

Creating and Managing Tables

EX_NO:1

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 dept(id int, name varchar(25));
```

OUTPUT:

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

```
1 create table department(
2   id int,
3   name varchar(25)
4 );
```

The 'Run' button is highlighted in green at the top right of the editor. Below the editor, the Results tab is selected, showing the output:

Table created.
0.03 seconds

At the bottom, the footer includes user information (220701055@rajalakshmi.edu.in, dash04, en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and version (Oracle APEX 23.2.4).

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 int, Last_Name varchar(25),First_Name varchar(25),Dept_id int);
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, the tabs 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery' are visible. On the right side, the user 'darshita m' and schema 'WKSP_DARSH04' are displayed. The main area is titled 'SQL Commands'. It contains a code editor with the following SQL command:

```
1 create table emp(
2   id int,
3   last_name varchar(25),
4   first_name varchar(25),
5   dept_id int
6 );
```

Below the code editor, there are buttons for 'Clear Command', 'Find Tables', 'Save', and 'Run'. The status bar at the bottom shows the user's email '220701055@rajalakshmi.edu.in', the session ID 'darsh04', and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' The footer also indicates 'Oracle APEX 23.2.4'.

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 varchar(50));
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'darshita m darsh04', and a schema dropdown set to 'WKSP_DARSH04'. The main workspace is titled 'SQL Commands' and contains a command line with the following text:

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

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

```
Table altered.
```

Execution details at the bottom indicate it took 0.05 seconds.

Page footer information includes the URL '22070105@rajalakshmi.edu.in', session ID 'darsh04', and language 'en'. Copyright notice: 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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 int not null ,first_name varchar(20),Last_name varchar(25),Salary int,Dept_id int);

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected, along with 'SQL Workshop'. The schema dropdown shows 'WKSP_DARSH04'. The main area displays the SQL command for creating the 'employees2' table:

```
1 create table employees2(
2     id int not null,
3     first_name varchar(20),
4     last_name varchar(25) not null,
5     salary int,
6     dept_id int
7 );
```

Below the code, the 'Results' tab is active, showing the output: 'Table created.' and '0.05 seconds'. The bottom footer includes user information (220701055@rajalakshmi.edu.in, darsh04, en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the version (Oracle APEX 23.2.4).

5.Drop the EMP table.

QUERY:

```
Drop table emp;
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area contains a SQL command: 'drop table emp;'. Below the command, the 'Results' tab is active, showing the output: 'Table dropped.' and '0.09 seconds'. The bottom status bar displays session information: '220701055@rajalakshmi.edu.in', 'darsh04', and 'en'. The copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 25.2.4' are also visible.

6.Rename the EMPLOYEES2 table as EMP.

QUERY:

Rename employees2 to emp;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The top right corner shows the user's name 'darshita m' and schema 'darsh04'. The main workspace is titled 'SQL Commands' and contains the following SQL statement:

```
1  rename employees2 to emp;
```

Below the statement, the 'Results' tab is selected, showing the output of the executed command:

Statement processed.
0.05 seconds

At the bottom of the page, there are footer links for 'Copyright © 1999, 2023, Oracle and/or its affiliates.', 'Oracle APEX 23.2.4', and user information '220701055@rajalakshmi.edu.in', 'darsh04', and 'en'.

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

QUERY:

comment on table dept is 'Department info'; comment
on table emp is Employee info';

OUTPUT:

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

```
1 comment on table dept is'Department info';
2 comment on table emp is'employee info';
```

Below the code, the 'Results' tab is active. The output shows:

Statement processed.
0.03 seconds

At the bottom of the page, there are footer links: 220701055@rajalakshmi.edu.in, darsh04, en, Copyright © 1999, 2023, Oracle and/or its affiliates., and Oracle APEX 23.2.4.

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

```
1 alter table emp drop column first_name;
```

Below the command, the results are displayed:

```
Table altered.
```

Execution time: 0.05 seconds

At the bottom of the interface, there are user profile icons and copyright information: Copyright © 1999, 2025, Oracle and/or its affiliates. Oracle APEX 25.2.4

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

RESULT: Thus sql queries on creating and managing tables are executed and verified successfully.

MANIPULATING DATA

EX_NO:2

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 int not null, last_name varchar(25), first_name varchar(25), userid varchar(25), salary int);

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, a SQL command is entered to create the MY_EMPLOYEE table with the specified columns and constraints. The command is:

```
1 create table my_employee(
2     id int not null,
3     last_name varchar(25),
4     first_name varchar(25),
5     userid varchar(25),
6     salary int
7 );
```

After executing the command, the Results tab displays the output: "Table created." and "0.04 seconds". The status bar at the bottom right indicates "Activate Windows Go to Settings to activate Windows".

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 | Dances | Betty | bdances | 860 |
| 3 | Biri | Ben | bbiri | 1100 |
| 4 | Newman | Chad | Cnewman | 750 |
| 5 | Ropebur | Audrey | aropebur | 1550 |

QUERY: insert into my_employee

values(1,'patel','ralph','rpatel',895); Insert into my_employee
values(2,'dances','betty',bdances',860);

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, a single line of SQL code is entered:

```
1 insert into my_employee values (2,'Dances','Betty','bdances',860);
```

In the Results pane, the output is displayed:

1 row(s) inserted.
0.02 seconds

At the bottom right of the interface, there is a watermark: "Activate Windows Go to Settings to activate Windows".

3.Display the table with values.

QUERY:

```
select*from my_employee;
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side of the header shows the schema 'WKSP_DARSH04' and a user icon for 'darshita'. The main workspace has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab shows the query 'select* from my_employee;'. The Results tab displays the following table:

| ID | LAST_NAME | FIRST_NAME | USERID | SALARY |
|----|-----------|------------|--------|--------|
| 1 | Patel | Ralph | rpatel | 895 |
| 2 | Danes | Betty | bdanes | 860 |

Below the table, it says '2 rows returned in 0.02 seconds' and provides a 'Download' link. The bottom of the page includes copyright information for Oracle and a link to activate Windows.

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

QUERY:

```
insert into my_employee values(3,'biri','ben','bbiri',1100);
Insert into my_employee values(4,'newman','chad',cnewman',750);
```

OUTPUT:

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

```
1 insert into my_employee values(4,'Newman','Chad','Cnewman',750);
```

Below the editor, the 'Results' tab is active. The output shows:

1 row(s) inserted.
0.00 seconds

In the bottom right corner, there is a watermark: "Activate Windows Go to Settings to activate Windows." and the text "Copyright © 1999, 2023, Oracle and/or its affiliates." and "Oracle APEX 23.2.4".

5. Make the data additions permanent.

QUERY:

Select* from my_employee;

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 top right corner shows the user's name 'darshita m' and workspace 'darsh04'. The main area has tabs for SQL Commands and Results. The SQL Commands tab shows the query 'select* from my_employee;'. The Results tab displays a table with four rows of data:

| ID | LAST_NAME | FIRST_NAME | USERID | SALARY |
|----|-----------|------------|---------|--------|
| 1 | Patel | Ralph | rpatel | 895 |
| 2 | Danes | Betty | bdares | 860 |
| 3 | Birj | Ben | bbirj | 1100 |
| 4 | Newman | Chad | Cnewman | 750 |

Below the table, there is a note: 'Activate Windows' with a link 'Go to Settings to activate Windows.' The bottom of the screen shows the user's email '22070105@rajkashmi.edu.in', the workspace 'darsh04', and the session status 'en'. The footer includes copyright information 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.0'.

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

QUERY:

```
Update my_employee set last_name='Drexler' where last_name='Biri';
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. At the top, there are tabs for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, it shows the schema as WKSP_DARSH04 and the user as darsih04. Below the tabs, there's a search bar and some icons. The main area has tabs for SQL Commands, SQL (selected), and PL/SQL. Under SQL, the language is set to SQL, rows are set to 10, and there are buttons for Clear Command and Find Tables. The command entered is: `select* from my_employee;`. In the results section, there's a table with columns ID, LAST_NAME, FIRST_NAME, USERID, and SALARY. The data is as follows:

| ID | LAST_NAME | FIRST_NAME | USERID | SALARY |
|----|-----------|------------|---------|---|
| 1 | Patel | Ralph | rpatel | 895 |
| 2 | Danes | Betty | bdanes | 860 |
| 3 | Drexler | Ben | bbiri | 1100 Activate Windows |
| 4 | Newman | Chad | Cnewman | 750 Go to Settings to activate Windows. |

At the bottom left, it shows the session details: 220701055@rajaletakshri.edu.in, darsih04, en. At the bottom right, it says Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4.

7.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 APEX SQL Workshop interface. At the top, there are tabs for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there are buttons for Search, Help, and a user icon, along with the schema information "Schema: WKSP_DARSH04" and "User: dashita m".

In the SQL Commands section, the language is set to SQL, rows are set to 10, and the command is "Clear Command". The SQL code entered is:

```
1 update my_employee set salary='1000' where salary<900;
2 select* from my_employee;
```

Below the code, the Results tab is selected. The table contains four rows of employee data:

| Employee ID | Last Name | First Name | User Name | Salary |
|-------------|-----------|------------|-----------|--------|
| 1 | Patel | Ralph | rpatel | 1000 |
| 2 | Danes | Betty | bdanes | 1000 |
| 3 | Drexler | Ben | bbiri | 1100 |
| 4 | Newman | Chad | Cnewman | 1000 |

At the bottom of the results table, there is a note: "Activate Windows" with a link "Go to Settings to activate Windows".

At the very bottom of the page, there is footer text: "Copyright © 1999, 2023, Oracle and/or its affiliates." and "Oracle APEX 23.2.4".

8.Delete Betty from MY_EMPLOYEE table.

QUERY:

Delete from my_employee where first_name='Betty';

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 top right corner shows the user's name 'darshita m' and workspace 'WKS_DARSH04'. The main area is titled 'SQL Commands' with tabs for Language (set to SQL), Rows (set to 10), and various icons for Clear Command, Find Tables, Save, and Run. The SQL editor contains the following commands:

```
1 delete from my_employee where first_name='Betty';
2 select* from my_employee;
```

Below the SQL editor is a results grid. The grid has columns: ID, LAST_NAME, FIRST_NAME, USERID, and SALARY. The data rows are:

| ID | LAST_NAME | FIRST_NAME | USERID | SALARY |
|----|-----------|------------|---------|--|
| 1 | Patel | Ralph | rpatel | 1000 |
| 5 | Drexler | Ben | bbiri | 1100 |
| 4 | Newman | Chad | Cnewman | 1000 Activate Windows Go to Settings to activate Windows. |

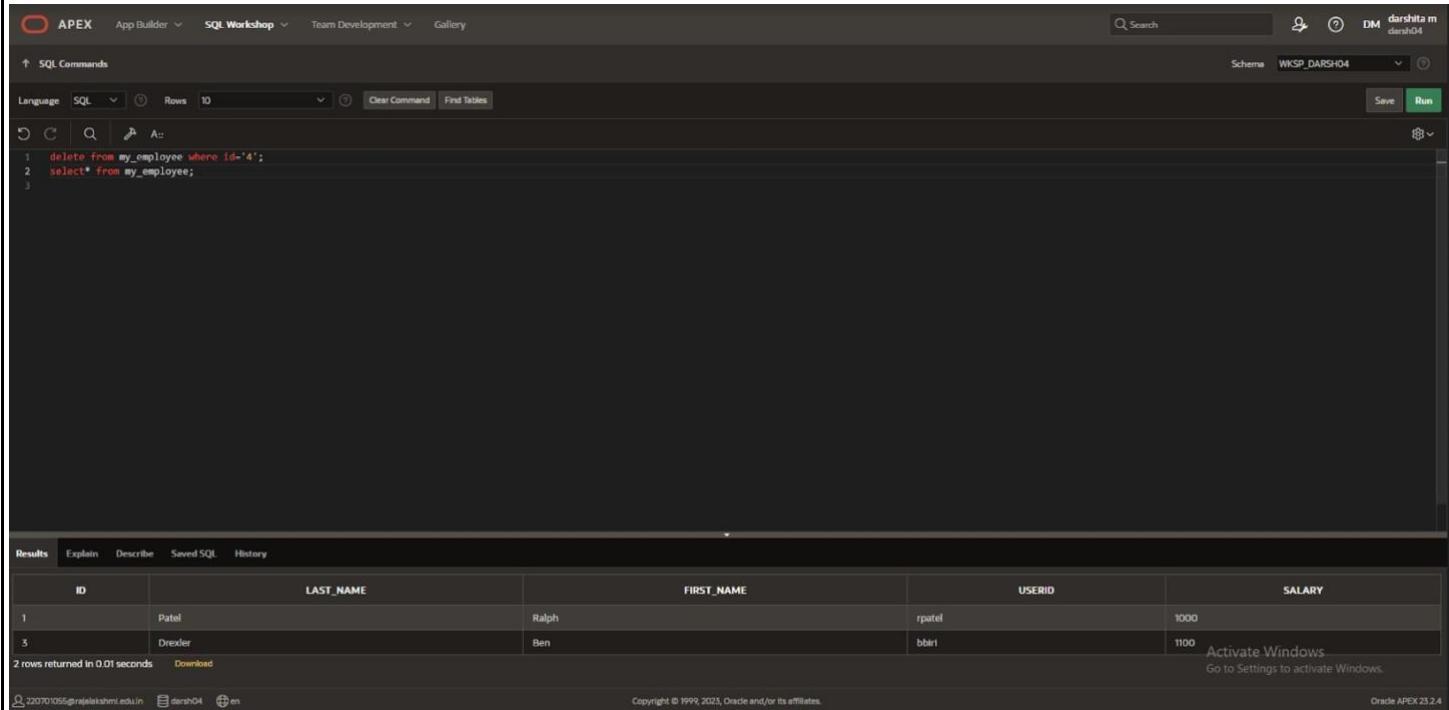
At the bottom of the results grid, it says '3 rows returned in 0.00 seconds' and provides a 'Download' link. The footer of the page includes copyright information: 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.0'.

9.Empty the fourth row of the emp table.

QUERY:

Delete from my_employee where id='4';

OUTPUT:



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

```
1 delete from my_employee where id='4';
2 select* from my_employee;
3
```

Below the code, the results section displays a table with two rows of data:

| ID | LAST_NAME | FIRST_NAME | USERID | SALARY |
|----|-----------|------------|--------|--------|
| 1 | Patel | Ralph | rpatel | 1000 |
| 5 | Drexler | Ben | bbiri | 1100 |

Text at the bottom of the results pane: "Activate Windows
Go to Settings to activate Windows".

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

RESULT: Thus sql queries on manipulating data is executed and verified successfully.

INCLUDING CONSTRAINTS

EX_NO:3

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:

```
create table emp2(id number(6), l_name varchar(25) not null, email varchar(25), salary number(8,2),  
constraint  
my_emp_id_pk primary key(id));
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands tab, a SQL script is being run. The script creates a table 'emp2' with columns 'id', 'l_name', 'email', and 'salary'. A primary key constraint 'my_emp_id_pk' is defined on the 'id' column. The output pane shows the command executed and the message 'Table created.' indicating successful execution.

```
1 create table emp2(  
2     id number(6),  
3     l_name varchar(25) not null,  
4     email varchar(25),  
5     salary number(8,2),  
6     constraint my_emp_id_pk PRIMARY KEY(id)  
7 );
```

Results Explain Describe Saved SQL History

Table created.
0.05 seconds

Activate Windows
Go to Settings to activate Windows.

Copyright © 1999, 2023, Oracle and/or its affiliates.
Oracle APEX 23.2.4

2.Create a PRIMARY 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 dept2(id number(6), l_name varchar(25) not null, email varchar(25), constraint my_dept_id_pk primary key(id));

OUTPUT:

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

```
1 create table dept2(
2     id number(6),
3     l_name varchar(25) not null,
4     email varchar(25),
5     constraint my_dept_id_pk PRIMARY KEY(id)
6 );
```

Below the code, the 'Results' tab is active, showing the output: 'Table created.' and '0.05 seconds'. At the bottom right, there is an 'Activate Windows' message: 'Activate Windows Go to Settings to activate Windows.'

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 emp2 add dept_id number(6); alter table emp2 add constraint my_emp_dept_id_fk foreign key(dept_id) references emp2(id);
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area contains the SQL command: 'alter table emp2 add dept_id number(6);'. The 'Run' button is visible at the bottom right of the command input field.

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area contains the SQL command: '1 alter table emp2 add constraint my_emp_dept_id_fk foreign key(dept_id) references emp2(id);'. The 'Run' button is visible at the bottom right of the command input field.

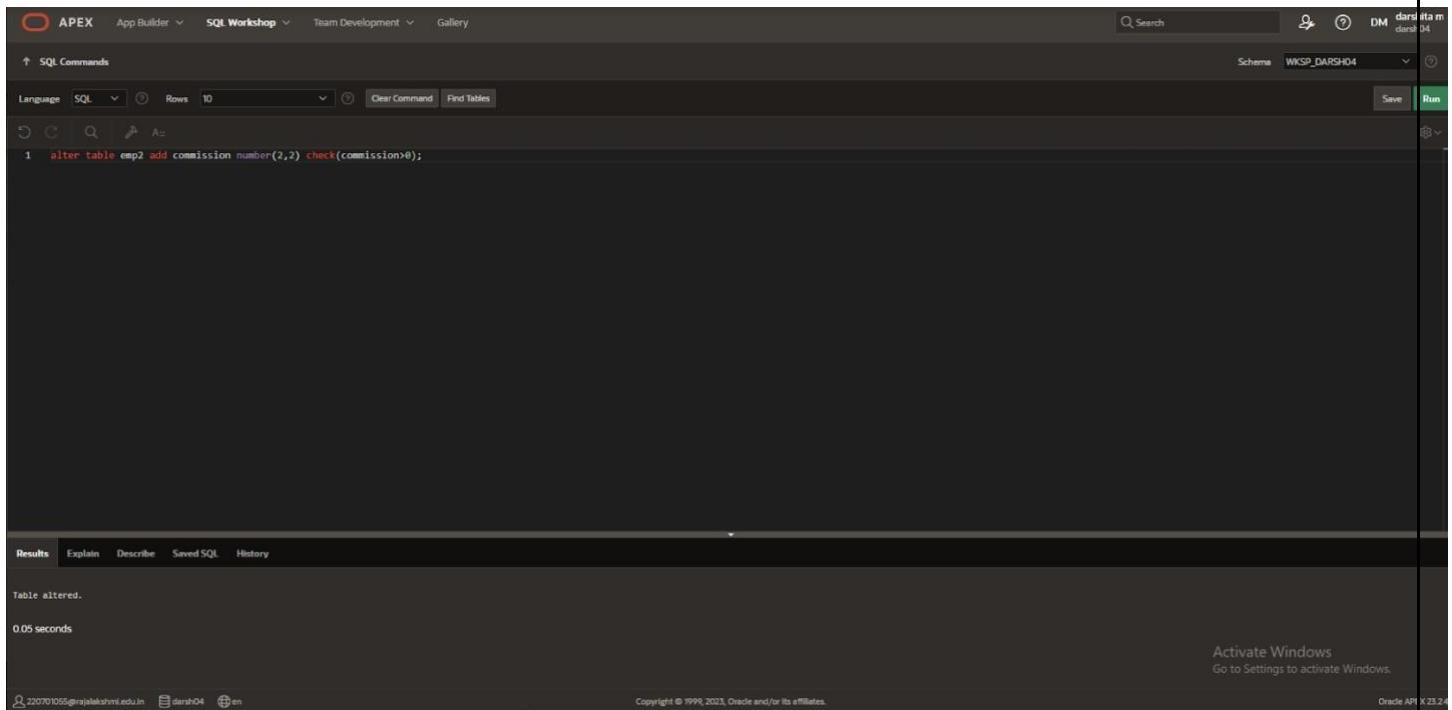
The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area displays the result of the executed command: 'Table altered.' Below the results, it shows '0.06 seconds' and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.'

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 emp2 add commission number(2,2) check (commission>0);

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area contains the SQL command: 'alter table emp2 add commission number(2,2) check (commission>0);'. Below the command, the results tab is active, showing the output: 'Table altered.' and '0.05 seconds'. The bottom right corner of the interface has an 'Activate Windows' watermark.

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

RESULT: Thus the sql queries on including constraints are successfully executed.

Writing Basic SQL SELECT Statements

EX_NO:4

DATE:

-
- 1.The following statement executes successfully.

Identify the Errors

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

QUERY:

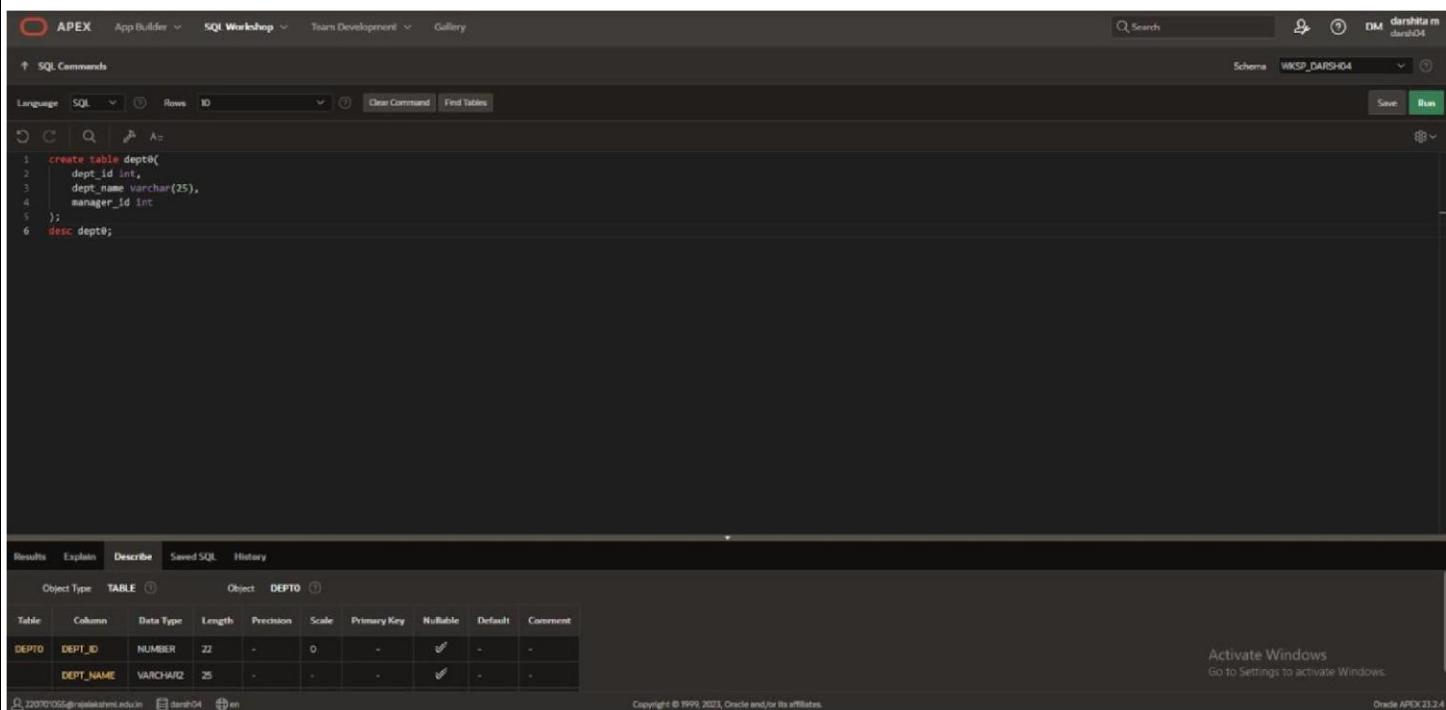
```
SELECT employee_id, last_name  
salary*12 as ANNUAL_SALARY  
FROM employees;
```

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

QUERY:

```
Desc dept0;  
select* from  
dept0;
```

OUTPUT:



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

```
create table dept0(  
dept_id int,  
dept_name varchar(25),  
manager_id int  
);  
desc dept0;
```

In the Results pane, the 'Describe' tab is selected, showing the structure of the DEPT0 table:

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|-------|-----------|-----------|--------|-----------|-------|-------------|----------|---------|---------|
| DEPT0 | DEPT_ID | NUMBER | 22 | - | 0 | + | ✓ | + | - |
| DEPT0 | DEPT_NAME | VARCHAR2 | 25 | - | - | + | ✓ | + | - |

At the bottom right of the Results pane, there is a message: "Activate Windows Go to Settings to activate Windows."

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 employee_number, last_name, job_code, hire_date frpm employee0;
```

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 user is logged in as 'darshtam' with schema 'WKSP_DARSH04'. The SQL Commands tab is selected, showing the query 'select * from employee0;'. The results tab displays the output of the query:

| EMP_NUMBER | LAST_NAME | JOB_CODE | HIRE_DATE |
|------------|-----------|----------|------------|
| 1 | aaa | 123 | 01/03/2019 |
| 2 | bbb | 456 | 11/11/2021 |

Below the table, it says '2 rows returned in 0.01 seconds' and provides a 'Download' link. The bottom of the page includes copyright information for Oracle and a link to activate Windows.

4. Provide an alias STARTDATE for the hire date.

QUERY:

Select hire_date as startdate from employee0;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. At the top, there are tabs for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The SQL Workshop tab has sub-options for SQL Commands, SQL, Rows (set to 10), Clear Command, Find Tables, Save, and Run. The schema is set to WKSP_DARSH04. The main area contains the following SQL code:

```
1 alter table employee0
2 rename column hire_date to startdate;
3 select* from employee0;
```

Below the code, the Results tab is selected, showing the output of the query:

| EMP_NUMBER | LAST_NAME | JOB_CODE | STARTDATE |
|------------|-----------|----------|------------|
| 1 | aaa | 123 | 01/03/2019 |
| 2 | bbb | 456 | 11/11/2021 |

Below the table, it says "2 rows returned in 0.01 seconds" and provides a "Download" link. The bottom of the page includes copyright information and links to Oracle settings and APEX version details.

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 and TITLE" from employee0;
```

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 schema is set to WKSP_DARSH04. The SQL command window contains the following code:

```
select last_name||', '||job_code as "EMPLOYEE and TITLE" from employee0;
```

The results section displays the output:

| EMPLOYEE and TITLE |
|--------------------|
| aaa, IZS |
| bbb, 456 |

Below the results, it says "2 rows returned in 0.01 seconds". There are "Download" and "Activate Windows" buttons. The bottom footer includes copyright information and a link to Oracle APEX 23.2.4.

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 emp_number||','|| lat_name||','||job_code||','||startdate as "THE_OUTPUT" from employee0;

OUTPUT:

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

```
1 select emp_number||','||last_name||','||job_code||','||startdate as "THE_OUTPUT" from employee0;
```

The results section displays the output:

| THE_OUTPUT |
|----------------------|
| 1,aaa,23.01.03/2019 |
| 2,bbb,456,11/11/2021 |

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

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

RESULT: Thus writing basic sql select statements are executed successfully.

RESTRICTING AND SORTING DATA

EX_NO:5

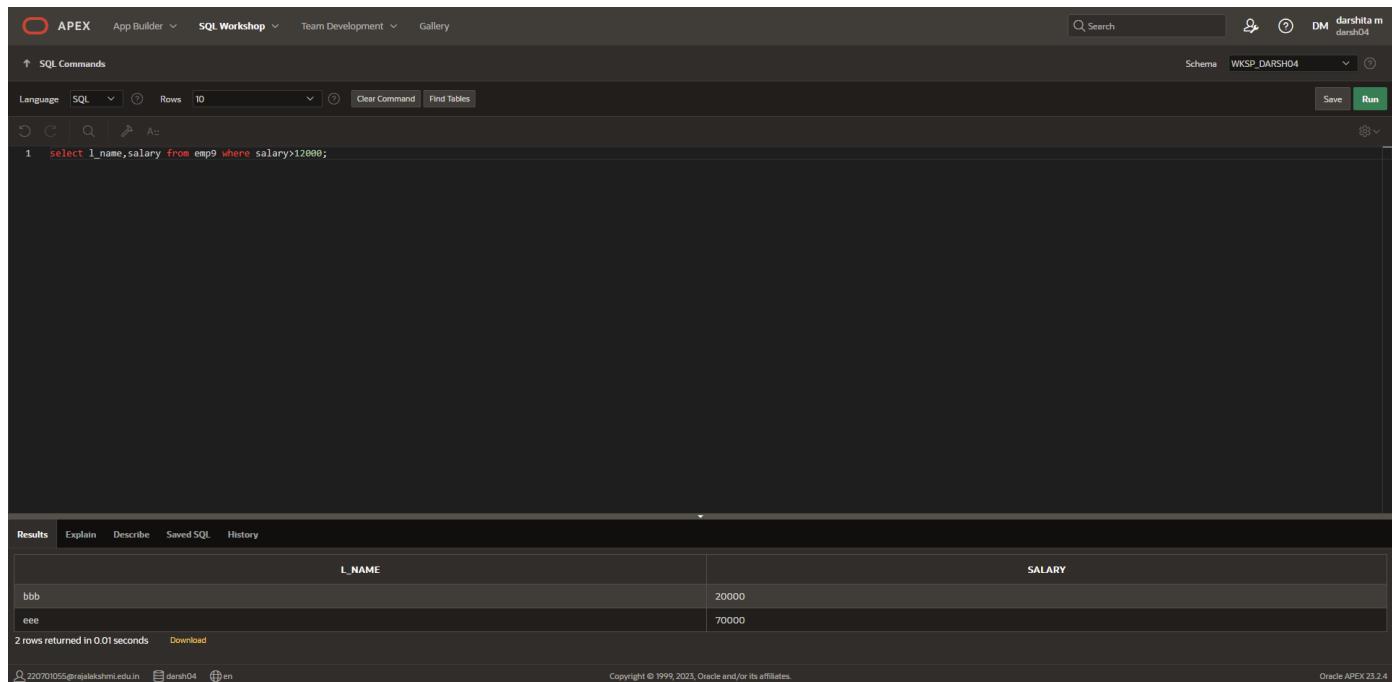
DATE:

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

QUERY:

Select l_name,salary from emp9 where salary>12000;

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 schema dropdown shows 'WKSP_DARSH04'. The main area contains the following SQL command:

```
1 select l_name,salary from emp9 where salary>12000;
```

The results section displays the output of the query:

| L_NAME | SALARY |
|--------|--------|
| bbb | 20000 |
| eee | 70000 |

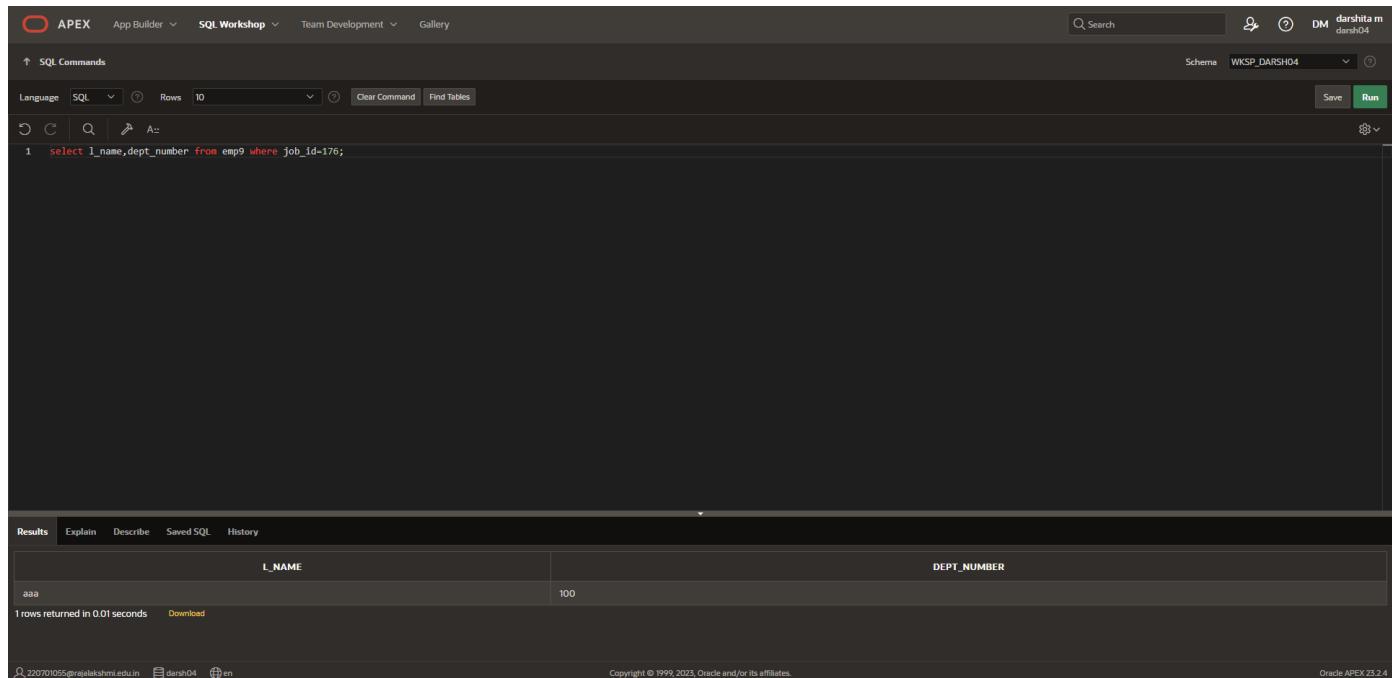
Below the table, it says "2 rows returned in 0.01 seconds". The bottom of the screen shows copyright information: "Copyright © 1999, 2023, Oracle and/or its affiliates." and "Oracle APEX 23.2.4".

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

QUERY:

Select l_name,dept_number from emp9 where job_id=176;

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The top right corner shows the user's name 'darshita m' and workspace 'darsh04'. The main area is titled 'SQL Commands' with tabs for Language (set to SQL), Rows (set to 10), and buttons for Clear Command and Find Tables. A green 'Run' button is visible on the right. The SQL command entered is: 'select l_name,dept_number from emp9 where job_id=176;'. Below the command is a results grid. The first row contains the column headers 'L_NAME' and 'DEPT_NUMBER'. The data row shows 'aaa' in the L_NAME column and '100' in the DEPT_NUMBER column. A note at the bottom left says '1 rows returned in 0.01 seconds'. The bottom right corner displays the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 23.2.4'.

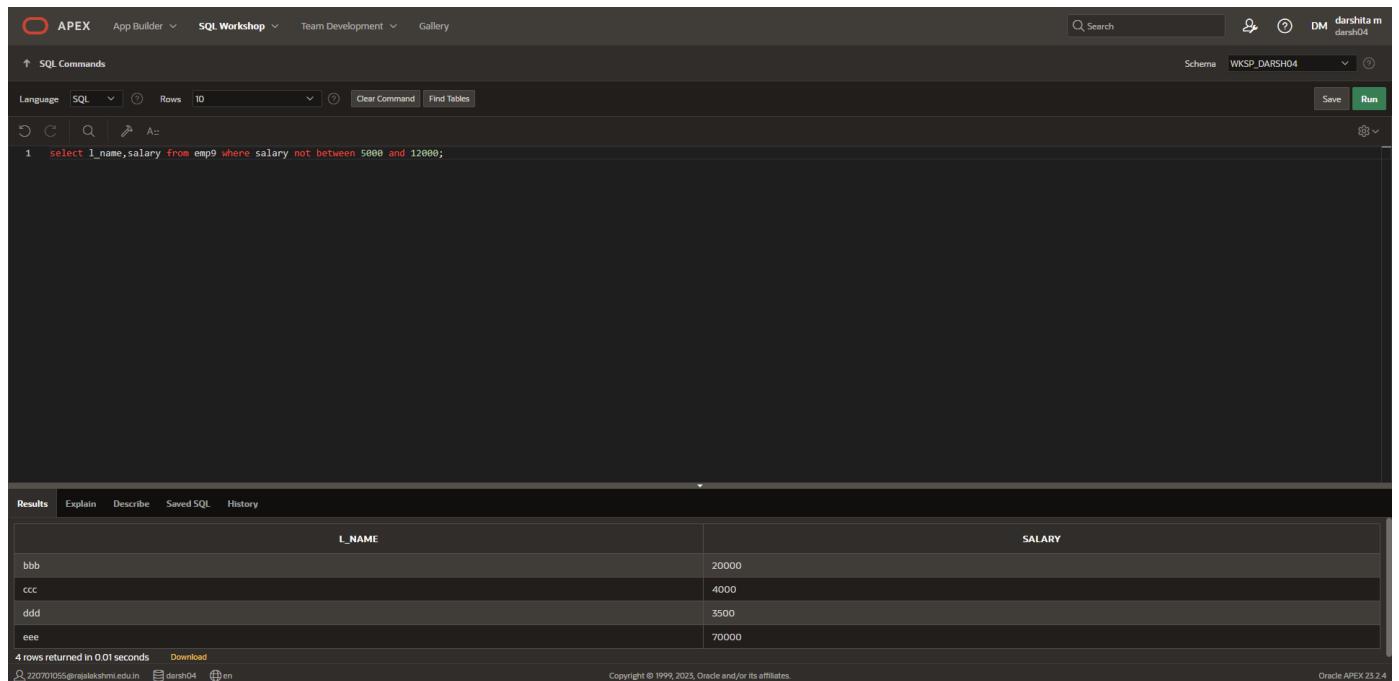
| L_NAME | DEPT_NUMBER |
|--------|-------------|
| aaa | 100 |

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 l_name,salary from emp9 where salary not between 5000 and 12000;

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side of the header shows the user 'darshita m' and schema 'WKSP_DARSH04'. The main workspace has tabs for SQL Commands, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the following code:

```
1 select l_name,salary from emp9 where salary not between 5000 and 12000;
```

The results section displays the output of the query:

| L_NAME | SALARY |
|--------|--------|
| bbb | 20000 |
| ccc | 4000 |
| ddd | 3500 |
| eee | 70000 |

Below the results, it says '4 rows returned in 0.01 seconds' and provides download and refresh options. The bottom footer includes copyright information for Oracle and the APEX version.

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

QUERY:

Select l_name,job_id,startdate from emp9 where startdate between ‘feb-20-1998’ and ‘may-1-1998’;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. At the top, there are tabs for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, it shows the user 'darshita m' and the schema 'WKSP_DARSH04'. Below the tabs, there's a search bar and a toolbar with icons for Save and Run.

In the main area, the SQL Commands tab is selected. The SQL code entered is:

```
1 select l_name,job_id,startdate from emp9 where startdate between 'feb-20-1998' and 'may-1-1998';
```

Below the code, the Results tab is selected, showing the query results in a grid:

| L_NAME | JOB_ID | STARTDATE |
|--------|--------|------------|
| bbb | 123 | 05/16/1998 |
| ccc | 456 | 05/01/1998 |
| aaa | 176 | 02/20/1998 |

At the bottom of the results grid, it says '3 rows returned in 0.00 seconds'. There are also download and refresh buttons.

At the very bottom of the page, there are footer links for 220701055@rajalakshmi.edu.in, dash04, and en, along with copyright information: Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4.

5.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 l_name,dept_number from emp9 where department in (20,50) order by l_name;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side of the header shows the schema (WKSP_DARSH04), user (darshita m dash04), and a search bar. The main area is titled 'SQL Commands' with a language dropdown set to 'SQL'. The query entered is: 'select l_name,dept_number from emp9 where department in (20,50) order by l_name;'. Below the query, there are buttons for Save and Run. The results tab is selected, displaying the output of the query. The output shows three rows with columns 'L_NAME' and 'DEPT_NUMBER':

| L_NAME | DEPT_NUMBER |
|--------|-------------|
| ccc | 987 |
| ddd | 671 |
| eee | 908 |

At the bottom left, it says '3 rows returned in 0.01 seconds'. At the bottom right, it says 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

QUERY:

Select l_name,salary from emp9 where salary between 5000 and 12000 and department in (20,50) order by l_name;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side of the header shows the schema 'WKSP_DARSH04' and a user 'darshita m'. The main workspace has a toolbar with various icons for SQL operations like Insert, Update, Delete, and Find Tables. Below the toolbar is a search bar and a 'Run' button. The SQL command entered is:

```
1 select l_name,salary from emp9 where salary between 5000 and 12000 and department in(20,50) order by l_name;
```

The results section displays the output of the query:

| L_NAME | SALARY |
|--------|--------|
| ddd | 5000 |
| eee | 12000 |

Below the table, it says '2 rows returned in 0.00 seconds' and provides a 'Download' link.

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

QUERY:

Select l_name, hire_date from emp9 where hire_date like '%94';

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. At the top, there are tabs for App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, the user is identified as 'darshita m' with session ID 'darsh04'. Below the tabs, there's a search bar and a toolbar with icons for Save, Run, and other functions.

In the main area, the SQL command is displayed:

```
1 select l_name, hire_date from emp9 where hire_date like '%94';
```

The results are shown in a table:

| L_NAME | HIRE_DATE |
|--------|------------|
| ddd | 05/04/1994 |
| ttf | 07/19/1994 |

Below the table, it says '2 rows returned in 0.02 seconds' and has a 'Download' link. At the bottom of the page, there are copyright notices for Oracle and APEX, along with user information: '220701055@rejalakshmi.edu.in' and 'darsh04'. The footer also includes a link to 'en'.

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

QUERY:

Select l_name,job_Ttle from emp9 where manager is null;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. At the top, there are tabs for App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a help icon, and a session identifier "DM darshita m darsh04". Below the tabs, there are buttons for Language (set to SQL), Rows (set to 10), Clear Command, Find Tables, Save, and Run. The SQL command entered is:

```
1 select l_name,job_title from emp9 where manager is null;
```

The Results tab is selected, showing the output of the query:

| L_NAME | JOB_TITLE |
|--------|--------------|
| ccc | team leader |
| ddd | designer |
| eee | project head |

Below the table, it says "3 rows returned in 0.03 seconds" and has a "Download" link. At the bottom, it shows the user information "220701055@rajalakshmi.edu.in darsh04", the copyright notice "Copyright © 1999-2023, Oracle and/or its affiliates.", and the version "Oracle APEX 23.2.4".

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

QUERY:

Select l_name,salary,commission from emp9 where commission is not null order by salary desc;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The schema is set to WKS_P_DARSH04. The SQL command entered is:

```
1 select l_name,salary,commission from emp9 where commission is not null order by salary desc;
```

The results section displays the following data:

| L_NAME | SALARY | COMMISSION |
|--------|--------|------------|
| eee | 12000 | 8 |
| aaa | 6000 | 20 |
| ddd | 5000 | 6 |
| ccc | 4000 | 10 |

Below the table, it says "4 rows returned in 0.01 seconds". The bottom of the screen shows user information (220701055@realekshmi.edu.in, darsh04, en) and copyright information (Copyright © 1999, 2023, Oracle and/or its affiliates). The footer also indicates Oracle APEX 23.2.4.

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

QUERY:

Select l_name from emp9 where l_name like '%__a';

OUTPUT:

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

```
1 select l_name from emp9 where l_name like '%__a';
```

Below the command window, the results tab is selected, showing a single row of data:

| L_NAME |
|--------|
| aaa |

At the bottom left of the results pane, it says "1 rows returned in 0.01 seconds".

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

QUERY:

Select l_name from emp9 where l_name like '%a%' and l_name like '%e%';

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area contains a SQL command: 'select l_name from emp9 where l_name like '%a%' and l_name like '%e%';'. Below the command, the results are displayed in a table with one row, showing the value 'aae'. The bottom of the screen shows user information and copyright details.

| L_NAME |
|--------|
| aae |

Results Explain Describe Saved SQL History

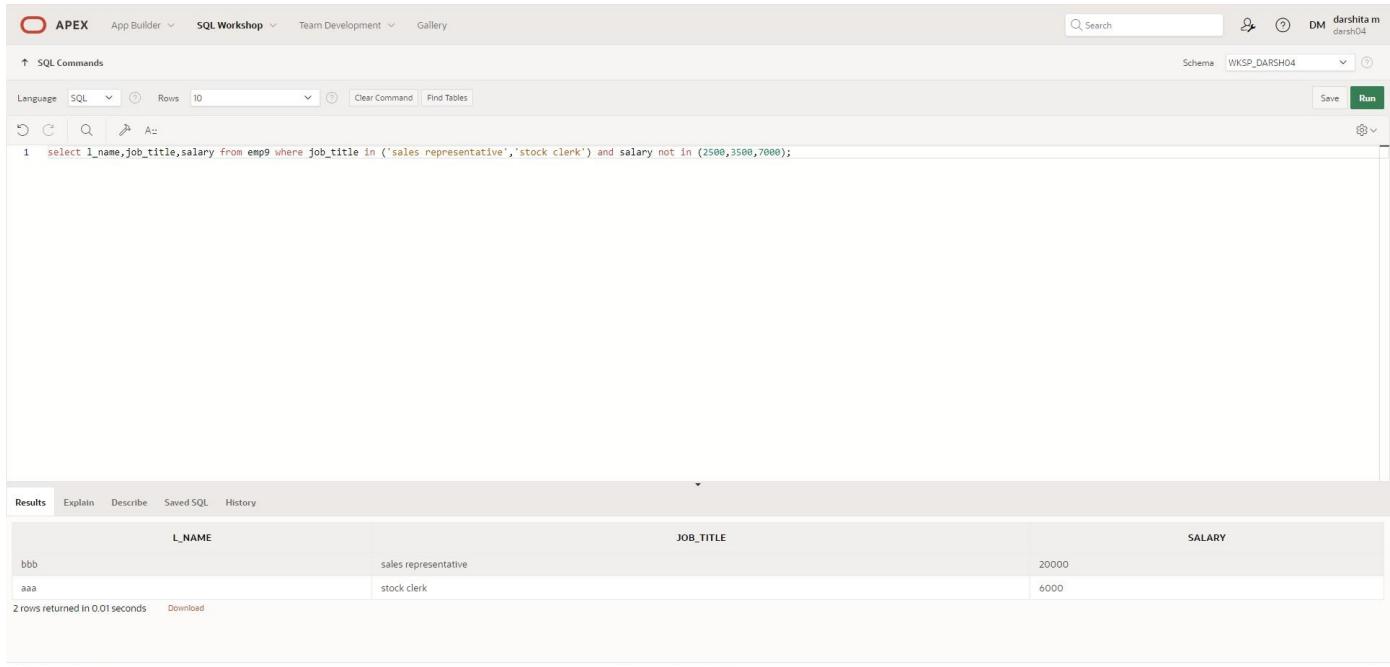
Copyright © 1999, 2023, Oracle and/or its affiliates.

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

QUERY:

Select l_name,job_Ttle, msalary from emp9 where job_Ttle in ('sales representative', 'stock clerk') and salary not in (2500,3500,7000);

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, it shows the user 'darshita m' and schema 'WKSP_DARSH04'. The main area is titled 'SQL Commands' with a search bar and a 'Run' button. Below the search bar are buttons for Undo, Redo, Find, and Paste. The SQL command entered is:

```
1 select l_name,job_title,salary from emp9 where job_title in ('sales representative','stock clerk') and salary not in (2500,3500,7000);
```

The results tab is selected, displaying the following data:

| L_NAME | JOB_TITLE | SALARY |
|--------|----------------------|--------|
| bbb | sales representative | 20000 |
| aaa | stock clerk | 6000 |

Below the table, it says '2 rows returned in 0.01 seconds' and has a 'Download' link. At the bottom, there are footer links for 220701055@rejislashmi.edu.in, darsh04, and en, along with copyright information: 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

QUERY:

Select l_name,salary,commission from emp9 where commission=.2;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, it shows the user 'darshita' and schema 'WKSP_DARSH04'. The main area has tabs for SQL Commands, Results (selected), Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the query: 'select l_name, salary, commission from emp9 where commission=.2;'. The Results tab shows the output: 'no data found'. The bottom footer includes user information (220701055@rajalakshmi.edu.in, darsh04, en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the version (Oracle APEX 23.2.4).

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

RESULT: Thus the sql queries on restricting and sorting data is successfully executed and verified.

SINGLE ROW FUNCTIONS

EX_NO:6

DATE:

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

QUERY:

```
select sysdate from dual;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the query `select sysdate from dual;` is entered. The Results pane displays the output: a single row with the column `SYSDATE` containing the value `03/09/2024`. The status bar at the bottom indicates "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_number, l_name, salary, salary+(15.5/100*salary) "new salary" from hr;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the query `select emp_number, l_name, salary, salary+ (salary*15.5/100) "NEW SALARY" from hr;` is entered. The Results pane displays a table with four columns: `EMP_NUMBER`, `L_NAME`, `SALARY`, and `NEW SALARY`. The data is as follows:

| EMP_NUMBER | L_NAME | SALARY | NEW SALARY |
|------------|--------|--------|------------|
| 3 | ccc | 6000 | 6930 |
| 5 | hhh | 9000 | 10395 |
| 2 | bbb | 3000 | 3465 |
| 4 | ddd | 8000 | 9240 |
| 1 | aaa | 12000 | 13860 |

The status bar at the bottom indicates "5 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_number, l_name, Salary, (Salary+(Salary*15.5/100))-Salary "Increase"  
From hr;
```

OUTPUT:

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

```
select emp_number, l_name, salary, salary+(15.5/100*salary) "NEW SALARY", (salary+(15.5/100*salary))-salary as "Increase" from hr;
```

The results table has columns: EMP_NUMBER, L_NAME, SALARY, NEW SALARY, and Increase. The data is:

| EMP_NUMBER | L_NAME | SALARY | NEW SALARY | Increase |
|------------|--------|--------|------------|----------|
| 3 | ccc | 6000 | 6930 | 930 |
| 5 | hhh | 9000 | 10395 | 1395 |
| 2 | bbb | 3000 | 3465 | 465 |
| 4 | jjj | 8000 | 9240 | 1240 |
| 1 | aaa | 12000 | 13860 | 1860 |

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

4.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(l_name), length(l_name) "Length of Name"  
from hr  
where l_name like 'J%' or l_name like 'A%' or l_name like 'M%'  
order by l_name;
```

OUTPUT:

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

```
select initcap(l_name), length(l_name) as "Length_of_last_name" from hr where l_name like 'j%' or l_name like 'a%' or l_name like 'm%' order by l_name asc;
```

The results table has columns: INITCAP(L_NAME) and Length_of_last_name. The data is:

| INITCAP(L_NAME) | Length_of_last_name |
|-----------------|---------------------|
| Aaa | 3 |
| Jjj | 3 |

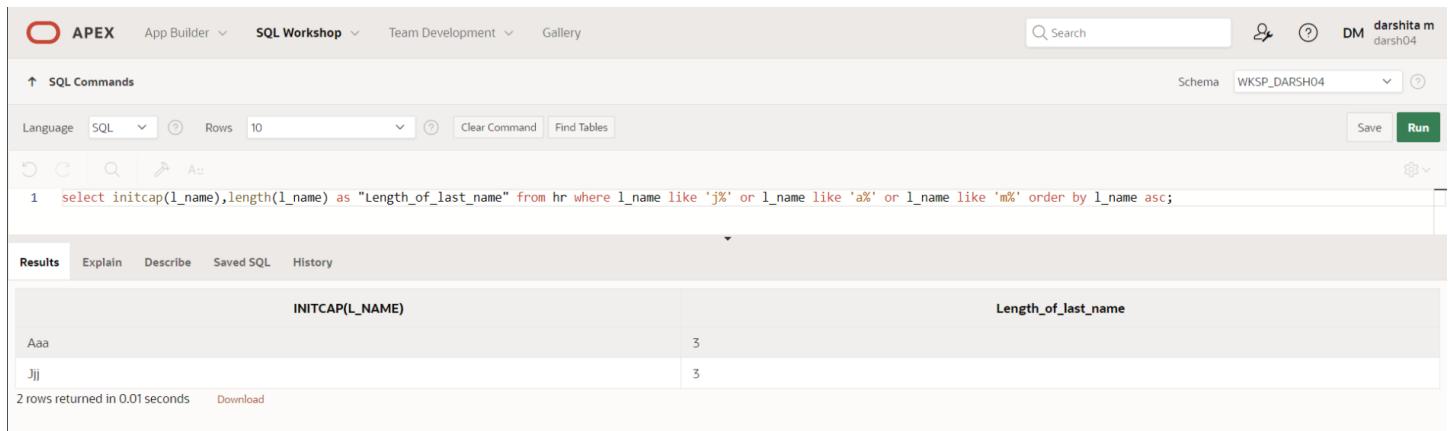
2 rows returned in 0.01 seconds [Download](#)

5. Rewrite the query so that the user is prompted to enter a letter that starts the last name. For example, if the user enters H when prompted for a letter, then the output should show all employees whose last name starts with the letter H.

QUERY:

```
select initcap(l_name) , length(l_name) "Length of Name"  
from hr  
where l_name like '&name%'  
order by l_name;
```

OUTPUT:



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, it shows the user 'darshita m' and session 'darsh04'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the following query:

```
1 select initcap(l_name),length(l_name) as "Length_of_last_name" from hr where l_name like 'j%' or l_name like 'a%' or l_name like 'm%' order by l_name asc;
```

The Results tab displays the output:

| INITCAP(L_NAME) | Length_of_last_name |
|-----------------|---------------------|
| Aaa | 3 |
| Jjj | 3 |

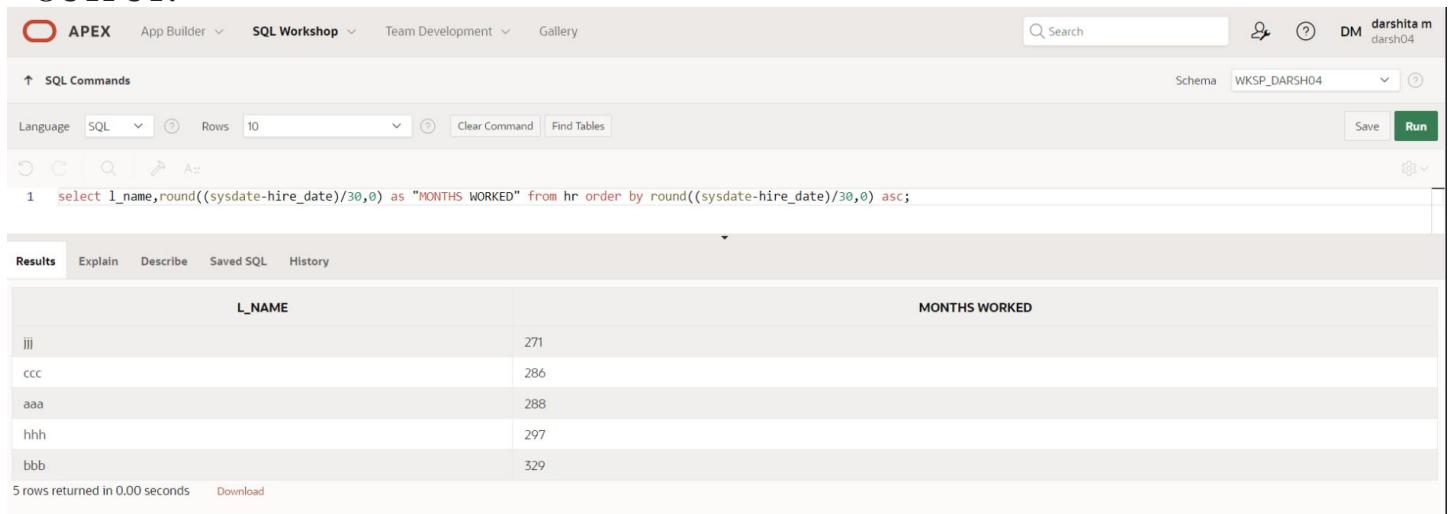
2 rows returned in 0.01 seconds. There is a 'Download' link at the bottom.

6. 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 l_name, round((sysdate,hire_date)/30,0) Months_worked  
from hr order by round((sysdate,hire_date)/30,0) ;
```

OUTPUT:



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, it shows the user 'darshita m' and session 'darsh04'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the following query:

```
1 select l_name,round((sysdate-hire_date)/30,0) as "MONTHS WORKED" from hr order by round((sysdate-hire_date)/30,0) asc;
```

The Results tab displays the output:

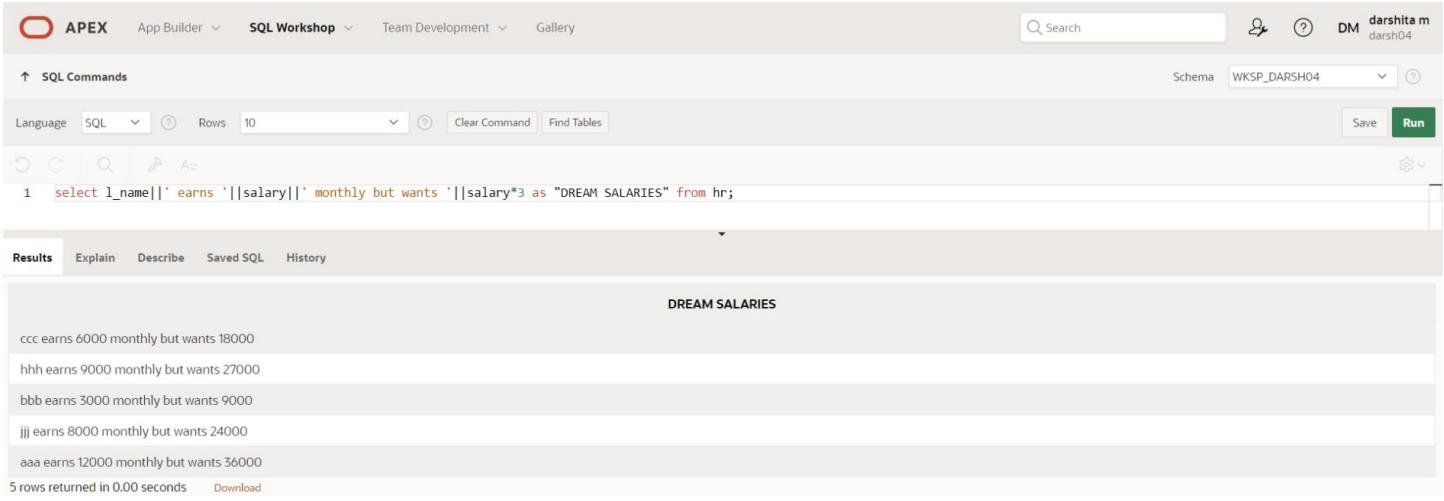
| L_NAME | MONTHS WORKED |
|--------|---------------|
| jjj | 271 |
| ccc | 286 |
| aaa | 288 |
| hhh | 297 |
| bbb | 329 |

5 rows returned in 0.00 seconds. There is a 'Download' link at the bottom.

7. 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 l_name||' earns'||salary||' monthly but wants'||salary*3 "Dream Salaries"
from hr;

OUTPUT:

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

```
1 select l_name||' earns'||salary||' monthly but wants'||salary*3 as "DREAM SALARIES" from hr;
```

The results tab is selected, displaying the output:

DREAM SALARIES

| |
|---|
| ccc earns 6000 monthly but wants 18000 |
| hhh earns 9000 monthly but wants 27000 |
| bbb earns 3000 monthly but wants 9000 |
| jjj earns 8000 monthly but wants 24000 |
| aaa earns 12000 monthly but wants 36000 |

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

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

RESULT: Thus the sql queries on single row functions are verified and executed successfully.

DISPLAYING DATA FROM MULTIPLE TABLES

EX_NO:7

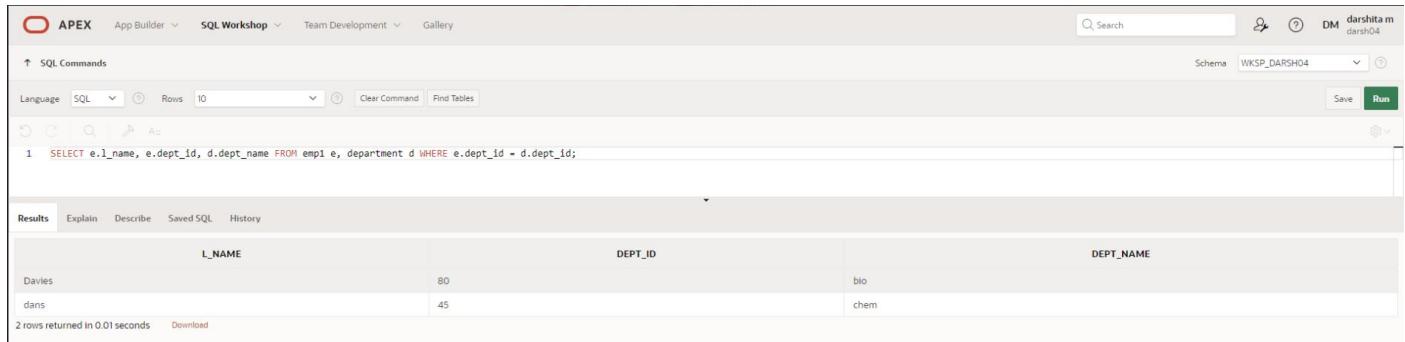
DATE:

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

QUERY:

```
SELECT e.l_name, e.dept_id, d.dept_name FROM emp1 e, department d WHERE e.dept_id = d.dept_id;
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The query window contains the following code:

```
1 SELECT e.l_name, e.dept_id, d.dept_name FROM emp1 e, department d WHERE e.dept_id = d.dept_id;
```

The results pane displays the following data:

| L_NAME | DEPT_ID | DEPT_NAME |
|--------|---------|-----------|
| Davies | 80 | bio |
| dans | 45 | chem |

2 rows returned in 0.01 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 emp1 e, department d WHERE e.dept_id = d.dept_id AND e.dept_id = 80;
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The query window contains the following code:

```
1 SELECT DISTINCT job_id, loc_id FROM emp1 e, department d WHERE e.dept_id = d.dept_id AND e.dept_id = 80;
```

The results pane displays the following data:

| JOB_ID | LOC_ID |
|--------|--------|
| 120 | 123 |

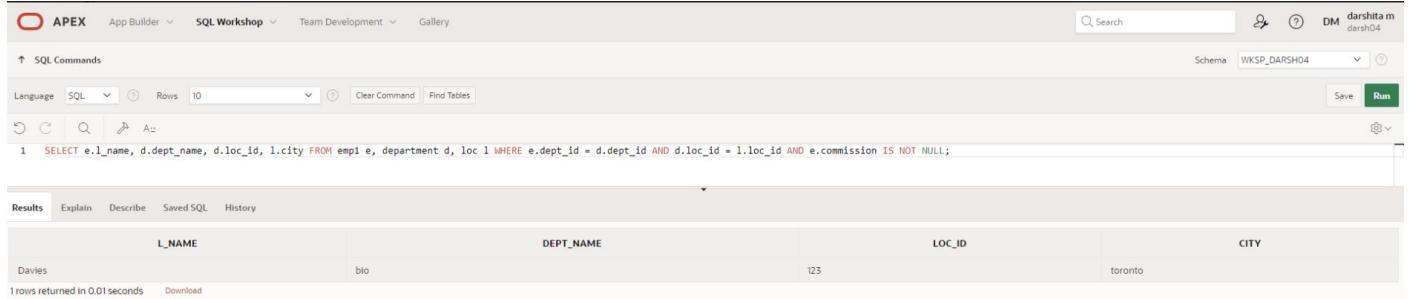
1 rows returned in 0.05 seconds

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 e.l_name, d.dept_name, d.loc_id, l.city FROM emp1 e, department d, loc l WHERE e.dept_id = d.dept_id AND d.loc_id = l.loc_id AND e.commission IS NOT NULL;
```

OUTPUT:



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

```
1 SELECT e.l_name, d.dept_name, d.loc_id, l.city FROM emp1 e, department d, loc l WHERE e.dept_id = d.dept_id AND d.loc_id = l.loc_id AND e.commission IS NOT NULL;
```

The results table has four columns: L_NAME, DEPT_NAME, LOC_ID, and CITY. The single row returned is:

| L_NAME | DEPT_NAME | LOC_ID | CITY |
|--------|-----------|--------|---------|
| Davies | bio | 123 | toronto |

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

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

QUERY:

```
select emp1.l_name, department.dept_name from emp1, department where emp1.dept_id=department.dept_id and l_name like '%a%';
```

OUTPUT:



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

```
1 select emp1.l_name, department.dept_name from emp1,department where emp1.dept_id=department.dept_id and l_name like '%a%';
```

The results table has two columns: L_NAME and DEPT_NAME. The two rows returned are:

| L_NAME | DEPT_NAME |
|--------|-----------|
| Davies | bio |
| dans | chem |

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

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

QUERY:

```
select e. l_name, e. job_id, e.dept_id,d.dept_name from empl e join department d on (e.dept_id=d.dept_id) join loc on (d. loc_id= loc. loc_id) where lower (loc.city)= 'toronto' ;
```

OUTPUT:

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

```
1 select e.l_name, e.job_id,e.dept_id,d.dept_name from empl e join department d on (e.dept_id=d.dept_id) join loc on (d.loc_id= loc.loc_id) where lower(loc.city)= 'toronto';
```

The results table has columns L_NAME, JOB_ID, DEPT_ID, and DEPT_NAME. One row is returned:

| L_NAME | JOB_ID | DEPT_ID | DEPT_NAME |
|--------|--------|---------|-----------|
| Davies | 120 | 80 | bio |

1 rows returned in 0.01 seconds

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

QUERY:

```
SELECT w.l_name "Employee", w.emp_id "emp#", m. l_name "manager", m.emp_id "Mgr#" FROM emp1 w join emp1 m ON (w.manager_id = m. emp_id);
```

OUTPUT:

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

```
1 SELECT w.l_name "Employee", w.emp_id "emp#", m.l_name "manager", m.emp_id "Mgr#" FROM emp1 w join emp1 m ON (w.manager_id = m.emp_id);
```

The results table has columns Employee, emp#, manager, and Mgr#. One row is returned:

| Employee | emp# | manager | Mgr# |
|----------|------|---------|------|
| gunwook | 1015 | gunwook | 1015 |

1 rows returned in 0.01 seconds

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

QUERY:

```
SELECT w.l_name "Employee", w.emp_id "EMP#", m. l_name "Manager", m. emp_id "Mgr#" FROM emp1 w LEFT OUTER JOIN emp1 m ON (w.manager_id = m. emp_id);
```

OUTPUT:



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

```
1 SELECT w.l_name "Employee", w.emp_id "EMP#", m. l_name "Manager", m. emp_id "Mgr#" FROM emp1 w LEFT OUTER JOIN emp1 m ON (w.manager_id = m. emp_id);
```

The results table has four columns: Employee, EMP#, Manager, and Mgr#. The data is:

| Employee | EMP# | Manager | Mgr# |
|----------|------|---------|------|
| gunwook | 1015 | gunwook | 1015 |
| dans | 491 | - | - |
| Davies | 467 | - | - |

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

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

QUERY:

```
SELECT e. dept_id department, e.l_name empi, c.l_name colleague FROM emp1 e JOIN emp1 C ON (e dept_id = c.dept_id) WHERE e. emp_id < c. emp_id ORDER BY e. dept_id, e.l_name, c.l._name;
```

OUTPUT:



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

```
1 SELECT e.dept_id department, e.l_name empi, c.l_name colleague FROM emp1 e JOIN emp1 c ON (e.dept_id = c.dept_id) WHERE e.emp_id < c.emp_id
2 ORDER BY e.dept_id, e.l_name, c.l_name
```

The results table has three columns: DEPARTMENT, EMPI, and COLLEAGUE. The data is:

| DEPARTMENT | EMPI | COLLEAGUE |
|------------|--------|-----------|
| 80 | Davies | dans |
| 80 | dans | Davies |

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

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

QUERY:

```
Desc job_grades;  
select e.l_name, e.job_id, d.dept_name, e.salary, j.grade_level from emple, department d, job_grades j  
where e.dept_id= d.dept_id and e.salary between j.low_sal and j.high_sal;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' and 'SQL Workshop' are selected. The main area shows the command 'desc job_grades;' entered in the SQL editor. Below it, the 'Describe' tab is selected in the results panel, showing the structure of the 'JOB_GRADES' table. The table has three columns: 'GRADE_LEVEL' (VARCHAR2(50)), 'LOW_SAL' (NUMBER), and 'HIGH_SAL' (NUMBER). The 'GRADE_LEVEL' column is the primary key.

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|------------|-------------|-----------|--------|-----------|-------|-------------|----------|---------|---------|
| JOB_GRADES | GRADE_LEVEL | VARCHAR2 | 50 | - | - | - | ✓ | - | - |
| | LOW_SAL | NUMBER | - | 6 | 0 | - | ✓ | - | - |
| | HIGH_SAL | NUMBER | - | 6 | 0 | - | ✓ | - | - |

The screenshot shows the Oracle SQL Workshop interface. The SQL editor contains the query: 'select e.l_name, e.job_id, d.dept_name, e.salary, j.grade_level from emple e, department d, job_grades j where e.dept_id= d.dept_id and e.salary between j.low_sal and j.high_sal;'. The 'Run' button is highlighted. The results panel shows the message 'no data found'.

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

QUERY:

```
select e.l_name, e.hire_date from emp1 e, emp1.davies where davies.l_name= 'davies' and  
davies.hire_date < e.hire_date;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL editor contains the query: 'select e.l_name, e.hire_date from emp1 e, emp1.davies where davies.l_name= 'davies' and davies.hire_date < e.hire_date;'. The 'Run' button is highlighted. The results panel shows the message 'no data found'.

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

RESULT: Thus the sql queries on displaying data from multiple tables is executed and verified successfully.

AGGREGATING DATA USING GROUP FUNCTIONS

EX_NO:8

DATE:

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

True/False

TRUE

2. Group functions include nulls in calculations.

True/False

FALSE

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

True/False

FALSE

4. Find the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum

, Sum, and Average, respectively. Round your results to the nearest whole number

QUERY:

```
SELECT ROUND(MAX(salary),0) "Maximum", ROUND(MIN(salary),0) "Minimum",
ROUND(SUM(salary),
,0) "Sum", ROUND(AVG(salary),0) "Average" FROM emp1;
```

OUTPUT:

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 'darshita m' and a 'Run' button. The main area has tabs for 'SQL Commands' and 'Results'. Under 'Results', there are tabs for 'Results' (selected), 'Explain', 'Describe', 'Saved SQL', and 'History'. The results table has four columns: 'Maximum', 'Minimum', 'Sum', and 'Average'. The data row shows values: Maximum 81265, Minimum 6544, Sum 103574, and Average 34525. A note at the bottom says '1 rows returned in 0.04 seconds'. There are also 'Download' and 'Run' buttons for the results table.

| Maximum | Minimum | Sum | Average |
|---------|---------|--------|---------|
| 81265 | 6544 | 103574 | 34525 |

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

QUERY:

```
SELECT job_id, ROUND(MAX(salary),0) "Maximum", ROUND(MIN(salary),0)  
"Minimum", ROUND(SUM(salary),0) "Sum", ROUND(AVG(salary),0) "Average" FROM emp1  
GROUP BY job_id;
```

OUTPUT:

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

```
1 SELECT job_id, ROUND(MAX(salary),0) "Maximum", ROUND(MIN(salary),0)  
2 "Minimum", ROUND(SUM(salary),0) "Sum", ROUND(AVG(salary),0) "Average" FROM emp1 GROUP BY job_id;
```

The results are displayed in a table:

| JOB_ID | Maximum | Minimum | Sum | Average |
|--------|---------|---------|-------|---------|
| 564 | 6544 | 6544 | 6544 | 6544 |
| 120 | 81265 | 15765 | 97030 | 48515 |

2 rows returned in 0.00 seconds

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

QUERY:

```
SELECT job_id, COUNT(*) FROM emp1 GROUP BY job_id;
```

OUTPUT:

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

```
1 SELECT job_id, COUNT(*) FROM emp1 GROUP BY job_id;
```

The results are displayed in a table:

| JOB_ID | COUNT(*) |
|--------|----------|
| 564 | 1 |
| 120 | 2 |

2 rows returned in 0.00 seconds

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

QUERY:

Select count(manager_id) as Number_of_Manager from emp1;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The query entered is: `1 Select count(manager_id) as Number_of_Manager from emp1;`. The results section displays a single row with the heading `NUMBER_OF_MANAGER` and the value `3`.

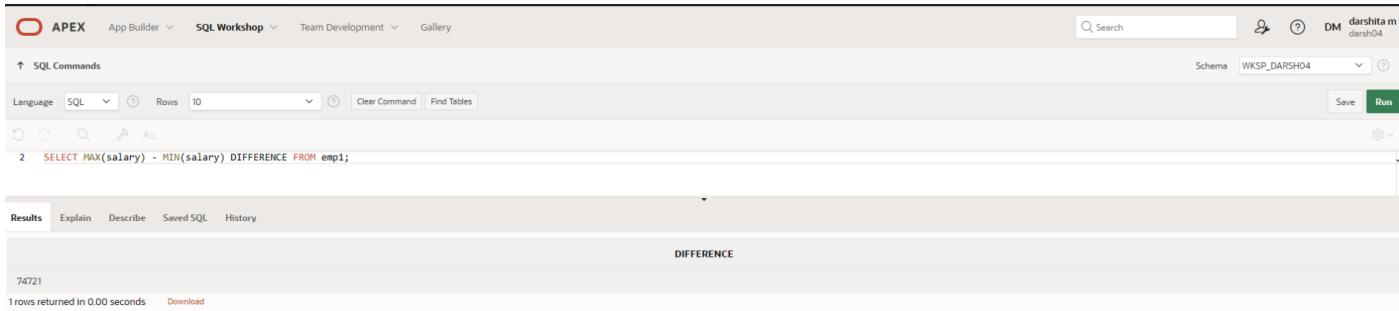
| NUMBER_OF_MANAGER |
|-------------------|
| 3 |

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

QUERY:

SELECT MAX(salary) - MIN(salary) DIFFERENCE FROM emp1;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The query entered is: `2 SELECT MAX(salary) - MIN(salary) DIFFERENCE FROM emp1;`. The results section displays a single row with the heading `DIFFERENCE` and the value `74721`.

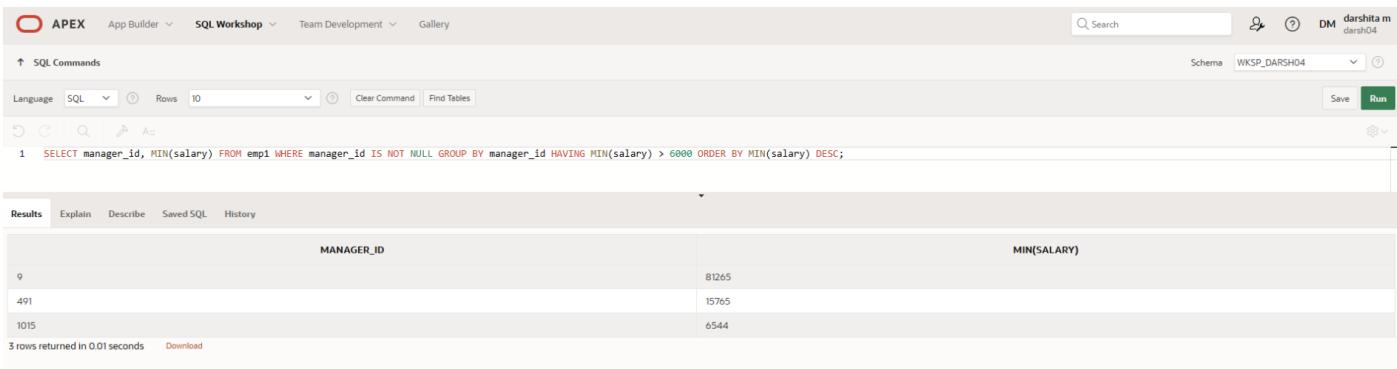
| DIFFERENCE |
|------------|
| 74721 |

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

QUERY:

```
SELECT manager_id, MIN(salary) FROM emp1 WHERE manager_id IS NOT NULL GROUP BY manager_id HAVING MIN(salary) > 6000 ORDER BY MIN(salary) DESC;
```

OUTPUT:



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, it shows the user 'darshita m' and workspace 'WKSP_DARSH04'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, displaying the following SQL code:

```
1 SELECT manager_id, MIN(salary) FROM emp1 WHERE manager_id IS NOT NULL GROUP BY manager_id HAVING MIN(salary) > 6000 ORDER BY MIN(salary) DESC;
```

The Results tab shows the output of the query:

| MANAGER_ID | MIN(SALARY) |
|------------|-------------|
| 9 | 81265 |
| 491 | 15765 |
| 1015 | 6544 |

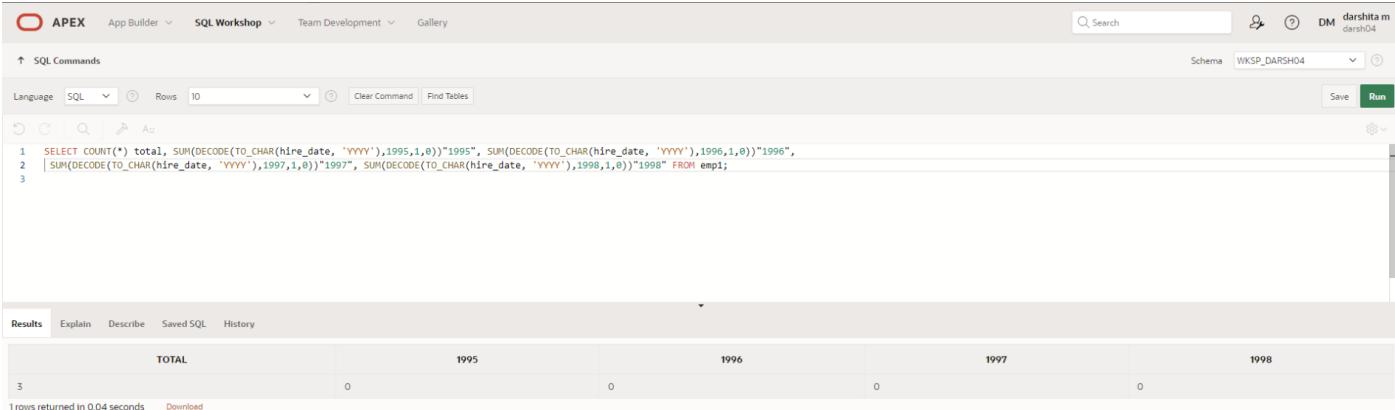
At the bottom left, it says '3 rows returned in 0.01 seconds'. There are 'Download' and 'Run' buttons at the bottom right.

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

QUERY:

```
SELECT COUNT(*) total, SUM(DECODE(TO_CHAR(hire_date,'YYY'),1995,1,0))"1995"  
,SUM(DECODE (TO_CHAR(hire_date,'W*'), 1996,1,0))"1996" ,  
 SUM(DECODE(TO_CHAR(hire_date, 'MYY'), 1997,1,0)) "1997", SUM(DECODE(TO_CHAR (hire_date,  
'MY'), 1998,1,0)) "1998" FROM emp1;
```

OUTPUT:



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

```
1 SELECT COUNT(*) total, SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),1995,1,0))"1995", SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),1996,1,0))"1996",  
2 | SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),1997,1,0))"1997", SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),1998,1,0))"1998" FROM emp1;  
3
```

The Results tab displays the output:

| | TOTAL | 1995 | 1996 | 1997 | 1998 |
|---|-------|------|------|------|------|
| 3 | 0 | 0 | 0 | 0 | 0 |

1 rows returned in 0.04 seconds Download

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

QUERY:

```
select dept_id, job_id,sum(salary) from emp1 where dept_id in(20,50,80,9) group by rollup(dept_id,job_id);
```

OUTPUT:



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

```
1 select dept_id,job_id,sum(salary) from emp1 where dept_id in(20,50,80,9) group by rollup(dept_id,job_id);
```

The results section displays the following data:

| DEPT_ID | JOB_ID | SUM(SALARY) |
|---------|--------|-------------|
| 80 | 120 | 97030 |
| 80 | - | 97030 |
| - | - | 97030 |

3 rows returned in 0.03 seconds

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

QUERY:

```
select d.dept_name as DNAME, l.loc_id as LOC, count(e.dept_id)as "Number of people",
round(avg(e.salary),2) as  "Salary" from department d, emp1 e, loc l where d.dept_id = e.dept_id group by
d.dept_name, l.loc_id, e.dept_id;
```

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 SQL Commands tab is active, showing the following SQL code:

```
1 select d.dept_name as DNAME, l.loc_id as LOC, count(e.dept_id)as "Number of people",
2 round(avg(e.salary),2) as  "Salary" from department d, emp1 e, loc l where d.dept_id = e.dept_id group by
3 d.dept_name, l.loc_id, e.dept_id;
```

The Results tab displays the output of the query:

| DNAME | LOC | Number of people | Salary |
|-------|-----|------------------|--------|
| blo | 456 | 2 | 48515 |
| blo | 123 | 2 | 48515 |

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

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

RESULT: Thus the sql queries on aggregating data using group functions is verified and executed successfully.

SUB QUERIES

EX_NO:9

DATE:28.03.2024

- 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 l_name, hire_date from emp1 where dept_id = (select dept_id from emp1 where l_name like 'Zlotkey')
```

```
and l_name <> 'Zlotkey';
```

OUTPUT:



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 search bar, a user icon for darshita m, and a schema dropdown set to WKSP_DARSH04. The main workspace has tabs for SQL Commands, SQL, and Run. The SQL tab contains the following code:

```
1 Select l_name, hire_date from emp1 where dept_id = (select dept_id from emp1 where l_name like 'zlotkey')and l_name<> 'zlotkey';
```

The Results tab shows the output:

| L_NAME | HIRE_DATE |
|--------|------------|
| Davies | 02/09/1990 |

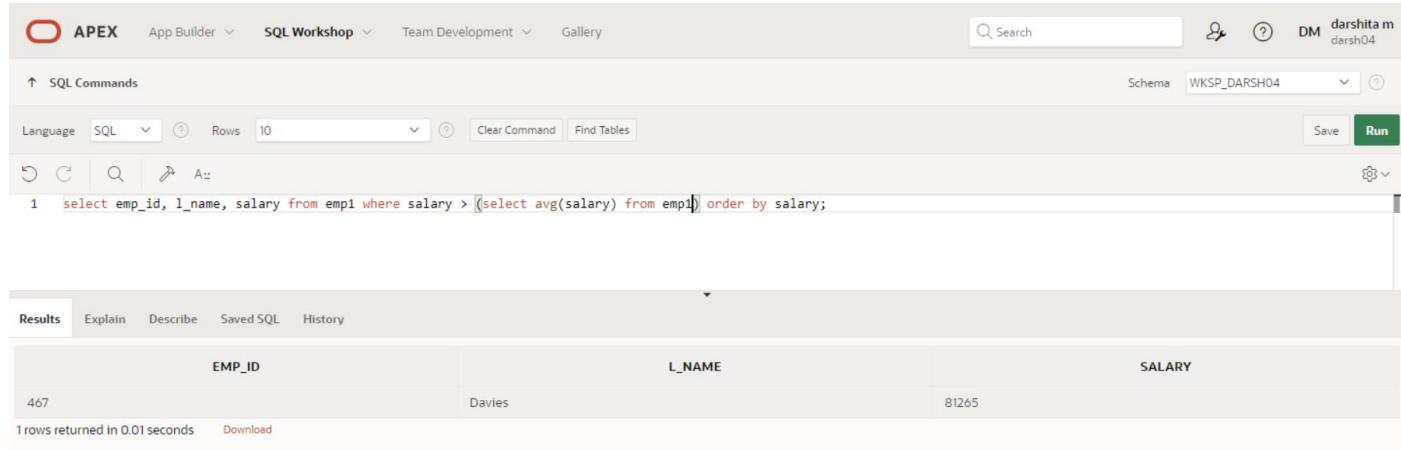
Below the results, it says "1 rows returned in 0.01 seconds" and has a Download link.

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,l_name,salary from emp1 where salary>(select avg(salary) from emp1) order by salary;
```

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 the user 'darshita m' and schema 'WKSP_DARSH04'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab shows the query: 'select emp_id, l_name, salary from emp1 where salary > (select avg(salary) from emp1) order by salary;'. The Results tab displays the output:

| EMP_ID | L_NAME | SALARY |
|--------|--------|--------|
| 467 | Davies | 81265 |

1 rows returned in 0.01 seconds Download

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,l_name from emp1 where dept_id=(select dept_id from emp1 where l_name like'%u%');
```

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 the user 'darshita m' and schema 'WKSP_DARSH04'. The main area is titled 'SQL Commands' with a 'Run' button. The code entered is:

```
1 select emp_id, l_name from emp1 where dept_id in (select dept_id from emp1 where l_name like '%u%');
```

The results tab is selected, showing a single row:

| EMP_ID | L_NAME |
|--------|---------|
| 1015 | gunwook |

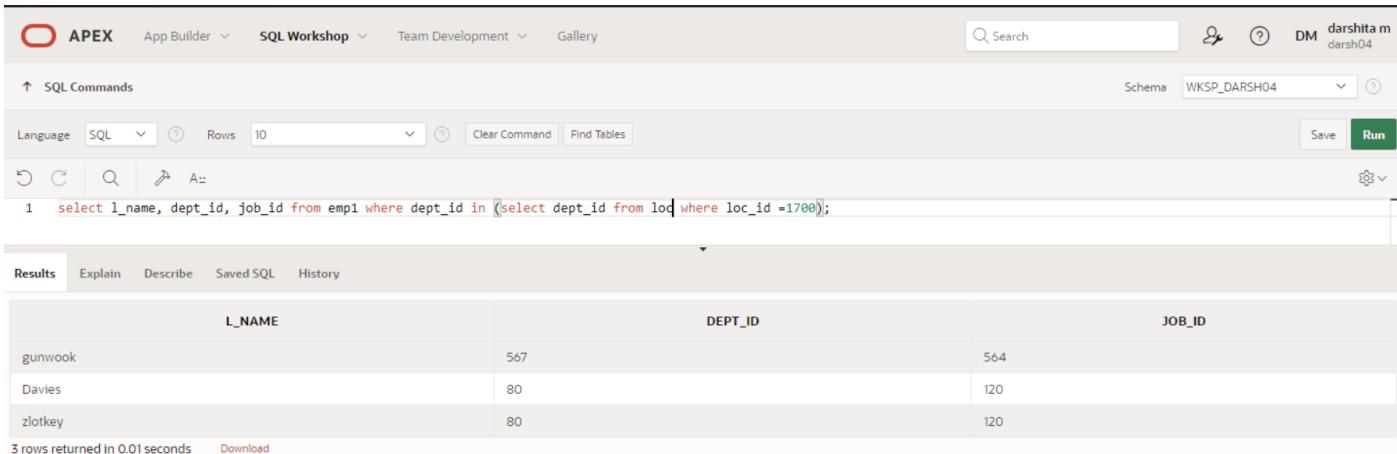
1 rows returned in 0.01 seconds

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 l_name,dept_id,job_id from emp1 where dept_id=(select dept_id from loc where loc_id=1700);
```

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 the user 'darshita m' and schema 'WKSP_DARSH04'. The main area is titled 'SQL Commands' with a 'Run' button. The code entered is:

```
1 select l_name, dept_id, job_id from emp1 where dept_id in (select dept_id from loc where loc_id =1700);
```

The results tab is selected, showing three rows:

| L_NAME | DEPT_ID | JOB_ID |
|---------|---------|--------|
| gunwook | 567 | 564 |
| Davies | 80 | 120 |
| zlotkey | 80 | 120 |

3 rows returned in 0.01 seconds

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

QUERY:

```
select l_name, salary from emp1 where manager_id in (select emp_id from emp1 where l_name='King');
```

OUTPUT:

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, it shows the user 'darshita m' and 'darsh04'. The main area has a 'SQL Commands' tab selected. The SQL command entered is:

```
1 select l_name, salary from emp1 where manager_id in (select emp_id from emp1 where l_name='King');
```

The results tab is selected, displaying the output:

| L_NAME | SALARY |
|--------|--------|
| king | 15765 |

1 rows returned in 0.00 seconds

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

QUERY:

```
select dept_id, l_name, job_id from emp1 where dept_id in (select dept_id from department where dept_name = 'Executive');
```

OUTPUT:

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, it shows the user 'darshita m' and 'darsh04'. The main area has a 'SQL Commands' tab selected. The SQL command entered is:

```
1 select dept_id, l_name, job_id from emp1 where dept_id in (select dept_id from department where dept_name = 'Executive');
```

The results tab is selected, displaying the output:

| DEPT_ID | L_NAME | JOB_ID |
|---------|--------|--------|
| 80 | Davies | 120 |
| 80 | king | 120 |

2 rows returned in 0.01 seconds

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

QUERY:

```
select emp_id,l_name,salary from emp1 where salary>(select avg(salary) from emp1 where l_name like '%u%');
```

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 the user 'darshtita m' and schema 'WKSP_DARSH04'. The main area has tabs for SQL Commands, Results (selected), Explain, Describe, Saved SQL, and History. The SQL Commands tab shows the query: 'select emp_id, l_name, salary from emp1 where salary > (select avg(salary) from emp1) and dept_id in (select dept_id from emp1 where l_name like '%u%');'. The Results tab displays the output in a grid:

| EMP_ID | L_NAME | SALARY |
|--------|--------|--------|
| 467 | Davies | 81265 |

1 rows returned in 0.01 seconds. There is a 'Download' link at the bottom.

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

RESULT: Thus the sql sub queries are executed and verified successfully.

USING THE SET OPERATORS

EX_NO:10

DATE:06.04.2024

- 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 dept_id from emp1 minus select dept_id from emp1 where job_id ='st_clerk';
```

OUTPUT:



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 search bar, a user icon for 'darshita m', and a schema dropdown set to 'WKSP_DARSH04'. The main area is titled 'SQL Commands' with a language dropdown set to 'SQL'. Below it, there are buttons for 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. The SQL command entered is:

```
1 select dept_id from emp1 minus select dept_id from emp1 where job_id ='st_clerk';
```

Below the command, the 'Results' tab is selected, showing a single row of output:

| DEPT_ID |
|---------|
| 80 |

At the bottom left, it says '1 rows returned in 0.01 seconds' and there's a 'Download' link.

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 country_id,state from loc minus select country_id,state from loc,department where  
loc.loc_id=department.loc_id;
```

OUTPUT:

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 search bar, user information (darshita m, darsh04), and a Run button. The main area has tabs for SQL Commands and Results. Under SQL Commands, the schema is set to WKSP_DARSH04. The command entered is:

```
1 select country_id,state from loc minus select country_id,state from loc,department where loc.loc_id=department.loc_id;
```

Under Results, the output is displayed in a table:

| COUNTRY_ID | STATE |
|------------|------------|
| 8 | tamil nadu |

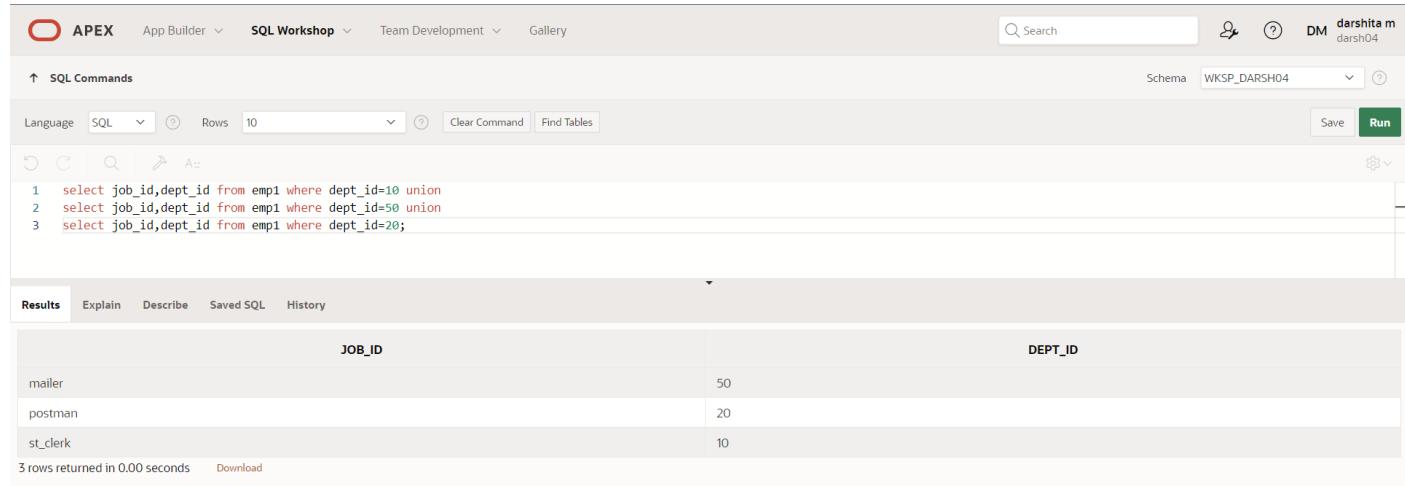
Below the table, it says "1 rows returned in 0.02 seconds" and has a Download link.

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_id,dept_id from emp1 where dept_id=10 union
select job_id,dept_id from emp1 where dept_id=50 union
select job_id,dept_id from emp1 where dept_id=20;
```

OUTPUT:



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 search bar, a user icon for 'darshita m darsh04', and a schema dropdown set to 'WKSP_DARSH04'. The main area is titled 'SQL Commands' with tabs for Language (set to SQL) and Rows (set to 10). Below these are buttons for Clear Command, Find Tables, Save, and Run. The SQL command area contains three numbered lines of code:

```
1 select job_id,dept_id from emp1 where dept_id=10 union
2 select job_id,dept_id from emp1 where dept_id=50 union
3 select job_id,dept_id from emp1 where dept_id=20;
```

The results section shows a table with two columns: JOB_ID and DEPT_ID. The data is as follows:

| JOB_ID | DEPT_ID |
|----------|---------|
| mailer | 50 |
| postman | 20 |
| st_clerk | 10 |

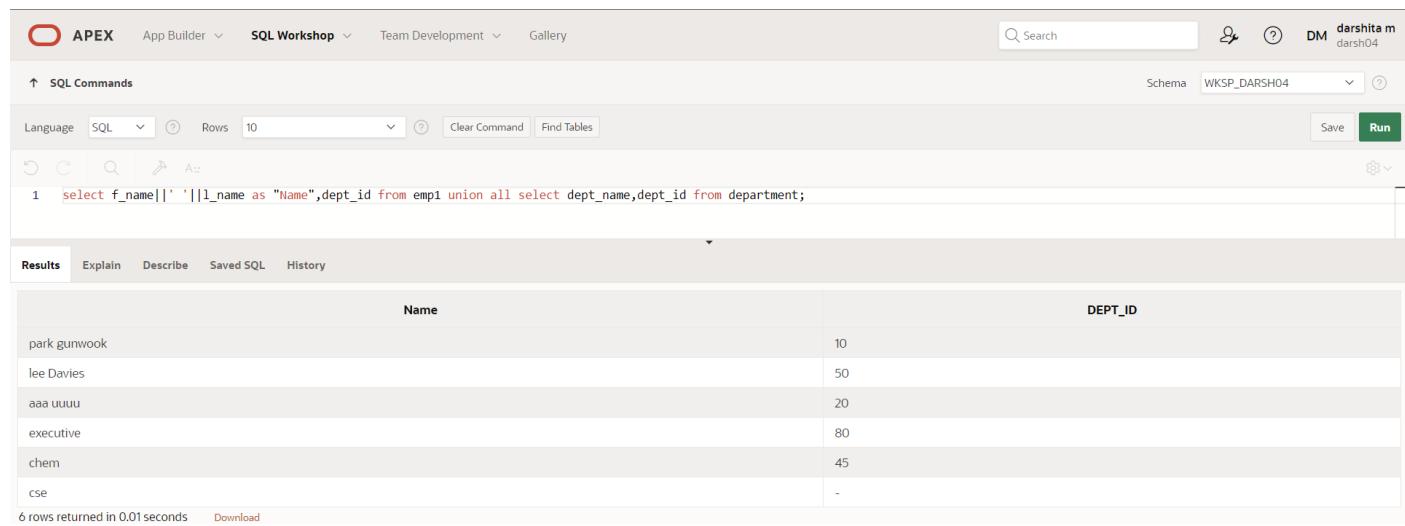
At the bottom left of the results section, it says '3 rows returned in 0.00 seconds' and has a 'Download' link.

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 f_name||"||l_name as "Name",dept_id from emp1 union all select dept_name,dept_id from department;
```

OUTPUT:



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

```
1 select f_name||' '||l_name as "Name",dept_id from emp1 union all select dept_name,dept_id from department;
```

The results are displayed in a table:

| Name | DEPT_ID |
|--------------|---------|
| park gunwook | 10 |
| lee Davies | 50 |
| aaa uuuu | 20 |
| executive | 80 |
| chem | 45 |
| cse | - |

6 rows returned in 0.01 seconds Download

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

RESULT: Thus the sql queries using set operations are executed and verified successfully.

CREATING VIEWS

EX_NO:11

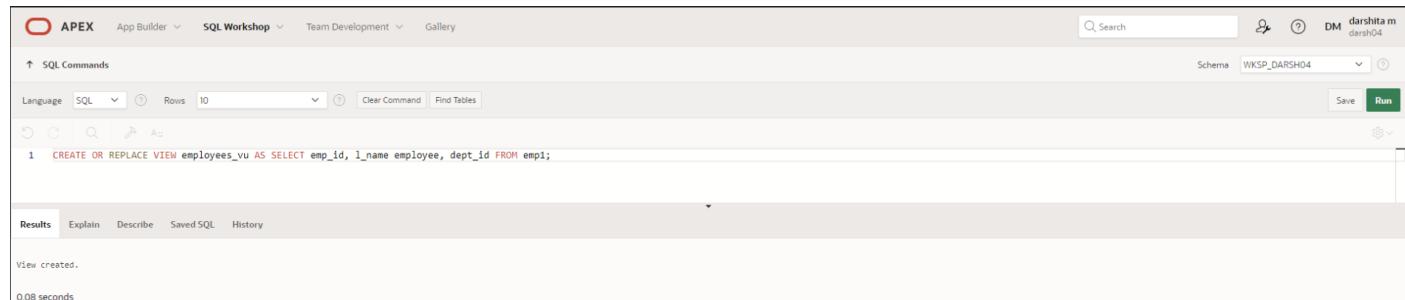
DATE:

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 OR REPLACE VIEW employees_vu AS SELECT emp_id, l_name employee, dept_id FROM emp1;
```

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The top navigation bar shows 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. On the right, it shows the user 'darshita m' and workspace 'WKSP_DARSH04'. The main area has tabs for 'SQL Commands', 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'SQL Commands' tab is active, showing the command: 'CREATE OR REPLACE VIEW employees_vu AS SELECT emp_id, l_name employee, dept_id FROM emp1;'. Below the command, the results show: 'View created.' and '0.08 seconds'. There are also icons for search, refresh, and run.

2.Display the contents of the EMPLOYEES_VU view.

QUERY:

```
select * from employees_vu;
```

OUTPUT:



A screenshot of the Oracle SQL Workshop interface, similar to the previous one but with different results. The 'Results' tab is active, showing the output of the query 'SELECT * FROM employees_vu;'. The results are displayed in a table with three columns: 'EMP_ID', 'EMPLOYEE', and 'DEPT_ID'. The data rows are: (1015, gunwook, 10), (467, Davies, 50), and (491, uuuu, 20). At the bottom, it says '3 rows returned in 0.03 seconds'.

| EMP_ID | EMPLOYEE | DEPT_ID |
|--------|----------|---------|
| 1015 | gunwook | 10 |
| 467 | Davies | 50 |
| 491 | uuuu | 20 |

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

QUERY:

SELECT view_name, text FROM user_views;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The query `SELECT view_name, text FROM user_views;` is run, and the results are displayed in a table. The table has two columns: `VIEW_NAME` and `TEXT`. The single row returned is `EMPLOYEES_VU` with the text `SELECT emp_id, l_name employee, dept_id FROM emp1`.

| VIEW_NAME | TEXT |
|--------------|---|
| EMPLOYEES_VU | SELECT emp_id, l_name employee, dept_id FROM emp1 |

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

QUERY:

SELECT employee, dept_id FROM employees_vu;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The query `SELECT employee, dept_id FROM employees_vu;` is run, and the results are displayed in a table. The table has two columns: `EMPLOYEE` and `DEPT_ID`. The three rows returned are `gunwook` (dept_id 10), `Davies` (dept_id 50), and `uuuu` (dept_id 20).

| EMPLOYEE | DEPT_ID |
|----------|---------|
| gunwook | 10 |
| Davies | 50 |
| uuuu | 20 |

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

QUERY:

```
CREATE VIEW dept50 AS SELECT emp_id empno, l_name employee, dept_id deptno FROM emp1  
WHERE dept_id = 50 WITH CHECK OPTION CONSTRAINT emp_dept_50;
```

OUTPUT:

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

```
1 CREATE VIEW dept50 AS SELECT emp_id empno, l_name employee, dept_id deptno FROM emp1 WHERE dept_id = 50 WITH CHECK OPTION CONSTRAINT emp_dept_50;
```

Below the command, the results tab is active, showing the message "View created." and a execution time of "0.06 seconds".

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

QUERY:

```
Desc dept50;
```

```
Select * from dept50;
```

OUTPUT:

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

```
1 DESCRIBE dept50;
```

Below the command, the 'Describe' tab is active, showing the structure of the DEPT50 view:

| Object Type | VIEW | Object | DEPT50 | | | | | | |
|-------------|----------|-----------|--------|-----------|-------|-------------|----------|---------|---------|
| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
| DEPT50 | EMPNO | NUMBER | - | 6 | 0 | - | ✓ | - | - |
| | EMPLOYEE | VARCHAR2 | 50 | - | - | - | ✓ | - | - |
| | DEPTNO | NUMBER | - | 6 | 0 | - | ✓ | - | - |

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area displays the SQL command for selecting all columns from the DEPT50 view:

```
1 SELECT * FROM dept50;
```

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

| EMPNO | EMPLOYEE | DEPTNO |
|-------|----------|--------|
| 467 | Davies | 50 |

At the bottom, it says "1 rows returned in 0.03 seconds".

7.) Attempt to reassign Matos to department 80

QUERY:

```
UPDATE dept50 SET deptno=80 WHERE employee='Matos';
```

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The top navigation bar shows "APEX", "App Builder", "SQL Workshop", "Team Development", and "Gallery". The right side of the header shows the schema "WKSP_DARSH04" and the user "darshita m". The main area has tabs for "SQL Commands", "Results", "Explain", "Describe", "Saved SQL", and "History". The "SQL Commands" tab is active, showing the query: "1 UPDATE dept50 SET deptno = 80 WHERE employee = 'Matos';". Below the query, the results show "0 row(s) updated." and a execution time of "0.05 seconds".

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

QUERY:

```
create or replace view salary_vu as select e.l_name "Employee",d.dept_name Department, e.salary  
"Salary",j.grade_level "Grades" from emp1 e, department d, job_grades j where e.dept_id=d.dept_id and  
e.salary between j.low_sal and j.high_sal;
```

OUTPUT:



A screenshot of the Oracle SQL Workshop interface, similar to the previous one. The "SQL Commands" tab is active, showing the creation of a view: "1 CREATE OR REPLACE VIEW salary_vu AS SELECT e.l_name "Employee", d.dept_name "Department", e.salary "Salary", j.grade_level "Grades" FROM emp1 e, department d, job_grades j 2 WHERE e.dept_id = d.dept_id AND e.salary BETWEEN j.low_sal and j.high_sal;". Below the query, the results show "View created." and an execution time of "0.09 seconds".

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

RESULT: thus the views are created in sql successfully.

EXERCISE 12

PRACTICE QUESTIONS

Intro to Constraints; NOT NULL and UNIQUE Constraints

Global Fast Foods has been very successful this past year and has opened several new stores. They need to add a table to their database to store information about each of their store's locations. The owners want to make sure that all entries have an identification number, date opened, address, and city and that no other entry in the table can have the same email address. Based on this information, answer the following questions about the global_locations table. Use the table for your answers.

| Global Fast Foods global_locations Table | | | | | | |
|--|------|--------|-----------|-------|----------|---------|
| NAME | TYPE | LENGTH | PRECISION | SCALE | NULLABLE | DEFAULT |
| Id | | | | | | |
| name | | | | | | |
| date_opened | | | | | | |
| address | | | | | | |
| city | | | | | | |
| zip/postal code | | | | | | |
| phone | | | | | | |
| email | | | | | | |
| manager_id | | | | | | |
| Emergency contact | | | | | | |

1. What is a “constraint” as it relates to data integrity?

Database can be as reliable as the data in it, and database rules are implemented as Constraint to maintain data integrity.

2. What are the limitations of constraints that may be applied at the column level and at the table level?

- Constraints referring to more than one column are defined at Table Level
- NOT NULL constraint must be defined at column level as per ANSI/ISO SQL standard.

3. Why is it important to give meaningful names to constraints?

- If a constraint is violated in a SQL statement execution, it is easy to identify the cause with user-named constraints.
- It is easy to alter names/drop constraint.

4. Based on the information provided by the owners, choose a datatype for each column. Indicate the length, precision, and scale for each NUMBER datatype.

| Global Fast Foods global_locations Table | | | | | | |
|--|------|----------|--------|-----------|-------|----------|
| NAME | TYPE | DataType | LENGTH | PRECISION | SCALE | NULLABLE |
| id | pk | NUMBER | 6 | 0 | | No |
| name | | VARCHAR2 | 50 | | | |
| date_opened | | DATE | | | | No |
| address | | VARCHAR2 | 50 | | | No |
| city | | VARCHAR2 | 30 | | | No |
| zip_postal_code | | VARCHAR2 | 12 | | | |
| phone | | VARCHAR2 | 20 | | | |
| email | uk | VARCHAR2 | 75 | | | |
| manager_id | | NUMBER | 6 | 0 | | |
| emergency_contact | | VARCHAR2 | 20 | | | |

5. Use “(nullable)” to indicate those columns that can have null values.

| Global Fast Foods global_locations Table | | | | | | |
|--|------|----------|--------|-----------|-------|----------|
| NAME | TYPE | DataType | LENGTH | PRECISION | SCALE | NULLABLE |
| id | pk | NUMBER | 6 | 0 | | No |
| name | | VARCHAR2 | 50 | | | Yes |
| date_opened | | DATE | | | | No |
| address | | VARCHAR2 | 50 | | | No |
| city | | VARCHAR2 | 30 | | | No |
| zip_postal_code | | VARCHAR2 | 12 | | | Yes |
| phone | | VARCHAR2 | 20 | | | Yes |
| email | uk | VARCHAR2 | 75 | | | Yes |
| manager_id | | NUMBER | 6 | 0 | | Yes |
| emergency_contact | | VARCHAR2 | 20 | | | Yes |

6. Write the CREATE TABLE statement for the Global Fast Foods locations table to define the constraints at the column level.

```
CREATE TABLE f_global_locations
( id NUMBER(6,0) CONSTRAINT f_gln_id_pk PRIMARY KEY ,
name VARCHAR2(50),
date_opened DATE CONSTRAINT f_gln_dt_opened_nn NOT NULL ENABLE,
address VARCHAR2(50) CONSTRAINT f_gln_add_nn NOT NULL ENABLE,
city VARCHAR2(30) CONSTRAINT f_gln_city_nn NOT NULL ENABLE,
zip_postal_code VARCHAR2(12),
phone VARCHAR2(20),
email VARCHAR2(75) CONSTRAINT f_gln_email_uk UNIQUE,
manager_id NUMBER(6,0),
emergency_contact VARCHAR2(20)
);
```

7. Execute the CREATE TABLE statement in Oracle Application Express.

Table Created.

8. Execute a DESCRIBE command to view the Table Summary information.

```
DESCRIBE f_global_locations;
```

9. Rewrite the CREATE TABLE statement for the Global Fast Foods locations table to define the UNIQUE constraints at the table level. Do not execute this statement.

| NAME | TYPE | LENGTH | PRECISION | SCALE | NULLABLE | DEFAULT |
|------------|----------|--------|-----------|-------|----------|---------|
| id | number | 4 | | | | |
| loc_name | varchar2 | 20 | | | X | |
| | date | | | | | |
| address | varchar2 | 30 | | | | |
| city | varchar2 | 20 | | | | |
| zip_postal | varchar2 | 20 | | | X | |
| phone | varchar2 | 15 | | | X | |
| email | varchar2 | 80 | | | X | |
| manager_id | number | 4 | | | X | |
| contact | varchar2 | 40 | | | X | |

```
CREATE TABLE f_global_locations
( id NUMBER(6,0) CONSTRAINT f_gln_id_pk PRIMARY KEY ,
name VARCHAR2(50),
date_opened DATE CONSTRAINT f_gln_dt_opened_nn NOT NULL ENABLE,
address VARCHAR2(50) CONSTRAINT f_gln_add_nn NOT NULL ENABLE,
city VARCHAR2(30) CONSTRAINT f_gln_city_nn NOT NULL ENABLE,
zip_postal_code VARCHAR2(12),
```

```
phone VARCHAR2(20),  
email VARCHAR2(75) ,  
manager_id NUMBER(6,0),  
emergency_contact VARCHAR2(20),  
CONSTRAINT f_gln_email_uk UNIQUE(email)  
);
```

PRIMARY KEY, FOREIGN KEY, and CHECK Constraints

1. What is the purpose of a
 - PRIMARY KEY
 - FOREIGN KEY
 - CHECK CONSTRAINT

a. PRIMARY KEY

Uniquely identify each row in table.

b. FOREIGN KEY

Referential integrity constraint links back parent table's primary/unique key to child table's column.

c. CHECK CONSTRAINT

Explicitly define condition to be met by each row's fields. This condition must be returned as true or unknown.

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.

```
CREATE TABLE animals
( animal_id NUMBER(6,0) CONSTRAINT anl_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
);
```

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

| ANIMAL_ID | NAME | LICENSE_TAG_NUM BER | ADMIT_DATE | ADOPTION_ID | VACCINATION_DATE |
|-----------|------|------------------------|-------------|-------------|------------------|
| 101 | Spot | 35540 | 10-Oct-2004 | 205 | 12-Oct-2004 |

```
INSERT INTO animals (animal_id, name, license_tag_number, admit_date, adoption_id, vaccination_date)
VALUES( 101, 'Spot', 35540, TO_DATE('10-Oct-2004', 'DD-Mon-YYYY'), 205, TO_DATE('12-Oct-2004', 'DD-Mon-YYYY'));
```

```
SELECT * FROM animals;
```

5. Write the syntax to create a foreign key (adoption_id) in the animals table that has a corresponding primary-key reference in the adoptions table. Show both the column-level and table-level syntax. Note that because you have not actually created an adoptions table, no adoption_id primary key exists, so the foreign key cannot be added to the animals table.

COLUMN LEVEL STATEMENT:

```
ALTER TABLE animals
MODIFY ( adoption_id NUMBER(5,0) CONSTRAINT anl_adopt_id_fk REFERENCES adoptions(id)
ENABLE );
```

TABLE LEVEL STATEMENT:

```
ALTER TABLE animals ADD CONSTRAINT anl_adopt_id_fk FOREIGN KEY (adoption_id)
REFERENCES adoptions(id) ENABLE;
```

6. What is the effect of setting the foreign key in the ANIMAL table as:

a. ON DELETE CASCADE
`ALTER TABLE animals
ADD CONSTRAINT anl_adopt_id_fk FOREIGN KEY (adoption_id)
REFERENCES adoptions(id) ON DELETE CASCADE ENABLE ;`

b. ON DELETE SET NULL
`ALTER TABLE animals
ADD CONSTRAINT anl_adopt_id_fk FOREIGN KEY (adoption_id)
REFERENCES adoptions(id) ON DELETE SET NULL ENABLE ;`

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

- I cannot specify check constraint for a view however in this case I could use WITH CHECK OPTION clause
- I am restricted to columns from self table and fields in self row.
- I cannot use subqueries and scalar subquery expressions.

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

RESULT: Thus constraints and keys in sql queries are executed and verified successfully.

PRACTICE PROBLEM

Managing Constraints

Using Oracle Application Express, click the SQL Workshop tab in the menu bar. Click the Object Browser and verify that you have a table named `copy_d_clients` and a table named `copy_d_events`. If you don't have these tables in your schema, create them before completing the exercises below. Here is how the original tables are related. The `d_clients` table has a primary key `client_number`. This has a primary-key constraint and it is referenced in the foreign-key constraint on the `d_events` table.

NOTE: The practice exercises use the `d_clients` and `d_events` tables in the DJs on Demand database. Students will work with copies of these two tables named `copy_d_clients` and `copy_d_events`. Make sure they have new copies of the tables (without changes made from previous exercises). Remember, tables copied using a subquery do not have the integrity constraints as established in the original tables. When using the `SELECT` statement to view the constraint name, the tablename must be all capital letters.

1. What are four functions that an `ALTER` statement can perform on constraints?

- ADD
- DROP
- ENABLE
- DISABLE

2. Since the tables are copies of the original tables, the integrity rules are not passed onto the new tables; only the column datatype definitions remain. You will need to add a PRIMARY KEY constraint to the `copy_d_clients` table. Name the primary key `copy_d_clients_pk`. What is the syntax you used to create the PRIMARY KEY constraint to the `copy_d_clients` table?

```
ALTER TABLE copy_d_clients
ADD CONSTRAINT copy_d_clt_client_number_pk PRIMARY KEY (client_number);
```

3. Create a FOREIGN KEY constraint in the `copy_d_events` table. Name the foreign key `copy_d_events_fk`. This key references the `copy_d_clients` table `client_number` column. What is the syntax you used to create the FOREIGN KEY constraint in the `copy_d_events` table?

```
ALTER TABLE copy_d_events
ADD CONSTRAINT copy_d_eve_client_number_fk FOREIGN KEY (client_number) REFERENCES
copy_d_clients (client_number) ENABLE;
```

4. Use a `SELECT` statement to verify the constraint names for each of the tables. Note that the tablename must be capitalized.

```
SELECT constraint_name, constraint_type, table_name
FROM user_constraints
WHERE table_name = UPPER('copy_d_events');
```

a. The constraint name for the primary key in the `copy_d_clients` table is _____.

COPY_D_CLT_CLIENT_NUMBER_PK

5. Drop the PRIMARY KEY constraint on the copy_d_clients table. Explain your results.

```
ALTER TABLE copy_d_clients
DROP CONSTRAINT COPY_D_CLT_CLIENT_NUMBER_PK CASCADE ;
```

6. Add the following event to the copy_d_events table. Explain your results.

| ID | NAME | EVENT_DATE | DESCRIPTION | COST | VENUE_ID | PACKAGE_CODE | THEME_CODE | CLIENT_NUMBER |
|-----|-------------------|-------------|--------------------------------|------|----------|--------------|------------|---------------|
| 140 | Cline Bas Mitzvah | 15-Jul-2004 | Church and Private Home formal | 4500 | 105 | 87 | 77 | 7125 |

```
INSERT INTO copy_d_events(client_number,id,name,event_date,description,cost,venue_id,package_code,theme_code)
VALUES(7125,140,'Cline Bas Mitzvah',TO_DATE('15-Jul-2004','dd-Mon-yyyy'),'Church and Private Home formal',4500,105,87,77);
```

RESULT: ORA-02291: integrity constraint (HKUMAR.COPY_D_EVE_CLIENT_NUMBER_FK) violated - parent key not found

7. Create an ALTER TABLE query to disable the primary key in the copy_d_clients table. Then add the values from #6 to the copy_d_events table. Explain your results.

```
ALTER TABLE copy_d_clients
DISABLE CONSTRAINT COPY_D_CLT_CLIENT_NUMBER_PK CASCADE;
```

8. Repeat question 6: Insert the new values in the copy_d_events table. Explain your results.

```
INSERT INTO
copy_d_events(client_number,id,name,event_date,description,cost,venue_id,package_code,theme_code)
VALUES(7125,140,'Cline Bas Mitzvah',TO_DATE('15-Jul-2004','dd-Mon-yyyy'),'Church and Private Home formal',4500,105,87,77);
```

1 row(s) inserted.

9. Enable the primary-key constraint in the copy_d_clients table. Explain your results.

```
ALTER TABLE copy_d_clients
ENABLE CONSTRAINT COPY_D_CLT_CLIENT_NUMBER_PK ;
```

10. If you wanted to enable the foreign-key column and reestablish the referential integrity between these two tables, what must be done?

```
DELETE FROM copy_d_events WHERE
client_number NOT IN ( SELECT client_number FROM copy_d_clients);
```

1 row(s) deleted.

```
ALTER TABLE copy_d_events  
ENABLE CONSTRAINT COPY_D_EVE_CLIENT_NUMBER_FK;
```

Table altered.

11. Why might you want to disable and then re-enable a constraint?

Generally to make bulk operations fast, where my input data is diligently sanitized and I am sure, it is safe to save some time in this clumsy process.

12. Query the data dictionary for some of the constraints that you have created. How does the data dictionary identify each constraint type?

Queries are same as in point 2,3, 4 above.

- C - Check constraint
 - Sub-case - if I see SEARCH_CONDITION something like "FIRST_NAME" IS NOT NULL , its a NOT NULL constraint.
- P - Primary key
- R - Referential integrity (fk)
 - U - Unique key

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

RESULT: Thus the constraints are managed in sql queries successfully.

EXERCISE 13

Creating Views

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';
```

3. SELECT * FROM view_d_songs. What was returned?

The screenshot shows a database query results interface. At the top, there are tabs: 'Results' (which is selected), 'Explain', 'Describe', 'Saved SQL', and 'History'. Below the tabs is a table with three columns: 'ID', 'Song Title', and 'ARTIST'. The first row has an ID of 47, a Song Title of 'Hurrah for Today', and an ARTIST of 'The Jubilant Trio'. The second row has an ID of 49, a Song Title of 'Lets Celebrate', and an ARTIST of 'The Celebrants'. At the bottom left, it says '2 rows returned in 0.00 seconds'. At the bottom right, there is a 'Download' link.

| ID | Song Title | ARTIST |
|----|------------------|-------------------|
| 47 | Hurrah for Today | The Jubilant Trio |
| 49 | Lets Celebrate | The Celebrants |

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';
```

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);
```

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);
```

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';
```

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;
```

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);
```

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;
```

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;
```

7. 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';
```

8. 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;
```

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

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

10. 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;
```

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

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;
```

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;
```

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

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

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);
```

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';
```

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';
```

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;
```

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));
```

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;
```

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

10. Drop the synonym that you created in question

```
DROP SYNONYM dj_tracks2;
```

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

RESULT: Thus the views are created successfully.

OTHER DATABASE OBJECTS

EX_NO:14

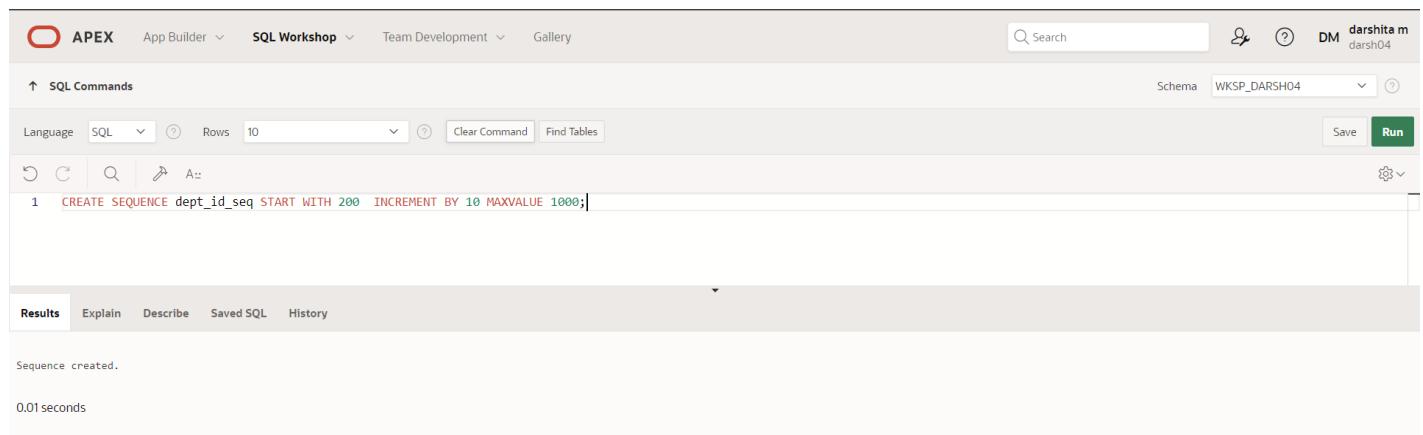
DATE:

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:



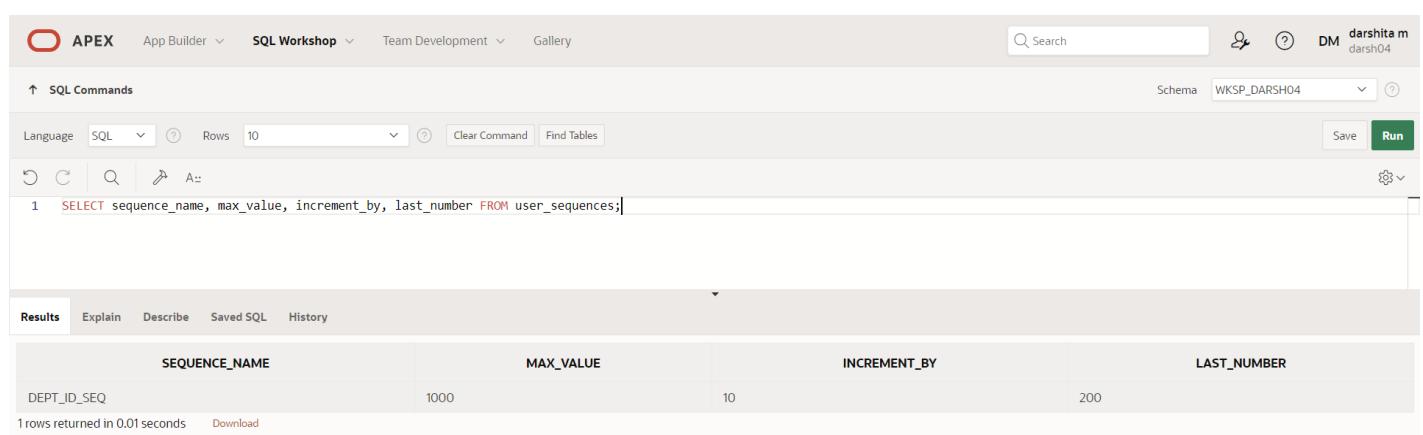
The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A single line of SQL code is entered in the command input field: 'CREATE SEQUENCE dept_id_seq START WITH 200 INCREMENT BY 10 MAXVALUE 1000;'. Below the input field, the results pane displays the message 'Sequence created.' and a execution time of '0.01 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. 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: 'SELECT sequence_name, max_value, increment_by, last_number FROM user_sequences;'. Below the input field, the results pane displays a table with four columns: SEQUENCE_NAME, MAX_VALUE, INCREMENT_BY, and LAST_NUMBER. The table contains one row for 'DEPT_ID_SEQ' with values 1000, 10, and 200 respectively. 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');
```

OUTPUT:

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 search bar, a user icon, and a 'DM' role indicator for 'darshita m darsh04'. The main area displays the results of a script named 'ex14'. The status is 'Complete'. The summary table shows one row inserted with an elapsed time of 0.01 seconds. Below the table, it says '1 row(s) inserted.' and 'row(s) 1 - 1 of 1'. At the bottom, performance metrics are shown: 1 statement processed, 1 successful, and 0 with errors.

| Number | Elapsed | Statement | Feedback | Rows |
|--------|---------|--|----------|--------------------|
| 1 | 0.01 | INSERT INTO job_history VALUES (dept_id_seq.nextval, 'Educat | | 1 row(s) inserted. |

Download

1
0.01
1 row(s) 1 - 1 of 1

1
Successful
0 With Errors

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

QUERY:

```
CREATE INDEX emp_dept_id_idx ON emp1 (dept_id);
```

OUTPUT:

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 search bar, a schema dropdown set to 'WKSP_DARSH04', and a 'Run' button. The main area shows the execution of a SQL command to create an index. The command is: 'CREATE INDEX emp_dept_id_idx ON emp1 (dept_id);'. Below the command, the results show 'Index created.' and a execution time of '0.04 seconds'.

Language: SQL Rows: 10

```
CREATE INDEX emp_dept_id_idx ON emp1 (dept_id);
```

Results Explain Describe Saved SQL History

Index created.
0.04 seconds

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:



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 information (darshita m, darsh04) are also present. The main area is titled 'SQL Commands' with a 'Language' dropdown set to 'SQL'. The query 'SELECT index_name,table_name,uniqueness FROM user_indexes WHERE table_name='EMPLOYEES';' is entered in the command field. Below the command are tabs for Results, Explain, Describe, Saved SQL, and History. The results section displays the message 'no data found'.

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

RESULT: Thus the sequences are executed in sql queries successfully.

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</u> <u>(5)</u> | |
| <u>Viva(5)</u> | |
| <u>Total (10)</u> | |
| <u>Faculty Signature</u> | |

RESULT: thus the sql queries on controlling user access is executed and verified successfully.

PL/SQL CONTROL STRUCTURES

EX_NO: 16

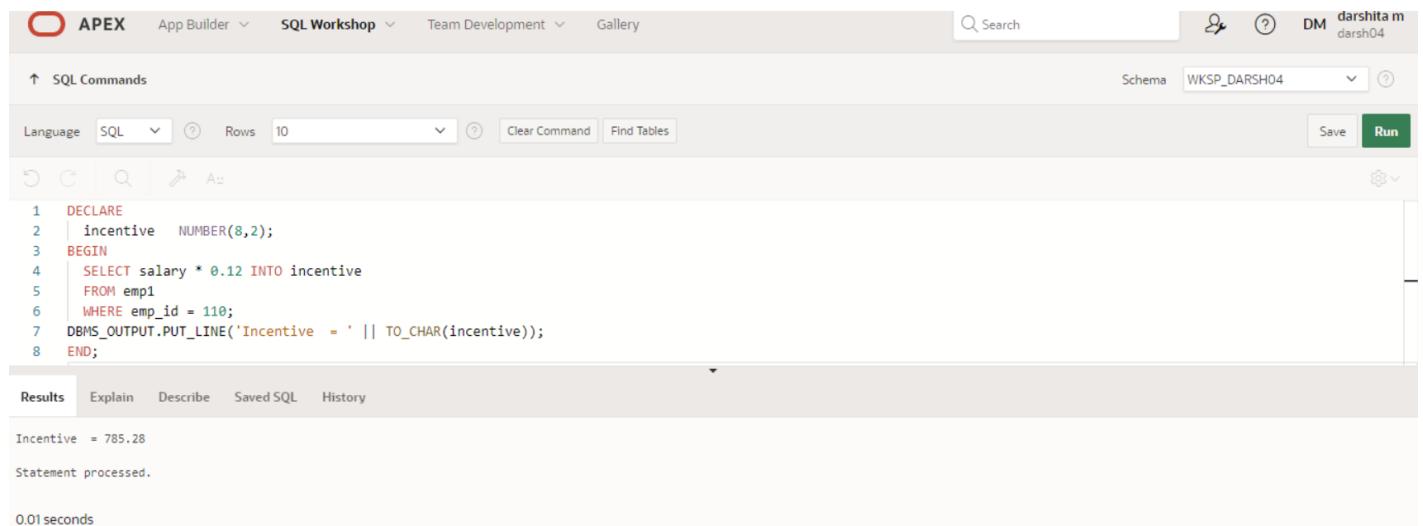
DATE:

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 top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user 'darshita m' and schema 'WKSP_DARSH04'. The main area is titled 'SQL Commands' with a language dropdown set to 'SQL'. The code editor contains the provided PL/SQL block. The results tab shows the output: 'Incentive = 785.28', 'Statement processed.', and '0.01 seconds'.

```
1 DECLARE
2     incentive NUMBER(8,2);
3 BEGIN
4     SELECT salary * 0.12 INTO incentive
5     FROM emp1
6     WHERE emp_id = 110;
7     DBMS_OUTPUT.PUT_LINE('Incentive = ' || TO_CHAR(incentive));
8 END;
```

Incentive = 785.28
Statement processed.
0.01 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;
/
```

OUTPUT:

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

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

```
1 DECLARE
2   "WELCOME" varchar2(10) := 'welcome';
3 BEGIN
4   DBMS_Output.Put_Line("Welcome");
5 END;
6 /
```

In the 'Results' tab, the output shows an error message:

```
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,

 addless NUMBER

) IS

BEGIN

 empsal := empsal + addless;

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 SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows the schema as WKSP_DARSH04 and the user as darshtam_darsh04. The main area is titled 'SQL Commands' and contains the following PL/SQL code:

```
1  DECLARE salary_of_emp NUMBER(8,2);
2  PROCEDURE approx_salary ( emp      NUMBER, empsal IN OUT NUMBER, addless   NUMBER) IS
3  BEGIN
4      empsal := empsal + addless;
5  END;
6  BEGIN
7      SELECT salary INTO salary_of_emp
8      FROM emp1
9      WHERE emp_id = 122;
10     DBMS_OUTPUT.PUT_LINE
11         ('Before invoking procedure, salary_of_emp: ' || salary_of_emp);
12     approx_salary (100, salary_of_emp, 1000);
13     DBMS_OUTPUT.PUT_LINE
14         ('After invoking procedure, salary_of_emp: ' || salary_of_emp);
15 END;
```

The 'Results' tab at the bottom shows the output of the code execution:

```
Before invoking procedure, salary_of_emp: 81265
After invoking procedure, salary_of_emp: 82265
Statement processed.
```

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 SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area displays the SQL command for creating the 'pri_bool' procedure. The code is as follows:

```
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
11        DBMS_OUTPUT.PUT_LINE(boo_name || ' = FALSE');
12    END IF;
13 END;
```

Below the code, the 'Results' tab is selected, showing the output of the procedure execution. The output includes the procedure creation message and a detailed breakdown of the execution paths for different input values of 'boo_val' (TRUE, NULL, FALSE).

Procedure created.

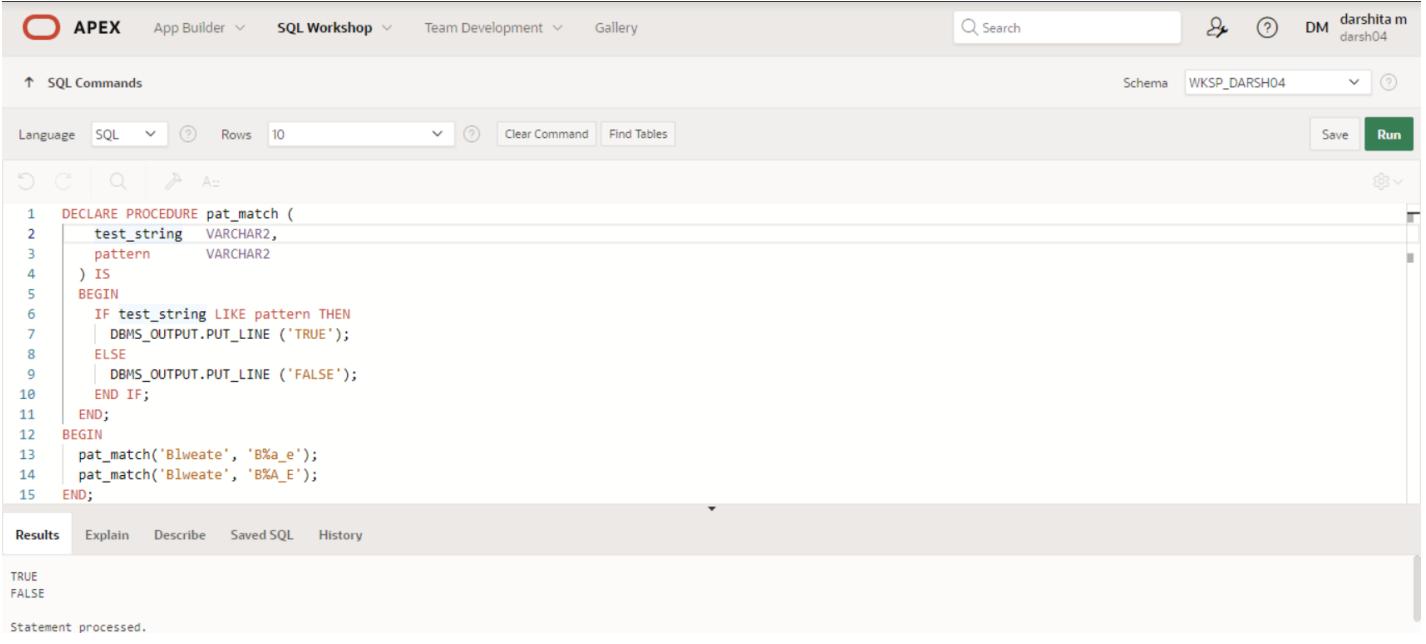
| Results | Explain | Describe | Saved SQL | History |
|---|---------|----------|-----------|---------|
| <pre>m = TRUE m AND n = TRUE ----- FOR m TRUE AND n NULL ----- m = TRUE n = NULL m AND n = NULL ----- FOR m FALSE AND n NULL----- m = FALSE n = NULL m AND n = FALSE ----- FOR m NULL AND n TRUE ----- m = NULL n = TRUE m AND n = NULL ----- FOR m NULL AND n FALSE ----- m = NULL n = FALSE m AND n = FALSE Statement processed.</pre> <p>0.01 seconds</p> | | | | |

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;
BEGIN
  pat_match('Blweate', 'B%a_e');
  pat_match('Blweate', 'B%A_E');
END;
/
```

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 'darshita m' and 'darsh04'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP_DARSH04'. The code editor contains the provided PL/SQL block. The results tab at the bottom shows the output: 'TRUE' and 'FALSE', indicating the procedure's execution results for the two pattern comparisons.

```
1  DECLARE PROCEDURE pat_match (
2    test_string  VARCHAR2,
3    pattern      VARCHAR2
4  ) IS
5  BEGIN
6    IF test_string LIKE pattern THEN
7      DBMS_OUTPUT.PUT_LINE ('TRUE');
8    ELSE
9      DBMS_OUTPUT.PUT_LINE ('FALSE');
10   END IF;
11 END;
12 BEGIN
13   pat_match('Blweate', 'B%a_e');
14   pat_match('Blweate', 'B%A_E');
15 END;
```

Results Explain Describe Saved SQL History

TRUE
FALSE

Statement processed.

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

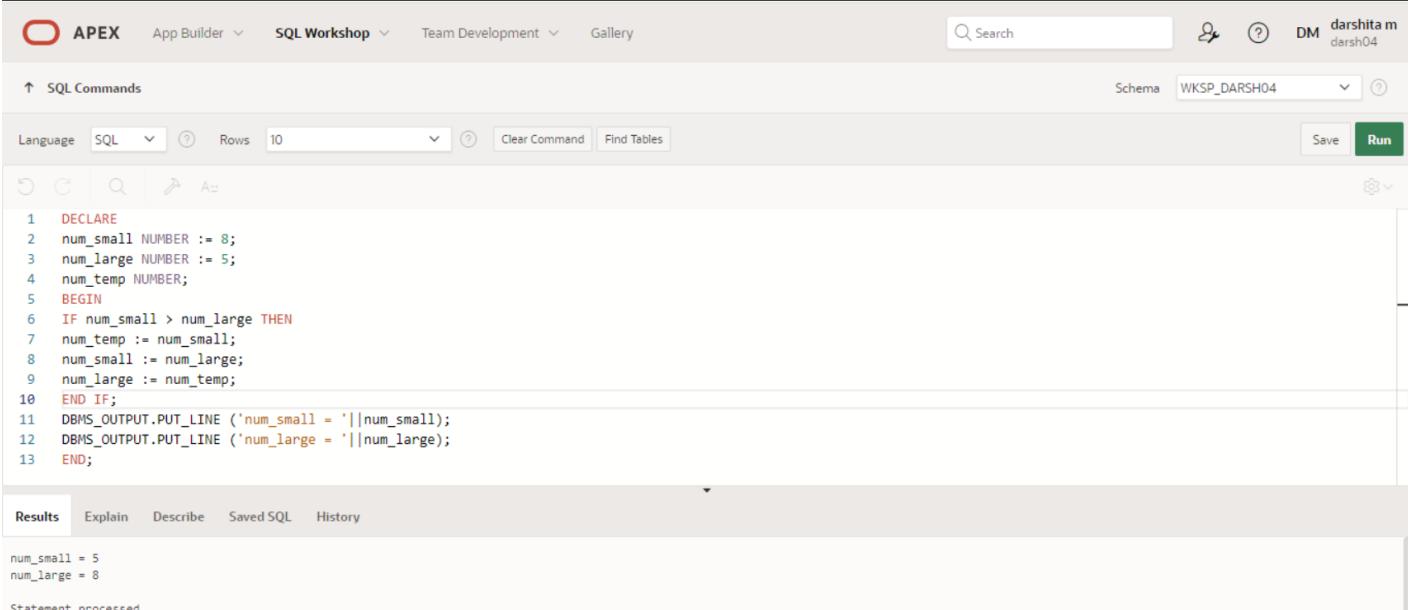
QUERY:

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:



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 the user 'darshita m' and schema 'WKSP_DARSH04'. The main area is titled 'SQL Commands' with a language dropdown set to 'SQL'. The code area contains a numbered PL/SQL block:

```
1  DECLARE  
2  num_small NUMBER := 8;  
3  num_large NUMBER := 5;  
4  num_temp NUMBER;  
5  BEGIN  
6  IF num_small > num_large THEN  
7  num_temp := num_small;  
8  num_small := num_large;  
9  num_large := num_temp;  
10 END IF;  
11 DBMS_OUTPUT.PUT_LINE ('num_small ='||num_small);  
12 DBMS_OUTPUT.PUT_LINE ('num_large ='||num_large);  
13 END;
```

The 'Results' tab is selected at the bottom, showing the output:

```
num_small = 5  
num_large = 8  
Statement processed.
```

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 SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user 'darshita m' and workspace 'darsh04'. The main area is titled 'SQL Commands' with a language dropdown set to 'SQL'. The code editor contains the following PL/SQL procedure:

```
1 DECLARE PROCEDURE test1 (sal_achieve NUMBER,target_qty NUMBER,emp_id NUMBER)
2 IS incentive NUMBER := 0;
3 updated VARCHAR2(3) := 'No';
4 BEGIN IF sal_achieve > (target_qty + 200) THEN incentive := (sal_achieve - target_qty)/4;
5 UPDATE emp1
6 SET salary = salary + incentive
7 WHERE emp_id = emp_id;
8 updated := 'Yes';
9 END IF;
10 DBMS_OUTPUT.PUT_LINE ('Table updated? ' || updated || ', ' || 'incentive = ' || incentive || '.');
11 END test1;
12 BEGIN test1(2300, 2000, 144); test1(3600, 3000, 145);
13 END;
```

The results tab shows the output of the procedure execution:

```
Table updated? Yes, incentive = 75.
Table updated? Yes, incentive = 150.

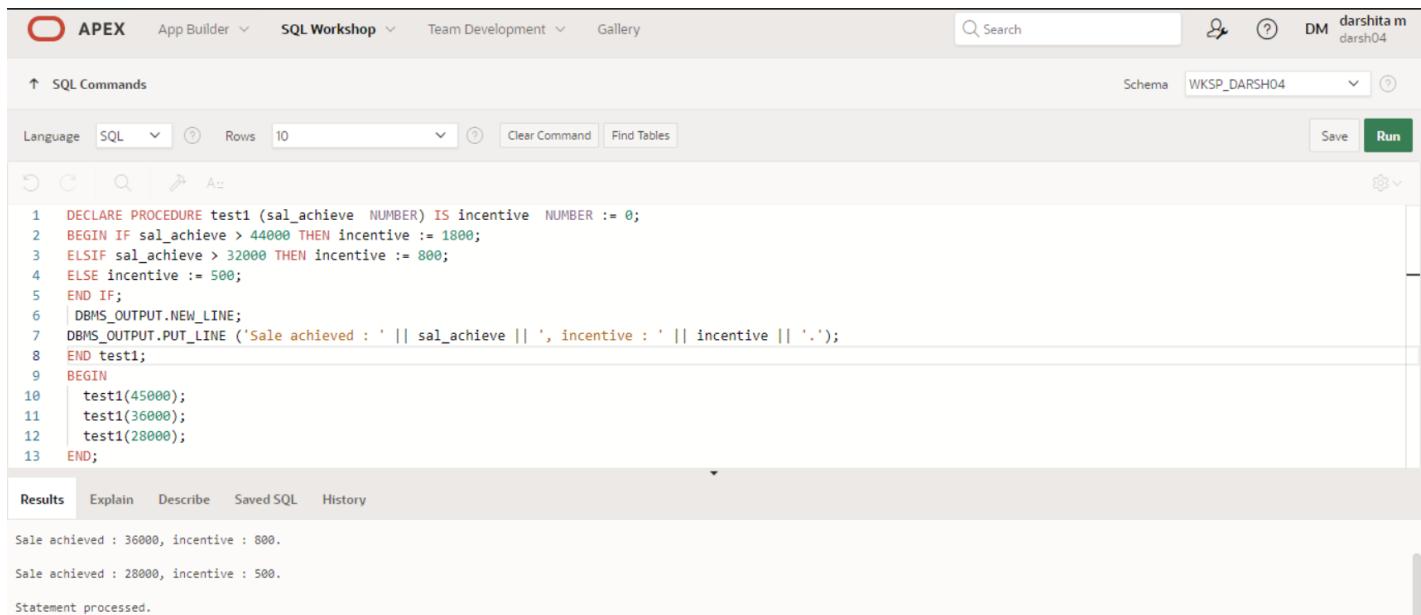
1 row(s) updated.
```

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 SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user 'darshita m' and schema 'WKSP_DARSH04'. The main area displays the PL/SQL code for the 'test1' procedure and its execution. The code defines a procedure that calculates an incentive based on a salary achievement and prints the result. It is then called from a BEGIN block with three different salary values: 45000, 36000, and 28000. The output window at the bottom shows the results for each call.

```
1  DECLARE PROCEDURE test1 (sal_achieve NUMBER) IS incentive NUMBER := 0;
2  BEGIN IF sal_achieve > 44000 THEN incentive := 1800;
3  ELSIF sal_achieve > 32000 THEN incentive := 800;
4  ELSE incentive := 500;
5  END IF;
6  |DBMS_OUTPUT.NEW_LINE;
7  DBMS_OUTPUT.PUT_LINE ('Sale achieved : ' || sal_achieve || ', incentive : ' || incentive || '.');
8  END test1;
9  BEGIN
10  | test1(45000);
11  | test1(36000);
12  | test1(28000);
13  END;
```

Results Explain Describe Saved SQL History

```
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:

```
APEX App Builder SQL Workshop Team Development Gallery Search DM darshita m darsh04  
Schema WKSP_DARSH04 Run  
SQL Commands Language SQL Rows 10 Clear Command Find Tables  
SQL Commands Language SQL Rows 10 Clear Command Find Tables Schema WKSP_DARSH04 Run  
1 DECLARE  
2     tot_emp NUMBER;  
3 BEGIN SELECT Count(*) INTO tot_emp FROM employees e join department d ON e.dept_id = d.dept_id WHERE e.dept_id = 50;  
4     dbms_output.Put_line ('The employees are in the department 50: '||To_char(tot_emp));  
5     IF tot_emp >= 45 THEN  
6         dbms_output.Put_line ('There are no vacancies in the department 50.');
```

The employees are in the department 50: 0
There are some vacancies in department 50.
Statement processed.
0.03 seconds

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 SQL Workshop interface. At the top, there are tabs for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'BHANU PRIYA bhanu23', and a 'Run' button. Below the tabs, the 'SQL Commands' section is selected. The code area contains a PL/SQL block:

```
10  ON emp WHERE e.department_id = get_dep_id;
11  WHERE e.department_id = get_dep_id;
12  dbms_output.Put_line ('The employees are in the department'||get_dep_id||' is: '
13  ||To_char(tot_emp));
14  IF tot_emp >= 45 THEN
15  dbms_output.Put_line ('There are no vacancies in the department'||get_dep_id);
16  ELSE
17  dbms_output.Put_line ('There are'||To_char(45-tot_emp)||' vacancies in department'|||
18  get_dep_id );
19  END IF;
20  END;
21 /
22
```

Below the code, the 'Results' tab is active, showing the output of the executed code:

```
The employees are in the department 80 is: 6
There are 39 vacancies in department 80

Statement processed.

0.01 seconds
```

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 SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows the user 'Bhanu Priya' with the schema 'WKSP_BHANU23'. The main area has tabs for 'SQL Commands' and 'Results'. The 'Results' tab is selected, displaying the output of a PL/SQL block. The code block is as follows:

```
1  DECLARE
2    v_employee_id employees.employee_id%TYPE;
3    v_full_name employees.first_name%TYPE;
4    v_job_id employees.job_id%TYPE;
5    v_hire_date employees.hire_date%TYPE;
6    v_salary employees.salary%TYPE;
7    CURSOR c_employees IS
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;

```

The output section shows the results of the query:

| Employee ID | Full Name | Job Title | Hire Date | Salary |
|-------------|----------------|------------|------------|--------|
| 2 | fleming Janu | ac_account | 03/01/1998 | 60000 |
| 1 | Stephen davies | sales_rep | 02/23/1996 | 50000 |
| 4 | Chris Jones | ac_account | 05/01/1998 | 48000 |
| 5 | Cameron Brown | st_clerk | 04/22/1997 | 80000 |
| 3 | john Williams | hr_rep | 02/19/1994 | 20000 |

Statement processed.

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;
7 BEGIN
8   OPEN emp_cursor;
9   FETCH emp_cursor INTO emp_record;
10  WHILE emp_cursor%FOUND LOOP
11    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp_record.employee_id);
12    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_record.first_name);
13    DBMS_OUTPUT.PUT_LINE('Manager Name: ' || emp_record.manager_name);
14    DBMS_OUTPUT.PUT_LINE('-----');
15  END LOOP;
16  CLOSE emp_cursor;
17 END;
18 /
```

The screenshot shows the Oracle SQL Workshop interface with the code entered into the SQL Commands pane. The Results pane displays the output of the PL/SQL block, listing three employees with their respective manager names. The output is as follows:

```
Employee ID: 4
Employee Name: Chris
Manager Name:
-----
Employee ID: 2
Employee Name: Fleming
Manager Name:
-----
Employee ID: 5
Employee Name: Cameron
Manager Name:
```

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

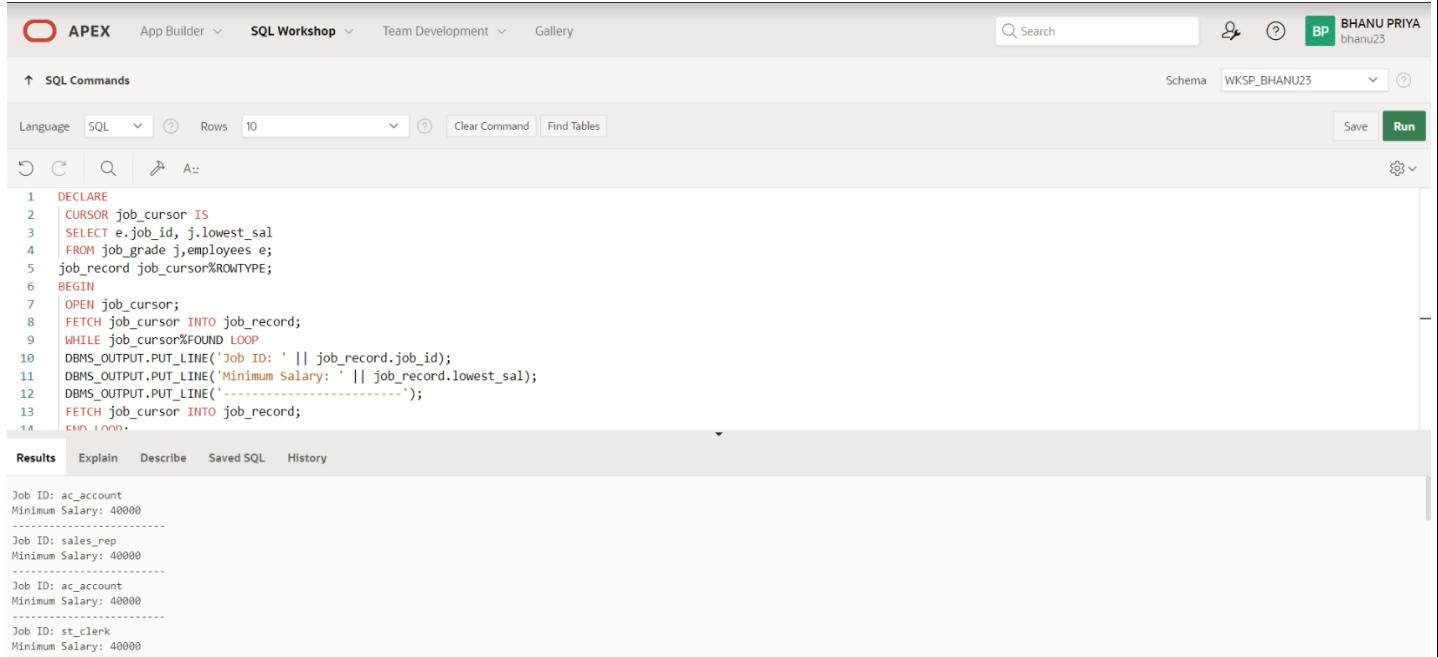
QUERY:

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. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The top right shows the schema 'WKSP_BHANU25' and a user icon 'BHANU PRIYA bhanu25'. The main area has tabs for 'SQL Commands' (selected) and 'Results'. The SQL Commands tab contains the provided PL/SQL code. The Results tab displays the output of the code execution, which lists four job entries with their IDs and minimum salaries:

| Job ID | Minimum Salary |
|------------|----------------|
| ac_account | 40000 |
| sales_rep | 40000 |
| ac_account | 40000 |
| st_clerk | 40000 |

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 SQL Workshop interface. At the top, there are tabs for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there is a search bar, user information for 'Bhanu Priya bhanu23', and a 'Run' button.

In the main area, under 'SQL Commands', a PL/SQL block is displayed:

```
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)
8  ||'Start Date');
9  dbms_output.put_line('.');
10 FOR emp_sal_rec IN employees_cur LOOP
11 -- find out most recent end_date in job_history
12   SELECT Max(end_date) + 1
13   INTO emp_start_date
14   FROM job_history;
```

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

| Employee ID | Last Name | Job Id | Start Date |
|-------------|-----------|------------|-------------|
| 2 | Janu | ac_account | 22-sep-1995 |
| 1 | davies | sales_rep | 16-mar-1997 |

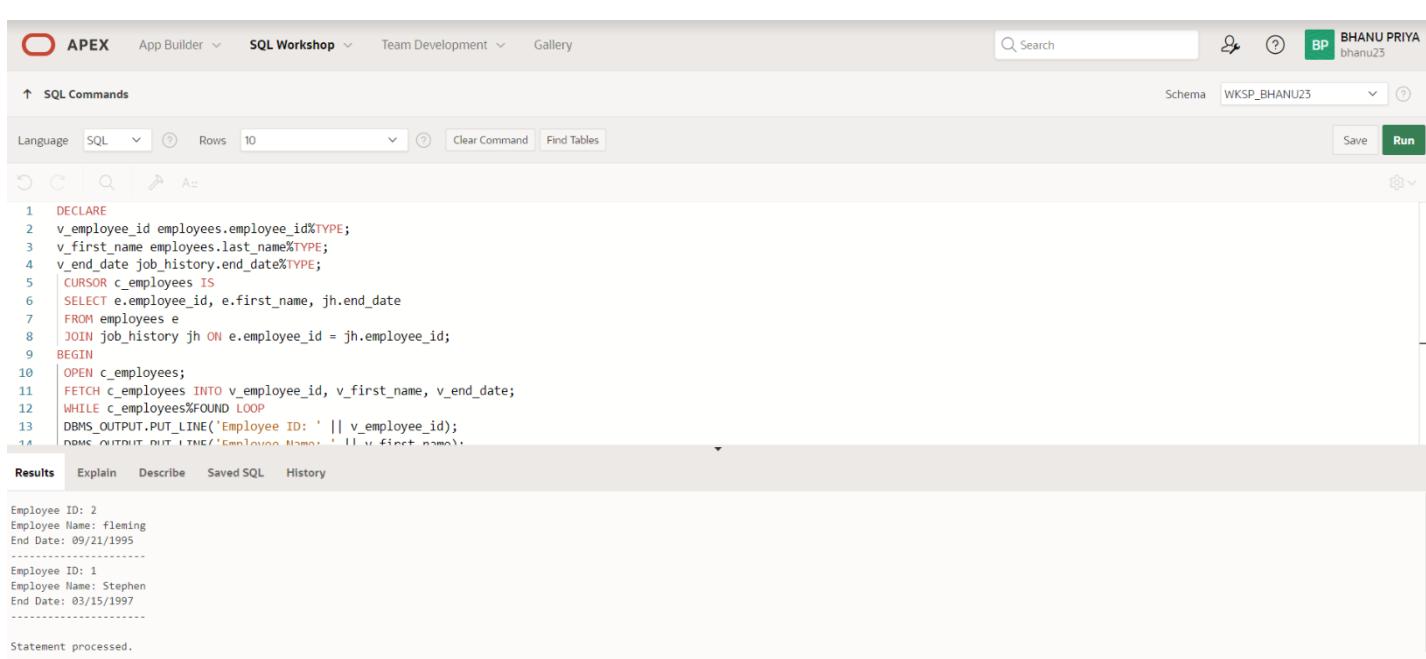
At the bottom left, it says 'Statement processed.'

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:



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 'Bhanu Priya' with the schema 'WKSP_BHANU23'. The main area is titled 'SQL Commands' with a 'Run' button. Below the toolbar, there are buttons for 'Language' (set to 'SQL'), 'Rows' (set to 10), and 'Clear Command'. The code area contains the PL/SQL block from the previous text. The 'Results' tab is selected at the bottom, showing the output of the program. The output displays two sets of employee information, separated by a double-dash separator ('-----').

```
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      WHILE c_employees%FOUND LOOP
13        DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_employee_id);
14        DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_first_name);
15        DBMS_OUTPUT.PUT_LINE('End Date: ' || v_end_date);
16        DBMS_OUTPUT.PUT_LINE('-----');
```

| Employee ID | Employee Name | End Date |
|-------------|---------------|------------|
| 2 | fleming | 09/21/1995 |
| ----- | | |
| 1 | Stephen | 03/15/1997 |
| ----- | | |

Statement processed.

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

RESULT: thus the pl/sql control structure statements are executed successfully.

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 SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user's name, darshita m, and workspace, WKSP_DARSH04. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the following PL/SQL code:

```
1 declare
2     fac number :=1;
3     n number := 5;
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;
11
```

The Results tab shows the output: "Statement processed." and "0.00 seconds".

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:

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

```

1 CREATE OR REPLACE PROCEDURE get_book_info (
2     p_book_id IN NUMBER,
3     p_title IN OUT VARCHAR2,
4     p_author OUT VARCHAR2,
5     p_year_published OUT NUMBER
6 )
7 AS
8 BEGIN
9     SELECT title, author, year_published INTO p_title, p_author, p_year_published
10    FROM books
11   WHERE book_id = p_book_id;
12
13    p_title := p_title || ' - Retrieved';
14 EXCEPTION
15    WHEN NO_DATA_FOUND THEN
16        p_title := NULL;
17        p_author := NULL;
18        p_year_published := NULL;
19 END;

```

Below the code, the 'Results' tab is active, showing the message "Procedure created.".

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

RESULT: thus the pl/sql statements are executed successfully.

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 SQL Workshop interface with the following details:

- Toolbar:** Includes APEX, App Builder, SQL Workshop, Team Development, and Gallery.
- Search Bar:** Search field and user 'darshita04'.
- Schema:** WKSP_DARSH04.
- Run Button:** Green button labeled 'Run'.
- SQL Commands Tab:** Selected tab.
- Language:** SQL.
- Rows:** 10.
- Buttons:** Clear Command, Find Tables.
- Code Area:** Displays the PL/SQL code for the trigger.
- Results Tab:** Shows the output 'Trigger created.' and execution time '0.04 seconds'.

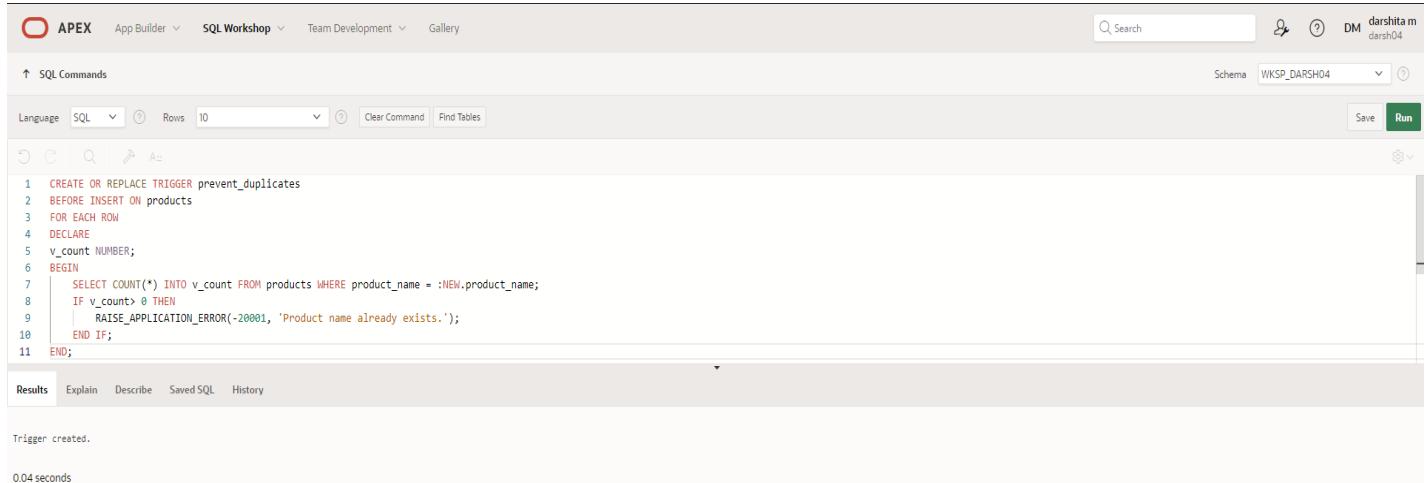
```
1 CREATE OR REPLACE TRIGGER prevent_parent_deletion
2 BEFORE DELETE ON departments
3 FOR EACH ROW DECLARE
4     v_count NUMBER;
5 BEGIN
6     SELECT COUNT(*) INTO v_count FROM employees WHERE department_id = :OLD.department_id;
7     IF v_count > 0 THEN
8         RAISE_APPLICATION_ERROR(-20001, 'Cannot delete department with associated employees.');
9     END IF;
10 END;
```

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 SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user's name (darshita m dars04) and the schema (WKSP_DARSH04). The main area displays the SQL code for creating a trigger named 'check_duplicates'. The code uses a cursor variable 'v_count' to count rows where the 'unique_col' matches the new value. If the count is greater than zero, it raises the 'duplicate_found' exception. An exception block handles this raise by calling 'RAISE_APPLICATION_ERROR' with the message 'Duplicate value found in unique_col.'. The bottom status bar indicates 'Trigger created.' and a execution time of '0.04 seconds'.

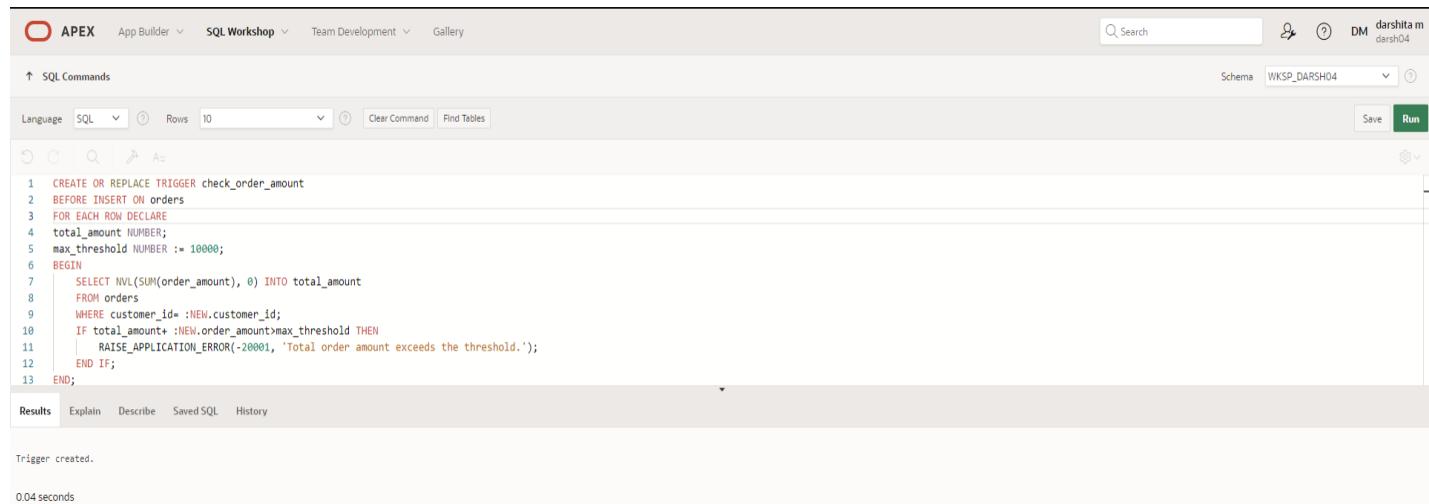
```
1 CREATE OR REPLACE TRIGGER prevent_duplicates
2 BEFORE INSERT ON products
3 FOR EACH ROW
4 DECLARE
5     v_count NUMBER;
6 BEGIN
7     SELECT COUNT(*) INTO v_count FROM products WHERE product_name = :NEW.product_name;
8     IF v_count > 0 THEN
9         RAISE_APPLICATION_ERROR(-20001, 'Product name already exists.');
10    END IF;
11 END;
```

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 SQL Workshop interface with the following details:

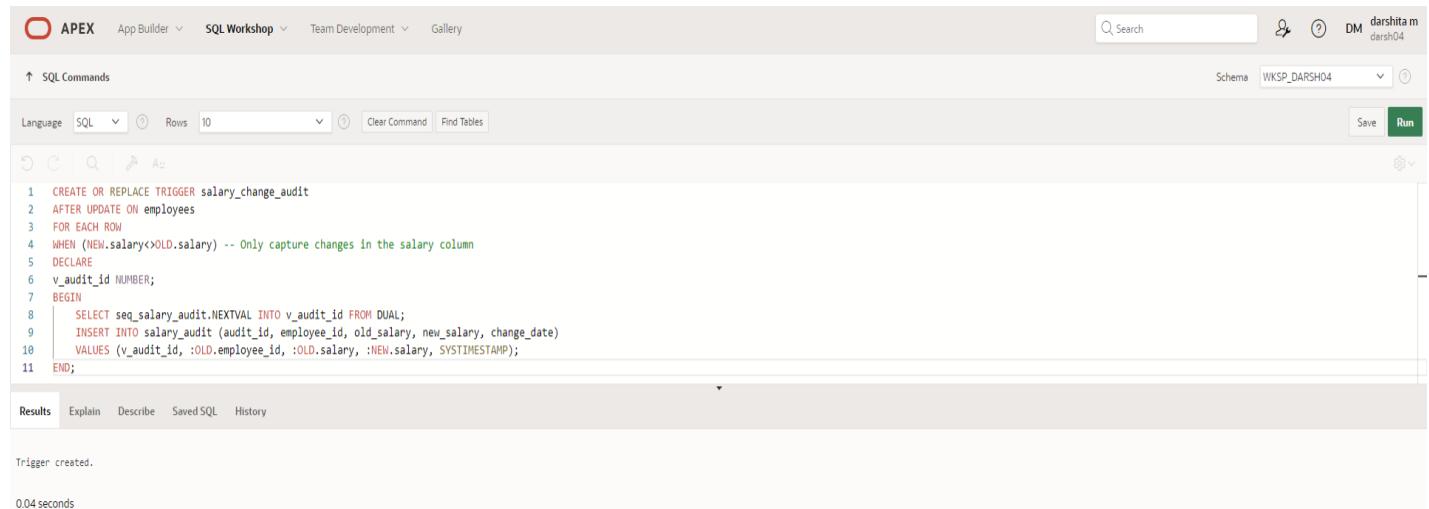
- Header:** APEX, App Builder, SQL Workshop (selected), Team Development, Gallery.
- User:** darsita m, Schema: WKSP_DARSH04.
- Toolbar:** Search, Run, Save.
- Query Editor:** Language: SQL, Rows: 10, Clear Command, Find Tables.
- Code:** The PL/SQL code for the trigger is displayed in the editor.
- Results Tab:** Shows the output: "Trigger created." and "0.04 seconds".

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 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, and a schema dropdown set to WKSP_DARSH04. The main workspace is titled "SQL Commands". It shows a code editor with the following PL/SQL code:

```
1 CREATE OR REPLACE TRIGGER salary_change_audit
2 AFTER UPDATE ON employees
3 FOR EACH ROW
4 WHEN (NEW.salary <> OLD.salary) -- Only capture changes in the salary column
5 DECLARE
6 v_audit_id NUMBER;
7 BEGIN
8     SELECT seq_salary_audit.NEXTVAL INTO v_audit_id FROM DUAL;
9     INSERT INTO salary_audit (audit_id, employee_id, old_salary, new_salary, change_date)
10    VALUES (v_audit_id, :OLD.employee_id, :OLD.salary, :NEW.salary, SYSTIMESTAMP);
11 END;
```

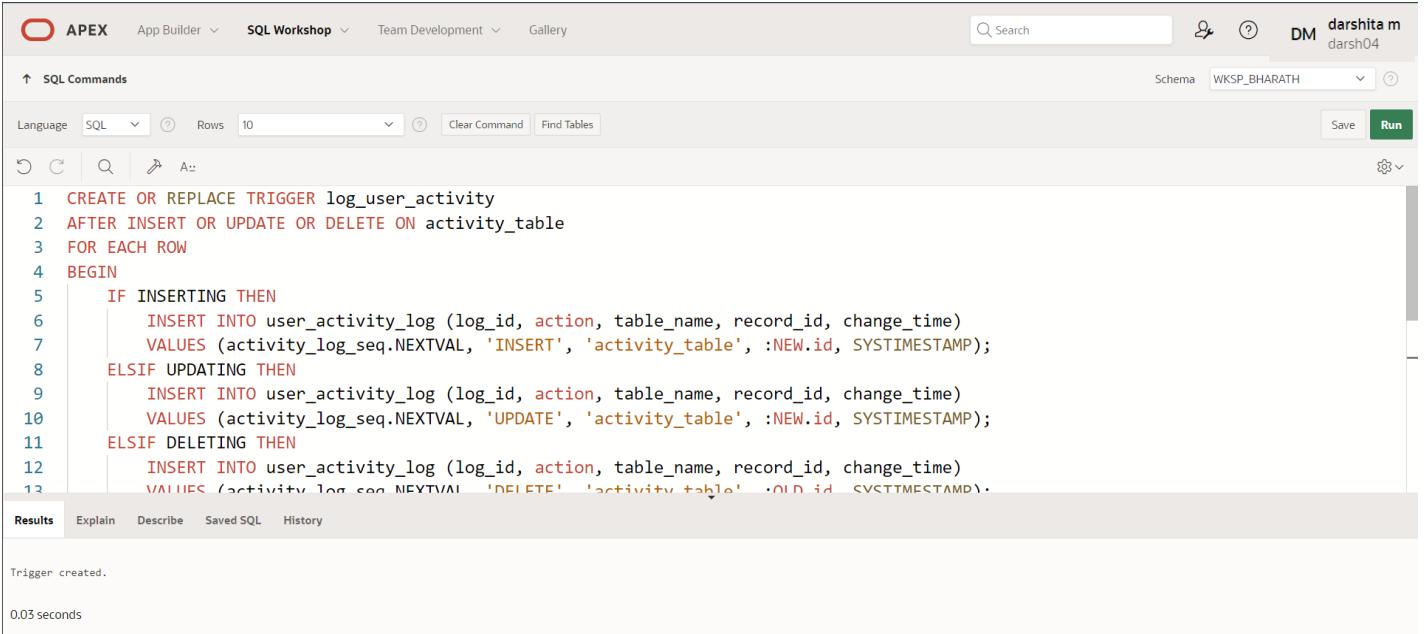
Below the code editor, there are tabs for Results, Explain, Describe, Saved SQL, and History. The status bar at the bottom indicates "Trigger created." and "0.04 seconds".

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 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 'darshita m', and a schema dropdown set to 'WKSP_BHARATH'. The main area is titled 'SQL Commands' with a language dropdown set to 'SQL'. Below the title are buttons for Undo, Redo, Find, and Run. The SQL code for the trigger is pasted into the command window. The code is color-coded: red for numbers and keywords like CREATE, OR, REPLACE, TRIGGER, AFTER, etc.; orange for identifiers like log_user_activity, user_activity_log, activity_log_seq, etc.; and black for comments and values. The code itself is identical to the one provided in the question. At the bottom, the 'Results' tab is selected, showing the output: 'Trigger created.' and '0.03 seconds'.

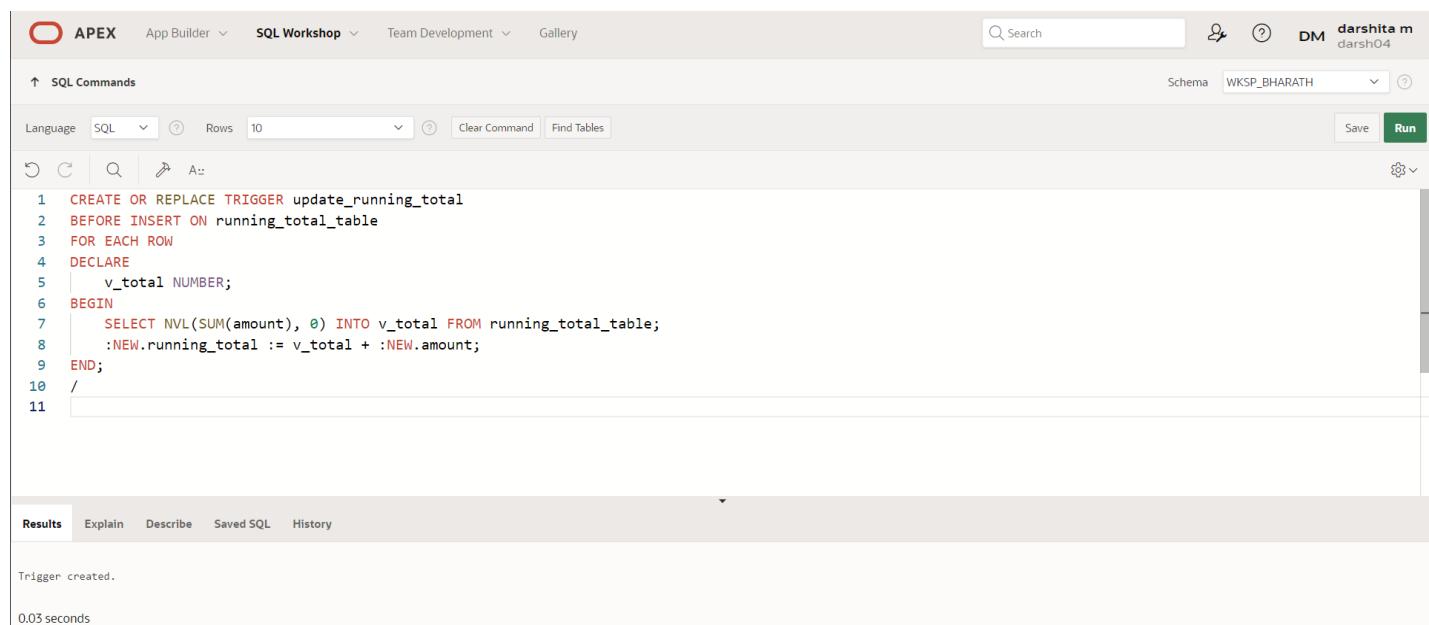
```
1 CREATE OR REPLACE TRIGGER log_user_activity
2 AFTER INSERT OR UPDATE OR DELETE ON activity_table
3 FOR EACH ROW
4 BEGIN
5     IF INSERTING THEN
6         INSERT INTO user_activity_log (log_id, action, table_name, record_id, change_time)
7         VALUES (activity_log_seq.NEXTVAL, 'INSERT', 'activity_table', :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);
```

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 SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user 'darshita m' and schema 'WKSP_BHARATH'. The main area is titled 'SQL Commands' with a 'Language' dropdown set to 'SQL'. Below the dropdown are buttons for Undo, Redo, Search, Clear Command, Find Tables, Save, and Run. The SQL code for the trigger is pasted into the command window. The code is as follows:

```
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 
```

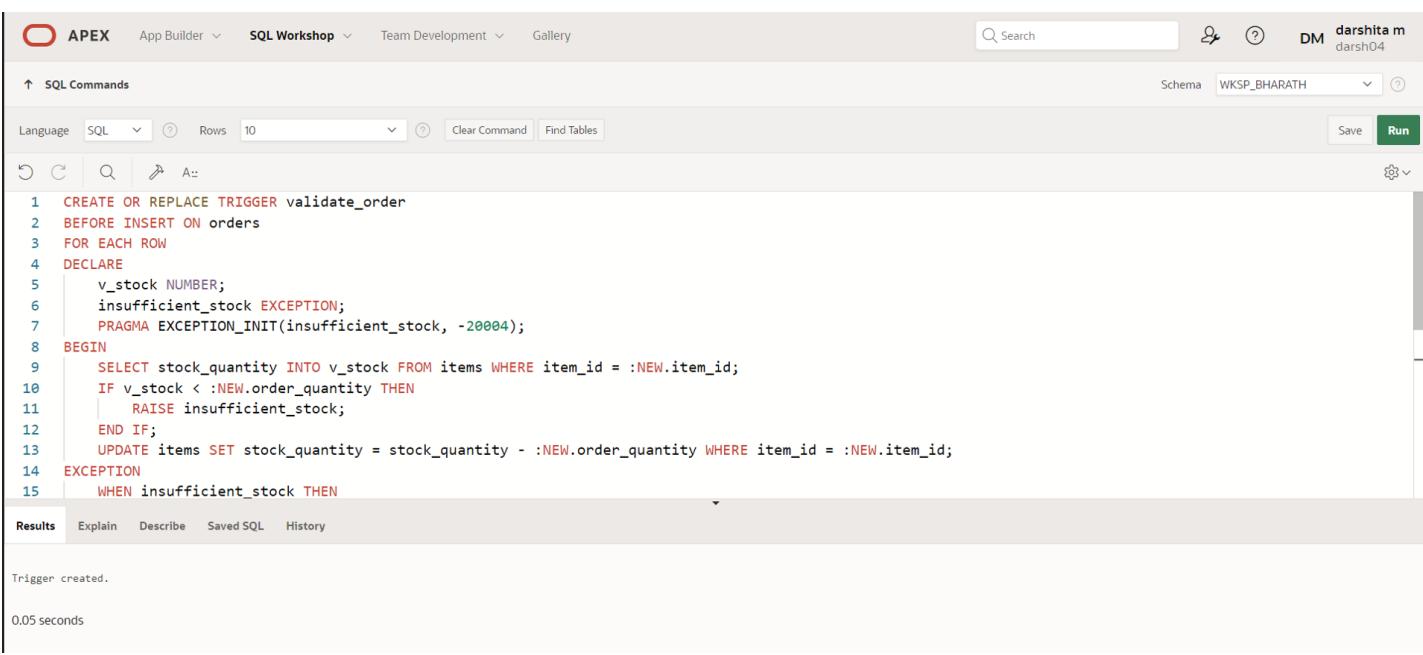
Below the code, the results tab is selected, showing the message 'Trigger created.' and a execution time of '0.03 seconds'.

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 SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The top right shows the user 'darshita m' and schema 'WKSP_BHARATH'. The main area is titled 'SQL Commands' with a language dropdown set to 'SQL'. Below the title are buttons for Undo, Redo, Find, Replace, and Run. The SQL command for creating the trigger is pasted into the editor. The command 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
```

The 'Results' tab at the bottom shows the output: 'Trigger created.' and '0.05 seconds'.

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

RESULT: thus the pl/sql trigger statements are executed successfully.

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:

```
darshita_55> db.restaurants.find({$and:[{ $or: [{cuisine: { $nin: ["American", "Chinese"] }}, {name: /^Wil/}]}], {restarestarestaurant_id: 1, name: 1, borough: 1, cuisine: 1})
[
  {
    _id: ObjectId('6650c743202862e984cdcdf6'),
    borough: 'Bronx',
    cuisine: 'Bakery',
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
```

- 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:

```
darshita_55> db.restaurants.find({ "grades": { $elemMatch: { "grade": "A", "score": 11, "date": ISODate("2014-08-11T00:00:00Z") } } }, { restaurant_id: 1, name: 1, grades: 1 })
darshita_55>
```

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:

```
darshita_55> 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 } )
darshita_55>
```

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:

```
darshita_55> db.restaurants.find({$and : [{"address.coord.1": {$gt : 42}}, {"address.coord.1": {$lte : 52}}]}, { _id:0, restaurant_id:1, name:1, address:1 })
```

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:

```
darshita_55> 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:

```
darshita_55> 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'
}]
```

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:

```
darshita_55> 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'
```

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:

```
darshita_55> db.restaurants.find({ "address.street": { $exists: true, $ne: "" } })
[ {
    _id: ObjectId('6650c743202862e984cdcdf6'),
    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: '30075045'
```

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:

```
darshita_55> db.restaurants.find({ "address.coord": { $elemMatch: { $type: "double" } } })
[
  {
    _id: ObjectId('6650c743202862e984cdcdf6'),
    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'
```

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:

```
darshita_55> db.restaurants.find({ "grades.score": { $mod: [7, 0] } }, { restaurant_id: 1, name: 1, grades: 1 })
[ {
  _id: ObjectId('6650c743202862e984cdcdf6'),
  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'
}]
```

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:

```
darshita_55> db.restaurants.find({ name: /mon/i }, { name: 1, borough: 1, "address.coord": 1, cuisine: 1 })
```

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

QUERY:

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

OUTPUT:

```
darshita_55> db.restaurants.find({ name: /^Mad/i }, { name: 1, borough: 1, "address.coord": 1, cuisine: 1 })
```

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

QUERY:

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

OUTPUT:

```
darshita_55> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } } })
[ {
  _id: ObjectId('6650c743202862e984cdcdf6'),
  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'
```

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:

```
darshita_55> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, "borough": "Manhattan" })
```

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:

```
darshita_55> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }] })
```

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, and their cuisine is not American.

QUERY:

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

OUTPUT:

```
darshita_55> 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:

```
darshita_55> 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:

```
darshita_55> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }] })
[ {
  _id: ObjectId('6650c743202862e984cdcdf6'),
  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:

```
darshita_55> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], "borough": "Manhattan" })
```

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:

```
darshita_55> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }] })
```

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:

```
darshita_55> 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:

```
darshita_55> 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:

```
darshita_55> db.restaurants.find({ $or: [{ "grades.score": 2 }, { "grades.score": 6 }] })
[ {
  _id: ObjectId('6650c743202862e984cdcdf6'),
  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'
```

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

RESULT: thus the mongodb statements are executed successfully.

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:

```
darshita_55> db.movies.find({ year: 1893 })
```

- 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:

```
darshita_55> db.movies.find({ runtime: { $gt: 120 } })
```

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

QUERY:

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

OUTPUT:

```
darshita_55> 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/MV5BMTU3NjE5NzYtYTYYNS00MDVmLWIwYjgtMmYwYWIxZDYyNzU2XkEyXkFqcGdeQXVyNzQzNzI@._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 00:27:59.17000000',
  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')
  }
}
```

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:

```
darshita_55> db.movies.find({ directors: 'William K.L. Dickson' })
```

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

QUERY:

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

OUTPUT:

```
darshita_55> 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')
    }
}
```

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

QUERY:

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

OUTPUT:

```
darshita_55> db.movies.find({ rated: 'UNRATED' })
```

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

QUERY:

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

OUTPUT:

```
darshita_55> 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/MV5BMTU3NjE5NzYtYTYYNS00MDVmLWIwYjgtMmYWIxZDYyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._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')
    }
  }
]
```

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:

```
darshita_55> 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/MV5BMTU3NjE5NzYtYTYYNS00MDVmLWIwYjgtMmYwYWIxZDYYNzU2XkEyXkFqcGdeQXVyNzQzNzI@._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')
  }
}
```

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:

```
darshita_55> db.movies.find({ 'tomatoes.viewer.rating': { $gt: 4 } })
```

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

QUERY:

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

OUTPUT:

```
darshita_55> 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/MV5BMTU3NjE5NzYtYTlyNS00MDVmLWIwYjgtMmYwYWIxZDYyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._V1_SV1000_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')
  }
}
```

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:

```
darshita_55> 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 }
... )
```

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:

```
darshita_55> db.movies.find(
...   { cast: 'Charles Kayser' },
...   { title: 1, languages: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 }
... )
```

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

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:

```
darshita_55> db.movies.find(
...   { released: ISODate("1893-05-09T00:00:00.000Z") },
...   { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 }
... )
```

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

QUERY:

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

OUTPUT:

```
darshita_55> db.movies.find(
...   { title: /scene/i },
...   { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 }
... )
```

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

RESULT: thus the mongodb queries are executed successfully.