

# 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 number(7),name varchar2(25));
```

OUTPUT:

The screenshot shows a SQL command interface with the following details:

- Language: SQL
- Rows: 10
- Schema: WKSP\_MYDEEPIKAWORKS1
- Commands:

```
1 create table dept(ID number(5),NAME Varchar(10));
```
- Results:

Table created.  
0.03 seconds

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

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

Length	7	25	25	7
--------	---	----	----	---

**QUERY:**

Create table EMP(id number(7),Last\_Name varchar2(25),First\_Name varchar2(25),Dept\_id number(7));

**OUTPUT:**

The screenshot shows a SQL command interface with the following details:

- Language:** SQL
- Schema:** WKSP\_MYDEEPIKAWORKS!
- Command:** CREATE TABLE EMP(id number(7), LAST\_NAME Varchar(25),FIRST\_NAME Varchar(25),DEPT\_ID number(7))
- Results:** Table created.
- Time:** 0.03 seconds

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

**QUERY:**

Alter table EMP modify(Last\_Name varchar2(25));

**OUTPUT:**

The screenshot shows a SQL command interface with the following details:

- Language:** SQL
- Schema:** WKSP\_MYDEEPIKAWORKS!
- Command:** ALTER TABLE EMP MODIFY LAST\_NAME Varchar(50)
- Results:** Table altered.
- Time:** 0.06 seconds

4. Create the EMPLOYEES2 table based on the structure of EMPLOYEES table. Include Only the Employee\_id, First\_name, Last\_name, Salary and Dept\_id columns. Name the columns Id, First\_name, Last\_name, salary and Dept\_id respectively.

QUERY:

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

OUTPUT:

The screenshot shows the SQL Commands tab in Oracle SQL Developer. The schema is set to 'WKSP\_MYDEEPIKAWORKS1'. The command entered is 'CREATE TABLE EMPLOYEES2 (id number(6),First\_name Varchar(25),Last\_name Varchar(25),Salary int,Dept\_id number(6))'. The results pane shows the message 'Table created.' and a execution time of '0.02 seconds'.

5. Drop the EMP table. QUERY:

```
Drop table EMP;
```

OUTPUT:

The screenshot shows the SQL Commands tab in Oracle SQL Developer. The schema is set to 'WKSP\_MYDEEPIKAWORKS1'. The command entered is 'DROP TABLE EMP'. The results pane shows the message 'Table dropped.' and a execution time of '0.07 seconds'.

6.Rename the EMPLOYEES2 table as EMP.

QUERY:

Rename EMPLOYEES2 to EMP; OUTPUT:

The screenshot shows a SQL command interface with the following details:

- SQL Commands** tab is selected.
- Schema dropdown: WKSP\_MYDEEPIKAWORKSF
- Language: SQL
- Rows: 10
- Buttons: Save, Run
- Toolbar icons: Undo, Redo, Search, Find Tables, etc.
- SQL Command: `1 RENAME EMPLOYEES2 to EMP`
- Results tab is selected.
- Output:
  - Statement processed.
  - 0.06 seconds

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

QUERY:

comment on table dept is 'Department Details';

comment on table EMP is 'Employee Details';

OUTPUT:

The screenshot shows a SQL command interface with the following details:

- SQL Commands** tab is selected.
- Schema dropdown: WKSP\_MYDEEPIKAWORKSF
- Language: SQL
- Rows: 10
- Buttons: Save, Run
- Toolbar icons: Undo, Redo, Search, Find Tables, etc.
- SQL Command: `1 Undo comment on table dept is 'Department details'`
- Results tab is selected.
- Output:
  - Statement processed.
  - 0.03 seconds

8.Drop the First\_name column from the EMP table and confirm it. QUERY:

Alter table EMP drop column first\_name; OUTPUT:

↑ SQL Commands

Schema WKSP\_MYDEEPIKAWORKS! ⓘ

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

⌚ 🔍 🗑️ A:: ⚙️ ⓘ

1 ALTER TABLE EMP DROP COLUMN First\_name

Results Explain Describe Saved SQL History

Table altered.

0.07 seconds

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

RESULT:

# 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 number(4),Last_name Varchar(25),First_name Varchar(25),Userid  
Varchar (25),Salary number(9,2))
```

OUTPUT:

The screenshot shows a SQL query editor interface. At the top, there's a toolbar with options like Language (set to SQL), Rows (set to 10), Clear Command, Find Tables, Save, and Run. The schema is set to WKSP\_MYDEEPIKAWORKSF. Below the toolbar, the SQL command is entered: `1 | Create TABLE MY_EMPLOYEE (ID number(4),Last_name Varchar(25),First_name Varchar(25),Userid Varchar(25),Salary number(9,2))`. The results tab is selected, showing the output: "Table created." and "0.04 seconds". Other tabs available are Explain, Describe, Saved SQL, and History.

2.Add the first and second rows data to MY\_EMPLOYEE table from the following sample data.

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

QUERY:

```
Insert into my_employee values(1,'Patel','Ralph','rpatel',895);
Insert into my_employee values(1,'Dancs','Betty','bdancs',860);
```

OUTPUT:

The screenshot shows a SQL command interface with the following details:

- Language: SQL
- Schema: WKSP\_MYDEEPIKAWORKS!
- Query: `Insert into my_employee values(1,'Dancs','Betty','bdancs',860);`
- Results:
  - 1 row(s) inserted.
  - 0.00 seconds

3.Display the table with values.

QUERY:

```
Select*from my_employee
```

OUTPUT:

The screenshot shows a SQL command interface with the following details:

- Language: SQL
- Schema: WKSP\_MYDEEPIKAWORKS!
- Query: `Select*from my_employee`
- Results:

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

2 rows returned in 0.02 seconds

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 a SQL command window with the following details:

- Schema: WKSP\_MYDEEPIKAWORKSF
- Language: SQL
- Rows: 10
- Command: Insert into my\_employee values(4,'Newman','Chad','Cnewman',750);
- Results tab selected.
- Output: 1 row(s) inserted.
- Time: 0.00 seconds

5.Make the data additions permanent.

QUERY:

```
Select*from my_employee;
```

OUTPUT:

The screenshot shows a SQL command window with the following details:

- Schema: WKSP\_MYDEEPIKAWORKSF
- Language: SQL
- Rows: 10
- Command: select \* from my\_employee
- Results tab selected.
- Output:

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
4	Newman	Chad	Cnewman	750
3	Biri	Ben	bbiri	1100
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	750
- Time: 0.02 seconds

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

QUERY:

```
Update my_employee set last_name='Drexler' where id=3;
```

## OUTPUT:

A screenshot of a SQL command window. The top bar shows "SQL Commands" and the schema "WKSP\_MYDEEPIKAWORKSF". The toolbar includes buttons for Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run. Below the toolbar, there are icons for Undo, Redo, Search, and others. The main area contains a single SQL command:  
1 Update my\_employee set last\_name='Drexler' where id=3;  
The results section shows "1 row(s) updated." and "0.01 seconds".

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:

A screenshot of a SQL command window. The top bar shows "SQL Commands" and the schema "WKSP\_MYDEEPIKAWORKSF". The toolbar includes buttons for Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run. Below the toolbar, there are icons for Undo, Redo, Search, and others. The main area contains a single SQL command:  
1 update my\_employee set salary=1000 where salary<900;  
The results section shows "3 row(s) updated." and "0.00 seconds".

8.Delete Betty dancs from MY\_EMPLOYEE table.

## QUERY:

```
DELETE from MY_EMPLOYEE where first_name='betty'
```

A screenshot of a SQL command window. The top bar shows "SQL Commands" and the schema "WKSP\_MYDEEPIKAWORKSF". The toolbar includes buttons for Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run. Below the toolbar, there are icons for Undo, Redo, Search, and others. The main area contains a single SQL command:  
1 DELETE from MY\_EMPLOYEE where First\_name='Betty';  
The results section shows "1 row(s) deleted." and "0.01 seconds".

9.Empty the fourth row of the emp table.

QUERY:

DELETE from MY\_EMPLOYEE where id=4

OUTPUT:

The screenshot shows a SQL query editor interface. At the top, there's a toolbar with various icons and buttons. Below the toolbar, the schema is set to 'WKSP\_MYDEEPIKAWORKSF'. The main area contains a SQL command: 'DELETE from MY\_EMPLOYEE where id=4'. The results section shows the output: '1 row(s) deleted.' and a time of '0.01seconds'. There are tabs for Results, Explain, Describe, Saved SQL, and History.

```
SQL Commands
Language: SQL | Rows: 10 | Clear Command | Find Tables | Save | Run
1  DELETE from MY_EMPLOYEE where id=4
Results | Explain | Describe | Saved SQL | History
1 row(s) deleted.
0.01seconds
```

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

RESULT:

# 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:

```
alter table EMP add constraint my_emp_id_pk primary key(id);
```

OUTPUT:

The screenshot shows a SQL command interface with the following details:

- Header: SQL Commands, Schema: WKSP\_MYDEEPIKAWORKSF
- Toolbar: Language (SQL), Rows (10), Clear Command, Find Tables, Save, Run
- Query Editor:

```
1 alter table emp add constraint my_emp_id_pk primary key(id);
```
- Results Tab: Selected
- Output:

```
Table altered.  
0.08 seconds
```

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:

```
Alter table DEPT add constraint my_dept_id_pk primary key(id);
```

OUTPUT:

The screenshot shows a SQL command interface with the following details:

- Header: SQL Commands, Schema: WKSP\_MYDEEPIKAWORKSF
- Toolbar: Language (SQL), Rows (10), Clear Command, Find Tables, Save, Run
- Query Editor:

```
1 Alter table DEPT add constraint my_dept_id_pk primary key(id);
```
- Results Tab: Selected
- Output:

```
Table altered.  
0.08 seconds
```

3.Add a column DEPT\_ID to the EMP table. Add a foreign key reference on the EMP table that ensures that the employee is not assigned to nonexistent department. Name the constraint my\_emp\_dept\_id\_fk.

QUERY:

```
alter table EMP add constraint my_emp_dept_id_fk foreign key(dept_id) references emp(id);
```

OUTPUT:

The screenshot shows the Oracle APEX SQL workspace interface. The top navigation bar includes 'SQL Commands', 'Schema (WKSP\_MYDEEPIKAWORKSF)', and a 'Run' button. Below the toolbar are buttons for Undo, Redo, Find, and Refresh. The main area contains the SQL command: '1 alter table emp add constraint my\_emp\_dept\_id\_fk foreign key(dept\_id) references emp(id);'. The results section shows the output: 'Table altered.' and '0.07 seconds'. At the bottom, it displays user information (220701058@rajalakshmi.edu.in, mydeepikaworkspace, en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the version (Oracle APEX 23.2.4).

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

QUERY:

```
Alter table EMP add commission number(2,2) check(commission>0);
```

OUTPUT:

The screenshot shows the Oracle APEX SQL workspace interface. The top navigation bar includes 'SQL Commands', 'Schema (WKSP\_MYDEEPIKAWORKSF)', and a 'Run' button. Below the toolbar are buttons for Undo, Redo, Find, and Refresh. The main area contains the SQL command: '1 Alter table emp add commission number(2,2) check(commission>0);'. The results section shows the output: 'Table altered.' and '0.07 seconds'. At the bottom, it displays user information (220701058@rajalakshmi.edu.in, mydeepikaworkspace, 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:

# 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  
sal*12 as 'ANNUAL SALARY'  
FROM employees;
```

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

QUERY:

```
desc DEPT
```

OUTPUT:

The screenshot shows a SQL query editor interface. At the top, there are buttons for 'SQL Commands', 'Language' (set to SQL), 'Rows' (set to 10), 'Clear Command', 'Find Tables', 'Save', and 'Run'. The main area contains the SQL command 'desc DEPT'. Below the command, there is a table titled 'DEPT' with two rows of data. The table has columns: Table, Column, Data Type, Length, Precision, Scale, Primary Key, Nullable, Default, and Comment. The first row has 'DEPT' in the Table column and 'ID' in the Column column. The second row has 'NAME' in the Column column. The Data Type for ID is NUMBER and for NAME is VARCHAR2. The Length for ID is 5 and for NAME is 10. The Precision for ID is 0 and for NAME is 1. The Scale for ID is 0 and for NAME is 0. The Primary Key is marked as 1 for ID and 0 for NAME. The Nullable column has a checkmark for NAME. The Default and Comment columns are both marked as '-'.

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPT	ID	NUMBER	-	5	0	1	-	-	-
	NAME	VARCHAR2	10	-	-	0	✓	-	-

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:

```
CREATE TABLE EMPP ( employee_id NUMBER(7),last_name VARCHAR(45),job_id  
NUMBER(5),hire_date DATE );  
INSERT INTO EMPP VALUES(100,'Ben',111,'04-05-2005');  
INSERT INTO EMPP VALUES(101,'Becky',112,'05-05-2005');  
select * from EMPP;
```

OUTPUT:

The screenshot shows the Oracle APEX SQL workspace interface. The top navigation bar includes 'SQL Commands', 'Schema (WKSP\_MYDEEPIKAWORKS1)', 'Save', and 'Run' buttons. Below the toolbar are buttons for Undo, Redo, Find, and Refresh. The main area contains the SQL command 'select \* from empp'. The results section displays the following data:

EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE
100	Ben	111	04/05/2005
101	Becky	112	05/05/2005

Below the table, it says '2 rows returned in 0.01 seconds' and has a 'Download' link. The bottom of the screen shows the user's email (220701058@rajalakshmi.edu.in), workspace name (mydeepikaworkspace), and language (en). Copyright information and Oracle APEX version (23.2.4) are also present.

4.Provide an alias STARTDATE for the hire date.

QUERY:

```
SELECT hire_date as "STARTDATE" from EMPP;
```

OUTPUT:

The screenshot shows the Oracle APEX SQL workspace interface. The top navigation bar includes 'SQL Commands', 'Schema (WKSP\_MYDEEPIKAWORKS1)', 'Save', and 'Run' buttons. Below the toolbar are buttons for Undo, Redo, Find, and Refresh. The main area contains the SQL command 'select hire\_date as "STARTDATE" from empp;'. The results section displays the following data:

STARTDATE
04/05/2005
05/05/2005

Below the table, it says '2 rows returned in 0.01 seconds' and has a 'Download' link. The bottom of the screen shows the user's email (220701058@rajalakshmi.edu.in), workspace name (mydeepikaworkspace), and language (en). Copyright information and Oracle APEX version (23.2.4) are also present.

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

QUERY:

Select distinct job\_id from EMPP;

OUTPUT:

The screenshot shows the Oracle APEX SQL workspace interface. The SQL command is: `select distinct job_id from empp;`. The results show two rows: 112 and 111. The output is displayed in a table with a single column labeled "JOB\_ID".

JOB_ID
112
111

2 rows returned in 0.01 seconds    Download

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\_id as "EMPLOYEE\_AND\_TITLE" from EMPP;

OUTPUT:

The screenshot shows the Oracle APEX SQL workspace interface. The SQL command is: `Select last_name||','||' '||job_id as "EMPLOYEE_AND_TITLE" from empp;`. The results show two rows: Ben,111 and Becky,112. The output is displayed in a table with a single column labeled "EMPLOYEE\_AND\_TITLE".

EMPLOYEE_AND_TITLE
Ben,111
Becky,112

2 rows returned in 0.01 seconds    Download

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 employee\_id||'.'||last\_name||'.'||job\_id||'.'||hire\_date as THE\_OUTPUT from EMPP;  
OUTPUT:

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

```
1 Select
2 employee_id||'.'||last_name||'.'||job_id||'.'||hire_date as THE_OUTPUT from EMPP;
```

The results section displays the output:

THE_OUTPUT
100,Ben,111,04/05/2005
101,Becky,112,05/05/2005

Below the results, it says "2 rows returned in 0.01 seconds" and has a "Download" link.

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

## RESULT:

# RESTRICTING AND SORTING DATA

EX\_NO:5

DATE: 07-03-2024

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

## QUERY:

Select last\_name from employees\_table where salary>12000 ;

## OUTPUT:

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

```
1 Select last_name from employees_table where salary>12000;
2
```

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

LAST_NAME
Mohan

Below the table, it says "1 rows returned in 0.00 seconds".

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

## QUERY:

Select last\_name,department\_id from employees\_table where employee\_id=176;

## OUTPUT:

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

```
1 Select last_name,department_id from employees_table where employee_id=176;
2
```

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

LAST_NAME	DEPARTMENT_ID
Hari	120

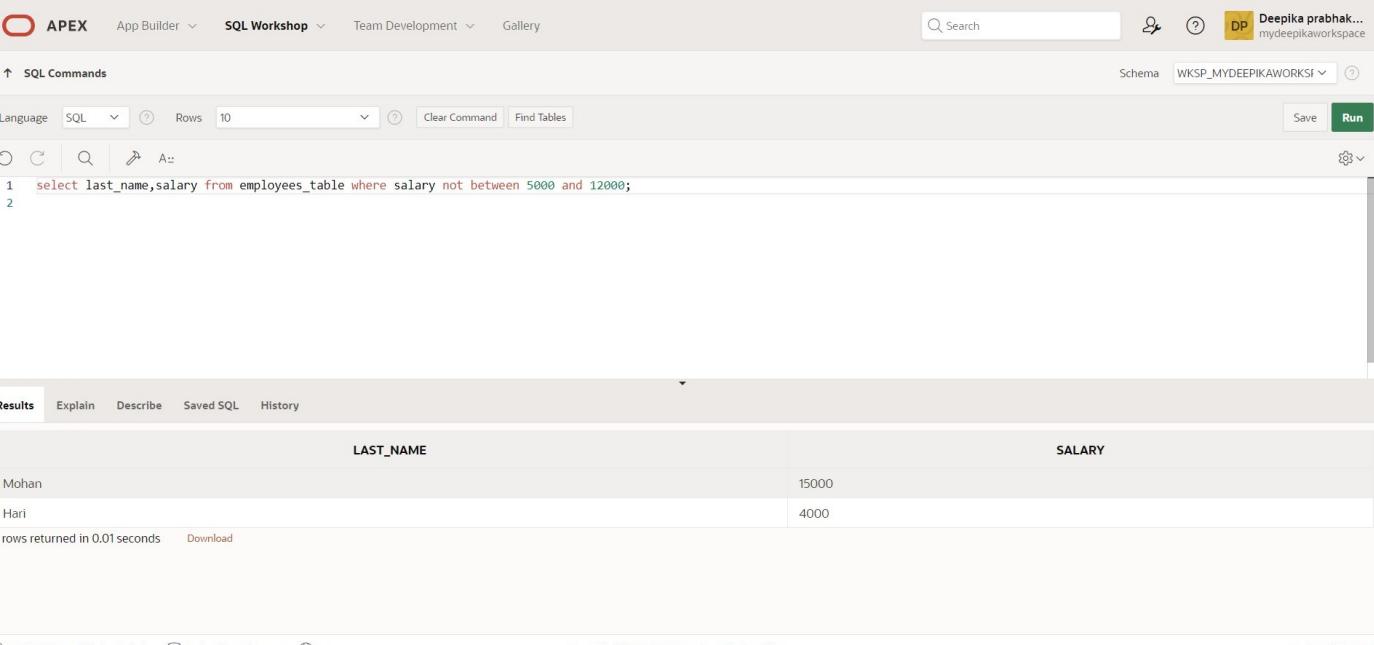
Below the table, it says "1 rows returned in 0.01 seconds".

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

#### QUERY:

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

#### OUTPUT:



LAST_NAME	SALARY
Mohan	15000
Hari	4000

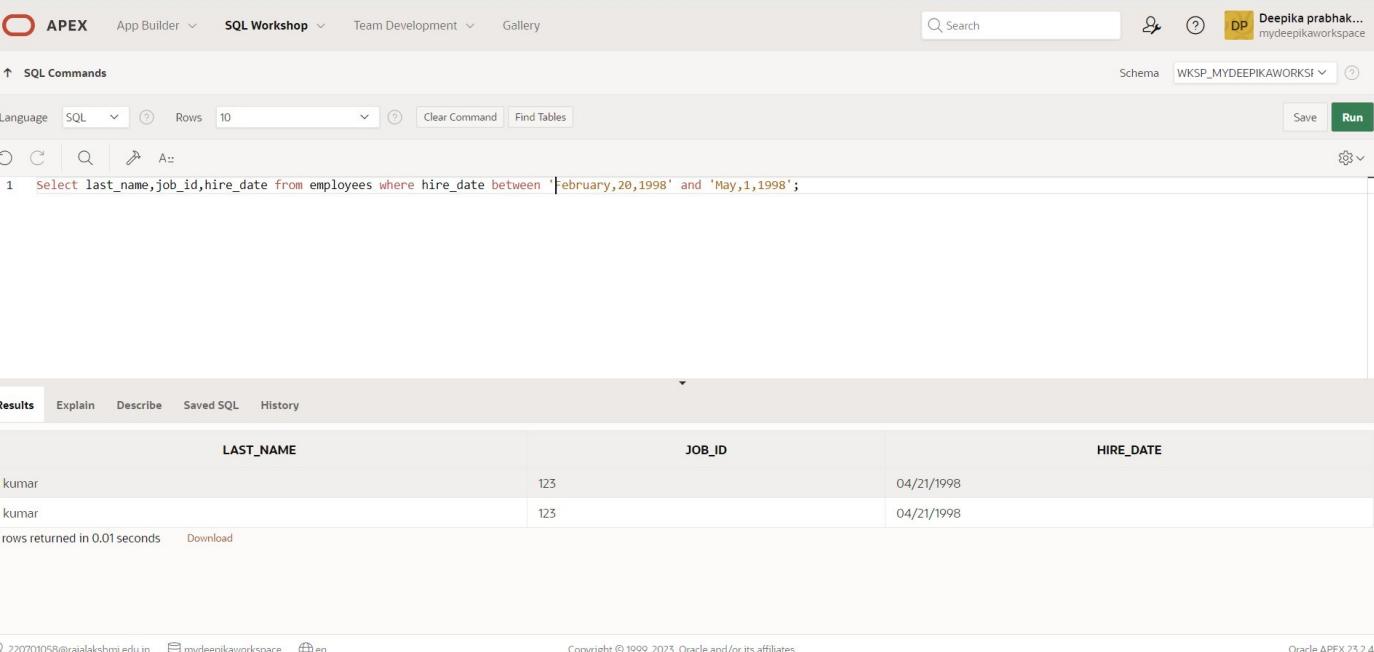
2 rows returned in 0.01 seconds [Download](#)

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

#### QUERY:

```
Select last_name,job_id,hire_date from employees_table where hire_date between 'February,20,1998' and 'May,1,1998';
```

#### OUTPUT:



LAST_NAME	JOB_ID	HIRE_DATE
kumar	123	04/21/1998
kumar	123	04/21/1998

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

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

**QUERY:**

```
select last_name,department_id from employees_table where department_id in(20,50) order by last_name;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, user profile, and schema dropdown set to 'WKSP\_MYDEEPIKAWORKS1'. 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'. The SQL command entered is:

```
1 select last_name,department_id from employees_table where department_id in(20,50) order by last_name;
2
```

The results section shows the output:

LAST_NAME	DEPARTMENT_ID
Janu	20

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

At the bottom, the URL is 220701058@rajalakshmi.edu.in, the workspace is mydeepikaworkspace, and the page is en. Copyright information and Oracle APEX 23.2.4 are also present.

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

**QUERY:**

```
select last_name as "EMPLOYEE",salary as "MONTHLY SALARY" from employees_table where (salary between 5000 and 12000) and (department_id in(20,50)) order by last_name asc;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, user profile, and schema dropdown set to 'WKSP\_MYDEEPIKAWORKS1'. 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'. The SQL command entered is:

```
1 select last_name as "EMPLOYEE",salary as "MONTHLY SALARY" from employees_table where (salary between 5000 and 12000) and (department_id in(20,50)) order by last_name asc;
2
```

The results section shows the output:

EMPLOYEE	MONTHLY SALARY
Janu	7000

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

At the bottom, the URL is 220701058@rajalakshmi.edu.in, the workspace is mydeepikaworkspace, and the page is en. Copyright information and Oracle APEX 23.2.4 are also present.

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

#### QUERY:

```
select last_name,hire_date from employees_table where hire_date like '1994';
```

#### 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. On the right, a user profile for "Deepika prabhak..." is shown, along with the schema "WKSP\_MYDEEPIKAWORKS1". The main area is titled "SQL Commands" with a "Language" dropdown set to "SQL". The query entered is: `select last_name,hire_date from employees_table where hire_date like '1994';`. Below the query, the results section displays the output:

LAST_NAME	HIRE_DATE
Hari	03/24/1994

1 rows returned in 0.01 seconds. The bottom of the screen shows copyright information for Oracle and the APEX version 23.2.4.

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

#### QUERY:

```
select last_name,job_id from employees_table where manager_id is null;
```

#### 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. On the right, a user profile for "Deepika prabhak..." is shown, along with the schema "WKSP\_MYDEEPIKAWORKS1". The main area is titled "SQL Commands" with a "Language" dropdown set to "SQL". The query entered is: `select last_name,job_id from employees_table where manager_id is null;`. Below the query, the results section displays the output:

LAST_NAME	JOB_ID
Mohan	ac_account
Hari	sales_rep

2 rows returned in 0.00 seconds. The bottom of the screen shows copyright information for Oracle and the APEX version 23.2.4.

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

### QUERY:

```
select last_name,salary,commission_pct from employees_table where commission_pct 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. On the right, there's a user profile for 'Deepika prabhak...' and a workspace dropdown for 'WKSP\_MYDEEPIKAWORKS...'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab shows the executed query: 'select last\_name,salary,commission\_pct from employees\_table where commission\_pct is not null order by salary desc;'. The Results tab displays the output in a grid format:

LAST_NAME	SALARY	COMMISSION_PCT
Mohan	15000	.2

Below the results, it says '1 rows returned in 0.01 seconds' and has a 'Download' link. The bottom footer includes copyright information for Oracle and workspace details.

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

### QUERY:

```
select last_name from employees_table where last_name like '_a%';
```

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface, identical to the previous one but with a different query. The top navigation bar and workspace information are the same. The SQL Commands tab shows the query: 'select last\_name from employees\_table where last\_name like '\_a%';'. The Results tab displays the output in a grid format:

LAST_NAME
Haari

Below the results, it says '1 rows returned in 0.01 seconds' and has a 'Download' link. The bottom footer includes copyright information for Oracle and workspace details.

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

#### QUERY:

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

#### OUTPUT:

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

```
1 select last_name from employees_table where last_name like '%a%' and last_name like '%e%';
2 |
```

The results section displays the output:

LAST_NAME
James

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

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

#### QUERY:

```
select last_name,job_id,salary from employees_table where job_id in ('sales representative','stock clerk') and salary not in(2500,3500,7000);
```

#### OUTPUT:

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

```
1 select last_name,job_id,salary from employees_table where job_id in ('sales_rep','stock_clerk') and salary not in(2500,3500,7000);
```

The results section displays the output:

LAST_NAME	JOB_ID	SALARY
Haari	sales_rep	4000

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

## 1. Write a query to dis

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

### QUERY:

```
select last_name,salary,commission_pct from employees_table where commission_pct=0.2;
```

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected, followed by 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. On the right, there's a search bar, a user icon, a help icon, and a workspace dropdown set to 'Deepika prabhak... mydeepikaworkspace'. Below the navigation is a toolbar with icons for 'SQL Commands', 'Language: SQL', 'Rows: 10', 'Clear Command', 'Find Tables', 'Save', and 'Run'. The main area contains the SQL command:

```
1 select last_name,salary,commission_pct from employees_table where commission_pct=0.2;
2
```

Below the command, the results section shows the output:

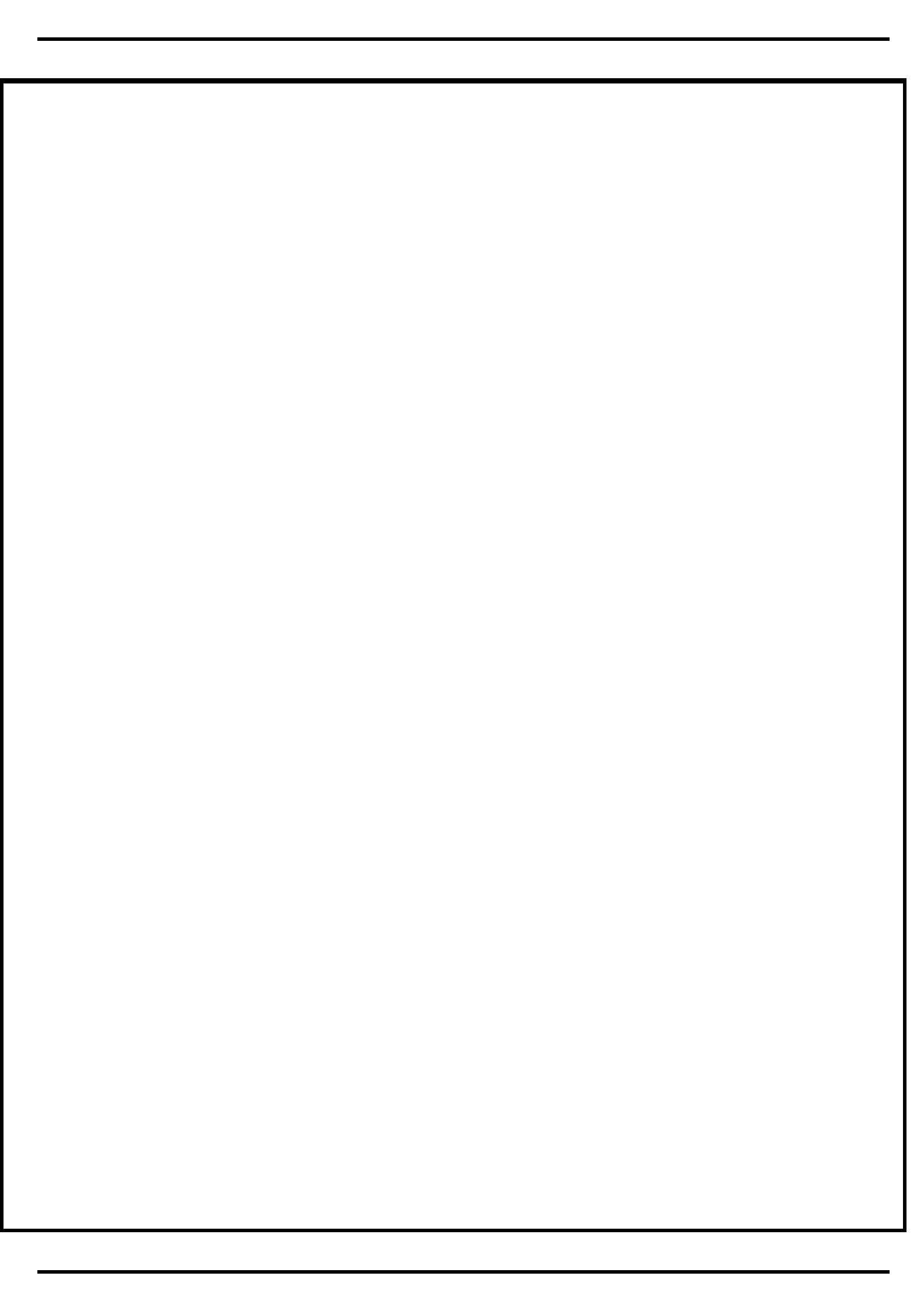
LAST_NAME	SALARY	COMMISSION_PCT
Mohan	15000	.2

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

At the bottom, the footer includes the URL 220701058@rajalakshmi.edu.in, the workspace name mydeepikaworkspace, and the language en. It also states Copyright © 1999, 2023, Oracle and/or its affiliates, and Oracle APEX 23.2.4.

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

### RESULT:



# 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:

**OUTPUT:**

The screenshot shows the Oracle APEX interface with the SQL Workshop tab selected. In the command window, the following SQL code is entered:

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

After running the query, the results are displayed in the Results tab:

SYSDATE
03/15/2024

Copyright © 1999, 2025, Oracle and/or its affiliates.

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 employee_id, last_name, salary, salary+(15.5/100*salary)  
"new_salary"from employees_table
```

**OUTPUT:**

The screenshot shows the Oracle APEX interface with the SQL Workshop tab selected. In the command window, the following SQL code is entered:

```
1 select employee_id, last_name, salary, salary+(15.5/100*salary) "new_salary"from employees_table;  
2
```

After running the query, the results are displayed in the Results tab:

EMPLOYEE_ID	LAST_NAME	SALARY	new_salary
152	Janu	7000	8085
2	Mohan	15000	17325
176	Nivas	4000	4620

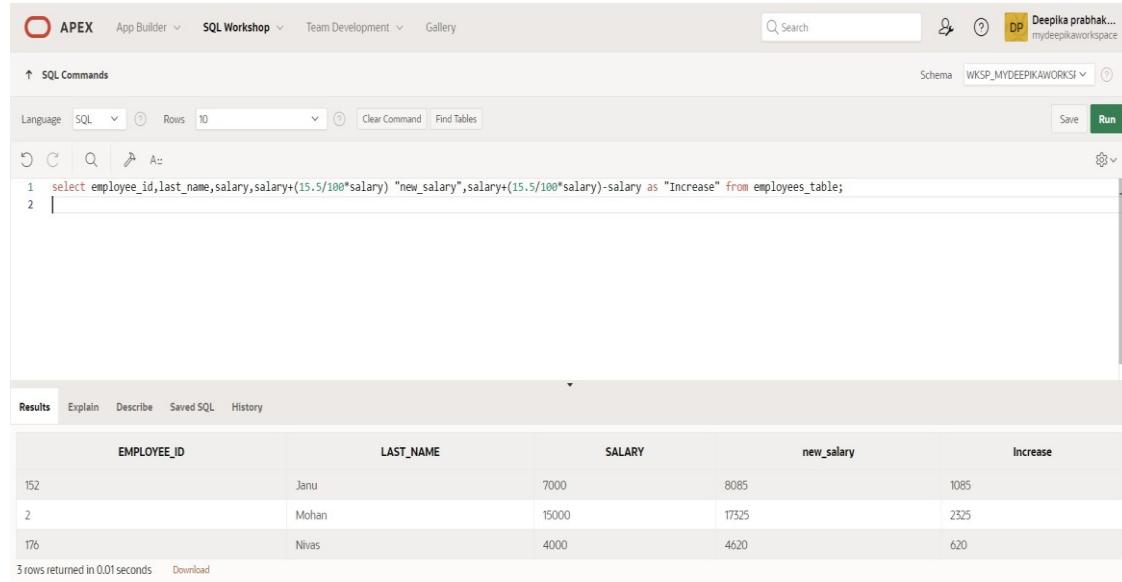
Copyright © 1999, 2025, Oracle and/or its affiliates.

**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 employee_id, last_name, salary, salary+(15.5/100*salary) "new_salary", salary+(15.5/100*salary)-salary as "Increase" from employees_table
```

**OUTPUT:**



EMPLOYEE_ID	LAST_NAME	SALARY	new_salary	Increase
152	Janu	7000	8085	1085
2	Mohan	15000	17325	2325
176	Nivas	4000	4620	620

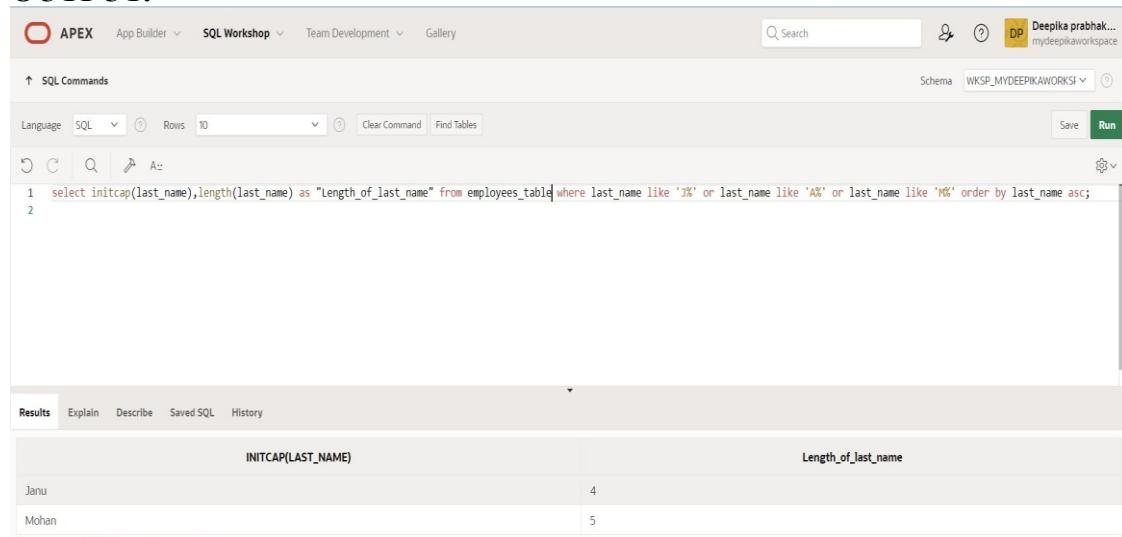
3 rows returned in 0.01 seconds [Download](#)

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

**QUERY:**

```
select initcap(last_name),length(last_name) as "Length_of_last_name" from employees_table where last_name like 'J%' or last_name like 'A%' or last_name like 'M%' order by last_name asc;
```

**OUTPUT:**



INITCAP(LAST_NAME)	Length_of_last_name
Janu	4
Mohan	5

2 rows returned in 0.00 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\_table whose last name starts with the letter H.

#### QUERY:

```
select initcap(last_name),length(last_name) as "Length_of_last_name" from employees_table where last_name like 'H%' order by last_name asc;
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user profile 'Deepika prabhak...' and the schema 'WKSP\_MYDEEPIKAWORKS1'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the query: 'select initcap(last\_name),length(last\_name) as "Length\_of\_last\_name" from employees\_table where last\_name like 'H%' order by last\_name asc;'. The Results tab displays the output:

INITCAP(LAST_NAME)	Length_of_last_name
Hari	4

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

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 last_name,round((sysdate-hire_date)/30,0) as "MONTHS_WORKED" from employees_table order by round((sysdate-hire_date)/30,0) asc;
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user profile 'Deepika prabhak...' and the schema 'WKSP\_MYDEEPIKAWORKS1'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the query: 'select last\_name,round((sysdate-hire\_date)/30,0) as "MONTHS\_WORKED" from employees\_table order by round((sysdate-hire\_date)/30,0) asc;'. The Results tab displays the output:

LAST_NAME	MONTHS_WORKED
Janu	317
Mohan	342
Hari	365

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

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 last_name||' earns'||salary||' monthly but wants'||salary*3 as "DREAM_SALARIES" from employees_table;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a user profile for 'Deepika prabhak...' and a workspace dropdown for 'WKSP\_MYDEEPIKAWORKSf'. The main area has tabs for 'SQL Commands', 'Language' (set to SQL), 'Rows' (set to 10), and buttons for 'Clear Command' and 'Find Tables'. Below this is a code editor with the following SQL command:

```
1 select last_name||' earns'||salary||' monthly but wants'||salary*3 as "DREAM_SALARIES" from employees_table;
2
```

Under the 'Results' tab, the output is displayed in a table with a single column labeled 'DREAM\_SALARIES'. The data rows are:

DREAM_SALARIES
Janu earns 7000 monthly but wants 21000
Mohan earns 15000 monthly but wants 45000
Hari earns 4000 monthly but wants 12000

At the bottom, it says '3 rows returned in 0.00 seconds' and has a 'Download' link. The footer includes the URL '220701058@rajalakshmi.edu.in', the workspace name 'mydeepikaworkspace', and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' The version 'Oracle APEX 23.2.4' is also mentioned.

8. Create a query to display the last name and salary for all employees\_table. Format the salary to be 15 characters long, left-padded with the \$ symbol. Label the column SALARY.

**QUERY:**

```
select last_name, lpad(salary, 15, '$') as "SALARY" from employees_table;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface, identical to the previous one but with a different query. The top navigation bar and workspace information are the same. The code editor contains the following SQL command:

```
1 select last_name, lpad(salary, 15, '$') as "SALARY" from employees_table;
2
```

Under the 'Results' tab, the output is displayed in a table with two columns: 'LAST\_NAME' and 'SALARY'. The data rows are:

LAST_NAME	SALARY
Janu	\$\$\$\$\$\$\$\$\$\$7000
Mohan	\$\$\$\$\$\$\$\$\$\$15000
Hari	\$\$\$\$\$\$\$\$\$\$4000

At the bottom, it says '3 rows returned in 0.00 seconds' and has a 'Download' link. The footer includes the URL '220701058@rajalakshmi.edu.in', the workspace name 'mydeepikaworkspace', and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' The version 'Oracle APEX 23.2.4' is also mentioned.

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

**QUERY:**

```
SELECT last_name,hire_date,TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6), 'MONDAY'),'FMDay, "the "FMDD "of "FMMonth, YYYY') AS REVIEW FROM employees_table;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows a user profile for 'Deepika prabhak...' and the workspace 'WKSP\_MYDEEPIKAWORKS1'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the executed query:

```
1 SELECT last_name,hire_date,TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6), 'MONDAY'),'FMDay, "the "FMDD "of "FMMonth, YYYY') AS REVIEW FROM employees_table;
```

The Results tab displays the output:

LAST_NAME	HIRE_DATE	REVIEW
Janu	02/25/1998	Monday, the 31 of August, 1998
Mohan	02/03/1996	Monday, the 05 of August, 1996
Hari	03/24/1994	Monday, the 26 of September, 1994

Below the table, it says '3 rows returned in 0.01 seconds' and there is a 'Download' link. At the bottom, the footer includes the URL '220701058@rajalakshmi.edu.in', the workspace 'mydeepikaworkspace', and the language 'en'. It also states 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 25.2.4'.

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

**QUERY:**

```
SELECT last_name,hire_date,TO_CHAR(hire_date,'Day') as Day from employees_table order by TO_CHAR(hire_date,'Day');
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows a user profile for 'Deepika prabhak...' and the workspace 'WKSP\_MYDEEPIKAWORKS1'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the executed query:

```
1 SELECT last_name,hire_date,TO_CHAR(hire_date,'Day') as Day from employees_table order by TO_CHAR(hire_date,'Day');
```

The Results tab displays the output:

LAST_NAME	HIRE_DATE	DAY
Mohan	02/03/1996	Saturday
Hari	03/24/1994	Thursday
Janu	02/25/1998	Wednesday

Below the table, it says '3 rows returned in 0.01 seconds' and there is a 'Download' link. At the bottom, the footer includes the URL '220701058@rajalakshmi.edu.in', the workspace 'mydeepikaworkspace', and the language 'en'. It also states 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 25.2.4'.

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

**RESULT:**

# 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.last_name,e.department_number,d.dept_id from empo21 e,dept23 d where e.department_number=d.dept_id;
```

**OUTPUT:**

The screenshot shows a SQL query editor with the following details:

- Language: SQL
- Rows: 10
- Clear Command
- Find Tables
- Run button
- SQL command: `select e.last_name,e.department_number,d.dept_id from empo21 e,dept23 d where e.department_number=d.dept_id;`
- Results tab selected
- Table headers: LAST\_NAME, DEPARTMENT\_NUMBER, DEPT\_ID
- Data row: Joe, 80, 80
- Message: 1 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 empo21 e,dept23 d where e.department_number=d.dept_id and e.department_number=80; OUTPUT:
```

The screenshot shows a SQL query editor with the following details:

- Language: SQL
- Rows: 10
- Clear Command
- Find Tables
- Run button
- SQL command: `select distinct job_id,loc_id from empo21 e,dept23 d where e.department_number=d.dept_id and e.department_number=80;`
- Results tab selected
- Table headers: JOB\_ID, LOC\_ID
- Data row: 8978, 49
- Message: 1 rows returned in 0.03 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.last_name,e.department_number,d.dept_name,d.loc_id,l.city from empo21 e,dept23 d,location l  
where e.department_number=d.dept_id and d.loc_id=l.location_id and e.commission_pct is not null;
```

#### OUTPUT:

The screenshot shows a SQL query execution interface. The query is:

```
1 select e.last_name,e.department_number,d.dept_name,d.loc_id,l.city from empo21 e,dept23 d,location l where e.department_number=d.dept_id and d.loc_id=l.location_id and  
2 e.commission_pct is not null;
```

The results table has columns: LAST\_NAME, DEPARTMENT\_NUMBER, DEPT\_NAME, LOC\_ID, and CITY. One row is returned:

LAST_NAME	DEPARTMENT_NUMBER	DEPT_NAME	LOC_ID	CITY
Joe	80	CSE	49	chennai

1 rows returned in 0.04 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 empo21.last_name,dept23.dept_name from empo21,dept23 where  
empo21.department_number=dept23.dept_id and last_name like '%a%';
```

#### OUTPUT:

The screenshot shows a SQL query execution interface. The query is:

```
1 select empo21.last_name,dept23.dept_name from empo21,dept23 where empo21.department_number=dept23.dept_id and last_name like '%a%';|
```

The results table has columns: LAST\_NAME and DEPT\_NAME. No data is found:

LAST_NAME	DEPT_NAME
no data found	

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.last_name,e.department_number,e.job_id,d.dept_name from empo21 e join dept d  
on(e.department_number=d.dept_id) join location on (d.location_id=location.location_id)  
where lower(location.city)=’toronto’;
```

**OUTPUT:**

The screenshot shows a SQL query editor interface. At the top, there are buttons for Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run. Below the toolbar is a code editor window containing the following SQL query:

```
1 select e.last_name,e.department_number,e.job_id,d.dept_name from empo21 e join dept d on(e.department_number=d.dept_id) join location on (d.location_id=location.location_id)  
2 where lower(location.city)=’toronto’;
```

Below the code editor is a results pane with tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, and it displays the message "no data found".

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.last_name “Employee”,w.emp_id “emp#”,m.last_name ‘manager’,m.emp_id “Mgr#” from empo21  
m on (w.manager_id=m.emp_id); OUTPUT:
```

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language SQL Rows 10 Clear Command Find Tables Schema WKSP\_BHANU25 Save Run

```
1 select w.last_name "Employee",w.emp_id "emp#",m.last_name "manager",m.emp_id "Mgr#" from empo21 w join empo21 m on (w.manager_id=m.emp_id);
```

Results Explain Describe Saved SQL History

Employee	emp#	manager	Mgr#
appu	12	appu	12

1 rows returned in 0.01 seconds Download

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

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.last_name "Employee",w.emp_id "emp#",m.last_name 'manager',m.emp_id "Mgr#" from empo21  
w left outer join empo21 m on (w.manager_id=m.emp_id);
```

### OUTPUT:

Language SQL Rows 10 Clear Command Find Tables Save Run

```
1 select w.last_name "Employee",w.emp_id "emp#",m.last_name "manager",m.emp_id "Mgr#" from empo21 w left outer join empo21 m on (w.manager_id=m.emp_id);
```

Results Explain Describe Saved SQL History

Employee	emp#	manager	Mgr#
appu	12	appu	12
prayae	1	-	-
Joe	4	-	-
Harsun	3	-	-

4 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.department_number dept23,e.last_name colleague from empo21 e join empo21 c on  
(e.department_number=c.department_number) where e.emp_id <> c.emp_id order by  
e.department_number,e.last_name,c.last_name;
```

## OUTPUT:

The screenshot shows a SQL query execution interface. The query is:

```
1 select e.department_number dept23,e.last_name empo21,c.last_name colleague from empo21 e join empo21 c on (e.department_number=c.department_number) where e.emp_id <> c.emp_id
2 order by e.department_number,e.last_name,c.last_name;
```

The results table has three columns: DEPT23, EMPO21, and COLLEAGUE. It contains two rows:

DEPT23	EMPO21	COLLEAGUE
111	appu	prayae
111	prayae	appu

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:

```
SELECT e.last_name, e.job_id, d.dept_name, e.salary, j.grade_level
FROM emp18 e JOIN dept18 d
ON (e.dept_id = d.dept_id) JOIN
job_grade j
ON (e.salary BETWEEN j.lowest_sal AND j.highest_sal);
```

## OUTPUT:

The screenshot shows a SQL query execution interface. The query is:

```
1 SELECT e.last_name, e.job_id, d.dept_name, e.salary, j.grade_level
2 FROM empo21 e JOIN dept23 d
3 ON (e.dept_id = d.dept_id)
4 JOIN job_grade j
5 ON (e.salary BETWEEN j.lowest_sal AND j.highest_sal);
```

The results table has five columns: last\_name, job\_id, dept\_name, salary, and grade\_level. The message "no data found" is displayed.

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

## QUERY:

```

SELECT e.last_name, e.hire_date FROM emp18 e, emp18 davies
WHERE davies.last_name = 'Davies'
AND davies.hire_date < e.hire_date;

```

## OUTPUT:

The screenshot shows a SQL query editor with the following details:

- Language:** SQL
- Rows:** 10
- Clear Command** and **Find Tables** buttons
- Run** button
- SQL Query:**

```

1 SELECT e.last_name, e.hire_date
2 FROM emp021 e, emp021 davies
3 WHERE davies.last_name = 'davies'
4 AND davies.hire_date < e.hire_date;
5

```
- Results Tab:** Active tab
- Table Headers:** LAST\_NAME, HIRE\_DATE
- Table Data:**

LAST_NAME	HIRE_DATE
Johnson	03/01/1998
Jones	05/01/1998
Brown	04/22/1997
- Timing:** 3 rows returned in 0.01 seconds
- Actions:** Download link

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

## QUERY:

```

SELECT e.last_name AS Employee, e.hire_date AS Emp_Hired, e.
manager_name AS Manager, m.hire_date AS Mgr_Hired
FROM emp18 e
JOIN emp18|m ON e.manager_name = m.last_name
WHERE e.hire_date < m.hire_date;

```

## OUTPUT:

The screenshot shows a SQL query editor interface. At the top, there are buttons for Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run. Below the toolbar, the query text is displayed:

```

1  SELECT e.last_name AS Employee, e.hire_date AS Emp_Hired,
2  [ ] e.last_name AS Manager, m.hire_date AS Mgr_Hired
3  FROM empoz1 e
4  JOIN empoz1 m ON e.last_name = m.last_name
5  WHERE e.hire_date < m.hire_date;
6

```

The second line of the query has a syntax error, indicated by a red bracket and a cursor. The results section below shows "no data found".

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

## RESULT

# 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 EMPB;
```

**OUTPUT:**

The screenshot shows a SQL query editor with the following details:

- Toolbar:** Language (SQL), Rows (10), Clear Command, Find Tables, Save, Run.
- Query Area:** Contains the SQL code:

```
1 select Round(Max (salary),0)"Maximum", Round (Min (salary),0) "Minimum", round(sum(salary),0)"sum",
2                                         round (avg(salary),0) "Average" from EMPB;
```
- Results Tab:** Active tab, showing the output of the query.
- Output Data:** A table with four columns: Maximum, Minimum, sum, and Average. The values are 90000, 60000, 300000, and 75000 respectively.
- Message Bar:** Shows "1 rows returned in 0.01 seconds" and a "Download" link.

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 EMPB group by job_id;
```

**OUTPUT:**

The screenshot shows a SQL query editor with the following details:

SQL Query:

```
1 select job_id ,Round(MAX(salary),0) "MAXIMUM",Round (Min(salary),0)"Minimum",Round  
2 (SUM(Salary),0)"sum" ,Round (AVg (salary),0)"average" from EMPB group by job_id;
```

Results:

JOB_ID	MAXIMUM	Minimum	sum	average
44	70000	70000	70000	70000
65	90000	90000	90000	90000
46	60000	60000	60000	60000
47	80000	80000	80000	80000

4 rows returned in 0.01 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 EMPB group by job_id ; select job_id,  
count(*) from EMPB where job_id='47' group by job_id ;
```

**OUTPUT:**

The screenshot shows a SQL query editor with the following details:

SQL Query:

```
1 select job_id, count(*) from EMPB where job_id='47' group by job_id ;  
2  
3
```

Results:

JOB_ID	COUNT(*)
47	1

1 rows returned in 0.01 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(distinct manager_id )"Number of managers" from empB;
```

### OUTPUT:

The screenshot shows a SQL query execution interface. The query is:

```
1 select count(distinct manager_id )"Number of managers" from empB;
2
3
```

The results table has one row with the header "Number of managers" and the value "3".

Number of managers
3

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

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

### QUERY:

```
select max(salary)-min(salary) difference from empB;
```

### OUTPUT:

The screenshot shows a SQL query execution interface. The query is:

```
1 select max(salary)-min(salary) difference from empB;
2
3
```

The results table has one row with the header "DIFFERENCE" and the value "30000".

DIFFERENCE
30000

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

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 empb where manager_id is not null group by manager_id  
having min(salary) >6000 order by min(salary) desc;
```

**OUTPUT:**

MANAGER_ID	MIN(SALARY)
5	80000
6	70000
5	60000

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,'YYYY'),1995,1,0)) "1995",sum(decode(to_char(hire_date,'YYYY'),1996,1,0)) "1996",sum(decode(to_char(hire_date,'YYYY'),1997,1,0)) "1997",sum(decode(to_char(hire_date,'YYYY'),1998,1,0)) "1998" from empb;
```

**OUTPUT:**

TOTAL	1995	1996	1997	1998
4	1	0	1	0

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 job_id "job", sum(decode(dept_id,20,salary))"Dept20",sum (decode(dept_id ,50, salary))  
"dept50",sum (decode(dept_id ,80, salary)) "dept80",sum (decode(dept_id ,90, salary)) "dept90",sum(salary)  
"TOTAL" from empb group by job_id
```

#### OUTPUT:

The screenshot shows a SQL query execution interface. The SQL code is:

```
1 select job_id "job", sum(decode(dept_id,20,salary))"Dept20",sum (decode(dept_id ,50, salary)) "dept50",  
2 sum (decode(dept_id ,80, salary)) "dept80",sum (decode(dept_id ,90, salary)) "dept90",sum(salary) "TOTAL" from empb group by job_id
```

The results table has columns: job, Dept20, dept50, dept80, dept90, and TOTAL. The data is:

job	Dept20	dept50	dept80	dept90	TOTAL
44	-	-	-	-	60000
65	-	-	-	-	90000
46	-	-	-	-	80000
47	-	-	70000	-	70000

4 rows returned in 0.01 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 "dept_name",d.loc as "department location", count(*) "Number of  
people",round(avg(salary),2) "salary" from dept111 d inner join empb e on(d.dpt_id =e.dept_id ) group by  
d.dept_name ,d.loc;
```

#### OUTPUT:

The screenshot shows a SQL query execution interface. The SQL code is:

```
1 select d.dept_name as "dept_name",d.loc as "department location",  
2 count(*) "Number of people",round(avg(salary),2) "salary" from dept111 d inner join empb e on(d.dept_id =e.dept_id ) group by d.dept_name ,d.loc;
```

The results table has columns: dept\_name, department location, Number of people, and salary. The data is:

dept_name	department location	Number of people	salary
CS	CHENNAI	1	70000
IT	MUMBAI	1	90000
IT	TORANTO	1	80000
CS	BANGLORE	1	60000

4 rows returned in 0.07 seconds

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

**RESULT:**

# SUB QUERIES

**EX\_NO:9**

**DATE:**

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

**QUERY:**

```
select last_name,hire_date from employees_table where department_id=(select department_id  
from employees_table where last_name like 'James') AND last_name <>'James';
```

**OUTPUT:**

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

```
1 select last_name,hire_date  
2 from employees_table  
3 where department_id=(  
4     select department_id from employees_table  
5     where last_name like 'James'  
6 )  
7 and last_name <>'James';
```

The results section shows the output:

LAST_NAME	HIRE_DATE
Janu	02/25/1998

1 rows returned in 0.01 seconds

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

**QUERY:**

```
select employee_id,last_name,salary from employees_table where salary>(select avg(salary) from  
employees_table ) order by salary;
```

**OUTPUT:**

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

```
1 select employee_id,last_name, salary  
2 from employees_table  
3 where salary > (  
4     select avg(salary) from employees_table  
5 )  
6 order by salary ;
```

The results section shows the output:

EMPLOYEE_ID	LAST_NAME	SALARY
81	Mohan	15000
205	James	16000

2 rows returned in 0.01 seconds

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

#### QUERY:

```
select employee_id, last_name from employees_table where department_id=(select department_id  
from employees_table where last_name like'%u%');
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The right side shows a user profile 'Deepika prabhak...' and the schema 'WKSP\_MYDEEPIKA@WORKS1'. The main area has tabs for 'SQL Commands' and 'Results'. The SQL command entered is:

```
1 select department_id, last_name  
2 from employees_table where department_id in(  
3     select department_id from employees_table  
4     where last_name like '%u%'  
5 )
```

The results tab shows the output:

DEPARTMENT_ID	LAST_NAME
67	Janu
67	James

Below the table, it says '2 rows returned in 0.01 seconds' and 'Download'.

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 e.last_name, e.department_id, job_id from employees_table where department_id=(select  
department_id from departments_table where location_id=1700);
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The right side shows a user profile 'Deepika prabhak...' and the schema 'WKSP\_MYDEEPIKA@WORKS1'. The main area has tabs for 'SQL Commands' and 'Results'. The SQL command entered is:

```
1 select e.department_id, e.last_name, e.job_id  
2 from employees_table e where e.department_id=(  
3     select d.department_id from departments_table d  
4     where d.location_id=1700 )  
5
```

The results tab shows the output:

DEPARTMENT_ID	LAST_NAME	JOB_ID
120	Davies	5678

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

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

#### QUERY:

```
select last_name,salary from employees_table where manager_id=(select manager_id from employee_table where manager_name='King');
```

#### OUTPUT:

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

```
1 SELECT last_name, salary
2 FROM employees_table
3 WHERE manager_id = (SELECT employee_id
4 FROM employees_table
5 WHERE last_name = 'King');
```

The results table has two columns: LAST\_NAME and SALARY. The data returned is:

LAST_NAME	SALARY
King	7000

1 rows returned in 0.01 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 department_id,last_name,job_id from employees where department_id in (select dept_id from departments where dept_name='Executive');
```

#### OUTPUT:

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

```
1 select last_name,job_id,department_id
2 from employees_table
3 where department_id IN(
4   select department_id from department_table
5   where department_name like '%Executive%'
6 )
```

The results table has three columns: LAST\_NAME, JOB\_ID, and DEPARTMENT\_ID. The data returned is:

LAST_NAME	JOB_ID	DEPARTMENT_ID
King	12345	67
Murugan	908	67

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 employee_id, last_name, salary from employees_table where salary > (select avg(salary) from employees_table where last_name like '%u%');
```

### 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. A search bar and a user profile for 'Deepika prabhak...' are also present. The main area is titled 'SQL Commands' and contains a code editor with the following query:

```
1 select employee_id, last_name, salary
2   from employees_table
3  where department_id IN(
4    select department_id from employees_table
5   where last_name like '%u%'
6   AND salary >(select avg(salary) from employees_table)
7 )
```

Below the code editor, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab displays the query results in a grid format:

EMPLOYEE_ID	LAST_NAME	SALARY
206	kingu	7000
205	Murugan	16000

Below the grid, it says '2 rows returned in 0.01 seconds' and has a 'Download' link. At the bottom of the page, there are footer links for email (220701058@rajalakshmi.edu.in), workspace (mydeepikaworkspace), and language (en). The copyright notice reads 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version is 'Oracle APEX 23.2.4'.

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

### RESULT:

# USING THE SET OPERATORS

EX\_NO:10

DATE:

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

**QUERY:**

```
select department_id from department_table minus select department_id from employees_table  
where job_id='st_clerk';
```

**OUTPUT:**

The screenshot shows the Oracle APEX interface with the SQL Workshop tab selected. The query entered is:

```
1 select department_id  
2 from department_table  
3 Minus  
4 select department_id  
5 from employees_table  
6 where job_id='ST_CLERK';
```

The results section displays the output:

DEPARTMENT_ID
67
100

2 rows returned in 0.04 seconds

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

**QUERY:**

```
select country_id,state_province from location minus select country_id,state_province from  
location,departments where location.location_id=departments.location_id;
```

**OUTPUT:**

The screenshot shows the Oracle APEX interface with the SQL Workshop tab selected. The query entered is:

```
1 select country_id,country_name  
2 from countries  
3 minus  
4 select l.country_id,c.c.country_name  
5 from countries c,location_table l  
6 where c.country_id=l.country_id
```

The results section displays the output:

COUNTRY_ID	COUNTRY_NAME
3	Italy
4	Russia

2 rows returned in 0.02 seconds

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

#### QUERY:

```
select job_id,department_id from employees_table where department_id=10 union all  
select job_id,department_id from employees_table where department_id=50 union all  
select job_id,department_id from employees_table where department_id=20 ;
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile for 'Deepika prabhak...' and the schema 'WKSP\_MYDEEPIKAWORKS1'. The main area has tabs for 'SQL Commands' and 'Results'. In the 'SQL Commands' tab, the following SQL code is entered:

```
1 select job_id,department_id from employees_table  
2 where department_id=10  
3 UNION all  
4 select job_id,department_id from employees_table  
5 where department_id=50  
6 UNION all  
7 select job_id,department_id from employees_table  
8 where department_id=20;
```

The 'Results' tab displays the output in a grid format:

JOB_ID	DEPARTMENT_ID
12345	10
908	50
1700	20

At the bottom, the footer includes the email '220701058@rajalakshmi.edu.in', the workspace name 'mydeeppikaworkspace', the language 'en', copyright information 'Copyright © 1999, 2023, Oracle and/or its affiliates.', and the version 'Oracle APEX 23.2.4'.

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

#### QUERY:

```
select job_id,employee_id from employees_table intersect select e.job_id,e.employee_id from employees_table e,job_history j where e.job_id=j.old_job_id;
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface, identical to the previous one but with a different query. The top navigation bar and user information are the same. The 'SQL Commands' tab contains the following SQL code:

```
1 SELECT job_id, employee_id  
2 FROM employees_table  
3 INTERSECT  
4 SELECT e.job_id, e.employee_id  
5 FROM employees_table e  
6 JOIN job_history j ON e.job_id = j.old_job_id;  
7
```

The 'Results' tab shows the output:

JOB_ID	EMPLOYEE_ID
110	2
123	20

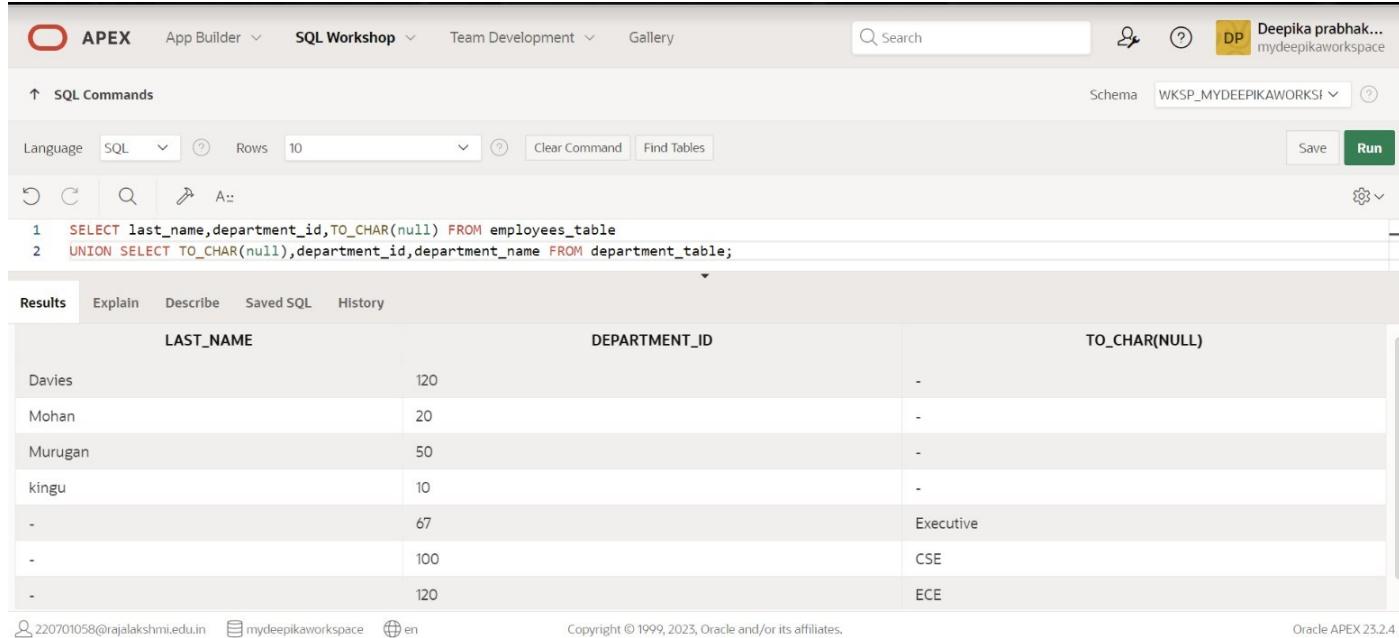
The footer includes the email '220701058@rajalakshmi.edu.in', the workspace name 'mydeeppikaworkspace', the language 'en', copyright information 'Copyright © 1999, 2023, Oracle and/or its affiliates.', and the version 'Oracle APEX 23.2.4'.

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 last_name,department_id ,TO_CHAR(null) from employees_table union select
department_name,department_id from department_table;
```

#### OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', 'Search', and user information 'Deepika prabhak... mydeepikaworkspace'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_MYDEEPIKAWORKSP'. Below the toolbar are buttons for Language (SQL), Rows (10), Clear Command, and Find Tables. The SQL editor contains two lines of code:

```
1 SELECT last_name,department_id,TO_CHAR(null) FROM employees_table
2 UNION SELECT TO_CHAR(null),department_id,department_name FROM department_table;
```

The results section shows a table with three columns: LAST\_NAME, DEPARTMENT\_ID, and TO\_CHAR(NULL). The data is as follows:

LAST_NAME	DEPARTMENT_ID	TO_CHAR(NULL)
Davies	120	-
Mohan	20	-
Murugan	50	-
kingu	10	-
-	67	Executive
-	100	CSE
-	120	ECE

At the bottom, there are footer links for '220701058@rajalakshmi.edu.in', 'mydeepikaworkspace', 'en', 'Copyright © 1999, 2023, Oracle and/or its affiliates.', and 'Oracle APEX 23.2.4'.

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

#### RESULT:

# 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 employee_id, last_name employee,
department_id FROM employees;
```

## OUTPUT:

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

```
1 CREATE OR REPLACE VIEW employees_vu AS
2  SELECT employee_id, last_name employee, department_id
3  FROM employees_table
```

Below the code, the results pane shows the message "View created." and "0.07 seconds". At the bottom, the URL is 22070105@rajalakshmi.edu.in and the workspace is mydeepikaworkspace.

2.) Display the contents of the EMPLOYEES\_VU view.

## QUERY:

```
select * from employees_vu;
```

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface displaying the results of the query. The results are presented in a grid format:

EMPLOYEE_ID	EMPLOYEE	DEPARTMENT_ID
1	kingu	10
10	Mohan	20
2	Munagan	50
20	Davies	120

At the bottom, it says "4 rows returned in 0.00 seconds". The URL and workspace information are at the bottom of the page.

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 APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The schema dropdown is set to 'WKSP\_MYDEEPIKWORKS1'. The SQL command input field contains the query: 'SELECT view\_name, text FROM user\_views;'. The results section displays three rows of data:

VIEW_NAME	TEXT
EMPLOYEE_VIEW	SELECT * FROM employees WHERE department = 'IT'
SALES_VIEW	SELECT * FROM sales WHERE year = 2023
PRODUCT_VIEW	SELECT p.product_name, s.quantity_sold, s.sale_date FROM products p JOIN sales s ON p.product_id = s.product_id

Below the results, it says '3 rows returned in 0.01 seconds'.

4.)Using your EMPLOYEES\_VU view, enter a query to display all employees names and department

#### QUERY:

```
SELECT employee, department_id FROM employees_vu;
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The schema dropdown is set to 'WKSP\_MYDEEPIKWORKS1'. The SQL command input field contains the query: 'SELECT employee, department\_id FROM employees\_vu;'. The results section displays four rows of data:

EMPLOYEE	DEPARTMENT_ID
Iingu	10
Mohan	20
Munagan	50
Davies	120

Below the results, it says '4 rows returned in 0.03 seconds'.

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 employee_id empno, last_name employee, department_id  
deptno FROM employees WHERE department_id = 50 WITH CHECK OPTION CONSTRAINT  
emp_dept_50;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is active. The main area is titled 'SQL Commands'. The schema dropdown shows 'WKSP\_MYDEEPIKAWORKS1'. The code editor contains the following SQL command:

```
1 create view dept50 as  
2 select employee_id empno,  
3 last_name employee,  
4 department_id deptno  
5 from employees table  
6 where department_id=50  
7 WITH CHECK OPTION CONSTRAINT emp_dept_50;  
8
```

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

```
View created.  
0.05 seconds
```

At the bottom of the page, the URL is '220704058@rajatashmi.edu.in mydeepikaworkspace' and the page footer says 'Copyright © 1999-2023, Oracle and/or its affiliates. Oracle APEX 23.2.4'.

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

**QUERY:**

```
Describe dept50;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is active. The main area is titled 'SQL Commands'. The schema dropdown shows 'WKSP\_MYDEEPIKAWORKS1'. The code editor contains the following SQL command:

```
1 DESCRIBE dept50  
2  
3 select * from dept50
```

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

```
Object Type: VIEW Object: DEPT50  
Table Column Data Type Length Precision Scale Primary Key Nullable Default Comment  
DEPT50 EMPNO NUMBER - 8 0 - - -  
EMPLOYEE VARCHAR2 25 - - - - -  
DEPTNO NUMBER - 4 0 - ✓ - -
```

At the bottom of the page, the URL is '220704058@rajatashmi.edu.in mydeepikaworkspace' and the page footer says 'Copyright © 1999-2023, Oracle and/or its affiliates. Oracle APEX 23.2.4'.

## 7.) Attempt to reassign Matos to department 80

### QUERY:

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

### OUTPUT:

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

```
1 CREATE OR REPLACE VIEW dept50 AS
2   SELECT employee_id AS empno,
3         last_name AS employee,
4         department_id AS deptno
5   FROM employees_table
6   WHERE department_id = 80
7   WITH CHECK OPTION CONSTRAINT emp_dept_50;
8
9
10 select * from dept50
11
```

The results section shows a single row returned by the query:

EMPNO	EMPLOYEE	DEPTNO
2	Matos	80

1 rows returned in 0.00 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.last_name "Employee",d.dept_name Department,
e.salary "Salary",j.grade_level "Grades" from employees e,departments d,job_grade j where
e.department_id=d.dept_id and e.salary between j.lowest_sal and j.highest_sal;
```

### OUTPUT:

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

```
1 CREATE OR REPLACE VIEW salary_vu
2   AS
3   SELECT e.last_name "Employee",
4         d.department_name "Department",
5         e.salary "Salary",
6         j.grade_level "Grades"
7   FROM employees_table e,
8        departments_table d,
9        job_grades j
10  WHERE e.department_id = d.department_id
11    AND e.salary BETWEEN j.lowest_sal AND j.highest_sal;
12
13  select * from SALARY_VU
```

The results section shows a single row returned by the query:

Employee	Department	Salary	Grades
Davies	ECE	4000	grade 1

1 rows returned in 0.02 seconds

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

**RESULT:**

# 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. Use "(nullable)" to indicate those columns that can have null values.

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

CREATE TABLE

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				

f\_global\_locations

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	

```
( 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_NUMBER	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-MonYYYY'));
```

SELECT \* FROM animals;

5. Write the syntax to create a foreign key (adoption\_id) in the animals table that has a corresponding primarykey 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	

# 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:**

# 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?

Results	Explain	Describe	Saved SQL	History									
<table border="1"><thead><tr><th>ID</th><th>Song Title</th><th>ARTIST</th></tr></thead><tbody><tr><td>47</td><td>Hurrah for Today</td><td>The Jubilant Trio</td></tr><tr><td>49</td><td>Lets Celebrate</td><td>The Celebrants</td></tr></tbody></table>					ID	Song Title	ARTIST	47	Hurrah for Today	The Jubilant Trio	49	Lets Celebrate	The Celebrants
ID	Song Title	ARTIST											
47	Hurrah for Today	The Jubilant Trio											
49	Lets Celebrate	The Celebrants											
2 rows returned in 0.00 seconds				Download									

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:**



# 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 a SQL query editor interface. The top bar includes 'Language' (set to SQL), 'Rows' (set to 10), 'Clear Command', 'Find Tables', 'Save', and 'Run' buttons. Below the toolbar is a toolbar with icons for refresh, search, and other functions. The main area contains the SQL command: 'CREATE SEQUENCE dept\_id\_seq START WITH 200 INCREMENT BY 10 MAXVALUE 1000;'. The results section below shows the output: 'Sequence created.' and a execution time of '0.02 seconds'.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a SQL query editor interface. The top bar includes 'Language' (set to SQL), 'Rows' (set to 10), 'Clear Command', 'Find Tables', 'Save', and 'Run' buttons. Below the toolbar is a toolbar with icons for refresh, search, and other functions. The main area contains the SQL command: 'SELECT sequence\_name, max\_value, increment\_by, last\_number FROM user\_sequences;'. The results section below displays a table with four columns: SEQUENCE\_NAME, MAX\_VALUE, INCREMENT\_BY, and LAST\_NUMBER. The single row shows: DEPT\_ID\_SEQ, 1000, 10, and 200. A note at the bottom indicates '1 rows returned in 0.02 seconds'.

SEQUENCE_NAME	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
DEPT_ID_SEQ	1000	10	200

3.) Write a script to insert two rows into the DEPT table. Name your script lab12\_3.sql. Be sure to use the sequence that you created for the ID column. Add two departments named Education and Administration. Confirm your additions. Run the commands in your script.

**QUERY:**

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

### OUTPUT:

SQL Scripts \ Results			
Script: ex14	Status: Complete	Create App Edit Script	
View:	Detail Summary	Rows 15	Go
1	0.02	INSERT INTO dept VALUES (dept_id_seq.nextval, 'Education')	1 row(s) inserted.
Download			
row(s) 1 - 1 of 1			
1	1	Successful	0
Statements Processed		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 EMPLOYEES (department_id);
```

### OUTPUT:

Language SQL		Rows 10	Clear Command	Find Tables	Save Run
↻	C	Q	A..	✖	✖
1	CREATE INDEX emp_dept_id_idx ON EMPLOYEES (department_id);				
Index created.					
0.03 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:

Language SQL Rows 10 Clear Command Find Tables Save Run

1 SELECT index\_name,table\_name,uniqueness FROM user\_indexes WHERE table\_name='EMPLOYEES';

**Results** Explain Describe Saved SQL History

INDEX_NAME	TABLE_NAME	UNIQUENESS
EMP_DEPT_ID_IDX	EMPLOYEES	NONUNIQUE

1 rows returned in 0.04 seconds [Download](#)

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

**RESULT:**

# CONTROLLING USER ACCESS

**EX\_NO:15**

**DATE:**

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

The CREATE SESSION system privilege

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

The CREATE TABLE privilege

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

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

4. You are the DBA. You are creating many users who require the same system privileges. What should you use to make your job easier?

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

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

The ALTER USER statement

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

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

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

7. Query all the rows in your DEPARTMENTS table.

SELECT \* FROM departments;

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

Team 1 executes this INSERT statement. INSERT INTO departments(department\_id, department\_name) VALUES (500, 'Education'); COMMIT;

Team 2 executes this INSERT statement. INSERT INTO departments(department\_id, department\_name) VALUES (510, 'Administration'); COMMIT;

9. Query the USER\_TABLES data dictionary to see information about the tables that you own. SELECT table\_name FROM user\_tables;

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

Team 1 revokes the privilege.

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

Team 2 revokes the privilege.

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

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

Team 1 executes this INSERT statement.

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

Team 2 executes this INSERT statement.

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

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

**RESULT:**



# PL/SQL

## CONTROL STRUCTURES

**EX\_NO:16**

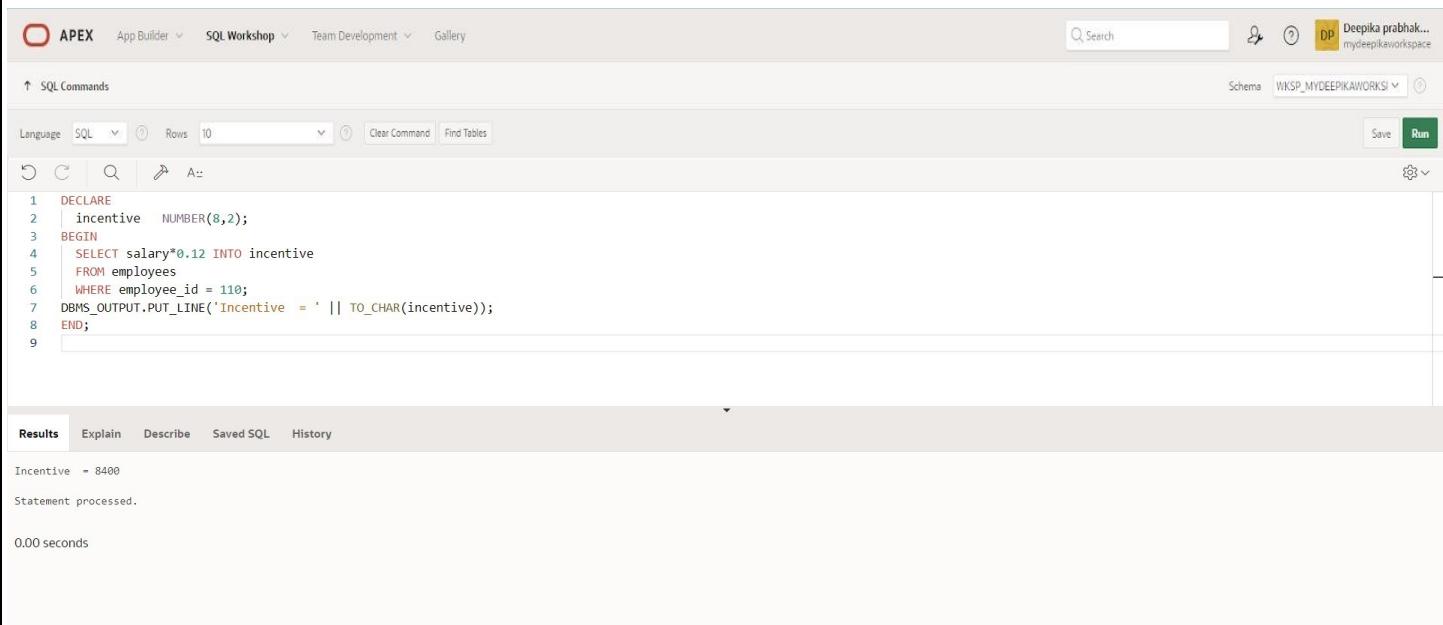
**DATE:**

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area displays the PL/SQL code for calculating an incentive. Below the code, the 'Results' tab is active, showing the output: 'Incentive = 8400'. The status message 'Statement processed.' is also visible. The bottom of the screen shows a timer indicating '0.00 seconds'.

```
1 DECLARE
2   incentive NUMBER(8,2);
3 BEGIN
4   SELECT salary*0.12 INTO incentive
5   FROM employees
6   WHERE employee_id = 110;
7   DBMS_OUTPUT.PUT_LINE('Incentive = ' || TO_CHAR(incentive));
8 END;
9 
```

Incentive = 8400  
Statement processed.  
0.00 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:**

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

```
1 DECLARE
2   "WELCOME" varchar2(10) := 'welcome';
3 BEGIN
4   DBMS_Output.Put_Line("Welcome");
5 END;
6 /
7
```

In the results pane, an error message is displayed in a yellow box:

```
Error at line 4/25: ORA-06550: line 4, column 25:
PLS-00201: identifier 'Welcome' must be declared
ORA-06512: at "SYS.0MW_DBMS_SQL_APEX_230200", line 801
ORA-06550: line 4, column 3:
PL/SQL: Statement ignored
```

Below the error message, the valid code is shown again:

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

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

```
1 DECLARE
2   | WELCOME varchar2(10) := 'welcome';
3 BEGIN
4   | DBMS_Output.Put_Line("Welcome");
5 END;
6 /
7
```

In the results pane, an error message is displayed in a yellow box:

```
Error at line 4/25: ORA-06550: line 4, column 25:
PLS-00201: identifier 'Welcome' must be declared
ORA-06512: at "SYS.0MW_DBMS_SQL_APEX_230200", line 801
ORA-06550: line 4, column 3:
PL/SQL: Statement ignored
```

Below the error message, the valid code is shown again:

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

3.)

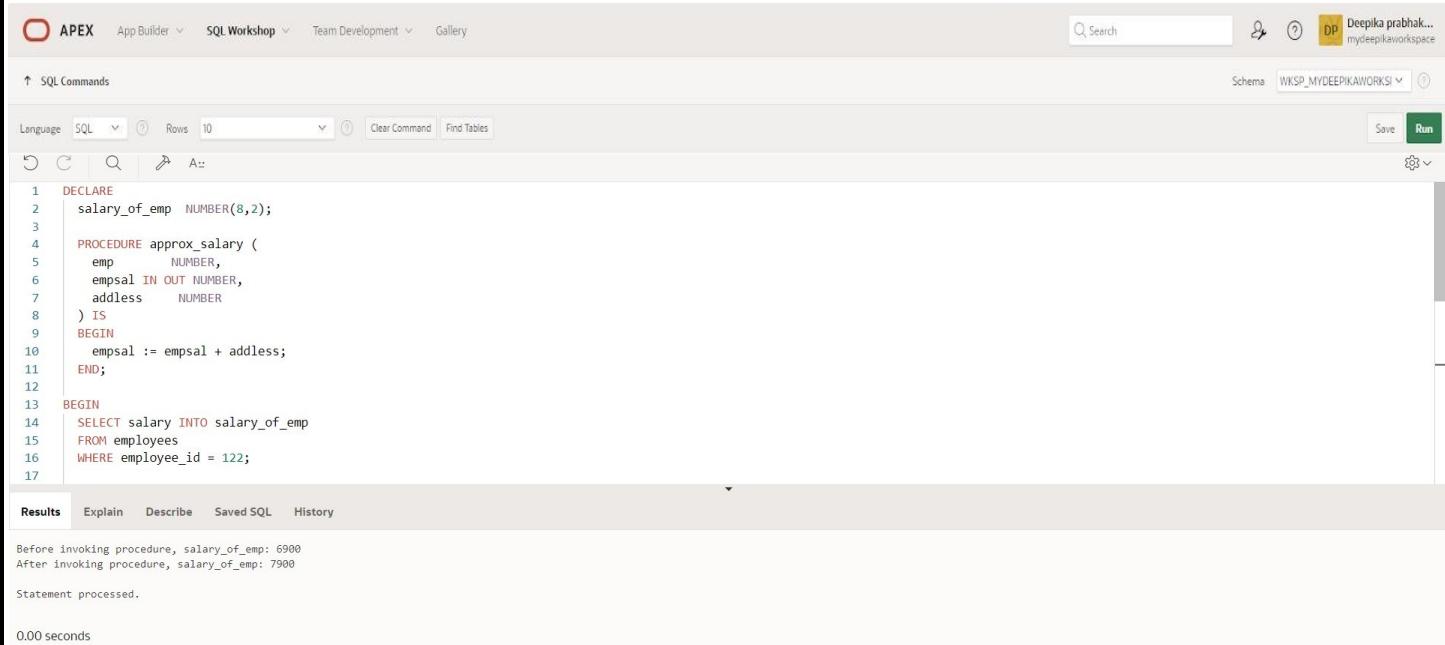
Write a PL/SQL block to adjust the salary of the employee whose ID 122.

#### 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\_MYDEEPIKAWORKS1 and the user as Deepika prabhak... mydeepikaworkspace. The main area displays the PL/SQL code with syntax highlighting. The code declares a variable salary\_of\_emp, defines a procedure approx\_salary with parameters emp, empsal (IN OUT), and addless, and then performs a SELECT statement to find the salary for employee\_id 122. It also includes DBMS\_OUTPUT.PUT\_LINE statements to print the salary before and after invoking the procedure. The code is numbered from 1 to 17. Below the code, the Results tab is selected, showing the output of the executed statements. The output indicates that before invoking the procedure, the salary\_of\_emp is 6900, and after invoking it with a call to approx\_salary(100, salary\_of\_emp, 1000), the salary\_of\_emp becomes 7900. The total execution time is 0.00 seconds.

```
1  DECLARE
2      salary_of_emp NUMBER(8,2);
3
4  PROCEDURE approx_salary (
5      emp      NUMBER,
6      empsal IN OUT NUMBER,
7      addless  NUMBER
8  ) IS
9  BEGIN
10     empsal := empsal + addless;
11
12
13 BEGIN
14     SELECT salary INTO salary_of_emp
15     FROM employees
16     WHERE employee_id = 122;
17
Results Explain Describe Saved SQL History
Before invoking procedure, salary_of_emp: 6900
After invoking procedure, salary_of_emp: 7900
Statement processed.
0.00 seconds
```

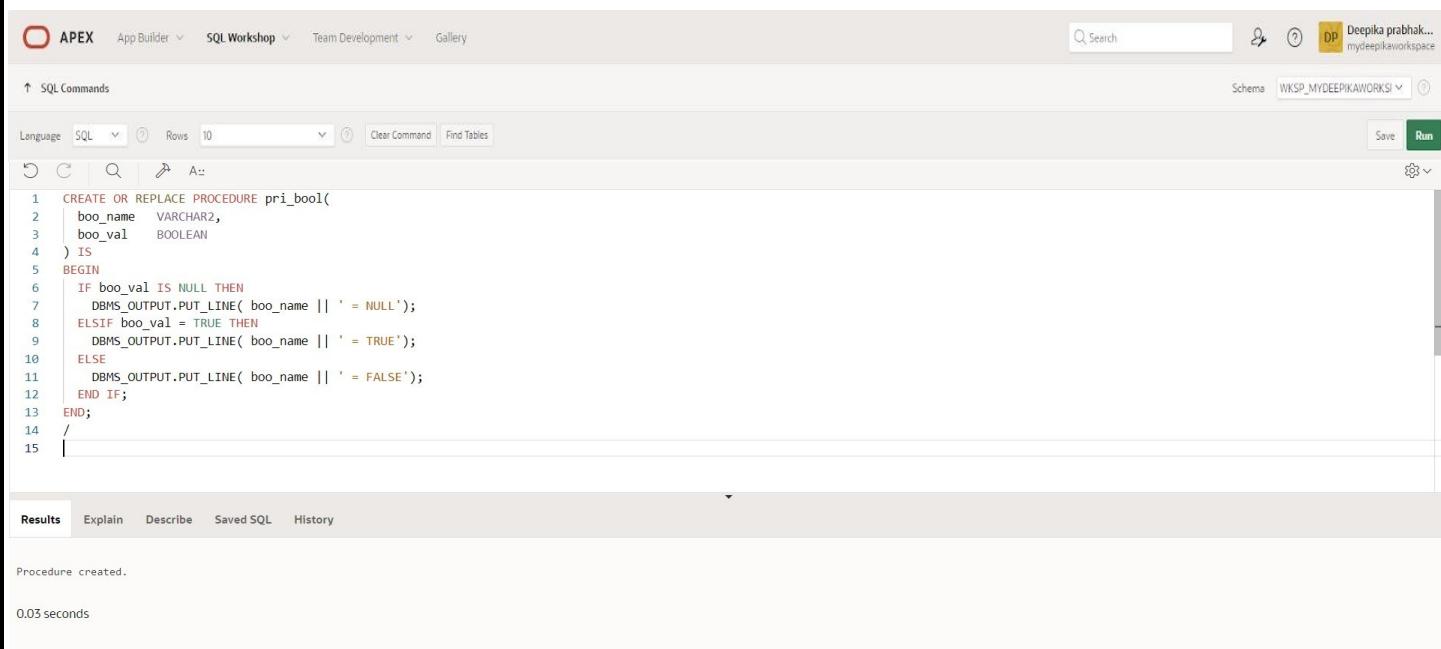
4.)

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 PL/SQL code for the 'pri\_bool' procedure. The code uses the 'DBMS\_OUTPUT.PUT\_LINE' function to print the value of 'boo\_name' followed by its state ('NULL', 'TRUE', or 'FALSE'). The code is correctly executed, as indicated by the 'Procedure created.' message in the results pane. The execution time is listed as '0.03 seconds'.

```
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;
14 /
```

Results Explain Describe Saved SQL History

Procedure created.

0.03 seconds

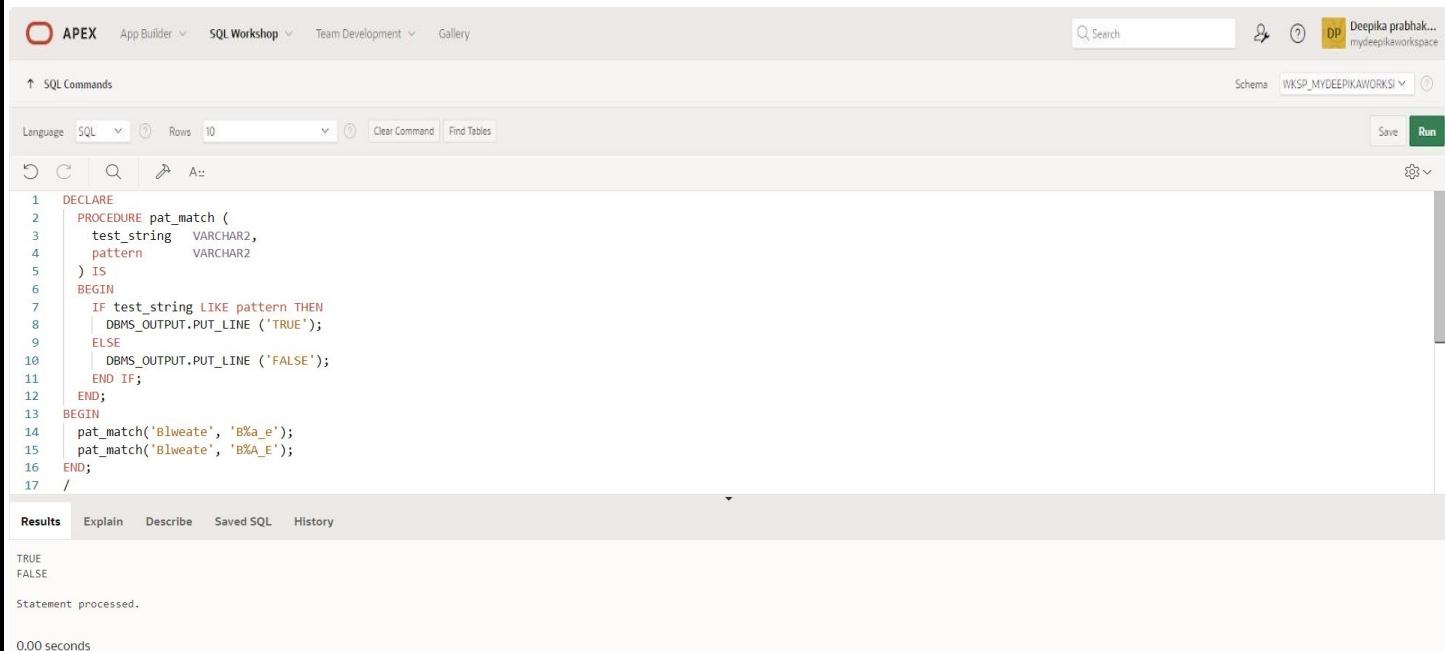
5.)

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 the user's profile (Deepika prabhak... mydeepikaworkspace). The main area is a SQL editor with the following content:

```
1 DECLARE
2 PROCEDURE pat_match (
3   test_string  VARCHAR2,
4   pattern      VARCHAR2
5 ) IS
6 BEGIN
7   IF test_string LIKE pattern THEN
8     | DBMS_OUTPUT.PUT_LINE ('TRUE');
9   ELSE
10     DBMS_OUTPUT.PUT_LINE ('FALSE');
11   END IF;
12 END;
13 BEGIN
14   pat_match('Blweate', 'B%a_e');
15   pat_match('Blweate', 'B%A_E');
16 END;
17 /
```

The results tab shows the output of the executed code:

```
TRUE
FALSE
```

Below the results, a message states "Statement processed." and "0.00 seconds".

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. On the right, there are search, help, and user profile icons. The main area has tabs for SQL Commands, Results (selected), Explain, Describe, Saved SQL, and History. The SQL Commands tab displays the PL/SQL code with line numbers. The Results tab shows the output of the executed code.

```
2  num_small NUMBER := 8;
3  num_large NUMBER := 5;
4  num_temp NUMBER;
5  BEGIN
6
7  IF num_small > num_large THEN
8    num_temp := num_small;
9    num_small := num_large;
10   num_large := num_temp;
11  END IF;
12
13 DBMS_OUTPUT.PUT_LINE ('num_small = '||num_small);
14 DBMS_OUTPUT.PUT_LINE ('num_large = '||num_large);
15 END;
16 /
17
```

Results

```
num_small = 5
num_large = 8

Statement processed.

0.00 seconds
```

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;
```

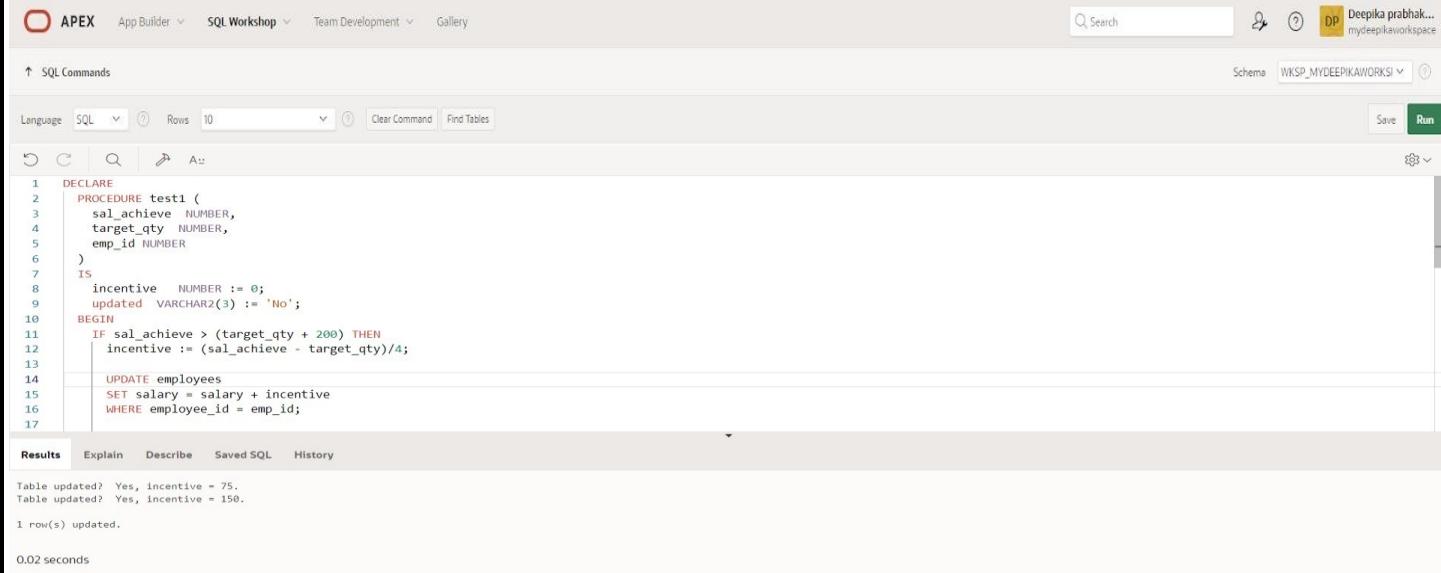
BEGIN

```
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, Team Development, and Gallery. On the right, there are search, schema selection, and workspace dropdowns. The main area is titled 'SQL Commands' and contains the PL/SQL code from the previous steps. The code is executed successfully, with the output showing two rows updated and the incentive values printed. The bottom status bar indicates 1 row(s) updated and a execution time of 0.02 seconds.

```
1 DECLARE
2     PROCEDURE test1 (
3         sal_achieve NUMBER,
4         target_qty NUMBER,
5         emp_id NUMBER
6     )
7     IS
8         incentive NUMBER := 0;
9         updated VARCHAR2(3) := 'No';
10    BEGIN
11        IF sal_achieve > (target_qty + 200) THEN
12            incentive := (sal_achieve - target_qty)/4;
13            UPDATE employees
14            SET salary = salary + incentive
15            WHERE employee_id = emp_id;
16        updated := 'Yes';
17    END IF;
18    DBMS_OUTPUT.PUT_LINE (
19        'Table updated? ' || updated || ',' ||
20        'incentive = ' || incentive || '
21    );
22 END test1;
```

Results Explain Describe Saved SQL History

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

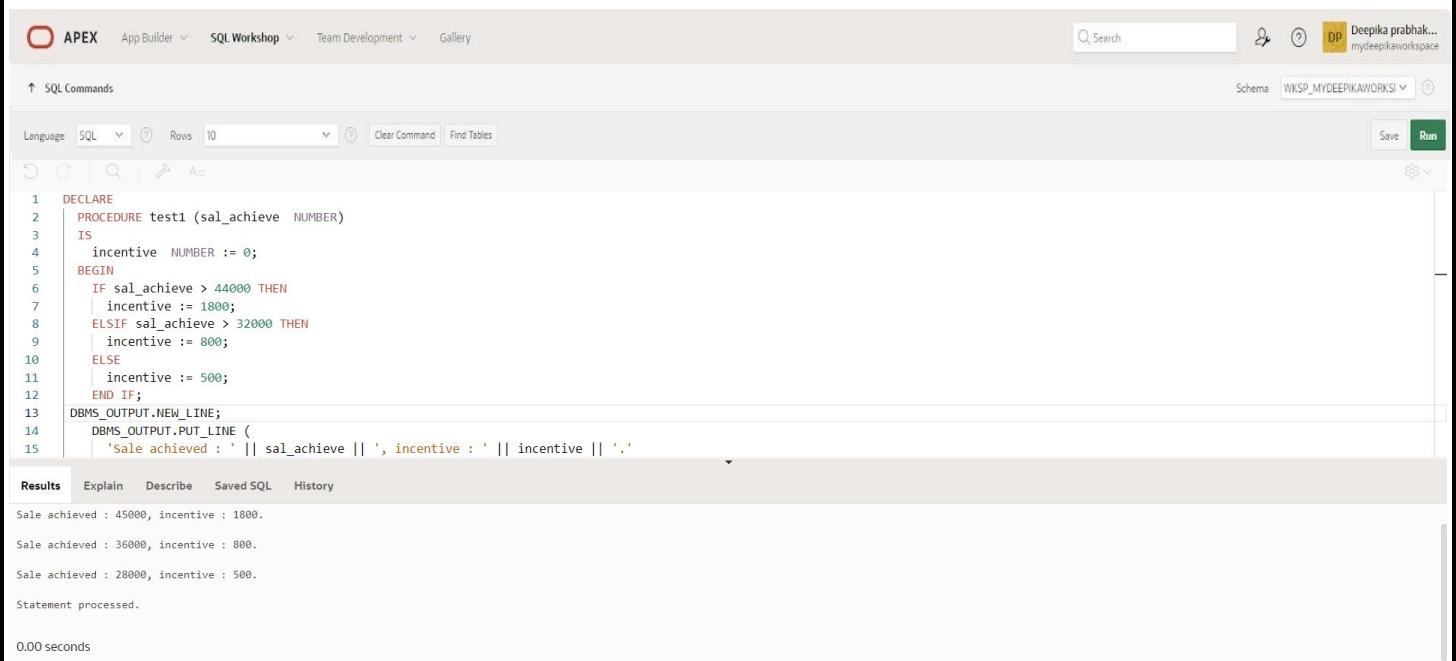
1 row(s) updated.  
0.02 seconds

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;
/
```



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows a user profile for 'Deepika prabhak...' with a workspace named 'mydeepikaworkspace'. The main area is titled 'SQL Commands' with tabs for Language, SQL, Rows, Clear Command, Find Tables, Save, and Run. The code editor contains the PL/SQL procedure 'test1' and its call from the 'BEGIN' block. The results tab displays the output of the executed code, showing three rows of sales and incentives: 45000, 36000, and 28000. The bottom status bar indicates '0.00 seconds'.

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

Results Explain Describe Saved SQL History
Sale achieved : 45000, incentive : 1800.
Sale achieved : 36000, incentive : 800.
Sale achieved : 28000, incentive : 500.
Statement processed.

0.00 seconds
```

9.)

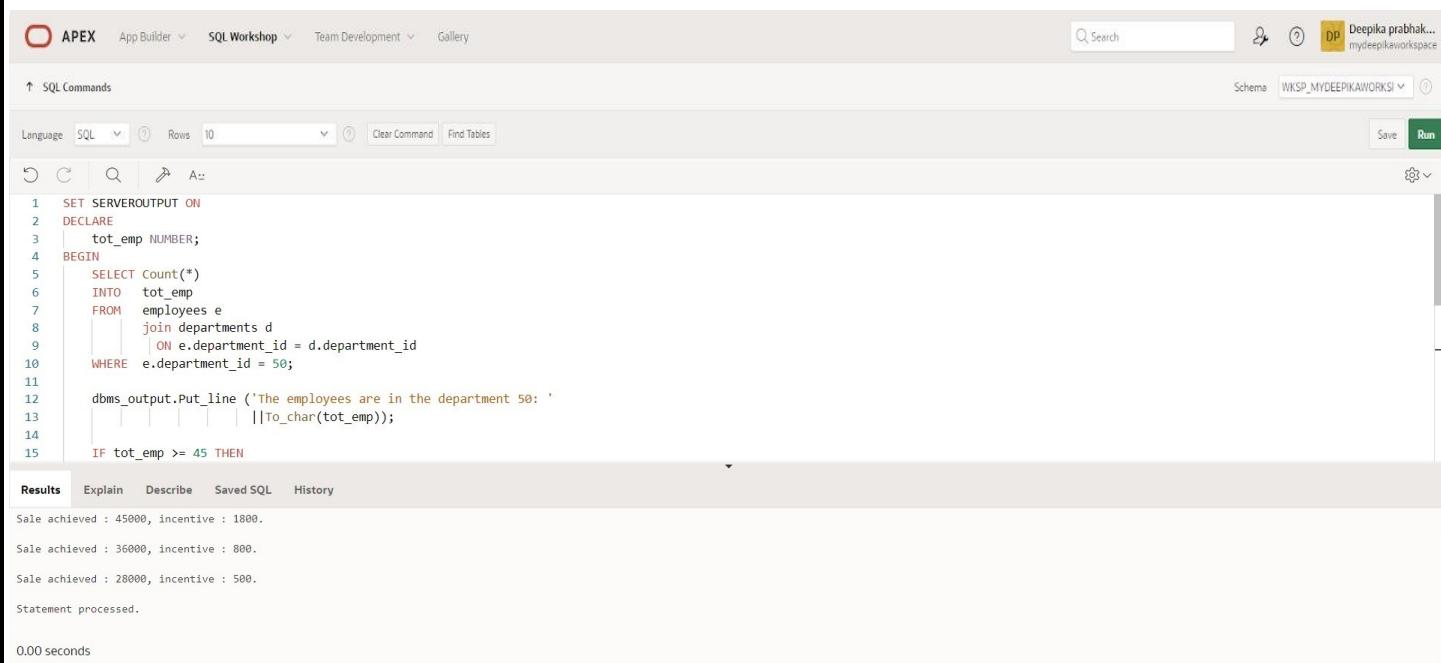
Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

**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:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a user profile for 'Deepika prabhak...' and a schema dropdown for 'WKSP\_MYDEEPIKAWORKS1'. The main area is titled 'SQL Commands' with a language selector set to SQL. Below the title are standard edit tools: Undo, Redo, Find, Replace, and Auto. The SQL editor contains the provided PL/SQL code. The code is numbered from 1 to 15. Lines 1-14 show the declaration of variables, setting up a cursor, and performing a select into. Line 15 starts an if-then block. The bottom pane is the 'Results' tab, which displays the output of the executed code. The output shows three rows of data: 'Sale achieved : 45000, incentive : 1800.', 'Sale achieved : 36000, incentive : 800.', and 'Sale achieved : 28000, incentive : 500.'. It also indicates 'Statement processed.' and a execution time of '0.00 seconds'.

```
1 SET SERVEROUTPUT ON
2 DECLARE
3     tot_emp NUMBER;
4 BEGIN
5     SELECT Count(*)
6         INTO tot_emp
7         FROM employees e
8         join departments d
9             ON e.department_id = d.department_id
10    WHERE e.department_id = 50;
11
12    dbms_output.Put_line ('The employees are in the department 50: '
13                           ||To_char(tot_emp));
14
15    IF tot_emp >= 45 THEN
```

## 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:

```
APEX App Builder SQL Workshop Team Development Gallery Search Schema WKSP_MYDEEPIKAWORKSPACE Run  
↑ SQL Commands Language SQL Rows 10 Clear Command Find Tables Save  
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));
```

The employees are in the department 80 is: 4  
There are 41 vacancies in department 80  
Statement processed.  
0.03 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:

```
Employee ID | Full Name | Job Title | Hire Date | Salary
-----|-----|-----|-----|-----
110 | John Smith | sales_rep | 02/28/1995 | 70000
125 | Emily Johnson | hr_rep | 04/26/1998 | 50000
122 | Jaunty Janu | ac_account | 03/05/2024 | 69000
114 | den Davies | st_clerk | 02/03/1999 | 11000
142 | Jane Doe | ac_account | 03/05/1997 | 30000
115 | Vijaya Mohan | st_clerk | 02/22/1998 | 40000
```

Statement processed.

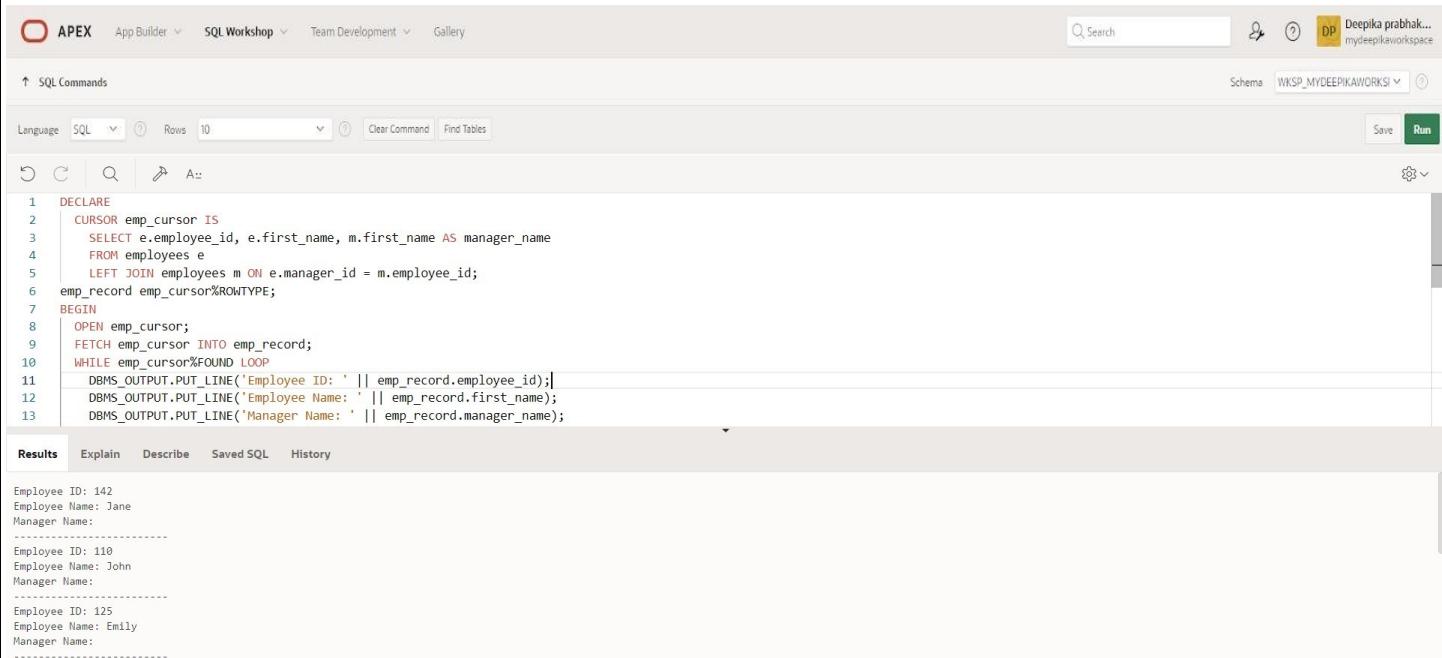
## 12.)

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:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema is set to WKSP\_MYDEEPIKAWORKS1. The main area displays the PL/SQL code and its execution results.

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

The results pane shows the output of the PL/SQL code:

```
Employee ID: 142
Employee Name: Jane
Manager Name:
-----
Employee ID: 110
Employee Name: John
Manager Name:
-----
Employee ID: 125
Employee Name: Emily
Manager Name:
-----
```

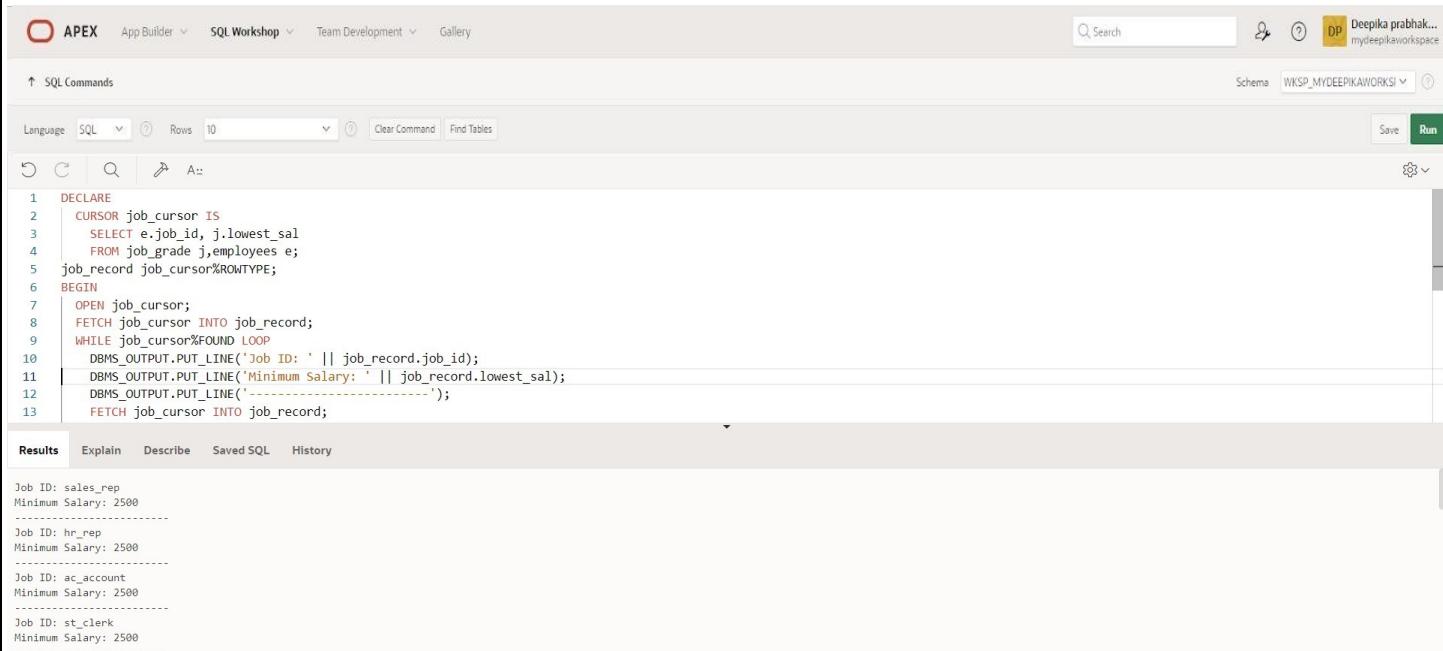
### 13.)

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. On the right, there's a search bar, a user icon, and a workspace named "Deepika prabhak... mydeepikaworkspace". The main area has tabs for Language (SQL selected), Rows (10), Clear Command, Find Tables, Save, and Run. Below these are icons for Undo, Redo, Search, and Run. The code editor displays a PL/SQL block with numbered lines 1 through 13. Lines 1-5 define a cursor and declare a record type. Lines 6-13 implement a loop to fetch records and output them using DBMS\_OUTPUT.PUT\_LINE. The results pane at the bottom shows the output of the executed code, listing four job entries: sales\_rep, hr\_rep, ac\_account, and st\_clerk, each with a minimum salary of 2500.

```
1  DECLARE
2  CURSOR job_cursor IS
3    SELECT e.job_id, j.lowest_sal
4      FROM job_grade j,employees e;
5  job_record job_cursor%ROWTYPE;
6 BEGIN
7  OPEN job_cursor;
8  FETCH job_cursor INTO job_record;
9  WHILE job_cursor%FOUND LOOP
10   DBMS_OUTPUT.PUT_LINE('Job ID: ' || job_record.job_id);
11   DBMS_OUTPUT.PUT_LINE('Minimum Salary: ' || job_record.lowest_sal);
12   DBMS_OUTPUT.PUT_LINE('-----');
13   FETCH job_cursor INTO job_record;
```

Results	Explain	Describe	Saved SQL	History
Job ID: sales_rep Minimum Salary: 2500 ----- Job ID: hr_rep Minimum Salary: 2500 ----- Job ID: ac_account Minimum Salary: 2500 ----- Job ID: st_clerk Minimum Salary: 2500 -----				

**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:**

```
APEX App Builder SQL Workshop Team Development Gallery Search Schema WKSP_MYDEEPAKWORKS! Run
SQL Commands Language SQL Rows 10 Clear Command Find Tables
1 DECLARE
2     CURSOR employees_cur IS
3         SELECT employee_id,
4             last_name,
5             job_id,
6             start_date
7         FROM employees
8             NATURAL JOIN job_history;
9     emp_start_date DATE;
10 BEGIN
11     dbms_output.Put_line(Rpad('Employee ID', 15)
12                         || Rpad('Last Name', 25)
13                         || Rpad('Job Id', 35)
14                         || 'Start Date');
15
16     dbms_output.Put_line('-----');
17
Results Explain Describe Saved SQL History
Employee ID Last Name Job Id Start Date
125 Johnson hr_rep 22-apr-1999
125 Johnson hr_rep 22-apr-1999
Statement processed.
```

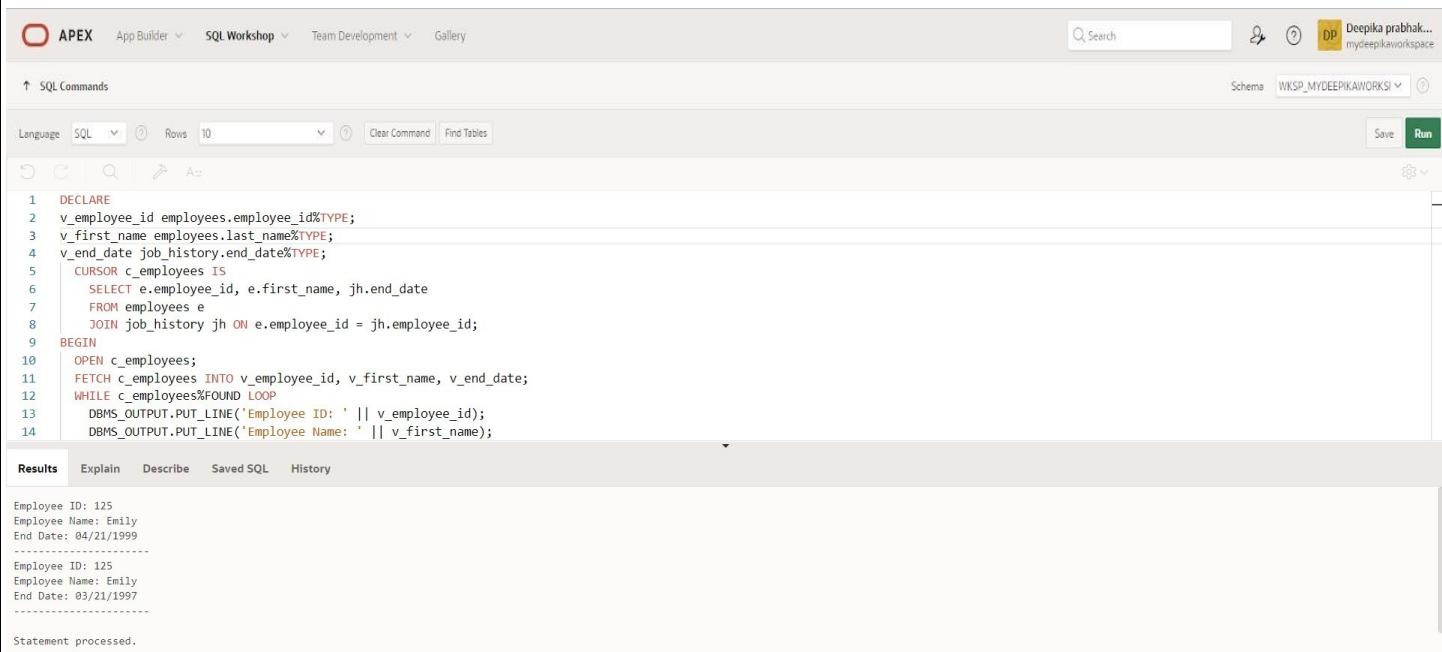
## 15.)

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, Team Development, and Gallery. The right side shows the schema as WKSP\_MYDEEPIKAWORKSPACE and a run button. The main area is a SQL editor with tabs for Language (SQL), Rows (10), Clear Command, and Find Tables. The code area contains the PL/SQL block from above. The results tab at the bottom shows the output of the program, which consists of two rows of data. Each row contains the Employee ID, Employee Name, and End Date, separated by a vertical line and followed by a horizontal line.

```
Employee ID: 125
Employee Name: Emily
End Date: 04/21/1999
-----
Employee ID: 125
Employee Name: Emily
End Date: 03/21/1997
-----
Statement processed.
```

**RESULT:**

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

# 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. In the top navigation bar, 'APEX' is selected. The main area displays a PL/SQL block:

```
1 DECLARE
2     fac NUMBER := 1;
3     n NUMBER := :1; -- Use a bind variable instead of "&1"
4 BEGIN
5     WHILE n > 0 LOOP
6         fac := n * fac;
7         n := n - 1;
8     END LOOP;
9     DBMS_OUTPUT.PUT_LINE(fac);
10 END;
```

The code is highlighted in green. Below the code, the results section shows the output:

120  
Statement processed.  
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. At the top, there are tabs for APEX, App Builder, SQL Workshop (which is selected), Team Development, and Gallery. On the right, there's a search bar, a help icon, and a user profile for "Deepika prabhak...". The main area is titled "SQL Commands" and shows a code editor with the following PL/SQL block:

```
1  DECLARE
2      v_book_id NUMBER := 1;
3      v_title VARCHAR2(100);
4      v_author VARCHAR2(100);
5      v_year_published NUMBER;
6  BEGIN
7      v_title := 'Initial Title';
8
9      get_book_info(p_book_id => v_book_id, p_title => v_title, p_author => v_author, p_year_published => v_year_published);
10
11     DBMS_OUTPUT.PUT_LINE('Title: ' || v_title);
12     DBMS_OUTPUT.PUT_LINE('Author: ' || v_author);
13     DBMS_OUTPUT.PUT_LINE('Year Published: ' || v_year_published);
14 END;
```

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

```
Title: To Kill a Mockingbird - Retrieved
Author: Harper Lee
Year Published: 1960

Statement processed.

0.01 seconds
```

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

## RESULT:

# TRIGGER

EX\_NO: 18

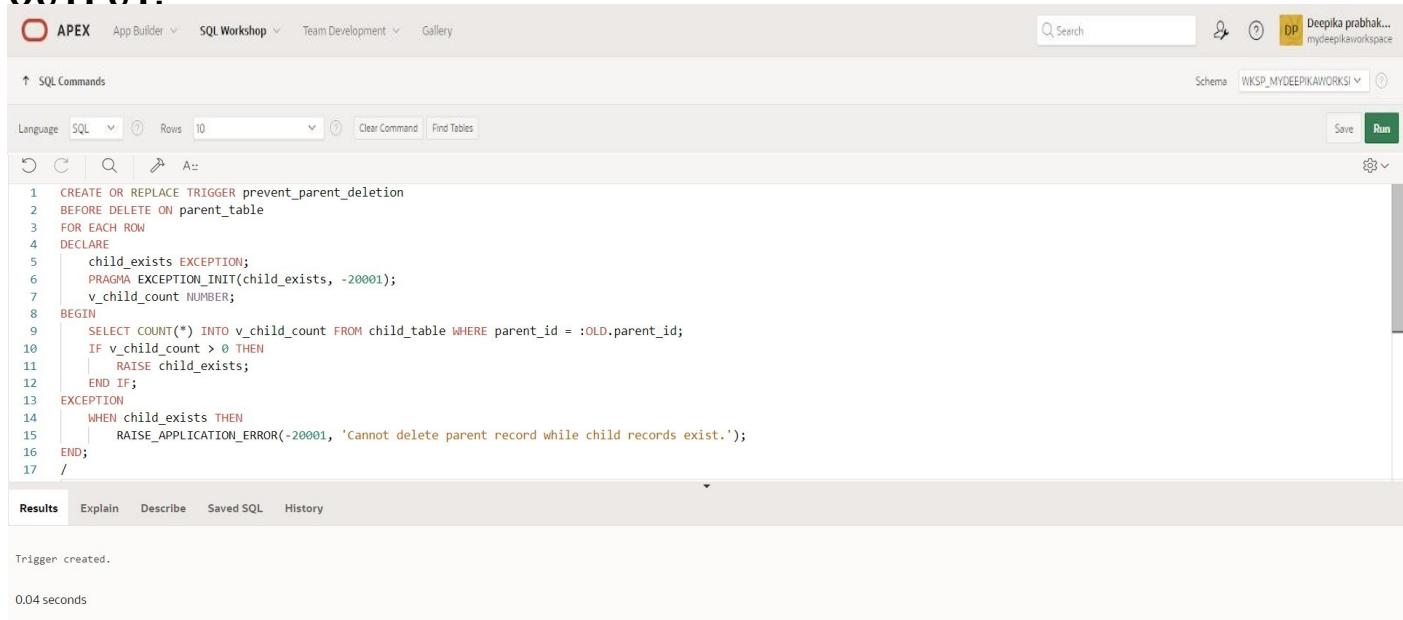
DATE:

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

QUERY:

```
CREATE OR REPLACE TRIGGER prevent_parent_deletion
BEFORE DELETE ON parent_table
FOR EACH ROW
DECLARE
    child_exists EXCEPTION;
    PRAGMA EXCEPTION_INIT(child_exists, -20001);
    v_child_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_child_count FROM child_table WHERE parent_id =
:OLD.parent_id;
    IF v_child_count > 0 THEN
        RAISE child_exists;
    END IF;
EXCEPTION
    WHEN child_exists THEN
        RAISE_APPLICATION_ERROR(-20001, 'Cannot delete parent record while child
records exist.');
END;
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, a user icon, and the workspace name "Deepika probhak... mydeepikaworkspace". The main area is titled "SQL Commands" and contains the PL/SQL code for the trigger. Below the code, the "Results" tab is selected, showing the output: "Trigger created." and "0.04 seconds".

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

Trigger created.  
0.04 seconds



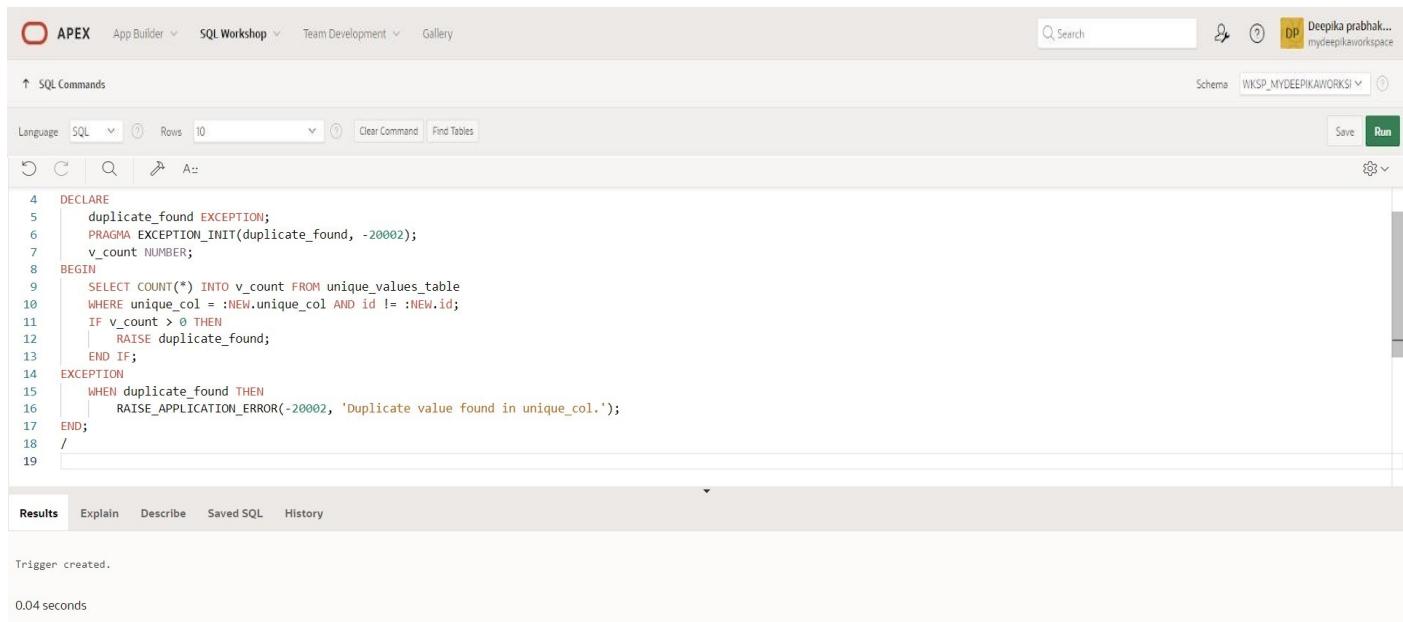
## 2.)

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

### QUERY:

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

### OUTPUT:



The screenshot shows the Oracle SQL Developer interface with the SQL Workshop tab selected. A trigger named 'check\_duplicates' is being created in the 'unique\_values\_table'. The code is displayed in the central pane, and the bottom pane shows the results of the command, indicating that the trigger was successfully created.

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

Results

Trigger created.

0.04 seconds

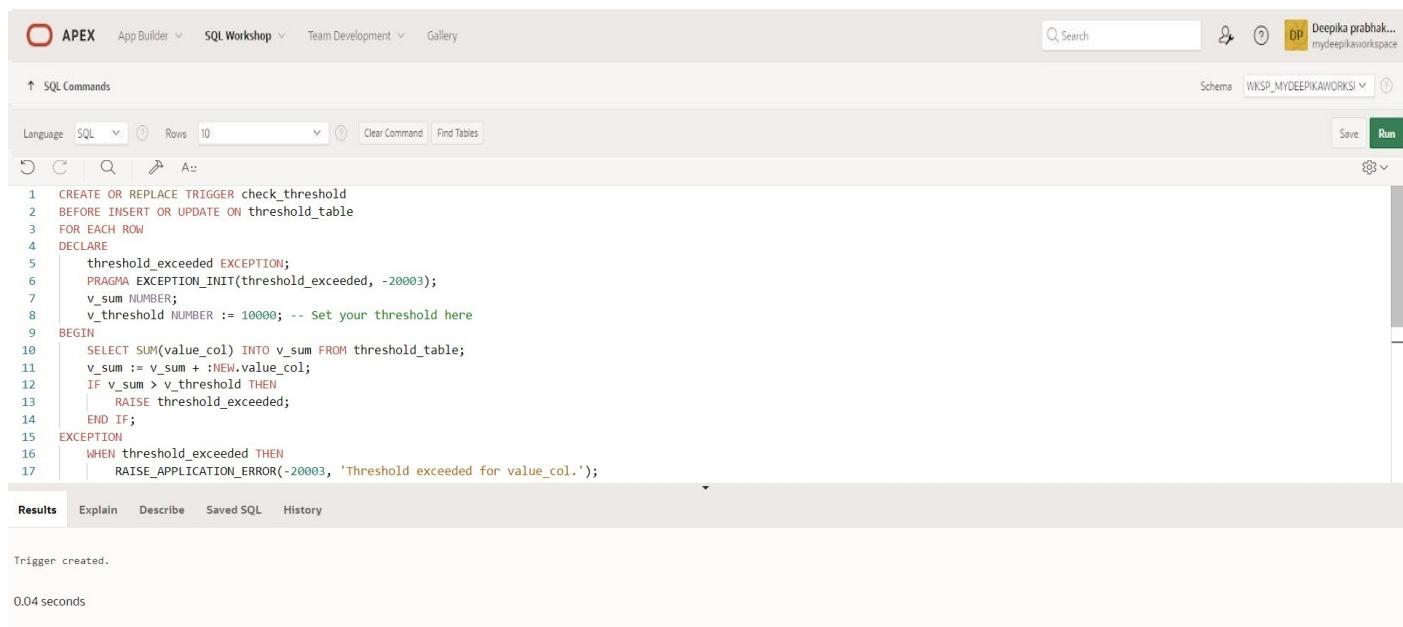
3.)

**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. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there are tabs for Schema (WKSP\_MYDEEPIKAWORKS1), a user icon (DP Deepika prabhak...), and a search bar. The main workspace is titled "SQL Commands". It contains a code editor with the trigger definition. Below the code editor are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, displaying the message "Trigger created." and "0.04 seconds".

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

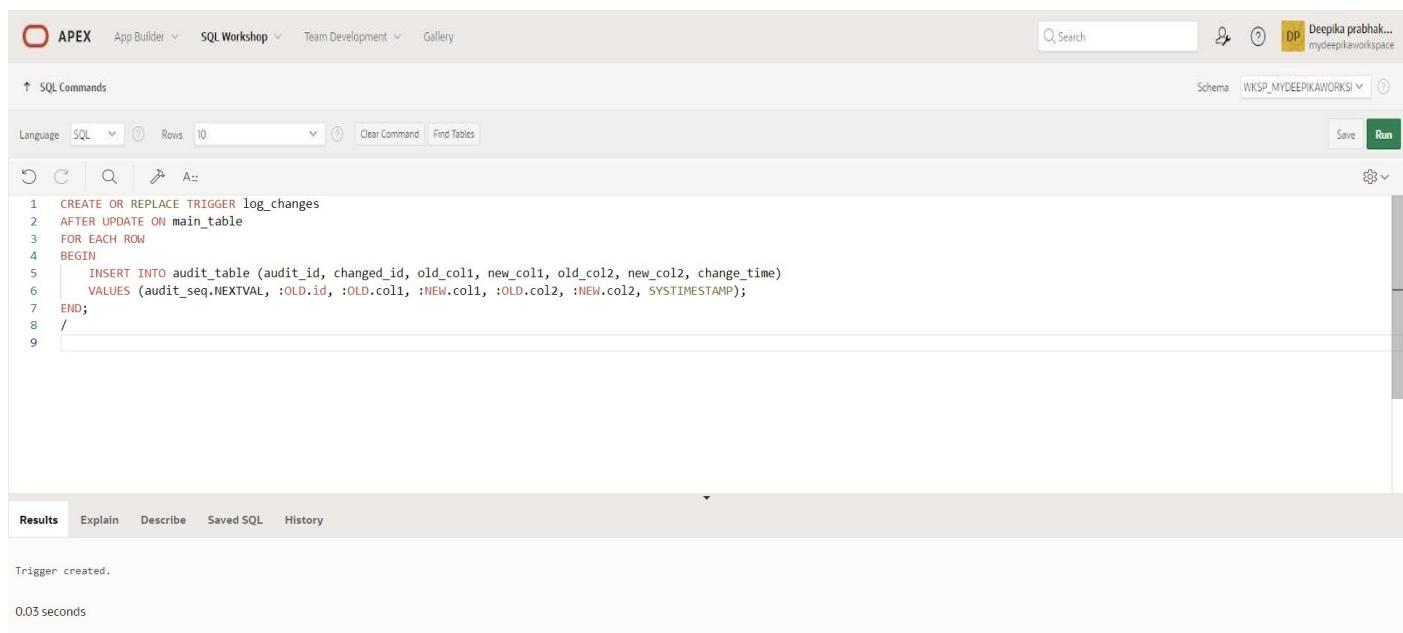
#### 4.)

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

#### QUERY:

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

#### OUTPUT:



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

```
1 CREATE OR REPLACE TRIGGER log_changes
2 AFTER UPDATE ON main_table
3 FOR EACH ROW
4 BEGIN
5     INSERT INTO audit_table (audit_id, changed_id, old_col1, new_col1, old_col2,
6     new_col2, change_time)
7     VALUES (audit_seq.NEXTVAL, :OLD.id, :OLD.col1, :NEW.col1, :OLD.col2, :NEW.col2,
8     SYSTIMESTAMP);
9 END;
9 /
```

Below the code, the 'Results' tab is active, showing the message "Trigger created." and a execution time of "0.03 seconds".

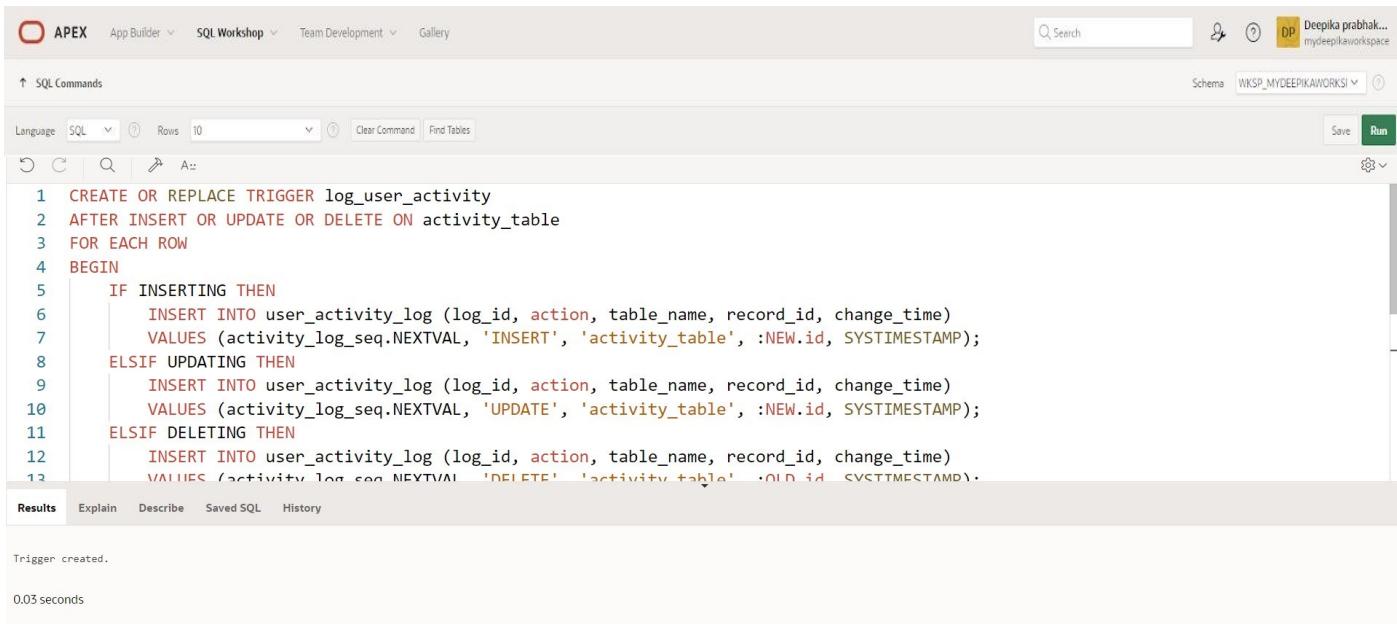
5.)

**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. In the top navigation bar, 'APEX' is selected. The main area displays the SQL command for creating the trigger:

```
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);
```

Below the code, the 'Results' tab is active, showing the message "Trigger created." and "0.03 seconds".

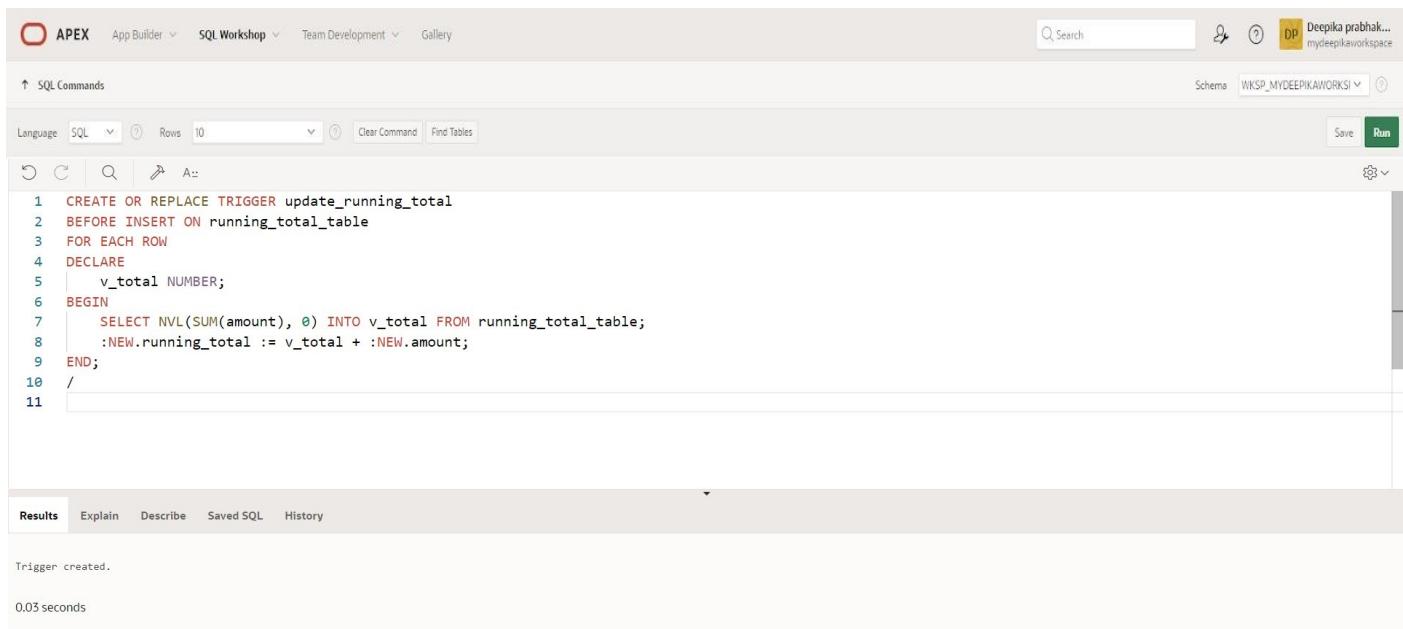
## 6.)

**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. In the top navigation bar, 'APEX' is selected. The main area displays the SQL command for creating the trigger, which is then executed successfully, resulting in the message 'Trigger created.' in the results pane.

```
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 
```

Results

Trigger created.

0.03 seconds

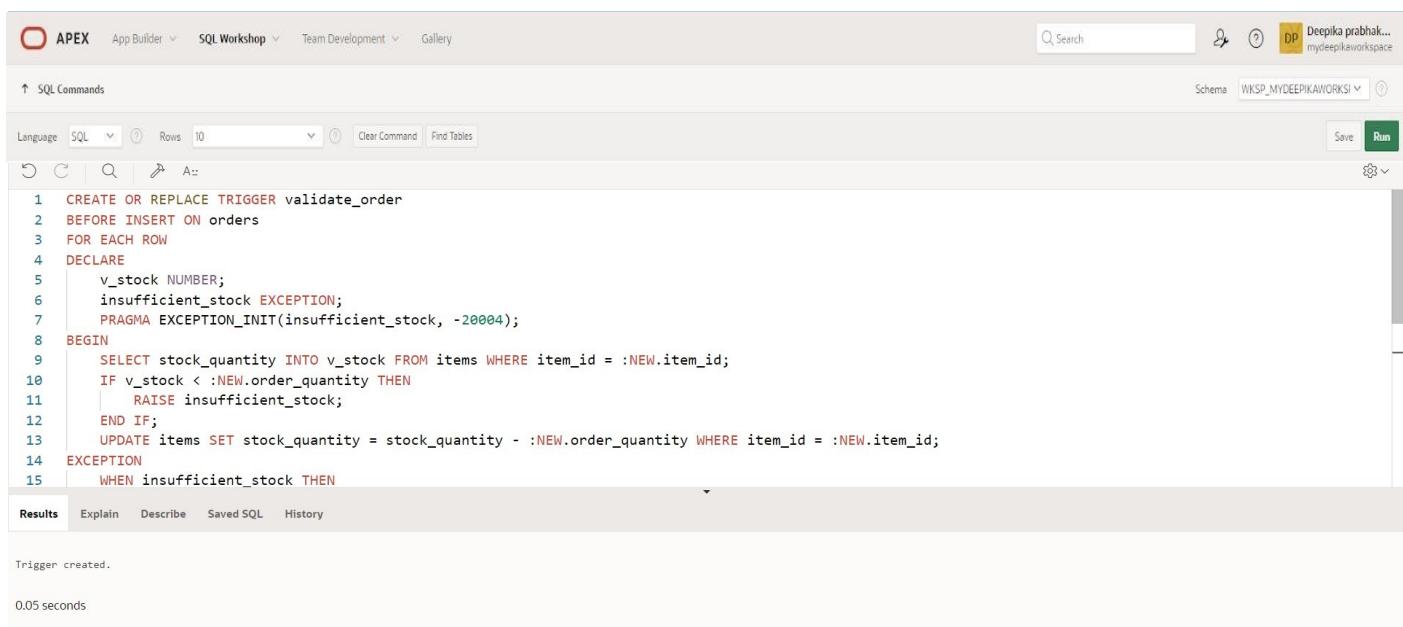
7.)

**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. On the right, there's a search bar, a help icon, and a workspace dropdown for 'Deepika prabhak... mydeepikaworkspace'. The main area is titled 'SQL Commands' with tabs for Language (set to SQL), Rows (set to 10), Clear Command, and Find Tables. Below these are standard edit tools: Undo, Redo, Cut, Copy, Paste, Find, Replace, and Auto. The SQL editor contains the PL/SQL code for the 'validate\_order' trigger. The code is numbered from 1 to 15. Lines 1 through 14 are part of the trigger definition, while line 15 is the exception handling section. The code uses standard Oracle syntax, including the 'RAISE\_APPLICATION\_ERROR' function. At the bottom of the SQL editor, there are 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History' tabs. The results pane displays the message 'Trigger created.' and '0.05 seconds'.

```
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
```

**RESULT:**

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

# 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:**

```
Deepika_58> db.restaurants.find( { $or: [ { name: /^Wil/ }, { cuisine: { $nin: ['American', 'Chinese'] } } ] }, { restaurant_id: 1, name: 1, borough: 1, cuisine: 1 } );
[ {
  _id: ObjectId('664f3c798752f54dc3cdcf7'),
  borough: 'Bronx',
  cuisine: 'Bakery',
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445'
}
]
Deepika_58>
```

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:**

```
Deepika_58> db.restaurants.find( { grades: { $elemMatch: { grade: "A", score: 11, date: ISODate("2014-08-11T00:00:00Z") } } }, { restaurant_id: 1, name: 1, grades: 1 } );
[ ]
Deepika_58>
```

**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:**

```
Deepika_58> 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 } );
Deepika_58> |
```

**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:**

```
Deepika_58> db.restaurants.find({$and : [{"address.coord.1": {$gt : 42}}, {"address.coord.1": {$lte : 52}}]}, {_id:0, restaurant_id:1, name:1, address:1})
Deepika_58> |
```

**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:**

```
Deepika_58> 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'
}
]
Deepika_58>
```

**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:**

```
Deepika_58> 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'
}
]
Deepika_58>
```

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

**QUERY:**

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

**OUTPUT:**

```
Deepika_58> 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'
  }
]
Deepika_58> |
```

**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:**

```
Deepika_58> db.restaurants.find({ "address.street": { $exists: true, $ne: "" } })
[
  {
    _id: ObjectId('664f3c798752f54dc3cdcdf7'),
    address: {
      building: '1007',
      coord: [ -73.856077, 40.848447 ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: ISODate('2014-03-03T00:00:00.000Z'),
        grade: 'A',
        score: 2
      },
      {
        date: ISODate('2013-09-11T00:00:00.000Z'),
        grade: 'A',
        score: 6
      },
      {
        date: ISODate('2013-01-24T00:00:00.000Z'),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate('2011-11-23T00:00:00.000Z'),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate('2011-03-10T00:00:00.000Z'),
        grade: 'B',
        score: 14
      }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
Deepika_58> |
```

**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:**

```
Deepika_58> db.restaurants.find({ "address.coord": { $elemMatch: { $type: "double" } } })
[ {
    _id: ObjectId('664f3c798752f54dc3cdcdf7'),
    address: {
        building: '1007',
        coord: [ -73.856077, 40.848447 ],
        street: 'Morris Park Ave',
        zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
        {
            date: ISODate('2014-03-03T00:00:00.000Z'),
            grade: 'A',
            score: 2
        },
        {
            date: ISODate('2013-09-11T00:00:00.000Z'),
            grade: 'A',
            score: 6
        },
        {
            date: ISODate('2013-01-24T00:00:00.000Z'),
            grade: 'A',
            score: 10
        },
        {
            date: ISODate('2011-11-23T00:00:00.000Z'),
            grade: 'A',
            score: 9
        },
        {
            date: ISODate('2011-03-10T00:00:00.000Z'),
            grade: 'B',
            score: 14
        }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
}
]
Deepika_58>
```

**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:**

```
Deepika_58> db.restaurants.find({ "grades.score": { $mod: [7, 0] } }, { restaurant_id: 1, name: 1, grades: 1 })
[ {
    _id: ObjectId('664f3c798752f54dc3cdcdf7'),
    grades: [
        {
            date: ISODate('2014-03-03T00:00:00.000Z'),
            grade: 'A',
            score: 2
        },
        {
            date: ISODate('2013-09-11T00:00:00.000Z'),
            grade: 'A',
            score: 6
        },
        {
            date: ISODate('2013-01-24T00:00:00.000Z'),
            grade: 'A',
            score: 10
        },
        {
            date: ISODate('2011-11-23T00:00:00.000Z'),
            grade: 'A',
            score: 9
        },
        {
            date: ISODate('2011-03-10T00:00:00.000Z'),
            grade: 'B',
            score: 14
        }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
}
]
Deepika_58> |
```

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:**

```
Deepika_58> db.restaurants.find({ name: /mon/i }, { name: 1, borough: 1, "address.coord": 1, cuisine: 1 })
Deepika_58>
```

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:**

```
Deepika_58> db.restaurants.find({ name: /^Mad/i }, { name: 1, borough: 1, "address.coord": 1, cuisine: 1 })
Deepika_58>
```

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:**

```
Deepika_58> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } } })
[ {
    _id: ObjectId('664f3c798752f54dc3cdcdf7'),
    address: {
        building: '1007',
        coord: [-73.856077, 40.848447],
        street: 'Morris Park Ave',
        zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
        {
            date: ISODate('2014-03-03T00:00:00.000Z'),
            grade: 'A',
            score: 2
        },
        {
            date: ISODate('2013-09-11T00:00:00.000Z'),
            grade: 'A',
            score: 6
        },
        {
            date: ISODate('2013-01-24T00:00:00.000Z'),
            grade: 'A',
            score: 10
        },
        {
            date: ISODate('2011-11-23T00:00:00.000Z'),
            grade: 'A',
            score: 9
        },
        {
            date: ISODate('2011-03-10T00:00:00.000Z'),
            grade: 'B',
            score: 14
        }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
}
]
Deepika_58>
```

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:**

```
Deepika_58> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, "borough": "Manhattan" })
Deepika_58>
```

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:**

```
Deepika_58> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }] })
```

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

**QUERY:**

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

**OUTPUT:**

```
Deepika_58> 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:

```
Deepika_58> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $nin: ["American", "Chinese"] } })
Deepika_58>
```

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:

```
Deepika_58> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }] })
{
  "_id": ObjectId('664f3c798752f54dc3cdcf7'),
  "address": {
    "building": '1007',
    "coord": [-73.856077, 40.848447],
    "street": 'Morris Park Ave',
    "zipcode": '10462'
  },
  "borough": 'Bronx',
  "cuisine": 'Bakery',
  "grades": [
    {
      "date": ISODate('2014-03-03T00:00:00.000Z'),
      "grade": 'A',
      "score": 2
    },
    {
      "date": ISODate('2013-09-11T00:00:00.000Z'),
      "grade": 'A',
      "score": 6
    },
    {
      "date": ISODate('2013-01-24T00:00:00.000Z'),
      "grade": 'A',
      "score": 10
    },
    {
      "date": ISODate('2011-11-23T00:00:00.000Z'),
      "grade": 'A',
      "score": 9
    },
    {
      "date": ISODate('2011-03-10T00:00:00.000Z'),
      "grade": 'B',
      "score": 14
    }
  ],
  "name": 'Morris Park Bake Shop',
  "restaurant_id": '50075445'
}
BHARATH_KUMAR_43> |
```

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:**

```
Deepika_58> 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:**

```
Deepika_58> 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:**

```
Deepika_58> 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:**

```
Deepika_58> 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:

```
Deepika_58> db.restaurants.find({ $or: [{ "grades.score": 2 }, { "grades.score": 6 }] })
[{"_id": ObjectId('664f3c798752f54dc3cdcf7'), "address": {"building": "1007", "coord": [-73.856077, 40.848447], "street": "Morris Park Ave", "zipcode": "10462"}, "borough": "Bronx", "cuisine": "Bakery", "grades": [{"date": ISODate('2014-03-03T00:00:00.000Z'), "grade": "A", "score": 2}, {"date": ISODate('2013-09-11T00:00:00.000Z'), "grade": "A", "score": 6}, {"date": ISODate('2013-01-24T00:00:00.000Z'), "grade": "A", "score": 10}, {"date": ISODate('2011-11-23T00:00:00.000Z'), "grade": "A", "score": 9}, {"date": ISODate('2011-03-10T00:00:00.000Z'), "grade": "B", "score": 14}], "name": "Morris Park Bake Shop", "restaurant_id": "30075445"}]
Deepika_58> |
```

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

#### RESULT:

# MONGO DB

**EX\_NO:** 20

**DATE:**

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

**QUERY:**

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

**OUTPUT:**

```
Deepika_58> db.movies.find({ year: 1893 })
Deepika_58> |
```



**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:**

```
Deepika_58> db.movies.find({ runtime: { $gt: 120 } })
BHARATH_KUMAR_43> |
Deepika_58>
```



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

**QUERY:**

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

**OUTPUT:**

```
Deepika_58> 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/MV5BMTU3NjE5NzYtYTByNS00MDVmLWIwYjgtMmYwYWIxZDYyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._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')
  }
]
Deepika_58> |
```

### 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:**

```
Deepika_58> db.movies.find({ directors: 'William K.L. Dickson' })
DIADETHU HUMAN: ~ ~ ~
Deepika_58>
```

**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:**

```
Deepika_58> 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/MV5BMTU3NjE5NzYtYTYYNS00MDVmLWIwYjgtMmYwYWIxZDYyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._V1_SY1000_SX677_AL_.jpg',
  title: 'The Great Train Robbery',
  fullplot: "Among the earliest existing films in American cinema - notable as the first film that presented a narrative story to tell - it depicts a group of cowboy outlaws who hold up a train and rob the passengers. They are then pursued by a Sheriff's posse. Several scenes have color included - all hand tinted.",
  languages: [ 'English' ],
  released: ISODate('1903-12-01T00:00:00.000Z'),
  directors: [ 'Edwin S. Porter' ],
  rated: 'TV-G',
  awards: { wins: 1, nominations: 0, text: '1 win.' },
  lastupdated: '2015-08-13 00:27:59.177000000',
  year: 1903,
  imdb: { rating: 7.4, votes: 9847, id: 439 },
  countries: [ 'USA' ],
  type: 'movie',
  tomatoes: {
    viewer: { rating: 3.7, numReviews: 2559, meter: 75 },
    fresh: 6,
    critic: { rating: 7.6, numReviews: 6, meter: 100 },
    rotten: 0,
    lastUpdated: ISODate('2015-08-08T19:16:10.000Z')
  }
}
Deepika_58> |
```

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

**QUERY:**

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

**OUTPUT:**

```
Deepika_58> db.movies.find({ rated: 'UNRATED' })
Deepika_58> |
```

**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:**

```
Deepika_58> 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/MV5BMTU3NjE5NzYtYTYYNS00MDVmLWIwYjgtMmYwYWIxZDyNzU2XkEyXkFqcGdeQXVyNzQzQxNzI@._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')
    }
  }
]
Deepika_58> |
```

**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:**

```
Deepika_58> 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/MV5BMTU3NjE5NzYtYTYYNS00MDVmLWIwYjgtMmYwYWIxZDyNzU2XkEyXkFqcGdeQXVyNzQzQxNzI@._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')
    }
  }
]
Deepika_58> |
```

**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:**

```
Deepika_58> db.movies.find({ 'tomatoes.viewer.rating': { $gt: 4 } })
Deepika_58>
```

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

**QUERY:**

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

**OUTPUT:**

```
Deepika_58> 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. Adabie',
    'Gilbert M. "Broncho Billy" Anderson',
    'George Barnes',
    'Justus D. Barnes'
  ],
  poster: 'https://m.media-amazon.com/images/M/MV5BMTU3NjE5NzYtYTYYNS00MDVmLWIwYjgtMmYwYWIxZDYYNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._V1_SY1000_SX677_AL_.jpg',
  title: 'The Great Train Robbery',
  fullplot: 'Among the earliest existing films in American cinema - notable as the first film that presented a narrative story to tell - it depicts a group of cowboy outlaws who hold up a train and rob the passengers. They are then pursued by a Sheriff's posse. Several scenes have color included - all hand tinted.',
  languages: [ 'English' ],
  released: ISODate('1903-12-01T00:00:00.000Z'),
  directors: [ 'Edwin S. Porter' ],
  rated: 'TV-G',
  awards: { wins: 1, nominations: 0, text: '1 win.' },
  lastupdated: '2015-08-13 00:27:59.177000000',
  year: 1903,
  imdb: { rating: 7.4, votes: 9847, id: 439 },
  countries: [ 'USA' ],
  type: 'movie',
  tomatoes: {
    viewer: { rating: 3.7, numReviews: 2559, meter: 75 },
    fresh: 6,
    critic: { rating: 7.6, numReviews: 6, meter: 100 },
    rotten: 0,
    lastUpdated: ISODate('2015-08-08T19:16:10.000Z')
  }
}
Deepika_58>
```

**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:**

```
Deepika_58> 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 }
... )
BHARATH_KUMAR_43> |
Deepika_58>
```

**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:**

```
Deepika_58> 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 }
... )
BHARATH_KUMAR_43> |
Deepika_58>
```

**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:**

```
Deepika_58> db.movies.find(  
...   { released: ISODate("1893-05-09T00:00:00.000Z") },  
...   { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 }  
... )  
Deepika_58> |
```

**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:**

```
Deepika_58> db.movies.find(
...   { title: /scene/i },
...   { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 }
... )
```

**RESULT:**

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