

# 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     |
|---------------------|--------|----------|
| <b>Key Type</b>     |        |          |
| <b>Nulls/Unique</b> |        |          |
| <b>FK table</b>     |        |          |
| <b>FK column</b>    |        |          |
| <b>Data Type</b>    | Number | Varchar2 |
| <b>Length</b>       | 7      | 25       |

## QUERY:

```
CREATE TABLE DEPT (DEPT_ID number(7) not null, DEPT_NAME varchar(25));
```

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'SQL Commands' and a schema dropdown set to 'WKSP\_SHRAVANTHI902CSI'. The toolbar has buttons for Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run. Below the toolbar, there are icons for Undo, Redo, Find, Replace, and Paste. The main area contains a single SQL command: 'CREATE TABLE DEPT(DEPT\_ID number(7) not null, DEPT\_NAME varchar(25));'. The results tab is selected, showing the output 'Table created.' and a execution time of '0.02 seconds'. The bottom footer includes user information (220701902@rajalakshmi.edu.in, shravanti902cse), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the version 'Oracle APEX 23.2.4'.

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

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

**QUERY:**

```
CREATE TABLE EMP (EMP_ID number(7) not null, FIRST_NAME  
    varchar(25),LAST_NAME varchar(25),DEPT_ID number(7));
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'SQL Commands' and a schema dropdown set to 'WKSP\_SHRAVANTHI902CSI'. Below the toolbar, there are buttons for Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run. The main area contains the SQL command for creating the EMP table. The results section below shows the message 'Table created.' and a execution time of '0.03 seconds'. The bottom footer includes user information (220701902@rajalakshmi.edu.in, shrawanthi902cse), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the version 'Oracle APEX 23.2.4'.

```
1 CREATE TABLE EMP(EMP_ID number(7) not null, FIRST_NAME varchar(25),LAST_NAME varchar(25),DEPT_ID number(7));
```

Results Explain Describe Saved SQL History

Table created.  
0.03 seconds

220701902@rajalakshmi.edu.in shrawanthi902cse Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

**QUERY:**

```
ALTER TABLE EMP MODIFY(LAST_NAME varchar(50));
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The query entered is: `ALTER TABLE EMP MODIFY (LAST_NAME varchar(50));`. The results section shows the output: `Table altered.`. The bottom status bar indicates the user is 220701902@rajalakshmi.edu.in and the session ID is shravanthi902cse. The copyright notice is Copyright © 1999, 2023, Oracle and/or its affiliates. The version is Oracle APEX 23.2.4.

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

**QUERY:**

```
CREATE TABLE EMPLOYEE2(EMPLOYEEID number(6)not null, FIRSTNAME  
varchar(25), LASTNAME varchar(25),SALARY number(10), DEPTID number(6)  
not null);
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The queries entered are: `CREATE TABLE EMP2(EMP_ID number(6),FIRST_NAME varchar(24),LAST_NAME varchar(20),  
SALARY number(8),DEPT_ID number(6) not null);`. The results section shows the output: `Table created.`. The bottom status bar indicates the user is 220701902@rajalakshmi.edu.in and the session ID is shravanthi902cse. The copyright notice is Copyright © 1999, 2023, Oracle and/or its affiliates. The version is Oracle APEX 23.2.4.

5. Drop the EMP table.

**QUERY:**

```
DROP TABLE EMP;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The command entered is 'DROP TABLE EMP;'. The results tab shows the output: 'Table dropped.' and '0.09 seconds'. The bottom status bar includes session information and the text 'Oracle APEX 23.2.4'.

↑ SQL Commands

Schema: WKSP\_SHRAVANTHI902CSI

Language: SQL Rows: 10

Clear Command Find Tables Save Run

1 **DROP TABLE** EMP;

Results Explain Describe Saved SQL History

Table dropped.  
0.09 seconds

220701902@rajalakshmi.edu.in shrawanthi902cse Copyright © 1999, 2023, Oracle and/or its affiliates. https://apex.oracle.com/pls/apex/r/apex/app-builder/apps?session=3198729214433 Oracle APEX 23.2.4

6. Rename the EMPLOYEES2 table as EMP.

**QUERY:**

```
RENAME EMP2 TO EMP;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The command entered is 'RENAME EMP2 TO EMP;'. The results tab shows the output: 'Statement processed.' and '0.06 seconds'. The bottom status bar includes session information and the text 'Oracle APEX 23.2.4'.

↑ SQL Commands

Schema: WKSP\_SHRAVANTHI902CSI

Language: SQL Rows: 10

Clear Command Find Tables Save Run

1 **RENAME** EMP2 TO EMP;

Results Explain Describe Saved SQL History

Statement processed.  
0.06 seconds

220701902@rajalakshmi.edu.in shrawanthi902cse Copyright © 1999, 2023, Oracle and/or its affiliates. https://apex.oracle.com/pls/apex/r/apex/app-builder/apps?session=3198729214433 Oracle APEX 23.2.4

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

**QUERY:**

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

**OUTPUT:**

↑ SQL Commands Schema WKSP\_SHRAVANTHI902CSI ⓘ

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

DESCRIBE DEPT;

Results Explain Describe Saved SQL History

Object Type TABLE ⓘ Object DEPT ⓘ

| Table | Column    | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|-------|-----------|-----------|--------|-----------|-------|-------------|----------|---------|---------|
| DEPT  | DEPT_ID   | NUMBER    | -      | 7         | 0     | -           | -        | -       | -       |
|       | DEPT_NAME | VARCHAR2  | 25     | -         | -     | -           | ✓        | -       | -       |

✉ 220701902@rajalakshmi.edu.in 📄 shravanti902cse Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

↑ SQL Commands Schema WKSP\_SHRAVANTHI902CSI ⓘ

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

DESCRIBE EMP;

Results Explain Describe Saved SQL History

Object Type TABLE ⓘ Object EMP ⓘ

| Table | Column     | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|-------|------------|-----------|--------|-----------|-------|-------------|----------|---------|---------|
| EMP   | EMP_ID     | NUMBER    | -      | 6         | 0     | -           | ✓        | -       | -       |
|       | FIRST_NAME | VARCHAR2  | 24     | -         | -     | -           | ✓        | -       | -       |
|       | LAST_NAME  | VARCHAR2  | 20     | -         | -     | -           | ✓        | -       | -       |
|       | SALARY     | NUMBER    | -      | 8         | 0     | -           | ✓        | -       | -       |
|       | DEPT_ID    | NUMBER    | -      | 6         | 0     | -           | -        | -       | -       |

✉ 220701902@rajalakshmi.edu.in 📄 shravanti902cse ⓘ en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

**QUERY:**

```
ALTER TABLE EMP DROP COLUMN FIRST_NAME;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The top navigation bar includes 'SQL Commands', 'Schema' set to 'WKSP\_SHRAVANTHI902CSI', and a 'Run' button. Below the bar are buttons for Language (SQL), Rows (10), Clear Command, and Find Tables. The main area contains the SQL command: 'ALTER TABLE EMP DROP COLUMN FIRST\_NAME;'. The results tab is selected, showing the output: 'Table altered.' and '0.07 seconds'. The bottom footer displays user information (220701902@rajalakshmi.edu.in, shravanthi902cse), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and version information (Oracle APEX 23.2.4).

| 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 MYEMPLOYEE(ID number(4) not null ,LAST_NAME  
varchar(25),FIRST_NAME varchar(25),USER_ID varchar(25),SALARY  
number(9,2));
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'SQL Commands', 'Schema' set to 'WKSP\_SHRAVANTHI902CSE', and a 'Run' button. Below the toolbar are buttons for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. The main area contains the SQL code for creating the table:

```
1 CREATE TABLE MYEMPLOYEE(ID number(4) not null ,LAST_NAME varchar(25),  
2 FIRST_NAME varchar(25),USER_ID varchar(25),SALARY number(9,2));
```

Below the code, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, displaying the message 'Table created.' and a execution time of '0.04 seconds'. At the bottom, the footer includes user information (email: 220701902@rajalakshmi.edu.in, workspace: shravanthi902cse), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the version 'Oracle APEX 23.2.4'.

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    | 860    |
| 4  | Newman    | Chad       | Cnewman  | 750    |
| 5  | Ropebur   | Audrey     | aropebur | 1550   |

### QUERY:

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

### OUTPUT:

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

```
1 INSERT INTO MYEMPLOYEE VALUES(2,'Dancs','Betty','bdancs',860);
```

The results section shows:

1 row(s) inserted.  
0.01 seconds

At the bottom, the footer includes:

220701902@rajalakshmi.edu.in shrawanthi902cse Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

```
1 INSERT INTO MYEMPLOYEE VALUES(1,'Patel','Ralph','rpatel',895);
```

The results section shows:

1 row(s) inserted.  
0.03 seconds

At the bottom, the footer includes:

220701902@rajalakshmi.edu.in shrawanthi902cse Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

3. Display the table with values.

**QUERY:**

```
SELECT * FROM MYEMPLOYEE;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Worksheet interface. The query `SELECT * FROM MYEMPLOYEE;` has been run, returning two rows of data:

| ID | LAST_NAME | FIRST_NAME | USER_ID | SALARY |
|----|-----------|------------|---------|--------|
| 2  | Dancs     | Betty      | bdancs  | 860    |
| 1  | Patel     | Ralph      | rpatel  | 895    |

Below the table, it says "2 rows returned in 0.01 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 MYEMPLOYEE VALUES (4,'Newman','Chad','cnewman',750);
INSERT INTO MYEMPLOYEE VALUES (5,'Ropebur','Audrey','aropebur',1550);
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Worksheet interface after running the insert statements. The table now contains five rows of data:

| ID | LAST_NAME | FIRST_NAME | USER_ID  | SALARY |
|----|-----------|------------|----------|--------|
| 2  | Dancs     | Betty      | bdancs   | 860    |
| 3  | Biri      | Ben        | bbiri    | 860    |
| 4  | Chad      | Newman     | cnewman  | 750    |
| 5  | Ropebur   | Audrey     | aropebur | 1550   |
| 1  | Patel     | Ralph      | rpatel   | 895    |

Below the table, it says "5 rows returned in 0.00 seconds".

5. Make the data additions permanent.

**QUERY:**

```
SELECT * FROM MYEMPLOYEE;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Worksheet interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The query entered is 'SELECT \* FROM MYEMPLOYEE;'. The results section displays a table with the following data:

| ID | LAST_NAME | FIRST_NAME | USER_ID   | SALARY |
|----|-----------|------------|-----------|--------|
| 2  | Dancs     | Betty      | bdancs    | 860    |
| 3  | Biri      | Ben        | bbiri     | 860    |
| 4  | Chad      | Newman     | cnewman   | 750    |
| 4  | Ropebur   | Audrey     | arophebur | 1550   |
| 1  | Patel     | Ralph      | rpatel    | 895    |

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

User information at the bottom: 220701902@rajalakshmi.edu.in, shravanthi902cse, en. Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

**QUERY:**

```
UPDATE MYEMPLOYEE SET LAST_NAME = 'DREXLER' WHERE ID=3;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Worksheet interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The query entered is 'UPDATE MYEMPLOYEE SET LAST\_NAME = 'DREXLER' WHERE ID=3;'. The results section shows the message '1 row(s) updated.' and '0.01 seconds'.

User information at the bottom: 220701902@rajalakshmi.edu.in, shravanthi902cse, en. Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

**QUERY:**

```
UPDATE MYEMPLOYEE SET SALARY=1000 WHERE SALARY<900;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL workspace interface. The top navigation bar includes 'SQL Commands', 'Schema (WKSP\_SHRAVANTHI902CSF)', and a help icon. Below the schema dropdown are buttons for 'Save' and 'Run'. The main area has tabs for 'Language' (set to SQL), 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. The SQL command entered is:

```
1 UPDATE MYEMPLOYEE
2 SET SALARY=1000
3 WHERE SALARY<900;
```

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

4 row(s) updated.  
0.01 seconds

At the bottom, it displays user information (220701902@rajalakshmi.edu.in, shravanthi902cse), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the version (Oracle APEX 23.2.4).

8.Delete Betty from MY\_EMPLOYEE table.

**QUERY:**

```
DELETE FROM MYEMPLOYEE WHERE FIRST_NAME='Betty';
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL workspace interface. The top navigation bar includes 'SQL Commands', 'Schema (WKSP\_SHRAVANTHI902CSF)', and a help icon. Below the schema dropdown are buttons for 'Save' and 'Run'. The main area has tabs for 'Language' (set to SQL), 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. The SQL command entered is:

```
1 DELETE FROM MYEMPLOYEE WHERE FIRST_NAME='Betty';
```

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

1 row(s) deleted.  
0.01 seconds

At the bottom, it displays user information (220701902@rajalakshmi.edu.in, shravanthi902cse), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the version (Oracle APEX 23.2.4). The URL https://apex.oracle.com/pls/apex/r/apex/workspace/home?session=3198729214433 is also visible.

9.Empty the fourth row of the emp table.

**QUERY:**

```
DELETE FROM EMP WHERE EMP_ID=10004;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL workspace interface. At the top, there's a toolbar with 'SQL Commands' and a schema dropdown set to 'WKSP\_SHRAVANTHI902CSI'. Below the toolbar, there are buttons for Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run. The main area contains the SQL command: '1 DELETE FROM EMP WHERE EMP\_ID=10004;'. The results tab is selected, showing the output: '1 row(s) deleted.' and '0.03 seconds'. The bottom of the screen displays user information (220701902@rajalakshmi.edu.in, shrawanthi902cse) and copyright information (Copyright © 1999, 2023, Oracle and/or its affiliates). The URL https://apex.oracle.com/pls/apex/r/apex/workspace/home?session=12944390302411 is also visible.

| 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(EMP_ID);
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Worksheet interface. The SQL Commands tab is active, displaying the following code:

```
1 ALTER TABLE EMP
2 ADD CONSTRAINT my_emp_id_pk PRIMARY KEY(EMP_ID);
3
```

The Run button is highlighted. Below the code, the results show:

Table altered.  
0.09 seconds

At the bottom, the user information is shown as 220701902@rajalakshmi.edu.in and shrawanthi902cse, with copyright notice for Oracle APEX 23.2.4.

2. Create a PRIMARY KEY constraint to the DEPT table using the ID column. The constraint should be named at creation. Name the constraint my\_dept\_id\_pk.

**QUERY:**

```
ALTER TABLE DEPT
ADD CONSTRAINT my_dept_id_pk PRIMARY KEY(DEPT_ID);
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Worksheet interface. The SQL Commands tab is active, displaying the following code:

```
1 ALTER TABLE DEPT
2 ADD CONSTRAINT my_dept_id_pk PRIMARY KEY(DEPT_ID);
3
```

The Run button is highlighted. Below the code, the results show:

Table altered.  
0.07 seconds

At the bottom, the user information is shown as 220701902@rajalakshmi.edu.in and shrawanthi902cse, with copyright notice for Oracle APEX 23.2.4.

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(EMP_ID);
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL workspace interface. At the top, there's a toolbar with 'SQL Commands' and a schema dropdown set to 'WKSP\_SHRAVANTHI902CSI'. Below the toolbar, the SQL editor contains the following code:

```
1 ALTER TABLE EMP
2 ADD CONSTRAINT my_emp_dept_id_fk FOREIGN KEY(DEPT_ID) REFERENCES EMP(EMP_ID);
```

Under the 'Results' tab, the output is displayed:

```
Table altered.  
0.07 seconds
```

At the bottom, it shows the user information (220701902@rajalakshmi.edu.in, shrawanthi902cse), 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) CONSTRAINT emp_comm_check
CHECK (commission > 0);
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL workspace interface. At the top, there's a toolbar with 'SQL Commands' and a schema dropdown set to 'WKSP\_SHRAVANTHI902CSI'. Below the toolbar, the SQL editor contains the following code:

```
1 ALTER TABLE emp
2 ADD COMMISSION NUMBER(2, 2) CONSTRAINT emp_comm_check CHECK (commission > 0);
3
```

Under the 'Results' tab, the output is displayed:

```
Table altered.  
0.05 seconds
```

At the bottom, it shows the user information (220701902@rajalakshmi.edu.in, shrawanthi902cse), 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 employeeid, last_name ,Salary*12 as Annual salary  
From employees;
```

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

## QUERY:

```
DESCRIBE DEPT;
```

## OUTPUT:

The screenshot shows the Oracle APEX interface with the following details:

- SQL Commands:** The input field contains the command `1 DESCRIBE DEPT;`.
- Results:** The tab is selected, showing the output of the `DESCRIBE DEPT` command.
- Object Type:** TABLE
- Object:** DEPT
- Table Data:**

| Table | Column    | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|-------|-----------|-----------|--------|-----------|-------|-------------|----------|---------|---------|
| DEPT  | DEPT_ID   | NUMBER    | -      | 7         | 0     | 1           | -        | -       | -       |
|       | DEPT_NAME | VARCHAR2  | 25     | -         | -     | -           | ✓        | -       | -       |
- Page Footer:** Includes user information (220701902@rajalakshmi.edu.in, shrawanthi902cse), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and version (Oracle APEX 23.2.4).

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

**QUERY:**

```
SELECT EMPLOYEE_ID, LAST_NAME, JOB_ID, HIRE_DATE FROM EMPLOYEE_DETAILS;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The query `SELECT EMPLOYEE_ID, LAST_NAME, JOB_ID, HIRE_DATE FROM EMPLOYEE_DETAILS;` is entered in the command field. The results are displayed in a table with columns: EMPLOYEE\_ID, LAST\_NAME, JOB\_ID, and HIRE\_DATE. The data is as follows:

| EMPLOYEE_ID | LAST_NAME | JOB_ID | HIRE_DATE  |
|-------------|-----------|--------|------------|
| 2222        | cooper    | 020220 | 02/02/2020 |
| 4444        | miller    | 020221 | 02/02/2021 |
| 5555        | day       | 050520 | 05/05/2020 |
| 1111        | LEE       | 122320 | 02/12/2020 |
| 3333        | wilson    | 021120 | 02/11/2020 |

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

4. Provide an alias STARTDATE for the hire date.

**QUERY:**

```
SELECT HIRE_DATE AS STARTDATE FROM EMPLOYEE_DETAILS;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The query `SELECT HIRE_DATE AS STARTDATE FROM EMPLOYEE_DETAILS;` is entered in the command field. The results are displayed in a table with a single column labeled STARTDATE. The data is as follows:

| STARTDATE  |
|------------|
| 02/02/2020 |
| 02/02/2021 |
| 05/05/2020 |
| 02/12/2020 |
| 02/11/2020 |

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

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

**QUERY:**

```
SELECT UNIQUE JOB_ID FROM EMPLOYEE_DETAILS;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL workspace interface. The top navigation bar includes 'SQL Commands', 'Schema' set to 'WKSP\_SHRAVANTHI902CSI', and a 'Run' button. The main area has tabs for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. Below these are icons for refresh, search, and copy. The SQL command entered is: `1 SELECT UNIQUE JOB_ID FROM EMPLOYEE_DETAILS;`. The results tab is selected, showing a single column 'JOB\_ID' with five rows of data: 122320, 021120, 020221, 020220, and 050520. A note at the bottom says '5 rows returned in 0.01 seconds'. The bottom of the screen displays user information (email: 220701902@rajalakshmi.edu.in, name: shravanti902cse, language: en) and copyright information (Copyright © 1999, 2023, Oracle and/or its affiliates). The version is Oracle APEX 23.2.4.

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 TITLE FROM EMPLOYEE_DETAILS;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL workspace interface. The top navigation bar includes 'SQL Commands', 'Schema' set to 'WKSP\_SHRAVANTHI902CSI', and a 'Run' button. The main area has tabs for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. Below these are icons for refresh, search, and copy. The SQL command entered is: `1 SELECT LAST_NAME || ',' || JOB_ID AS TITLE FROM EMPLOYEE_DETAILS;`. The results tab is selected, showing a single column 'TITLE' with five rows of data: cooper,020220, miller,020221, day,050520, LEE,122320, and wilson,021120. A note at the bottom says '5 rows returned in 0.01 seconds'. The bottom of the screen displays user information (email: 220701902@rajalakshmi.edu.in, name: shravanti902cse, language: en) and copyright information (Copyright © 1999, 2023, Oracle and/or its affiliates). The version is Oracle APEX 23.2.4.

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

**QUERY:**

```
SELECT EMP_ID || ',' || LAST_NAME || ',' || JOB_ID || ',' || EMAIL_ID || ',' ||  
SALARY || ',' || HIRE_DATE AS "THE_OUTPUT" FROM EMPLOYEE_DETAILS;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The query is executed successfully, returning five rows of data under the column 'THE\_OUTPUT'. The data consists of five employee records separated by commas.

| THE_OUTPUT   |
|--|
| 2222,cooper,020220,cooper@gmail.com,20000,02/02/2020   |
| 4444,miller,020221,millerday@gmail.com,9000,02/02/2021 |
| 5555,day,050520,daymiller@gmail.com,50000,05/05/2020   |
| 1111,LEE,122320,jonlee@gmail.com,12000,02/12/2020      |
| 3333,wilson,021120,awilson@gmail.com,9000,02/11/0020   |

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

**RESULT:**

# RESTRICTING AND SORTING DATA

**EX\_NO:5**

**DATE:**

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

**QUERY:**

```
SELECT LAST_NAME, SALARY FROM EMPLOYEE_DETAILS WHERE SALARY>12000;
```

**OUTPUT:**

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

```
1 SELECT LAST_NAME, SALARY
2 FROM EMPLOYEE_DETAILS
3 WHERE SALARY>12000
```

The results table displays two rows:

| LAST_NAME | SALARY |
|-----------|--------|
| cooper    | 20000  |
| day       | 50000  |

Below the table, it says "2 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, SALARY FROM EMPLOYEE_DETAILS WHERE EMPLOYEE_ID=176;
```

**OUTPUT:**

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

```
1 SELECT LAST_NAME, DEPT_ID
2 FROM EMPLOYEE_DETAILS
3 WHERE EMPLOYEE_ID=176;
```

The results table displays one row:

| LAST_NAME | DEPT_ID |
|-----------|---------|
| Fisher    | 6       |

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

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

#### QUERY:

```
SELECT LAST_NAME, SALARY FROM EMPLOYEE_DETAILS WHERE SALARY NOT BETWEEN 5000 AND 12000;
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Worksheet interface. The query `SELECT LAST\_NAME, SALARY FROM EMPLOYEE\_DETAILS WHERE SALARY NOT BETWEEN 5000 AND 12000;` has been run. The results are displayed in a table with columns `LAST\_NAME` and `SALARY`. The data shows three rows: Fisher (70000), cooper (20000), and day (50000). The schema is set to `WKSP\_SHRAVANTHI902CSI`.

| LAST_NAME | SALARY |
|-----------|--------|
| Fisher    | 70000  |
| cooper    | 20000  |
| day       | 50000  |

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

#### QUERY:

```
SELECT LAST_NAME, JOB_ID, HIRE_DATE FROM EMPLOYEE_DETAILS WHERE HIRE_DATE BETWEEN 'FEBRUARY,20,1998' AND 'MAY,1,1998';
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Worksheet interface. The query `SELECT LAST\_NAME, JOB\_ID, HIRE\_DATE FROM EMPLOYEE\_DETAILS WHERE HIRE\_DATE BETWEEN 'FEBRUARY,20,1998' AND 'MAY,1,1998';` has been run. The results are displayed in a table with columns `LAST\_NAME`, `JOB\_ID`, and `HIRE\_DATE`. The data shows one row: parker (accountant, 02/22/1998). The schema is set to `WKSP\_SHRAVANTHI902CSI`.

| LAST_NAME | JOB_ID     | HIRE_DATE  |
|-----------|------------|------------|
| parker    | accountant | 02/22/1998 |

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

**QUERY:**

```
SELECT LAST_NAME,DEPT_ID FROM EMPLOYEE_DETAILS WHERE DEPT_ID IN(20,50)  
ORDER BY LAST_NAME;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top bar includes 'SQL Commands', 'Schema (WKSP\_SHRAVANTHI902CSI)', and a 'Run' button. The command input area contains the query: 'SELECT LAST\_NAME,DEPT\_ID FROM EMPLOYEE\_DETAILS WHERE DEPT\_ID IN(20,50) ORDER BY LAST\_NAME;'. The results tab is selected, displaying the output:

| LAST_NAME | DEPT_ID |
|-----------|---------|
| parker    | 20      |

1 rows returned in 0.01 seconds

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

**QUERY:**

```
SELECT LAST_NAME AS "EMPLOYEE",SALARY AS "MONTHLY SALARY" FROM EMPLOYEE_DETAILS  
WHERE (SALARY BETWEEN 5000 AND 12000) AND (DEPT_ID IN(20,50)) ORDER BY LAST_NAME  
ASC;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top bar includes 'SQL Commands', 'Schema (WKSP\_SHRAVANTHI902CSI)', and a 'Run' button. The command input area contains the query: 'SELECT LAST\_NAME AS "EMPLOYEE",SALARY AS "MONTHLY SALARY" FROM EMPLOYEE\_DETAILS WHERE (SALARY BETWEEN 5000 AND 12000) AND (DEPT\_ID IN(20,50)) ORDER BY LAST\_NAME ASC;'. The results tab is selected, displaying the output:

| EMPLOYEE | MONTHLY SALARY |
|----------|----------------|
| parker   | 10000          |

1 rows returned in 0.01 seconds

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 EMPLOYEE_DETAILS WHERE HIRE_DATE LIKE '1994';
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The query entered is: `SELECT LAST_NAME,HIRE_DATE FROM EMPLOYEE_DETAILS WHERE HIRE_DATE LIKE '1994';`. The results section displays the message "no data found".

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

**QUERY:**

```
SELECT LAST_NAME,JOB_ID FROM EMPLOYEE_DETAILS WHERE MANAGER_ID IS NULL;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The query entered is: `SELECT LAST_NAME,JOB_ID FROM EMPLOYEE_DETAILS WHERE MANAGER_ID IS NULL;`. The results section displays the following data:

| LAST_NAME | JOB_ID |
|-----------|--------|
| day       | 050520 |
| LEE       | 122320 |
| wilson    | 021120 |

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

#### QUERY:

```
SELECT LAST_NAME, SALARY, COMMISSION_PCT FROM EMPLOYEE_DETAILS  
WHERE COMMISSION_PCT IS NOT NULL ORDER BY SALARY DESC;
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The query entered is:

```
1 SELECT LAST_NAME, SALARY, COMMISSION_PCT FROM EMPLOYEE_DETAILS  
2 WHERE COMMISSION_PCT IS NOT NULL ORDER BY SALARY DESC;
```

The results section displays the following data:

| LAST_NAME | SALARY | COMMISSION_PCT |
|-----------|--------|----------------|
| AMBROSE   | 89000  | 0              |
| JONES     | 70000  | 0              |

At the bottom, it shows the user information: 220701902@rajalakshmi.edu.in, shrawanthi902cse, Copyright © 1999, 2023, Oracle and/or its affiliates, and Oracle APEX 23.2.4.

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

#### QUERY:

```
SELECT LAST_NAME FROM EMPLOYEE_DETAILS WHERE LAST_NAME LIKE '__a%';
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The query entered is:

```
1 SELECT LAST_NAME FROM EMPLOYEE_DETAILS WHERE LAST_NAME LIKE '__a%';
```

The results section displays the message: 'no data found'.

At the bottom, it shows the user information: 220701902@rajalakshmi.edu.in, shrawanthi902cse, Copyright © 1999, 2023, Oracle and/or its affiliates, and Oracle APEX 23.2.4.

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

**QUERY:**

```
SELECT LAST_NAME FROM EMPLOYEE_DETAILS WHERE LAST_NAME LIKE '%a%' AND LAST_NAME LIKE '%e%';
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL workspace interface. The top navigation bar includes 'SQL Commands', 'Schema (WKSP\_SHRAVANTHI902CSI)', and a 'Run' button. Below the toolbar, there are buttons for back, forward, search, and refresh. The main area contains the SQL command: '1 SELECT LAST\_NAME FROM EMPLOYEE\_DETAILS WHERE LAST\_NAME LIKE '%a%' AND LAST\_NAME LIKE '%e%';'. The results section shows a single row with the value 'parker' under the 'LAST\_NAME' column. The bottom status bar indicates '1 rows returned in 0.01 seconds'.

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

**QUERY:**

```
SELECT LAST_NAME, JOB_ID, SALARY FROM EMPLOYEE_DETAILS WHERE JOB_ID IN ('SALES REPRESENTATIVE', 'STOCK CLERK') AND SALARY NOT IN(2500,3500,7000);
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL workspace interface. The top navigation bar includes 'SQL Commands', 'Schema (WKSP\_SHRAVANTHI902CSI)', and a 'Run' button. Below the toolbar, there are buttons for back, forward, search, and refresh. The main area contains the SQL command: '1 SELECT LAST\_NAME, JOB\_ID, SALARY FROM EMPLOYEE\_DETAILS WHERE JOB\_ID IN ('sales representative', 'STOCK CLERK') AND SALARY NOT IN(2500,3500,7000);'. The results section shows a single row with values 'JONES', 'sales representative', and '10000' under the 'LAST\_NAME', 'JOB\_ID', and 'SALARY' columns respectively. The bottom status bar indicates '1 rows returned in 0.00 seconds'.

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

**QUERY:**

```
SELECT LAST_NAME, SALARY, COMMISSION_PCT FROM EMPLOYEE_DETAILS WHERE  
COMMISSION_PCT=0.2;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The query `SELECT LAST_NAME, SALARY, COMMISSION_PCT FROM EMPLOYEE_DETAILS WHERE COMMISSION_PCT=0.2;` is entered in the command field. The results tab is selected, showing the message "no data found".

Schema: WKSP\_SHRAVANTHI902CSI

Language: SQL

Rows: 10

Clear Command Find Tables Save Run

Results Explain Describe Saved SQL History

no data found

220701902@rajalakshmi.edu.in shrawanthi902cse Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

**RESULT:**

# SINGLE ROW FUNCTIONS

**EX.NO.6**

**DATE:**

**Find the Solution for the following:**

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

**QUERY:**

```
SELECT SYSDATE AS "DATE" FROM DUAL;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The query `SELECT SYSDATE AS "DATE" FROM DUAL;` is entered in the command field. The results pane displays a single row with the column `DATE` containing the value `03/13/2024`.

| EMPLOYEE_ID | LAST_NAME | SALARY | NEW_SALARY |
|-------------|-----------|--------|------------|
| 8928        | JONES     | 10000  | 11550      |
| 7433        | parker    | 10000  | 11550      |
| 789         | AMBROSE   | 89000  | 102795     |
| 176         | Fisher    | 70000  | 80850      |
| 2222        | cooper    | 20000  | 23100      |
| 4444        | miller    | 9000   | 10395      |
| 5555        | day       | 50000  | 57750      |
| 1111        | LEE       | 12000  | 13860      |
| 3333        | wilson    | 9000   | 10395      |

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  
EMPLOYEE_DETAILS;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The query `SELECT EMPLOYEE_ID, LAST_NAME, SALARY, SALARY+(15.5/100*SALARY) "NEW_SALARY" FROM EMPLOYEE_DETAILS;` is entered in the command field. The results pane displays a table with columns `EMPLOYEE_ID`, `LAST_NAME`, `SALARY`, and `NEW_SALARY`. The data is as follows:

| EMPLOYEE_ID | LAST_NAME | SALARY | NEW_SALARY |
|-------------|-----------|--------|------------|
| 8928        | JONES     | 10000  | 11550      |
| 7433        | parker    | 10000  | 11550      |
| 789         | AMBROSE   | 89000  | 102795     |
| 176         | Fisher    | 70000  | 80850      |
| 2222        | cooper    | 20000  | 23100      |
| 4444        | miller    | 9000   | 10395      |
| 5555        | day       | 50000  | 57750      |
| 1111        | LEE       | 12000  | 13860      |
| 3333        | wilson    | 9000   | 10395      |

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+(SALARY*15.5/100))-SALARY "INCREASE" FROM  
EMPLOYEE_DETAILS;
```

**OUTPUT:**

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

```
1 SELECT EMPLOYEE_ID, LAST_NAME, SALARY, (SALARY+(SALARY*15.5/100))-SALARY "INCREASE" FROM EMPLOYEE_DETAILS;
```

The results table has four columns: EMPLOYEE\_ID, LAST\_NAME, SALARY, and INCREASE. The data is as follows:

| EMPLOYEE_ID | LAST_NAME | SALARY | INCREASE |
|-------------|-----------|--------|----------|
| 8928        | JONES     | 10000  | 1550     |
| 7433        | parker    | 10000  | 1550     |
| 789         | AMBROSE   | 89000  | 13795    |
| 176         | Fisher    | 70000  | 10850    |
| 2222        | cooper    | 20000  | 3100     |
| 4444        | miller    | 9000   | 1395     |
| 5555        | day       | 50000  | 7750     |
| 1111        | LEE       | 12000  | 1860     |
| 3333        | wilson    | 9000   | 1395     |

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

**QUERY:**

```
SELECT INITCAP(LAST_NAME) "NAME", LENGTH(LAST_NAME)  
"LENGTH OF NAME" FROM EMPLOYEE_DETAILS  
WHERE LAST_NAME LIKE 'J%' OR LAST_NAME LIKE 'A%' OR  
LAST_NAME LIKE 'M%' ORDER BY LAST_NAME;
```

**OUTPUT:**

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

```
1 SELECT INITCAP(LAST_NAME) "NAME", LENGTH(LAST_NAME) "LENGTH OF NAME" FROM EMPLOYEE_DETAILS  
2 WHERE LAST_NAME LIKE 'J%' OR LAST_NAME LIKE 'A%' OR LAST_NAME LIKE 'M%' ORDER BY LAST_NAME;
```

The results table has two columns: NAME and LENGTH OF NAME. The data is as follows:

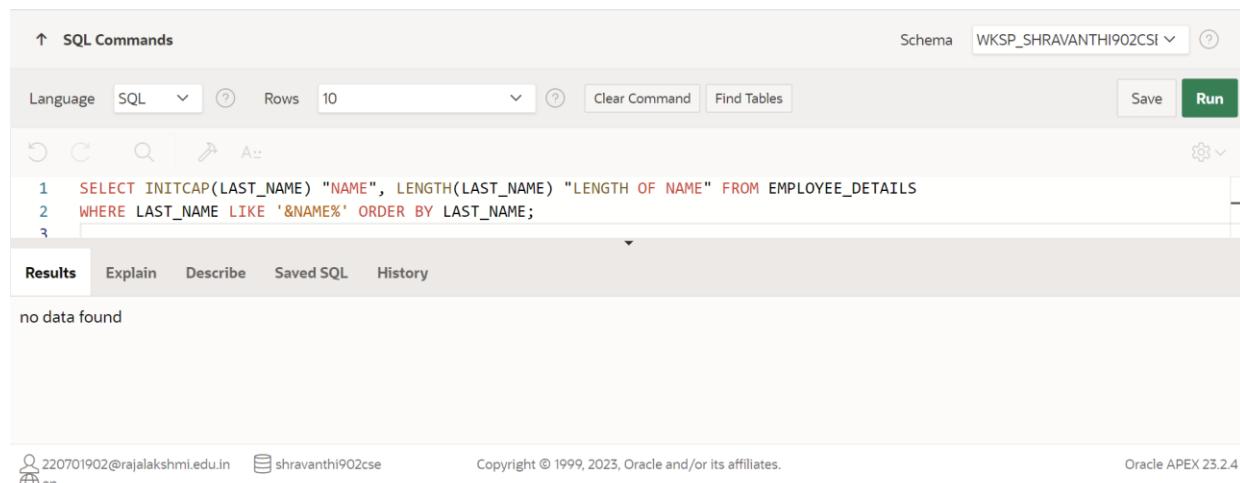
| NAME    | LENGTH OF NAME |
|---------|----------------|
| Ambrose | 7              |
| Jones   | 5              |

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

**QUERY:**

```
SELECT INITCAP(LAST_NAME) "NAME", LENGTH(LAST_NAME)  
"LENGTH OF NAME" FROM EMPLOYEE_DETAILS  
WHERE LAST_NAME LIKE  
'&NAME%' ORDER BY  
LAST_NAME;
```

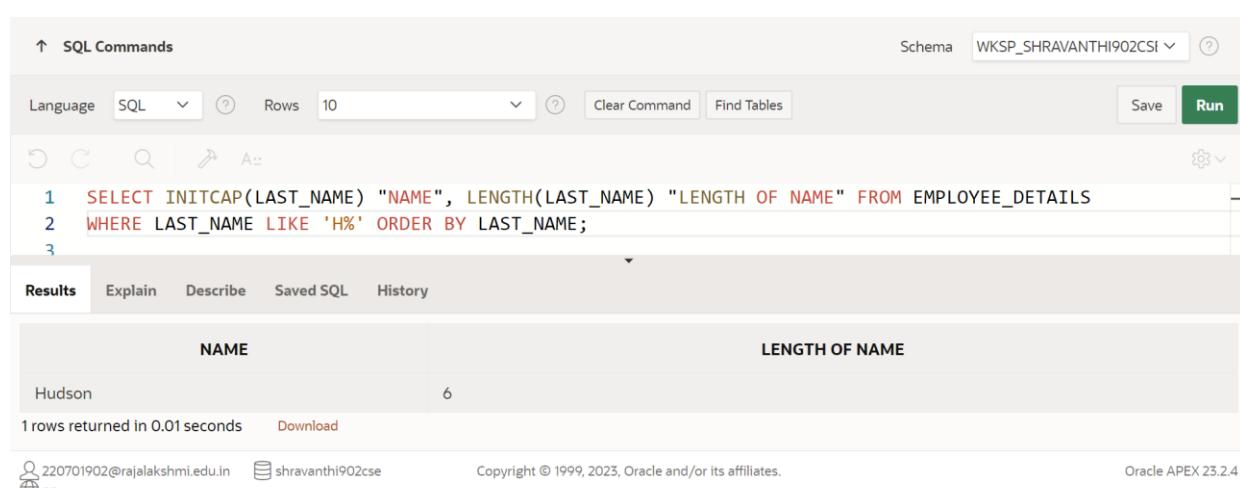
**OUTPUT:**



The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The SQL command entered is:

```
1 SELECT INITCAP(LAST_NAME) "NAME", LENGTH(LAST_NAME) "LENGTH OF NAME" FROM EMPLOYEE_DETAILS  
2 WHERE LAST_NAME LIKE '&NAME%' ORDER BY LAST_NAME;  
3
```

The results tab shows 'no data found'.



The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The SQL command entered is:

```
1 SELECT INITCAP(LAST_NAME) "NAME", LENGTH(LAST_NAME) "LENGTH OF NAME" FROM EMPLOYEE_DETAILS  
2 WHERE LAST_NAME LIKE 'H%' ORDER BY LAST_NAME;  
3
```

The results tab displays a table with two columns: 'NAME' and 'LENGTH OF NAME'. The single row returned is:

| NAME   | LENGTH OF NAME |
|--------|----------------|
| Hudson | 6              |

1 rows returned in 0.01 seconds

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(MONTHS_BETWEEN(SYSDATE,HIRE_DATE),0) MONTHS_WORKED  
FROM EMPLOYEE_DETAILS ORDER BY 2;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. At the top, there's a toolbar with 'SQL Commands' and a schema dropdown set to 'WKSP\_SHRAVANTHI902CSE'. Below the toolbar is a search bar with 'Language: SQL' and a 'Run' button. The main area contains the SQL query:

```
1 SELECT LAST_NAME, ROUND(MONTHS_BETWEEN(SYSDATE,HIRE_DATE),0) MONTHS_WORKED  
2 FROM EMPLOYEE_DETAILS ORDER BY 2;
```

Below the query is a results grid with two columns: 'LAST\_NAME' and 'MONTHS\_WORKED'. The data is as follows:

| LAST_NAME | MONTHS_WORKED |
|-----------|---------------|
| Fisher    | 34            |
| miller    | 37            |
| day       | 46            |
| cooper    | 49            |
| LEE       | 49            |
| GRANT     | 93            |

At the bottom left, there are user profile icons and the text '220701902@rajalakshmi.edu.in' and 'shravanthi902cse'. On the right, it says 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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 "DREAM SALARY" FROM EMPLOYEE_DETAILS;
```

**OUTPUT:**

The screenshot shows the Oracle APEX interface with the following details:

- SQL Commands:** The query is entered here:

```
1 SELECT LAST_NAME||' EARNS $'||SALARY||' MONTHLY BUT WANTS $'||SALARY*3 "DREAM SALARY"  
2 FROM EMPLOYEE_DETAILS;
```
- Results:** The results are displayed in a table with two columns: **DREAM SALARY**. The data is as follows:

| DREAM SALARY                                     |
|--|
| JONES EARNS \$10000 MONTHLY BUT WANTS \$30000    |
| parker EARNS \$10000 MONTHLY BUT WANTS \$30000   |
| AMBROSE EARNS \$89000 MONTHLY BUT WANTS \$267000 |
| Fisher EARNS \$70000 MONTHLY BUT WANTS \$210000  |
| cooper EARNS \$20000 MONTHLY BUT WANTS \$60000   |
| miller EARNS \$9000 MONTHLY BUT WANTS \$27000    |

- Footer:** Includes user information (220701902@rajalakshmi.edu.in, shravanti902cse, en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and version (Oracle APEX 23.2.4).

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

**QUERY:**

```
SELECT LAST_NAME,  
LPAD(SALARY,15,'$') SALARY  
FROM EMPLOYEE_DETAILS;
```

**OUTPUT:**

The screenshot shows the Oracle APEX interface with the following details:

- SQL Commands:** The query is entered here:

```
1 SELECT LAST_NAME, LPAD(SALARY,15,'$') SALARY FROM EMPLOYEE_DETAILS;
```
- Results:** The results are displayed in a table with two columns: **SALARY**. The data is as follows:

| LAST_NAME | SALARY                    |
|-----------|---------------------------|
| JONES     | \$\$\$\$\$\$\$\$\$\$10000 |
| parker    | \$\$\$\$\$\$\$\$\$\$10000 |
| AMBROSE   | \$\$\$\$\$\$\$\$\$\$89000 |
| Fisher    | \$\$\$\$\$\$\$\$\$\$70000 |
| cooper    | \$\$\$\$\$\$\$\$\$\$20000 |
| miller    | \$\$\$\$\$\$\$\$\$\$9000  |

- Footer:** Includes user information (220701902@rajalakshmi.edu.in, shravanti902cse, en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and version (Oracle APEX 23.2.4).

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(HIRE_DATE, 'MONDAY')), 'FMDAY, " THE "DDSPTH  
"OF" MONTH,YYYY') "REVIEW" FROM EMPLOYEE_DETAILS;
```

**OUTPUT:**

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

```
1 SELECT LAST_NAME, HIRE_DATE,  
2 TO_CHAR((NEXT_DAY(HIRE_DATE, 'MONDAY')), 'FMDAY, " THE "DDSPTH "OF" MONTH,YYYY') "REVIEW"  
3 FROM EMPLOYEE_DETAILS;
```

The results table has columns: LAST\_NAME, HIRE\_DATE, and REVIEW. The data is:

| LAST_NAME | HIRE_DATE  | REVIEW                                    |
|-----------|------------|---|
| JONES     | 05/06/1998 | MONDAY, THE ELEVENTH OF MAY,1998          |
| parker    | 02/22/1998 | MONDAY, THE TWENTY-THIRD OF FEBRUARY,1998 |
| AMBROSE   | 07/07/1994 | MONDAY, THE ELEVENTH OF JULY,1994         |
| Fisher    | 05/09/2021 | MONDAY, THE TENTH OF MAY,2021             |
| cooper    | 02/02/2020 | MONDAY, THE THIRD OF FEBRUARY,2020        |
| miller    | 02/02/2021 | MONDAY, THE EIGHTH OF FEBRUARY,2021       |
| day       | 05/05/2020 | MONDAY, THE ELEVENTH OF MAY,2020          |

At the bottom, it says Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4.

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') "DAY"  
FROM EMPLOYEE_DETAILS  
ORDER BY TO_CHAR(HIRE_DATE-1, 'D');
```

**OUTPUT:**

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

```
1 SELECT LAST_NAME, HIRE_DATE, TO_CHAR(HIRE_DATE, 'DAY') "DAY" FROM EMPLOYEE_DETAILS  
2 ORDER BY TO_CHAR(HIRE_DATE-1, 'D');
```

The results table has columns: LAST\_NAME, HIRE\_DATE, and DAY. The data is:

| LAST_NAME | HIRE_DATE  | DAY       |
|-----------|------------|-----------|
| HUDSON    | 08/10/2015 | MONDAY    |
| miller    | 02/02/2021 | TUESDAY   |
| day       | 05/05/2020 | TUESDAY   |
| GRANT     | 06/07/2016 | TUESDAY   |
| JONES     | 05/06/1998 | WEDNESDAY |
| LEE       | 02/12/2020 | WEDNESDAY |
| AMBROSE   | 07/07/1994 | THURSDAY  |
| cooper    | 02/02/2020 | SUNDAY    |
| wilson    | 02/11/2020 | SUNDAY    |

At the bottom, it says Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4.

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

## **RESULT:**

# 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.dept_id, d.dept_name FROM employee_details e,
department d
WHERE e.dept_id = d.dept_id;
```

OUTPUT:

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

```
1 SELECT e.last_name, e.dept_id, d.dept_name FROM employee_details e, department d
2 WHERE e.dept_id = d.dept_id;
```

The results are displayed in a grid:

| LAST_NAME | DEPT_ID | DEPT_NAME       |
|-----------|---------|-----------------|
| JONES     | 39      | Sales           |
| Fisher    | 6       | R&D             |
| cooper    | 2       | HUMAN RESOURCES |
| miller    | 5       | ADMINISTRATION  |
| day       | 4       | MARKETING       |
| LEE       | 1       | FINANCE         |
| wilson    | 3       | IT              |

7 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, location_id FROM employee_details, department
WHERE employee_details.dept_id = department.dept_id AND
employee_details.dept_id = 80;
```

OUTPUT:

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

```
1 SELECT DISTINCT job_id, location_id FROM employee_details, department
2 WHERE employee_details.dept_id = department.dept_id AND employee_details.dept_id = 80;
```

The results are displayed in a grid:

| JOB_ID          | LOCATION_ID |
|-----------------|-------------|
| sales executive | 78          |

1 rows returned in 0.00 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, d.dept_name, d.location_id, l.city FROM
employee_details e, department d, location_l WHERE e.dept_id
= d.dept_id AND d.location_id = l.location_id AND
e.commission_pct IS NOT NULL;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The query has been run and returned one row:

| LAST_NAME | DEPT_NAME | LOCATION_ID | CITY    |
|-----------|-----------|-------------|---------|
| HUDSON    | sales     | 1700        | CHENNAI |

1 rows returned in 0.01 seconds

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

**QUERY:**

```
SELECT last_name, dept_name
FROM employee_details, department
WHERE employee_details.dept_id = department.dept_id AND last_name LIKE
'%a%';
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The query has been run and returned one row:

| LAST_NAME | DEPT_NAME       |
|-----------|-----------------|
| cooper    | HUMAN RESOURCES |

1 rows returned in 0.00 seconds

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.job_id, e.dept_id, d.dept_name  
FROM employee_details e JOIN department d ON (e.dept_id = d.dept_id)  
JOIN location_ l ON (d.location_id = l.location_id)  
WHERE LOWER(l.city) = 'toronto';
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Worksheet interface. The SQL command is displayed in the editor, and the results are shown in a table below. The table has four columns: LAST\_NAME, JOB\_ID, DEPT\_ID, and DEPT\_NAME. Two rows are returned, both for department 80 (sales). The first row is for JONES (sales representative) and the second for HUDSON (sales executive).

| LAST_NAME | JOB_ID               | DEPT_ID | DEPT_NAME |
|-----------|----------------------|---------|-----------|
| JONES     | sales representative | 80      | sales     |
| HUDSON    | sales executive      | 80      | sales     |

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.employee_id "EMP#", m.name "Manager",  
m.manager_id "Mgr#"  
FROM employee_details w join MANAGER m ON (w.manager_id = m.manager_id);
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Worksheet interface. The SQL command is displayed in the editor, and the results are shown in a table below. The table has four columns: Employee, EMP#, Manager, and Mgr#. Three rows are returned, each mapping an employee to their manager. The first row is for cooper (179, LEE, 10002), the second for miller (178, JONES, 10001), and the third for GRANT (877, LEE, 10002).

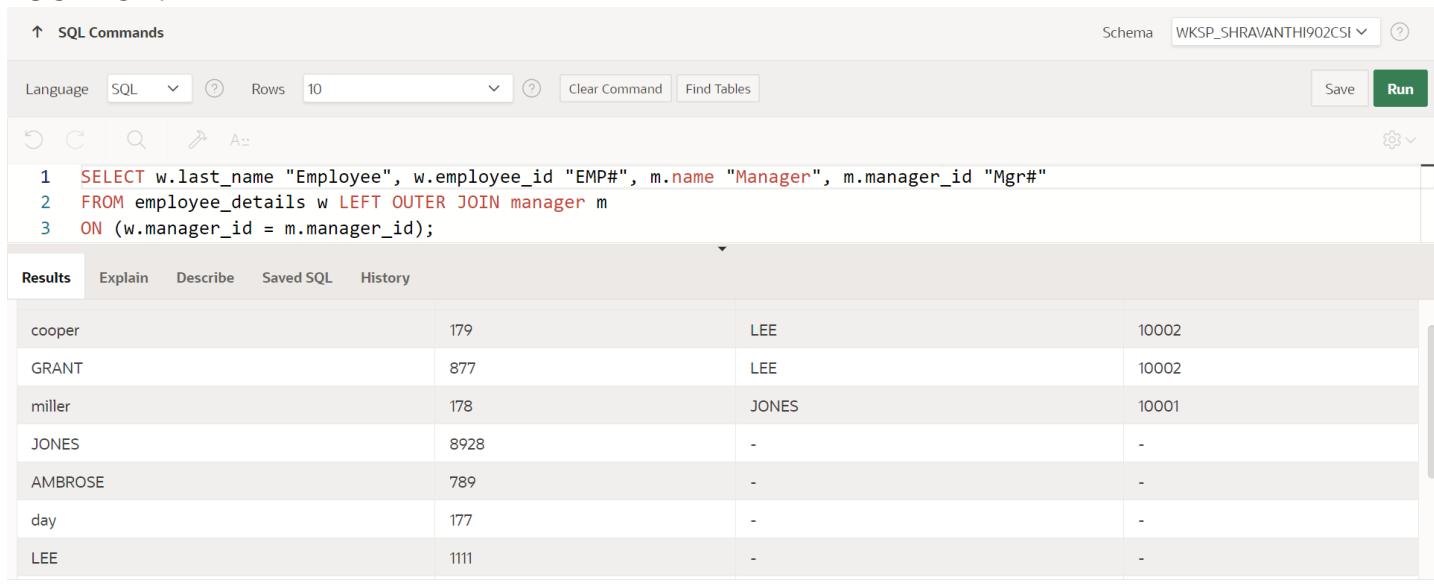
| Employee | EMP# | Manager | Mgr#  |
|----------|------|---------|-------|
| cooper   | 179  | LEE     | 10002 |
| miller   | 178  | JONES   | 10001 |
| GRANT    | 877  | LEE     | 10002 |

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.employee_id "EMP#", m.name "Manager",  
m.manager_id "Mgr#"  
  
FROM employee_details w LEFT OUTER JOIN manager m  
ON (w.manager_id = m.manager_id);
```

#### OUTPUT:



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

```
1 SELECT w.last_name "Employee", w.employee_id "EMP#", m.name "Manager", m.manager_id "Mgr#"  
2 FROM employee_details w LEFT OUTER JOIN manager m  
3 ON (w.manager_id = m.manager_id);
```

The results section displays the following data:

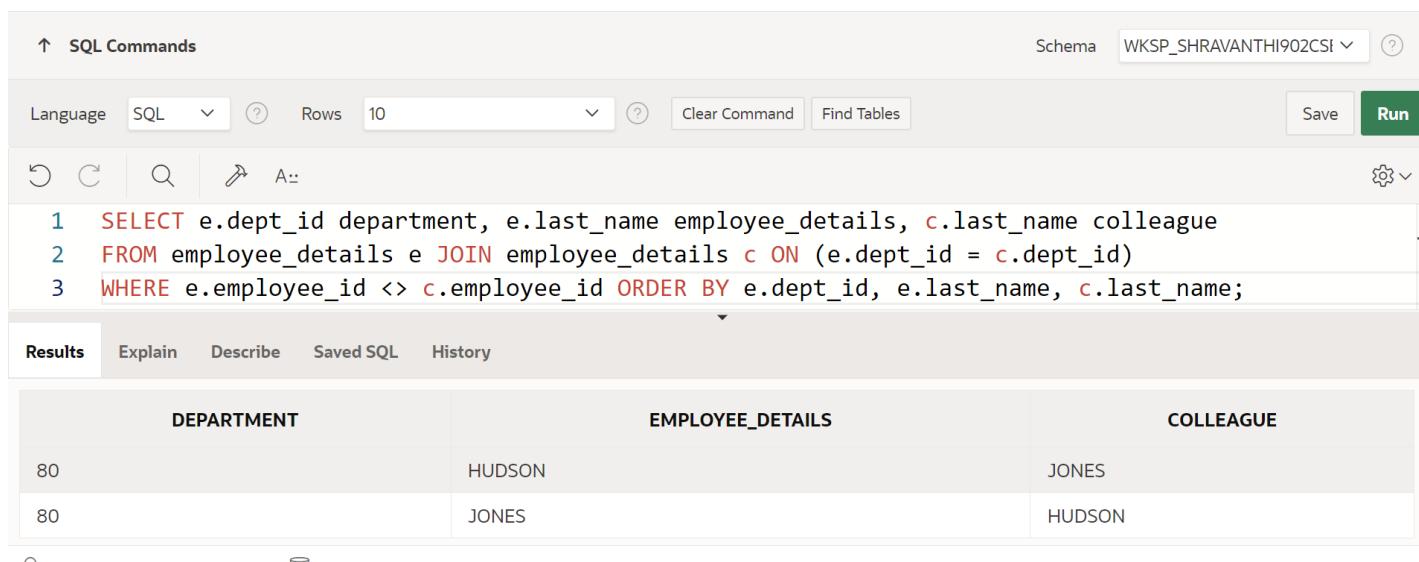
| Employee | Employee ID | Manager | Manager ID |
|----------|-------------|---------|------------|
| cooper   | 179         | LEE     | 10002      |
| GRANT    | 877         | LEE     | 10002      |
| miller   | 178         | JONES   | 10001      |
| JONES    | 8928        | -       | -          |
| AMBROSE  | 789         | -       | -          |
| day      | 177         | -       | -          |
| LEE      | 1111        | -       | -          |

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

#### QUERY:

```
SELECT e.dept_id department, e.last_name employee_details, c.last_name colleague  
FROM employee_details e JOIN employee_details c ON (e.dept_id = c.dept_id)  
WHERE e.employee_id <> c.employee_id ORDER BY e.dept_id,  
e.last_name, c.last_name;
```

#### OUTPUT:



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

```
1 SELECT e.dept_id department, e.last_name employee_details, c.last_name colleague  
2 FROM employee_details e JOIN employee_details c ON (e.dept_id = c.dept_id)  
3 WHERE e.employee_id <> c.employee_id ORDER BY e.dept_id, e.last_name, c.last_name;
```

The results section displays the following data:

| DEPARTMENT | EMPLOYEE_DETAILS | COLLEAGUE |
|------------|------------------|-----------|
| 80         | HUDSON           | JONES     |
| 80         | JONES            | HUDSON    |

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 employee_details e JOIN department d ON (e.dept_id = d.dept_id)  
JOIN jobgrade j ON (e.salary BETWEEN j.lowest_sal AND j.highest_sal);
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL workspace interface. The top navigation bar includes 'SQL Commands', 'Schema (WKSP\_SHRAVANTHI902CSI)', 'Save', and 'Run' buttons. Below the toolbar, there are filters for 'Language (SQL)', 'Rows (10)', 'Clear Command', and 'Find Tables'. The main area contains the SQL query and its results.

**SQL Query:**

```
1 SELECT e.last_name, e.job_id, d.dept_name, e.salary, j.grade_level  
2 FROM employee_details e JOIN department d ON (e.dept_id = d.dept_id) JOIN job  
3 ON (e.salary BETWEEN j.lowest_sal AND j.highest_sal);
```

**Results:**

| LAST_NAME | JOB_ID               | DEPT_NAME       | SALARY | GRADE_LEVEL |
|-----------|----------------------|-----------------|--------|-------------|
| LEE       | 122320               | FINANCE         | 12000  | 2           |
| wilson    | 021120               | IT              | 9000   | 2           |
| miller    | 020221               | ADMINISTRATION  | 9000   | 2           |
| cooper    | 020220               | HUMAN RESOURCES | 20000  | 2           |
| JONES     | sales representative | sales           | 10000  | 2           |
| JONES     | sales representative | sales           | 10000  | 2           |

Page footer: 220701902@rajalakshmi.edu.in shrawanthi902cse en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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 employee_details e, employee_details davies  
WHERE davies.last_name = 'Davies' AND davies.hire_date < e.hire_date;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL workspace interface. The top navigation bar includes 'SQL Commands', 'Schema (WKSP\_SHRAVANTHI902CSI)', 'Save', and 'Run' buttons. Below the toolbar, there are filters for 'Language (SQL)', 'Rows (10)', 'Clear Command', and 'Find Tables'. The main area contains the SQL query and its results.

**SQL Query:**

```
1 SELECT e.last_name, e.hire_date FROM employee_details e, employee_details davies  
2 WHERE davies.last_name = 'Davies' AND davies.hire_date < e.hire_date;
```

**Results:**

| LAST_NAME | HIRE_DATE  |
|-----------|------------|
| JONES     | 05/06/1998 |
| parker    | 02/22/1998 |
| Davies    | 05/09/2021 |
| cooper    | 02/02/2020 |
| miller    | 02/02/2021 |
| day       | 05/05/2020 |

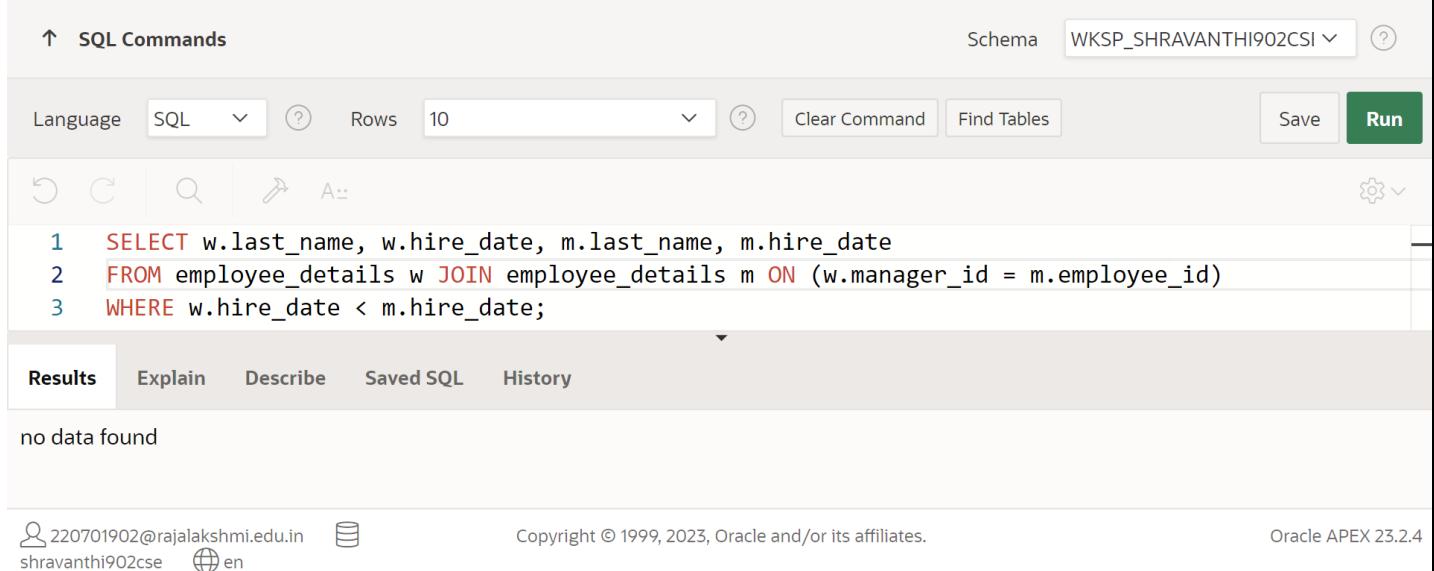
Page footer: 220701902@rajalakshmi.edu.in shrawanthi902cse en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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 w.last_name, w.hire_date, m.last_name, m.hire_date
FROM employee_details w JOIN employee_details m ON (w.manager_id =
m.employee_id)
WHERE w.hire_date < m.hire_date;
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'SQL Commands', 'Schema' set to 'WKSP\_SHRAVANTHI902CSI', and a 'Run' button. Below the toolbar are buttons for undo, redo, search, and other database operations. The main area contains the SQL query:

```
1 SELECT w.last_name, w.hire_date, m.last_name, m.hire_date
2 FROM employee_details w JOIN employee_details m ON (w.manager_id = m.employee_id)
3 WHERE w.hire_date < m.hire_date;
```

The 'Results' tab is selected, showing the message 'no data found'. At the bottom, user information (220701902@rajalakshmi.edu.in, shravanthi902cse) and system details (Copyright © 1999, 2023, Oracle and/or its affiliates, Oracle APEX 23.2.4) are displayed.

| 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
employee_details;
```

**OUTPUT:**

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

```
1 select ROUND(MAX(salary),0) "Maximum",ROUND(MIN(salary),0) "Minimum",
2 ROUND(SUM(salary),0) "Sum",ROUND(AVG(salary),0) "Average"FROM employee_details;
```

The results section displays the following output:

| Maximum | Minimum | Sum    | Average |
|---------|---------|--------|---------|
| 90000   | 7000    | 376000 | 34182   |

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

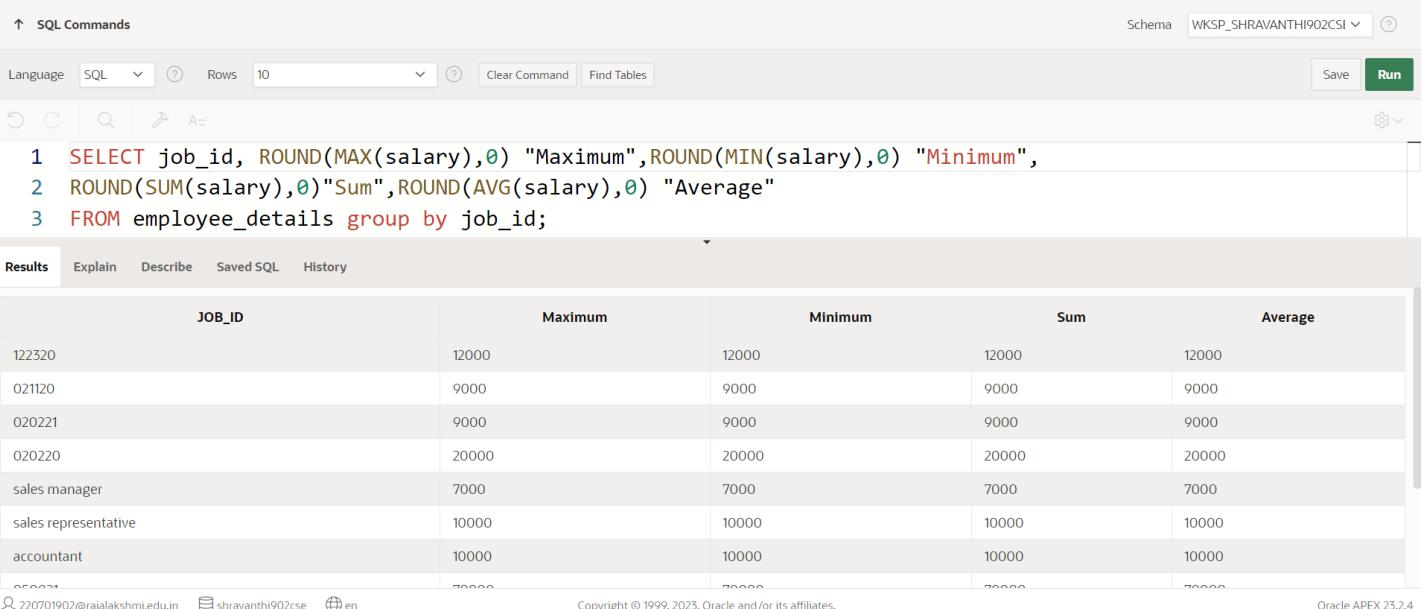
Page footer: 220701902@rajalakshmi.edu.in, shrawanthi902cse, Copyright © 1999, 2023, Oracle and/or its affiliates, Oracle APEX 23.2.4

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 employee_details group by job_id;
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the modified query. The results window displays a table with columns: JOB\_ID, Maximum, Minimum, Sum, and Average. The data shows that all job types have the same values for Maximum, Minimum, Sum, and Average, which is incorrect based on the query. The results are as follows:

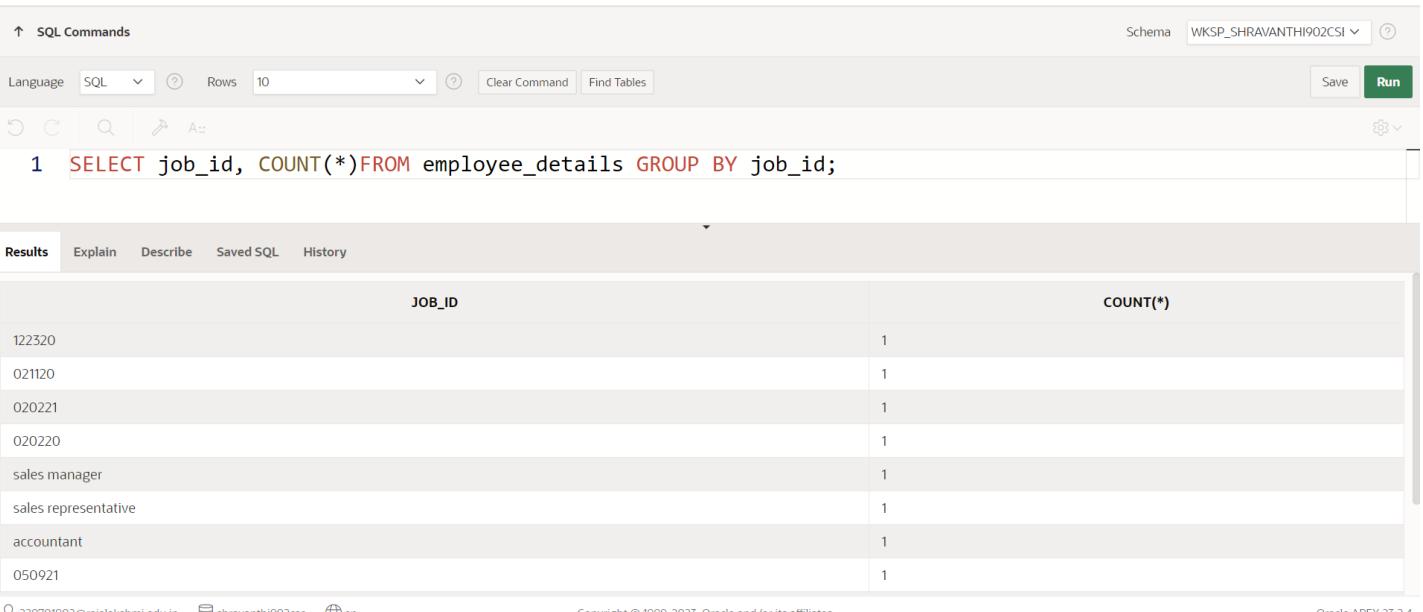
| JOB_ID               | Maximum | Minimum | Sum   | Average |
|----------------------|---------|---------|-------|---------|
| 122320               | 12000   | 12000   | 12000 | 12000   |
| 021120               | 9000    | 9000    | 9000  | 9000    |
| 020221               | 9000    | 9000    | 9000  | 9000    |
| 020220               | 20000   | 20000   | 20000 | 20000   |
| sales manager        | 7000    | 7000    | 7000  | 7000    |
| sales representative | 10000   | 10000   | 10000 | 10000   |
| accountant           | 10000   | 10000   | 10000 | 10000   |
| 050921               | 70000   | 70000   | 70000 | 70000   |

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 employee_details GROUP BY job_id;
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the generalized query. The results window displays a table with columns: JOB\_ID and COUNT(\*). The data shows that each job type has exactly one person assigned to it. The results are as follows:

| JOB_ID               | COUNT(*) |
|----------------------|----------|
| 122320               | 1        |
| 021120               | 1        |
| 020221               | 1        |
| 020220               | 1        |
| sales manager        | 1        |
| sales representative | 1        |
| accountant           | 1        |
| 050921               | 1        |

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 employee_details;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL workspace interface. The query `SELECT COUNT(DISTINCT manager_id) "Number of Managers" FROM employee_details;` is entered in the command field. The results show a single row with the value 7, labeled "Number of Managers".

| Number of Managers |
|--------------------|
| 7                  |

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

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

**QUERY:**

```
SELECT MAX(salary) - MIN(salary) DIFFERENCE FROM employee_details;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL workspace interface. The query `SELECT MAX(salary) - MIN(salary) DIFFERENCE FROM employee_details;` is entered in the command field. The results show a single row with the value 83000, labeled "DIFFERENCE".

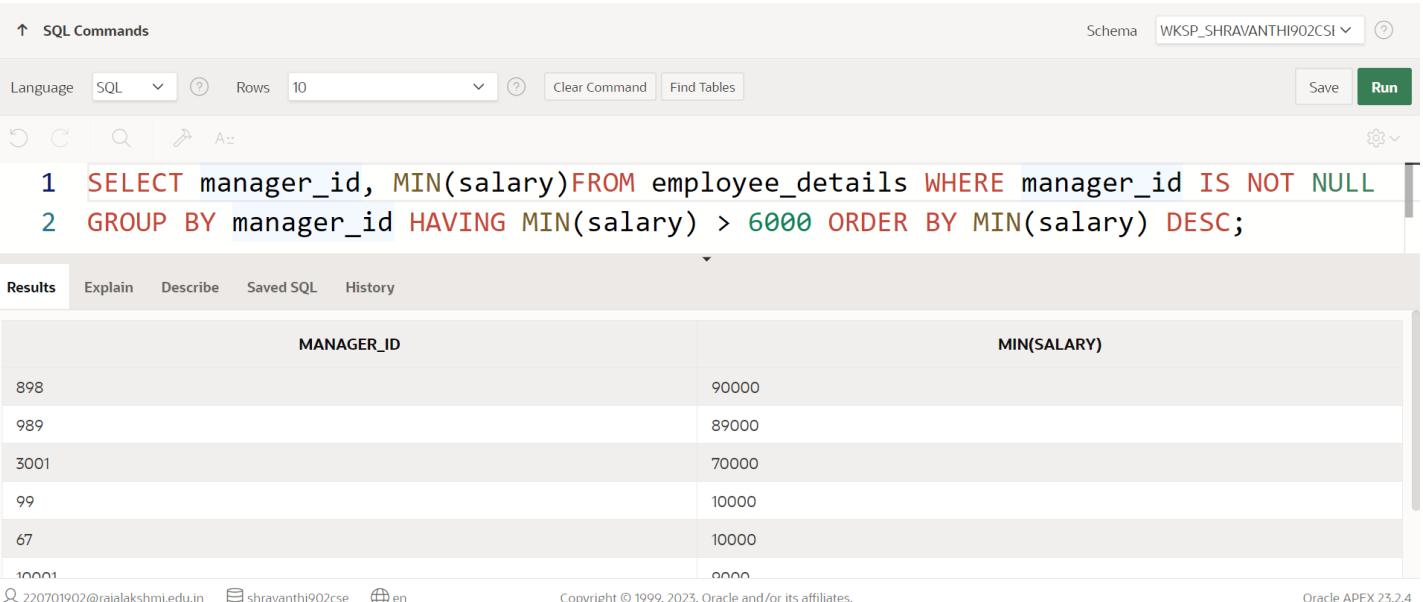
| DIFFERENCE |
|------------|
| 83000      |

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 employee_details WHERE manager_id IS NOT NULL  
GROUP BY manager_id HAVING MIN(salary) > 6000 ORDER BY MIN(salary) DESC;  
OUTPUT:
```



The screenshot shows an Oracle APEX SQL Worksheet interface. The SQL command entered is:

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

The Results tab displays the output:

| MANAGER_ID | MIN(SALARY) |
|------------|-------------|
| 898        | 90000       |
| 989        | 89000       |
| 3001       | 70000       |
| 99         | 10000       |
| 67         | 10000       |
| 10001      | 8000        |

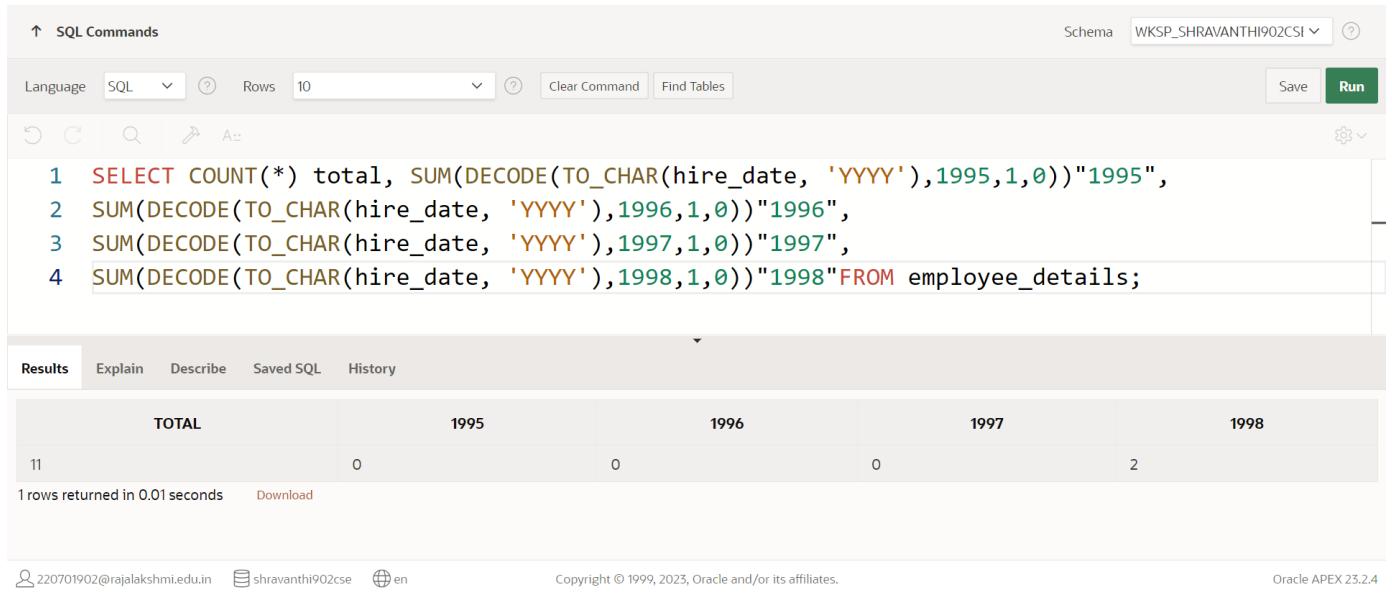
At the bottom, it shows the user information: 220701902@rajalakshmi.edu.in, shrawanthi902cse, en, and the copyright notice: Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4.

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 employee_details;
```

**OUTPUT:**



The screenshot shows an Oracle APEX SQL Worksheet interface. The SQL command entered is:

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

The Results tab displays the output:

| TOTAL | 1995 | 1996 | 1997 | 1998 |
|-------|------|------|------|------|
| 11    | 0    | 0    | 0    | 2    |

At the bottom, it shows the user information: 220701902@rajalakshmi.edu.in, shrawanthi902cse, en, and the copyright notice: Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4.

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)) "Dept 20", SUM(DECODE(dept_id , 50, salary)) "Dept 50", SUM(DECODE(dept_id , 80, salary)) "Dept 80", SUM(DECODE(dept_id , 90, salary)) "Dept 90", SUM(salary) "Total" FROM employee_details GROUP BY job_id;
```

#### OUTPUT:

The screenshot shows the Oracle APEX interface with the following details:

- SQL Commands:** The query is displayed in the command editor.
- Results:** The results are presented in a grid format with columns: Job, Dept 20, Dept 50, Dept 80, Dept 90, and Total.
- Data:**

| Job           | Dept 20 | Dept 50 | Dept 80 | Dept 90 | Total |
|---------------|---------|---------|---------|---------|-------|
| 122320        | -       | -       | -       | -       | 12000 |
| 021120        | -       | -       | -       | -       | 9000  |
| 020221        | -       | -       | -       | -       | 9000  |
| 020220        | -       | -       | -       | -       | 20000 |
| sales manager | -       | -       | -       | -       | 7000  |
- Session Information:** User 220701902@rajalakshmi.edu.in, Schema WKSP\_SHRAVANTHI902CSI, Language SQL, Rows 10.
- Copyright:** Copyright © 1999, 2023, Oracle and/or its affiliates.
- Version:** Oracle APEX 23.2.4

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 "deptname",d.location_id as "department location",  
count(*) as "Number of people",  
round(avg(e.salary),2) as "salary"  
from department d inner join employee_details e on (d.dept_id=e.dept_id)  
Group by d.dept_name,d.location_id;
```

#### OUTPUT:

The screenshot shows the Oracle APEX interface with the following details:

- SQL Commands:** The query is displayed in the command editor.
- Results:** The results are presented in a grid format with columns: deptname, department location, Number of people, and salary.
- Data:**

| deptname        | department location | Number of people | salary |
|-----------------|---------------------|------------------|--------|
| HUMAN RESOURCES | 20                  | 1                | 20000  |
| ADMINISTRATION  | 50                  | 1                | 9000   |
| IT              | 30                  | 1                | 9000   |
| sales           | 4000                | 2                | 50000  |
| MARKETING       | 40                  | 1                | 50000  |
| FINANCE         | 10                  | 1                | 12000  |
- Session Information:** User 220701902@rajalakshmi.edu.in, Schema WKSP\_SHRAVANTHI902CSI, Language SQL, Rows 10.
- Copyright:** Copyright © 1999, 2023, Oracle and/or its affiliates.
- Version:** Oracle APEX 23.2.4

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

**RESULT:**

# SUB-QUERIES

EX.NO:9

DATE:

Find the Solution for the following:

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 employee_details  
where dept_id =(select dept_id from employee_details where last_name  
like 'zlotkey')  
and last_name <> 'zlotkey';
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'SQL Commands', 'Schema' set to 'WKSP\_SHRAVANTHI902CSI', and a 'Run' button. Below the toolbar, there are buttons for Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run. The main area contains the SQL query:

```
1 Select last_name, hire_date from employee_details  
2 where dept_id =(select dept_id from employee_details where last_name like 'zlotkey')  
3 and last_name <> 'zlotkey'
```

The 'Results' tab is selected, showing the message 'no data found'.

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 employee_details where salary > (select avg(salary) from employee_details) order by salary;
```

**OUTPUT:**

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

```
1 select employee_id, last_name, salary from employee_details
2 where salary > (select avg(salary) from employee_details) order by salary;
```

The results table has columns: EMPLOYEE\_ID, LAST\_NAME, and SALARY. The data is:

| EMPLOYEE_ID | LAST_NAME | SALARY |
|-------------|-----------|--------|
| 177         | day       | 50000  |
| 176         | Davies    | 70000  |
| 789         | Davies    | 89000  |
| 840         | HILTON    | 90000  |

At the bottom, it shows the user information: 220701902@rajalakshmi.edu.in, shravanthi902cse, Copyright © 1999, 2023, Oracle and/or its affiliates, and Oracle APEX 23.2.4.

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 employee_details
where dept_id in (select dept_id from employee_details where
last_name like '%u%');
```

**OUTPUT:**

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

```
1 select employee_id, last_name from employee_details
2 where dept_id in (select dept_id from employee_details where last_name like '%u%');
```

The results table has columns: EMPLOYEE\_ID and LAST\_NAME. The message at the top says "no data found".

At the bottom, it shows the user information: 220701902@rajalakshmi.edu.in, shravanthi902cse, Copyright © 1999, 2023, Oracle and/or its affiliates, and Oracle APEX 23.2.4.

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

**QUERY:**

```
select last_name, dept_id, job_id from
employee_details
where dept_id in (select dept_id from department where
location_id =1700);
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The query is:

```
1 select last_name, dept_id, job_id from employee_details
2 where dept_id in (select dept_id from department where location_id =1700);
```

The results table has columns 'LAST\_NAME', 'DEPT\_ID', and 'JOB\_ID'. One row is returned:

| LAST_NAME | DEPT_ID | JOB_ID     |
|-----------|---------|------------|
| parker    | 20      | accountant |

1 rows returned in 0.01 seconds. The user is 'shravanthi902cse'.

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 employee_details
where manager_id in (select employee_id from employee_details
where last_name='King');
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The query is:

```
1 select last_name, salary from employee_details
2 where manager_id in (select employee_id from employee_details
3 where last_name='King');
```

The results table has columns 'LAST\_NAME' and 'SALARY'. No data is found.

| LAST_NAME     | SALARY |
|---------------|--------|
| no data found |        |

no data found. The user is 'shravanthi902cse'.

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

**QUERY:**

```
select dept_id, last_name, job_id from employee_details where dept_id in  
(select dept_id from department where dept_name = 'executive');
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The query entered is:

```
1 select dept_id, last_name, job_id from employee_details  
2 where dept_id in (select dept_id from department where dept_name = 'executive');  
3
```

The results tab shows the message "no data found". The bottom of the screen displays user information (220701902@rajalakshmi.edu.in, shrawanthi902cse), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the version (Oracle APEX 23.2).

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 employee_details  
where salary > (select avg(salary) from employee_details) and  
dept_id in (select dept_id from employee_details where last_name like  
'%u%');
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The query entered is:

```
1 select employee_id, last_name, salary from employee_details  
2 where salary > (select avg(salary) from employee_details) and  
3 dept_id in (select dept_id from employee_details where last_name like '%u%');
```

The results tab shows the message "no data found". The bottom of the screen displays user information (220701902@rajalakshmi.edu.in, shrawanthi902cse), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the version (Oracle APEX 23.2).

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

**RESULT:**

# USING THE SET OPERATORS

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

OUTPUT:

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

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

The results section displays the following data:

| ID | NAME      |
|----|-----------|
| 10 | Marketing |
| 50 | technical |

2 rows returned in 0.05 seconds

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

QUERY:

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

OUTPUT:

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

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

The results section displays the following data:

| ID | NAME   |
|----|--------|
| 1  | India  |
| 3  | london |

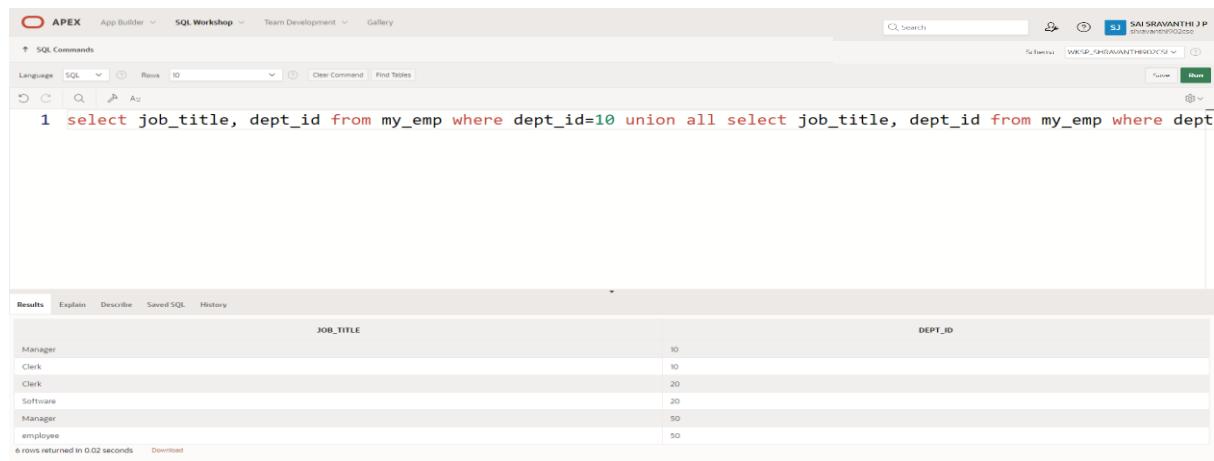
2 rows returned in 0.05 seconds

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

**QUERY:**

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

**OUTPUT:**



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

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

The results section displays the output:

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

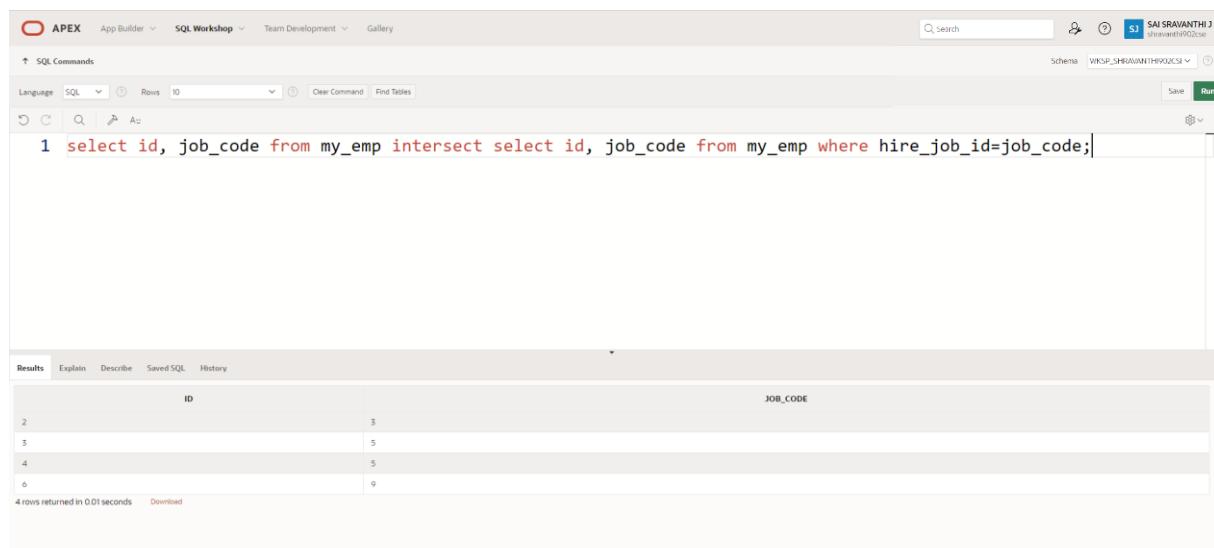
6 rows returned in 0.02 seconds [Download](#)

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

**QUERY:**

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

**OUTPUT:**



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

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

The results section displays the output:

| ID | JOB_CODE |
|----|----------|
| 2  | 3        |
| 3  | 5        |
| 4  | 5        |
| 6  | 9        |

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

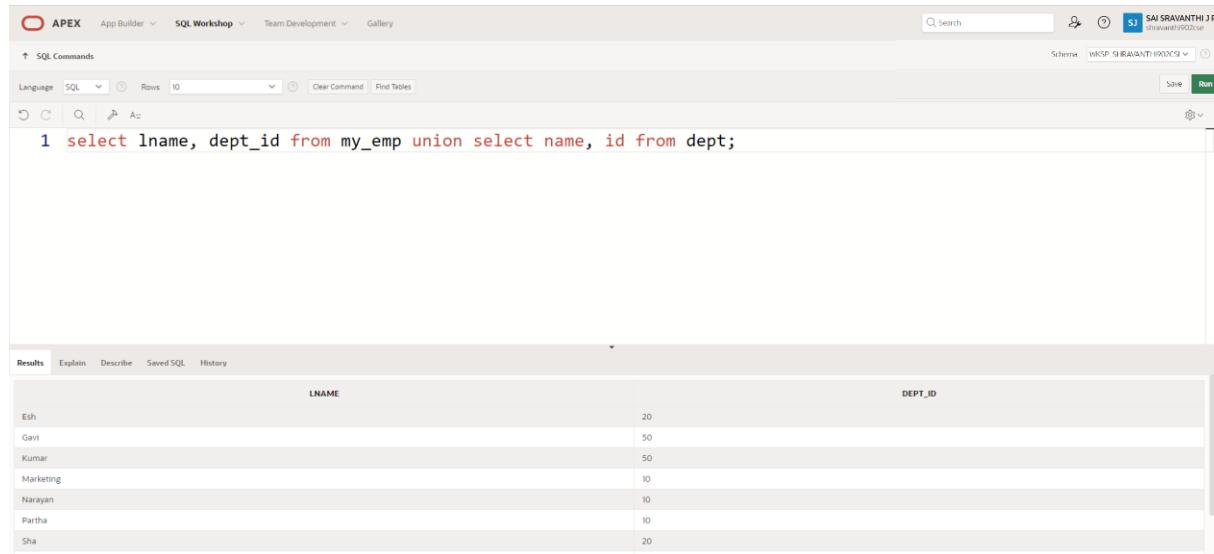
**5. The HR department needs a report with the following specifications:**

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The SQL tab is selected. The command window contains the following SQL code:

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

The results window displays the output of the query:

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

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

**RESULT:**

# CREATING VIEWS

EXNO: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 view employee_vu as
Select employee_id as empno, first_name || ' ' || last_name as
employee, department_id as deptno
From employees;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, a CREATE VIEW statement is being typed into the editor. The schema is set to WKSP\_SHRAVANTHI902CSI. The Run button is highlighted. Below the editor, the Results tab is selected, showing the message "View created." and the execution time "0.02 seconds". The bottom status bar includes user information and the Oracle APEX version.

```
1 CREATE VIEW EMPLOYEE_VU AS
2 SELECT EMPLOYEE_ID AS EMPNO,
3       FIRST_NAME || ' ' || LAST_NAME AS EMPLOYEE,
4       DEPARTMENT_ID AS DEPTNO
5  FROM EMPLOYEES;
```

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

**QUERY:**

```
SELECT * FROM EMPLOYEE_VU;
```

**OUTPUT:**

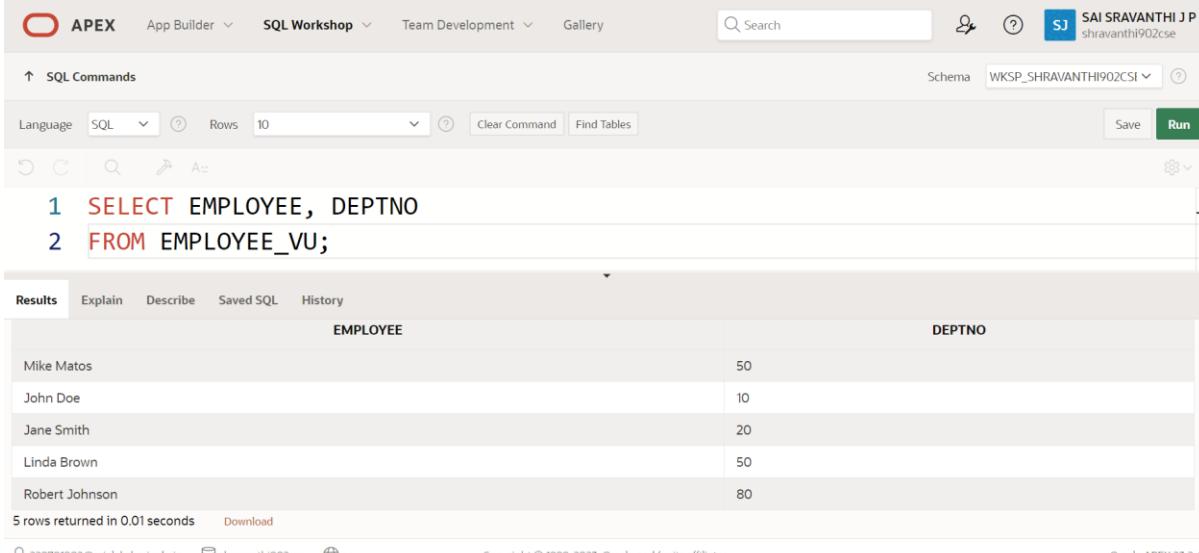
The screenshot shows the Oracle APEX SQL Workshop interface with the Results tab selected. The query "SELECT \* FROM EMPLOYEE\_VU;" is run, and the results are displayed in a table. The table has three columns: EMPNO, EMPLOYEE, and DEPTNO. The data rows are: (3, Mike Matos, 50), (1, John Doe, 10), (2, Jane Smith, 20), (4, Linda Brown, 50), and (5, Robert Johnson, 80). The bottom status bar indicates "5 rows returned in 0.02 seconds".

| EMPNO | EMPLOYEE       | DEPTNO |
|-------|----------------|--------|
| 3     | Mike Matos     | 50     |
| 1     | John Doe       | 10     |
| 2     | Jane Smith     | 20     |
| 4     | Linda Brown    | 50     |
| 5     | Robert Johnson | 80     |

**3. using your EMPLOYEES\_VU view, enter a query to display all employees' names and department.**

**QUERY:** `SELECT EMPLOYEE, DEPTNO  
FROM EMPLOYEE_VU;`

**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 user information for 'SAI SRAVANTHI J P' are also present. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_SHRAVANTHI902CSI'. The SQL editor contains the following code:

```
1 SELECT EMPLOYEE, DEPTNO
2 FROM EMPLOYEE_VU;
```

The results section displays the output of the query:

| EMPLOYEE       | DEPTNO |
|----------------|--------|
| Mike Matos     | 50     |
| John Doe       | 10     |
| Jane Smith     | 20     |
| Linda Brown    | 50     |
| Robert Johnson | 80     |

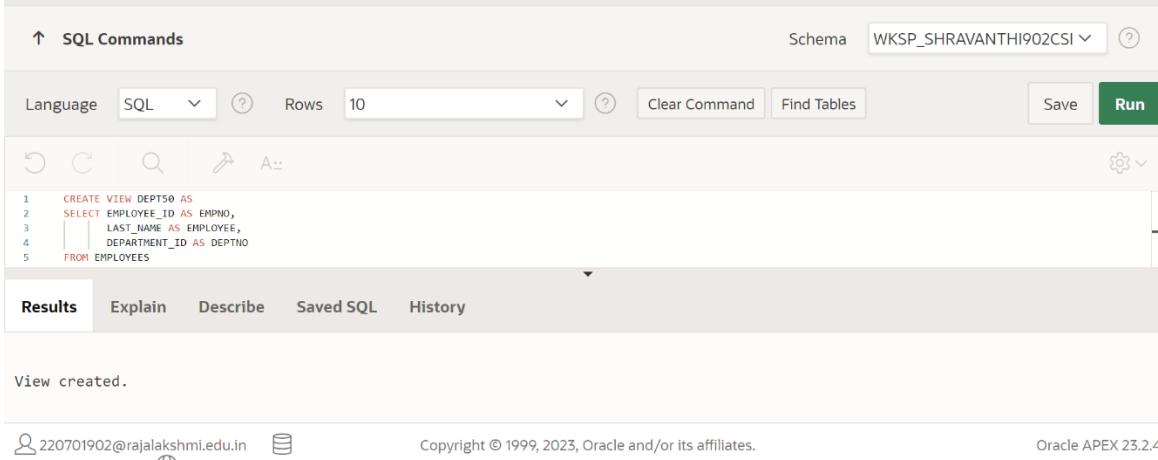
Below the table, it says '5 rows returned in 0.01 seconds' and provides download options. The bottom of the screen shows copyright information and the version 'Oracle APEX 23.2.4'.

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar and schema selection are identical to the previous screenshot. The SQL editor contains the code for creating the view:

```
1 CREATE VIEW DEPT50 AS
2 SELECT EMPLOYEE_ID AS EMPNO,
3       LAST_NAME AS EMPLOYEE,
4       DEPARTMENT_ID AS DEPTNO
5  FROM EMPLOYEES
```

The results section shows the message 'View created.' Below the message, the user information and copyright notice are visible.

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

**QUERY:**  
DESCRIBE DEPT50;

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (which is selected), 'Team Development', 'Gallery', and a search bar. The schema dropdown is set to 'WKSP\_SHRAVANTHI902CSI'. The main area shows the SQL Commands section with the query 'DESCRIBE DEPT50;'. Below the query, the 'Describe' tab is selected in the results panel, showing the structure of the view 'DEPT50'. The results table has columns: Table, Column, Data Type, Length, Precision, Scale, Primary Key, Nullable, Default, and Comment. The data shows three columns: EMPNO (NUMBER, 22), EMPLOYEE (VARCHAR2, 20), and DEPTNO (NUMBER, 22). Both EMPLOYEE and DEPTNO are marked as nullable. At the bottom, it shows the user '220701902@rajalakshmi.edu.in' and 'shravanthi902cse', the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.', and the version 'Oracle APEX 23.2.4'.

Select \* from dept50;

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', and a search bar. The schema dropdown is set to 'WKSP\_SHRAVANTHI902CSI'. The main area shows the SQL Commands section with the query 'SELECT \* FROM DEPT50;'. Below the query, the 'Results' tab is selected in the results panel, displaying the output of the query. The output shows two rows of data: row 3 with EMPNO 3, EMPLOYEE Matos, and DEPTNO 50; and row 4 with EMPNO 4, EMPLOYEE Brown, and DEPTNO 50. At the bottom, it shows the user '220701902@rajalakshmi.edu.in' and 'shravanthi902cse', the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.', and the version 'Oracle APEX 23.2.4'.

## 6. 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. In the SQL Commands tab, there is an error message: "Error at line 2/5: ORA-00904: "DEPARTMENT\_ID": invalid identifier". The error is highlighted in a yellow box. The SQL command is:

```
1. UPDATE DEPT50
2. SET DEPARTMENT_ID = 80
3. WHERE EMPLOYEE = 'Matos';
```

The Results tab shows the error message. At the bottom, it displays user information and copyright details.

## 7. Create a view called SALARY\_VU based on the employee last names, department names, salaries, and salary grades for all employees. Use the Employees, DEPARTMENTS and JOB\_GRADE tables. Label the column Employee, Department, salary, and Grade respectively.

QUERY: create view salary\_vu as select lname EMPLOYEE, dept\_name  
DEPARTMENT, salary SALARY, sal\_grade GRADE from my\_emp;

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command to create the view is:

```
1. CREATE VIEW SALARY_VU AS
2. SELECT E.LAST_NAME AS EMPLOYEE,
3.        D.DEPARTMENT_NAME AS DEPARTMENT,
4.        E.SALARY AS SALARY,
5.        J.GRADE_LEVEL AS GRADE
6.   FROM EMPLOYEES E
7.  JOIN DEPARTMENTS D ON E.DEPARTMENT_ID = D.DEPARTMENT_ID
8.  JOIN JOB_GRADES J ON E.SALARY BETWEEN J.LOWEST_SAL AND J.HIGHEST_SAL;
9.
```

The Results tab shows the message "View created." At the bottom, it displays user information and copyright details.

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

**RESULT:**

# PRIMARY KEY, FOREIGN KEY AND CHECK CONSTRAINTS

EXNO:12

DATE:

## 1. What is the purpose of a

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

## 2. Using the column information for the animals table below, name constraints where applicable at the table level, otherwise name them at the column level. Define the primary key (animal\_id). The license\_tag\_number must be unique. The admit\_date and vaccination\_date columns cannot contain null values.

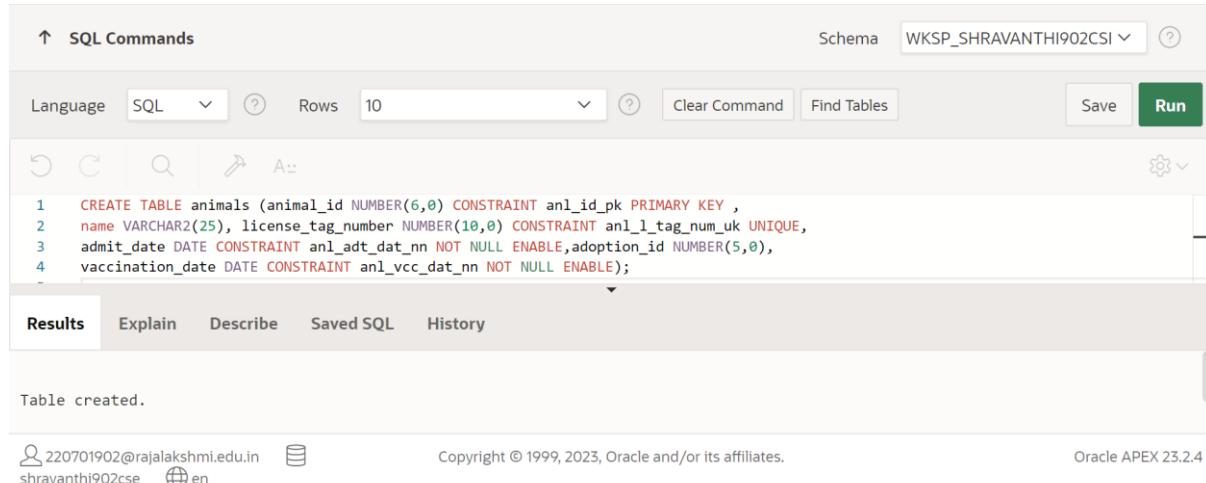
animal\_id NUMBER(6) - PRIMARY KEY  
name VARCHAR2(25)  
license\_tag\_number NUMBER(10)- UNIQUE  
admit\_date DATE- NOT NULL  
adoption\_id NUMBER(5),  
vaccination\_date DATE- NOT NULL

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

### QUERY:

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

### OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'SQL Commands', 'Schema' set to 'WKSP\_SHRAVANTHI902CSI', and a 'Run' button. Below the toolbar are buttons for Undo, Redo, Find, and Save. The main area displays the SQL command for creating the 'animals' table. The command is as follows:

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

The 'Results' tab at the bottom shows the message 'Table created.'.

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

**QUERY:**

```
INSERT INTO ANIMALS (ANIMAL_ID, NAME, LICENSE_TAG_NUMBER, ADMIT_DATE,
ADOPTION_ID,
VACCINATION_DATE) VALUES (101, 'SPOT', 35540, '10/10/2004',
205, '12/10/2004');
```

**OUTPUT:**

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

```
1 INSERT INTO ANIMALS (ANIMAL_ID, NAME, LICENSE_TAG_NUMBER, ADMIT_DATE, ADOPTION_ID,
2 VACCINATION_DATE) VALUES (101, 'SPOT', 35540, '10/10/2004', 205, '12/10/2004');
```

The results section shows the message "Table created." indicating the success of the insert operation.

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

**Ans: 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
(adooption_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 (adooption_id)
REFERENCES adoptions(id) ON DELETE CASCADE ENABLE ;
```

b. ON DELETE SET NULL

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

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

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

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

**RESULT:**

# CREATING VIEWS

EX.NO.13

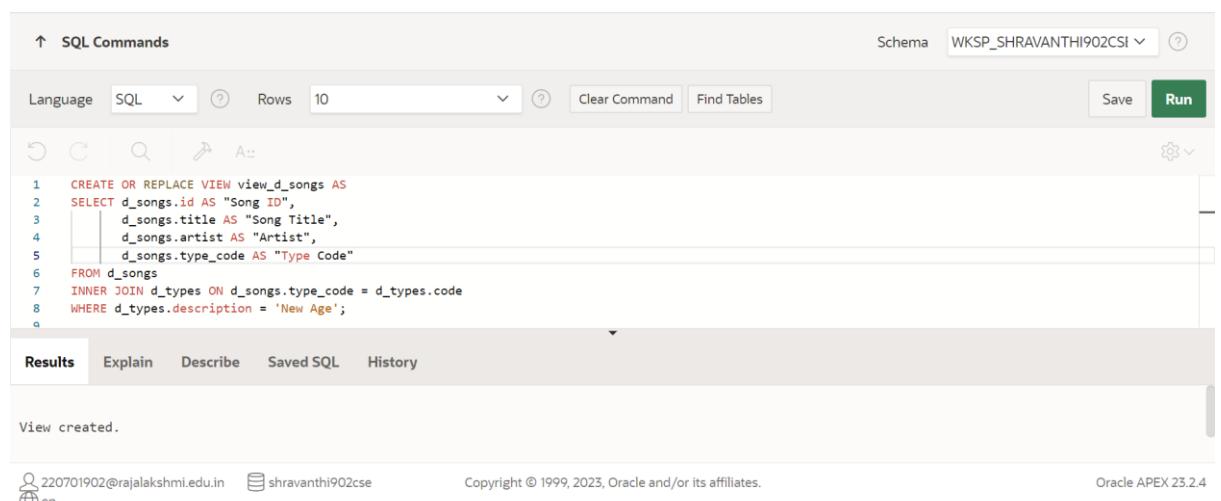
DATE:

- 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.

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

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

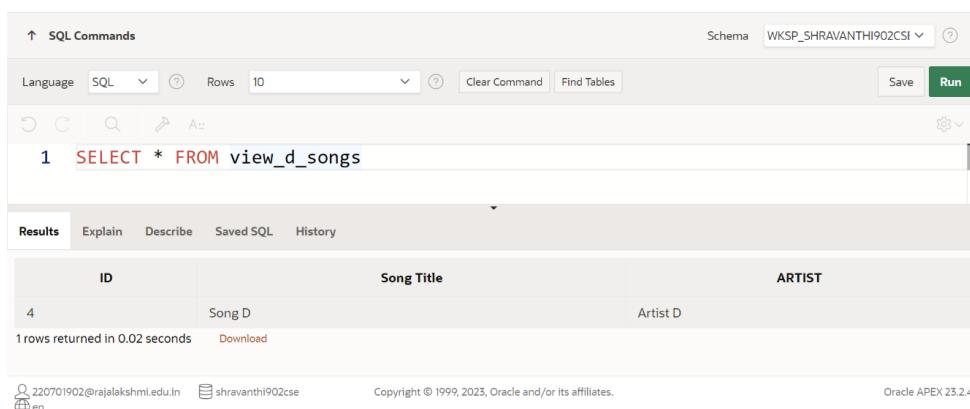


The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The SQL command entered is:

```
1 CREATE OR REPLACE VIEW view_d_songs AS
2 SELECT d_songs.id AS "Song ID",
3        d_songs.title AS "Song Title",
4        d_songs.artist AS "Artist",
5        d_songs.type_code AS "Type Code"
6   FROM d_songs
7  INNER JOIN d_types ON d_songs.type_code = d_types.code
8 WHERE d_types.description = 'New Age';
q
```

The 'Run' button is highlighted in green. Below the command, the message 'View created.' is displayed. The bottom right corner indicates 'Oracle APEX 23.2.4'.

- SELECT \* FROM view\_d\_songs. What was returned?



The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The SQL command entered is:

```
1 SELECT * FROM view_d_songs
```

The 'Run' button is highlighted in green. The results section shows a single row:

| ID | Song Title | ARTIST   |
|----|------------|----------|
| 4  | Song D     | Artist D |

Below the table, it says '1 rows returned in 0.02 seconds' and has a 'Download' link. The bottom right corner indicates 'Oracle APEX 23.2.4'.

4. REPLACE view\_d\_songs. Add type\_code to the column list. Use aliases for all columns.

Or use alias after the CREATE statement as shown.

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



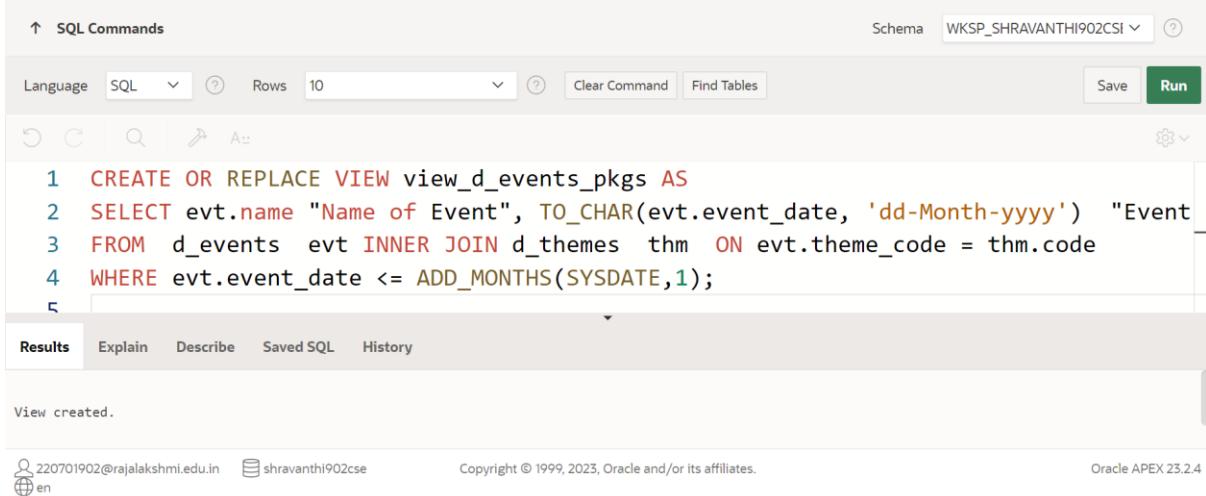
The screenshot shows the Oracle APEX SQL Commands interface. The SQL command is displayed in the editor area:

```
1 CREATE OR REPLACE VIEW view_d_songs AS
2 SELECT d_songs.id, d_songs.title "Song Title", d_songs.artist, d_songs.type_code
3 FROM d_songs INNER JOIN d_types ON d_songs.type_code = d_types.code
4 WHERE d_types.description = 'New Age';
5
```

The interface includes a toolbar with icons for Undo, Redo, Find, Replace, and Save/Run. The Schema dropdown is set to WKSP\_SHRAVANTHI902CSI. The Results tab is selected, showing the message "View created." Below the interface, the footer displays user information and copyright details.

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

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



The screenshot shows the Oracle APEX SQL Commands interface. The SQL command is displayed in the editor area:

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

The interface includes a toolbar with icons for Undo, Redo, Find, Replace, and Save/Run. The Schema dropdown is set to WKSP\_SHRAVANTHI902CSI. The Results tab is selected, showing the message "View created." Below the interface, the footer displays user information and copyright details.

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

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

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'SQL Commands', 'Schema' set to 'WKSP\_SHRAVANTHI902CSI', and a 'Run' button. Below the toolbar are buttons for Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run. The main area contains the SQL code for creating the view, with line numbers 1 through 4. The code is as follows:

```
1 CREATE OR REPLACE VIEW view_min_max_avg_dpt_salary ("Department Id", "Department Name",  
2 SELECT dpt.department_id, dpt.department_name, MAX(NVL(emp.salary,0)), MIN(NVL(emp.  
3 FROM departments dpt LEFT OUTER JOIN employees emp ON dpt.department_id = emp.department_id  
4 GROUP BY (dpt.department_id, dpt.department_name);
```

Below the code, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, displaying the message 'View created.' At the bottom, it shows the user '220701902@rajalakshmi.edu.in' and 'shravanthi902cse', the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.', and the version 'Oracle APEX 23.2.4'.

# DML Operations and Views

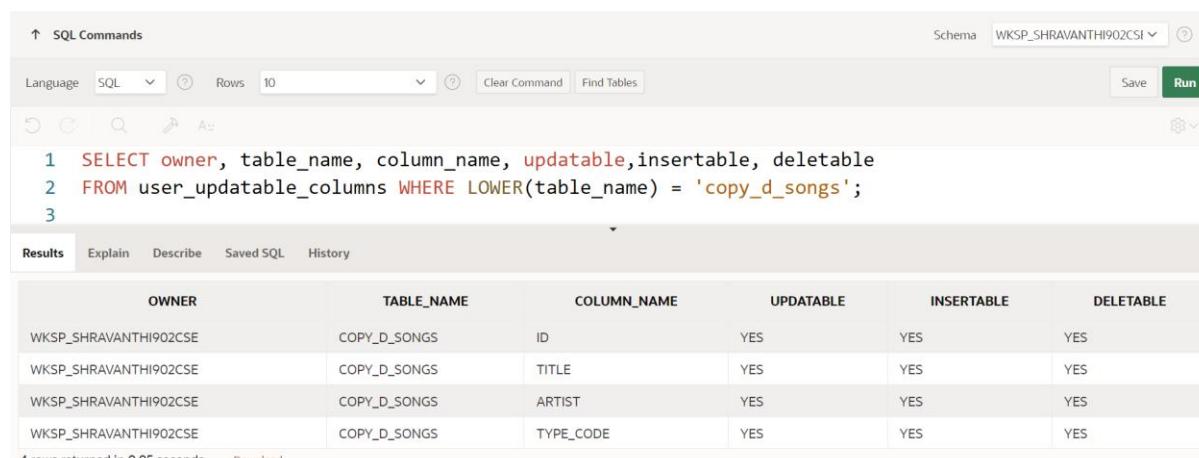
Use the DESCRIBE statement to verify that you have tables named copy\_d\_songs, copy\_d\_events, copy\_d\_cds, and copy\_d\_clients in your schema. If you don't, write a query to create a copy of each.

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

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

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

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



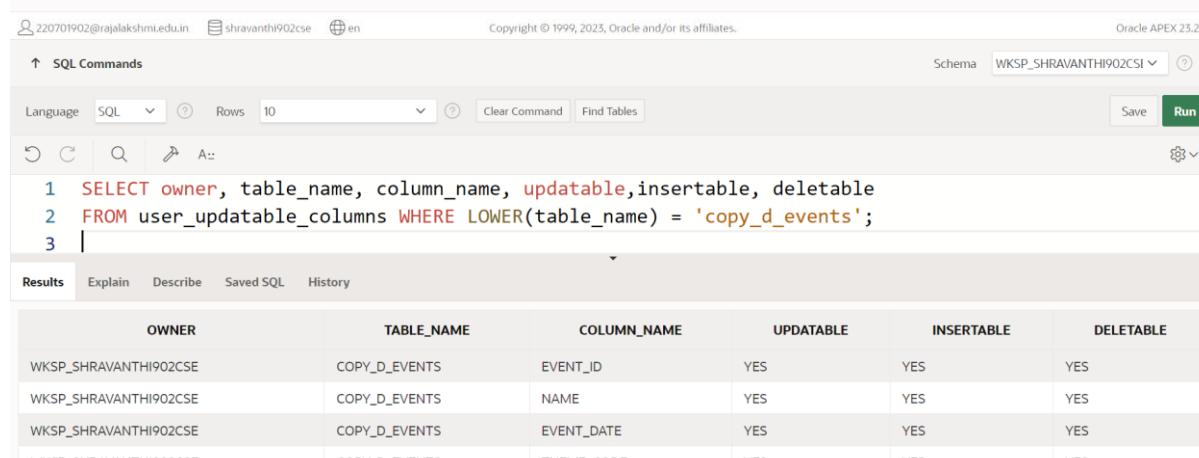
The screenshot shows the Oracle SQL Developer interface with the following details:

- SQL Commands tab is selected.
- Language dropdown is set to SQL.
- Rows dropdown is set to 10.
- Schema dropdown is set to WKSP\_SHRAVANTHI902CSI.
- Run button is visible.
- SQL code entered:

```
1 SELECT owner, table_name, column_name, updatable,insertable, deletable
2 FROM user_updatable_columns WHERE LOWER(table_name) = 'copy_d_songs';
3
```
- Results tab is selected.
- A table is displayed with the following data:

| OWNER                 | TABLE_NAME   | COLUMN_NAME | UPDATABLE | INSERTABLE | DELETABLE |
|-----------------------|--------------|-------------|-----------|------------|-----------|
| WKSP_SHRAVANTHI902CSE | COPY_D_SONGS | ID          | YES       | YES        | YES       |
| WKSP_SHRAVANTHI902CSE | COPY_D_SONGS | TITLE       | YES       | YES        | YES       |
| WKSP_SHRAVANTHI902CSE | COPY_D_SONGS | ARTIST      | YES       | YES        | YES       |
| WKSP_SHRAVANTHI902CSE | COPY_D_SONGS | TYPE_CODE   | YES       | YES        | YES       |

- Message at the bottom: "4 rows returned in 0.05 seconds" and "Download".



The screenshot shows the Oracle SQL Developer interface with the following details:

- SQL Commands tab is selected.
- Language dropdown is set to SQL.
- Rows dropdown is set to 10.
- Schema dropdown is set to WKSP\_SHRAVANTHI902CSI.
- Run button is visible.
- SQL code entered:

```
1 SELECT owner, table_name, column_name, updatable,insertable, deletable
2 FROM user_updatable_columns WHERE LOWER(table_name) = 'copy_d_events';
3
```
- Results tab is selected.
- A table is displayed with the following data:

| OWNER                 | TABLE_NAME    | COLUMN_NAME | UPDATABLE | INSERTABLE | DELETABLE |
|-----------------------|---------------|-------------|-----------|------------|-----------|
| WKSP_SHRAVANTHI902CSE | COPY_D_EVENTS | EVENT_ID    | YES       | YES        | YES       |
| WKSP_SHRAVANTHI902CSE | COPY_D_EVENTS | NAME        | YES       | YES        | YES       |
| WKSP_SHRAVANTHI902CSE | COPY_D_EVENTS | EVENT_DATE  | YES       | YES        | YES       |
| WKSP_SHRAVANTHI902CSE | COPY_D_EVENTS | THEME_CODE  | YES       | YES        | YES       |

- Message at the bottom: "4 rows returned in 0.04 seconds" and "Download".

↑ SQL Commands

Language: SQL Rows: 10 Schema: WKSP\_SHRAVANTHI902CSI

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

Results Explain Describe Saved SQL History

| OWNER                 | TABLE_NAME | COLUMN_NAME | UPDATABLE | INSERTABLE | DELETABLE |
|-----------------------|------------|-------------|-----------|------------|-----------|
| WKSP_SHRAVANTHI902CSE | COPY_D_CDS | CD_NUMBER   | YES       | YES        | YES       |
| WKSP_SHRAVANTHI902CSE | COPY_D_CDS | TITLE       | YES       | YES        | YES       |
| WKSP_SHRAVANTHI902CSE | COPY_D_CDS | ARTIST      | YES       | YES        | YES       |
| WKSP_SHRAVANTHI902CSE | COPY_D_CDS | YEAR        | YES       | YES        | YES       |

4 rows returned in 0.04 seconds Download

220701902@rajalakshmi.edu.in shravanthi902cse Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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;
```

↑ SQL Commands

Language: SQL Rows: 10 Schema: WKSP\_SHRAVANTHI902CSI

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

Results Explain Describe Saved SQL History

View created.

0.02 seconds

220701902@rajalakshmi.edu.in shravanthi902cse Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

↑ SQL Commands

Language: SQL Rows: 10 Schema: WKSP\_SHRAVANTHI902CSI

```
2 SELECT * FROM view_copy_d_songs;
3
4
```

Results Explain Describe Saved SQL History

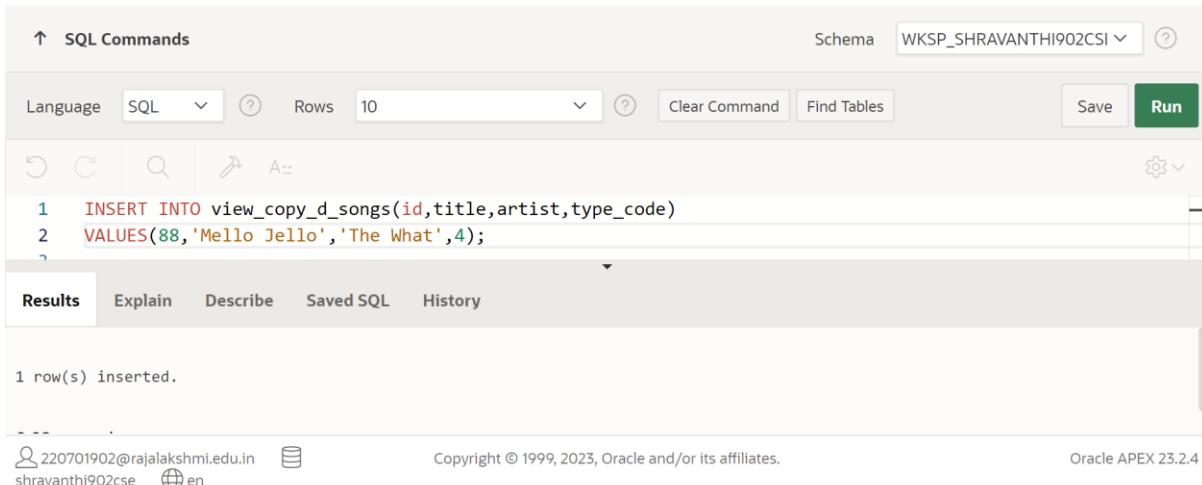
no data found

220701902@rajalakshmi.edu.in shravanthi902cse Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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,artist,type_code)
VALUES(88,'Mello Jello','2 min','The What',4);
```



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

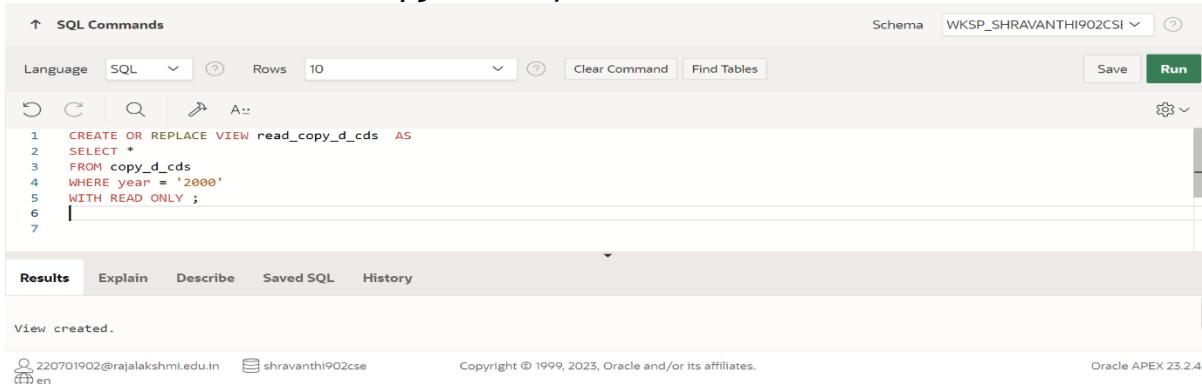
```
1 INSERT INTO view_copy_d_songs(id,title,artist,type_code)
2 VALUES(88,'Mello Jello','The What',4);
3
```

The results section shows the message: "1 row(s) inserted."

4. Create a view based on the DJs on Demand COPY\_D\_CDS table. Name the view read\_copy\_d\_cds. Select all columns to be included in the view. Add a WHERE clause to restrict the year to 2000. Add the WITH READ ONLY option.

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

```
SELECT * FROM read_copy_d_cds;
```



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

```
1 CREATE OR REPLACE VIEW read_copy_d_cds AS
2 SELECT *
3 FROM copy_d_cds
4 WHERE year = '2000'
5 WITH READ ONLY ;
6
7
```

The results section shows the message: "View created."

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The SQL command window contains the following code:

```
1  SELECT * FROM read_copy_d_cds;
2
```

The results tab displays the following data:

| CD_NUMBER | TITLE | ARTIST   | YEAR |
|-----------|-------|----------|------|
| 1         | CD A  | Artist A | 2000 |
| 3         | CD C  | Artist C | 2000 |

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

At the bottom, it shows the user information: 220701902@rajalakshmi.edu.in and shrawanthi902cse, along with the copyright notice: Copyright © 1999, 2023, Oracle and/or its affiliates. and the version: Oracle APEX 23.2.4.

5. Using the read\_copy\_d\_cds view, execute a DELETE FROM read\_copy\_d\_cds WHERE cd\_number = 90;

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

6. Use REPLACE to modify read\_copy\_d\_cds. Replace the READ ONLY option with WITH CHECK OPTION CONSTRAINT ck\_read\_copy\_d\_cds. Execute a SELECT \* statement to verify that the view exists.

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

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The SQL command window contains the following code:

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

The results tab displays the message: View created.

0.02 seconds

At the bottom, it shows the user information: 220701902@rajalakshmi.edu.in and shrawanthi902cse, along with the copyright notice: Copyright © 1999, 2023, Oracle and/or its affiliates. and the version: Oracle APEX 23.2.4.

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';
```

The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The SQL command entered is: `DELETE FROM read_copy_d_cds WHERE year = '2000';`. The results section shows the message: `2 row(s) deleted.` and a time taken of `0.01 seconds`. The copyright notice at the bottom is: `Copyright © 1999, 2023, Oracle and/or its affiliates.` and the version is `Oracle APEX 23.2.4`.

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;
```

The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The SQL command entered is: `DELETE FROM read_copy_d_cds WHERE cd_number = 90;`. The results section shows the message: `0 row(s) deleted.` and a time taken of `0.01 seconds`. The copyright notice at the bottom is: `Copyright © 1999, 2023, Oracle and/or its affiliates.` and the version is `Oracle APEX 23.2.4`.

9. Use the read\_copy\_d\_cds view to delete year 2001 records.

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

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

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

The results section shows:

0 row(s) deleted.

0.01 seconds

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

Oracle APEX 23.2.4

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;
```

The screenshot shows the Oracle APEX SQL Workshop interface. At the top, there are two tabs: 'SQL Commands' and 'Results'. The 'SQL Commands' tab contains the SQL code for creating the view and executing a select statement. The 'Results' tab displays the output of the query, showing one row with 'Mello Jello' in the 'TITLE' column and 'The What' in the 'ARTIST' column. The bottom of the screen shows user information and copyright details.

| TITLE       | ARTIST   |
|-------------|----------|
| Mello Jello | The What |

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

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;
```

The screenshot shows the Oracle APEX SQL Workshop interface. The 'SQL Commands' tab contains the SQL code for dropping the view. The 'Results' tab displays the message 'View dropped.' and the time '0.03 seconds'. The bottom of the screen shows user information and copyright details.

↑ SQL Commands

Schema: WKSP\_SHRAVANTHI902CSI

Language: SQL Rows: 10 Save Run

SELECT \* FROM view\_copy\_d\_songs;

Results Explain Describe Saved SQL History

X Error at line 1/15: ORA-00942: table or view does not exist

220701902@rajalakshmi.edu.in shrawanthi902cse Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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 last_name, salary
FROM (
    SELECT last_name, salary,
           DENSE_RANK() OVER (ORDER BY salary DESC) AS salary_rank
    FROM employees
)
WHERE salary_rank <= 3;
```

↑ SQL Commands

Schema: WKSP\_SHRAVANTHI902CSI

Language: SQL Rows: 10 Save Run

SELECT last\_name, salary
FROM (
 SELECT last\_name, salary,
 DENSE\_RANK() OVER (ORDER BY salary DESC) AS salary\_rank
 FROM employees
)
WHERE salary\_rank <= 3;

Results Explain Describe Saved SQL History

| LAST_NAME | SALARY |
|-----------|--------|
| Brown     | 6000   |
| Johnson   | 5500   |
| Matos     | 4500   |

3 rows returned in 0.01 seconds Download

220701902@rajalakshmi.edu.in shrawanthi902cse Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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 last_name, salary, department_id, max_salary
FROM (
    SELECT last_name, salary, department_id,
           MAX(salary) OVER (PARTITION BY department_id) AS max_salary
      FROM employees
);

```

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the query provided above. The results window displays the following data:

| LAST_NAME | SALARY | DEPARTMENT_ID | MAX_SALARY |
|-----------|--------|---------------|------------|
| Doe       | 2000   | 10            | 2000       |
| Smith     | 3500   | 20            | 3500       |
| Brown     | 6000   | 50            | 6000       |
| Matos     | 4500   | 50            | 6000       |
| Johnson   | 5500   | 80            | 5500       |

5 rows returned in 0.01 seconds.

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

```
SELECT last_name, salary
FROM employees
ORDER BY salary ASC;
```

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the query provided above. The results window displays the following data:

| LAST_NAME | SALARY |
|-----------|--------|
| Doe       | 2000   |
| Smith     | 3500   |
| Matos     | 4500   |
| Johnson   | 5500   |
| Brown     | 6000   |

5 rows returned in 0.01 seconds.

# INDEXES AND SYNONYMS

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

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

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

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

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

3. When will an index be created automatically?

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

4. Create a nonunique index (foreign key) for the DJs on Demand column (cd\_number) in the D\_TRACK\_LISTINGS table. Use the Oracle Application Express SQL Workshop Data Browser to confirm that the index was created.

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

The screenshot shows the Oracle Application Express SQL Workshop Data Browser interface. The top navigation bar includes 'SQL Commands', 'Schema' set to 'WKSP\_SHRAVANTHI902CSI', and a 'Run' button. Below the toolbar, there are buttons for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. The main query editor window contains the SQL command: '1 CREATE INDEX d\_tlg\_cd\_number\_fk\_i on d\_track\_listings (cd\_number);'. The results tab shows the output: 'Index created.' and '0.03 seconds'. At the bottom, footer information includes email addresses (220701902@rajalakshmi.edu.in, shrawanthi902cse), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and software version (Oracle APEX 23.2.4).

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';
```

The screenshot shows the Oracle APEX SQL Commands interface. The query is executed successfully, returning one row of data:

| INDEX_NAME       | COLUMN_NAME | COLUMN_POSITION | UNIQUENESS |
|------------------|-------------|-----------------|------------|
| SYS_C00160023971 | ID          | 1               | UNIQUE     |

Execution details: 1 rows returned in 0.18 seconds. Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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';
```

The screenshot shows the Oracle APEX SQL Commands interface. The query is executed successfully, returning one row of data:

| INDEX_NAME       | TABLE_NAME | UNIQUENESS |
|------------------|------------|------------|
| SYS_C00160023988 | D_EVENTS   | UNIQUE     |

Execution details: 1 rows returned in 0.07 seconds. Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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;
```

The screenshot shows the Oracle APEX SQL Commands interface. The command `CREATE SYNONYM dj_tracks FOR d_track_listings;` is entered in the SQL pane. The results pane shows the message "Synonym created." and a execution time of "0.02 seconds". The schema is set to "WKSP\_SHRAVANTHI902CSI".

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

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

The screenshot shows the Oracle APEX SQL Commands interface. The command `CREATE INDEX d_ptr_last_name_idx ON d_partners(LOWER(last_name));` is entered in the SQL pane. The results pane shows the message "Index created." and a execution time of "0.04 seconds". The schema is set to "WKSP\_SHRAVANTHI902CSI".

9. Create a synonym for the D\_TRACK\_LISTINGS table. Confirm that it has been created by querying the data dictionary.

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

The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The command entered is 'CREATE SYNONYM dj\_tracks2 FOR d\_track\_listings;'. The results pane shows the message 'Synonym created.'.

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

The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The command entered is 'SELECT \* FROM user\_synonyms WHERE table\_NAME = UPPER('d\_track\_listings');'. The results pane displays two rows of data:

| DJ_TRACKS  | WKSP_SHRAVANTHI902CSE | D_TRACK_LISTINGS | - | 0 |
|------------|-----------------------|------------------|---|---|
| DJ_TRACKS2 | WKSP_SHRAVANTHI902CSE | D_TRACK_LISTINGS | - | 0 |

2 rows returned in 0.03 seconds

10. Drop the synonym that you created in question :DROP SYNONYM dj\_tracks2;

The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The command entered is 'DROP SYNONYM dj\_tracks2;'. The results pane shows the message 'Synonym dropped.'

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

**RESULT:**

# OTHER DATABASE OBJECTS

EX.NO:14

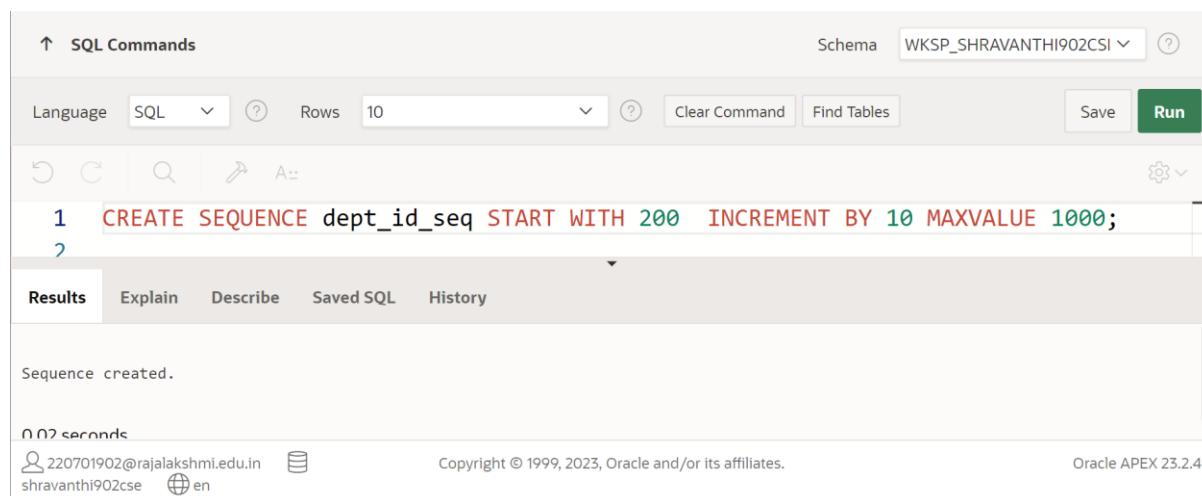
DATE:

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

QUERY:

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

OUTPUT:



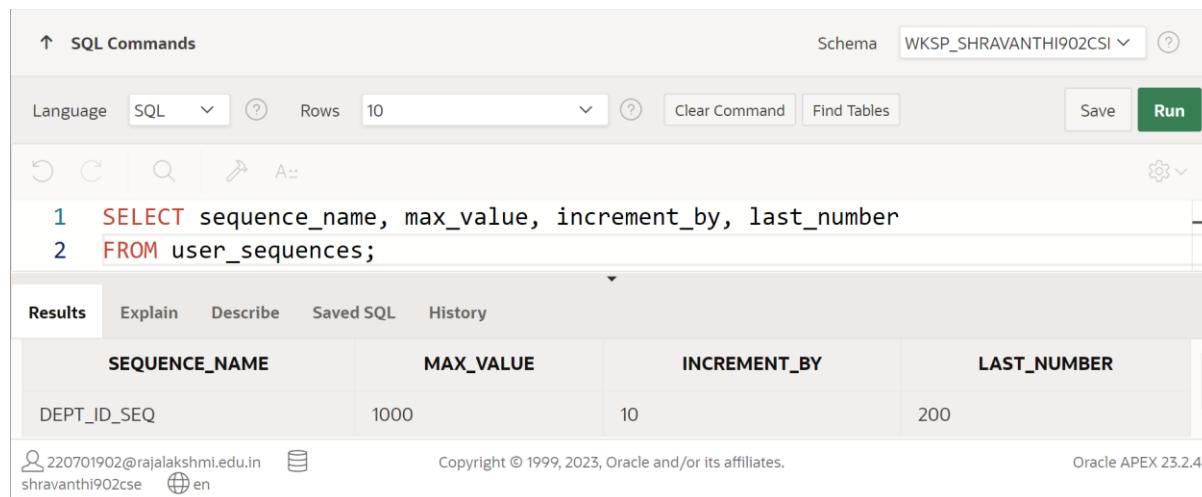
The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The SQL command entered is: `CREATE SEQUENCE dept_id_seq START WITH 200 INCREMENT BY 10 MAXVALUE 1000;`. The 'Run' button is highlighted in green. Below the command, the output shows: 'Sequence created.' and '0.02 seconds'. The copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4' are visible at the bottom.

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The SQL command entered is: `SELECT sequence_name, max_value, increment_by, last_number FROM user_sequences;`. The 'Run' button is highlighted in green. Below the command, the output shows a table with four columns: SEQUENCE\_NAME, MAX\_VALUE, INCREMENT\_BY, and LAST\_NUMBER. The data row is: DEPT\_ID\_SEQ, 1000, 10, 200. The copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4' are visible at the bottom.

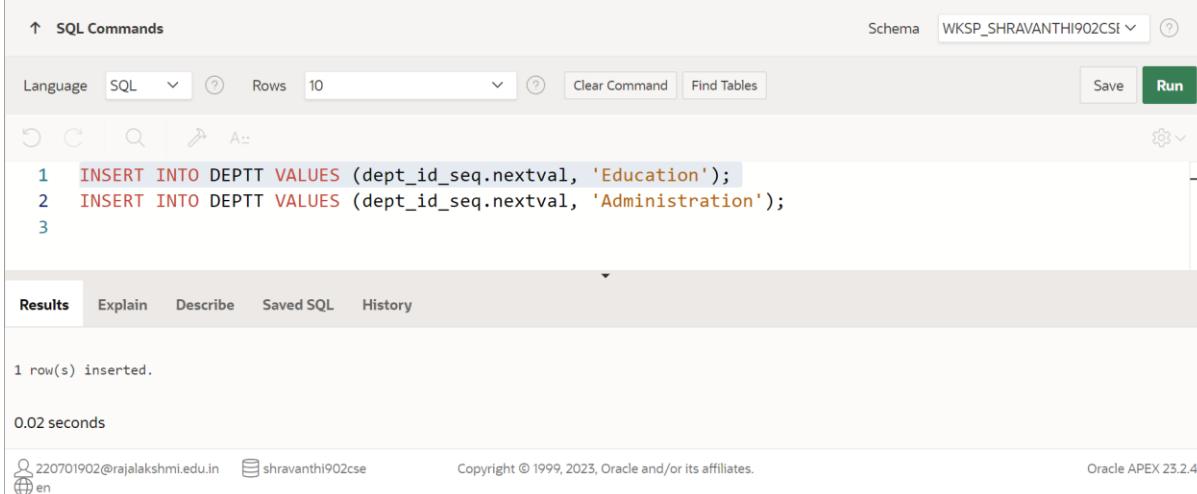
| SEQUENCE_NAME | MAX_VALUE | INCREMENT_BY | LAST_NUMBER |
|---------------|-----------|--------------|-------------|
| DEPT_ID_SEQ   | 1000      | 10           | 200         |

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:**



The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to WKSP\_SHRAVANTHI902CSI. The query entered is:

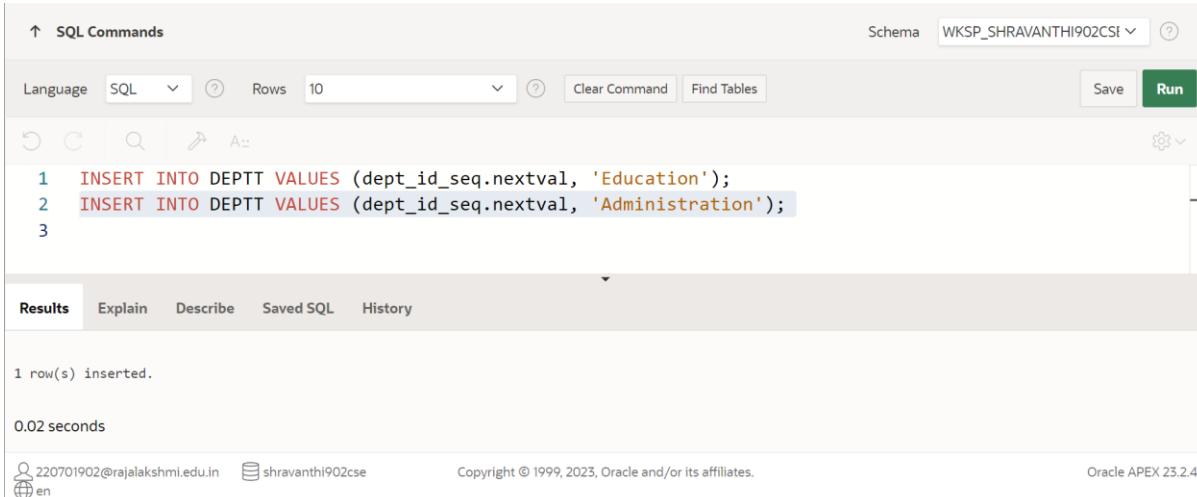
```
1 INSERT INTO DEPTT VALUES (dept_id_seq.nextval, 'Education');
2 INSERT INTO DEPTT VALUES (dept_id_seq.nextval, 'Administration');
3
```

The results section shows the output:

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

Execution time: 0.02 seconds

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



The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to WKSP\_SHRAVANTHI902CSI. The query entered is:

```
1 INSERT INTO DEPTT VALUES (dept_id_seq.nextval, 'Education');
2 INSERT INTO DEPTT VALUES (dept_id_seq.nextval, 'Administration');
3
```

The results section shows the output:

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

Execution time: 0.02 seconds

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

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:**

The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. A SQL command is being run to create an index named 'emp\_dept\_id\_idx' on the 'EMPLOYEES' table, specifically on the 'department\_id' column. The results show that the index was created successfully in 0.03 seconds. The interface includes standard navigation buttons like back, forward, and search, along with tabs for Results, Explain, Describe, Saved SQL, and History.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface displaying the results of a query on the 'user\_indexes' data dictionary view. The query selects index names, table names, and uniqueness constraints for the 'EMPLOYEES' table. Two rows are returned: one for the 'EMP\_DEPT\_ID\_IDX' index which is non-unique, and another for the 'SYS\_C00160014856' index which is unique. The interface includes standard navigation buttons and a results table with columns for INDEX\_NAME, TABLE\_NAME, and UNIQUENESS.

| INDEX_NAME       | TABLE_NAME | UNIQUENESS |
|------------------|------------|------------|
| EMP_DEPT_ID_IDX  | EMPLOYEES  | NONUNIQUE  |
| SYS_C00160014856 | EMPLOYEES  | UNIQUE     |

| 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</u><br><u>(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 APEX SQL Commands interface. The SQL command window contains the provided PL/SQL code. The results tab shows the output: 'Incentive = 660' and 'Statement processed.' The bottom status bar indicates the session user is 'shrawanthi902cse' and the version is 'Oracle APEX 23.2.4'.

```
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;
```

Incentive = 660  
Statement processed.

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

#### QUERY:

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

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

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, a PL/SQL block is entered. The code attempts to declare a variable named 'WELCOME' (without quotes) and use it in a DBMS\_OUTPUT.PUT\_LINE statement. This results in an ORA-06550 error because the identifier is not declared. The error message is displayed in a yellow box in the Results tab.

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

Error at line 4/23: ORA-06550: line 4, column 23:  
PLS-00201: identifier 'Welcome' must be declared  
ORA-06512: at "SYS.IMG\_DBMS\_SQL\_APEX\_230200", line 801  
ORA-06550: line 4, column 1:  
PL/SQL: Statement ignored

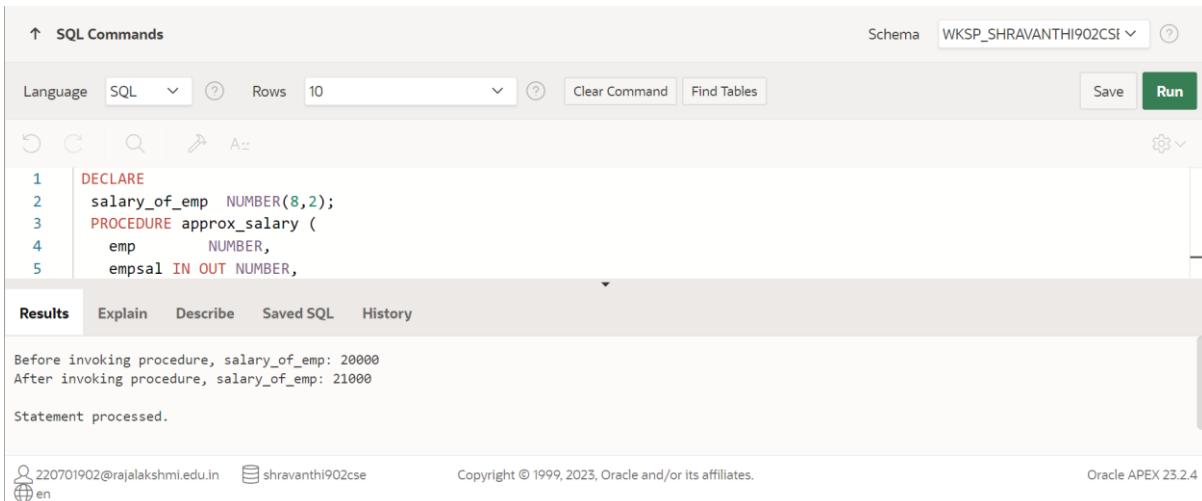
```
2. WELCOME varchar2(10) := 'welcome';
```

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

**QUERY:**

```
DECLARE
    salary_of_emp  NUMBER(8,2);
    PROCEDURE approx_salary (
        emp          NUMBER,
        empsal IN OUT NUMBER,
        adddress     NUMBER
    ) IS
    BEGIN
        empsal := empsal + adddress;
    END;
/
BEGIN
    SELECT salary INTO salary_of_emp
    FROM employees
    WHERE employee_id = 122;
    DBMS_OUTPUT.PUT_LINE
        ('Before invoking procedure, salary_of_emp: ' || salary_of_emp);
    approx_salary (100, salary_of_emp, 1000);
    DBMS_OUTPUT.PUT_LINE
        ('After invoking procedure, salary_of_emp: ' || salary_of_emp);
END;
/
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Commands interface. The code area contains the PL/SQL block provided above. The results pane shows the output of the DBMS\_OUTPUT.PUT\_LINE statements. The bottom status bar indicates the statement was processed successfully.

```
SQL Commands
Schema: WKSP_SHRAVANTHI902CSI
Language: SQL
Rows: 10
Clear Command Find Tables Save Run

1 DECLARE
2     salary_of_emp  NUMBER(8,2);
3     PROCEDURE approx_salary (
4         emp          NUMBER,
5         empsal IN OUT NUMBER,
```

| Results   | Explain | Describe | Saved SQL | History |
|---|---------|----------|-----------|---------|
| Before invoking procedure, salary_of_emp: 20000<br>After invoking procedure, salary_of_emp: 21000<br><br>Statement processed. |         |          |           |         |

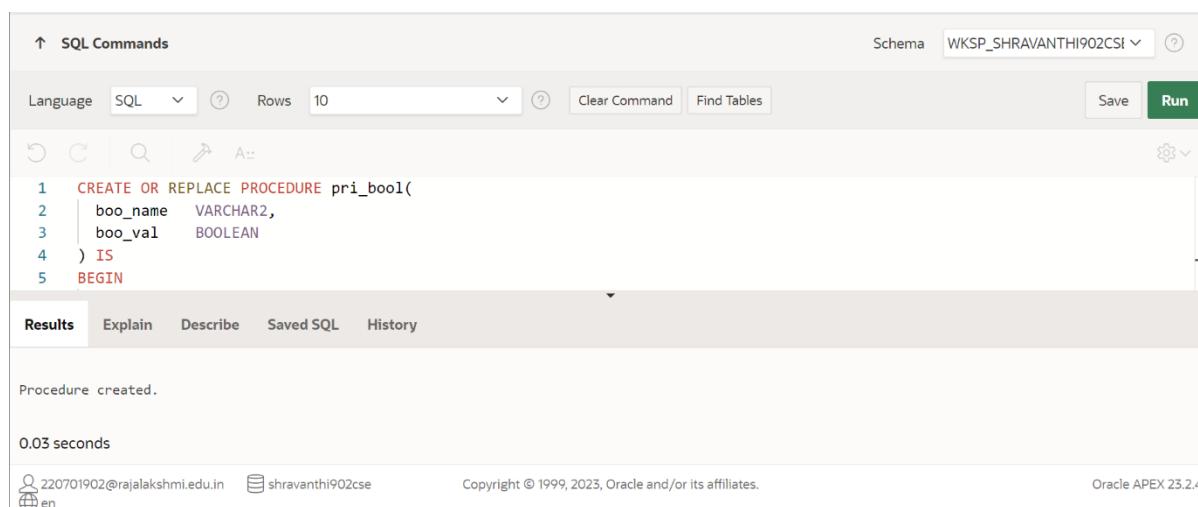
220701902@rajalakshmi.edu.in shravanthi902cse Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Commands interface. The schema is set to 'WKSP\_SHRAVANTHI902CSI'. The code entered is the PL/SQL procedure 'pri\_bool' with its logic for handling NULL, TRUE, and FALSE values. After running the command, the results panel displays the message 'Procedure created.' and a execution time of '0.03 seconds'. The bottom of the screen shows copyright information for Oracle and the APEX version 'Oracle APEX 23.2.4'.

```
1 CREATE OR REPLACE PROCEDURE pri_bool(
2     boo_name    VARCHAR2,
3     boo_val     BOOLEAN
4 ) IS
5 BEGIN
```

Procedure created.  
0.03 seconds

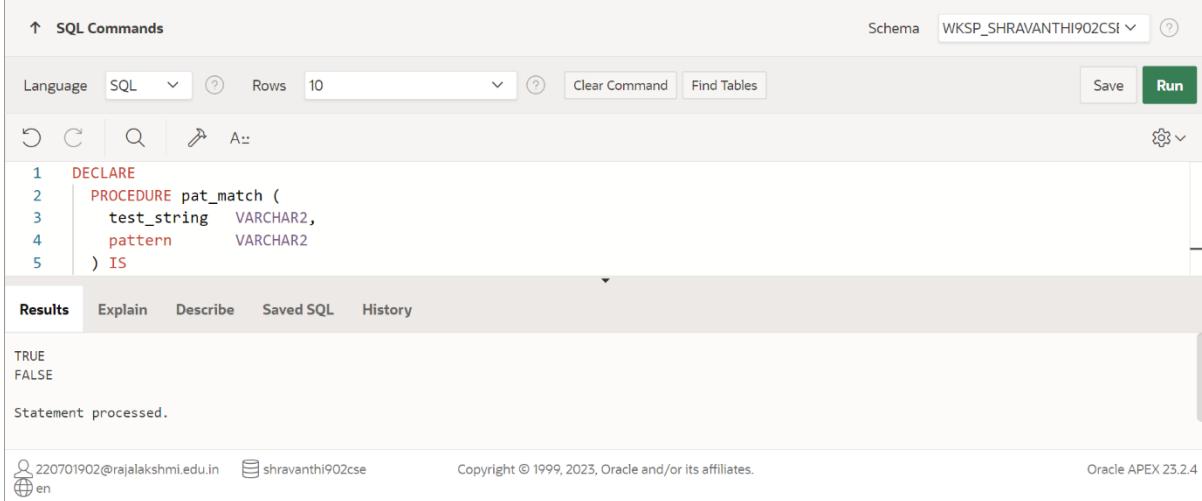
Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

**QUERY:**

```
DECLARE
  PROCEDURE pat_match (
    test_string  VARCHAR2,
    pattern      VARCHAR2
  ) IS
BEGIN
  IF test_string LIKE pattern THEN
    DBMS_OUTPUT.PUT_LINE ('TRUE');
  ELSE
    DBMS_OUTPUT.PUT_LINE ('FALSE');
  END IF;
END;
/
BEGIN
  pat_match('Blweate', 'B%a_e');
  pat_match('Blweate', 'B%A_E');
END;
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Commands interface. The code area contains the provided PL/SQL block. The results section shows the output of the `DBMS\_OUTPUT.PUT\_LINE` statements: "TRUE" and "FALSE". The bottom status bar indicates the session user and Oracle version.

| Results                               | Explain | Describe | Saved SQL | History |
|---------------------------------------|---------|----------|-----------|---------|
| TRUE<br>FALSE<br>Statement processed. |         |          |           |         |

220701902@rajalakshmi.edu.in shrawanthi902cse Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

6.) Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num\_small variable and large number will store in num\_large variable

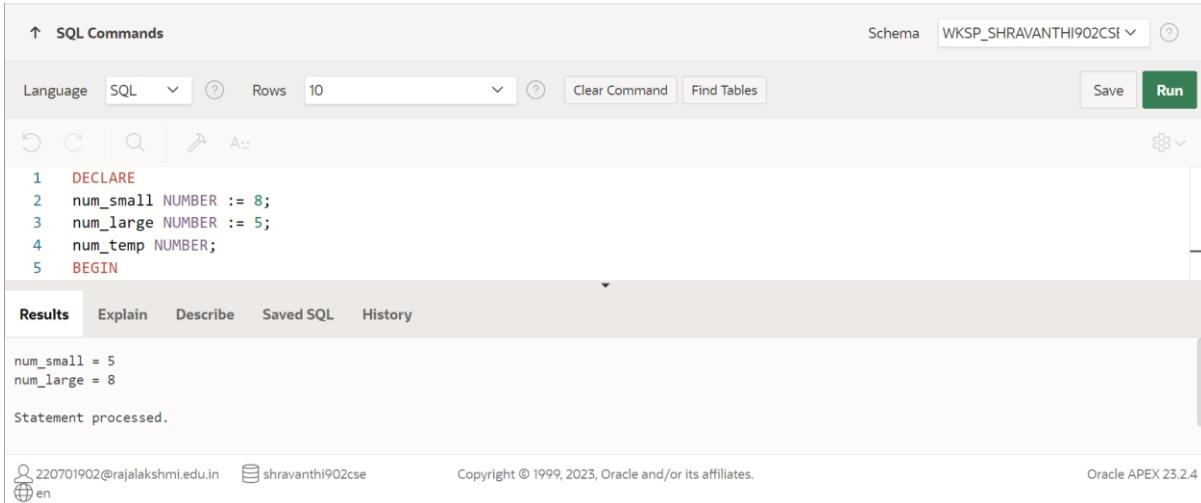
**QUERY:**

```
DECLARE
num_small NUMBER := 8;
num_large NUMBER := 5;
num_temp NUMBER;
BEGIN

IF num_small > num_large THEN
num_temp := num_small;
num_small := num_large;
num_large := num_temp;
END IF;

DBMS_OUTPUT.PUT_LINE ('num_small = '||num_small);
DBMS_OUTPUT.PUT_LINE ('num_large = '||num_large);
END;
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Commands interface. The query window contains the provided PL/SQL code. The results tab shows the output of the DBMS\_OUTPUT.PUT\_LINE statements: "num\_small = 5" and "num\_large = 8". The status bar at the bottom indicates "Statement processed."

```
1  DECLARE
2    num_small NUMBER := 8;
3    num_large NUMBER := 5;
4    num_temp NUMBER;
5    BEGIN

num_small = 5
num_large = 8

Statement processed.
```

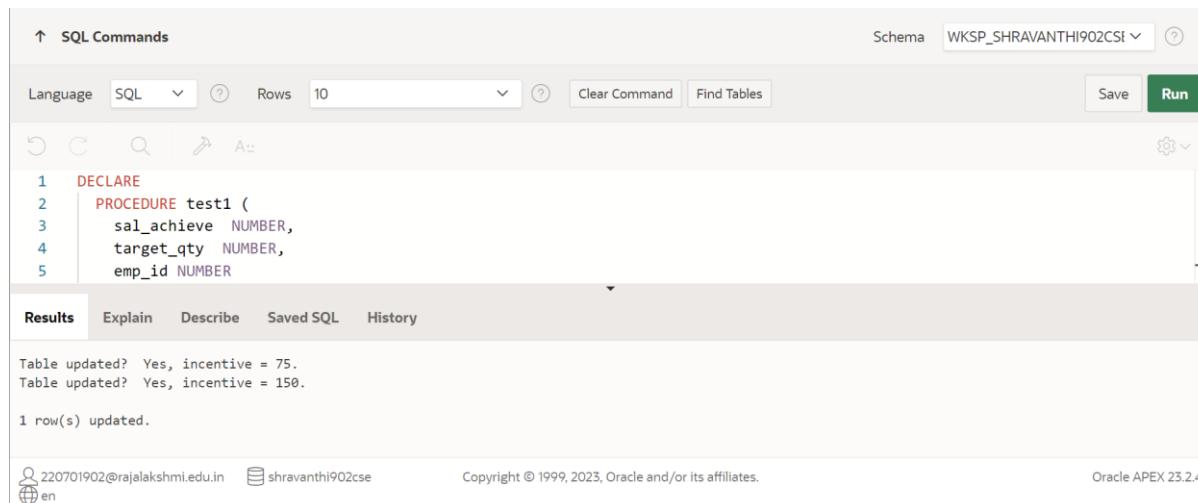
7.) Write a PL/SQL procedure to calculate the incentive on a target achieved and display the message either the record updated or not.

#### QUERY:

DECLARE

```
PROCEDURE test1 (
    sal_achieve NUMBER,
    target_qty NUMBER,
    emp_id NUMBER
)
IS
    incentive NUMBER := 0;
    updated VARCHAR2(3) := 'No';
BEGIN
    IF sal_achieve > (target_qty + 200) THEN
        incentive := (sal_achieve - target_qty)/4;
        UPDATE employees
        SET salary = salary + incentive
        WHERE employee_id = emp_id;
        updated := 'Yes';
    END IF;
    DBMS_OUTPUT.PUT_LINE (
        'Table updated? ' || updated || ', '
        'incentive = ' || incentive || '.'
    );
END test1;
BEGIN
    test1(2300, 2000, 144);
    test1(3600, 3000, 145);
END;/
```

#### OUTPUT:



The screenshot shows the Oracle APEX SQL workspace interface. The top navigation bar includes 'SQL Commands', 'Schema' set to 'WKSP\_SHRAVANTHI902CSF', and a 'Run' button. Below the toolbar are standard database navigation icons. The main area contains the PL/SQL code for the 'test1' procedure. After running the code, the results pane displays two rows of output:

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

1 row(s) updated.
```

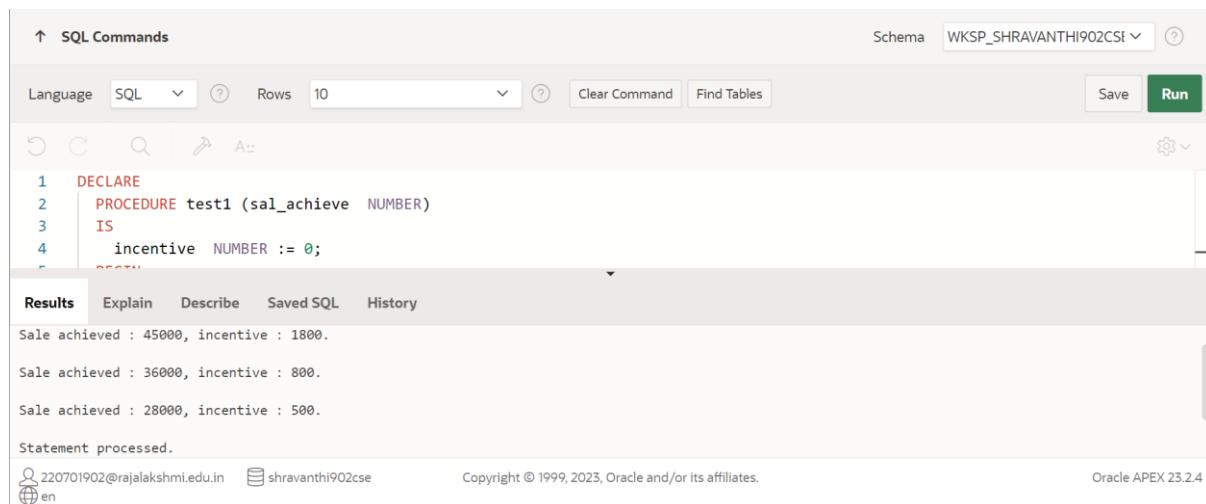
At the bottom of the page, there are footer links for user information (220701902@rajalakshmi.edu.in), the workspace name (shrawanthi902cse), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates), and the software version (Oracle APEX 23.2.4).

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

**QUERY:**

```
DECLARE
  PROCEDURE test1 (sal_achieve  NUMBER)
  IS
    incentive  NUMBER := 0;
  BEGIN
    IF sal_achieve > 44000 THEN
      incentive := 1800;
    ELSIF sal_achieve > 32000 THEN
      incentive := 800;
    ELSE
      incentive := 500;
    END IF;
    DBMS_OUTPUT.NEW_LINE;
    DBMS_OUTPUT.PUT_LINE (
      'Sale achieved : ' || sal_achieve || ', incentive : ' ||
      incentive || '.'
    );
  END test1;
BEGIN
  test1(45000);
  test1(36000);
  test1(28000);
END;/
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Worksheet interface. The code area contains the provided PL/SQL procedure. The results pane displays three rows of output corresponding to the executed test1 procedure calls with arguments 45000, 36000, and 28000.

| Result                                   |
|--|
| Sale achieved : 45000, incentive : 1800. |
| Sale achieved : 36000, incentive : 800.  |
| Sale achieved : 28000, incentive : 500.  |

At the bottom, it shows "Statement processed." and the copyright notice "Copyright © 1999, 2023, Oracle and/or its affiliates."

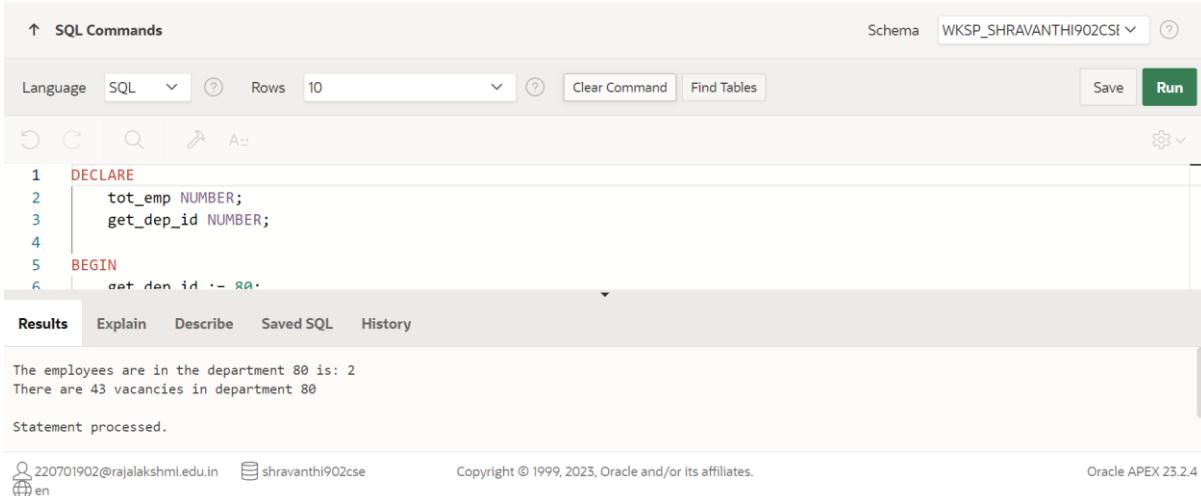
9.) Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

**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 APEX SQL Workshop interface. The top navigation bar includes 'SQL Commands', 'Schema' set to 'WKSP\_SHRAVANTHI902CS1', and a 'Run' button. Below the toolbar, there are icons for undo, redo, search, and other database operations. The main area displays the PL/SQL code in a code editor. The code is as follows:

```
1  DECLARE
2      tot_emp NUMBER;
3      get_dep_id NUMBER;
4
5  BEGIN
6      get_dep_id := 80;
```

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

```
The employees are in the department 80 is: 2
There are 43 vacancies in department 80

Statement processed.
```

At the bottom of the interface, there are footer links for '220701902@rajalakshmi.edu.in', 'shravanthi902cse', 'Copyright © 1999, 2023, Oracle and/or its affiliates.', and 'Oracle APEX 23.2.4'.

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

**QUERY:**

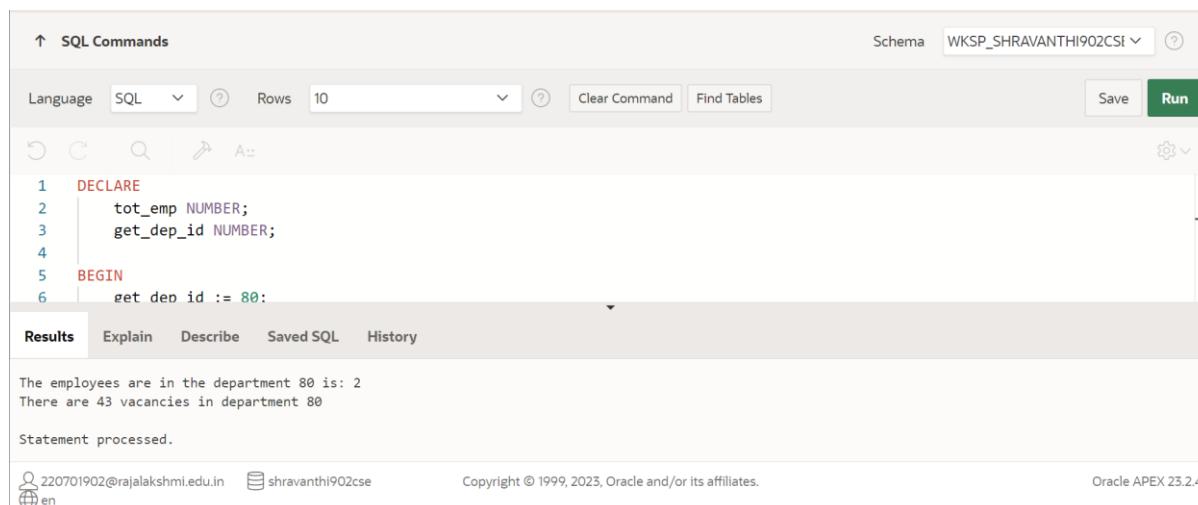
```
DECLARE
    tot_emp NUMBER;
get_dep_id NUMBER;

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

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

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

**OUTPUT:**



The screenshot shows the Oracle APEX SQL workspace interface. The top navigation bar includes 'SQL Commands', 'Schema' set to 'WKSP\_SHRAVANTHI902CSE', and a 'Run' button. Below the toolbar, there are icons for undo, redo, search, and refresh. The main area displays the PL/SQL code and its output. The code declares variables for total employees and department ID, performs a join to count employees in department 80, and then checks if there are any vacancies. The output window shows the results of the query and the final message indicating the number of vacancies.

```
1  DECLARE
2      tot_emp NUMBER;
3      get_dep_id NUMBER;
4
5  BEGIN
6      get dep id := 80;
```

The employees are in the department 80 is: 2  
There are 43 vacancies in department 80  
Statement processed.

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

**QUERY:**

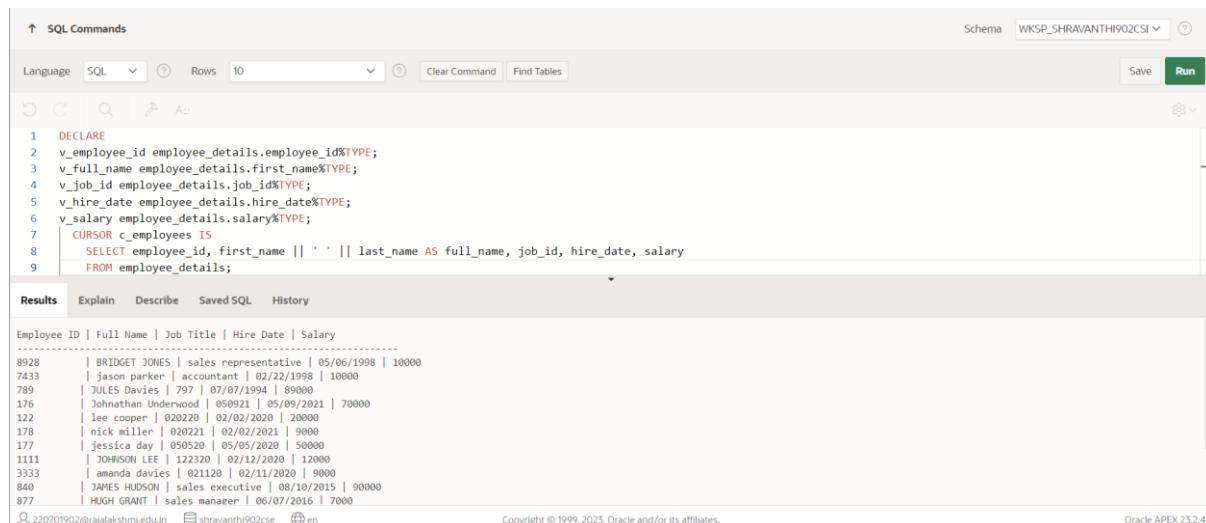
DECLARE

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

CURSOR c_employees IS
    SELECT employee_id, first_name || ' ' || last_name AS full_name,
job_id, hire_date, salary
    FROM employees;

BEGIN
    DBMS_OUTPUT.PUT_LINE('Employee ID | Full Name | Job Title | Hire
Date | Salary');
    DBMS_OUTPUT.PUT_LINE('-----');
    OPEN c_employees;
    FETCH c_employees INTO v_employee_id, v_full_name, v_job_id,
v_hire_date, v_salary;
    WHILE c_employees%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(v_employee_id || ' ' || ' ' || v_full_name
|| ' ' || v_job_id || ' ' || v_hire_date || ' ' || v_salary);
        FETCH c_employees INTO v_employee_id, v_full_name, v_job_id,
v_hire_date, v_salary;
    END LOOP;
    CLOSE c_employees;
END;
/
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'SQL Commands', 'Schema' (set to 'WKSP\_SHRAVANTHI902CSI'), 'Save', and 'Run'. The main area has tabs for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. Below these are icons for 'Copy', 'Search', and 'Help'. The code editor contains the PL/SQL block from above. The 'Results' tab is selected, displaying the output of the query. The output table has columns: Employee ID | Full Name | Job Title | Hire Date | Salary. The data rows are:

| Employee ID | Full Name           | Job Title            | Hire Date  | Salary |
|-------------|---------------------|----------------------|------------|--------|
| 8928        | BRIDGET JONES       | sales representative | 05/06/1998 | 10000  |
| 7433        | jason parker        | accountant           | 02/22/1998 | 10000  |
| 789         | JULES Davies        | 797                  | 07/07/1994 | 89000  |
| 176         | Johnathan Underwood | 050921               | 05/09/2021 | 70000  |
| 122         | lee cooper          | 020228               | 02/02/2020 | 20000  |
| 178         | nick miller         | 020221               | 02/02/2021 | 9000   |
| 177         | jessica day         | 050520               | 05/05/2020 | 50000  |
| 1111        | JOHNSON LEE         | 122320               | 02/12/2020 | 12000  |
| 3333        | amanda davies       | 021120               | 02/11/2020 | 9000   |
| 840         | JAMES HUDSON        | sales executive      | 08/10/2015 | 90000  |
| 877         | HUGH GRANT          | sales manager        | 06/07/2016 | 7000   |

At the bottom, the footer includes the URL '220701902@rajalakshmi.edu.in', the name 'shranthi902cse', and the language 'en'. It also states 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

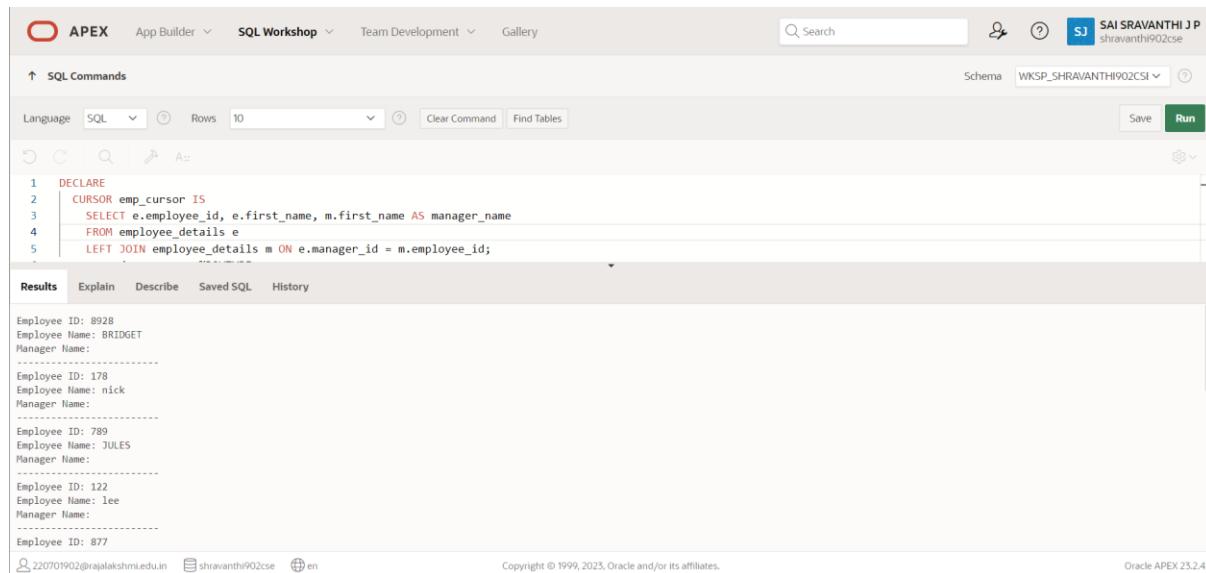
**12.)** Write a PL/SQL program to display the employee IDs, names, and department names of all employees.

**QUERY:**

DECLARE

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

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is signed in as SAI SRAVANTHI J P (shravanthi902cse). The SQL Commands tab is active, displaying the following PL/SQL code:

```
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;
```

The Results tab is selected, showing the output of the executed code:

```
Employee ID: 8928
Employee Name: BRIDGET
Manager Name:
-----
Employee ID: 178
Employee Name: nick
Manager Name:
-----
Employee ID: 789
Employee Name: JULES
Manager Name:
-----
Employee ID: 122
Employee Name: lee
Manager Name:
-----
Employee ID: 877
```

At the bottom of the results pane, there are footer links for Copyright © 1999, 2023, Oracle and/or its affiliates, and Oracle APEX 23.2.4.

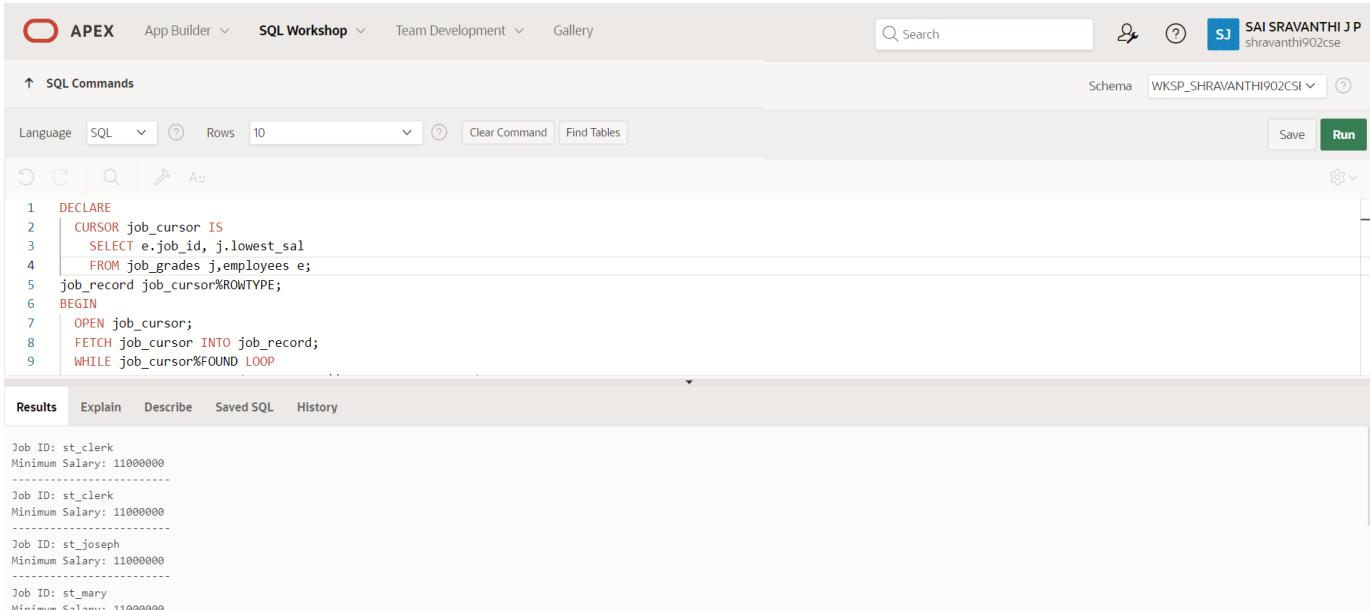
13.) Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all jobs

**QUERY:**

DECLARE

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

**OUTPUT:**



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

```
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
```

In the results pane below, the output is shown as:

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

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

**QUERY:**

DECLARE

```
CURSOR employees_cur IS
```

```
    SELECT employee_id, last_name, job_id, start_date
```

```
    FROM employees NATURAL JOIN job_history;
```

```
    emp_start_date DATE;
```

BEGIN

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

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

```
FOR emp_sal_rec IN employees_cur LOOP
```

```
    -- find out most recent end_date in job_history
```

```
    SELECT Max(end_date) + 1
```

```
    INTO emp_start_date
```

```
    FROM job_history
```

```
    WHERE employee_id = emp_sal_rec.employee_id;
```

```
    IF emp_start_date IS NULL THEN
```

```
        emp_start_date := emp_sal_rec.start_date;
```

```
    END IF;
```

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

```
                    || Rpad(emp_sal_rec.last_name, 25)
```

```
                    || Rpad(emp_sal_rec.job_id, 35)
```

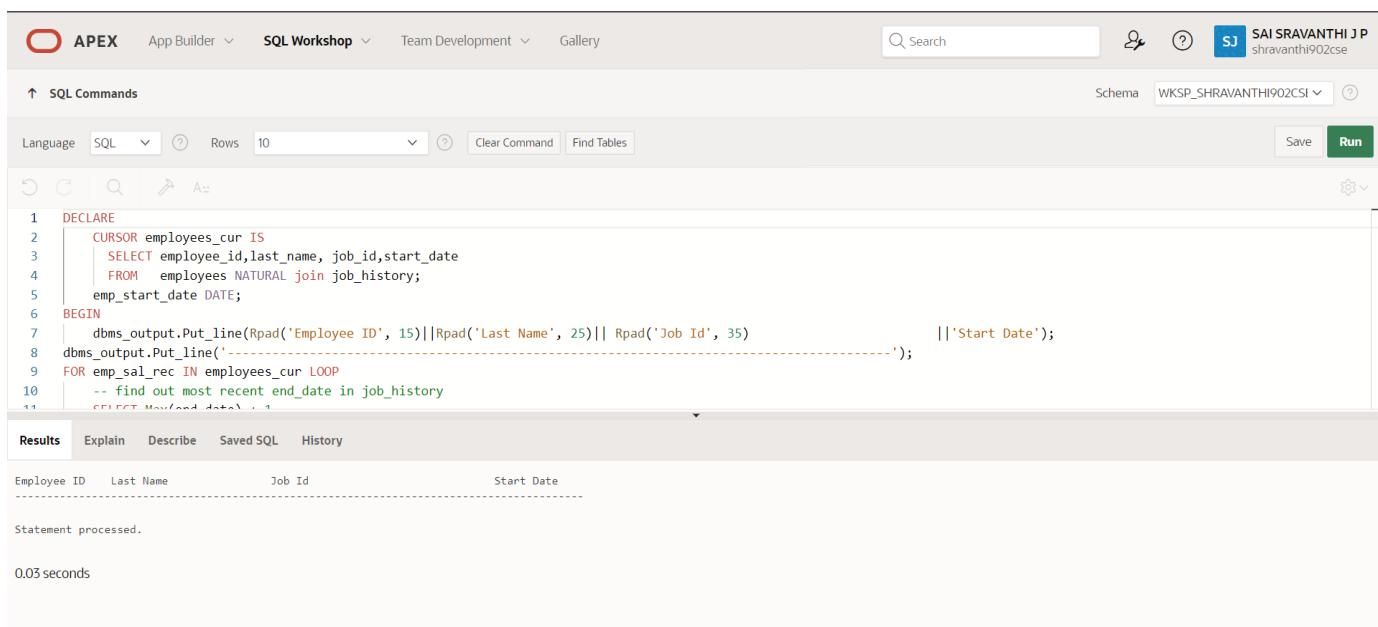
```
                    || To_char(emp_start_date, 'dd-mon-yyyy'));
```

```
END LOOP;
```

```
END;
```

```
/
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is signed in as SAI SRAVANTHI J P (shravanthi902ce). The schema is set to WKSP\_SHRAVANTHI902CSI. The SQL Commands tab is active, displaying the PL/SQL code. The code itself is identical to the one shown in the previous text block. Below the code, the Results tab is selected, showing the output of the program. The output consists of a single row with columns Employee ID, Last Name, Job Id, and Start Date. The values are: Employee ID 101, Last Name King, Job Id 10, and Start Date 01-JAN-1981. Below the results, it says 'Statement processed.' and '0.03 seconds'.

| Employee ID | Last Name | Job Id | Start Date  |
|-------------|-----------|--------|-------------|
| 101         | King      | 10     | 01-JAN-1981 |

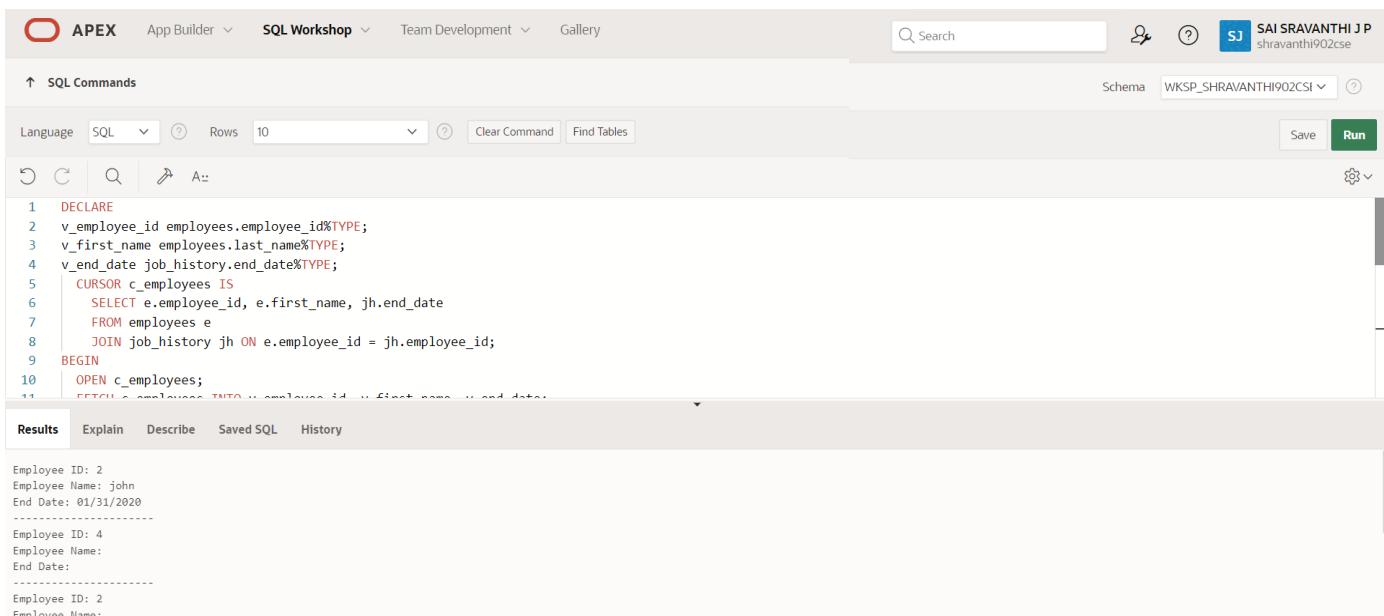
Statement processed.  
0.03 seconds

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a user profile for SAI SRAVANTHI J P (shravanthi902cse) and a schema dropdown for WKSP\_SHRAVANTHI902CSI. The main workspace is titled 'SQL Commands' and contains the following content:

```
1 DECLARE
2 v_employee_id employees.employee_id%TYPE;
3 v_first_name employees.last_name%TYPE;
4 v_end_date job_history.end_date%TYPE;
5 CURSOR c_employees IS
6     SELECT e.employee_id, e.first_name, jh.end_date
7     FROM employees e
8     JOIN job_history jh ON e.employee_id = jh.employee_id;
9 BEGIN
10    OPEN c_employees;
11    FETCH c_employees INTO v_employee_id, v_first_name, v_end_date;
```

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

```
Employee ID: 2
Employee Name: john
End Date: 01/31/2020
-----
Employee ID: 4
Employee Name:
End Date:
-----
Employee ID: 2
Employee Name:
```

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

**RESULT:**

# PROCEDURES AND FUNCTIONS

EX.NO: 17

DATE:

1.) Factorial of a number using function.

QUERY:

DECLARE

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

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The SQL Workshop tab is active, showing the schema as WKSP\_SHRAVANTHI902CSI. The main area displays the PL/SQL code for calculating factorial:

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

The results pane at the bottom shows the output of the query:

```
120
Statement processed.
0.01 seconds
```

2.) Write a PL/SQL program using Procedures IN,IN,OUT,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 (selected), Team Development, and Gallery. A search bar and a user profile for SAI SRAVANTHI J P are also at the top. The main area displays the following SQL code:

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

The "Results" tab is selected, showing the output of the executed code:

```
Title: 1984
Author: George Orwell
Year Published: 1949

Statement processed.
```

| 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. In the top navigation bar, 'APEX' is selected. Below it, the tabs include 'App Builder', 'SQL Workshop' (which is active), 'Team Development', and 'Gallery'. On the right side, the user 'SAI SRAVANTHI J P' is logged in, and the schema 'WKSP\_SRAVANTHI902CSI' is selected. The main workspace displays the SQL command for creating the trigger:

```
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 /
```

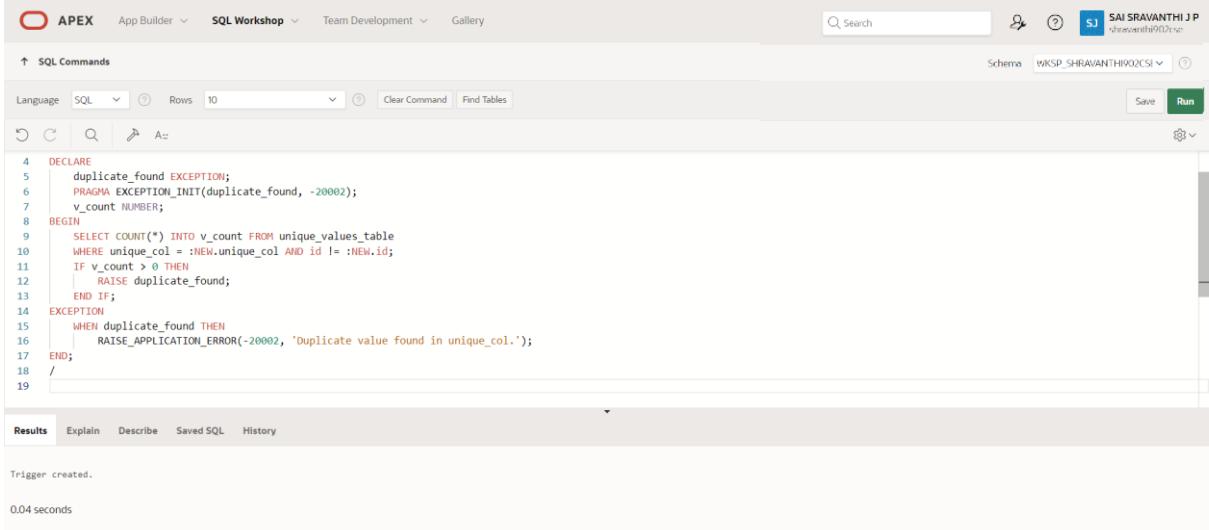
Below the code, the 'Results' tab is selected, showing the output: 'Trigger created.' and '0.04 seconds'. Other tabs available include 'Explain', 'Describe', 'Saved SQL', and 'History'.

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

#### QUERY:

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

#### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery' are visible. On the right, there's a user icon for 'SAI SRAVANTHI J P' and a 'Run' button. The main area is a SQL editor window with the following content:

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

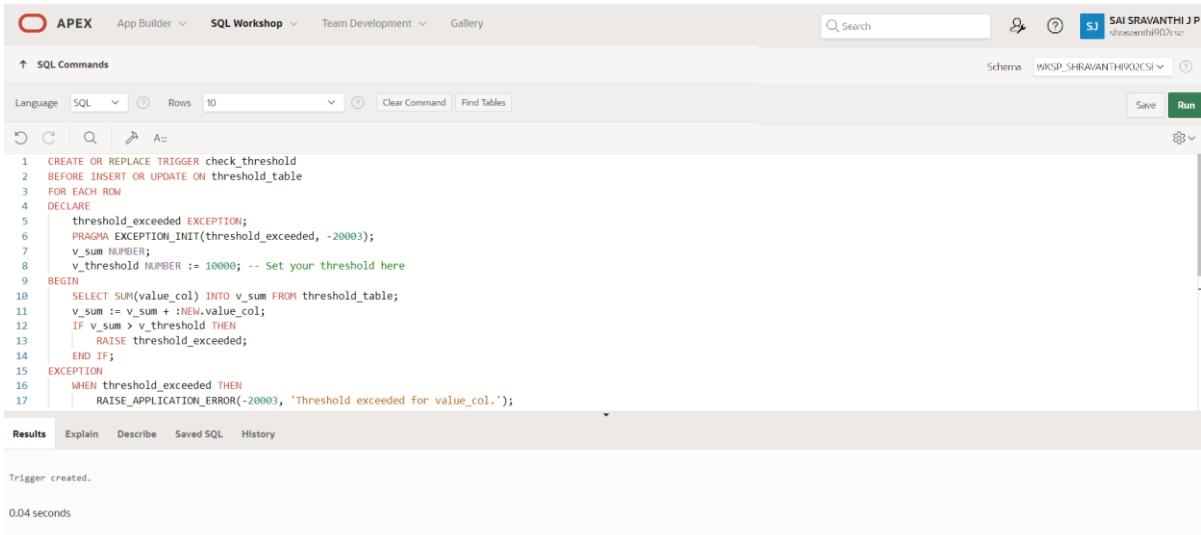
Below the editor, a 'Results' tab is selected, showing the message 'Trigger created.' and a execution time of '0.04 seconds'.

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. The right side shows the user profile 'SAI SRAVANTI J P' and the schema 'WKSP\_SRAVANTI902CSI'. The main area is titled 'SQL Commands' and contains the PL/SQL code for the trigger. The code is highlighted in green. Below the code, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The results section shows the message 'Trigger created.' and a execution time of '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.');

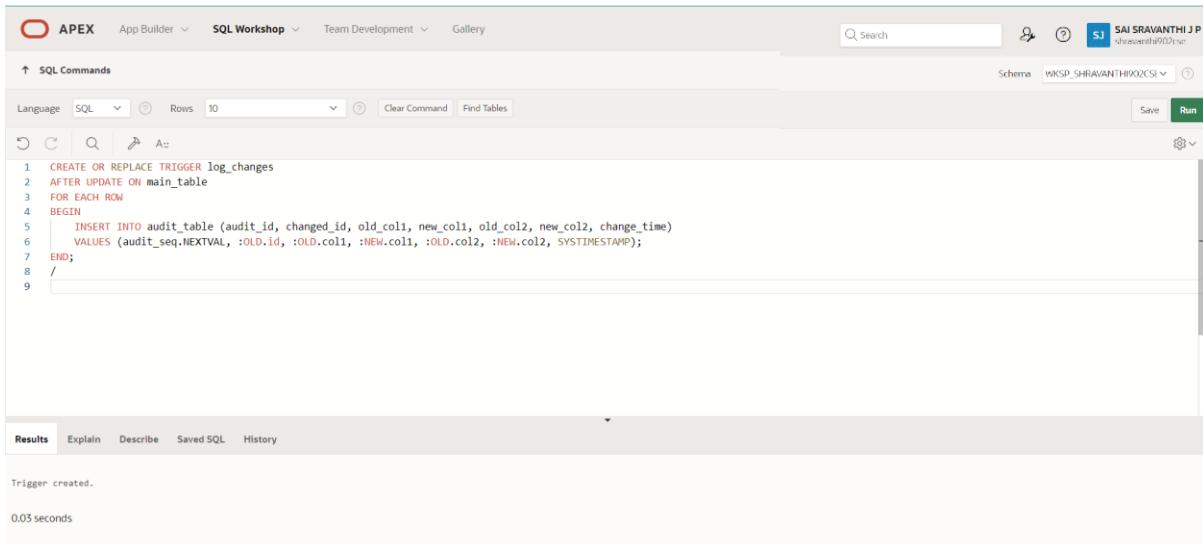
```

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,
6     old_col2, new_col2, change_time)
7     VALUES (audit_seq.NEXTVAL, :OLD.id, :OLD.col1, :NEW.col1,
8     :OLD.col2, :NEW.col2, SYSTIMESTAMP);
9 END;
/
```

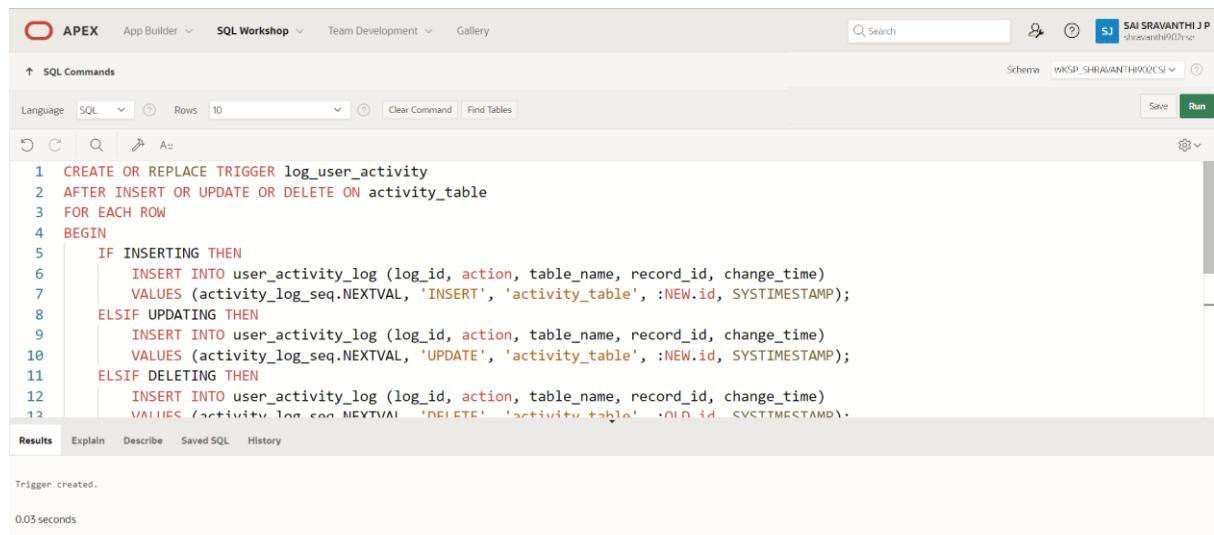
Below the command, the 'Results' tab is active, showing the output: "Trigger created." and "0.03 seconds".

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. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows the user's profile: SAI SRAVANTHI J P, shrawanthip02@se. The main area is titled 'SQL Commands' with tabs for Language (set to SQL), Rows (set to 10), Clear Command, Find Tables, Save, and Run. Below this is a toolbar with icons for Undo, Redo, Search, and others. The SQL editor contains the PL/SQL code for the trigger. The code is numbered from 1 to 12. Lines 1-4 define the trigger, while lines 5-12 handle the three types of database changes (INSERT, UPDATE, DELETE). The code uses bind variables (:NEW.id and :OLD.id) to log the affected row IDs. The output pane at the bottom shows the message 'Trigger created.' and a execution time of '0.03 seconds'.

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

```

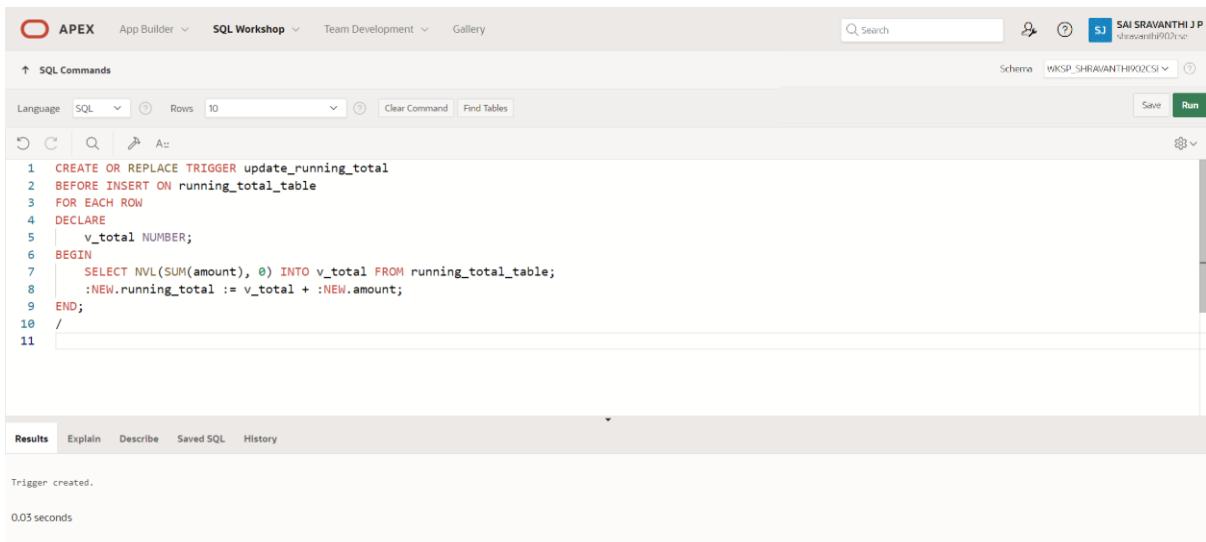
Trigger created.  
0.03 seconds

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:

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

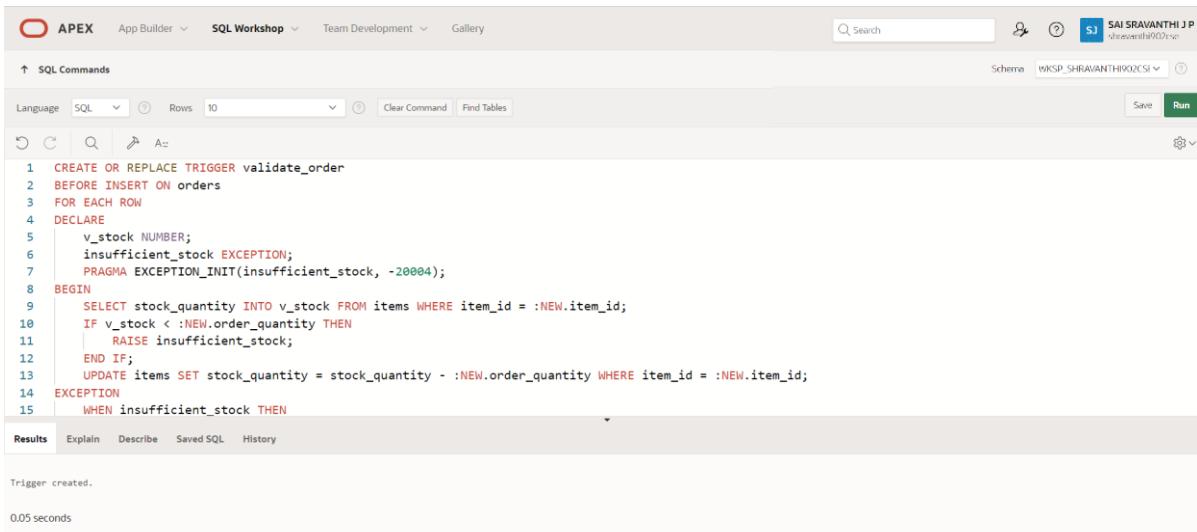
Below the code, the 'Results' tab is active, showing the message "Trigger created." and a execution time of "0.03 seconds".

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

#### QUERY:

```
CREATE OR REPLACE TRIGGER validate_order
BEFORE INSERT ON orders
FOR EACH ROW
DECLARE
    v_stock NUMBER;
    insufficient_stock EXCEPTION;
    PRAGMA EXCEPTION_INIT(insufficient_stock, -20004);
BEGIN
    SELECT stock_quantity INTO v_stock FROM items WHERE item_id =
:NEW.item_id;
    IF v_stock < :NEW.order_quantity THEN
        RAISE insufficient_stock;
    END IF;
    UPDATE items SET stock_quantity = stock_quantity -
:NEW.order_quantity WHERE item_id = :NEW.item_id;
EXCEPTION
    WHEN insufficient_stock THEN
        RAISE_APPLICATION_ERROR(-20004, 'Insufficient stock for the
item.');
END;
```

#### OUTPUT:

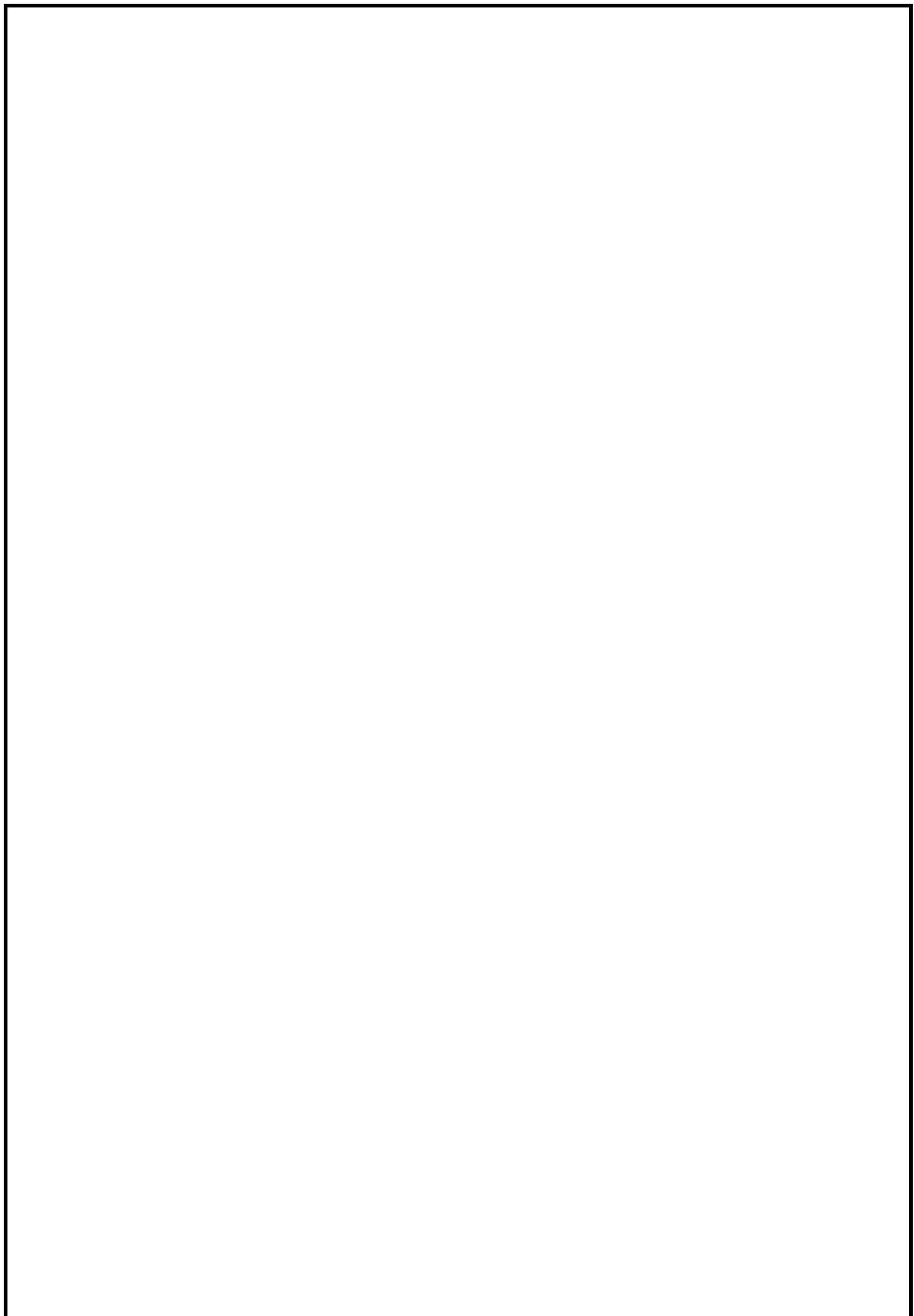


The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, tabs for 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery' are visible. The main area is titled 'SQL Commands'. A toolbar with various icons is at the top, followed by a search bar and a schema dropdown set to 'WKSP\_SHRAVANTHIP902CS1'. The SQL editor contains the PL/SQL code for the 'validate\_order' trigger. The code is highlighted in blue and red, indicating syntax and identifiers. The code itself is identical to the one provided in the question. At the bottom of the SQL editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The status bar at the bottom shows '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
        RAISE_APPLICATION_ERROR(-20004, 'Insufficient stock for the
item.');
```

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

**RESULT:**



# MONGO DB

**EX\_NO: 19**

**DATE:**

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the MongoDB shell interface. At the top, there is a header bar with the user information "220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K". Below the header are two buttons: "MongoDB" with a dropdown arrow and a help icon. To the right are "Run" and "Save" buttons. The main area is divided into two sections: "Output" on the left and "mycompiler\_mongodb> ..." on the right. The "Output" section contains the MongoDB command and its execution results.

```
1 db.restaurants.find(
2   { $or: [
3     { name: /^Wil/ },
4     { cuisine: { $nin: ['American', 'Chinese'] } }
5   ] },
6   { restaurant_id: 1, name: 1, borough: 1, cuisine: 1 }
7 );
8
```

Output

```
mycompiler_mongodb> ... . . .
mycompiler_mongodb>
[Execution complete with exit code 0]
```

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:**

The screenshot shows the MongoDB shell interface. At the top, there is a header bar with the user information "220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K". Below the header are two buttons: "MongoDB" with a dropdown arrow and a help icon. To the right are "Run" and "Save" buttons. The main area is divided into two sections: "Output" on the left and "mycompiler\_mongodb> ..." on the right. The "Output" section contains the MongoDB command and its execution results.

```
1 db.restaurants.find(
2   { grades: { $elemMatch: { grade: "A", score: 11, date: I
3   { restaurant_id: 1, name: 1, grades: 1 }
4 });
5
```

Output

```
mycompiler_mongodb> ... . . .
mycompiler_mongodb>
[Execution complete with exit code 0]
```

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:**

The screenshot shows the MongoDB shell interface. At the top, it displays the user's details: 220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K. Below this is a toolbar with a MongoDB icon, dropdown menus, and buttons for Run and Save. The main area is divided into two sections: 'Query' and 'Output'. In the 'Query' section, the command is pasted. In the 'Output' section, the response from the MongoDB server is shown, starting with 'mycompiler\_mongodb> ...' followed by '[Execution complete with exit code 0]'. The interface has a light gray background with dark blue header and footer bars.

```
220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K
```

MongoDB ▾ ⓘ

Run Save

```
1 db.restaurants.find(
2   { "grades.1.grade": "A", "grades.1.score": 9, "grades.1.date": ISODate("2014-08-11T00:00:00Z") },
3   { restaurant_id: 1, name: 1, grades: 1 }
4 );
5
```

Output

```
mycompiler_mongodb> ...
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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:**

The screenshot shows the MongoDB shell interface. At the top, it displays the user's details: 220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K. Below this is a toolbar with a MongoDB icon, dropdown menus, and buttons for Run and Save. The main area is divided into two sections: 'Query' and 'Output'. In the 'Query' section, the command is pasted. In the 'Output' section, the response from the MongoDB server is shown, starting with 'mycompiler\_mongodb> ...' followed by '[Execution complete with exit code 0]'. The interface has a light gray background with dark blue header and footer bars.

```
220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K
```

MongoDB ▾ ⓘ

Run Save

```
1 db.restaurants.find(
2   { $and: [
3     { "address.coord.1": { $gt: 42 } },
4     { "address.coord.1": { $lte: 52 } }
5   ] },
6   { _id: 0, restaurant_id: 1, name: 1, address: 1 }
7 );
8
```

Output

```
mycompiler_mongodb> ...
mycompiler_mongodb>

[Execution complete with exit code 0]
```

**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:**

The screenshot shows the MongoDB shell interface. At the top, there is a header bar with the user's name '220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K' and a 'Ctrl+Enter' button. Below the header are two buttons: 'MongoDB ▾' and 'i'. On the right side are 'Run' and 'Save' buttons. The main area has two tabs: 'Query' and 'Output'. The 'Query' tab contains the command: 'db.restaurants.find({}, { \_id: 0 }).sort({ name: 1 });'. The 'Output' tab shows the response from the database: 'mycompiler\_mongodb>', 'mycompiler\_mongodb>', '[Execution complete with exit code 0]'. The background of the interface is light gray.

**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:**

The screenshot shows the MongoDB shell interface. At the top, there is a header bar with the user's name '220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K' and a 'Ctrl+Enter' button. Below the header are two buttons: 'MongoDB ▾' and 'i'. On the right side are 'Run' and 'Save' buttons. The main area has two tabs: 'Query' and 'Output'. The 'Query' tab contains the command: 'db.restaurants.find({}, { \_id: 0 }).sort({ name: -1 });'. The 'Output' tab shows the response from the database: 'mycompiler\_mongodb>', 'mycompiler\_mongodb>', '[Execution complete with exit code 0]'. The background of the interface is light gray.

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:**

The screenshot shows a MongoDB query editor interface. At the top, it displays the user's information: "220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K". Below this are two tabs: "MongoDB" and "Run/Save". The code input area contains the following MongoDB command:

```
1 db.restaurants.find({}, { _id: 0 }).sort({ cuisine: 1, borou Output
2
3
```

The output window shows the results of the command:

```
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

At the bottom left, there is a link to the website: <https://www.mycompiler.io>.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB query editor interface. At the top, it displays the user's information: "220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K". Below this are two tabs: "MongoDB" and "Run/Save". The code input area contains the following MongoDB command:

```
1 db.restaurants.find({ "address.street": { $exists: true, $ne Output
2
```

The output window shows the results of the command:

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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:**

```
220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K
```

MongoDB ▾ ⓘ

Run Save

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

**Output**

```
mycompiler_mongodb>
mycompiler_mongodb>
```

```
[Execution complete with exit code 0]
```

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:**

```
220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K
```

MongoDB ▾ ⓘ

Run Save

```
1 db.restaurants.find(
2   { "grades.score": { $mod: [7, 0] } },
3   { restaurant_id: 1, name: 1, grades: 1 }
4 );
5
```

**Output**

```
mycompiler_mongodb> ... ... ...
mycompiler_mongodb>
```

```
[Execution complete with exit code 0]
```

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:**

The screenshot shows a MongoDB shell interface. At the top, there is a header bar with the text "220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K". Below the header are buttons for "MongoDB", "Run", and "Save". The main area has two sections: "Input" and "Output". The "Input" section contains the MongoDB query:1 db.restaurants.find(  
2 { name: /mon/i },  
3 { name: 1, borough: 1, "address.coord": 1, cuisine: 1 }  
4 );  
5The "Output" section shows the command prompt "mycompiler\_mongodb> ... . . ." followed by "[Execution complete with exit code 0]".

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:**

The screenshot shows a MongoDB shell interface. At the top, there is a header bar with the text "220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K". Below the header are buttons for "Ctrl+Enter", "Run", and "Save". The main area has two sections: "Input" and "Output". The "Input" section contains the MongoDB query:1 db.restaurants.find(  
2 { name: /^Mad/i },  
3 { name: 1, borough: 1, "address.coord": 1, cuisine: 1 }  
4 );  
5The "Output" section shows the command prompt "mycompiler\_mongodb> ... . . ." followed by "[Execution complete with exit code 0]".

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:**

220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K

MongoDB ▾



Run

Save

```
1 db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } } })
2
```

```
mycompiler_mongodb> ... ... ...
mycompiler_mongodb>
```

```
[Execution complete with exit code 0]
```

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:**

220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K

MongoDB ▾



Run

Save

```
1 db.restaurants.find(
2   { "grades": { $elemMatch: { "score": { $lt: 5 } } }, "bo
3 );
4
```

**Output**

```
mycompiler_mongodb> ... ... ...
mycompiler_mongodb>
```

```
[Execution complete with exit code 0]
```

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:

The screenshot shows the MongoDB shell interface. The top bar displays the user's name and ID: 220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K. Below the bar are buttons for MongoDB dropdown, info, Run, and Save. The code input area contains the following command:

```
1 db.restaurants.find(  
2   { "grades": { $elemMatch: { "score": { $lt: 5 } } } },  
3 );  
4
```

The output window is titled "Output" and shows the results of the query execution:

```
mycompiler_mongodb> ... ... ...  
mycompiler_mongodb>  
[Execution complete with exit code 0]
```

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:

The screenshot shows the MongoDB shell interface. The top bar displays the user's name and ID: 220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K. Below the bar are buttons for MongoDB dropdown, info, Run, and Save. The code input area contains the following command:

```
1 db.restaurants.find(  
2   { "grades": { $elemMatch: { "score": { $lt: 5 } } } },  
3 );  
4
```

The output window is titled "Output" and shows the results of the query execution:

```
mycompiler_mongodb> ... ... ...  
mycompiler_mongodb>  
[Execution complete with exit code 0]
```

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:**

The screenshot shows the MongoDB shell interface. At the top, it displays the user information: 220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K. Below this is a toolbar with a MongoDB icon, dropdown menus, and buttons for Run and Save. The main area is divided into two sections: 'Input' and 'Output'. The 'Input' section contains the MongoDB query from above. The 'Output' section shows the command being run: 'mycompiler\_mongodb> ... . . . . .' followed by '[Execution complete with exit code 0]'. This indicates that the query was successfully executed.

```
220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K  
MongoDB ▾ ⓘ Run Save  
1 db.restaurants.find(  
2   { "grades": { $elemMatch: { "score": { $lt: 5 } } }, "bo  
3 );  
4  
Output  
mycompiler_mongodb> ... . . . .  
mycompiler_mongodb>  
[Execution complete with exit code 0]
```

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:**

The screenshot shows the MongoDB shell interface. At the top, it displays the user information: 220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K. Below this is a toolbar with a MongoDB icon, dropdown menus, and buttons for Run and Save. The main area is divided into two sections: 'Input' and 'Output'. The 'Input' section contains the MongoDB query from above. The 'Output' section shows the command being run: 'mycompiler\_mongodb> ... . . . . .' followed by '[Execution complete with exit code 0]'. This indicates that the query was successfully executed.

```
220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K  
MongoDB ▾ ⓘ Run Save  
1 db.restaurants.find(  
2   { "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or  
3 );  
4  
Output  
mycompiler_mongodb> ... . . . .  
mycompiler_mongodb>  
[Execution complete with exit code 0]
```

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:**

```
220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K
```



▶ Run Save

```
1 db.restaurants.find(  
2   { "grades": { $elemMatch: { "score": { $lt: 5 } } } }, $or  
3 );  
4
```

**Output**

```
mycompiler_mongodb> ... . . .  
mycompiler_mongodb>
```

```
[Execution complete with exit code 0]
```

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:**

```
220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K
```



▶ Run Save

```
1 db.restaurants.find(  
2   { $and: [{ "grades.score": 2 }, { "grades.score": 6 }], $or: [{ "borough":  
3 }];  
4 )
```

**Output**

```
mycompiler_mongodb> ... . . .  
mycompiler_mongodb>
```

```
[Execution complete with exit code 0]
```

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:

The screenshot shows a MongoDB shell interface. At the top, there is a header bar with the text '220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K'. Below the header are two buttons: 'MongoDB' with a dropdown arrow and a small info icon. To the right are 'Run' and 'Save' buttons. The main area has two sections: 'Input' and 'Output'. The 'Input' section contains the MongoDB query. The 'Output' section shows the command being run ('mycompiler\_mongodb> ...'), followed by a prompt ('mycompiler\_mongodb>'), and the message '[Execution complete with exit code 0]'.

```
1 db.restaurants.find(
2   { $and: [{ "grades.score": 2 }, { "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $ne: "American" } });
3
4
```

Output

```
mycompiler_mongodb> ...
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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:

The screenshot shows a MongoDB shell interface. At the top, there is a header bar with the text '220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K'. Below the header are two buttons: 'MongoDB' with a dropdown arrow and a small info icon. To the right are 'Run' and 'Save' buttons. The main area has two sections: 'Input' and 'Output'. The 'Input' section contains the MongoDB query. The 'Output' section shows the command being run ('mycompiler\_mongodb> ...'), followed by a prompt ('mycompiler\_mongodb>'), and the message '[Execution complete with exit code 0]'.

```
1 db.restaurants.find(
2   { $and: [{ "grades.score": 2 }, { "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $nin: ["American", "Chinese"] } });
3
4 |
```

Output

```
mycompiler_mongodb> ...
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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:**

The screenshot shows a MongoDB shell interface. At the top, there is a header bar with the text '220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K'. Below the header are two buttons: 'MongoDB' with a dropdown arrow and a small info icon. To the right are 'Run' and 'Save' buttons. The main area has a light gray background. On the left, a code editor window displays the following MongoDB query:

```
1 db.restaurants.find(  
2   { $or: [{ "grades.score": 2 }, { "grades.score": 6 }]}  
3 );  
4 |
```

On the right, a panel titled 'Output' shows the results of the query execution:

```
mycompiler_mongodb> ... . . .  
mycompiler_mongodb>  
[Execution complete with exit code 0]
```

| 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:

The screenshot shows the MongoDB shell interface. At the top, there is a header bar with the user's name and ID: "220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K". Below the header are two buttons: "MongoDB" with a dropdown arrow and a small info icon. To the right are "Run" and "Save" buttons. The main area is divided into two sections: "Query" and "Output". In the "Query" section, the command `db.movies.find({ year: 1893 })` is typed. In the "Output" section, the response shows the prompt `mycompiler\_mongodb>` followed by the command again, and the message "[Execution complete with exit code 0]".

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:

The screenshot shows the MongoDB shell interface. At the top, there is a header bar with the user's name and ID: "220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K". Below the header are two buttons: "MongoDB" with a dropdown arrow and a small info icon. To the right are "Run" and "Save" buttons. The main area is divided into two sections: "Query" and "Output". In the "Query" section, the command `db.movies.find({ runtime: { \$gt: 120 } })` is typed. In the "Output" section, the response shows the prompt `mycompiler\_mongodb>` followed by the command again, and the message "[Execution complete with exit code 0]".

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

**QUERY:**

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

**OUTPUT:**

```
220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K
```

MongoDB ▾ 

Run

Save

```
1 db.movies.find({ genres: 'Short' })  
2  
3
```

**Output**

```
mycompiler_mongodb>  
mycompiler_mongodb>
```

```
[Execution complete with exit code 0]
```

**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:**

```
220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K
```

MongoDB ▾ 

Run

Save

```
1 db.movies.find({ directors: 'William K.L. Dickson' })  
2  
3  
4
```

**Output**

```
mycompiler_mongodb>  
mycompiler_mongodb>
```

```
[Execution complete with exit code 0]
```

**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:**

220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K

MongoDB ▾



Run

Save

```
1 db.movies.find({ countries: 'USA' })
2
3
4
5
```

Output

```
mycompiler_mongodb>
mycompiler_mongodb>
```

```
[Execution complete with exit code 0]
```

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

**QUERY:**

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

**OUTPUT:**

220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K

Ctrl+Enter

MongoDB ▾



Run

Save

```
1 db.movies.find({ rated: 'UNRATED' })
2
3
4
5
6
```

Output

```
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>
```

```
[Execution complete with exit code 0]
```

**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:**

The screenshot shows a MongoDB query interface. At the top, it displays the user's details: 220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K. Below this is a toolbar with a MongoDB icon, a dropdown menu, and buttons for 'Run' and 'Save'. The code input area contains the following query:

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

The output panel is titled 'Output' and shows the results of the query. It lists seven documents, each starting with 'mycompiler\_mongodb>'. The results are identical to the previous screenshot. At the bottom of the output panel, a message indicates: '[Execution complete with exit code 0]'. A small URL at the bottom left of the interface is <https://www.mycompiler.io>.

**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:**

The screenshot shows a MongoDB query interface. At the top, it displays the user's details: 220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K. Below this is a toolbar with a MongoDB icon, a dropdown menu, and buttons for 'Run' and 'Save'. The code input area contains the following query:

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

The output panel is titled 'Output' and shows the results of the query. It lists two documents, both starting with 'mycompiler\_mongodb>'. The results are identical to the previous screenshots. At the bottom of the output panel, a message indicates: '[Execution complete with exit code 0]'. A small URL at the bottom left of the interface is <https://www.mycompiler.io>.

**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:**

220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K

MongoDB ▾ 

```
1 db.movies.find({ 'tomatoes.viewer.rating': { $gt: 4 } })  
2 |  
3
```

 Run

 Save

**Output**

```
mycompiler_mongodb>  
mycompiler_mongodb>
```

[Execution complete with exit code 0]

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

**QUERY:**

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

**OUTPUT:**

220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K

MongoDB ▾ 

```
1 db.movies.find({ 'awards.wins': { $gt: 0 } })  
2  
3  
4
```

 Run

 Save

**Output**

```
mycompiler_mongodb>  
mycompiler_mongodb>  
mycompiler_mongodb>  
mycompiler_mongodb>
```

[Execution complete with exit code 0]

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

**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:**

The screenshot shows the MongoDB shell interface. At the top, it displays the user's details: 220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K. Below this is the command input area where the query is typed. To the right is the 'Output' panel which shows the results of the query execution. The output consists of four blank lines followed by the message '[Execution complete with exit code 0]'. The 'Run' and 'Save' buttons are visible at the top right of the interface.

```
1 db.movies.find(
2   { 'awards.nominations': { $gt: 0 } },
3   {
4     title: 1,
5     languages: 1,
6     released: 1,
7     directors: 1,
8     writers: 1,
9     awards: 1,
10    year: 1,
11    genres: 1,
12    runtime: 1,
13    cast: 1,
14    countries: 1
15  }
16 )
17
```

Output

```
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

**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:**

The screenshot shows the MongoDB shell interface. At the top, it displays the user's details: 220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K. Below this is the command input area where the query is typed. To the right is the 'Output' panel which shows the results of the query execution. The output consists of several blank lines followed by the message '[Execution complete with exit code 0]'. The 'Run' and 'Save' buttons are visible at the top right of the interface.

```
1 db.movies.find(
2   { cast: 'Charles Kayser' },
3   {
4     title: 1,
5     languages: 1,
6     released: 1,
7     directors: 1,
8     writers: 1,
9     awards: 1,
10    year: 1,
11    genres: 1,
12    runtime: 1,
13    cast: 1,
14    countries: 1
15  }
16 )
17
```

Output

```
mycompiler_mongodb> .....
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

**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:**

```
220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K
```

MongoDB ▾ ⓘ

▶ Run

Save

```
1 db.movies.find(  
2     { released: ISODate("1893-05-09T00:00:00.000Z") },  
3     {  
4         title: 1,  
5         languages: 1,  
6         released: 1,  
7         directors: 1,  
8         writers: 1,  
9         countries: 1  
10    }  
11 )  
12  
13  
14  
15  
16
```

Output

```
mycompiler_mongodb> ... . . . . .  
mycompiler_mongodb>  
mycompiler_mongodb>  
mycompiler_mongodb>  
mycompiler_mongodb>
```

[Execution complete with exit code 0]

**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:**

```
220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K
```

MongoDB ▾ ⓘ

▶ Run

Save

```
1 db.movies.find(  
2     { title: /scene/i },  
3     {  
4         title: 1,  
5         languages: 1,  
6         released: 1,  
7         directors: 1,  
8         writers: 1,  
9         countries: 1  
10    }  
11 )  
12  
13  
14  
15  
16
```

Output

```
mycompiler_mongodb> ... . . . . .  
mycompiler_mongodb>  
mycompiler_mongodb>  
mycompiler_mongodb>  
mycompiler_mongodb>  
mycompiler_mongodb>
```

[Execution complete with exit code 0]

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

**RESULT:**