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 Notebook

CS23332 DATABASE MANAGEMENT SYSTEMS

NaME	VISHWA J
Roll No	230701384
DEPT	CSE
SEC	F

Ex.No.: 1		**CREATION OF BASE TABLE AND DML OPERATIONS**
Date:	31/7/24	

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	NNewman	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	NNewman	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:	5/8/24	

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:	6/8/24	

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.: 5		CREATING VIEWS
Date:	14/8/24	

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_I
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_
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	EMPNO	NUMBER	-	6	0	-	-	-	-
	EMPLOYEE	VARCHAR2	25	-	-	-	-	-	-
	DEPTNO	NUMBER	-	4	0	-	✓	-	-

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

SELECT * FROM DEPT50;

EMPNO	EMPLOYEE	DEPTN
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:	14/8/24	

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 nul,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

LAST_NAME	SALARY
Smith	12500
Davis	15000
Wilson	13500
Brown	16000

Ex.No.: 7		USING SET OPERATORS
Date:	28/8/24	

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

Ex.No.: 8	WORKINGWITHMULTIPLETABLES
Date: 10/9/24	

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: 10/9/24	

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:	11/9/24	

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.: 12		WORKING WITH CURSOR, PROCEDURES AND FUNCTIONS
Date:	23.10.2024	

Program 1

FACTORIAL OF A NUMBER USING FUNCTION

```
create or replace function fact (a number) return number is
fact number:=1;
b number;
begin
b:=a;
while b>0
loop
fact:=fact*b;
b:=b-1;
end loop;
return(fact);
end;
/

declare
a number(2);
f number(10);
begin
a := :n ;
f:=fact(a);
dbms_output.put_line('The factorial is '||f);
end;
/
```

Input : 5

```
The factorial is 120
```

Program 2

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

```
--PROCEDURE FOR IN PARAMETER

create procedure proc(a in number) is bprice number;
begin
select price into bprice from library where book_id=a;
dbms_output.put_line('The price of the book is '||bprice);
end;

declare
a number(2);
begin
a := :n;
proc(a);
end;
```

Input: 5

```
The price of the book is 9.75
```

```
--PROCEDURE FOR OUT PARAMETER

create or replace procedure proc(a in number,n out number) is
begin
select publication_year into n from library where book_id=a;
end;

declare
a number(2);
n number(4);
begin
a := :b;
proc(a,n);
dbms_output.put_line('The year of publication of the book is '||n);
end;
```

Input 7

```
The year of publication of the book is 1951
```

```
--PROCEDURE FOR INOUT PARAMETER

create or replace procedure proc(a in out number) is
begin
a:=a+10;
end;

declare
a number(2);
id number(2);
begin
id := :b;
select price into a from library where book_id=id;
proc(a);
dbms_output.put_line('The updated price of the book is '||a);
end;
```

Input 3

```
The updated price of the book is 23
```

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

PROGRAM 1

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

```
declare
a employees.employee_id%type;
b employees.salary%type;
begin
Select salary into a from employees where employee_id = 110;
b:=0.05*a;
dbms_output.put_line('Salary after incentive : '||(a+b));
end;
```

```
Salary after incentive : 6300

Statement processed.

0.01 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
non_quoted_variable varchar2(10) := 'Hi';
"quoted_variable" varchar2(10) := 'Hello';
begin
dbms_output.put_line(NON_QUOTED_VARIABLE);
dbms_output.put_line("quoted_variable");
dbms_output.put_line("QUOTED_VARIABLE");
end;
```

```
Hi
Hello

Statement processed.
```

```
ORA-06550: line 7, column 23:
PLS-00201: identifier 'QUOTED_VARIABLE' must be declared
ORA-06550: line 7, column 1:
PL/SQL: Statement ignored
```

PROGRAM 3

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

```
declare
old_salary employees.salary%type;
new_salary employees.salary%type;
begin
new_salary:= :sal;
Select salary into old_salary from employees where employee_id = 122;
dbms_output.put_line('Before updation: '||old_salary);
Update employees set salary = salary + new_salary where employee_id = 122;
Select salary into new_salary from employees where employee_id = 122;
dbms_output.put_line('After updation: '||new_salary);
end;
```

```
Before updation: 8000
After updation: 9000

Statement processed.

0.00 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 proc1( a boolean, b boolean) IS
BEGIN
if(a is not null) and (b is not null) then
if(a = TRUE and b = TRUE) then
dbms_output.put_line('TRUE');
else
dbms_output.put_line('FALSE');
end if;
else
dbms_output.put_line('NULL VALUES in arguments');
end if;
end proc1;


BEGIN
proc1(TRUE,TRUE);
proc1(TRUE,FALSE);
proc1(NULL,NULL);
end;
```

```
TRUE
FALSE
NULL VALUES in arguments

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
name varchar2(20);
num number(3);
Begin
num := :n;
Select first_name into name from employees where employee_id=num;
if name like 'D%' then
dbms_output.put_line('Name starts with "D"');
end if;
if name like 'Dan_el%' then
dbms_output.put_line('Name contains "Dan" followed by one character');
end if;
name := 'Daniel_Andrea';
if name like 'Daniel\_Andrea' escape '\' then
dbms_output.put_line('Name contains "Daniel_Andrea"');
end if;
end;
```

```
Name starts with "D"
Name contains "Dan" followed by one character
Name contains "Daniel_Andrea"

Statement processed.
```

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
a number(2);
b number(2);
num_small number(2);
num_large number(2);
begin
a := :s;
b := :l;
dbms_output.put_line('Value in a : '||a);
dbms_output.put_line('Value in b : '||b);
if a>b then
num_small := b;
num_large := a;
else
num_small :=a;
num_large :=b;
end if;
dbms_output.put_line('Smaller number is '||num_small);
dbms_output.put_line('Larger number is '||num_large);
end;
```

```
Value in a : 10
Value in b : 5
Smaller number is 5
Larger number is 10

Statement processed.

0.00 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 calc_incen(emp_id number,achievement number,target number)
AS
incentive number;
rowcount number;
Begin
if achievement > target then
incentive:= achievement*0.2;
else
incentive:=0;
end if;
Update employees set salary = salary + incentive where employee_id = emp_id;
rowcount:= SQL%ROWCOUNT;
if rowcount>0 then
dbms_output.put_line('Record(s) updated');
else
dbms_output.put_line('No Record(s) updated');
end if;
end;

Declare
id number;
achievement number;
target number;
Begin
id := :emp_id;
achievement := :achieve;
target := :target_;
calc_incen(id,achievement,target);
end;
```

```
Record(s) updated

Statement processed.
```

PROGRAM 8

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

```
Create or replace procedure calc_incen(emp_id number,sales number) AS
incentive number;
rowcount number;
Begin
if sales < 1000 then
incentive:= 0;
elsif sales > 1000 and sales < 2000 then
incentive := sales * 0.2;
else
incentive := sales * 0.5;
end if;
Update employees set salary = salary + incentive where employee_id = emp_id;
rowcount:= SQL%ROWCOUNT;
if rowcount>0 then
dbms_output.put_line('Record(s) updated');
else
dbms_output.put_line('No Record(s) updated');
end if;
end;

Declare
id number;
sales number;
sal number;
Begin
id := :emp_id;
sales := :sale;
select salary into sal from employees where employee_id = id;
dbms_output.put_line('Before incentive calculation: '||sal);
calc_incen(id,sales);
select salary into sal from employees where employee_id = id;
dbms_output.put_line('After incentive calculation: '||sal);
end;
```

```
Before incentive calculation: 21000
Record(s) updated
After incentive calculation: 23500

Statement processed.
```

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
emp_count number;
vacancy number := 20;
begin
Select count(*) into emp_count from employees where department_id = 10;
dbms_output.put_line('Total seats : '||vacancy);
dbms_output.put_line('Number of employees in Department 50 : '||emp_count);
if emp_count>vacancy then
dbms_output.put_line('No vacancies available');
else
dbms_output.put_line('Available vacancies : '||(vacancy-emp_count));
end if;
end;
```

```
Total seats : 20
Number of employees in Department 50 : 3
Available vacancies : 17

Statement processed.
```

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
dept_id number;
emp_count number;
vacancy number := 10;
begin
dept_id := :id;
Select count(*) into emp_count from employees where department_id = dept_id;
dbms_output.put_line('Total seats : '||vacancy);
dbms_output.put_line('Number of employees in Department : '||emp_count);
if emp_count>vacancy then
dbms_output.put_line('No vacancies available');
else
dbms_output.put_line('Available vacancies : '||(vacancy-emp_count));
end if;
end;
```

```
Total seats : 10
Number of employees in Department : 2
Available vacancies : 8

Statement processed.
```

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, job_id, hire_date, salary from employees)
loop
dbms_output.put_line('employee id: ' || i.employee_id);
dbms_output.put_line('name: ' || i.first_name);
dbms_output.put_line('job title: ' || i.job_id);
dbms_output.put_line('hire date: ' || to_char(i.hire_date, 'dd-mon-yyyy'));
dbms_output.put_line('salary: ' || i.salary);
dbms_output.put_line(' ------------------------------ ');
end loop;
end;
```

```
employee id: 101
name: John
job title: IT_PROG
hire date: 01-jan-1994
salary: 6020
------------------------------
employee id: 176
name: Jane
job title: HR_REP
hire date: 20-feb-2019
salary: 12500
------------------------------
employee id: 103
name: Mike
job title: SA_MAN
hire date: 01-mar-1998
salary: 7200
------------------------------
employee id: 104
name: Emily
job title: AC_ACCOUNT
hire date: 01-jan-1998
salary: 15000
------------------------------
employee id: 105
name: Robert
job title: ST_CLERK
hire date: 25-jul-2018
salary: 6200
------------------------------
```

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.job_id from employees e)
loop
dbms_output.put_line('employee id: ' || i.employee_id);
dbms_output.put_line('name: ' || i.first_name);
dbms_output.put_line('department name: ' || i.job_id);
dbms_output.put_line(' ------------------------------ ');
end loop;
end;
```

```
employee id: 101
name: John
department name: IT_PROG
--------------------------------
employee id: 176
name: Jane
department name: HR_REP
--------------------------------
employee id: 103
name: Mike
department name: SA_MAN
--------------------------------
employee id: 104
name: Emily
department name: AC_ACCOUNT
--------------------------------
employee id: 105
name: Robert
department name: ST_CLERK
--------------------------------
```

PROGRAM 13

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

```
Begin
for i in (select job_id,job_title,min_salary from jobs)
loop
dbms_output.put_line('job id: ' || i.job_id);
dbms_output.put_line('job title: ' || i.job_title);
dbms_output.put_line('minimum salary: ' || i.min_salary);
dbms_output.put_line(' ------------------------------ ');
end loop;
end;
```

```
job id: 101
job title: Software Engineer
minimum salary: 60000
--------------------------------
job id: 102
job title: Data Analyst
minimum salary: 50000
--------------------------------
job id: 103
job title: Project Manager
minimum salary: 70000
--------------------------------
job id: 104
job title: HR Manager
minimum salary: 55000
--------------------------------
job id: 105
job title: Marketing Specialist
minimum salary: 45000
--------------------------------
```

PROGRAM 14

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

```
Begin
for i in (select employee_id,employee_name,start_date from job_history)
loop
dbms_output.put_line('employee id: ' || i.employee_id);
dbms_output.put_line('name: ' || i.employee_name);
dbms_output.put_line('start date: ' ||to_char(i.start_date, 'dd-mon-yyyy'));
dbms_output.put_line(' ------------------------------ ');
end loop;
end;
```

```
employee id: 201
name: James
start date: 01-jan-2010
--------------------------------
employee id: 202
name: King
start date: 01-jan-2012
--------------------------------
employee id: 203
name: Smith
start date: 01-jan-2013
--------------------------------
employee id: 204
name: Steve
start date: 01-jan-2014
--------------------------------
employee id: 205
name: Robert
start date: 01-jan-2015
--------------------------------
```

PROGRAM 15

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

```
Begin
for i in (select employee_id,employee_name,end_date from job_history)
loop
dbms_output.put_line('employee id: ' || i.employee_id);
dbms_output.put_line('name: ' || i.employee_name);
dbms_output.put_line('end date: ' ||to_char(i.end_date, 'dd-mon-yyyy'));
dbms_output.put_line(' ------------------------------ ');
end loop;
end;
```

```
employee id: 201
name: James
end date: 10-oct-2015
--------------------------------
employee id: 202
name: King
end date: 15-sep-2016
--------------------------------
employee id: 203
name: Smith
end date: 20-mar-2017
--------------------------------
employee id: 204
name: Steve
end date: 05-apr-2018
--------------------------------
employee id: 205
name: Robert
end date: 12-may-2019
--------------------------------
```

Ex.No.: 13		WORKING WITH TRIGGER **TRIGGER**
Date:	**29.10.2024**	

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 parent_table
FOR EACH ROW
DECLARE
   child_count NUMBER;
BEGIN
   SELECT COUNT(*) INTO child_count
   FROM child_table
   WHERE parent_id = :OLD.parent_id;

   IF child_count > 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Cannot delete parent record as child records
exist.');
   END IF;
END;
```

Testing of Trigger

```
DELETE FROM parent_table WHERE parent_id = 1;
```

```
ORA-20001: Cannot delete parent record as child records exist.
```

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 check_duplicate_value

BEFORE INSERT OR UPDATE ON table_name

FOR EACH ROW

DECLARE

  v_count NUMBER;

BEGIN

  -- Check if the new value already exists in the table

  SELECT COUNT(*) INTO v_count

  FROM table_name

  WHERE specific_column = :NEW.specific_column;



  -- If a duplicate is found, raise an error

  IF v_count > 0 THEN

        RAISE_APPLICATION_ERROR(-20002, 'Duplicate value detected in specific column.');

  END IF;

END;

/
```

Output:

ORA-20002: Duplicate value detected in specific column.

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_insertion

BEFORE INSERT ON table_name

FOR EACH ROW

DECLARE

  v_total NUMBER;

  v_threshold CONSTANT NUMBER := 10000; -- Set your threshold here

BEGIN

  -- Calculate the total sum of the column values

  SELECT SUM(column_name) INTO v_total FROM table_name;



  -- Prevent insertion if the threshold is exceeded

  IF v_total + :NEW.column_name > v_threshold THEN

        RAISE_APPLICATION_ERROR(-20003, 'Cannot insert, total column value exceeds threshold.');

  END IF;

END;

/
```

Output:

ORA-20003: Cannot insert, total column value exceeds threshold.

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 log_column_changes

AFTER UPDATE ON table_name

FOR EACH ROW

BEGIN

  -- Check if specific columns have been modified

  IF :OLD.column_name1 != :NEW.column_name1 OR :OLD.column_name2 !=
:NEW.column_name2 THEN

        -- Insert the old and new values into the audit table

        INSERT INTO audit_table (user_id, change_time, old_value, new_value)

        VALUES (USER, SYSDATE, :OLD.column_name1 || ', ' || :OLD.column_name2,
:NEW.column_name1 || ', ' || :NEW.column_name2);

  END IF;

END;

/
```

Output:

User_ID	Change_Time	Old_Value	New_Value
SYSTEM	2024-09-19 10:05:00	OldValue1, OldValue2	NewValue, AnotherNewValue

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 audit_user_activity
AFTER INSERT OR UPDATE OR DELETE ON table_name
FOR EACH ROW
BEGIN
  IF INSERTING THEN
        INSERT INTO audit_log (user_id, operation, record_id, change_time)
        VALUES (USER, 'INSERT', :NEW.id_column, SYSDATE);

  ELSIF UPDATING THEN
        INSERT INTO audit_log (user_id, operation, record_id, change_time)
        VALUES (USER, 'UPDATE', :NEW.id_column, SYSDATE);

  ELSIF DELETING THEN
        INSERT INTO audit_log (user_id, operation, record_id, change_time)
        VALUES (USER, 'DELETE', :OLD.id_column, SYSDATE);
  END IF;
END;
/
```

Output:

User_ID	Operation	Record_ID	Change_Time
SYSTEM	INSERT	1	2024-09-19 10:10:00
SYSTEM	UPDATE	1	2024-09-19 10:15:00
SYSTEM	DELETE	1	2024-09-19 10:20:00

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 OR REPLACE TRIGGER update_running_total

AFTER INSERT ON table_name

FOR EACH ROW

BEGIN

  -- Update the running total column in the total_table

  UPDATE total_table

  SET running_total = running_total + :NEW.value_column

  WHERE total_id = :NEW.total_id;

END;

/
```

Output:

Total_ID	**Running_Total**
1	1500

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 OR REPLACE TRIGGER validate_item_availability

BEFORE INSERT ON orders

FOR EACH ROW

DECLARE

  v_stock_level NUMBER;

  v_pending_orders NUMBER;

BEGIN

  SELECT stock INTO v_stock_level FROM inventory WHERE item_id = :NEW.item_id;

  -- Check pending orders

  SELECT SUM(quantity) INTO v_pending_orders

  FROM orders

  WHERE item_id = :NEW.item_id AND status = 'Pending';

  -- Ensure stock is available for the order

  IF v_stock_level - v_pending_orders < :NEW.order_quantity THEN

        RAISE_APPLICATION_ERROR(-20004, 'Insufficient stock available for this
order.');

  END IF;

END;

/
```

Output:

ORA-20004: Insufficient stock available for this order.

Ex.No.: 14		MongoDB
Date:	30/10/24	

Restaurant Collection

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

```
db.restaurants.find(
   {
      $or: [
         {
            cuisine: { $nin: ['American', 'Chinese'] } // Cuisines other than 'American' and
'Chinese'
         },
         {
            name: { $regex: '^Wil', $options: 'i' } // Restaurant names that begin with 'Wil'
(case-insensitive)
         }
      ]
   },
   {
      _id: 1,       // Retrieve the restaurant ID
      name: 1,      // Retrieve the restaurant name
      borough: 1,   // Retrieve the borough
      cuisine: 1    // Retrieve the cuisine
   }
)
```

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 "2014-08-11 T00:00:00Z" among many of survey dates.**

```
db.restaurants.find(
   {
      "grades": {
         $elemMatch: {
            "grade": "A",                          // Grade must be "A"
            "score": 11,                           // Score must be 11
            "date": ISODate("2014-08-11T00:00:00Z") // Date must match the specified
ISODate
         }
      }
   },
   {
```

```
        _id: 1,        // Retrieve the restaurant ID
        name: 1,       // Retrieve the restaurant name
        grades: 1      // Retrieve the grades
    }
)
```

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": { // Accessing the 2nd element (index 1) of the grades array
         "grade": "A",                          // Grade must be "A"
         "score": 9,                           // Score must be 9
         "date": ISODate("2014-08-11T00:00:00Z") // Date must match the specified
ISODate
      }
   },
   {
      _id: 1,       // Retrieve the restaurant ID
      name: 1,      // Retrieve the restaurant name
      grades: 1     // Retrieve the grades
   }
)
```

4. **Write a MongoDB query to find the restaurant Id, name, address and geographical location for those restaurants where 2nd element of the coord array contains a value which is more than 42 and up to 52.**

```
db.restaurants.find(
   {
      "coord.1": { $gt: 42, $lte: 52 } // Accessing the 2nd element (index 1) of the coord
array
   },
   {
      _id: 1,       // Retrieve the restaurant ID
      name: 1,      // Retrieve the restaurant name
      address: 1,  // Retrieve the address
      coord: 1      // Retrieve the geographical location (coord)
   }
)
```

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 order along with all the columns.**

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

7. **Write a MongoDB query to arrange 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: false } })
```

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

```
db.restaurants.find({
   "coord": { $type: "double" } // or you can use $type: 1
})
```

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

```
db.restaurants.find(
   {
      "grades": {
         $elemMatch: {
            $expr: {
               $eq: [{ $mod: ["$score", 7] }, 0] // Check if score % 7 == 0
            }
         }
      }
   },
   {
      _id: 1,       // Retrieve the restaurant ID
      name: 1,      // Retrieve the restaurant name
      grades: 1     // Retrieve the grades
```

```
    }
)
```

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 } // Regex to find 'mon' anywhere in the name
(case-insensitive)
    },
    {
        name: 1,       // Retrieve the restaurant name
        borough: 1,    // Retrieve the borough
        "coord.0": 1, // Retrieve longitude (assuming longitude is the first element in the
coord array)
        "coord.1": 1, // Retrieve latitude (assuming latitude is the second element in the
coord array)
        cuisine: 1,    // Retrieve the cuisine
        _id: 0         // Exclude the restaurant ID from the results
    }
)
```

12. Write a mongodb query to find the restaurant name, borough, longitude and attitude and cuisine for those restaurants which contains 'Mad' as first three letters in its name.

```
db.restaurants.find(
    {
        name: { $regex: /^Mad/i } // Regex to find names starting with 'Mad'
(case-insensitive)
    },
    {
        name: 1,       // Retrieve the restaurant name
        borough: 1,    // Retrieve the borough
        "coord.0": 1, // Retrieve longitude (assuming longitude is the first element in the
coord array)
        "coord.1": 1, // Retrieve latitude (assuming latitude is the second element in the
coord array)
        cuisine: 1,    // Retrieve the cuisine
        _id: 0         // Exclude the restaurant ID from the results

    }
)
```

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 } // Score must be less than 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", // Condition to filter by borough
      "grades": {
         $elemMatch: {
            score: { $lt: 5 } // Condition to filter grades with score less than 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(
   {
      $or: [
         { borough: "Manhattan" }, // Condition to filter by borough Manhattan
         { borough: "Brooklyn" }      // Condition to filter by borough Brooklyn
      ],
      "grades": {
         $elemMatch: {
            score: { $lt: 5 } // Condition to filter grades with score less than 5
         }
      }


   }
)
```

```
    }
)
```

Movies Collection

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

```
db.movies.find(
   {
      releaseYear: 1893 // Assuming the field for the release year is named 'releaseYear'
   }
)
```

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 } // Assuming the field for runtime is named 'runtime'
   }
)
```

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

```
db.movies.find(
   {
      genres: "Short" // Assuming the field for genres is an array named 'genres'
   }
)
```

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(
   {
      director: "William K. L. Dickson" // Assuming the field for the director is named
'director'
   }
)
```

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

```
db.movies.find(
```

```
    {
        country: "USA" // Assuming the field for the release country is named 'country'
    }
)
```

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

```
db.movies.find(
    {
        rating: "UNRATED" // Assuming the field for the rating is named 'rating'
    }
)
```

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

```
db.movies.find(
    {
        votes: { $gt: 1000 } // Assuming the field for votes is named 'votes'
    }
)
```

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

```
db.movies.find(
    {
        imdbRating: { $gt: 7 } // Assuming the field for IMDb rating is named 'imdbRating'
    }
)
```

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: { $gt: 4 } } // Assuming the viewer rating is nested within a
'tomatoes' object
    }
)
```

Ex.No.: 15		OTHER DATABASE OBJECTS
Date:	04.11.2024	

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_sequence start with 200 increment by 10 maxvalue 1000;

```
Sequence created.
```

SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	CYCLE_FLAG	ORDER_FLAG	CACHE_SIZE	LAST_NUMBER
DEMO_CUST_SEQ	1	9999999999999999999999999999	1	N	N	20	21
DEMO_ORDER_ITEMS_SEQ	1	9999999999999999999999999999	1	N	N	20	61
DEMO_ORD_SEQ	1	9999999999999999999999999999	1	N	N	20	11
DEMO_PROD_SEQ	1	9999999999999999999999999999	1	N	N	20	21
DEMO_USERS_SEQ	1	9999999999999999999999999999	1	N	N	20	21
DEPT_ID_SEQUENCE	1	1000	10	N	N	20	200

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 AS increment_size,last_number FROM user_sequences WHERE sequence_name = 'DEPT_ID_SEQUENCE';

SEQUENCE_NAME	MAX_VALUE	INCREMENT_SIZE	LAST_NUMBER
DEPT_ID_SEQUENCE	1000	10	200

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 departments values(dept_id_sequence.nextval,'HR',111,1010,'US','United States');
Insert into departments values(dept_id_seq.nextval,'Admin',112,1011,'IN','India');

200	HR	111	1010	US	United States
210	Admin	112	1011	IN	India

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

Create index emp_dept_index on Employees(department_id);

EMPLOYEE_INDEX	NORMAL	VISHWAK16	EMPLOYEES	TABLE	NONUNIQUE	DISABLED	-	USERS	2

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

SELECT index_name, uniqueness FROM user_indexes WHERE table_name = 'Employees';

Output :
Index_name : EMPLOYEE_INDEX
Uniqueness : NONUNIQUE

Ex.No.: 16		CONTROLLING USER ACCESS
Date:	06.11.2024	

1. What privilege should a user be given to log on to the Oracle Server? Is this a system or an object privilege?

~ The privilege is CREATE SESSION. This privilege allows a user to log on to the Oracle Server. It is a system privilege, not an object privilege.

2. What privilege should a user be given to create tables?

~ The privilege is CREATE TABLE . This is a system privilege that allows a user to create tables in their own schema.

3. If you create a table, who can pass along privileges to other users on your table?

~ The owner of the table (the user who created it) can pass along privileges to other users. This is done using the GRANT command.
For example:
GRANT SELECT ON my_table TO other_user;

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?

~ Use a role to bundle common system privileges.
Assign this role to users instead of granting privileges individually.

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

~ ALTER USER username IDENTIFIED BY new_password;

6. Grant another user access to your DEPARTMENTS table. Have the user grant you query Access to his or her DEPARTMENTS table.

~ GRANT SELECT ON DEPARTMENTS TO other_user;
~ GRANT SELECT ON DEPARTMENTS TO your_username;

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 should execute:

```
INSERT INTO DEPARTMENTS (DEPARTMENT_ID, DEPARTMENT_NAME) VALUES
(500, 'Education');
COMMIT;
```

~ Team 2 should execute:

```
INSERT INTO DEPARTMENTS (DEPARTMENT_ID, DEPARTMENT_NAME) VALUES
(510, 'Human Resources');
COMMIT;
```

~ To query the other team’s table:

```
SELECT * FROM other_user.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.

~ To revoke the SELECT privilege on your table from the other team:

```
REVOKE SELECT ON DEPARTMENTS FROM other_user;
```

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

~ To remove the row you inserted into the DEPARTMENTS table and save the changes:
For Team 1 (removing the Education department with ID 500):

```
DELETE FROM DEPARTMENTS WHERE DEPARTMENT_ID = 500;
COMMIT;
```

For Team 2 (removing the Human Resources department with ID 510):

```
DELETE FROM DEPARTMENTS WHERE DEPARTMENT_ID = 510;
COMMIT;
```