

# Creating and Managing Tables

EX\_NO:1

DATE:

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

Column name	ID	NAME
<b>Key Type</b>		
<b>Nulls/Unique</b>		
<b>FK table</b>		
<b>FK column</b>		
<b>Data Type</b>	Number	Varchar2
<b>Length</b>	7	25

## QUERY:

```
Create table department(id number(7),name varchar2(25));
```

## OUTPUT:

The screenshot shows the Oracle Application Express interface. The URL is 127.0.0.1:8080/apex/f?p=4500:1003:399704901457789::NO:::. The page title is SQL Commands. The SQL command entered is:

```
Create table DEPARTMENT(
ID Number(7),
Name varchar2(25)
);
```

The results section shows the output:

```
Table created.  
0.02 seconds
```

At the bottom, it says Application Express 4.0.2.00.09 and Workspace: BALAJI36 User: BALAJI36 Language: en | Copyright © 1995-2010, Oracle. All rights reserved.

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 EMI(id number(7),Last\_Name varchar2(25),First\_Name varchar2(25),Dept\_id number(7));

### OUTPUT:

The screenshot shows the Oracle Application Express SQL Workshop interface. The URL in the address bar is 127.0.0.1:8080/apex/f?p=4500:1003:399704901457789::NO:::. The page title is "SQL Commands". The top navigation bar includes Home, Application Builder, SQL Workshop (selected), Team Development, Administration, and a user icon. The SQL Commands tab is active. The main area contains the SQL command for creating the EMI table:

```
create table EMI(
ID Number(7),
LAST_NAME varchar2(25),
FIRST_NAME varchar2(25),
DEPT_ID Number(7)
);
```

Below the SQL command, the results are displayed:

Table created.  
0.00 seconds

At the bottom, the footer indicates "Application Express 4.0.2.00.09" and "Workspace: BALAJI36 User: BALAJI36".

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

**QUERY:**

```
Alter table EMI modify(Last_Name varchar2(25));
```

**OUTPUT:**

The screenshot shows a browser window for Oracle Application Express. The URL is 127.0.0.1:8080/apex/f?p=4500:1003:399704901457789::NO:::. The page title is "SQL Commands". The main content area contains the SQL command: "Alter table EMI modify(LAST\_NAME varchar2(50));". Below the command, the results show "Table altered." and "0.01 seconds". The bottom status bar indicates "Workspace: BALAJI36 User: BALAJI36" and "Application Express 4.0.2.0.0.09".

```
Alter table EMI modify(LAST_NAME varchar2(50));
```

Table altered.  
0.01 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 E2(id number(7),First\_name varchar2(25),Last\_name varchar2(25),Salary int,Dept\_id number(7));

**OUTPUT:**

The screenshot shows the Oracle Application Express interface. In the top navigation bar, the path is Home > SQL Workshop > SQL Commands. The main area contains the following SQL code:

```
Create table E2(
ID Number(6),
First_name varchar(20),
Last_name varchar(25),
Salary Number(8,2),
Dept_id Number(4)
);
```

Below the code, the results of the execution are shown:

Table created.  
0.00 seconds

In the bottom right corner, the footer indicates "Application Express 4.0.2.00.09" and "Language: en | Copyright © 1999, 2010, Oracle. All rights reserved."

5.Drop the EMP table.

**QUERY:**

Drop table emp;

## OUTPUT:

The screenshot shows the Oracle Application Express interface. In the top navigation bar, the path is Home > SQL Workshop > SQL Commands. The main area contains the following SQL command:

```
Drop table EM1;
```

Below the command, the results are displayed:

```
Table dropped.  
0.03 seconds
```

At the bottom of the page, the footer includes the text "Workspace: BALAJI36 User: BALAJI36" and "Application Express 4.0.2.00.09".

6.Rename the EMPLOYEES2 table as EMP.

## QUERY:

```
Rename employees2 to emp;
```

## OUTPUT:

The screenshot shows the Oracle Application Express interface. In the top navigation bar, the path is Home > SQL Workshop > SQL Commands. The main area contains the following SQL command:

```
Rename E2 to ENI;
```

Below the command, the results show:

Statement processed.  
0.01 seconds

At the bottom, it indicates the workspace is BALAJI36 and the user is BALAJI36. The application version is Application Express 4.0.2.0.0.0.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX interface. In the top navigation bar, the path is Home > SQL Workshop > SQL Commands. The main area contains the following SQL commands:

```
1 COMMENT ON TABLE DEPARTMENT IS 'DEPARTMNET INFO';
2 COMMENT ON TABLE EMPLOYEE IS 'EMPLOYEE INFO';
3
```

Below the commands, the results show:

Statement processed.  
0.03 seconds

At the bottom, it indicates the session user is 220701036@rajalakshmi.edu.in and the schema is WKSP\_D36. The application version is Oracle APEX 23.2.4. The system tray at the bottom shows weather information (29°C, High winds soon), system icons, and the date/time (04-03-2024, 10:59).

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

**QUERY:**

```
Alter table emp drop column First_name;
```

**OUTPUT:**

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

```
Alter table emp drop column First_Name;
```

Below the command, the output shows:

```
Table altered.  
0.05 seconds
```

In the bottom right corner of the interface, it says "Application Express 4.0.2.00.09".

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

**RESULT:**

# MANIPULATING DATA

**EX\_NO:**2

**DATE:**

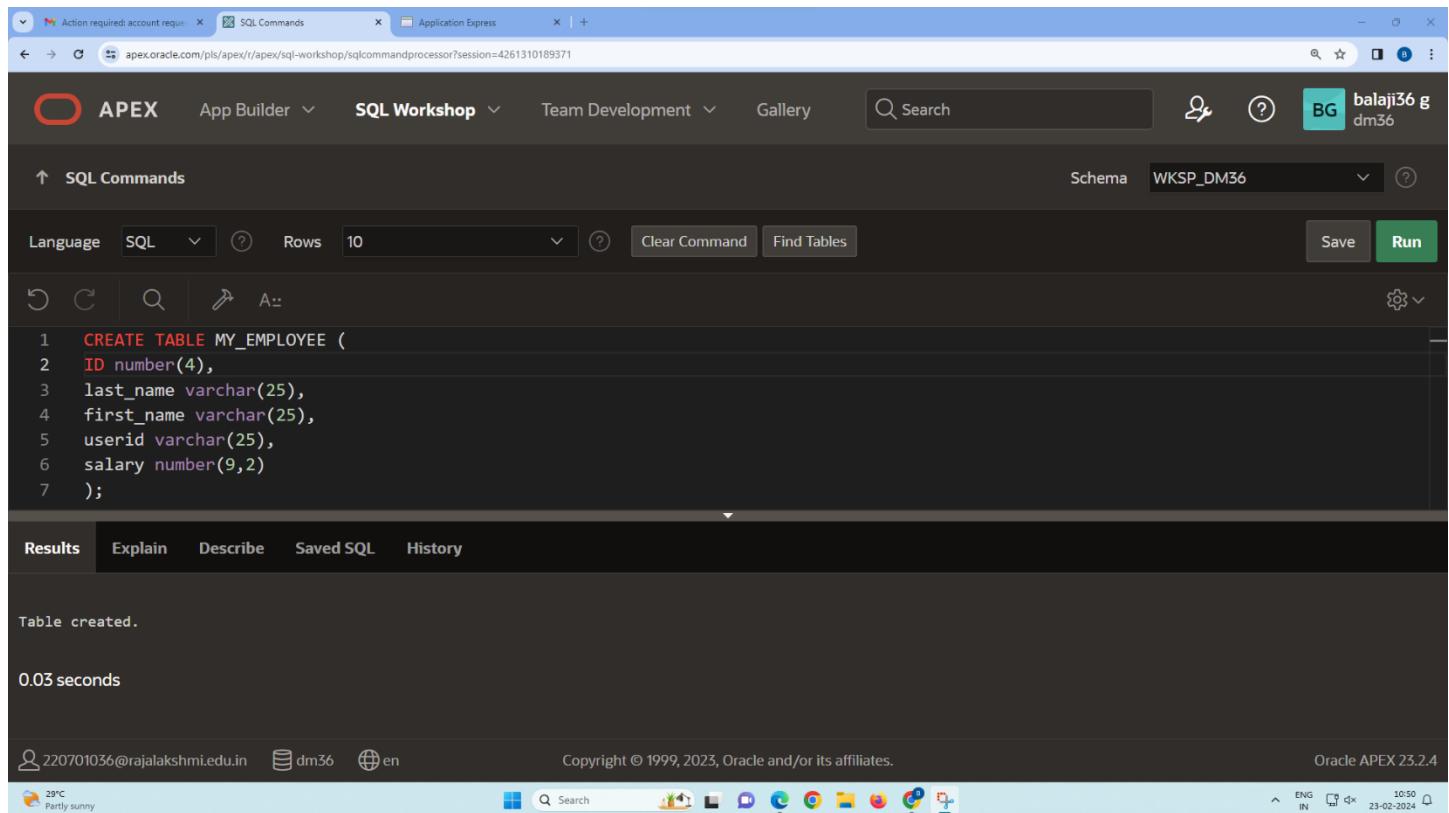
1.Create MY\_EMPLOYEE table with the following structure

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

**QUERY:**

Create table MY\_EMPLOYEE(ID number(4),last\_name varchar(25),first\_name varchar(25),userid varchar(25),salary number(9,2));

## OUTPUT:



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

```
1 CREATE TABLE MY_EMPLOYEE (
2   ID number(4),
3   last_name varchar(25),
4   first_name varchar(25),
5   userid varchar(25),
6   salary number(9,2)
7 );
```

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

```
Table created.
```

Execution time: 0.03 seconds

The bottom status bar shows the user's email (220701036@rajalakshmi.edu.in), session (dm36), and locale (en). It also displays the copyright notice "Copyright © 1999, 2023, Oracle and/or its affiliates." and the version "Oracle APEX 23.2.4".

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

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

## QUERY:

Insert into MY\_EMPLOYEE VALUES(4,'Newman','chad','cnewman',750);

## OUTPUT:

Action required: account required

SQL Commands Application Express

apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=4261310189371

APEX App Builder SQL Workshop Team Development Gallery Search Schema: WKSP\_DM36

SQL Commands

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

1. `INSERT INTO MY_EMPLOYEE VALUES(4, 'Newman', 'Chad', 'Cnewman', 750);`

2.

3.

Results Explain Describe Saved SQL History

1 row(s) inserted.

0.01 seconds

220701036@rajalakshmi.edu.in dm36 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

Today's high To break record

Search

ENG IN 11:11 23-02-2024

3. Display the table with values.

**QUERY:**

`Select * from MY_EMPLOYEE;`

**OUTPUT:**

Action required: account required

SQL Commands Application Express

apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=4261310189371

APEX App Builder SQL Workshop Team Development Gallery Search Schema: WKSP\_DM36

SQL Commands

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

1. `select * from MY_EMPLOYEE;`

2.

3.

Results Explain Describe Saved SQL History

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
4	Newman	Chad	Cnewman	750
3	Biri	Ben	bbiri	1100

220701036@rajalakshmi.edu.in dm36 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

30°C Haze

Search

ENG IN 11:16 23-02-2024

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

#### QUERY:

**Insert into MY\_EMPLOYEE VALUES(2,'Dancs','Betty','bdancs',860);**

#### OUTPUT:

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

```
1  INSERT INTO MY_EMPLOYEE VALUES(2, 'Dancs', 'Betty', 'bdancs', 860);
2
3
```

After running the command, the Results tab displays the output:

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

The status bar at the bottom indicates "0.00 seconds" for the execution time.

5.Make the data additions permanent.

#### QUERY:

**Select \* from MY\_EMPLOYEE;**

## OUTPUT:

The screenshot shows the Oracle APEX interface with the 'SQL Workshop' tab selected. In the main area, there is a table titled 'Results' displaying employee data. The table has columns: ID, LAST\_NAME, FIRST\_NAME, USERID, and SALARY. There are two rows: one for employee ID 4 with last name Newman, first name Chad, user ID Cnewman, and salary 1000; and another for employee ID 3 with last name Drexler, first name Ben, user ID bbiri, and salary 1100.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
4	Newman	Chad	Cnewman	1000
3	Drexler	Ben	bbiri	1100

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

## QUERY:

```
UPDATE MY_EMPLOYEE SET last_name = 'Drexler' where ID=3;
```

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, there are tabs for 'APEX', 'App Builder', 'SQL Workshop' (which is currently selected), 'Team Development', 'Gallery', and a search bar. On the right, there's a user profile for 'balaji36 g dm36'. Below the navigation, the 'SQL Commands' page is displayed. The 'Language' dropdown is set to 'SQL'. The 'Rows' dropdown is set to '10'. There are buttons for 'Clear Command' and 'Find Tables'. On the far right, there are 'Save' and 'Run' buttons. The main area contains the following SQL code:

```
1 UPDATE MY_EMPLOYEE SET last_name='Drexler'
2 where ID=3;
3
4
```

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

1 row(s) updated.  
0.01 seconds

The bottom status bar shows the user's email (220701036@rajalakshmi.edu.in), session (dm36), and language (en). It also displays the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 23.2.4'. On the far right, it shows system information like '30°C Haze', 'ENG IN', and the date '23-02-2024'.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface, identical to the previous one but with different SQL code. The 'SQL Commands' page is selected, and the 'Language' dropdown is set to 'SQL'. The 'Rows' dropdown is set to '10'. There are buttons for 'Clear Command' and 'Find Tables'. On the far right, there are 'Save' and 'Run' buttons. The main area contains the following SQL code:

```
1 UPDATE MY_EMPLOYEE SET salary=1000
2 where salary<900;
3
4
```

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

3 row(s) updated.  
0.00 seconds

The bottom status bar shows the user's email (220701036@rajalakshmi.edu.in), session (dm36), and language (en). It also displays the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 23.2.4'. On the far right, it shows system information like 'Breaking news Unfolding now', 'ENG IN', and the date '23-02-2024'.

8.Delete Betty dancs from MY\_EMPLOYEE table.

**QUERY:**

**Delete from MY\_EMPLOYEE where first\_name='Betty';**

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, a single-line delete statement is entered:

```
1 delete from MY_EMPLOYEE WHERE First_name='Betty';
2
3
```

After running the command, the Results tab displays the output:

1 row(s) deleted.  
0.01 seconds

The bottom status bar shows the user's email (220701036@rajalakshmi.edu.in), session ID (dm36), and language (en). It also includes a weather widget (30°C Haze), a system menu, and system status indicators (ENG IN, 4x, 11:27, 23-02-2024).

9.Empty the fourth row of the emp table.

**QUERY:**

**Delete from MY\_EMPLOYEE where ID=4;**

**OUTPUT:**

Action required: account required

SQL Commands Application Express

apex.oracle.com/pls/apex/i/apex/sql-workshop/sqlcommandprocessor?session=4261310189571

**APEX** App Builder **SQL Workshop** Team Development Gallery Search **balaji36 g dm36**

**SQL Commands** Schema WKSP\_DM36

Language SQL Rows 10 Clear Command Find Tables Save Run

undo redo search filter A..

```
1 delete from MY_EMPLOYEE WHERE First_name='Betty';
2
3
```

**Results** Explain Describe Saved SQL History

1 row(s) deleted.  
0.01 seconds

220701036@rajalakshmi.edu.in dm36 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

30°C Haze

Search ENG IN 11:27 23-02-2024

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

## RESULT:

# INCLUDING CONSTRAINTS

EX\_NO:3

DATE:

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

**QUERY:**

**Alter table EMP**

**Add constraint my\_emp\_id\_pk PRIMARY KEY(ID);**

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, the user is in the 'SQL Commands' section. The schema is set to 'WKSP\_D36'. The SQL editor contains the following code:

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

The 'Run' button is highlighted in green. Below the editor, the results pane shows the output of the command:

```
Table altered.
0.08 seconds
```

On the right side of the screen, there is a sidebar titled 'History' which displays a search result for 'blackbox ai' from 78 days ago, showing links to 'Blackbox' and 'blackbox.ai'.

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

#### QUERY:

```
Create table Dept(My_dept_id_pk int,Dept_name varchar(20),Manager_name varchar(25),Location_id varchar(25),Primary key(My_dept_id_pk));
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, there are tabs for 'APEX', 'App Builder', 'SQL Workshop' (which is selected), 'Team Development', and 'Gallery'. Below the navigation bar, there is a search bar and a dropdown for 'Schema' set to 'WKSP\_DM36'. The main area is titled 'SQL Commands' and contains a code editor with the following SQL command:

```
1 CREATE TABLE Dept
2 (
3   My_dept_id_pk int,
4   Dept_name varchar(20),
5   Manager_name varchar(25),
6   Location_id varchar(25),
7   PRIMARY KEY(My_dept_id_pk)
8 );
```

Below the code editor, there are buttons for 'Save' and 'Run'. The results tab is selected, showing the output of the query:

```
Table created.
0.07 seconds
```

At the bottom of the page, there is footer information including the session ID and copyright notice: 'Copyright © 1999-2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

#### QUERY:

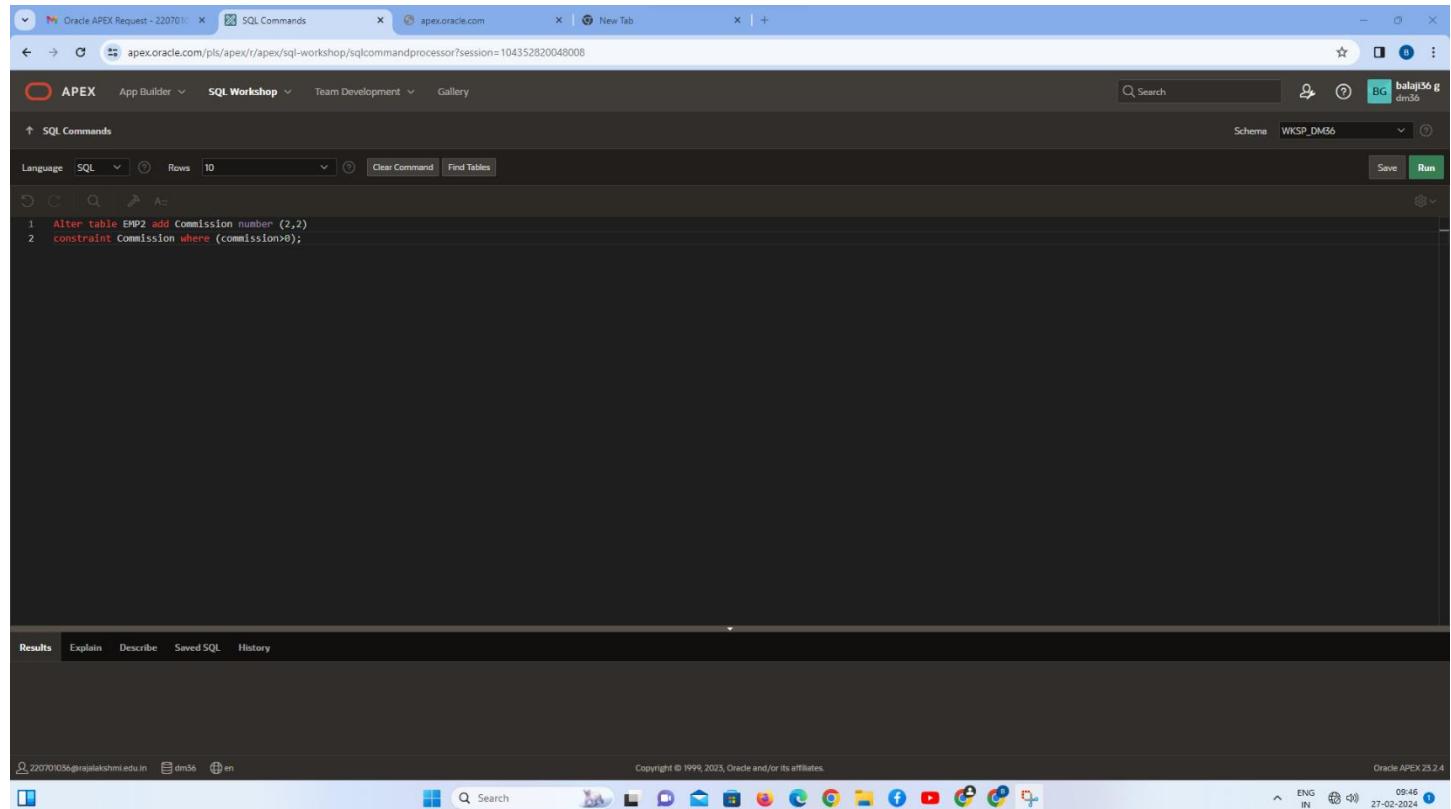
## OUTPUT:

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

## QUERY:

**Alter table EMP2 add Commission number (2,2)  
Constraint Commission where (Commission>0);**

## OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes tabs for Oracle APEX Request - 220701, SQL Commands, apex.oracle.com, New Tab, and various APEX modules like App Builder, SQL Workshop, Team Development, and Gallery. The user is logged in as 'balajis6 g' with schema 'WKSP\_DM36'. The SQL Commands tab is active, showing the following SQL code:

```
1 Alter table EMP2 add Commission number (2,2)
2 constraint Commission where (commission>0);
```

The 'Run' button at the bottom right of the editor is highlighted in green. Below the editor, there are tabs for Results, Explain, Describe, Saved SQL, and History. The status bar at the bottom displays the user's email (220701036@rajalakshmi.edu.in), the session ID (dm36), and the language (en). It also shows the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 25.2.4'. On the far right, it shows the date and time (27-02-2024 09:46) and system status indicators (ENG IN).

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

**RESULT:**

# Writing Basic SQL SELECT Statements

EX\_NO:4

DATE:

- 1.The following statement executes successfully.

## Identify the Errors

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

**QUERY:**

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

**QUERY:**

```
Desc Department;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The user has run the command `DESC DEPARTMENT;`. The results pane displays the structure of the `DEPARTMENT` table, which has two columns: `ID` (NUMBER type) and `NAME` (VARCHAR2 type). The `ID` column is the primary key and nullable. The `NAME` column is nullable.

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPARTMENT	ID	NUMBER	-	7	0	-	✓	-	-
	NAME	VARCHAR2	25	-	-	-	✓	-	-

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

**QUERY:**

**Select Number,Last\_name,job\_id,Hire\_Date,Emp\_ID from Employee;**

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. In the top-left corner, there's a browser window with the URL `apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=11836161183619`. The main area is titled "SQL Commands". The query entered is:

```
1 SELECT EMPLOYEE_ID, LAST_NAME, JOB_ID, HIRE_DATE
2 FROM EMPLOYEE;
```

Below the query, the "Results" tab is selected, displaying the following data:

EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE
40	Balaji	CEO	04/01/2023
23	Smith	MANAGER	03/01/2023
22	Smith	SALES	03/01/2024

At the bottom of the interface, there are various system status icons and a timestamp: "23:49 01-03-2024".

4.Provide an alias STARTDATE for the hire date.

**QUERY:**

**Select hire\_date as Start\_Date from employee;**

## OUTPUT:

The screenshot shows a browser window for apex.oracle.com with the URL apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=13448931157032. The page is titled "SQL Commands". The navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The top right corner shows a user profile for "balaji36 g" and a schema dropdown set to "WKSP\_D36". The main area contains a SQL editor with the following code:

```
1 : SELECT HIRE_DATE AS STARTDATE
2 : FROM EMPLOYEE;
```

Below the editor, a results grid displays the output:

STARTDATE
04/01/2023
03/01/2023
03/01/2024

The bottom of the screen shows a Windows taskbar with various application icons and system status indicators.

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

**QUERY:**

Select Unique JOB CODES from employee;

## OUTPUT:

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

```
1 SELECT UNIQUE JOB_ID
2 FROM EMPLOYEE;
```

The results section displays the output:

JOB_ID
CEO
MANAGER
SALES

Below the results, the status bar shows "2 rows returned in 0.00 seconds". The system information bar at the bottom right indicates "Oracle APEX 23.2.4".

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

### QUERY:

Select Last\_name || ',' || JOB\_ID as Title from employee;

### OUTPUT:

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

```
1 SELECT last_name || ',' || job_id AS title
2 FROM employee;
```

The results section displays the output:

TITLE
Balaji, CEO
Bsmith, MANAGER
Smith, SALES

Below the results, the status bar shows "2 rows returned in 0.00 seconds". The system information bar at the bottom right indicates "Oracle APEX 23.2.4".

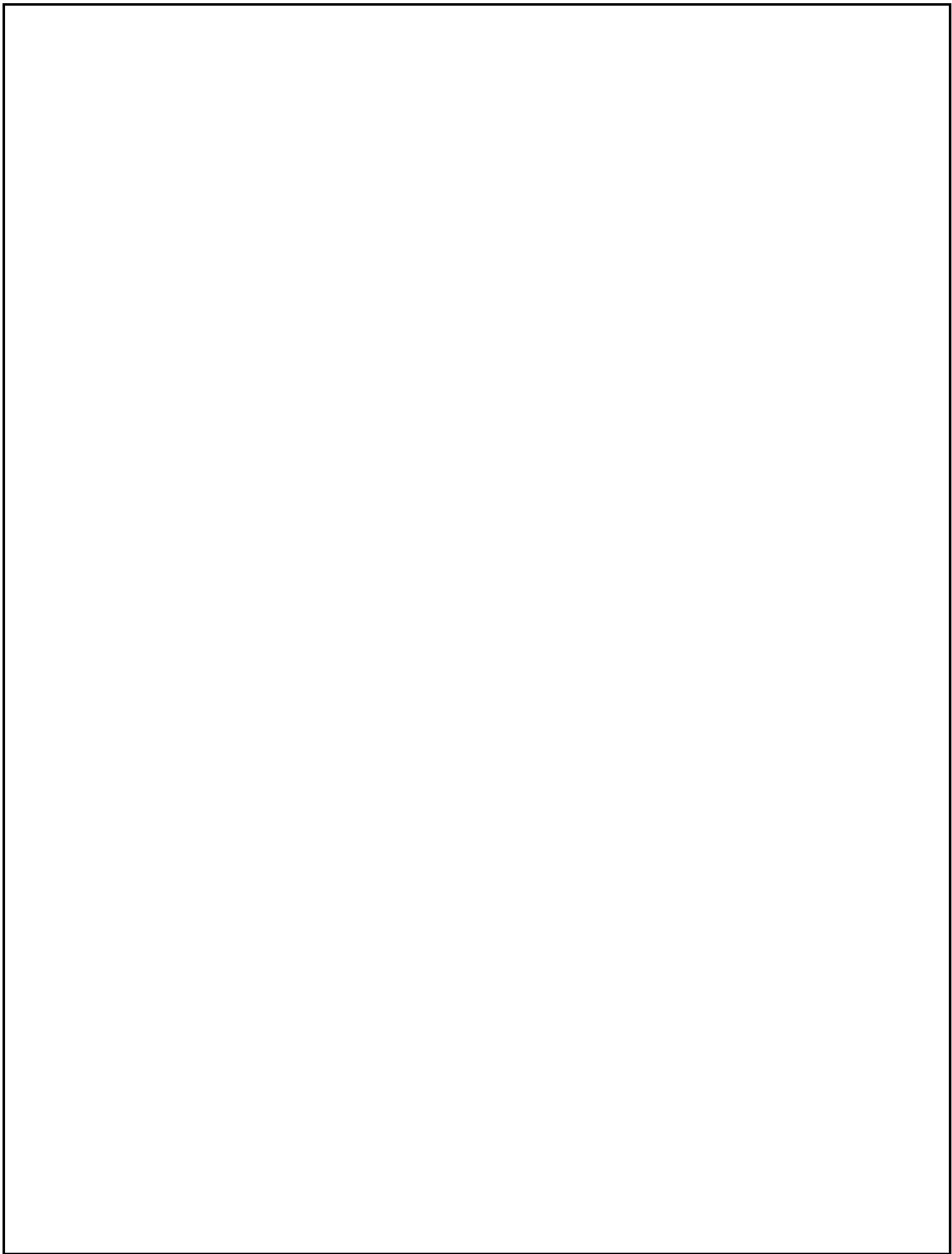
7.Create a query to display all the data from the employees table. Separate each column by a comma. Name the column THE\_OUTPUT.

**QUERY:**

**OUTPUT:**

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

**RESULT:**



# RESTRICTING AND SORTING DATA

EX\_NO:5

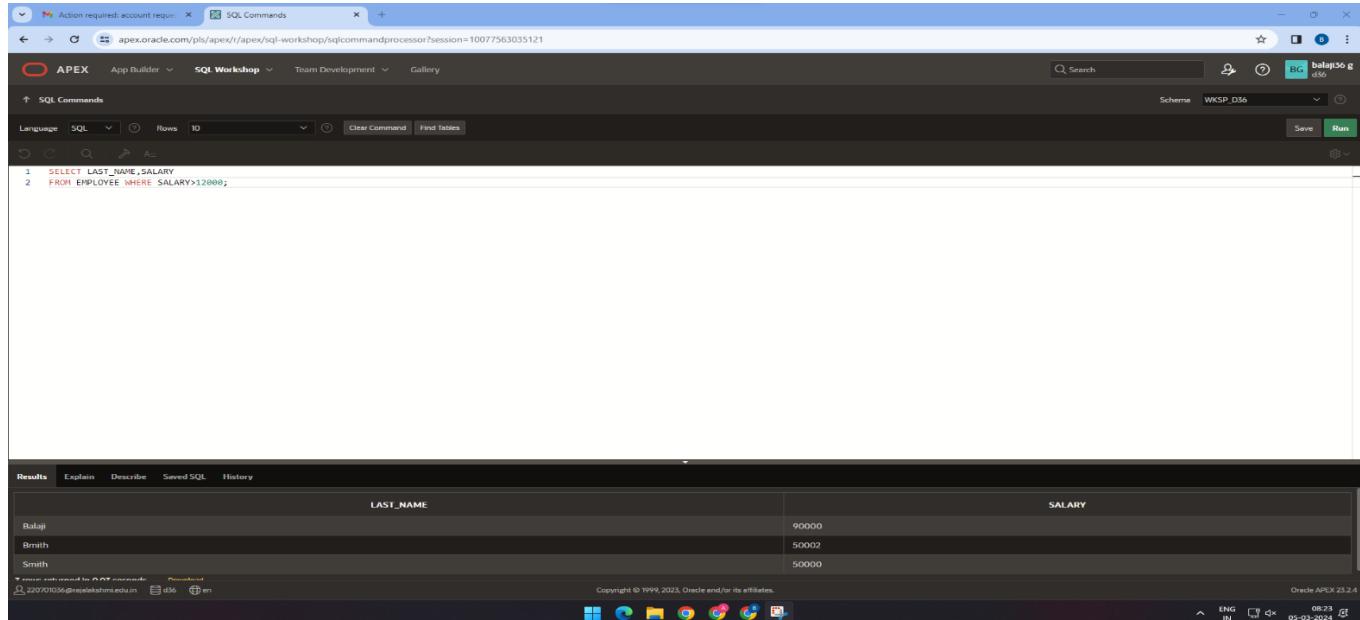
DATE:

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

**QUERY:**

**SELECT LAST\_NAME, SALARY FROM EMPLOYEE WHERE SALARY > 12000;**

**OUTPUT:**



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

```
1 SELECT LAST_NAME, SALARY
2 FROM EMPLOYEE WHERE SALARY > 12000;
```

The results are displayed in a table:

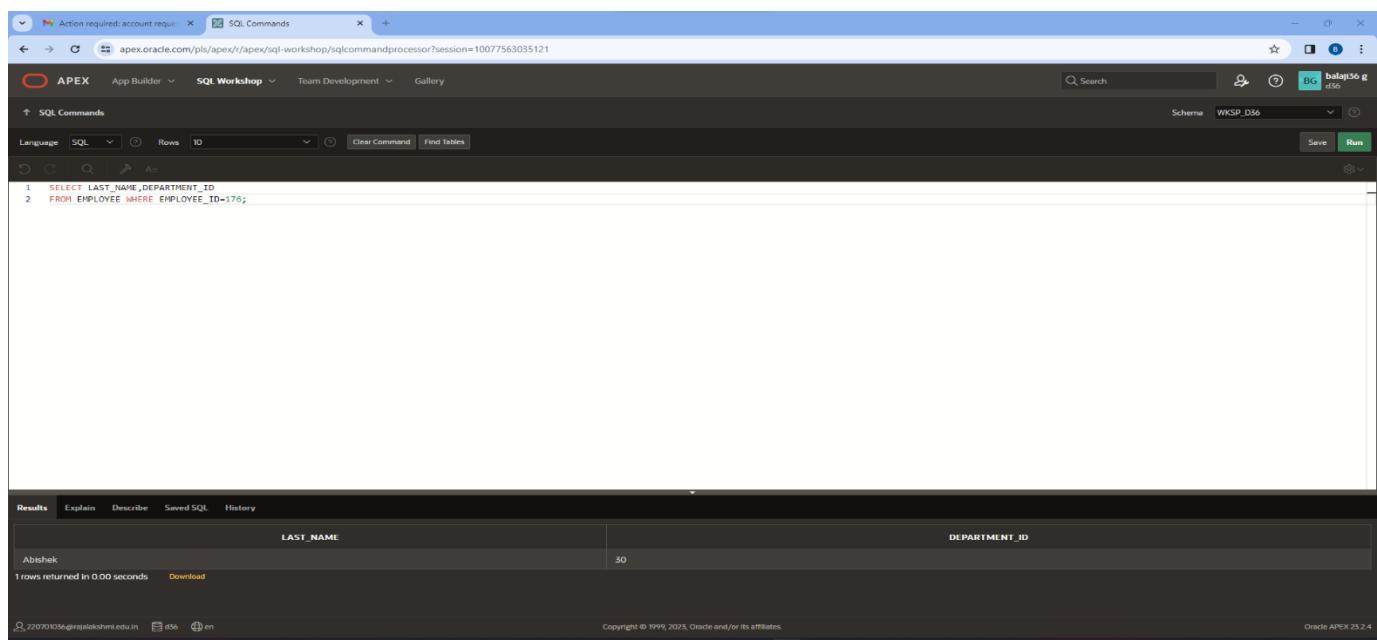
LAST_NAME	SALARY
Balaji	90000
Brith	50002
Smith	50000

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

**QUERY:**

**SELECT LAST\_NAME, DEPARTMENT\_ID FROM EMPLOYEE WHERE ID=176;**

**OUTPUT:**



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

```
1 SELECT LAST_NAME, DEPARTMENT_ID
2 FROM EMPLOYEE WHERE EMPLOYEE_ID=176;
```

The results are displayed in a table:

LAST_NAME	DEPARTMENT_ID
Abshek	30

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

#### QUERY:

**SELECT LAST\_NAME,SALARY FROM EMPLOYEE WHERE SALARY NOT BETWEEN 5000 AND 12000;**

#### OUTPUT:

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

```
1 SELECT LAST_NAME,SALARY
2 FROM EMPLOYEE WHERE SALARY NOT BETWEEN 5000 AND 12000;
```

The results window displays the following data:

LAST_NAME	SALARY
Balaji	90000
Abishek	340000
Bimith	5002

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

#### QUERY:

**SELECT LAST\_NAME,JOB\_ID,HIRE\_DATE FROM EMPLOYEE WHERE HIRE\_DATE BETWEEN '02/21/1998' AND '05/1/1998' ORDER BY HIRE\_DATE;**

#### OUTPUT:

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

```
1 SELECT LAST_NAME,JOB_ID,HIRE_DATE
2 FROM EMPLOYEE
3 WHERE HIRE_DATE BETWEEN '02/21/1998' AND '05/1/1998'
4 ORDER BY HIRE_DATE;
```

The results window displays the following data:

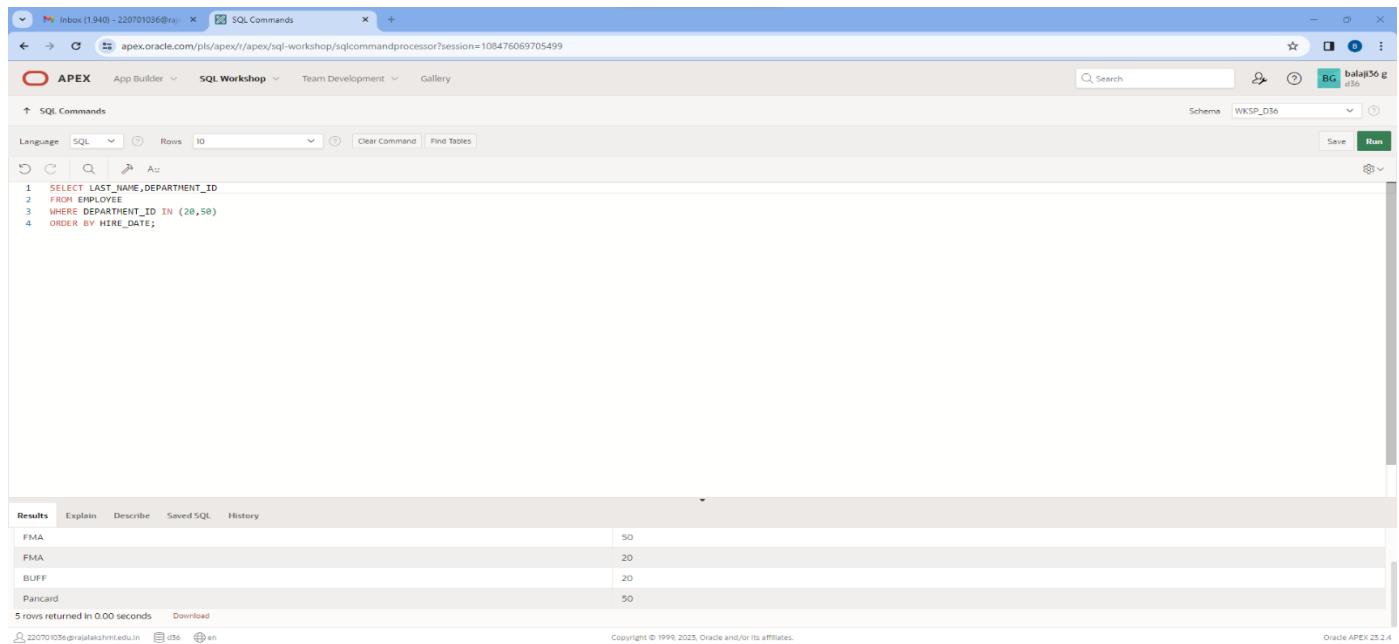
LAST_NAME	JOB_ID	HIRE_DATE
catcher	EMPLOYEE	02/21/1998
Tatcher	EMPLOYEE	02/21/1998
atcher	EMPLOYEE	02/21/1998

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

**QUERY:**

```
SELECT LAST_NAME,DEPARTMENT_ID FROM EMPLOYEE WHERE DEPARTMENT_ID IN(20,50) ORDER BY NAME;
```

**OUTPUT:**



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

```
1 SELECT LAST_NAME,DEPARTMENT_ID
2   FROM EMPLOYEE
3  WHERE DEPARTMENT_ID IN (20,50)
4 ORDER BY HIRE_DATE;
```

The results table displays the following data:

LAST_NAME	DEPARTMENT_ID
FMA	50
FMA	20
BUFF	20
Pancard	50

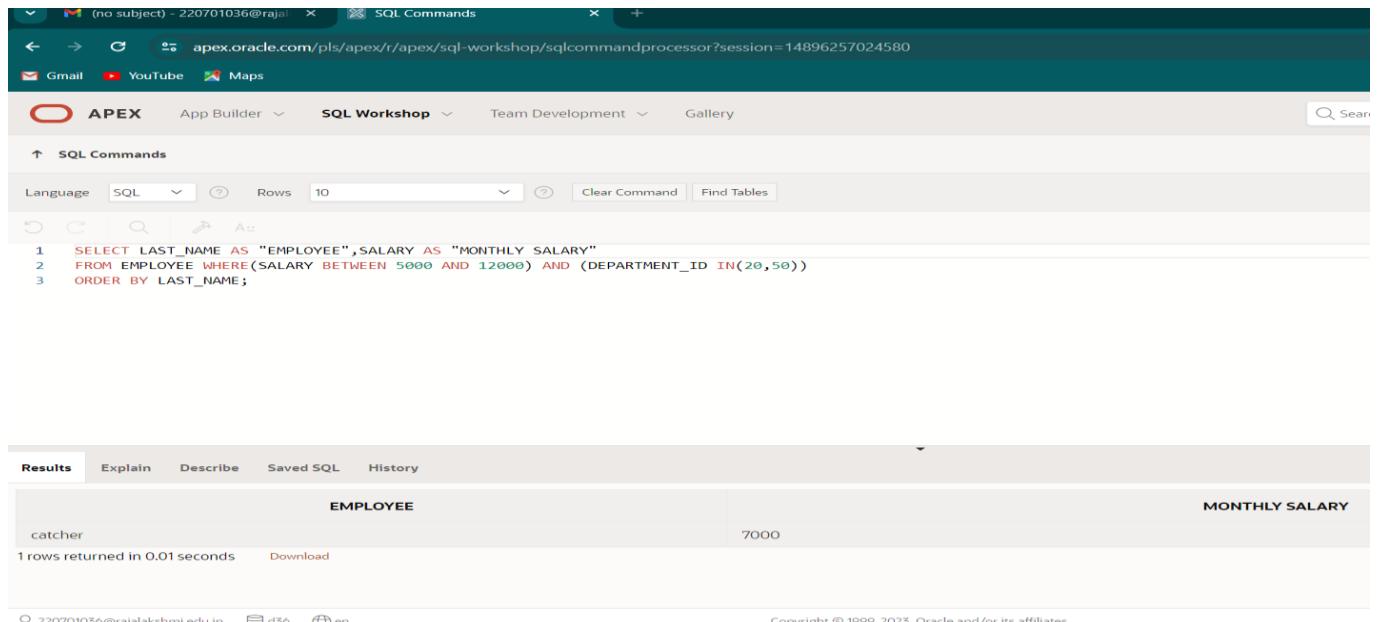
5 rows returned in 0.00 seconds

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

**QUERY:**

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

**OUTPUT:**



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

```
1 SELECT LAST_NAME AS "EMPLOYEE",SALARY AS "MONTHLY SALARY"
2   FROM EMPLOYEE WHERE(SALARY BETWEEN 5000 AND 12000) AND (DEPARTMENT_ID IN(20,50))
3 ORDER BY LAST_NAME;
```

The results table displays the following data:

EMPLOYEE	MONTHLY SALARY
catcher	7000

1 rows returned in 0.01 seconds

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

**QUERY:**

**SELECT LAST\_NAME,HIRE\_DATE FROM EMPLOYEE WHERE HIRE\_DATE='02/03/1994';**

**OUTPUT:**

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

```
1 SELECT LAST_NAME,HIRE_DATE
2 FROM EMPLOYEE
3 WHERE HIRE_DATE='02/03/1994';
```

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

LAST_NAME	HIRE_DATE
Kavi	02/03/1994
FMA	02/03/1994

2 rows returned in 0.00 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\_ID FROM EMPLOYEE WHERE MANAGER\_ID IS NULL;**

**OUTPUT:**

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

```
1 SELECT LAST_NAME,JOB_ID
2 FROM EMPLOYEE
3 WHERE MANAGER_ID IS NULL;
```

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

LAST_NAME	JOB
Abi	DEPUTY CEO
Abi	DEPUTY CEO

2 rows returned in 0.02 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,orderby)

#### QUERY:

**SELECT LAST\_NAME,SALARY,COMMISSION\_PCT FROM EMPLOYEE ORDER BY SALARY,COMMISSION\_PCT DESC;**

#### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for Gmail, YouTube, Maps, APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Commands tab is selected. The SQL editor contains the following code:

```
1 SELECT LAST_NAME, SALARY, COMMISSION_PCT
2 FROM EMPLOYEE
3 ORDER BY SALARY, COMMISSION_PCT DESC;
```

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

LAST_NAME	SALARY	COMMISSION_PCT
catcher	7000	.6
atcer	10000	.2
BUFF	23000	.9

At the bottom left, the session ID is shown as 220701036@rajalakshmi.edu.in, and the page number is d36. At the bottom right, the copyright notice reads "Copyright © 1999, 2023, Oracle and/or its affiliates."

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

#### QUERY:

**SELECT LAST\_NAME FROM EMPLOYEE WHERE LAST\_NAME LIKE '\_%a';**

#### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for Gmail, YouTube, Maps, APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Commands tab is selected. The SQL editor contains the following code:

```
1 SELECT LAST_NAME, SALARY, COMMISSION_PCT
2 FROM EMPLOYEE
3 ORDER BY SALARY, COMMISSION_PCT DESC;
```

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

LAST_NAME	SALARY	COMMISSION_PCT
catcher	7000	.6
atcer	10000	.2
BUFF	23000	.9

At the bottom left, the session ID is shown as 220701036@rajalakshmi.edu.in, and the page number is d36. At the bottom right, the copyright notice reads "Copyright © 1999, 2023, Oracle and/or its affiliates."

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

#### QUERY:

```
SELECT LAST_NAME FROM EMPLOYEE WHERE LAST_NAME LIKE %a% AND  
(LAST_NAME LIKE %e%);
```

#### OUTPUT:

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

```
1 SELECT LAST_NAME, SALARY, COMMISSION_PCT  
2 FROM EMPLOYEE  
3 ORDER BY SALARY, COMMISSION_PCT DESC;
```

The results section displays the following data:

LAST_NAME	SALARY	COMMISSION_PCT
catcher	7000	.6
atcer	10000	.2
BUFF	23000	.9

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

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

#### QUERY:

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

#### OUTPUT:

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

```
1 SELECT LAST_NAME, JOB_ID, SALARY  
2 FROM EMPLOYEE  
3 WHERE JOB_ID IN ('SALES', 'STOCK CLERK')  
4 AND SALARY NOT IN (2500, 3500, 7000);
```

The results section displays the following data:

LAST_NAME	JOB_ID	SALARY
Pancard	SALES	50000
Smith	SALES	50000

2 rows returned in 0.00 seconds    Download

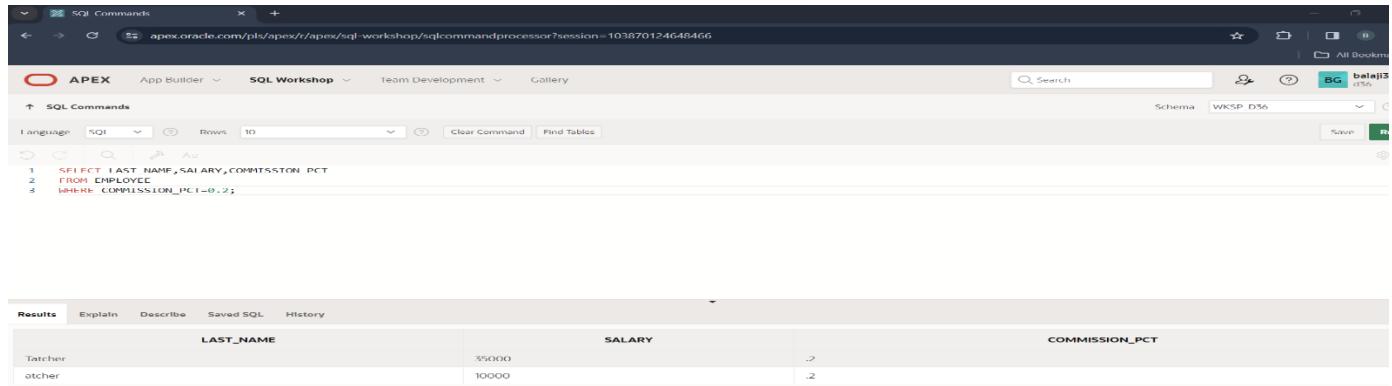
Copyright © 1999, 2023, Oracle and/or its affiliates.

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

#### QUERY:

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

#### OUTPUT:



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

```
1 SELECT LAST_NAME, SALARY, COMMISSION_PCT  
2 FROM EMPLOYEE  
3 WHERE COMMISSION_PCT=0.2;
```

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

LAST_NAME	SALARY	COMMISSION_PCT
Taticher	35000	.2
Atcher	10000	

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

#### RESULT:

# SINGLE ROW FUNCTIONS

EX\_NO:6

DATE:

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

QUERY:

```
select sysdate from dual;
```

OUTPUT:

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

```
1 SELECT SYSDATE AS "COLUMN DATE"
2 FROM DUAL;
```

The results tab shows the output:

COLUMN DATE
03/12/2024

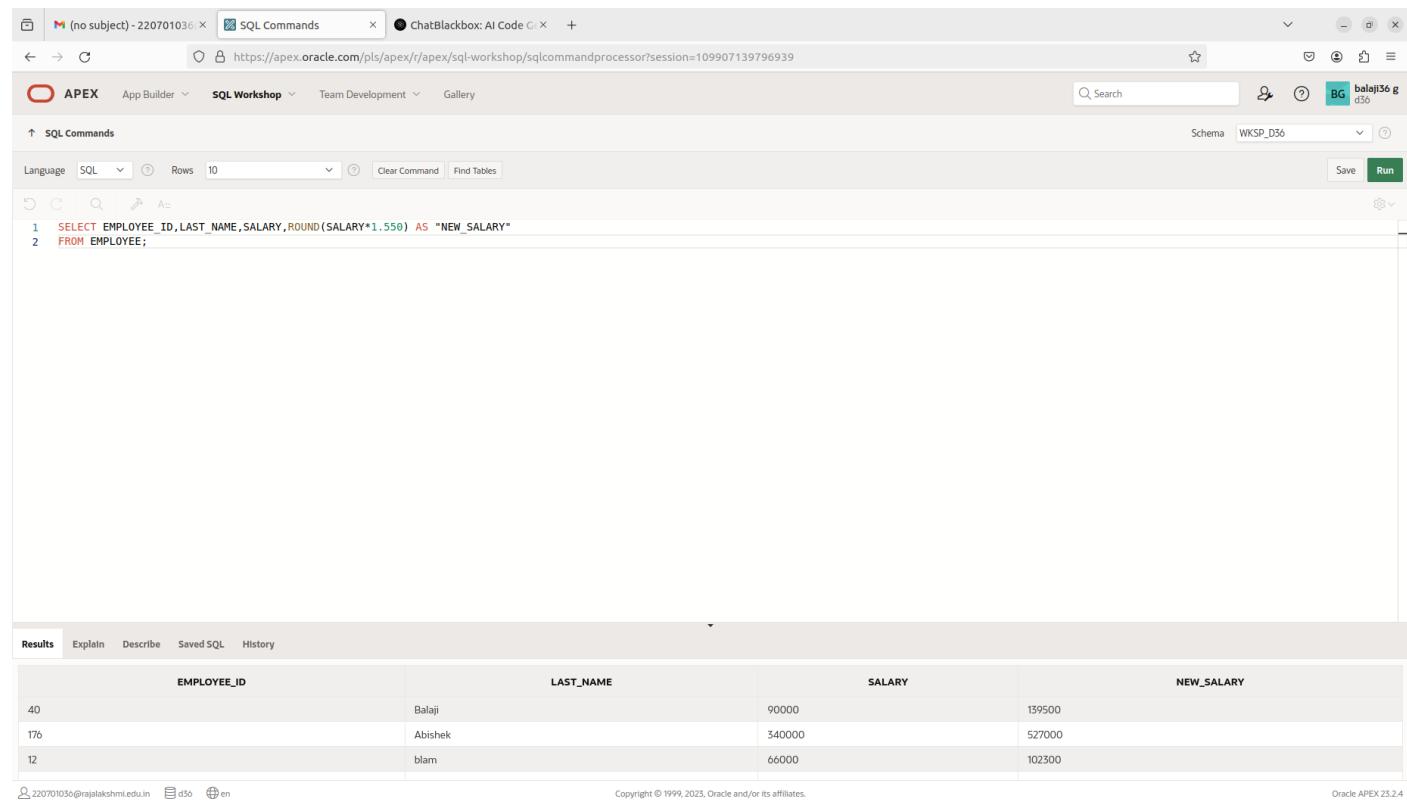
Below the results, it says "1 rows returned in 0.02 seconds".

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

QUERY:

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

## OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes tabs for App Builder, SQL Workshop (selected), Team Development, and Gallery. The main area is titled 'SQL Commands' with a sub-tab 'SQL'. The schema is set to 'WKSP\_D56'. The code editor contains the following SQL query:

```
1 SELECT EMPLOYEE_ID, LAST_NAME, SALARY, ROUND(SALARY*1.1550) AS "NEW_SALARY"
2 FROM EMPLOYEE;
```

Below the code editor is a results grid. The 'Results' tab is selected. The grid has four columns: EMPLOYEE\_ID, LAST\_NAME, SALARY, and NEW\_SALARY. The data is as follows:

EMPLOYEE_ID	LAST_NAME	SALARY	NEW_SALARY
40	Balaji	90000	103500
176	Abishek	340000	389400
12	blam	66000	75910

At the bottom of the page, there are footer links for 'Copyright © 1999, 2025, Oracle and/or its affiliates.', 'Oracle APEX 25.2.4', and session information '220701036@rajalakshmi.edu.in'.

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

## QUERY:

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

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes tabs for 'APEX' (selected), 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The main area is titled 'SQL Commands' with a sub-tab 'Find Tables'. The SQL editor contains the following query:

```
SELECT EMPLOYEE_ID, LAST_NAME, SALARY, ROUND(SALARY*0.155,0) AS "NEW_SALARY", SALARY - ROUND(SALARY*0.155,0) AS "INCREASE"
FROM EMPLOYEE;
```

The results section displays a table with the following data:

EMPLOYEE_ID	LAST_NAME	SALARY	NEW_SALARY	INCREASE
40	Balaji	90000	13950	76050
176	Abhishek	340000	52700	287300
12	blam	66000	10230	55770
12	FMA	66000	10230	55770
22	Pancard	50000	7750	42250
4	catcher	7000	1085	5915
2	Tatcher	35000	5425	29575
23	Bsmith	50002	7750	42252
12	Kavi	66000	10230	55770
2	Tatcher	35000	5425	29575

Below the table, a message indicates 'More than 10 rows available. Increase rows selector to view more rows.' and '10 rows returned in 0.01 seconds'. There is also a 'Download' link.

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

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. At the top, there are tabs for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user profile, and a schema dropdown set to 'WKSP\_D36'. Below the tabs, the SQL Commands section has a language dropdown set to SQL, row limit set to 10, and buttons for Clear Command and Find Tables. The main area contains a SQL command:

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

Below the command is a results grid:

EMPLOYEE_ID	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID	CITY
40	Balaji	Balaji.smith@company.com	3234567890	04/01/2023	CEO	90000	.8	120	200	-

At the bottom left are session details: 220701036@rajalakshmi.edu.in, d36, en. In the center is a copyright notice: Copyright © 1999, 2023, Oracle and/or its affiliates. On the right is the text: Oracle APEX 23.2.4.

**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:**

**OUTPUT:**

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. At the top, there are various links and a search bar. Below that is the main toolbar with tabs like 'APEX', 'App Builder', 'SQL Workshop' (which is selected), 'Team Development', and 'Gallery'. A search bar and a user profile are also present.

In the central workspace, under the 'SQL Commands' tab, a SQL query is entered:

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

The results are displayed in a table:

EMPLOYEE_ID	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID	CITY
40	Balaji	Balaji.smith@company.com	3234567890	04/01/2023	CEO	90000	.8	120	200	-

At the bottom, there are footer links and the text 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

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

select last\_name||' earns'||salary||' monthly but wants'||salary\*3 as "DREAM\_SALARIES" from employees;

**OUTPUT:**

This screenshot is identical to the one above, showing the Oracle APEX SQL Workshop interface. The query entered is:

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

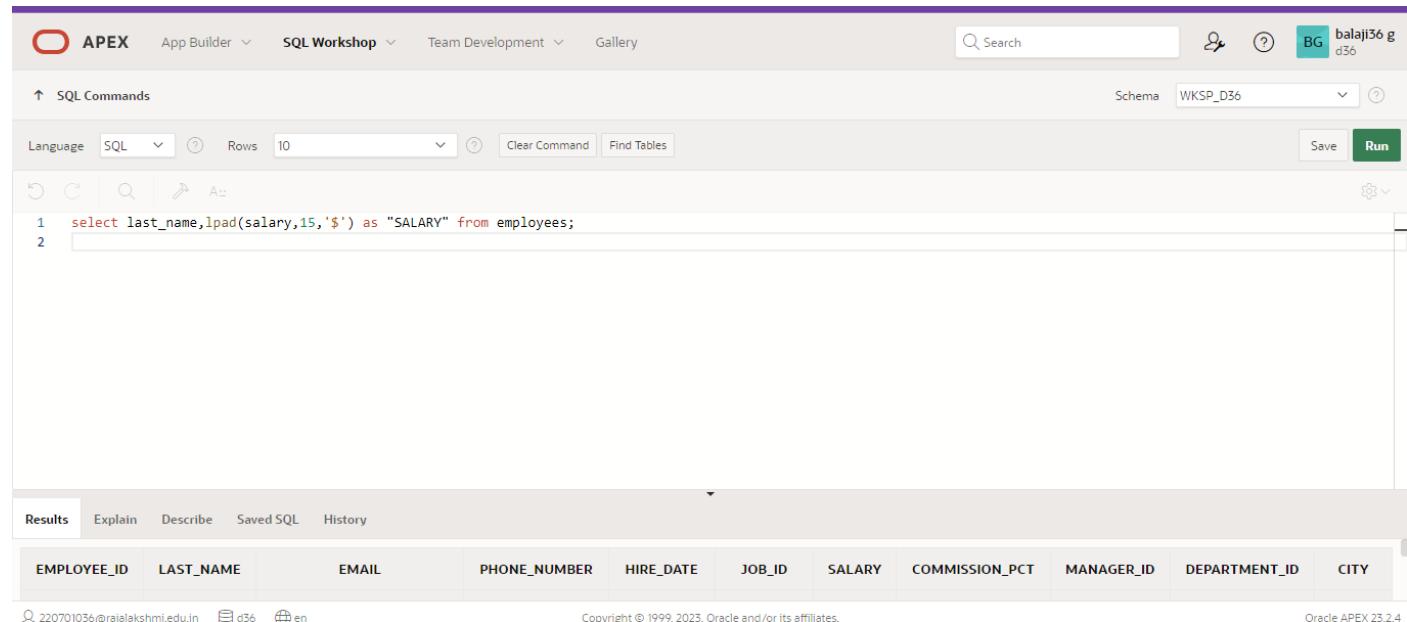
The results table is empty, indicating no data was returned for this specific query.

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

**QUERY:**

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

**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, user profile, and session information (Schema: WKSP\_D36, User: balaji36 g d36). The main workspace is titled 'SQL Commands' and contains the following SQL code:

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

Below the code, the results tab is selected, showing the output of the query. The columns listed are EMPLOYEE\_ID, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, DEPARTMENT\_ID, and CITY. The results table is currently empty, indicating no data has been returned yet.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. At the top, there are tabs for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'balaji36 g d36', and a schema dropdown set to 'WKSP\_D36'. Below the header, the SQL Commands section has a language dropdown set to SQL, a rows dropdown set to 10, and buttons for Clear Command and Find Tables. The main area contains the following SQL code:

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

Below the code, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, showing a table with columns: EMPLOYEE\_ID, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, DEPARTMENT\_ID, and CITY. The table contains several rows of employee data.

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

#### QUERY:

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

#### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface, identical to the previous one but with a different query. The SQL Commands section contains the following code:

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

The Results tab is selected, showing the same table structure and data as the previous screenshot, displaying employee information including their hire date converted to the day of the week.

**RESULT:**

# DISPLAYING DATA FROM MULTIPLE TABLES

EX\_NO:7

DATE:

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

QUERY:

```
select e.name,e.dept_id,d.dept_name from employees e,departments d  
where e.dept_id=d.dept_id;
```

OUTPUT:

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

```
1 SELECT e.last_name, e.dept_id, d.dept_name  
2 FROM employees e, departments d  
3 WHERE e.dept_id = d.dept_id;  
4
```

The results pane displays the output:

LAST_NAME	DEPT_ID	DEPT_NAME
banner	22	csd

1 rows returned in 0.00 seconds

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

QUERY:

```
Select distinct e.employee_id,d.dept_no,d.dept_name from employee e ,department1 d where  
e.employee_id==d.dept_id and d.dept_id=80;
```

OUTPUT:

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

```
1 select distinct e.employee_id,d.dept_no,location_id  
2 from employee e,department1 d  
3 where e.employee_id = d.dept_no  
4 and d.dept_no=80;  
5
```

The results pane displays the output:

EMPLOYEE_ID	DEPT_NO	LOCATION_ID
101	80	40

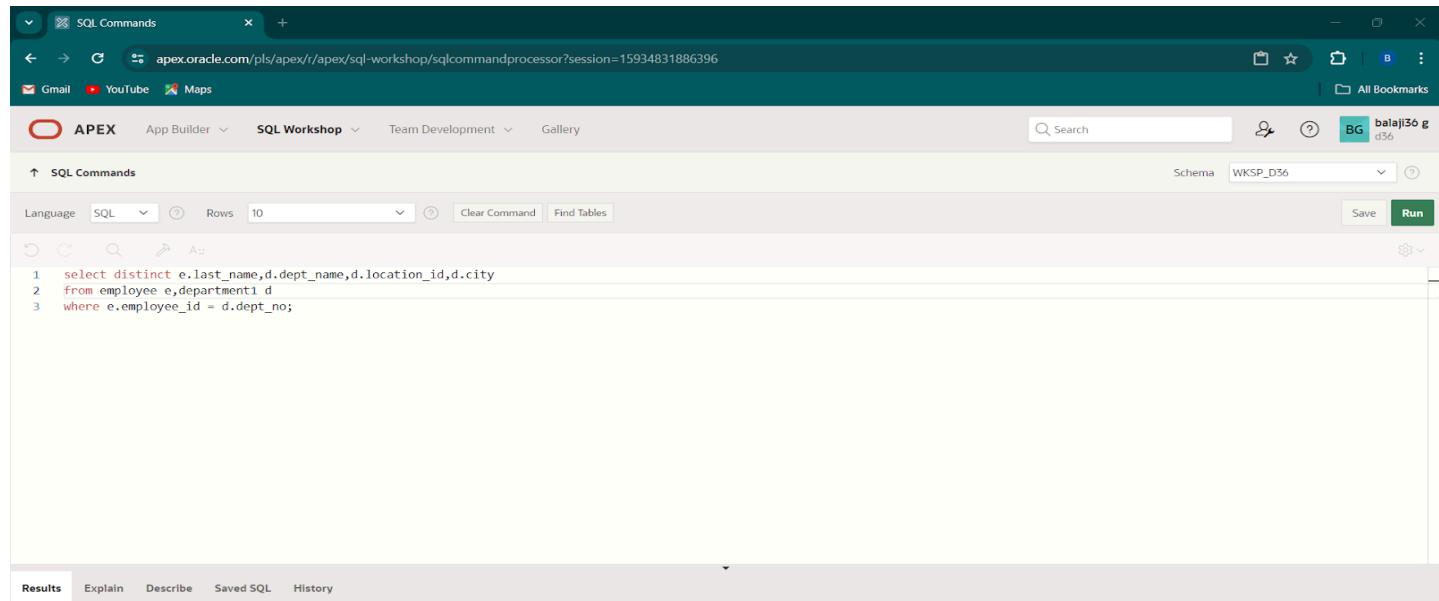
no data found

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 from employee e ,department1 d where e.employee_id==d.dept_no;
```

**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. The main area is titled "SQL Commands". The schema is set to "WKSP\_D36". The code editor contains the following SQL query:

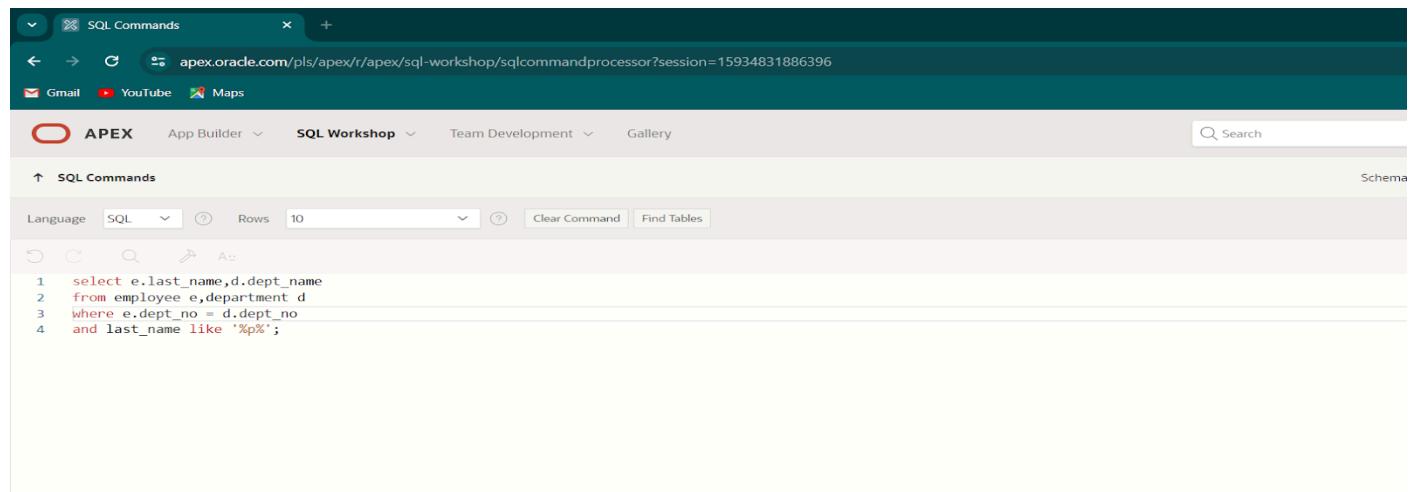
```
1 select distinct e.last_name,d.dept_name,d.location_id,d.city
2 from employee e,department1 d
3 where e.employee_id = d.dept_no;
```

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_namefrom employees,departments where employees.dept_id=departments.dept_id  
And last_name like '%a%';
```

**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. The main area is titled "SQL Commands". The schema is set to "WKSP\_D36". The code editor contains the following SQL query:

```
1 select e.last_name,d.dept_name
2 from employee e,department d
3 where e.dept_no = d.dept_no
4 and last_name like '%a%';
```

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

**QUERY:**

```
Select e.last_name,e.job_id,e.dept_id,d.dept_name from employees e join departments d on (e.dept_id=d.dept_id)
Join locations l on(d.location_id=l.location_id) where lower(l.city)='toronto';
```

**OUTPUT:**

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

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

The results tab displays the following table:

LAST_NAME	JOB_ID	DEPT_ID	DEPT_NAME
roger	80	80	cse
smith	55	80	cse

2 rows returned in 0.00 seconds

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

**QUERY:**

```
Select w.last_name "Employee",w.emp_id "EMP#",m.last_name "Manager",m.emp_id "Mgr#"
From employes w join employes m on (w.manager_id=m.emp_id);
```

**OUTPUT:**

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

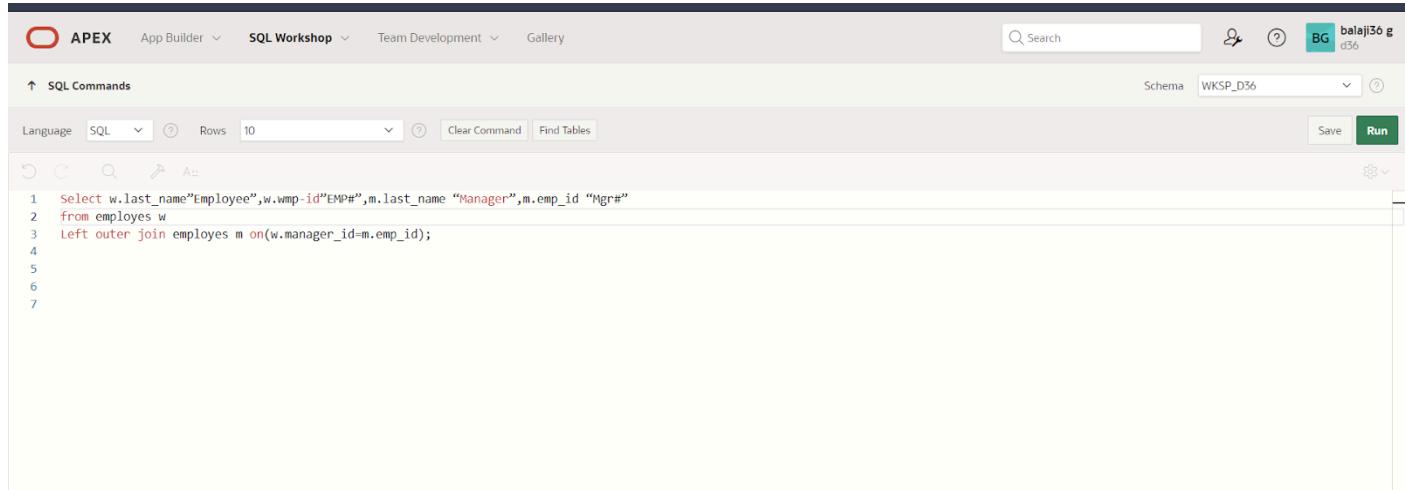
```
1 Select w.last_name "Employee",w.emp_id "EMP#",m.last_name "Manager",m.emp_id "Mgr#"
2 From employees w
3 join employees m on (w.manager_id=m.emp_id);
```

7. Modify lab4\_6.sql to display all employees including King, who has no manager. Order the results by the employee number.

**QUERY:**

```
Select w.last_name "Employee", w.emp_id "EMP#", m.last_name "Manager", m.emp_id "Mgr#" from employees w  
Left outer join employees m on(w.manager_id=m.emp_id);
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon, and a session name 'balaaji36 g d56'. Below the navigation is a toolbar with icons for SQL, Rows, Clear Command, Find Tables, Save, and Run. The main area is titled 'SQL Commands' and contains the following SQL code:

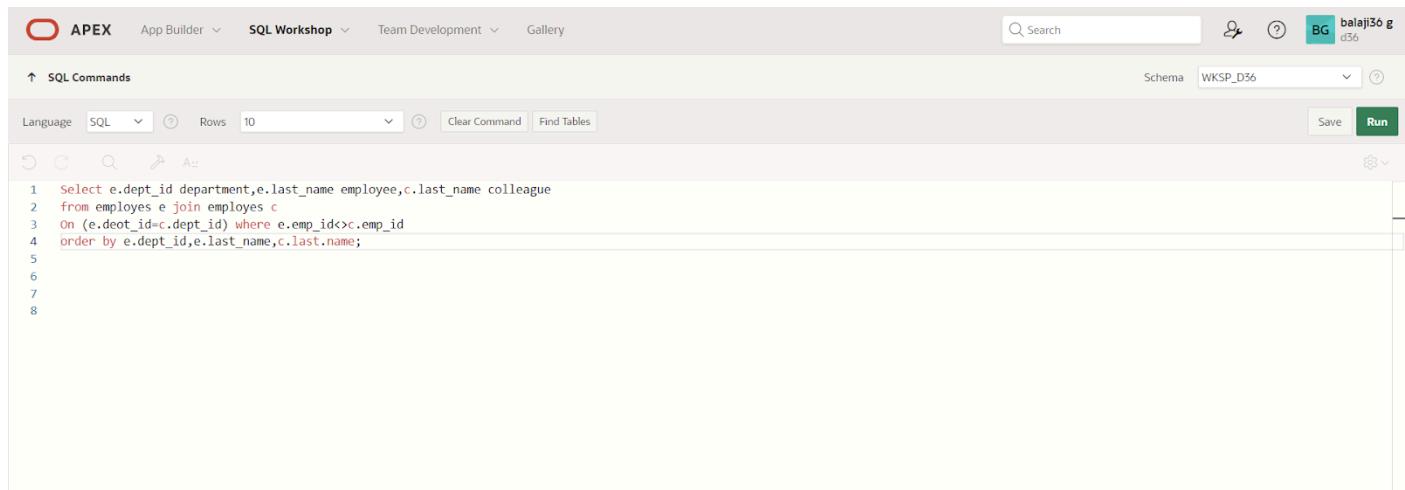
```
1 Select w.last_name "Employee", w.emp_id "EMP#", m.last_name "Manager", m.emp_id "Mgr"  
2 from employees w  
3 Left outer join employees m on(w.manager_id=m.emp_id);  
4  
5  
6  
7
```

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 employees e join employees 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 (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon, and a session name 'balaaji36 g d56'. Below the navigation is a toolbar with icons for SQL, Rows, Clear Command, Find Tables, Save, and Run. The main area is titled 'SQL Commands' and contains the following SQL code:

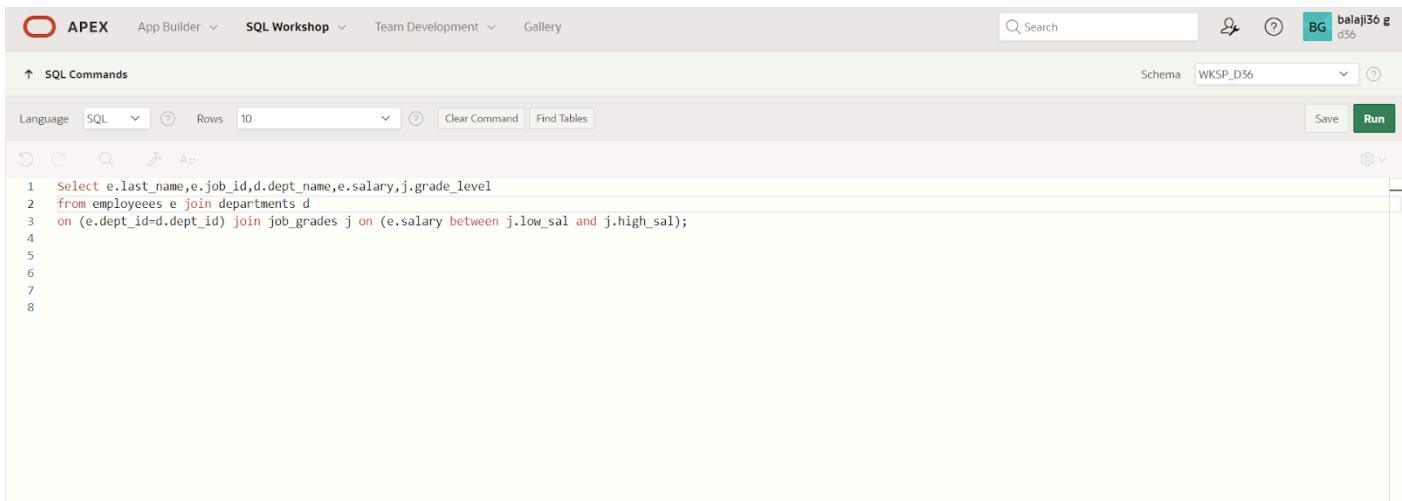
```
1 Select e.dept_id department, e.last_name employee, c.last_name colleague  
2 from employees e join employees c  
3 On (e.dept_id=c.dept_id) where e.emp_id<>c.emp_id  
4 order by e.dept_id, e.last_name, c.last.name;  
5  
6  
7  
8
```

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

**QUERY:**

```
Select e.last_name,e.job_id,d.dept_name,e.salary,j.grade_level from employees e join departments d on (e.dept_id=d.dept_id) join job_grades j on (e.salary between j.low_sal and j.high_sal);
```

**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 a user icon are also present. The main area is titled 'SQL Commands' and shows the following SQL code:

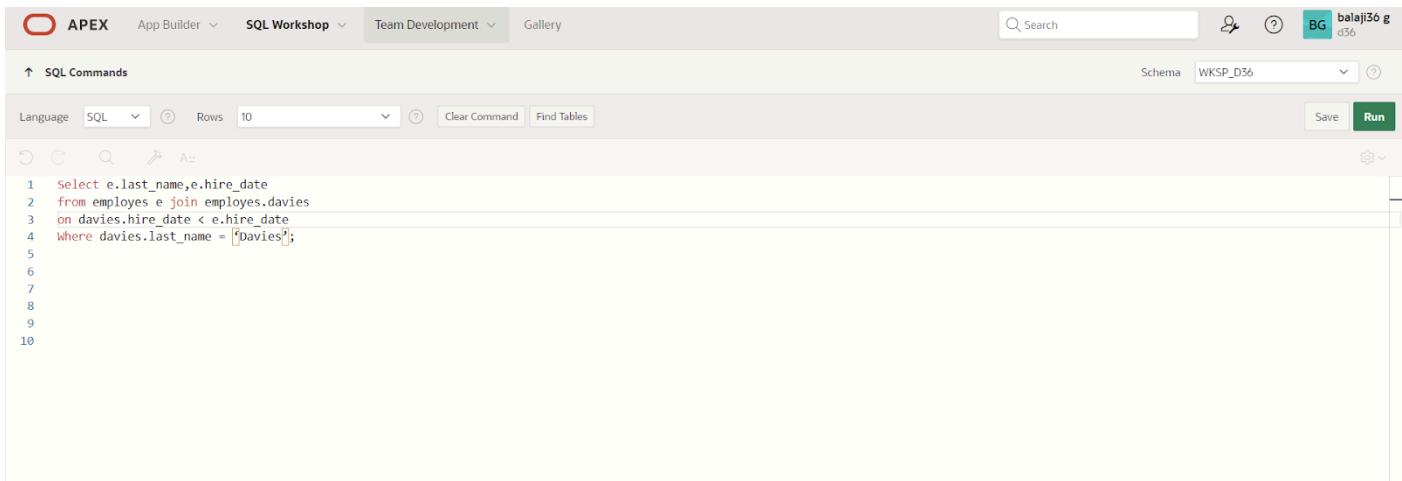
```
1 Select e.last_name,e.job_id,d.dept_name,e.salary,j.grade_level
2 from employees e join departments d
3 on (e.dept_id=d.dept_id) join job_grades j on (e.salary between j.low_sal and j.high_sal);
4
5
6
7
8
```

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 employees e join employees.davies on davies.hire_date < e.hire_date
Where davies.last_name = 'Davies';
```

**OUTPUT:**



A screenshot of the Oracle SQL Workshop interface, identical to the previous one but with a different query. The top navigation bar and main area are the same, but the SQL code is:

```
1 Select e.last_name,e.hire_date
2 from employees e join employees.davies
3 on davies.hire_date < e.hire_date
4 Where davies.last_name = 'Davies';
5
6
7
8
9
10
```

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

#### QUERY:

```
select e.last_name as Employee,e.hire_date as Emp_hired,e.man_name as manager,m.hire_date as mgr_hired from employees join employees m on e.man_name=m.last_name where e.hire_date<m.hire_date;
```

#### 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 'balaaji36 g'. The main area is titled 'SQL Commands' with tabs for Language (SQL selected), Rows (10), and Find Tables. Below these are standard icons for Undo, Redo, Search, and Paste. The SQL command window contains the following code:

```
1 select e.last_name as Employee,e.hire_date as Emp_hired,e.man_name as manager,m.hire_date as mgr_hired
2 from employees join employees m
3 on e.man_name=m.last_name
4 where e.hire_date<m.hire_date;
5
6
7
8
9
```

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

#### RESULT:

# AGGREGATING DATA USING GROUP FUNCTIONS

**EX\_NO:8**

**DATE:**

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

**TRUE**

2. Group functions include nulls in calculations.  
True/False

**FALSE**

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

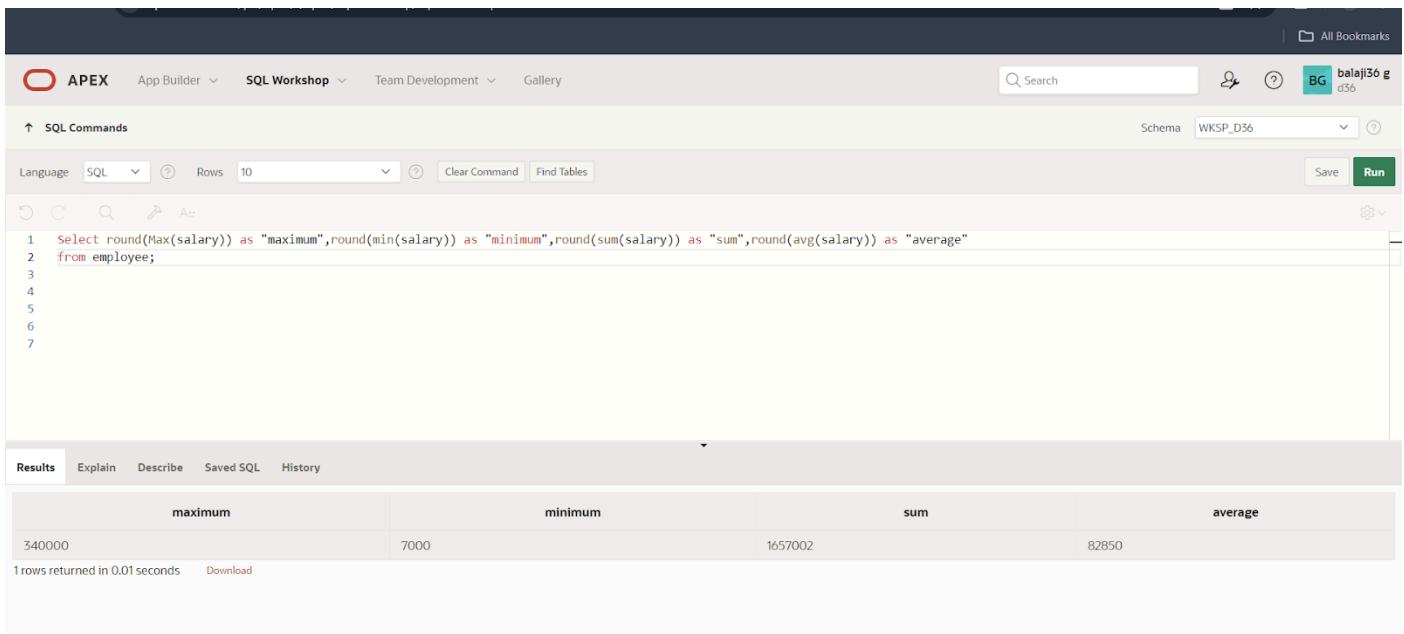
**FALSE**

4. Find the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole number

**QUERY:**

Select  
round(max(salary),0)"Maximum",round(min(salary),0)"Minimum",round(sum(salary),0)"sum",round(avg(salary),0)  
"average" from EMPA;

**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 are user profile icons and a 'Run' button. The main workspace is titled 'SQL Commands'. It contains a code editor with the following SQL query:

```
1 Select round(max(salary)) as "maximum",round(min(salary)) as "minimum",round(sum(salary)) as "sum",round(avg(salary)) as "average"
2 from employee;
3
4
5
6
7
```

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:

	maximum	minimum	sum	average
340000	7000	1657002	82850	

The status bar at the bottom indicates "1 rows returned in 0.01 seconds" and has a "Download" link.

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

### QUERY:

Select

```
job_id,round(max(salary),0)"MAXIMUM",round(Min(salary),0)"Minimun",round(sum(salary),0)"sum",round(Avg(salary),0)"average" from EMPA group by job_id;
```

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL Commands tab contains the following query:

```
1 select round(max(salary)) as "maximum",round(min(salary)) as "minimum",round(sum(salary)) as "sum",round(avg(salary)) as "average"
2 from employee
3 group by job_id;
4
5
6
7
8
```

The Results tab displays the output in a grid format:

	maximum	minimum	sum	average
90000	90000	90000	90000	90000
66000	66000	66000	66000	66000
234500	234500	234500	469000	234500
66000	50002	50002	248002	62001

At the bottom, the URL is 220701050@rajalakshmi.edu.in, the session ID is d56, and the environment is en. The copyright notice is Copyright © 1999, 2023, Oracle and/or its affiliates. The version is Oracle APEX 23.2.4.

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

### QUERY:

Select job\_id,count(\*) from EMPA where job\_id='47' group by job\_id;

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL Commands tab contains the following query:

```
1 select job_id,count(job_id)
2 from employee
3 group by job_id;
4
5
6
7
8
```

The Results tab displays the output in a grid format:

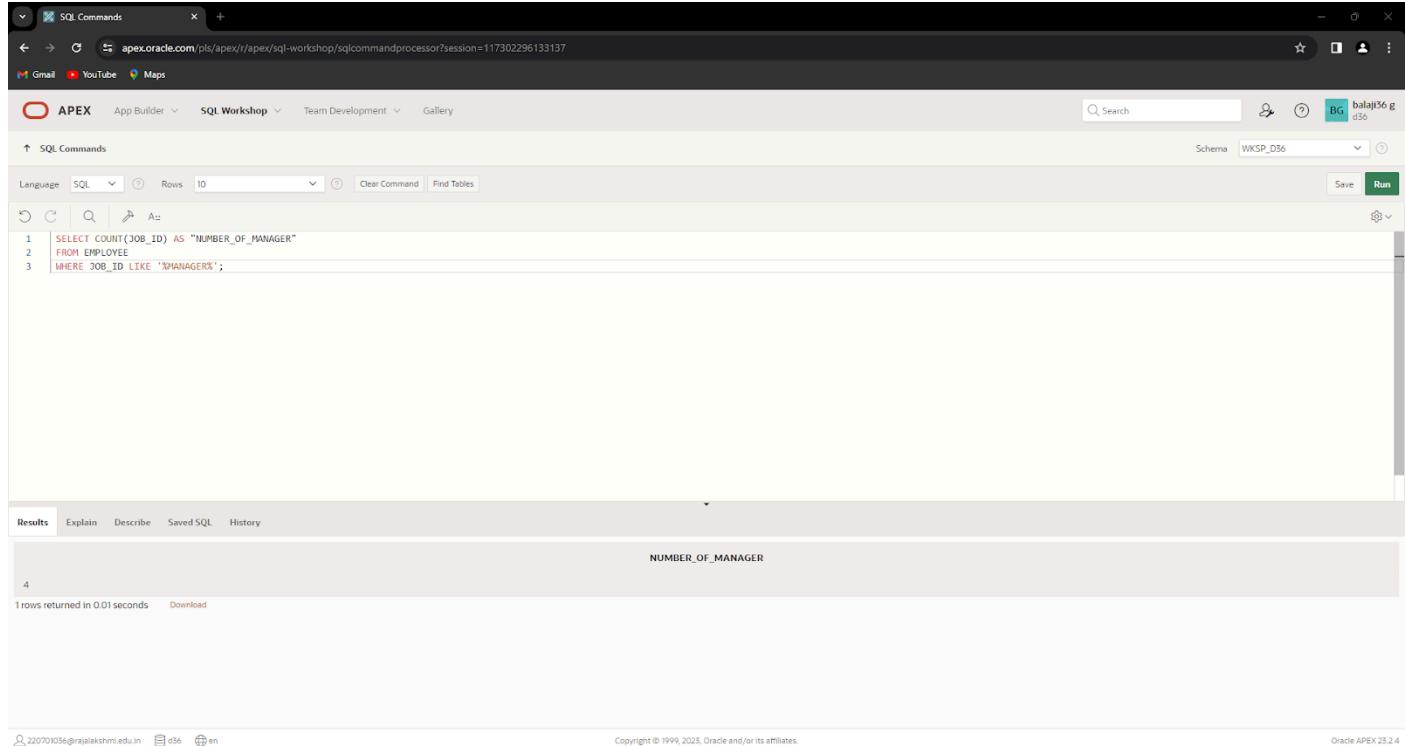
JOB_ID	COUNT(JOB_ID)
CEO	1
WORKER	1
DEPUTY CEO	2
MANAGER	4

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

### QUERY:

Select count(distinct manager\_id)"Number of managers" from empa;

### OUTPUT:



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

```
1 | SELECT COUNT(JOB_ID) AS "NUMBER_OF_MANAGER"
2 | FROM EMPLOYEE
3 | WHERE JOB_ID LIKE '%MANAGER%';
```

The results section displays the output:

NUMBER_OF_MANAGER
4

1 rows returned in 0.01 seconds

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

### QUERY:

Select max(salary)-min(salary) difference from empa;

### OUTPUT:

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

```

1 Select (max(salary)-min(salary)) as "difference"
2 from employee;
3
4
5
6
7

```

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

difference
333000

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

The status bar at the bottom shows the user's email (220701056@rajalakshmi.edu.in), session ID (d56), and language (en). The system also displays the date and time (Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4).

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

### QUERY:

Select manager\_id,min(salary) from empa where manager\_id is not null group by manager\_id having min(salary)>6000 order by min(salary) desc

### OUTPUT:

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

```

1 Select manager_id,min(salary)
2 from employee where manager_id is not null
3 group by manager_id
4 having min(salary)>6000 order by min(salary) desc
5
6
7
8
9
10

```

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

MANAGER_ID	MIN(SALARY)
145	340000
111	50002
45	50000
1001	50000

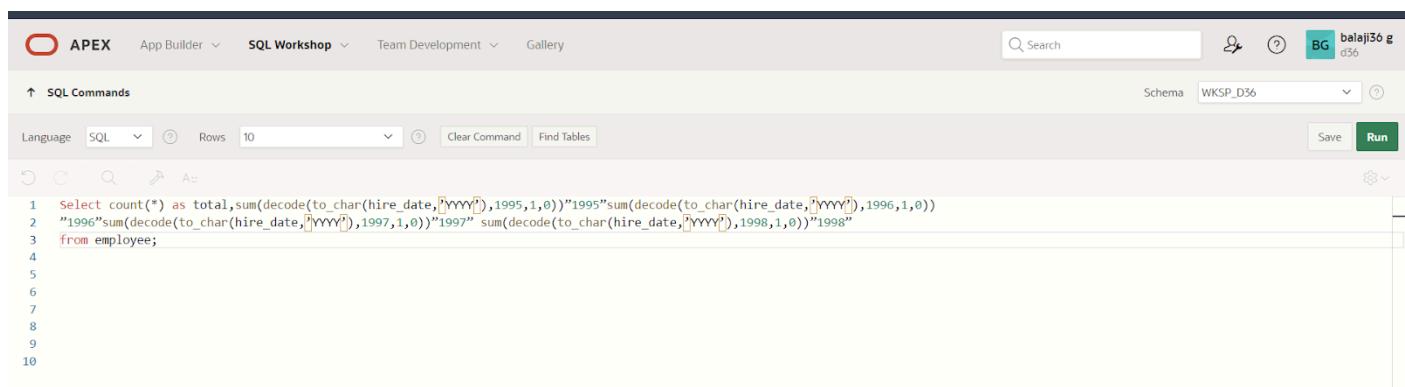
The status bar at the bottom shows the user's email (220701056@rajalakshmi.edu.in), session ID (d56), and language (en). The system also displays the date and time (Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4).

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

### QUERY:

Select count(\*) as  
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 empa;

**OUTPUT:**



A screenshot of the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon, and a schema dropdown set to WKSP\_D56. Below the navigation is a toolbar with icons for Undo, Redo, Search, and Find Tables. The main area shows the SQL command being run:

```
1 select count(*) as total,sum(decode(to_char(hire_date,'YYYY'),1995,1,0))"1995"sum(decode(to_char(hire_date,'YYYY'),1996,1,0))
2 "1996"sum(decode(to_char(hire_date,'YYYY'),1997,1,0))"1997" sum(decode(to_char(hire_date,'YYYY'),1998,1,0))"1998"
3 from employee;
4
5
6
7
8
9
10
```

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

**QUERY:**

**OUTPUT:**

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

**QUERY:**

Select d.dept\_name as "dept\_name",d.loc as "department location",count(\*) as "Number of people",round(avg(e.salary),2) as "salary" from departments d inner join emp a e on (d.dept\_id=e.dept\_id)  
Group by d.dept\_name,d.loc;

**OUTPUT:**



A screenshot of the Oracle SQL Workshop interface, identical to the previous one but with a different query. The top navigation bar and schema dropdown are the same. The main area shows the new SQL command:

```
1 Select d.dept_name as "dept_name",d.loc as "department location",count(*) as "Number of people",round(avg(e.salary),2) as "salary"
2 from departments d inner join emp a e on (d.dept_id=e.dept_id)
3 Group by d.dept_name,d.loc;
4
5
6
7
8
9
10
11
```

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

**RESULT:**



## SUB-QUERIES

EX.NO:9

DATE:

Find the Solution for the following:

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

**QUERY:**

```
Select lastname, hiredate  
from employees  
where deptid =  
(select deptid from employees where lastname like 'zlotkey')and lastname <> '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 search bar is empty. The schema is set to WKSP\_D56.

In the SQL Commands section, the code is displayed:

```
1 Select last_name, hire_date  
2   from employee  
3  where department_id =  
4 (select department_id from employee where last_name like 'zlotkey')and last_name <> 'zlotkey';|
```

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

LAST_NAME	HIRE_DATE
san	07/01/2020
sanam	07/01/2023

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

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

### QUERY:

```
select employeeid, lastname, salary from employees
```

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

### OUTPUT:

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

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

The results window displays the following data:

EMPLOYEE_ID	LAST_NAME	SALARY
10	zlotkey	230000
10	san	230000
12	sanam	230004
23	Abi	234500

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 employeeid, lastname
```

```
from employees
```

```
where deptid in (select deptid from employees where lastname like '%u%');
```

### OUTPUT:

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

```
1 select employee_id, last_name
2   from employee
3  where department_id in (select department_id from employee where last_name like '%u%');
```

The results window displays the following data:

EMPLOYEE_ID	LAST_NAME
10	san
10	zlotkey
10	saun
12	sanam

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 lastname, deptid, jobid  
from employees  
where deptid in (select deptid from dept where locationid =1700);
```

### 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 shows a user profile for 'balaji36 g' and a schema dropdown set to 'WKSP\_D36'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the following SQL code:

```
1 select last_name, department_id, job_id  
2 from employee  
3 where department_id in (select department_id from department where location_id =1700);
```

The Results tab displays the query results in a grid format:

LAST_NAME	DEPARTMENT_ID	JOB_ID
Balaji	200	CEO
Abishek	30	EMPLOYEE
blam	26	MANAGER
FMA	20	stock

At the bottom, the footer includes the URL '220701036@rajalakshmi.edu.in', session information 'd36 en', copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.', and the version 'Oracle APEX 23.2.4'.

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

### QUERY:

```
select lastname, salary  
from employees  
where managerid in (select employeeid from employees where lastname='King');
```

### 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 shows a user profile for 'balaji36 g' and a schema dropdown set to 'WKSP\_D36'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the following SQL code:

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

The Results tab displays the query results in a grid format:

LAST_NAME	SALARY
King	50000

The message 'no data found' is displayed below the results grid.

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 deptid, lastname, jobid  
from employees  
where deptid in (select deptid from dept where deptname = 'executive');
```

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile 'balaji36 g d56'. The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the query is pasted:

```
1 select department_id, last_name, job_id  
2   from employee  
3  where department_id in (select department_id from department1 where dept_name = 'executive');  
4
```

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

DEPARTMENT_ID	LAST_NAME	JOB_ID
200	Balaji	CEO
30	Abishek	EMPLOYEE
26	blam	MANAGER
20	FMA	stock

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 employeeid, lastname, salary  
from employees  
where salary > (select avg(salary) from employees) and  
deptid in (select deptid from employees where lastname like '%u%');
```

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface, identical to the previous one but with a different query. The top navigation bar and user profile are the same. The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the modified query is pasted:

```
1 select department_id, last_name, job_id  
2   from employee  
3  where department_id in (select department_id from department1 where dept_name = 'executive');  
4
```

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

DEPARTMENT_ID	LAST_NAME	JOB_ID
200	Balaji	CEO
30	Abishek	EMPLOYEE
26	blam	MANAGER
20	FMA	stock

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

**RESULT:**



# USING THE SET OPERATORS

**EX\_NO:10**

**DATE:**

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

**QUERY:**

```
select department_id from employees minus select department_id from employees where job_id='st_clerk';
```

**OUTPUT:**

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

```
1 select department_id from
2 employee minus select department_id from employee| where job_id='st_clerk';
```

In the Results pane, the output is displayed as:

DEPARTMENT_ID
9
11
20
23

The system status bar at the bottom indicates the user is connected to 220701056@rajalakshmi.edu.in, the session ID is d36, the language is en, and the date and time are 15-04-2024 09:27.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX 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, and a session identifier 'balaji36 g d56'. Below the navigation is a toolbar with icons for SQL, Rows, Clear Command, Find Tables, Save, and Run. The main area contains the SQL command and its execution results.

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

**Results**

DEPARTMENT_ID
9
11
20
23

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

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

**QUERY:**

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

**OUTPUT:**

APEX App Builder SQL Workshop Team Development Gallery

Search Schema WKSP\_D36

SQL Commands

Language SQL Rows 10 Clear Command Find Tables Save Run

undo redo search find replace

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

Results Explain Describe Saved SQL History

DEPARTMENT_ID
9
11
20
23

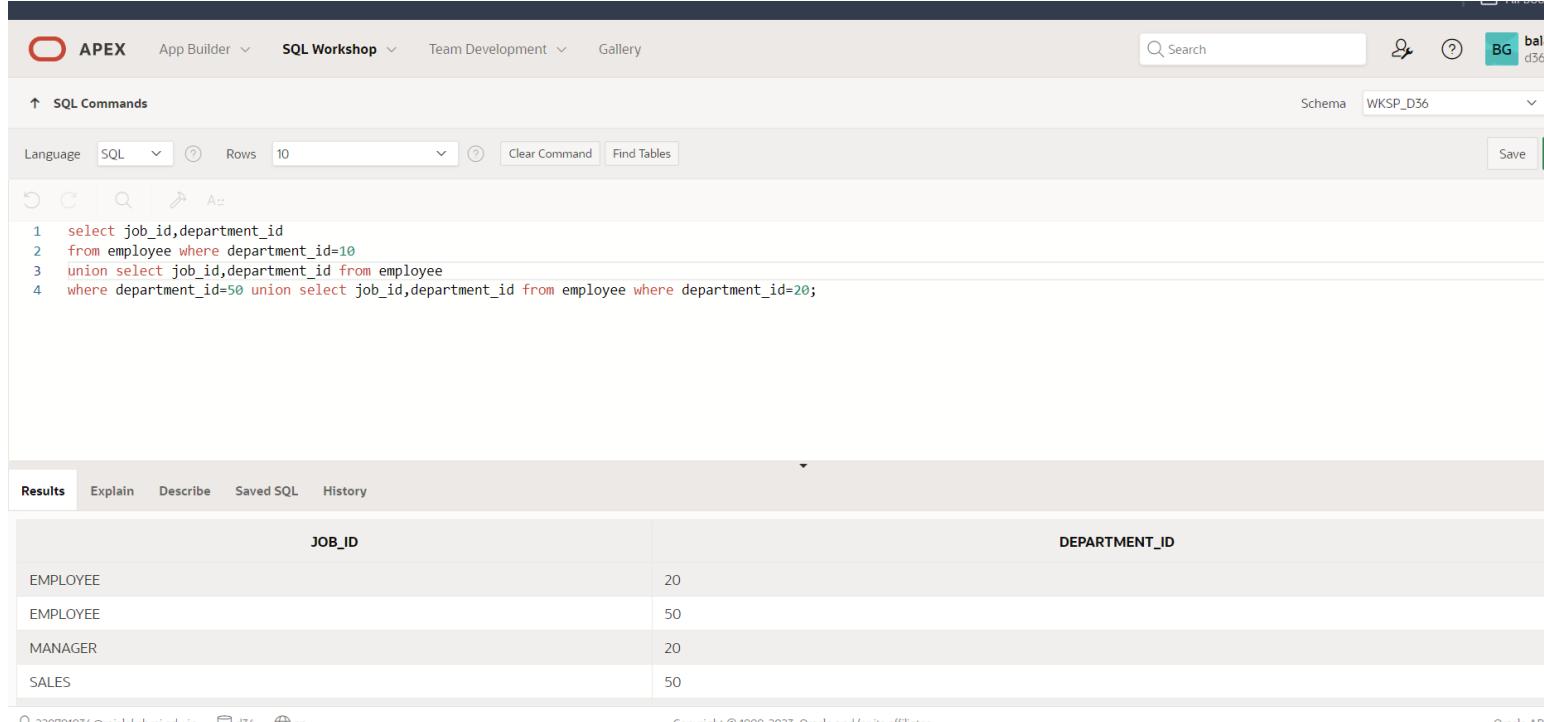
220701036@rajalakshmi.edu.in d36 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.0

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

**QUERY:**

```
select job_id,employee_id from employees intersect select e.job_id,e.employee_id from employees  
e,job_history j where e.job_id=j.old_job_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. 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 shows the following query:

```
1 select job_id,department_id  
2 from employee where department_id=10  
3 union select job_id,department_id from employee  
4 where department_id=50 union select job_id,department_id from employee where department_id=20;
```

The Results tab displays the output of the query:

JOB_ID	DEPARTMENT_ID
EMPLOYEE	20
EMPLOYEE	50
MANAGER	20
SALES	50

At the bottom, there are footer links for 220701036@rajalakshmi.edu.in, d36, en, Copyright © 1999, 2023, Oracle and/or its affiliates, and Oracle AP.

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 first_name||' '||last_name as "Name",department_id from employees union all select  
dept_name,dept_id from departments;
```

**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 are also present. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_D36'. The code editor contains the following SQL query:

```
1 select job_id,employee_id  
2 from employees intersect select e.job_id,e.employee_id  
3 from employees e,job_history j where e.job_id=j.old_job_id;
```

The results section displays the output of the query:

EMPLOYEE_ID	JOB_ID
40	CEO
176	EMPLOYEE
12	MANAGER
12	stock

At the bottom, the footer includes the URL '220701036@rajalakshmi.edu.in', session information 'd36 en', copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.', and the version 'Oracle APEX 23.2.4'.

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

**RESULT:**

# CREATING VIEWS

EX\_NO:11

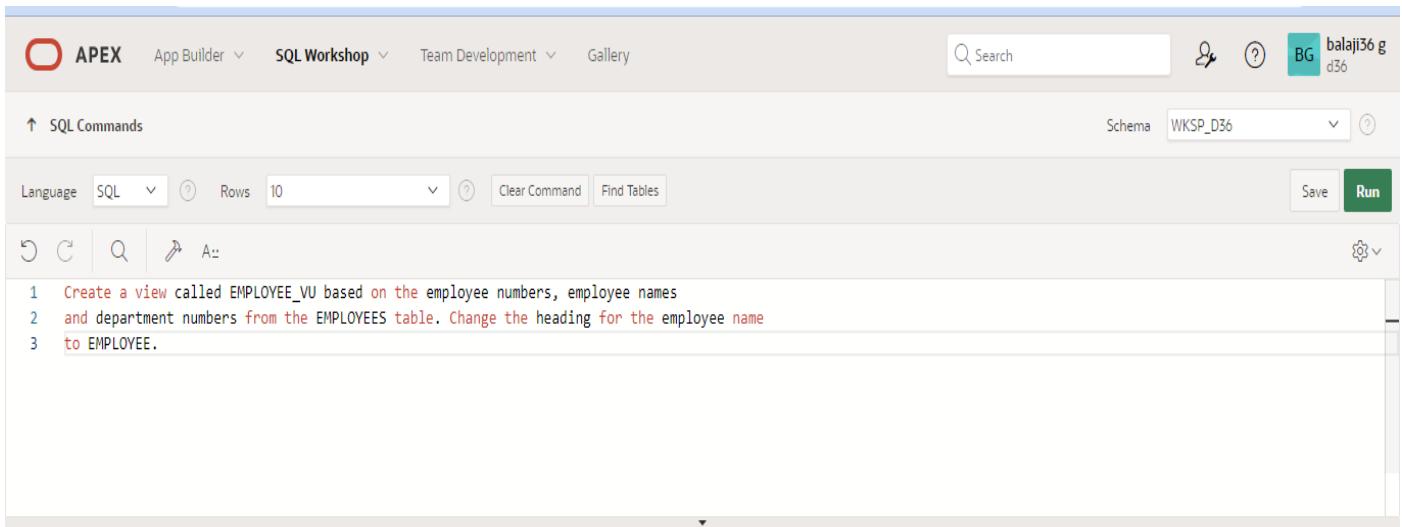
DATE:

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. 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 session identifier 'balaji36 g d36'. Below the navigation is a toolbar with icons for 'SQL Commands', 'Schema' set to 'WKSP\_D36', 'Save', and 'Run'. The main area contains a list of three commands:

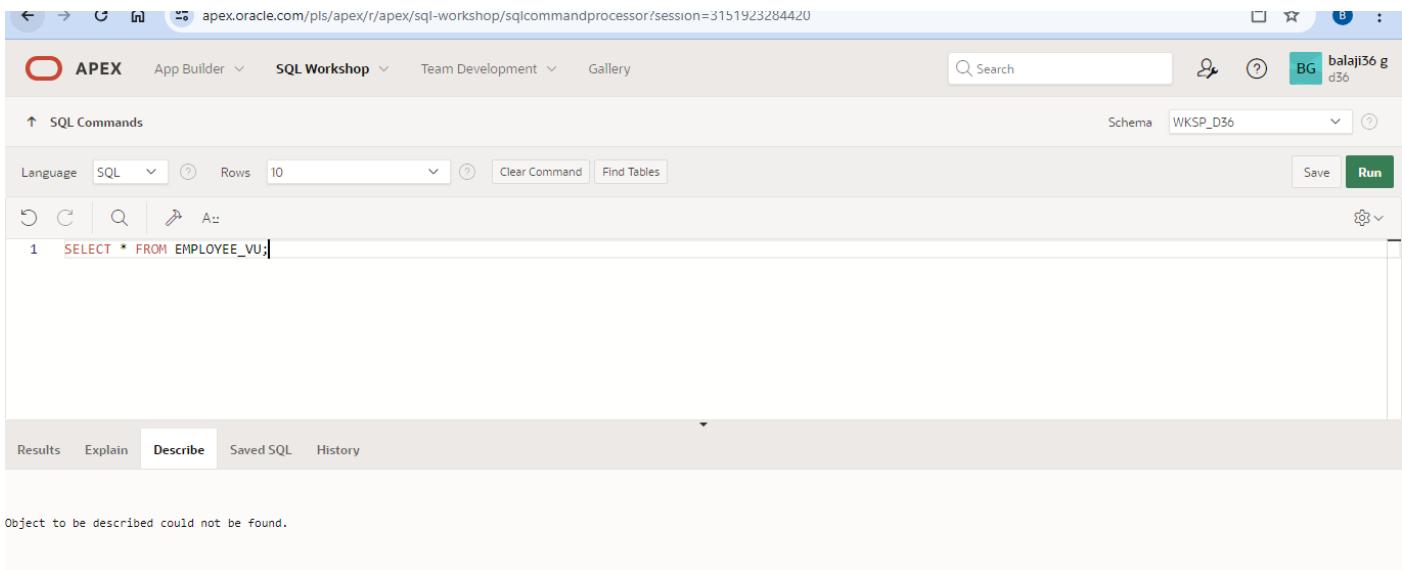
- 1 Create a view called EMPLOYEE\_VU based on the employee numbers, employee names and department numbers from the EMPLOYEES table. Change the heading for the employee name
- 2 to EMPLOYEE.
- 3 to EMPLOYEE.

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

**QUERY:**

```
select * from employees_vu;
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface with the 'Describe' tab selected. The top navigation bar and schema selection are identical to the previous screenshot. The main area shows the executed SQL command:

```
1 SELECT * FROM EMPLOYEE_VU;
```

Below the results, a message states: "Object to be described could not be found." At the bottom, the footer includes the URL "220701036@rajalakshmi.edu.in", the session ID "d36", the language "en", copyright information "Copyright © 1999, 2023, Oracle and/or its affiliates.", and the version "Oracle APEX 23.2.4".

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is active. The schema dropdown is set to 'WKSP\_D36'. The main area contains the SQL command: '1 SELECT view\_name, text FROM user\_views;'. Below the command, the 'Describe' tab is selected, showing the message 'Object to be described could not be found.' The bottom status bar displays the user's email (220701036@rajalakshmi.edu.in), session ID (d36), and language (en). The footer indicates Copyright © 1999, 2023, Oracle and/or its affiliates, and Oracle APEX 23.2.4.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is active. The schema dropdown is set to 'WKSP\_D36'. The main area contains the SQL command: '1 SELECT employee, department\_id FROM employees\_vu;'. Below the command, the 'Describe' tab is selected, showing the message 'Object to be described could not be found.' The bottom status bar displays the user's email (220701036@rajalakshmi.edu.in), session ID (d36), and language (en). The footer indicates Copyright © 1999, 2023, Oracle and/or its affiliates, and Oracle APEX 23.2.4.

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

#### QUERY:

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

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Commands tab is selected. The schema dropdown is set to WKSP\_D36. The SQL language dropdown is set to SQL. The command input area contains the following SQL code:

```
1 CREATE VIEW DEPT50 (EMPNO, EMPLOYEE, DEPTNO) AS
2 SELECT employee_number, last_name, department_number
3 FROM employees
4 WHERE department_number = 50
5 WITH CHECK OPTION;
```

The results section below the input area displays the message: "Object to be described could not be found." At the bottom of the page, the footer includes user information (220701036@rajalakshmi.edu.in), session details (d36), and copyright information (Copyright © 1999, 2023, Oracle and/or its affiliates). The page is identified as Oracle APEX 23.2.4.

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

#### QUERY:

```
Describe dept50;
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface, similar to the previous one. The SQL Commands tab is selected, and the schema is set to WKSP\_D36. The SQL language dropdown is set to SQL. The command input area contains the following SQL code:

```
1 DESCRIBE DEPT50;
2 SELECT * FROM DEPT50;
```

The results section below the input area displays the message: "Object to be described could not be found." At the bottom of the page, the footer includes user information (220701036@rajalakshmi.edu.in), session details (d36), and copyright information (Copyright © 1999, 2023, Oracle and/or its affiliates). The page is identified as Oracle APEX 23.2.4.

7.) Attempt to reassign Matos to department 80

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, there is a single line of code: `1 UPDATE DEPTS`. Below the code, an error message is displayed: `Object to be described could not be found.`. The bottom of the screen shows the user's email address, session ID, and copyright information.

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 SQL Commands tab, there is a multi-line query statement: `1 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 2 JOB_GRADE tables. Label the column Employee, Department, salary, and Grade respectively.|`. The bottom of the screen shows the user's email address, session ID, and copyright information.

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

**RESULT:**

# EXERCISE 12

## PRACTICE QUESTIONS

### Intro to Constraints; NOT NULL and UNIQUE Constraints

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## OUTPUT:

The screenshot shows the Oracle Application Express (APEX) interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. The schema dropdown shows 'WKSP\_D36'. The code editor contains the following SQL statement:

```
1 CREATE TABLE global_locations1 (
2     id NUMBER(6) NOT NULL PRIMARY KEY,
3     name VARCHAR2(50) NOT NULL,
4     date_opened DATE NOT NULL,
5     address VARCHAR2(50) NOT NULL,
6     city VARCHAR2(30) NOT NULL,
7     zip_postal_code VARCHAR2(12),
8     phone VARCHAR2(20),
9     email VARCHAR2(75) NOT NULL UNIQUE,
10    manager_id NUMBER(6)
11 );
```

Below the code editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, displaying the message 'Table created.'.

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

Table Created.

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

DESCRIBE f\_global\_locations;

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

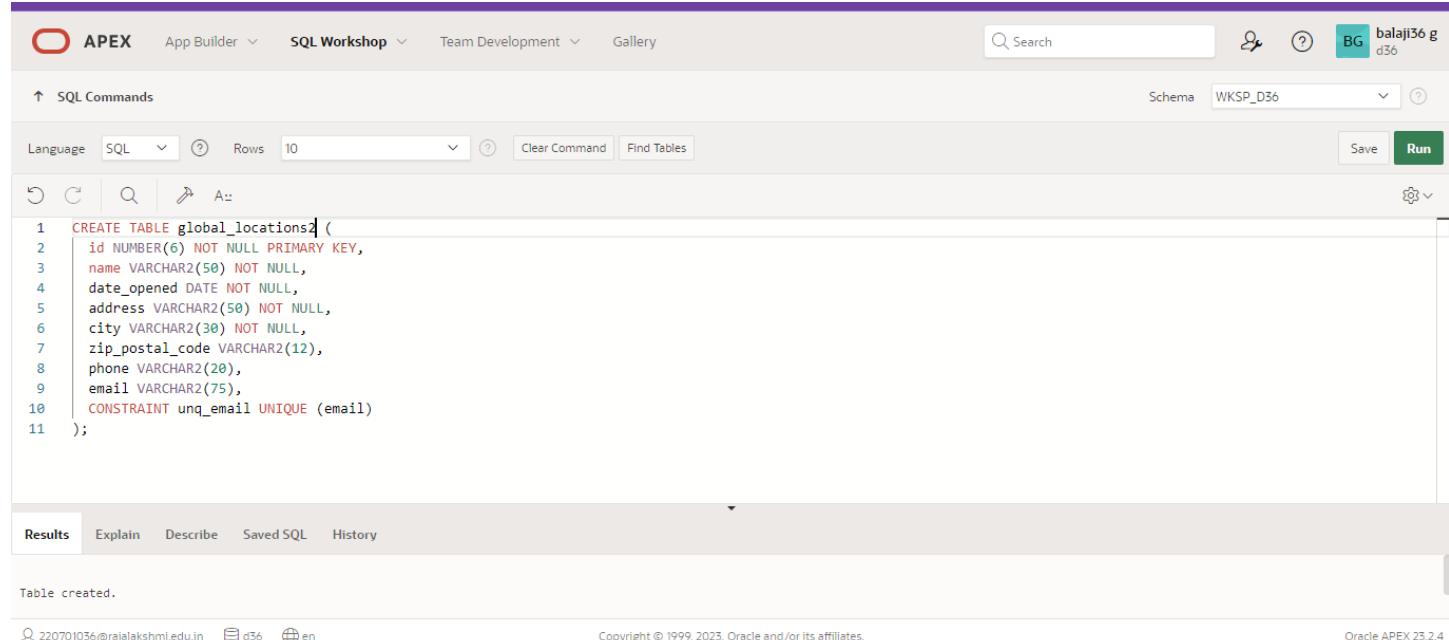
NAME	TYPE	LENGTH	PRECISION	SCALE	NULLABLE	DEFAULT
id	number	4				
loc_name	varchar2	20			X	
	date					
address	varchar2	30				
city	varchar2	20				
zip_postal	varchar2	20			X	
phone	varchar2	15			X	
email	varchar2	80			X	
manager_id	number	4			X	
contact	varchar2	40			X	

```

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

```

## 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 are user profile icons for 'balaji56 g' and 'd36'. The main workspace is titled 'SQL Commands' and contains the SQL code for creating the 'global\_locations' table. The code is numbered from 1 to 11. The 'Run' button is visible at the bottom right of the command input area. Below the command input, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The results tab displays the message 'Table created.'.

```

1 CREATE TABLE global_locations(
2   id NUMBER(6) NOT NULL PRIMARY KEY,
3   name VARCHAR2(50) NOT NULL,
4   date_opened DATE NOT NULL,
5   address VARCHAR2(50) NOT NULL,
6   city VARCHAR2(30) NOT NULL,
7   zip_postal_code VARCHAR2(12),
8   phone VARCHAR2(20),
9   email VARCHAR2(75),
10  CONSTRAINT unq_email UNIQUE (email)
11 );

```

Table created.

# 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_idNUMBER(6)	- PRIMARY KEY
name VARCHAR2(25)	
license_tag_numberNUMBER(10)	- UNIQUE
admit_date DATE	-NOT NULL
adoption_idNUMBER(5),	
vaccination_date DATE	-NOT NULL

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

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

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

ANIMAL_ID	NAME	LICENSE_TAG_NUMBE R	ADMIT_DAT E	ADOPTION_I D	VACCINATION_DAT E
101	Spot	35540	10-Oct-2004	205	12-Oct-2004

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

```
SELECT * FROM animals;
```

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

**COLUMN LEVEL STATEMENT:**

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

**TABLE LEVEL STATEMENT:**

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

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

a. ON DELETE CASCADE

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

b. ON DELETE SET NULL

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

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

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

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

# PRACTICE PROBLEM

## Managing Constraints

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

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

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

- ADD
- DROP
- ENABLE
- DISABLE

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

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

3. Create a FOREIGN KEY constraint in the copy\_d\_events table. Name the foreign key copy\_d\_events\_fk. This key references the copy\_d\_clients table client\_number column. What is the syntax you used to create the FOREIGN KEY constraint in the copy\_d\_events table?

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

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

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

a. The constraint name for the primary key in the copy\_d\_clients table is\_\_\_\_\_.

**COPY\_D\_CLT\_CLIENT\_NUMBER\_PK**

5. Drop the PRIMARY KEY constraint on the copy\_d\_clients table. Explain your results.

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

6. Add the following event to the copy\_d\_events table. Explain your results.

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

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

**RESULT:** ORA-02291: integrity constraint (HKUMAR.COPY\_D\_EVE\_CLIENT\_NUMBER\_FK) violated - parent key not found

7. Create an ALTER TABLE query to disable the primary key in the copy\_d\_clients table. Then add the values from #6 to the copy\_d\_events table. Explain your results.

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

8. Repeat question 6: Insert the new values in the copy\_d\_events table. Explain your results.

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

**1 row(s) inserted.**

9. Enable the primary-key constraint in the copy\_d\_clients table. Explain your results.

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

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

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

**1 row(s) deleted.**

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

**Table altered.**

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

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

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

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

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

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

# EXERCISE 13

## Creating Views

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

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

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

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

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

Results Explain Describe Saved SQL History

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

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

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

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

### 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 'balaji36 g d36'. The main area is titled 'SQL Commands' with a language dropdown set to 'SQL'. Below the dropdown are buttons for Undo, Redo, Find, Replace, and Autocomplete. The command window displays the following SQL code:

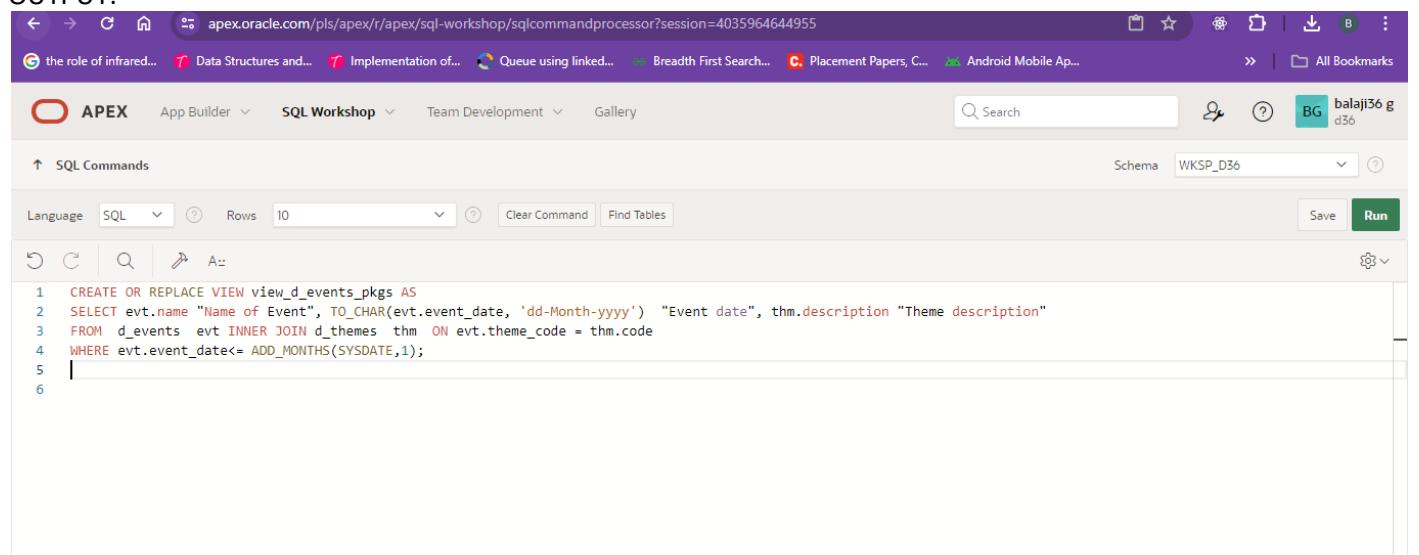
```
1 CREATE OR REPLACE VIEW view_d_songs AS
2 SELECT d_songs.id, d_songs.title "Song Title", d_songs.artist, d_songs.type_code
3 from d_songs INNER JOIN d_types ON d_songs.type_code = d_types.code
4 where d_types.description = 'New Age';
5
6
```

The code is numbered 1 through 6. The first four lines are highlighted in red, indicating they are part of the current command being run or edited. The fifth line is a blank line, and the sixth line is a comment. The 'Run' button is visible at the bottom right of the command window.

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);
```

#### OUTPUT:



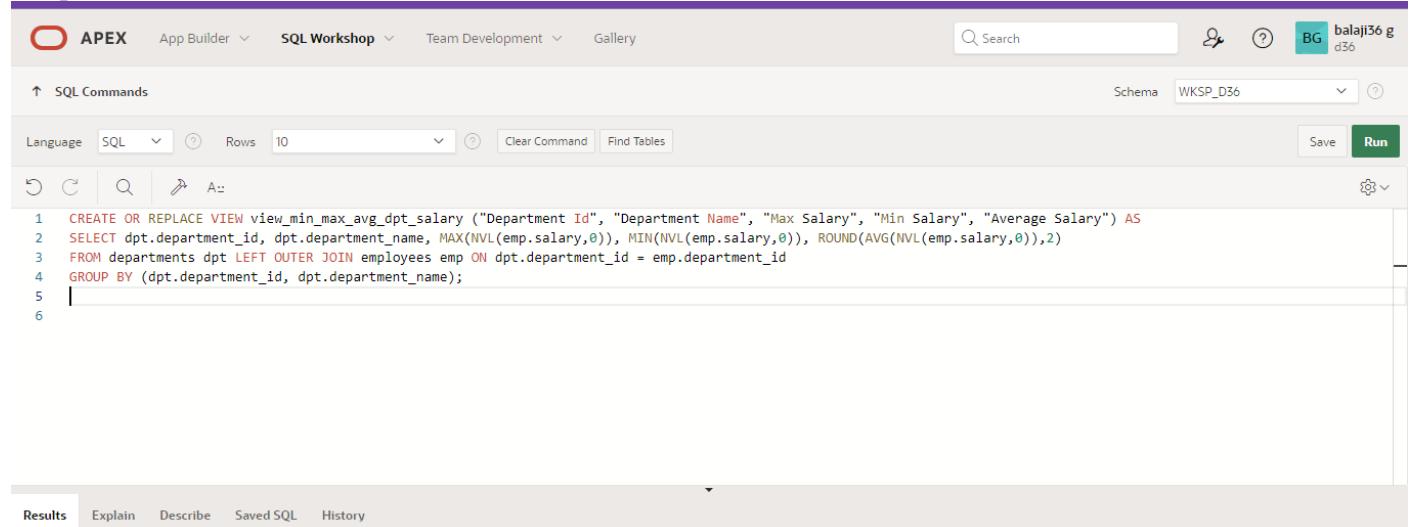
The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for 'the role of infrared...', 'Data Structures and...', 'Implementation of...', 'Queue using linked...', 'Breadth First Search...', 'Placement Papers, C...', 'Android Mobile Ap...', 'All Bookmarks', and user profile 'balaji36 g'. The main menu bar has items like 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar is at the top right. Below the menu, there's a toolbar with icons for 'SQL Commands', 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', 'Find Tables', 'Save', and 'Run'. The main workspace contains the SQL code for creating the 'view\_d\_events\_pkgs' view, with line numbers 1 through 6 on the left. The code is identical to the one provided in the question.

```
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",
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
6
```

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);
```

#### Output:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon, and a background switcher showing 'balaji56 g d36'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_D36'. Below it, there are buttons for Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run. The code area contains numbered lines 1 through 6 of the CREATE VIEW statement. The bottom navigation bar has tabs for Results, Explain, Describe, Saved SQL, and History, with 'Results' currently selected.

```
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  
6
```

# DML Operations and Views

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

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

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

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

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

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

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

```
SELECT * FROM view_copy_d_songs;
```

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

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

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

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

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

```
SELECT * FROM read_copy_d_cds;
```

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

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

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

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

7. Use the read\_copy\_d\_cds view to delete any CD of year 2000 from the underlying copy\_d\_cds.

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

8. Use the read\_copy\_d\_cds view to delete cd\_number 90 from the underlying copy\_d\_cds table.

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

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

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

10. Execute a SELECT \* statement for the base table copy\_d\_cds. What rows were deleted?

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

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

**DELETE,INSERT,MODIFY restricted if it contains:**

**Group functions**  
**GROUP BY CLAUSE**  
**DISTINCT**  
**pseudocolumnROWNUM Keyword**

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

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

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

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

# Managing Views

1. Create a view from the copy\_d\_songs table called view\_copy\_d\_songs that includes only the title and artist. Execute a SELECT \* statement to verify that the view exists.

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

```
SELECT * FROM view_copy_d_songs;
```

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

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

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

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

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

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

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

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

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

# **Indexes and Synonyms**

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

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

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

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

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

3. When will an index be created automatically?

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

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

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

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

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

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

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

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

```
CREATE SYNONYM dj_tracks FOR d_track_listings;
```

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

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

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

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

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

10. Drop the synonym that you created in question

**DROP SYNONYM dj\_tracks2;**

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

**RESULT:**

# OTHER DATABASE OBJECTS

**EX\_NO:14**

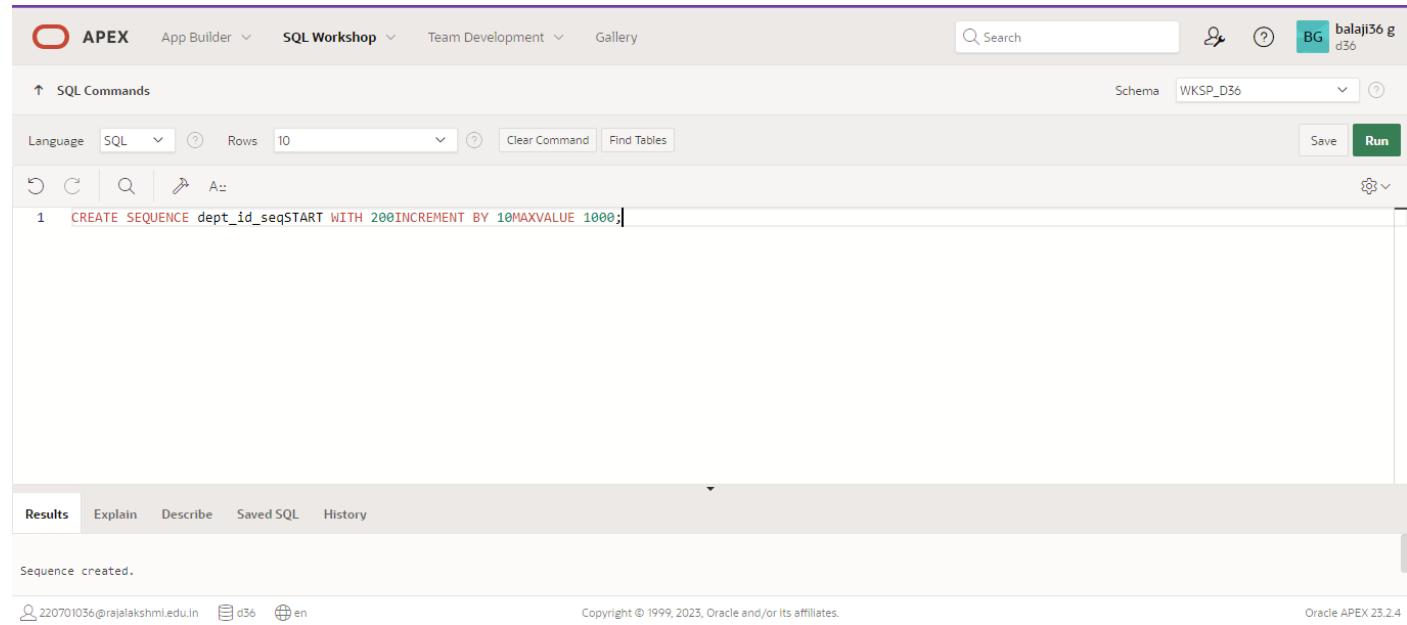
**DATE:**

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows 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 are user profile icons for 'balaji56 g' and 'd36'. The main workspace is titled 'SQL Commands' and shows the following command in the editor:

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

Below the editor, there are tabs for Results, Explain, Describe, Saved SQL, and History. The 'Results' tab is selected, displaying the message 'Sequence created.'.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL 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, user profile information (BG balaji56 d36), and a schema dropdown set to WKSP\_D36. The main workspace is titled 'SQL Commands' and contains a single line of SQL code: '1 SELECT sequence\_name, max\_value, increment\_by, last\_numberFROM user\_sequences;'. Below the code, the 'Results' tab is selected, showing the output: 'Sequence created.'. Other tabs include Explain, Describe, Saved SQL, and History. The bottom of the screen displays the user's email (220701036@rajalakshmi.edu.in), the session ID (d36), and the language (en). Copyright information for Oracle (© 1999, 2023) and the APEX version (25.2.4) are also present.

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

**QUERY:**

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

**OUTPUT:**

This screenshot is identical to the one above, showing the Oracle APEX SQL Workshop interface. It displays the same SQL command '1 INSERT INTO dept VALUES (dept\_id\_seq.nextval, 'Education');' in the workspace. The 'Results' tab is again selected, showing the output 'Sequence created.' This indicates that the sequence was successfully created before the insert command was run.

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

**QUERY:**

```
CREATE INDEX emp_dept_id_idx ON EMPLOYEES (department_id);
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL 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 'balaji36 g d36', and a schema dropdown set to 'WKSP\_D36'. The main workspace is titled 'SQL Commands' and contains a single line of SQL code: 'CREATE INDEX emp\_dept\_id\_idx ON EMPLOYEES (department\_id);'. Below the workspace are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The status bar at the bottom displays the user's email ('220701036@rajalakshmi.edu.in'), session ID ('d36'), and language ('en'). Copyright information ('Copyright © 1999, 2023, Oracle and/or its affiliates.') and the version ('Oracle APEX 23.2.4') are also visible.

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

**QUERY:**

```
SELECT index_name,table_name,uniquenessFROM user_indexesWHERE table_name='EMPLOYEES';
```

**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 'balaji36 g d36', and a schema dropdown set to 'WKSP\_D36'. The main workspace is titled 'SQL Commands' and contains a single line of SQL code: 'SELECT index\_name,table\_name,uniquenessFROM user\_indexesWHERE table\_name='EMPLOYEES';'. Below the workspace are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The status bar at the bottom displays the user's email ('220701036@rajalakshmi.edu.in'), session ID ('d36'), and language ('en'). Copyright information ('Copyright © 1999, 2023, Oracle and/or its affiliates.') and the version ('Oracle APEX 23.2.4') are also visible.

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

**RESULT:**

# CONTROLLING USER ACCESS

**EX\_NO:15**

**DATE:**

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

The CREATE SESSION system privilege

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

The CREATE TABLE privilege

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

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

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

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

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

The ALTER USER statement

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

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

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

7. Query all the rows in your DEPARTMENTS table.

SELECT \* FROM departments;

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

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

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

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

SELECT table\_name FROM user\_tables;

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

Team 1 revokes the privilege.

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

Team 2 revokes the privilege.

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

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

Team 1 executes this INSERT statement.

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

Team 2 executes this INSERT statement.

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

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

**RESULT:**

# PL/SQL

## CONTROL STRUCTURES

**EX\_NO:**

**DATE:**

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, there are tabs for "APEX", "App Builder", "SQL Workshop" (which is selected), "Team Development", and "Gallery". The URL in the browser bar is [apex.oracle.com/pls/apex/f?p=100:100:101498651039364](https://apex.oracle.com/pls/apex/f?p=100:100:101498651039364). The main workspace is titled "SQL Commands" and contains the following PL/SQL code:

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

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

```
Incentive = 5237.76
Statement processed.

0.01seconds
```

The bottom status bar displays the user's email (220701036@rajalakshmi.edu.in), the session ID (d56), and the system language (en). It also shows the date and time (21-05-2024 08:23) and the Oracle APEX version (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;
/
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, a PL/SQL block is written:

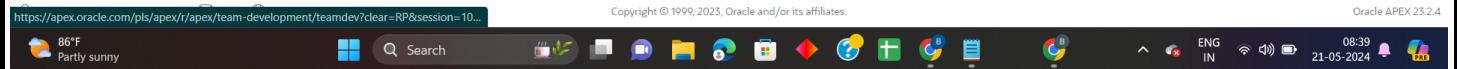
```
1 DECLARE
2   v_employee_id  NUMBER;
3 BEGIN
4   SELECT EMPLOYEE_ID INTO v_EMPLOYEE_ID
5   FROM "EMPLOYEE"
6   WHERE "EMPLOYEE".EMPLOYEE_ID = 110;
7
8   DBMS_OUTPUT.PUT_LINE('Employee ID = ' || TO_CHAR(v_employee_id));
9
10 -- This will raise an error because Oracle is case-sensitive for unquoted identifiers
11 SELECT employee_id INTO v_employee_id
12 FROM Emplovee;
```

The Results tab displays the output:

```
Employee ID = 110
Employee ID = 110

Statement processed.

0.01seconds
```



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

#### QUERY:

DECLARE

```
salary_of_emp NUMBER(8,2);
```

```
PROCEDURE approx_salary (
```

```
    emp      NUMBER,
```

```
    empsal IN OUT NUMBER,
```

```
    adddress  NUMBER
```

```
) IS
```

```
BEGIN
```

```
    empsal := empsal + adddress;
```

```
END;
```

```
BEGIN
```

```
    SELECT salary INTO salary_of_emp
```

```
    FROM employees
```

```
    WHERE employee_id = 122;
```

```
    DBMS_OUTPUT.PUT_LINE
```

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

```
    approx_salary (100, salary_of_emp, 1000);
```

```
    DBMS_OUTPUT.PUT_LINE
```

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

```
END;
```

```
/
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The code area contains a PL/SQL block. The output pane shows the results of the execution, including a message about salary adjustment and timing information. The system status bar at the bottom indicates network connection, battery level, and system time.

```
1 DECLARE
2     v_salary    NUMBER(8,2);
3 BEGIN
4     SELECT salary INTO v_salary
5     FROM employee
6     WHERE employee_id = 122;
7
8     -- Adjust the salary by adding a bonus of 10%
9     v_salary := v_salary * 1.10;
10
11    UPDATE employee
12    SET salary = v_salary
```

Salary of employee with ID 122 has been adjusted to 8321.5  
1 row(s) updated.  
0.01 seconds

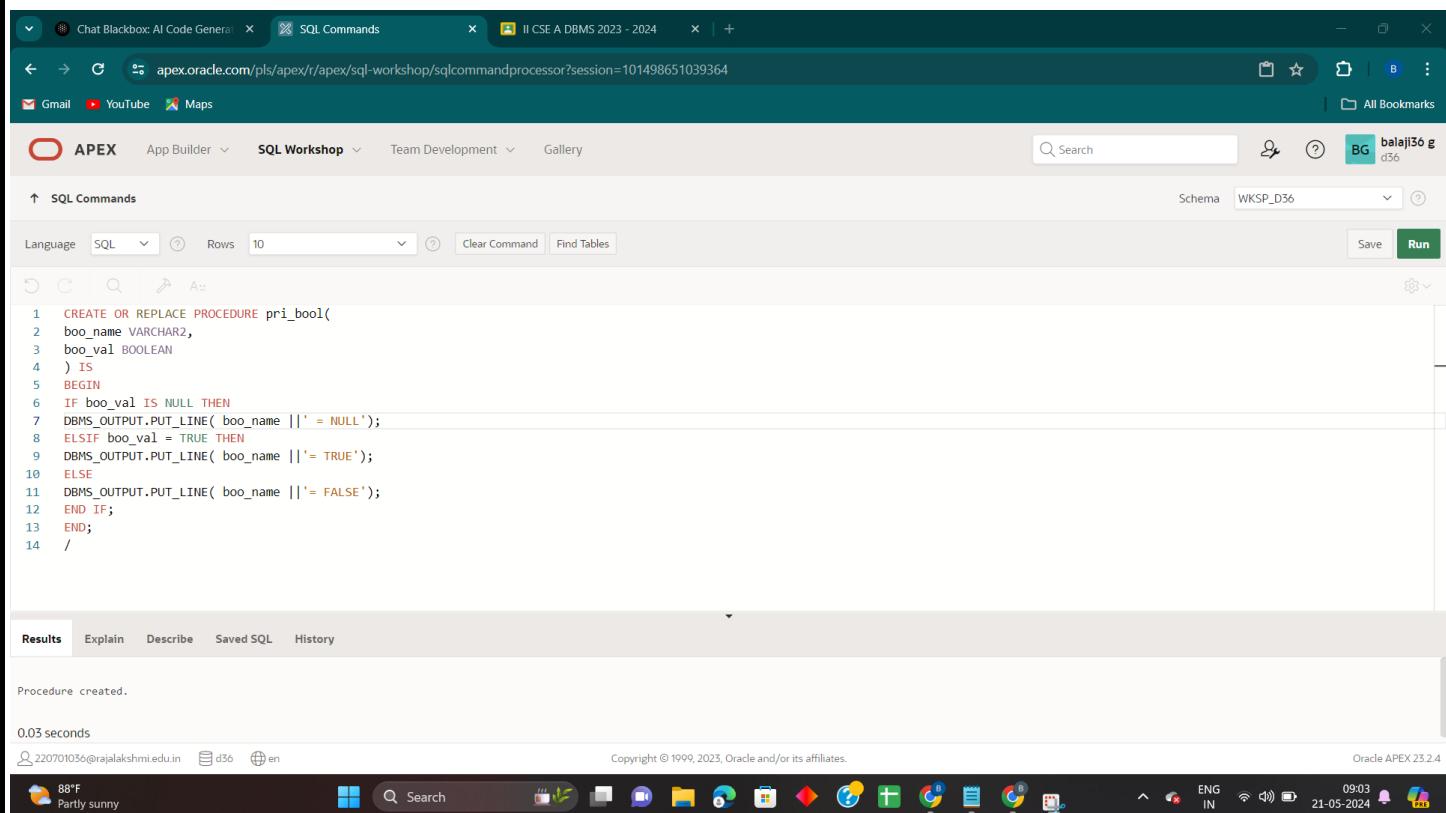
220701036@rajalakshmi.edu.in d36 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4 88°F Partly sunny Search ENG IN 08:52 21-05-2024

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. In the top navigation bar, there are tabs for Chat Blackbox: AI Code General, SQL Commands, and II CSE A DBMS 2023 - 2024. Below the tabs, the URL is apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=101498651039364. The main workspace is titled 'APEX' and shows the 'SQL Workshop' tab selected. The schema is set to WKSP\_D36. The code area contains the PL/SQL procedure definition provided in the question. The results section at the bottom shows the message 'Procedure created.' and a timestamp of '0.03 seconds'. The status bar at the bottom right indicates the user is connected to a Windows 10 system with a weather icon showing 88°F Partly sunny, and the date and time are 21-05-2024 09:03.

```
1 CREATE OR REPLACE PROCEDURE pri_bool(
2     boo_name VARCHAR2,
3     boo_val  BOOLEAN
4 ) IS
5 BEGIN
6 IF boo_val IS NULL THEN
7     DBMS_OUTPUT.PUT_LINE( boo_name || ' = NULL');
8 ELSIF boo_val = TRUE THEN
9     DBMS_OUTPUT.PUT_LINE( boo_name || ' = TRUE');
10 ELSE
11     DBMS_OUTPUT.PUT_LINE( boo_name || ' = FALSE');
12 END IF;
13 END;
14 /
```

Results Explain Describe Saved SQL History

Procedure created.

0.03 seconds

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

Oracle APEX 23.2.4

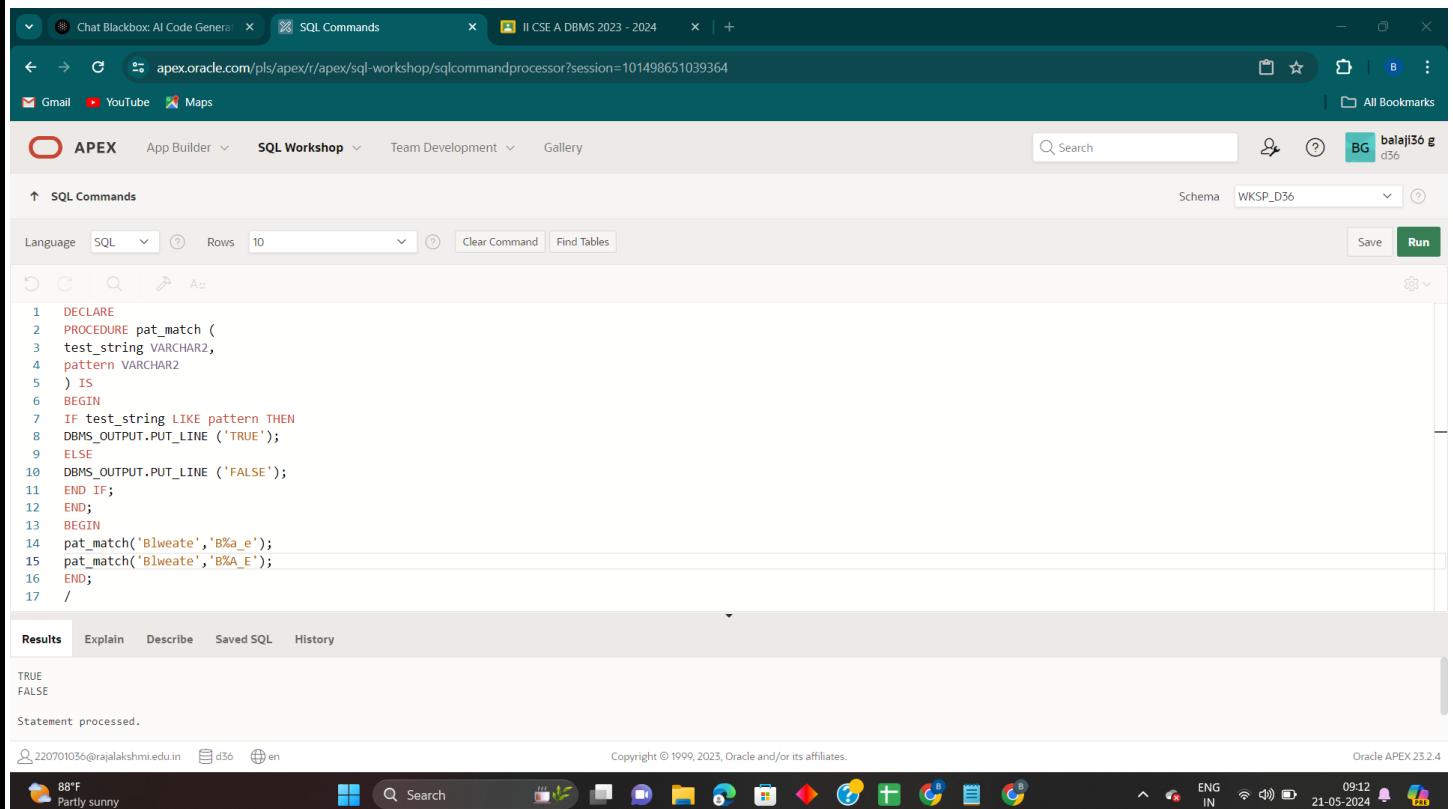
88°F Partly sunny Search ENG IN 21-05-2024 09:03

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

#### QUERY:

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

#### OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The code editor displays the provided PL/SQL block. The results tab shows the output: 'TRUE' followed by 'FALSE'. The status bar at the bottom indicates the statement was processed successfully.

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

Results

TRUE  
FALSE

Statement processed.

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

#### QUERY:

DECLARE

```
num_small NUMBER := 8;
```

```
num_large NUMBER := 5;
```

```
num_temp NUMBER;
```

```
BEGIN
```

```
IF num_small > num_large THEN
```

```
    num_temp := num_small;
```

```
    num_small := num_large;
```

```
    num_large := num_temp;
```

```
END IF;
```

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

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

```
END;
```

```
/
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The code area contains a PL/SQL block. The results tab shows the output of the DBMS\_OUTPUT.PUT\_LINE statements. The status bar at the bottom indicates the statement was processed.

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

Results

```
num_small = 5
num_large = 8

Statement processed.
```

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

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

#### QUERY:

DECLARE

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

BEGIN

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

END;

/

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes tabs for Chat Blackbox AI Code General, SQL Commands, II CSE A DBMS 2023 - 2024, and others. Below the navigation is a toolbar with various icons. 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      IF num_small > num_large THEN
7          num_temp := num_small;
8          num_small := num_large;
9          num_large := num_temp;
10     END IF;
11    DBMS_OUTPUT.PUT_LINE ('num_small = '||num_small);
12    DBMS_OUTPUT.PUT_LINE ('num_large = '||num_large);
13 END;
14 /
```

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

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

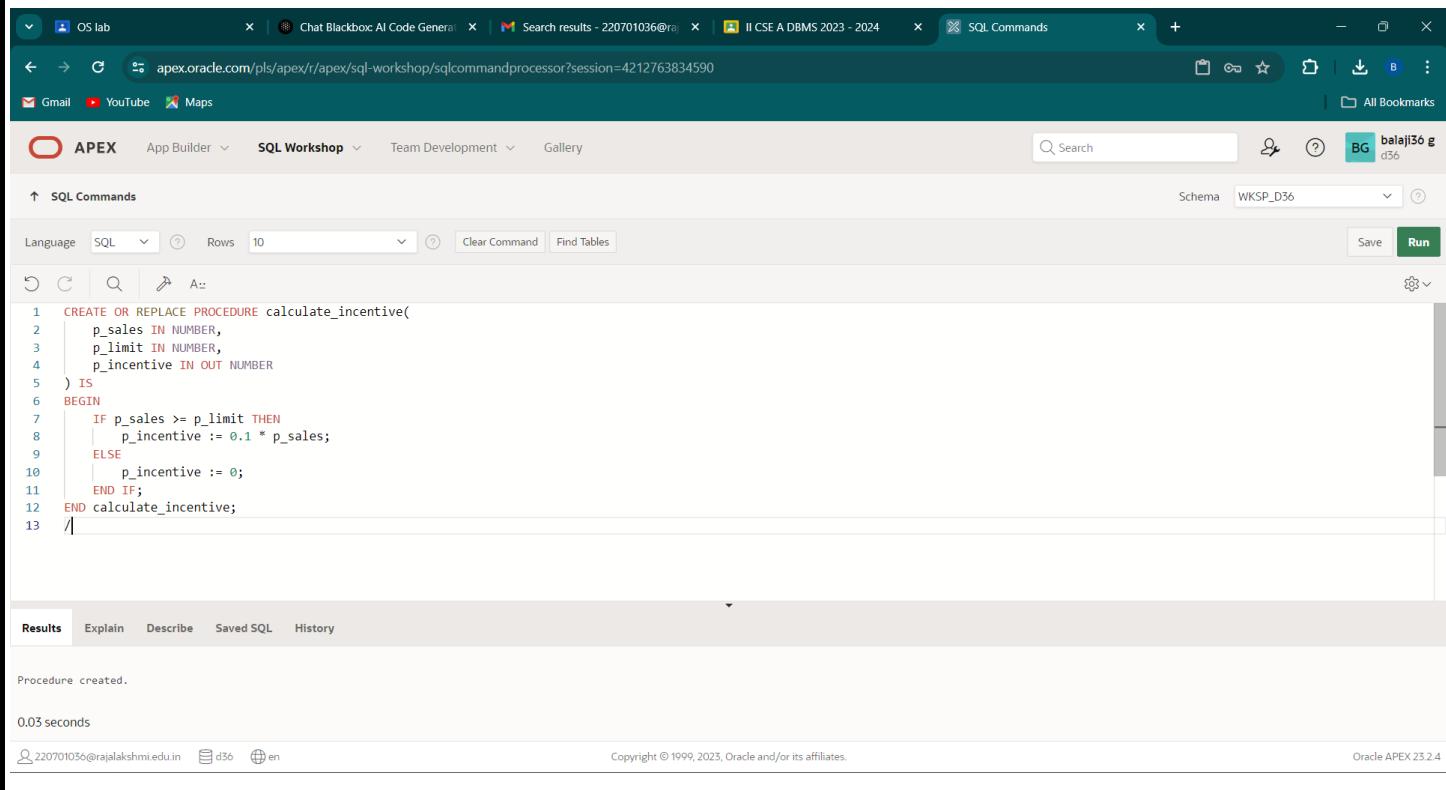
The bottom of the screen shows the operating system's taskbar with various application icons and the date/time: 21-05-2024 09:25.

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. In the top navigation bar, the 'SQL Commands' tab is selected. The main workspace displays the following PL/SQL code:

```
1 CREATE OR REPLACE PROCEDURE calculate_incentive(
2   p_sales IN NUMBER,
3   p_limit IN NUMBER,
4   p_incentive IN OUT NUMBER
5 ) IS
6 BEGIN
7   IF p_sales >= p_limit THEN
8     p_incentive := .01 * p_sales;
9   ELSE
10    p_incentive := 0;
11  END IF;
12 END calculate_incentive;
13 /
```

Below the code, the 'Results' tab is active, showing the output: "Procedure created." and "0.03 seconds". At the bottom, the footer includes the URL "220701036@rajalakshmi.edu.in", the session ID "d56", the language "en", copyright information "Copyright © 1999, 2023, Oracle and/or its affiliates.", and the version "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

DM employees e

join departments d

ON e.department\_id = d.department\_id

```
WHERE e.department_id = get_dep_id;
```

ut.Put\_line ('The employee

| | To\_char(

F tot\_emp >= 45 THEN

dbn

```
    dbms_output.Put_line ('There are '||to_char(45-tot_emp)||' vacancies in department'||  
    c_dep_id);
```

- 1 -

END:

1

## OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for OS lab, Chat Blackbox: AI Code General, Search results - 220701036@ra, II CSE A DBMS 2023 - 2024, and SQL Commands. Below the bar, the URL is apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=4212763834590. The main workspace displays a PL/SQL script:

```
1 DECLARE
2   v_dept_id NUMBER := 50;
3   v_employees NUMBER;
4   v_vacancies NUMBER := 45;
5   v_available_vacancies NUMBER;
6 BEGIN
7   SELECT COUNT(*) INTO v_employees
8   FROM employees
9   WHERE department_id = v_dept_id;
10
11   v_available_vacancies := v_vacancies - v_employees;
12
13   DBMS_OUTPUT.PUT_LINE('Number of employees in department ' || v_dept_id || ': ' || v_employees);
14
15   IF v_available_vacancies > 0 THEN
16     DBMS_OUTPUT.PUT_LINE('Department ' || v_dept_id || ' has ' || v_available_vacancies || ' vacancies available.');
17   ELSE
```

The script calculates the number of employees in department 50 and the available vacancies. It then prints these values to the screen. The IF block handles the case where there are available vacancies.

Procedure created.

0.03 seconds

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

#### QUERY:

DECLARE

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

BEGIN

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

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

IF tot\_emp >= 45 THEN

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

ELSE

```
dbms_output.Put_line ('There are'||to_char(45-tot_emp)||' vacancies in department'||  
get_dep_id );
```

END IF;

END;

/

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The code area contains a PL/SQL block. The results pane at the bottom shows the output: "Procedure created." and "0.03 seconds". The status bar at the bottom right indicates "Oracle APEX 23.2.4".

```
1  DECLARE  
2      v_dept_id NUMBER;  
3      v_employees NUMBER;  
4      v_vacancies NUMBER;  
5      v_available_vacancies NUMBER;  
6  BEGIN  
7      -- Get the department ID from the user  
8      DBMS_OUTPUT.PUT_LINE('Enter the department ID:');  
9      v_dept_id := &dept_id;  
10     -- Count the number of employees in the specified department  
11     SELECT COUNT(*) INTO v_employees  
12     FROM employees  
13     WHERE department_id = v_dept_id;  
14  
15     -- Get the number of vacancies in the specified department  
16     SELECT vacancies INTO v_vacancies  
17
```

Results Explain Describe Saved SQL History

Procedure created.  
0.03 seconds

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

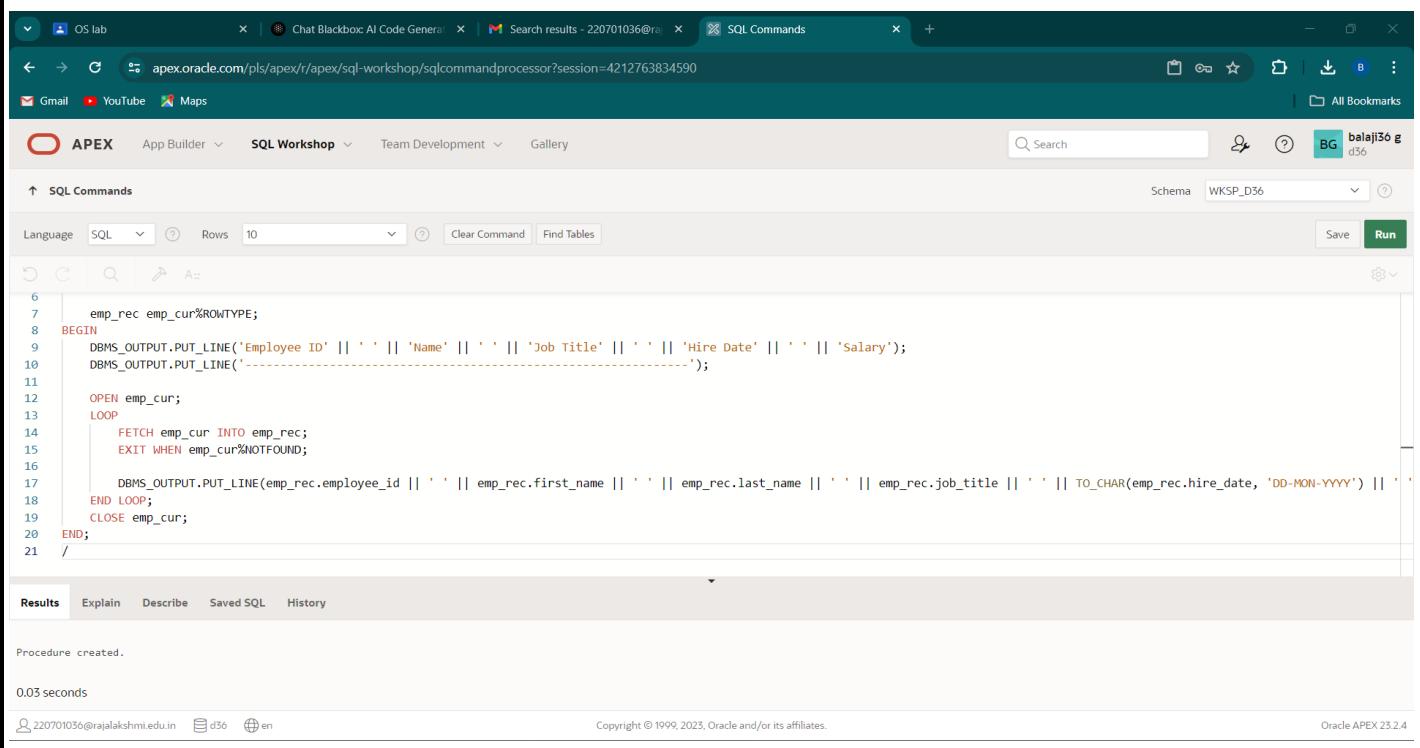
Oracle APEX 23.2.4

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

**QUERY:**

```
DECLARE
v_employee_id employees.employee_id%TYPE;
v_full_name employees.first_name%TYPE;
v_job_id employees.job_id%TYPE;
v_hire_date employees.hire_date%TYPE;
v_salary employees.salary%TYPE;
CURSOR c_employees IS
  SELECT employee_id, first_name || ' ' || last_name AS full_name, job_id, hire_date, salary
  FROM employees;
BEGIN
  DBMS_OUTPUT.PUT_LINE('Employee ID | Full Name | Job Title | Hire Date | Salary');
  DBMS_OUTPUT.PUT_LINE('-----');
  OPEN c_employees;
  FETCH c_employees INTO v_employee_id, v_full_name, v_job_id, v_hire_date, v_salary;
  WHILE c_employees%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(v_employee_id || ' ' || v_full_name || ' ' || v_job_id || ' ' ||
v_hire_date || ' ' || v_salary);
    FETCH c_employees INTO v_employee_id, v_full_name, v_job_id, v_hire_date, v_salary;
  END LOOP;
  CLOSE c_employees;
END;
/
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The code area contains the PL/SQL block provided above. The output pane at the bottom displays the message "Procedure created." and "0.03 seconds". The status bar at the bottom right indicates "Oracle APEX 23.2.4".

```
6
7   emp_rec emp_cur%ROWTYPE;
8
9 BEGIN
10   DBMS_OUTPUT.PUT_LINE('Employee ID' || ' ' || 'Name' || ' ' || 'Job Title' || ' ' || 'Hire Date' || ' ' || 'Salary');
11   DBMS_OUTPUT.PUT_LINE('-----');
12
13   OPEN emp_cur;
14   LOOP
15     FETCH emp_cur INTO emp_rec;
16     EXIT WHEN emp_cur%NOTFOUND;
17
18     DBMS_OUTPUT.PUT_LINE(emp_rec.employee_id || ' ' || emp_rec.first_name || ' ' || emp_rec.last_name || ' ' || emp_rec.job_title || ' ' || TO_CHAR(emp_rec.hire_date, 'DD-MON-YYYY') || ' ');
19   END LOOP;
20   CLOSE emp_cur;
21 /
```

Procedure created.  
0.03 seconds

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

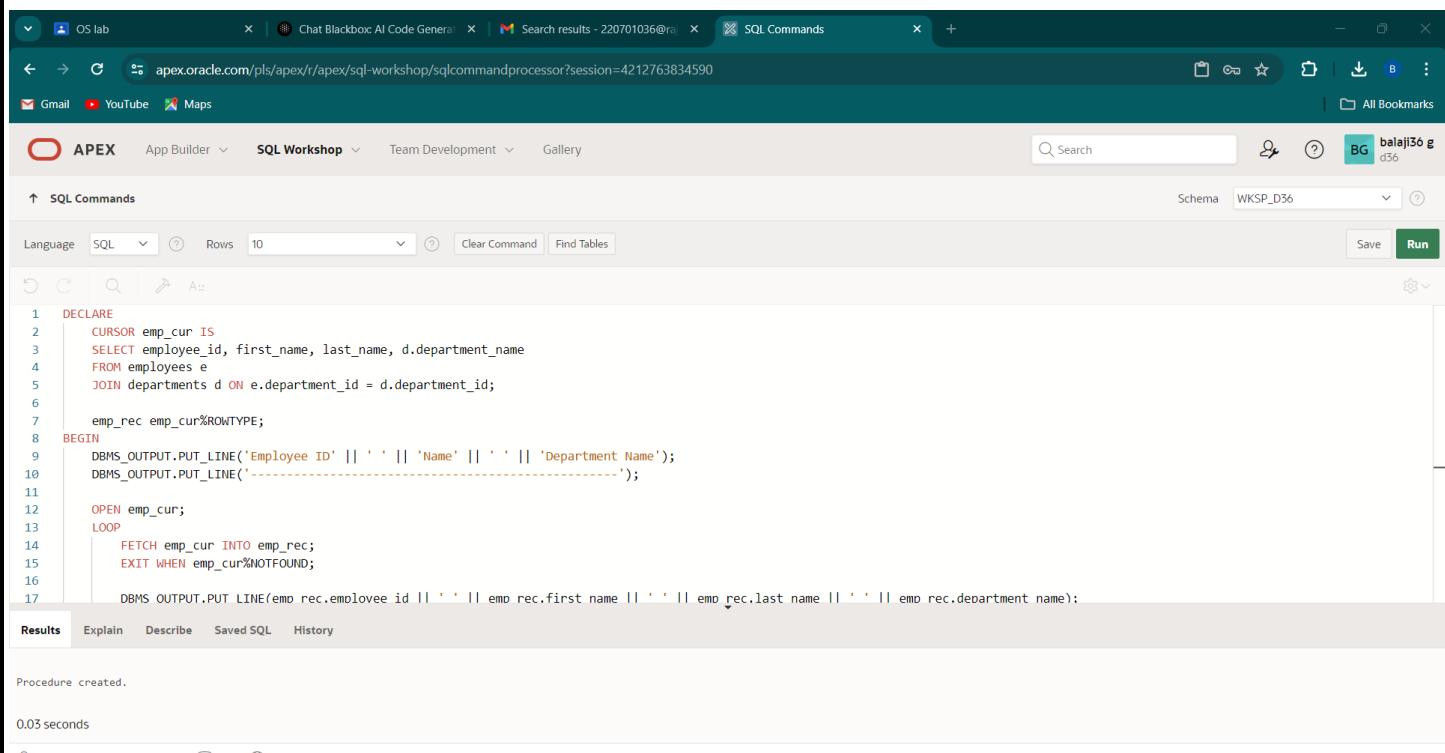
**QUERY:**

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 code area contains a PL/SQL block that retrieves employee information from the employees and departments tables, using DBMS\_OUTPUT to print the results. The output pane at the bottom shows the message "Procedure created." and a timestamp of "0.03 seconds". The status bar at the bottom right indicates the session is "balaji36 g d36" and the version is "Oracle APEX 23.2.4".

```
1  DECLARE
2    CURSOR emp_cur IS
3      SELECT employee_id, first_name, last_name, d.department_name
4      FROM employees e
5      JOIN departments d ON e.department_id = d.department_id;
6
7    emp_rec emp_cur%ROWTYPE;
8
9    BEGIN
10      DBMS_OUTPUT.PUT_LINE('Employee ID' || ' ' || 'Name' || ' ' || 'Department Name');
11      DBMS_OUTPUT.PUT_LINE('-----');
12
13      OPEN emp_cur;
14      LOOP
15        FETCH emp_cur INTO emp_rec;
16        EXIT WHEN emp_cur%NOTFOUND;
17
18        DBMS_OUTPUT.PUT_LINE(emp_rec.employee_id || ' ' || emp_rec.first_name || ' ' || emp_rec.last_name || ' ' || emp_rec.department_name);

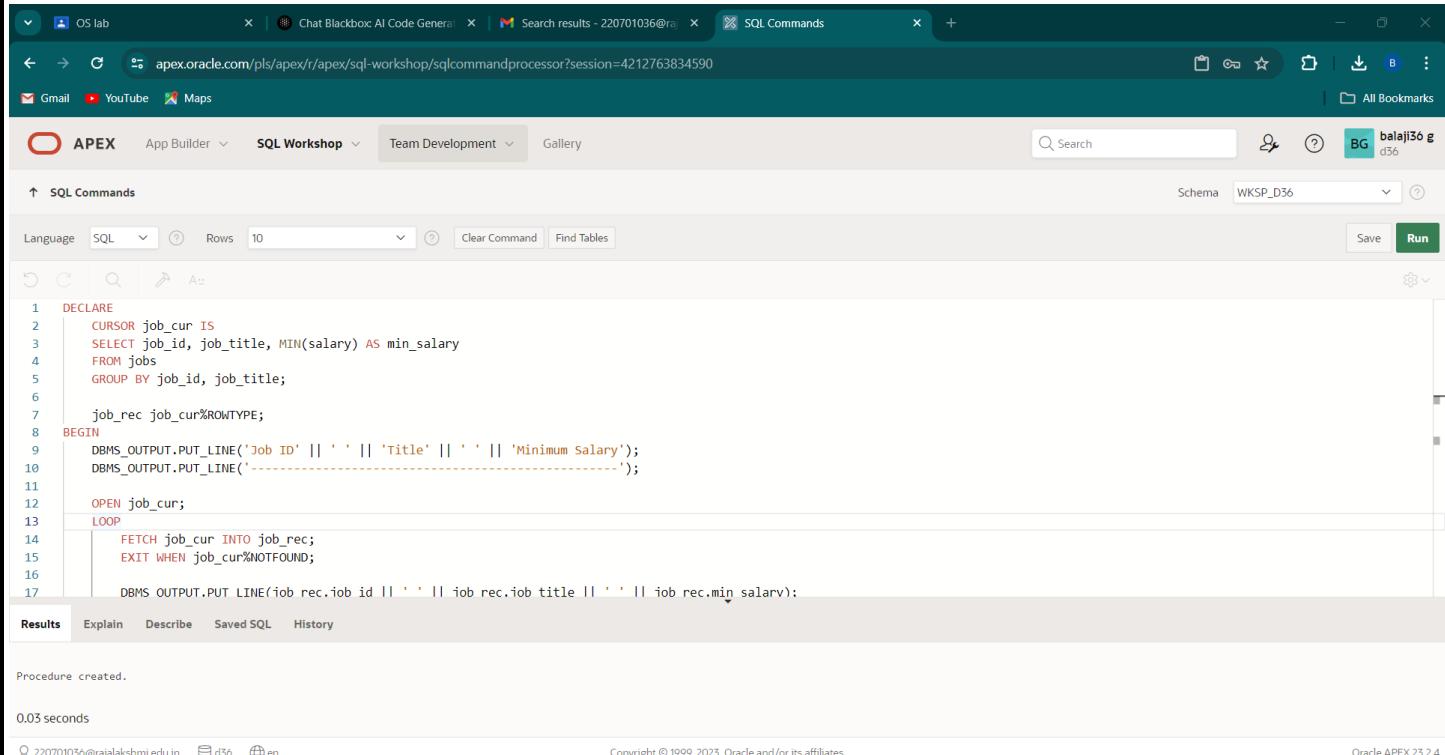
```

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes tabs for OS lab, Chat Blackbox AI Code General, Search results - 220701036@r..., and SQL Commands. Below the navigation bar, the browser address bar shows apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=4212763834590. The main workspace is titled "APEX" and contains a "SQL Workshop" tab. The SQL editor window displays the provided PL/SQL code. The code uses a cursor to select job IDs, titles, and minimum salaries from the jobs and employees tables, then prints them to the output using DBMS\_OUTPUT.PUT\_LINE. The code is numbered from 1 to 17. The bottom of the screen shows the "Results" tab is selected, displaying the message "Procedure created." and "0.03 seconds". The footer includes copyright information for Oracle APEX 23.2.4.

```
1  DECLARE
2    CURSOR job_cur IS
3      SELECT job_id, job_title, MIN(salary) AS min_salary
4        FROM jobs
5       GROUP BY job_id, job_title;
6
7    job_rec job_cur%ROWTYPE;
8
9 BEGIN
10   DBMS_OUTPUT.PUT_LINE('Job ID' || ' ' || 'Title' || ' ' || 'Minimum Salary');
11   DBMS_OUTPUT.PUT_LINE('-----');
12
13   OPEN job_cur;
14
15   LOOP
16     FETCH job_cur INTO job_rec;
17     EXIT WHEN job_cur%NOTFOUND;
18
19     DBMS_OUTPUT.PUT_LINE(job_rec.job_id || ' ' || job_rec.job_title || ' ' || job_rec.min_salary);
20
21   END LOOP;
22
23   CLOSE job_cur;
24
25 END;
26
```

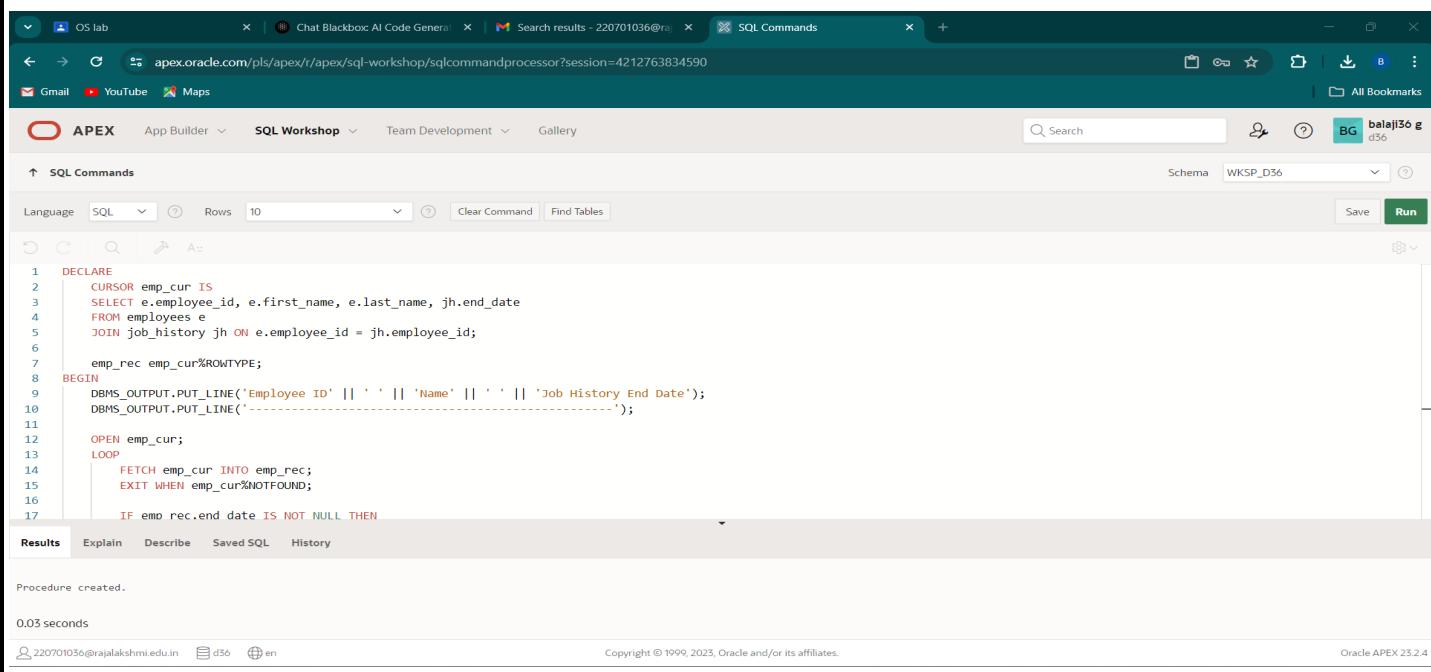
**14.)** Write a PL/SQL program to display the employee IDs, names, and job history start dates of all employees.

**QUERY:**

DECLARE

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

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The code has been entered into the SQL Commands editor. The output pane at the bottom displays the message "Procedure created." and "0.03 seconds". The status bar at the bottom right indicates "Oracle APEX 23.2.4".

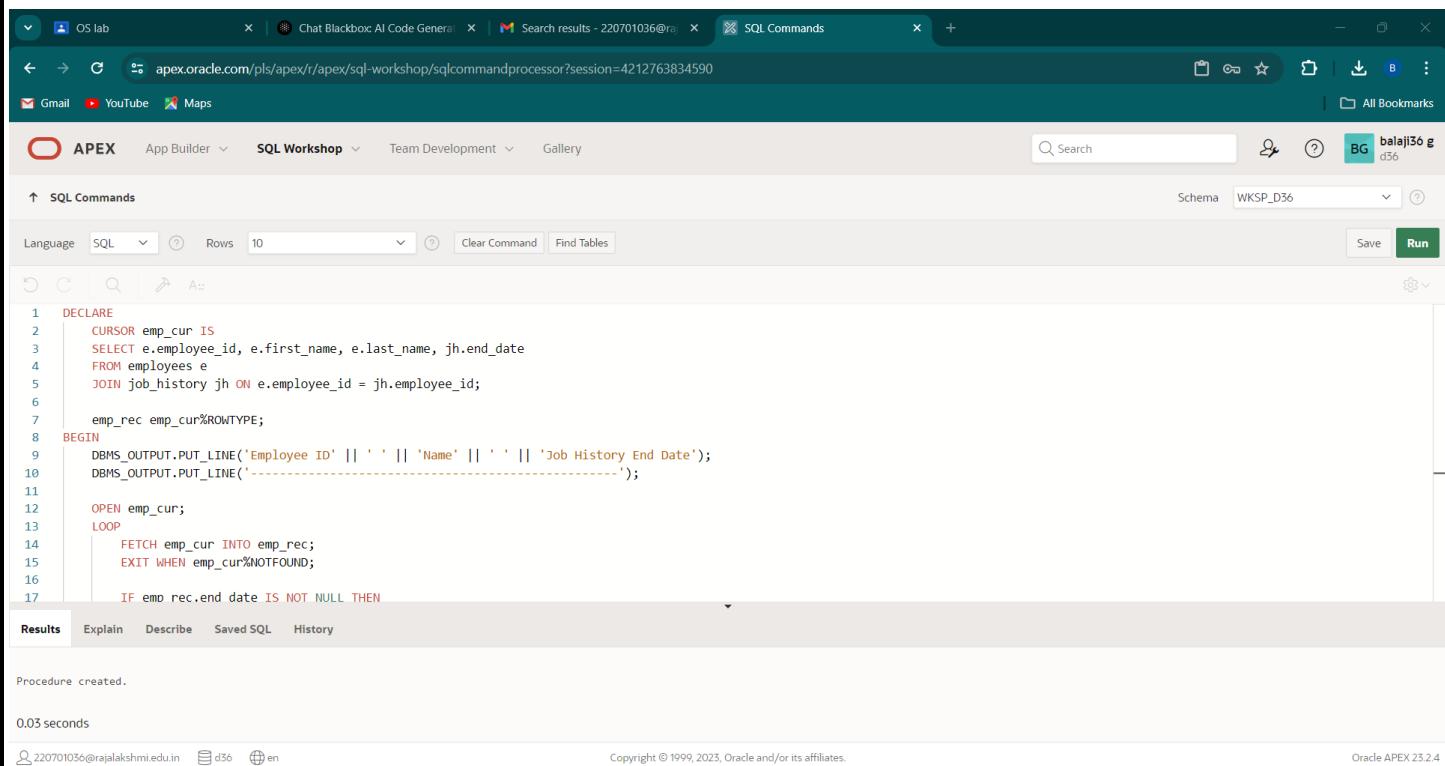
```
1  DECLARE
2    CURSOR emp_cur IS
3      SELECT e.employee_id, e.first_name, e.last_name, jh.end_date
4      FROM employees e
5      JOIN job_history jh ON e.employee_id = jh.employee_id;
6
7    emp_rec emp_cur%ROWTYPE;
8  BEGIN
9    DBMS_OUTPUT.PUT_LINE('Employee ID' || ' ' || 'Name' || ' ' || 'Job History End Date');
10   DBMS_OUTPUT.PUT_LINE('-----');
11
12   OPEN emp_cur;
13   LOOP
14     FETCH emp_cur INTO emp_rec;
15     EXIT WHEN emp_cur%NOTFOUND;
16
17     IF emp_rec.end_date IS NOT NULL THEN
```

**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 OS lab, Chat Blackbox AI Code General, Search results - 220701036@r..., and SQL Commands. Below the bar, the main workspace displays the PL/SQL code. The code is numbered from 1 to 17. Lines 1 through 16 show the declaration of a cursor, its selection statement, and a loop structure. Line 17 contains an IF statement. The bottom of the screen shows the results of the execution, indicating "Procedure created." and a execution time of "0.03 seconds". The status bar at the bottom right shows "Oracle APEX 23.2.4".

```
1  DECLARE
2      CURSOR emp_cur IS
3          SELECT e.employee_id, e.first_name, e.last_name, jh.end_date
4          FROM employees e
5          JOIN job_history jh ON e.employee_id = jh.employee_id;
6
7      emp_rec emp_cur%ROWTYPE;
8
9      BEGIN
10         DBMS_OUTPUT.PUT_LINE('Employee ID' || ' ' || 'Name' || ' ' || 'Job History End Date');
11         DBMS_OUTPUT.PUT_LINE('-----');
12
13         OPEN emp_cur;
14         LOOP
15             FETCH emp_cur INTO emp_rec;
16             EXIT WHEN emp_cur%NOTFOUND;
17             IF emp_rec.end_date IS NOT NULL THEN
```

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

Faculty Signature	
-------------------	--

# PROCEDURES AND FUNCTIONS

EX\_NO: 17

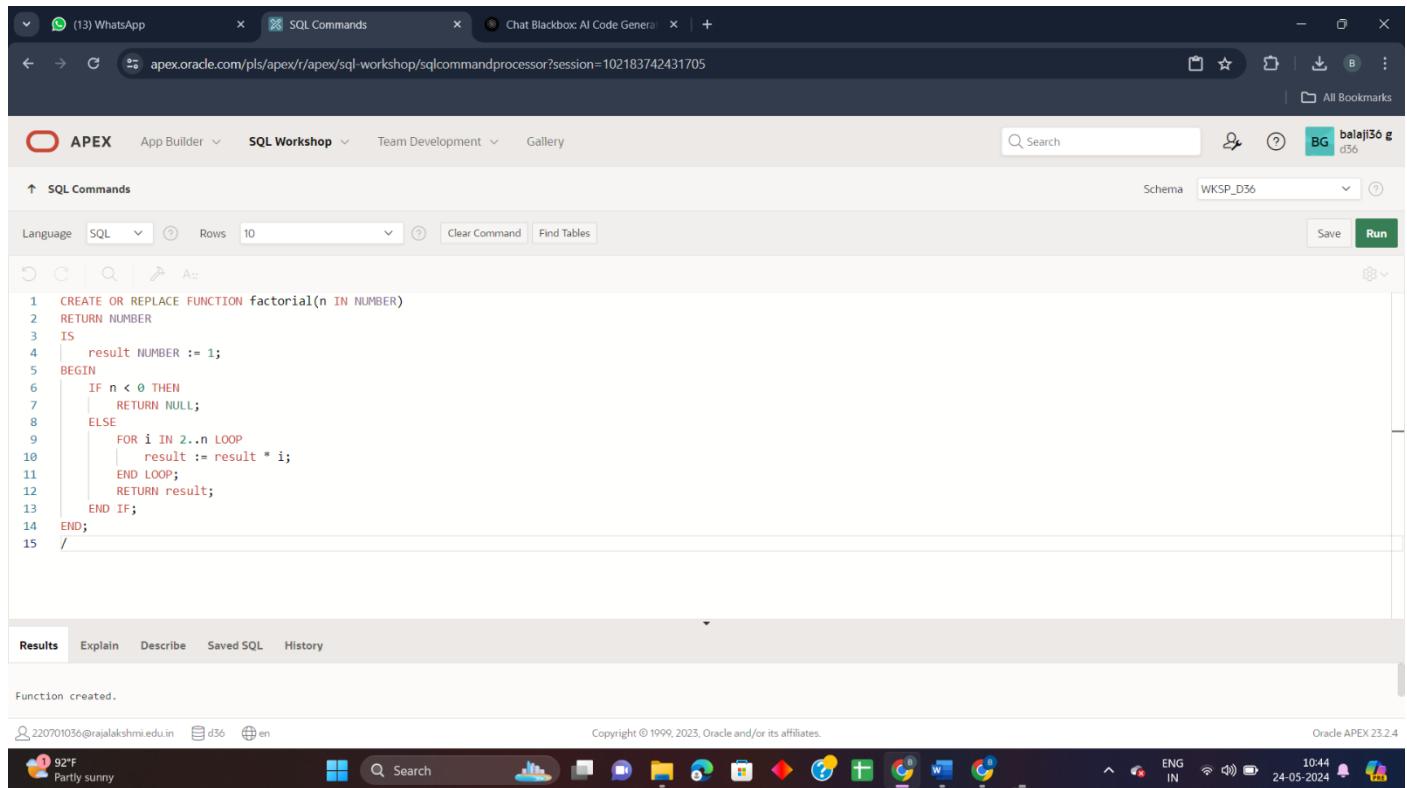
DATE:

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. In the main editor area, a PL/SQL code block is displayed for creating a function named 'factorial'. The code uses a FOR loop to calculate the factorial of a given number 'n'. The schema is set to 'WKSP\_D36'. The results tab at the bottom shows the message 'Function created.'.

```
1 CREATE OR REPLACE FUNCTION factorial(n IN NUMBER)
2 RETURN NUMBER
3 IS
4     result NUMBER := 1;
5 BEGIN
6     IF n < 0 THEN
7         RETURN NULL;
8     ELSE
9         FOR i IN 2..n LOOP
10            result := result * i;
11        END LOOP;
12        RETURN result;
13    END IF;
14 END;
15 /
```

**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. In the top navigation bar, the tabs "APEX", "App Builder", "SQL Workshop" (which is selected), and "Team Development" are visible. The main area is titled "SQL Commands". The code editor contains the following PL/SQL function definition:

```
1 CREATE OR REPLACE FUNCTION get_book_info_cursor(p_book_id IN NUMBER)
2 RETURN SYS_REFCURSOR
3 IS
4   v_cursor SYS_REFCURSOR;
5 BEGIN
6   OPEN v_cursor FOR
7     SELECT id, title, author
8       FROM books
9      WHERE id = p_book_id;
10  RETURN v_cursor;
11 END;
12 /
```

Below the code editor, there are tabs for "Results", "Explain", "Describe", "Saved SQL", and "History". The "Results" tab is selected, showing the message "Function created.". The status bar at the bottom displays system information, including the user's email (220701036@rajalakshmi.edu.in), the session ID (d36), the language (en), the copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the version (Oracle APEX 23.2.4). It also shows the date and time (24-05-2024 10:48) and the weather forecast (92°F, Mostly sunny).

Evaluation Procedure	Marks awarded
Query(5)	

Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

**RESULT:**

# TRIGGER

EX\_NO: 18

DATE:

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

QUERY:

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

OUTPUT:

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

```
1 CREATE OR REPLACE FUNCTION factorial(p_num IN NUMBER)
2 RETURN NUMBER IS
3     l_result NUMBER := 1;
4 BEGIN
5     FOR i IN 1..p_num LOOP
6         l_result := l_result * i;
7     END LOOP;
8     RETURN l_result;
9 END factorial;
10 /
11
12
```

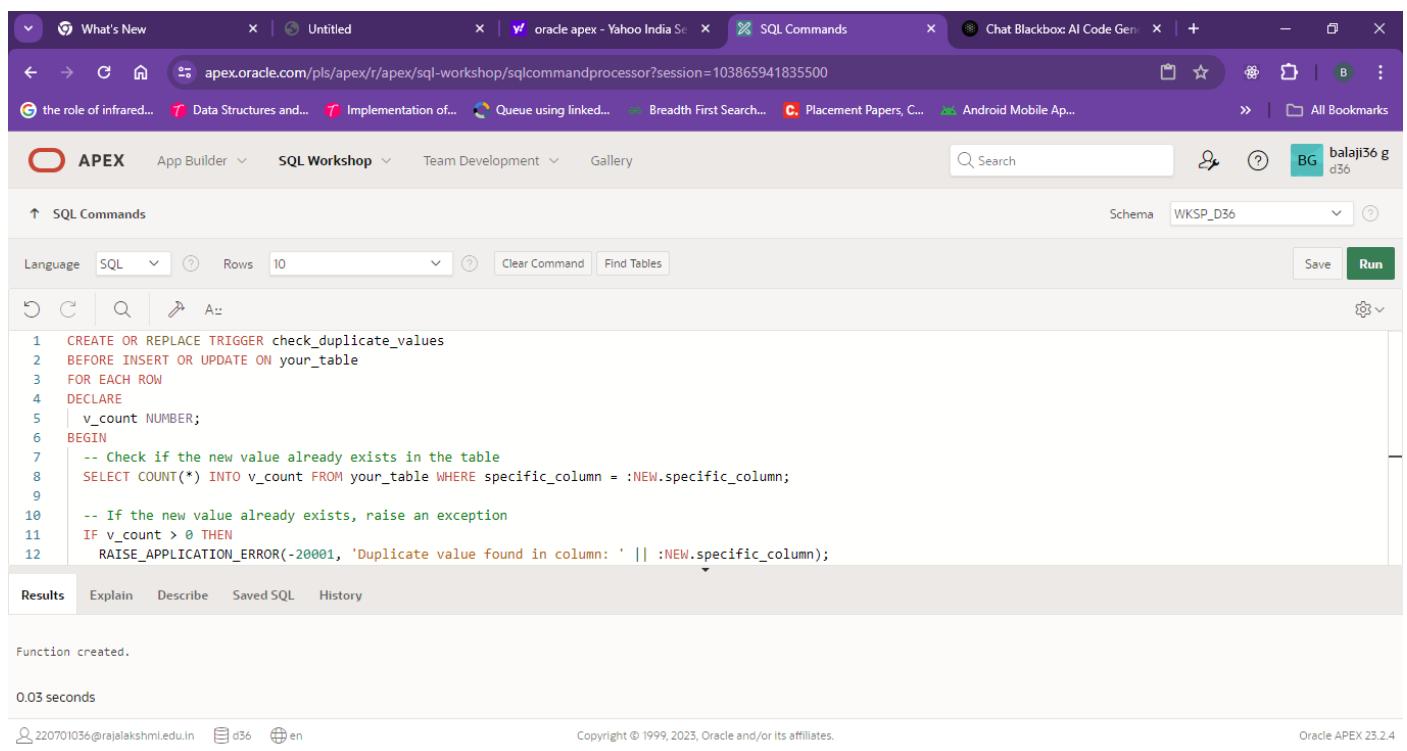
Below the code, the 'Results' tab is active, showing the message "Function created." and "0.02 seconds". At the bottom of the page, the footer includes the URL "220701036@rajalakshmi.edu.in", the session ID "d36", the language "en", copyright information "Copyright © 1999, 2023, Oracle and/or Its affiliates.", and the version "Oracle APEX 23.2.4".

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

### QUERY:

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

### OUTPUT:



The screenshot shows the Oracle Apex SQL Workshop interface. The top navigation bar includes tabs for 'What's New', 'Untitled', 'oracle apex - Yahoo India Se...', 'SQL Commands', and 'Chat Blackbox: AI Code Gen...'. Below the navigation is a toolbar with various icons. The main workspace is titled 'APEX' and 'SQL Workshop'. It features a 'Search' bar and a 'Schema' dropdown set to 'WKSP\_D36'. The code editor contains the PL/SQL trigger definition provided above. The 'Results' tab at the bottom is active, displaying the message 'Function created.' and '0.03 seconds'. The status bar at the bottom right indicates 'Oracle APEX 23.2.4'.

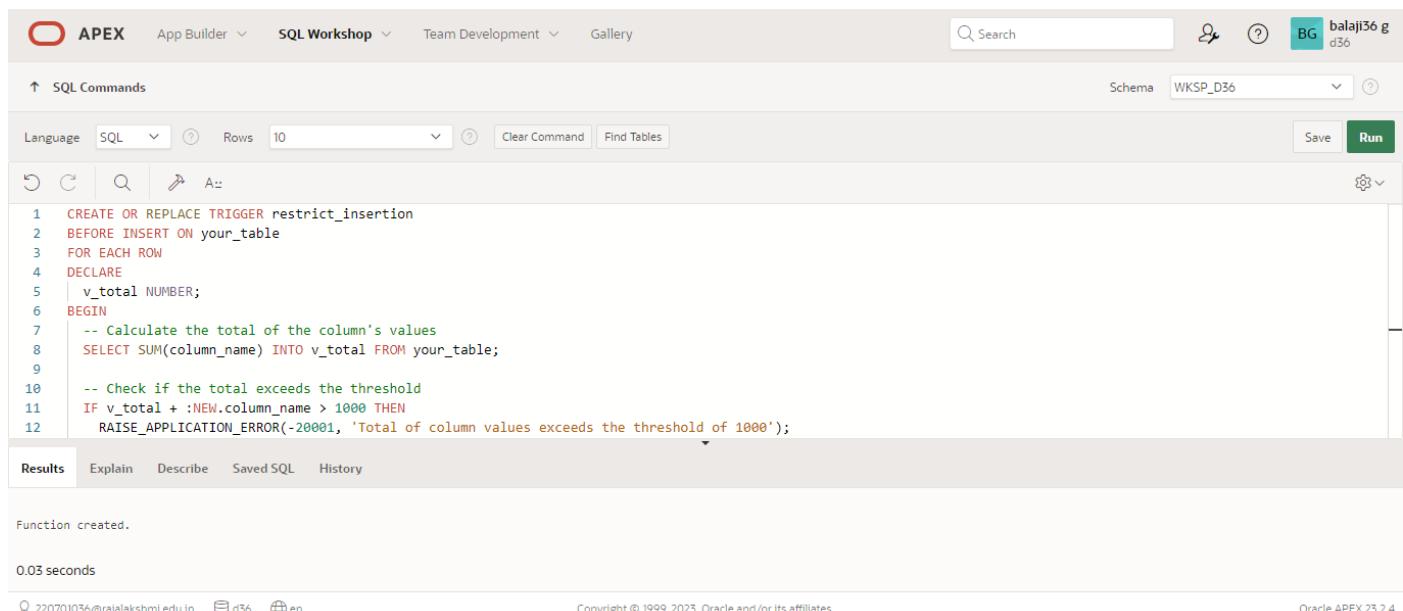
```
1 CREATE OR REPLACE TRIGGER check_duplicate_values
2 BEFORE INSERT OR UPDATE ON your_table
3 FOR EACH ROW
4 DECLARE
5     v_count NUMBER;
6 BEGIN
7     -- Check if the new value already exists in the table
8     SELECT COUNT(*) INTO v_count FROM your_table WHERE specific_column = :NEW.specific_column;
9
10    -- If the new value already exists, raise an exception
11    IF v_count > 0 THEN
12        RAISE_APPLICATION_ERROR(-20001, 'Duplicate value found in column: ' || :NEW.specific_column);
```

**3.) Write a code in PL/SQL to create a trigger that restricts the insertion of new rows if the total of a column's values exceeds a certain threshold**

**QUERY:**

```
CREATE OR REPLACE TRIGGER check_threshold
BEFORE INSERT OR UPDATE ON threshold_table
FOR EACH ROW
DECLARE
    threshold_exceeded EXCEPTION;
    PRAGMA EXCEPTION_INIT(threshold_exceeded, -20003);
    v_sum NUMBER;
    v_threshold NUMBER := 10000; -- Set your threshold here
BEGIN
    SELECT SUM(value_col) INTO v_sum FROM threshold_table;
    v_sum := v_sum + :NEW.value_col;
    IF v_sum > v_threshold THEN
        RAISE threshold_exceeded;
    END IF;
EXCEPTION
    WHEN threshold_exceeded THEN
        RAISE_APPLICATION_ERROR(-20003, 'Threshold exceeded for value_col.');
END;
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. A search bar and a user profile icon are also present. The main workspace is titled "SQL Commands". It features a toolbar with icons for Undo, Redo, Find, Replace, and Run. Below the toolbar, there are dropdown menus for Language (set to SQL), Rows (set to 10), and buttons for Clear Command and Find Tables. The SQL editor contains the PL/SQL code for the trigger. The code defines a trigger named "check\_threshold" that fires before inserting or updating rows in the "threshold\_table". It declares variables "v\_sum" and "v\_threshold" (set to 10000), and handles an exception "threshold\_exceeded" which is raised if the sum of the current row's value and the table's total exceeds the threshold. The "Run" button is highlighted in green at the bottom right of the editor.

```
1 CREATE OR REPLACE TRIGGER restrict_insertion
2 BEFORE INSERT ON your_table
3 FOR EACH ROW
4 DECLARE
5     v_total NUMBER;
6 BEGIN
7     -- Calculate the total of the column's values
8     SELECT SUM(column_name) INTO v_total FROM your_table;
9
10    -- Check if the total exceeds the threshold
11    IF v_total + :NEW.column_name > 1000 THEN
12        RAISE_APPLICATION_ERROR(-20001, 'Total of column values exceeds the threshold of 1000');
```

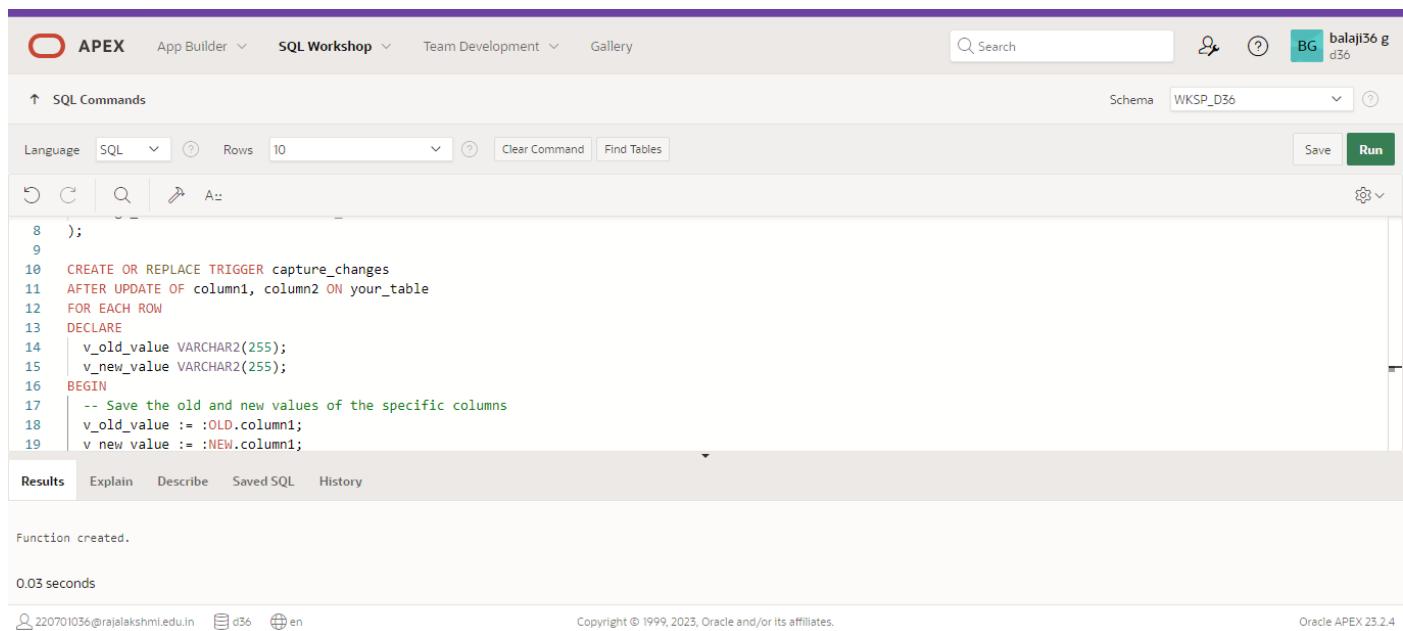
The results tab shows the message "Function created." and a execution time of "0.03 seconds". The bottom footer includes the user information "220701036@rajalakshmi.edu.in d56 en", copyright notice "Copyright © 1999, 2023, Oracle and/or its affiliates.", and the software version "Oracle APEX 23.2.4".

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile 'balaji36 g d36'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_D36'. The code editor contains the PL/SQL code for the 'log\_changes' trigger. The code is as follows:

```
8 );
9
10 CREATE OR REPLACE TRIGGER capture_changes
11 AFTER UPDATE OF column1, column2 ON your_table
12 FOR EACH ROW
13 DECLARE
14     v_old_value VARCHAR2(255);
15     v_new_value VARCHAR2(255);
16 BEGIN
17     -- Save the old and new values of the specific columns
18     v_old_value := :OLD.column1;
19     v_new_value := :NEW.column1;
```

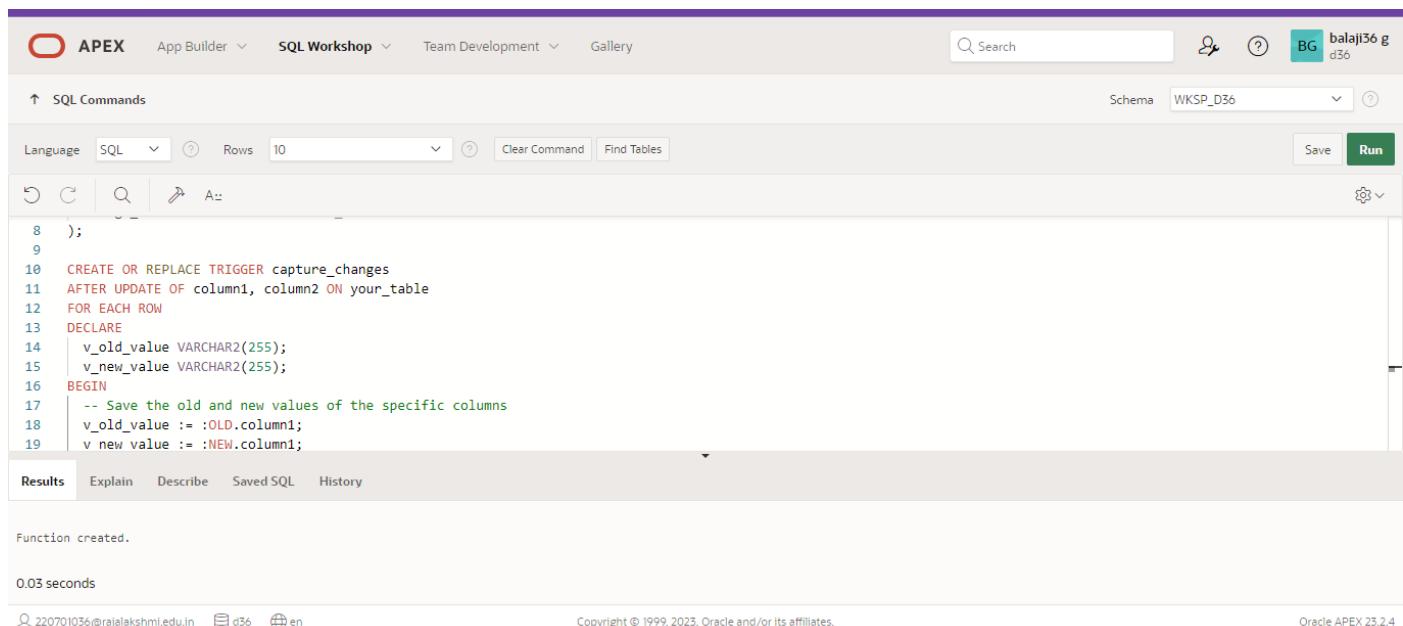
The 'Results' tab is selected at the bottom, showing the message 'Function created.' and a execution time of '0.03 seconds'. The footer includes copyright information for Oracle and the APEX version 'Oracle APEX 23.2.4'.

**5.) Write a code in PL/SQL to implement a trigger that records user activity (inserts, updates, deletes) in an audit log for a given set of tables.**

**QUERY:**

```
CREATE OR REPLACE TRIGGER log_user_activity
AFTER INSERT OR UPDATE OR DELETE ON activity_table
FOR EACH ROW
BEGIN
  IF INSERTING THEN
    INSERT INTO user_activity_log (log_id, action, table_name, record_id, change_time)
    VALUES (activity_log_seq.NEXTVAL, 'INSERT', 'activity_table', :NEW.id, SYSTIMESTAMP);
  ELSIF UPDATING THEN
    INSERT INTO user_activity_log (log_id, action, table_name, record_id, change_time)
    VALUES (activity_log_seq.NEXTVAL, 'UPDATE', 'activity_table', :NEW.id,
SYSTIMESTAMP);
  ELSIF DELETING THEN
    INSERT INTO user_activity_log (log_id, action, table_name, record_id, change_time)
    VALUES (activity_log_seq.NEXTVAL, 'DELETE', 'activity_table', :OLD.id, SYSTIMESTAMP);
  END IF;
END;
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, a help icon, and a user profile for 'balaji36 g d36'. The main workspace is titled 'SQL Commands' and shows the schema 'WKSP\_D36'. The code area contains the PL/SQL trigger definition provided in the previous section. The bottom status bar indicates the command was run in 0.03 seconds and shows the user's email and the Oracle APEX version.

```
8  );
9
10 CREATE OR REPLACE TRIGGER capture_changes
11 AFTER UPDATE OF column1, column2 ON your_table
12 FOR EACH ROW
13 DECLARE
14   v_old_value VARCHAR2(255);
15   v_new_value VARCHAR2(255);
16 BEGIN
17   -- Save the old and new values of the specific columns
18   v_old_value := :OLD.column1;
19   v new value := :NEW.column1;
```

Results Explain Describe Saved SQL History

Function created.

0.03 seconds

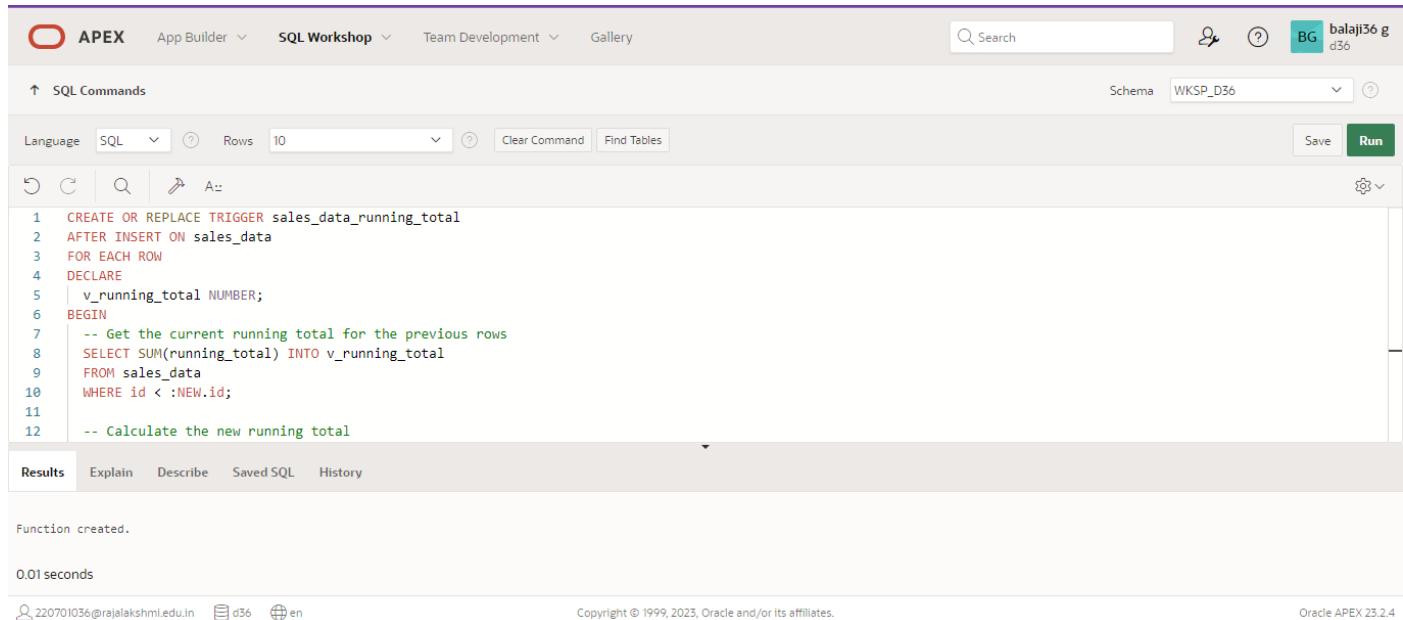
220701036@rajalakshmi.edu.in d36 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

**6.) Write a code in PL/SQL to implement a trigger that automatically calculates and updates a running total column for a table whenever new rows are inserted**

**QUERY:**

```
CREATE OR REPLACE TRIGGER update_running_total
BEFORE INSERT ON running_total_table
FOR EACH ROW
DECLARE
    v_total NUMBER;
BEGIN
    SELECT NVL(SUM(amount), 0) INTO v_total FROM running_total_table;
    :NEW.running_total := v_total + :NEW.amount;
END;
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side of the header shows a user profile for 'balaji36 g d36'. Below the header, the main area is titled 'SQL Commands' with a 'Language' dropdown set to 'SQL'. There are buttons for 'Rows' (set to 10), 'Clear Command', 'Find Tables', 'Save', and 'Run'. The SQL editor contains the code for the trigger 'sales\_data\_running\_total'. The code is as follows:

```
1 CREATE OR REPLACE TRIGGER sales_data_running_total
2 AFTER INSERT ON sales_data
3 FOR EACH ROW
4 DECLARE
5     v_running_total NUMBER;
6 BEGIN
7     -- Get the current running total for the previous rows
8     SELECT SUM(running_total) INTO v_running_total
9     FROM sales_data
10    WHERE id < :NEW.id;
11
12    -- Calculate the new running total

```

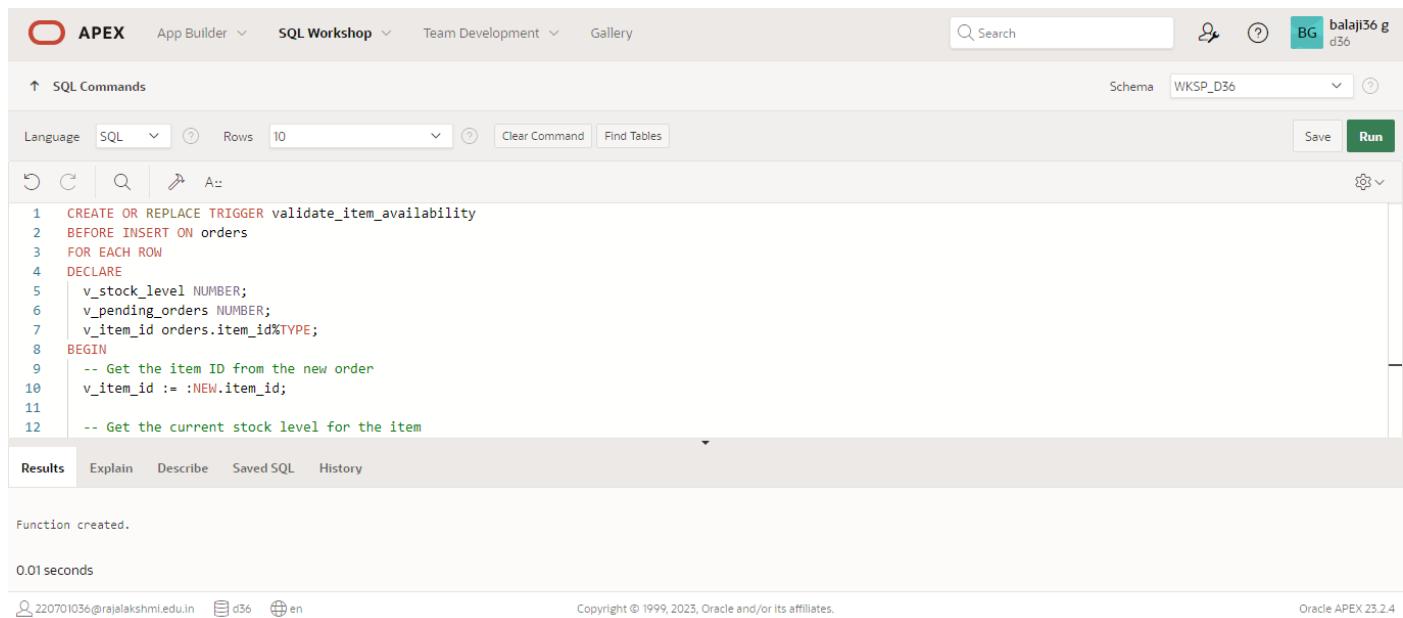
Below the SQL editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected. The output section displays the message 'Function created.' and '0.01 seconds'. At the bottom, it shows the user's email '220701036@rajalakshmi.edu.in', the session ID 'a56', and the language 'en'. The footer also includes the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 23.2.4'.

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile 'balaji36 g d36'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_D36'. The SQL editor contains the PL/SQL code for the 'validate\_order' trigger. The code is highlighted in green and red, indicating syntax and comments. The bottom section shows the results of the command execution.

```
1 CREATE OR REPLACE TRIGGER validate_order
2 BEFORE INSERT ON orders
3 FOR EACH ROW
4 DECLARE
5     v_stock_level NUMBER;
6     v_pending_orders NUMBER;
7     v_item_id orders.item_id%TYPE;
8 BEGIN
9     -- Get the item ID from the new order
10    v_item_id := :NEW.item_id;
11
12    -- Get the current stock level for the item
```

Results

Function created.

0.01 seconds

220701036@rajalakshmi.edu.in d36 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

**RESULT:**

# MONGO DB

**EX\_NO: 19**

**DATE:**

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the myCompiler interface. In the top bar, there are buttons for English, Recent, Login, and Sign up. Below the bar, there is a search input field labeled "Enter a title...". Underneath it, there are two tabs: "MongoDB" (selected) and "SQL". On the right side, there are "Run" and "Save" buttons. The main area contains the MongoDB query:

```
1 db.restaurants.find({  
2   $or: [  
3     { cuisine: { $not: { $in: ["American", "Chinese"] } } },  
4     { name: /^wil/ }  
5   ],  
6 }, {  
7   restaurant_id: 1,  
8   name: 1,  
9   borough: 1,  
10  cuisine: 1  
11});
```

To the right of the code editor is the "Output" panel, which displays the command prompt and the results of the execution:

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

At the bottom of the output panel, there is a small advertisement for FigJam AI.

2.) Write a MongoDB query to find the restaurant Id, name, and grades for those restaurants which achieved a grade of "A" and scored 11 on an ISODate "2014-08-11T00:00:00Z" among many of survey dates.

**QUERY:**

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

**OUTPUT:**

The screenshot shows the myCompiler interface. In the top bar, there are buttons for English, Recent, Login, and Sign up. Below the bar, there is a search input field labeled "Enter a title...". Underneath it, there are two tabs: "MongoDB" (selected) and "SQL". On the right side, there are "Run" and "Save" buttons. The main area contains the MongoDB query:

```
1 db.restaurants.find({  
2   "grades": {  
3     $elemMatch: {  
4       "date": ISODate("2014-08-11T00:00:00Z"),  
5       "grade": "A",  
6       "score": 11  
7     }  
8   },  
9 }, {  
10  "_id": 0,  
11  "restaurant_id": 1,  
12  "name": 1,  
13  "grades": 1  
14});
```

To the right of the code editor is the "Output" panel, which displays the command prompt and the results of the execution:

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

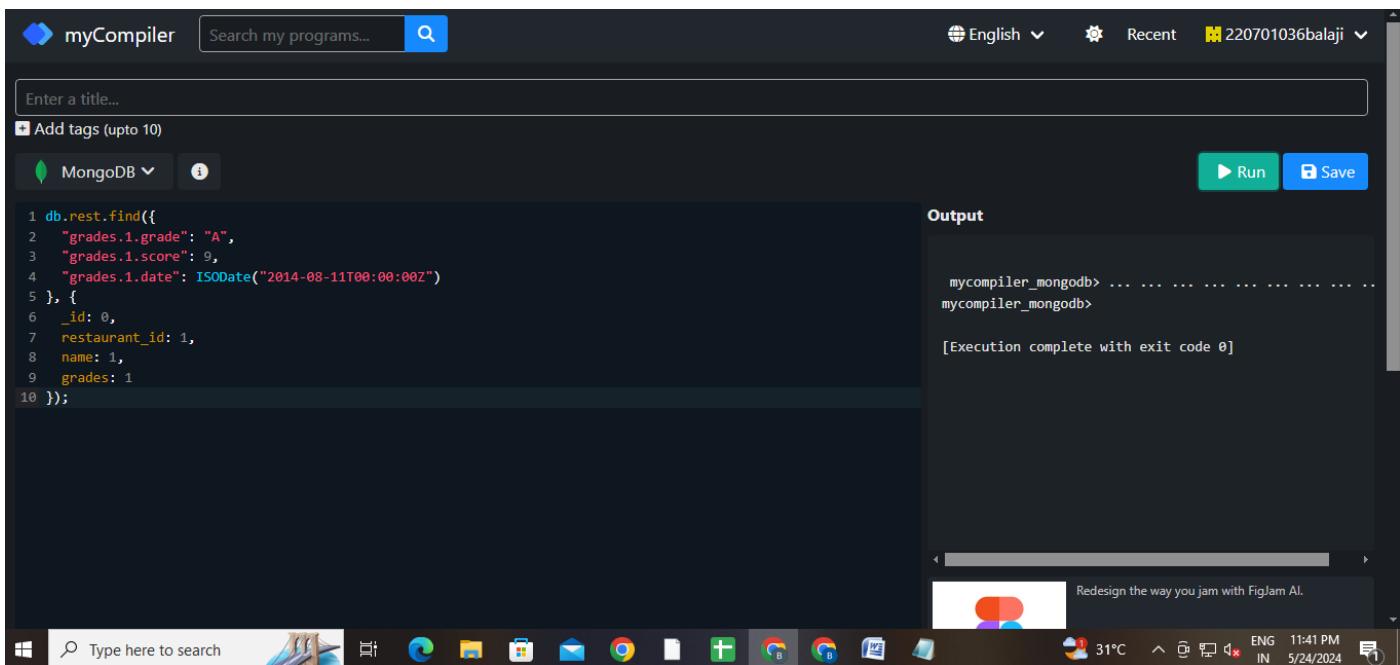
At the bottom of the output panel, there is a small advertisement for FigJam AI.

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("2014-08-1T00:00:00Z") }, { restaurant_id: 1, name: 1, grades: 1 } );
```

## OUTPUT:



The screenshot shows the myCompiler application interface. In the top bar, there's a search bar for programs and a language selection dropdown set to English. On the right, there are buttons for Recent projects and a user ID (220701036balaji). Below the header, there's a text input field for titles and a button to add tags (up to 10). A dropdown menu shows 'MongoDB' is selected. On the left, the code editor contains the following MongoDB query:

```
1 db.restaurants.find({  
2   "grades.1.grade": "A",  
3   "grades.1.score": 9,  
4   "grades.1.date": ISODate("2014-08-11T00:00:00Z")  
5 }, {  
6   _id: 0,  
7   restaurant_id: 1,  
8   name: 1,  
9   grades: 1  
10});
```

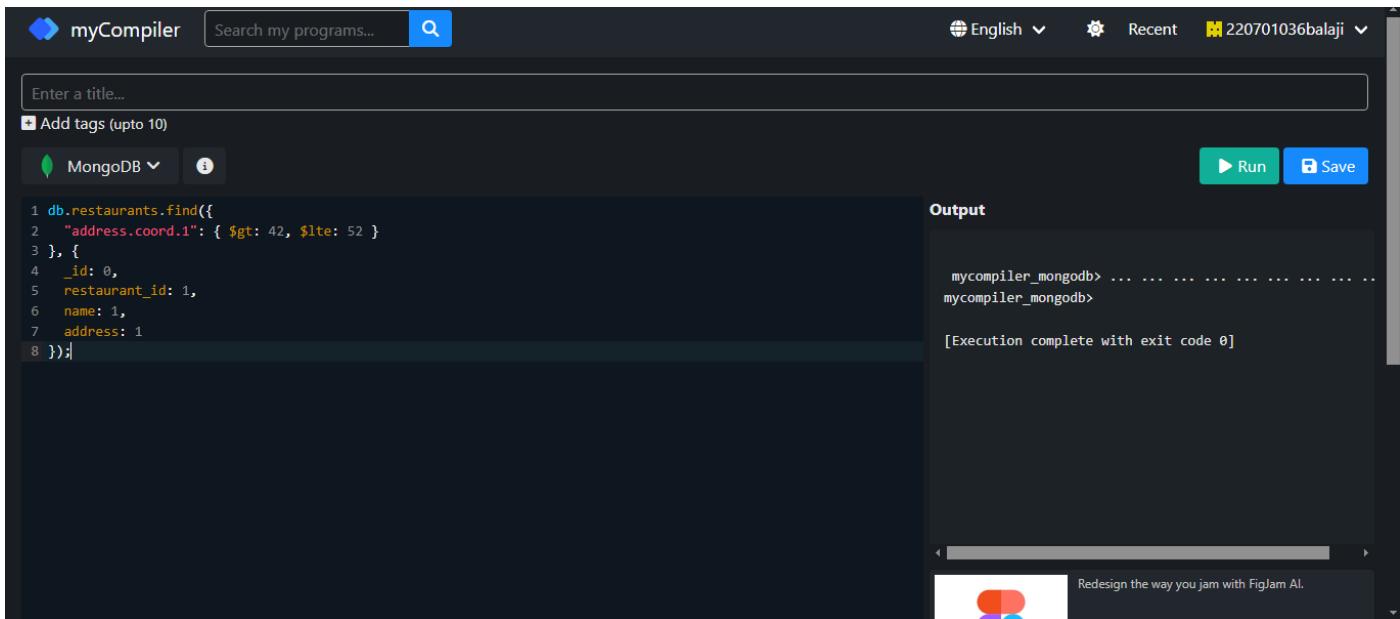
On the right, the 'Output' panel shows the command being run and the completion message: [Execution complete with exit code 0]. At the bottom of the window, there's a toolbar with various icons and a status bar showing the date and time (5/24/2024, 11:41 PM).

4.) Write a MongoDB query to find the restaurant Id, name, address and geographical location for those restaurants where 2nd element of coord array contains a value which is more than 42 and upto 52

## QUERY:

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

## OUTPUT:



The screenshot shows the myCompiler application interface. In the top bar, there's a search bar for programs and a language selection dropdown set to English. On the right, there are buttons for Recent projects and a user ID (220701036balaji). Below the header, there's a text input field for titles and a button to add tags (up to 10). A dropdown menu shows 'MongoDB' is selected. On the left, the code editor contains the following MongoDB query:

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

On the right, the 'Output' panel shows the command being run and the completion message: [Execution complete with exit code 0]. At the bottom of the window, there's a toolbar with various icons and a status bar showing the date and time (5/24/2024, 11:41 PM).

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

## QUERY:

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

## OUTPUT:

The screenshot shows the myCompiler application interface. In the top bar, there is a search bar for programs and a language selection dropdown set to English. On the right, there are buttons for Recent projects and a user profile. Below the header, there is a toolbar with a MongoDB icon, a help icon, and buttons for Run and Save. A text input field contains the MongoDB query: `1 db.restaurants.find().sort({name: 1})`. To the right, under the heading "Output", the terminal window displays the command prompt "mycompiler\_mongodb> ...". The output shows the command being run and the message "[Execution complete with exit code 0]". At the bottom of the interface, there is a watermark for FigJam AI.

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

## QUERY:

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

## OUTPUT:

The screenshot shows the myCompiler application interface. In the top bar, there is a search bar for programs and a language selection dropdown set to English. On the right, there are buttons for Recent projects and a user profile. Below the header, there is a toolbar with a MongoDB icon, a help icon, and buttons for Run and Save. A text input field contains the MongoDB query: `1 db.restaurants.find().sort({name: -1})`. To the right, under the heading "Output", the terminal window displays the command prompt "mycompiler\_mongodb> ...". The output shows the command being run and the message "[Execution complete with exit code 0]". At the bottom of the interface, there is a watermark for FigJam AI.

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:

The screenshot shows the myCompiler MongoDB interface. In the code editor, a query is written: `1 db.restaurants.find({}, { _id:0, cuisine:1, borough:1}).sort({cuisine: 1, borough: -1})`. The output panel shows the results of the query execution.

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

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:

The screenshot shows the myCompiler MongoDB interface. In the code editor, a query is written: `1 db.restaurants.find({ "address.street": { $exists: true } }).count() == db.restaurants.count()`. The output panel shows the results of the query execution.

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

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

#### QUERY:

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

#### OUTPUT:

The screenshot shows the myCompiler interface. In the top left, there's a logo and the text "myCompiler". A search bar says "Search my programs..." with a magnifying glass icon. On the right, there are language selection ("English"), recent projects ("Recent"), and user information ("220701036balaji"). Below the search bar is a text input field with placeholder "Enter a title...". Underneath it is a button "Add tags (upto 10)" with a plus sign icon. The main area has tabs for "MongoDB" (selected) and "SQL". A code editor window contains the following MongoDB query:

```
1 db.restaurants.find({ "address.coord.0": { $type: "double" }, "address.coord.1": { $type: "double" } })
```

To the right is an "Output" panel with the text:

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

10. Write a MongoDB query which will select the restaurant Id, name and grades for those restaurants which returns 0 as a remainder after dividing the score by 7.

#### QUERY:

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

#### OUTPUT:

The screenshot shows the myCompiler interface. The setup is identical to the previous one, with the "MongoDB" tab selected. The code editor window contains the following MongoDB query:

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

The "Output" panel shows the results:

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

11. Write a MongoDB query to find the restaurant name, borough, longitude and attitude and cuisine for those restaurants which contains 'mon' as three letters somewhere in its name.

#### QUERY:

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

#### OUTPUT:

The screenshot shows the myCompiler interface. In the top left, there's a logo and the text "myCompiler". A search bar says "Search my programs..." with a magnifying glass icon. On the right, there are language selection ("English"), recent projects ("Recent"), and user information ("220701036balaji"). Below the search bar is a text input field with placeholder "Enter a title...". Underneath it is a button "Add tags (upto 10)". On the left, there's a "MongoDB" dropdown and an "Info" button. On the right, there are "Run" and "Save" buttons. The main area contains a code editor with the following MongoDB query:

```
1 db.restaurants.find({ name: { $regex: /mon/i } }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 })
```

To the right of the code editor is an "Output" panel. It shows the command "mycompiler\_mongodb> ..." followed by "[Execution complete with exit code 0]". At the bottom of the output panel is a watermark: "Redesign the way you jam with FigJam AI.".

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

#### QUERY:

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

#### OUTPUT:

The screenshot shows the myCompiler interface. The layout is identical to the previous one, with the "MongoDB" dropdown and "Info" button on the left, and "Run" and "Save" buttons on the right. The code editor contains the same MongoDB query as the previous screenshot:

```
1 db.restaurants.find({ name: { $regex: /^Mad/i } }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 })
```

The "Output" panel shows the command "mycompiler\_mongodb> ..." followed by "[Execution complete with exit code 0]".

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

#### QUERY:

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

#### OUTPUT:

The screenshot shows the myCompiler interface. In the top left, there's a logo and the text "myCompiler". A search bar with the placeholder "Search my programs..." and a magnifying glass icon is next to it. On the right, there are language selection ("English"), recent projects ("Recent"), and user information ("220701036balaji"). Below the search bar is a text input field with placeholder "Enter a title...". Underneath it is a checkbox labeled "Add tags (upto 10)". To the right of the input fields are two buttons: "Run" (green) and "Save" (blue). The main area contains a code editor with the following MongoDB query:

```
1 db.restaurants.find({ "grades.score": { $lt: 5 } }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 })
```

To the right of the code editor is a "Output" panel. It shows the command "mycompiler\_mongodb> ..." followed by the results of the query. The results are empty, indicated by "[Execution complete with exit code 0]".

14. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan.

#### QUERY:

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

#### OUTPUT:

The screenshot shows the myCompiler interface. The setup is identical to the previous one: logo, search bar, language selection, and user info. The code editor contains the same MongoDB query as the previous screenshot:

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

The "Output" panel shows the command "mycompiler\_mongodb> ..." followed by the results of the query. The results are empty, indicated by "[Execution complete with exit code 0]".

15. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn.

#### QUERY:

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

#### OUTPUT:

The screenshot shows the myCompiler interface. At the top, there's a search bar for programs and a language selector set to English. Below the search bar is a title input field with placeholder text "Enter a title..." and a "Add tags (upto 10)" button. On the left, there's a MongoDB dropdown and a help icon. On the right, there are "Run" and "Save" buttons. The main area contains a code editor with the following MongoDB query:

```
1, {borough: "Brooklyn"}], "grades.score": { $lt: 5 } }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 }) Output
```

Below the code editor is a terminal window showing the command "mycompiler\_mongodb> ...". The output from the MongoDB shell shows the results of the query. The message "[Execution complete with exit code 0]" is displayed at the bottom of the terminal.

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

### QUERY:

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

### OUTPUT:

The screenshot shows the myCompiler interface. The setup is identical to the previous query, with the MongoDB dropdown selected and the "Run" button visible. The code editor contains the same MongoDB query as the previous screenshot.

The terminal window shows the command "mycompiler\_mongodb> ...". The output from the MongoDB shell shows the results of the query. The message "[Execution complete with exit code 0]" is displayed at the bottom of the terminal.

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

### QUERY:

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

### OUTPUT:

The screenshot shows the myCompiler MongoDB interface. In the top left, there's a logo and the text "myCompiler". A search bar with placeholder text "Search my programs..." and a magnifying glass icon is next to it. On the right, there are language selection ("English"), recent projects ("Recent"), and user information ("220701036balaji"). Below the search bar is a text input field with placeholder "Enter a title...". Underneath it is a button "Add tags (upto 10)" with a plus sign icon. The main area has tabs for "MongoDB" (selected) and "i". On the far right are "Run" and "Save" buttons. The code input field contains the following MongoDB query:

```
1$lt: 5 }, cuisine: { $nin: ["American", "Chinese"] } }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 })
```

The "Output" section shows the results of the query execution:

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

18. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6.

### QUERY:

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

### OUTPUT:

The screenshot shows the myCompiler MongoDB interface. The setup is identical to the previous one, with the "MongoDB" tab selected. The code input field contains the same query as above:

```
1$and: [{ "grades.score": { $in: [2, 6] } }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 })
```

The "Output" section shows the results of the query execution:

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

19. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan.

### QUERY:

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

### OUTPUT:

```
Enter a title...
+ Add tags (upto 10)
MongoDB ▾ i Run Save
1grades.score": { $in: [2, 6] }, borough: "Manhattan" }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 }) Output
```

```
mycompiler_mongodb> ...
mycompiler_mongodb>

[Execution complete with exit code 0]
```



Redesign the way you jam with FigJam AI.  
ADS VIA CARBON

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:

```
Enter a title...
+ Add tags (upto 10)
MongoDB ▾ i Run Save
1rough: "Brooklyn"]}, "grades.score": { $in: [2, 6] } }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 }) Output
```

```
mycompiler_mongodb> ...
mycompiler_mongodb>

[Execution complete with exit code 0]
```



Redesign the way you jam with FigJam AI.  
ADS VIA CARBON

21. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American.

### QUERY:

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

### OUTPUT:

The screenshot shows a MongoDB query editor interface. At the top, there's a search bar labeled "Enter a title..." and a button "Add tags (upto 10)". Below that is a dropdown "MongoDB" and a help icon. On the right are "Run" and "Save" buttons. The main area is titled "Output" and contains the following text:  
mycompiler\_mongodb> ...  
mycompiler\_mongodb>  
[Execution complete with exit code 0]  
At the bottom right, there's an advertisement for FigJam AI.

22. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American or Chinese.

#### QUERY:

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

#### OUTPUT:

The screenshot shows a MongoDB query editor interface, identical to the one above. The "Output" section contains the same text as the previous screenshot:  
mycompiler\_mongodb> ...  
mycompiler\_mongodb>  
[Execution complete with exit code 0]  
At the bottom right, there's an advertisement for FigJam AI.

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:

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

## RESULT:

# MONGO DB

**EX\_NO: 20**

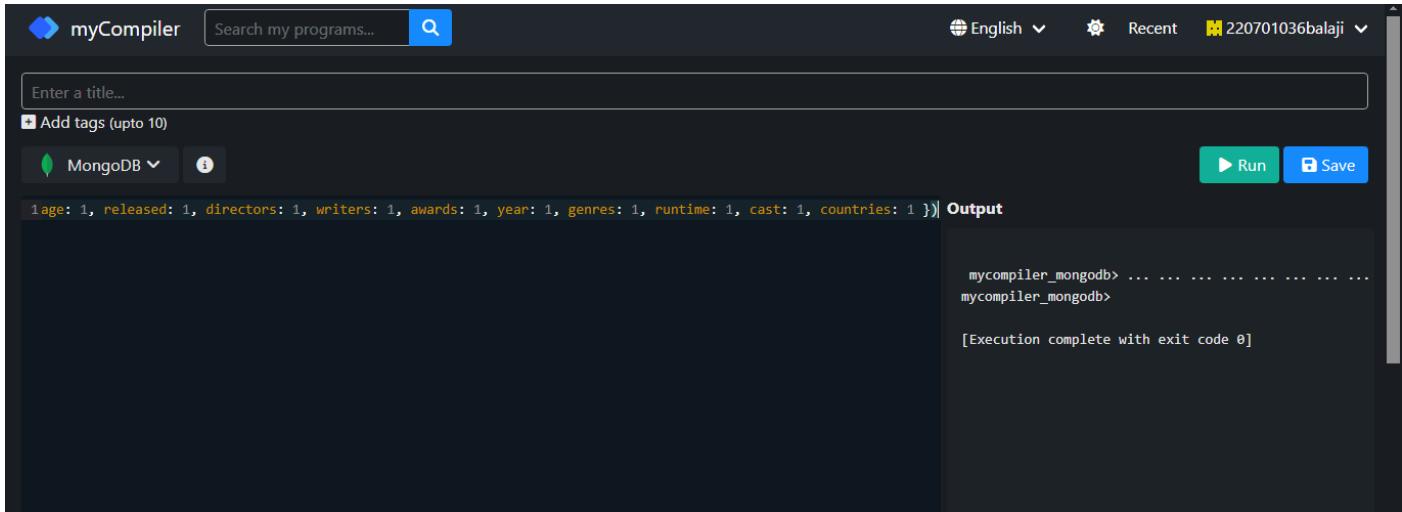
**DATE:**

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

**QUERY:**

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

**OUTPUT:**



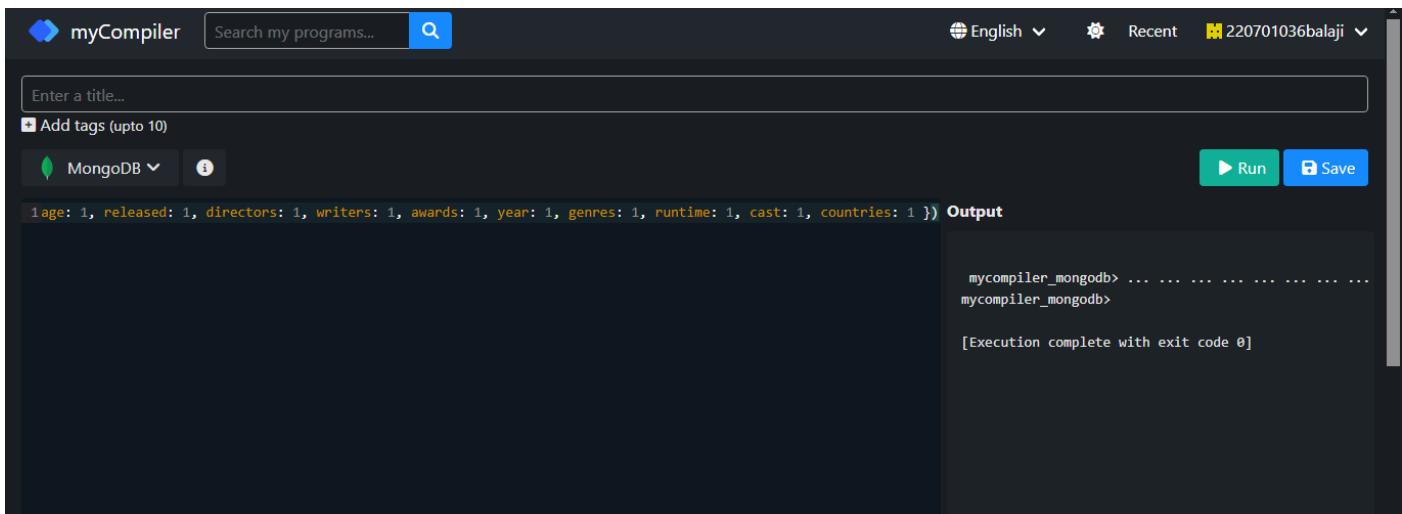
The screenshot shows the myCompiler interface. At the top, there's a search bar for programs and a language dropdown set to English. Below the search bar is a text input field with placeholder "Enter a title...". Underneath it is a checkbox for "Add tags (upto 10)". On the left, there's a MongoDB icon and a dropdown menu. On the right, there are "Run" and "Save" buttons. The main area contains a command-line interface window with the following text:  
`1age: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 }`| Output  
mycompiler\_mongodb> ... . . . . .  
mycompiler\_mongodb>  
[Execution complete with exit code 0]

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

**QUERY:**

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

**OUTPUT:**



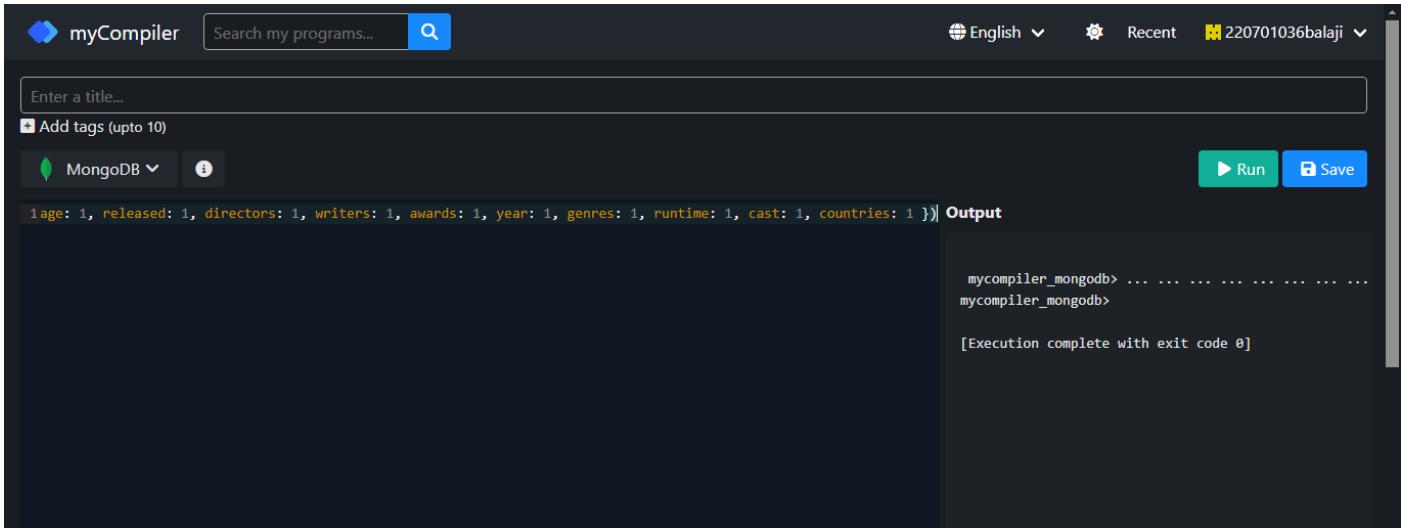
The screenshot shows the myCompiler interface. At the top, there's a search bar for programs and a language dropdown set to English. Below the search bar is a text input field with placeholder "Enter a title...". Underneath it is a checkbox for "Add tags (upto 10)". On the left, there's a MongoDB icon and a dropdown menu. On the right, there are "Run" and "Save" buttons. The main area contains a command-line interface window with the following text:  
`1age: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 }`| Output  
mycompiler\_mongodb> ... . . . . .  
mycompiler\_mongodb>  
[Execution complete with exit code 0]

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

**QUERY:**

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

**OUTPUT:**



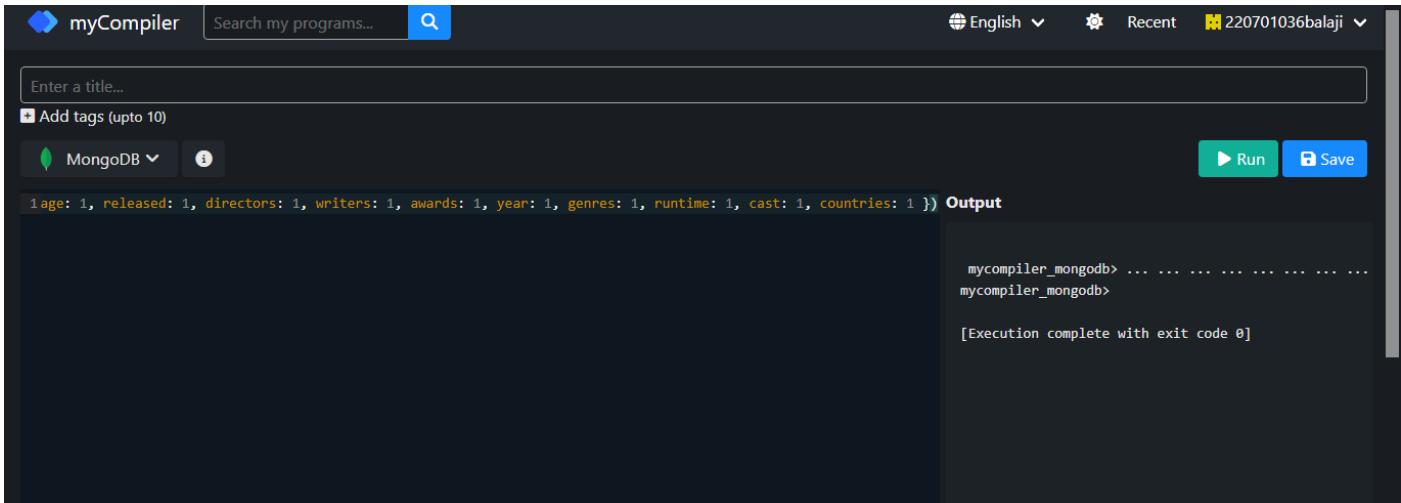
The screenshot shows the myCompiler interface. At the top, there's a search bar for programs and a language selector set to English. Below the search bar is a text input field for titles and a checkbox for adding tags. A MongoDB icon is present, along with a dropdown menu and a help icon. On the right, there are 'Run' and 'Save' buttons. The main area contains a command line interface window titled 'Output'. The command entered is 'db.movies.find({ genres: "Short" })'. The output shows the results of the query, which is empty (indicated by three dots). The command line prompt is 'mycompiler\_mongodb> ...'. Below the command line, a message says '[Execution complete with exit code 0]'. The entire interface has a dark theme.

**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:**



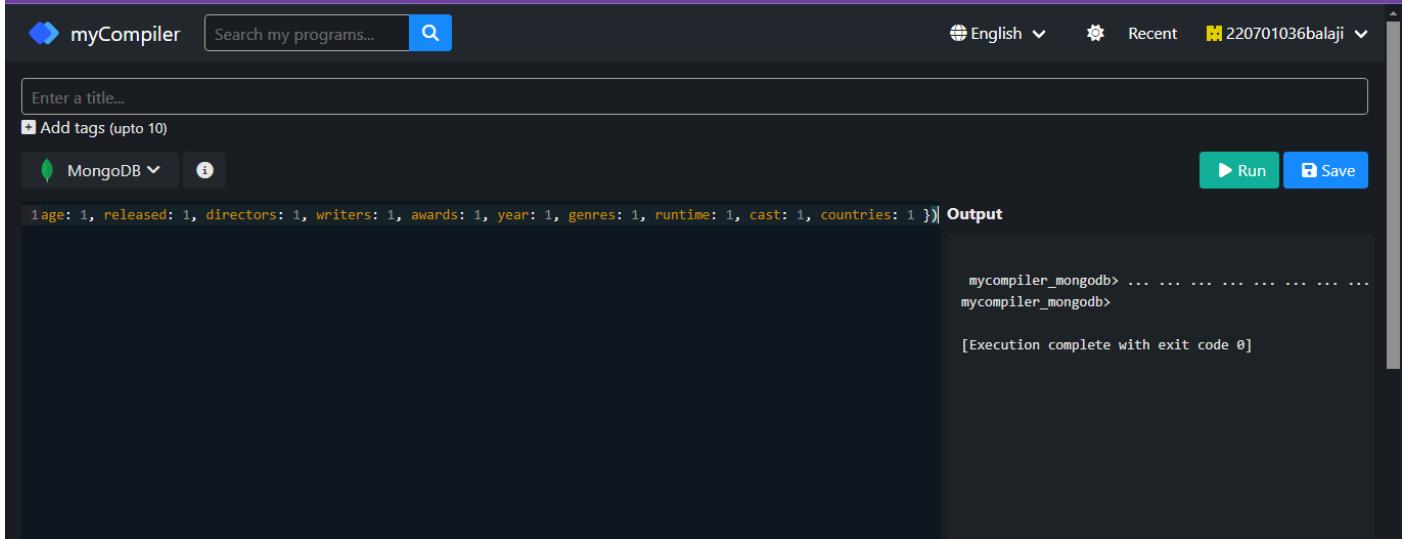
This screenshot is identical to the one above, showing the myCompiler interface. It features the same dark theme, search bar, and MongoDB connection details. The 'Output' window displays the command 'db.movies.find({ directors: "William K.L. Dickson" })' and its result, which is also empty (three dots). The command line prompt is 'mycompiler\_mongodb> ...'. The message '[Execution complete with exit code 0]' is visible at the bottom. The overall layout is consistent with the first screenshot.

**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:**



The screenshot shows the myCompiler interface. At the top, there's a search bar for programs and a language selector set to English. Below the search bar is a text input field labeled "Enter a title...". Underneath it is a checkbox for "Add tags (upto 10)". On the left, there are buttons for MongoDB and a help icon. On the right, there are "Run" and "Save" buttons. The main area contains a command-line interface window with the following text:

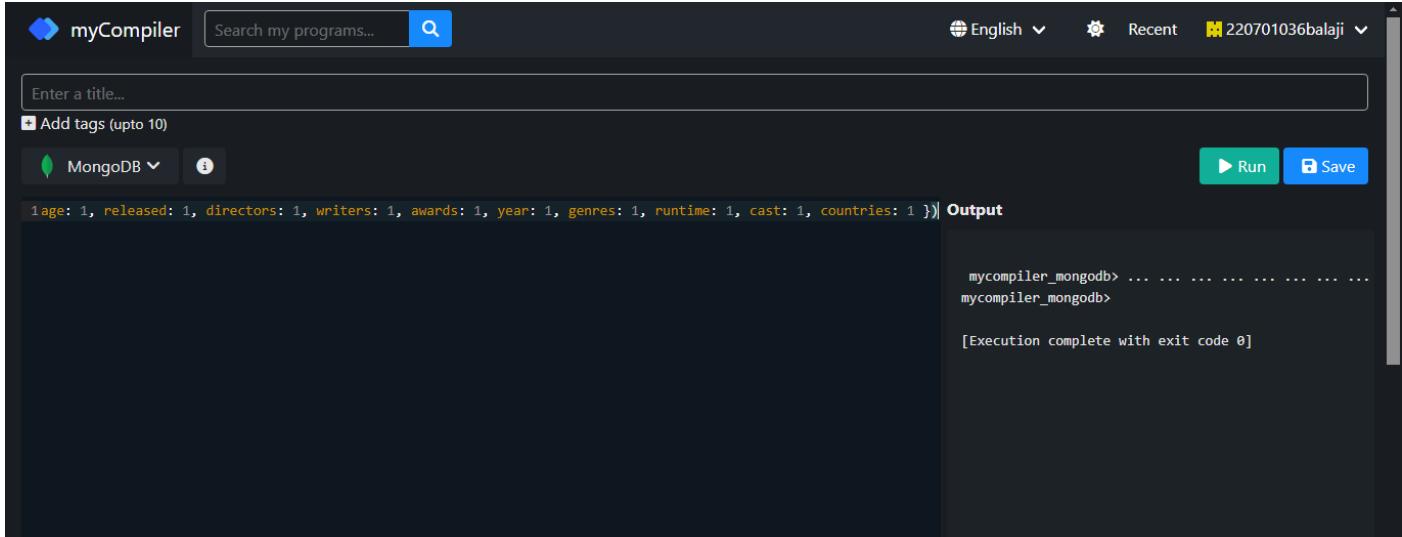
```
1age: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 })| Output  
mycompiler_mongodb> ...  
mycompiler_mongodb>  
[Execution complete with exit code 0]
```

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the myCompiler interface. The setup is identical to the previous one: English language, search bar, title input, tags checkbox, MongoDB button, and Run/Save buttons. The command-line interface window shows the same output as the previous screenshot:

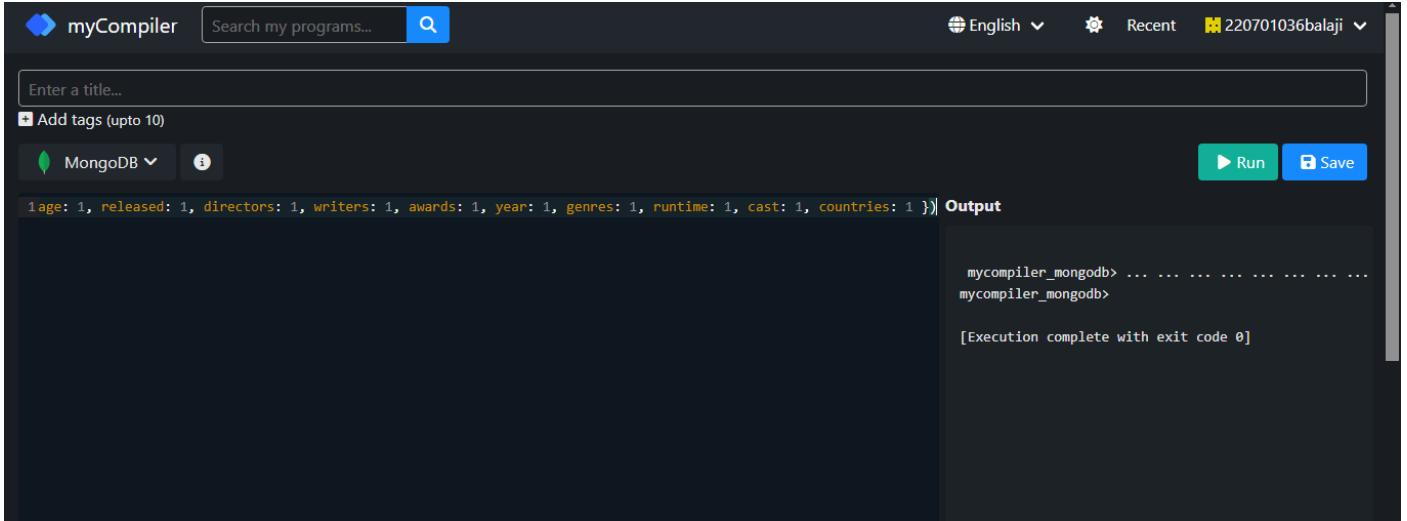
```
1age: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 })| Output  
mycompiler_mongodb> ...  
mycompiler_mongodb>  
[Execution complete with exit code 0]
```

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

**QUERY:**

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

**OUTPUT:**



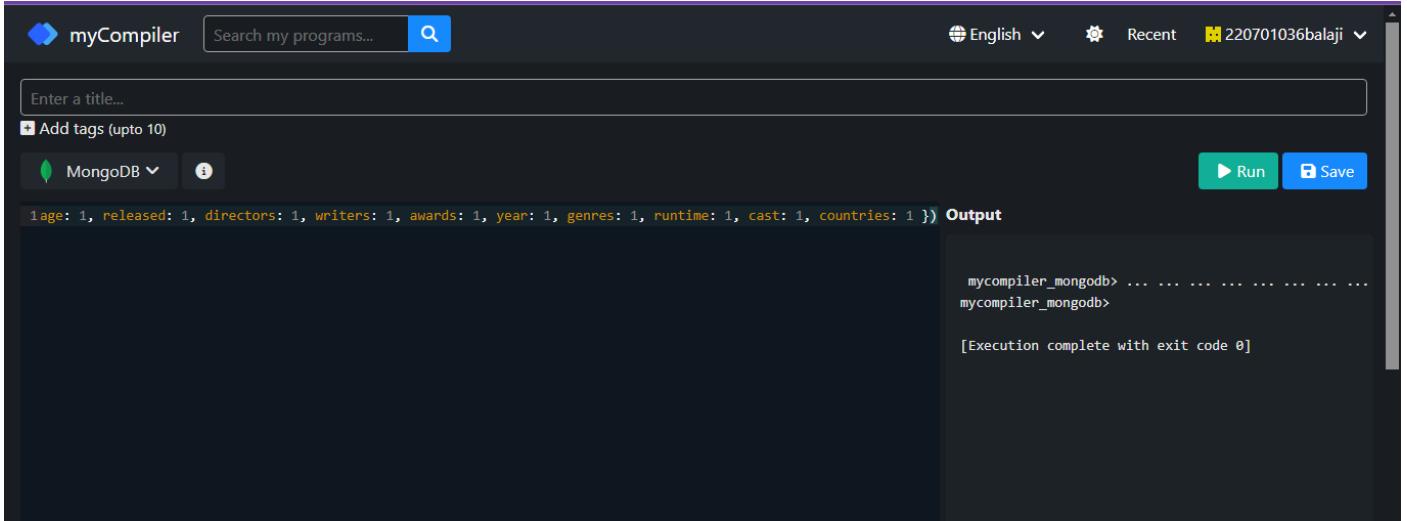
The screenshot shows the myCompiler interface with the MongoDB extension selected. In the top right, there are dropdowns for 'English', 'Recent', and a user ID '220701036balaji'. Below the header is a search bar 'Search my programs...' with a magnifying glass icon. A text input field 'Enter a title...' is followed by a checkbox 'Add tags (upto 10)' and a 'Run' button. On the left, there's a 'MongoDB' dropdown and a help icon. The main area is titled 'Output' and contains the command: 'db.movies.find({ "imdb.votes": { \$gt: 1000 } })'. The output window shows the command being run and completed successfully with exit code 0. The terminal output is: 'mycompiler\_mongodb> ...' and '[Execution complete with exit code 0]'. There is also a 'Save' button in the bottom right of the output panel.

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the myCompiler interface with the MongoDB extension selected. In the top right, there are dropdowns for 'English', 'Recent', and a user ID '220701036balaji'. Below the header is a search bar 'Search my programs...' with a magnifying glass icon. A text input field 'Enter a title...' is followed by a checkbox 'Add tags (upto 10)' and a 'Run' button. On the left, there's a 'MongoDB' dropdown and a help icon. The main area is titled 'Output' and contains the command: 'db.movies.find({ "imdb.rating": { \$gt: 7 } })'. The output window shows the command being run and completed successfully with exit code 0. The terminal output is: 'mycompiler\_mongodb> ...' and '[Execution complete with exit code 0]'. There is also a 'Save' button in the bottom right of the output panel.

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:

The screenshot shows the myCompiler IDE interface. At the top, there's a navigation bar with a logo, the text "myCompiler", a search bar containing "Search my programs...", and a magnifying glass icon. To the right are links for "English", "Recent", and a user profile "220701036balaji". Below the navigation bar is a search bar with placeholder text "Enter a title...". Underneath it is a button labeled "+ Add tags (upto 10)". On the left side, there's a dropdown menu for "MongoDB" with a green leaf icon and a help icon. On the right side, there are two buttons: "Run" (green with white play icon) and "Save" (blue with white save icon). The main workspace contains a command-line interface window. The command entered is "age: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 }" followed by "Output". The output shows the command prompt "mycompiler\_mongodb> " followed by several dots, and then "[Execution complete with exit code 0]".

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

## QUERY:

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

## **OUTPUT:**

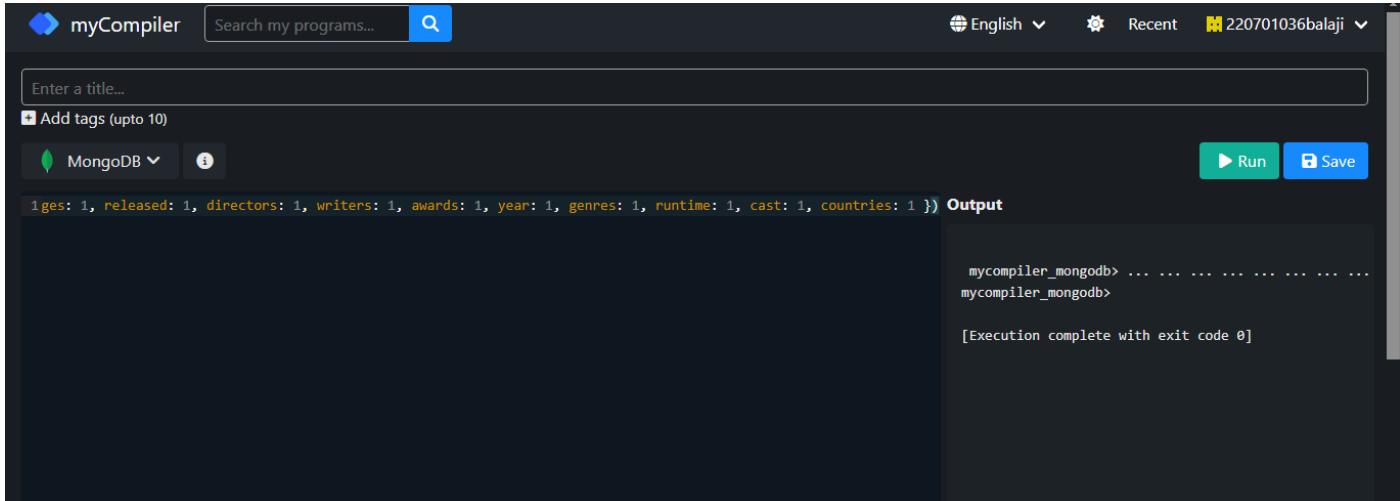
The screenshot shows the myCompiler interface. At the top, there's a search bar with placeholder text "Search my programs..." and a magnifying glass icon. To the right are language selection ("English"), recent projects, and user information ("220701036balaji"). Below the header is a text input field with placeholder "Enter a title...". Underneath it is a button labeled "+ Add tags (upto 10)". On the left, there are two dropdowns: "MongoDB" and "Info". On the right, there are "Run" and "Save" buttons. The main content area displays a MongoDB query result: "1 age: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 }" followed by an "Output" label. To the right, a terminal window shows the command "mycompiler\_mongodb> . . . . . . . . . . . . . . ." and the message "[Execution complete with exit code 0]".

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

**QUERY:**

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

**OUTPUT:**



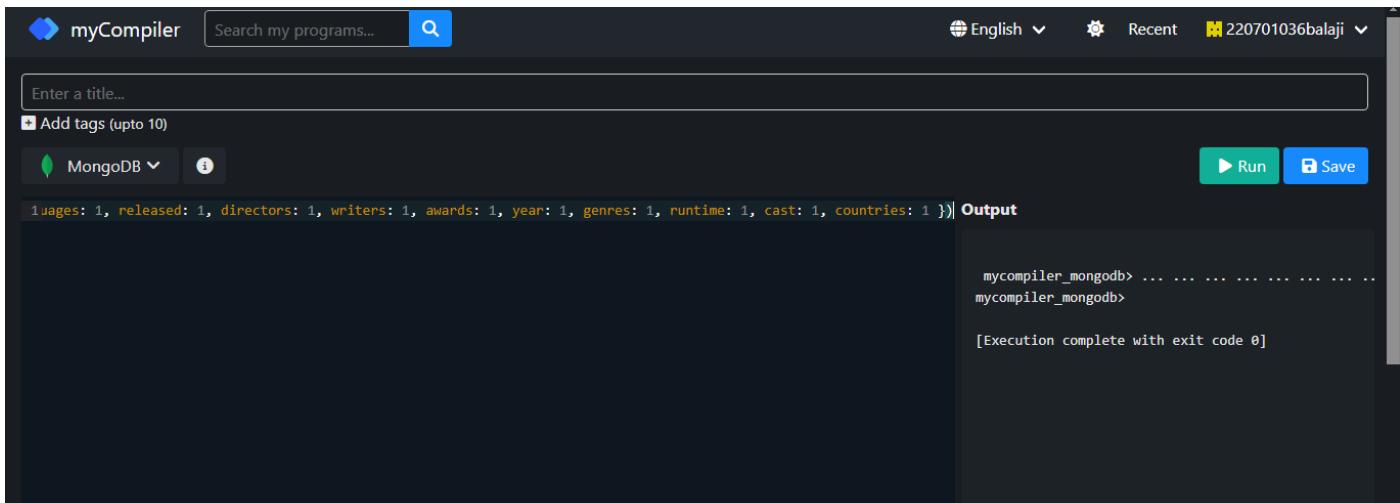
The screenshot shows the myCompiler interface. In the top bar, there are tabs for 'myCompiler', 'Search my programs...', and a magnifying glass icon. On the right, there are language selection ('English'), recent projects ('Recent'), and session ID ('220701036balaji') buttons. Below the top bar, there's a search bar labeled 'Enter a title...' and a checkbox for 'Add tags (upto 10)'. Underneath, there are two buttons: 'MongoDB' with a dropdown arrow and an info icon. To the right are 'Run' and 'Save' buttons. The main area is titled 'Output' and contains the MongoDB command: '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 } )'. The output window shows the command being run and the response: 'mycompiler\_mongodb> ...'. It also indicates '[Execution complete with exit code 0]'. The background of the interface is dark grey.

**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:**



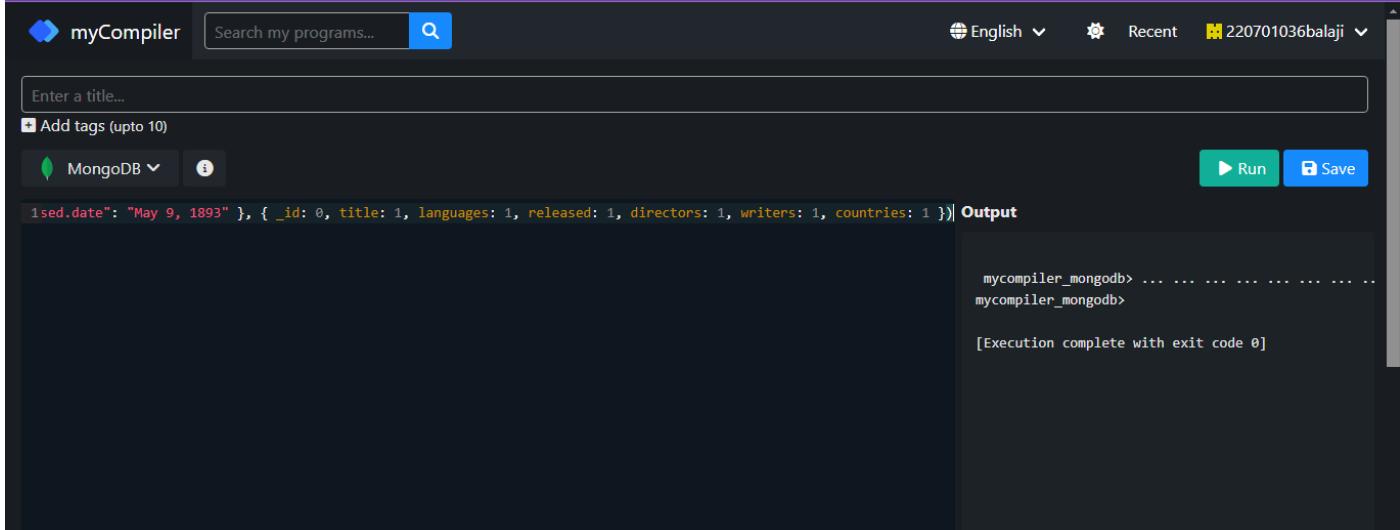
This screenshot is identical to the previous one, showing the myCompiler interface. The top bar, search bar, and session information are the same. The main output window shows the MongoDB command: 'db.movies.find( { "cast": "Charles Kayser" }, { "title": 1, "languages": 1, "released": 1, "directors": 1, "writers": 1, "awards": 1, "year": 1, "genres": 1, "runtime": 1, "cast": 1, "countries": 1 } )'. The output window shows the command being run and the response: 'mycompiler\_mongodb> ...'. It also indicates '[Execution complete with exit code 0]'. The background is dark grey.

**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:**



The screenshot shows the 'myCompiler' interface. In the top bar, there is a search bar 'Search my programs...' and a magnifying glass icon. On the right, there are language selection ('English'), recent projects ('Recent'), and user information ('220701036balaji'). Below the search bar, there is a text input field 'Enter a title...' and a checkbox 'Add tags (upto 10)'. A dropdown menu shows 'MongoDB' selected. On the right side of the interface, there are two buttons: 'Run' (green) and 'Save' (blue). The main area contains a code editor with the following MongoDB query:

```
1sed.date": "May 9, 1893" }, { _id: 0, title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 })| Output
```

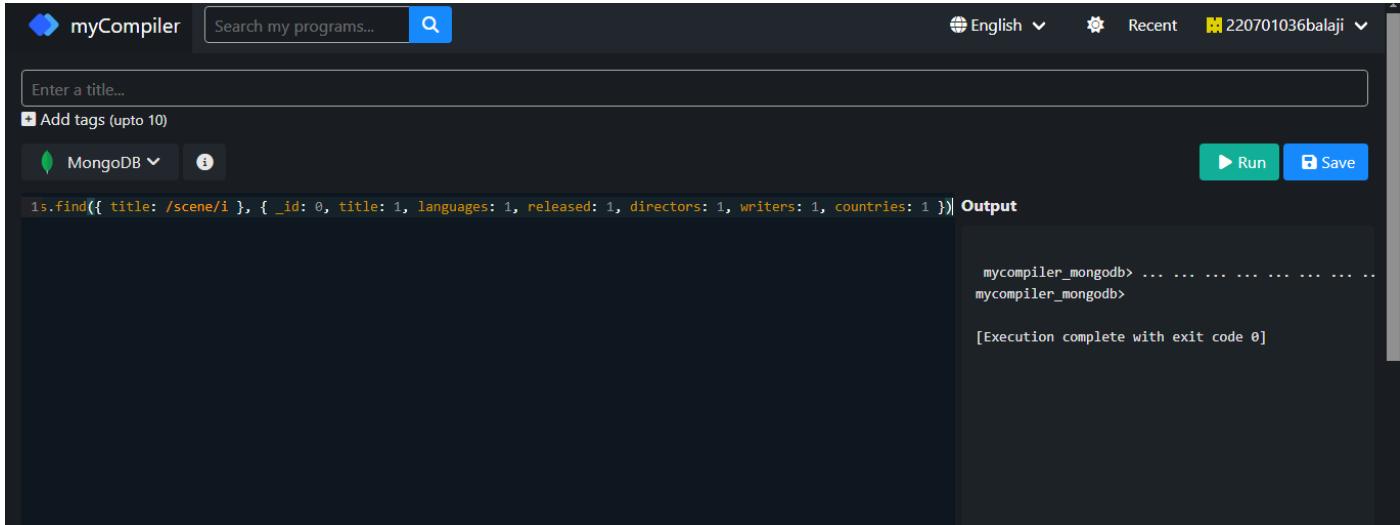
Below the code editor, the output window shows the command prompt 'mycompiler\_mongodb>' followed by several dots indicating continuation. At the bottom of the output window, it says '[Execution complete with exit code 0]'. The entire interface has a dark theme.

**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:**



The screenshot shows the 'myCompiler' interface. The layout is identical to the previous screenshot, with a search bar, language selection, and user information at the top. The code editor contains the same MongoDB query as the previous screenshot:

```
15.find({ title: /scene/i }, { _id: 0, title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 })| Output
```

The output window shows the command prompt 'mycompiler\_mongodb>' followed by several dots. At the bottom, it says '[Execution complete with exit code 0]'. The interface maintains its dark theme.

Evaluation Procedure	Marks awarded

Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

**RESULT:**