

# CREATING AND MANAGING TABLES

EX.NO.1

DATE:18/02/2024

1. Create the DEPT table based on the DEPARTMENT following the table instance chart below. Confirm that the table is created.

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

**QUERY :**

Create table DEPARTMENT1(Dept\_id number(6),Dept\_name varchar(20),manager\_id number(6),Location\_id number(4));

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'Bhuvaneshwari M mbhuv', and a schema dropdown set to 'WKSP\_MBHUV'. The main workspace is titled 'SQL Commands' and contains a command input field with the SQL code: 'Create table DEPARTMENT1(Dept\_id number(6),Dept\_name varchar(20),manager\_id number(6),Location\_id number(4));'. Below the input field, the status bar shows 'Table created.' and '0.04 seconds'.

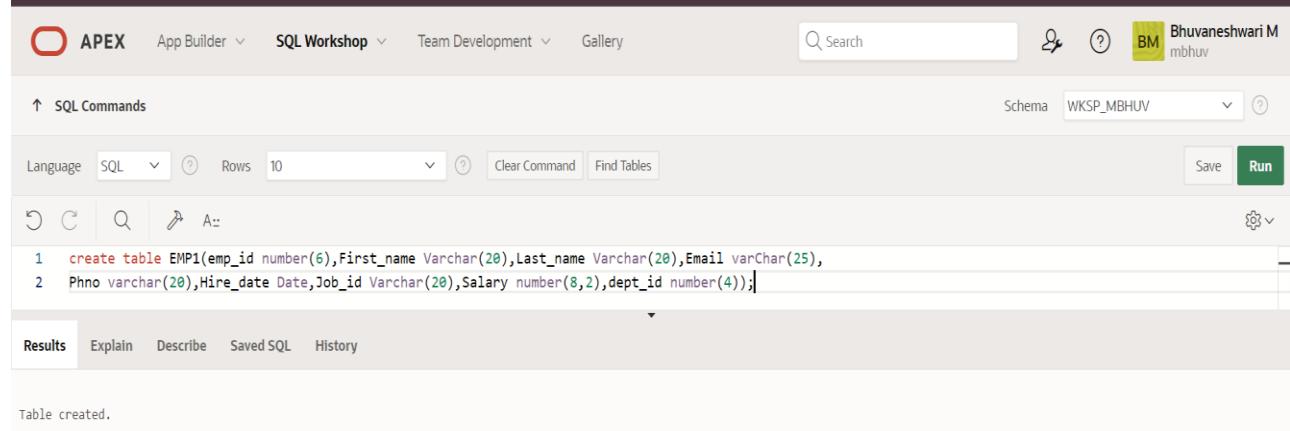
2. Create the EMP table based on the following instance chart. Confirm that the table is created.

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

#### QUERY :

```
create table EMP1(emp_id number(6),First_name Varchar(20),Last_name  
Varchar(20),Email varChar(25),Phno varchar(20),Hire_date Date,Job_id Varchar(20),Salary  
number(8,2),dept_id number(4));
```

#### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'Bhuvaneshwari M mbhuv', and a 'Run' button. The main workspace is titled 'SQL Commands'. It has dropdowns for 'Language' (set to SQL) and 'Rows' (set to 10). Below these are icons for Undo, Redo, Find, and Paste. The SQL editor contains the following code:

```
1 create table EMP1(emp_id number(6),First_name Varchar(20),Last_name Varchar(20),Email varChar(25),  
2 Phno varchar(20),Hire_date Date,Job_id Varchar(20),Salary number(8,2),dept_id number(4));
```

At the bottom, there are tabs for 'Results' (selected), Explain, Describe, Saved SQL, and History. The status bar at the bottom left shows 'Table created.' and '0.04 seconds'.

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

**QUERY:**

```
alter table EMP1 modify (last_name varchar(50));
```

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'Bhuvaneswari M mbhuv'. The main area is titled 'SQL Commands' with a search bar and a schema dropdown set to 'WKSP\_MBHUV'. Below the search bar are buttons for Save and Run. The SQL editor contains the command: '1 alter table EMP1 modify (last\_name varchar(50));'. The results tab shows the output: 'Table altered.' and a execution time of '0.05 seconds'.

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

**QUERY:**

```
create table EMPLOYEES2(id number(6),First_name Varchar(20),Last_name  
Varchar(20),Salary number(8,2),dept_id number(4));
```

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'Bhuvaneswari M mbhuv'. The main area is titled 'SQL Commands' with a search bar and a schema dropdown set to 'WKSP\_MBHUV'. Below the search bar are buttons for Save and Run. The SQL editor contains the command: '1 create table EMPLOYEES2(id number(6),First\_name Varchar(20),Last\_name Varchar(20),Salary number(8,2),dept\_id number(4));'. The results tab shows the output: 'Table created.' and a execution time of '0.03 seconds'.

5. Drop the EMP table.

**QUERY:**

**Drop table EMP1;**

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile 'Bhuvaneshwari M mbhuv' are also present. The main area is titled 'SQL Commands' with a sub-section 'Language: SQL'. It displays the command 'drop table EMP1;'. Below the command, the 'Results' tab is selected, showing the output 'Table dropped.' and a execution time of '0.08 seconds'.

6. Rename the EMPLOYEES2 table as EMP.

**QUERY:**

**alter table EMPLOYEES2 rename to EMP;**

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile 'Bhuvaneshwari M mbhuv' are also present. The main area is titled 'SQL Commands' with a sub-section 'Language: SQL'. It displays the command 'alter table EMPLOYEES2 rename to EMP;'. Below the command, the 'Results' tab is selected, showing the output 'Table altered.' and a execution time of '0.06 seconds'.

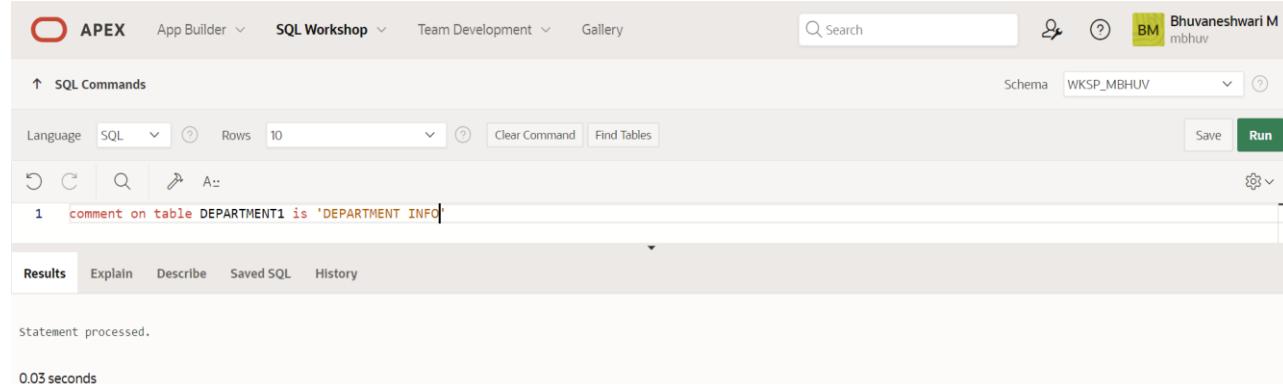
7. Add a comment on DEPT and EMP tables. Confirm the modification by describing the table.

**QUERY:**

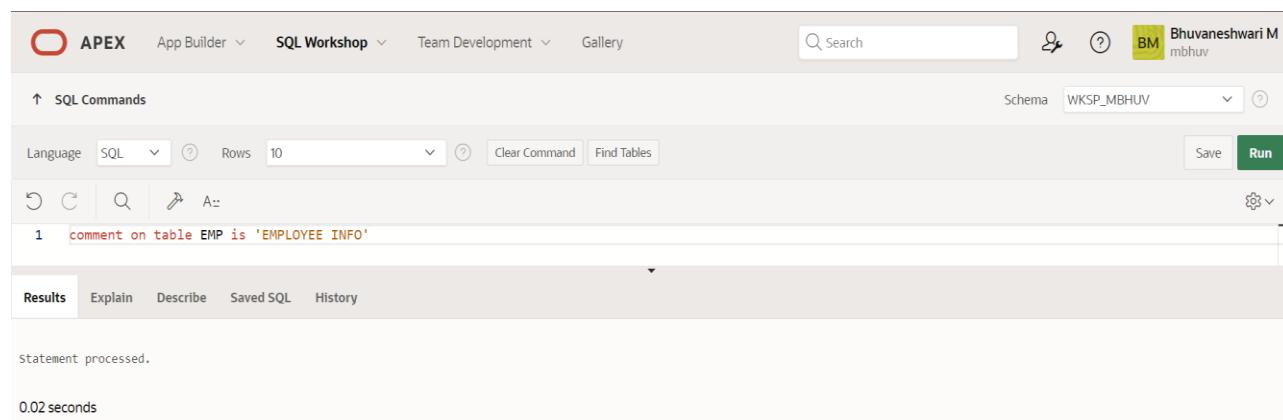
**comment on table DEPARTMENT1 is 'DEPARTMENT INFO'**

**comment on table EMP is 'EMPLOYEE INFO'**

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user profile 'Bhuvaneshwari M mbhuv'. The main area is titled 'SQL Commands' with a search bar and a schema dropdown set to 'WKSP\_MBHUV'. Below the title are buttons for Language (SQL), Rows (10), Clear Command, and Find Tables. The command input field contains '1 comment on table DEPARTMENT1 is 'DEPARTMENT INFO''. The results tab is selected, showing the output 'Statement processed.' and a execution time of '0.03 seconds'.



The screenshot shows the Oracle SQL Workshop interface, identical to the previous one but with a different command. The command input field now contains '1 comment on table EMP is 'EMPLOYEE INFO''. The results tab shows the output 'Statement processed.' and a execution time of '0.02 seconds'.

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

**QUERY:**

```
alter table emp drop column First_name;  
describe emp;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands pane, the command `alter table emp drop column First_name;` is entered. Below the command, the output shows "Table altered." and a execution time of "0.08 seconds".

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands pane, the command `describe emp;` is entered. The Results tab is selected, and the output displays the structure of the EMP table:

| Table | Column    | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|-------|-----------|-----------|--------|-----------|-------|-------------|----------|---------|---------|
| EMP   | EMP_ID    | NUMBER    | -      | 6         | 0     | -           | ✓        | -       | -       |
|       | LAST_NAME | VARCHAR2  | 20     | -         | -     | -           | ✓        | -       | -       |
|       | SALARY    | NUMBER    | -      | 8         | 2     | -           | ✓        | -       | -       |
|       | DEPT_ID   | NUMBER    | -      | 4         | 0     | -           | ✓        | -       | -       |

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

**RESULT:**

The Queries for the given Creating and Managing tables has been executed successfully.

# MANIPULATING DATA

EX.NO.2

DATE:22/02/2024

## Find the Solution for the following:

1. Create MY\_EMPLOYEE table with the following structure

| NAME       | NULL?    | TYPE        |
|------------|----------|-------------|
| ID         | Not null | Number(4)   |
| Last_name  |          | Varchar(25) |
| First_name |          | Varchar(25) |
| Userid     |          | Varchar(25) |
| Salary     |          | Number(9,2) |

### QUERY:

Create table MY\_EMP (ID Number(4),Last\_name varchar(25), First\_name varchar(25),Userid varchar2(25), Salary Number(9,2));

### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for Bhuvaneshwari M mbhuv, and a schema dropdown set to WKSP\_MBHUV. Below the toolbar, the SQL Commands section has Language set to SQL, Rows to 10, and includes Clear Command and Find Tables buttons. The main area contains the SQL code for creating the table MY\_EMP. The results tab is selected, showing the output "Table created." and a execution time of "0.05 seconds".

```
1 create table MY_EMP(ID Number(4), LastName Varchar(25), FirstName Varchar(25), Userid varchar2(25), Salary Number(9,2));
2
```

Results Explain Describe Saved SQL History

Table created.

0.05 seconds

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

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

#### QUERY:

```
INSERT INTO MY_EMPL0 values(1, 'patel','Ralph','rpatel',895);
```

```
INSERT INTO MY_EMPL0 values(2, 'Dancs','Betty','bdancs',860);
```

#### 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 identified as Bhuvaneshwari M mbhuv. The main workspace is titled "SQL Commands". The language is set to SQL, and the number of rows is set to 10. The command entry area contains the following SQL code:

```
1  INSERT INTO MY_EMPL0 values(1, 'patel','Ralph','rpatel',895);
2
```

The results tab shows the output: "1 row(s) inserted." Below the results, a note indicates "0.03 seconds".

3. Display the table with values.

**QUERY:**

```
SELECT * from MY_EMP;
```

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, a user icon, and a profile picture for 'Bhuvaneshwari M mbhuv'. Below the navigation is a toolbar with icons for Undo, Redo, Search, and Find Tables. The main area shows a SQL command line with the query 'Select \* from MY\_EMP;'. The results tab is selected, displaying a table with two rows of data:

| ID | LASTNAME | FIRSTNAME | USERID | SALARY |
|----|----------|-----------|--------|--------|
| 1  | Patel    | Ralph     | rpatel | 895    |
| 2  | Dancs    | Betty     | bdancs | 860    |

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

**QUERY:**

```
INSERT INTO MY_EMPL0 values(3, 'Biri','Ben','bbiri',1100);
INSERT INTO MY_EMPL0 values(4, 'Newman','Chad','Cnewman',750);
```

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar and toolbar are identical to the previous screenshot. The main area shows a SQL command line with the query 'INSERT INTO NY\_EMP values(3, 'Ben','Biri','bbiri',1100);'. The results tab is selected, showing the message '1 row(s) inserted.' and '0.00 seconds'.

5. Make the data additions permanent.

**QUERY:**

```
SELECT * from MY_EMPL0;
```

**OUTPUT:**

The screenshot shows the Oracle APEX interface with the SQL Workshop tab selected. The schema is set to WKSP\_MBHV. In the command editor, the following SQL statement is run:

```
1 select * from MY_EMP;
```

The results page displays the following data:

| ID | LASTNAME | FIRSTNAME | USERID  | SALARY |
|----|----------|-----------|---------|--------|
| 1  | Patel    | Ralph     | rpatel  | 895    |
| 2  | Dancs    | Betty     | bdancs  | 860    |
| 3  | Ben      | Birl      | bbirl   | 1100   |
| 4  | Newman   | Chad      | Cnewman | 750    |

4 rows returned in 0.00 seconds

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

**QUERY:**

**UPDATE MY\_EMP SET LastName='Drexler' where id=3;**

**OUTPUT:**

The screenshot shows the Oracle APEX interface with the SQL Workshop tab selected. The schema is set to WKSP\_MBHV. In the command editor, the following SQL statement is run:

```
1 UPDATE MY_EMP SET LastName='Drexler' where id=3;
```

The results page shows the output:

1 row(s) updated.

0.01 seconds

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

**QUERY:**

**UPDATE MY\_EMP SET Salary=1000 where Salary<900;**

**OUTPUT:**

The screenshot shows the Oracle APEX interface with the SQL Workshop tab selected. The schema is set to WKSP\_MBHV. In the command editor, the following SQL statement is run:

```
1 UPDATE MY_EMP SET Salary=1000 where Salary<900;
```

The results page shows the output:

3 row(s) updated.

0.01 seconds

8. Delete Betty dancs from MY\_EMPLOYEE table.

**QUERY:**

**DELETE from MY\_EMP where Firstname ='Betty' and LastName='Dancs';**

## OUTPUT:

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

```
↑ SQL Commands
Language: SQL Rows: 10 Clear Command Find Tables
1 DELETE FROM MY_EMP WHERE FirstName='Betty' and LastName='Dance';
2
Results Explain Describe Saved SQL History
```

The results section shows the output: "1 row(s) deleted." and "0.03 seconds".

9. Empty the fourth row of the emp table.

### QUERY:

**Delete from MY\_EMP where id=4;**

## OUTPUT:

A screenshot of the Oracle SQL Workshop interface, identical to the previous one but with a different query. The SQL command window contains:

```
↑ SQL Commands
Language: SQL Rows: 10 Clear Command Find Tables
1 delete from MY_EMP where id=4;
2
Results Explain Describe Saved SQL History
```

The results show "1 row(s) deleted." and "0.01 seconds".

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

## RESULT:

The Queries for manipulating data has been executed successfully.

# INCLUDING CONSTRAINTS

EX.NO.3

DATE:29/02/2024

1. Add a table-level PRIMARY KEY constraint to the EMP table on the ID column. The constraint should be named at creation. Name the constraint my\_emp\_id\_pk.

## QUERY:

```
Create table Employee1(emp_id number(6),last_name varchar(25) NOT NULL,email  
varchar(25),Salary number(8,2),CONSTRAINT my_emp_id_pk PRIMARY KEY(emp_id));
```

## **OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX' logo, 'App Builder', 'SQL Workshop', 'Team Development', 'Gallery', a search bar, and user information for 'Bhuvaneshwari M mbhuv'. Below the navigation is a toolbar with icons for Undo, Redo, Find, Replace, and Save. The main area displays the following SQL command:

```
1 Create table Employee1(emp_id number(6) ,last_name varchar(25) NOT NULL,email varchar(25),Salary number(8,2),CONSTRAINT my_emp_id_pk PRIMARY KEY(emp_id));  
2 |
```

Below the SQL editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The status bar at the bottom indicates 'Table created.' and a execution time of '0.07 seconds'.

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

## QUERY:

```
Create table MYDEPT (dept_id number(6),first_name varchar(25) NOT NULL,email  
varchar(25),Salary number(8,2),CONSTRAINT my_dept_id_pk PRIMARY KEY(dept_id));
```

## OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user profile (Bhuwaneshwari M mbhuv), and a 'BM' button. The main workspace is titled 'SQL Commands'. It features a toolbar with Language (SQL selected), Rows (set to 10), Clear Command, Find Tables, Save, and Run buttons. Below the toolbar, the SQL command is entered:

```
1 Create table MYDEPT(dept_id number(6) ,first_name varchar(25) NOT NULL,email varchar(25),Salary number(8,2),CONSTRAINT my_dept_id_pk PRIMARY KEY(dept_id));
2
```

The results pane at the bottom shows the message "Table created." and a execution time of "0.07 seconds".

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

**QUERY:**

```
ALTER TABLE Employee1 ADD DEPT1_ID number(6);
alter table Employee1 ADD CONSTRAINT my_emp_dept_id_fk FOREIGN KEY(dept1_id)
REFERENCES Employee1(emp_id);
```

**OUTPUT:**

The screenshot shows two separate sessions in the Oracle SQL Workshop interface. Both sessions are titled 'APEX' and have the schema 'WKSP\_MBHV' selected. The top session shows the creation of a new column 'DEPT1\_ID' with a length of 6 digits. The bottom session shows the addition of a foreign key constraint named 'my\_emp\_dept\_id\_fk' that references the 'Employee1' table's 'emp\_id' column. Both sessions show a successful execution with a message 'Table altered.' and a duration of approximately 0.05 to 0.07 seconds.

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

**QUERY:**

```
alter table Employee1 ADD COMMISSION number (2,2) check (COMMISSION > 0);
```

**OUTPUT:**

The screenshot shows a single session in the Oracle SQL Workshop interface. The session is titled 'APEX' and has the schema 'WKSP\_MBHV' selected. It displays the command to add a new column 'COMMISSION' of type 'number (2,2)' with a check constraint ensuring values are greater than zero. The execution is successful, resulting in a message 'Table altered.' and a duration of 0.06 seconds.

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

## **RESULT:**

The Queries for including constraints has been executed successfully.

# Writing Basic SQL SELECT Statements

**EX.NO.4**

**DATE:02/03/2024**

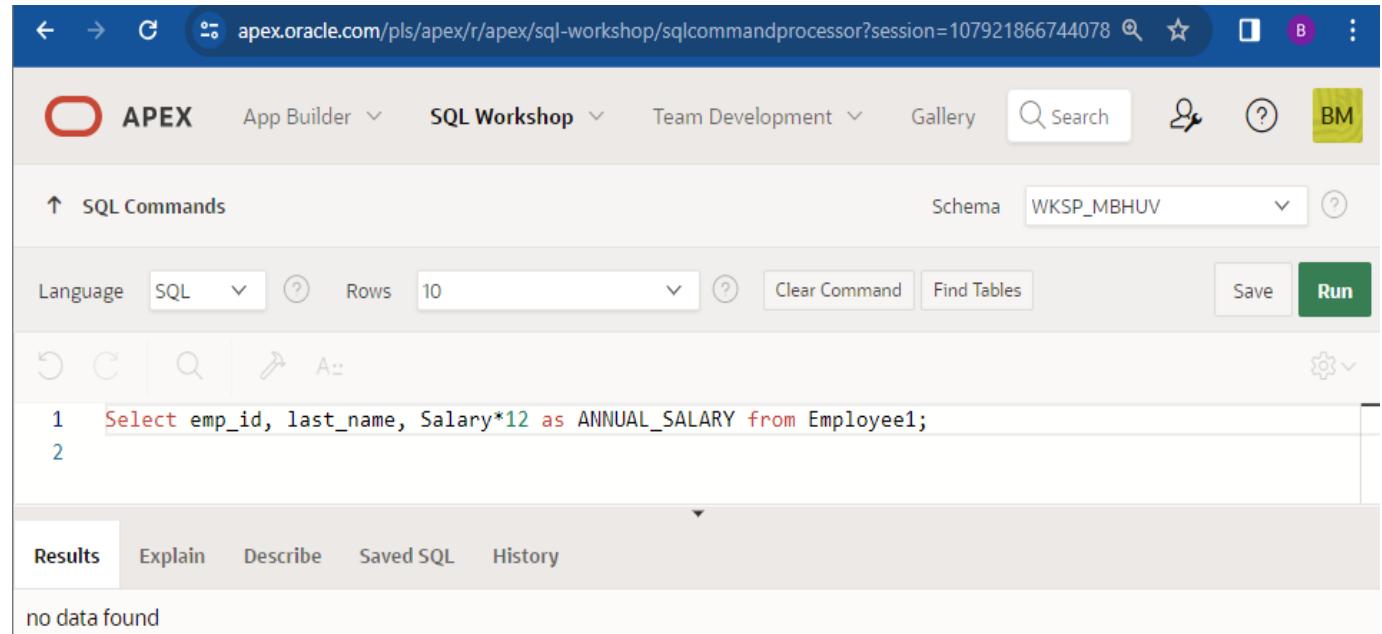
## 1.Identify the Errors

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

**Correct Query:**

**Select emp\_id , last\_name,Salary\*12 as ANNUAL\_SALARY from Employee1;**

## OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (which is selected), Team Development, Gallery, and various search and help icons. The main workspace is titled "SQL Commands" and shows the schema "WKSP\_MBHUV". The language is set to "SQL" and the number of rows is set to "10". The "Run" button is visible. Below the toolbar, there are several icons for navigating between tabs and performing common operations like saving and clearing. The SQL command entered is:

```
1 Select emp_id, last_name, Salary*12 as ANNUAL_SALARY from Employee1;  
2
```

The results tab is active, showing the message "no data found".

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

**QUERY:**

```
Create table DEPARTMENT (dept_id number(6), first_name varchar(25) NOT
NULL ,email varchar(25),Salary number(8,2));
DESC DEPARTMENT;
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below the navigation, there's a search bar and a user profile for 'Bhuvaneshwari M mbhuv'. The main area is titled 'SQL Commands'. The schema dropdown is set to 'WKSP\_MBHV'. The SQL editor contains the following code:

```
1 Create table DEPARTMENT(dept_id number(6) ,first_name varchar(25) NOT NULL ,email varchar(25),Salary number(8,2));
2
```

Below the editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active. The output shows the message 'Table created.' and a execution time of '0.04 seconds'.



This screenshot shows the same APEX interface after running the 'DESC DEPARTMENT;' command. The SQL editor now contains:

```
1 DESC DEPARTMENT;
```

The 'Describe' tab is now active in the results panel. It displays the structure of the 'DEPARTMENT' table with four columns: 'DEPT\_ID', 'FIRST\_NAME', 'EMAIL', and 'SALARY'. The table structure is as follows:

| Table      | Column     | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|------------|------------|-----------|--------|-----------|-------|-------------|----------|---------|---------|
| DEPARTMENT | DEPT_ID    | NUMBER    | -      | 6         | 0     | -           | ✓        | -       | -       |
|            | FIRST_NAME | VARCHAR2  | 25     | -         | -     | -           | -        | -       | -       |
|            | EMAIL      | VARCHAR2  | 25     | -         | -     | -           | ✓        | -       | -       |
|            | SALARY     | NUMBER    | -      | 8         | 2     | -           | ✓        | -       | -       |



This screenshot shows the 'Describe' output for the 'DEPARTMENT' table. The table has four columns: 'DEPT\_ID', 'FIRST\_NAME', 'EMAIL', and 'SALARY'. The 'DEPT\_ID' column is defined as a NUMBER type with a precision of 6 and a scale of 0. The 'FIRST\_NAME' column is defined as a VARCHAR2 type with a length of 25. The 'EMAIL' column is defined as a VARCHAR2 type with a length of 25. The 'SALARY' column is defined as a NUMBER type with a precision of 8 and a scale of 2. Primary key status is indicated by a checkmark in the 'Primary Key' column for 'DEPT\_ID' and 'EMAIL'.

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

**QUERY:**

```
Create table Employees(emp_id number(6), job_id varchar(25) NOT NULL,email
varchar(25),Salary number(8,2), hire_date date);
```

```
Insert into Employees values(1,21,'AAA','abc@gmail.com',10000,01/01/2024);
```

```
SELECT emp_id, last_name, job_id, hire_date from Employees;
```

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, the schema is set to 'WKSP\_MBHV' and the user is 'Bhuvaneshwari M mbhuv'. The main area displays the SQL command for creating the 'Employees' table:

```
1 Create table Employees(emp_id number(6), job_id varchar(25),last_name varchar(25) NOT NULL,email varchar(25),Salary number(8,2), hire_date number(8));
2
```

The 'Results' tab is selected, showing the output: 'Table created.' Below it, the execution time is listed as '0.04 seconds'.

The screenshot shows the Oracle SQL Workshop interface. The schema is 'WKSP\_MBHV' and the user is 'Bhuvaneshwari M mbhuv'. The main area displays the SQL command for inserting data into the 'Employees' table:

```
1 Insert into Employees values(1,21,'AAA','abc@gmail.com',10000,01/01/24);
```

The 'Results' tab is selected, showing the output: '1 row(s) inserted.' Below it, the execution time is listed as '0.02 seconds'.

The screenshot shows the Oracle SQL Workshop interface. The schema is 'WKSP\_MBHV' and the user is 'Bhuvaneshwari M mbhuv'. The main area displays the SQL command for selecting data from the 'Employees' table:

```
1 Select emp_id,last_name,job_id,hire_date from Employees;
```

The 'Results' tab is selected, displaying the query results in a table format:

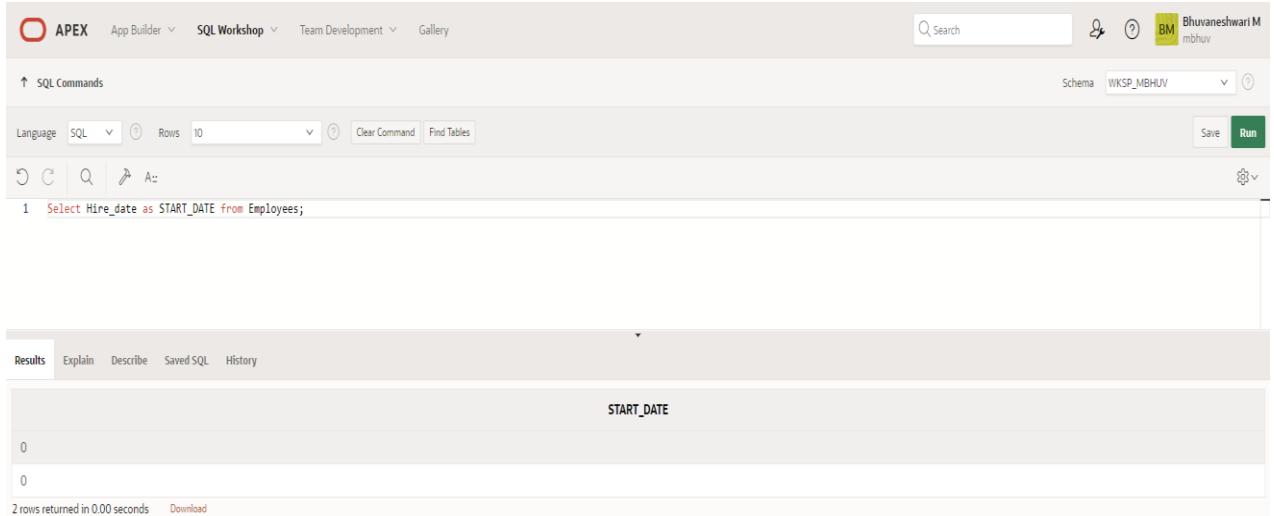
| EMP_ID | LAST_NAME | JOB_ID | HIRE_DATE  |
|--------|-----------|--------|------------|
| 1      | AAA       | 21     | 01/01/2024 |

4. Provide an alias STARTDATE for the hire date.

**QUERY:**

**Select Hire\_date as STARTDATE from Employees;**

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'Bhuvaneshwari M mbhuv'. The main area has tabs for SQL Commands, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, showing the query: 'Select Hire\_date as START\_DATE from Employees;'. The results tab shows a single column named 'START\_DATE' with two rows containing the value '0'. Below the results, it says '2 rows returned in 0.00 seconds'.

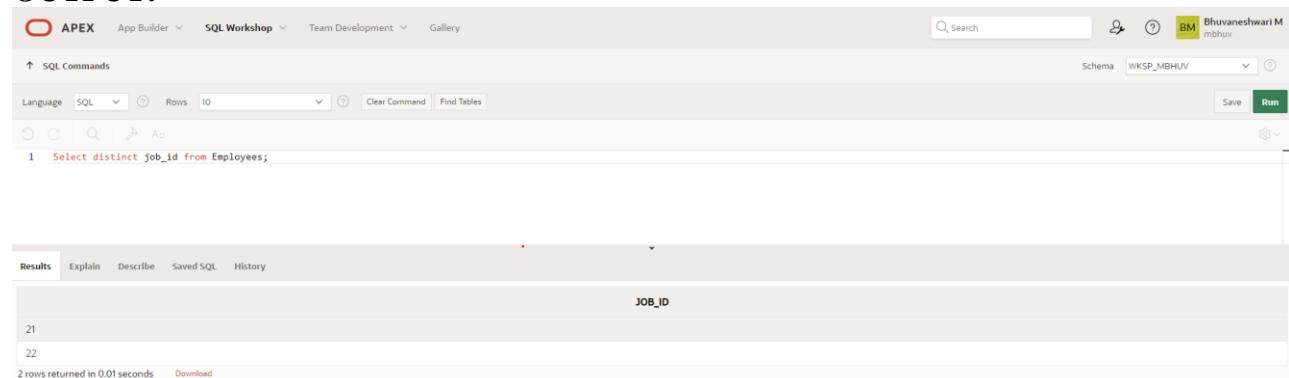
| START_DATE |
|------------|
| 0          |
| 0          |

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

**QUERY:**

**SELECT DISTINCT job\_id from Employees;**

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'Bhuvaneshwari M mbhuv'. The main area has tabs for SQL Commands, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, showing the query: 'Select distinct job\_id from Employees;'. The results tab shows a single column named 'JOB\_ID' with two rows containing the values '21' and '22'. Below the results, it says '2 rows returned in 0.01 seconds'.

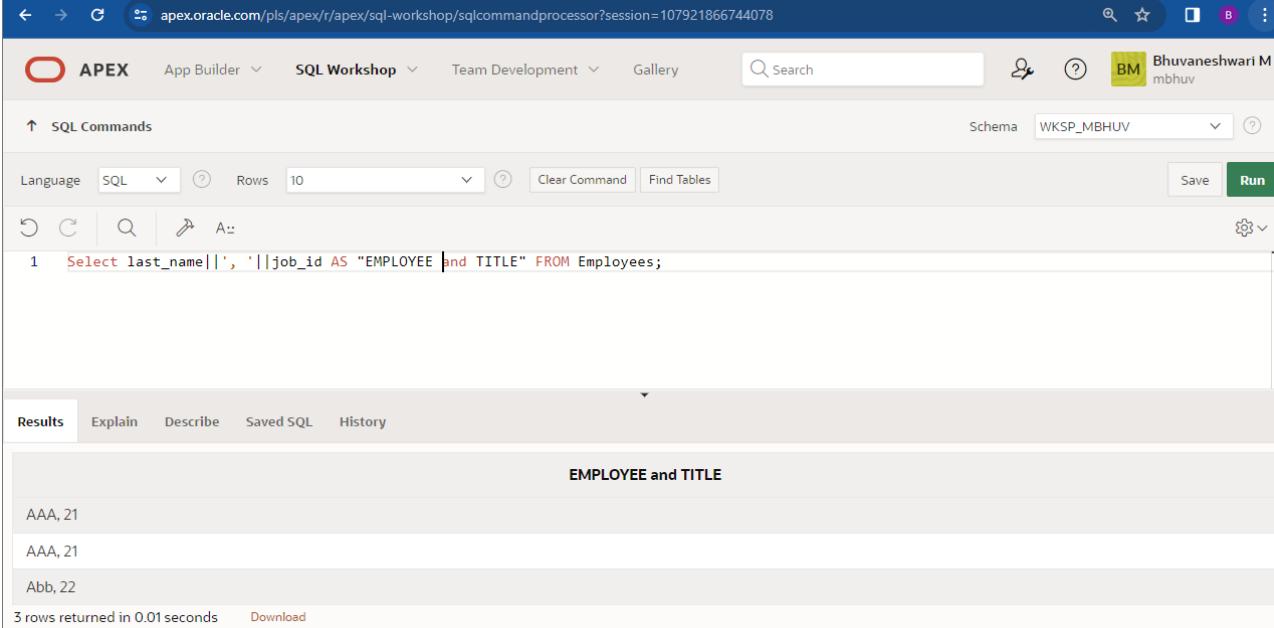
| JOB_ID |
|--------|
| 21     |
| 22     |

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

**QUERY:**

Select last\_name||','||job\_id AS "EMPLOYEE and TITLE" FROM Employees;

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (which is selected), Team Development, and Gallery. A search bar and user profile are also present. The main area is titled 'SQL Commands' and contains a code editor with the following SQL statement:

```
1 Select last_name||','||job_id AS "EMPLOYEE and TITLE" FROM Employees;
```

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

**EMPLOYEE and TITLE**

|         |
|---------|
| AAA, 21 |
| AAA, 21 |
| Abb, 22 |

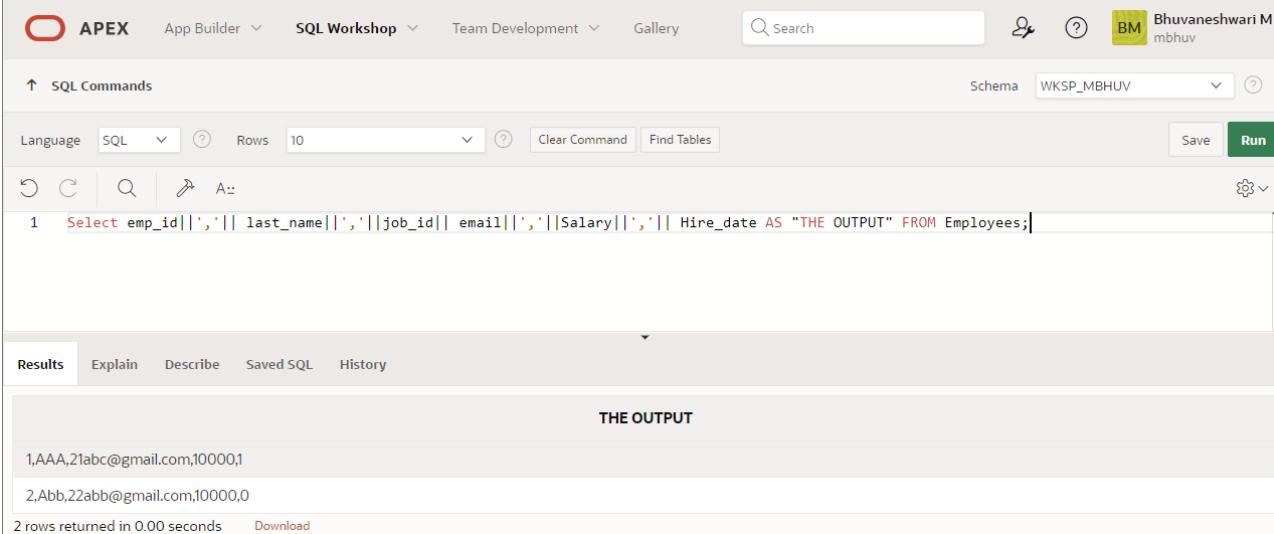
3 rows returned in 0.01 seconds    [Download](#)

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

**QUERY:**

Select emp\_id||','|| last\_name||','||job\_id|| email||','||Salary||','|| Hire\_date AS "THE OUTPUT" FROM Employees;

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface, identical to the previous one but with a different query. The top navigation bar and code editor are the same. The Results tab is selected, displaying the output of the query:

**THE OUTPUT**

|                               |
|-------------------------------|
| 1,AAA,21abc@gmail.com,10000,1 |
| 2,Abb,22abb@gmail.com,10000,0 |

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

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

### **RESULT:**

The Queries for writing basic SQL statements has been executed successfully.

# RESTRICTING AND SORTING DATA

**EX.NO.5**

**DATE:07/03/2024**

Find the Solution for the following:

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

**QUERY:**

**Select last\_name ,Salary from Employe;**

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the query `Select last_name,Salary from Employe;` is entered. The Results pane displays the output:

| LAST_NAME | SALARY |
|-----------|--------|
| MOHANRAJ  | 20000  |
| MOHANRAJ  | 10000  |

2 rows returned in 0.01 seconds

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

**QUERY:**

**Select last\_name, dept\_id from Employe where e\_id=176;**

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the query `Select last_name, dept_id from Employe where e_id=176;` is entered. The Results pane displays the output:

| LAST_NAME | DEPT_ID |
|-----------|---------|
| KUMAR     | 6       |

1 rows returned in 0.01 seconds

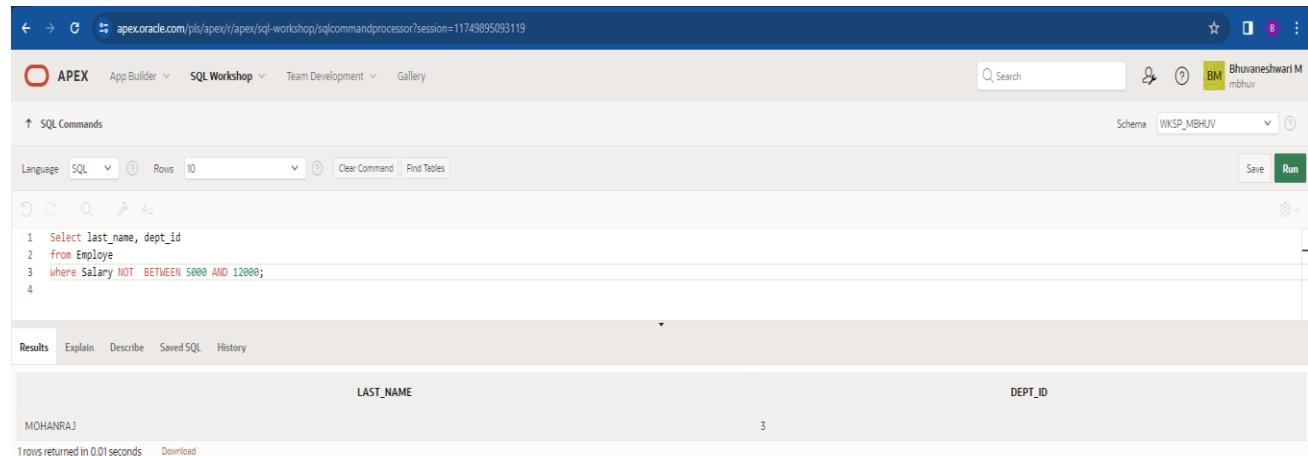
3. Create a query to display the last name and salary of employees whose salary is not in the range of 5000 and 12000. (hints: not between )

**QUERY:**

**SELECT last\_name , dept\_id from Employe**

**Where Salary NOT BETWEEN 5000 AND 12000**

**OUTPUT:**



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

```
1 Select last_name, dept_id
2 from Employe
3 where Salary NOT BETWEEN 5000 AND 12000;
4
```

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

| LAST_NAME | DEPT_ID |
|-----------|---------|
| MOHANRAJ  | 3       |

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

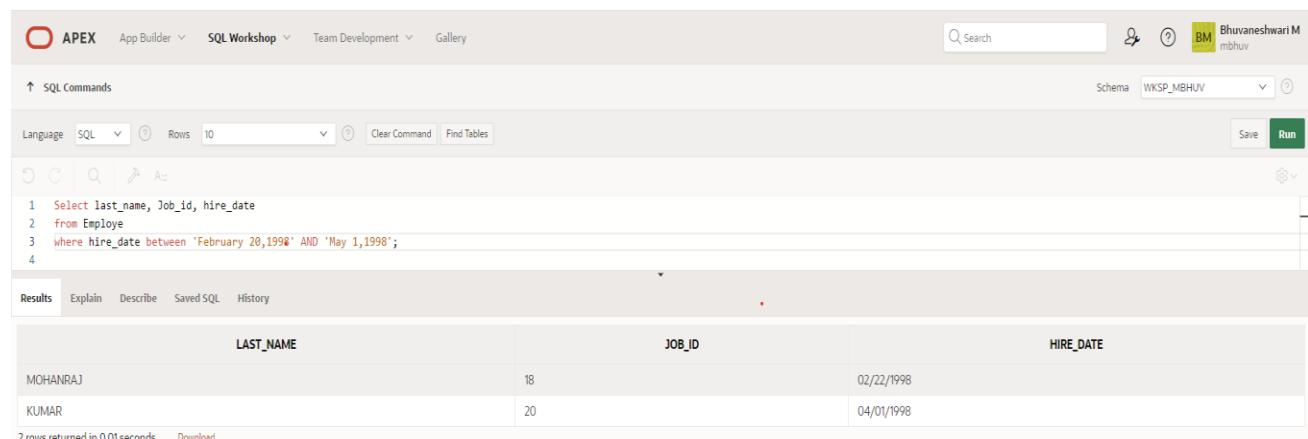
4. Display the employee last name, job ID, and start date of employees hired between February 20,1998 and May 1,1998.order the query in ascending order by start date.(hints: between)

**QUERY:**

**Select last\_name, Job\_id, hire\_date from Employe**

**where hire\_date between 'February 20,1998' AND 'May 1,1998';**

**OUTPUT:**



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

```
1 Select last_name, Job_id, hire_date
2 from Employe
3 where hire_date between 'February 20,1998' AND 'May 1,1998';
4
```

The results table has columns LAST\_NAME, JOB\_ID, and HIRE\_DATE. Two rows are returned:

| LAST_NAME | JOB_ID | HIRE_DATE  |
|-----------|--------|------------|
| MOHANRAJ  | 18     | 02/22/1998 |
| KUMAR     | 20     | 04/01/1998 |

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

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,Salary from Employee where (Salary BETWEEN 5000 AND 12000) AND (dept\_id IN(20,50)) order by last\_name AS EMPLOYEE, MONTHLY SALARY;**

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows a user profile for 'Bhuvaneshwari M mbhuv'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the following code:

```
1 Select last_name, dept_id from Employee
2 where dept_id IN(20,50) order by last_name;
```

The Results tab displays the output:

| LAST_NAME | DEPT_ID |
|-----------|---------|
| PAVENDAN  | 50      |
| PRIYA     | 20      |

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

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 where**

**(Salary BETWEEN 5000 AND 12000) AND (dept\_id IN(20,50)) order by last\_name ;**

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows a user profile for 'Bhuvaneshwari M mbhuv'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the following code:

```
1 Select last_name AS "EMPLOYEE",Salary AS "MONTHLY SALARY" from Employee where (Salary BETWEEN 5000 AND 12000) AND (dept_id IN(20,50)) order by last_name ;
```

The Results tab displays the output:

| EMPLOYEE | MONTHLY SALARY |
|----------|----------------|
| PRIYA    | 6000           |

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

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 Employe where hire\_date like '%1994'**

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'Bhuvaneshwari M mbhuv'. The main area has tabs for SQL Commands, Results (selected), Explain, Describe, Saved SQL, and History. The SQL command entered is: `1 Select last_name ,hire_date from Employe where hire_date like '%1994';`. The results table has columns LAST\_NAME and HIRE\_DATE, showing one row: PAVENDAN and 05/01/1994. A note at the bottom says '1 rows returned in 0.03 seconds'.

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\_title from Employe where manager\_id is NULL;**

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'Bhuvaneshwari M mbhuv'. The main area has tabs for SQL Commands, Results (selected), Explain, Describe, Saved SQL, and History. The SQL command entered is: `1 Select last_name ,job_title from Employe where manager_id is NULL;`. The results table has columns LAST\_NAME and JOB\_TITLE, showing two rows: PAVENDAN with Sales Manager and PRABHU with manager. A note at the bottom says '2 rows returned in 0.00 seconds'.

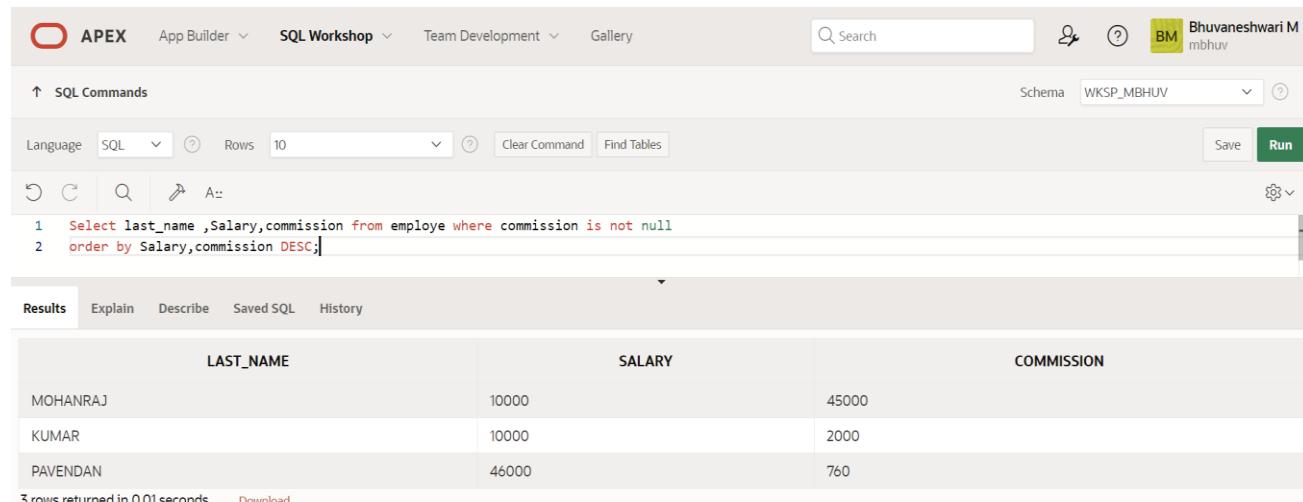
9. Display the last name, salary, and commission for all employees who earn commissions.

Sort data in descending order of salary and commissions.(hints: is not null,order by)

#### QUERY:

```
Select last_name ,Salary,commission from employe where commission is not null  
order by Salary,commission DESC;
```

#### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for Bhuvaneshwari M (bmhuv). The main area is titled "SQL Commands". The schema dropdown is set to WKSP\_MBHUV. The SQL editor contains the following code:

```
1 Select last_name ,Salary,commission from employe where commission is not null  
2 order by Salary,commission DESC;
```

The results tab is selected, displaying the output:

| LAST_NAME | SALARY | COMMISSION |
|-----------|--------|------------|
| MOHANRAJ  | 10000  | 45000      |
| KUMAR     | 10000  | 2000       |
| PAVENDAN  | 46000  | 760        |

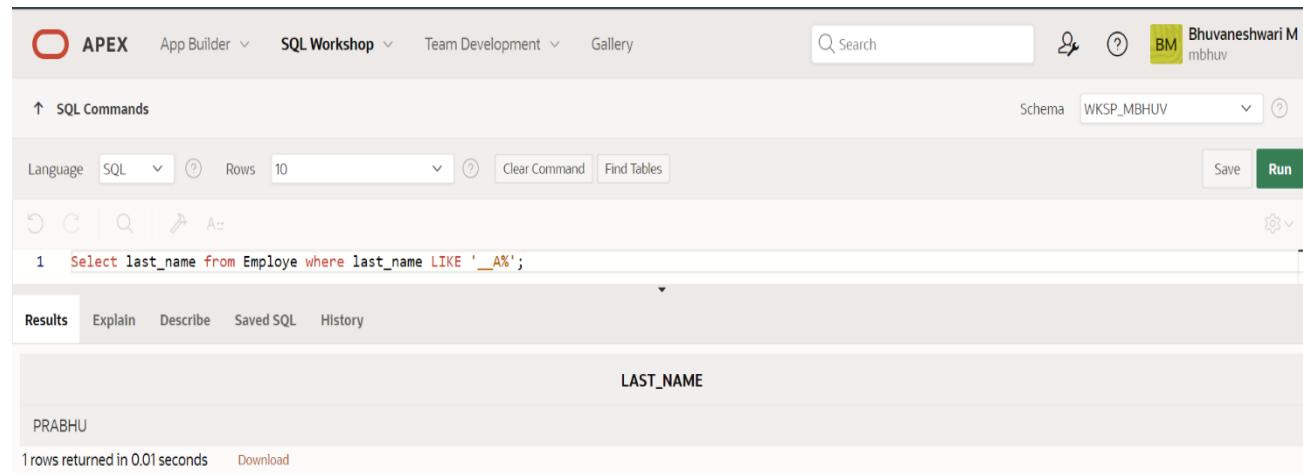
3 rows returned in 0.01 seconds [Download](#)

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

#### QUERY:

```
Select last_name from Employe where last_name LIKE '__A%';
```

#### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for Bhuvaneshwari M (bmhuv). The main area is titled "SQL Commands". The schema dropdown is set to WKSP\_MBHUV. The SQL editor contains the following code:

```
1 Select last_name from Employe where last_name LIKE '__A%';
```

The results tab is selected, displaying the output:

| LAST_NAME |
|-----------|
| PRABHU    |

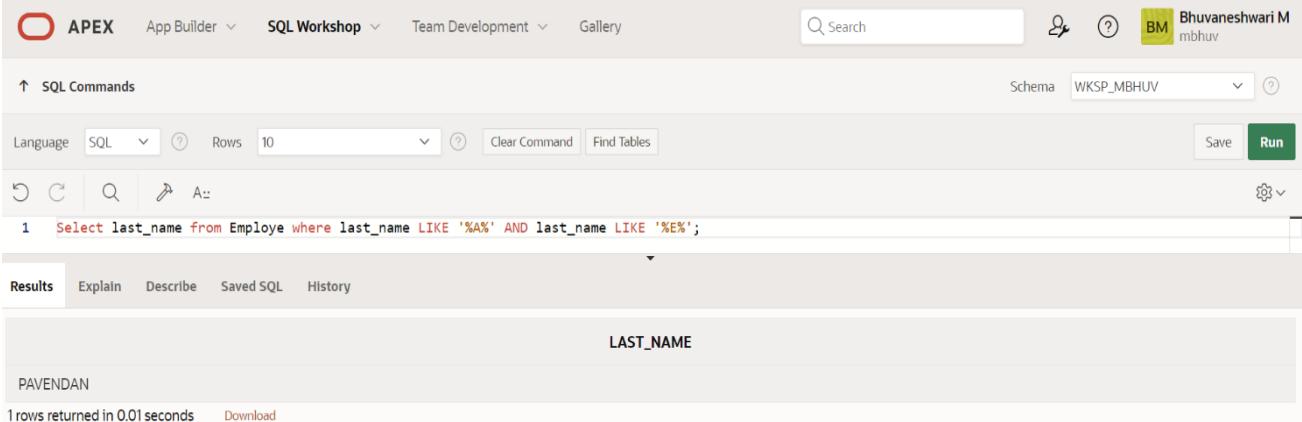
1 rows returned in 0.01 seconds [Download](#)

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 ,job\_title ,Salary from Employe where last\_name LIKE '%A%' AND last\_name LIKE '%E%';**

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows a user profile for 'Bhuvaneshwari M mbhuv'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_MBHV'. The SQL editor contains the query: 'Select last\_name from Employe where last\_name LIKE '%A%' AND last\_name LIKE '%E%';'. The results tab is selected, displaying a single row with the value 'PAVENDAN' under the column 'LAST\_NAME'. A note at the bottom says '1 rows returned in 0.01 seconds'.

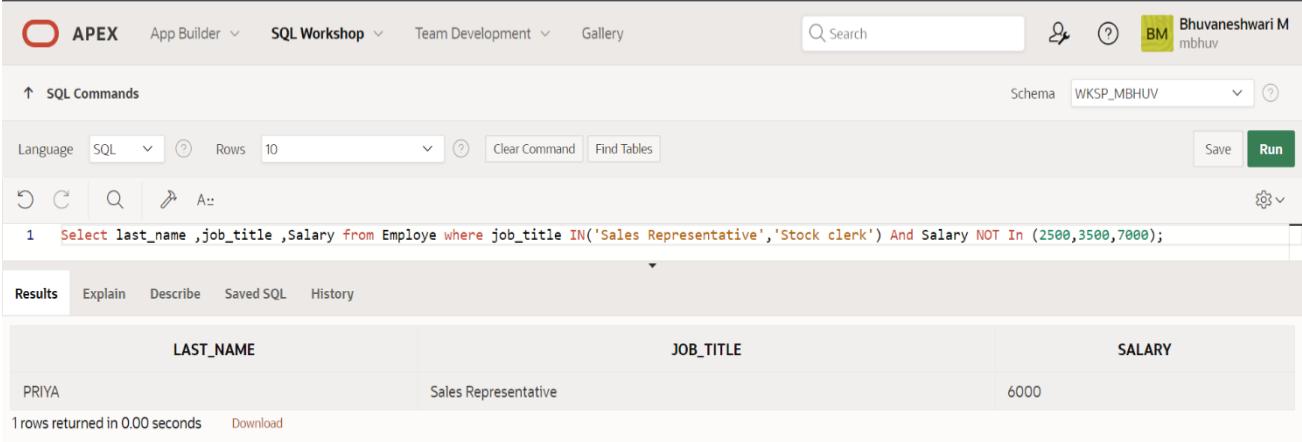
12. Display the last name and job and salary for all employees whose job is sales representative or stock clerk and whose salary is not equal to 2500 ,3500 or 7000.(hints:in,not in)

**QUERY:**

**Select last\_name AS "EMPLOYEE",Salary AS "MONTHLY SALARY" from Employe where**

**(Salary BETWEEN 5000 AND 12000) AND (dept\_id IN(20,50)) order by last\_name ;**

**OUTPUT:**



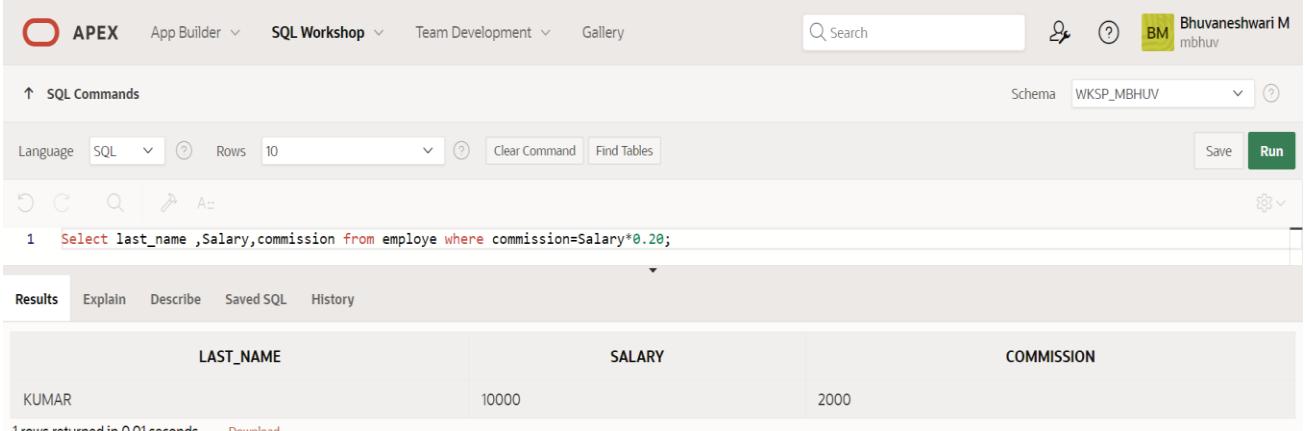
The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows a user profile for 'Bhuvaneshwari M mbhuv'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_MBHV'. The SQL editor contains the query: 'Select last\_name ,job\_title ,Salary from Employe where job\_title IN('Sales Representative','Stock clerk') And Salary NOT In (2500,3500,7000);'. The results tab is selected, displaying a single row with values 'PRIYA', 'Sales Representative', and '6000' under the columns 'LAST\_NAME', 'JOB\_TITLE', and 'SALARY' respectively. A note at the bottom says '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 from employe where commission=Salary\*0.20;**

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for Bhuvaneshwari M (mbhuv). The main area has tabs for SQL Commands, SQL (selected), Clear Command, and Find Tables. The SQL pane contains the query: "Select last\_name ,Salary,commission from employe where commission=Salary\*0.20;". The results pane shows a single row with columns LAST\_NAME, SALARY, and COMMISSION. The data is: KUMAR, 10000, 2000. A note at the bottom says "1 rows returned in 0.01 seconds".

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

**RESULT:**

The Queries for restricting and sorting data has been executed successfully.

# SINGLE ROW FUNCTIONS

**EX.NO.6**

**DATE:09/03/2024**

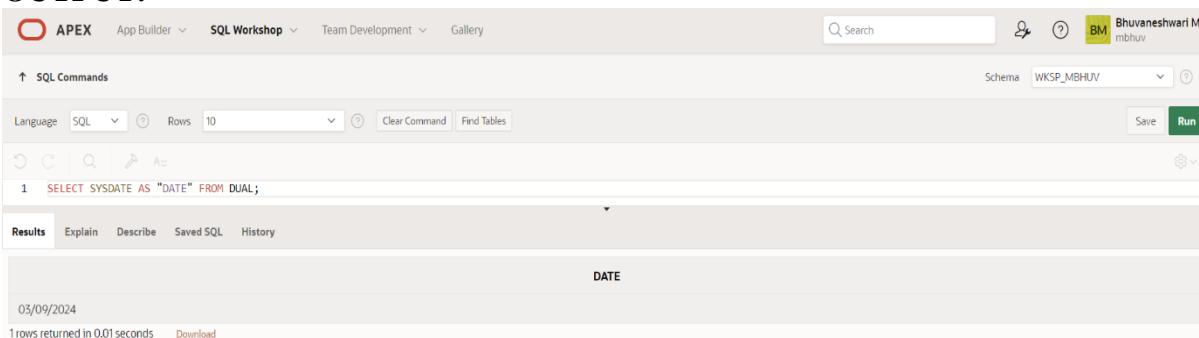
**Find the Solution for the following:**

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

**QUERY:**

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

**OUTPUT:**



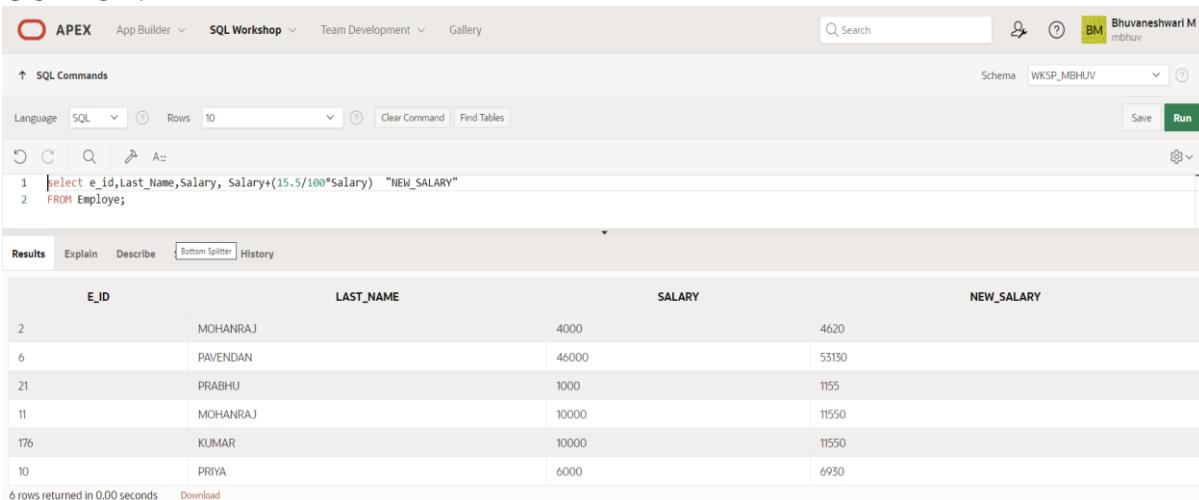
The screenshot shows the Oracle SQL Workshop interface. The command window contains the query: `1 SELECT SYSDATE AS "DATE" FROM DUAL;`. The results pane shows a single row with the column `DATE` containing the value `03/09/2024`.

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 e\_id, last\_name, Salary, Salary+(15.5/100\*Salary) "NEW\_SALARY"  
From Employe;**

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The command window contains the query: `1 select e_id,Last_Name,Salary, Salary+(15.5/100*Salary) "NEW_SALARY"  
2 FROM Employe;`. The results pane displays a table with columns `E_ID`, `LAST_NAME`, `SALARY`, and `NEW_SALARY`. The data is as follows:

| E_ID | LAST_NAME | SALARY | NEW_SALARY |
|------|-----------|--------|------------|
| 2    | MOHANRAJ  | 4000   | 4620       |
| 6    | PAVENDAN  | 46000  | 53150      |
| 21   | PRABHU    | 1000   | 1155       |
| 11   | MOHANRAJ  | 10000  | 11550      |
| 176  | KUMAR     | 10000  | 11550      |
| 10   | PRIYA     | 6000   | 6930       |

6 rows returned in 0.00 seconds

3. Modify your query lab\_03\_02.sql to add a column that subtracts the old salary from the new salary. Label the column Increase.

**QUERY:**

**SELECT e\_id, last\_name, Salary, (Salary+(Salary\*15.5/100))-Salary "Increase"  
From Employe;**

**OUTPUT:**

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

```
1 select e_id, last_name, salary, salary+(salary*15.5/100) "New Salary", (salary+(salary*15.5/100))-salary "Increse"
2 from employe;
```

The results table has columns: E\_ID, LAST\_NAME, SALARY, New Salary, and Increse. The data is:

| E_ID | LAST_NAME | SALARY | New Salary | Increse |
|------|-----------|--------|------------|---------|
| 2    | MOHANRAJ  | 4000   | 4620       | 620     |
| 6    | PAVENDAN  | 46000  | 53130      | 7130    |
| 21   | PRABHU    | 1000   | 1155       | 155     |
| 11   | MOHANRAJ  | 10000  | 11550      | 1550    |
| 176  | KUMAR     | 10000  | 11550      | 1550    |
| 10   | PRIYA     | 6000   | 6930       | 930     |

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

4. Write a query that displays the last name (with the first letter uppercase and all other letters lowercase) and the length of the last name 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 Employe  
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 SQL Workshop interface. The query entered is:

```
1 select initcap(last_name) "Name", length(last_name) "Length of Name"
2 from Employe
3 where last_name like 'J%' or last_name like 'A%' or last_name like 'M%'
4 order by last_name;
```

The results table has columns: Name and Length of Name. The data is:

| Name     | Length of Name |
|----------|----------------|
| Mohanraj | 8              |
| Mohanraj | 8              |

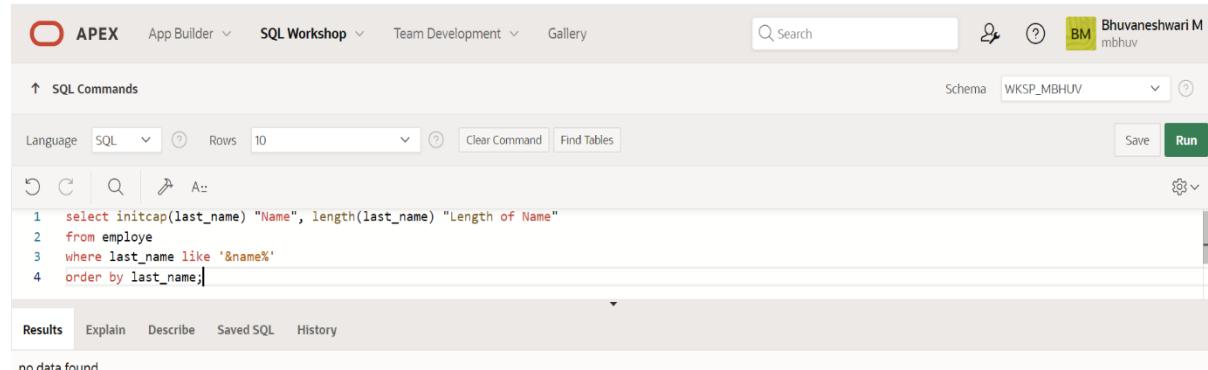
2 rows returned in 0.02 seconds [Download](#)

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

**QUERY:**

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

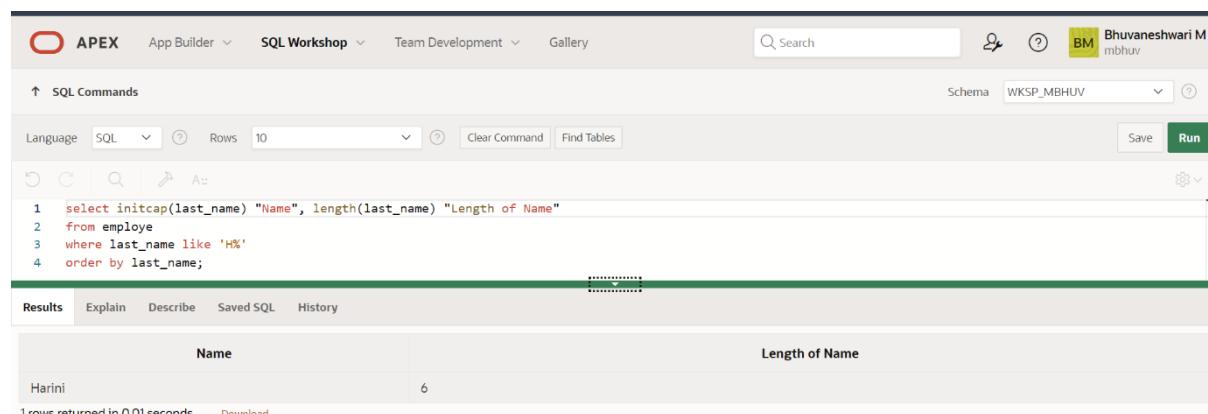
**OUTPUT:**



This 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 'Bhuvaneshwari M mbhuv'. The main area is titled 'SQL Commands' with a search bar and a schema dropdown set to 'WKSP\_MBHV'. The SQL editor contains the following code:

```
1 select initcap(last_name) "Name", length(last_name) "Length of Name"  
2 from employe  
3 where last_name like '&name%'  
4 order by last_name;
```

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



This screenshot shows the same Oracle SQL Workshop interface after running the query. The results tab now displays a single row of data:

| Name   | Length of Name |
|--------|----------------|
| Harini | 6              |

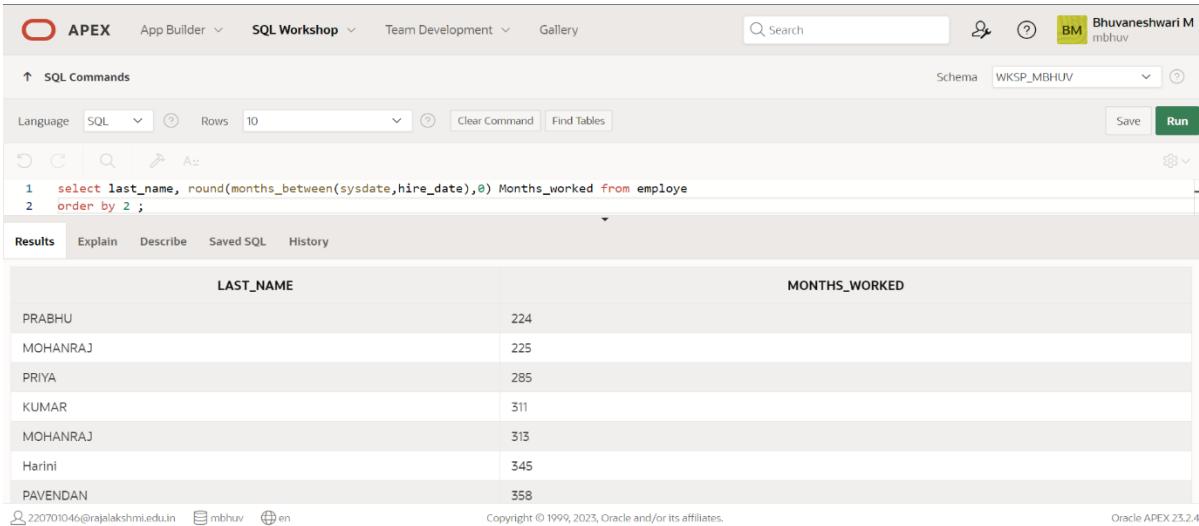
Below the table, it says '1 rows returned in 0.01 seconds' and has a 'Download' link.

5. The HR department wants to find the length of employment for each employee. For each employee, display the last name and calculate the number of months between today and the date on which the employee was hired. Label the column MONTHS\_WORKED. Order your results by the number of months employed. Round the number of months up to the closest whole number.

**QUERY:**

```
select last_name, round(months_between(sysdate,hire_date),0) Months_worked  
from employe order by 2;
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (which is selected), Team Development, and Gallery. A search bar and user profile are also present. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the executed query:

```
1 select last_name, round(months_between(sysdate,hire_date),0) Months_worked  
2 from employe order by 2;
```

The Results tab displays the output in a table format:

| LAST_NAME | MONTHS_WORKED |
|-----------|---------------|
| PRABHU    | 224           |
| MOHANRAJ  | 225           |
| PRIYA     | 285           |
| KUMAR     | 311           |
| MOHANRAJ  | 313           |
| Harini    | 345           |
| PAVENDAN  | 358           |

At the bottom of the interface, there are footer links for copyright information and Oracle APEX version 23.2.4.

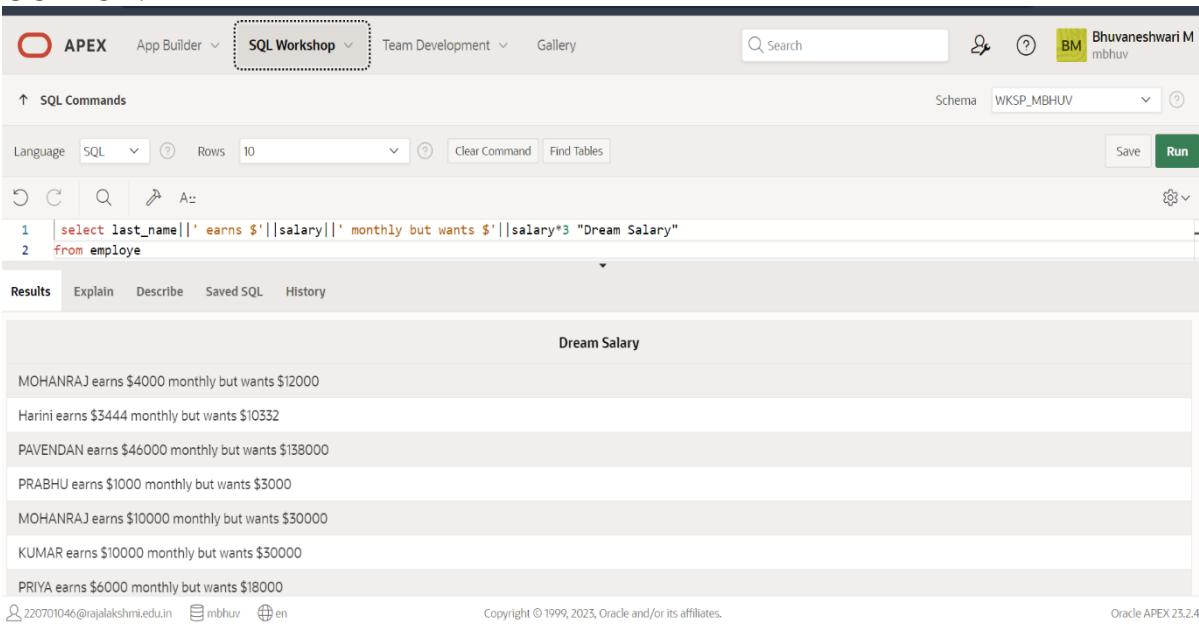
6. Create a report that produces the following for each employee:

<employee last name> earns <salary> monthly but wants <3 times salary>. Label the column Dream Salaries.

**QUERY:**

```
Select last_name||' earns $'||salary||' monthly but wants $'||salary*3 "Dream Salary"  
from employe
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface, identical to the previous one but with a different query. The Results tab displays the output:

| Dream Salary                                      |
|---|
| MOHANRAJ earns \$4000 monthly but wants \$12000   |
| Harini earns \$3444 monthly but wants \$10332     |
| PAVENDAN earns \$46000 monthly but wants \$138000 |
| PRABHU earns \$1000 monthly but wants \$3000      |
| MOHANRAJ earns \$10000 monthly but wants \$30000  |
| KUMAR earns \$10000 monthly but wants \$30000     |
| PRIYA earns \$6000 monthly but wants \$18000      |

At the bottom of the interface, there are footer links for copyright information and Oracle APEX version 23.2.4.

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

**QUERY:**

**Select last\_name, lpad(salary,15,'\$') Salary  
from employee;**

**OUTPUT:**

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

```
1 select last_name, lpad(salary,15,'$') Salary
2 | from employee;
```

The results table displays the last names and formatted salaries:

| LAST_NAME | SALARY                    |
|-----------|---------------------------|
| MOHANRAJ  | \$\$\$\$\$\$\$\$\$\$4000  |
| Harini    | \$\$\$\$\$\$\$\$\$\$3444  |
| PAVENDAN  | \$\$\$\$\$\$\$\$\$\$46000 |
| PRABHU    | \$\$\$\$\$\$\$\$\$\$1000  |
| MOHANRAJ  | \$\$\$\$\$\$\$\$\$\$10000 |
| KUMAR     | \$\$\$\$\$\$\$\$\$\$10000 |
| PRIYA     | \$\$\$\$\$\$\$\$\$\$6000  |

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

**QUERY:**

**select last\_name, hire\_date, to\_char((next\_day(hire\_date,'Monday')),'fmday," the "ddspth  
"of" month,yyyy') "REVIEW" from employee;**

**OUTPUT:**

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

```
1 select last_name, hire_date, to_char((next_day(hire_date,'Monday')),'fmday," the "ddspth
2 | "of" month,yyyy') "REVIEW"
3 | from employee;
```

The results table displays the last names, hire dates, and formatted review dates:

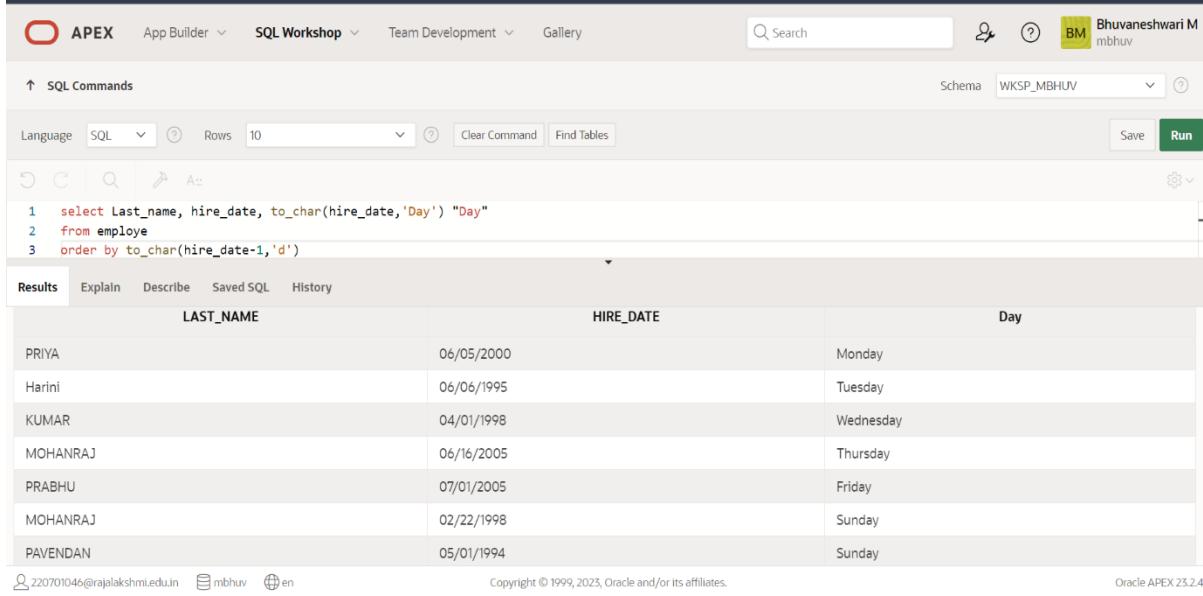
| LAST_NAME | HIRE_DATE  | REVIEW                                    |
|-----------|------------|---|
| MOHANRAJ  | 02/22/1998 | monday, the twenty-third of february,1998 |
| Harini    | 06/06/1995 | monday, the twelfth of june,1995          |
| PAVENDAN  | 05/01/1994 | monday, the second of may,1994            |
| PRABHU    | 07/01/2005 | monday, the fourth of july,2005           |
| MOHANRAJ  | 06/16/2005 | monday, the twentieth of june,2005        |
| KUMAR     | 04/01/1998 | monday, the sixth of april,1998           |
| PRIYA     | 06/05/2000 | monday, the twelfth of june,2000          |

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

**QUERY:**

```
Select Last_name, hire_date, to_char(hire_date,'Day') "Day"  
from employe  
order by to_char(hire_date-1,'d')
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The query is executed successfully, displaying the following results:

| LAST_NAME | HIRE_DATE  | Day       |
|-----------|------------|-----------|
| PRIYA     | 06/05/2000 | Monday    |
| Harini    | 06/06/1995 | Tuesday   |
| KUMAR     | 04/01/1998 | Wednesday |
| MOHANRAJ  | 06/16/2005 | Thursday  |
| PRABHU    | 07/01/2005 | Friday    |
| MOHANRAJ  | 02/22/1998 | Sunday    |
| PAVENDAN  | 05/01/1994 | Sunday    |

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

**RESULT:**

The Queries for Single row functions has been executed successfully.

## DISPLAYING DATA FROM MULTIPLE TABLES

EX.NO:7

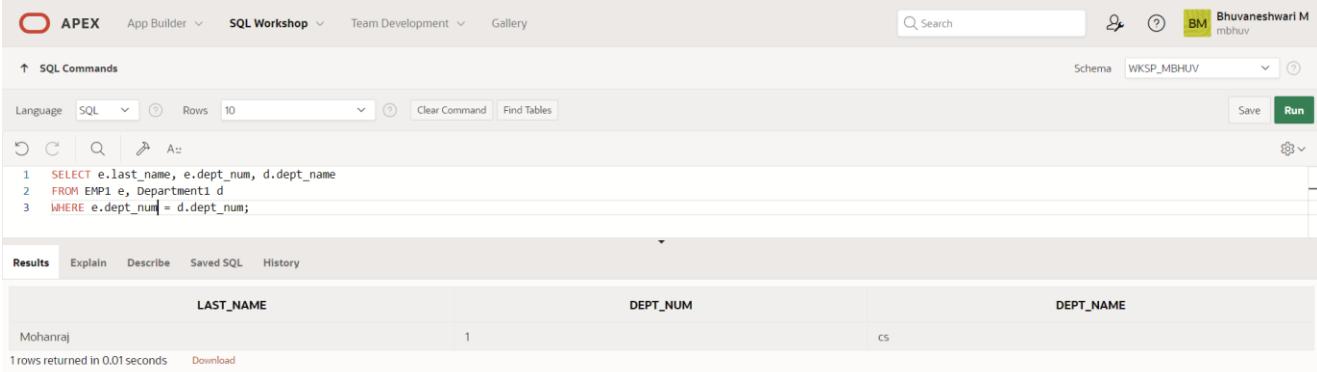
DATE:14/03/2024

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

**QUERY:**

```
SELECT e.last_name, e.dept_num, d.dept_name
FROM EMP1 e, Department1 d
WHERE e.dept_num = d.dept_num;
```

**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 'Bhuvaneshwari M mbhuv'. The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the query is pasted:

```
1 SELECT e.last_name, e.dept_num, d.dept_name
2 FROM EMP1 e, Department1 d
3 WHERE e.dept_num = d.dept_num;
```

Under 'Results', the output is displayed in a table:

| LAST_NAME | DEPT_NUM | DEPT_NAME |
|-----------|----------|-----------|
| Mohanraj  | 1        | CS        |

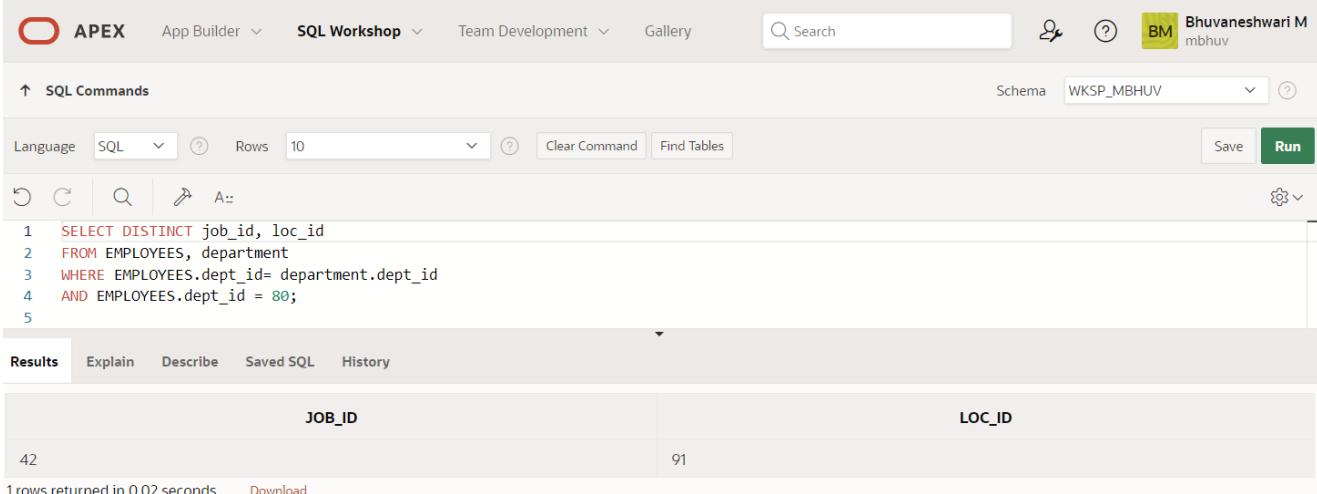
1 rows returned in 0.01 seconds [Download](#)

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, department
WHERE employees.dept_id = department.dept_id
AND employees.dept_id = 80;
```

**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 'Bhuvaneshwari M mbhuv'. The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the query is pasted:

```
1 SELECT DISTINCT job_id, loc_id
2 FROM EMPLOYEES, department
3 WHERE EMPLOYEES.dept_id= department.dept_id
4 AND EMPLOYEES.dept_id = 80;
5
```

Under 'Results', the output is displayed in a table:

| JOB_ID | LOC_ID |
|--------|--------|
| 42     | 91     |

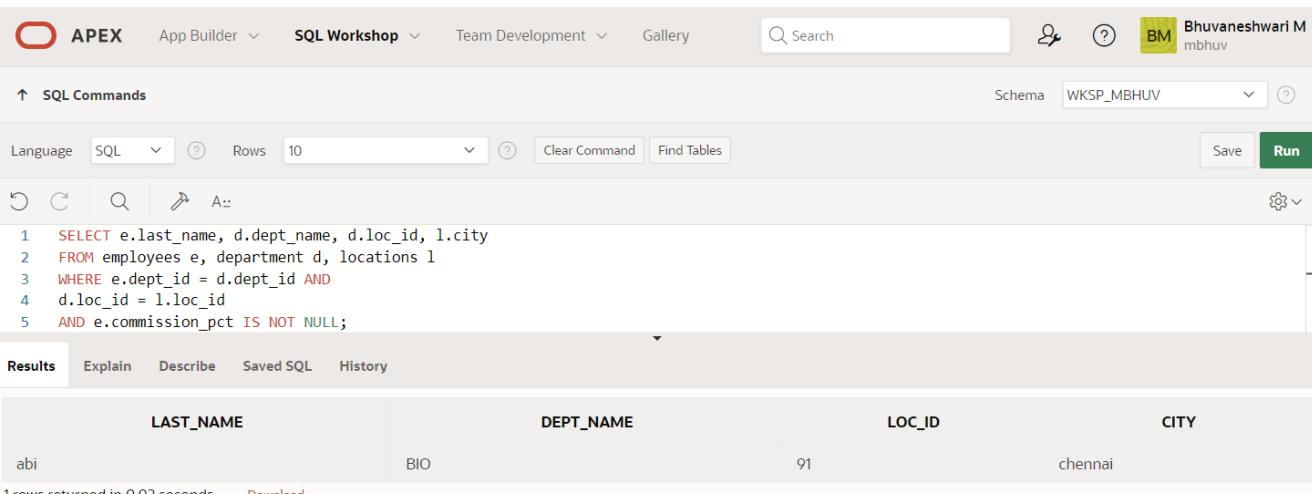
1 rows returned in 0.02 seconds [Download](#)

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 employees e, department d, locations l
WHERE e.dept_id = d.dept_id
AND
d.loc_id = l.loc_id
AND e.commission_pct IS NOT NULL;
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. A search bar and user profile 'Bhuvaneshwari M mbhuv' are also present. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_MBHV'. The code editor contains the following SQL query:

```
1 SELECT e.last_name, d.dept_name, d.location_id, l.city
2 FROM employees e, department d, locations l
3 WHERE e.dept_id = d.dept_id AND
4 d.loc_id = l.loc_id
5 AND e.commission_pct IS NOT NULL;
```

The results tab shows the output:

| LAST_NAME | DEPT_NAME | LOC_ID | CITY    |
|-----------|-----------|--------|---------|
| abi       | BIO       | 91     | chennai |

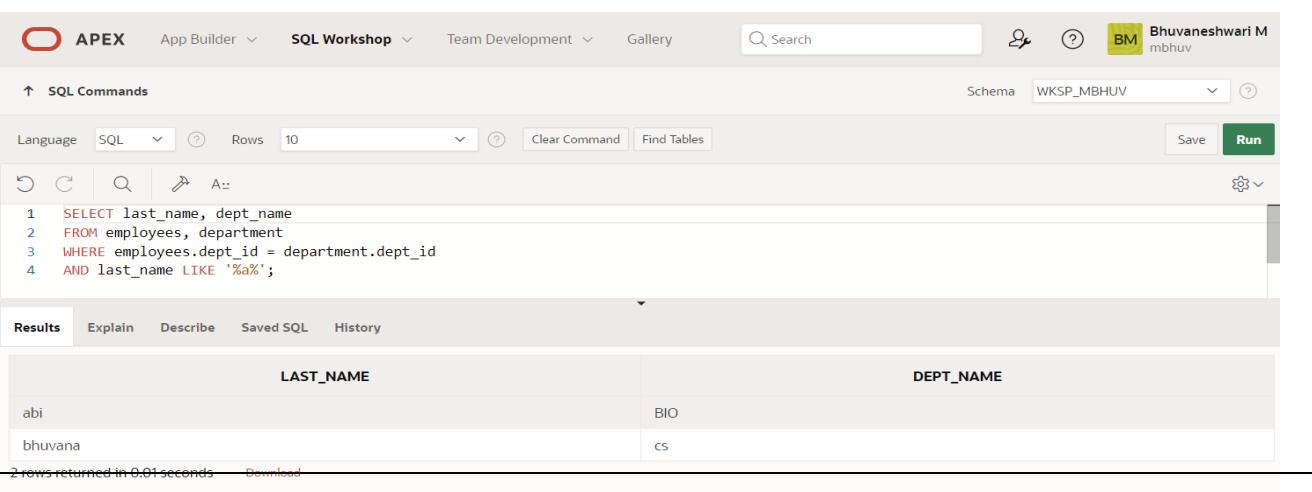
1 rows returned in 0.02 seconds [Download](#)

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface, identical to the previous one but with a different query. The top navigation bar and user profile are the same. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_MBHV'. The code editor contains the following SQL query:

```
1 SELECT last_name, dept_name
2 FROM employees, department
3 WHERE employees.dept_id = department.dept_id
4 AND last_name LIKE '%a%';
```

The results tab shows the output:

| LAST_NAME | DEPT_NAME |
|-----------|-----------|
| abi       | BIO       |
| bhuvana   | cs        |

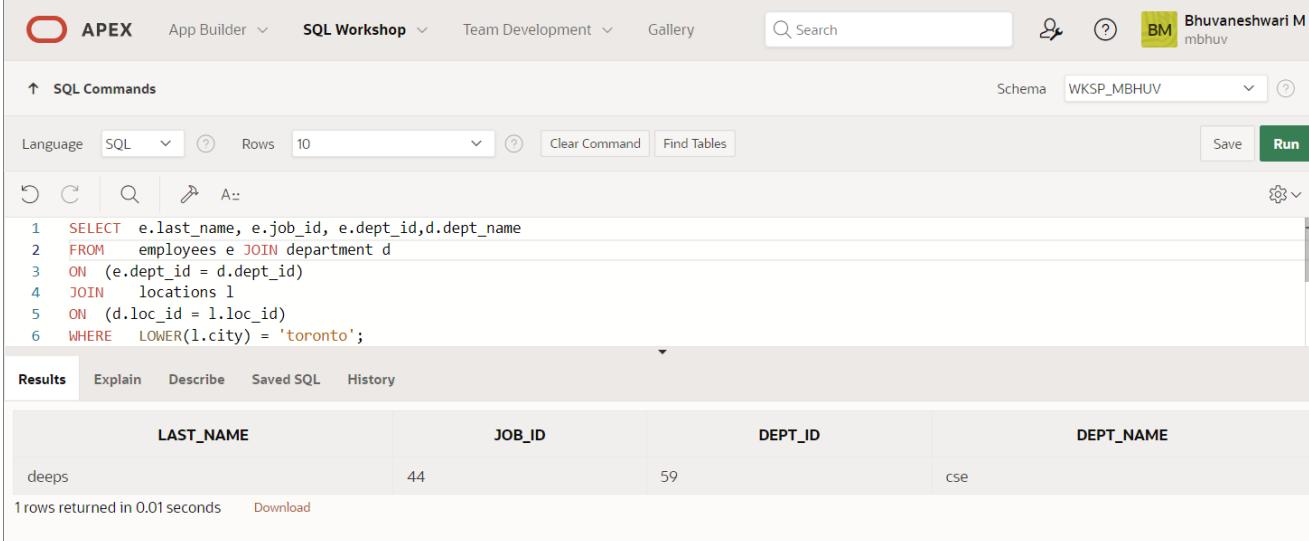
2 rows returned in 0.01 seconds [Download](#)

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

**QUERY:**

```
SELECT e.last_name, e.job_id, e.dept_id, d.department_name
FROM employees e JOIN department d
ON (e.dept_id = d.dept_id)
JOIN location l
ON (d.loc_id = l.loc_id)
WHERE LOWER(l.city) = 'toronto';
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. A search bar and user profile are also present. The main area is titled "SQL Commands". The language is set to SQL, and the schema is WKSP\_MBHV. The query entered is:

```
1 SELECT e.last_name, e.job_id, e.dept_id, d.department_name
2 FROM employees e JOIN department d
3 ON (e.dept_id = d.dept_id)
4 JOIN locations l
5 ON (d.loc_id = l.loc_id)
6 WHERE LOWER(l.city) = 'toronto';
```

The results tab is selected, displaying the output:

| LAST_NAME | JOB_ID | DEPT_ID | DEPT_NAME |
|-----------|--------|---------|-----------|
| deeps     | 44     | 59      | cse       |

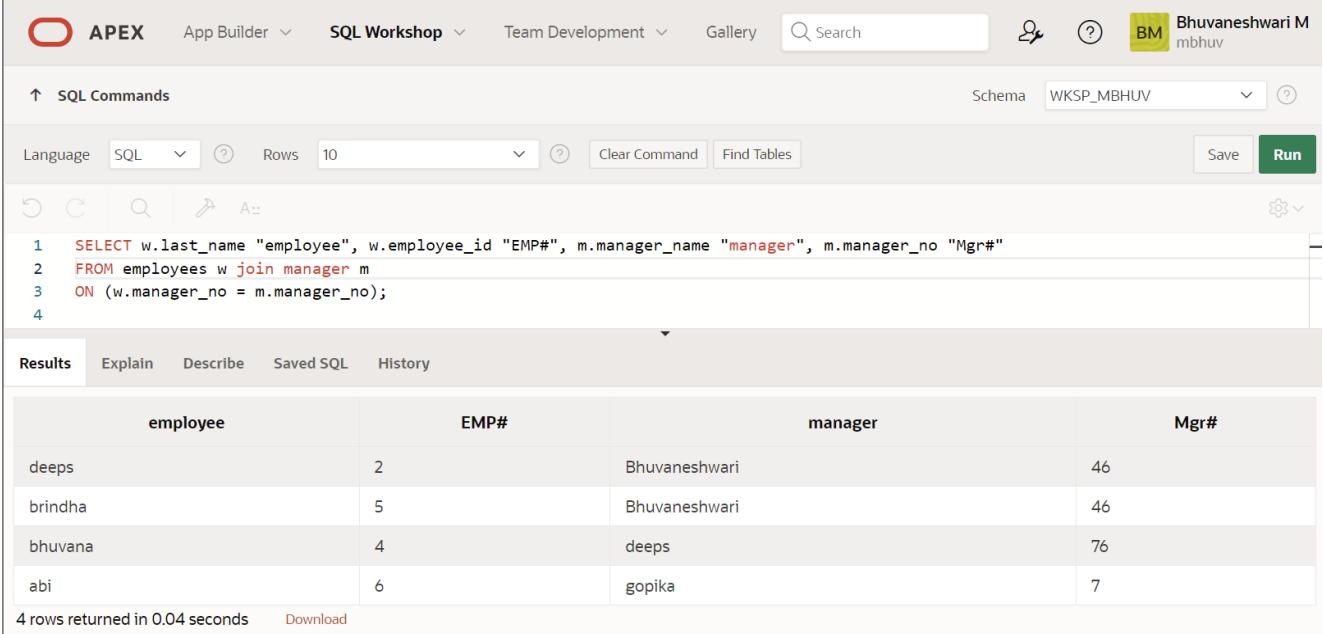
1 rows returned in 0.01 seconds [Download](#)

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.manager_name  
"manager", m.manager_no "Mgr#"  
FROM employee w join manager m  
ON (w.manager_no = m.manager_no);
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, a search bar, and a user profile for Bhuvaneshwari M mbhuv.

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

```
1  SELECT w.last_name "employee", w.employee_id "EMP#", m.manager_name "manager", m.manager_no "Mgr#"  
2  FROM employees w join manager m  
3  ON (w.manager_no = m.manager_no);  
4
```

The Results tab is selected, displaying the query results in a grid format:

| employee | EMP# | manager       | Mgr# |
|----------|------|---------------|------|
| deep     | 2    | Bhuvaneshwari | 46   |
| brindha  | 5    | Bhuvaneshwari | 46   |
| bhuvana  | 4    | deep          | 76   |
| abi      | 6    | gopika        | 7    |

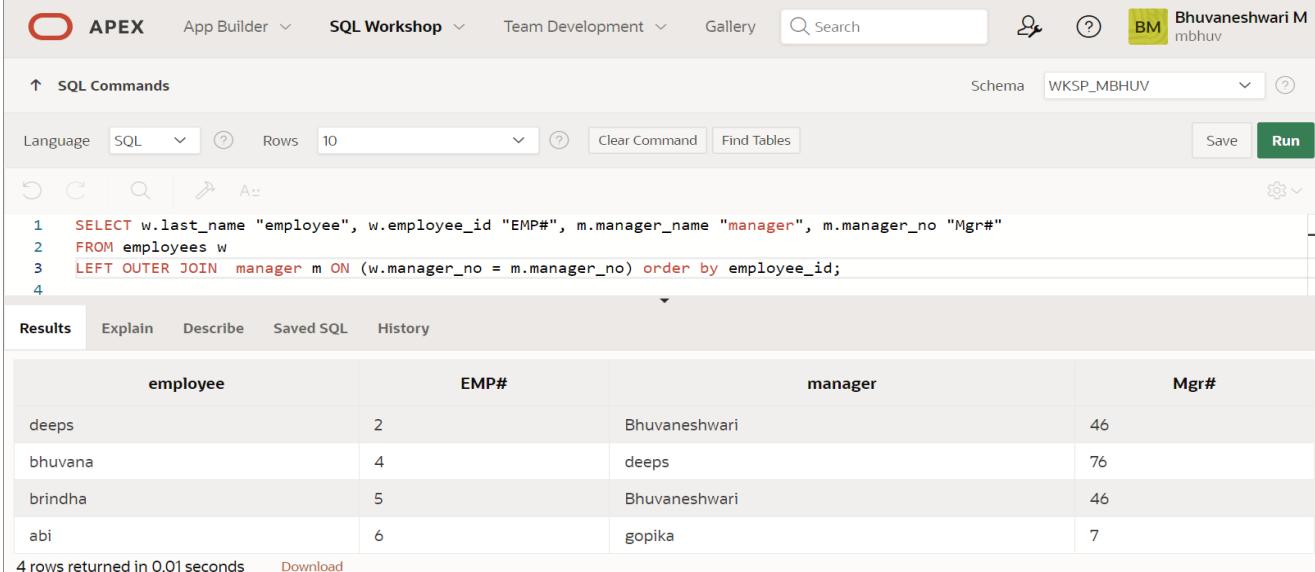
Below the table, it says "4 rows returned in 0.04 seconds" and there is a "Download" link.

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.manager_name "manager", m.manager_no "Mgr#"  
FROM employee w  
LEFT OUTER JOIN manager m ON (w.manager_no = m.manager_no)  
order by employee_id;
```

#### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, a search bar, and user information (Bhuvaneshwari M mbhuv). The main area is titled 'SQL Commands' with tabs for Language (SQL selected), Rows (10), Clear Command, Find Tables, Save, and Run. The SQL command entered is:

```
1 SELECT w.last_name "employee", w.employee_id "EMP#", m.manager_name "manager", m.manager_no "Mgr#"  
2 FROM employees w  
3 LEFT OUTER JOIN manager m ON (w.manager_no = m.manager_no) order by employee_id;  
4
```

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

| employee | EMP# | manager       | Mgr# |
|----------|------|---------------|------|
| deeps    | 2    | Bhuvaneshwari | 46   |
| bhuvana  | 4    | deeps         | 76   |
| brindha  | 5    | Bhuvaneshwari | 46   |
| abi      | 6    | gopika        | 7    |

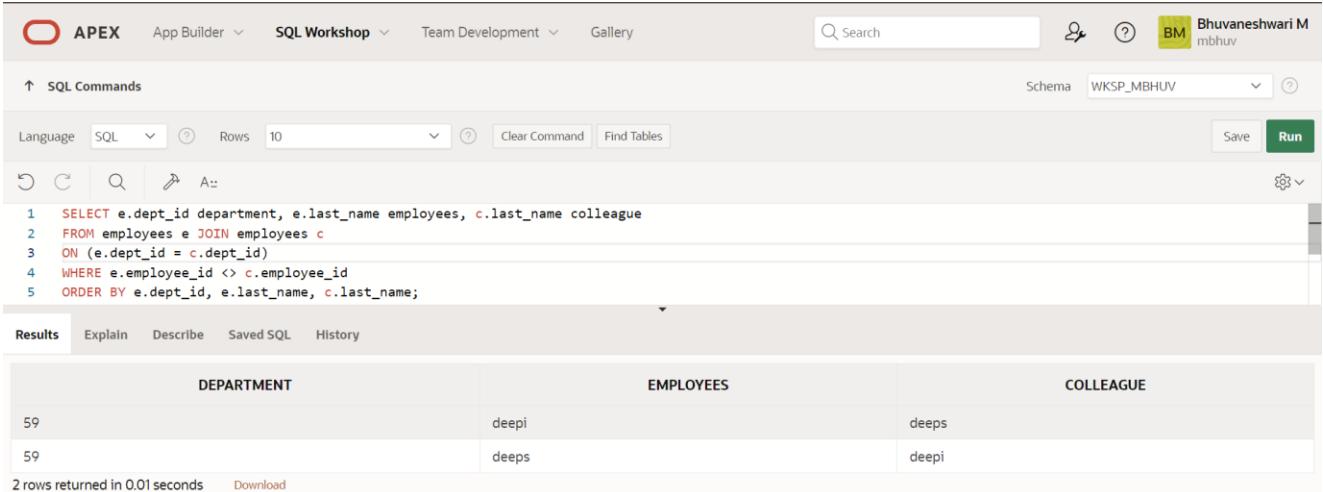
Below the table, it says '4 rows returned in 0.01 seconds' and there is a 'Download' link.

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, c.last_name colleague
FROM employee e JOIN employee c
ON (e.dept_id = c.dept_id)
WHERE e.emp_id <> c.emp_id
ORDER BY e.dept_id, e.last_name, c.last_name;
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, user information for 'Bhuvaneshwari M mbhuv', and a 'Run' button. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the executed query. The Results tab displays the output in a grid format.

| DEPARTMENT | EMPLOYEES | COLLEAGUE |
|------------|-----------|-----------|
| 59         | deepi     | deeps     |
| 59         | deeps     | deepi     |

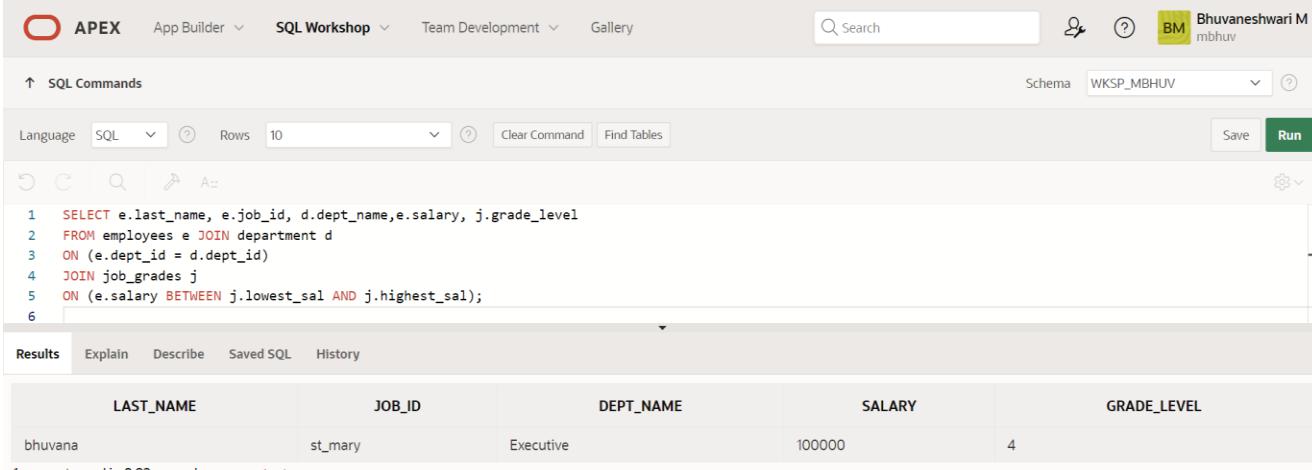
2 rows returned in 0.01 seconds [Download](#)

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

**QUERY:**

```
SELECT e.last_name, e.job_id, d.department_name, e.salary, j.grade_level
FROM employee e JOIN department d
ON (e.dept_id = d.dept_id)
JOIN job_grades j
ON (e.salary BETWEEN j.lowest_sal AND j.highest_sal);
```

**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 Bhuvaneshwari M (mbhuv). The schema is set to WKSP\_MBHUV. The main area shows the SQL command entered:

```
1 SELECT e.last_name, e.job_id, d.department_name, e.salary, j.grade_level
2 FROM employee e JOIN department d
3 ON (e.dept_id = d.dept_id)
4 JOIN job_grades j
5 ON (e.salary BETWEEN j.lowest_sal AND j.highest_sal);
6
```

The results tab is selected, displaying the following output:

| LAST_NAME | JOB_ID  | DEPT_NAME | SALARY | GRADE_LEVEL |
|-----------|---------|-----------|--------|-------------|
| bhuvana   | st_mary | Executive | 100000 | 4           |

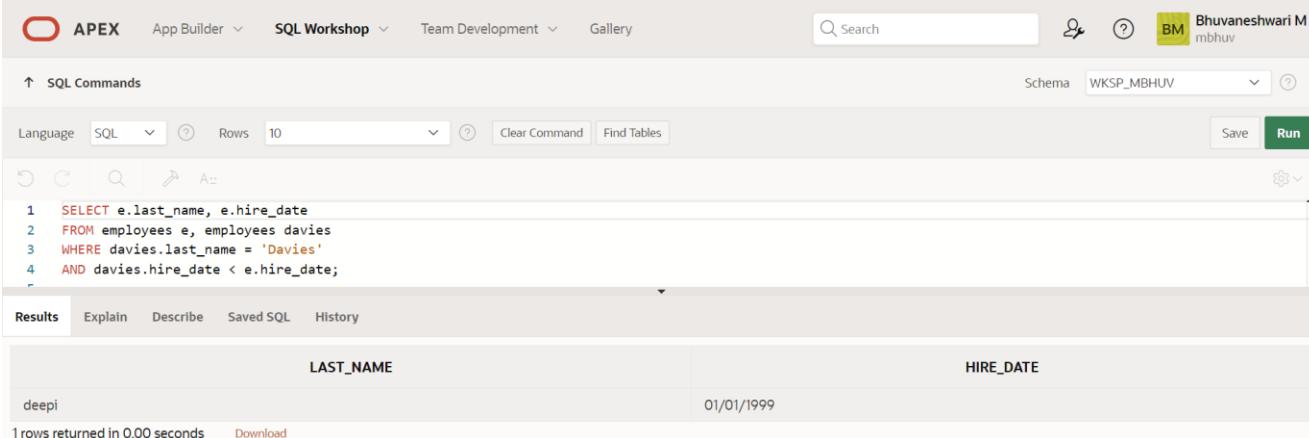
1 rows returned in 0.02 seconds [Download](#)

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows a user profile for Bhuvaneshwari M (mbhuv). The main area is titled 'SQL Commands' with a sub-tab 'Results'. The SQL editor contains the following query:

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

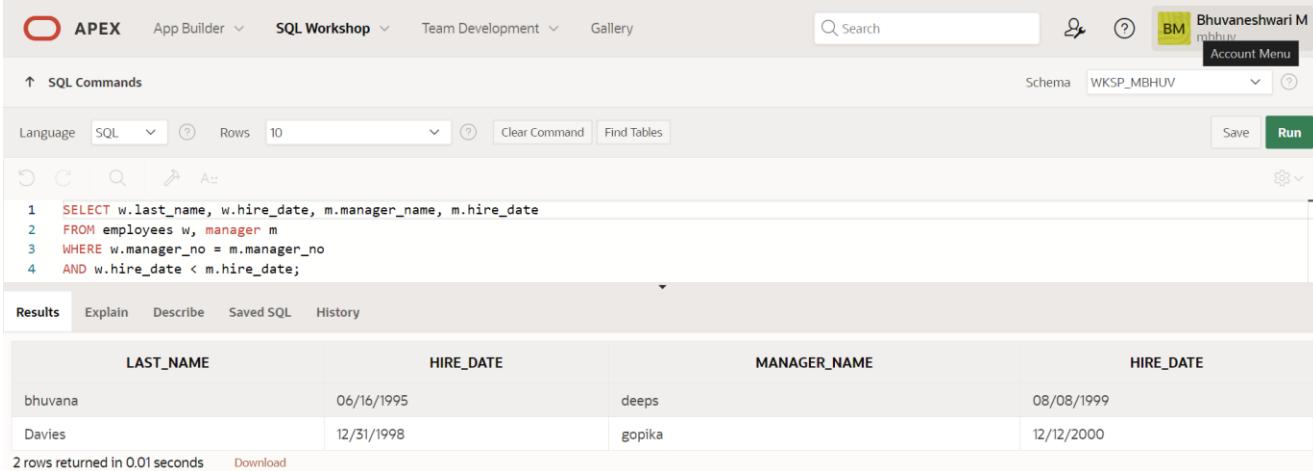
The results table has two columns: LAST\_NAME and HIRE\_DATE. One row is returned, showing 'deepi' in the LAST\_NAME column and '01/01/1999' in the HIRE\_DATE column. The status bar at the bottom indicates '1 rows returned in 0.00 seconds'.

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.manager_name, m.hire_date
FROM employee w, manager m
WHERE w.manager_id = m.manager_id
AND w.hire_date < m.hire_date;
```

**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 logged in as Bhuvaneshwari M with the ID mbhuv. The schema is set to WKSP\_MBHV. The main area shows the SQL command entered:

```
1 SELECT w.last_name, w.hire_date, m.manager_name, m.hire_date
2 FROM employees w, manager m
3 WHERE w.manager_no = m.manager_no
4 AND w.hire_date < m.hire_date;
```

The results tab displays the output:

| LAST_NAME | HIRE_DATE  | MANAGER_NAME | HIRE_DATE  |
|-----------|------------|--------------|------------|
| bhuvana   | 06/16/1995 | deepa        | 08/08/1999 |
| Davies    | 12/31/1998 | gopika       | 12/12/2000 |

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

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

**RESULT:**

The Queries for displaying data from multiple tables has been executed successfully.

## AGGREGATING DATA USING GROUP FUNCTIONS

EX.NO.8

DATE:24/03/2024

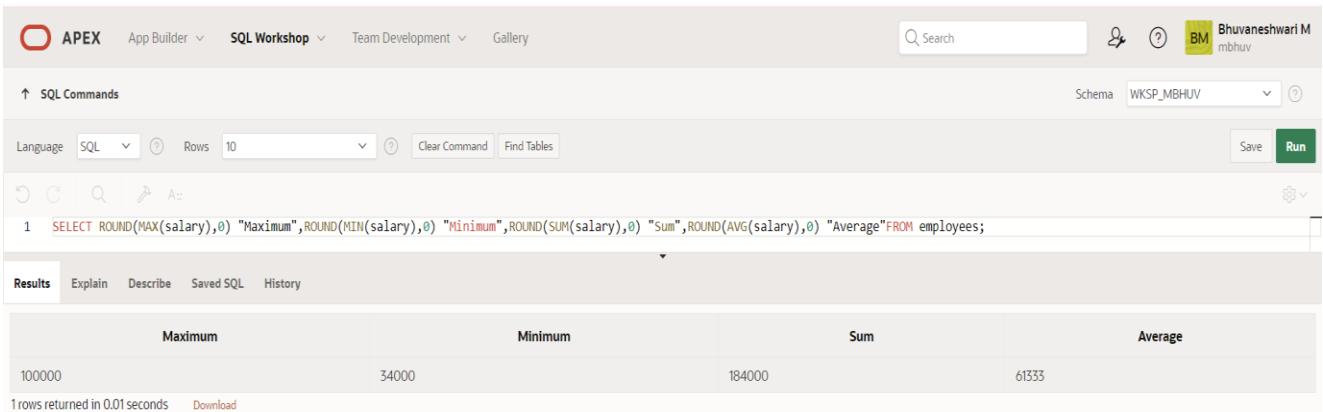
The HR department needs the following reports:

- 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 employees;
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon, and the schema name WKSP\_MBHV. The main area is titled 'SQL Commands' with a 'Run' button. The SQL editor contains the query:

```
1 SELECT ROUND(MAX(salary),0) "Maximum",  
2     ROUND(MIN(salary),0) "Minimum",  
3     ROUND(SUM(salary),0) "Sum",  
4     ROUND(AVG(salary),0) "Average"  
5  FROM employees;
```

The results tab is selected, displaying the following output:

|  | Maximum | Minimum | Sum    | Average |
|--|---------|---------|--------|---------|
|  | 100000  | 34000   | 184000 | 61333   |

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

2. 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 employees
GROUP BY job_id;
```

**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 identified as Bhuvaneshwari M mbhuv. The SQL Commands tab is active, displaying the following SQL code:

```
1 SELECT job_id, ROUND(MAX(salary),0) "Maximum",
2 ROUND(MIN(salary),0) "Minimum",
3 ROUND(SUM(salary),0) "Sum",
4 ROUND(AVG(salary),0) "Average"
5 FROM employees
```

The Results tab displays the output of the query:

| JOB_ID    | Maximum | Minimum | Sum    | Average |
|-----------|---------|---------|--------|---------|
| st_joseph | 24000   | 3200    | 27200  | 13600   |
| st_mary   | 100000  | 100000  | 100000 | 100000  |
| st_clerk  | 50000   | 12222   | 96222  | 32074   |

Below the results, it says "3 rows returned in 0.05 seconds" and there is a "Download" link.

3. 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 employees
WHERE job_id = '&JOBID' GROUP BY job_id;
```

**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 identified as Bhuvaneshwari M mbhuv. The SQL Commands tab is active, displaying the following SQL code:

```
1 SELECT job_id, COUNT(*) FROM employees WHERE job_id = '42' GROUP BY job_id;
2
```

The Results tab displays the output of the query:

| JOB_ID | COUNT(*) |
|--------|----------|
| 42     | 2        |

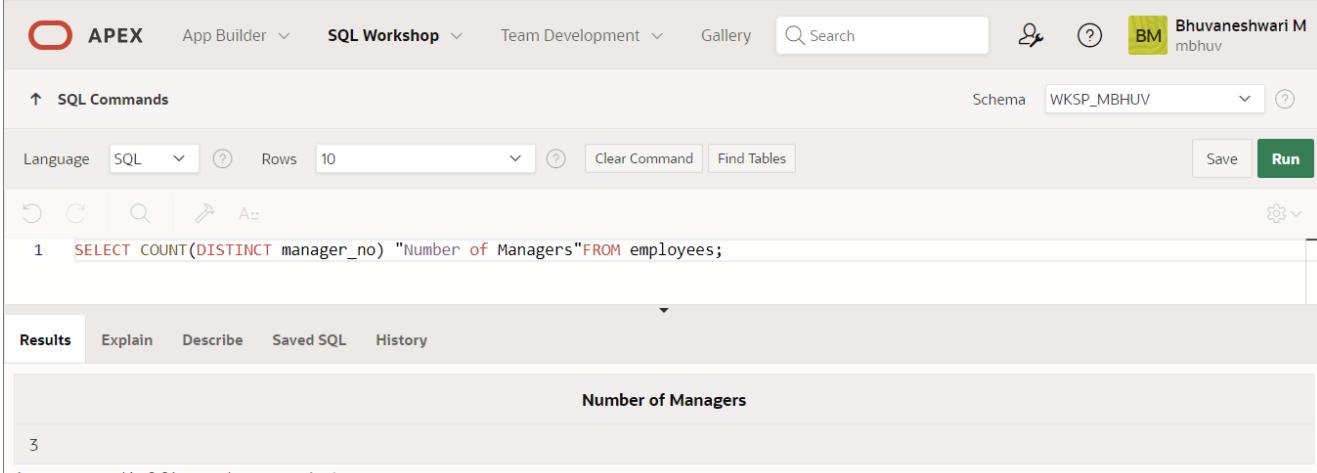
Below the results, it says "1 rows returned in 0.00 seconds" and there is a "Download" link.

4. 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_no) "Number of Managers"  
FROM employees;
```

**OUTPUT:**



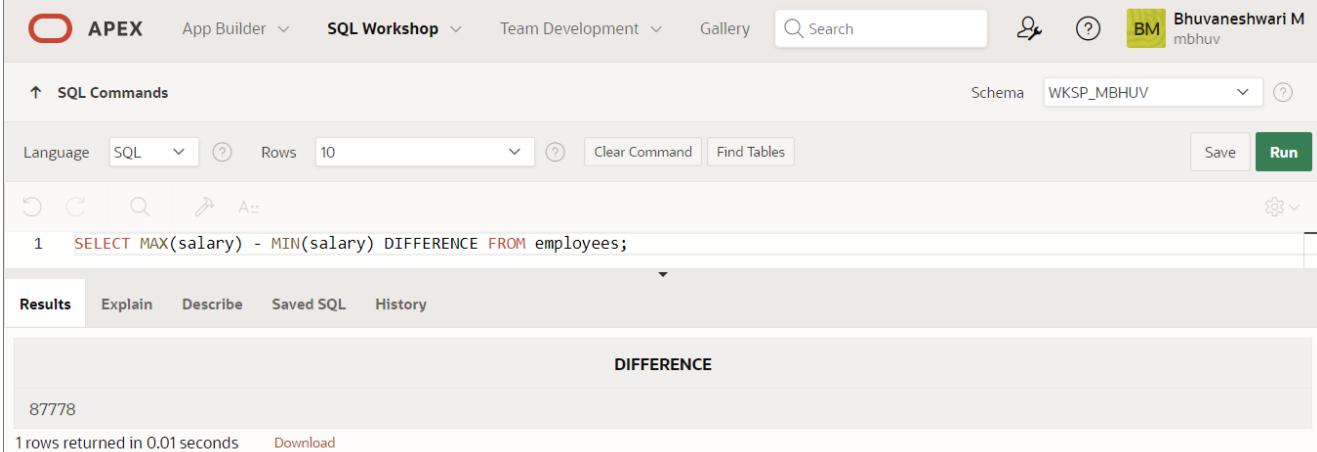
The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, and a search bar. The user is signed in as Bhuvaneshwari M (mbhuv). The SQL Commands panel shows the query: `SELECT COUNT(DISTINCT manager_no) "Number of Managers" FROM employees;`. The Results tab displays the output: `Number of Managers` with a value of `3`. The status message indicates `1 rows returned in 0.01 seconds`.

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

**QUERY:**

```
SELECT MAX(Salary)- MIN(Salary) DIFFERENCE FROM employees;
```

**OUTPUT:**



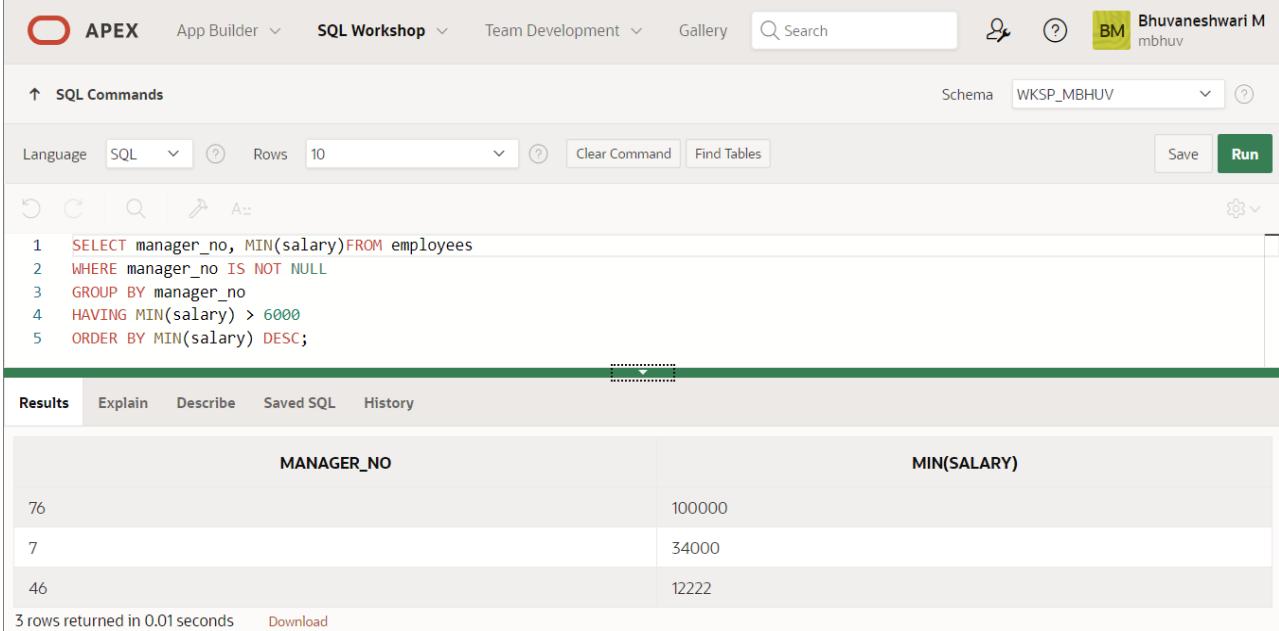
The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, and a search bar. The user is signed in as Bhuvaneshwari M (mbhuv). The SQL Commands panel shows the query: `SELECT MAX(salary) - MIN(salary) DIFFERENCE FROM employees;`. The Results tab displays the output: `DIFFERENCE` with a value of `87778`. The status message indicates `1 rows returned in 0.01 seconds`.

6. 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_no, MIN(salary)FROM employees  
WHERE manager_no IS NOT NULL  
GROUP BY manager_no  
HAVING MIN(salary) > 6000  
ORDER BY MIN(salary) DESC;
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, a search bar, and a user profile for Bhuvaneshwari M (mbhuv). The SQL Workshop tab is active. Below the toolbar, the schema is set to WKSP\_MBHUV. The main area contains the SQL command text and its execution results.

**SQL Commands:**

```
1 SELECT manager_no, MIN(salary)FROM employees  
2 WHERE manager_no IS NOT NULL  
3 GROUP BY manager_no  
4 HAVING MIN(salary) > 6000  
5 ORDER BY MIN(salary) DESC;
```

**Results:**

| MANAGER_NO | MIN(SALARY) |
|------------|-------------|
| 76         | 100000      |
| 7          | 34000       |
| 46         | 12222       |

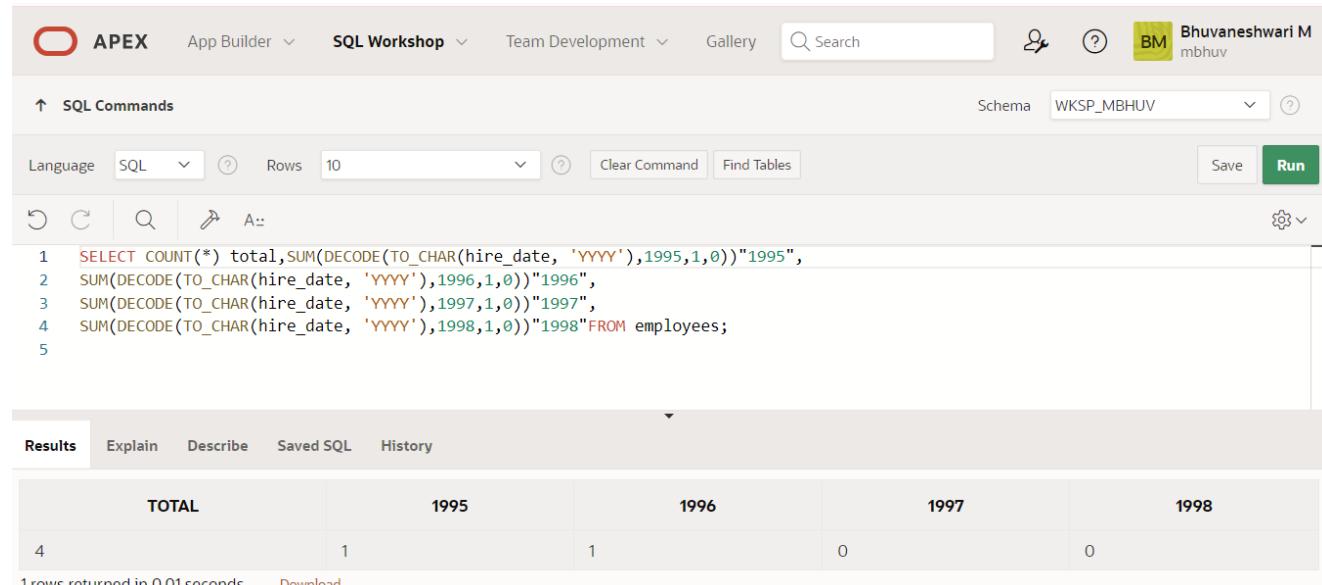
3 rows returned in 0.01 seconds    [Download](#)

7. 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 employees;
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, a search bar, and user information for Bhuvaneshwari M. The main area is titled 'SQL Commands' and contains the executed SQL code. Below the code, the 'Results' tab is selected, displaying a grid with the query's output. The output shows a single row with the total count of employees and the counts for each year from 1995 to 1998.

|   | TOTAL | 1995 | 1996 | 1997 | 1998 |
|---|-------|------|------|------|------|
| 4 | 1     | 1    | 0    | 0    |      |

1 rows returned in 0.01 seconds    Download

8. 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 employees GROUP BY job_id;
```

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and user information 'Bhuvaneshwari M mbhuv'. Below the navigation is a toolbar with icons for undo, redo, search, and refresh, followed by 'Schema' set to 'WKSP\_MBHV', 'Save', and 'Run' buttons. The main area is titled 'SQL Commands' and contains the executed SQL code. The results tab is selected, displaying a matrix output with columns for Job, Dept 20, Dept 50, Dept 80, Dept 90, and Total. The data shows three rows corresponding to job IDs 44, 42, and 40.

| Job | Dept 20 | Dept 50 | Dept 80 | Dept 90 | Total  |
|-----|---------|---------|---------|---------|--------|
| 44  | -       | -       | -       | -       | 50000  |
| 42  | -       | -       | 34000   | -       | 46222  |
| 40  | -       | -       | -       | -       | 100000 |

3 rows returned in 0.01 seconds    Download

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user profile 'Bhuvaneshwari M mbhuv'. The main area has tabs for SQL Commands, Language (SQL selected), Rows (10), Clear Command, Find Tables, Save, and Run. The code editor contains the query from above. The Results tab is selected, displaying the output:

| dept_name | department location | Number of people | salary |
|-----------|---------------------|------------------|--------|
| cse       | Tamil nadu          | 1                | 3200   |
| BIO       | Banglore            | 1                | 12222  |
| Executive | italy               | 1                | 100000 |

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

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

**RESULT:** The Queries for aggregating data using single row functions has been executed successfully.

## SUB-QUERIES

EX.NO:9

DATE:06/04/2024

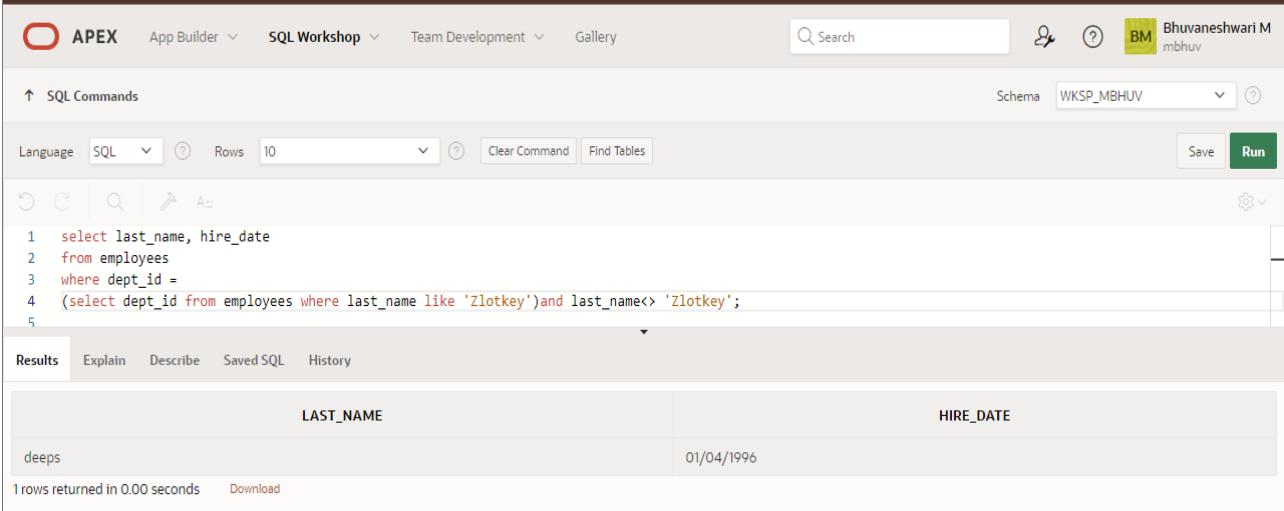
Find the Solution for the following:

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 employees
where dept_id =
(select dept_id from employees where last_name like 'Zlotkey')and last_name <>
'Zlotkey';
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for Bhuvaneshwari M (mbhuv). The main area has a toolbar with icons for Undo, Redo, Search, and Run. The SQL command window contains the following code:

```
1 select last_name, hire_date
2   from employees
3  where dept_id =
4 (select dept_id from employees where last_name like 'Zlotkey')and last_name<> 'Zlotkey';
5
```

The Results tab is selected, displaying the output:

| LAST_NAME | HIRE_DATE  |
|-----------|------------|
| deeps     | 01/04/1996 |

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

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

#### QUERY:

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

#### OUTPUT:

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

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

The results pane shows the output:

| EMPLOYEE_ID | LAST_NAME | SALARY |
|-------------|-----------|--------|
| 2           | deep      | 50000  |
| 4           | bhuvana   | 100000 |

2 rows returned in 0.01 seconds

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

#### QUERY:

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

#### OUTPUT:

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

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

The results pane shows the output:

| EMPLOYEE_ID | LAST_NAME |
|-------------|-----------|
| 4           | bhuvana   |

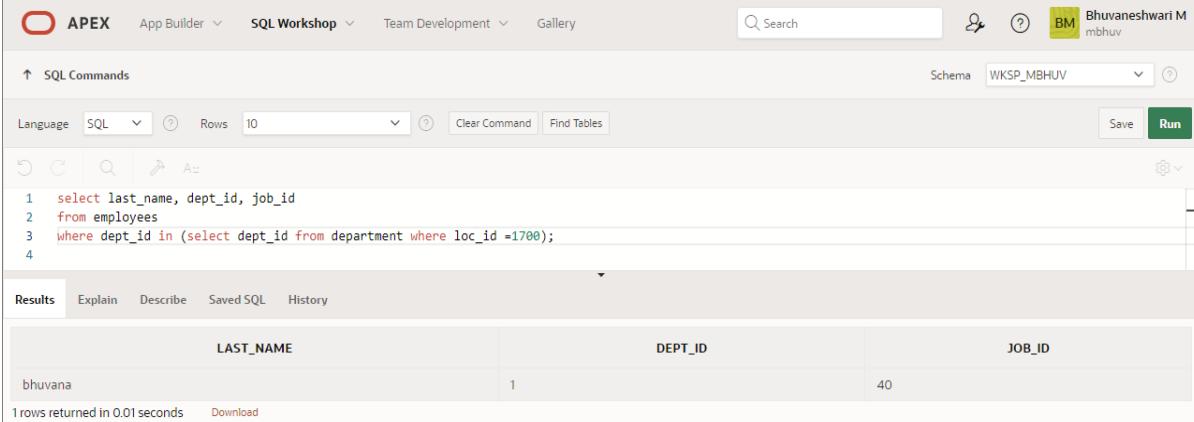
1 rows returned in 0.02 seconds

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

#### QUERY:

```
select last_name, dept_id, job_id  
from employees  
where dept_id in (select dept_id from department where loc_id =1700);
```

#### OUTPUT:



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

```
1 select last_name, dept_id, job_id  
2 from employees  
3 where dept_id in (select dept_id from department where loc_id =1700);  
4
```

The results table shows one row:

| LAST_NAME | DEPT_ID | JOB_ID |
|-----------|---------|--------|
| bhuvana   | 1       | 40     |

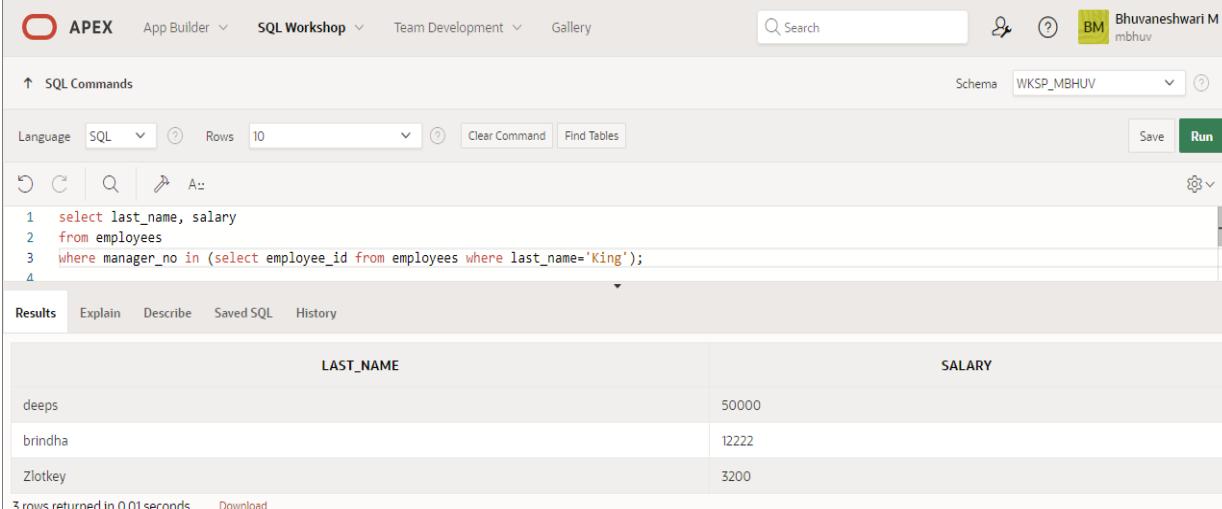
1 rows returned in 0.01 seconds

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

#### QUERY:

```
select last_name, salary  
from employees  
where manager_no in (select employee_id from employees where last_name='King');
```

#### OUTPUT:



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

```
1 select last_name, salary  
2 from employees  
3 where manager_no in (select employee_id from employees where last_name='King');  
4
```

The results table shows three rows:

| LAST_NAME | SALARY |
|-----------|--------|
| deepa     | 50000  |
| brindha   | 12222  |
| Zlotkey   | 3200   |

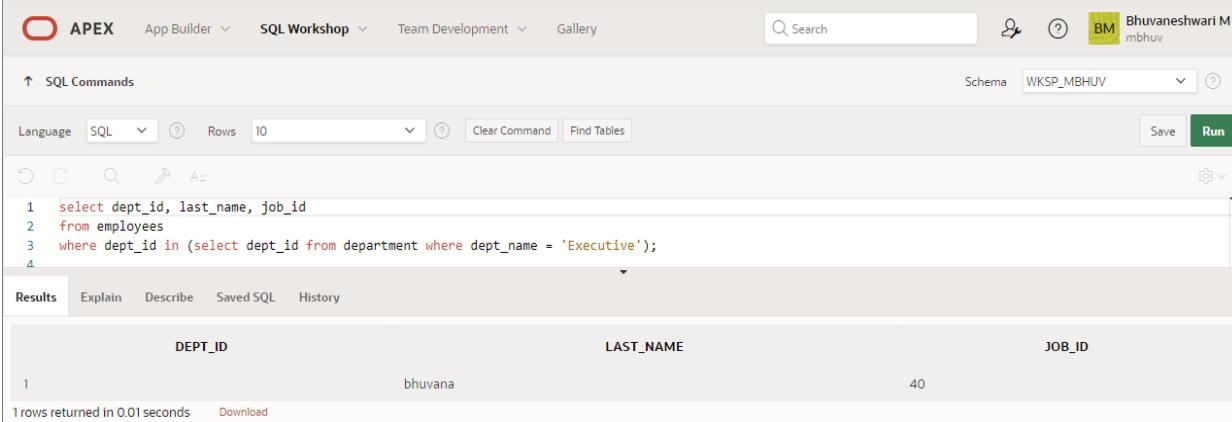
3 rows returned in 0.01 seconds

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

#### QUERY:

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

#### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and user information 'Bhuvaneshwari M mbhuv'. Below the navigation is a toolbar with 'Language' set to 'SQL', 'Rows' set to 10, and buttons for 'Clear Command', 'Find Tables', 'Save', and 'Run'. The main area contains the SQL command:

```
1 select dept_id, last_name, job_id  
2 from employees  
3 where dept_id in (select dept_id from department where dept_name = 'Executive');  
4
```

The results tab is selected, showing a single row of data:

| DEPT_ID | LAST_NAME | JOB_ID |
|---------|-----------|--------|
| 1       | bhuvana   | 40     |

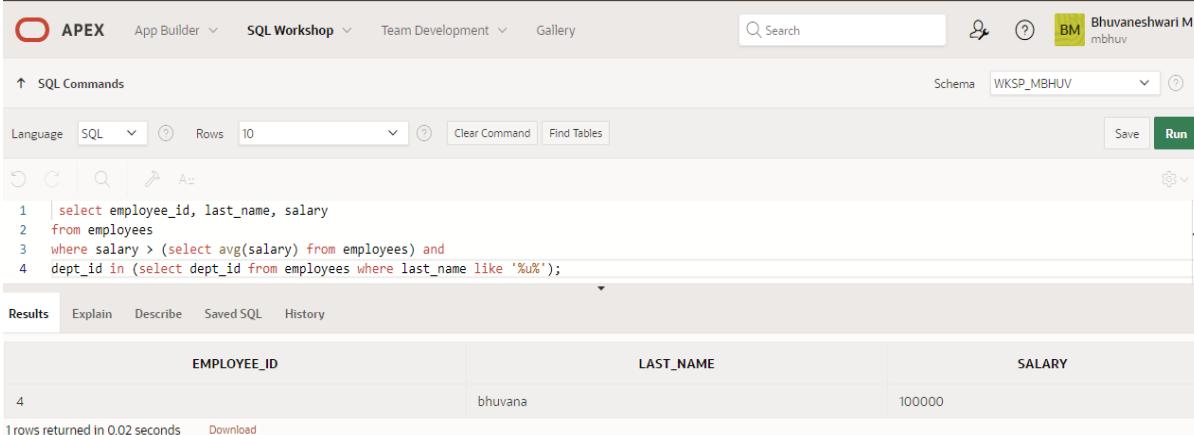
At the bottom, it says '1 rows returned in 0.01 seconds' and has 'Download' and 'Run' buttons.

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

#### QUERY:

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

#### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface, identical to the previous one but with a different query. The top navigation bar and toolbar are the same. The main area contains the modified SQL command:

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

The results tab is selected, showing a single row of data:

| EMPLOYEE_ID | LAST_NAME | SALARY |
|-------------|-----------|--------|
| 4           | bhuvana   | 100000 |

At the bottom, it says '1 rows returned in 0.02 seconds' and has 'Download' and 'Run' buttons.

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

**RESULT:**

The Queries for sub-queries has been executed successfully.

## USING THE SET OPERATORS

EX.NO:10

DATE:16/04/2024

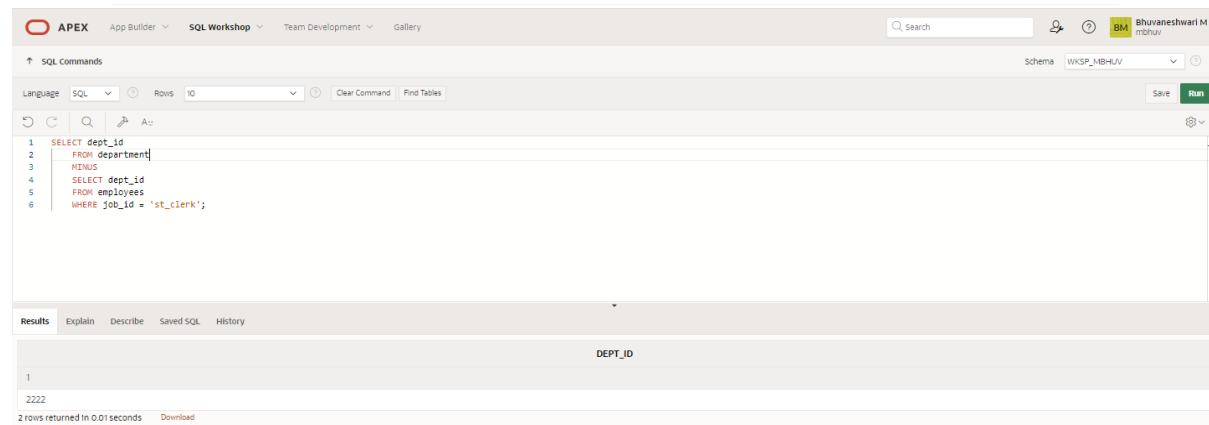
Find the Solution for the following:

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

**QUERY:**

```
SELECT dept_id
FROM department
MINUS
SELECT dept_id
FROM employees
WHERE job_id = 'st_clerk';
```

**OUTPUT:**



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

```
1 SELECT dept_id
2   FROM department|
3
4   MINUS
5
6   SELECT dept_id
7   FROM employees
8   WHERE job_id = 'st_clerk';
```

The results window displays a single row of data:

| DEPT_ID |
|---------|
| 2222    |

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

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

#### QUERY:

```
SELECT country_id,country_name
FROM countries
MINUS
SELECT l.country_id,c.country_name
FROM locations l, countries c
WHERE l.country_id = c.country_id;
```

#### OUTPUT:



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

```
1 SELECT country_id,country_name
2   FROM countries
3
4 MINUS
5
6 SELECT l.country_id,c.country_name
7   FROM locations l, countries c
8   WHERE l.country_id = c.country_id;
```

The results pane displays a single row of data:

| COUNTRY_ID | COUNTRY_NAME |
|------------|--------------|
| 101        | India        |

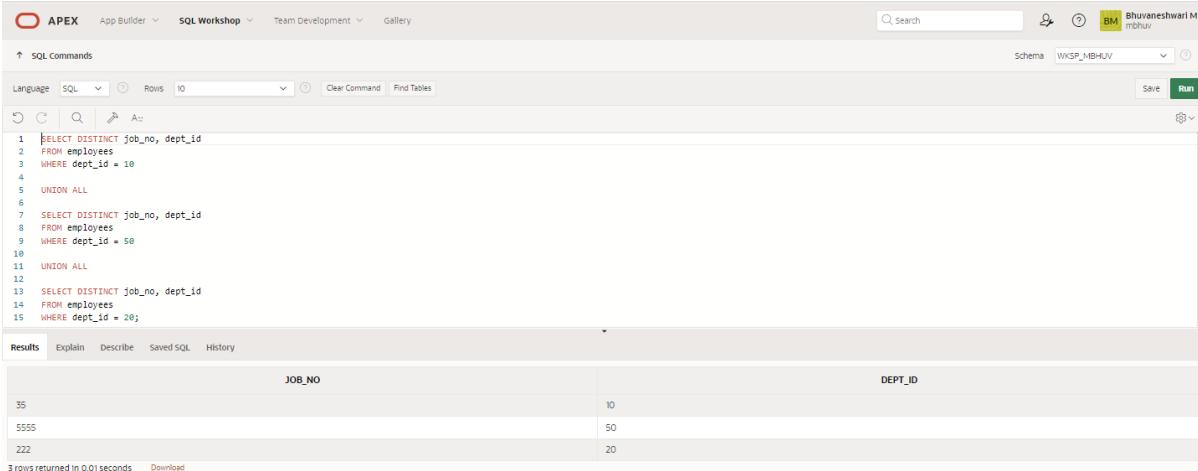
Below the results, it says "1 rows returned in 0.05 seconds".

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 DISTINCT job_no, dept_id
FROM employees
WHERE dept_id = 10
UNION ALL
SELECT DISTINCT job_no, dept_id
FROM employees
WHERE dept_id = 50
UNION ALL
SELECT DISTINCT job_no, dept_id
FROM employees
WHERE dept_id = 20;
```

**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: Bhuvaneshwari M (bm). The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the query code. The Results tab displays the output in a table format.

| JOB_NO | DEPT_ID |
|--------|---------|
| 35     | 10      |
| 5555   | 50      |
| 222    | 20      |

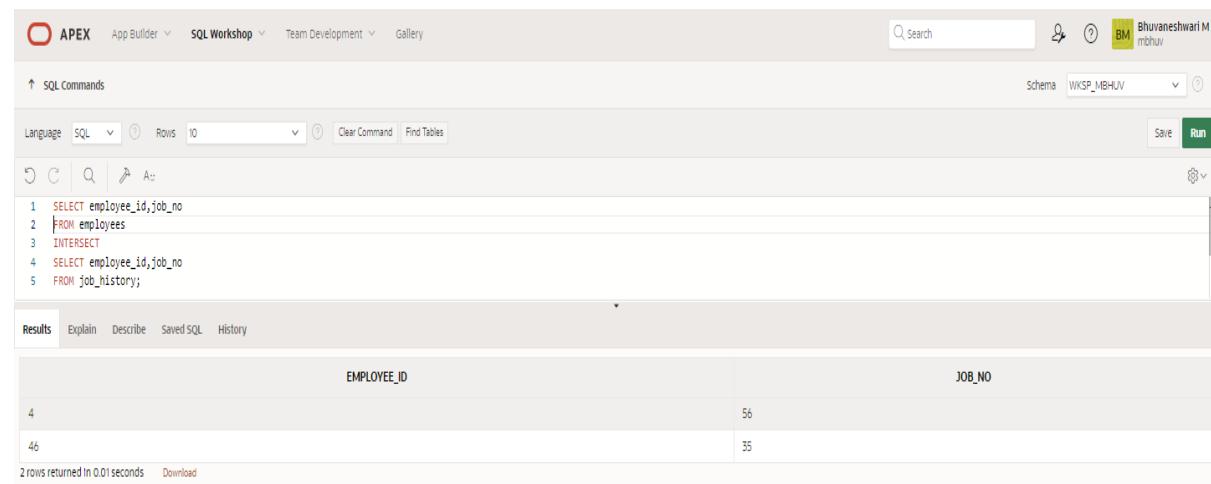
3 rows returned in 0.01 seconds [Download](#)

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 employee_id,job_no
FROM employees
INTERSECT
SELECT employee_id,job_no
FROM job_history;
```

**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: Bhuvaneswari M (bmhu) with a green icon. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the following code:

```
1 SELECT employee_id,job_no
2 FROM employees
3 INTERSECT
4 SELECT employee_id,job_no
5 FROM job_history;
```

The Results tab displays the output of the query:

| EMPLOYEE_ID | JOB_NO |
|-------------|--------|
| 4           | 56     |
| 46          | 35     |

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

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

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

#### QUERY:

```
SELECT last_name,dept_id,TO_CHAR(null)
```

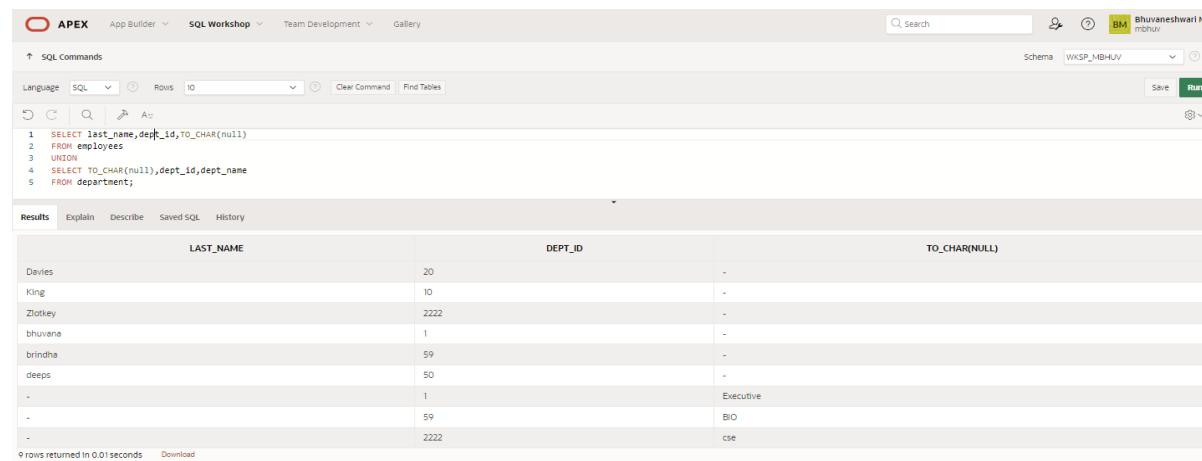
```
FROM employees
```

```
UNION
```

```
SELECT TO_CHAR(null),dept_id,dept_name
```

```
FROM department;
```

#### OUTPUT:



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

```
1 SELECT last_name,dept_id,TO_CHAR(null)
2 FROM employees
3 UNION
4 SELECT TO_CHAR(null),dept_id,dept_name
5 FROM department;
```

The results window displays the output of the query:

| LAST_NAME | DEPT_ID | TO_CHAR(NULL) |
|-----------|---------|---------------|
| Davies    | 20      | -             |
| King      | 10      | -             |
| Zlotkey   | 2222    | -             |
| bhuvena   | 1       | -             |
| brindha   | 59      | -             |
| deepa     | 50      | -             |
| -         | 1       | Executive     |
| -         | 59      | BIO           |
| -         | 2222    | cse           |

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

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

**RESULT:** The Queries using set operators has been executed successfully.

# CREATING VIEWS

EX.NO.11

DATE:20/04/2024

Find the Solution for the following:

1. Create a view called EMPLOYEE\_VU based on the employee numbers, employee names and department numbers from the EMPLOYEES table. Change the heading for the employee name to EMPLOYEE.

**QUERY:**

```
CREATE OR REPLACE VIEW employees_vu AS
SELECT employee_id, last_name employee, department_id
FROM employees;
```

**OUTPUT:**



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

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

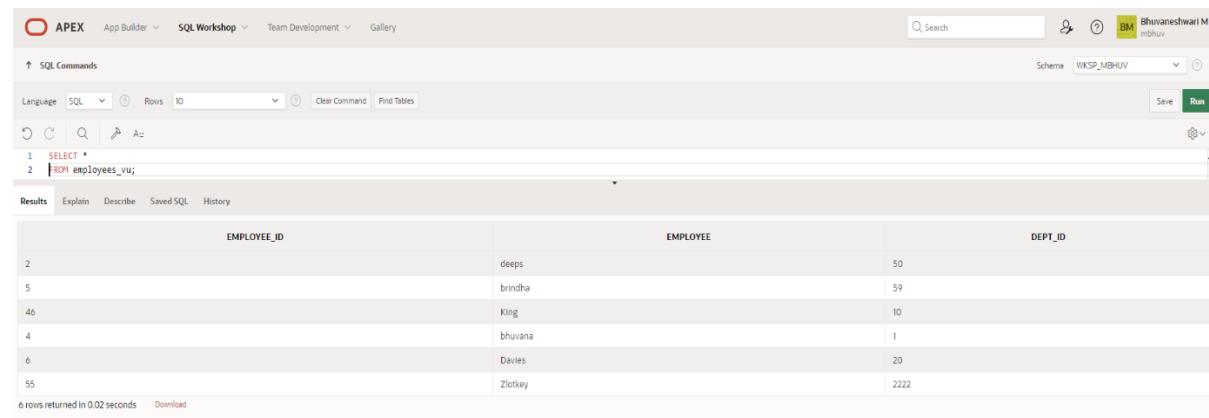
Below the code, the message "View created." is displayed. At the bottom, it shows "0.08 seconds".

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

**QUERY:**

```
select * from employees_vu;
```

**OUTPUT:**



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

| EMPLOYEE_ID | EMPLOYEE | DEPT_ID |
|-------------|----------|---------|
| 2           | deepa    | 50      |
| 5           | brinda   | 59      |
| 46          | King     | 10      |
| 4           | bhuvana  | 1       |
| 6           | Davies   | 20      |
| 55          | Zlotkey  | 2222    |

At the bottom left, it says "6 rows returned in 0.02 seconds".

3. Select the view name and text from the USER\_VIEWS data dictionary views.

#### QUERY:

```
SELECT view_name, text  
FROM user_views;
```

#### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows the user profile "Bhuvaneswari M mbhuv". The main area is titled "SQL Commands" with a "Schema" dropdown set to "WKSP\_MBHV". The query "SELECT view\_name, text FROM user\_views;" is entered in the command field. The results tab is selected, displaying three rows of data:

| VIEW_NAME    | TEXT  |
|--------------|---|
| DEPT50       | SELECT employee_id empno, last_name employee, dept_id deptno FROM employees WHERE dept_id = 50 WITH CHECK OPTION  |
| EMPLOYEES_VU | SELECT employee_id, last_name employee, dept_id FROM employees  |
| SALARY_VU    | SELECT e.last_name "Employee", d.dept_name "Department", e.salary "Salary", j.grade_level "Grades" FROM employees e, department d, job_grades j WHERE e.dept_id = d.dept_id AND e.salary BETWEEN j.lowest_sal AND j.highest_sal |

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

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

#### QUERY:

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

#### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface, identical to the previous one but with a different query. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows the user profile "Bhuvaneswari M mbhuv". The main area is titled "SQL Commands" with a "Schema" dropdown set to "WKSP\_MBHV". The query "SELECT employee, department\_id FROM employees\_vu;" is entered in the command field. The results tab is selected, displaying six rows of data:

| EMPLOYEE | DEPT_ID |
|----------|---------|
| deepa    | 50      |
| brindha  | 59      |
| King     | 10      |
| bhuvana  | 1       |
| Davies   | 20      |
| Zlotkey  | 2222    |

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

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

#### QUERY:

```
CREATE VIEW dept50 AS
SELECT employee_id empno, last_name employee,
department_id deptno
FROM employees
WHERE department_id = 50
WITH CHECK OPTION CONSTRAINT emp_dept_50;
```

#### OUTPUT:



The screenshot shows the Oracle 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 Bhuvaneshwari M (mbhuv). The main area is titled 'SQL Commands' with tabs for Language (SQL selected), Rows (10), Clear Command, and Find Tables. Below these are icons for Undo, Redo, Copy, Paste, and Run. The SQL editor contains the code for creating the 'dept50' view. The results tab at the bottom shows the message 'View created.' and a execution time of '0.03 seconds'.

```
1 CREATE VIEW dept50 AS
2 SELECT employee_id empno, last_name employee,
3 department_id deptno
4 FROM employees
5 WHERE department_id = 50
6 WITH CHECK OPTION CONSTRAINT emp_dept_50;
```

Results Explain Describe Saved SQL History

View created.  
0.03 seconds

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

#### QUERY:

**Describe dept50;**

**SELECT \* from dept50;**

#### 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 (Bhuvaneshwar M mbhuv) and the schema (WKSP\_MBHV). The main area is titled 'SQL Commands' with a search bar and a 'Run' button. Below it, there are tabs for Results, Explain, Describe, Saved SQL, and History. The 'Describe' tab is selected, showing the structure of the DEPT50 view. The results table has columns: Table, Column, Data Type, Length, Precision, Scale, Primary Key, Nullable, Default, and Comment. The DEPT50 view contains three columns: EMPNO, NUMBER, 22; EMPLOYEE, VARCHAR2, 25; and DEPTNO, NUMBER, 6. The EMPNO column is primary key and nullable, while the DEPTNO column is nullable.

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 (Bhuvaneshwar M mbhuv) and the schema (WKSP\_MBHV). The main area is titled 'SQL Commands' with a search bar and a 'Run' button. Below it, there are tabs for Results, Explain, Describe, Saved SQL, and History. The 'Results' tab is selected, showing the output of the following query:

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

The results table has columns: EMPNO, EMPLOYEE, and DEPTNO. It shows one row with EMPNO as 2, EMPLOYEE as 'deeps', and DEPTNO as 50. A note at the bottom says '1 rows returned in 0.00 seconds'.

7. Attempt to reassign Matos to department 80.

#### QUERY:

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

#### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows the user's profile (Bhuvaneshwar M mbhuv) and the schema (WKSP\_MBHV). The main area is titled 'SQL Commands' with a search bar and a 'Run' button. Below it, there are tabs for Results, Explain, Describe, Saved SQL, and History. The 'Results' tab is selected, showing the output of the following query:

```
1 update dept50
2 set deptno=80 where employee='Matos'
```

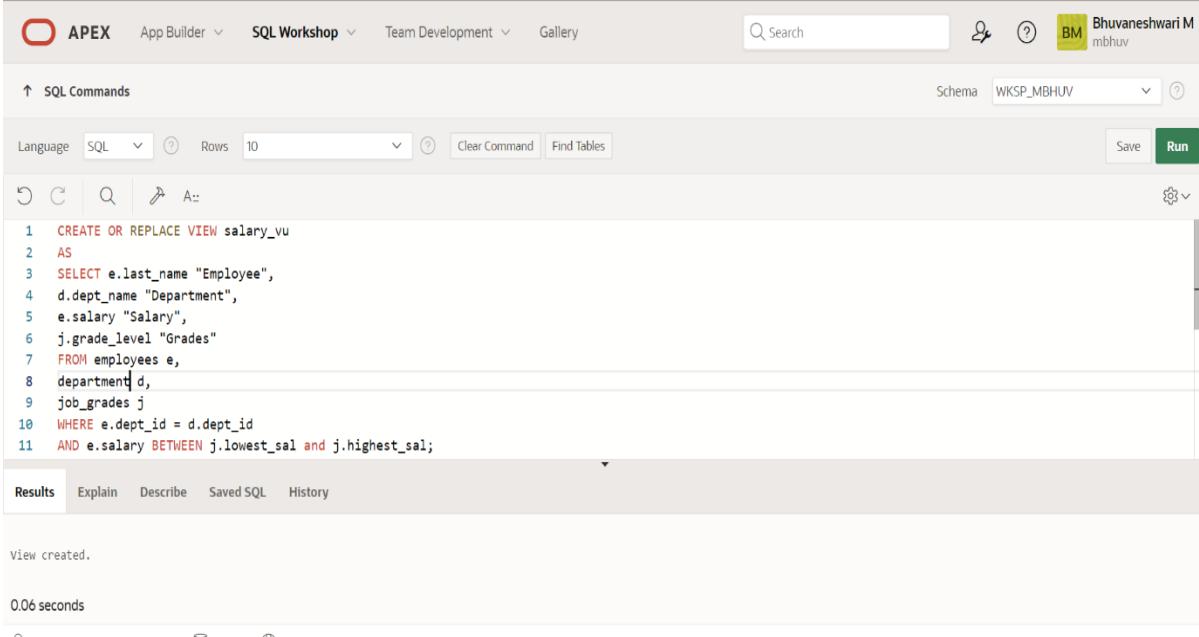
A yellow box highlights the error message: 'ORA-01402: view WITH CHECK OPTION where-clause violation'. At the bottom, it says '0.03 seconds'.

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

**QUERY:**

```
create or replace view salary_vu as
select e.last_name "Employee",d.dept_name "Department",e.salary "Salary",j.grade_level
"Grades"
from employees e,departments d,job_grade j
where e.department_id=d.dept_id and e.salary between j.lowest_sal and j.highest_sal;
```

**OUTPUT:**



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

```
1 CREATE OR REPLACE VIEW salary_vu
2 AS
3 SELECT e.last_name "Employee",
4 d.dept_name "Department",
5 e.salary "Salary",
6 j.grade_level "Grades"
7 FROM employees e,
8 department d,
9 job_grades j
10 WHERE e.dept_id = d.dept_id
11 AND e.salary BETWEEN j.lowest_sal and j.highest_sal;
```

The 'Results' tab is active at the bottom. The output message 'View created.' is displayed, along with a execution time of '0.06 seconds'. The bottom right corner shows 'Oracle APEX 23.2.4'.

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

**RESULT:**

The Queries for creating views has been executed successfully.

## Intro to Constraints: NOT NULL and UNIQUE Constraints.

EX.NO.12

DATE:04/04/2024

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

Global Fast Foods global\_locations Table

| NAME              | TYPE | LENGTH | PRECISION | SCALE | NULABLE | DEFAULT |
|-------------------|------|--------|-----------|-------|---------|---------|
| Id                | pk   |        |           |       | No      |         |
| name              |      |        |           |       |         |         |
| date_opened       |      |        |           |       | No      |         |
| address           |      |        |           |       | No      |         |
| city              |      |        |           |       | No      |         |
| zip/postal code   |      |        |           |       |         |         |
| phone             |      |        |           |       |         |         |
| email             | Uk   |        |           |       |         |         |
| manager_id        |      |        |           |       |         |         |
| Emergency contact |      |        |           |       |         |         |

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

**Database can be as reliable as the data in it, and database rules are implemented as Constraint to maintain data integrity. For example these constraints may prohibit deletion of a table or some row when insertion, updation or deletion is executed. Type of constraints:**

- PRIMARY KEY Constraint
- UNIQUE Constraint
- FOREIGN KEY Constraint
- CHECK Constraint with condition applied on the column/columns (they work at row level)
- NOT NULL Constraint (implemented at row level using special CHECK Constraint having condition IS NOT NULL for single column)

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

- **Constraints referring to more than one column are defined at Table Level**
- **NOT NULL constraint must be defined at column level as per ANSI/ISO SQL standard.**
- **If word CONSTRAINT is used in a CREATE TABLE statement, I must specify constraint name. Also, that is why, Table level constraint must be user-named.**

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

**If a constraint is violated in a SQL statement execution, it is easy to identify the cause with user-named constraints.**

- **It is easy to alter names/drop constraint.**
- **Handling production issues may be faster with user-named constraints**

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

| Global Fast Foods global_locations Table |       |           |         |            |        |           |
|--|-------|-----------|---------|------------|--------|-----------|
| NAME                                     | TYP E | DataType  | LENGT H | PRECISIO N | SCAL E | NULLABL E |
| id                                       | pk    | NUMBER    | 6       | 0          |        | No        |
| name                                     |       | VARCHAR 2 | 50      |            |        |           |
| date_opened                              |       | DATE      |         |            |        | No        |
| address                                  |       | VARCHAR 2 | 50      |            |        | No        |
| city                                     |       | VARCHAR 2 | 30      |            |        | No        |
| zip_postal_code                          |       | VARCHAR 2 | 12      |            |        |           |
| phone                                    |       | VARCHAR 2 | 20      |            |        |           |
| email                                    | uk    | VARCHAR 2 | 75      |            |        |           |
| manager_id                               |       | NUMBER    | 6       | 0          |        |           |
| emergency_contact                        |       | VARCHAR 2 | 20      |            |        |           |

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

**Global Fast Foods global\_locations Table**

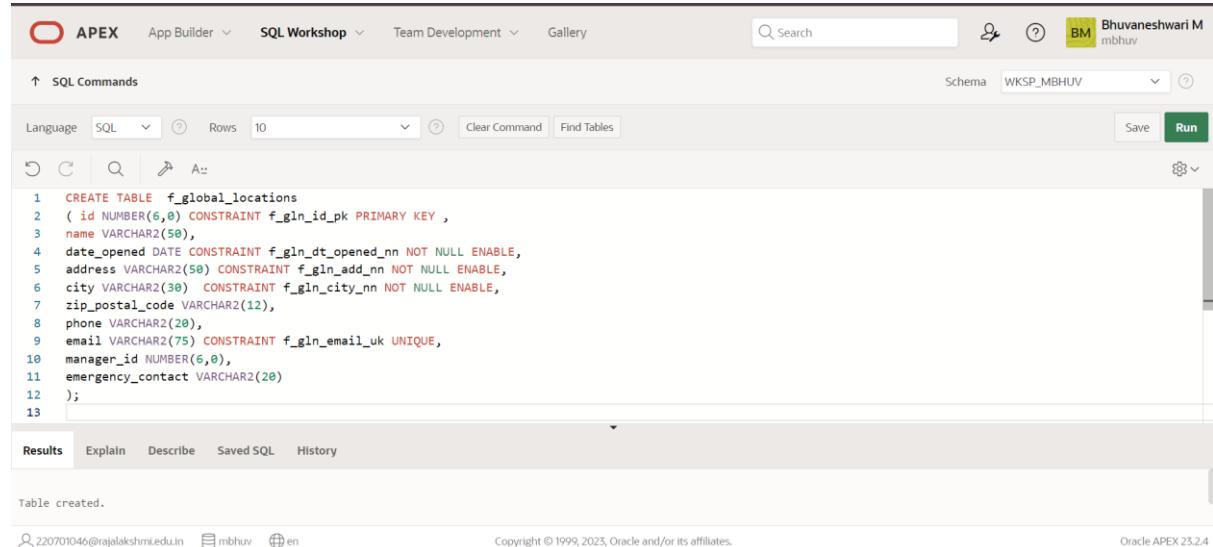
| NAME              | TYPE | DataType | LENGTH | PRECISION | SCALE | NULLABLE |
|-------------------|------|----------|--------|-----------|-------|----------|
| id                | pk   | NUMBER   | 6      | 0         |       | No       |
| name              |      | VARCHAR2 | 50     |           |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

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

QUERY:

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

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



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user profile of Bhuvaneshwari M (mbhuv). The main area is titled "SQL Commands" with a "Run" button. The schema is set to WKSP\_MBHV. The code editor contains the following SQL statement:

```
1 CREATE TABLE f_global_locations
2   ( id NUMBER(6,0) CONSTRAINT f_gln_id_pk PRIMARY KEY ,
3     name VARCHAR2(50),
4     date_opened DATE CONSTRAINT f_gln_dt_opened_nn NOT NULL ENABLE,
5     address VARCHAR2(50) CONSTRAINT f_gln_add_nn NOT NULL ENABLE,
6     city VARCHAR2(30) CONSTRAINT f_gln_city_nn NOT NULL ENABLE,
7     zip_postal_code VARCHAR2(12),
8     phone VARCHAR2(20),
9     email VARCHAR2(75) CONSTRAINT f_gln_email_uk UNIQUE,
10    manager_id NUMBER(6,0),
11    emergency_contact VARCHAR2(20)
12  );
13
```

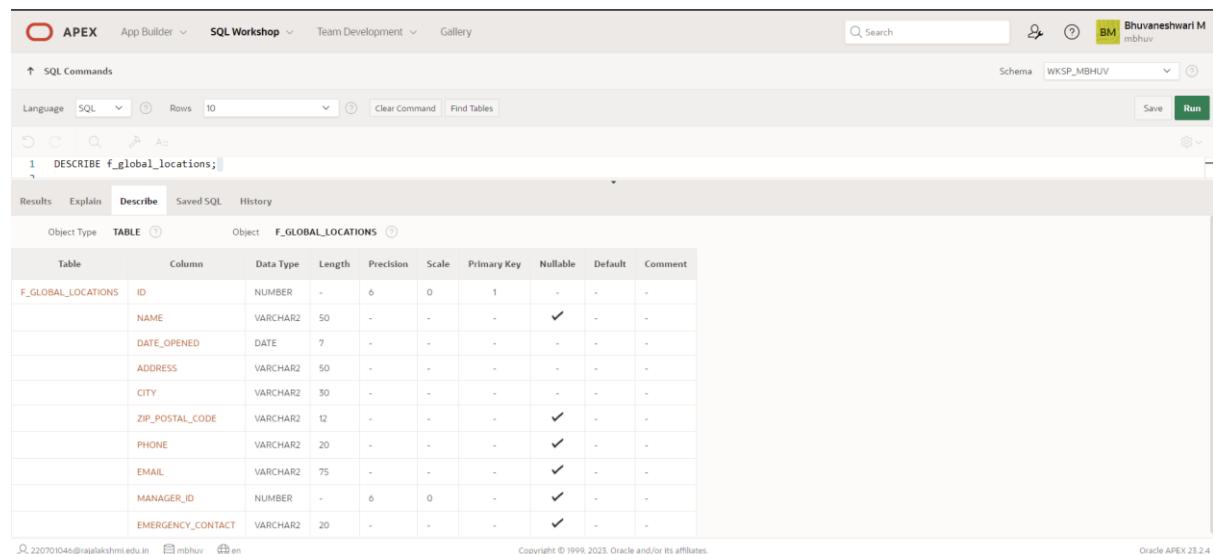
The results tab shows the message "Table created." Below the interface, footer information includes the email 220701046@rajalakshmi.edu.in, the user mbhuv, and the language en. Copyright information from Oracle is also present.

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

QUERY:

**DESCRIBE f\_global\_locations;**

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface with the "Describe" tab selected. The results table displays the structure of the F\_GLOBAL\_LOCATIONS table:

| Table              | Column            | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|--------------------|-------------------|-----------|--------|-----------|-------|-------------|----------|---------|---------|
| F_GLOBAL_LOCATIONS | ID                | NUMBER    | -      | 6         | 0     | 1           | -        | -       | -       |
|                    | NAME              | VARCHAR2  | 50     | -         | -     | -           | ✓        | -       | -       |
|                    | DATE_OPENED       | DATE      | 7      | -         | -     | -           | -        | -       | -       |
|                    | ADDRESS           | VARCHAR2  | 50     | -         | -     | -           | -        | -       | -       |
|                    | CITY              | VARCHAR2  | 50     | -         | -     | -           | -        | -       | -       |
|                    | ZIP_POSTAL_CODE   | VARCHAR2  | 12     | -         | -     | -           | ✓        | -       | -       |
|                    | PHONE             | VARCHAR2  | 20     | -         | -     | -           | ✓        | -       | -       |
|                    | EMAIL             | VARCHAR2  | 75     | -         | -     | -           | ✓        | -       | -       |
|                    | MANAGER_ID        | NUMBER    | -      | 6         | 0     | -           | ✓        | -       | -       |
|                    | EMERGENCY_CONTACT | VARCHAR2  | 20     | -         | -     | -           | ✓        | -       | -       |

Below the table, the footer information includes the email 220701046@rajalakshmi.edu.in, the user mbhuv, and the language en. Copyright information from Oracle is also present.

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

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

## **PRIMARY KEY, FOREIGN KEY, and CHECK Constraints**

1. What is the purpose of a

- PRIMARY KEY
- FOREIGN KEY
- CHECK CONSTRAINT

**a. PRIMARY KEY**

Uniquely identify each row in table.

**b. FOREIGN KEY**

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

**c. CHECK CONSTRAINT**

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

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

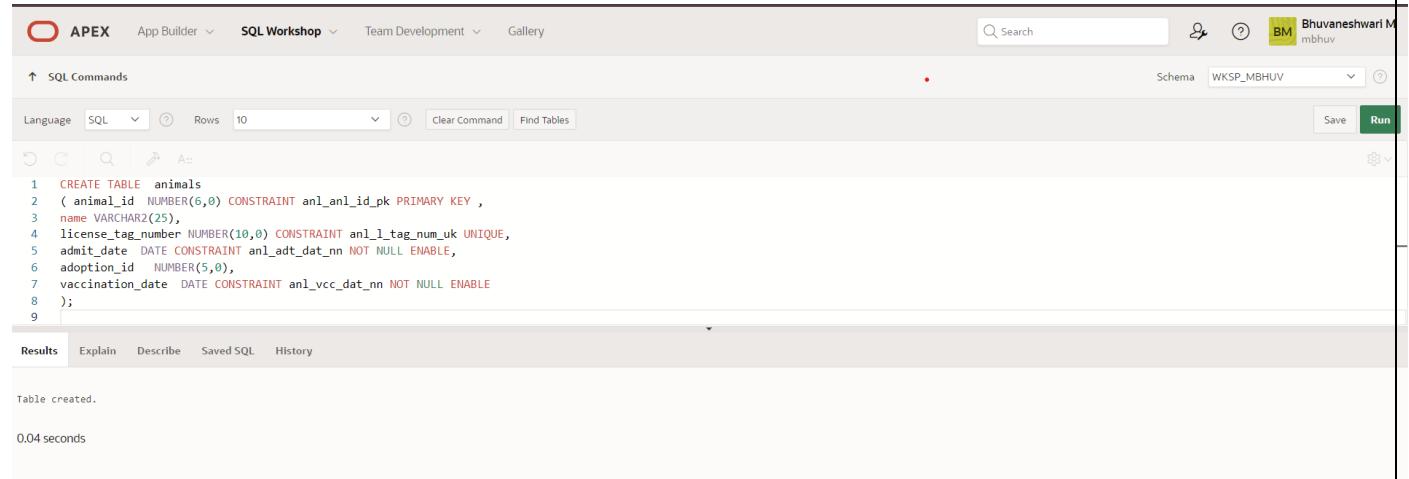
|                               |               |
|-------------------------------|---------------|
| animal_id NUMBER(6)           | - PRIMARY KEY |
| name VARCHAR2(25)             |               |
| license_tag_number NUMBER(10) | - UNIQUE      |
| admit_date DATE               | -NOT NULL     |
| adoption_id NUMBER(5),        |               |
| vaccination_date DATE         | -NOT NULL     |

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side of the header shows the user 'Bhuvaneshwari M mbhuv' and a 'Run' button. The main workspace is titled 'SQL Commands'. It contains a code editor with the SQL script for creating the 'animals' table. Below the code editor are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The results tab displays the output: 'Table created.' and '0.04 seconds'.

```
1 CREATE TABLE animals
2 (animal_id NUMBER(6,0) CONSTRAINT anl_anl_id_pk PRIMARY KEY ,
3 name VARCHAR2(25),
4 license_tag_number NUMBER(10,0) CONSTRAINT anl_l_tag_num_uk UNIQUE,
5 admit_date DATE CONSTRAINT anl_adt_dat_nn NOT NULL ENABLE,
6 adoption_id NUMBER(5,0),
7 vaccination_date DATE CONSTRAINT anl_vcc_dat_nn NOT NULL ENABLE
8 );
9 
```

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

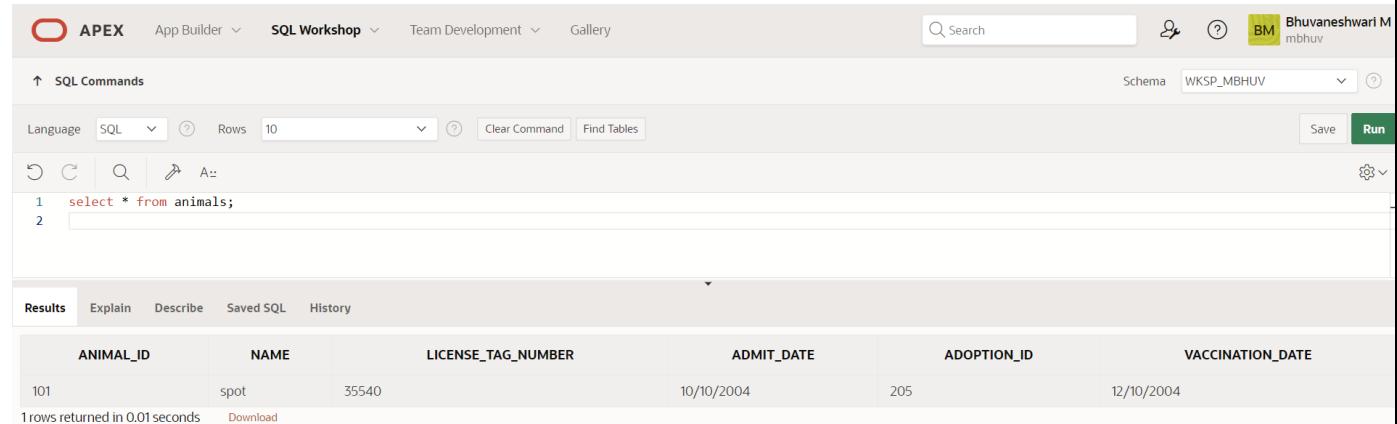
| <b>ANIMAL_ID</b> | <b>NAME</b> | <b>LICENSE_TAG_NUMBER</b> | <b>ADMIT_DATE</b> | <b>ADOPTION_ID</b> | <b>VACCINATION_DATE</b> |
|------------------|-------------|---------------------------|-------------------|--------------------|-------------------------|
| 101              | Spot        | 35540                     | 10-Oct-2004       | 205                | 12-Oct-2004             |

#### QUERY:

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

**SELECT \* FROM animals;**

#### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the command `select * from animals;` is entered and executed. The Results pane displays the following output:

| ANIMAL_ID | NAME | LICENSE_TAG_NUMBER | ADMIT_DATE | ADOPTION_ID | VACCINATION_DATE |
|-----------|------|--------------------|------------|-------------|------------------|
| 101       | spot | 35540              | 10/10/2004 | 205         | 12/10/2004       |

1 rows returned in 0.01 seconds

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

#### COLUMN LEVEL STATEMENT:

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

#### TABLE LEVEL STATEMENT:

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

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

- a. ON DELETE CASCADE

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

- b. ON DELETE SET NULL

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

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

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

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

**RESULT:**

The Queries for Not Null and Unique constraints has been executed successfully.

## Creating Views

EX.NO.13

DATE: 12/05/2024

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

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

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

**CREATE VIEW view\_d\_songs AS**

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

The screenshot shows the Oracle SQL Workshop interface. The SQL command to create the view is entered in the command window:

```
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 results tab shows the message "View created." and a execution time of "0.03 seconds".

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

The screenshot shows the Oracle SQL Workshop interface. The query "SELECT \* FROM view\_d\_songs;" is executed, and the results table shows the following data:

| ID | Song Title | ARTIST        |
|----|------------|---------------|
| 54 | YYY        | SHREYA GOSHAL |

1 rows returned in 0.01 seconds

4. REPLACE view\_d\_songs. Add type\_code to the column list. Use aliases for all columns. Or use alias after the CREATE statement as shown.

**CREATE OR REPLACE VIEW view\_d\_songs AS**

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



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there are search, schema dropdown (WKSP\_MBHV), and user information (Bhuvaneshwari M). The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, showing the following code:

```
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';
```

Below the code, the results show "View created." and a execution time of "0.01 seconds".

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

**CREATE OR REPLACE VIEW view\_d\_events\_pkgs AS**

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



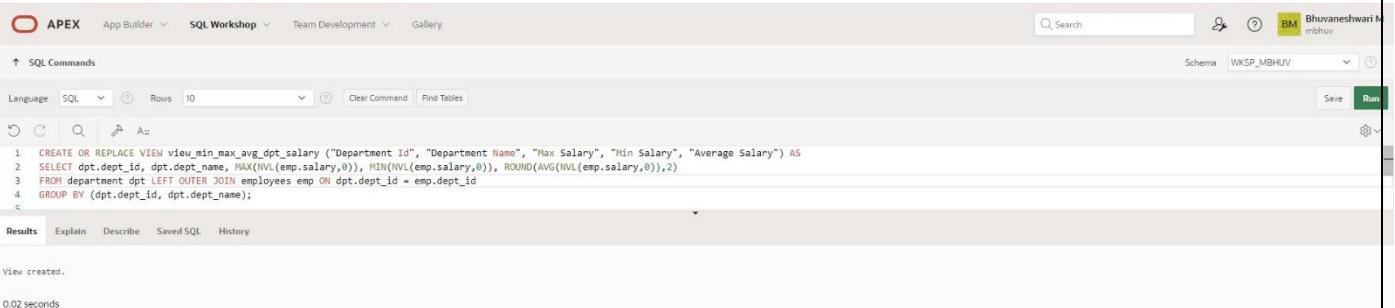
The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there are search, schema dropdown (WKSP\_MBHV), and user information (Bhuvaneshwari M). The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, showing the following code:

```
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
```

Below the code, the results show "View created." and a execution time of "0.02 seconds".

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

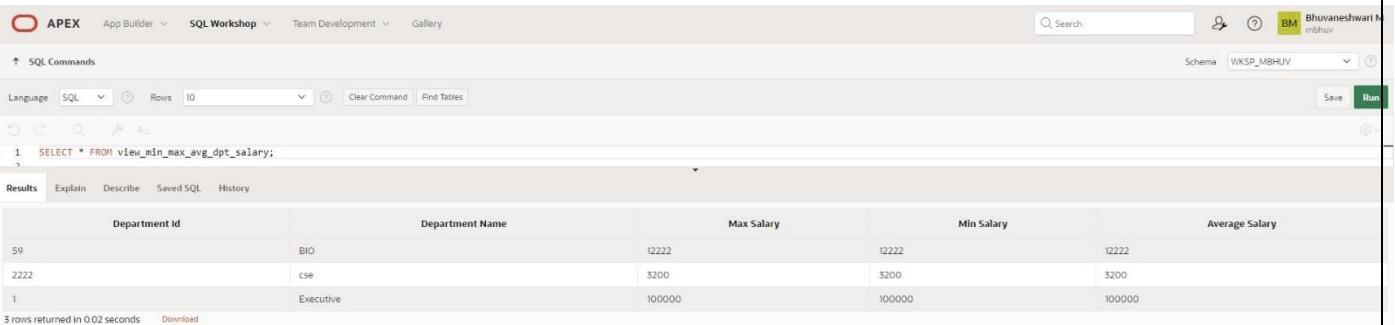
```
CREATE OR REPLACE VIEW view_min_max_avg_dpt_salary ("Department Id", "Department Name", "Max Salary", "Min Salary", "Average Salary") AS
SELECT dpt.department_id, dpt.department_name, MAX(NVL(emp.salary,0)),
MIN(NVL(emp.salary,0)), ROUND(AVG(NVL(emp.salary,0)),2)
FROM departments dpt LEFT OUTER JOIN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



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is active. The main area is titled 'SQL Commands'. A code editor window contains the SQL code for creating the view. The code is as follows:

```
1 CREATE OR REPLACE VIEW view_min_max_avg_dpt_salary ("Department Id", "Department Name", "Max Salary", "Min Salary", "Average Salary") AS
2 SELECT dpt.department_id, dpt.department_name, MAX(NVL(emp.salary,0)), MIN(NVL(emp.salary,0)), ROUND(AVG(NVL(emp.salary,0)),2)
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);
5
```

Below the code editor, the 'Results' tab is selected. The message 'View created.' is displayed. The execution time is shown as '0.02 seconds'.



The screenshot shows the Oracle SQL Workshop interface again. The 'Results' tab is selected under the 'SQL Commands' tab. The command executed is:

```
1 SELECT * FROM view_min_max_avg_dpt_salary;
```

Below the command, the results are displayed in a table:

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

At the bottom left, it says '3 rows returned in 0.02 seconds'. At the bottom right, there is a 'Download' link.

## 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';
```



This screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for Bhuvaneshwari M (mbhuv). The main area has a search bar and a schema dropdown set to WKSP\_MBHV. Below is a toolbar with Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run buttons. The SQL Commands pane contains the following code:

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

The Results pane displays the output of the query:

| OWNER     | TABLE_NAME   | COLUMN_NAME | UPDATABLE | INSERTABLE | DELETABLE |
|-----------|--------------|-------------|-----------|------------|-----------|
| WKSP_MBHV | COPY_D_SONGS | TITLE       | YES       | YES        | YES       |
| WKSP_MBHV | COPY_D_SONGS | DURATION    | YES       | YES        | YES       |
| WKSP_MBHV | COPY_D_SONGS | TYPE_CODE   | YES       | YES        | YES       |

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



This screenshot shows the Oracle SQL Workshop interface, identical to the previous one but with a different query in the SQL Commands pane:

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

The Results pane displays the output of the query:

| OWNER     | TABLE_NAME | COLUMN_NAME | UPDATABLE | INSERTABLE | DELETABLE |
|-----------|------------|-------------|-----------|------------|-----------|
| WKSP_MBHV | COPY_D_CDS | CD_NUMBER   | YES       | YES        | YES       |
| WKSP_MBHV | COPY_D_CDS | PRODUCER    | YES       | YES        | YES       |
| WKSP_MBHV | COPY_D_CDS | TITLE       | YES       | YES        | YES       |
| WKSP_MBHV | COPY_D_CDS | YEAR        | YES       | YES        | YES       |

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

2. Use the CREATE or REPLACE option to create a view of *all* the columns in the copy\_d\_songs table called view\_copy\_d\_songs.

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

```
SELECT * FROM view_copy_d_songs;
```

The screenshot shows two instances of the Oracle SQL Workshop interface. The top instance displays the creation of a view:

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

The message "View created." is shown below the command. The bottom instance shows the execution of the view:

```
1 SELECT * FROM view_copy_d_songs;
```

The message "no data found" is displayed, indicating that the view has been successfully created but no data is present in the underlying table.

3. Use view\_copy\_d\_songs to INSERT the following data into the underlying copy\_d\_songs table. Execute a SELECT \* from copy\_d\_songs to verify your DML command. See the graphic.

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

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

The screenshot shows the Oracle SQL Workshop interface with the following SQL command:

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

The message "1 row(s) inserted." is displayed, confirming the successful insertion. Below the command, a "Results" tab is open, showing the output of a SELECT \* query:

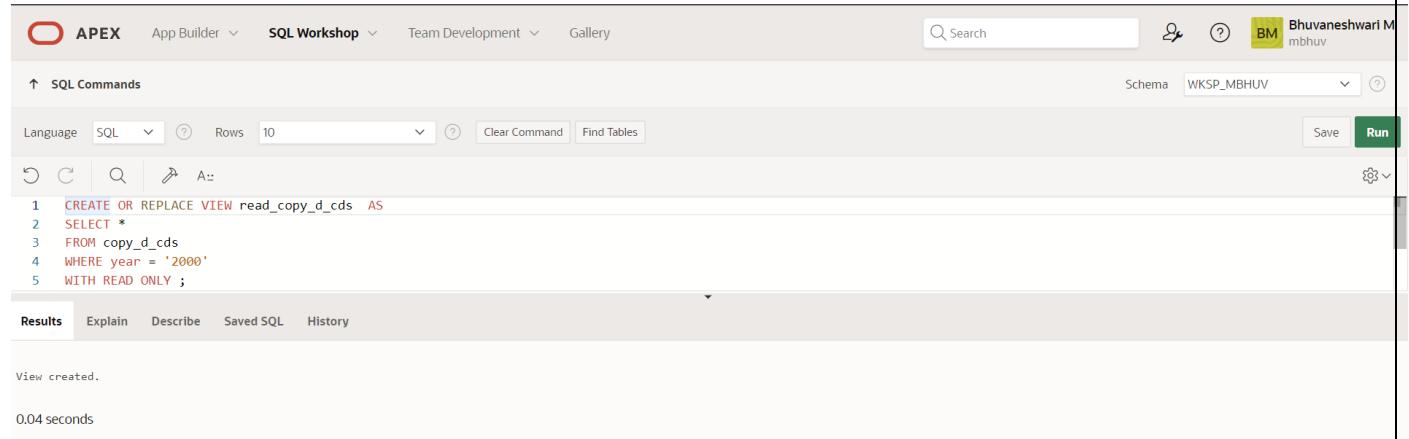
```
1 SELECT * FROM view_copy_d_songs;
```

The results table shows one row of data: ID 88, Title 'Mello Jello', Duration '2 min', Artist 'The What', and Type Code 4.

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

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

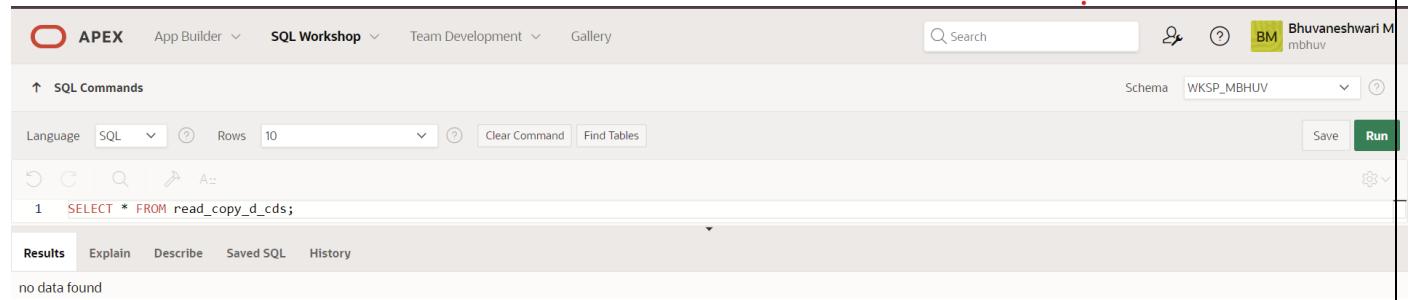
```
SELECT * FROM read_copy_d_cds;
```



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for Bhuvaneshwari M (mbhuv). The main area is titled "SQL Commands". It contains a code editor with the following SQL statements:

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

Below the code editor, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab shows the message "View created." and a execution time of "0.04 seconds".



The screenshot shows the Oracle SQL Workshop interface again. The top navigation bar and user profile are identical. The main area is titled "SQL Commands". It contains the following SQL statement:

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

Below the code editor, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab shows the message "no data found".

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. In the top navigation bar, the 'SQL Workshop' tab is selected. The main area displays the following SQL command:

```
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 is active, showing the message "View created." and a execution time of "0.04 seconds".

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

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

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

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

```
1 DELETE FROM read_copy_d_cds WHERE cd_number = 90;
```

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

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 SQL Workshop interface. At the top, there are tabs for APEX, App Builder, SQL Workshop (which is selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'Bhuvaneshwari M mbhuv', and a 'Run' button. Below the tabs, there's a toolbar with icons for Undo, Redo, Search, and others. The main area is titled 'SQL Commands' and contains a code editor with the following SQL command:

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

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

0 row(s) deleted.  
0.00 seconds

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 SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile 'Bhuvaneshwari M mbhuv' are also present. The main area is titled 'SQL Commands' with a sub-section 'CREATE OR REPLACE VIEW'. The command entered is:

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

The results tab shows the message 'View created.' and a execution time of '0.03 seconds'.

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile 'Bhuvaneshwari M mbhuv' are also present. The main area is titled 'SQL Commands' with a sub-section 'SELECT \*'. The command entered is:

```
1 SELECT * FROM view_copy_d_songs;
```

The results tab displays the output:

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

1 rows returned in 0.01 seconds

2. Issue a DROP view\_copy\_d\_songs. Execute a SELECT \* statement to verify that the view has been deleted.

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

**ORA-00942: table or view does not exist**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile 'Bhuvaneshwari M mbhuv' are also present. The main area is titled 'SQL Commands' with a sub-section 'SELECT \*'. The command entered is:

```
1 SELECT * FROM view_copy_d_songs;
```

The results tab displays the error message 'ORA-00942: table or view does not exist'.

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

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

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The search bar contains 'Search'. The user's profile is 'Bhuvaneshwari M mbhuv'. The schema is set to 'WKSP\_MBHV'. The main area shows the SQL command:

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

The results tab displays the output:

| LAST_NAME | SALARY |
|-----------|--------|
| bhuvana   | 100000 |
| Matos     | 50000  |
| Davies    | 34000  |

3 rows returned in 0.02 seconds. There is a 'Download' button at the bottom.

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

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

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The search bar contains 'Search'. The user's profile is 'Bhuvaneshwari M mbhuv'. The schema is set to 'WKSP\_MBHV'. The main area shows the SQL command:

```
1 SELECT empm.last_name, empm.salary, dptmx.dept_id
2 FROM (SELECT dpt.dept_id, MAX(NVL(emp.salary,0)) max_dpt_sal FROM department dpt
3 LEFT OUTER JOIN employees emp ON dpt.dept_id = emp.dept_id GROUP BY dpt.dept_id)
4 dptmx LEFT OUTER JOIN employees empm ON dptmx.dept_id = empm.dept_id WHERE NVL(empm.salary,0) = dptmx.max_dpt_sal;
```

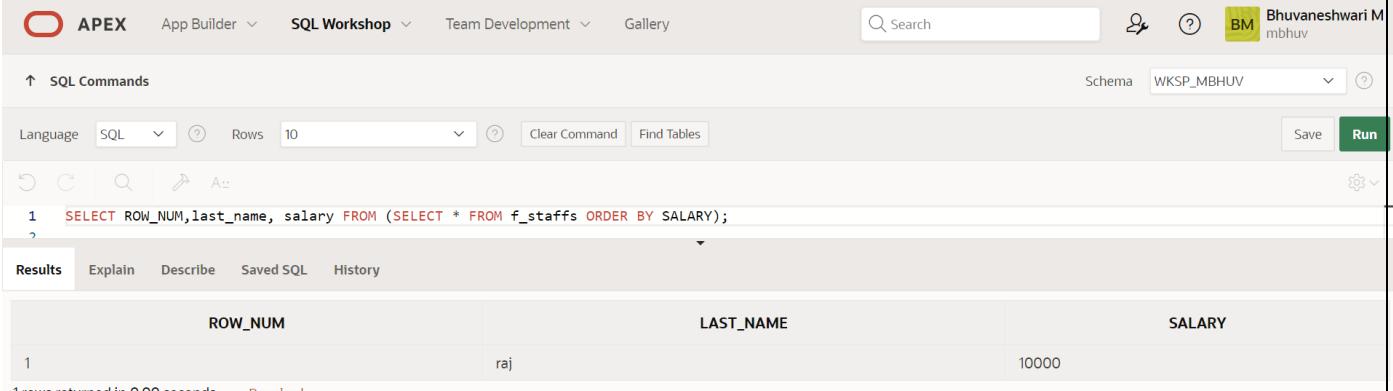
The results tab displays the output:

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

3 rows returned in 0.01 seconds. There is a 'Download' button at the bottom.

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

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



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user profile, and session information for Bhuvaneshwari M (mbhuv). The main area has tabs for SQL Commands, Results (selected), Explain, Describe, Saved SQL, and History. The SQL tab contains the executed query:

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

The Results tab displays the output:

| ROW_NUM | LAST_NAME | SALARY |
|---------|-----------|--------|
| 1       | raj       | 10000  |

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

## Indexes and Synonyms

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

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

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

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

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

3. When will an index be created automatically?

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

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

**CREATE INDEX d\_tlg\_cd\_number\_fk\_i  
on d\_track\_listings (cd\_number);**

The screenshot shows the Oracle Application Express SQL Workshop Data Browser interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is signed in as Bhuvaneshwari M mbhuv. The schema selected is WKSP\_MBHV. The main area is titled "SQL Commands" with a "Run" button. The code entered is:

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

The results section shows the message "Index created." and a execution time of "0.02 seconds".

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

**SELECT ucm.index\_name, ucm.column\_name, ucm.column\_position, uix.uniqueness  
FROM user\_indexes uix INNER JOIN user\_ind\_columns ucm ON uix.index\_name =  
ucm.index\_name  
WHERE ucm.table\_name = 'D\_SONGS';**

The screenshot shows the Oracle Application Express SQL Workshop Data Browser interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is signed in as Bhuvaneshwari M mbhuv. The schema selected is WKSP\_MBHV. The main area is titled "SQL Commands" with a "Run" button. The code entered is:

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

The results section shows the message "no data found".

6. Use a SELECT statement to display the index\_name, table\_name, and uniqueness from the data dictionary USER\_INDEXES for the DJs on Demand D\_EVENTS table.

**SELECT index\_name, table\_name,uniqueness FROM user\_indexes where table\_name =  
'D\_EVENTS';**

The screenshot shows the Oracle Application Express SQL Workshop Data Browser interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is signed in as Bhuvaneshwari M mbhuv. The schema selected is WKSP\_MBHV. The main area is titled "SQL Commands" with a "Run" button. The code entered is:

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

The results section shows the message "no data found".

7. Write a query to create a synonym called dj\_tracks for the DJs on Demand d\_track\_listings table.

**CREATE SYNONYM dj\_tracks FOR d\_track\_listings;**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, user profile, and schema dropdown set to WKSP\_MBHUV. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the command: `CREATE SYNONYM dj_tracks FOR d_track_listings;`. Below the command, the results show "Synonym created." and a execution time of "0.01 seconds".

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

**CREATE INDEX d\_ptr\_last\_name\_idx  
ON d\_partners(LOWER(last\_name));**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, user profile, and schema dropdown set to WKSP\_MBHUV. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the command: `CREATE INDEX d_ptr_last_name_idx ON D_PARTNERS(LOWER(last_name));`. Below the command, the results show "Index created." and a execution time of "0.02 seconds".

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

**CREATE SYNONYM dj\_tracks2 FOR d\_track\_listings;**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, user profile, and schema dropdown set to WKSP\_MBHUV. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the command: `CREATE SYNONYM dj_tracks2 FOR d_track_listings;`. Below the command, the results show "Synonym created." and a execution time of "0.01 seconds".

**SELECT \* FROM user\_synonyms WHERE table\_NAME = UPPER('d\_track\_listings');**

APEX App Builder SQL Workshop Team Development Gallery Search Schema WKSP\_MBHUV

↑ SQL Commands

Language SQL Rows 10 Clear Command Find Tables Save Run

1 SELECT \* FROM user\_synonyms WHERE table\_NAME = UPPER('d\_track\_listings');

Results Explain Describe Saved SQL History

| SYNONYM_NAME | TABLE_OWNER | TABLE_NAME       | DB_LINK | ORIGIN_CON_ID |
|--------------|-------------|------------------|---------|---------------|
| DJ_TRACKS    | WKSP_MBHUV  | D_TRACK_LISTINGS | -       | 0             |
| DJ_TRACKS2   | WKSP_MBHUV  | D_TRACK_LISTINGS | -       | 0             |

2 rows returned in 0.03 seconds Download

10. Drop the synonym that you created in question

**DROP SYNONYM dj\_tracks2;**

APEX App Builder SQL Workshop Team Development Gallery Search Schema WKSP\_MBHUV

↑ SQL Commands

Language SQL Rows 10 Clear Command Find Tables Save Run

1 DROP SYNONYM dj\_tracks2;

Results Explain Describe Saved SQL History

Synonym dropped.

0.02 seconds

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

## RESULT:

The Queries for creating viewa has been executed successfully.

## OTHER DATABASE OBJECTS

EX.NO:14

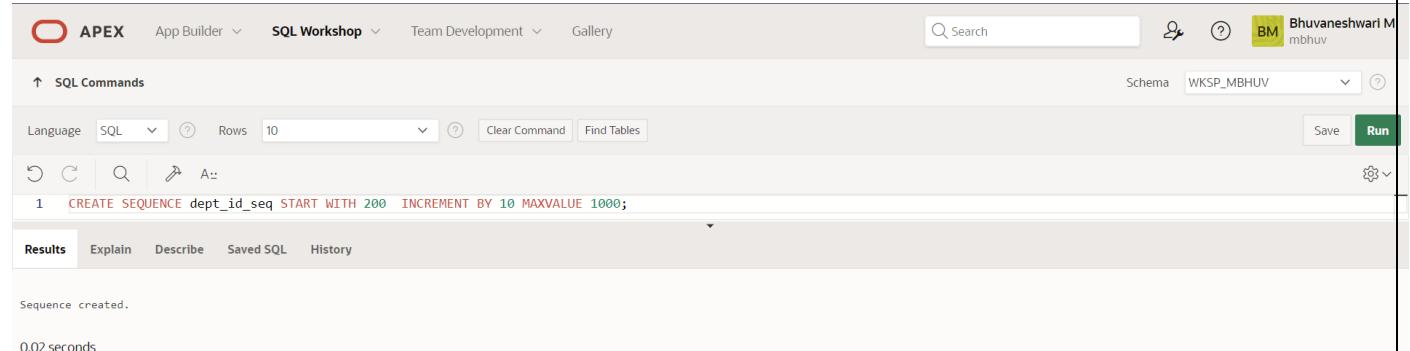
DATE : 16/05/2024

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

**QUERY:**

**CREATE SEQUENCE dept\_id\_seq START WITH 200 INCREMENT BY 10  
MAXVALUE 1000;**

**OUTPUT:**



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

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

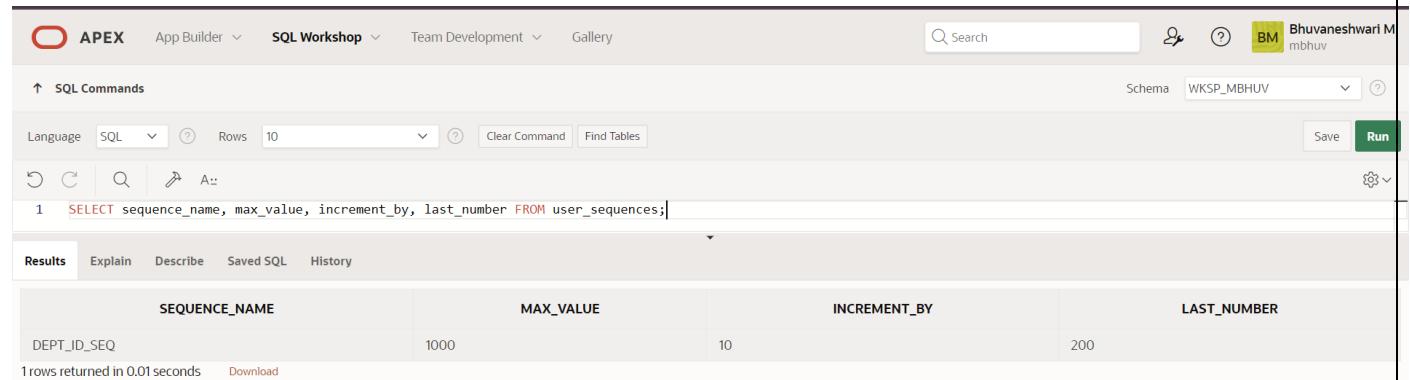
The 'Results' tab is active, showing the output: "Sequence created." Below the results, it says "0.02 seconds".

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

**QUERY:**

**SELECT sequence\_name, max\_value, increment\_by, last\_number FROM user\_sequences;**

**OUTPUT:**



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

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

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

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

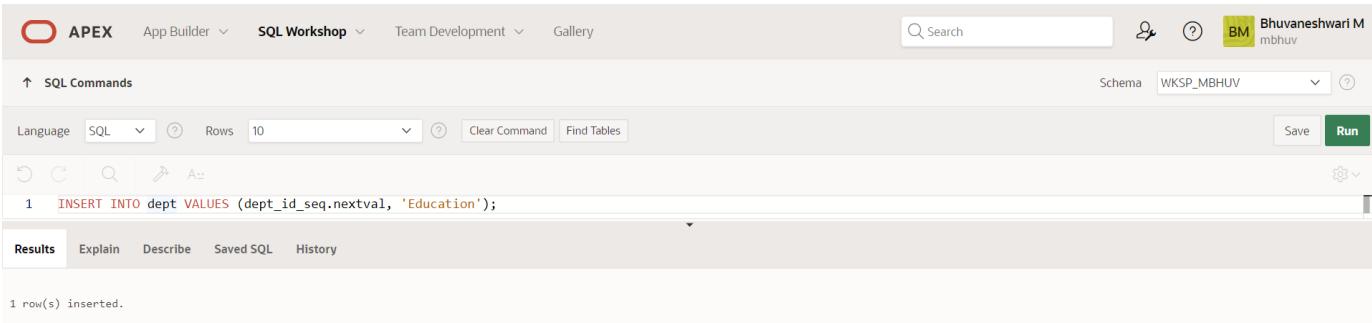
Below the results, it says "1 rows returned in 0.01 seconds".

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 SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user information (Bhuvaneshwari M mbhuv), and a schema dropdown set to WKSP\_MBHV. The main area is titled 'SQL Commands' with a 'Run' button. Below it, the command history shows two INSERT statements:

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

Under the 'Results' tab, the output shows:

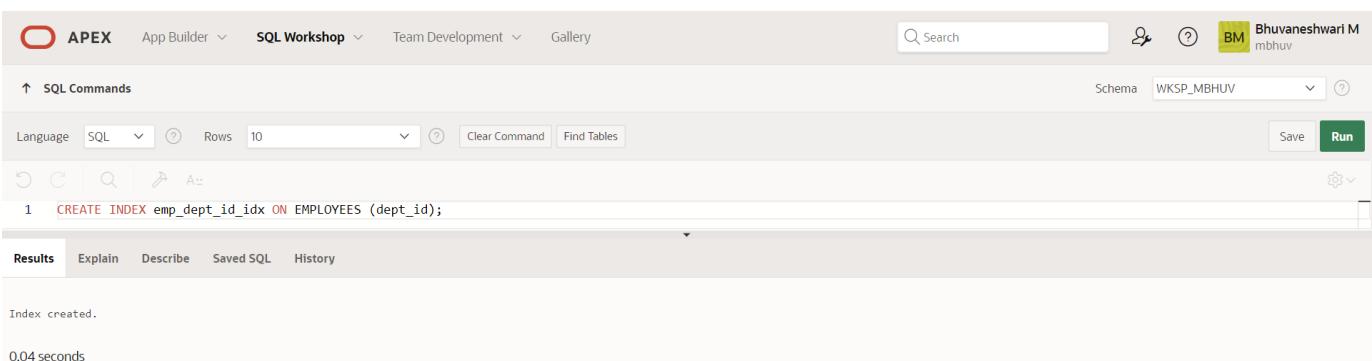
1 row(s) inserted.  
0.02 seconds

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 SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user information (Bhuvaneshwari M mbhuv), and a schema dropdown set to WKSP\_MBHV. The main area is titled 'SQL Commands' with a 'Run' button. Below it, the command history shows a CREATE INDEX statement:

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

Under the 'Results' tab, the output shows:

Index created.  
0.04 seconds

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user information (Bhuvaneshwari M mbhuv), and a schema dropdown set to WKSP\_MBHV. The main area is titled 'SQL Commands' with a 'Run' button. Below it, the command history shows a SELECT statement:

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

Under the 'Results' tab, the output shows a table with three columns: INDEX\_NAME, TABLE\_NAME, and UNIQUENESS. The data is:

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

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

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

### **RESULT:**

The Queries for executing other data objects has been executed successfully.

## **CONTROLLING USER ACCESS**

**EX.NO:15**

**DATE:16/05/2024**

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

The CREATE SESSION system privilege

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

The CREATE TABLE privilege

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

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

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

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

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

The ALTER USER statement

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

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

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

7. Query all the rows in your DEPARTMENTS table.

SELECT \* FROM departments;

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

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

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

9. Query the USER\_TABLES data dictionary to see information about the tables that you own.

SELECT table\_name FROM user\_tables;

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

Team 1 revokes the privilege.

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

Team 2 revokes the privilege.

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

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

Team 1 executes this INSERT statement.

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

Team 2 executes this INSERT statement.

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

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

**RESULT:** The Queries for controlling user objects has been executed successfully.

## PL/SQL CONTROL STRUCTURES

EX.NO:16

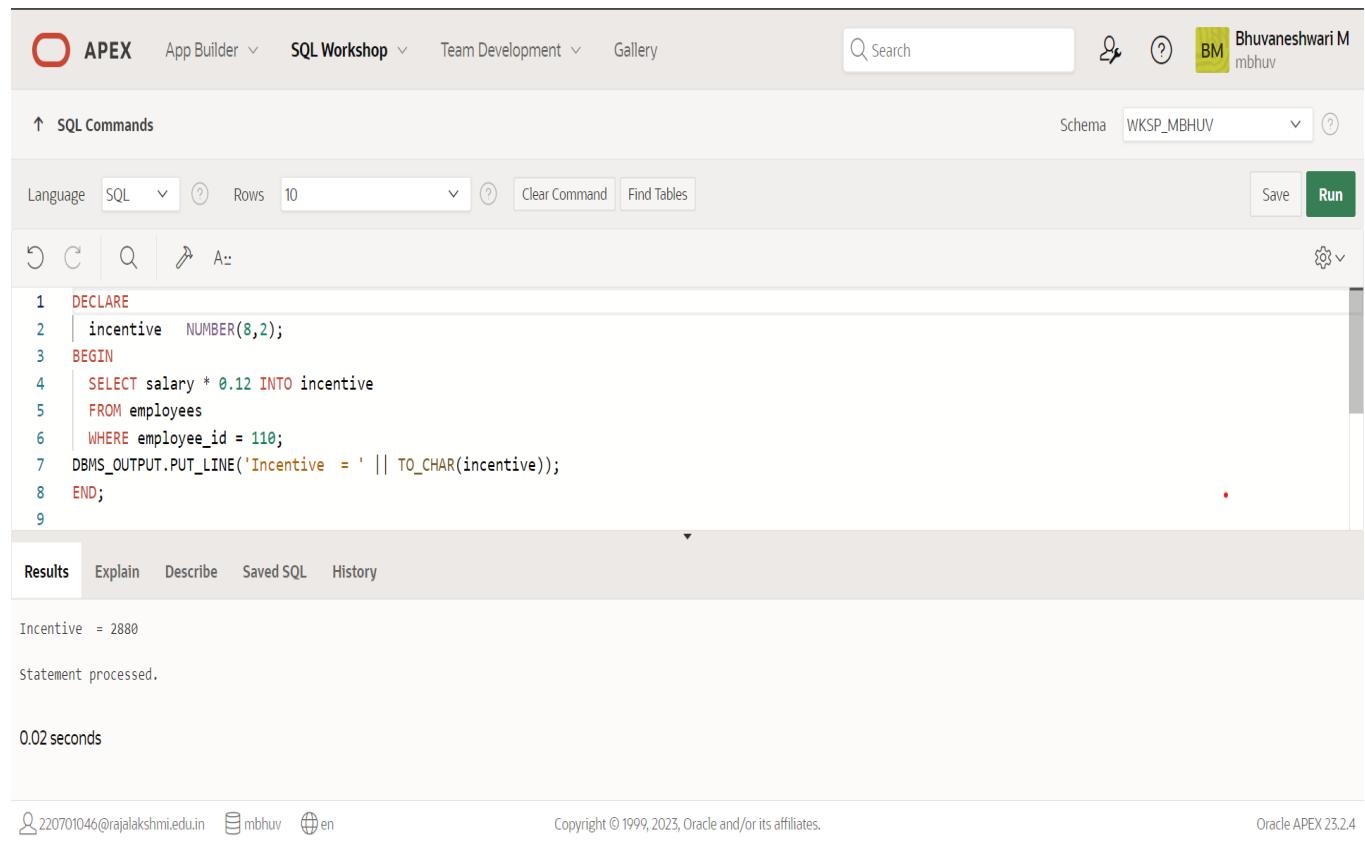
DATE:18/05/2024

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 Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there is a search bar, a user profile icon for 'Bhuvaneshwari M mbhuv', and a help icon. Below the navigation, the schema is set to 'WKSP\_MBHUV'. The main workspace is titled 'SQL Commands' and contains a code editor with the following PL/SQL block:

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

Below the code editor, there are tabs for Results, Explain, Describe, Saved SQL, and History. The results section displays the output of the executed query:

```
Incentive = 2880
Statement processed.

0.02 seconds
```

At the bottom, there are footer links for user information (220701046@rajalakshmi.edu.in, mbhuv, en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and version information (Oracle APEX 23.2.4).

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 written:

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

An error message is displayed in a yellow box:

```
Error at line 4/25: ORA-06550: line 4, column 25:
PLS-00201: identifier 'Welcome' must be declared
ORA-06512: at "SYS.WWV_DBMS_SQL_APEX_230200", line 801
ORA-06550: line 4, column 3:
PL/SQL: Statement ignored
```

Below the code, the error message is repeated:

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

At the bottom, the footer includes:

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

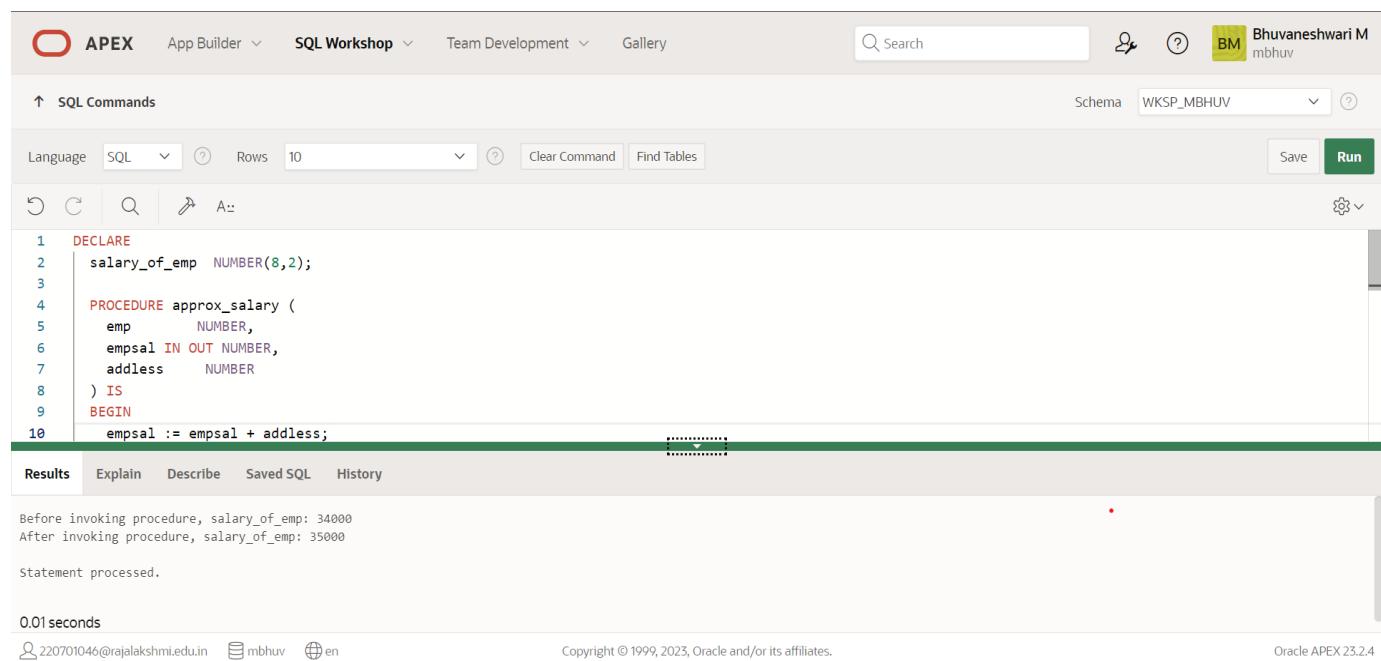
3.) Write a PL/SQL block to adjust the salary of the employee whose ID 122.

#### QUERY:

```
DECLARE
    salary_of_emp NUMBER(8,2);
    PROCEDURE approx_salary (
        emp      NUMBER,
        empsal IN OUT NUMBER,
        addless  NUMBER
    ) IS
    BEGIN
        empsal := empsal + addless;
    END;
```

```
BEGIN
    SELECT salary INTO salary_of_emp
    FROM employees
    WHERE employee_id = 122;
    DBMS_OUTPUT.PUT_LINE
    ('Before invoking procedure, salary_of_emp: ' || salary_of_emp);
    approx_salary (100, salary_of_emp, 1000);
    DBMS_OUTPUT.PUT_LINE
    ('After invoking procedure, salary_of_emp: ' || salary_of_emp);
END;
/
```

#### OUTPUT:



The screenshot shows the Oracle 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 profile 'Bhuvaneshwari M mbhuv' are also present. The main workspace is titled 'SQL Commands' and contains the following code:

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

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

```
Before invoking procedure, salary_of_emp: 34000
After invoking procedure, salary_of_emp: 35000

Statement processed.
```

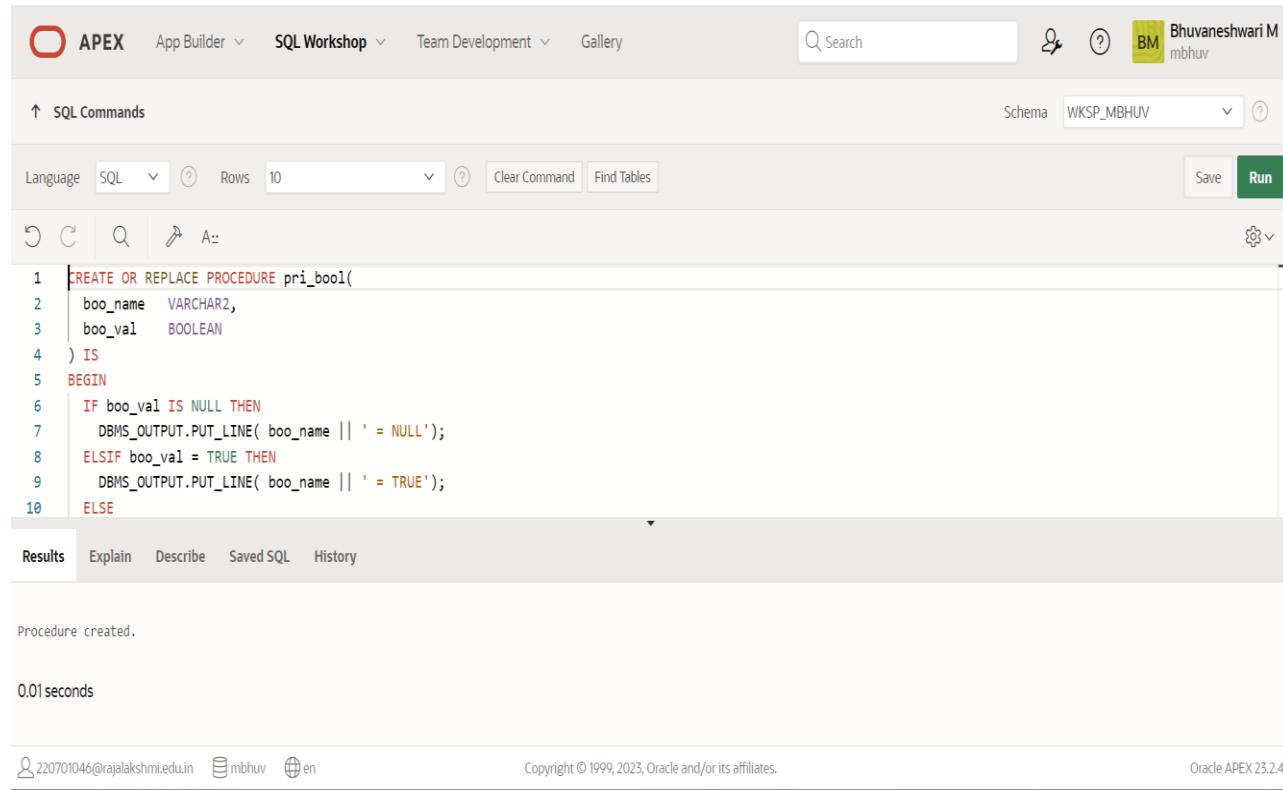
At the bottom, it shows '0.01 seconds' and copyright information: 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

#### QUERY:

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

#### OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, a user icon for 'Bhuvaneshwari M mbluv', and a schema dropdown set to 'WKSP\_MBHUV'. The main workspace is titled 'SQL Commands' and contains the PL/SQL code for the 'pri\_bool' procedure. Below the code, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The results section displays the message 'Procedure created.' and a execution time of '0.01 seconds'. At the bottom, there are footer links for '220701046@rajalakshmi.edu.in', 'mbluv', and 'en', along with copyright information for Oracle and a reference to 'Oracle APEX 23.2.4'.

```
1 CREATE OR REPLACE PROCEDURE pri_bool(
2   boo_name  VARCHAR2,
3   boo_val   BOOLEAN
4 ) IS
5 BEGIN
6   IF boo_val IS NULL THEN
7     DBMS_OUTPUT.PUT_LINE( boo_name || ' = NULL');
8   ELSIF boo_val = TRUE THEN
9     DBMS_OUTPUT.PUT_LINE( boo_name || ' = TRUE');
10  ELSE
11  END IF;
12END;
13/
```

Procedure created.  
0.01 seconds

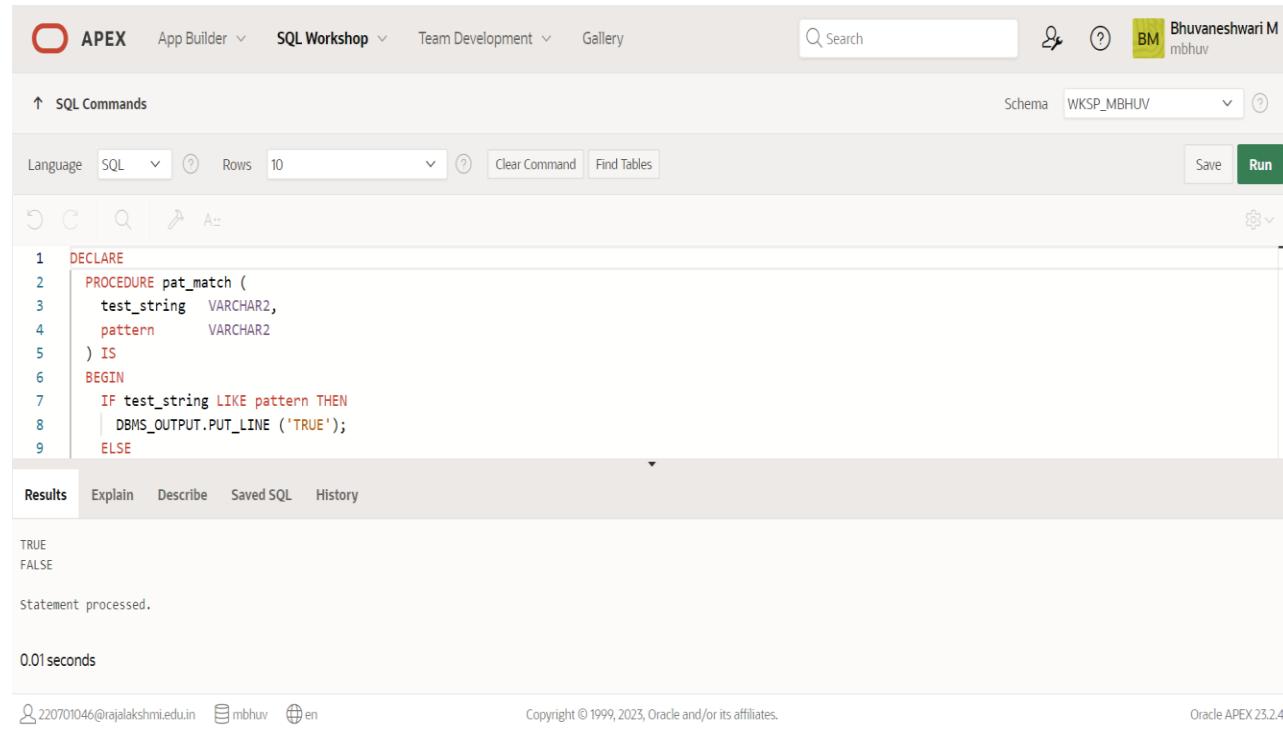
220701046@rajalakshmi.edu.in mbluv en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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 interface with the SQL Workshop tab selected. The code area displays the PL/SQL block provided above. The results section shows the output of the execution:

```
TRUE
FALSE
```

Below the results, a message states "Statement processed." and the execution time is shown as "0.01seconds".

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;

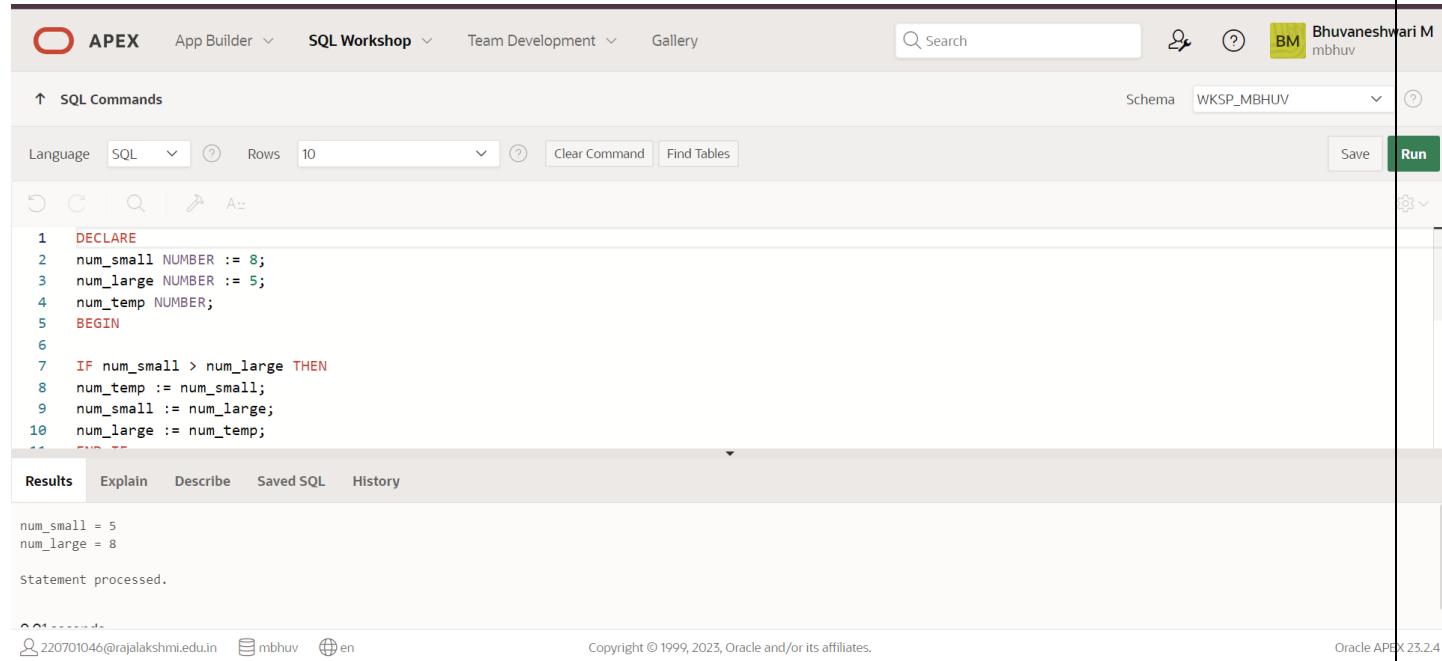
END IF;

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 Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. A search bar and user profile 'Bhuvaneshwari M mbhuv' are also present. The main workspace is titled 'SQL Commands' and contains the following PL/SQL code:

```
1 DECLARE
2 num_small NUMBER := 8;
3 num_large NUMBER := 5;
4 num_temp NUMBER;
5 BEGIN
6
7 IF num_small > num_large THEN
8 num_temp := num_small;
9 num_small := num_large;
10 num_large := num_temp;
11 END IF;
12
```

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

```
num_small = 5
num_large = 8

Statement processed.
```

At the bottom, footer information includes the URL '220701046@rajalakshmi.edu.in', the schema 'mbhuv', and the language 'en'. Copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 23.2.4' are also visible.

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

### QUERY:

DECLARE

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

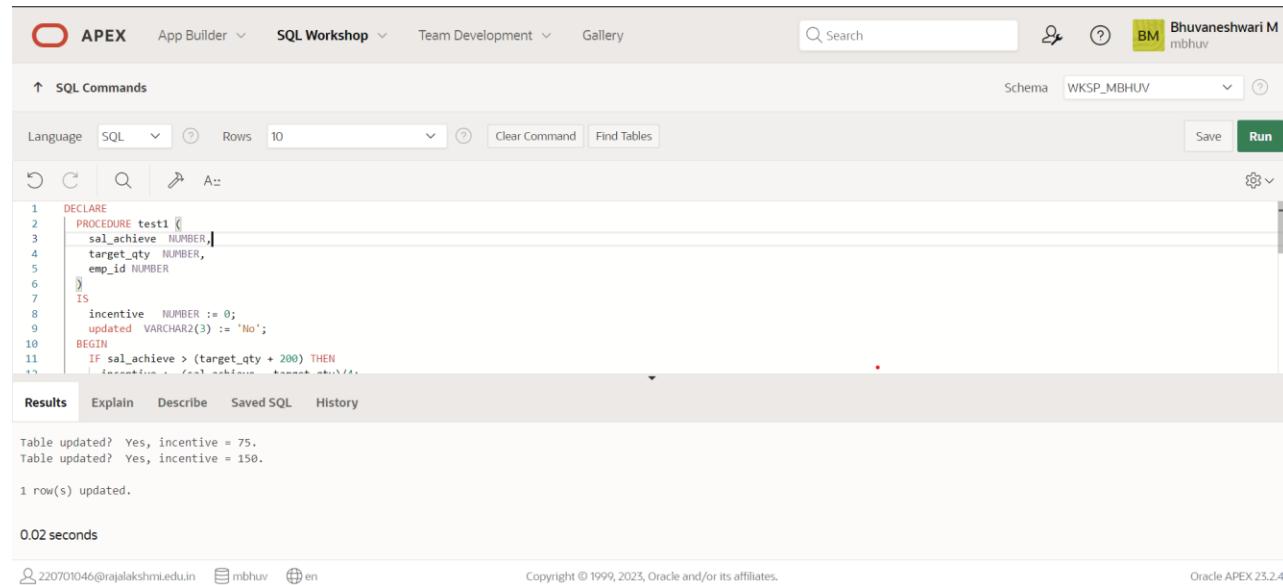
BEGIN

```
test1(2300, 2000, 144);
test1(3600, 3000, 145);
```

END;

/

### OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a user profile for 'Bhuvaneshwari M mbhuv'. The main workspace is titled 'SQL Commands' and contains the PL/SQL code provided above. The code is executed successfully, with the output showing two rows updated: one with an incentive of 75 and another with 150. The execution time is listed as 0.02 seconds. At the bottom, footer information includes the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 23.2.4'.

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

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

1 row(s) updated.

0.02 seconds

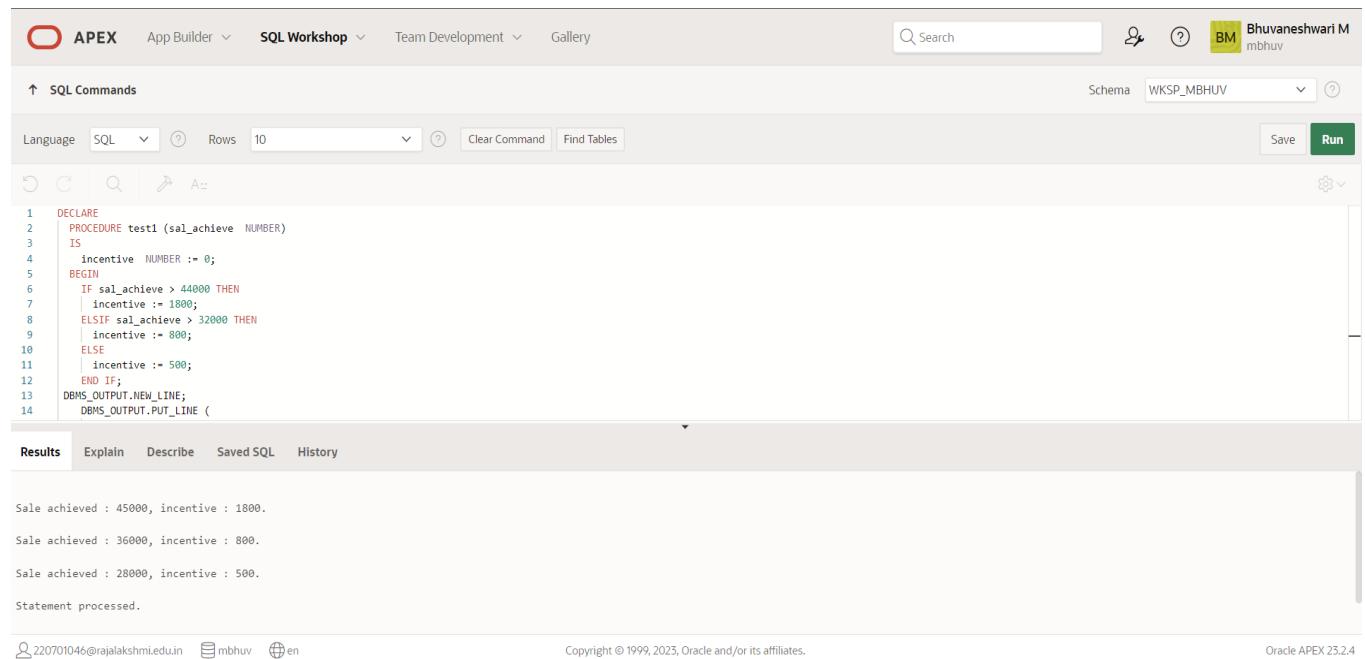
220701046@rajalakshmi.edu.in mbhuv en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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 Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for Bhuvaneshwari M (mbhuv), and a schema dropdown set to WKSP\_MBHV. The main workspace is titled "SQL Commands". It contains the PL/SQL code from the previous block. Below the code, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is active, displaying the output of the executed code: "Sale achieved : 45000, incentive : 1800.", "Sale achieved : 36000, incentive : 800.", "Sale achieved : 28000, incentive : 500.", and "Statement processed.". At the bottom, footer links include 220701046@rajalakshmi.edu.in, mbhuv, and en, along with copyright information: Copyright © 1999, 2023, Oracle and/or its affiliates. and Oracle APEX 23.2.4.

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:

```
1  DECLARE  
2      tot_emp NUMBER;  
3      get_dep_id NUMBER;  
4  BEGIN  
5      get_dep_id := 80;  
6      SELECT Count(*)  
7          INTO tot_emp  
8          FROM employees e  
9              join departments d  
10                 ON e.department_id = d.department_id  
11            WHERE e.department_id = get_dep_id;  
12            dbms_output.Put_line ('The employees are in the department'||get_dep_id||' is: '  
13                                ||To_char(tot_emp));  
14            IF tot_emp >= 45 THEN  
15                dbms_output.Put_line ('There are no vacancies in the department'||get_dep_id);  
16            ELSE  
17                dbms_output.Put_line ('There are'||to_char(45-tot_emp)||' vacancies in department'|| get_dep_id );  
18            END IF;  
19  END;  
20
```

The employees are in the department 80 is: 1  
There are 44 vacancies in department 80  
Statement processed.  
0.01seconds

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

### QUERY:

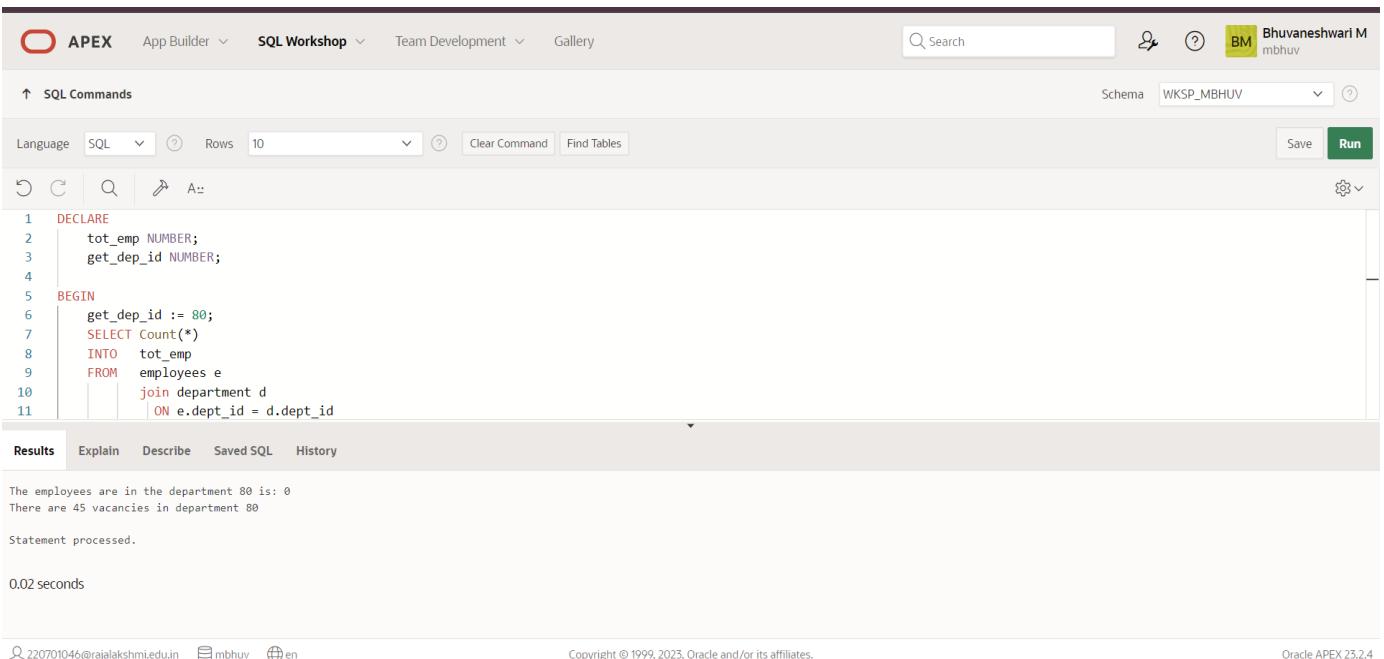
DECLARE

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

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

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

### OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows the user 'Bhuvaneshwari M mbhuv'. The main area is titled 'SQL Commands' with a 'Run' button. Below the toolbar, there are buttons for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. The code editor contains the provided PL/SQL block. The results tab shows the output of the executed code:

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

```

The employees are in the department 80 is: 0  
There are 45 vacancies in department 80  
Statement processed.  
0.02 seconds

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

**QUERY:**

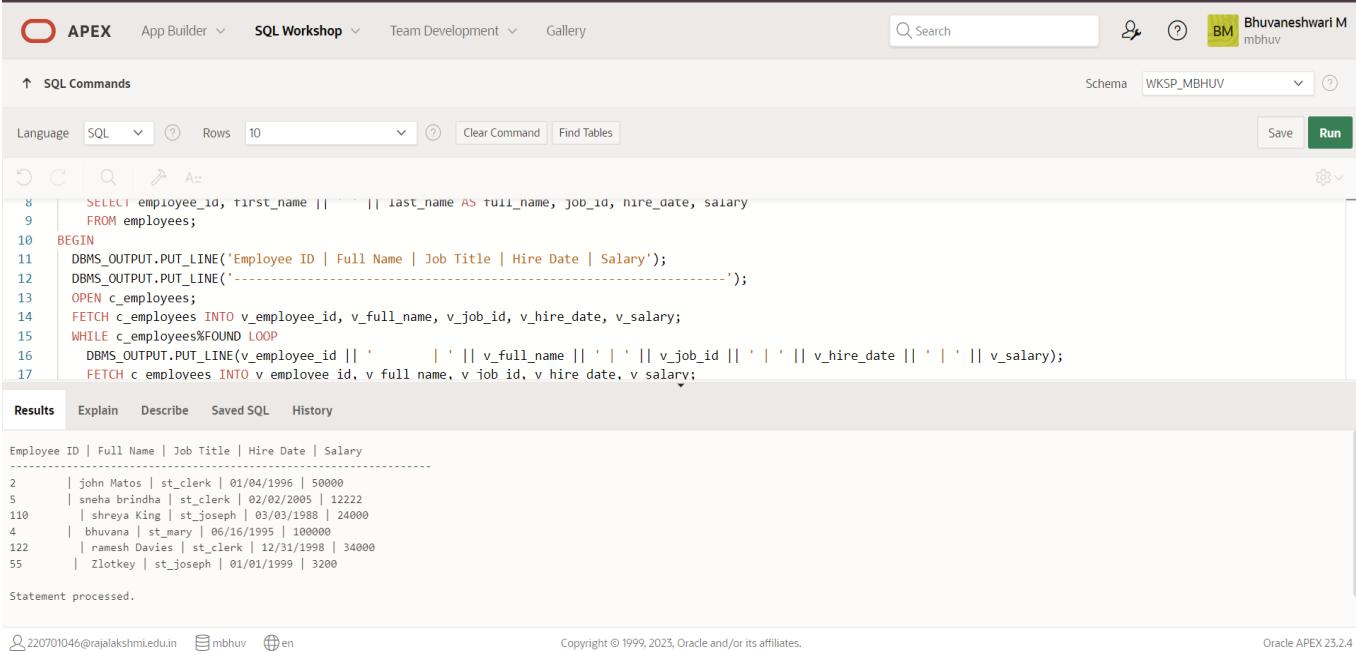
DECLARE

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

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

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

**OUTPUT:**



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

Statement processed.

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

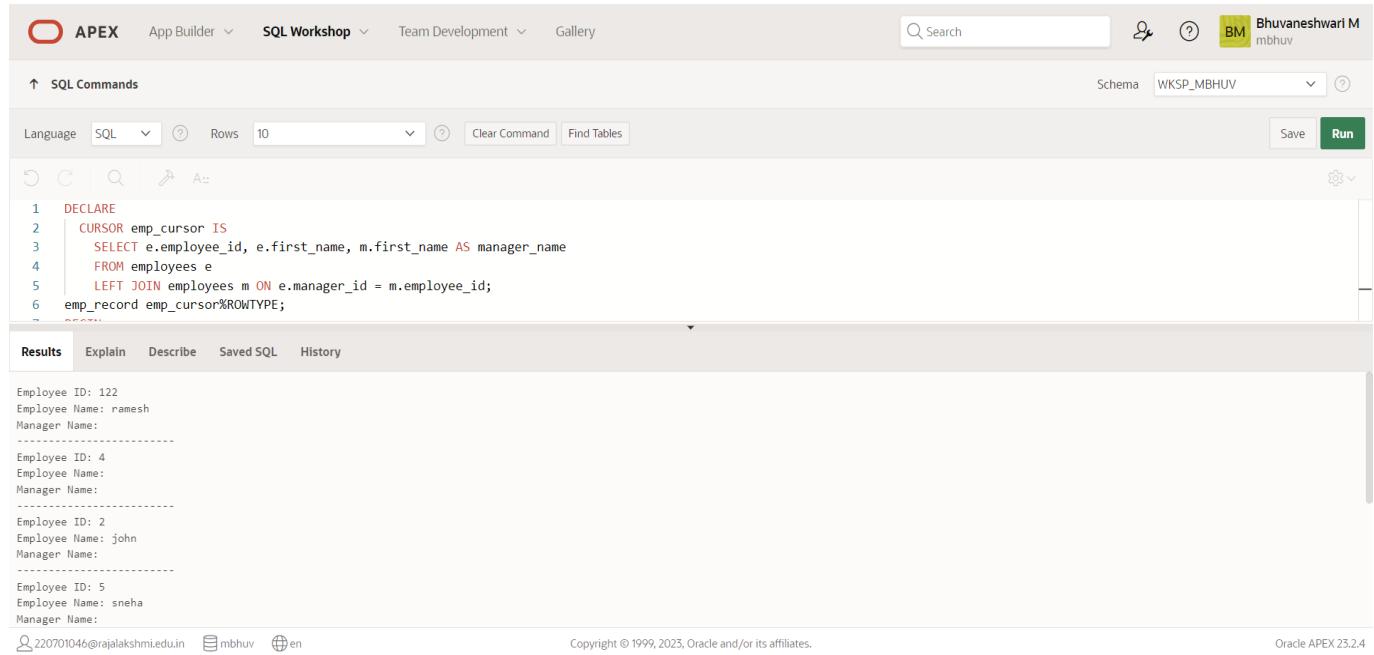
**12.) Write a PL/SQL program to display the employee IDs, names, and department names of all employees.**

**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 right side of the header shows the user 'Bhuvaneshwari M mbhuv' and the schema 'WKSP\_MBHV'. The main workspace has tabs for SQL Commands, Explain, Describe, Saved SQL, and History. The SQL Commands tab displays the PL/SQL code. The Results tab shows the output of the executed code, which lists four employees with their respective employee IDs, first names, and manager names. The output is as follows:

| Employee ID | Employee Name | Manager Name |
|-------------|---------------|--------------|
| 122         | ramesh        |              |
| 4           |               |              |
| 2           | john          |              |
| 5           | sneha         |              |

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile 'Bhuvaneshwari M mbhuv' are also present. The main workspace displays the PL/SQL code in the 'SQL Commands' tab. 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
```

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

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

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

**QUERY:**

DECLARE

```
CURSOR employees_cur IS
  SELECT employee_id, last_name, job_id, start_date
  FROM employees NATURAL JOIN job_history;
  emp_start_date DATE;
BEGIN
  dbms_output.Put_line(Rpad('Employee ID', 15) || Rpad('Last Name', 25) || Rpad('Job Id', 35)
  || 'Start Date');
  dbms_output.Put_line('-----');
FOR emp_sal_rec IN employees_cur LOOP
  -- find out most recent end_date in job_history
  SELECT Max(end_date) + 1
  INTO emp_start_date
  FROM job_history
  WHERE employee_id = emp_sal_rec.employee_id;
  IF emp_start_date IS NULL THEN
    emp_start_date := emp_sal_rec.start_date;
  END IF;
  dbms_output.Put_line(Rpad(emp_sal_rec.employee_id, 15)
    || Rpad(emp_sal_rec.last_name, 25)
    || Rpad(emp_sal_rec.job_id, 35)
    || To_char(emp_start_date, 'dd-mon-yyyy'));
END LOOP;
END;
/
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is signed in as 'Bhuvaneshwari M mbhuv'. The main workspace is titled 'SQL Commands' and contains the PL/SQL code from the previous step. The code is syntax-highlighted and shows the execution path with line numbers. Below the code, there is a results grid with columns for Employee ID, Last Name, Job Id, and Start Date. The results section displays the processed statement and execution time. At the bottom, there are footer links for copyright information and Oracle APEX version.

| Employee ID | Last Name | Job Id | Start Date |
|-------------|-----------|--------|------------|
|             |           |        |            |

Statement processed.  
0.03 seconds

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

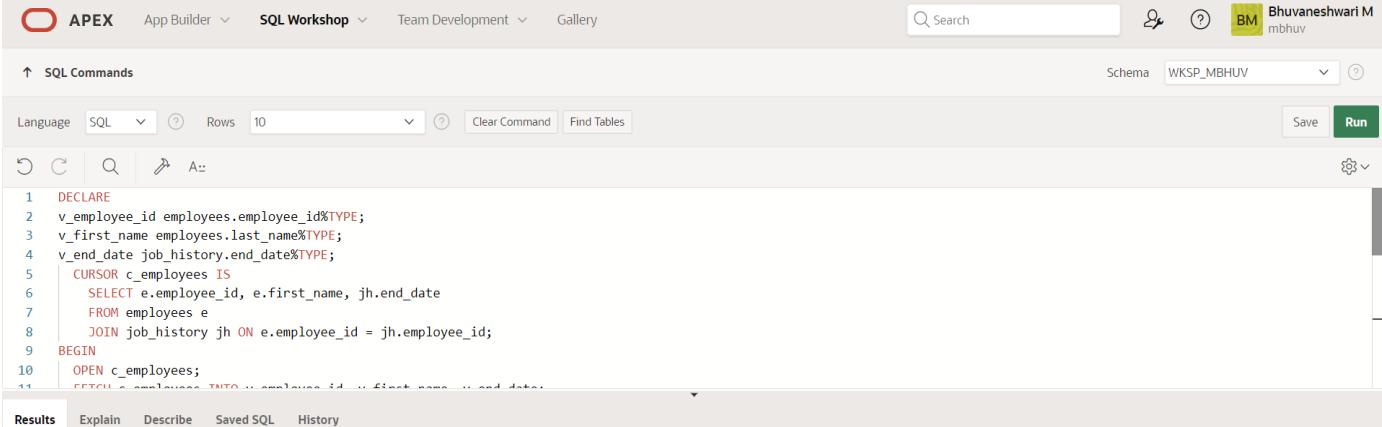
Oracle APEX 23.2.4

**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 APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. A search bar and user profile 'Bhuvaneshwari M mbhuv' are also present. The main workspace displays the PL/SQL code in the SQL Commands tab. The code declares variables for employee ID, first name, and end date, defines a cursor for employees, and uses a loop to fetch data and print it to the screen. The results tab shows the output of the code execution, which lists three employees with their IDs, names, and end dates.

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

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

**RESULT:**

The PL/SQL control structures has been successfully executed.

## PROCEDURES AND FUNCTIONS

EX.NO: 17

DATE:18/05/2024

**1.)Factorial of a number using function.**

**QUERY:**

DECLARE

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side of the header shows a user profile for 'Bhuvaneshwari M mbhuv' and a search bar. The main workspace is titled 'SQL Commands'. It contains a code editor with the following PL/SQL block:

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

Below the code editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, showing the output: '120'. Below the results, it says 'Statement processed.' and '0.01 seconds'. At the bottom of the page, there are footer links for '220701046@rajalakshmi.edu.in', 'mbhuv', and 'en', along with copyright information: 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

**QUERY:**

```
CREATE OR REPLACE PROCEDURE get_book_info (
    p_book_id IN NUMBER,
    p_title IN OUT VARCHAR2,
    p_author OUT VARCHAR2,
    p_year_published OUT NUMBER
)
AS
BEGIN
    SELECT title, author, year_published INTO p_title, p_author, p_year_published
    FROM books
    WHERE book_id = p_book_id;

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

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

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

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

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows the user profile "Bhuvaneshwari M mbhuv". 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     o_author OUT VARCHAR2.
```

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

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

Statement processed.

At the bottom, the footer includes the URL "220701046@rajalakshmi.edu.in", the schema "mbhuv", and the copyright notice "Copyright © 1999, 2023, Oracle and/or its affiliates". The version "Oracle APEX 23.2.4" is also mentioned.

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

## RESULT:

# TRIGGER

EX.NO: 18

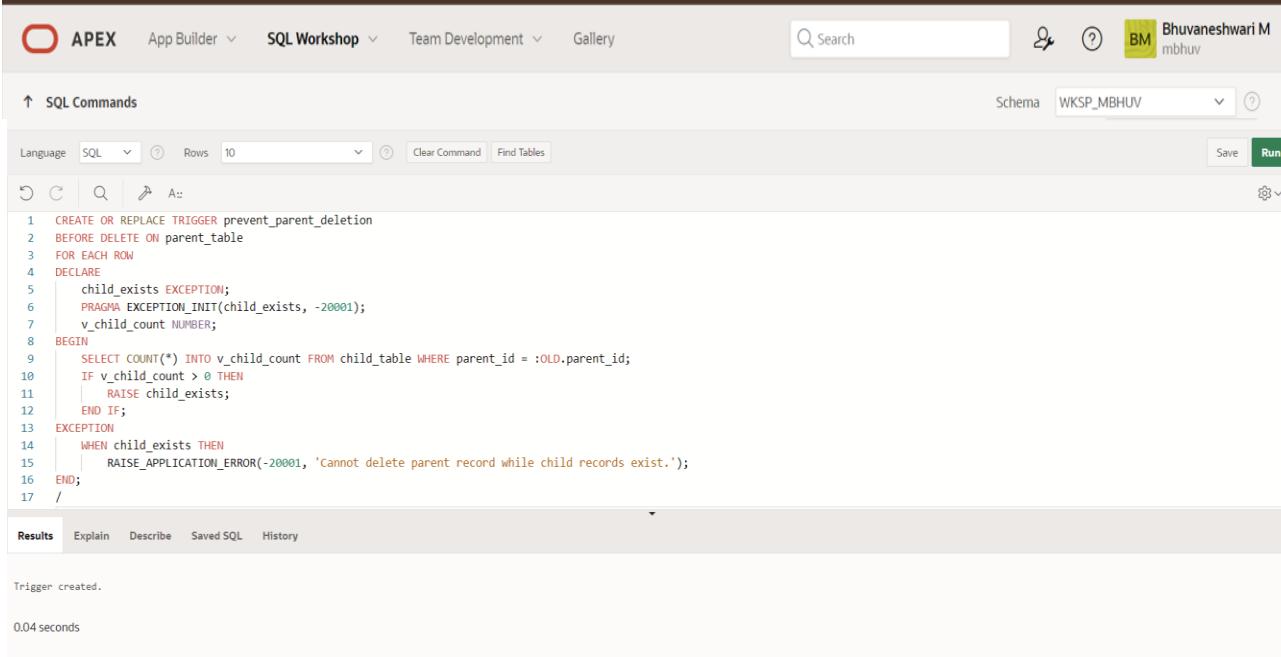
DATE:18/05/2024

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

## QUERY:

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

## OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right side, there's a user profile for 'Bhuvaneshwari M' (mbhuv). The main workspace is titled 'SQL Commands'. It contains the PL/SQL code for the trigger 'prevent\_parent\_deletion'. The code is syntax-highlighted, with keywords in blue and identifiers in green. The 'Run' button at the bottom right of the command input area is highlighted in green. Below the command input, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The results section displays the message 'Trigger created.' and '0.04 seconds'.

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

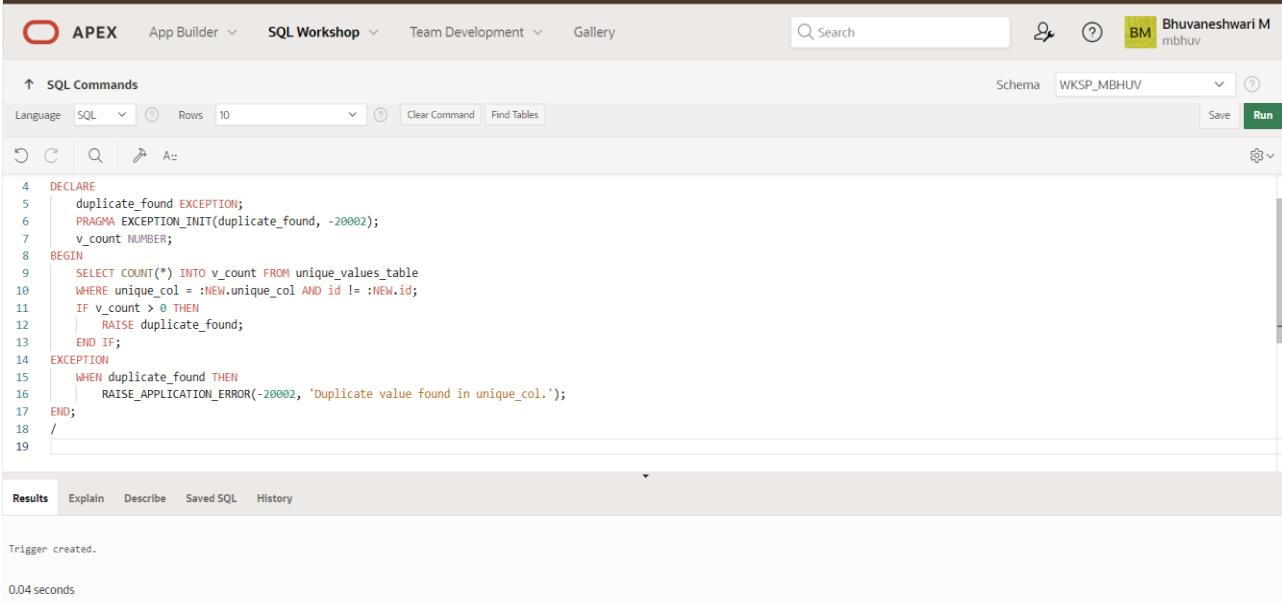
Trigger created.  
0.04 seconds

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

**QUERY:**

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

**OUTPUT:**



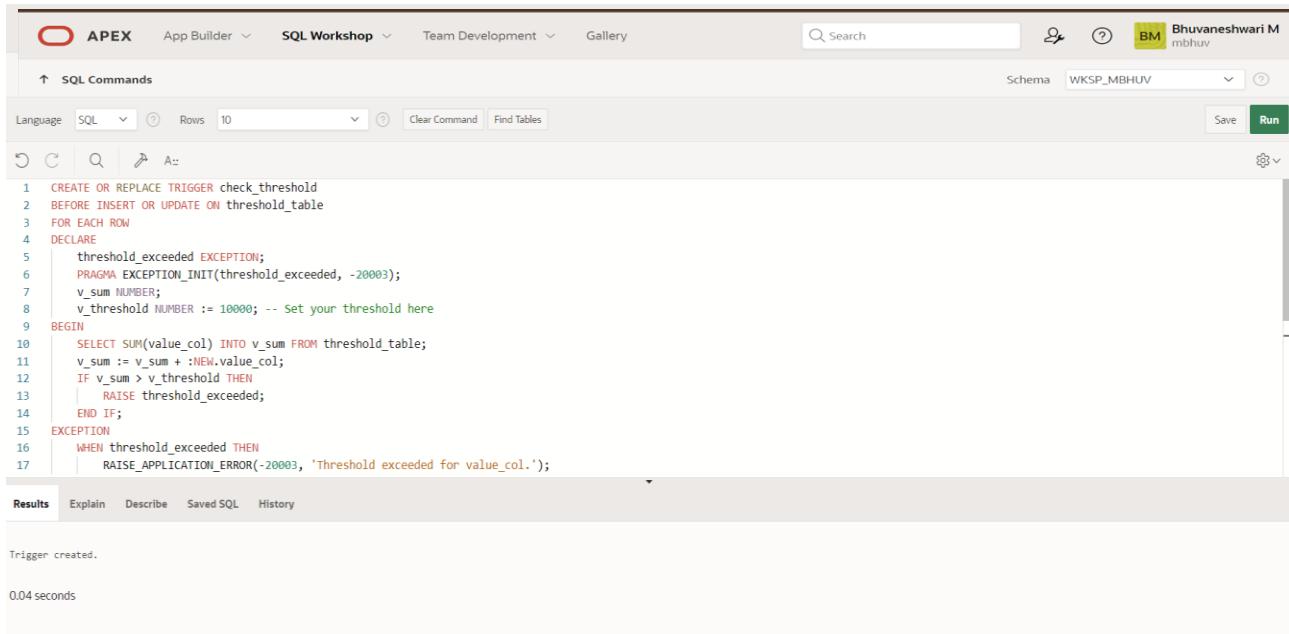
The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side of the header shows the user 'Bhuvaneshwari M mbhuv' and a 'Run' button. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_MBHUV'. The code area contains the PL/SQL trigger definition shown above. Below the code, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The status bar at the bottom indicates 'Trigger created.' and '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' (selected), 'Team Development', and 'Gallery'. A search bar and user profile 'Bhuvaneswari M mbhuv' are also present. The main workspace is titled 'SQL Commands' and shows the following SQL code:

```
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.');

```

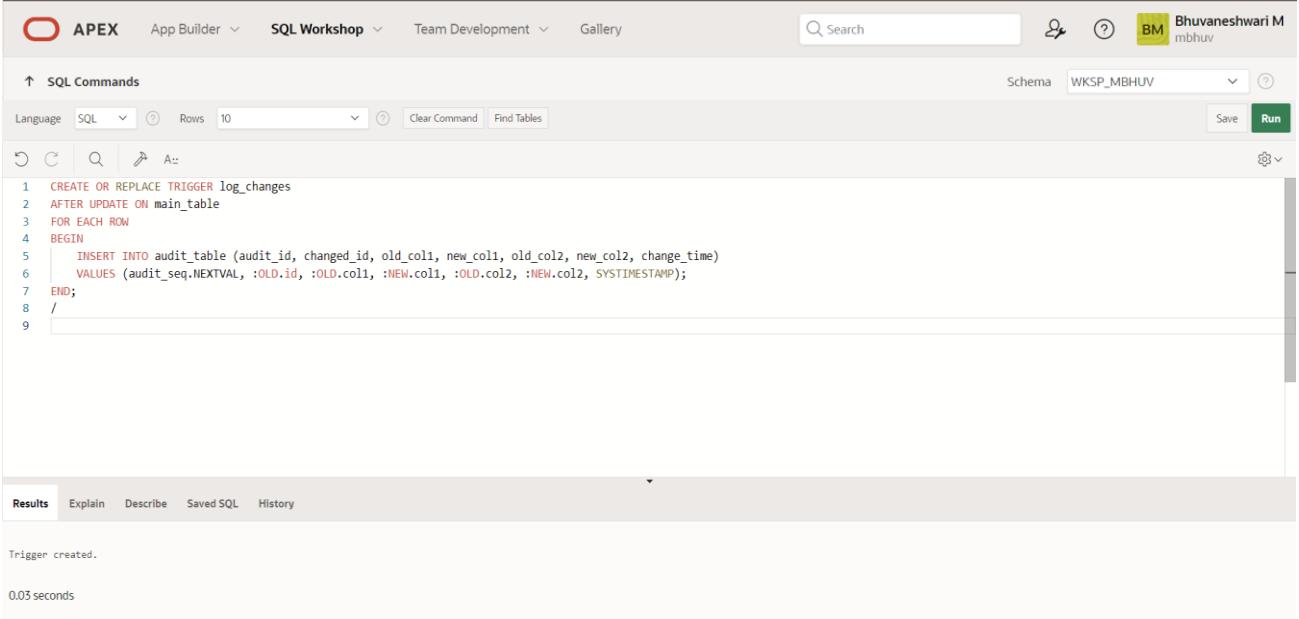
Below the code, the 'Results' tab is selected, displaying the message 'Trigger created.' and '0.04 seconds'.

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a user profile for 'Bhuvaneshwari M mbhuv'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_MBHV'. The code editor contains the PL/SQL trigger definition provided in the question. Below the code, the results tab is selected, showing the message 'Trigger created.' and a execution time of '0.03 seconds'.

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

Results Explain Describe Saved SQL History

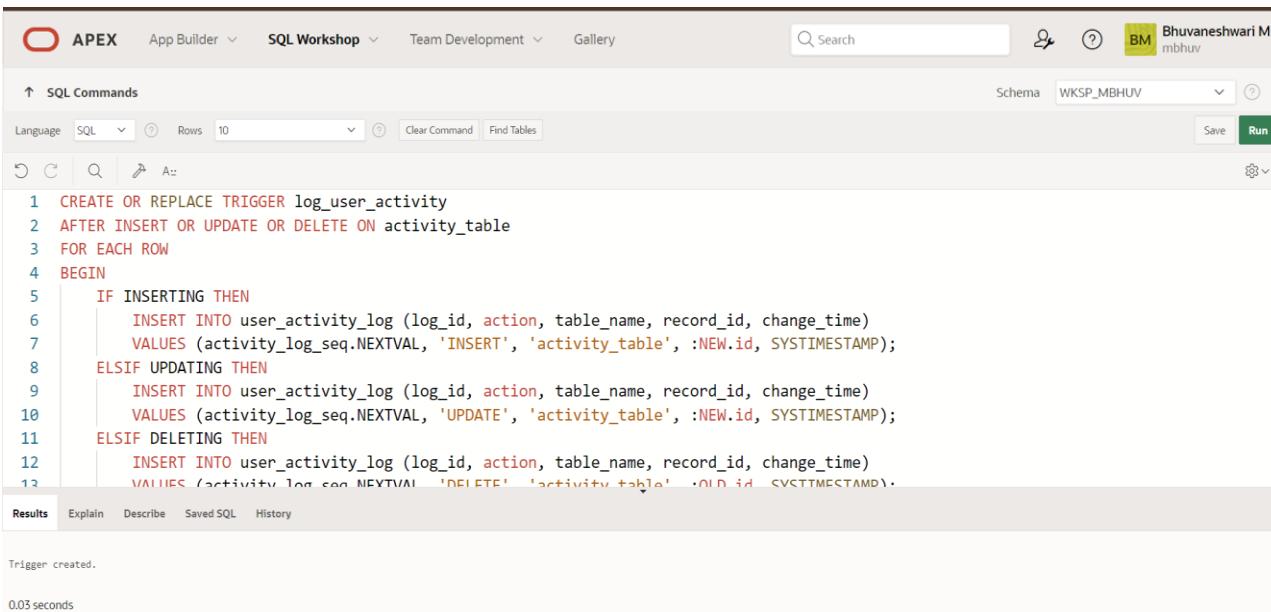
Trigger created.  
0.03 seconds

**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 (selected), Team Development, and Gallery. The right side shows a user profile for Bhuvaneshwari M (mbhuv). The main area is titled 'SQL Commands' with a 'Schema' dropdown set to WKSP\_MBHUV. The code area contains the PL/SQL trigger definition, which is highlighted in red. The code itself is as follows:

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

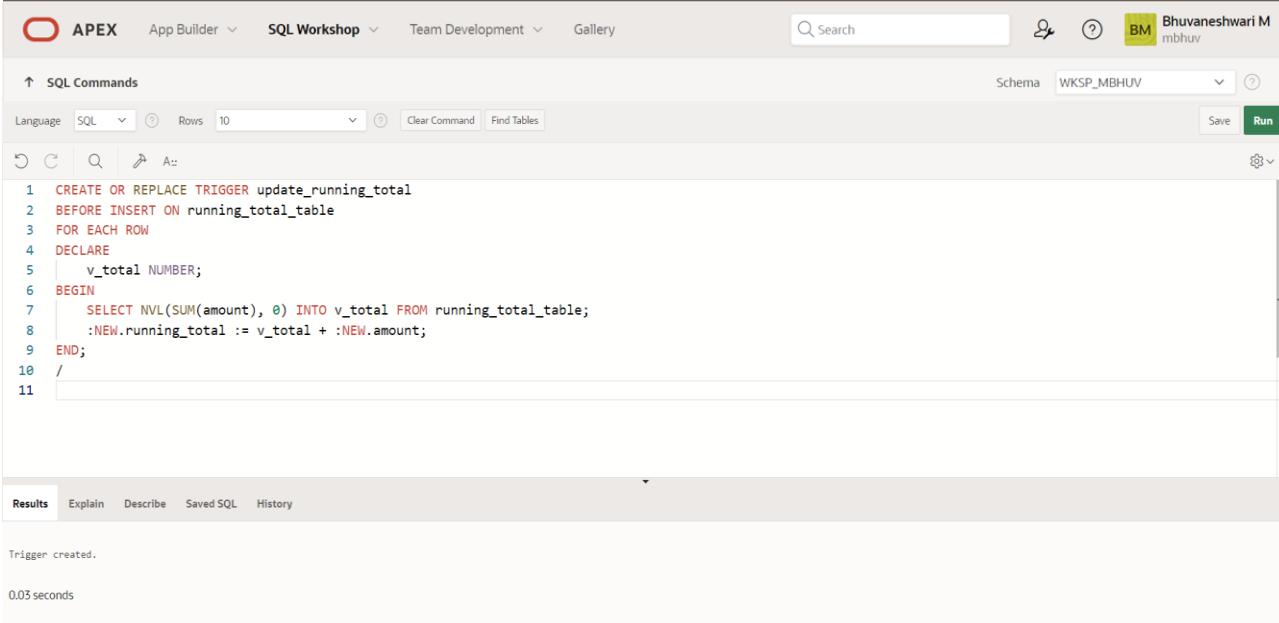
Below the code, the 'Results' tab is selected, showing the message "Trigger created." and a execution time of "0.03 seconds".

**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. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user's profile: Bhuvaneshwari M (mbhuv). The main area is titled "SQL Commands". The schema dropdown is set to WKSP\_MBHUV. The code editor contains the PL/SQL trigger definition provided in the question. The code is numbered from 1 to 11. Below the code, the "Results" tab is selected, showing the message "Trigger created." and "0.03 seconds".

```
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
```

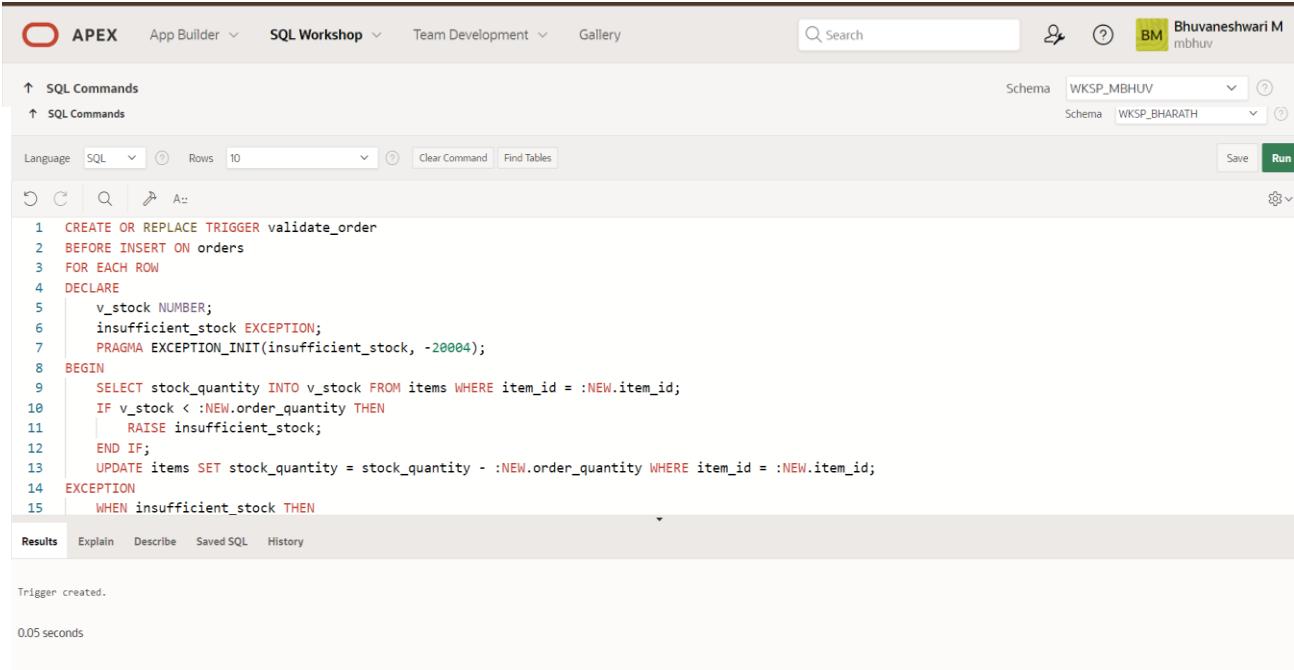
Trigger created.  
0.03 seconds

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user information ('Bhuvaneshwari M mbhuv') are also present. The main workspace displays the PL/SQL code for the 'validate\_order' trigger. The code is highlighted in red and black, indicating syntax. Below the code, the 'Results' tab is selected, showing the message 'Trigger created.' and a execution time of '0.05 seconds'. Other tabs like 'Explain', 'Describe', 'Saved SQL', and 'History' are visible at the bottom.

```
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
```

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

## **RESULT:**

The procedures and functions has been successfully executed.

# MONGO DB

EX.NO: 19

DATE:23/05/2024

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:**

```
bhuvaneshwari_46> db.restaurants.find( { grades: { $elemMatch: { grade: "A", score: 11, date: ISODate("2014-08-11T00:00:00Z") } } }, { restaurant_id: 1, name: 1, grades: 1 } );
```

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:**

```
bhuvaneshwari_46> db.restaurants.find( { grades: { $elemMatch: { grade: "A", score: 11, date: ISODate("2014-08-11T00:00:00Z") } } }, { restaurant_id: 1, name: 1, grades: 1 } );
```

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:**

```
bhuvaneshwari_46> 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 } );
```

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:**

```
bhuvaneshwari_46> db.restaurants.find({$and : [{ "address.coord.1": { $gt : 42 } }, {"address.coord.1": { $lte : 52 } }]}, {_id:0, restaurant_id:1, name:1, address:1})
```

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

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.restaurants.find( {}, { _id: 0 }).sort({ name: 1 })
[ {
    address: {
        building: '1009',
        coord: [ -73.856077, 40.848447 ],
        street: 'Morris Park Ave',
        zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
        {
            date: ISODate('2014-03-03T00:00:00.000Z'),
            grade: 'A',
            score: 2
        },
        {
            date: ISODate('2013-09-11T00:00:00.000Z'),
            grade: 'A',
            score: 6
        },
        {
            date: ISODate('2013-01-24T00:00:00.000Z'),
            grade: 'A',
            score: 10
        },
        {
            date: ISODate('2011-11-23T00:00:00.000Z'),
            grade: 'A',
            score: 9
        },
        {
            date: ISODate('2011-03-10T00:00:00.000Z'),
            grade: 'B',
            score: 14
        }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
}
```

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

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.restaurants.find( {}, { _id: 0 }).sort({ name: -1 })
[ {
    address: {
        building: '1007',
        coord: [ -73.856077, 40.848447 ],
        street: 'Morris Park Ave',
        zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
        {
            date: ISODate('2014-03-03T00:00:00.000Z'),
            grade: 'A',
            score: 2
        },
        {
            date: ISODate('2013-09-11T00:00:00.000Z'),
            grade: 'A',
            score: 6
        },
        {
            date: ISODate('2013-01-24T00:00:00.000Z'),
            grade: 'A',
            score: 10
        },
        {
            date: ISODate('2011-11-23T00:00:00.000Z'),
            grade: 'A',
            score: 9
        },
        {
            date: ISODate('2011-03-10T00:00:00.000Z'),
            grade: 'B',
            score: 14
        }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '36075445'
}]
```

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

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.restaurants.find( {}, { _id: 0 }).sort({ cuisine: 1, borough: -1 })
[ {
    address: {
        building: '1007',
        coord: [ -73.856077, 40.848447 ],
        street: 'Morris Park Ave',
        zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
        {
            date: ISODate('2014-03-03T00:00:00.000Z'),
            grade: 'A',
            score: 2
        },
        {
            date: ISODate('2013-09-11T00:00:00.000Z'),
            grade: 'A',
            score: 6
        },
        {
            date: ISODate('2013-01-24T00:00:00.000Z'),
            grade: 'A',
            score: 10
        },
        {
            date: ISODate('2011-11-23T00:00:00.000Z'),
            grade: 'A',
            score: 9
        },
        {
            date: ISODate('2011-03-10T00:00:00.000Z'),
            grade: 'B',
            score: 14
        }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '36075445'
}]
```

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

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46>db.restaurants.find({ "address.street": { $exists: true, $ne: "" } })
[ {
    _id: ObjectId('664f3c798752f54dc3cdcf7'),
    address: {
        building: '1809',
        coord: [ -73.856077, 40.848447 ],
        street: 'Morris Park Ave',
        zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
        {
            date: ISODate('2014-03-03T00:00:00.000Z'),
            grade: 'A',
            score: 2
        },
        {
            date: ISODate('2013-09-11T00:00:00.000Z'),
            grade: 'A',
            score: 6
        },
        {
            date: ISODate('2013-01-24T00:00:00.000Z'),
            grade: 'A',
            score: 10
        },
        {
            date: ISODate('2011-11-23T00:00:00.000Z'),
            grade: 'A',
            score: 9
        },
        {
            date: ISODate('2011-03-10T00:00:00.000Z'),
            grade: 'B',
            score: 14
        }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
}
```

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:**

```
bhuvaneshwari_46>db.restaurants.find({ "address.coord": { $elemMatch: { $type: "double" } } })
[ {
    _id: ObjectId('664f3c798752f54dc3cdcf7'),
    address: {
        building: '1809',
        coord: [ -73.856077, 40.848447 ],
        street: 'Morris Park Ave',
        zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
        {
            date: ISODate('2014-03-03T00:00:00.000Z'),
            grade: 'A',
            score: 2
        },
        {
            date: ISODate('2013-09-11T00:00:00.000Z'),
            grade: 'A',
            score: 6
        },
        {
            date: ISODate('2013-01-24T00:00:00.000Z'),
            grade: 'A',
            score: 10
        },
        {
            date: ISODate('2011-11-23T00:00:00.000Z'),
            grade: 'A',
            score: 9
        },
        {
            date: ISODate('2011-03-10T00:00:00.000Z'),
            grade: 'B',
            score: 14
        }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
}
```

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

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.restaurants.find({ "grades.score": { $mod: [7, 0] } }, { restaurant_id: 1, name: 1, grades: 1 })
[ {
    _id: ObjectId('664f3c798752f54dc3cdcdf7'),
    grades: [
        {
            date: ISODate('2014-03-03T00:00:00.000Z'),
            grade: 'A',
            score: 2
        },
        {
            date: ISODate('2013-09-11T00:00:00.000Z'),
            grade: 'A',
            score: 6
        },
        {
            date: ISODate('2013-01-24T00:00:00.000Z'),
            grade: 'A',
            score: 10
        },
        {
            date: ISODate('2011-11-23T00:00:00.000Z'),
            grade: 'A',
            score: 9
        },
        {
            date: ISODate('2011-03-10T00:00:00.000Z'),
            grade: 'B',
            score: 14
        }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
}]
```

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:**

```
bhuvaneshwari_46> db.restaurants.find({ name: /mon/i }, { name: 1, borough: 1, "address.coord": 1, cuisine: 1 })
```

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

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.restaurants.find({ name: /^Mad/i }, { name: 1, borough: 1, "address.coord": 1, cuisine: 1 })
```

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

**QUERY:**

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

## OUTPUT:

```
bhuvaneshwari_46> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } } })  
[  
  {  
    _id: ObjectId('664f3c798752f54dc3cdcdf7'),  
    address: {  
      building: '1007',  
      coord: [ -73.856077, 40.848447 ],  
      street: 'Morris Park Ave',  
      zipcode: '10462'  
    },  
    borough: 'Bronx',  
    cuisine: 'Bakery',  
    grades: [  
      {  
        date: ISODate('2014-03-03T00:00:00.000Z'),  
        grade: 'A',  
        score: 2  
      },  
      {  
        date: ISODate('2013-09-11T00:00:00.000Z'),  
        grade: 'A',  
        score: 6  
      },  
      {  
        date: ISODate('2013-01-24T00:00:00.000Z'),  
        grade: 'A',  
        score: 10  
      },  
      {  
        date: ISODate('2011-11-23T00:00:00.000Z'),  
        grade: 'A',  
        score: 9  
      },  
      {  
        date: ISODate('2011-03-10T00:00:00.000Z'),  
        grade: 'B',  
        score: 14  
      }  
    ],  
    name: 'Morris Park Bake Shop',  
    restaurant_id: '30075445'  
  }  
]
```

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:

```
bhuvaneshwari_46> restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, "borough": "Manhattan" })  
[  
  {  
    _id: ObjectId('664f3c798752f54dc3cdcdf7'),  
    address: {  
      building: '1007',  
      coord: [ -73.856077, 40.848447 ],  
      street: 'Morris Park Ave',  
      zipcode: '10462'  
    },  
    borough: 'Bronx',  
    cuisine: 'Bakery',  
    grades: [  
      {  
        date: ISODate('2014-03-03T00:00:00.000Z'),  
        grade: 'A',  
        score: 2  
      },  
      {  
        date: ISODate('2013-09-11T00:00:00.000Z'),  
        grade: 'A',  
        score: 6  
      },  
      {  
        date: ISODate('2013-01-24T00:00:00.000Z'),  
        grade: 'A',  
        score: 10  
      },  
      {  
        date: ISODate('2011-11-23T00:00:00.000Z'),  
        grade: 'A',  
        score: 9  
      },  
      {  
        date: ISODate('2011-03-10T00:00:00.000Z'),  
        grade: 'B',  
        score: 14  
      }  
    ],  
    name: 'Morris Park Bake Shop',  
    restaurant_id: '30075445'  
  }  
]
```

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:

```
bhuvaneshwari_46> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" } ])  
[  
  {  
    _id: ObjectId('664f3c798752f54dc3cdcdf7'),  
    address: {  
      building: '1007',  
      coord: [ -73.856077, 40.848447 ],  
      street: 'Morris Park Ave',  
      zipcode: '10462'  
    },  
    borough: 'Bronx',  
    cuisine: 'Bakery',  
    grades: [  
      {  
        date: ISODate('2014-03-03T00:00:00.000Z'),  
        grade: 'A',  
        score: 2  
      },  
      {  
        date: ISODate('2013-09-11T00:00:00.000Z'),  
        grade: 'A',  
        score: 6  
      },  
      {  
        date: ISODate('2013-01-24T00:00:00.000Z'),  
        grade: 'A',  
        score: 10  
      },  
      {  
        date: ISODate('2011-11-23T00:00:00.000Z'),  
        grade: 'A',  
        score: 9  
      },  
      {  
        date: ISODate('2011-03-10T00:00:00.000Z'),  
        grade: 'B',  
        score: 14  
      }  
    ],  
    name: 'Morris Park Bake Shop',  
    restaurant_id: '30075445'  
  }  
]
```

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:

```
bhuvaneshwari_46> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $ne: "American" } })
```

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

#### QUERY:

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

#### OUTPUT:

```
bhuvaneshwari_46> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $nin: ["American", "Chinese"] } })
```

18. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6.

#### QUERY:

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

#### OUTPUT:

```
bhuvaneshwari_46> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }] })
[ {
    _id: ObjectId('664f3c798752f54dc3cdcdf7'),
    address: {
        building: '1007',
        coord: [-73.856077, 40.848447],
        street: 'Morris Park Ave',
        zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
        {
            date: ISODate('2014-03-03T00:00:00.000Z'),
            grade: 'A',
            score: 2
        },
        {
            date: ISODate('2013-09-11T00:00:00.000Z'),
            grade: 'A',
            score: 6
        },
        {
            date: ISODate('2013-01-24T00:00:00.000Z'),
            grade: 'A',
            score: 10
        },
        {
            date: ISODate('2011-11-23T00:00:00.000Z'),
            grade: 'A',
            score: 9
        },
        {
            date: ISODate('2011-03-10T00:00:00.000Z'),
            grade: 'B',
            score: 14
        }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
}
```

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:**

```
bhuvaneshwari_46> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], "borough": "Manhattan" })
```

20. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn.

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }] })
```

21. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American.

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $ne: "American" } })
```

22. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American or Chinese.

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $nin: ["American", "Chinese"] } })
```

23. Write a MongoDB query to find the restaurants that have a grade with a score of 2 or a grade with a score of 6.

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.restaurants.find({ $or: [{ "grades.score": 2 }, { "grades.score": 6 }] })
[ {
    _id: ObjectId('664f3c798752f54dc3cdcdf7'),
    address: {
        building: '1007',
        coord: [ -73.856077, 40.848447 ],
        street: 'Morris Park Ave',
        zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
        {
            date: ISODate('2014-03-03T00:00:00.000Z'),
            grade: 'A',
            score: 2
        },
        {
            date: ISODate('2013-09-11T00:00:00.000Z'),
            grade: 'A',
            score: 6
        },
        {
            date: ISODate('2013-01-24T00:00:00.000Z'),
            grade: 'A',
            score: 10
        },
        {
            date: ISODate('2011-11-23T00:00:00.000Z'),
            grade: 'A',
            score: 9
        },
        {
            date: ISODate('2011-03-10T00:00:00.000Z'),
            grade: 'B',
            score: 14
        }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30875445'
}]
```

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

**RESULT:**

The Mongo DB queries are executed successfully.

## MONGO DB

EX.NO: 20

DATE:23/05/2024

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

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.movies.find({ year: 1893 })
```

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

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.movies.find({ runtime: { $gt: 120 } })
```

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

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.movies.find({ genres: 'Short' })
[ {
    _id: ObjectId('573a1390f29313caabcd42e8'),
    plot: 'A group of bandits stage a brazen train hold-up, only to find a determined posse hot on their heels.',
    genres: [ 'Short', 'Western' ],
    runtime: 11,
    cast: [
        'A.C. Abadie',
        "Gilbert M. 'Broncho Billy' Anderson",
        'George Barnes',
        'Justus D. Barnes'
    ],
    poster: 'https://m.media-amazon.com/images/M/MV5BMTU3NjE5NzYtYTYYNS00MDVmLWIwYjgtMmYwYWIxZDYYNzU2XkEyXkFqcGdeQXVvNzQzNzQxNzI@._V1_SV1000_SX677_AL_.jpg',
    title: 'The Great Train Robbery',
    fullplot: 'Among the earliest existing films in American cinema - notable as the first film that presented a narrative story to tell - it depicts a group of cowboy outlaws who hold up a train and rob the passengers. They are then pursued by a Sheriff's posse. Several scenes have color included - all hand tinted.',
    languages: [ 'English' ],
    released: ISODate('1903-12-01T00:00:00.000Z'),
    directors: [ 'Edwin S. Porter' ],
    rated: 'TV-G',
    awards: { wins: 1, nominations: 0, text: '1 win.' },
    lastupdated: '2015-08-13 00:27:59.177000000',
    year: 1903,
    imdb: { rating: 7.4, votes: 9847, id: 439 },
    countries: [ 'USA' ],
    type: 'movie',
    tomatoes: {
        viewer: { rating: 3.7, numReviews: 2559, meter: 75 },
        fresh: 6,
        critic: { rating: 7.6, numReviews: 6, meter: 100 },
        rotten: 0,
        lastUpdated: ISODate('2015-08-08T19:16:10.000Z')
    }
} ]
```

**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:**

```
bhuvaneshwari_46> db.movies.find({ directors: 'William K.L. Dickson' })
```

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

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.movies.find({ countries: 'USA' })
[
  {
    _id: ObjectId('573a1390f29313caabcd42e8'),
    plot: 'A group of bandits stage a brazen train hold-up, only to find a determined posse hot on their heels.',
    genres: ['Short', 'Western'],
    runtime: 11,
    cast: [
      'A.C. Abadie',
      "Gilbert M. 'Broncho Billy' Anderson",
      'George Barnes',
      'Justus D. Barnes'
    ],
    poster: 'https://m.media-amazon.com/images/M/MV5BMTU3NjE5NzYtYTYYNS00MDVmLWIwYjgtMmYwYWIxZDYYNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._V1_SY1000_SX677_AL_.jpg',
    title: 'The Great Train Robbery',
    fullplot: 'Among the earliest existing films in American cinema - notable as the first film that presented a narrative story to tell - it depicts a group of cowboy outlaws who hold up a train and rob the passengers. They are then pursued by a Sheriff's posse. Several scenes have color included - all hand tinted.',
    languages: ['English'],
    released: ISODate('1903-12-01T00:00:00.000Z'),
    directors: ['Edwin S. Porter'],
    rated: 'TV-G',
    awards: { wins: 1, nominations: 0, text: '1 win.' },
    lastupdated: '2015-08-13 00:27:59.177000000',
    year: 1903,
    imbd: { rating: 7.4, votes: 9847, id: 439 },
    countries: ['USA'],
    type: 'movie',
    tomatoes: {
      viewer: { rating: 3.7, numReviews: 2559, meter: 75 },
      fresh: 6,
      critic: { rating: 7.6, numReviews: 6, meter: 100 },
      rotten: 0,
      lastUpdated: ISODate('2015-08-08T19:16:10.000Z')
    }
  }
]
```

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

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.movies.find({ rated: 'UNRATED' })
```

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

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.movies.find({ 'imdb.votes': { $gt: 1000 } })
[ {
  _id: ObjectId('573a1390f29313caabcd42e8'),
  plot: 'A group of bandits stage a brazen train hold-up, only to find a determined posse hot on their heels.',
  genres: [ 'Short', 'Western' ],
  runtime: 11,
  cast: [
    'A.C. Abadie',
    "Gilbert M. 'Broncho Billy' Anderson",
    'George Barnes',
    'Justus D. Barnes'
  ],
  poster: 'https://m.media-amazon.com/images/M/MV5BMTU3NjE5NzYtYTYYNS00MDVmLWIwYjgtMmYwYWIxZDYyNzU2XkEyXkFqcGdeQXVvNzQzNzQxNzI@._V1_SY1000_SX677_AL_.jpg',
  title: 'The Great Train Robbery',
  fullplot: "Among the earliest existing films in American cinema - notable as the first film that presented a narrative story to tell - it depicts a group of cowboy outlaws who hold up a train and rob the passengers. They are then pursued by a Sheriff's posse. Several scenes have color included - all hand tinted.",
  languages: [ 'English' ],
  released: ISODate('1903-12-01T00:00:00.000Z'),
  directors: [ 'Edwin S. Porter' ],
  rated: 'TV-G',
  awards: { wins: 1, nominations: 0, text: '1 win.' },
  lastupdated: '2015-08-13 00:27:59.177000000',
  year: 1903,
  imdb: { rating: 7.4, votes: 9847, id: 439 },
  countries: [ 'USA' ],
  type: 'movie',
  tomatoes: {
    viewer: { rating: 3.7, numReviews: 2559, meter: 75 },
    fresh: 6,
    critic: { rating: 7.6, numReviews: 6, meter: 100 },
    rotten: 0,
    lastUpdated: ISODate('2015-08-08T19:16:10.000Z')
  }
}
```

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

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.movies.find({ 'imdb.rating': { $gt: 7 } })
[ {
  _id: ObjectId('573a1390f29313caabcd42e8'),
  plot: 'A group of bandits stage a brazen train hold-up, only to find a determined posse hot on their heels.',
  genres: [ 'Short', 'Western' ],
  runtime: 11,
  cast: [
    'A.C. Abadie',
    "Gilbert M. 'Broncho Billy' Anderson",
    'George Barnes',
    'Justus D. Barnes'
  ],
  poster: 'https://m.media-amazon.com/images/M/MV5BMTU3NjE5NzYtYTYYNS00MDVmLWIwYjgtMmYwYWIxZDYyNzU2XkEyXkFqcGdeQXVvNzQzNzQxNzI@._V1_SY1000_SX677_AL_.jpg',
  title: 'The Great Train Robbery',
  fullplot: "Among the earliest existing films in American cinema - notable as the first film that presented a narrative story to tell - it depicts a group of cowboy outlaws who hold up a train and rob the passengers. They are then pursued by a Sheriff's posse. Several scenes have color included - all hand tinted.",
  languages: [ 'English' ],
  released: ISODate('1903-12-01T00:00:00.000Z'),
  directors: [ 'Edwin S. Porter' ],
  rated: 'TV-G',
  awards: { wins: 1, nominations: 0, text: '1 win.' },
  lastupdated: '2015-08-13 00:27:59.177000000',
  year: 1903,
  imdb: { rating: 7.4, votes: 9847, id: 439 },
  countries: [ 'USA' ],
  type: 'movie',
  tomatoes: {
    viewer: { rating: 3.7, numReviews: 2559, meter: 75 },
    fresh: 6,
    critic: { rating: 7.6, numReviews: 6, meter: 100 },
    rotten: 0,
    lastUpdated: ISODate('2015-08-08T19:16:10.000Z')
  }
}
```

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

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.movies.find({ 'tomatoes.viewer.rating': { $gt: 4 } })
```

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

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.movies.find({ 'awards.wins': { $gt: 0 } })
[ {
  _id: ObjectId('573a1390f29313caabed42e8'),
  plot: 'A group of bandits stage a brazen train hold-up, only to find a determined posse hot on their heels.',
  genres: [ 'Short', 'Western' ],
  runtime: 11,
  cast: [
    'A.C. Abadie',
    "Gilbert M. 'Broncho Billy' Anderson",
    'George Barnes',
    'Justus D. Barnes'
  ],
  poster: 'https://m.media-amazon.com/images/M/MV5BMTU3NjE5NzYtYTYYNS00MDVmLWIwYjgtMmYWIxZDYyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._V1_SY1000_SX677_AL_.jpg',
  title: 'The Great Train Robbery',
  fullplot: "Among the earliest existing films in American cinema - notable as the first film that presented a narrative story to tell - it depicts a group of cowboy outlaws who hold up a train and rob the passengers. They are then pursued by a Sheriff's posse. Several scenes have color included - all hand tinted.",
  languages: [ 'English' ],
  released: ISODate('1903-12-01T00:00:00.000Z'),
  directors: [ 'Edwin S. Porter' ],
  rated: 'TV-G',
  awards: { wins: 1, nominations: 0, text: '1 win.' },
  lastupdated: '2015-08-13 00:27:59.177000000',
  year: 1903,
  imbd: { rating: 7.4, votes: 9847, id: 439 },
  countries: [ 'USA' ],
  type: 'movie',
  tomatoes: {
    viewer: { rating: 3.7, numReviews: 2559, meter: 75 },
    fresh: 6,
    critic: { rating: 7.6, numReviews: 6, meter: 100 },
    rotten: 0,
    lastUpdated: ISODate('2015-08-08T19:16:10.000Z')
  }
}
```

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

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.movies.find(
...   { 'awards.nominations': { $gt: 0 } },
...   { title: 1, languages: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 }
```

**12.) Find all movies with title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries from the 'movies' collection in MongoDB with cast including "Charles Kayser".**

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.movies.find(
...   { cast: 'Charles Kayser' },
...   { title: 1, languages: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 }
... )
```

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

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.movies.find(
...   { released: ISODate("1893-05-09T00:00:00.000Z") },
...   { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 }
... )
```

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

**QUERY:**

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

**OUTPUT:**

```
bhuvaneshwari_46> db.movies.find(
...   { title: /scene/i },
...   { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 }
... )
```

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

## **RESULT:**

The Mongo DB queries are executed successfully.

