

EXNO:1**CREATING AND MANAGING TABLE****DATE:**

- 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

QUERY: Create table DEPARTMENT(id number(7),name varchar2(25));

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is active. The main area is titled 'SQL Commands'. A single line of SQL code is entered: 'create table dept(id number(7),name varchar2(25));'. The 'Run' button is visible at the bottom right of the command input field. In the results pane, the output is: 'Table created.' and '0.03 seconds'.

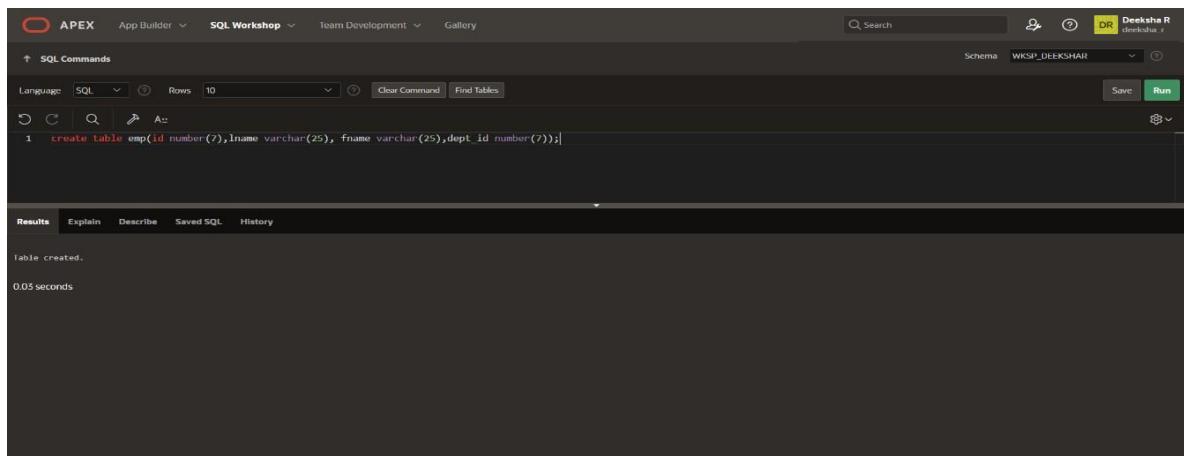
- 2.Create the EMP table based on the following instance chart. Confirm that the table is created.

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

QUERY: Create table emp(id number(7),Last_Name varchar(25),First_Name varchar(25),Dept_id number(7));

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands tab, the following SQL command is entered and executed:

```
1 create table emp(id number(7),lname varchar(25), fname varchar(25),dept_id number(7));
```

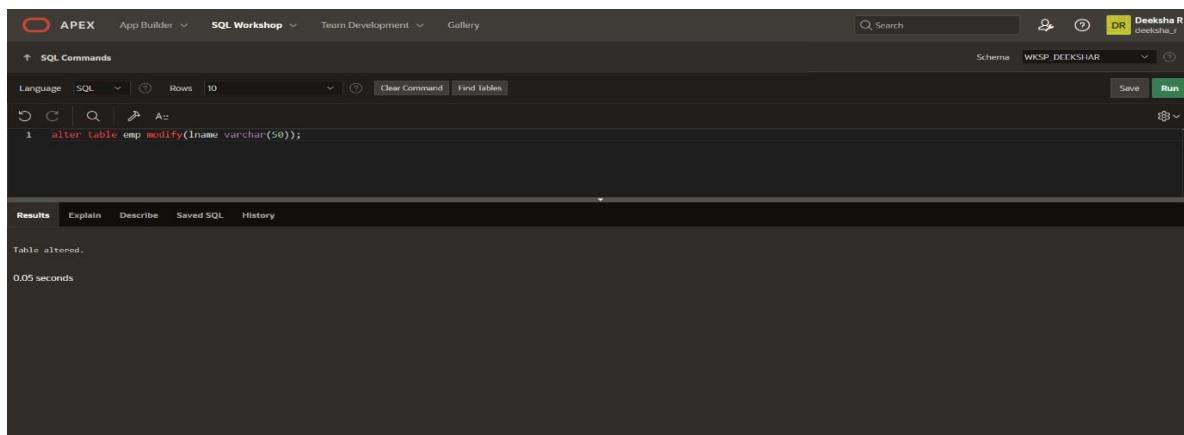
The Results tab displays the output:

```
Table created.  
0.05 seconds
```

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

QUERY: Alter table emp modify(Last_Name varchar2(50));

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands tab, the following SQL command is entered and executed:

```
1 alter table emp modify(lname varchar(50));
```

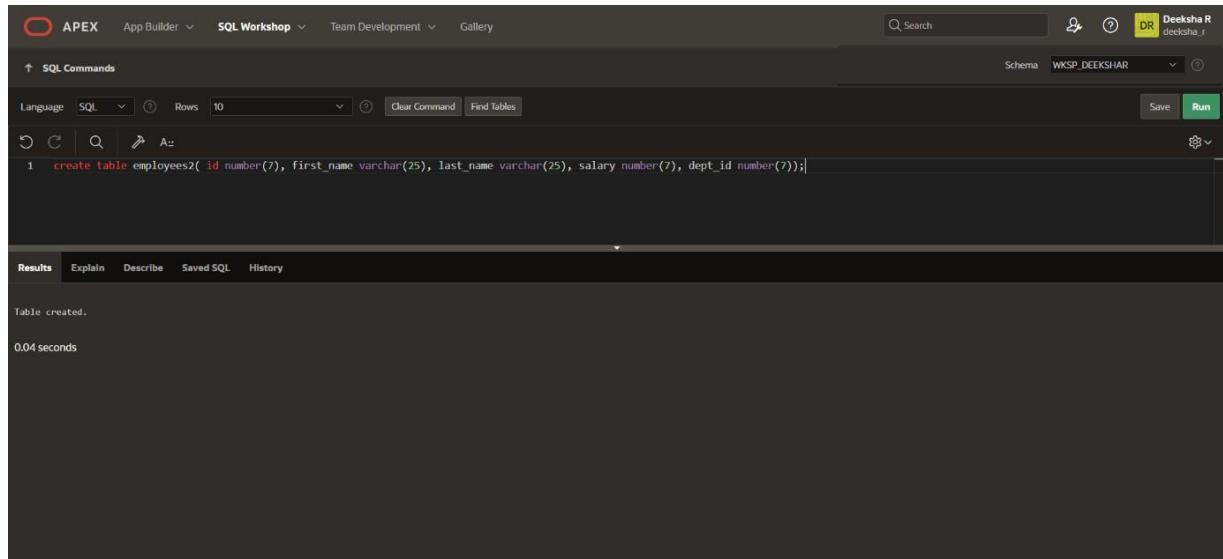
The Results tab displays the output:

```
Table altered.  
0.05 seconds
```

**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.**

QUERY: Create table employees2(id number(7),first_name varchar2(25),Last_name varchar2(25),Salary int,Dept_id number(7));

OUTPUT:

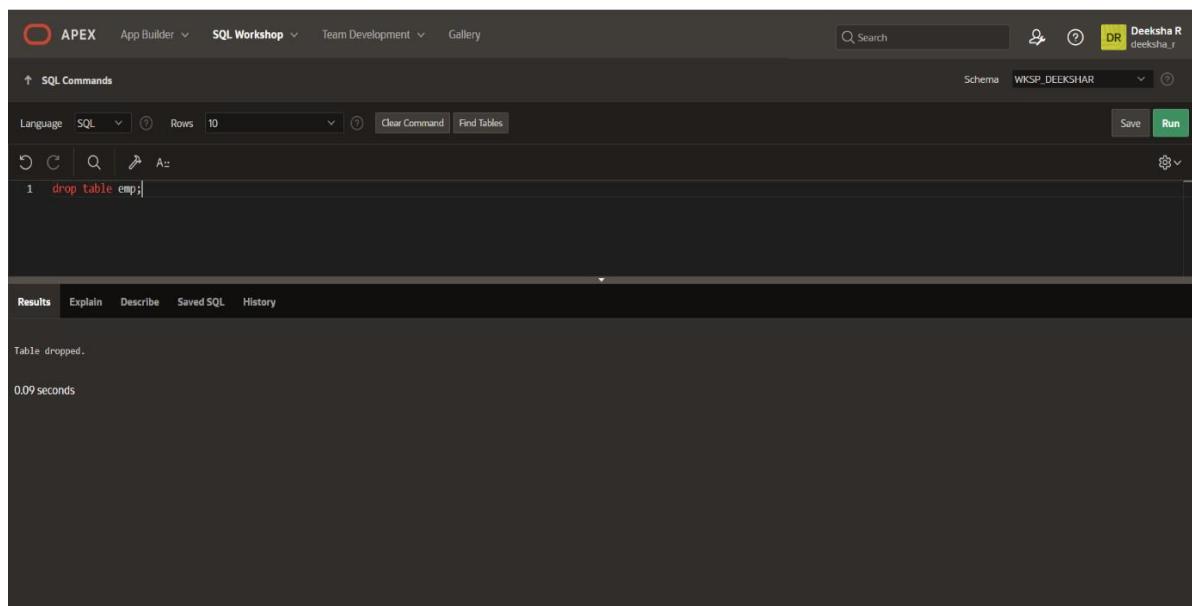


A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar shows "APEX", "App Builder", "SQL Workshop", "Team Development", and "Gallery". The right side shows a user profile for "Deeksha R" and the schema "WKSP_DEEKSHAR". The main area is titled "SQL Commands" with a "Language" dropdown set to "SQL". The command entered is "create table employees2(id number(7), first_name varchar2(25), last_name varchar2(25), salary number(7), dept_id number(7));". The results section shows the output: "Table created." and "0.04 seconds".

5. Drop table emp.

QUERY: Drop table emp;

OUTPUT:

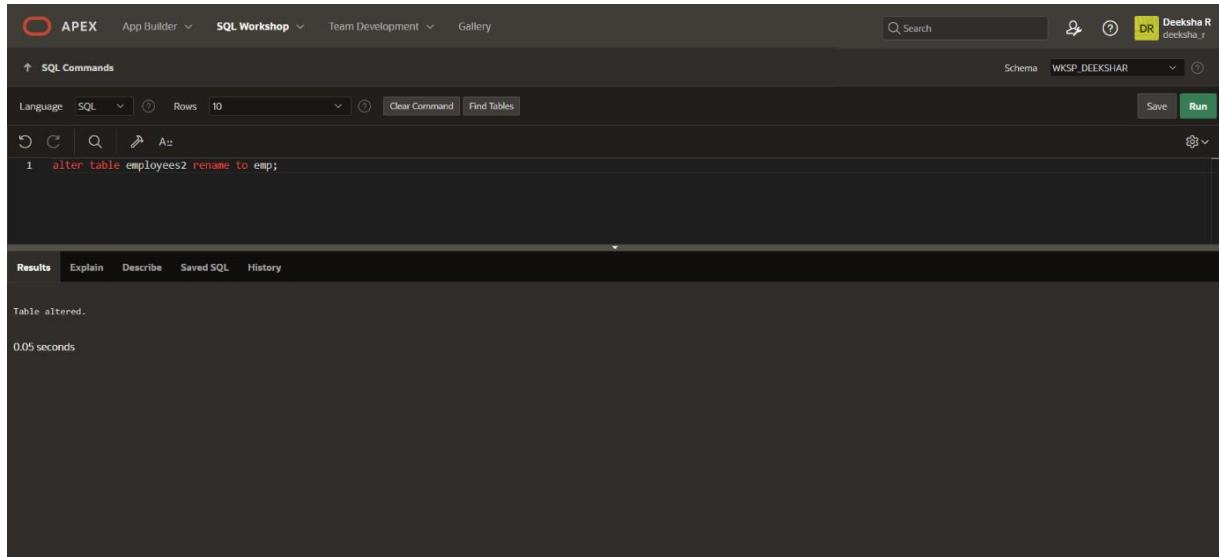


A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar shows "APEX", "App Builder", "SQL Workshop", "Team Development", and "Gallery". The right side shows a user profile for "Deeksha R" and the schema "WKSP_DEEKSHAR". The main area is titled "SQL Commands" with a "Language" dropdown set to "SQL". The command entered is "drop table emp;". The results section shows the output: "Table dropped." and "0.09 seconds".

6.Rename the EMPLOYEES2 table as EMP.

QUERY: alter table employees2 rename to emp;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A single line of SQL code is entered: 'alter table employees2 rename to emp;'. Below the command, the output shows 'Table altered.' and a execution time of '0.05 seconds'. The schema is set to 'WKSP_DEEKSHAR'.

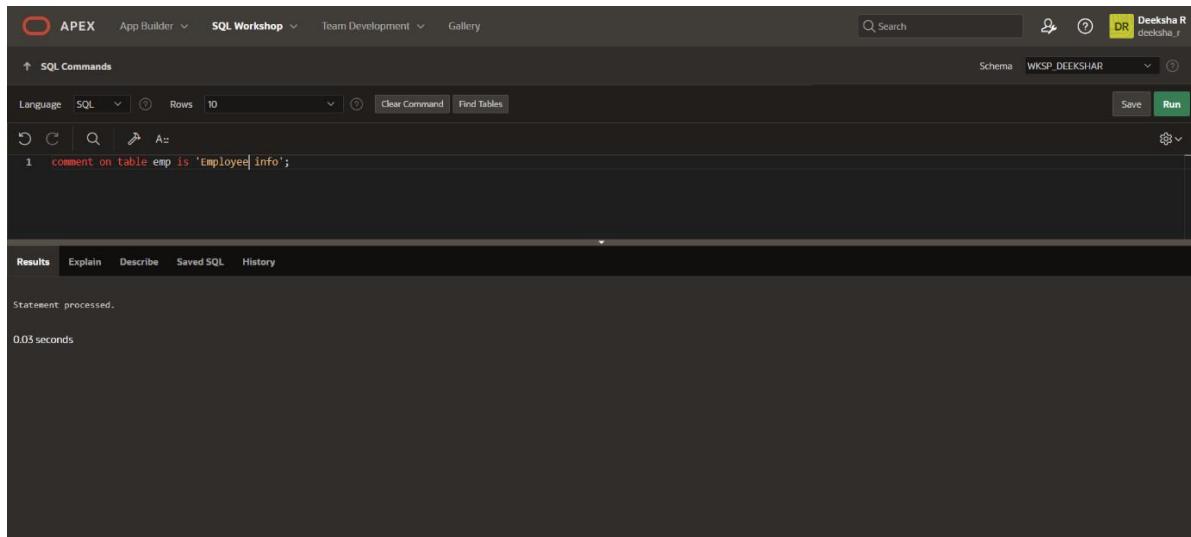
```
1 alter table employees2 rename to emp;
```

Table altered.
0.05 seconds

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

QUERY: comment on table dept is 'Employee info';

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A single line of SQL code is entered: 'comment on table dept is 'Employee info'';. Below the command, the output shows 'Statement processed.' and a execution time of '0.03 seconds'. The schema is set to 'WKSP_DEEKSHAR'.

```
1 comment on table dept is 'Employee info';
```

Statement processed.
0.03 seconds

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

QUERY: Alter table emp drop column first_name;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands tab, the command `alter table emp drop column first_name;` is entered. The Results tab displays the output: `Table altered.` and `0.06 seconds`.

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

RESULT:

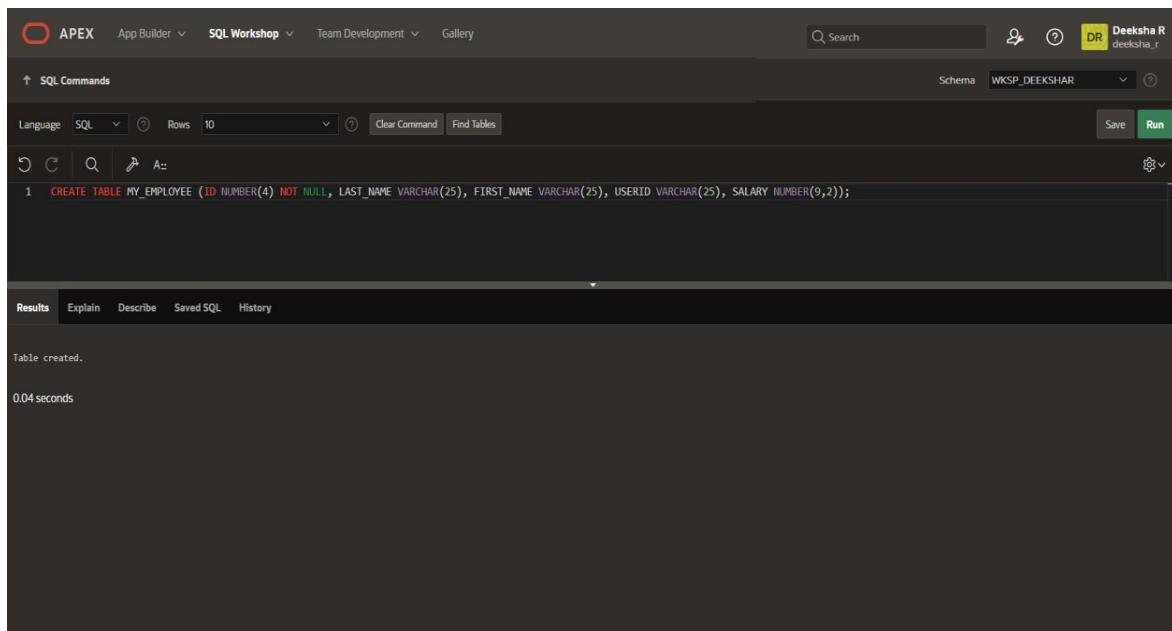
EXNO:2**MANIPULATING DATA****DATE:**

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)

QUERY: Create table My_employee(ID number(4) not null, last_name varchar(25),first_name varchar(25),userid varchar(25),salary number(9,2));

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A single line of SQL code is entered in the command input field:

```
1 CREATE TABLE MY_EMPLOYEE (ID NUMBER(4) NOT NULL, LAST_NAME VARCHAR(25), FIRST_NAME VARCHAR(25), USERID VARCHAR(25), SALARY NUMBER(9,2));
```

Below the command, the results section displays the output:

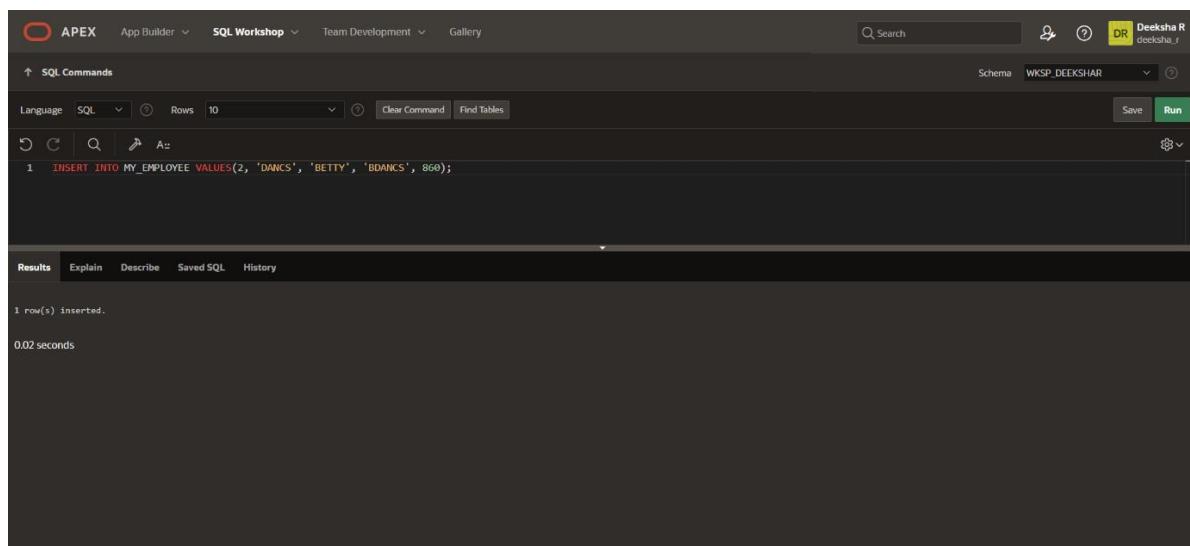
Table created.
0.04 seconds

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

QUERY: insert into my_employee(1,'patel','ralph','rpatel',895); insert into my_employee(2,'dancs','betty','bdancs',860);

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area displays a SQL command window with the following content:

```
1 INSERT INTO MY_EMPLOYEE VALUES(2, 'DANCS', 'BETTY', 'BDANCS', 860);
```

Below the command window, the 'Results' tab is active, showing the output of the query:

```
1 row(s) inserted.  
0.02 seconds
```

3.Display the table with values.

QUERY: select * from My_Employee;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the command `SELECT * FROM MY_EMPLOYEE;` is entered. The Results pane displays the following data:

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
2	DANCS	BETTY	BDANCS	860
1	PATEL	RALPH	RPATEL	895

2 rows returned in 0.02 seconds [Download](#)

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

QUERY: update My_employee set lname="drexler" where id=3;

OUTPUT:

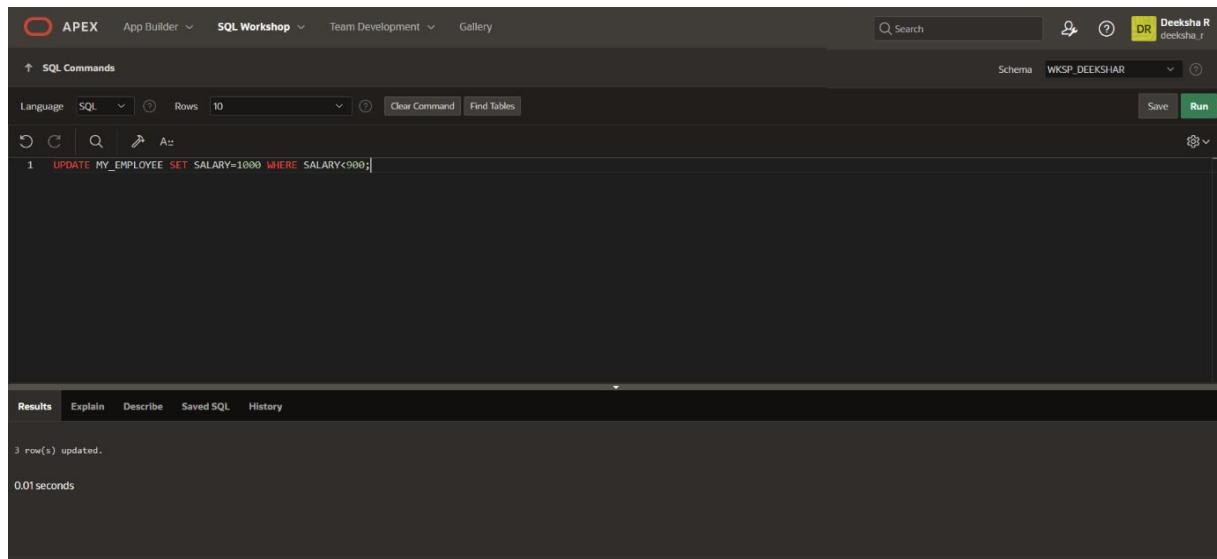
The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the command `UPDATE MY_EMPLOYEE SET LAST_NAME='DREXLER' WHERE ID=3;` is entered. The Results pane displays the following message:

1 row(s) updated.
0.04 seconds

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

QUERY: update My_Employee set salary=1000 where salary<900;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The SQL Commands tab is active, displaying the following command:

```
1 UPDATE MY_EMPLOYEE SET SALARY=1000 WHERE SALARY<900;
```

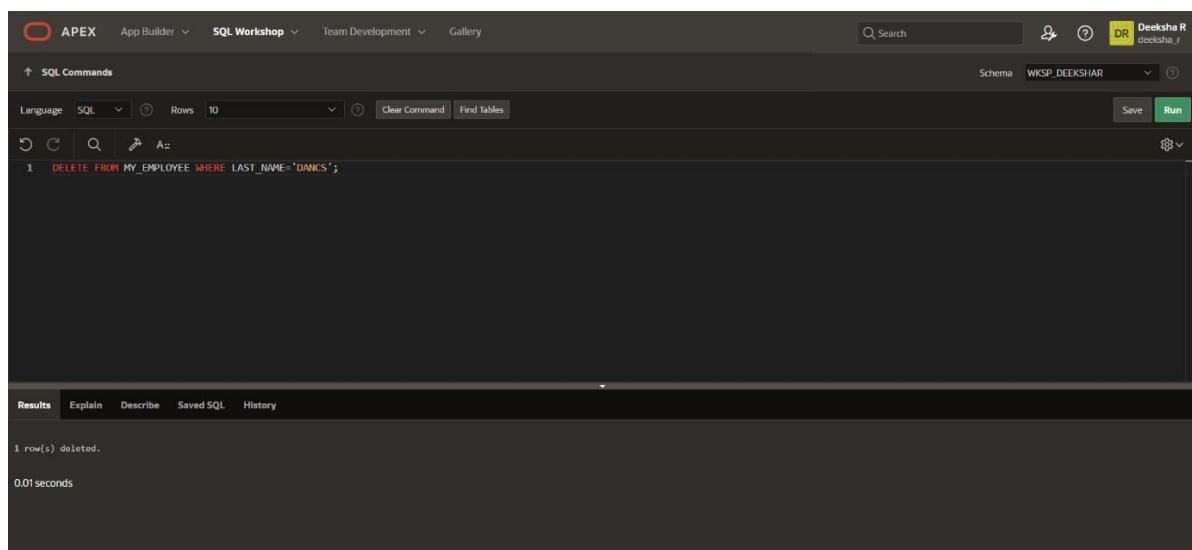
Below the command, the results show:

3 row(s) updated.
0.01 seconds

6.Delete Betty dancs from MY_EMPLOYEE table.

QUERY: delete from My_Employee where last_name='dancs';

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The SQL Commands tab is active, displaying the following command:

```
1 DELETE FROM MY_EMPLOYEE WHERE LAST_NAME='DANCS';
```

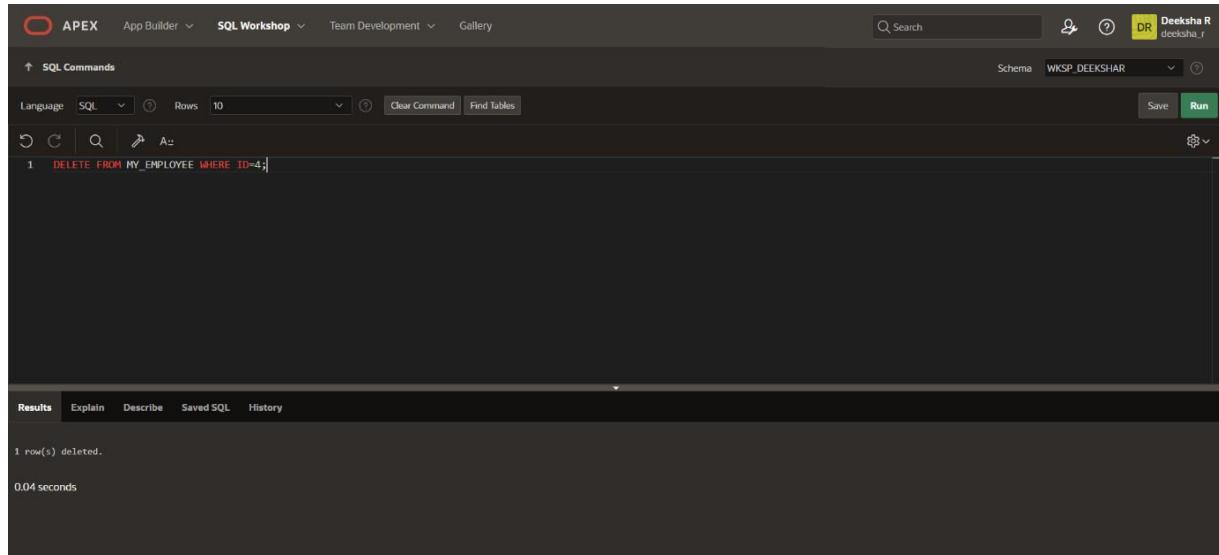
Below the command, the results show:

1 row(s) deleted.
0.01 seconds

7.Empty the fourth row of the emp table.

QUERY: delete from My_Employee where id=4;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A single line of SQL code is entered: 'DELETE FROM MY_EMPLOYEE WHERE ID=4;'. Below the command, the results tab is selected, showing the output: '1 row(s) deleted.' and '0.04 seconds' execution time.

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

RESULT:

EXNO:3

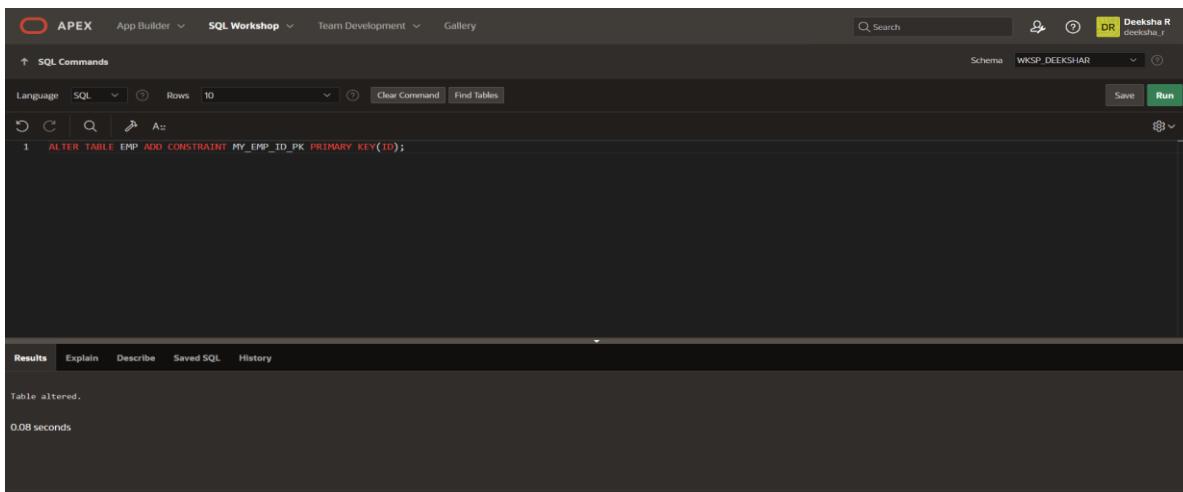
INCLUDING CONSTRAINTS

DATE:

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.

QUERY: alter table emp add constraint my_emp_id_pk primary key(id);

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following command is entered and executed:

```
1 ALTER TABLE EMP ADD CONSTRAINT MY_EMP_ID_PK PRIMARY KEY(ID);
```

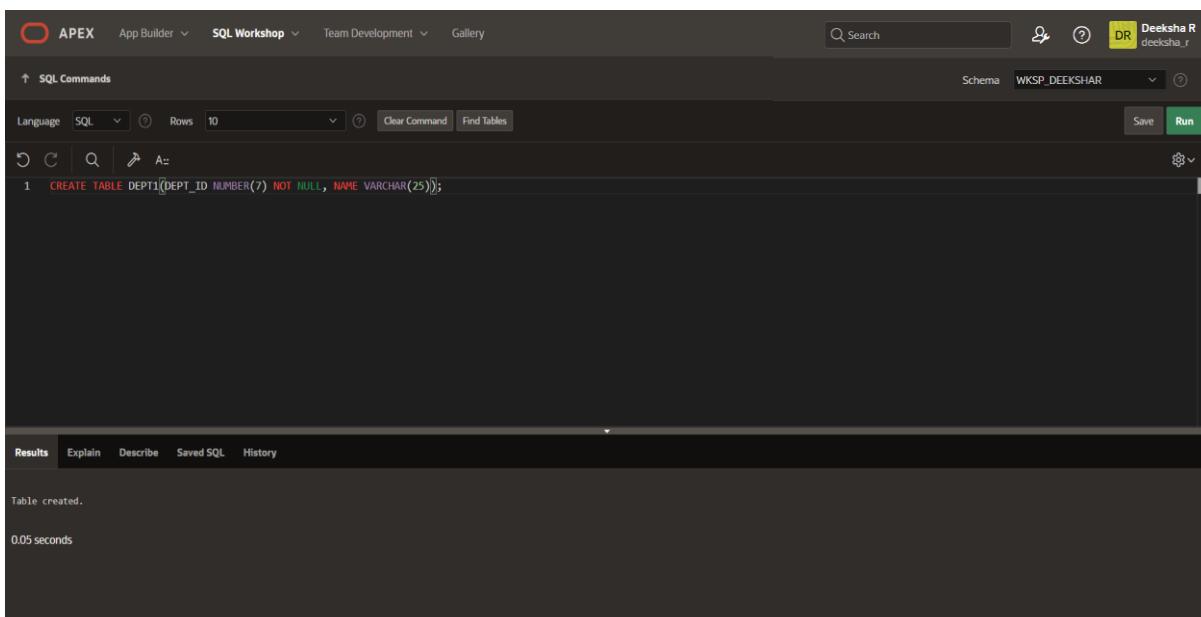
The Results pane displays the output:

```
Table altered.  
0.08 seconds
```

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

QUERY: create table dept1(dept_id number(7) not null, name varchar(25));

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following command is entered and executed:

```
1 CREATE TABLE DEPT1(DEPT_ID NUMBER(?) NOT NULL, NAME VARCHAR(25));
```

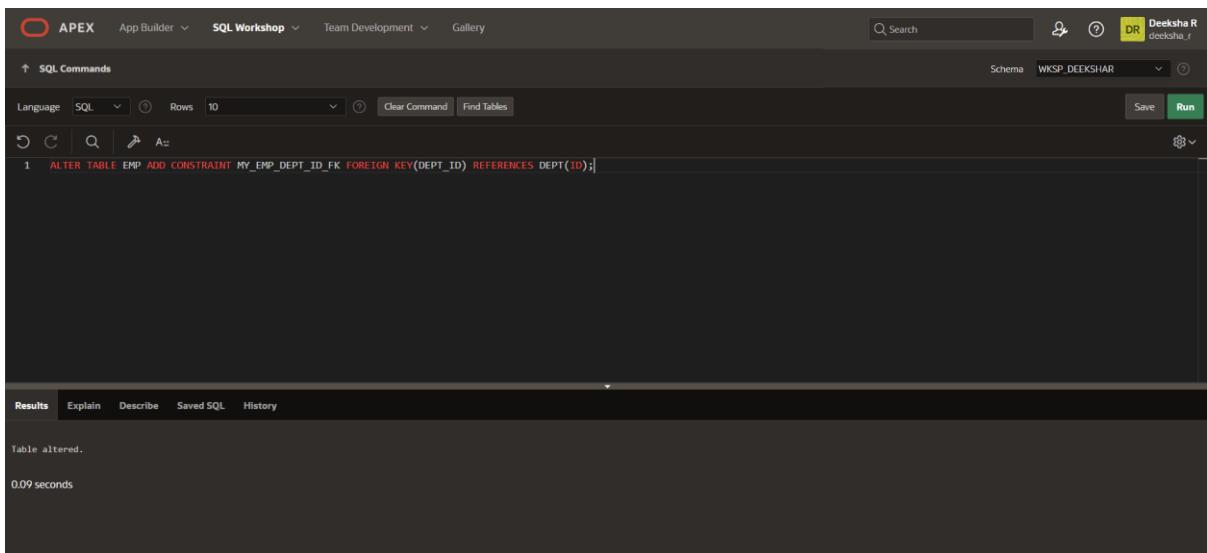
The Results pane displays the output:

```
Table created.  
0.05 seconds
```

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.

QUERY: alter table emp add constraint my_emp_dept_id_fk foreign key(dept_id) references dept(id);

OUTPUT:

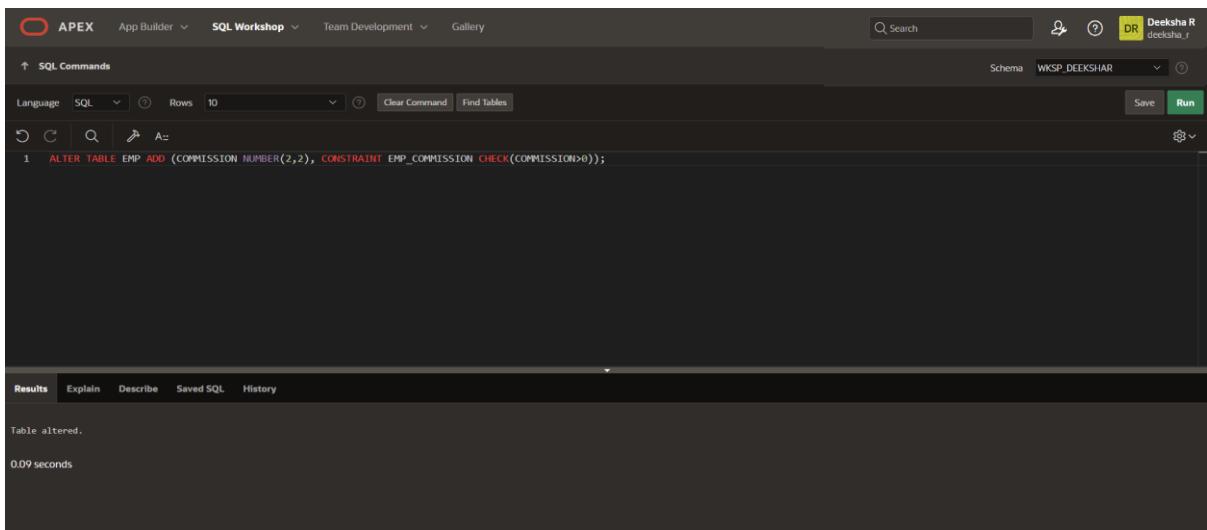


The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A single line of SQL code is entered in the command input field: 'ALTER TABLE EMP ADD CONSTRAINT MY_EMP_DEPT_ID_FK FOREIGN KEY(DEPT_ID) REFERENCES DEPT(ID);'. Below the input field, there are buttons for 'Save' and 'Run'. The results section is active, displaying the output of the command. The output shows the message 'Table altered.' and a timestamp '0.09 seconds'.

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.

QUERY: alter table emp add (commission number(2,2), constraint emp_commission check(commission>0));

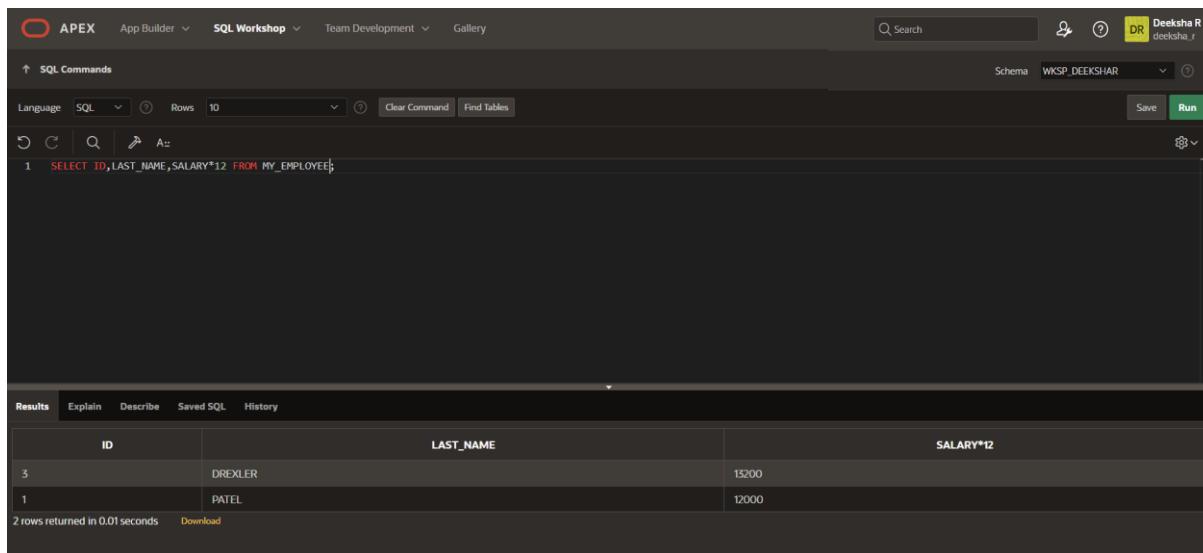
OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A single line of SQL code is entered in the command input field: 'ALTER TABLE EMP ADD (COMMISSION NUMBER(2,2), CONSTRAINT EMP_COMMISION CHECK(COMMISSION>0));'. Below the input field, there are buttons for 'Save' and 'Run'. The results section is active, displaying the output of the command. The output shows the message 'Table altered.' and a timestamp '0.09 seconds'.

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

RESULT:

EXNO:4**WRITING BASIC SQL SELECT STATEMENTS****DATE:****1.Identify the Errors****SELECT employee_id, last_name ,sal*12 ANNUAL SALARY FROM employees;****QUERY: select id,last_name,salary*12 from my_employee;****OUTPUT:**

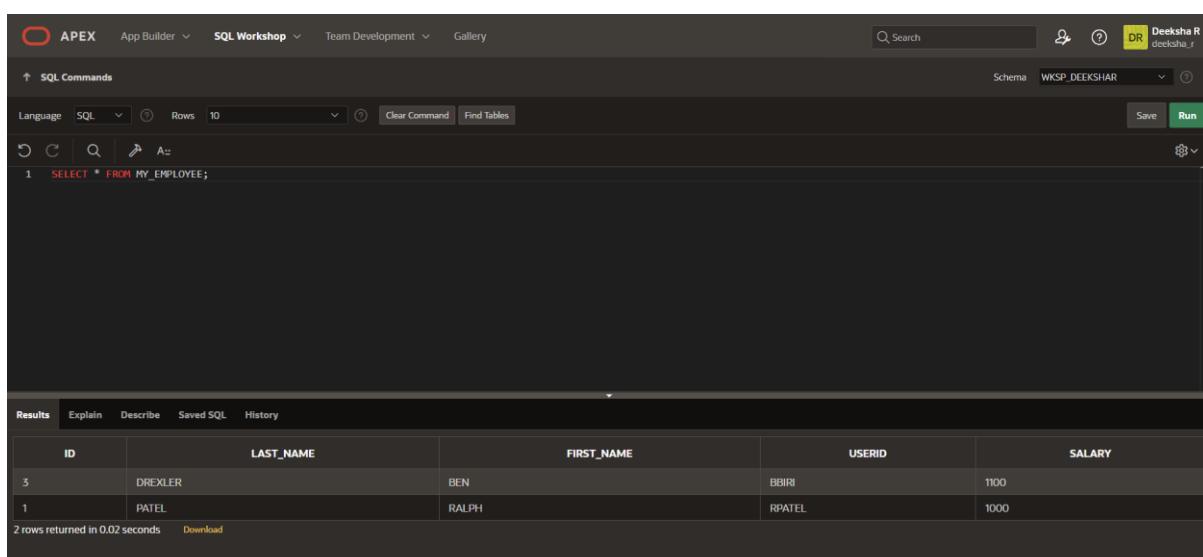
The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following query is entered:

```
1 SELECT ID, LAST_NAME, SALARY*12 FROM MY_EMPLOYEE;
```

In the Results pane, the output is displayed in a table:

ID	LAST_NAME	SALARY*12
3	DREXLER	15200
1	PATEL	12000

2 rows returned in 0.01 seconds

2. Show the structure of departments the table. Select all the data from it.**QUERY: select * from my_employee;****OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following query is entered:

```
1 SELECT * FROM MY_EMPLOYEE;
```

In the Results pane, the output is displayed in a table:

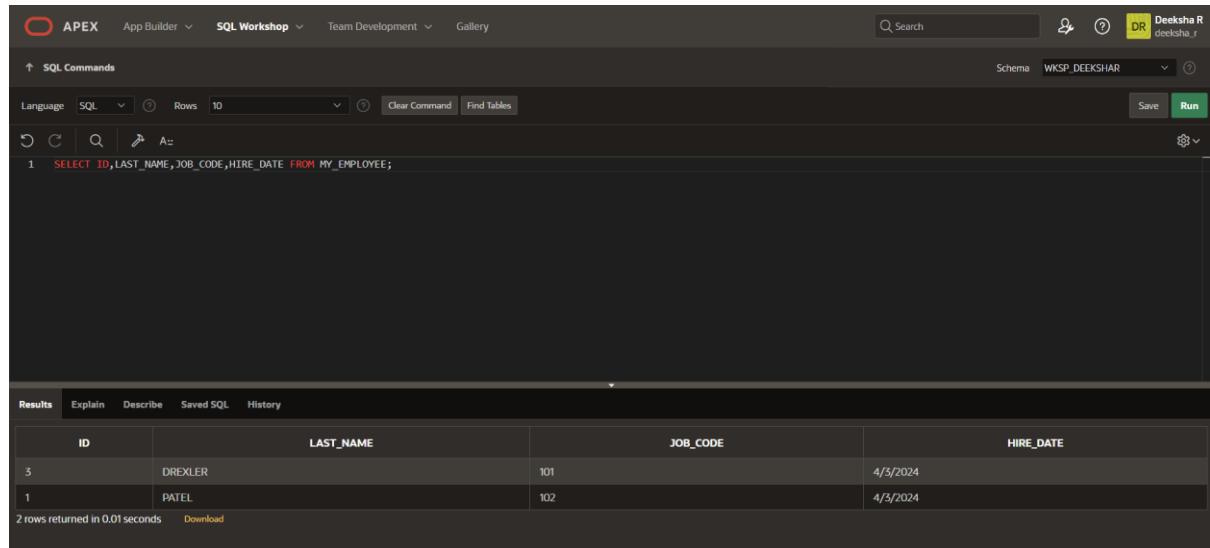
ID	LAST_NAME	FIRST_NAME	USERID	SALARY
3	DREXLER	BEN	BBIRI	1100
1	PATEL	RALPH	RPATEL	1000

2 rows returned in 0.02 seconds

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

QUERY: select id,last_name,job_code,hire_date from my_employee;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following query is entered:

```
1  SELECT ID, LAST_NAME, JOB_CODE, HIRE_DATE FROM MY_EMPLOYEE;
```

In the Results pane, the output is displayed as a table:

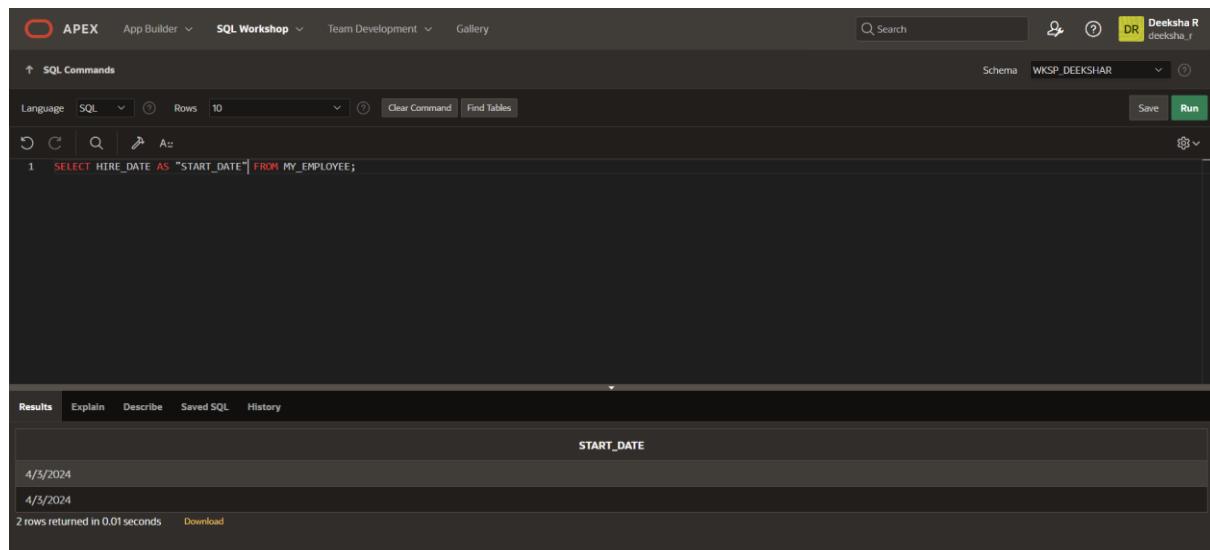
ID	LAST_NAME	JOB_CODE	HIRE_DATE
3	DREXLER	101	4/3/2024
1	PATEL	102	4/3/2024

Below the table, it says "2 rows returned in 0.01 seconds".

4. Provide an alias STARTDATE for the hire date.

QUERY: Select hire_date as “start_date” from my_employee;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following query is entered:

```
1  SELECT HIRE_DATE AS "START_DATE" FROM MY_EMPLOYEE;
```

In the Results pane, the output is displayed as a table:

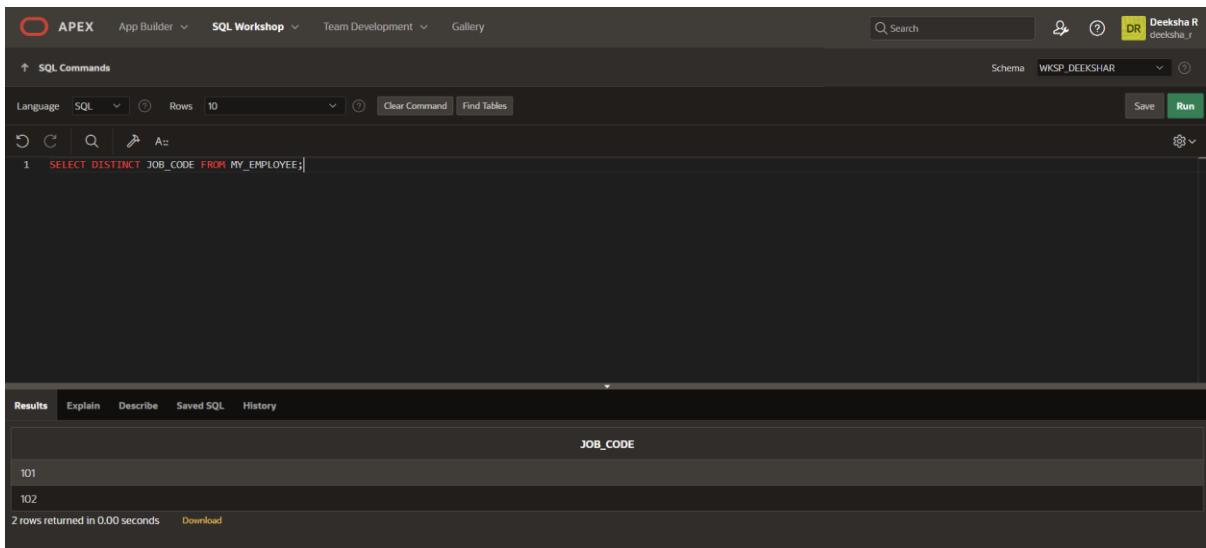
START_DATE
4/3/2024
4/3/2024

Below the table, it says "2 rows returned in 0.01 seconds".

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

QUERY: select distinct job_code from my_employee;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the command `SELECT DISTINCT JOB_CODE FROM MY_EMPLOYEE;` is entered. The Results pane displays the output:

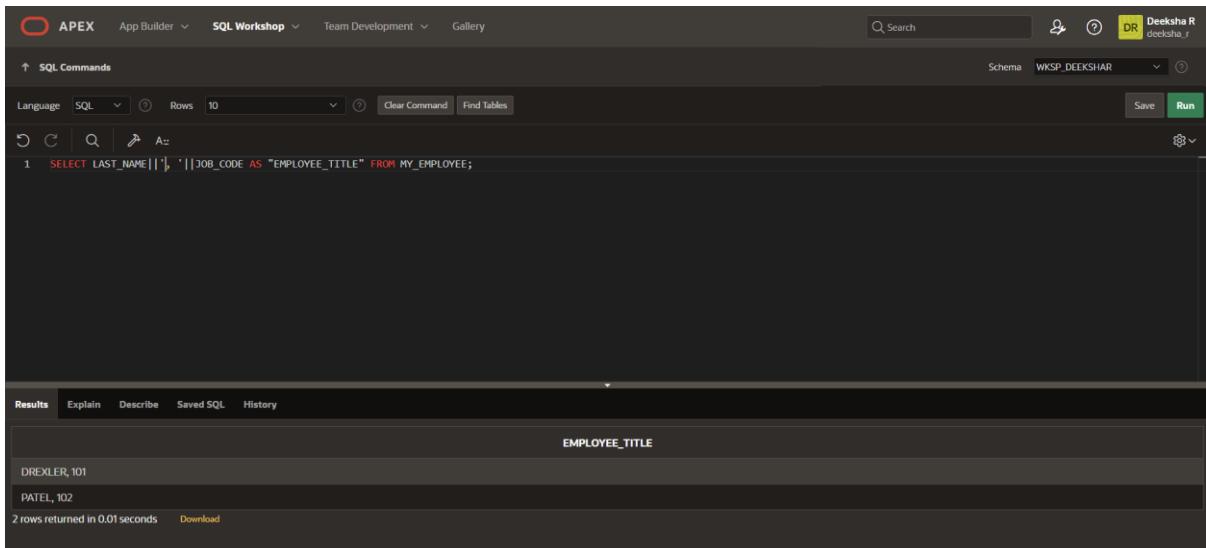
JOB_CODE
101
102

2 rows returned in 0.00 seconds

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

QUERY: select last_name||', '|job_code as "employee_title" from my_employee;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the command `SELECT LAST_NAME||', '|JOB_CODE AS "EMPLOYEE_TITLE" FROM MY_EMPLOYEE;` is entered. The Results pane displays the output:

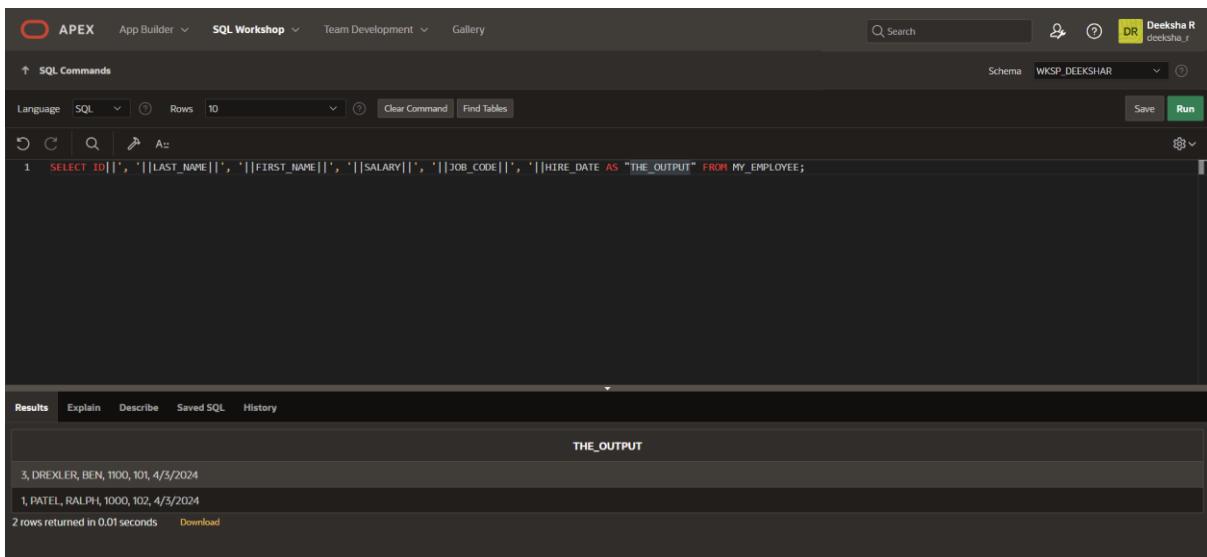
EMPLOYEE_TITLE
DREXLER, 101
PATEL, 102

2 rows returned in 0.01 seconds

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

QUERY: select id||', '||last_name||', '||first_name||', '||salary||', '||job_code||', '||hire_date as "the_output" from my_employee;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, a query is entered:

```
1 SELECT ID||', '||LAST_NAME||', '||FIRST_NAME||', '||SALARY||', '||JOB_CODE||', '||HIRE_DATE AS "THE_OUTPUT" FROM MY_EMPLOYEE;
```

In the Results pane, the output is displayed in a table with one column labeled "THE_OUTPUT". The data shows two rows of employee information:

THE_OUTPUT
3, DREXLER, BEN, 1100, 101, 4/3/2024
1, PATEL, RALPH, 1000, 102, 4/3/2024

Below the table, it says "2 rows returned in 0.01 seconds" and there is a "Download" link.

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

RESULT:

EXNO:5

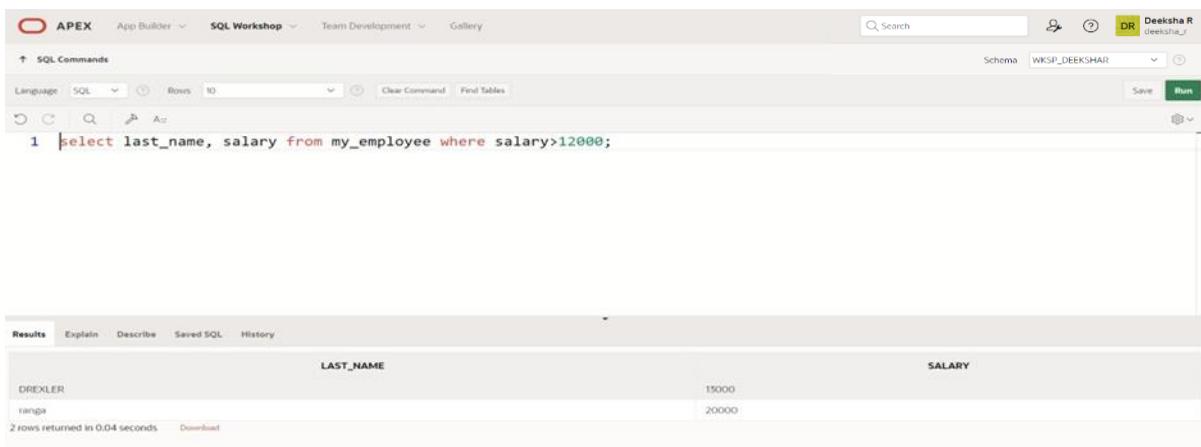
RESTRICTING AND SORTING DATA

DATE:

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

QUERY: select last_name, salary from My_employee where salary > 12000;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following query is entered:

```
1 select last_name, salary from my_employee where salary>12000;
```

In the Results pane, the output is displayed as a table:

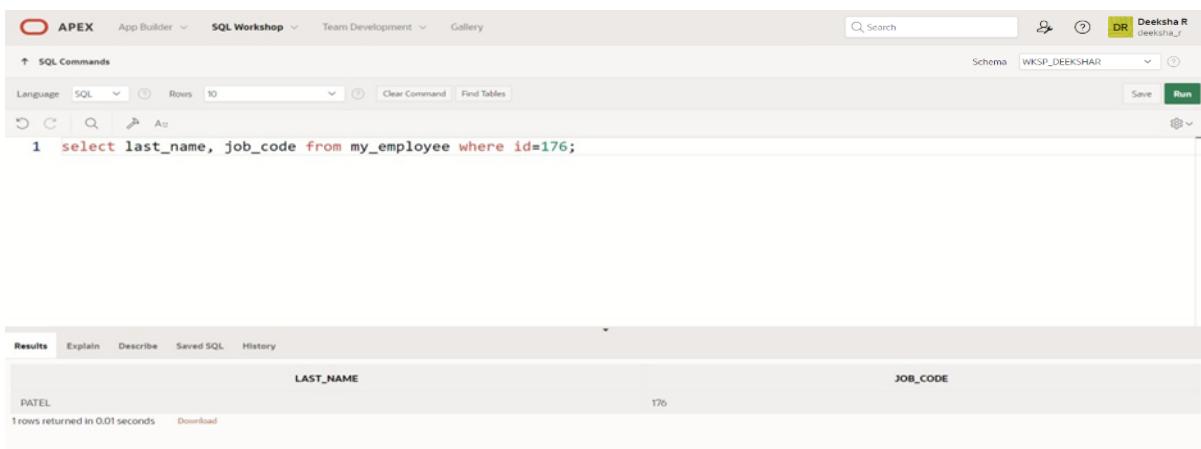
LAST_NAME	SALARY
DREXLER	15000
ranga	20000

Below the table, it says "2 rows returned in 0.04 seconds".

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

QUERY: select last_name, dept_id from My_employee where emp_id = 176;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following query is entered:

```
1 select last_name, job_code from my_employee where id=176;
```

In the Results pane, the output is displayed as a table:

LAST_NAME	JOB_CODE
PATEL	176

Below the table, it says "1 rows returned in 0.01 seconds".

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)

QUERY: select last_name, salary from My_employee where salary>12000 or salary<5000;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
1 select last_name, salary from my_employee where salary not between 5000 and 12000;
```

The results table displays the following data:

LAST_NAME	SALARY
DREXLER	13000
PATEL	1000
ranga	20000

3 rows returned in 0.03 seconds

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

QUERY: select last_name, dept_id from My_employee where dept_id in (20,50) order by last_name;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
1 select last_name, dept_id from My_employee where dept_id in (20,50) order by last_name;
```

The results table displays the following data:

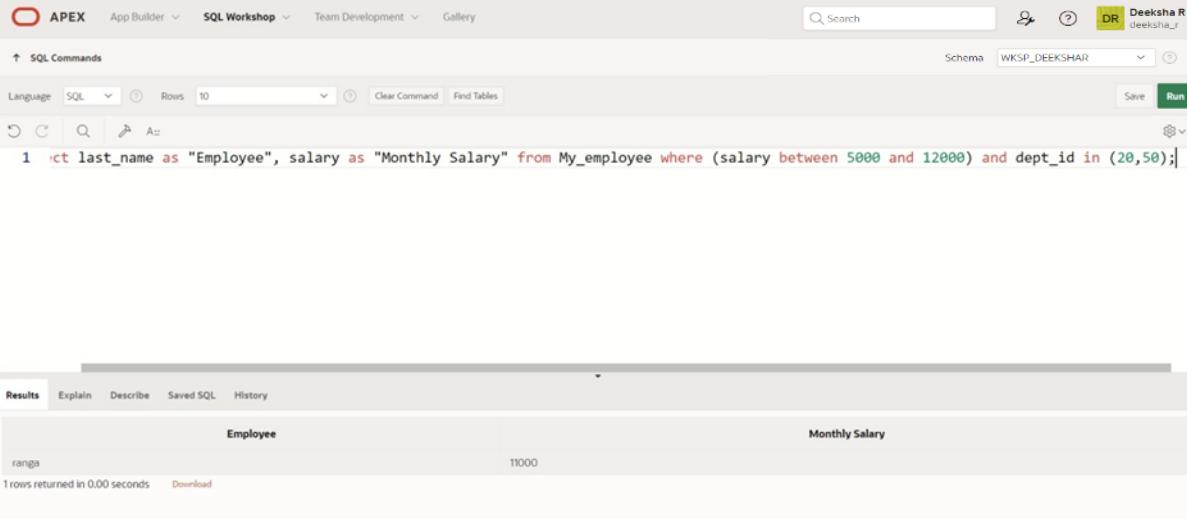
LAST_NAME	DEPT_ID
PATEL	50
ranga	20

2 rows returned in 0.00 seconds

5. 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)

QUERY: select last_name as “Employee”, salary as “Monthly Salary” from My_employee where (salary between 5000 and 12000) and dept_id in (20,50);

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'Deeksha R' (deeksha_r). The main area has a search bar and a schema dropdown set to 'WKSP_DEEKSHAR'. The SQL command window contains the following code:

```
1 select last_name as "Employee", salary as "Monthly Salary" from My_employee where (salary between 5000 and 12000) and dept_id in (20,50);
```

The results section shows the output:

Employee	Monthly Salary
ranga	11000

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

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

QUERY: select last_name, hire_date from My_employee where hire_date like ‘%94’;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'Deeksha R' (deeksha_r). The main area has a search bar and a schema dropdown set to 'WKSP_DEEKSHAR'. The SQL command window contains the following code:

```
1 select last_name, hire_date from My_employee where hire_date like '%94';
```

The results section shows the output:

LAST_NAME	HIRE_DATE
DREXLER	4/5/1994
PATEL	4/27/1994

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

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

QUERY: select last_name, job_title from My_employee where manager_id is null;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
1 select last_name, job_title from My_employee where manager_id is null;
```

The results table displays two rows:

LAST_NAME	JOB_TITLE
DREXLER	
PATEL	manager
ranga	-

3 rows returned in 0.02 seconds

8.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)

QUERY: select last_name, salary, commission from My_employee where commission is not null order by salary, commission;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
1 select last_name, salary, commission from My_employee where commission is not null order by salary, commission;
```

The results table displays two rows:

LAST_NAME	SALARY	COMMISSION
DREXLER	7000	20
ranga	11000	50

2 rows returned in 0.02 seconds

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

QUERY: select last_name from My_employee where last_name like '_____a%';

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
1 select last_name from My_employee where last_name like '_____a%';
```

The results section shows a single row returned:

LAST_NAME
rrema

1 rows returned in 0.01 seconds

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

QUERY: select last_name from My_employee where last_name like '%a%' and last_name like '%e%';

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
1 select last_name from My_employee where last_name like '%a%' and last_name like '%e%';
```

The results section shows a single row returned:

LAST_NAME
patel

1 rows returned in 0.00 seconds

11. 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)

QUERY: select last_name, job_title, salary from My_employee where job_title in('sales','clerk') and salary not in(2500,3500,7000);

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
1 select last_name, job_title, salary from My_employee where job_title in('sales','clerk') and salary not in(2500,3500,7000);
```

The results section displays the following data:

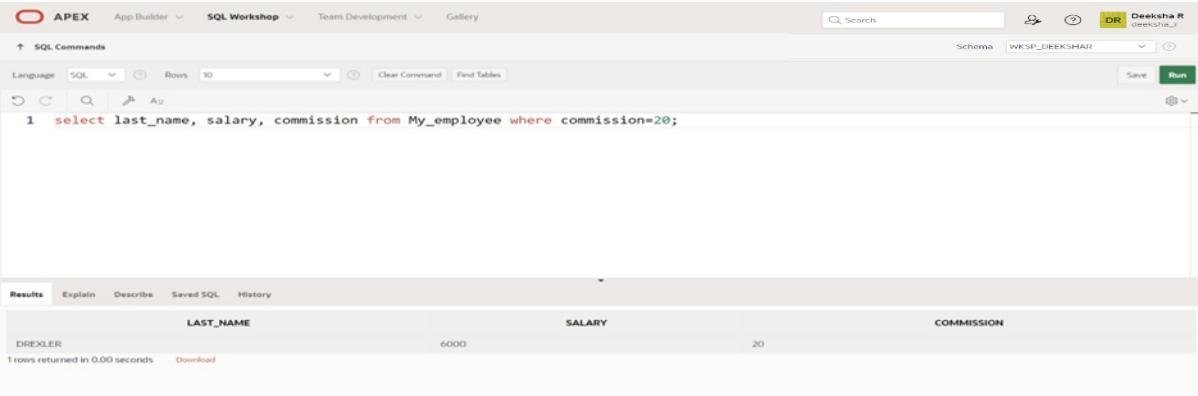
LAST_NAME	JOB_TITLE	SALARY
DREXLER	sales	6000
rrnaga	clerk	11000

2 rows returned in 0.01 seconds

12. Display the last name, salary, and commission for all employees whose commission amount is 20%. (hints: use predicate logic)

QUERY: select last_name, salary, commission from My_employee where commission=20;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
1 select last_name, salary, commission from My_employee where commission=20;
```

The results section displays the following data:

LAST_NAME	SALARY	COMMISSION
DREXLER	6000	20

1 rows returned in 0.00 seconds

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

RESULT:

EXNO:6

SINGLE ROW FUNCTIONS

DATE:

1. Write a query to display the current date. Label the column Date.

QUERY: select sysdate as "DATE" from dual;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following query is entered:

```
1 select sysdate from dual;
2
```

In the Results pane, the output is displayed in a table:

SYSDATE
04/04/2024

Below the table, it says "1 rows returned in 0.02 seconds".

2. The HR department needs a report to display the employee number, last name, salary, and increased by 15.5% (expressed as a whole number) for each employee. Label the column New Salary.

QUERY: select emp_id, lname, salary, salary+(salary*15.5/100) as "New Salary" from My_emp;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following query is entered:

```
1 | select id, lname, salary, salary+(salary*15.5/100) as "New Salary" from My_emp;
```

In the Results pane, the output is displayed in a table:

ID	LNAME	SALARY	New Salary
2	Esh	400000	462000
5	Kumar	1200000	1386000
1	Narayan	2000000	2310000
4	Sha	600000	693000
6	Gavl	10000	11550
3	Partha	700000	808500

Below the table, it says "6 rows returned in 0.01 seconds".

3. Modify your query lab_03_02.sql to add a column that subtracts the old salary from

the new salary. Label the column Increase.

QUERY: select emp_id, lname, salary, salary+(salary*15.5/100) as "New Salary", (salary+(salary*15.5/100))-salary as "Increase" from My_emp;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The query entered is:

```
1 select id, lname, salary, salary+(salary*15.5/100) as "New Salary", (salary+(salary*15.5/100))-salary as "Increase" from My_emp;
```

The results section displays the following data:

ID	LNAME	SALARY	New Salary	Increase
2	Esh	400000	462000	62000
5	Kumar	1200000	1386000	186000
1	Narayan	2000000	2310000	310000
4	Sha	600000	693000	93000
6	Gavi	10000	11550	1550
3	Partha	700000	808500	108500

6 rows returned in 0.05 seconds [Download](#)

4. Write a query that displays the last name (with the first letter uppercase and all other letters lowercase) and the length of the last name for all employees whose name starts with the letters J, A, or M. Give each column an appropriate label. Sort the results by the employees' last names.

QUERY: select initcap(lname) as "Name", length(lname) as "Length of Name" from My_emp where lname like 'J%' or lname like 'A%' or lname like 'M%' order by lname;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The query entered is:

```
1 select initcap(lname) as "Name", length(lname) as "Length of Name" from My_emp where lname like 'J%' or lname like 'A%' or lname like 'M%' order by lname;
```

The results section displays the following data:

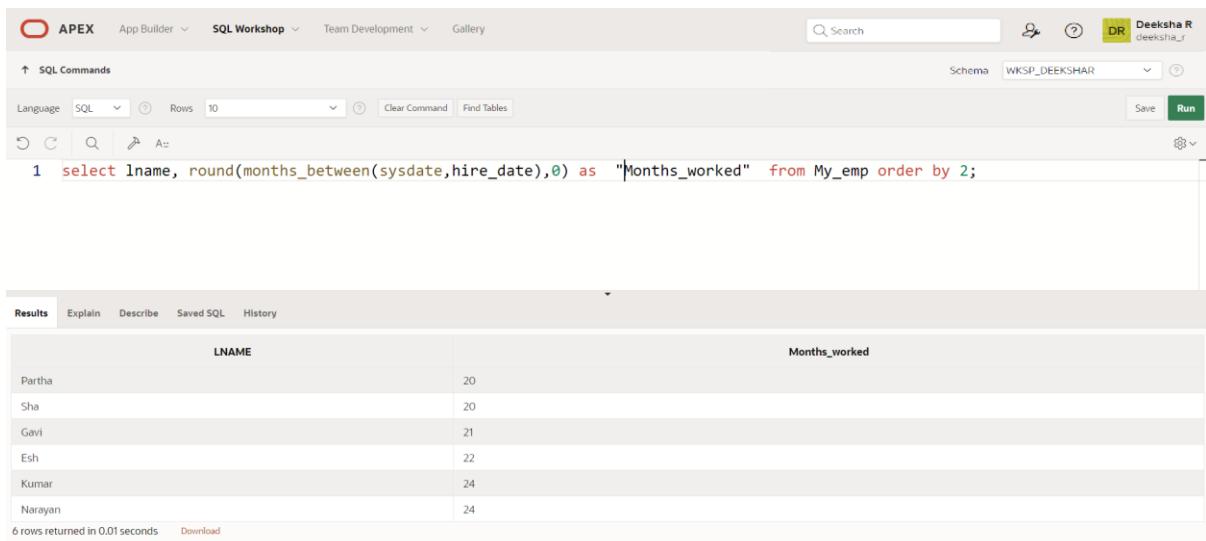
Name	Length of Name
	no data found

5. The HR department wants to find the length of employment for each employee. For each employee, display the last name and calculate the number of months between

today and the date on which the employee was hired. Label the column MONTHS_WORKED. Order your results by the number of months employed. Round the number of months up to the closest whole number.

QUERY: select lname, round(months_between(sysdate,h_date),0) as "Months_worked" from My_emp order by 2;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
1 select lname, round(months_between(sysdate,hire_date),0) as "Months_worked" from My_emp order by 2;
```

The results section displays the following data:

LNAME	Months_worked
Partha	20
Sha	20
Gavi	21
Esh	22
Kumar	24
Narayan	24

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

6. Create a report that produces the following for each employee:<employee last name> earns <salary> monthly but wants <3 times salary>. Label the column Dream Salaries.

QUERY: select lname||' earns \$'||salary||' monthly but wants \$'||salary*3 as "Dream Salary" from My_emp;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
1 select lname||' earns $'||salary||' monthly but wants $'||salary*3 as "Dream Salary" from My_emp;
```

The results section displays the following data:

Dream Salary
Esh earns \$400000 monthly but wants \$1200000
Kumar earns \$1200000 monthly but wants \$3600000
Narayan earns \$2000000 monthly but wants \$6000000
Sha earns \$600000 monthly but wants \$1800000
Gavi earns \$100000 monthly but wants \$300000
Partha earns \$700000 monthly but wants \$2100000

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

7. Create a query to display the last name and salary for all employees. Format the

salary to be 15 characters long, left-padded with the \$ symbol. Label the column SALARY.

QUERY: select lname, lpad(salary,15,'\$') Salary from My_emp;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
1 select lname, lpad(salary,15,'$') Salary from My_emp;
```

The results section displays the following data:

LNAME	SALARY
Esh	\$\$\$\$\$\$\$400000
Kumar	\$\$\$\$\$\$\$1200000
Narayan	\$\$\$\$\$\$\$2000000
Sha	\$\$\$\$\$\$\$6000000
Gavi	\$\$\$\$\$\$\$100000
Partha	\$\$\$\$\$\$\$700000

6 rows returned in 0.01 seconds

8. Display each employee's last name, hire date, and salary review date, which is the first Monday after six months of service. Label the column REVIEW. Format the dates to appear in the format similar to "Monday, the Thirty-First of July, 2000."

QUERY: select lname, h_date, to_char((next_day(h_date,'Monday')),'fmday," the "ddspth "of" month,yyyy') as "REVIEW" from My_emp;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
1 select lname, hire_date, to_char((next_day(hire_date,'Monday')),'fmday," the "ddspth "of" month,yyyy') as "REVIEW" from My_emp;
```

The results section displays the following data:

LNAME	HIRE_DATE	REVIEW
Esh	6/4/2022	monday, the sixth of june,2022
Kumar	4/2/2022	monday, the fourth of april,2022
Narayan	4/2/2022	monday, the fourth of april,2022
Sha	8/3/2022	monday, the eighth of august,2022
Gavi	7/23/2022	monday, the twenty-fifth of july,2022
Partha	8/3/2022	monday, the eighth of august,2022

6 rows returned in 0.01 seconds

9. Display the last name, hire date, and day of the week on which the employee started. Label the column DAY. Order the results by the day of the week, starting with

Monday.

QUERY: select Lname, h_date, to_char(h_date,'Day')as "Day" from My_emp order by to_char(h_date-1,'d');

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. At the top, there are tabs for APEX, App Builder, SQL Workshop (which is selected), Team Development, and Gallery. On the right, there's a user profile for 'Deeksha R' (deeksha_r). Below the tabs, there's a toolbar with icons for Undo, Redo, Search, and Run. The main area has a 'SQL Commands' tab selected. The SQL command entered is: `1 select lname, hire_date from My_emp;`. The results section shows a table with two columns: LNAME and HIRE_DATE. The data is as follows:

LNAME	HIRE_DATE
Esh	7/13/1998
Kumar	7/13/2000
Narayan	7/13/1993
Sha	7/13/1998
Gavi	7/13/1994
Partha	7/13/2001

At the bottom of the results pane, it says '6 rows returned in 0.01 seconds' and has a 'Download' link.

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

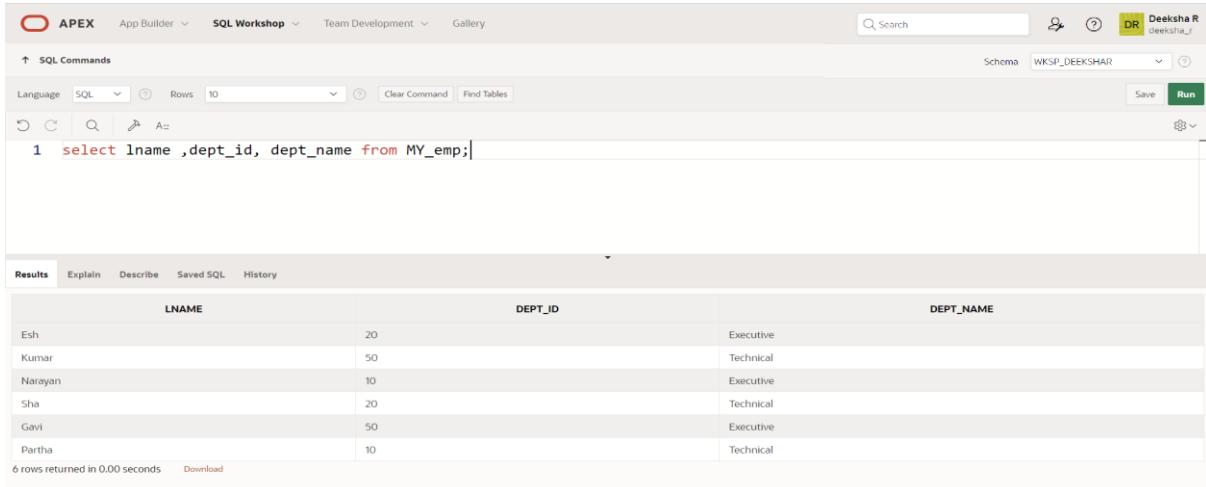
RESULT:

EXNO:7**DISPLAYING DATA FROM MULTIPLE TABLES****DATE:**

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

QUERY: select lname ,dept_id, dept_name from MY_emp;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the query `select lname ,dept_id, dept_name from MY_emp;` is entered. The Results pane displays the output in a table:

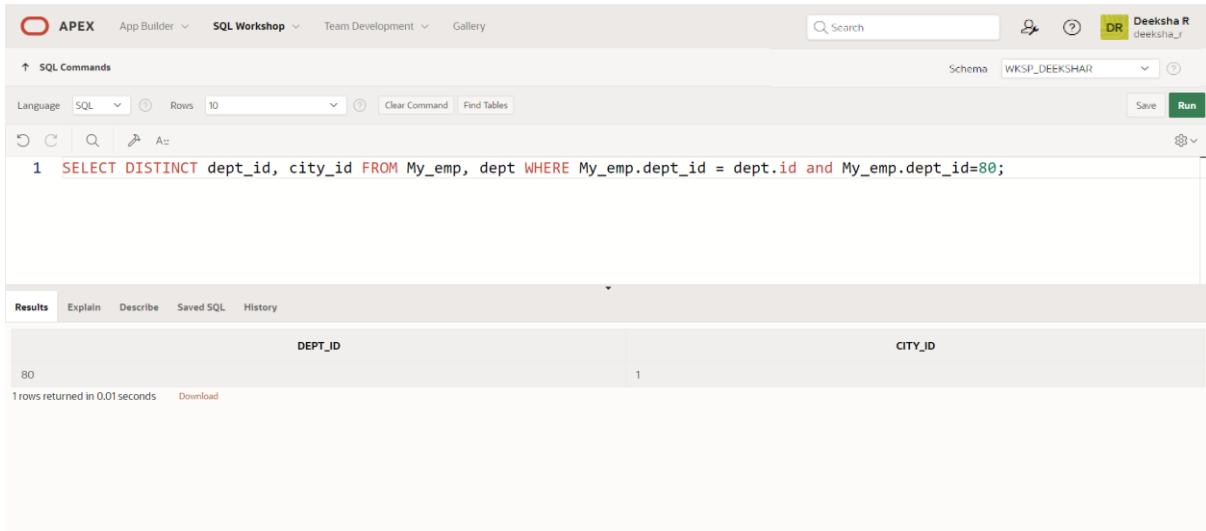
LNAME	DEPT_ID	DEPT_NAME
Esh	20	Executive
Kumar	50	Technical
Narayan	10	Executive
Sha	20	Technical
Gavi	50	Executive
Partha	10	Technical

6 rows returned in 0.00 seconds

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

QUERY: SELECT DISTINCT job_id, loc_id FROM My_emp, dept WHERE My_emp.dept_id = dept.dept_id and My_emp.d_id=80;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the query `SELECT DISTINCT dept_id, city_id FROM My_emp, dept WHERE My_emp.dept_id = dept.id and My_emp.dept_id=80;` is entered. The Results pane displays the output in a table:

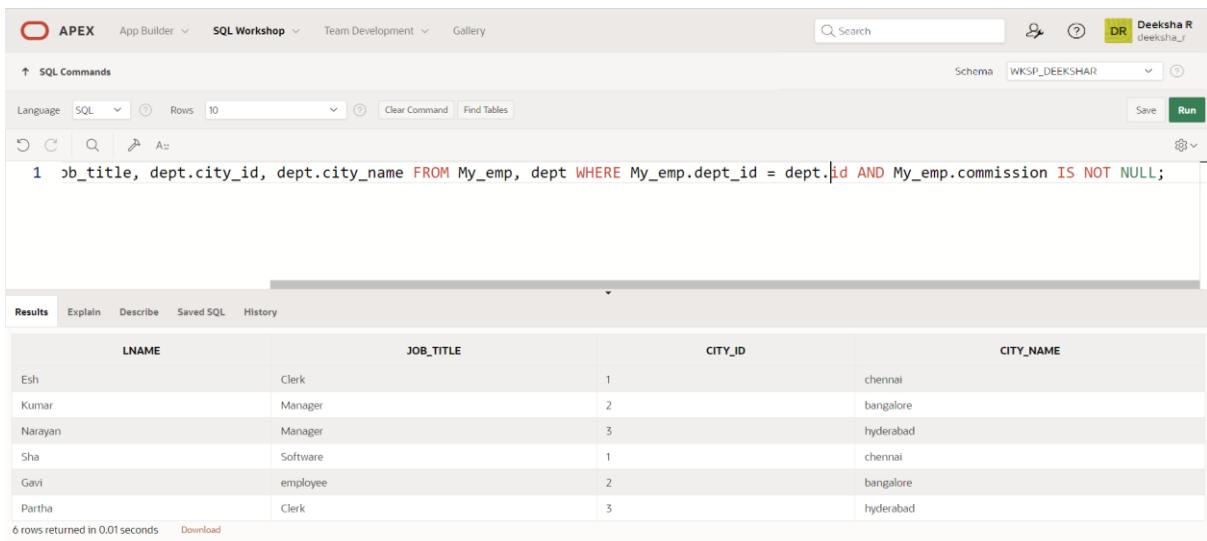
DEPT_ID	CITY_ID
80	1

1 rows returned in 0.01 seconds

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

QUERY: `SELECT My_emp.lname, My_emp.job_title, dept.loc_id, My_emp.city
FROM My_emp, dept WHERE My_emp.dept_id = dept.dept_id AND
My_emp.commission IS NOT NULL;`

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows a user profile for 'Deeksha R' (deeksha_r). The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP_DEEKSHAR'. The query entered is:

```
1  job_title, dept.city_id, dept.city_name FROM My_emp, dept WHERE My_emp.dept_id = dept.id AND My_emp.commission IS NOT NULL;
```

The results tab is selected, displaying the following data:

LNAME	JOB_TITLE	CITY_ID	CITY_NAME
Esh	Clerk	1	chennai
Kumar	Manager	2	bangalore
Narayan	Manager	3	hyderabad
Sha	Software	1	chennai
Gavi	employee	2	bangalore
Partha	Clerk	3	hyderabad

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

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

RESULT:

EXNO:8 AGGREGATING DATA USING GROUP FUNCTIONS DATE:

1. 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.

QUERY: select max(salary) as "MAXIMUM", min(salary) as "MINIMUM", sum(salary) as "SUM", round(avg(salary),2) as "average" from My_emp;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
1 select max(salary) as "MAXIMUM", min(salary) as "MINIMUM", sum(salary) as "SUM", round(avg(salary),2) as "average" from My_emp;
```

The results section displays the following data:

	MAXIMUM	MINIMUM	SUM	average
	2000000	10000	4910000	818355.35

1 rows returned in 0.00 seconds

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

QUERY: select job_title,max(salary) as "MAXIMUM", min(salary) as "MINIMUM",sum(salary) as "SUM",round(avg(salary),2) as "average" from My_emp group by job_title;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
1 select job_title, max(salary) as "MAXIMUM", min(salary) as "MINIMUM", sum(salary) as "SUM", round(avg(salary),2) as "average" from My_emp group by job_title;
```

The results section displays the following data:

JOB_TITLE	MAXIMUM	MINIMUM	SUM	average
employee	10000	10000	10000	10000
Software	600000	600000	600000	600000
Clerk	700000	400000	1100000	550000
Manager	2000000	1200000	3200000	1600000

4 rows returned in 0.01 seconds

3. 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.

QUERY: select count(mg_id) as "MANAGER" from My_emp;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the query is:

```
1 select count(manager_id) as "MANAGER" from My_emp;
```

In the Results pane, the output is:

MANAGER
2

1 rows returned in 0.00 seconds

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

QUERY: select max(salary)-min(salary) as "DIFFERENCE"from My_emp;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the query is:

```
1 select max(salary)-min(salary) as "DIFFERENCE"from My_emp;
```

In the Results pane, the output is:

DIFFERENCE
1990000

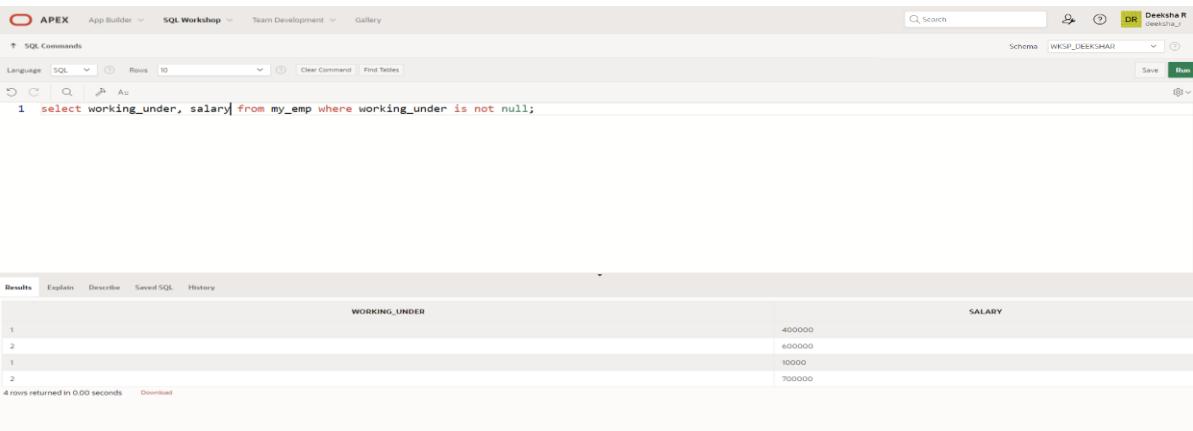
1 rows returned in 0.00 seconds

5.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 \$6000 or less. Sort the output in descending

order of salary

QUERY: select working_under, min(salary) from my_emp where working_under is not null;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The query entered is:

```
1 select working_under, salary from my_emp where working_under is not null;
```

The results table has two columns: WORKING_UNDER and SALARY. The data is:

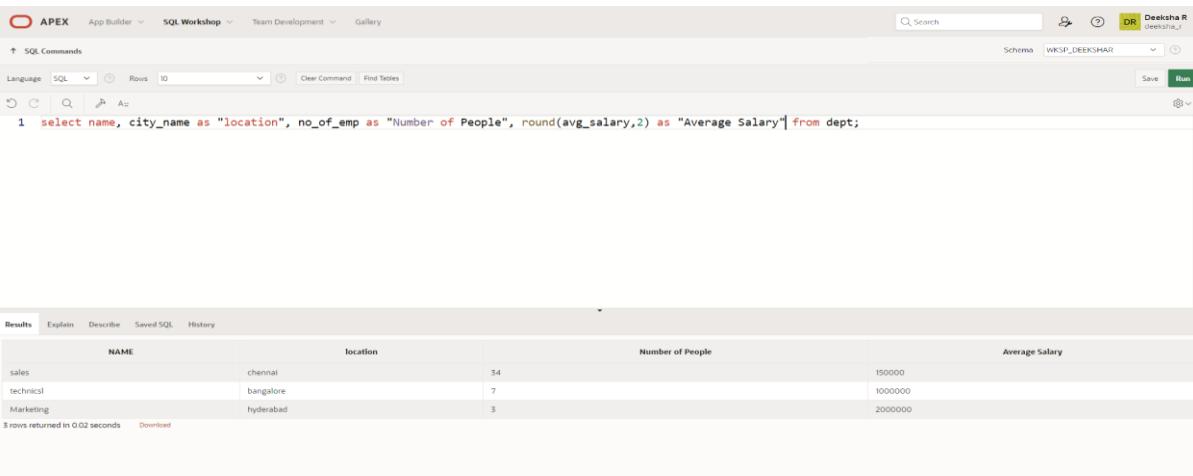
WORKING_UNDER	SALARY
1	400000
2	600000
1	10000
2	700000

4 rows returned in 0.00 seconds

6. Write a query to display each department's name, location, number of employees and the average salary for all the employees in that department. Name the column name- Location, Number of people and salary respectively. Round the average salary to 2 decimal places.

QUERY: select name, city_name as "location", no_of_emp as "Number of People", round(avg_salary,2) as "Average Salary" from dept;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The query entered is:

```
1 select name, city_name as "location", no_of_emp as "Number of People", round(avg_salary,2) as "Average Salary" from dept;
```

The results table has four columns: NAME, location, Number of People, and Average Salary. The data is:

NAME	location	Number of People	Average Salary
sales	chennai	34	150000
technicel	bangalore	7	1000000
Marketing	hyderabad	3	2000000

3 rows returned in 0.02 seconds

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

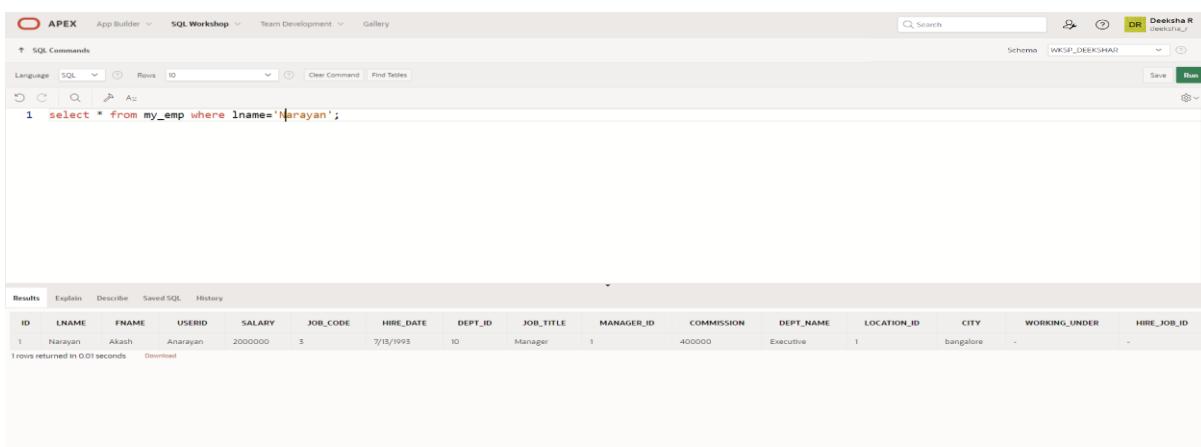
RESULT:

EXNO:9**SUB QUERIES****DATE:**

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)

QUERY: select * from My_emp where lname='zlotkey';

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The SQL command window contains the following code:

```
1 select * from my_emp where lname='Narayan';
```

The results window shows a single row of data:

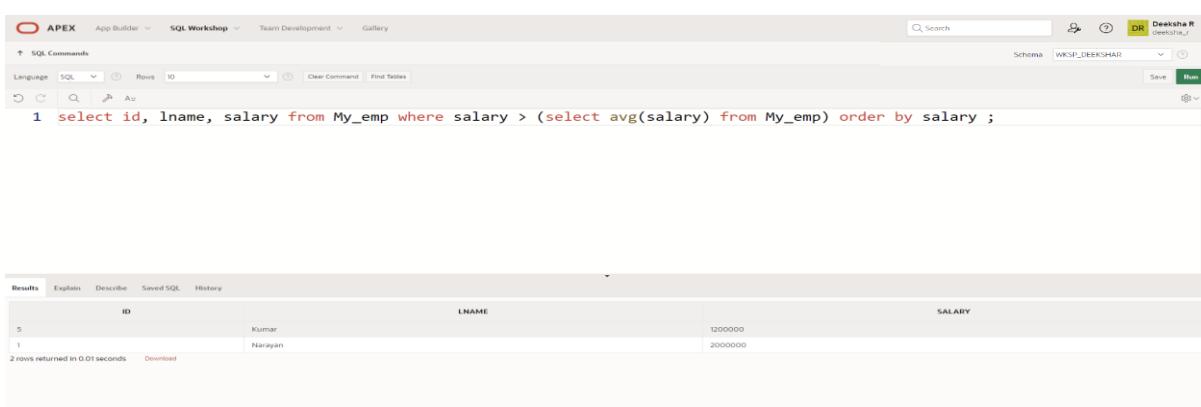
ID	LNAME	FNAME	USERID	SALARY	JOB_CODE	HIRE_DATE	DEPT_ID	JOB_TITLE	MANAGER_ID	COMMISSION	DEPT_NAME	LOCATION_ID	CITY	WORKING_UNDER	HIRE_JOB_ID
1	Narayan	Aakash	Anarayan	2000000	3	7/13/1993	10	Manager	1	400000	Executive	1	bangalore	-	-

1 rows returned in 0.01 seconds

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.

QUERY: select emp_id, lname, salary from My_emp where salary > (select avg(salary) from My_emp) order by salary asc;

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The SQL command window contains the following code:

```
1 select id, lname, salary from My_emp where salary > (select avg(salary) from My_emp) order by salary ;
```

The results window shows two rows of data:

ID	LNAME	SALARY
5	Kumar	1200000
1	Narayan	2000000

2 rows returned in 0.01 seconds

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.

QUERY: select emp_id, lname from My_emp where id_dept in(select id_dept from My_emp where lname like '%u%');

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
1 select id, lname from My_emp where lname in(select lname from My_emp where lname like '%u%');
```

The results table shows one row:

ID	LNAME
5	Kumar

1 rows returned in 0.01 seconds

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.

QUERY: select emp_id, lname from my_emp where loc_id in(select loc_id from My_emp where loc_id=1700);

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
1 select id, lname from my_emp where location_id in(select location_id from My_emp where location_id=1700);
```

The results table shows two rows:

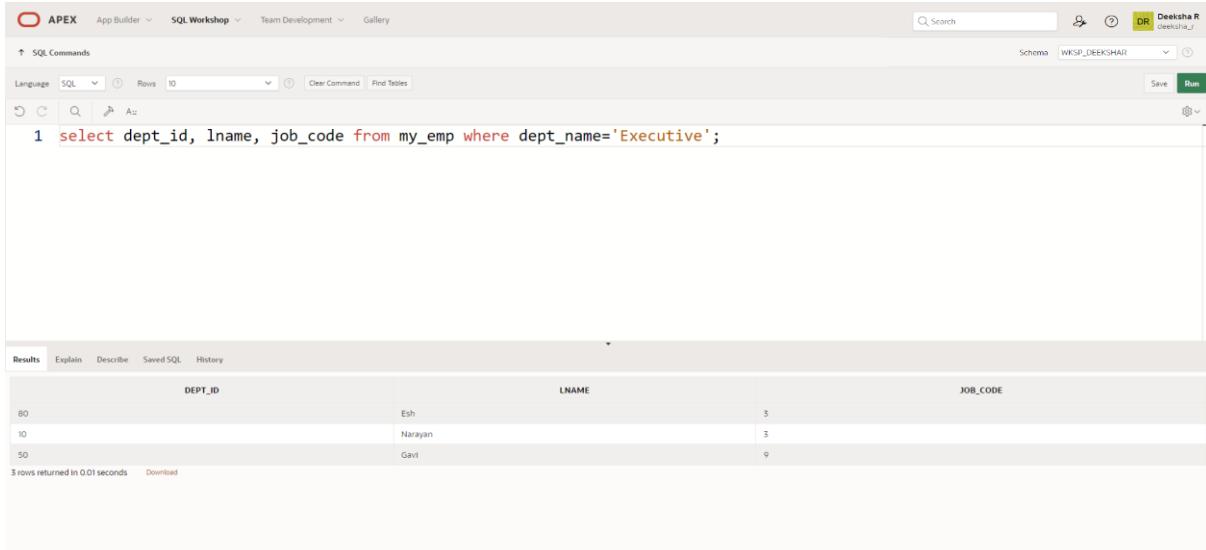
ID	LNAME
1	Narayan
4	Sha

2 rows returned in 0.01 seconds

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

QUERY: select emp_id, lname, id_dept from My_emp where id_dept in (select id_dept from My_emp where job_title = 'ex');

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
1 select dept_id, lname, job_code from my_emp where dept_name='Executive';
```

The results section displays the following data:

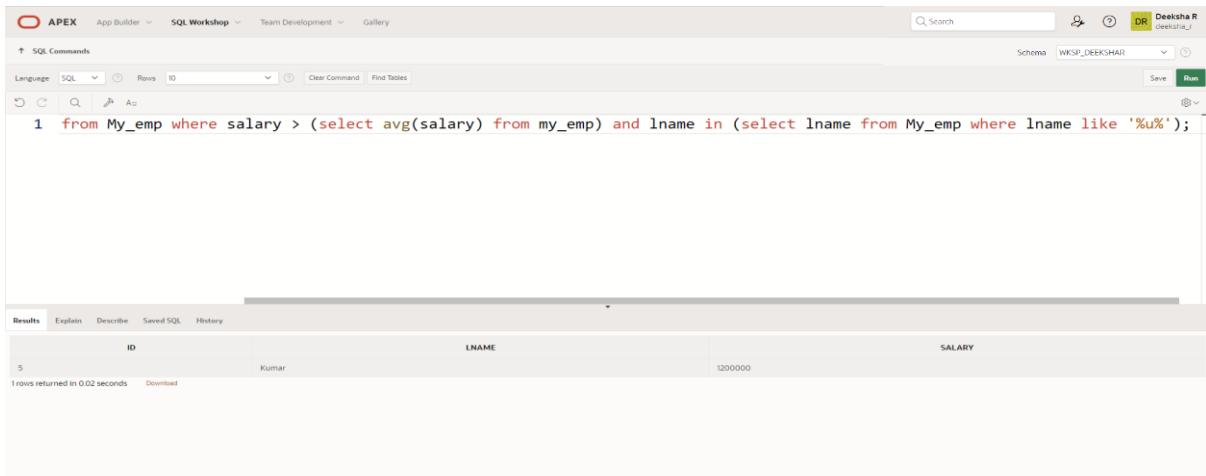
DEPT_ID	LNAME	JOB_CODE
80	Esh	3
10	Narayan	3
50	Gavil	9

3 rows returned in 0.01 seconds

6. 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.

QUERY: select emp_id, lname, salary from My_emp where salary > (select avg(salary) from my_emp) and id_dept in (select id_dept from My_emp where lname like '%u%');

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
1 from My_emp where salary > (select avg(salary) from my_emp) and lname in (select lname from My_emp where lname like '%u%');
```

The results section displays the following data:

ID	LNAME	SALARY
5	Kumar	1200000

1 rows returned in 0.02 seconds

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

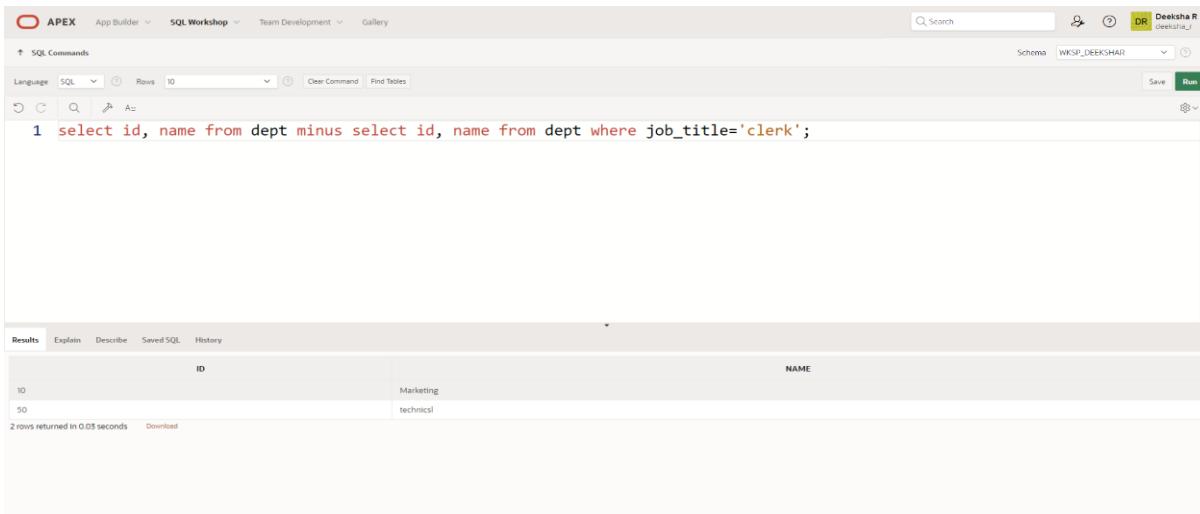
RESULT:

EXNO:10**USING THE SET OPERATORS****DATE:**

- 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.

QUERY: select id, name from dept minus select id, name from dept where job_title='clerk';

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following query is entered:

```
1 select id, name from dept minus select id, name from dept where job_title='clerk';
```

In the Results pane, the output is displayed as a table:

ID	NAME
10	Marketing
50	technicsl

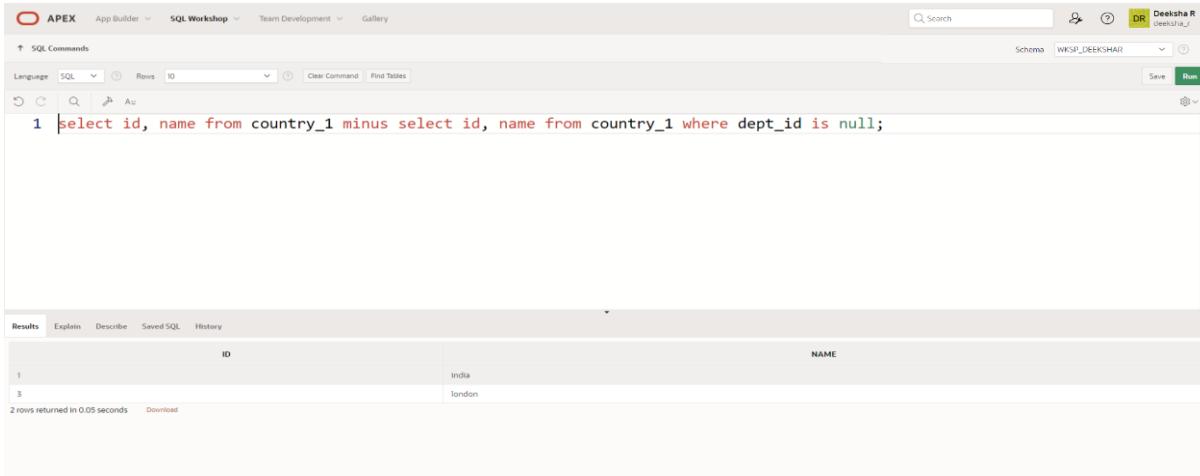
Below the table, it says "2 rows returned in 0.03 seconds".

- 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.

QUERY: select id, name from country_1 minus select id, name from country_1 where dept_id is null;

OUTPUT:

0



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following query is entered:

```
1 select id, name from country_1 minus select id, name from country_1 where dept_id is null;
```

In the Results pane, the output is displayed as a table:

ID	NAME
1	India
3	london

Below the table, it says "2 rows returned in 0.05 seconds".

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

QUERY: select job_title, dept_id from my_emp where dept_id=10 union all select job_title, dept_id from my_emp where dept_id=20 union all select job_title, dept_id from my_emp where dept_id=50;;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
1 select job_title, dept_id from my_emp where dept_id=10 union all select job_title, dept_id from my_emp where dept_id=20 union all select job_title, dept_id from my_emp where dept_id=50;;
```

The results section displays the output:

JOB_TITLE	DEPT_ID
Manager	10
Clerk	10
Clerk	20
Software	20
Manager	50
employee	50

6 rows returned in 0.02 seconds Download

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).

QUERY:

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is:

```
1 select id, job_code from my_emp intersect select id, job_code from my_emp where hire_job_id=job_code;
```

The results section displays the output:

ID	JOB_CODE
2	3
3	5
4	5
6	9

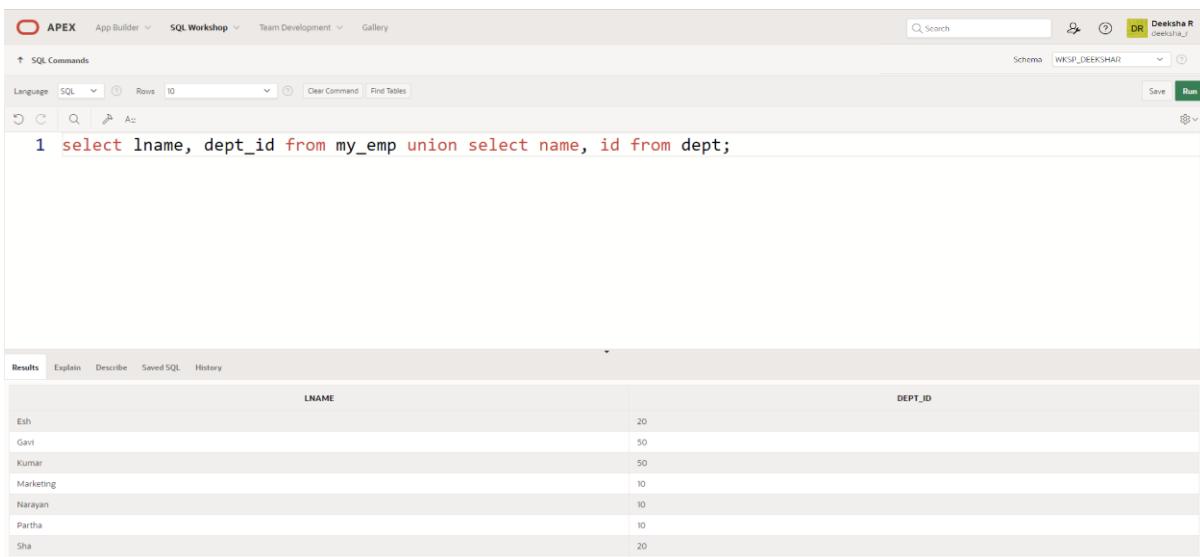
4 rows returned in 0.01 seconds Download

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.

QUERY: select lname, dept_id from my_emp union select name, id from dept;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following SQL code is entered:

```
1 select lname, dept_id from my_emp union select name, id from dept;
```

In the Results pane, the output is displayed in a table format:

LNAME	DEPT_ID
Esh	20
Gavi	50
Kumar	50
Marketing	10
Narayan	10
Partha	10
Sha	20

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

RESULT:

EXNO:11

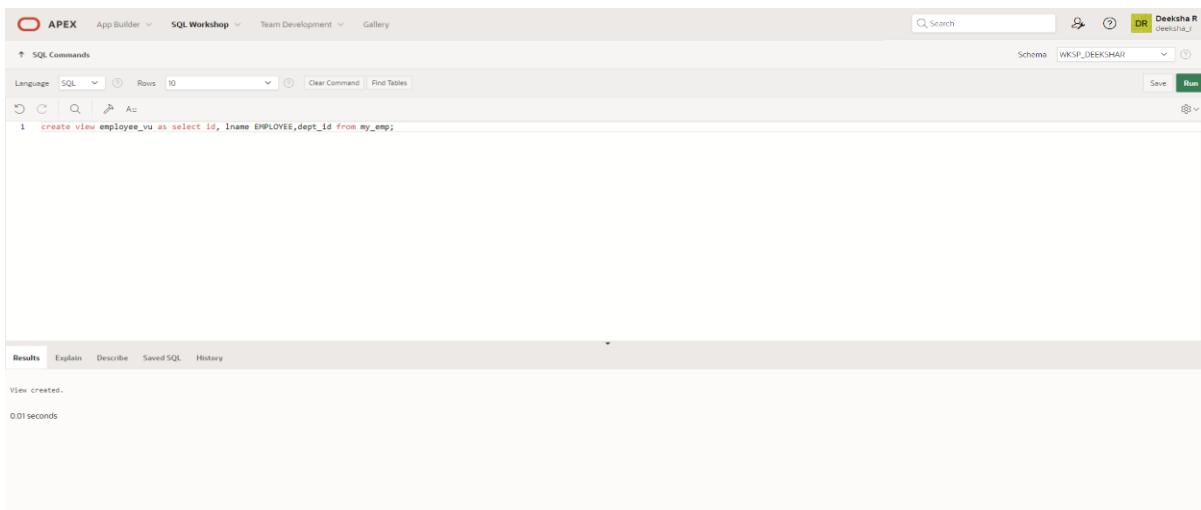
CREATING VIEWS

DATE:

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.

QUERY: create view employee_vu as select id, lname EMPLOYEE,dept_id from my_emp;

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile 'Deeksha R' and a schema dropdown 'WKSP_DEEKSHAR'. The main area has tabs for 'SQL Commands', 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The SQL command entered is:

```
1 create view employee_vu as select id, lname EMPLOYEE,dept_id from my_emp;
```

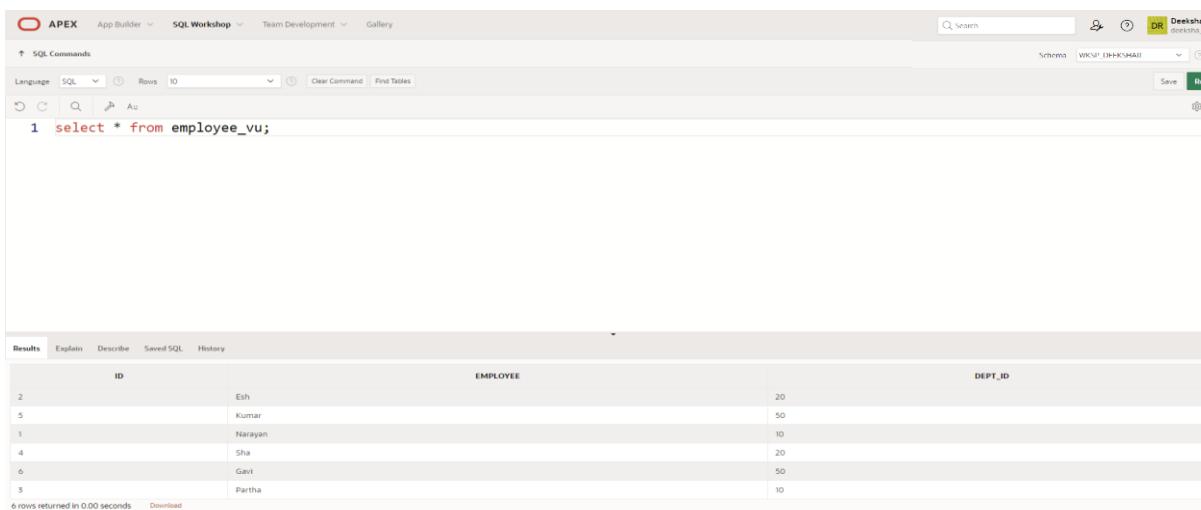
The results section shows the output:

```
View created.  
0.01seconds
```

2.Display the contents of the EMPLOYEES_VU view.

QUERY: select * from employee_vu;

OUTPUT:



A screenshot of the Oracle SQL Workshop interface, similar to the previous one but with a different schema 'WKSP_DHKSHAR'. The SQL command entered is:

```
1 select * from employee_vu;
```

The results section displays the data from the view:

ID	EMPLOYEE	DEPT_ID
2	Esh	20
5	Kumar	50
1	Narayan	10
4	Sha	20
6	Gavt	50
3	Partha	10

6 rows returned in 0.00 seconds

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

QUERY: select employee, dept_id from employee_vu;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following query is entered:

```
1 select employee, dept_id from employee_vu;
```

In the Results pane, the output is displayed as a table:

EMPLOYEE	DEPT_ID
Esh	20
Kumar	50
Narayan	10
Sha	20
Gavi	50
Partha	10

Below the table, it says "6 rows returned in 0.01 seconds" and "Download".

4.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.

QUERY: create view dept50 as select id EMPNO, lname EMPLOYEE, dept_id DEPTNO from my_emp where dept_id=50 with read only;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following query is entered:

```
1 create view dept50 as select id EMPNO, lname EMPLOYEE, dept_id DEPTNO from my_emp where dept_id=50 with read only;
```

In the Results pane, the output is displayed as follows:

View created.
0.03 seconds

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

QUERY: select * from dept50;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The command entered is "select * from dept50;". The results pane displays a table with three columns: EMPNO, EMPLOYEE, and DEPTNO. The data is as follows:

EMPNO	EMPLOYEE	DEPTNO
5	Kumar	50
1	Narayan	50
6	Gavi	50
3	Partha	50

4 rows returned in 0.01 seconds

6. Attempt to reassign Matos to department 80.

QUERY: update dept50 set deptno=80 where employee='Matos';

OUTPUT:

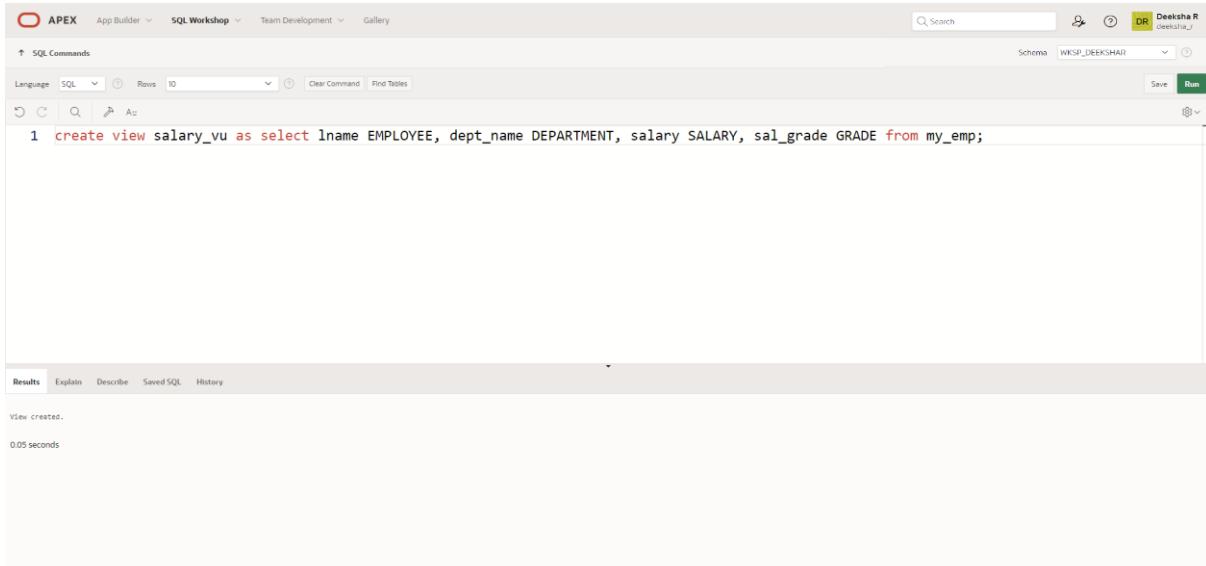
The screenshot shows the Oracle SQL Workshop interface. The command entered is "update dept50 set deptno=80 where employee='Matos';". A yellow box highlights the error message: "Error at line 1/19: ORA-42399: cannot perform a DML operation on a read-only view".

0.01 seconds

7.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.

QUERY: create view salary_vu as select lname EMPLOYEE, dept_name DEPARTMENT, salary SALARY, sal_grade GRADE from my_emp;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, a single line of SQL code is entered: "create view salary_vu as select lname EMPLOYEE, dept_name DEPARTMENT, salary SALARY, sal_grade GRADE from my_emp;". The Results pane below shows the output: "View created." and "0.05 seconds".

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

RESULT:

EXNO:12 PRIMARY KEY, FOREIGN KEY AND CHECK CONSTRAINTS

1. What is the purpose of a

- **PRIMARY KEY** – They provide a unique value that can identify a specific row in a table
- **FOREIGN KEY** - to link data between tables
- **CHECK CONSTRAINT** - to limit the value range that can be placed in a column

2. Using the column information for the animals table below, name constraints where applicable at the table level, otherwise name them at the column level. Define the primary key (animal_id). The license_tag_number must be unique. The admit_date and vaccination_date columns cannot contain null values.

animal_id NUMBER(6) - PRIMARY KEY

name VARCHAR2(25)

license_tag_number NUMBER(10)- UNIQUE

admit_date DATE- NOT NULL

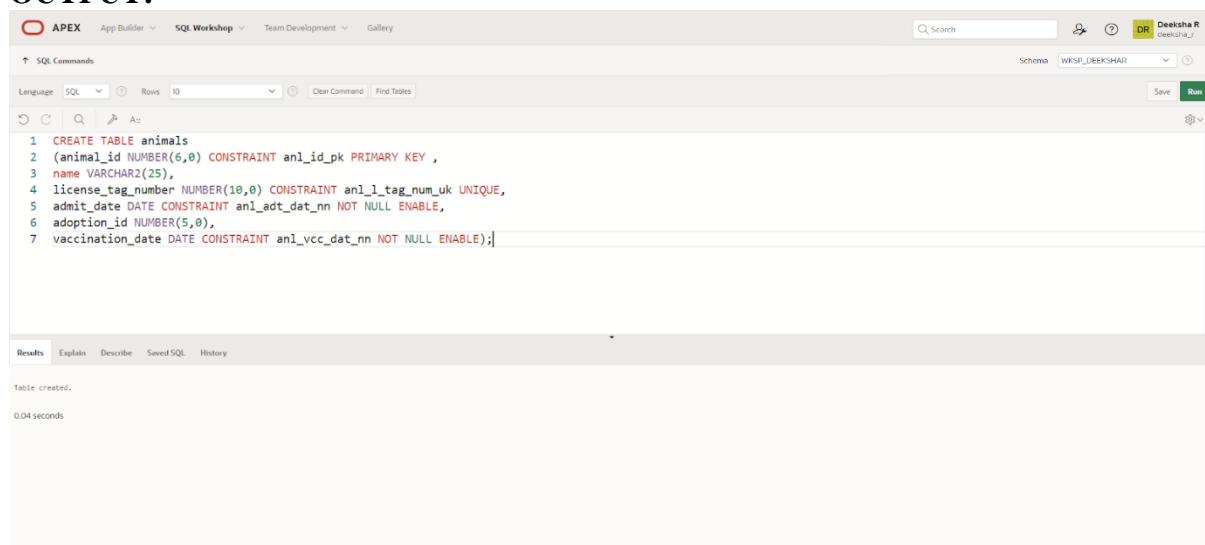
adoption_id NUMBER(5),

vaccination_date DATE- NOT NULL

3. Create the animals table. Write the syntax you will use to create the table.

QUERY: CREATE TABLE animals (animal_id NUMBER(6,0) CONSTRAINT anl_id_pk PRIMARY KEY , name VARCHAR2(25), license_tag_number NUMBER(10,0) CONSTRAINT anl_l_tag_num_uk UNIQUE, admit_date DATE CONSTRAINT anl_adt_dat_nn NOT NULL ENABLE,adoption_id NUMBER(5,0), vaccination_date DATE CONSTRAINT anl_vcc_dat_nn NOT NULL ENABLE);

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery' are visible. On the right, there's a user icon for 'Deeksha R' and a 'Run' button. The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the code for creating the 'animals' table is pasted:

```
1 CREATE TABLE animals
2 (animal_id NUMBER(6,0) CONSTRAINT anl_id_pk PRIMARY KEY ,
3 name VARCHAR2(25),
4 license_tag_number NUMBER(10,0) CONSTRAINT anl_l_tag_num_uk UNIQUE,
5 admit_date DATE CONSTRAINT anl_adt_dat_nn NOT NULL ENABLE,
6 adoption_id NUMBER(5,0),
7 vaccination_date DATE CONSTRAINT anl_vcc_dat_nn NOT NULL ENABLE);
```

In the 'Results' tab, the output shows:

```
Table created.
0.04 seconds
```

4. Enter one row into the table. Execute a SELECT * statement to verify your input.

QUERY:insert into
animals(animal_id,name,license_tag_number,admit_date,adoption_id,
vaccination_date) values (101,'spot',35540,'10/10/2004',205,'12/10/2004');

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area contains the SQL command:

```
1 insert into animals(animal_id, name, license_tag_number, admit_date, adoption_id, vaccination_date) values (101, 'spot', 35540, '10/10/2004', 205, '12/10/2004');
```

Below the command, the results tab is selected, showing the output:

```
1 row(s) inserted.  
0.03 seconds
```

5. What are the restrictions on defining a CHECK constraint?

Ans: If you define a CHECK constraint on a column it will allow only certain values for this column. If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

RESULT:

Creating Views

EX.NO.13

DATE:

1. What are three uses for a view from a DBA's perspective?

- **Restrict access and display selective columns**
- **Reduce complexity of queries from other internal systems. So, providing a way to view same data in a different manner.**
- **Let the app code rely on views and allow the internal implementation of tables to be modified later.**

2. Create a simple view called view_d_songs that contains the ID, title and artist from the DJs on Demand table for each "New Age" type code. In the subquery, use the alias "Song Title" for the title column.

CREATE VIEW view_d_songs AS

```
SELECT d_songs.id, d_songs.title "Song Title", d_songs.artist
from d_songs INNER JOIN d_types ON d_songs.type_code = d_types.code
where d_types.description = 'New Age';
```



The screenshot shows the Oracle SQL Workshop interface. The SQL Commands tab is active, displaying the SQL code for creating the view. The code is as follows:

```
1 CREATE VIEW view_d_songs AS
2 SELECT d_songs.id, d_songs.title "Song Title", d_songs.artist
3 from d_songs INNER JOIN d_types ON d_songs.type_code = d_types.code
4 where d_types.description = 'New Age';
```

The results pane shows the message "View created." and a execution time of "0.03 seconds".

3. **SELECT * FROM view_d_songs.** What was returned?



The screenshot shows the Oracle SQL Workshop interface. The SQL Commands tab is active, displaying the query "SELECT * FROM view_d_songs;". The results pane shows the output of the query:

ID	Song Title	ARTIST
54	YYY	SHREYA GOSHAL

The results pane also indicates "1 rows returned in 0.01 seconds".

4. REPLACE view_d_songs. Add type_code to the column list. Use aliases for all columns. Or use alias after the CREATE statement as shown.

```
CREATE OR REPLACE VIEW view_d_songs AS
SELECT d_songs.id, d_songs.title "Song Title", d_songs.artist, d_songs.type_code
from d_songs INNER JOIN d_types ON d_songs.type_code = d_types.code
where d_types.description = 'New Age';
```



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a user profile for 'Deeksha R' and a schema dropdown set to 'WKSP_DEEKSHAR'. The main area is titled 'SQL Commands' with a 'Run' button. Below it, the SQL code for creating the view is pasted:

```
1 CREATE OR REPLACE VIEW view_d_songs AS
2 SELECT d_songs.id, d_songs.title "Song Title", d_songs.artist, d_songs.type_code
3 from d_songs INNER JOIN d_types ON d_songs.type_code = d_types.code
4 where d_types.description = 'New Age';
```

Under the 'Results' tab, the message 'View created.' is displayed, along with a timestamp '0.01 seconds'.

5. Jason Tsang, the disk jockey for DJs on Demand, needs a list of the past events and those planned for the coming months so he can make arrangements for each event's equipment setup. As the company manager, you do not want him to have access to the price that clients paid for their events. Create a view for Jason to use that displays the name of the event, the event date, and the theme description. Use aliases for each column name.

```
CREATE OR REPLACE VIEW view_d_events_pkgs AS
SELECT evt.name "Name of Event", TO_CHAR(evt.event_date, 'dd-Month-
yyyy') "Event date", thm.description "Theme description"
FROM d_events evt INNER JOIN d_themes thm ON evt.theme_code = thm.code
WHERE evt.event_date <= ADD_MONTHS(SYSDATE,1);
```



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a user profile for 'Deeksha R' and a schema dropdown set to 'WKSP_DEEKSHAR'. The main area is titled 'SQL Commands' with a 'Run' button. Below it, the SQL code for creating the view is pasted:

```
1 CREATE OR REPLACE VIEW view_d_events_pkgs AS
2 SELECT evt.name "Name of Event", TO_CHAR(evt.event_date, 'dd-Month-yyyy') "Event date", thm.description "Theme description"
3 FROM d_events evt INNER JOIN d_themes thm ON evt.theme_code = thm.code
4 WHERE evt.event_date <= ADD_MONTHS(SYSDATE,1);
5
```

Under the 'Results' tab, the message 'View created.' is displayed, along with a timestamp '0.02 seconds'.

6. It is company policy that only upper-level management be allowed access to individual employee salaries. The department managers, however, need to know the minimum, maximum, and average salaries, grouped by department. Use the Oracle database to prepare a view that displays the needed information for department managers.

```
CREATE OR REPLACE VIEW view_min_max_avg_dpt_salary ("Department Id", "Department Name", "Max Salary", "Min Salary", "Average Salary") AS
SELECT dpt.department_id, dpt.department_name, MAX(NVL(emp.salary,0)),
MIN(NVL(emp.salary,0)), ROUND( AVG(NVL(emp.salary,0)),2)
FROM departments dpt LEFT OUTER JOIN employees emp ON dpt.department_id =
emp.department_id
GROUP BY (dpt.department_id, dpt.department_name);
```



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is active. The schema dropdown is set to 'WKSP_DEEKSHAR'. The main area contains the SQL code for creating the view. The 'Results' tab is selected, showing the message 'View created.' and a execution time of '0.02 seconds'.



The screenshot shows the Oracle SQL Workshop interface again. The 'Results' tab is selected under a new SQL command. The query is 'SELECT * FROM view_min_max_avg_dpt_salary;'. The results table displays the following data:

Department Id	Department Name	Max Salary	Min Salary	Average Salary
59	BIO	12222	12222	12222
2222	cse	3200	3200	3200
1	Executive	100000	100000	100000

Below the table, it says '3 rows returned in 0.02 seconds'.

DML Operations and Views

Use the DESCRIBE statement to verify that you have tables named copy_d_songs, copy_d_events, copy_d_cds, and copy_d_clients in your schema. If you don't, write a query to create a copy of each.

1. Query the data dictionary USER_UPDATABLE_COLUMNS to make sure the columns in the base tables will allow UPDATE, INSERT, or DELETE. All table names in the data dictionary are stored in uppercase.

```
SELECT owner, table_name, column_name, updatable,insertable, deletable
FROM user_updatable_columns WHERE LOWER(table_name) = 'copy_d_songs';
```

```
SELECT owner, table_name, column_name, updatable,insertable, deletable
FROM user_updatable_columns WHERE LOWER(table_name) = 'copy_d_events';
```

```
SELECT owner, table_name, column_name, updatable,insertable, deletable
FROM user_updatable_columns WHERE LOWER(table_name) = 'copy_d_cds';
```

The screenshot shows two separate sessions in the Oracle SQL Workshop interface. Both sessions are connected to the schema 'WKSP_DEEKSHAR'.

Session 1 (Top): The user has run the following SQL command:

```
1 SELECT owner, table_name, column_name, updatable,insertable, deletable
2 FROM user_updatable_columns WHERE LOWER(table_name) = 'copy_d_songs';
3
```

The results grid displays the following data:

OWNER	TABLE_NAME	COLUMN_NAME	UPDATABLE	INSERTABLE	DELETABLE
WKSP_MBHUV	COPY_D_SONGS	TITLE	YES	YES	YES
WKSP_MBHUV	COPY_D_SONGS	DURATION	YES	YES	YES
WKSP_MBHUV	COPY_D_SONGS	TYPE_CODE	YES	YES	YES

3 rows returned in 0.04 seconds

Session 2 (Bottom): The user has run the following SQL command:

```
1 SELECT owner, table_name, column_name, updatable,insertable, deletable
2 FROM user_updatable_columns WHERE LOWER(table_name) = 'copy_d_cds';
3
```

The results grid displays the following data:

OWNER	TABLE_NAME	COLUMN_NAME	UPDATABLE	INSERTABLE	DELETABLE
WKSP_MBHUV	COPY_D_CDS	CD_NUMBER	YES	YES	YES
WKSP_MBHUV	COPY_D_CDS	PRODUCER	YES	YES	YES
WKSP_MBHUV	COPY_D_CDS	TITLE	YES	YES	YES
WKSP_MBHUV	COPY_D_CDS	YEAR	YES	YES	YES

4 rows returned in 0.05 seconds

2. Use the CREATE or REPLACE option to create a view of *all* the columns in the copy_d_songs table called view_copy_d_songs.

```
CREATE OR REPLACE VIEW view_copy_d_songs AS
SELECT *
FROM copy_d_songs;
```

```
SELECT * FROM view_copy_d_songs;
```

The screenshot shows two consecutive screenshots of the Oracle SQL Workshop interface. In the first screenshot, a SQL command is being run to create a view named 'view_copy_d_songs' that selects all columns from the 'copy_d_songs' table. The command is:

```
CREATE OR REPLACE VIEW view_copy_d_songs AS
SELECT *
FROM copy_d_songs;
```

The second screenshot shows the results of running a select query against the newly created view. The command is:

```
SELECT * FROM view_copy_d_songs;
```

The results pane indicates 'no data found'.

3. Use view_copy_d_songs to INSERT the following data into the underlying copy_d_songs table. Execute a SELECT * from copy_d_songs to verify your DML command. See the graphic.

ID	TITLE	DURATION	ARTIST	TYPE_CODE
88	Mello Jello	2	The What	4

```
INSERT INTO view_copy_d_songs(id,title,duration,artist,type_code)
VALUES(88,'Mello Jello','2 min','The What',4);
```

The screenshot shows the Oracle SQL Workshop interface with a SQL command being run to insert data into the 'view_copy_d_songs' view. The command is:

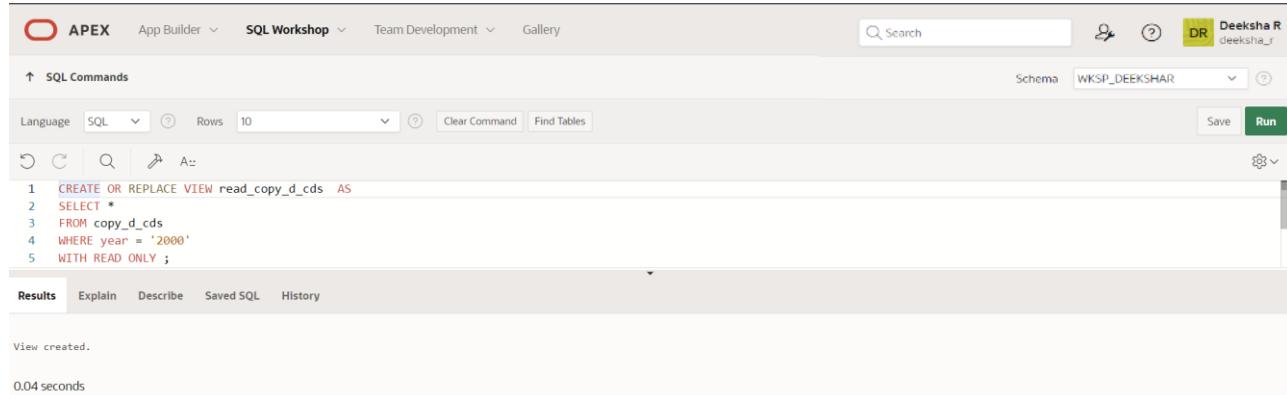
```
INSERT INTO view_copy_d_songs(id,title,duration,artist,type_code)
VALUES(88,'Mello Jello','2 min','The What',4);
```

The results pane shows the message '1 row(s) inserted.' and a execution time of '0.03 seconds'.

4. Create a view based on the DJs on Demand COPY_D_CDS table. Name the view read_copy_d_cds. Select all columns to be included in the view. Add a WHERE clause to restrict the year to 2000. Add the WITH READ ONLY option.

```
CREATE OR REPLACE VIEW read_copy_d_cds AS
SELECT *
FROM copy_d_cds
WHERE year = '2000'
WITH READ ONLY;

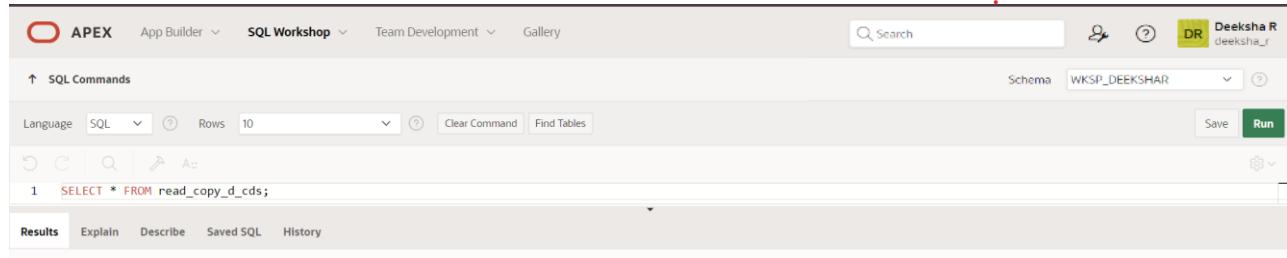
SELECT * FROM read_copy_d_cds;
```



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'SQL Workshop' is selected. The schema dropdown shows 'WKSP_DEEKSHAR'. The main area displays the SQL command for creating the view:

```
1 CREATE OR REPLACE VIEW read_copy_d_cds AS
2 SELECT *
3 FROM copy_d_cds
4 WHERE year = '2000'
5 WITH READ ONLY;
```

Below the command, the 'Results' tab is selected, showing the message 'View created.' and a execution time of '0.04 seconds'.



The screenshot shows the Oracle SQL Workshop interface with the same setup as the previous screenshot. The 'Results' tab is selected, displaying the message 'no data found'.

```
1 SELECT * FROM read_copy_d_cds;
```

5. Using the read_copy_d_cds view, execute a DELETE FROM read_copy_d_cds WHERE cd_number = 90;

ORA-42399: cannot perform a DML operation on a read-only view

6. Use REPLACE to modify read_copy_d_cds. Replace the READ ONLY option with WITH CHECK OPTION CONSTRAINT ck_read_copy_d_cds. Execute a SELECT * statement to verify that the view exists.

```
CREATE OR REPLACE VIEW read_copy_d_cds AS
SELECT *
FROM copy_d_cds
WHERE year = '2000'
WITH CHECK OPTION CONSTRAINT ck_read_copy_d_cds;
```

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'SQL Workshop' is selected. The schema dropdown is set to 'WKSP_DEEKSHAR'. The main area displays the SQL command for creating the view:

```
1 CREATE OR REPLACE VIEW read_copy_d_cds AS
2 SELECT *
3 FROM copy_d_cds
4 WHERE year = '2000'
5 WITH CHECK OPTION CONSTRAINT ck_read_copy_d_cds;
```

Below the code, the 'Results' tab is active, showing the message 'View created.' and a execution time of '0.04 seconds'.

6. Use the read_copy_d_cds view to delete any CD of year 2000 from the underlying copy_d_cds.

```
DELETE FROM read_copy_d_cds
WHERE year = '2000';
```

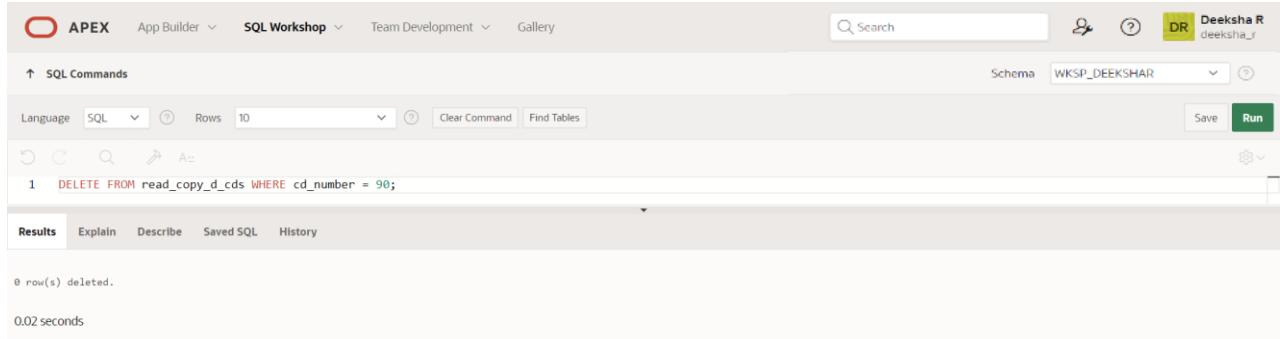
The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'SQL Workshop' is selected. The schema dropdown is set to 'WKSP_DEEKSHAR'. The main area displays the SQL command for deleting rows from the view:

```
1 DELETE FROM read_copy_d_cds WHERE year = '2000';
```

Below the code, the 'Results' tab is active, showing the message '0 row(s) deleted.' and a execution time of '0.02 seconds'.

7. Use the read_copy_d_cds view to delete cd_number 90 from the underlying copy_d_cds table.

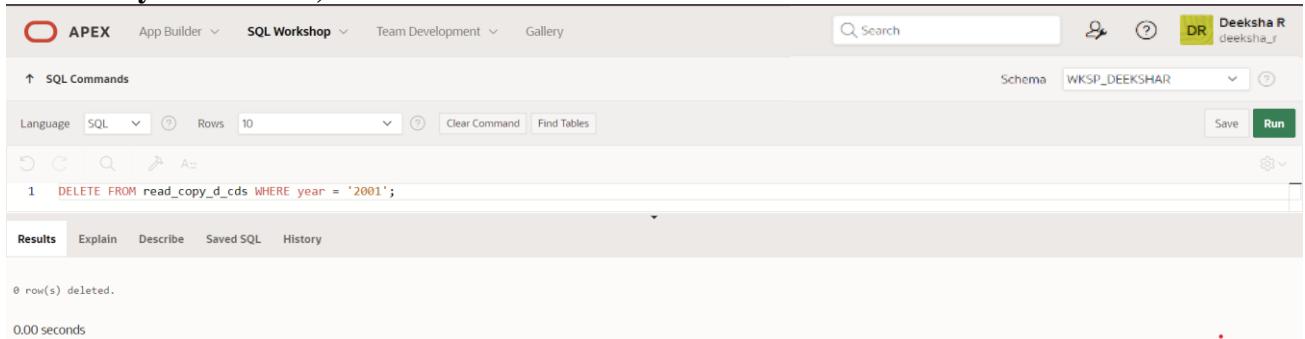
**DELETE FROM read_copy_d_cds
WHERE cd_number = 90;**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile 'Deeksha R deeksha_r' are on the right. The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the query '1 DELETE FROM read_copy_d_cds WHERE cd_number = 90;' is entered. The results tab shows '0 row(s) deleted.' and a execution time of '0.02 seconds'.

7. Use the read_copy_d_cds view to delete year 2001 records.

**DELETE FROM read_copy_d_cds
WHERE year = '2001';**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile 'Deeksha R deeksha_r' are on the right. The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the query '1 DELETE FROM read_copy_d_cds WHERE year = '2001'' is entered. The results tab shows '0 row(s) deleted.' and a execution time of '0.00 seconds'.

8. Execute a SELECT * statement for the base table copy_d_cds. What rows were deleted?

Only the one in problem 7 above, not the one in 8 and 9

11. What are the restrictions on modifying data through a view?

DELETE,INSERT,MODIFY restricted if it contains:

Group functions

GROUP BY CLAUSE
DISTINCT
pseudocolumn ROWNUM Keyword

12. What is Moore's Law? Do you consider that it will continue to apply indefinitely? Support your opinion with research from the internet.

It roughly predicted that computing power nearly doubles every year. But Moore also said in 2005 that as per nature of exponential functions, this trend may not continue forever.

13. What is the "singularity" in terms of computing?

Singularity is the hypothesis that the invention of artificial superintelligence will abruptly trigger runaway technological growth, resulting in unfathomable changes to human civilization

Managing Views

1. Create a view from the copy_d_songs table called view_copy_d_songs that includes only the title and artist. Execute a SELECT * statement to verify that the view exists.

```
CREATE OR REPLACE VIEW view_copy_d_songs AS
SELECT title, artist
FROM copy_d_songs;
```

```
SELECT * FROM view_copy_d_songs;
```

The screenshot shows two consecutive executions of SQL commands in the Oracle SQL Workshop interface. In the first execution, a view named 'view_copy_d_songs' is created, mapping to the 'copy_d_songs' table with columns 'title' and 'artist'. In the second execution, a SELECT * statement is run against this newly created view, returning a single row with 'Mello Jello' in the title and 'The What' in the artist column.

```
1 CREATE OR REPLACE VIEW view_copy_d_songs AS SELECT title, artist FROM copy_d_songs;
View created.
0.03 seconds

1 SELECT * FROM view_copy_d_songs;
TITLE          ARTIST
Mello Jello    The What
1 rows returned in 0.01 seconds
```

2. Issue a DROP view_copy_d_songs. Execute a SELECT * statement to verify that the view has been deleted.

```
DROP VIEW view_copy_d_songs;
SELECT * FROM view_copy_d_songs;
```

ORA-00942: table or view does not exist

The screenshot shows the execution of a DROP command followed by a SELECT * command. The first part of the screenshot shows the execution of 'DROP VIEW view_copy_d_songs;'. The second part shows the execution of 'SELECT * FROM view_copy_d_songs;', which results in an error message indicating that the table or view does not exist.

```
1 DROP VIEW view_copy_d_songs;
2
View dropped.

0.04 seconds

1 SELECT * FROM view_copy_d_songs;
```

3. Create a query that selects the last name and salary from the Oracle database. Rank the salaries from highest to lowest for the top three employees.

```
SELECT * FROM
(SELECT last_name, salary FROM employees ORDER BY salary DESC)
WHERE ROWNUM <= 3;
```

LAST_NAME	SALARY
bhuvana	100000
Matos	50000
Davies	34000

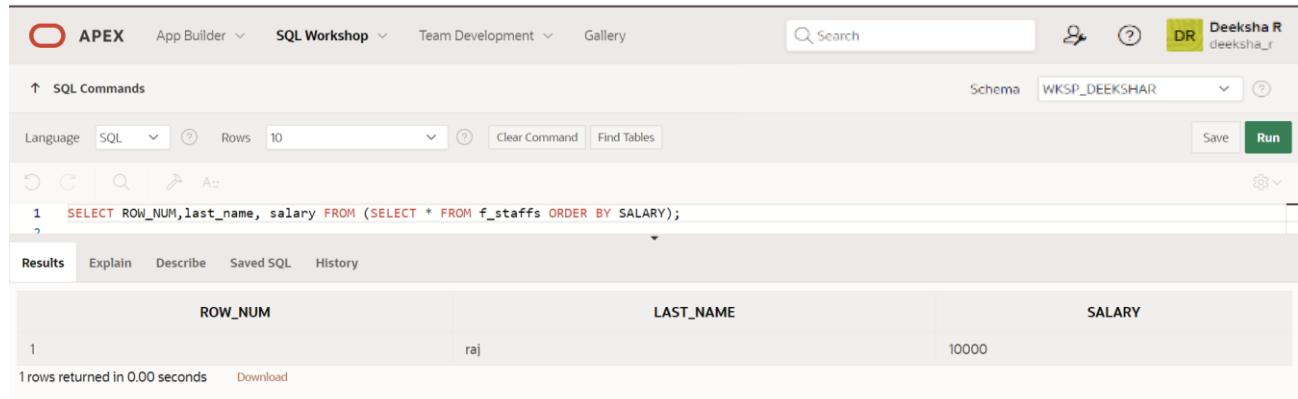
4. Construct an inline view from the Oracle database that lists the last name, salary, department ID, and maximum salary for each department. Hint: One query will need to calculate maximum salary by department ID.

```
SELECT empm.last_name, empm.salary, dptmx.department_id
FROM
(SELECT dpt.department_id, MAX(NVL(emp.salary,0)) max_dpt_sal
FROM departments dpt LEFT OUTER JOIN employees emp ON dpt.department_id =
emp.department_id
GROUP BY dpt.department_id) dptmx LEFT OUTER JOIN employees empm ON
dptmx.department_id = empm.department_id
WHERE NVL(empm.salary,0) = dptmx.max_dpt_sal;
```

LAST_NAME	SALARY	DEPT_ID
brindha	12222	59
bhuvana	100000	1
Zlotkey	3200	2222

5. Create a query that will return the staff members of Global Fast Foods ranked by salary from lowest to highest.

```
SELECT ROW_NUM, last_name, salary
FROM
(SELECT * FROM f_staffs ORDER BY SALARY);
```



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. A search bar and user profile 'Deeksha R' are also present. The main area is titled 'SQL Commands' with a language dropdown set to 'SQL'. The command entered is:

```
1  SELECT ROW_NUM, last_name, salary FROM (SELECT * FROM f_staffs ORDER BY SALARY);
2
```

The results tab is selected, displaying the output:

ROW_NUM	LAST_NAME	SALARY
1	raj	10000

Below the table, it says '1 rows returned in 0.00 seconds' and has a 'Download' link.

Indexes and Synonyms

1. What is an index and what is it used for?

Definition: These are schema objects which make retrieval of rows from table faster.

Purpose: An index provides direct and fast access to row in table. They provide indexed path to locate data quickly, so hereby reduce necessity of heavy disk input/output operations.

2. What is a ROWID, and how is it used?

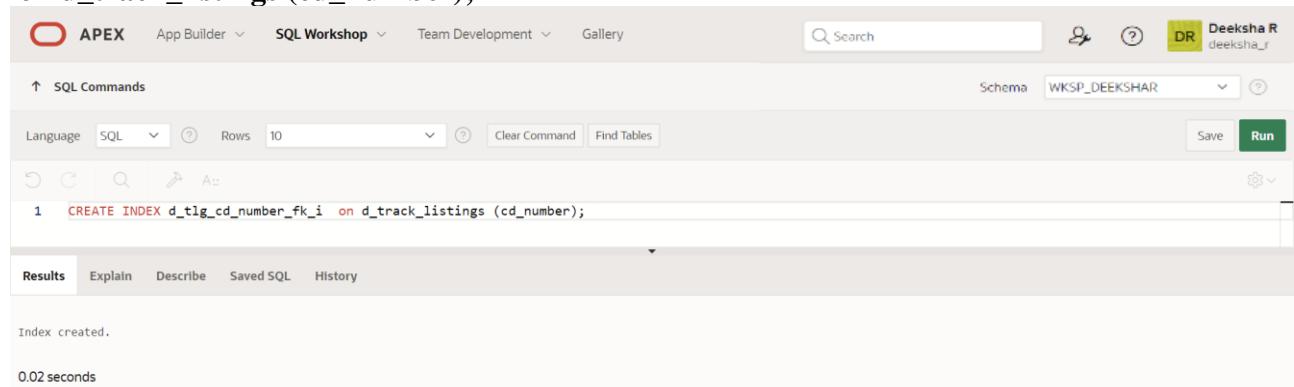
Indexes use ROWID's (base 64 string representation of the row address containing block identifier, row location in the block and the database file identifier) which is the fastest way to access any particular row.

3. When will an index be created automatically?

Primary key/unique key use already existing unique index but if index is not present already, it is created while applying unique/primary key constraint.

4. Create a nonunique index (foreign key) for the DJs on Demand column (cd_number) in the D_TRACK_LISTINGS table. Use the Oracle Application Express SQL Workshop Data Browser to confirm that the index was created.

```
CREATE INDEX d_tlg_cd_number_fk_i  
on d_track_listings (cd_number);
```



The screenshot shows the Oracle Application Express SQL Workshop Data Browser interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there is a user icon labeled 'Deeksha R' and 'WKSP_DEEKSHAR'. The main area is titled 'SQL Commands' with a search bar and a 'Run' button. Below the title, there are filters for Language (SQL), Rows (10), and buttons for Clear Command and Find Tables. The SQL command entered is: 'CREATE INDEX d_tlg_cd_number_fk_i on d_track_listings (cd_number);'. The results section at the bottom shows the message 'Index created.' and a execution time of '0.02 seconds'.

5. Use the join statement to display the indexes and uniqueness that exist in the data dictionary for the DJs on Demand D_SONGS table.

```
SELECT ucm.index_name, ucm.column_name, ucm.column_position, uix.uniqueness  
FROM user_indexes uix INNER JOIN user_ind_columns ucm ON uix.index_name =  
ucm.index_name  
WHERE ucm.table_name = 'D_SONGS';
```

A screenshot of the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile 'DR Deeksha R deeksha_r' are on the right. The main area shows a SQL command window with the following code:

```

1 SELECT ucm.index_name, ucm.column_name, ucm.column_position, uix.uniqueness
2 FROM user_indexes uix INNER JOIN user_ind_columns ucm ON uix.index_name = ucm.index_name
3 WHERE ucm.table_name = 'D_SONGS';

```

The results tab shows 'no data found'.

6. Use a SELECT statement to display the index_name, table_name, and uniqueness from the data dictionary USER_INDEXES for the DJs on Demand D_EVENTS table.

SELECT index_name, table_name,uniqueness FROM user_indexes where table_name = 'D_EVENTS';

A screenshot of the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile 'DR Deeksha R deeksha_r' are on the right. The main area shows a SQL command window with the following code:

```

1 SELECT index_name, table_name,uniqueness FROM user_indexes where table_name = 'D_EVENTS';

```

The results tab shows 'no data found'.

7. Write a query to create a synonym called dj_tracks for the DJs on Demand d_track_listings table.

CREATE SYNONYM dj_tracks FOR d_track_listings;

A screenshot of the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile 'DR Deeksha R deeksha_r' are on the right. The main area shows a SQL command window with the following code:

```

1 CREATE SYNONYM dj_tracks FOR d_track_listings;

```

The results tab shows 'Synonym created.' and '0.01 seconds'.

8. Create a function-based index for the last_name column in DJs on Demand D_PARTNERS table that makes it possible not to have to capitalize the table name for searches. Write a SELECT statement that would use this index.

**CREATE INDEX d_ptr_last_name_idx
ON d_partners(LOWER(last_name));**

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'SQL Workshop' is selected. The schema is set to 'WKSP_DEEKSHAR'. The main area displays a SQL command to create an index:

```
1 CREATE INDEX d_ptr_last_name_idx ON D_PARTNERS(LOWER(last_name));
```

The results pane shows the output of the command:

Index created.
0.02 seconds

9. Create a synonym for the D_TRACK_LISTINGS table. Confirm that it has been created by querying the data dictionary.

CREATE SYNONYM dj_tracks2 FOR d_track_listings;

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'SQL Workshop' is selected. The schema is set to 'WKSP_DEEKSHAR'. The main area displays a SQL command to create a synonym:

```
1 CREATE SYNONYM dj_tracks2 FOR d_track_listings;
```

The results pane shows the output of the command:

Synonym created.
0.01 seconds

SELECT * FROM user_synonyms WHERE table_NAME = UPPER('d_track_listings');

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'SQL Workshop' is selected. The schema is set to 'WKSP_DEEKSHAR'. The main area displays a SQL command to query the data dictionary:

```
1 SELECT * FROM user_synonyms WHERE table_NAME = UPPER('d_track_listings');
```

The results pane shows the output of the command:

SYNONYM_NAME	TABLE_OWNER	TABLE_NAME	DB_LINK	ORIGIN_CON_ID
DJ_TRACKS	WKSP_MBHVU	D_TRACK_LISTINGS	-	0
DJ_TRACKS2	WKSP_MBHVU	D_TRACK_LISTINGS	-	0

2 rows returned in 0.03 seconds [Download](#)

10. Drop the synonym that you created in question

DROP SYNONYM dj_tracks2;

The screenshot shows the Oracle SQL Workshop interface. At the top, there are navigation links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there is a search bar, a user profile for 'Deeksha R' (deeksha_1), and a schema dropdown set to 'WKSP_DEEKSHAR'. Below the header, the 'SQL Commands' tab is selected. The main area contains a SQL editor with the following content:

```
1 DROP SYNONYM dj_tracks2;
```

Below the editor, there are tabs for Results, Explain, Describe, Saved SQL, and History. The 'Results' tab is active. The output section displays the message "Synonym dropped." and a time stamp "0.02 seconds".

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

RESULT:

OTHER DATABASE OBJECTS

EX.NO:14

DATE:

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

QUERY:

CREATE SEQUENCE dept_id_seq START WITH 200 INCREMENT BY 10 MAXVALUE 1000;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'SQL Workshop' is selected. The main area displays the SQL command: 'CREATE SEQUENCE dept_id_seq START WITH 200 INCREMENT BY 10 MAXVALUE 1000;'. Below the command, the results pane shows the message 'Sequence created.' and a execution time of '0.02 seconds'.

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

QUERY:

SELECT sequence_name, max_value, increment_by, last_number FROM user_sequences;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The results of the query 'SELECT sequence_name, max_value, increment_by, last_number FROM user_sequences;' are displayed in a table. The table has four columns: SEQUENCE_NAME, MAX_VALUE, INCREMENT_BY, and LAST_NUMBER. There is one row with the values: DEPT_ID_SEQ, 1000, 10, and 200. At the bottom of the results pane, it says '1 rows returned in 0.01 seconds'.

SEQUENCE_NAME	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
DEPT_ID_SEQ	1000	10	200

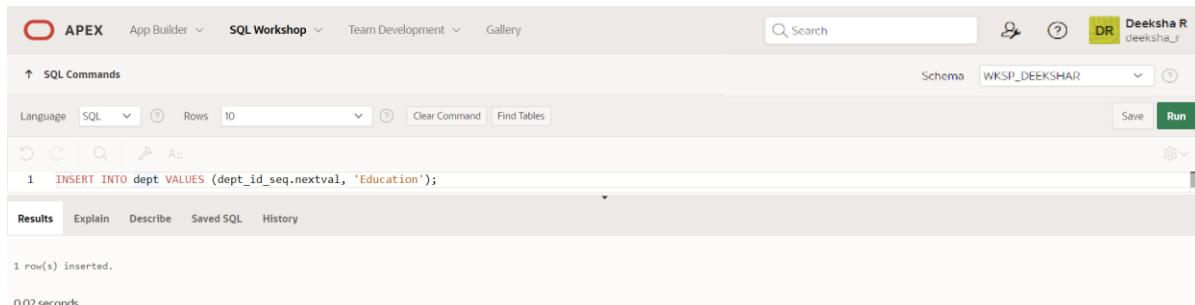
3.) Write a script to insert two rows into the DEPT table. Name your script lab12_3.sql. Be sure to use the sequence that you created for the ID column. Add two departments named Education and Administration. Confirm your additions. Run the commands in your script.

QUERY:

INSERT INTO dept VALUES (dept_id_seq.nextval, 'Education');

INSERT INTO dept VALUES (dept_id_seq.nextval, 'Administration');

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, a user icon for 'Deeksha R', and a schema dropdown set to 'WKSP_DEEKSHAR'. The main area shows a SQL command window with the following content:

```
1  INSERT INTO dept VALUES (dept_id_seq.nextval, 'Education');
```

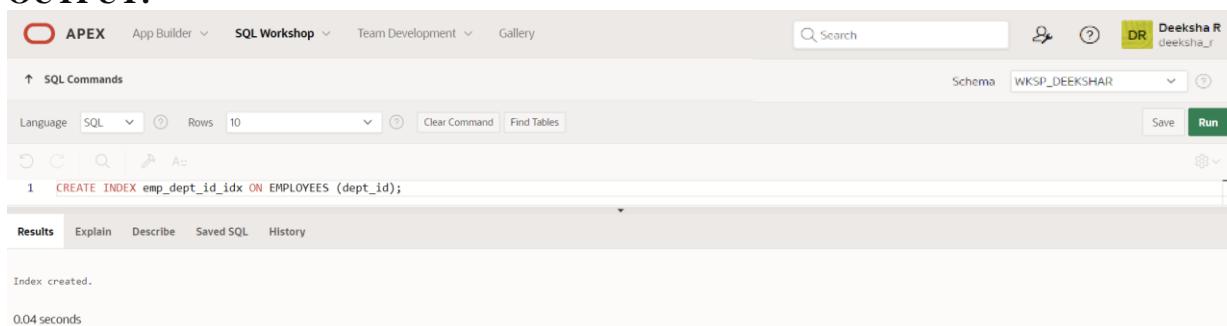
The results tab shows the output: '1 row(s) inserted.' and '0.02 seconds' execution time.

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

QUERY:

CREATE INDEX emp_dept_id_idx ON EMPLOYEES (department_id);

OUTPUT:



A screenshot of the Oracle SQL Workshop interface, similar to the previous one. The SQL command window contains:

```
1  CREATE INDEX emp_dept_id_idx ON EMPLOYEES (dept_id);
```

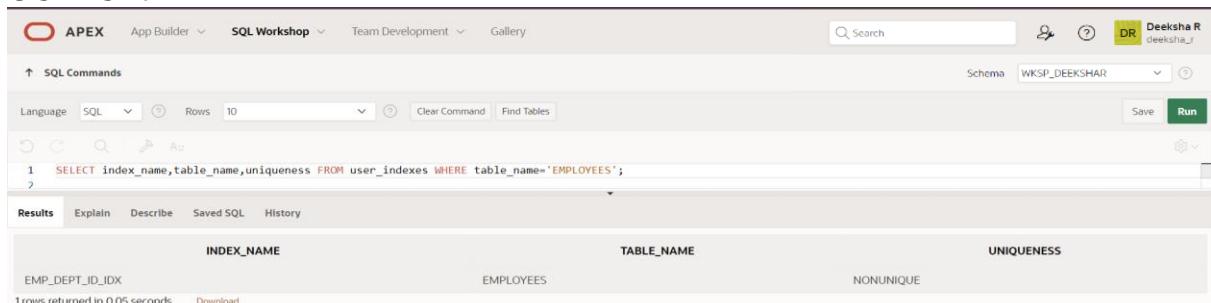
The results tab shows 'Index created.' and '0.04 seconds' execution time.

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

QUERY:

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

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The SQL command window contains:

```
1  SELECT index_name,table_name,uniqueness FROM user_indexes WHERE table_name='EMPLOYEES';
2
```

The results tab displays a table with three columns: 'INDEX_NAME', 'TABLE_NAME', and 'UNIQUENESS'. The data is as follows:

INDEX_NAME	TABLE_NAME	UNIQUENESS
EMP_DEPT_ID_IDX	EMPLOYEES	NONUNIQUE

1 rows returned in 0.05 seconds

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

RESULT:

CONTROLLING USER ACCESS

EX.NO:15

DATE:

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

The CREATE SESSION system privilege

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

The CREATE TABLE privilege

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

You can, or anyone you have given those privileges to by using the WITH GRANT OPTION.

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?

Create a role containing the system privileges and grant the role to the users

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

The ALTER USER statement

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

Team 2 executes the GRANT statement. GRANT select ON departments TO <user1>;

Team 1 executes the GRANT statement. GRANT select ON departments TO <user2>;

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 executes this INSERT statement. INSERT INTO departments(department_id, department_name) VALUES (500, 'Education'); COMMIT;

Team 2 executes this INSERT statement. INSERT INTO departments(department_id, department_name) VALUES (510, 'Administration'); COMMIT;

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

```
SELECT table_name FROM user_tables;
```

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

Team 1 revokes the privilege.

```
REVOKE select  
ON departments  
FROM user2;
```

Team 2 revokes the privilege.

```
REVOKE select  
ON departments  
FROM user1;
```

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

Team 1 executes this INSERT statement.

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

Team 2 executes this INSERT statement.

```
DELETE FROM departments  
WHERE department_id = 510;  
COMMIT;
```

<u>Evaluation Procedure</u>	<u>Marks awarded</u>
<u>Practice Evaluation (5)</u>	
<u>Viva(5)</u>	
<u>Total (10)</u>	
<u>Faculty Signature</u>	

RESULT:

PL/SQL CONTROL STRUCTURES

EX.NO:16

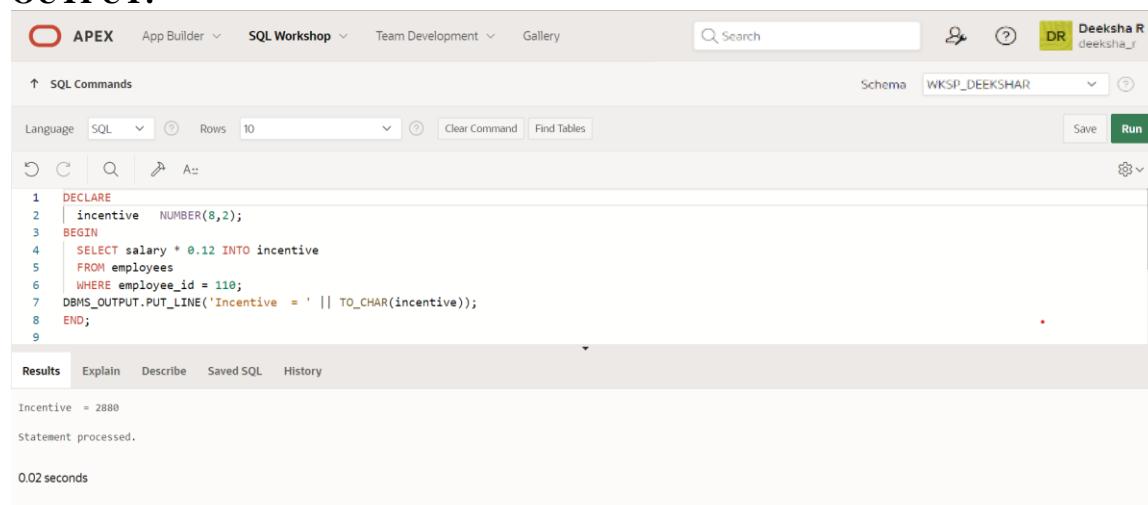
DATE:

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

QUERY:

```
DECLARE
    incentive  NUMBER(8,2);
BEGIN
    SELECT salary*0.12 INTO incentive
    FROM employees
    WHERE employee_id = 110;
    DBMS_OUTPUT.PUT_LINE('Incentive = ' || TO_CHAR(incentive));
END;
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The code entered is:

```
1 DECLARE
2 | incentive  NUMBER(8,2);
3 BEGIN
4 |   SELECT salary * .12 INTO incentive
5 |   FROM employees
6 |   WHERE employee_id = 110;
7 |   DBMS_OUTPUT.PUT_LINE('Incentive = ' || TO_CHAR(incentive));
8 |
9 
```

The results pane shows the output:

```
Incentive = 2880
Statement processed.
0.02 seconds
```

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

QUERY:

```
DECLARE
WELCOME varchar2(10) := 'welcome';
BEGIN
DBMS_Output.Put_Line("Welcome");
END;
```

```
DECLARE
WELCOME varchar2(10) := 'welcome';
BEGIN
DBMS_Output.Put_Line("Welcome");
END;
```

/

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area displays a PL/SQL block:

```
1 DECLARE
2   WELCOME varchar2(10) := 'welcome'; -- identifier without quotation
3 BEGIN
4   DBMS_Output.Put_Line("Welcome"); --reference to the identifier with quotation and different case
5 END;
```

In the 'Results' tab, an error message is displayed:

```
Error at line 4/25: ORA-06550: line 4, column 25:
PLS-00201: identifier 'Welcome' must be declared
ORA-06512: at "SYS.WMV_DBMS_SQL_APEX_230200", line 801
ORA-06550: line 4, column 3:
PL/SQL: Statement ignored
```

The error message is highlighted with a yellow box.

3.) Write a PL/SQL block to adjust the salary of the employee whose ID 122.

QUERY:

DECLARE

 salary_of_emp NUMBER(8,2);

PROCEDURE approx_salary (

 emp NUMBER,

 empsal IN OUT NUMBER,

 adddress NUMBER

) IS

BEGIN

 empsal := empsal + adddress;

END;

BEGIN

 SELECT salary INTO salary_of_emp

 FROM employees

 WHERE employee_id = 122;

 DBMS_OUTPUT.PUT_LINE

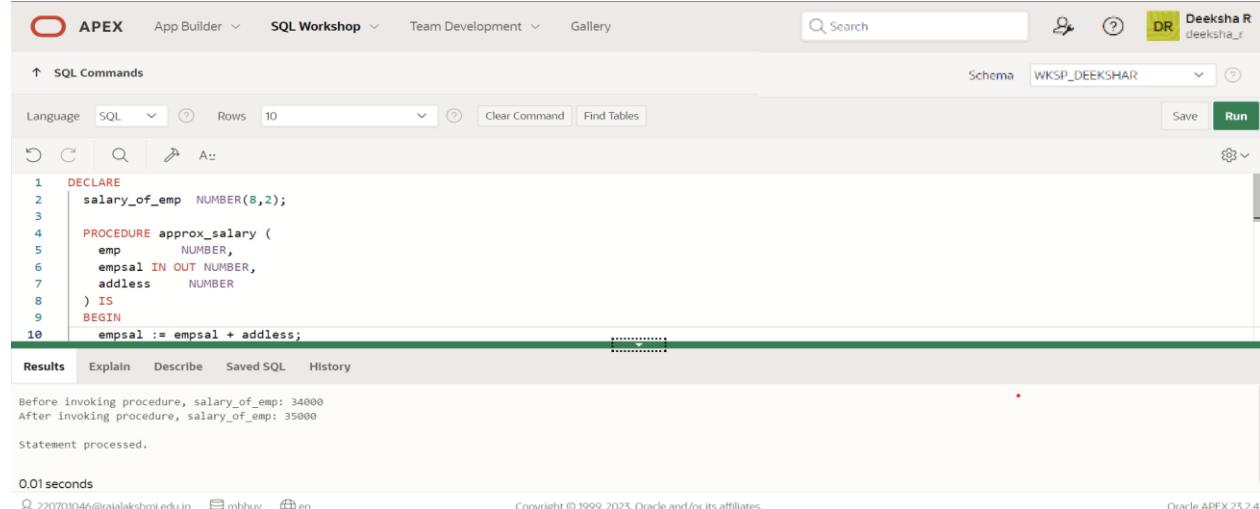
 ('Before invoking procedure, salary_of_emp: ' || salary_of_emp);

 approx_salary (100, salary_of_emp, 1000);

 DBMS_OUTPUT.PUT_LINE

```
('After invoking procedure, salary_of_emp: ' || salary_of_emp);
END;
/
```

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile for 'Deeksha R' (deeksha_r). The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP_DEEKSHAR'. Below the title are buttons for 'Save' and 'Run'. The code editor contains the following PL/SQL block:

```
1  DECLARE
2      salary_of_emp  NUMBER(8,2);
3
4      PROCEDURE approx_salary (
5          emp        NUMBER,
6          empsal IN OUT NUMBER,
7          address    NUMBER
8      ) IS
9      BEGIN
10         empsal := empsal + address;
```

Below the code, the results section displays the output of the procedure execution:

```
Before invoking procedure, salary_of_emp: 34000
After invoking procedure, salary_of_emp: 35000
Statement processed.
```

Execution time is listed as '0.01 seconds'. The bottom of the screen shows copyright information and the version 'Oracle APEX 23.2.4'.

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.

QUERY:

```
CREATE OR REPLACE PROCEDURE pri_bool(
    boo_name  VARCHAR2,
    boo_val   BOOLEAN
) IS
BEGIN
    IF boo_val IS NULL THEN
        DBMS_OUTPUT.PUT_LINE( boo_name || ' = NULL');
    ELSIF boo_val = TRUE THEN
        DBMS_OUTPUT.PUT_LINE( boo_name || ' = TRUE');
    ELSE
        DBMS_OUTPUT.PUT_LINE( boo_name || ' = FALSE');
    END IF;
END;
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The main area displays a PL/SQL code editor with the following content:

```

1 CREATE OR REPLACE PROCEDURE pri_bool(
2     boo_name    VARCHAR2,
3     boo_val     BOOLEAN
4 ) IS
5 BEGIN
6     IF boo_val IS NULL THEN
7         DBMS_OUTPUT.PUT_LINE( boo_name || ' = NULL');
8     ELSIF boo_val = TRUE THEN
9         DBMS_OUTPUT.PUT_LINE( boo_name || ' = TRUE');
10    ELSE

```

Below the code editor, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, showing the output: "Procedure created." and "0.01 seconds". At the bottom, the user information is shown as 220701046@rajalakshmi.edu.in, mbhuv, and the copyright notice "Copyright © 1999, 2023, Oracle and/or its affiliates." The footer also indicates "Oracle APEX 25.2.4".

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

QUERY:

DECLARE

```

PROCEDURE pat_match (
    test_string  VARCHAR2,
    pattern      VARCHAR2
) IS
BEGIN
    IF test_string LIKE pattern THEN
        DBMS_OUTPUT.PUT_LINE ('TRUE');
    ELSE
        DBMS_OUTPUT.PUT_LINE ('FALSE');
    END IF;
END;
/

```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The main area displays a PL/SQL procedure named 'pat_match'. The code is as follows:

```

1  DECLARE
2      PROCEDURE pat_match (
3          test_string    VARCHAR2,
4          pattern        VARCHAR2
5      ) IS
6      BEGIN
7          IF test_string LIKE pattern THEN
8              DBMS_OUTPUT.PUT_LINE ('TRUE');
9          ELSE

```

Below the code, the results show:

```

TRUE
FALSE
Statement processed.
0.01 seconds

```

At the bottom, the footer includes copyright information and the version Oracle APEX 23.2.4.

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

QUERY:

DECLARE

num_small NUMBER := 8;

num_large NUMBER := 5;

num_temp NUMBER;

BEGIN

IF num_small > num_large THEN

num_temp := num_small;

num_small := num_large;

num_large := num_temp;

END IF;

DBMS_OUTPUT.PUT_LINE ('num_small ='||num_small);

DBMS_OUTPUT.PUT_LINE ('num_large ='||num_large);

END;

OUTPUT:

```

1  DECLARE
2    num_small NUMBER := 8;
3    num_large NUMBER := 5;
4    num_temp NUMBER;
5    BEGIN
6
7    IF num_small > num_large THEN
8      num_temp := num_small;
9      num_small := num_large;
10     num_large := num_temp;
11   END IF;

```

Results Explain Describe Saved SQL History

```

num_small = 5
num_large = 8

Statement processed.

```

220701046@rajalakshmi.edu.in mbhuv en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

QUERY:

DECLARE

```

PROCEDURE test1 (
  sal_achieve NUMBER,
  target_qty NUMBER,
  emp_id NUMBER
)
IS
  incentive NUMBER := 0;
  updated VARCHAR2(3) := 'No';
BEGIN
  IF sal_achieve > (target_qty + 200) THEN
    incentive := (sal_achieve - target_qty)/4;
    UPDATE employees
    SET salary = salary + incentive
    WHERE employee_id = emp_id;
    updated := 'Yes';
  END IF;
  DBMS_OUTPUT.PUT_LINE (
    'Table updated? ' || updated || ', '
    'incentive = ' || incentive || ''
  );
END test1;

```

```

BEGIN
  test1(2300, 2000, 144);
  test1(3600, 3000, 145);
END;
/

```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile for 'Deeksha R' (deeksha_r). The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP_DEEKSHAR'. Below the title are buttons for 'Save' and 'Run'. The code editor contains the following PL/SQL procedure:

```

1  DECLARE
2    PROCEDURE test1 (
3      sal_achieve NUMBER,
4      target_qty NUMBER,
5      emp_id NUMBER
6    )
7    IS
8      incentive NUMBER := 0;
9      updated VARCHAR2(3) := 'No';
10     BEGIN
11       IF sal_achieve > (target_qty + 200) THEN
12         incentive := 150;
13       ELSE
14         incentive := 75;
15       END IF;
16       DBMS_OUTPUT.NEW_LINE;
17       DBMS_OUTPUT.PUT_LINE (
18         'Sale achieved : ' || sal_achieve || ', incentive : ' || incentive || '.'
19       );
20     END;
21   BEGIN
22     test1(45000);
23     test1(36000);
24   END;

```

The results pane shows the output of the procedure execution:

```

Table updated? Yes, incentive = 75.
Table updated? Yes, incentive = 150.
1 row(s) updated.

0.02 seconds

```

At the bottom, it shows the URL '220701046@rajalakshmi.edu.in', the session ID 'mbhuv', and the language 'en'. The copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 23.2.4' are also visible.

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

QUERY:

```

DECLARE
  PROCEDURE test1 (sal_achieve NUMBER)
  IS
    incentive NUMBER := 0;
  BEGIN
    IF sal_achieve > 44000 THEN
      incentive := 1800;
    ELSIF sal_achieve > 32000 THEN
      incentive := 800;
    ELSE
      incentive := 500;
    END IF;
    DBMS_OUTPUT.NEW_LINE;
    DBMS_OUTPUT.PUT_LINE (
      'Sale achieved : ' || sal_achieve || ', incentive : ' || incentive || '.'
    );
  END test1;
BEGIN
  test1(45000);
  test1(36000);

```

```
test1(28000);
```

```
END;
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The search bar contains 'Search'. On the right, there's a user icon, a help icon, and a yellow box labeled 'DR Deeksha R deeksha_r'. The schema dropdown is set to 'WKSP_DEEKSHAR'. Below the toolbar, the language is set to 'SQL' and rows are set to '10'. There are buttons for 'Clear Command' and 'Find Tables'. The main area displays the PL/SQL code for a procedure named 'test1'. The code uses IF-THEN-ELSIF-ELSE logic to calculate an incentive based on a salary achievement. It also includes DBMS_OUTPUT.NEW_LINE and DBMS_OUTPUT.PUT_LINE statements. The results section shows the output of the procedure execution, which includes three rows of data and a final message 'Statement processed.'.

```
1  DECLARE
2      PROCEDURE test1 (sal_achieve NUMBER)
3  IS
4      incentive NUMBER := 0;
5  BEGIN
6      IF sal_achieve > 44000 THEN
7          incentive := 1800;
8      ELSIF sal_achieve > 32000 THEN
9          incentive := 800;
10     ELSE
11         incentive := 500;
12     END IF;
13     DBMS_OUTPUT.NEW_LINE;
14     DBMS_OUTPUT.PUT_LINE (

```

Sale achieved : 45000, incentive : 1800.
Sale achieved : 36000, incentive : 800.
Sale achieved : 28000, incentive : 500.
Statement processed.

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.

QUERY:

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
    tot_emp NUMBER;
    get_dep_id NUMBER;
```

```
BEGIN
```

```
    get_dep_id := 80;
    SELECT Count(*)
    INTO tot_emp
    FROM employees e
    join departments d
        ON e.department_id = d.department_id
    WHERE e.department_id = get_dep_id;
```

```
    dbms_output.Put_line ('The employees are in the department'||get_dep_id|| is: '
```

```
    ||To_char(tot_emp));
```

```
    IF tot_emp >= 45 THEN
```

```
        dbms_output.Put_line ('There are no vacancies in the department'||get_dep_id);
```

```
    ELSE
```

```

    dbms_output.Put_line ('There are'||to_char(45-tot_emp)||" vacancies in department '||get_dep_id );
    END IF;
END;
/
OUTPUT:

```

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.

QUERY:

```

DECLARE
    tot_emp NUMBER;
    get_dep_id NUMBER;

BEGIN
    get_dep_id := 80;
    SELECT Count(*)
    INTO tot_emp
    FROM employees e
        join departments d
        ON e.department_id = d.dept_id
    WHERE e.department_id = get_dep_id;

    dbms_output.Put_line ('The employees are in the department'||get_dep_id||' is: '
        ||To_char(tot_emp));

    IF tot_emp >= 45 THEN
        dbms_output.Put_line ('There are no vacancies in the department'||get_dep_id);
    ELSE
        dbms_output.Put_line ('There are'||to_char(45-tot_emp)||" vacancies in department '||get_dep_id );
    END IF;
END;

```

/

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows the user 'Deeksha R' (deeksha_r) and the schema 'WKSP_DEEKSHAR'. The main area is titled 'SQL Commands' with a 'Language' dropdown set to 'SQL'. The code editor contains the following PL/SQL block:

```
1  DECLARE
2      tot_emp NUMBER;
3      get_dep_id NUMBER;
4
5  BEGIN
6      get_dep_id := 80;
7      SELECT Count(*)
8          INTO tot_emp
9         FROM employees e
10        JOIN department d
11           ON e.dept_id = d.dept_id
```

Below the code, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab displays the output:

```
The employees are in the department 80 is: 0
There are 45 vacancies in department 80

Statement processed.

0.02 seconds
```

At the bottom, it shows the session details '220701046@rajalakshmi.edu.in mbhuv en' and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' followed by 'Oracle APEX 25.2.4'.

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

QUERY:

DECLARE

```
v_employee_id employees.employee_id%TYPE;
v_full_name employees.first_name%TYPE;
v_job_id employees.job_id%TYPE;
v_hire_date employees.hire_date%TYPE;
v_salary employees.salary%TYPE;
```

CURSOR c_employees IS

```
SELECT employee_id, first_name || ' ' || last_name AS full_name, job_id, hire_date, salary
  FROM employees;
```

BEGIN

```
DBMS_OUTPUT.PUT_LINE('Employee ID | Full Name | Job Title | Hire Date | Salary');
```

```
DBMS_OUTPUT.PUT_LINE('-----');
```

```
OPEN c_employees;
```

```
FETCH c_employees INTO v_employee_id, v_full_name, v_job_id, v_hire_date, v_salary;
WHILE c_employees%FOUND LOOP
```

```
    DBMS_OUTPUT.PUT_LINE(v_employee_id || ' ' || v_full_name || ' ' || v_job_id || ' '
    || v_hire_date || ' ' || v_salary);
```

```
    FETCH c_employees INTO v_employee_id, v_full_name, v_job_id, v_hire_date,
v_salary;
```

```
END LOOP;
```

```
CLOSE c_employees;
```

```
END;
```

/

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a user profile for 'Deeksha R' (deeksha_r) and a schema dropdown set to 'WKSP_DEEKSHAR'. The main area has tabs for 'SQL Commands' and 'Results' (which is selected). The SQL Commands tab contains the following PL/SQL code:

```
8  SELECT employee_id, first_name || ' ' || last_name AS full_name, job_id, hire_date, salary
9  FROM employees;
10 BEGIN
11   DBMS_OUTPUT.PUT_LINE('Employee ID | Full Name | Job Title | Hire Date | Salary');
12   DBMS_OUTPUT.PUT_LINE('-----');
13   OPEN c_employees;
14   FETCH c_employees INTO v_employee_id, v_full_name, v_job_id, v_hire_date, v_salary;
15   WHILE c_employees%FOUND LOOP
16     DBMS_OUTPUT.PUT_LINE(v_employee_id || ' ' || v_full_name || ' ' || v_job_id || ' ' || v_hire_date || ' ' || v_salary);
17     FETCH c_employees INTO v_employee_id, v_full_name, v_job_id, v_hire_date, v_salary;

```

The Results tab displays the output of the program, which is a table with columns 'Employee ID | Full Name | Job Title | Hire Date | Salary'. The data rows are:

Employee ID Full Name Job Title Hire Date Salary
2 john Matos st_clerk 01/04/1996 50000
5 sneha brindha st_clerk 02/02/2005 12222
110 shreya King st_joseph 03/03/1988 24000
4 bhuvana st_mary 06/16/1995 100000
122 ramesh Davies st_clerk 12/31/1998 34000
55 Zlotkey st_joseph 01/01/1999 3200

Below the results, a message says 'Statement processed.' and the footer includes copyright information: 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 25.2.4'.

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

QUERY:

DECLARE

 CURSOR emp_cursor IS

```
    SELECT e.employee_id, e.first_name, m.first_name AS manager_name
      FROM employees e
      LEFT JOIN employees m ON e.manager_id = m.employee_id;
    emp_record emp_cursor%ROWTYPE;
  BEGIN
    OPEN emp_cursor;
    FETCH emp_cursor INTO emp_record;
    WHILE emp_cursor%FOUND LOOP
      DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp_record.employee_id);
      DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_record.first_name);
      DBMS_OUTPUT.PUT_LINE('Manager Name: ' || emp_record.manager_name);
      DBMS_OUTPUT.PUT_LINE('-----');
      FETCH emp_cursor INTO emp_record;
    END LOOP;
    CLOSE emp_cursor;
  END;
/
```

OUTPUT:

```

1  DECLARE
2      CURSOR emp_cursor IS
3          SELECT e.employee_id, e.first_name, m.first_name AS manager_name
4          FROM employees e
5              LEFT JOIN employees m ON e.manager_id = m.employee_id;
6  emp_record emp_cursor%ROWTYPE;

```

Results

Employee ID	Employee Name	Manager Name
122	ramesh	
4	john	
5	sneha	

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

```

DECLARE
    CURSOR job_cursor IS
        SELECT e.job_id, j.lowest_sal
        FROM job_grade j,employees e;
    job_record job_cursor%ROWTYPE;
BEGIN
    OPEN job_cursor;
    FETCH job_cursor INTO job_record;
    WHILE job_cursor%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE('Job ID: ' || job_record.job_id);
        DBMS_OUTPUT.PUT_LINE('Minimum Salary: ' || job_record.lowest_sal);
        DBMS_OUTPUT.PUT_LINE('-----');
        FETCH job_cursor INTO job_record;
    END LOOP;
    CLOSE job_cursor;
END;
/

```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is active. The main area is titled 'SQL Commands'. The code entered is:

```

1  DECLARE
2      CURSOR job_cursor IS
3          SELECT e.job_id, j.lowest_sal
4              FROM job_grades j,employees e;
5      job_record job_cursor%ROWTYPE;
6  BEGIN
7      OPEN job_cursor;
8      FETCH job_cursor INTO job_record;
9      WHILE job_cursor%FOUND LOOP

```

The 'Results' tab is selected, displaying the output of the query:

```

Job ID: st_clerk
Minimum Salary: 11000000
-----
Job ID: st_clerk
Minimum Salary: 11000000
-----
Job ID: st_joseph
Minimum Salary: 11000000
-----
Job ID: st_mary
Minimum Salary: 11000000

```

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

QUERY:

DECLARE

```

CURSOR employees_cur IS
    SELECT employee_id,last_name,job_id,start_date
        FROM employees NATURAL join job_history;
    emp_start_date DATE;

```

BEGIN

```

dbms_output.Put_line(Rpad('Employee ID', 15)||Rpad('Last Name', 25)|| Rpad('Job Id', 35)
||'Start Date');

```

```

dbms_output.Put_line('-----'
-----');

```

FOR emp_sal_rec IN employees_cur LOOP

-- find out most recent end_date in job_history

```

SELECT Max(end_date) + 1

```

```

INTO emp_start_date

```

```

FROM job_history

```

```

WHERE employee_id = emp_sal_rec.employee_id;

```

```

IF emp_start_date IS NULL THEN

```

```

    emp_start_date := emp_sal_rec.start_date;

```

```

END IF;

```

```

dbms_output.Put_line(Rpad(emp_sal_rec.employee_id, 15)

```

```

    ||Rpad(emp_sal_rec.last_name, 25)

```

```

    || Rpad(emp_sal_rec.job_id, 35)

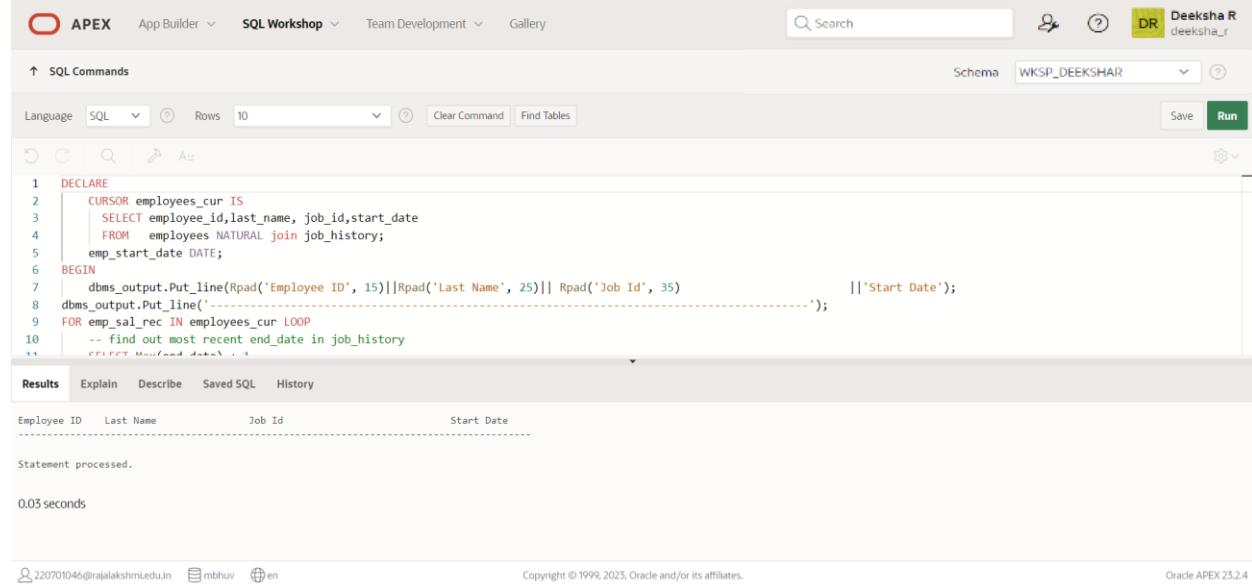
```

```

    || To_char(emp_start_date, 'dd-mon-yyyy'));
END LOOP;
END;
/

```

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The code in the editor is:

```

1  DECLARE
2      CURSOR employees_cur IS
3          SELECT employee_id, last_name, job_id, start_date
4          FROM employees NATURAL JOIN job_history;
5          emp_start_date DATE;
6      BEGIN
7          dbms_output.Put_line(Rpad('Employee ID', 15) || Rpad('Last Name', 25) || Rpad('Job Id', 35) || 'Start Date');
8          dbms_output.Put_line('-----');
9          FOR emp_sal_rec IN employees_cur LOOP
10              -- find out most recent end_date in job_history
11              SELECT Max(end_date) ... ;

```

The results pane shows the output:

Employee ID	Last Name	Job Id	Start Date

Statement processed.
0.03 seconds

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

QUERY:

DECLARE

```

v_employee_id employees.employee_id%TYPE;
v_first_name employees.last_name%TYPE;
v_end_date job_history.end_date%TYPE;

CURSOR c_employees IS
    SELECT e.employee_id, e.first_name, jh.end_date
    FROM employees e
    JOIN job_history jh ON e.employee_id = jh.employee_id;
BEGIN
    OPEN c_employees;
    FETCH c_employees INTO v_employee_id, v_first_name, v_end_date;
    WHILE c_employees%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_employee_id);
        DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_first_name);
        DBMS_OUTPUT.PUT_LINE('End Date: ' || v_end_date);
        DBMS_OUTPUT.PUT_LINE('-----');
        FETCH c_employees INTO v_employee_id, v_first_name, v_end_date;
    END LOOP;
    CLOSE c_employees;
END;

```

OUTPUT:

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language: SQL Rows: 10 Clear Command Find Tables Save Run

```

1 DECLARE
2   v_employee_id employees.employee_id%TYPE;
3   v_first_name employees.last_name%TYPE;
4   v_end_date job_history.end_date%TYPE;
5   CURSOR c_employees IS
6     SELECT e.employee_id, e.first_name, jh.end_date
7       FROM employees e
8      JOIN job_history jh ON e.employee_id = jh.employee_id;
9   BEGIN
10    OPEN c_employees;
11    FETCH c_employees INTO v_employee_id, v_first_name, v_end_date;
12    DBMS_OUTPUT.PUT_LINE('Employee ID: '||v_employee_id||' Employee Name: '||v_first_name);
13    DBMS_OUTPUT.PUT_LINE('End Date: '||v_end_date);
14  END;
15  /

```

Results Explain Describe Saved SQL History

Employee ID: 2
Employee Name: john
End Date: 01/31/2020

Employee ID: 4
Employee Name:
End Date:

Employee ID: 2
Employee Name:

Copyright © 1999, 2023, Oracle and/or its affiliates.

Oracle APEX 23.2.0

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

RESULT:

PROCEDURES AND FUNCTIONS

EX.NO: 17

DATE:

1.) Factorial of a number using function.

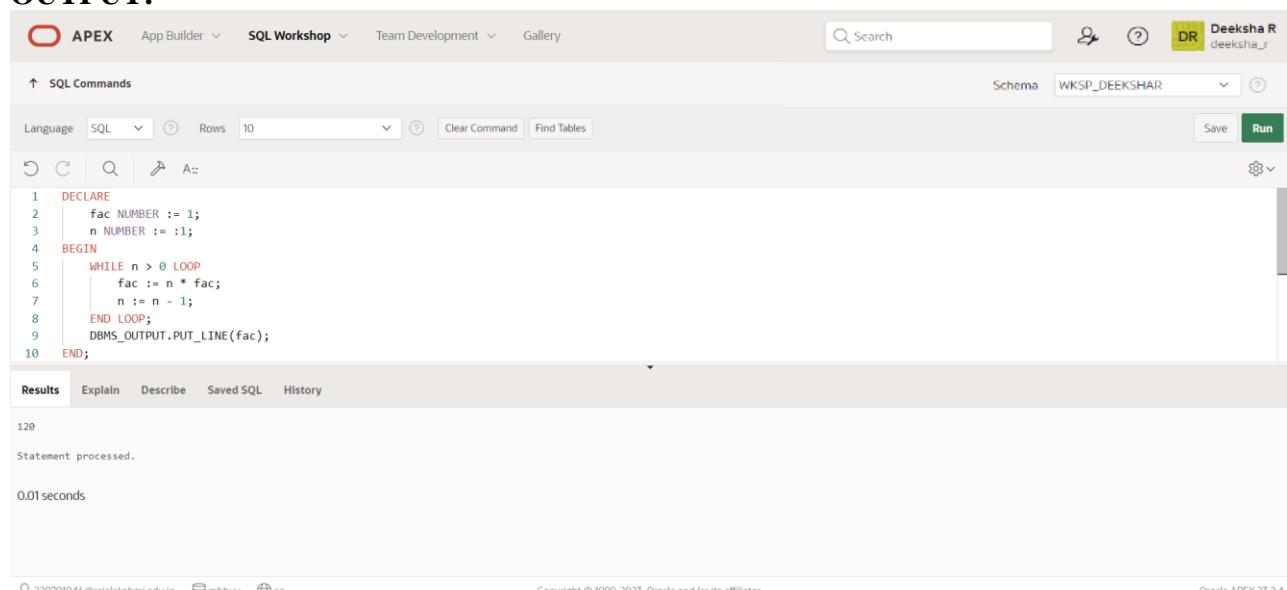
QUERY:

DECLARE

```

    fac NUMBER := 1;
    n NUMBER := :1;
BEGIN
    WHILE n > 0 LOOP
        fac := n * fac;
        n := n - 1;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE(fac);
END;
```

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a user profile for 'Deeksha R' and a search bar. The main workspace has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, displaying the following PL/SQL code:

```

1  DECLARE
2      fac NUMBER := 1;
3      n NUMBER := :1;
4  BEGIN
5      WHILE n > 0 LOOP
6          fac := n * fac;
7          n := n - 1;
8      END LOOP;
9      DBMS_OUTPUT.PUT_LINE(fac);
10 END;
```

Below the code, the Results tab shows the output:

```

120
Statement processed.

0.01 seconds
```

At the bottom, the footer includes copyright information: Copyright © 1999, 2023, Oracle and/or its affiliates. and Oracle APEX 23.2.4.

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

QUERY:

```

CREATE OR REPLACE PROCEDURE get_book_info (
    p_book_id IN NUMBER,
    p_title IN OUT VARCHAR2,
    p_author OUT VARCHAR2,
    p_year_published OUT NUMBER
)
```

```

AS
BEGIN
    SELECT title, author, year_published INTO p_title, p_author, p_year_published
    FROM books
    WHERE book_id = p_book_id;

    p_title := p_title || ' - Retrieved';
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        p_title := NULL;
        p_author := NULL;
        p_year_published := NULL;
END;

```

DECLARE

```

v_book_id NUMBER := 1;
v_title VARCHAR2(100);
v_author VARCHAR2(100);
v_year_published NUMBER;
BEGIN
    v_title := 'Initial Title';

    get_book_info(p_book_id => v_book_id, p_title => v_title, p_author => v_author,
    p_year_published => v_year_published);

    DBMS_OUTPUT.PUT_LINE('Title: ' || v_title);
    DBMS_OUTPUT.PUT_LINE('Author: ' || v_author);
    DBMS_OUTPUT.PUT_LINE('Year Published: ' || v_year_published);
END;

```

OUTPUT:

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language: SQL Rows: 10 Schema: WKSP_DEEKSHAR Save Run

```

1 CREATE TABLE books (
2     book_id NUMBER PRIMARY KEY,
3     title VARCHAR2(100),
4     author VARCHAR2(100),
5     year_published NUMBER
6 );
7 INSERT INTO books (book_id, title, author, year_published) VALUES (1, '1984', 'George Orwell', 1949);
8 INSERT INTO books (book_id, title, author, year_published) VALUES (2, 'To Kill a Mockingbird', 'Harper Lee', 1960);
9 INSERT INTO books (book_id, title, author, year_published) VALUES (3, 'The Great Gatsby', 'F. Scott Fitzgerald', 1925);
10
11 CREATE OR REPLACE PROCEDURE get_book_info (
12     p_book_id IN NUMBER,
13     p_title OUT VARCHAR2,
14     p_author OUT VARCHAR2
15 );

```

Results Explain Describe Saved SQL History

Title: 1984
Author: George Orwell
Year Published: 1949
Statement processed.

220701046@rajalakshmi.edu.in mbhuv en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

RESULT:

TRIGGER

EX_NO: 18

DATE:

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

QUERY:

```
CREATE OR REPLACE TRIGGER prevent_parent_deletion
```

```
BEFORE DELETE ON parent_table
```

```
FOR EACH ROW
```

```
DECLARE
```

```
    child_exists EXCEPTION;
```

```
    PRAGMA EXCEPTION_INIT(child_exists, -20001);
```

```
    v_child_count NUMBER;
```

```
BEGIN
```

```
    SELECT COUNT(*) INTO v_child_count FROM child_table WHERE  
parent_id = :OLD.parent_id;
```

```
    IF v_child_count > 0 THEN
```

```
        RAISE child_exists;
```

```
    END IF;
```

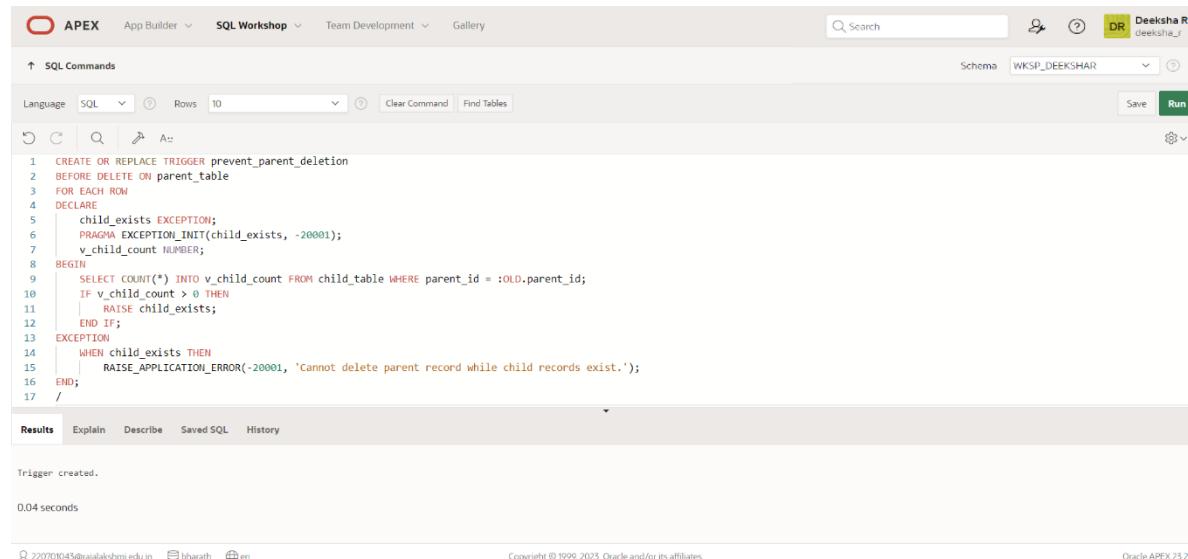
```
EXCEPTION
```

```
    WHEN child_exists THEN
```

```
        RAISE_APPLICATION_ERROR(-20001, 'Cannot delete parent record  
while child records exist.');
```

```
END;
```

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The right side shows a user profile for 'Deeksha R' (Deeksha_R). The main workspace has a toolbar with icons for Undo, Redo, Search, and Run. Below the toolbar, there are dropdown menus for Language (SQL), Rows (10), and a 'Clear Command' button. The schema is set to 'WKSP_DEEKSHAR'. The bottom of the workspace shows tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The results area displays the SQL code for the trigger and its execution results.

```
1 CREATE OR REPLACE TRIGGER prevent_parent_deletion
2 BEFORE DELETE ON parent_table
3 FOR EACH ROW
4 DECLARE
5     child_exists EXCEPTION;
6     PRAGMA EXCEPTION_INIT(child_exists, -20001);
7     v_child_count NUMBER;
8 BEGIN
9     SELECT COUNT(*) INTO v_child_count FROM child_table WHERE parent_id = :OLD.parent_id;
10    IF v_child_count > 0 THEN
11        RAISE child_exists;
12    END IF;
13 EXCEPTION
14    WHEN child_exists THEN
15        RAISE_APPLICATION_ERROR(-20001, 'Cannot delete parent record while child records exist.');
16 END;
17 /
```

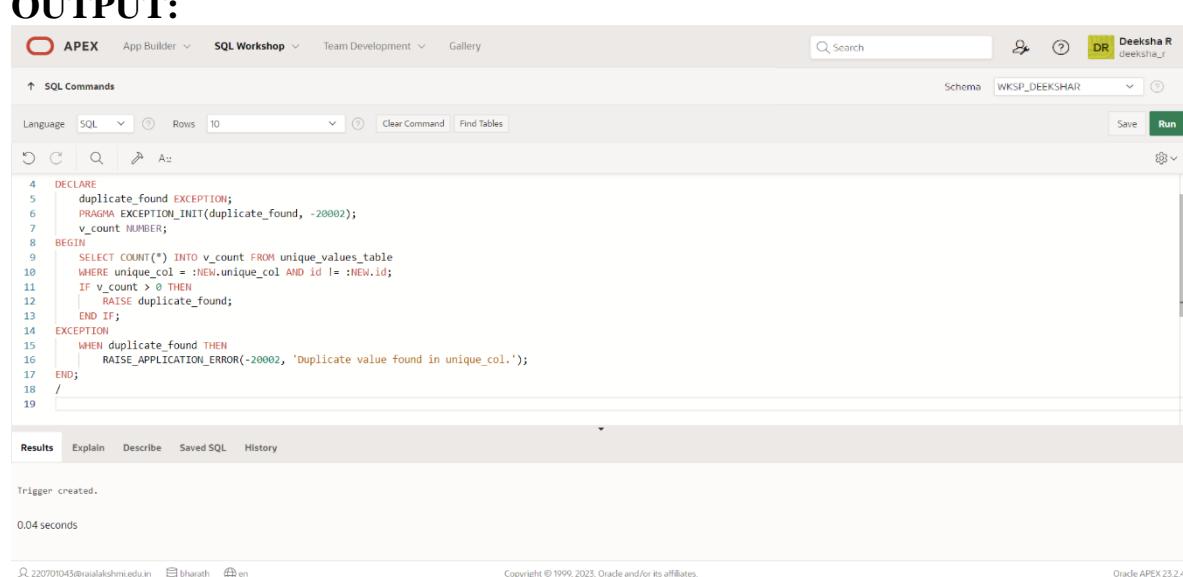
Trigger created.
0.04 seconds

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

QUERY:

```
CREATE OR REPLACE TRIGGER check_duplicates
BEFORE INSERT OR UPDATE ON unique_values_table
FOR EACH ROW
DECLARE
    duplicate_found EXCEPTION;
    PRAGMA EXCEPTION_INIT(duplicate_found, -20002);
    v_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_count FROM unique_values_table
    WHERE unique_col = :NEW.unique_col AND id != :NEW.id;
    IF v_count > 0 THEN
        RAISE duplicate_found;
    END IF;
EXCEPTION
    WHEN duplicate_found THEN
        RAISE_APPLICATION_ERROR(-20002, 'Duplicate value found in
unique_col.');
END;
```

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows the user 'Deeksha R' and the schema 'WKSP_DEEKSHAR'. The main area is titled 'SQL Commands' with a 'Save' and 'Run' button. The code area contains the PL/SQL trigger definition. The results tab at the bottom shows the message 'Trigger created.' and '0.04 seconds' execution time.

```
4  DECLARE
5      duplicate_found EXCEPTION;
6      PRAGMA EXCEPTION_INIT(duplicate_found, -20002);
7      v_count NUMBER;
8  BEGIN
9      SELECT COUNT(*) INTO v_count FROM unique_values_table
10     WHERE unique_col = :NEW.unique_col AND id != :NEW.id;
11     IF v_count > 0 THEN
12         RAISE duplicate_found;
13     END IF;
14 EXCEPTION
15     WHEN duplicate_found THEN
16         RAISE_APPLICATION_ERROR(-20002, 'Duplicate value found in unique_col.');
17 END;
18 /
19 
```

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

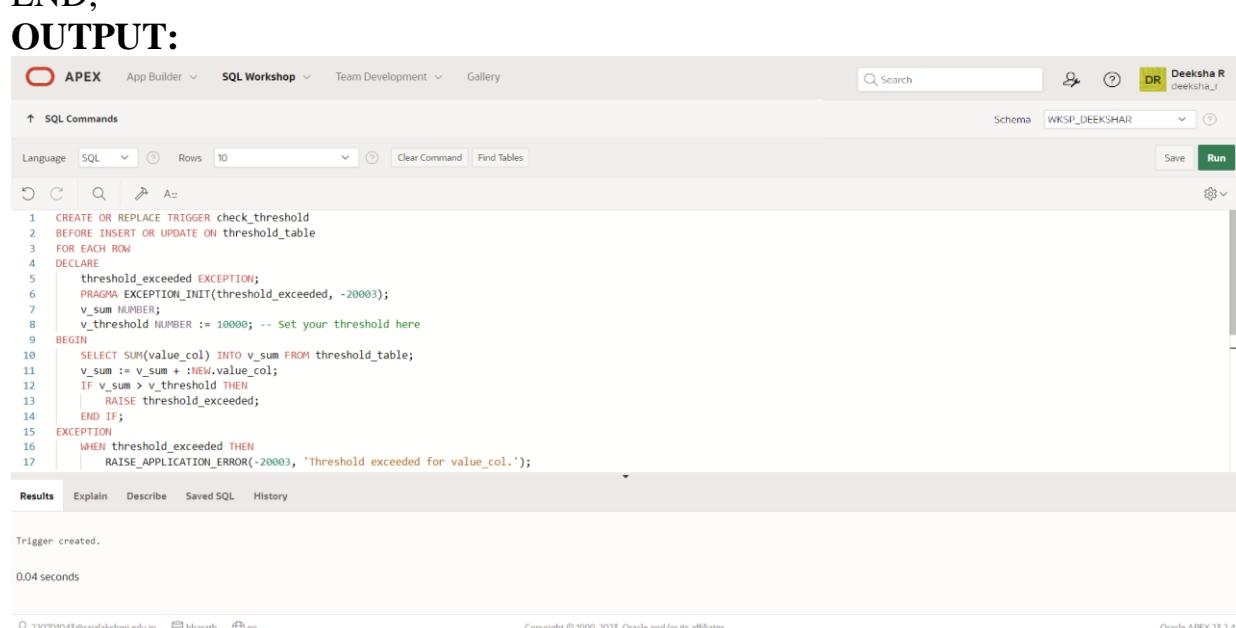
QUERY:

```
CREATE OR REPLACE TRIGGER check_threshold
BEFORE INSERT OR UPDATE ON threshold_table
FOR EACH ROW
```

DECLARE

```
    threshold_exceeded EXCEPTION;
    PRAGMA EXCEPTION_INIT(threshold_exceeded, -20003);
    v_sum NUMBER;
    v_threshold NUMBER := 10000; -- Set your threshold here
BEGIN
    SELECT SUM(value_col) INTO v_sum FROM threshold_table;
    v_sum := v_sum + :NEW.value_col;
    IF v_sum > v_threshold THEN
        RAISE threshold_exceeded;
    END IF;
EXCEPTION
    WHEN threshold_exceeded THEN
        RAISE_APPLICATION_ERROR(-20003, 'Threshold exceeded for
value_col.');
END;
```

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side of the header shows the user 'Deeksha R' and the schema 'WKSP_DEEKSHAR'. The main workspace is titled 'SQL Commands' and contains the PL/SQL code for the trigger. The code is numbered from 1 to 17. The 'Run' button is visible at the top right of the code area. Below the code, the 'Results' tab is selected, showing the output 'Trigger created.' and '0.04 seconds'. At the bottom, there are footer links for copyright information and language settings.

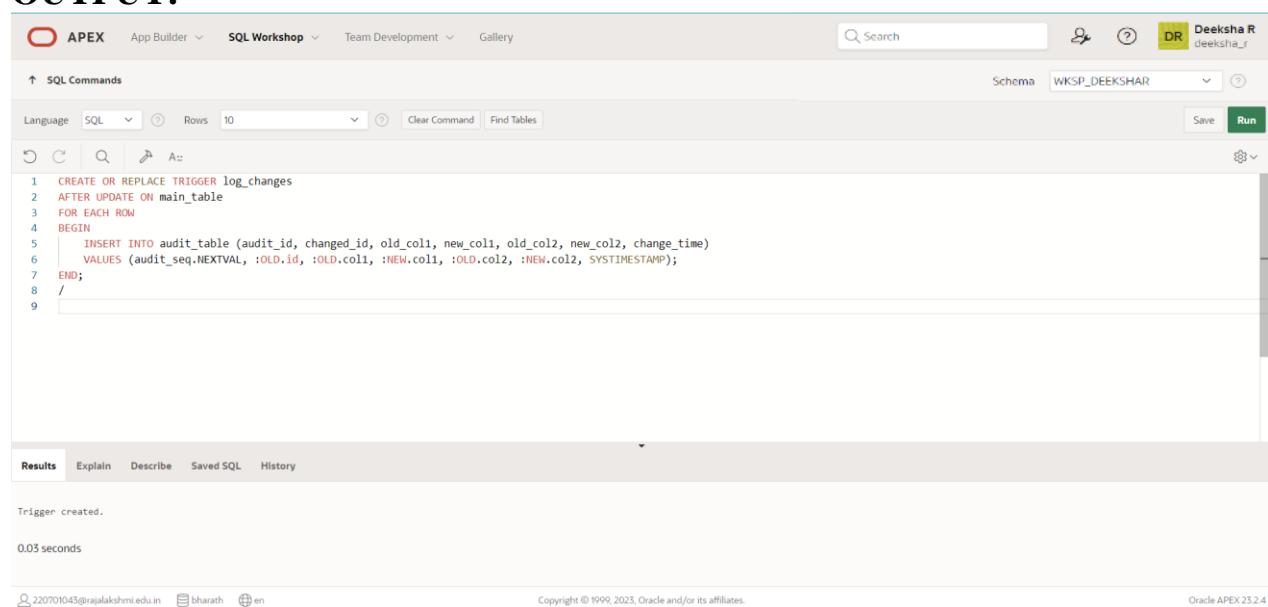
```
1 CREATE OR REPLACE TRIGGER check_threshold
2 BEFORE INSERT OR UPDATE ON threshold_table
3 FOR EACH ROW
4 DECLARE
5     threshold_exceeded EXCEPTION;
6     PRAGMA EXCEPTION_INIT(threshold_exceeded, -20003);
7     v_sum NUMBER;
8     v_threshold NUMBER := 10000; -- Set your threshold here
9 BEGIN
10     SELECT SUM(value_col) INTO v_sum FROM threshold_table;
11     v_sum := v_sum + :NEW.value_col;
12     IF v_sum > v_threshold THEN
13         RAISE threshold_exceeded;
14     END IF;
15 EXCEPTION
16     WHEN threshold_exceeded THEN
17         RAISE_APPLICATION_ERROR(-20003, 'Threshold exceeded for value_col.');
```

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.

QUERY:

```
CREATE OR REPLACE TRIGGER log_changes
AFTER UPDATE ON main_table
FOR EACH ROW
BEGIN
    INSERT INTO audit_table (audit_id, changed_id, old_col1, new_col1,
old_col2, new_col2, change_time)
    VALUES (audit_seq.NEXTVAL, :OLD.id, :OLD.col1, :NEW.col1,
:OLD.col2, :NEW.col2, SYSTIMESTAMP);
END;
```

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' and 'SQL Workshop' are selected. The schema is set to 'WKSP_DEEKSHAR'. The main area displays the SQL code for the trigger:

```
1 CREATE OR REPLACE TRIGGER log_changes
2 AFTER UPDATE ON main_table
3 FOR EACH ROW
4 BEGIN
5     INSERT INTO audit_table (audit_id, changed_id, old_col1, new_col1, old_col2, new_col2, change_time)
6     VALUES (audit_seq.NEXTVAL, :OLD.id, :OLD.col1, :NEW.col1, :OLD.col2, :NEW.col2, SYSTIMESTAMP);
7 END;
8 /
```

Below the code, the 'Results' tab is active, showing the output: "Trigger created." and "0.03 seconds". The bottom status bar indicates the user is '220701043@rajalakshmi.edu.in' and the session is 'bharath'. The copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 23.2.4' are also visible.

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.

QUERY:

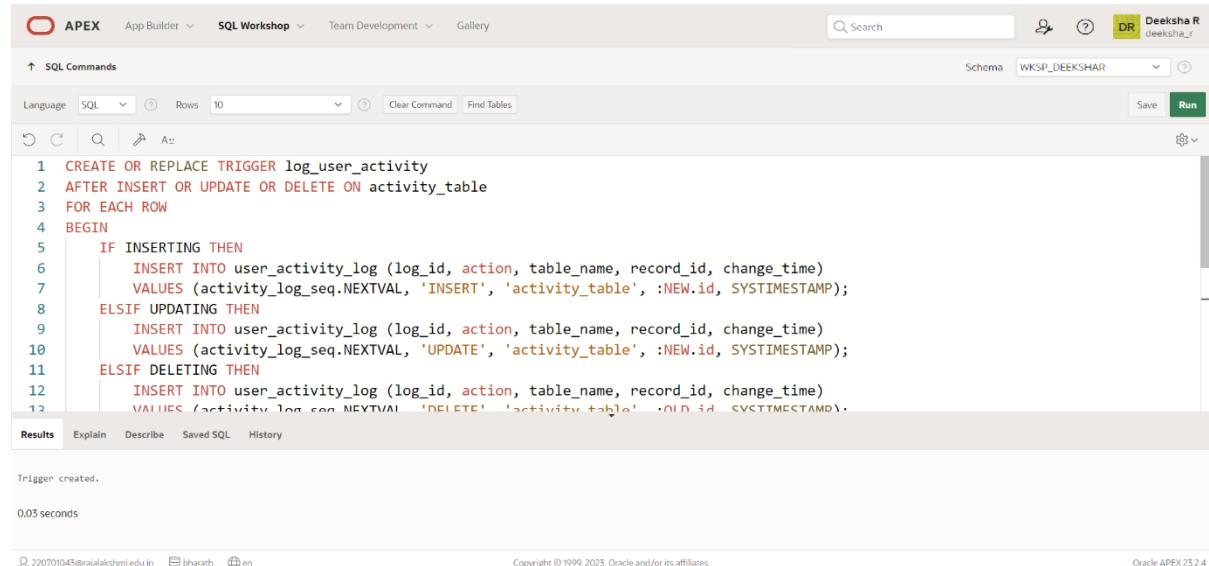
```
CREATE OR REPLACE TRIGGER log_user_activity
AFTER INSERT OR UPDATE OR DELETE ON activity_table
FOR EACH ROW
BEGIN
```

```

IF INSERTING THEN
    INSERT INTO user_activity_log (log_id, action, table_name, record_id,
change_time)
        VALUES (activity_log_seq.NEXTVAL, 'INSERT', 'activity_table',
:NEW.id, SYSTIMESTAMP);
ELSIF UPDATING THEN
    INSERT INTO user_activity_log (log_id, action, table_name, record_id,
change_time)
        VALUES (activity_log_seq.NEXTVAL, 'UPDATE', 'activity_table',
:NEW.id, SYSTIMESTAMP);
ELSIF DELETING THEN
    INSERT INTO user_activity_log (log_id, action, table_name, record_id,
change_time)
        VALUES (activity_log_seq.NEXTVAL, 'DELETE', 'activity_table',
:OLD.id, SYSTIMESTAMP);
END IF;
END;

```

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile 'Deeksha R' and a schema 'WKSP_DEEKSHAR'. The main area is titled 'SQL Commands' with a 'Language' dropdown set to 'SQL'. Below it, there's a toolbar with icons for 'Clear Command' and 'Find Tables'. The code editor contains the PL/SQL trigger definition:

```

1 CREATE OR REPLACE TRIGGER log_user_activity
2 AFTER INSERT OR UPDATE OR DELETE ON activity_table
3 FOR EACH ROW
4 BEGIN
5     IF INSERTING THEN
6         INSERT INTO user_activity_log (log_id, action, table_name, record_id, change_time)
7             VALUES (activity_log_seq.NEXTVAL, 'INSERT', 'activity_table', :NEW.id, SYSTIMESTAMP);
8     ELSIF UPDATING THEN
9         INSERT INTO user_activity_log (log_id, action, table_name, record_id, change_time)
10            VALUES (activity_log_seq.NEXTVAL, 'UPDATE', 'activity_table', :NEW.id, SYSTIMESTAMP);
11     ELSIF DELETING THEN
12         INSERT INTO user_activity_log (log_id, action, table_name, record_id, change_time)
13             VALUES (activity_log_seq.NEXTVAL, 'DELETE', 'activity_table', :OLD.id, SYSTIMESTAMP);

```

The 'Results' tab is selected at the bottom, showing the message 'Trigger created.' and a execution time of '0.03 seconds'. The bottom footer includes copyright information: 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

QUERY:

```

CREATE OR REPLACE TRIGGER update_running_total
BEFORE INSERT ON running_total_table
FOR EACH ROW
DECLARE

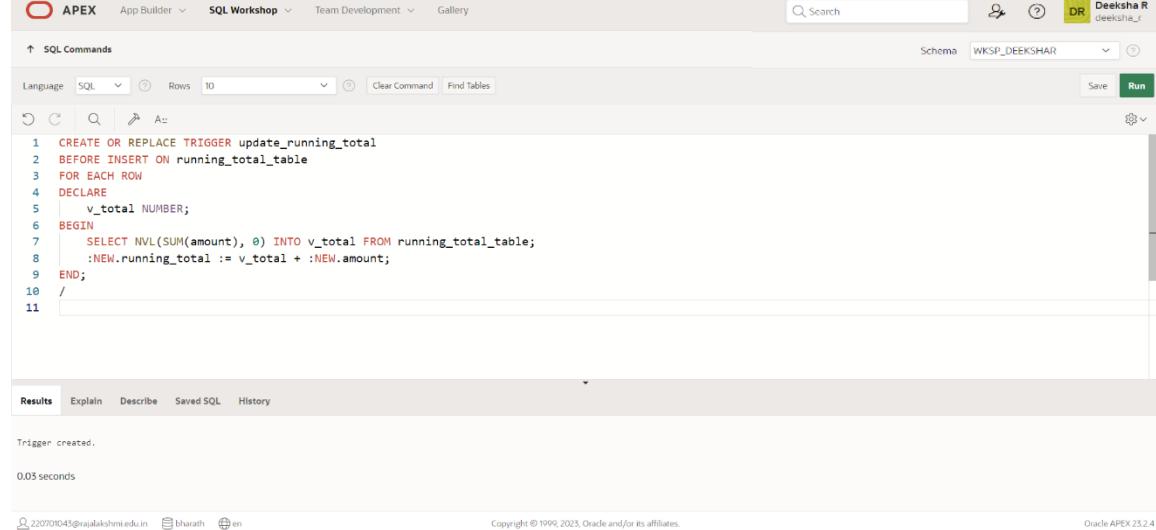
```

```

v_total NUMBER;
BEGIN
    SELECT NVL(SUM(amount), 0) INTO v_total FROM running_total_table;
    :NEW.running_total := v_total + :NEW.amount;
END;

```

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' and 'SQL Workshop' are selected. The main area displays the following PL/SQL code:

```

1 CREATE OR REPLACE TRIGGER update_running_total
2 BEFORE INSERT ON running_total_table
3 FOR EACH ROW
4 DECLARE
5     v_total NUMBER;
6 BEGIN
7     SELECT NVL(SUM(amount), 0) INTO v_total FROM running_total_table;
8     :NEW.running_total := v_total + :NEW.amount;
9 END;
10 /
11

```

Below the code, the 'Results' tab is active, showing the output: "Trigger created." and "0.03 seconds". The bottom right corner of the interface indicates "Oracle APEX 23.2.4".

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

QUERY:

```

CREATE OR REPLACE TRIGGER validate_order
BEFORE INSERT ON orders
FOR EACH ROW
DECLARE
    v_stock NUMBER;
    insufficient_stock EXCEPTION;
    PRAGMA EXCEPTION_INIT(insufficient_stock, -20004);
BEGIN
    SELECT stock_quantity INTO v_stock FROM items WHERE item_id =
    :NEW.item_id;
    IF v_stock < :NEW.order_quantity THEN

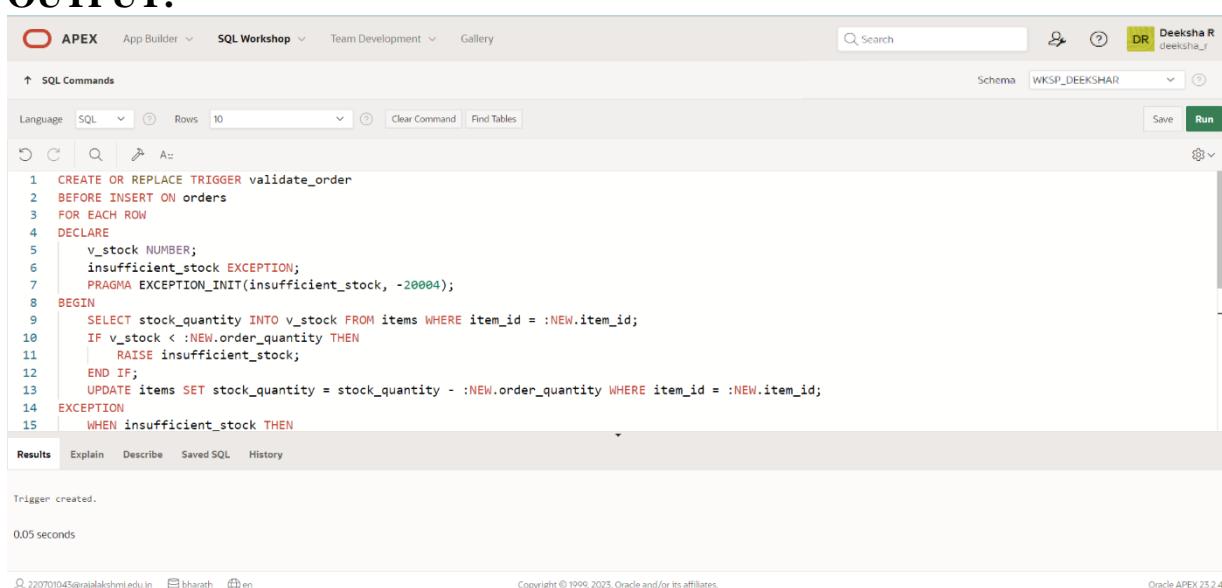
```

```

    RAISE insufficient_stock;
END IF;
UPDATE items SET stock_quantity = stock_quantity - :NEW.order_quantity
WHERE item_id = :NEW.item_id;
EXCEPTION
    WHEN insufficient_stock THEN
        RAISE_APPLICATION_ERROR(-20004, 'Insufficient stock for the
item.');
END;

```

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile 'Deeksha R' and the schema 'WKSP_DEEKSHAR'. The main area is titled 'SQL Commands' with a 'Language' dropdown set to 'SQL'. Below it is a code editor containing the PL/SQL code for the trigger. The code is as follows:

```

1 CREATE OR REPLACE TRIGGER validate_order
2 BEFORE INSERT ON orders
3 FOR EACH ROW
4 DECLARE
5     v_stock NUMBER;
6     insufficient_stock EXCEPTION;
7     PRAGMA EXCEPTION_INIT(insufficient_stock, -20004);
8 BEGIN
9     SELECT stock_quantity INTO v_stock FROM items WHERE item_id = :NEW.item_id;
10    IF v_stock < :NEW.order_quantity THEN
11        RAISE insufficient_stock;
12    END IF;
13    UPDATE items SET stock_quantity = stock_quantity - :NEW.order_quantity WHERE item_id = :NEW.item_id;
14 EXCEPTION
15    WHEN insufficient_stock THEN

```

Below the code editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The results section displays the message 'Trigger created.' and '0.05 seconds'. At the bottom, it shows the URL '220701043@rajalakshmi.edu.in', the session ID 'bharad', and the page number 'en'. The footer indicates 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

RESULT:

MONGO DB

EX_NO: 19

DATE:

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'.

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.restaurants.find( { $or: [ { name: /^Wil/ }, { cuisine: { $nin: ['American', 'Chinese'] } } ] , { restaurant_id: 1, name: 1, borough: 1, cuisine: 1 } } );
{
  _id: ObjectId('664f3c798752f54dc3cdcf7'),
  borough: 'Bronx',
  cuisine: 'Bakery',
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445'
}
c:\deeksha_56> |
```

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.

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.restaurants.find( { grades: { $elemMatch: { grade: "A", score: 11, date: ISODate("2014-08-11T00:00:00Z") } } }, { restaurant_id: 1, name: 1, grades: 1 } );  
c:\deeksha_56> |
```

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".

QUERY:

```
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 } );
```

OUTPUT:

```
c:\deeksha_56> 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 } );  
c:\deeksha_56> |
```

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

QUERY:

```
db.restaurants.find({$and : [{"address.coord.1": {$gt : 42}}, {"address.coord.1": {$lte : 52}}]}, {_id:0, restaurant_id:1, name:1, address:1})
```

OUTPUT:

```
c:\deeksha_56> db.restaurants.find({$and : [{"address.coord.1": {$gt : 42}}, {"address.coord.1": {$lte : 52}}]}, {_id:0, restaurant_id:1, name:1, address:1})  
c:\deeksha_56> |
```

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

QUERY:

```
db.restaurants.find({}, { _id: 0 }).sort({ name: 1 });
```

OUTPUT:

```
c:\deeksha_56> db.restaurants.find({}, { _id: 0 }).sort({ name: 1 })  
[  
  {  
    address: {  
      building: '1007',  
      coord: [ -73.856077, 40.848447 ],  
      street: 'Morris Park Ave',  
      zipcode: '10462'  
    },  
    borough: 'Bronx',  
    cuisine: 'Bakery',  
    grades: [  
      {  
        date: ISODate('2014-03-03T00:00:00.000Z'),  
        grade: 'A',  
        score: 2  
      },  
      {  
        date: ISODate('2013-09-11T00:00:00.000Z'),  
        grade: 'A',  
        score: 6  
      },  
      {  
        date: ISODate('2013-01-24T00:00:00.000Z'),  
        grade: 'A',  
        score: 10  
      },  
      {  
        date: ISODate('2011-11-23T00:00:00.000Z'),  
        grade: 'A',  
        score: 9  
      },  
      {  
        date: ISODate('2011-03-10T00:00:00.000Z'),  
        grade: 'B',  
        score: 14  
      }  
    ],  
    name: 'Morris Park Bake Shop',  
    restaurant_id: '30075445'  
  }  
]
```

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

QUERY:

```
db.restaurants.find({}, { _id: 0 }).sort({ name: -1 })
```

OUTPUT:

```
c:\deeksha_56> db.restaurants.find({}, { _id: 0 }).sort({ name: -1 })
[
  {
    address: {
      building: '1007',
      coord: [ -73.856077, 40.848447 ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: ISODate('2014-03-03T00:00:00.000Z'),
        grade: 'A',
        score: 2
      },
      {
        date: ISODate('2013-09-11T00:00:00.000Z'),
        grade: 'A',
        score: 6
      },
      {
        date: ISODate('2013-01-24T00:00:00.000Z'),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate('2011-11-23T00:00:00.000Z'),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate('2011-03-10T00:00:00.000Z'),
        grade: 'B',
        score: 14
      }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
c:\deeksha_56> |
```

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.

QUERY:

```
db.restaurants.find({}, { _id: 0 }).sort({ cuisine: 1, borough: -1 })
```

OUTPUT:

```
c:\deeksha_56> db.restaurants.find({}, { _id: 0 }).sort({ cuisine: 1, borough: -1 })
[
  {
    address: {
      building: '1007',
      coord: [ -73.856077, 40.848447 ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: ISODate('2014-03-03T00:00:00.000Z'),
        grade: 'A',
        score: 2
      },
      {
        date: ISODate('2013-09-11T00:00:00.000Z'),
        grade: 'A',
        score: 6
      },
      {
        date: ISODate('2013-01-24T00:00:00.000Z'),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate('2011-11-23T00:00:00.000Z'),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate('2011-03-10T00:00:00.000Z'),
        grade: 'B',
        score: 14
      }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
c:\deeksha_56> |
```

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

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.restaurants.find({ "address.street": { $exists: true, $ne: "" } })  
{  
  "_id": ObjectId('664f3c798752f54dc3cdcf7'),  
  "address": {  
    "building": '1007',  
    "coord": [ -73.856077, 40.848447 ],  
    "street": 'Morris Park Ave',  
    "zipcode": '10462'  
  },  
  "borough": 'Bronx',  
  "cuisine": 'Bakery',  
  "grades": [  
    {  
      "date": ISODate('2014-03-03T00:00:00.000Z'),  
      "grade": 'A',  
      "score": 2  
    },  
    {  
      "date": ISODate('2013-09-11T00:00:00.000Z'),  
      "grade": 'A',  
      "score": 6  
    },  
    {  
      "date": ISODate('2013-01-24T00:00:00.000Z'),  
      "grade": 'A',  
      "score": 10  
    },  
    {  
      "date": ISODate('2011-11-23T00:00:00.000Z'),  
      "grade": 'A',  
      "score": 9  
    },  
    {  
      "date": ISODate('2011-03-10T00:00:00.000Z'),  
      "grade": 'B',  
      "score": 14  
    }  
  ],  
  "name": 'Morris Park Bake Shop',  
  "restaurant_id": '30075445'  
}  
c:\deeksha_56> |
```

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

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.restaurants.find({ "address.coord": { $elemMatch: { $type: "double" } } })  
{  
  "_id": ObjectId('664f3c798752f54dc3cdcf7'),  
  "address": {  
    "building": '1007',  
    "coord": [ -73.856077, 40.848447 ],  
    "street": 'Morris Park Ave',  
    "zipcode": '10462'  
  },  
  "borough": 'Bronx',  
  "cuisine": 'Bakery',  
  "grades": [  
    {  
      "date": ISODate('2014-03-03T00:00:00.000Z'),  
      "grade": 'A',  
      "score": 2  
    },  
    {  
      "date": ISODate('2013-09-11T00:00:00.000Z'),  
      "grade": 'A',  
      "score": 6  
    },  
    {  
      "date": ISODate('2013-01-24T00:00:00.000Z'),  
      "grade": 'A',  
      "score": 10  
    },  
    {  
      "date": ISODate('2011-11-23T00:00:00.000Z'),  
      "grade": 'A',  
      "score": 9  
    },  
    {  
      "date": ISODate('2011-03-10T00:00:00.000Z'),  
      "grade": 'B',  
      "score": 14  
    }  
  ],  
  "name": 'Morris Park Bake Shop',  
  "restaurant_id": '30075445'  
}  
c:\deeksha_56> |
```

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.

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.restaurants.find({ "grades.score": { $mod: [7, 0] } }, { restaurant_id: 1, name: 1, grades: 1 })  
{  
  _id: ObjectId('664f3c798752f54dc3cdcdf7'),  
  grades: [  
    {  
      date: ISODate('2014-03-03T00:00:00.000Z'),  
      grade: 'A',  
      score: 2  
    },  
    {  
      date: ISODate('2013-09-11T00:00:00.000Z'),  
      grade: 'A',  
      score: 6  
    },  
    {  
      date: ISODate('2013-01-24T00:00:00.000Z'),  
      grade: 'A',  
      score: 10  
    },  
    {  
      date: ISODate('2011-11-23T00:00:00.000Z'),  
      grade: 'A',  
      score: 9  
    },  
    {  
      date: ISODate('2011-03-10T00:00:00.000Z'),  
      grade: 'B',  
      score: 14  
    }  
  ],  
  name: 'Morris Park Bake Shop',  
  restaurant_id: '30075445'  
}  
c:\deeksha_56> |
```

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.

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.restaurants.find({ name: /mon/i }, { name: 1, borough: 1, "address.coord": 1, cuisine: 1 })  
c:\deeksha_56> |
```

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.

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.restaurants.find({ name: /^Mad/i }, { name: 1, borough: 1, "address.coord": 1, cuisine: 1 })  
c:\deeksha_56> |
```

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

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } } })  
{  
  "_id": ObjectId('664f3c798752f54dc3cdcf7'),  
  "address": {  
    "building": "1007",  
    "coord": [ -73.856077, 40.848447 ],  
    "street": "Morris Park Ave",  
    "zipcode": "10462"  
  },  
  "borough": "Bronx",  
  "cuisine": "Bakery",  
  "grades": [  
    {  
      "date": ISODate('2014-03-03T00:00:00.000Z'),  
      "grade": "A",  
      "score": 2  
    },  
    {  
      "date": ISODate('2013-09-11T00:00:00.000Z'),  
      "grade": "A",  
      "score": 6  
    },  
    {  
      "date": ISODate('2013-01-24T00:00:00.000Z'),  
      "grade": "A",  
      "score": 10  
    },  
    {  
      "date": ISODate('2011-11-23T00:00:00.000Z'),  
      "grade": "A",  
      "score": 9  
    },  
    {  
      "date": ISODate('2011-03-10T00:00:00.000Z'),  
      "grade": "B",  
      "score": 14  
    }  
  ],  
  "name": "Morris Park Bake Shop",  
  "restaurant_id": "30075445"  
}  
c:\deeksha_56> |
```

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.

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, "borough": "Manhattan" })  
c:\deeksha_56> |
```

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.

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }] })
```

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.

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $ne: "American" } })
```

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.

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $nin: ["American", "Chinese"] } })
```

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.

QUERY:

```
db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }] })
```

OUTPUT:

```
c:\deeksha_56> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }] })
```

```
{ _id: ObjectId('664f3c798752f54dc3cdcf7'),
  address: {
    building: '1007',
    coord: [ -73.856077, 40.848447 ],
    street: 'Morris Park Ave',
    zipcode: '10462'
  },
  borough: 'Bronx',
  cuisine: 'Bakery',
  grades: [
    {
      date: ISODate('2014-03-03T00:00:00.000Z'),
      grade: 'A',
      score: 2
    },
    {
      date: ISODate('2013-09-11T00:00:00.000Z'),
      grade: 'A',
      score: 6
    },
    {
      date: ISODate('2013-01-24T00:00:00.000Z'),
      grade: 'A',
      score: 10
    },
    {
      date: ISODate('2011-11-23T00:00:00.000Z'),
      grade: 'A',
      score: 9
    },
    {
      date: ISODate('2011-03-10T00:00:00.000Z'),
      grade: 'B',
      score: 14
    }
  ],
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445'
}
```

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.

QUERY:

```
db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], "borough": "Manhattan" })
```

OUTPUT:

```
c:\deeksha_56> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], "borough": "Manhattan" })  
c:\deeksha_56> |
```

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.

QUERY:

```
db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }] })
```

OUTPUT:

```
c:\deeksha_56> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }] })  
c:\deeksha_56> |
```

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.

QUERY:

```
db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $ne: "American" } })
```

OUTPUT:

```
c:\deeksha_56> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $ne: "American" } })
```

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.

QUERY:

```
db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $nin: ["American", "Chinese"] } })
```

OUTPUT:

```
c:\deeksha_56> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $nin: ["American", "Chinese"] } })
```

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.

QUERY:

```
db.restaurants.find({ $or: [{ "grades.score": 2 }, { "grades.score": 6 }] })
```

OUTPUT:

```
c:\deeksha_56> db.restaurants.find({ $or: [{ "grades.score": 2 }, { "grades.score": 6 }] })
[ {
  _id: ObjectId('664f3c798752f54dc3cdcdf7'),
  address: {
    building: '1007',
    coord: [ -73.856077, 40.848447 ],
    street: 'Morris Park Ave',
    zipcode: '10462'
  },
  borough: 'Bronx',
  cuisine: 'Bakery',
  grades: [
    {
      date: ISODate('2014-03-03T00:00:00.000Z'),
      grade: 'A',
      score: 2
    },
    {
      date: ISODate('2013-09-11T00:00:00.000Z'),
      grade: 'A',
      score: 6
    },
    {
      date: ISODate('2013-01-24T00:00:00.000Z'),
      grade: 'A',
      score: 10
    },
    {
      date: ISODate('2011-11-23T00:00:00.000Z'),
      grade: 'A',
      score: 9
    },
    {
      date: ISODate('2011-03-10T00:00:00.000Z'),
      grade: 'B',
      score: 14
    }
  ],
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445'
}
]c:\deeksha_56> |
```

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

RESULT:

MONGO DB

EX_NO: 20

DATE:

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

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.movies.find({ year: 1893 })
c:\deeksha_56>
```

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

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.movies.find({ runtime: { $gt: 120 } })
c:\deeksha_56> |
```

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

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.movies.find({ genres: 'Short' })

{
  _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/MV5BMTU3NjE5NzYtYTYYNS00MDVmLWIwYjgtMmYwYWIxZDYyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._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')
  }
}

c:\deeksha_56> |
```

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

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.movies.find({ directors: 'William K.L. Dickson' })
c:\deeksha_56> |
```

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

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.movies.find({ countries: 'USA' })

{
  _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/MV5BMTU3NjE5NzYtYTYYNS00MDVmLWIwYjgtMmYwYWIxZDYyNzU2XkEyXkFqcGdeQXVyNzQzQxNzI@._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')
  }
}

c:\deeksha_56|
```

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

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.movies.find({ rated: 'UNRATED' })
c:\deeksha_56|
```

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

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.movies.find({ 'imdb.votes': { $gt: 1000 } })
{
  _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/MV5BMTU3NjE5NzYtYTYYNS00MDVmLWIwYjgtMmYwYWIxZDYyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._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')
  }
}
c:\deeksha_56> |
```

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

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.movies.find({ 'imdb.rating': { $gt: 7 } })
{
  _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/MV5BMTU3NjE5NzYtYTYYNS00MDVmLWIwYjgtMmYwYWIxZDYyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._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')
  }
}
c:\deeksha_56> |
```

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

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.movies.find({ 'tomatoes.viewer.rating': { $gt: 4 } })  
c:\deeksha_56> |
```

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

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.movies.find({ 'awards.wins': { $gt: 0 } })  
{  
  _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/MV5BMTU3NjE5NzYtYTYYNS00MDVmLwIwYjgtMmYwYWIxZDYyNzU2XkEyXkFqcGdeQXVyNzQzNzQzNzI@._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')  
  }  
}  
c:\deeksha_56> |
```

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.

QUERY:

```
db.movies.find( { 'awards.nominations': { $gt: 0 } }, { title: 1, languages: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 } )
```

OUTPUT:

```
c:\deeksha_56> db.movies.find(
..   { 'awards.nominations': { $gt: 0 } },
..   { title: 1, languages: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 }
.. )
c:\deeksha_56> |
```

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".

QUERY:

```
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 } )
```

OUTPUT:

```
c:\deeksha_56> 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 }
.. )
c:\deeksha_56> |
```

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

QUERY:

```
db.movies.find( { released: ISODate("1893-05-09T00:00:00.000Z") }, { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 } )
```

OUTPUT:

```
c:\deeksha_56> db.movies.find(
..   { released: ISODate("1893-05-09T00:00:00.000Z") },
..   { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 }
.. )
c:\deeksha_56> |
```

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.

QUERY:

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

OUTPUT:

```
c:\deeksha_56> db.movies.find(
..   { title: /scene/i },
..   { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 }
.. )
c:\deeksha_56> |
```

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

RESULT: