel	Brown
Kevin	Anderson
Rachel	Thomas

(d) List out the employees whose last name is AUSTIN'

SELECT Employee_id, First_name, Last_name FROM EMPLOYEES WHERE Last_name = 'AUSTIN';

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
109	Kevin	AUSTIN

(e) Find the names of the employees who works in departments 60,70 and 80

```
SELECT First_name, Last_name FROM EMPLOYEES WHERE Department_id IN (60, 70, 80);
```

FIRST_NAME	LAST_NAME
John	Doe
Jane	Smith
Mike	Johnson
Emily	Davis
Robert	Miller
Sophia	Wilson
Daniel	Brown
Lisa	Taylor
Kevin	AUSTIN
Rachel	Thomas

(f) Display the unique Manager_Id.

```
SELECT DISTINCT Manager_id FROM EMPLOYEES;
```

MANAGER_ID
100
102
101
104
105
103

Create an Emp table with the following fields: (EmpNo, EmpName, Job, Basic, DA, HRA, PF, GrossPay, NetPay) (Calculate DA as 30% of Basic and HRA as 40% of Basic)

(a) Insert Five Records and calculate GrossPay and NetPay.

```
INSERT INTO EMP (EmpNo, EmpName, Job, Basic, DA, HRA, PF, GrossPay, NetPay)
VALUES (1, 'John Doe', 'Manager', 50000, 0.30 * 50000, -- DA as 30% of Basic
       0.40 * 50000, -- HRA as 40% of Basic, 0.12 * 50000, -- PF as 12% of Basic
       50000 + (0.30 * 50000) + (0.40 * 50000), -- GrossPay (50000 + (0.30 * 50000) + (0.40 *
       50000)) - (0.12 * 50000) -- NetPay
     );
```

```
INSERT INTO EMP (EmpNo, EmpName, Job, Basic, DA, HRA, PF, GrossPay, NetPay)
VALUES (2, 'Jane Smith', 'Clerk', 30000, 0.30 * 30000, 0.40 * 30000,
       0.12 * 30000,
       30000 + (0.30 * 30000) + (0.40 * 30000),
       (30000 + (0.30 * 30000) + (0.40 * 30000)) - (0.12 * 30000)
     );
```

```
INSERT INTO EMP (EmpNo, EmpName, Job, Basic, DA, HRA, PF, GrossPay, NetPay)
VALUES (3, 'Mike Johnson', 'Salesman', 40000,
       0.30 * 40000,
       0.40 * 40000,
       0.12 * 40000,
       40000 + (0.30 * 40000) + (0.40 * 40000),
       (40000 + (0.30 * 40000) + (0.40 * 40000)) - (0.12 * 40000)
     );
```

```
INSERT INTO EMP (EmpNo, EmpName, Job, Basic, DA, HRA, PF, GrossPay, NetPay)
VALUES (4, 'Emily Davis', 'Accountant', 35000,
       0.30 * 35000,
       0.40 * 35000,
       0.12 * 35000,
       35000 + (0.30 * 35000) + (0.40 * 35000),
       (35000 + (0.30 * 35000) + (0.40 * 35000)) - (0.12 * 35000)
     );
```

```
INSERT INTO EMP (EmpNo, EmpName, Job, Basic, DA, HRA, PF, GrossPay, NetPay)
VALUES (5, 'Robert Miller', 'Clerk', 25000,
       0.30 * 25000,
       0.40 * 25000,
       0.12 * 25000,
       25000 + (0.30 * 25000) + (0.40 * 25000),
       (25000 + (0.30 * 25000) + (0.40 * 25000)) - (0.12 * 25000)
     );
```

EMPNO	EMPNAME	JOB	BASIC	DA	HRA	PF	GROSSPAY	NETPAY
1	John Doe	Manager	50000	15000	20000	6000	85000	79000
2	Jane Smith	Clerk	30000	9000	12000	3600	51000	47400
3	Mike Johnson	Salesman	40000	12000	16000	4800	68000	63200
4	Emily Davis	Accountant	35000	10500	14000	4200	59500	55300
5	Robert Miller	Clerk	25000	7500	10000	3000	42500	39500

(b) Display the employees whose Basic is lowest in each department.

```
SELECT EmpNo, EmpName, Job, Basic FROM EMP E1 WHERE Basic = (
    SELECT MIN(Basic) FROM EMP E2 WHERE E2.Job = E1.Job);
```

EMPNO	EMPNAME	JOB	BASIC
1	John Doe	Manager	50000
3	Mike Johnson	Salesman	40000
4	Emily Davis	Accountant	35000
5	Robert Miller	Clerk	25000

(c) If Net Pay is less than 50000, display employee number, name and net pay

```
SELECT EmpNo, EmpName, NetPay FROM EMP WHERE NetPay < 50000;
```

EMPNO	EMPNAME	NETPAY
2	Jane Smith	47400
5	Robert Miller	39500

DEPARTMENT TABLE

NAME	NULL?	TYPE
Dept_id	Not null	Number(6)
Dept_name	Not null	Varchar(20)
Manager_id		Number(6)
Location_id		Number(4)

JOB GRADE TABLE

NAME	NULL?	TYPE
Grade_level		Varchar(2)
Lowest_sal		Number
Highest_sal		Number

LOCATION TABLE

NAME	NULL?	TYPE
Location_id	Not null	Number(4)
St_addr		Varchar(40)
Postal_code		Varchar(12)
City	Not null	Varchar(30)
State_province		Varchar(25)
Country_id		Char(2)

1. Create the DEPT table based on the DEPARTMENT following the table instance chart below. Confirm that the table is created.

Column name	ID	NAME
Key Type		
Nulls/Unique		
FK table		
FK column		
Data Type	Number	Varchar2
Length	7	25

```
CREATE TABLE DEPT (Dept_id NUMBER(6) NOT NULL, Dept_name VARCHAR2(20)
NOT NULL, Manager_id NUMBER(6), Location_id NUMBER(4), CONSTRAINT
my_dept_id_pk PRIMARY KEY (Dept_id));
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPT	DEPT_ID	NUMBER	-	6	0	1	-	-	-
	DEPT_NAME	VARCHAR2	20	-	-	-	-	-	-
	MANAGER_ID	NUMBER	-	6	0	-	✓	-	-
	LOCATION_ID	NUMBER	-	4	0	-	✓	-	-

2. Create the EMP table based on the following instance chart. Confirm that the table is created.

Column name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				
Nulls/Unique				
FK table				
FK column				
Data Type	Number	Varchar2	Varchar2	Number
Length	7	25	25	7

```
CREATE TABLE EMP (EmpNo NUMBER(7) PRIMARY KEY,Last_name VARCHAR2(25)
NOT NULL,First_name VARCHAR2(25),Dept_id NUMBER(7),
CONSTRAINT my_emp_dept_id_fk FOREIGN KEY (Dept_id) REFERENCES
DEPT(Dept_id));
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMP	EMPNO	NUMBER	-	7	0	1	-	-	-
	LAST_NAME	VARCHAR2	25	-	-	-	-	-	-
	FIRST_NAME	VARCHAR2	25	-	-	-	✓	-	-
	DEPT_ID	NUMBER	-	7	0	-	✓	-	-
1 - 4									

3. Modify the EMP table to allow for longer employee last names. Confirm the modification.(Hint: Increase the size to 50)

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

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMP	EMPNO	NUMBER	-	7	0	1	-	-	-
	LAST_NAME	VARCHAR2	50	-	-	-	-	-	-
	FIRST_NAME	VARCHAR2	25	-	-	-	✓	-	-
	DEPT_ID	NUMBER	-	7	0	-	✓	-	-

4. Create the EMPLOYEES2 table based on the structure of EMPLOYEES table. Include Only the Employee_id, First_name, Last_name, Salary and Dept_id coloumns. Name the columns Id, First_name, Last_name, salary and Dept_id respectively.

```
CREATE TABLE EMPLOYEES2 (Id NUMBER(6) PRIMARY
KEY,First_name VARCHAR2(20),Last_name VARCHAR2(25),
Salary NUMBER(8,2),Dept_id NUMBER(4));
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMPLOYEES2	ID	NUMBER	-	6	0	1	-	-	-
	FIRST_NAME	VARCHAR2	20	-	-	-	✓	-	-
	LAST_NAME	VARCHAR2	25	-	-	-	✓	-	-
	SALARY	NUMBER	-	8	2	-	✓	-	-
	DEPT_ID	NUMBER	-	4	0	-	✓	-	-

5. Drop the EMP Table

```
DROP TABLE EMP;
```

Table dropped.

6. Rename the EMPLOYEES2 table as EMP.

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

Table altered.

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMP	ID	NUMBER	-	6	0	1	-	-	-
	FIRST_NAME	VARCHAR2	20	-	-	-	✓	-	-
	LAST_NAME	VARCHAR2	25	-	-	-	✓	-	-
	SALARY	NUMBER	-	8	2	-	✓	-	-
	DEPT_ID	NUMBER	-	4	0	-	✓	-	-

7. Add a comment on DEPT and EMP tables. Confirm the modification by describing the table.

```
COMMENT ON TABLE DEPT IS 'This table contains department information.';  
COMMENT ON TABLE EMP IS 'This table contains employee information.';
```

TABLE_NAME	TABLE_TYPE	COMMENTS
DEPT	TABLE	This table contains department information.
EMP	TABLE	This table contains employee information.
DEMO_CUSTOMERS	TABLE	-
MY_EMPLOYEE	TABLE	-
APEX\$_ACL	TABLE	-
STUDENTS	TABLE	-
APEX\$_WS_TAGS	TABLE	-
APEX\$_WS_WEBPG_SECTIONS	TABLE	-
APEX\$_WS_LINKS	TABLE	-
MANAGER	TABLE	-

8. Drop the First_name column from the EMP table and confirm it.

```
ALTER TABLE EMP DROP COLUMN First_name;
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMP	ID	NUMBER	-	6	0	1	-	-	-
	LAST_NAME	VARCHAR2	25	-	-	-	✓	-	-
	SALARY	NUMBER	-	8	2	-	✓	-	-
	DEPT_ID	NUMBER	-	4	0	-	✓	-	-

Ex.No.: 3	WRITING BASIC SQL SELECT STATEMENTS
Date:	09 / 08 / 24

Find the Solution for the following:

True OR False

1. The following statement executes successfully.

Identify the Errors

```
SELECT employee_id, last_name
      sal*12 ANNUAL SALARY
   FROM employees;
```

False ->Corrected Query and Output

Select employee_id, last_name, salary*12 AS "Annual Salary" from Employees;

EMPLOYEE_ID	LAST_NAME	Annual Salary
101	Doe	72000
102	Smith	54000
103	Johnson	86400
104	Davis	60000
105	Miller	74400
106	Wilson	67200
107	Brown	69600
108	Taylor	55200
109	AUSTIN	85200
110	Thomas	63600

2. Show the structure of departments the table. Select all the data from it.

DESC department;

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPARTMENT	DEPT_ID	NUMBER	-	6	0	-	-	-	-
	DEPT_NAME	VARCHAR2	20	-	-	-	-	-	-
	MANAGER_ID	NUMBER	-	6	0	-	✓	-	-
	LOCATION_ID	NUMBER	-	4	0	-	✓	-	-

Select * from Department;

DEPT_ID	DEPT_NAME	MANAGER_ID	LOCATION_ID
10	Admin	101	1000
20	Marketing	102	1001
30	Purchasing	103	1002
40	HR	104	1003
50	IT	105	1004
60	Sales	106	1005
70	Customer Service	107	1006
80	Accounting	108	1007
90	R&D	109	1008
100	Legal	110	1009

3. Create a query to display the last name, job code, hire date, and employee number for each employee, with employee number appearing first.

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

EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE
101	Doe	IT_PROG	01/15/2020
102	Smith	HR_REP	02/20/2019
103	Johnson	SA_MAN	05/30/2021
104	Davis	AC_ACCOUNT	10/10/2020
105	Miller	MK_MAN	07/25/2018
106	Wilson	SA_REP	03/12/2022
107	Brown	IT_PROG	11/05/2017
108	Taylor	HR_REP	12/15/2019
109	AUSTIN	AC_MGR	08/22/2021
110	Thomas	MK_REP	04/01/2020

4. Provide an alias STARTDATE for the hire date.

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

EMPLOYEE_ID	LAST_NAME	JOB_ID	STARTDATE
101	Doe	IT_PROG	01/15/2020
102	Smith	HR_REP	02/20/2019
103	Johnson	SA_MAN	05/30/2021
104	Davis	AC_ACCOUNT	10/10/2020
105	Miller	MK_MAN	07/25/2018
106	Wilson	SA_REP	03/12/2022
107	Brown	IT_PROG	11/05/2017
108	Taylor	HR_REP	12/15/2019
109	AUSTIN	AC_MGR	08/22/2021
110	Thomas	MK_REP	04/01/2020

5. Create a query to display unique job codes from the employee table.

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

JOB_ID
IT_PROG
AC_ACCOUNT
AC_MGR
SA_MAN
MK_MAN
SA_REP
MK_REP
HR_REP

6. Display the last name concatenated with the job ID , separated by a comma and space, and name the column EMPLOYEE and TITLE.

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

EMPLOYEE and TITLE
Doe, IT_PROG
Smith, HR_REP
Johnson, SA_MAN
Davis, AC_ACCOUNT
Miller, MK_MAN
Wilson, SA_REP
Brown, IT_PROG
Taylor, HR_REP
AUSTIN, AC_MGR
Thomas, MK_REP

7. Create a query to display all the data from the employees table. Separate each column by a comma. Name the column THE_OUTPUT.

```
SELECT employee_id || ',' || last_name || ',' || job_id || ',' || hire_date AS THE_OUTPUT  
FROM employees;
```

THE_OUTPUT
101, Doe, IT_PROG, 01/15/2020
102, Smith, HR_REP, 02/20/2019
103, Johnson, SA_MAN, 05/30/2021
104, Davis, AC_ACCOUNT, 10/10/2020
105, Miller, MK_MAN, 07/25/2018
106, Wilson, SA_REP, 03/12/2022
107, Brown, IT_PROG, 11/05/2017
108, Taylor, HR_REP, 12/15/2019
109, AUSTIN, AC_MGR, 08/22/2021
110, Thomas, MK_REP, 04/01/2020

Ex.No.: 4		WORKING WITH CONSTRAINTS	
Date:	09 / 08 / 24		

1. Add a table-level PRIMARY KEY constraint to the EMP table on the ID column. The constraint should be named at creation. Name the constraint my_emp_id_pk.

```
CREATE TABLE EMP ( EmpNo NUMBER(7) PRIMARY KEY,
Last_name VARCHAR2(25) NOT NULL,First_name VARCHAR2(25));
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMP	ID	NUMBER	-	6	0	1	-	-	-
	LAST_NAME	VARCHAR2	25	-	-	-	✓	-	-
	SALARY	NUMBER	-	8	2	-	✓	-	-

2. Create a PRIMAY KEY constraint to the DEPT table using the ID colum. The constraint should be named at creation. Name the constraint my_dept_id_pk.

```
CREATE TABLE DEPT (Dept_id NUMBER(6) NOT NULL,Dept_name
VARCHAR2(20) NOT NULL,Manager_id NUMBER(6), Location_id
NUMBER(4),CONSTRAINT my_dept_id_pk PRIMARY KEY (Dept_id));
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPT	DEPT_ID	NUMBER	-	6	0	1	-	-	-
	DEPT_NAME	VARCHAR2	20	-	-	-	-	-	-
	MANAGER_ID	NUMBER	-	6	0	-	✓	-	-
	LOCATION_ID	NUMBER	-	4	0	-	✓	-	-

3. Add a column DEPT_ID to the EMP table. Add a foreign key reference on the EMP table that ensures that the employee is not assigned to nonexistent department. Name the constraint my_emp_dept_id_fk.

```
ALTER TABLE EMP ADD Dept_id NUMBER(6);
ALTER TABLE EMP ADD CONSTRAINT my_emp_dept_id_fk FOREIGN KEY
(Dept_id) REFERENCES DEPT (Dept_id);
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMP	EMPNO	NUMBER	-	7	0	1	-	-	-
	LAST_NAME	VARCHAR2	25	-	-	-	-	-	-
	FIRST_NAME	VARCHAR2	25	-	-	-	✓	-	-
	DEPT_ID	NUMBER	-	6	0	-	✓	-	-

4. Modify the EMP table. Add a COMMISSION column of NUMBER data type, precision 2, scale 2. Add a constraint to the commission column that ensures that a commission value is greater than zero.

```
ALTER TABLE EMP
ADD Commission NUMBER(8,2) CONSTRAINT commission_check CHECK
(Commission > 0);
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMP	EMPNO	NUMBER	-	7	0	1	-	-	-
	LAST_NAME	VARCHAR2	25	-	-	-	-	-	-
	FIRST_NAME	VARCHAR2	25	-	-	-	✓	-	-
	DEPT_ID	NUMBER	-	6	0	-	✓	-	-
	COMMISSION	NUMBER	-	8	2	-	✓	-	-

Ex.No.: 5	CREATING VIEWS	
Date:	21 / 08 / 24	

1. Create a view called EMPLOYEE_VU based on the employee numbers, employee names and department numbers from the EMPLOYEES table. Change the heading for the employee name to EMPLOYEE.

CREATE VIEW EMPLOYEE_VU AS SELECT employee_id, last_name AS EMPLOYEE, department_id FROM EMPLOYEES;

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMPLOYEE_VU	EMPLOYEE_ID	NUMBER	-	6	0	-	-	-	-
	EMPLOYEE	VARCHAR2	25	-	-	-	-	-	-
	DEPARTMENT_ID	NUMBER	-	4	0	-	✓	-	-

2. Display the contents of the EMPLOYEES_VU view.

```
SELECT * FROM EMPLOYEE_VU;
```

EMPLOYEE_ID	EMPLOYEE	DEPARTMENT_ID
101	Doe	60
102	Smith	70
103	Johnson	80
104	Davis	60
105	Miller	70
106	Wilson	80
107	Brown	60
108	Taylor	70
109	AUSTIN	80
110	Thomas	60

3. Select the view name and text from the USER_VIEWS data dictionary views.

```
SELECT view_name, text FROM USER_VIEWS WHERE view_name = 'EMPLOYEE_VU';
```

VIEW_NAME	TEXT
EMPLOYEE_VU	SELECT employee_id, last_name AS EMPLOYEE, department_id FROM EMPLOYEES

4. Using your EMPLOYEES_VU view, enter a query to display all employees names and department.

```
SELECT EMPLOYEE, department_id FROM EMPLOYEE_VU;
```

EMPLOYEE	DEPARTMENT_ID
Doe	60
Smith	70
Johnson	80
Davis	60
Miller	70
Wilson	80
Brown	60
Taylor	70
AUSTIN	80
Thomas	60

5. Create a view named DEPT50 that contains the employee number, employee last names and department numbers for all employees in department 50. Label the view columns EMPNO, EMPLOYEE and DEPTNO. Do not allow an employee to be reassigned to another department through the view.

```
CREATE OR REPLACE VIEW DEPT50 (EMPNO, EMPLOYEE, DEPTNO) AS
SELECT employee_id, last_name, department_id
FROM EMPLOYEES
WHERE department_id = 50
WITH CHECK OPTION;
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPT50	<u>EMPNO</u>	NUMBER	-	6	0	-	-	-	-
	<u>EMPLOYEE</u>	VARCHAR2	25	-	-	-	-	-	-
	<u>DEPTNO</u>	NUMBER	-	4	0	-	✓	-	-

6. Display the structure and contents of the DEPT50 view.

```
SELECT * FROM DEPT50;
```

EMPNO	EMPLOYEE	DEPTNO
101	Doe	50
103	Johnson	50
107	Brown	50
109	AUSTIN	50

7. Attempt to reassign Matos to department 80.

```
UPDATE DEPT50 SET DEPTNO = 80 WHERE EMPLOYEE = 'Matos';
```

ORA-01402: view WITH CHECK OPTION where-clause violation

8. Create a view called SALARY_VU based on the employee last names, department names, salaries, and salary grades for all employees. Use the Employees, DEPARTMENTS and JOB_GRADE tables. Label the column Employee,Department, salary, and Grade respectively.

```
CREATE VIEW SALARY_VU AS
SELECT e.last_name AS Employee,
       d.department_name AS Department,
       e.salary AS Salary,
       j.grade_level AS Grade
  FROM EMPLOYEES e
```

```

JOIN DEPARTMENTS d ON e.department_id = d.department_id
JOIN JOB_GRADE j ON e.salary BETWEEN j.lowest_sal AND j.highest_sal;

```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
SALARY_VU	EMPLOYEE	VARCHAR2	25	-	-	-	-	-	-
	DEPARTMENT	VARCHAR2	20	-	-	-	-	-	-
	SALARY	NUMBER	-	8	2	-	✓	-	-
	GRADE	VARCHAR2	2	-	-	-	✓	-	-

Ex.No.: 6		RESTRICTING AND SORTING DATA
Date:	23 / 08 / 24	

1. Create a query to display the last name and salary of employees earning more than 12000.

```
SELECT last_name, salary FROM employees WHERE salary > 12000;
```

LAST_NAME	SALARY
Smith	12500
Davis	15000
Wilson	13500
Brown	16000

2. Create a query to display the employee last name and department number for employee number 176.

```
SELECT last_name, department_id FROM employees WHERE employee_id = 176;
```

LAST_NAME	DEPARTMENT_ID
Smith	70

3. Create a query to display the last name and salary of employees whose salary is not in the range of 5000 and 12000. (hints: not between).

```
SELECT last_name, salary FROM employees WHERE salary NOT BETWEEN 5000 AND 12000;
```

LAST_NAME	SALARY
Smith	12500
Davis	15000
Wilson	13500
Brown	16000
Taylor	4600

4. Display the employee last name, job ID, and start date of employees hired between February 20,1998 and May 1,1998.order the query in ascending order by start date.(hints: between)

```
SELECT last_name, job_id, hire_date FROM employees WHERE hire_date BETWEEN '02-20-1998' AND '05-01-1998' ORDER BY hire_date ASC;
```

LAST_NAME	JOB_ID	HIRE_DATE
Johnson	SA_MAN	03/01/1998

5. Display the last name and department number of all employees in departments 20 and 50 in alphabetical order by name.(hints: in, orderby)

```
SELECT last_name, department_id FROM employees WHERE department_id IN (20, 50)
ORDER BY last_name ASC;
```

LAST_NAME	DEPARTMENT_ID
AUSTIN	50
Brown	50
Johnson	50
Matos	50

6. Display the last name and salary of all employees who earn between 5000 and 12000 and are in departments 20 and 50 in alphabetical order by name. Label the columns EMPLOYEE, MONTHLY SALARY respectively.(hints: between, in)

```
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 ASC;
```

EMPLOYEE	MONTHLY SALARY
AUSTIN	7100
Johnson	7200
Matos	6000

7. Display the last name and hire date of every employee who was hired in 1994.(hints: like)

```
SELECT last_name, hire_date FROM employees WHERE hire_date LIKE '%1994%';
```

LAST_NAME	HIRE_DATE
Matos	01/01/1994

8. Display the last name and job title of all employees who do not have a manager.(hints: is null)

```
SELECT last_name, job_id FROM employees WHERE manager_id IS NULL;
```

LAST_NAME	JOB_ID
Austin	AC_MGR

9. Display the last name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions.(hints: is not null,orderby)

```
SELECT last_name, salary, commission_pct FROM employees WHERE commission_pct IS NOT NULL ORDER BY salary DESC, commission_pct DESC;
```

LAST_NAME	SALARY	COMMISSION_PCT
Wilson	13500	.1
Johnson	7200	.15
Thomas	5300	.08

10. Display the last name of all employees where the third letter of the name is *a*.(hints:like)

```
SELECT last_name FROM employees WHERE last_name LIKE '__a%';
```

LAST_NAME
Brawn

11. Display the last name of all employees who have an *a* and an *e* in their last name.(hints: like)

```
SELECT last_name FROM employees WHERE last_name LIKE '%a%' AND last_name
```

LIKE '%e%';

LAST_NAME
Andrea

12. Display the last name and job and salary for all employees whose job is sales representative or stock clerk and whose salary is not equal to 2500 ,3500 or 7000.(hints:in,not in)

SELECT last_name, job_id, salary FROM employees WHERE job_id IN ('SA_REP', 'ST_CLERK') AND salary NOT IN (2500, 3500, 7000);

LAST_NAME	JOB_ID	SALARY
Wilson	SA_REP	13500

Ex.No.: 7	USING SET OPERATORS
Date:	28 / 08 / 24

1. The HR department needs a list of department IDs for departments that do not contain the job ID ST_CLERK. Use set operators to create this report.

```
SELECT department_id FROM departments MINUS SELECT department_id  
FROM employees WHERE job_id = 'ST_CLERK';
```

DEPARTMENT_ID
10
20
30
40
50
80
90
100

2. The HR department needs a list of countries that have no departments located in them. Display the country ID and the name of the countries. Use set operators to create this report.

```
SELECT country_id, country_name FROM countries MINUS SELECT country_id,  
country_name FROM departments;
```

CN	China
BR	Brazil

3. Produce a list of jobs for departments 10, 50, and 20, in that order. Display job ID and department ID using set operators.

```

SELECT job_id, department_id FROM employees WHERE department_id = 10
UNION
SELECT job_id, department_id FROM employees WHERE department_id = 50
UNION
SELECT job_id, department_id FROM employees WHERE department_id = 20;

```

JOB_ID	DEPARTMENT_ID
AC_ACCOUNT	20
AC_MGR	50
HR_REP	20
IT_PROG	10
IT_PROG	50
SA_MAN	50
ST_CLERK	10

4. Create a report that lists the employee IDs and job IDs of those employees who currently have a job title that is the same as their job title when they were initially hired by the company (that is, they changed jobs but have now gone back to doing their original job).

```
SELECT employee_id, job_id FROM employees  
INTERSECT  
SELECT employee_id, job_id FROM job_history;
```

EMPLOYEE_ID	JOB_ID
201	IT_PROG
202	HR_REP
203	SA_REP
204	IT_PROG
205	HR_REP
206	SA_REP
207	IT_PROG
208	SA_REP
209	IT_PROG
210	HR_REP

5. The HR department needs a report with the following specifications:

- Last name and department ID of all the employees from the EMPLOYEES table, regardless of whether or not they belong to a department.

- Department ID and department name of all the departments from the DEPARTMENTS table, regardless of whether or not they have employees working in them Write a compound query to accomplish this.

```
SELECT last_name, department_id FROM employees  
UNION  
SELECT department_name, department_id FROM departments;
```

Andrea	10
Austin	50
Brown	-
Clark	-
Silva	-
Smith	70
Tanaka	-
Taylor	20
Thomas	60
Wei	-
Wilson	80

Date:	04 / 09 / 24	
--------------	--------------	--

1. Write a query to display the last name, department number, and department name for all employees.

```
SELECT e.last_name, e.department_id, d.department_name FROM employees e JOIN
departments d ON e.department_id = d.department_id;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Miller	10	Admin
Andrea	10	Admin
Davis	20	ST_CLERK
Taylor	20	ST_CLERK
Matos	50	IT
Johnson	50	IT
Austin	50	IT
Thomas	60	ST_CLERK
Smith	70	Customer Service
Wilson	80	ST_CLERK

2. Create a unique listing of all jobs that are in department 80. Include the location of the department in the output.

```
SELECT DISTINCT e.job_id, d.location_id FROM employees e JOIN departments d ON
e.department_id = d.department_id WHERE e.department_id = 80;
```

JOB_ID	LOCATION_ID
SA_REP	1007

3. Write a query to display the employee last name, department name, location ID, and city of all employees who earn a commission

```
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;
```

LAST_NAME	DEPARTMENT_NAME	LOCATION_ID	CITY
Johnson	IT	1004	London
Thomas	ST_CLERK	1005	Sydney
Wilson	ST_CLERK	1007	Dubai

4. Display the employee last name and department name for all employees who have an a(lowercase) in their last names. P

```
SELECT e.last_name, d.department_name FROM employees e JOIN departments d ON e.department_id = d.department_id WHERE e.last_name LIKE '%a%';
```

LAST_NAME	DEPARTMENT_NAME
Matos	IT
Davis	ST_CLERK
Andrea	Admin
Taylor	ST_CLERK
Thomas	ST_CLERK

5. Write a query to display the last name, job, department number, and department name for all employees who work in Toronto.

```
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';
```

LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
Andrea	IT_PROG	10	Admin
Miller	ST_CLERK	10	Admin

6. Display the employee last name and employee number along with their manager's last name and manager number. Label the columns Employee, Emp#, Manager, and Mgr#, Respectively

```
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;
```

EMPLOYEE	EMP#	MANAGER	MGR#
Andrea	107	Matos	101
Davis	104	Matos	101
Smith	176	Matos	101
Wilson	106	Johnson	103
Thomas	110	Miller	105
Silva	210	-	-
Wei	209	-	-
Tanaka	208	-	-
Wilson	207	-	-
Miller	206	-	-

7. Modify lab4_6.sql to display all employees including King, who has no manager. Order the results by the employee number.

```
SELECT e.last_name, e.employee_id, m.last_name AS Manager FROM employees e LEFT JOIN employees m ON e.manager_id = m.employee_id ORDER BY e.employee_id;
```

LAST_NAME	EMPLOYEE_ID	MANAGER
Matos	101	-
Johnson	103	-
Davis	104	Matos
Miller	105	-
Wilson	106	Johnson
Andrea	107	Matos
Taylor	108	-
Austin	109	-
Thomas	110	Miller
Smith	176	Matos

8. Create a query that displays employee last names, department numbers, and all the employees who work in the same department as a given employee. Give each column an appropriate label

```
SELECT e1.last_name AS Employee, e2.last_name AS Colleague FROM employees e1 JOIN employees e2 ON e1.department_id = e2.department_id WHERE e1.employee_id = :employee_id;
```

EMPLOYEE	COLLEAGUE
Matos	Matos
Matos	Johnson
Matos	Austin

9. Show the structure of the JOB_GRADES table. Create a query that displays the name, job, department name, salary, and grade for all employees

DESC job_grades;

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
JOB_GRADES	GRADE_LEVEL	VARCHAR2	2	-	-	-	✓	-	-
	LOWEST_SAL	NUMBER	22	-	-	-	✓	-	-
	HIGHEST_SAL	NUMBER	22	-	-	-	✓	-	-
	DEPTNO	NUMBER	22	-	-	-	✓	-	-

SELECT e.last_name, e.job_id, d.department_name, e.salary, j.grade_level FROM employees
e JOIN departments d ON e.department_id = d.department_id JOIN job_grades j ON e.salary
BETWEEN j.lowest_sal AND j.highest_sal;

LAST_NAME	JOB_ID	DEPARTMENT_NAME	SALARY	GRADE_LEVEL
Davis	AC_ACCOUNT	ST_CLERK	15000	G2
Wilson	SA_REP	ST_CLERK	13500	G1
Smith	HR REP	Customer Service	12500	F2
Johnson	SA_MAN	IT	7200	D1
Austin	AC_MGR	IT	7100	D1
Miller	ST_CLERK	Admin	6200	C2
Matos	IT_PROG	IT	6000	C1
Thomas	ST_CLERK	ST_CLERK	5300	C1
Taylor	HR REP	ST_CLERK	4600	B2

10. Create a query to display the name and hire date of any employee hired after employee Davies.

SELECT last_name, hire_date FROM employees WHERE hire_date > (SELECT hire_date FROM employees WHERE last_name = 'Davies');

LAST_NAME	HIRE_DATE
Smith	02/20/2019
Johnson	03/01/1998
Davis	01/01/1998
Miller	07/25/2018
Wilson	03/12/2022
Andrea	11/05/2017
Taylor	12/15/2019
Austin	08/22/2021
Thomas	04/01/2020
Doe	10/10/2015

11. Display the names and hire dates for all employees who were hired before their managers, along with their manager's names and hire dates. Label the columns Employee, Emp Hired, Manager, and Mgr Hired, respectively.

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;

EMPLOYEE	EMP_HIRED	MANAGER	MGR_HIRED
Smith	02/20/2019	Matos	01/01/1994
Davis	01/01/1998	Matos	01/01/1994
Andrea	11/05/2017	Matos	01/01/1994
Wilson	03/12/2022	Johnson	03/01/1998
Thomas	04/01/2020	Miller	07/25/2018

Ex.No.: 9	SUB QUERIES
-----------	-------------

Date:	11 / 09 / 24	
--------------	--------------	--

1. The HR department needs a query that prompts the user for an employee last name. The query then displays the last name and hire date of any employee in the same department as the employee whose name they supply (excluding that employee). For example, if the user enters Zlotkey, find all employees who work with Zlotkey (excluding Zlotkey).

```
SELECT e.last_name, e.hire_date FROM employees e JOIN employees e2 ON
e.department_id = e2.department_id WHERE e2.last_name = :emp_name
AND e.employee_id != e2.employee_id;
```

LAST_NAME	HIRE_DATE
Johnson	03/01/1998
Austin	08/22/2021

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

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

EMPLOYEE_ID	LAST_NAME	SALARY
176	Smith	12500
106	Wilson	13500
104	Davis	15000
107	Andrea	16000

3. Write a query that displays the employee number and last name of all employees who work in a department with any employee whose last name contains a *u*.

```
SELECT DISTINCT e1.employee_id, e1.last_name FROM employees e1 JOIN employees e2 ON e1.department_id = e2.department_id WHERE e2.last_name LIKE '%u%';
```

EMPLOYEE_ID	LAST_NAME
101	Matos
103	Johnson
109	Austin

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

```
SELECT e.last_name, e.department_id, e.job_id FROM employees e  
JOIN departments d ON e.department_id = d.department_id WHERE d.location_id = 1700;
```

LAST_NAME	DEPARTMENT_ID	JOB_ID
Miller	10	ST_CLERK
Andrea	10	IT_PROG

5. Create a report for HR that displays the last name and salary of every employee who reports to King.

```
SELECT e.last_name, e.salary FROM employees e JOIN employees m ON e.manager_id =  
m.employee_id WHERE m.last_name = 'King';
```

LAST_NAME	SALARY
Smith	12500
Davis	15000
Andrea	16000

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

```
SELECT e.department_id, e.last_name, e.job_id FROM employees e JOIN departments d ON e.department_id = d.department_id WHERE d.department_name = 'Executive';
```

DEPARTMENT_ID	LAST_NAME	JOB_ID
50	Matos	IT_PROG
50	Johnson	SA_MAN
50	Austin	AC_MGR

7. Modify the query 3 to display the employee number, last name, and salary of all employees who earn more than the average salary and who work in a department with any employee whose last name contains a *u*.

```
SELECT e1.employee_id, e1.last_name, e1.salary FROM employees e1  
JOIN employees e2 ON e1.department_id = e2.department_id WHERE e2.last_name LIKE  
'%u%' AND e1.salary > (SELECT AVG(salary) FROM employees);
```

EMPLOYEE_ID	LAST_NAME	SALARY
106	Wilson	13500
104	Davis	15000

Ex.No.: 10	AGGREGATING DATA USING GROUP FUNCTIONS
Date:	21 / 09 / 24

Determine the validity of the following three statements. Circle either True or False.

1. Group functions work across many rows to produce one result per group. True/False

TRUE

2. Group functions include nulls in calculations.
True/False

FALSE

3. The WHERE clause restricts rows prior to inclusion in a group calculation. True/False

TRUE

The HR department needs the following reports:

4. Find the highest, lowest, sum, and average salary of all employees.
Label the columns Maximum, Minimum, Sum, and Average, respectively.
Round your results to the nearest whole number

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

Maximum	Minimum	Sum	Average
16000	4600	158500	7925

5. Modify the above query to display the minimum, maximum, sum, and average salary for each job type.

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

JOB_ID	Maximum	Minimum	Sum	Average
IT_PROG	16000	6000	51600	8600
AC_ACCOUNT	15000	15000	15000	15000
AC_MGR	7100	7100	7100	7100
SA_MAN	7200	7200	7200	7200
SA_REP	13500	5500	30800	7700
HR_REP	12500	4600	35300	7060
ST_CLERK	6200	5300	11500	5750

6. Write a query to display the number of people with the same job. Generalize the query so that the user in the HR department is prompted for a job title.

```
SELECT COUNT(*) AS "Number of People" FROM employees WHERE job_id =
'&job_title';
```

Number of People
6

7. Determine the number of managers without listing them. Label the column Number of Managers. *Hint: Use the MANAGER_ID column to determine the number of managers.*

```
SELECT COUNT(DISTINCT manager_id) AS "Number of Managers" FROM employees
WHERE manager_id IS NOT NULL;
```

Number of Managers
5

8. Find the difference between the highest and lowest salaries. Label the column DIFFERENCE.

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

DIFFERENCE
11400

9. Create a report to display the manager number and the salary of the lowest-paid employee for that manager. Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is \$6,000 or less. Sort the output in descending order of salary.

```
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 MIN(salary) DESC;
```

MANAGER_ID	Lowest Salary
103	13500
101	12500

10. Create a query to display the total number of employees and, of that total, the number of employees hired in 1995, 1996, 1997, and 1998. Create appropriate column headings.

```
SELECT  
    COUNT(*) AS "Total Employees",  
    SUM(CASE WHEN TO_CHAR(hire_date, 'YYYY') = '1995' THEN 1 ELSE 0 END) AS  
    "Hired in 1995",  
    SUM(CASE WHEN TO_CHAR(hire_date, 'YYYY') = '1996' THEN 1 ELSE 0 END) AS  
    "Hired in 1996",  
    SUM(CASE WHEN TO_CHAR(hire_date, 'YYYY') = '1997' THEN 1 ELSE 0 END) AS  
    "Hired in 1997",  
    SUM(CASE WHEN TO_CHAR(hire_date, 'YYYY') = '1998' THEN 1 ELSE 0 END) AS  
    "Hired in 1998" FROM employees;
```

Total Employees	Hired in 1995	Hired in 1996	Hired in 1997	Hired in 1998
20	1	1	2	3

11. Create a matrix query to display the job, the salary for that job based on department number, and the total salary for that job, for departments 20, 50, 80, and 90, giving each column an appropriate heading.

```

SELECT job_id,
       SUM(CASE WHEN department_id = 20 THEN salary ELSE 0 END) AS "Dept 20",
       SUM(CASE WHEN department_id = 50 THEN salary ELSE 0 END) AS "Dept 50",
       SUM(CASE WHEN department_id = 80 THEN salary ELSE 0 END) AS "Dept 80",
       SUM(CASE WHEN department_id = 90 THEN salary ELSE 0 END) AS "Dept 90",
       SUM(salary) AS "Total Salary"
FROM employees WHERE department_id IN (20, 50, 80, 90) GROUP BY job_id;

```

JOB_ID	Dept 20	Dept 50	Dept 80	Dept 90	Total Salary
IT_PROG	0	6000	0	0	6000
AC_ACCOUNT	15000	0	0	0	15000
AC_MGR	0	7100	0	0	7100
SA_MAN	0	7200	0	0	7200
SA_REP	0	0	13500	0	13500
HR REP	4600	0	0	0	4600

12. Write a query to display each department's name, location, number of employees, and the average salary for all the employees in that department. Label the column name-Location, Number of people, and salary respectively. Round the average salary to two decimal places.

```

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

```

Department Name	Location	Number of People	Average Salary
IT	London	3	6766.67
ST_CLERK	Dubai	1	13500
ST_CLERK	Sydney	1	5300
Customer Service	Mumbai	1	12500
Admin	New York	2	11100
ST_CLERK	San Francisco	2	9800

Ex.No.: 11	PL SQL PROGRAMS
Date: 28 / 9 / 24	

PROGRAM 1

Write a PL/SQL block to calculate the incentive of an employee whose ID is 110.

DECLARE

```
pl_emp_id employees.employee_id%TYPE := 110;
```

```
pl_salary employees.salary%TYPE;
```

```
pl_incentive NUMBER;
```

BEGIN

```
SELECT salary INTO pl_salary
```

```
FROM employees
```

```
WHERE employee_id = pl_emp_id;
```

```
pl_incentive := pl_salary * 0.10;
```

```
UPDATE employees
```

```
SET incentive = pl_incentive
```

```
WHERE employee_id = pl_emp_id;
```

```
DBMS_OUTPUT.PUT_LINE('Incentive for employee ID ' || pl_emp_id || ' is ' || pl_incentive);
```

```
COMMIT;
```

```
END;
```

Results	Explain	Describe	Saved SQL	History
Incentive for employee ID 110 is 820 1 row(s) updated. 0.00 seconds				

PROGRAM 2

Write a PL/SQL block to show an invalid case-insensitive reference to a quoted and without quoted user-defined identifier.

```
DECLARE
    employeeName VARCHAR2(100);
    "EmployeeID" NUMBER;
BEGIN
    employeeName := 'John Doe';
    "EmployeeID" := 40;

    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || employeeName);
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || "EmployeeID");
END;
```

Results	Explain	Describe	Saved SQL	History
Employee Name: John Doe Employee ID: 40 Statement processed. 0.01 seconds				

PROGRAM 3

Write a PL/SQL block to adjust the salary of the employee whose ID 122.
Sample table: employees

```

DECLARE
    v_employee_id NUMBER := 122;
    v_salary NUMBER;
    v_new_salary NUMBER;
    v_increase_percentage NUMBER := 0.40;
BEGIN
    SELECT salary INTO v_salary
    FROM employees
    WHERE employee_id = v_employee_id;

    v_new_salary := v_salary + (v_salary * v_increase_percentage / 100);

    UPDATE employees
    SET salary = v_new_salary
    WHERE employee_id = v_employee_id;

    DBMS_OUTPUT.PUT_LINE('Employee ID ' || v_employee_id || ' new salary: ' ||
    v_new_salary);
END;

```

Results	Explain	Describe	Saved SQL	History
Employee ID 122 new salary: 9036.036				
1 row(s) updated.				
0.01 seconds				

PROGRAM 4

Write a PL/SQL block to create a procedure using the "IS [NOT] NULL Operator" and show AND operator returns TRUE if and only if both operands are TRUE.

```

CREATE OR REPLACE PROCEDURE check_null
IS
    value1 NUMBER := 10;
    value2 NUMBER := NULL;
BEGIN
    IF value1 IS NOT NULL AND value2 IS NULL THEN
        DBMS_OUTPUT.PUT_LINE('Null value found');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Both values are not null!!');
    END IF;
END;
/

```

BEGIN

```

check_null;
END;
/

```

Results	Explain	Describe	Saved SQL	History
Both values are not null!!				
Statement processed.				
0.00 seconds				

PROGRAM 5

Write a PL/SQL block to describe the usage of LIKE operator including wildcard characters and escape character.

```

DECLARE
    v_employeename employees.first_name%TYPE;
    v_employeeid NUMBER := 122;
BEGIN
    SELECT first_name INTO v_employeename
    FROM employees
    WHERE first_name LIKE '%e%'
        AND employee_id = v_employeeid;

    DBMS_OUTPUT.PUT_LINE(v_employeename);
END;

```

PROGRAM 6

Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num_small variable and large number will store in num_large variable.

```

DECLARE
    ab NUMBER := 10;
    cd NUMBER := 20;
    num_small NUMBER;
    num_large NUMBER;
BEGIN
    IF ab > cd THEN
        num_small := cd;
        num_large := ab;
    ELSE
        num_small := ab;
    END IF;
END;

```

```

num_large := cd;
END IF;

DBMS_OUTPUT.PUT_LINE('small number = ' || num_small);
DBMS_OUTPUT.PUT_LINE('large number = ' || num_large);
END;

```

```

small number = 10
large number = 20

Statement processed.
```

0.01 seconds

PROGRAM 7

Write a PL/SQL procedure to calculate the incentive on a target achieved and display the message either the record updated or not.

```

CREATE OR REPLACE PROCEDURE calculate_incentive (
    p_emp_id employees.employee_id%TYPE,
    p_target NUMBER
)
IS
    v_incentive NUMBER(7,2);
    v_salary employees.salary%TYPE;
BEGIN
    SELECT salary INTO v_salary
    FROM employees
    WHERE employee_id = p_emp_id;

    IF p_target >= 100000 THEN
        v_incentive := v_salary * 0.1;
        DBMS_OUTPUT.PUT_LINE('Incentive of ' || v_incentive || ' calculated for employee ID ' ||
p_emp_id);
    ELSE
        DBMS_OUTPUT.PUT_LINE('No incentive for employee ID ' || p_emp_id);
    END IF;
END;

```

```

Incentive of 750 calculated for employee ID 176
```

```

Statement processed.
```

0.02 seconds

PROGRAM 8

Write a PL/SQL procedure to calculate incentive achieved according to the specific sale limit.

```
CREATE OR REPLACE PROCEDURE incentive_sale (
    p_emp_id employees.employee_id%TYPE,
    p_sales NUMBER
)
IS
    v_incentive NUMBER(7,2);
BEGIN
    IF p_sales > 100000 THEN
        v_incentive := p_sales * 0.1;
    ELSIF p_sales BETWEEN 50000 AND 100000 THEN
        v_incentive := p_sales * 0.05;
    ELSE
        v_incentive := 0;
    END IF;

    DBMS_OUTPUT.PUT_LINE('Incentive for employee ID ' || p_emp_id || ' is: ' || v_incentive);
END;

BEGIN
    incentive_sale(122, 500000);
END;
```

Incentive for employee ID 122 is: 50000

Statement processed.

0.01 seconds

PROGRAM 9

Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

```
DECLARE
    no_of_emp NUMBER;
    vacancies NUMBER := 45;
BEGIN
    SELECT count(*) INTO no_of_emp
    FROM employees
    WHERE department_id = 50;

    IF no_of_emp < vacancies THEN
        DBMS_OUTPUT.PUT_LINE('Vacancies are available');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Vacancies are not available');
```

```
END IF;  
END;
```

```
vacancies are available  
  
Statement processed.  
  
0.01 seconds
```

PROGRAM 10

Write a PL/SQL program to count number of employees in a specific department and check whether this department have any vacancies or not. If any vacancies, how many vacancies are in that department.

```
DECLARE  
    v_department_id NUMBER := 55;  
    v_emp_count NUMBER;  
    v_vacancies NUMBER := 50;  
BEGIN  
    SELECT count(*) INTO v_emp_count  
    FROM employees  
    WHERE department_id = v_department_id;  
  
    IF v_emp_count < v_vacancies THEN  
        DBMS_OUTPUT.PUT_LINE('Vacancies available: ' || (v_vacancies - v_emp_count));  
    ELSE  
        DBMS_OUTPUT.PUT_LINE('No vacancies available.');//  
    END IF;  
END;
```

```
Vacancies available: 47  
  
Statement processed.  
  
0.01 seconds
```

PROGRAM 11

Write a PL/SQL program to display the employee IDs, names, job titles, hire dates, and salaries of all employees.

```
BEGIN  
FOR i IN (  
    SELECT employee_id, first_name || ' ' || last_name AS name, job_id, hire_date, salary
```

```

        FROM employees
    )
LOOP
    DBMS_OUTPUT.PUT_LINE('ID: ' || i.employee_id || ', Name: ' || i.name || ', Job: ' || i.job_id
    ||
        ', Hire Date: ' || i.hire_date || ', Salary: ' || i.salary);
END LOOP;
END;

```

```

ID: 2, Name: Emma Austen, Job: ST_CLERK, Hire Date: 11/06/1990, Salary: 5500
ID: 10, Name: Paul Rudd, Job: #pr010, Hire Date: 04/06/1969, Salary: 2500
ID: 11, Name: Brie Zlotkey, Job: #b1011, Hire Date: 10/01/1989, Salary: 7200
ID: 20, Name: Elizabeth Olsen, Job: #eo020, Hire Date: 02/16/1989, Salary: 7300
ID: 25, Name: Cate Abu, Job: #cb025, Hire Date: 05/14/1969, Salary: 13500
ID: 27, Name: Jeff Goldblum, Job: ST_CLERK, Hire Date: 10/22/1952, Salary: 3500
ID: 122, Name: Robert Downey, Job: #rd003, Hire Date: 04/04/1965, Salary: 9036.04
ID: 18, Name: Karen Gillan, Job: #kg018, Hire Date: 11/28/1987, Salary: 6900
ID: 21, Name: Anthony Mackie, Job: ST_CLERK, Hire Date: 09/23/1978, Salary: 4000
ID: 22, Name: Sebastian Stan, Job: #ss022, Hire Date: 08/13/1982, Salary: 9000
ID: 28, Name: Karl Austin, Job: #ka028, Hire Date: 06/07/1972, Salary: 13500
ID: 176, Name: Chris Morris, Job: #ce005, Hire Date: 05/07/1994, Salary: 7500
ID: 6, Name: Mark Ruffalo, Job: #mr006, Hire Date: 11/22/1967, Salary: 7200
ID: 12, Name: Chadwick Boseman, Job: #cb012, Hire Date: 11/29/1976, Salary: 8000
ID: 24, Name: Tom Hiddleston, Job: #th024, Hire Date: 02/09/1981, Salary: 6500
ID: 1, Name: Justin Beiber, Job: ST_CLERK, Hire Date: 09/21/1996, Salary: 4900
ID: 8, Name: Jeremy Wilson, Job: #ja008, Hire Date: 01/07/1971, Salary: 13500
ID: 7, Name: Chris Hemsworth, Job: #ch007, Hire Date: 08/11/1983, Salary: 7800
ID: 9, Name: Tom Holland, Job: ST_CLERK, Hire Date: 06/01/1996, Salary: 6000
ID: 13, Name: Chris Austin, Job: #ca013, Hire Date: 06/21/1979, Salary: 13500
ID: 17, Name: Dave Bautista, Job: #db017, Hire Date: 01/18/1969, Salary: 6500
ID: 26, Name: Tessa Thompson, Job: ST_CLERK, Hire Date: 10/03/1983, Salary: 5200
ID: 14, Name: Zoe Austin, Job: #za014, Hire Date: 06/19/1978, Salary: 13500
ID: 19, Name: Pom Davies, Job: #pk019, Hire Date: 05/03/1986, Salary: 1100
ID: 42, Name: Matos roy, Job: #mr042, Hire Date: 02/23/1991, Salary: 7000
ID: 4, Name: Scarlett Austin, Job: #sa004, Hire Date: 11/22/1984, Salary: 13500
ID: 15, Name: Bradley Hook, Job: ST_CLERK, Hire Date: 01/05/1975, Salary: 4500
ID: 16, Name: Vin Diesel, Job: #vd016, Hire Date: 07/18/1967, Salary: 8000
ID: 110, Name: Benedict andru, Job: #bc023, Hire Date: 07/19/1976, Salary: 8200
ID: 30, Name: Taika Waititi, Job: #tw030, Hire Date: 08/16/1975, Salary: 7700
ID: 40, Name: John Doe , Job: #jd040 , Hire Date: 08/10/1995, Salary: 6000
ID: 29, Name: Idris Elba, Job: #ie029, Hire Date: 09/06/1972, Salary: 7400
ID: 41, Name: Matos charles, Job: #mc041, Hire Date: 09/18/1993, Salary: 8900

```

Statement processed.

PROGRAM 12

Write a PL/SQL program to display the employee IDs, names, and department names of all employees.

```

BEGIN
    FOR i IN (
        SELECT e.employee_id, e.first_name || ' ' || e.last_name AS name, d.dept_name
        FROM employees e
        JOIN department d ON e.department_id = d.department_id
    )
    LOOP
        DBMS_OUTPUT.PUT_LINE('ID: ' || i.employee_id || ', Name: ' || i.name || ', Department: ' ||
        i.dept_name);
    END LOOP;
END;

```

```
ID: 25, Name: Cate Abu, Department: executive  
ID: 15, Name: Bradley Hook, Department: sales manager  
ID: 30, Name: Taika Waititi, Department: accounts manager  
  
Statement processed.
```

0.03 seconds

PROGRAM 13

Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all jobs.

```
BEGIN  
FOR rec IN (  
    SELECT e.employee_id, d.dept_name, MIN(salary) AS min_salary  
    FROM employees e  
    JOIN department d ON e.department_id = d.department_id  
    GROUP BY e.employee_id, d.dept_name  
)  
LOOP  
    DBMS_OUTPUT.PUT_LINE('Job ID: ' || rec.employee_id || ', Title: ' || rec.dept_name ||  
        ', Min Salary: ' || rec.min_salary);  
END LOOP;  
END;
```

```
Job ID: 30, Title: accounts manager, Min Salary: 7700  
Job ID: 25, Title: executive, Min Salary: 13500  
Job ID: 15, Title: sales manager, Min Salary: 4500
```

Statement processed.

0.05 seconds

PROGRAM 14

Write a PL/SQL program to display the employee IDs, names, and job history start dates of all employees.

```
BEGIN  
FOR rec IN (  
    SELECT employee_id, first_name || ' ' || last_name AS name, hire_date  
    FROM employees  
)  
LOOP  
    DBMS_OUTPUT.PUT_LINE('ID: ' || rec.employee_id || ', Name: ' || rec.name || ', Start Date:  
        ' || rec.hire_date);  
END LOOP;  
END;
```

```

ID: 2, Name: Emma Austen, Start Date: 11/06/1990
ID: 10, Name: Paul Rudd, Start Date: 04/06/1969
ID: 11, Name: Brie Zlotkey, Start Date: 10/01/1989
ID: 20, Name: Elizabeth Olsen, Start Date: 02/16/1989
ID: 25, Name: Cate Abu, Start Date: 05/14/1969
ID: 27, Name: Jeff Goldblum, Start Date: 10/22/1952
ID: 122, Name: Robert Downey, Start Date: 04/04/1965
ID: 18, Name: Karen Gillan, Start Date: 11/28/1987
ID: 21, Name: Anthony Mackie, Start Date: 09/23/1978
ID: 22, Name: Sebastian Stan, Start Date: 08/13/1982
ID: 28, Name: Karl Austin, Start Date: 06/07/1972
ID: 176, Name: Chris Morris, Start Date: 05/07/1994
ID: 6, Name: Mark Ruffalo, Start Date: 11/22/1967
ID: 12, Name: Chadwick Boseman, Start Date: 11/29/1976
ID: 24, Name: Tom Hiddleston, Start Date: 02/09/1981
ID: 1, Name: Justin Beiber, Start Date: 09/21/1996
ID: 8, Name: Jeremy Wilson, Start Date: 01/07/1971
ID: 7, Name: Chris Hemsworth, Start Date: 08/11/1983
ID: 9, Name: Tom Holland, Start Date: 06/01/1996
ID: 13, Name: Chris Austin, Start Date: 06/21/1979
ID: 17, Name: Dave Bautista, Start Date: 01/18/1969
ID: 26, Name: Tessa Thompson, Start Date: 10/03/1983
ID: 14, Name: Zoe Austin, Start Date: 06/19/1978
ID: 19, Name: Pom Davies, Start Date: 05/03/1986
ID: 42, Name: Matos roy, Start Date: 02/23/1991
ID: 4, Name: Scarlett Austin, Start Date: 11/22/1984
ID: 15, Name: Bradley Hook, Start Date: 01/05/1975
ID: 16, Name: Vin Diesel, Start Date: 07/18/1967
ID: 110, Name: Benedict andru, Start Date: 07/19/1976
ID: 30, Name: Taika Waititi, Start Date: 08/16/1975
ID: 40, Name: John Doe , Start Date: 08/10/1995
ID: 29, Name: Idris Elba, Start Date: 09/06/1972

```

PROGRAM 15

Write a PL/SQL program to display the employee IDs, names, and job history end dates of all employees.

```

BEGIN
  FOR rec IN (
    SELECT employee_id, first_name || ' ' || last_name AS name, end_date
    FROM employees
  )
  LOOP
    DBMS_OUTPUT.PUT_LINE('ID: ' || rec.employee_id || ', Name: ' || rec.name ||
      ', End Date: ' || NVL(TO_CHAR(rec.end_date, 'YYYY-MM-DD'), 'Still Active'));
  END LOOP;
END;

```

```

ID: 2, Name: Emma Austen, End Date: Still Active
ID: 10, Name: Paul Rudd, End Date: Still Active
ID: 11, Name: Brie Zlotkey, End Date: Still Active
ID: 20, Name: Elizabeth Olsen, End Date: Still Active
ID: 25, Name: Cate Abu, End Date: Still Active
ID: 27, Name: Jeff Goldblum, End Date: Still Active
ID: 122, Name: Robert Downey, End Date: Still Active
ID: 18, Name: Karen Gillan, End Date: Still Active
ID: 21, Name: Anthony Mackie, End Date: Still Active
ID: 22, Name: Sebastian Stan, End Date: Still Active
ID: 28, Name: Karl Austin, End Date: Still Active
ID: 176, Name: Chris Morris, End Date: Still Active
ID: 6, Name: Mark Ruffalo, End Date: Still Active
ID: 12, Name: Chadwick Boseman, End Date: Still Active
ID: 24, Name: Tom Hiddleston, End Date: Still Active
ID: 1, Name: Justin Beiber, End Date: Still Active
ID: 8, Name: Jeremy Wilson, End Date: Still Active
ID: 7, Name: Chris Hemsworth, End Date: Still Active
ID: 9, Name: Tom Holland, End Date: Still Active
ID: 13, Name: Chris Austin, End Date: Still Active
ID: 17, Name: Dave Bautista, End Date: Still Active
ID: 26, Name: Tessa Thompson, End Date: Still Active
ID: 14, Name: Zoe Austin, End Date: Still Active
ID: 19, Name: Pom Davies, End Date: Still Active
ID: 42, Name: Matos roy, End Date: Still Active
ID: 4, Name: Scarlett Austin, End Date: Still Active
ID: 15, Name: Bradley Hook, End Date: Still Active
ID: 16, Name: Vin Diesel, End Date: Still Active
ID: 110, Name: Benedict andru, End Date: Still Active
ID: 30, Name: Taika Waititi, End Date: Still Active
ID: 40, Name: John Doe , End Date: Still Active
ID: 29, Name: Idris Elba, End Date: Still Active

```

Ex.No.: 12	WORKING WITH CURSOR, PROCEDURES AND FUNCTIONS
Date: 5 / 10 / 24	

Program 1

FACTORIAL OF A NUMBER USING FUNCTION

DECLARE

```

n NUMBER := 10;
result NUMBER;
```

```

FUNCTION itfact(num NUMBER) RETURN NUMBER IS
    fact NUMBER := 1;
BEGIN
    FOR i IN 1..num LOOP
```

```

fact := fact * i;
END LOOP;
RETURN fact;
END;

BEGIN
result := itfact(n);
DBMS_OUTPUT.PUT_LINE('The factorial of ' || n || ' is ' || result);
END;

```

Results	Explain	Describe	Saved SQL	History
The factorial of 10 is 3628800				
Statement processed.				
0.01 seconds				

Program 2

Write a PL/SQL program using Procedures IN,INOUT,OUT parameters to retrieve the corresponding book information in library

```

CREATE OR REPLACE PROCEDURE book_info (
    p_book_id IN NUMBER,
    p_author OUT VARCHAR2,
    p_title OUT VARCHAR2,
    p_published_date OUT DATE
) AS
BEGIN
    SELECT author, title, published_date
    INTO p_author, p_title, p_published_date
    FROM books
    WHERE book_id = p_book_id;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        p_author := NULL;

```

```

p_title := NULL;
p_published_date := NULL;
WHEN OTHERS THEN
    RAISE;
END book_info;

DECLARE
    v_author VARCHAR2(100);
    v_title VARCHAR2(100);
    v_published_date DATE;
    v_book_id NUMBER := 1;
BEGIN
    book_info(v_book_id, v_author, v_title, v_published_date);

    IF v_author IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('Book ID: ' || v_book_id);
        DBMS_OUTPUT.PUT_LINE('Author: ' || v_author);
        DBMS_OUTPUT.PUT_LINE('Title: ' || v_title);
        DBMS_OUTPUT.PUT_LINE('Published Date: ' || TO_CHAR(v_published_date, 'YYYY-MM-DD'));
    ELSE
        DBMS_OUTPUT.PUT_LINE('No book found with ID: ' || v_book_id);
    END IF;
END;

```

Book ID: 1
Author: William Shaespeare
Title: Hamlet
Published Date: 1590-12-12

Statement processed.

0.02 seconds

Ex.No.: 13	
Date:	18 / 10 /24

WORKING WITH TRIGGER TRIGGER

Program 1

Write a code in PL/SQL to develop a trigger that enforces referential integrity by preventing the deletion of a parent record if child records exist.

```

CREATE OR REPLACE TRIGGER prevent_parent_deletion
    BEFORE DELETE ON employees
    FOR EACH ROW
DECLARE
    pl_dept_count NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO pl_dept_count
    FROM department
    WHERE dept_id = :OLD.employee_id;

    IF pl_dept_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Cannot delete employee record as department
records exist.');
    END IF;
END;

-- Deleting employee with ID 70 (This part would be executed outside the trigger)
DELETE FROM employees
WHERE employee_id = 70;

```

Program 2

Write a code in PL/SQL to create a trigger that checks for duplicate values in a specific column and raises an exception if found.

```

CREATE OR REPLACE TRIGGER prevent_duplicate_manager_id
    BEFORE INSERT OR UPDATE ON employees
    FOR EACH ROW
DECLARE
    pl_count NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO pl_count
    FROM employees
    WHERE manager_id = :NEW.manager_id
    AND employee_id != :NEW.employee_id;

```

```

IF pl_count > 0 THEN
    RAISE_APPLICATION_ERROR(-20003, 'Duplicate manager_id found: ' ||
:NEW.manager_id);
END IF;
END;
/

```

INSERT INTO employees (

- employee_id, first_name, last_name, email, phone_number,
- hire_date, job_id, salary, commission_pct, manager_id, department_id

) VALUES (

- 202, 'Jane', 'Smith',
- 'john006@gmail.com', 7383922241, TO_DATE('11/9/2000', 'MM/DD/YYYY'),
- 'ST_CLERK', 10000, 0.15, 400, 80

);

Program 3

Write a code in PL/SQL to create a trigger that restricts the insertion of new rows if the total of a column's values exceeds a certain threshold.

```

CREATE OR REPLACE TRIGGER restrict_salary_insertion
BEFORE INSERT ON employees
FOR EACH ROW
DECLARE
    total_salary NUMBER;
    threshold NUMBER := 100000;
BEGIN
    SELECT SUM(salary)
    INTO total_salary
    FROM employees;

    IF (total_salary + :NEW.salary) > threshold THEN
        RAISE_APPLICATION_ERROR(-20004, 'Insertion denied: Total salary exceeds the
threshold of ' || threshold);
    END IF;
END;
/

```

INSERT INTO employees (

- employee_id, first_name, last_name, email, phone_number,
- hire_date, job_id, salary, commission_pct, manager_id, department_id

) VALUES (

- 203, 'Charlie', 'Brown', 'charlie203@gmail.com', '9122334455',
- TO_DATE('03/01/2021', 'MM/DD/YYYY'), '#cb203', 5000, 0.20, 1000, 50

);

Program 4

Write a code in PL/SQL to design a trigger that captures changes made to specific columns and logs them in an audit table.

```
CREATE OR REPLACE TRIGGER audit_changes
AFTER UPDATE OF salary, job_id ON employees
FOR EACH ROW
BEGIN
    IF :OLD.salary != :NEW.salary OR :OLD.job_id != :NEW.job_id THEN
        INSERT INTO employee_audit (
            employee_id,
            old_salary,
            new_salary,
            old_job_title,
            new_job_title,
            change_timestamp,
            changed_by
        ) VALUES (
            :OLD.employee_id,
            :OLD.salary,
            :NEW.salary,
            :OLD.job_id,
            :NEW.job_id,
            SYSTIMESTAMP,
            USER
        );
    END IF;
END;
/
```

```
UPDATE employees
SET salary = 55000, job_id = 'ST_CLERK'
WHERE employee_id = 176;
```

```
SELECT * FROM employee_audit;
```

AUDIT_ID	EMPLOYEE_ID	OLD_SALARY	NEW_SALARY	OLD_JOB_ID	NEW_JOB_ID	CHANGE_TIMESTAMP	CHANGED_BY
1	20	50000	55000	manager	manager	15-OCT-24 10:00:00.000000 AM	admin
2	122	60000	65000	Manager	Manager	15-OCT-24 10:35:00.000000 AM	admin
5	27	45000	47000	Analyst	Senior Analyst	15-OCT-24 10:30:00.000000 AM	user1
22	176	2500	55000	IT005	ST_CLERK	15-OCT-24 10:45:06.252580 PM	APEX_PUBLIC_USER
3	9	70000	75000	Senior Developer	Lead Developer	15-OCT-24 10:45:00.000000 AM	user2
4	4	80000	85000	Team Lead	Project Manager	15-OCT-24 11:00:00.000000 AM	admin

Program 5

Write a code in PL/SQL to implement a trigger that records user activity (inserts, updates, deletes) in an audit log for a given set of tables.

```

CREATE OR REPLACE TRIGGER trg_audit_employees
AFTER INSERT OR UPDATE OR DELETE ON employees
FOR EACH ROW
DECLARE
    v_old_values CLOB;
    v_new_values CLOB;
BEGIN
    IF INSERTING THEN
        v_old_values := NULL;
        v_new_values := 'employee_id: ' || :NEW.employee_id || ',' ||
                        'first_name: ' || :NEW.first_name || ',' ||
                        'salary: ' || :NEW.salary;
        INSERT INTO audit_log (
            action, table_name, record_id, changed_by, new_values
        ) VALUES (
            'INSERT', 'employees', :NEW.employee_id, USER, v_new_values
        );
    ELSIF UPDATING THEN
        v_old_values := 'employee_id: ' || :OLD.employee_id || ',' ||
                        'first_name: ' || :OLD.first_name || ',' ||
                        'salary: ' || :OLD.salary;
        v_new_values := 'employee_id: ' || :NEW.employee_id || ',' ||
                        'first_name: ' || :NEW.first_name || ',' ||
                        'salary: ' || :NEW.salary;
        INSERT INTO audit_log (
            action, table_name, record_id, changed_by, old_values, new_values
        ) VALUES (
            'UPDATE', 'employees', :NEW.employee_id, USER, v_old_values, v_new_values
        );
    ELSIF DELETING THEN
        v_old_values := 'employee_id: ' || :OLD.employee_id || ',' ||
                        'first_name: ' || :OLD.first_name || ',' ||
                        'salary: ' || :OLD.salary;
        v_new_values := NULL;
        INSERT INTO audit_log (
            action, table_name, record_id, changed_by, old_values
        ) VALUES (
            'DELETE', 'employees', :OLD.employee_id, USER, v_old_values
        );
    END IF;
END trg_audit_employees;
/

```

INSERT INTO employees (employee_id, first_name, salary) VALUES (3, 'Ball', 50000)

```
);
UPDATE employees SET salary = 55000 WHERE employee_id = 3;
DELETE FROM employees WHERE employee_id = 3;
SELECT * FROM audit_log;
```

1 row(s) inserted.

0.12 seconds

1 row(s) updated.

0.06 seconds

AUDIT_ID	ACTION	TABLE_NAME	RECORD_ID	CHANGED_BY	CHANGE_TIMESTAMP	OLD_VALUES	NEW_VALUES
1	INSERT	employees	3	APEX_PUBLIC_USER	16-OCT-24 04.39.17957308 PM	-	employee_id: 3, first_name: Ball, salary: 50000
3	DELETE	employees	3	APEX_PUBLIC_USER	16-OCT-24 04.41.49.077471 PM	employee_id: 3, first_name: Ball, salary: 55000	-
2	UPDATE	employees	3	APEX_PUBLIC_USER	16-OCT-24 04.40.03.193035 PM	employee_id: 3, first_name: Ball, salary: 50000	employee_id: 3, first_name: Ball, salary: 55000

3 rows returned in 0.00 seconds [Download](#)

Program 6

Write a code in PL/SQL to implement a trigger that automatically calculates and updates a running total column for a table whenever new rows are inserted.

```
CREATE TABLE transactions (
    transaction_id NUMBER PRIMARY KEY,
    amount NUMBER,
    running_total NUMBER
);
```

```
CREATE OR REPLACE TRIGGER update_running_total
FOR INSERT ON transactions
COMPOUND TRIGGER
    TYPE amount_array IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
    new_amounts amount_array;
```

BEFORE EACH ROW IS

BEGIN

new_amounts(:NEW.transaction_id) := :NEW.amount;

END BEFORE EACH ROW;

AFTER STATEMENT IS

BEGIN

DECLARE

v_total NUMBER;

BEGIN

SELECT NVL(MAX(running_total), 0)

INTO v_total

FROM transactions;

FOR i IN new_amounts.FIRST .. new_amounts.LAST LOOP

v_total := v_total + new_amounts(i);

UPDATE transactions

SET running_total = v_total

WHERE transaction_id = i;

END LOOP;

END;

END AFTER STATEMENT;

END update_running_total;

/

INSERT INTO transactions (transaction_id, amount)

VALUES (1, 10000);

```
INSERT INTO transactions (transaction_id, amount)
VALUES (2, 20000);
```

Results				Explain	Describe	Saved SQL	History	
	TRANSACTION_ID	AMOUNT	RUNNING_TOTAL					
1		10000	10000					
2		20000	30000					
2 rows returned in 0.01 seconds								
Download								

Program 7

Write a code in PL/SQL to create a trigger that validates the availability of items before allowing an order to be placed, considering stock levels and pending orders.

```
CREATE TABLE inventory (
    item_id NUMBER PRIMARY KEY,
    item_name VARCHAR2(100),
    stock_level NUMBER
);
```

```
CREATE TABLE orders (
    order_id NUMBER PRIMARY KEY,
    item_id NUMBER,
    quantity NUMBER,
    order_status VARCHAR2(20),
    CONSTRAINT fk_item FOREIGN KEY (item_id) REFERENCES inventory(item_id)
);
```

```
CREATE OR REPLACE TRIGGER validate_stock_before_order
BEFORE INSERT ON orders
FOR EACH ROW
```

```
DECLARE
    v_stock_level NUMBER;
    v_pending_orders NUMBER;
```

```
BEGIN
    SELECT stock_level
    INTO v_stock_level
    FROM inventory
    WHERE item_id = :NEW.item_id;
```

```
    SELECT NVL(SUM(quantity), 0)
    INTO v_pending_orders
    FROM orders
    WHERE item_id = :NEW.item_id
    AND order_status = 'Pending';
```

```
    IF (:NEW.quantity + v_pending_orders) > v_stock_level THEN
```

```

    RAISE_APPLICATION_ERROR(-20001, 'Insufficient stock for item: ' || :NEW.item_id);
END IF;
END;
/

```

```

INSERT INTO orders (order_id, item_id, quantity, order_status)
VALUES (1, 101, 5, 'Pending');
INSERT INTO orders (order_id, item_id, quantity, order_status) VALUES (1, 101, 5, 'Pending');
INSERT INTO orders (order_id, item_id, quantity, order_status) VALUES (2, 103, 20,
'Pending');

```

ITEM_ID	ITEM_NAME	STOCK_LEVEL
101	hp_laptop	10
102	keyboard	20
103	mouse	15

1 rows returned in 0.01 seconds [Download](#)

ORDER_ID	ITEM_ID	QUANTITY	ORDER_STATUS
1	101	5	Pending

1 rows returned in 0.01 seconds [Download](#)

Ex.No.: 14	MONGO DB
Date: 19 / 10 /24	

Structure of 'restaurants' collection:

```

{
  "address": {
    "building": "1007",
    "coord": [ -73.856077, 40.848447 ],
    "street": "Morris Park Ave",
    "zipcode": "10462"
  },
  "borough": "Bronx",
  "cuisine": "Bakery",

  "grades": [
    { "date": { "$date": 1393804800000 }, "grade": "A", "score": 2 },
    { "date": { "$date": 1378857600000 }, "grade": "A", "score": 6 },
    { "date": { "$date": 1358985600000 }, "grade": "A", "score": 10 },
    { "date": { "$date": 1322006400000 }, "grade": "A", "score": 9 },
    { "date": { "$date": 1299715200000 }, "grade": "B", "score": 14 }
  ],
  "name": "Morris Park Bake Shop",
  "restaurant_id": "30075445"
}

```

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinees' or restaurant's name begins with letter 'Wil'.

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

2. Write a MongoDB query to find the restaurant Id, name, and grades for those restaurants which achieved a grade of "A" and scored 11 on an ISODate "2014-08-11T00:00:00Z" among many of survey dates..

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

3. Write a MongoDB query to find the restaurant Id, name and grades for those restaurants where the 2nd element of grades array contains a grade of "A" and score 9 on an ISODate "2014-08-11T00:00:00Z". 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. Write a MongoDB query to find the restaurant Id, name, address and geographical location for those restaurants where 2nd element of coord array contains a value which is more than 42 and upto 52..

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

5. Write a MongoDB query to arrange the name of the restaurants in ascending order along with all the columns.

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

6. Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.

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

7. Write a MongoDB query to arranged the name of the cuisine in ascending order and for that same cuisine borough should be in descending order.

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

8. Write a MongoDB query to know whether all the addresses contains the street or not.

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

9. Write a MongoDB query which will select all documents in the restaurants collection where the coord field value is Double.

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

10. Write a MongoDB query which will select the restaurant Id, name and grades for those restaurants which returns 0 as a remainder after dividing the score by 7.

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

11. Write a MongoDB query to find the restaurant name, borough, longitude and attitude and cuisine for those restaurants which contains 'mon' as three letters somewhere in its name.

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

12. Write a MongoDB query to find the restaurant name, borough, longitude and latitude and cuisine for those restaurants which contain 'Mad' as first three letters of its name.

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

13. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5.

```
db.restaurants.find({  
    grades: { $elemMatch: { score: { $lt: 5 } } }  
});
```

14. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan.

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

15. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn.

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

16. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn, and their cuisine is not American.

```
db.restaurants.find({  
    borough: { $in: ["Manhattan", "Brooklyn"] },  
    cuisine: { $ne: "American" },  
});
```

```
grades: { $elemMatch: { score: { $lt: 5 } } }  
});
```

17. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn, and their cuisine is not American or Chinese.

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

18. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6.

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

19. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan.

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

```
{ $elemMatch: { score: 6 } }
```

```
}]
```

```
});
```

20. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn.

```
db.restaurants.find({
```

```
borough: { $in: ["Manhattan", "Brooklyn"] },
```

```
grades: { $all: [
```

```
    { $elemMatch: { score: 2 } },
```

```
    { $elemMatch: { score: 6 } }
```

```
]}
```

```
});
```

21. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American.

```
db.restaurants.find({
```

```
borough: { $in: ["Manhattan", "Brooklyn"] },
```

```
cuisine: { $ne: "American" },
```

```
grades: { $all: [
```

```
    { $elemMatch: { score: 2 } },
```

```
    { $elemMatch: { score: 6 } }
```

```
]}
```

```
});
```

22. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American or Chinese.

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

23. Write a MongoDB query to find the restaurants that have a grade with a score of 2 or a grade with a score of 6.

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

Sample document of 'movies' collection

```
{  
    _id: ObjectId("573a1390f29313caabcd42e8"), plot: 'A group of bandits stage a brazen  
train hold-up, only to find a determined posse hot on their heels.', genres: [ 'Short', 'Western'  
],  
runtime: 11, cast:  
[  
    'A.C. Abadie',  
    "Gilbert M. 'Broncho Billy' Anderson",  
    'George Barnes',  
    'Justus D. Barnes'  
],  
poster: 'https://m.media-  
amazon.com/images/M/MV5BMTU3NjE5NzYtYTYYyNS00MDVmLWIwYjgtMmYwYWIxZD  
YyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._V1_SY1000_SX677_AL_.jpg', title: 'The  
Great Train Robbery',  
fullplot: "Among the earliest existing films in American cinema - notable as the first film that  
presented a narrative story to tell - it depicts a group of cowboy outlaws who hold up a train and  
rob the passengers. They are then pursued by a Sheriff's posse. Several scenes have color  
included - all hand tinted.",  
languages: [ 'English' ],  
released: ISODate("1903-12-01T00:00:00.000Z"),  
directors: [ 'Edwin S. Porter' ], rated: 'TV-G',  
awards: { wins: 1, nominations: 0, text: '1 win.'  
, lastupdated: '2015-08-13  
00:27:59.177000000', year: 1903,  
imdb: { rating: 7.4, votes: 9847, id: 439 },  
countries: [ 'USA' ], type: 'movie',  
tomatoes: {  
viewer: { rating: 3.7, numReviews: 2559, meter: 75 }, fresh:  
6,  
critic: { rating: 7.6, numReviews: 6, meter: 100 }, rotten:  
0,  
lastUpdated: ISODate("2015-08-08T19:16:10.000Z")  
}
```

1. Find all movies with full information from the 'movies' collection that released in the year 1893.

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

2. Find all movies with full information from the 'movies' collection that have a runtime greater than 120 minutes.

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

3. Find all movies with full information from the 'movies' collection that have "Short" genre.

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

4. Retrieve all movies from the 'movies' collection that were directed by "William K.L. Dickson" and include complete information for each movie.

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

5. Retrieve all movies from the 'movies' collection that were released in the USA and include complete information for each movie.

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

6. Retrieve all movies from the 'movies' collection that have complete information and are rated as "UNRATED".

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

7. Retrieve all movies from the 'movies' collection that have complete information and have received more than 1000 votes on IMDb.

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

8. Retrieve all movies from the 'movies' collection that have complete information and have an IMDb rating higher than 7.

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

9. Retrieve all movies from the 'movies' collection that have complete information and have a viewer rating higher than 4 on Tomatoes.

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

10. Retrieve all movies from the 'movies' collection that have received an award.

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

11. Find all movies with title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries from the 'movies' collection in MongoDB that have at least one nomination.

```
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  
  }  
)
```

12. Find all movies with title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries from the 'movies' collection in MongoDB with cast including "Charles Kayser".

```
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  
  }  
)
```

13. Retrieve all movies with title, languages, released, directors, writers, countries from the 'movies' collection in MongoDB that released on May 9, 1893.

```
db.movies.find(  
  { released: ISODate("1893-05-09T00:00:00Z") },  
  {  
    title: 1,  
    languages: 1,  
    released: 1,  
    directors: 1,  
    writers: 1,  
    countries: 1  
  }  
)
```

14. Retrieve all movies with title, languages, released, directors, writers, countries from the 'movies' collection in MongoDB that have a word "scene" in the title.

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

Ex.No.: 15	
Date: 25 / 10 /24	OTHER DATABASE OBJECTS

Find the Solution for the following:

1. Create a sequence to be used with the primary key column of the DEPT table. The sequence should start at 200 and have a maximum value of 1000. Have your sequence increment by ten numbers. Name the sequence DEPT_ID_SEQ.

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

2. Write a query in a script to display the following information about your sequences: sequence name, maximum value, increment size, and last number

```
SELECT sequence_name, max_value, increment_by, last_number
FROM USER_SEQUENCES
WHERE sequence_name = 'DEPT_ID_SEQ';
```

3. Write a script to insert two rows into the DEPT table. Name your script lab12_3.sql. Be sure to use the sequence that you created for the ID column. Add two departments named Education and Administration. Confirm your additions. Run the commands in your script.

```
INSERT INTO DEPT (DEPT_ID, DEPT_NAME)
VALUES (DEPT_ID_SEQ.NEXTVAL, 'Education');
```

```
INSERT INTO DEPT (DEPT_ID, DEPT_NAME)
VALUES (DEPT_ID_SEQ.NEXTVAL, 'Administration');
```

-- Verify the additions

```
SELECT * FROM DEPT WHERE DEPT_NAME IN ('Education', 'Administration');
```

4. Create a nonunique index on the foreign key column (DEPT_ID) in the EMP table.

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

5. Display the indexes and uniqueness that exist in the data dictionary for the EMP table.

```
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';
```

Ex.No.: 16	CONTROLLING USER ACCESS
Date: 26 / 10 /24	

Find the Solution for the following:

1. **What privilege should a user be given to log on to the Oracle Server? Is this a system or an object privilege?**
 - **Privilege:** The user should be granted the CREATE SESSION system privilege to log on to the Oracle server.
 - **Type:** This is a **system privilege**.

2. **What privilege should a user be given to create tables?**
 - **Privilege:** The user should be granted the CREATE TABLE system privilege to create tables.
 - **Type:** This is a **system privilege**

3. **If you create a table, who can pass along privileges to other users on your table?**
 - **Answer:** The owner of the table (i.e., the user who created the table) can pass along privileges to other users on that table. If the owner grants privileges using the WITH GRANT OPTION, the grantee can further grant those privileges to others.

4. **You are the DBA. You are creating many users who require the same system privileges. What should you use to make your job easier?**
 - **Answer:** You should use **roles**. By creating a role and granting it the necessary privileges, you can assign that role to multiple users, simplifying the process.

5. **What command do you use to change your password?**

Command: The ALTER USER command is used to change the user's password.
Example: ALTER USER scott IDENTIFIED BY lion;

6. **Grant another user access to your DEPARTMENTS table. Have the user grant you query access to his or her DEPARTMENTS table.**
 - **Granting access to your DEPARTMENTS table:**
GRANT select ON departments TO user_name;
 - **Granting you query access to their DEPARTMENTS table**
GRANT select ON their_table_name TO scott;

7.Query all the rows in your DEPARTMENTS table.

```
SELECT * FROM departments;
```

8. Add a new row to your DEPARTMENTS table. Team 1 should add Education as department number 500. Team 2 should add Human Resources department number 510. Query the other team's table.

- **Team 1 (Education):**

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

- **Team 2 (Human Resources):**

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

- **Query the other team's table** (Example for Team 1 querying Team 2's table):

```
SELECT * FROM team_2_departments;
```

9. Query the USER_TABLES data dictionary to see information about the tables that you own.

```
SELECT * FROM user_tables;
```

10. Revoke the SELECT privilege on your table from the other team.

```
REVOKE select ON departments FROM team_2_user;
```

11. Remove the row you inserted into the DEPARTMENTS table in step 8 and save the changes.

- **Remove the row:**

```
DELETE FROM departments WHERE department_id = 500;
```

- **Save the changes** (if using a transaction, this would be done with COMMIT):

```
COMMIT;
```