

EXNO:1**CREATING AND MANAGING TABLE****DATE:**

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

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

QUERY: Create table DEPARTMENT(id number(7),name varchar2(25));

OUTPUT:

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

```
1 create table dept(id number(7),name varchar2(25));
```

After running the command, the results show:

Table created.
0.03 seconds

At the bottom, the footer includes the URL <https://220701020@rajalakshmi.edu.in>, the schema [dbms20](#), and the copyright notice "Copyright © 1999, 2023, Oracle and/or its affiliates".

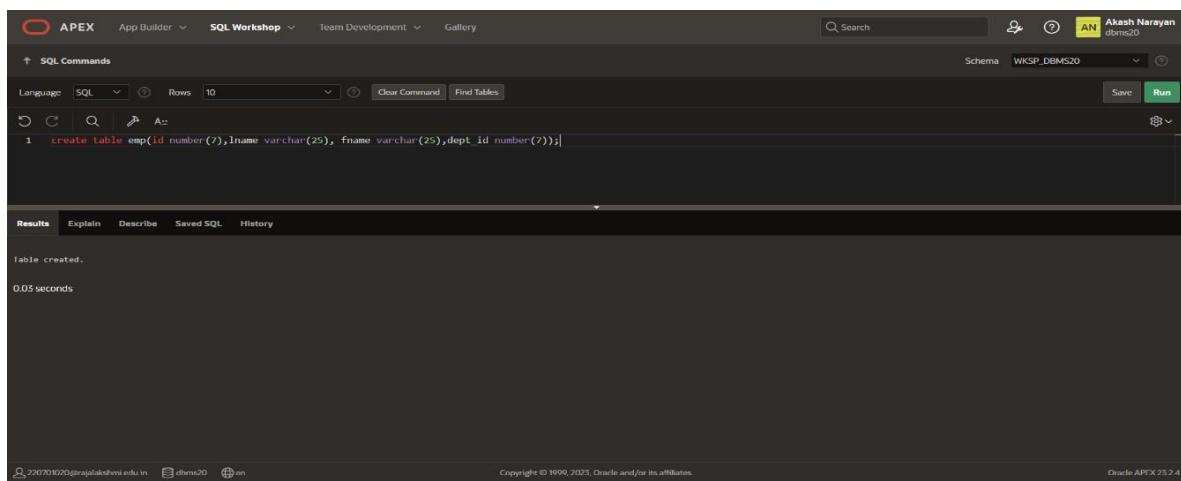
2. Create the EMP table based on the following instance chart. Confirm that the table is created.

Column name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				

Nulls/Unique				
FK table				
FK column				
Data Type	Number	Varchar2	Varchar2	Number
Length	7	25	25	7

QUERY: Create table emp(id number(7),Last_Name varchar(25),First_Name varchar(25),Dept_id number(7));

OUTPUT:



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

```
1 create table emp(id number(7),lname varchar(25), fname varchar(25),dept_id number(7));
```

The Results tab shows the output:

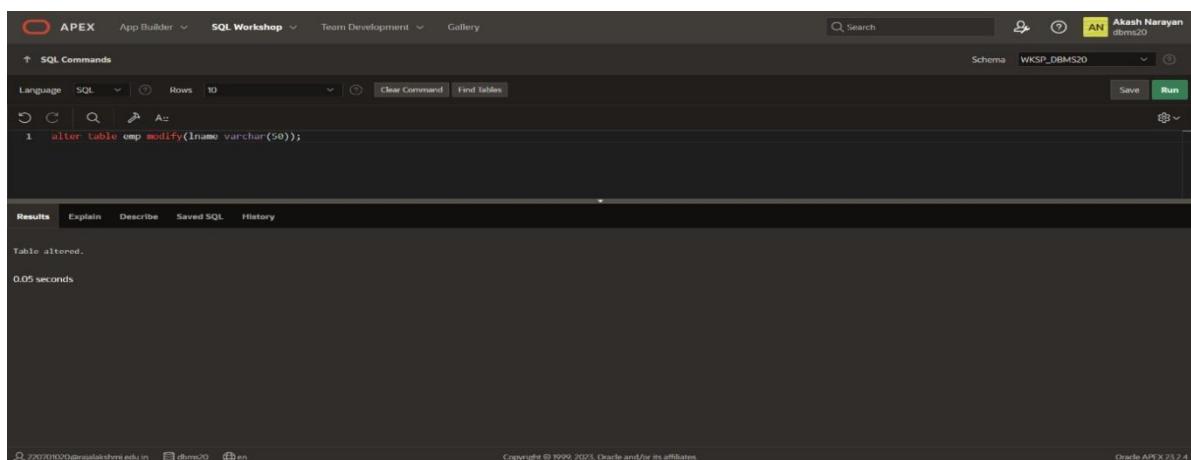
```
Table created.
0.05 seconds
```

At the bottom, the URL is 220701020@rajalakshmi.edu.in, the schema is WKSP_DBMS20, and the version is Oracle APEX 23.2.4.

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

QUERY: Alter table emp modify(Last_Name varchar2(50));

OUTPUT:



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

```
1 alter table emp modify(lname varchar2(50));
```

The Results tab shows the output:

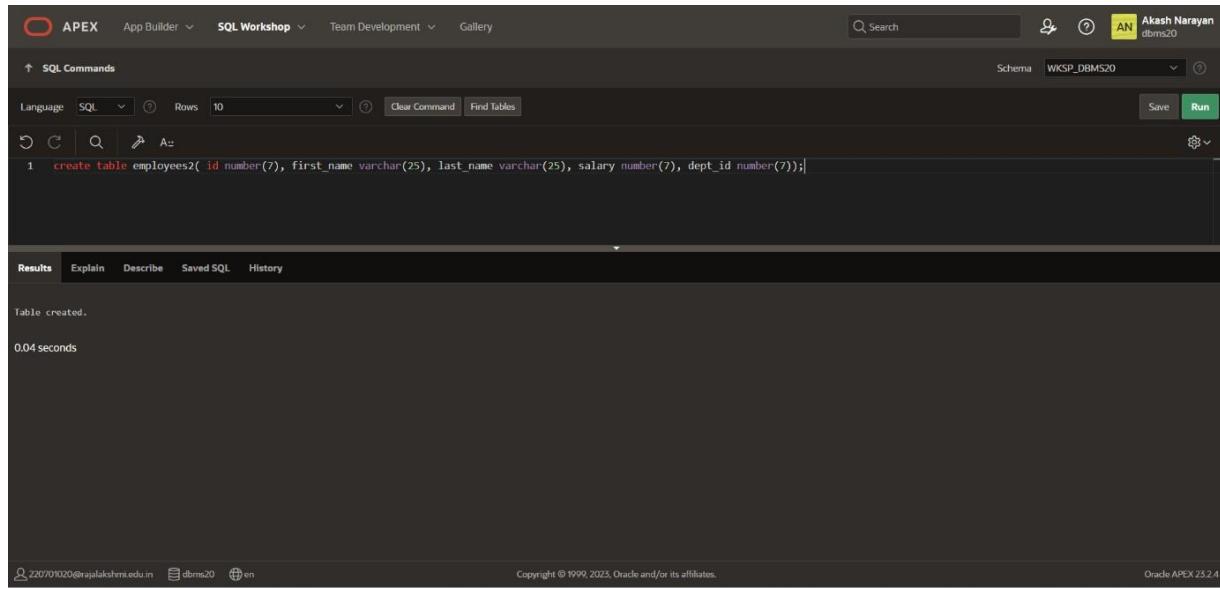
```
Table altered.
0.05 seconds
```

At the bottom, the URL is 220701020@rajalakshmi.edu.in, the schema is WKSP_DBMS20, and the version is Oracle APEX 23.2.4.

**4.Create the EMPLOYEES2 table based on the structure of EMPLOYEES table.
Include Only the Employee_id, First_name, Last_name, Salary and Dept_id coloumns.
Name the columns Id, First_name, Last_name, salary and Dept_id respectively.**

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

OUTPUT:

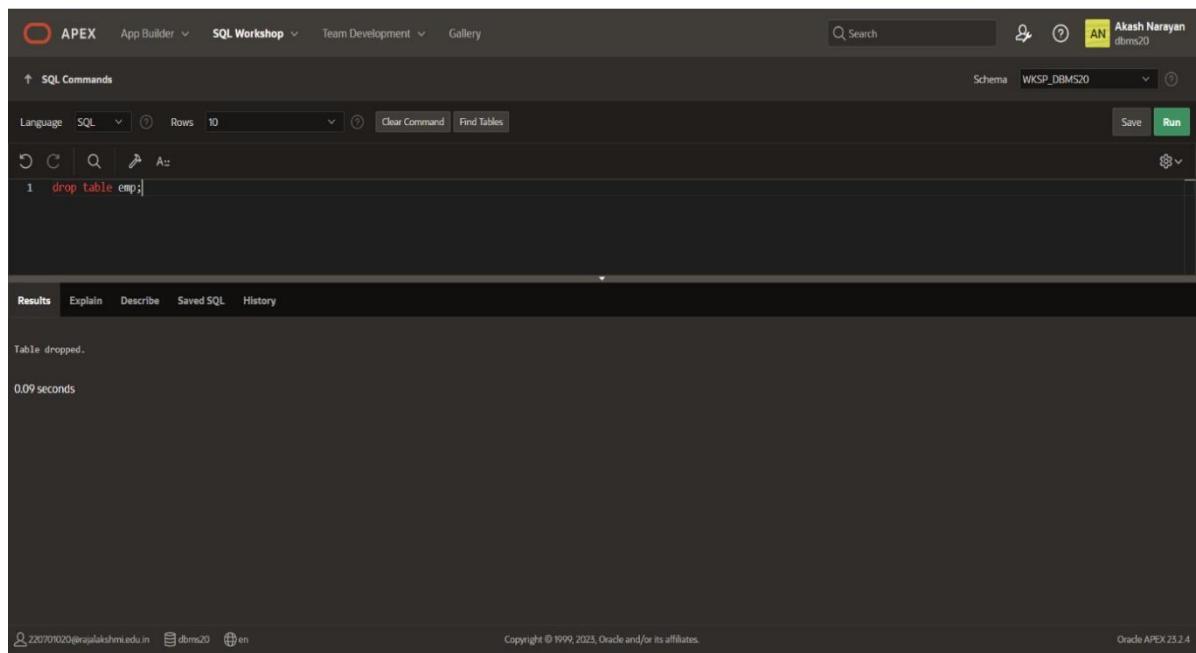


The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is active. The main area is titled 'SQL Commands'. A single line of SQL code is entered: 'create table employees2(id number(7), first_name varchar2(25), last_name varchar2(25), salary number(7), dept_id number(7));'. The 'Run' button is highlighted in green at the bottom right. The results section below shows the output: 'Table created.' and '0.04 seconds'. The bottom right corner of the interface displays 'Oracle APEX 23.2.4'.

5. Drop table emp.

QUERY: Drop table emp;

OUTPUT:

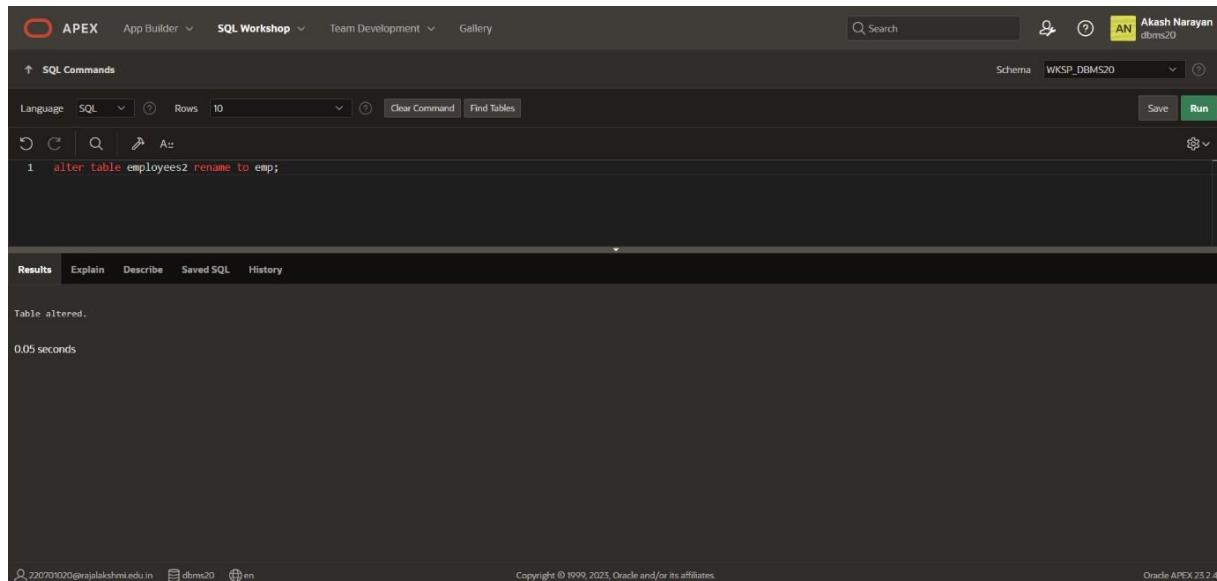


The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is active. The main area is titled 'SQL Commands'. A single line of SQL code is entered: 'drop table emp;'. The 'Run' button is highlighted in green at the bottom right. The results section below shows the output: 'Table dropped.' and '0.09 seconds'. The bottom right corner of the interface displays 'Oracle APEX 23.2.4'.

6.Rename the EMPLOYEES2 table as EMP.

QUERY: alter table employees2 rename to emp;

OUTPUT:

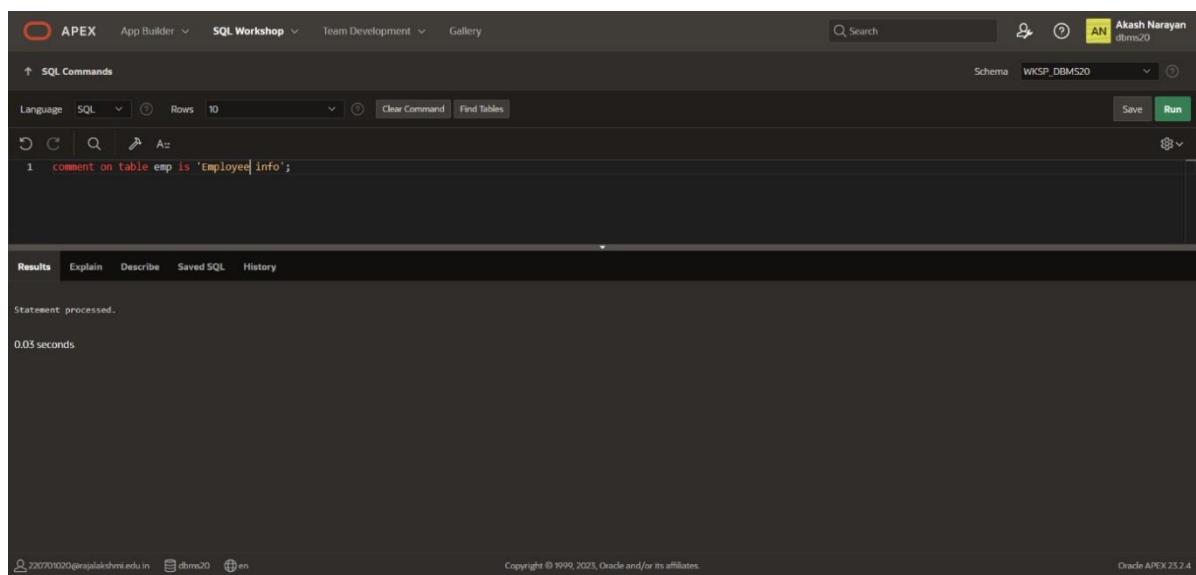


The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows the user 'Akash Narayan' and the schema 'WKSP_DBMS20'. The main area is titled 'SQL Commands' with a language dropdown set to 'SQL'. The command entered is 'alter table employees2 rename to emp;'. Below the command, the results show 'Table altered.' and a duration of '0.05 seconds'. The bottom footer includes copyright information for Oracle and the APEX version 'Oracle APEX 23.2.4'.

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

QUERY: comment on table dept is 'Employee info';

OUTPUT:

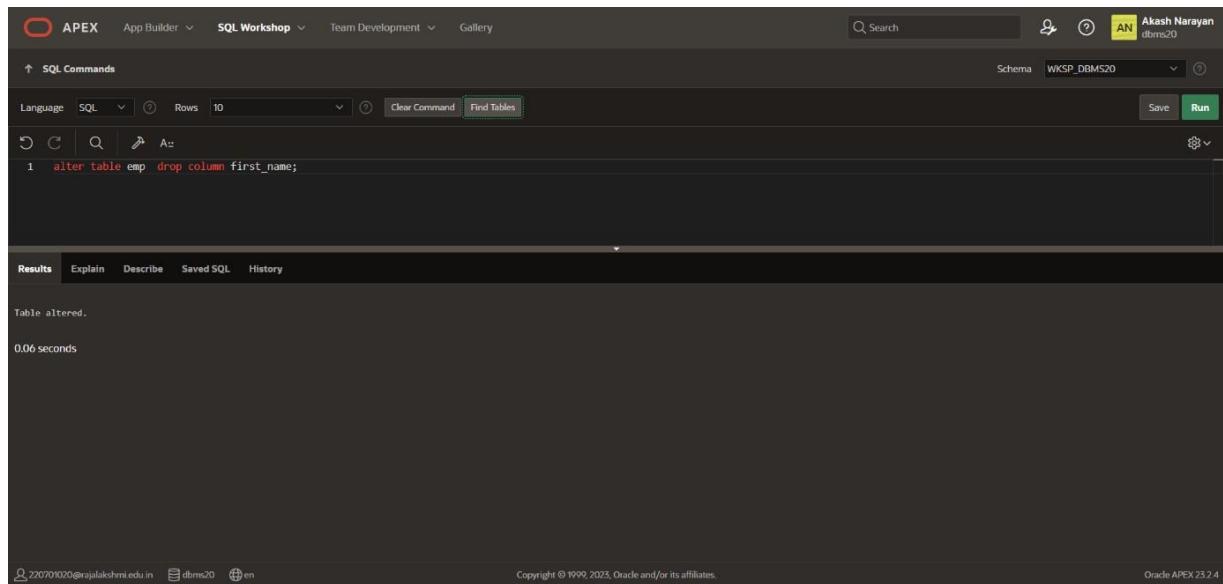


The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows the user 'Akash Narayan' and the schema 'WKSP_DBMS20'. The main area is titled 'SQL Commands' with a language dropdown set to 'SQL'. The command entered is 'comment on table emp is 'Employee info'';. Below the command, the results show 'Statement processed.' and a duration of '0.03 seconds'. The bottom footer includes copyright information for Oracle and the APEX version 'Oracle APEX 23.2.4'.

8.Drop the First_name column from the EMP table and confirm it.

QUERY: Alter table emp drop column first_name;

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, the command `alter table emp drop column first_name;` is entered. The Results tab displays the output: `Table altered.` and `0.06 seconds`. The top right corner shows the user `Akash Narayan` and the schema `WKSP_DBMS20`.

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

RESULT:

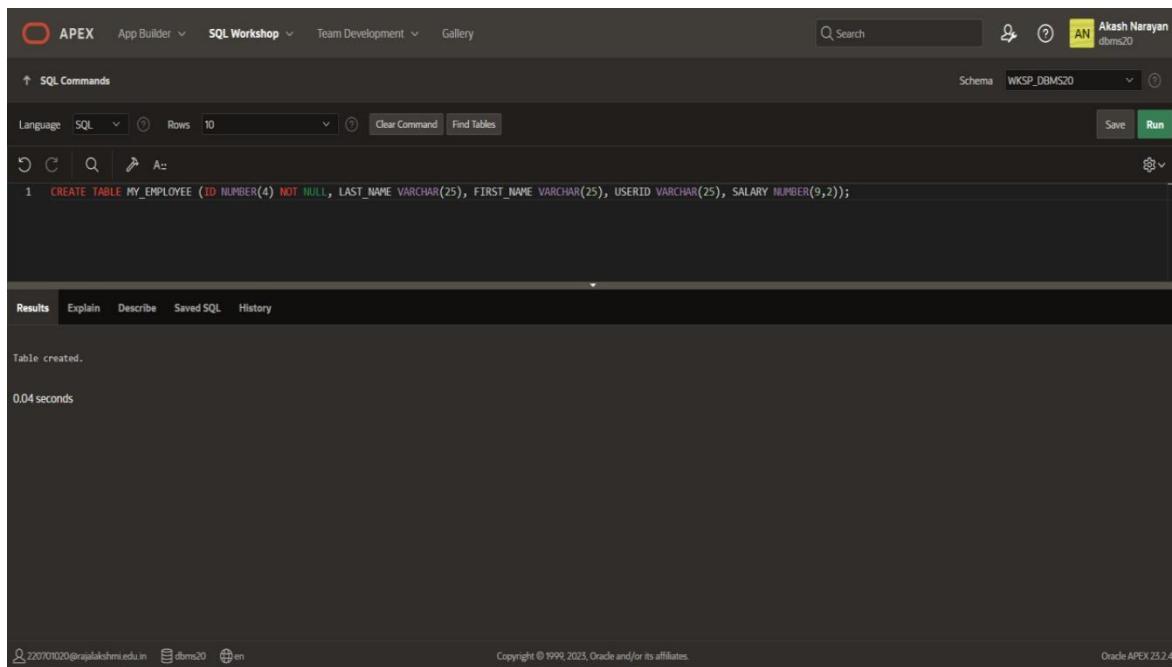
EXNO:2**MANIPULATING DATA****DATE:**

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) not null, 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 SQL Commands tab, a single line of SQL code is entered to create the MY_EMPLOYEE table with the specified structure. The table is successfully created, as indicated by the message "Table created." in the results area. The execution time was 0.04 seconds.

```
CREATE TABLE MY_EMPLOYEE (ID NUMBER(4) NOT NULL, LAST_NAME VARCHAR(25), FIRST_NAME VARCHAR(25), USERID VARCHAR(25), SALARY NUMBER(9,2));
```

Results Explain Describe Saved SQL History

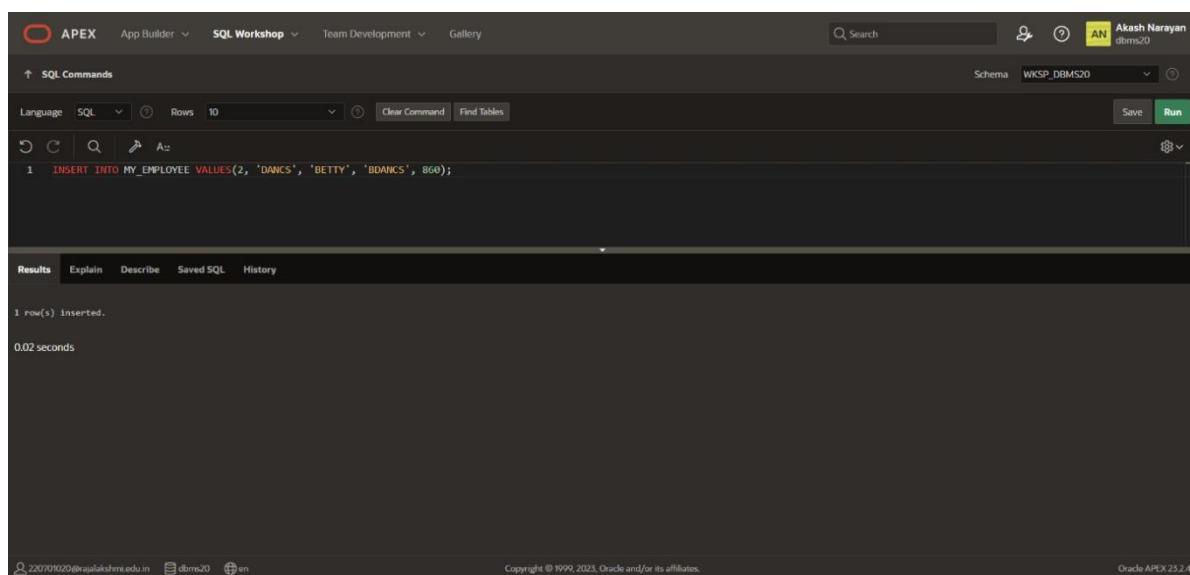
Table created.
0.04 seconds

2.Add the first and second rows data to MY_EMPLOYEE table from the following sample data.

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

QUERY: insert into my_employee(1,'patel','ralph','rpatel',895); insert into my_employee(2,'dancs','betty','bdancs',860);

OUTPUT:



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

```
1 INSERT INTO MY_EMPLOYEE VALUES(2, 'DANCS', 'BETTY', 'BDANCS', 860);
```

Below the command, the Results tab displays the output:

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

Execution time is listed as 0.02 seconds. The bottom of the screen shows standard Oracle APEX footer information.

3.Display the table with values.

QUERY: select * from My_Employee;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The top right shows the user 'Akash Narayan' and the schema 'WKSP_DBMS20'. The SQL command input field contains: 'SELECT * FROM MY_EMPLOYEE;'. The results tab is selected, displaying a table with two rows:

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
2	DANCS	BETTY	BDANCS	860
1	PATEL	RALPH	RPATEL	895

Below the table, it says '2 rows returned in 0.02 seconds' and has a 'Download' link. The bottom footer includes the URL '220701020@rajalakshmi.edu.in', the schema 'dbms20', and the language 'en'. It also states 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

QUERY: update My_employee set lname="drexler" where id=3;

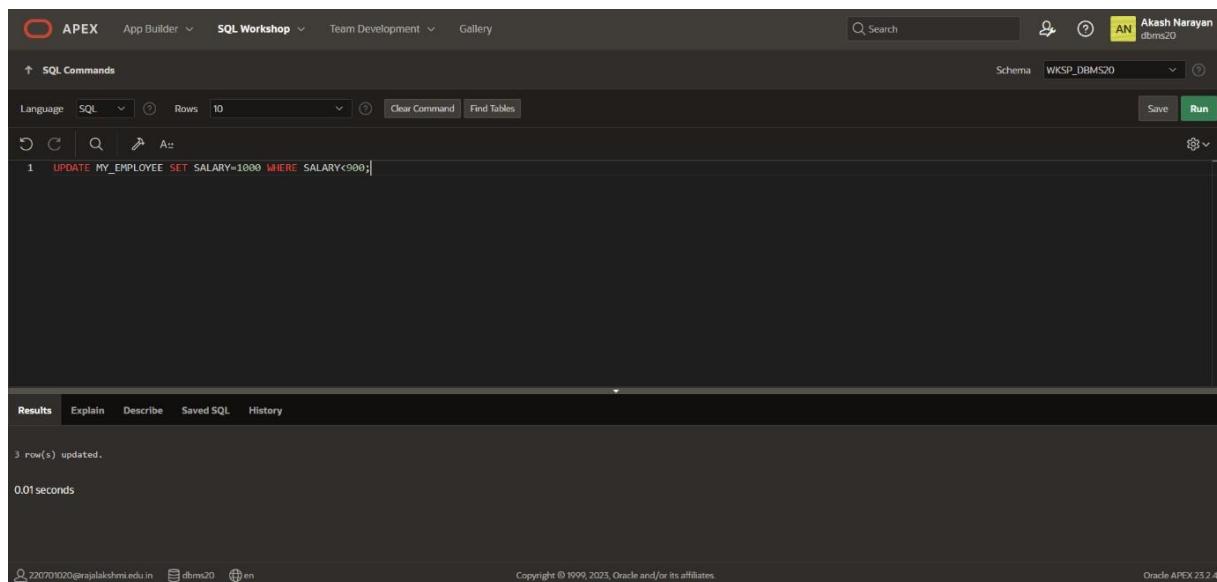
OUTPUT:

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 top right shows the user 'Akash Narayan' and the schema 'WKSP_DBMS20'. The SQL command input field contains: 'UPDATE MY_EMPLOYEE SET LAST_NAME='DREXLER' WHERE ID=3;'. The results tab is selected, displaying the message '1 row(s) updated.' and '0.04 seconds'. The bottom footer includes the URL '220701020@rajalakshmi.edu.in', the schema 'dbms20', and the language 'en'. It also states 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

5.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. In the SQL Commands pane, the following command is entered:

```
1 UPDATE MY_EMPLOYEE SET SALARY=1000 WHERE SALARY<900;
```

In the Results pane, the output is:

```
3 row(s) updated.  
0.01 seconds
```

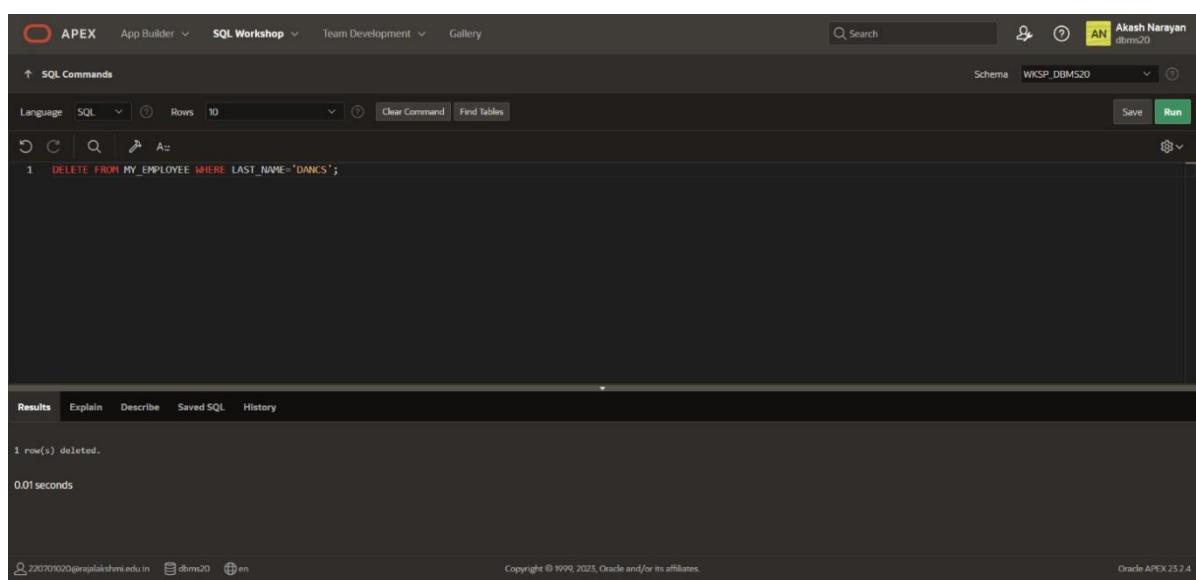
At the bottom, the footer includes:

```
220701020@rajalakshmi.edu.in dbms20 en  
Copyright © 1999, 2023, Oracle and/or its affiliates.  
Oracle APEX 23.2.4
```

6.Delete Betty dancs from MY_EMPLOYEE table.

QUERY: delete from My_Employee where last_name='dancs';

OUTPUT:



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

```
1 DELETE FROM MY_EMPLOYEE WHERE LAST_NAME='DANCS';
```

In the Results pane, the output is:

```
1 row(s) deleted.  
0.01 seconds
```

At the bottom, the footer includes:

```
220701020@rajalakshmi.edu.in dbms20 en  
Copyright © 1999, 2023, Oracle and/or its affiliates.  
Oracle APEX 23.2.4
```

7.Empty the fourth row of the emp table.

QUERY: delete from My_Employee where id=4;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A single command is entered: 'DELETE FROM NY_EMPLOYEE WHERE ID=4;'. Below the command, the results tab is selected, showing the output: '1 row(s) deleted.' and '0.04 seconds'. The bottom right corner of the interface displays 'Oracle APEX 23.2.4'.

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

RESULT:

EXNO:3

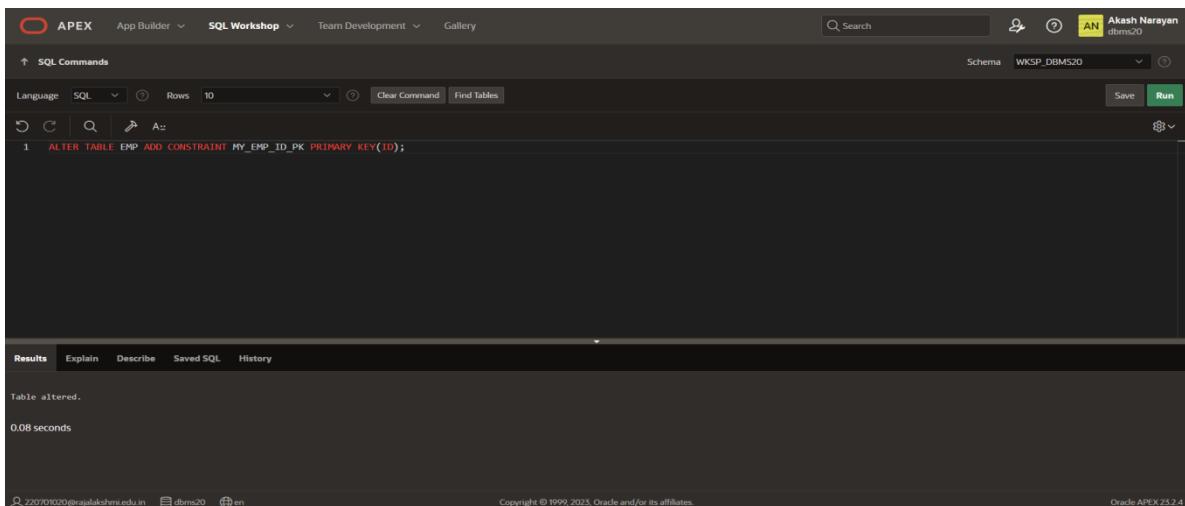
INCLUDING CONSTRAINTS

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, 'SQL Workshop' is selected. The main area contains the following SQL command:

```
1 ALTER TABLE EMP ADD CONSTRAINT MY_EMP_ID_PK PRIMARY KEY(ID);
```

Below the command, the results show:

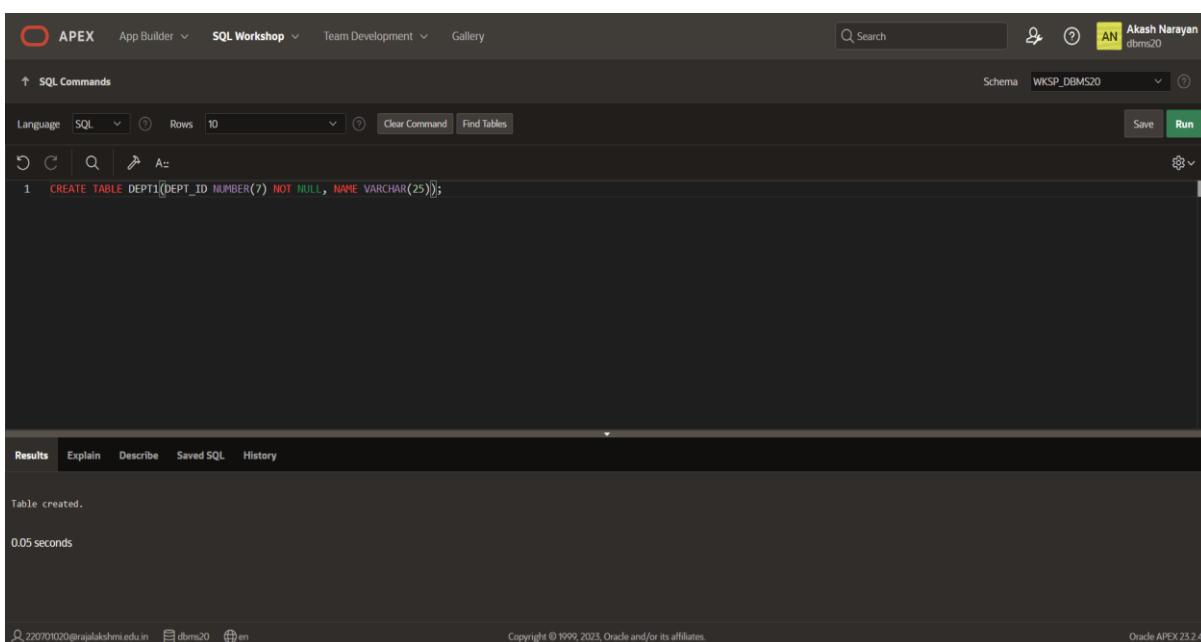
```
Table altered.  
0.08 seconds
```

At the bottom, the status bar indicates the user is connected to 'dbms20' and the session ID is '220701020@rajalakshmi.edu.in'. The copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 23.2.4' are also visible.

2.Create a PRIMAY KEY constraint to the DEPT table using the ID column. The constraint should be named at creation. Name the constraint my_dept_id_pk.

QUERY: create table dept1(dept_id number(7) not null, name varchar(25));

OUTPUT:



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

```
1 CREATE TABLE DEPT1(DEPT_ID NUMBER(?) NOT NULL, NAME VARCHAR(25));
```

Below the command, the results show:

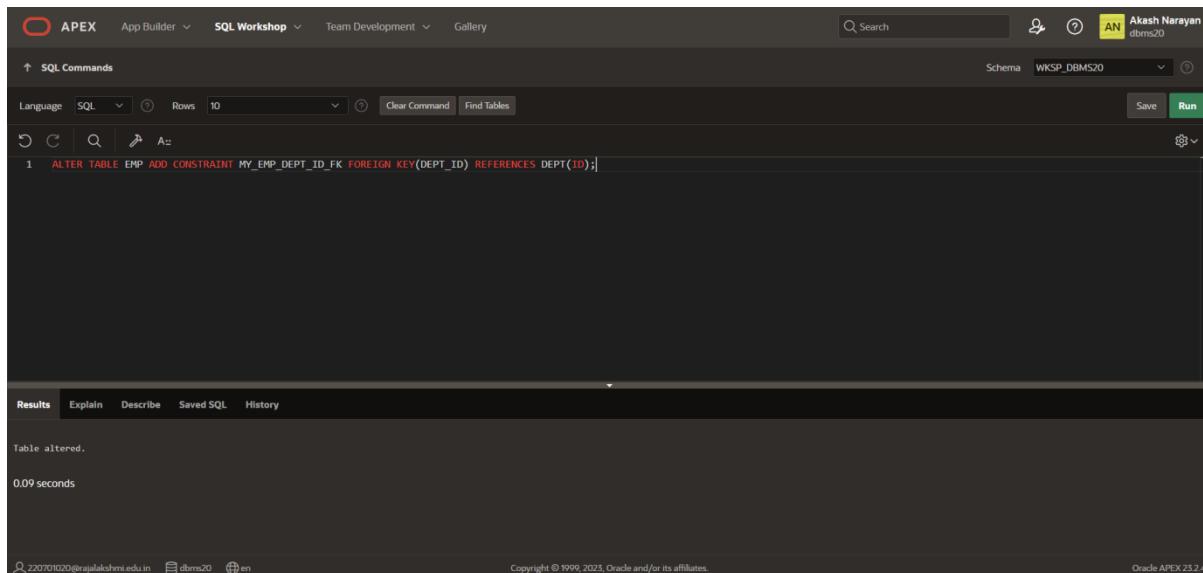
```
Table created.  
0.05 seconds
```

At the bottom, the status bar indicates the user is connected to 'dbms20' and the session ID is '220701020@rajalakshmi.edu.in'. The copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 23.2.4' are also visible.

3.Add a column DEPT_ID to the EMP table. Add a foreign key reference on the EMP table that ensures that the employee is not assigned to nonexistent department. Name the constraint my_emp_dept_id_fk.

QUERY: alter table emp add constraint my_emp_dept_id_fk foreign key(dept_id) references dept(id);

OUTPUT:

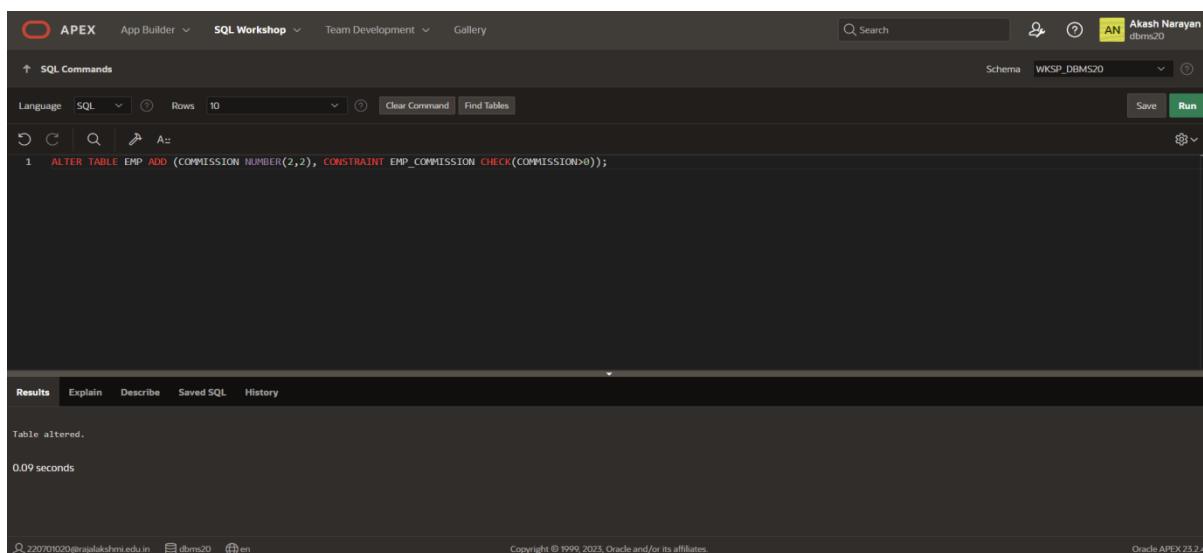


The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user 'Akash Narayan' and schema 'WKSP_DBMS20'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, showing the command: 'ALTER TABLE EMP ADD CONSTRAINT MY_EMP_DEPT_ID_FK FOREIGN KEY(DEPT_ID) REFERENCES DEPT(ID);'. Below the command, the results show: 'Table altered.' and '0.09 seconds'. The bottom footer includes copyright information and the version 'Oracle APEX 23.2.4'.

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

QUERY: alter table emp add (commission number(2,2), constraint emp_commission check(commission>0));

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user 'Akash Narayan' and schema 'WKSP_DBMS20'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, showing the command: 'ALTER TABLE EMP ADD (COMMISSION NUMBER(2,2), CONSTRAINT EMP_COMMISION CHECK(COMMISSION>0));'. Below the command, the results show: 'Table altered.' and '0.09 seconds'. The bottom footer includes copyright information and the version 'Oracle APEX 23.2.4'.

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

RESULT:

EXNO:4**WRITING BASIC SQL SELECT STATEMENTS****DATE:****1.Identify the Errors****SELECT employee_id, last_name ,sal*12 ANNUAL SALARY FROM employees;****QUERY: select id,last_name,salary*12 from my_employee;****OUTPUT:**

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

```
1 SELECT ID, LAST_NAME, SALARY*12 FROM MY_EMPLOYEE;
```

The results section displays the following data:

ID	LAST_NAME	SALARY*12
3	DREXLER	15200
1	PATEL	12000

2 rows returned in 0.01 seconds

2. Show the structure of departments the table. Select all the data from it.**QUERY: select * from my_employee;****OUTPUT:**

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

```
1 SELECT * FROM MY_EMPLOYEE;
```

The results section displays the following data:

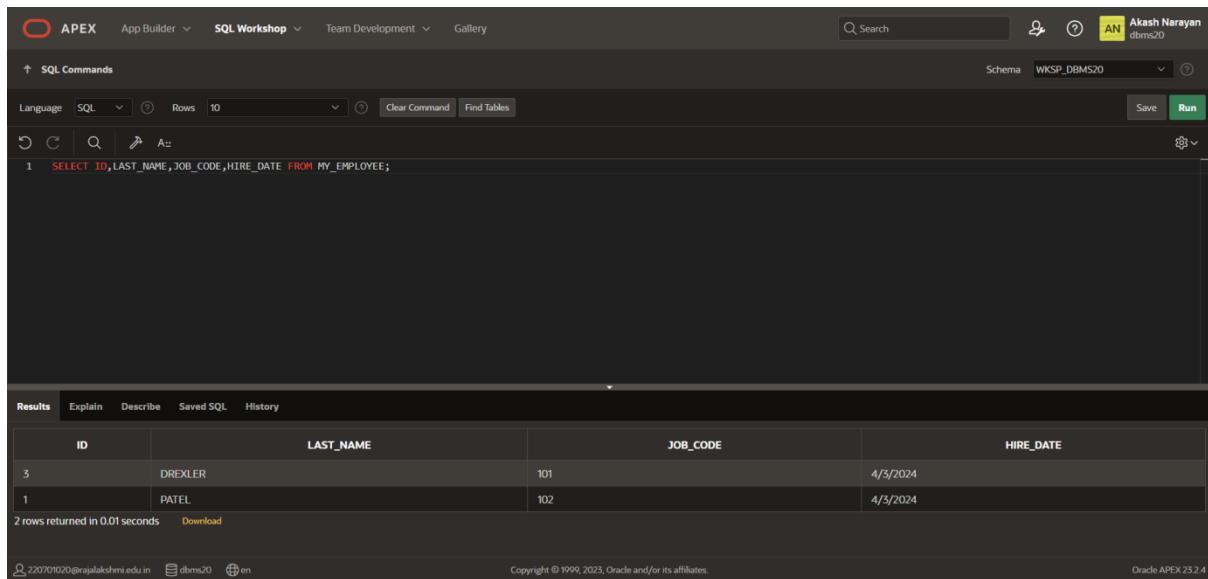
ID	LAST_NAME	FIRST_NAME	USERID	SALARY
3	DREXLER	BEN	BBIRI	1100
1	PATEL	RALPH	RPATEL	1000

2 rows returned in 0.02 seconds

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 id,last_name,job_code,hire_date from my_employee;

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows the user 'Akash Narayan' and the schema 'WKSP_DBMS20'. The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the query 'SELECT ID, LAST_NAME, JOB_CODE, HIRE_DATE FROM MY_EMPLOYEE;' is entered. Under 'Results', the output is displayed in a table:

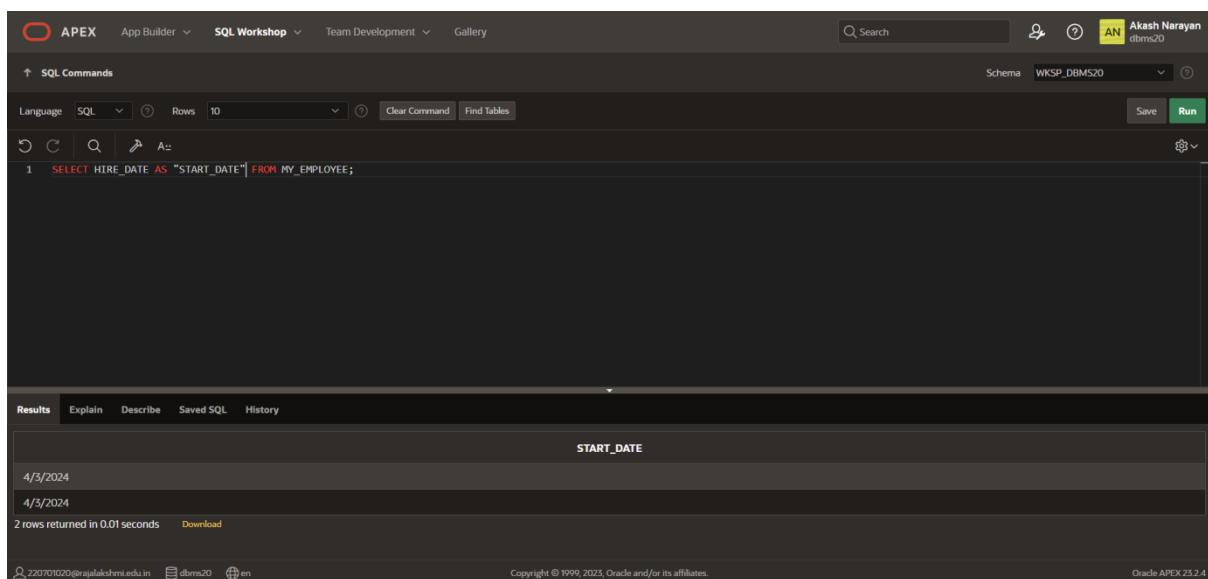
ID	LAST_NAME	JOB_CODE	HIRE_DATE
3	DREXLER	101	4/3/2024
1	PATEL	102	4/3/2024

Below the table, it says '2 rows returned in 0.01 seconds' and there's a 'Download' link. The bottom footer includes the URL '220701020@rajalakshmi.edu.in', the schema 'dbms20', and the language 'en'. Copyright information and the version 'Oracle APEX 23.2.4' are also present.

4. Provide an alias STARTDATE for the hire date.

QUERY: Select hire_date as "start_date" from my_employee;

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface, similar to the previous one. The top navigation bar and user information are the same. The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the query 'SELECT HIRE_DATE AS "START_DATE" FROM MY_EMPLOYEE;' is entered. Under 'Results', the output is displayed in a table:

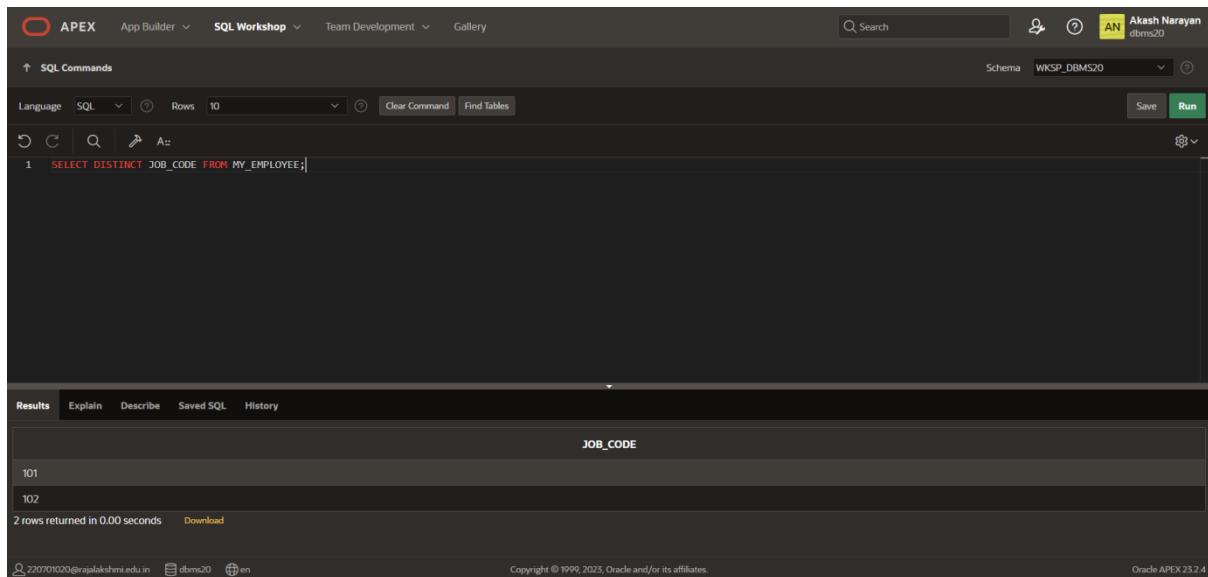
START_DATE
4/3/2024
4/3/2024

Below the table, it says '2 rows returned in 0.01 seconds' and there's a 'Download' link. The bottom footer includes the URL '220701020@rajalakshmi.edu.in', the schema 'dbms20', and the language 'en'. Copyright information and the version 'Oracle APEX 23.2.4' are also present.

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

QUERY: select distinct job_code from my_employee;

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 top right shows the user 'Akash Narayan' and the schema 'WKSP_DBMS20'. The main area has tabs for 'SQL Commands' and 'Results'. The SQL command entered is 'SELECT DISTINCT JOB_CODE FROM MY_EMPLOYEE;'. The results tab displays the output:

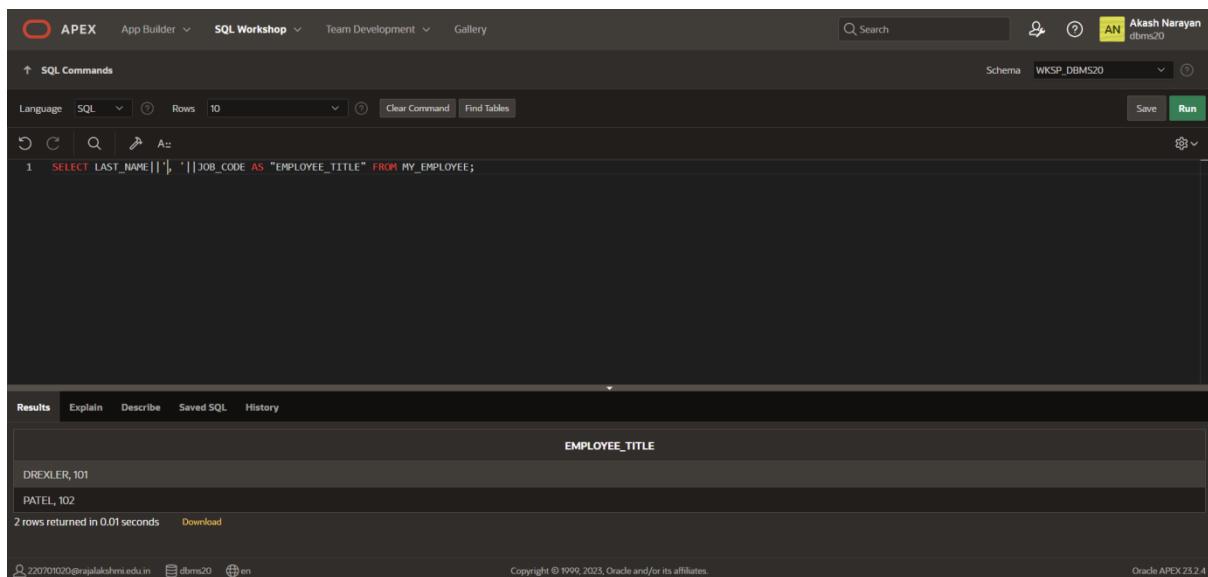
JOB_CODE
101
102

Below the results, it says '2 rows returned in 0.00 seconds' and 'Download'.

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

QUERY: select last_name||', '||job_code as "employee_title" from my_employee;

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 top right shows the user 'Akash Narayan' and the schema 'WKSP_DBMS20'. The main area has tabs for 'SQL Commands' and 'Results'. The SQL command entered is 'SELECT LAST_NAME||', '||JOB_CODE AS "EMPLOYEE_TITLE" FROM MY_EMPLOYEE;'. The results tab displays the output:

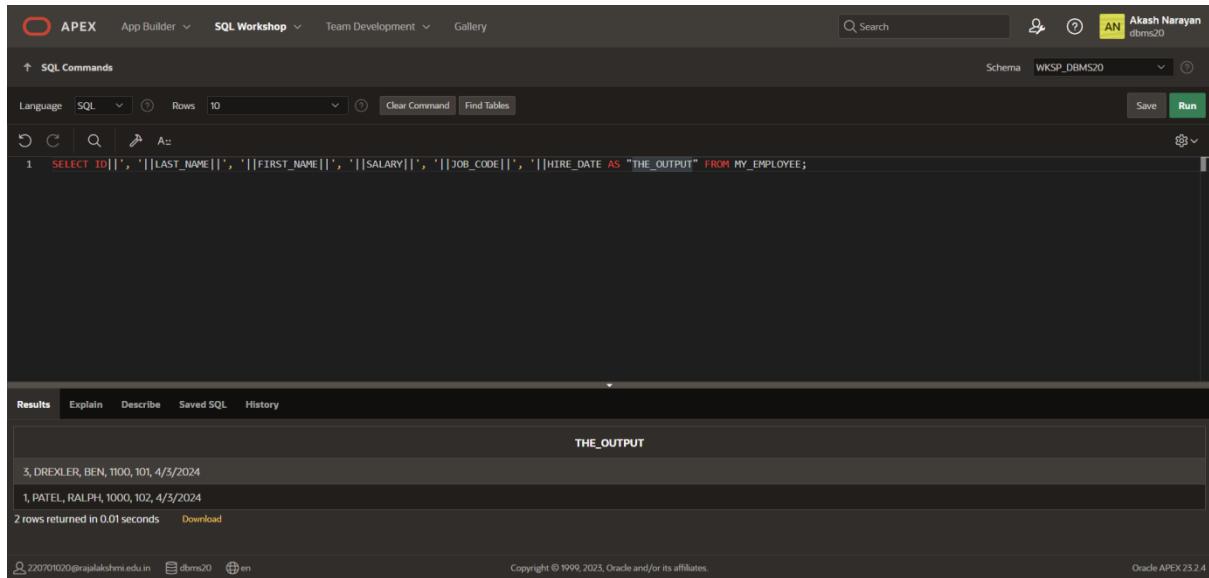
EMPLOYEE_TITLE
DREXLER, 101
PATEL, 102

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

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

QUERY: select id||', '||last_name||', '||first_name||', '||salary||', '||job_code||', '||hire_date as "the_output" from my_employee;

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows the user 'AN Akash Narayan' and the schema 'WKSP_DBMS20'. The main area has tabs for 'SQL Commands' and 'Results'. The SQL command entered is:

```
1 SELECT ID||', '||LAST_NAME||', '||FIRST_NAME||', '||SALARY||', '||JOB_CODE||', '||HIRE_DATE AS "THE_OUTPUT" FROM MY_EMPLOYEE;
```

The results tab displays the output:

THE_OUTPUT
3, DREXLER, BEN, 1100, 101, 4/3/2024
1, PATEL, RALPH, 1000, 102, 4/3/2024

Below the results, it says '2 rows returned in 0.01 seconds' and provides download options. The bottom footer includes the URL '220701020@rajalakshmi.edu.in', the schema 'dbms20', and the page number '1'. The copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 23.2.4' are also present.

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

RESULT:

EXNO:5

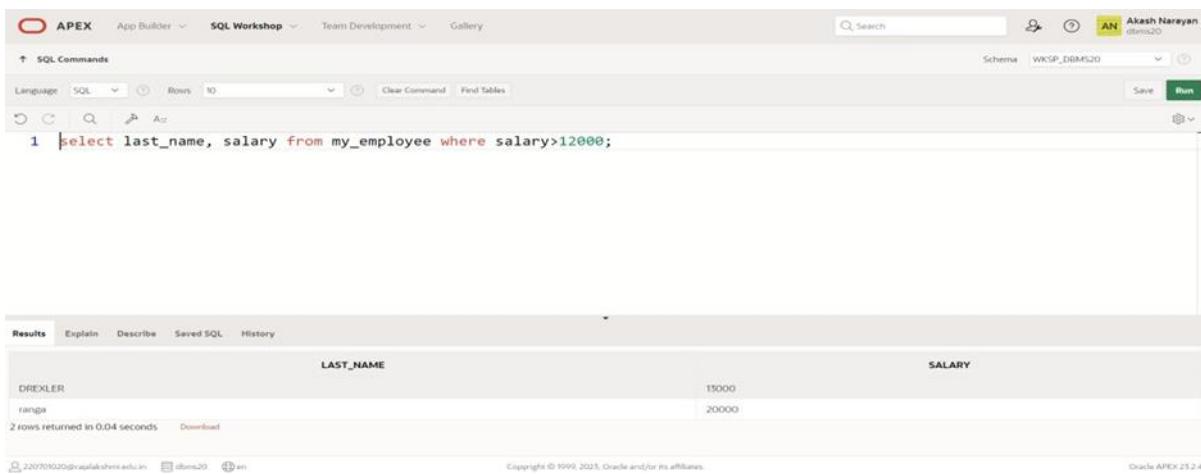
RESTRICTING AND SORTING DATA

DATE:

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

QUERY: select last_name, salary from My_employee where salary > 12000;

OUTPUT:



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

```
1 | select last_name, salary from my_employee where salary>12000;
```

The results section displays the following data:

LAST_NAME	SALARY
DREXLER	15000
ranga	20000

2 rows returned in 0.04 seconds.

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

QUERY: select last_name, dept_id from My_employee where emp_id = 176;

OUTPUT:



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

```
1 | select last_name, job_code from my_employee where id=176;
```

The results section displays the following data:

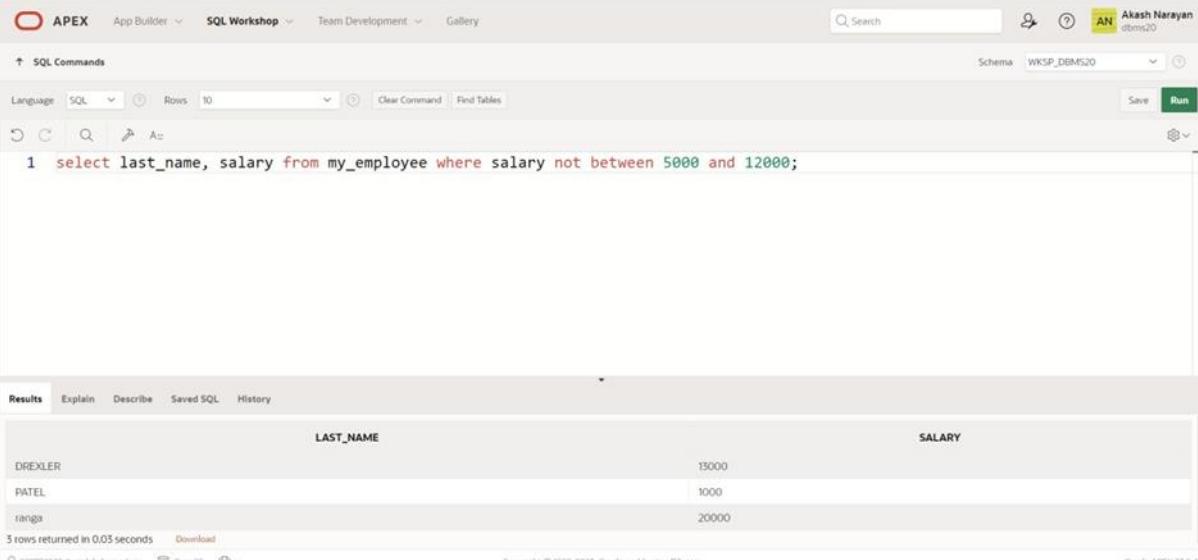
LAST_NAME	JOB_CODE
PATEL	176

1 rows returned in 0.01 seconds.

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

QUERY: select last_name, salary from My_employee where salary>12000 or salary<5000;

OUTPUT:



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

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

The results section displays the following data:

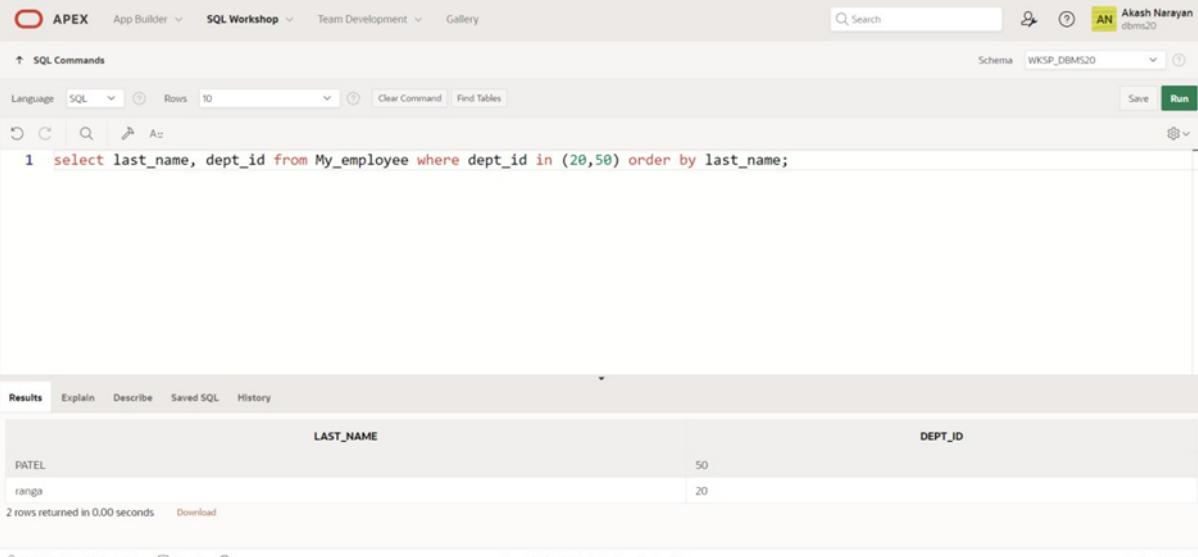
LAST_NAME	SALARY
DIREXLER	13000
PATEL	1000
ranga	20000

3 rows returned in 0.03 seconds

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

QUERY: select last_name, dept_id from My_employee where dept_id in (20,50) order by last_name;

OUTPUT:



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

```
1 select last_name, dept_id from My_employee where dept_id in (20,50) order by last_name;
```

The results section displays the following data:

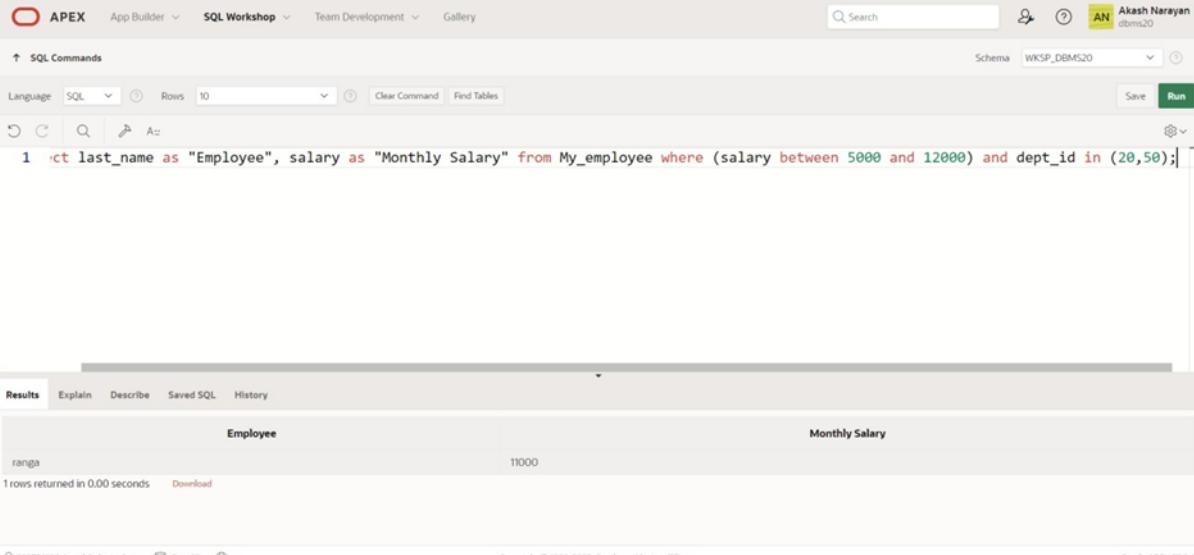
LAST_NAME	DEPT_ID
PATEL	50
ranga	20

2 rows returned in 0.00 seconds

5. 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 My_employee where (salary between 5000 and 12000) and dept_id in (20,50);

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile for 'Akash Narayan dbms20'. The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the query is written:

```
1 select last_name as "Employee", salary as "Monthly Salary" from My_employee where (salary between 5000 and 12000) and dept_id in (20,50);
```

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

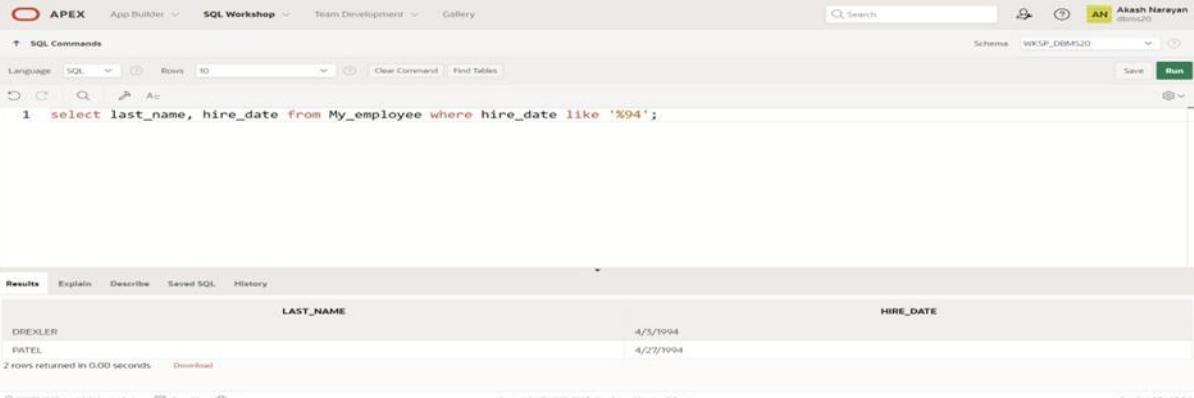
Employee	Monthly Salary
ranga	11000

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

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

QUERY: select last_name, hire_date from My_employee where hire_date like ‘%94’;

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile for 'Akash Narayan dbms20'. The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the query is written:

```
1 select last_name, hire_date from My_employee where hire_date like '%94';
```

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

LAST_NAME	HIRE_DATE
DREXLER	4/5/1994
PATEL	4/27/1994

Below the table, it says '2 rows returned in 0.00 seconds'.

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

QUERY: select last_name, job_title from My_employee where manager_id is null;

OUTPUT:

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

```
1 select last_name, job_title from My_employee where manager_id is null;
```

The results table has two columns: LAST_NAME and JOB_TITLE. The data returned is:

LAST_NAME	JOB_TITLE
DREXLER	
PATEL	manager
ranga	

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

8.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 from My_employee where commission is not null order by salary, commission;

OUTPUT:

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

```
1 select last_name, salary, commission from My_employee where commission is not null order by salary, commission;
```

The results table has three columns: LAST_NAME, SALARY, and COMMISSION. The data returned is:

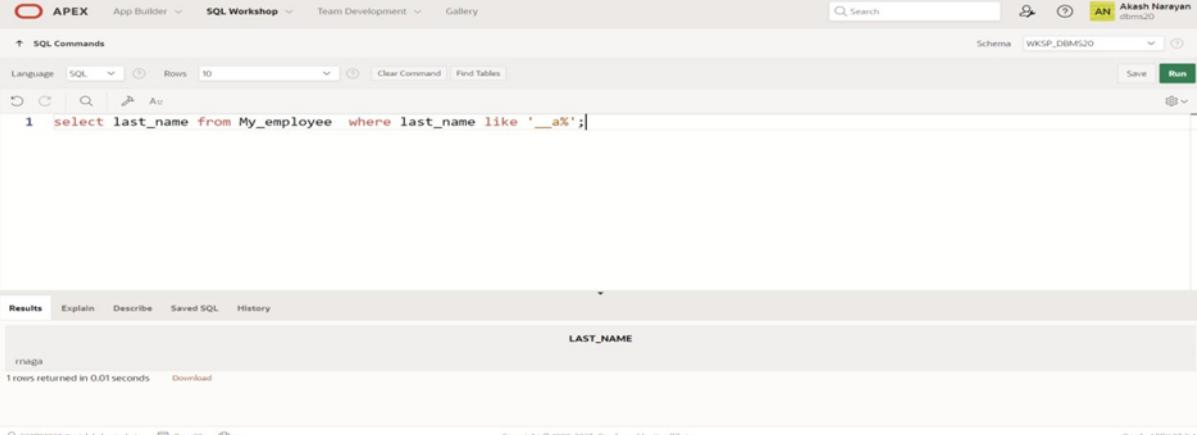
LAST_NAME	SALARY	COMMISSION
DREXLER	7000	20
ranga	11000	50

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

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

QUERY: select last_name from My_employee where last_name like ‘_____a%’;

OUTPUT:



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

```
1 select last_name from My_employee where last_name like '____a%';
```

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

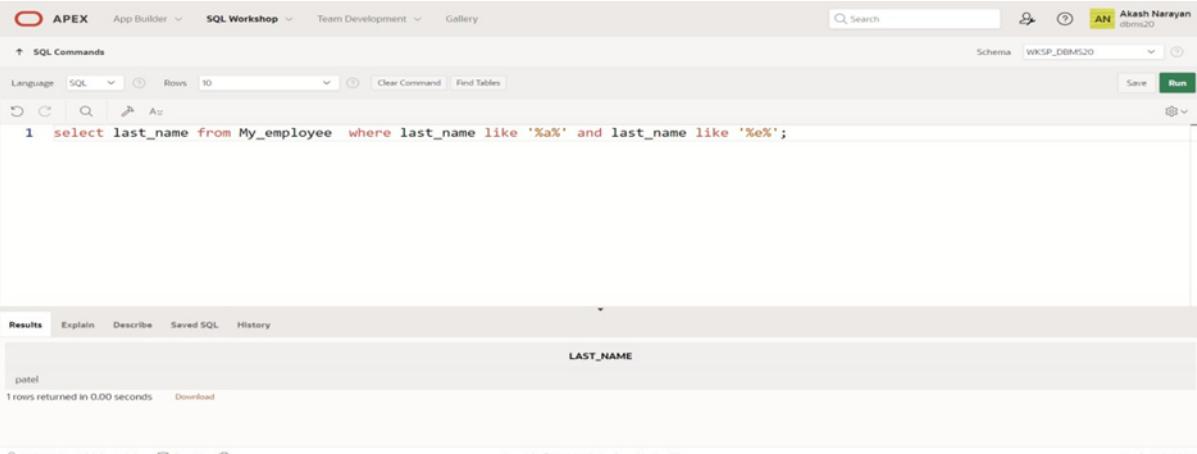
LAST_NAME
maga

1 rows returned in 0.01 seconds

10.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 My_employee where last_name like ‘%a%’ and last_name like ‘%e%’;

OUTPUT:



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

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

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

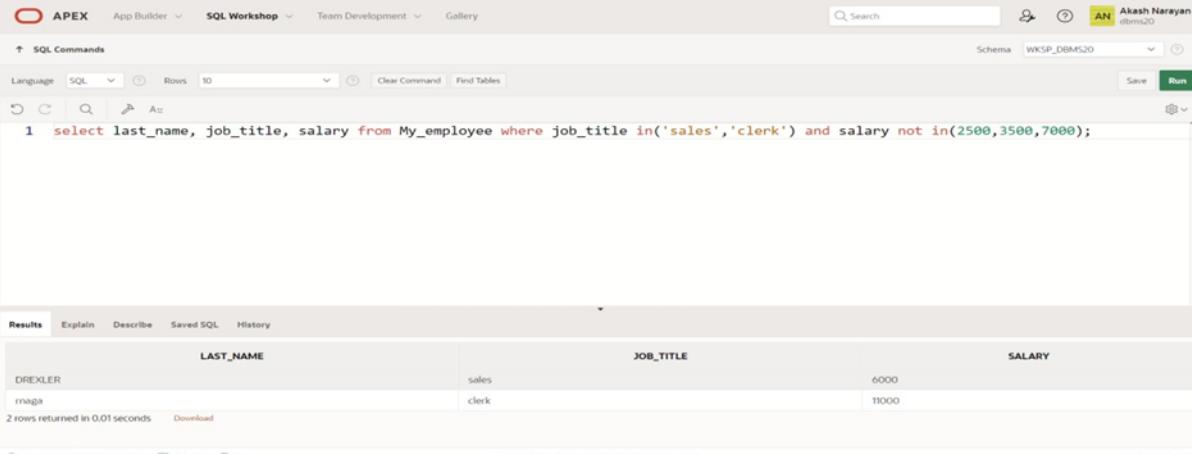
LAST_NAME
patel

1 rows returned in 0.00 seconds

11. 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_title, salary from My_employee where job_title in('sales','clerk') and salary not in(2500,3500,7000);

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows the user 'Akash Narayan' and the schema 'WKSP_DBMS20'. The main area has tabs for 'SQL Commands' and 'Results'. The SQL command entered is:

```
1 select last_name, job_title, salary from My_employee where job_title in('sales','clerk') and salary not in(2500,3500,7000);
```

The results section displays the following data:

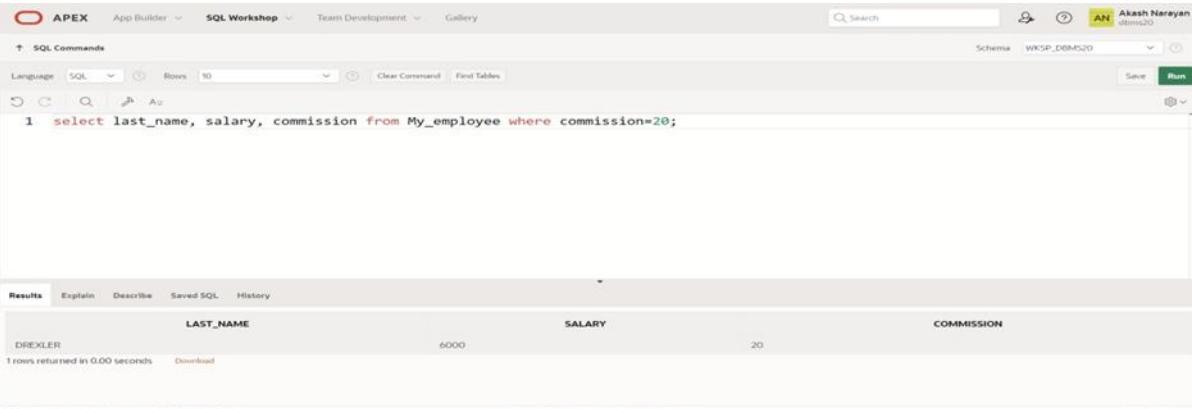
LAST_NAME	JOB_TITLE	SALARY
DREXLER	sales	6000
rnaga	clerk	11000

2 rows returned in 0.01 seconds. The bottom status bar shows the connection details: 220701020@rajalakshmi.edu.in, dbms20, en, and Oracle APEX 23.2.4.

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

QUERY: select last_name, salary, commission from My_employee where commission=20;

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows the user 'Akash Narayan' and the schema 'WKSP_DBMS20'. The main area has tabs for 'SQL Commands' and 'Results'. The SQL command entered is:

```
1 select last_name, salary, commission from My_employee where commission=20;
```

The results section displays the following data:

LAST_NAME	SALARY	COMMISSION
DREXLER	6000	20

1 rows returned in 0.00 seconds. The bottom status bar shows the connection details: 220701020@rajalakshmi.edu.in, dbms20, en, and Oracle APEX 23.2.4.

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

RESULT:

EXNO:6

SINGLE ROW FUNCTIONS

DATE:

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

QUERY: select sysdate as "DATE" from dual;

OUTPUT:

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

```
1 select sysdate from dual;
2
```

In the Results pane, the output is displayed as:

SYSDATE
04/04/2024

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 emp_id, lname, salary, salary+(salary*15.5/100) as "New Salary" from My_emp;

OUTPUT:

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

```
1 select id, lname, salary, salary+(salary*15.5/100) as "New Salary" from My_emp;
```

In the Results pane, the output is displayed as:

ID	LNAME	SALARY	New Salary
2	Esh	400000	462000
5	Kumar	1200000	1386000
1	Narayan	2000000	2310000
4	Sha	600000	693000
6	Givl	10000	11550
3	Partha	700000	808500

Below the results, it says "6 rows returned in 0.01 seconds".

3.Modify your query lab_03_02.sql to add a column that subtracts the old salary from the new salary. Label the column Increase.

QUERY: select emp_id, lname, salary, salary+(salary*15.5/100) as "New Salary", (salary+(salary*15.5/100))-salary as "Increase" from My_emp;

OUTPUT:

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

```
1 select id, lname, salary, salary+(salary*15.5/100) as "New Salary", (salary+(salary*15.5/100))-salary as "Increase" from My_emp;
```

The results section displays the following data:

ID	LNAME	SALARY	New Salary	Increase
2	Esh	400000	462000	62000
5	Kumar	1200000	1386000	186000
1	Narayan	2000000	2310000	310000
4	Sha	600000	693000	93000
6	Gavi	10000	11550	1550
3	Partha	700000	808500	108500

6 rows returned in 0.05 seconds

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(lname) as "Name", length(lname) as "Length of Name" from My_emp where lname like 'J%' or lname like 'A%' or lname like 'M%' order by lname;

OUTPUT:

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

```
1 select initcap(lname) as "Name", length(lname) as "Length of Name" from My_emp where lname like 'J%' or lname like 'A%' or lname like 'M%' order by lname;
```

The results section displays the following data:

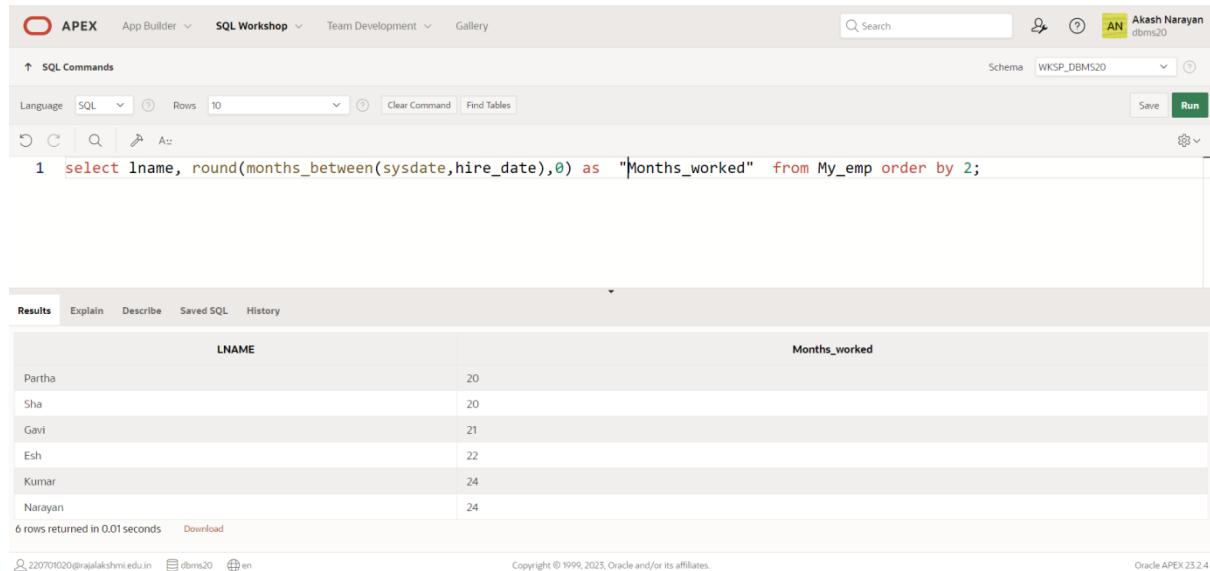
Name	Length of Name
	no data found

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

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

QUERY: select lname, round(months_between(sysdate,h_date),0) as "Months_worked" from My_emp order by 2;

OUTPUT:



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

```
1 select lname, round(months_between(sysdate,hire_date),0) as "Months_worked" from My_emp order by 2;
```

The results page displays the following data:

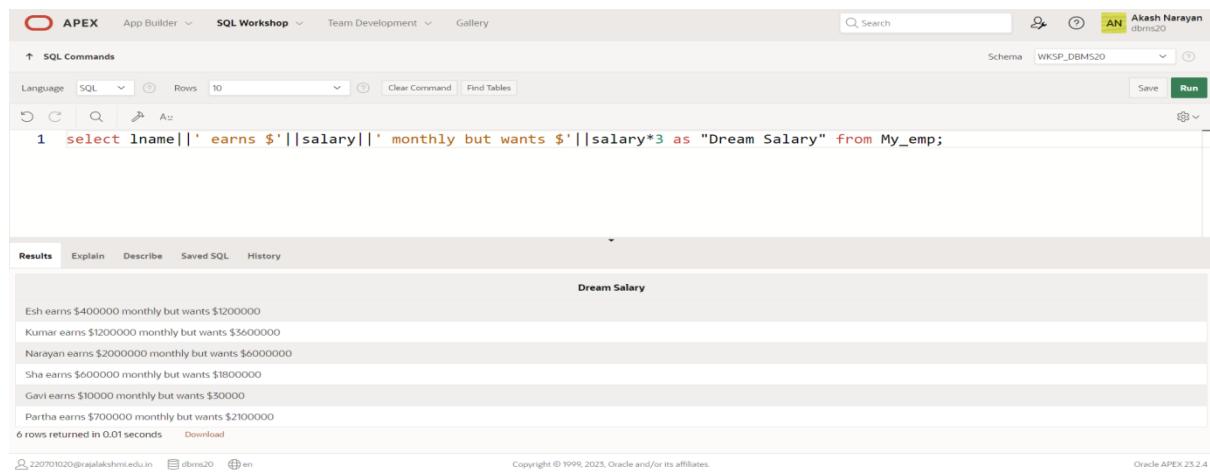
LNAME	Months_worked
Partha	20
Sha	20
Gavi	21
Esh	22
Kumar	24
Narayan	24

6 rows returned in 0.01 seconds

6. Create a report that produces the following for each employee:<employee last name> earns <salary> monthly but wants <3 times salary>. Label the column Dream Salaries.

QUERY: select lname||' earns \$'||salary||' monthly but wants \$'||salary*3 as "Dream Salary" from My_emp;

OUTPUT:



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

```
1 select lname||' earns $'||salary||' monthly but wants $'||salary*3 as "Dream Salary" from My_emp;
```

The results page displays the following data:

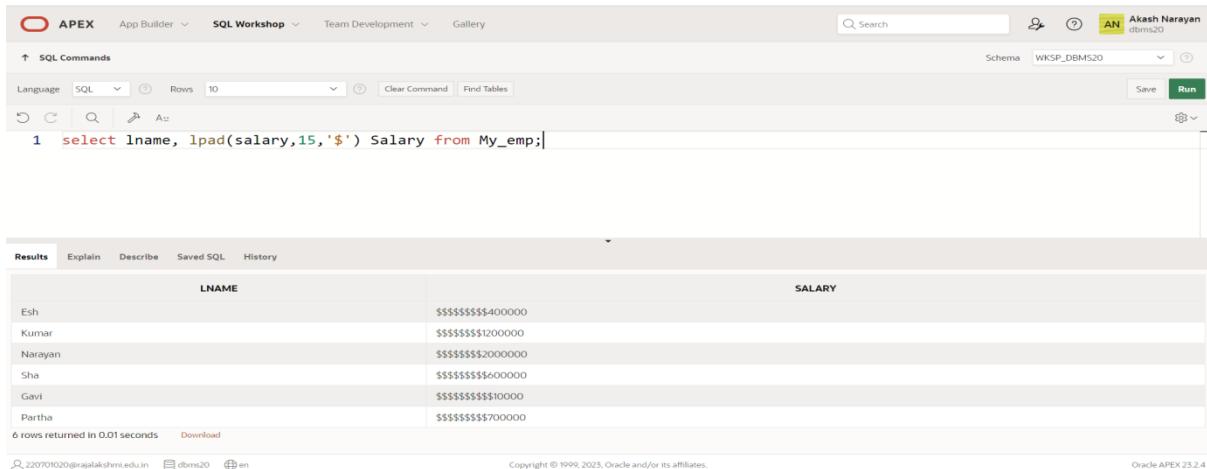
Dream Salary
Esh earns \$400000 monthly but wants \$1200000
Kumar earns \$1200000 monthly but wants \$3600000
Narayan earns \$2000000 monthly but wants \$6000000
Sha earns \$600000 monthly but wants \$1800000
Gavi earns \$100000 monthly but wants \$300000
Partha earns \$700000 monthly but wants \$2100000

6 rows returned in 0.01 seconds

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

QUERY: select lname, lpad(salary,15,'\$') Salary from My_emp;

OUTPUT:



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

```
1 select lname, lpad(salary,15,'$') Salary from My_emp;
```

The results window displays the output:

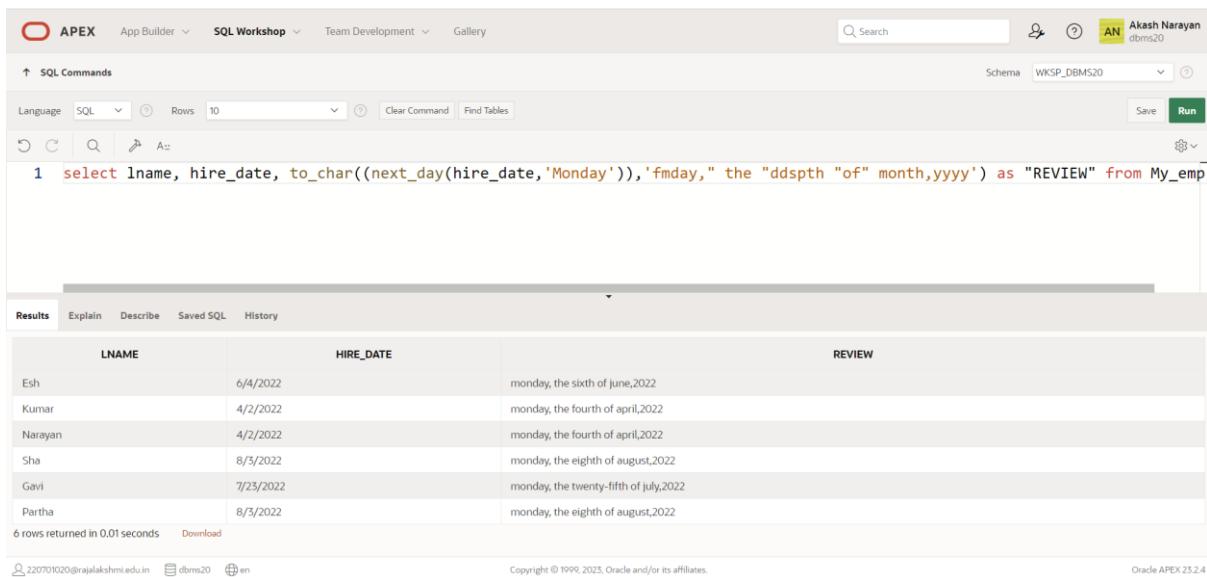
LNAME	SALARY
Esh	\$\$\$\$\$\$\$\$\$\$400000
Kumar	\$\$\$\$\$\$\$\$\$1200000
Narayan	\$\$\$\$\$\$\$\$\$2000000
Sha	\$\$\$\$\$\$\$\$\$600000
Gavi	\$\$\$\$\$\$\$\$\$100000
Partha	\$\$\$\$\$\$\$\$\$700000

6 rows returned in 0.01 seconds

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

QUERY: select lname, h_date, to_char((next_day(h_date,'Monday')),'fmday," the "ddspth "of" month,yyyy') as "REVIEW" from My_emp;

OUTPUT:



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

```
1 select lname, hire_date, to_char((next_day(hire_date,'Monday')),'fmday," the "ddspth "of" month,yyyy') as "REVIEW" from My_emp;
```

The results window displays the output:

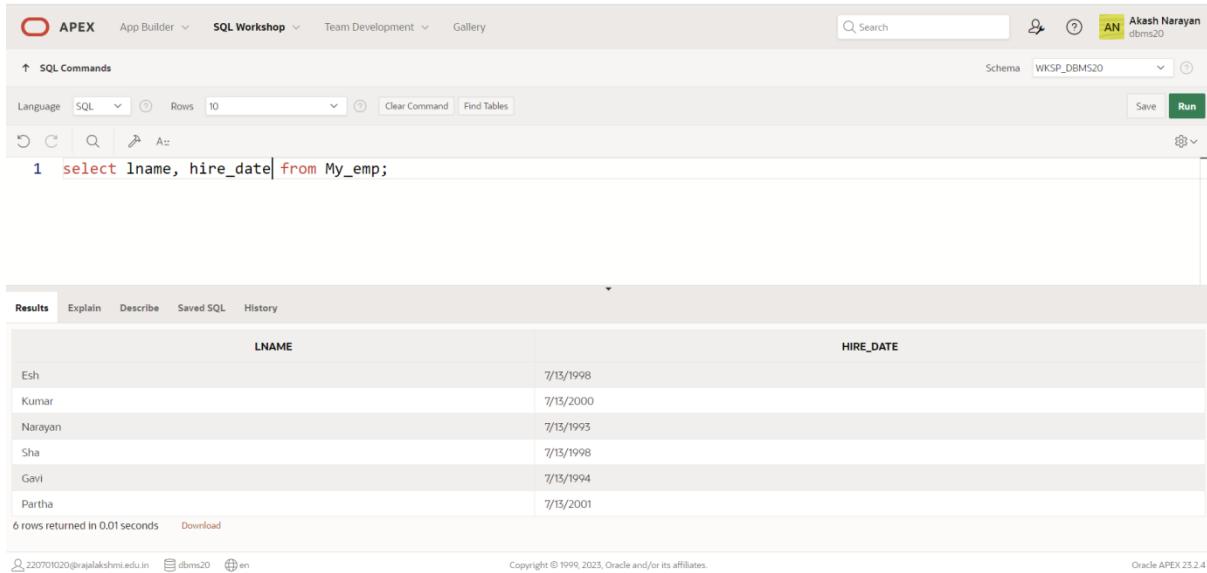
LNAME	HIRE_DATE	REVIEW
Esh	6/4/2022	monday, the sixth of june,2022
Kumar	4/2/2022	monday, the fourth of april,2022
Narayan	4/2/2022	monday, the fourth of april,2022
Sha	8/3/2022	monday, the eighth of august,2022
Gavi	7/23/2022	monday, the twenty-fifth of july,2022
Partha	8/3/2022	monday, the eighth of august,2022

6 rows returned in 0.01 seconds

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

QUERY: select Lname, h_date, to_char(h_date,'Day')as "Day" from My_emp order by to_char(h_date-1,'d');

OUTPUT:



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

```
1 select lname, hire_date| from My_emp;
```

The results section displays the following data:

LNAME	HIRE_DATE
Esh	7/13/1998
Kumar	7/13/2000
Narayan	7/13/1993
Sha	7/13/1998
Gavi	7/13/1994
Partha	7/13/2001

6 rows returned in 0.01 seconds

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

RESULT:

DISPLAYING DATA FROM MULTIPLE TABLES

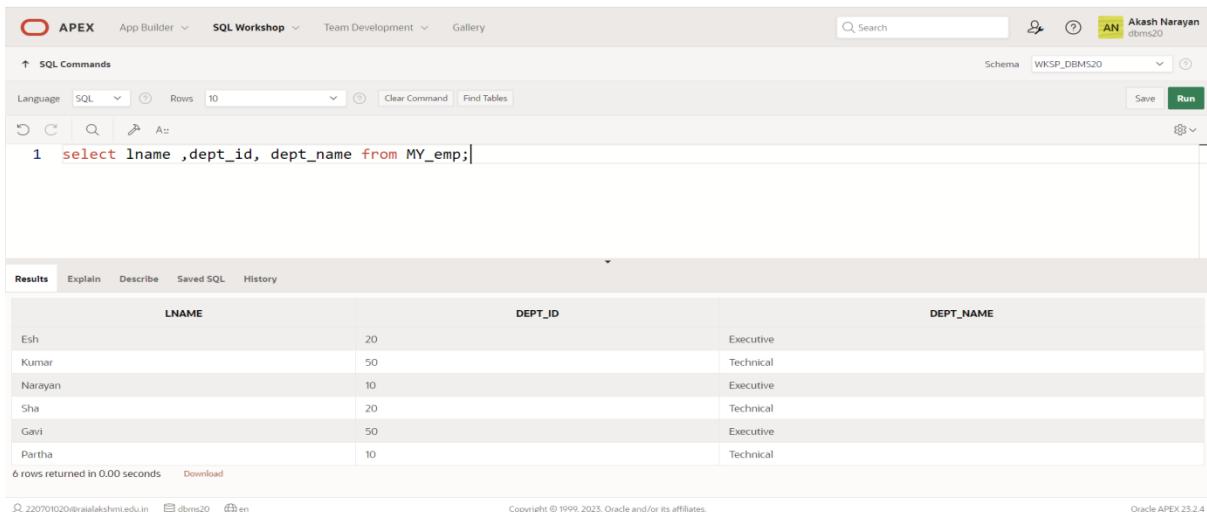
EXNO:7

DATE:

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

QUERY: select lname ,dept_id, dept_name from MY_emp;

OUTPUT:



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

```
1 select lname ,dept_id, dept_name from MY_emp;
```

The results section displays the following data:

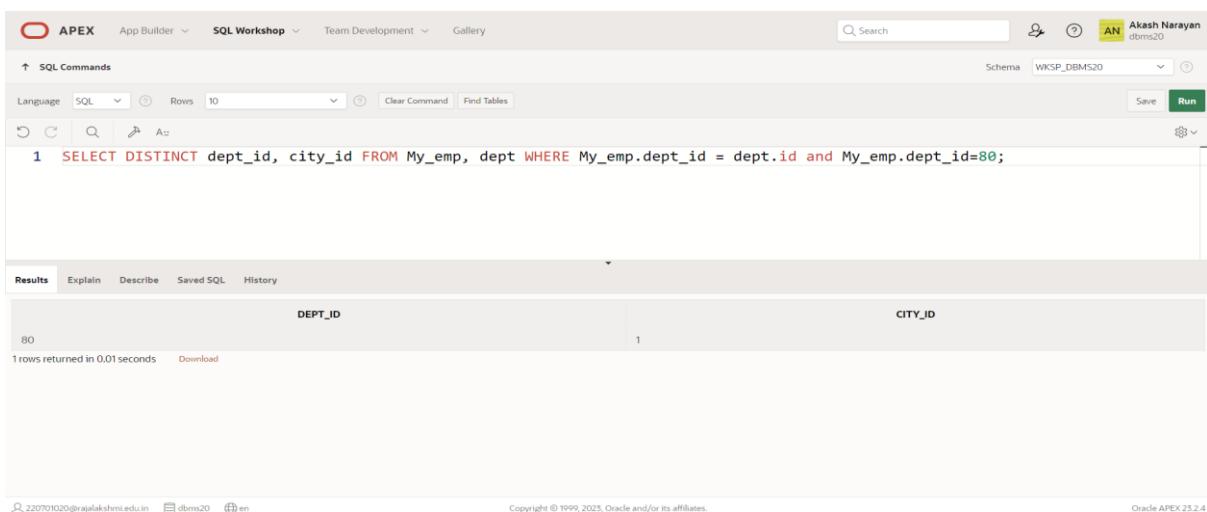
LNAME	DEPT_ID	DEPT_NAME
Esh	20	Executive
Kumar	50	Technical
Narayan	10	Executive
Sha	20	Technical
Gavi	50	Executive
Partha	10	Technical

6 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 job_id, loc_id FROM My_emp, dept WHERE My_emp.dept_id = dept.dept_id and My_emp.d_id=80;

OUTPUT:



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

```
1 SELECT DISTINCT dept_id, city_id FROM My_emp, dept WHERE My_emp.dept_id = dept.id and My_emp.dept_id=80;
```

The results section displays the following data:

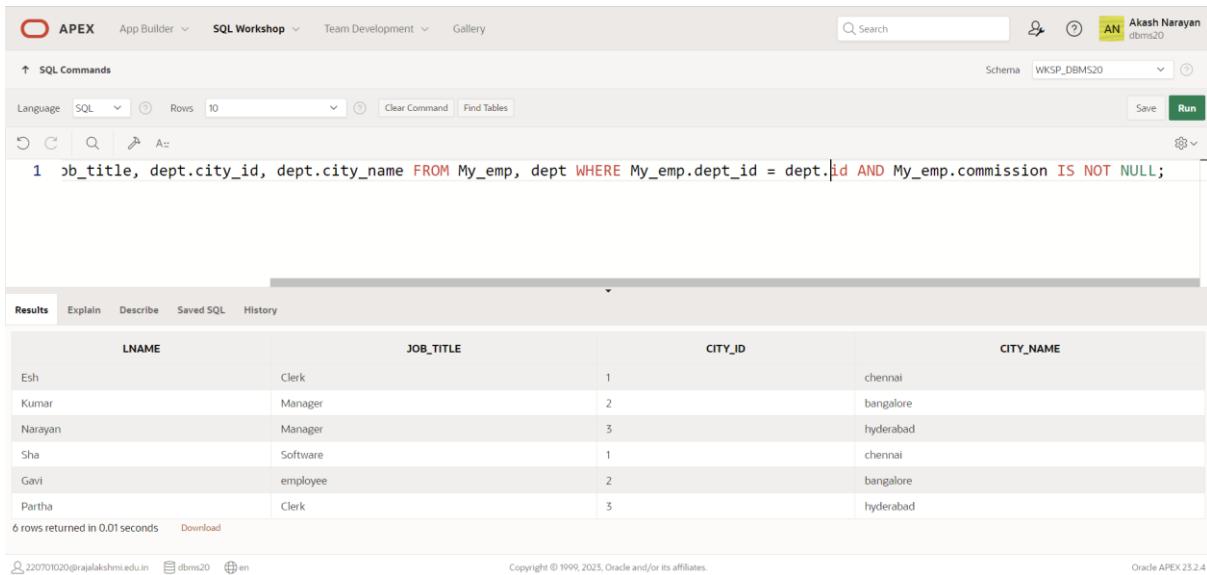
DEPT_ID	CITY_ID
80	1

1 rows returned in 0.01 seconds

3. Write a query to display the employee last name, department name, location ID, and city of all employees who earn a commission

QUERY: `SELECT My_emp.lname, My_emp.job_title, dept.loc_id, My_emp.city
FROM My_emp, dept WHERE My_emp.dept_id = dept.dept_id AND
My_emp.commission IS NOT NULL;`

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is identified as 'Akash Narayan' with the schema 'WKSP_DBMS20'. The main area is titled 'SQL Commands' with a 'Run' button. A SQL query is entered in the command line:

```
1 job_title, dept.city_id, dept.city_name FROM My_emp, dept WHERE My_emp.dept_id = dept.id AND My_emp.commission IS NOT NULL;
```

The results tab is selected, displaying the following data:

LNAME	JOB_TITLE	CITY_ID	CITY_NAME
Esh	Clerk	1	chennai
Kumar	Manager	2	bangalore
Narayan	Manager	3	hyderabad
Sha	Software	1	chennai
Gavi	employee	2	bangalore
Partha	Clerk	3	hyderabad

Below the table, it says '6 rows returned in 0.01 seconds'. The bottom of the page includes copyright information and a footer.

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

RESULT:

EXNO:8 AGGREGATING DATA USING GROUP FUNCTIONS DATE:

1. 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 max(salary) as "MAXIMUM", min(salary) as "MINIMUM", sum(salary) as "SUM", round(avg(salary),2) as "average" from My_emp;

OUTPUT:

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

```
1 select max(salary) as "MAXIMUM", min(salary) as "MINIMUM", sum(salary) as "SUM", round(avg(salary),2) as "average" from My_emp;
```

The results table displays the following data:

	MAXIMUM	MINIMUM	SUM	average
	2000000	10000	4910000	818355.33

1 rows returned in 0.00 seconds

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

QUERY: select job_title,max(salary) as "MAXIMUM", min(salary) as "MINIMUM",sum(salary) as "SUM",round(avg(salary),2) as "average" from My_emp group by job_title;

OUTPUT:

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

```
1 select job_title, max(salary) as "MAXIMUM", min(salary) as "MINIMUM", sum(salary) as "SUM", round(avg(salary),2) as "average" from My_emp group by job_title;
```

The results table displays the following data:

JOB_TITLE	MAXIMUM	MINIMUM	SUM	average
employee	10000	10000	10000	10000
Software	600000	600000	600000	600000
Clerk	700000	400000	1100000	550000
Manager	2000000	1200000	3200000	1600000

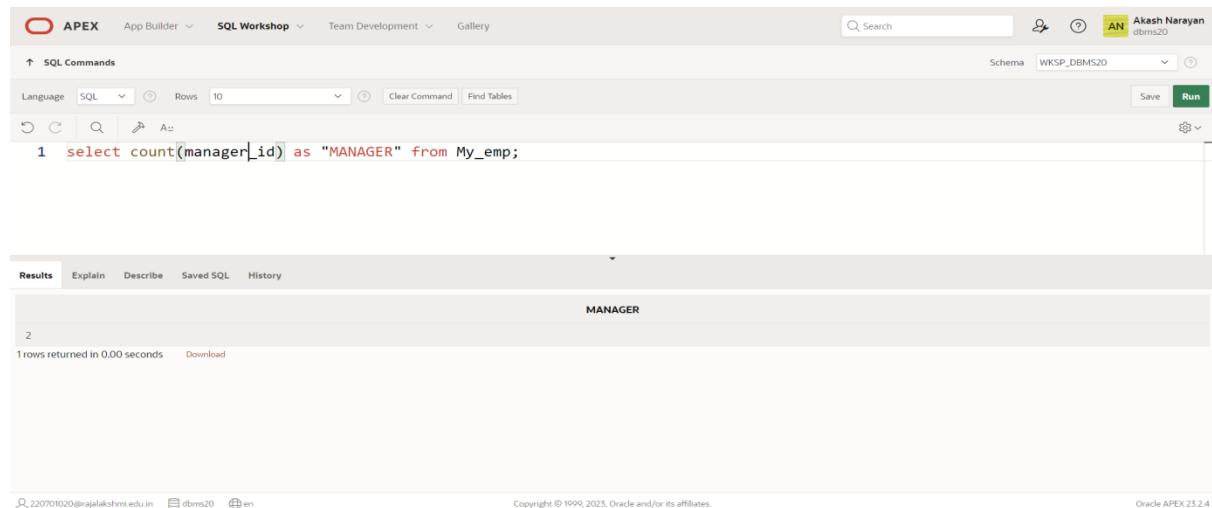
4 rows returned in 0.01 seconds

3. 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(mg_id) as "MANAGER" from My_emp;

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows the user 'AN Akash Narayan' and the schema 'WKSP_DBMS20'. The main area has tabs for 'SQL Commands', 'Results' (selected), 'Explain', 'Describe', 'Saved SQL', and 'History'. The SQL command entered is:

```
1 select count(manager_id) as "MANAGER" from My_emp;
```

The results section displays the output:

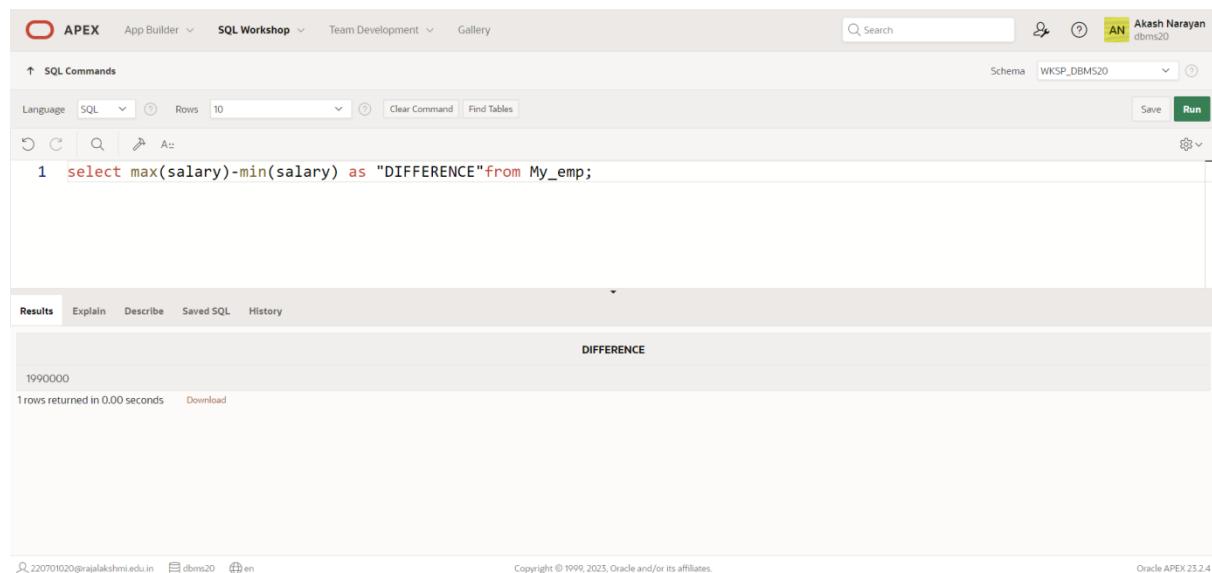
MANAGER
2

Below the results, it says '1 rows returned in 0.00 seconds' and provides a 'Download' link. The bottom footer includes the URL '220701020@rajalakshmi.edu.in', the schema 'dbms20', and the session ID 'en'. Copyright information 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 23.2.4' are also present.

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

QUERY: select max(salary)-min(salary) as "DIFFERENCE"from My_emp;

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface, identical to the previous one but with a different query. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows the user 'AN Akash Narayan' and the schema 'WKSP_DBMS20'. The main area has tabs for 'SQL Commands', 'Results' (selected), 'Explain', 'Describe', 'Saved SQL', and 'History'. The SQL command entered is:

```
1 select max(salary)-min(salary) as "DIFFERENCE"from My_emp;
```

The results section displays the output:

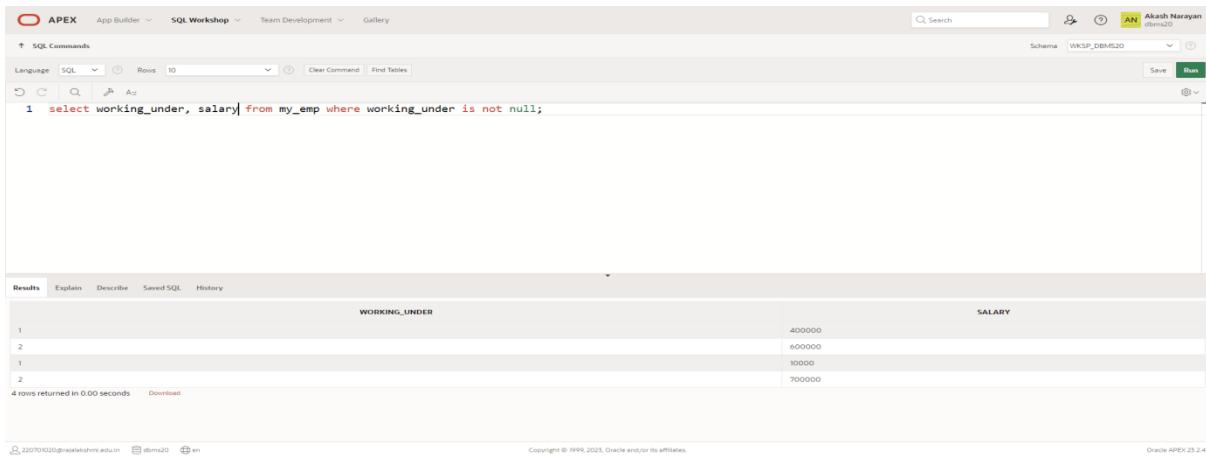
DIFFERENCE
1990000

Below the results, it says '1 rows returned in 0.00 seconds' and provides a 'Download' link. The bottom footer includes the URL '220701020@rajalakshmi.edu.in', the schema 'dbms20', and the session ID 'en'. Copyright information 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 23.2.4' are also present.

5.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 \$6000 or less. Sort the output in descending order of salary

QUERY: select working_under, min(salary) from my_emp where working_under is not null;

OUTPUT:



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

```
1 select working_under, salary from my_emp where working_under is not null;
```

The results section displays the following data:

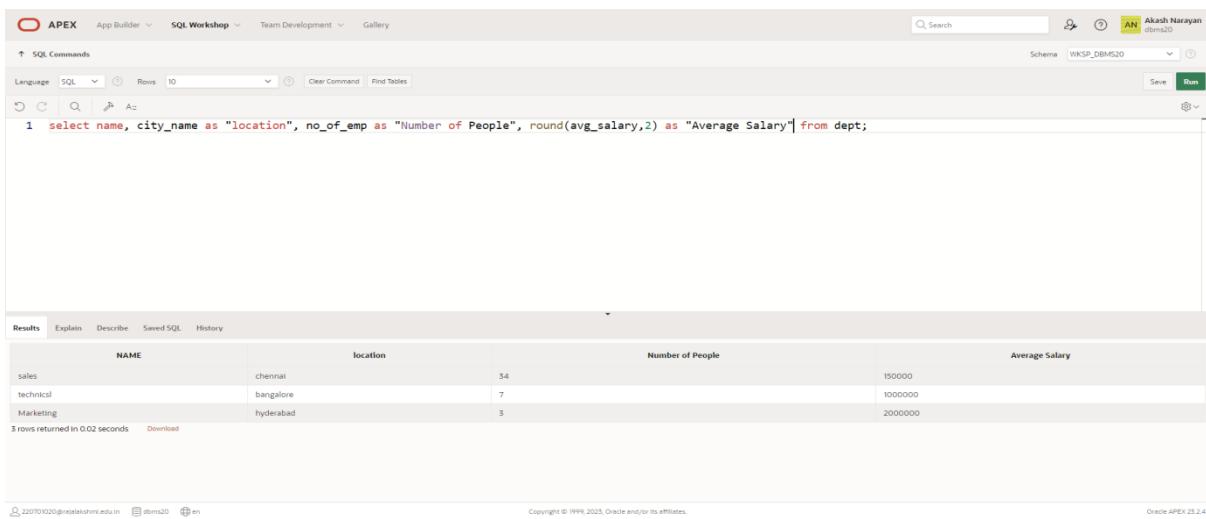
WORKING_UNDER	SALARY
1	400000
2	600000
1	10000
2	70000

4 rows returned in 0.00 seconds

6. Write a query to display each department's name, location, number of employees and the average salary for all the employees in that department. Name the column name- Location, Number of people and salary respectively. Round the average salary to 2 decimal places.

QUERY: select name, city_name as "location", no_of_emp as "Number of People", round(avg_salary,2) as "Average Salary" from dept;

OUTPUT:



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

```
1 select name, city_name as "location", no_of_emp as "Number of People", round(avg_salary,2) as "Average Salary" from dept;
```

The results section displays the following data:

NAME	location	Number of People	Average Salary
sales	chennai	34	150000
technical	bangalore	7	1000000
Marketing	hyderabad	3	2000000

3 rows returned in 0.02 seconds

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

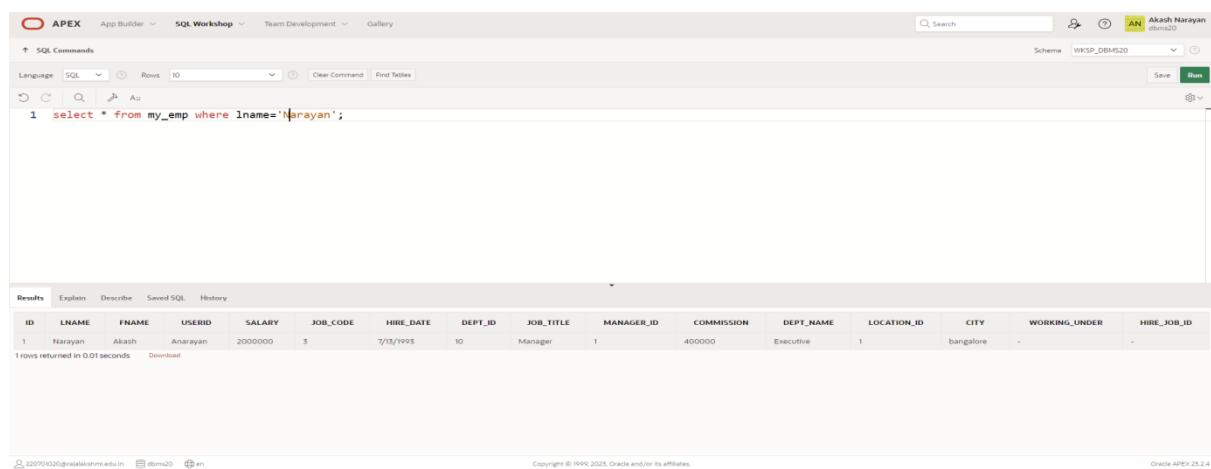
RESULT:

EXNO:9**SUB QUERIES****DATE:**

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

QUERY: select * from My_emp where lname='zlotkey';

OUTPUT:



A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar shows 'APEX' and 'SQL Workshop'. The SQL command window contains the following code:

```
1 select * from my_emp where lname='Narayan';
```

The results window shows a single row of data:

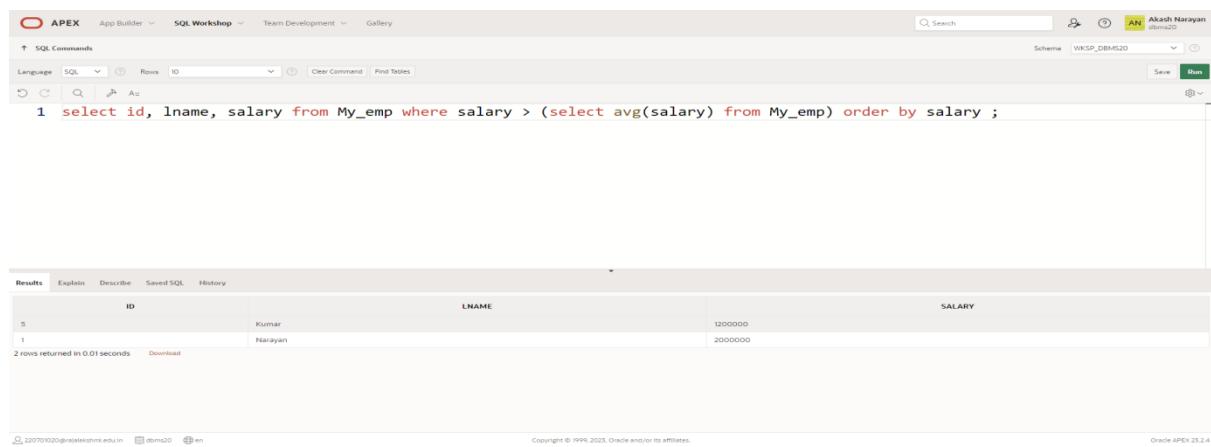
ID	LNAME	FNAME	USERID	SALARY	JOB_CODE	HIRE_DATE	DEPT_ID	JOB_TITLE	MANAGER_ID	COMMISSION	DEPT_NAME	LOCATION_ID	CITY	WORKING_UNDER	HIRE_JOB_ID
1	Narayan	Aakash	Anarayan	2000000	S	7/15/1993	10	Manager	1	400000	Executive	1	bangalore	-	-

1 rows returned in 0.01 seconds

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

QUERY: select emp_id, lname, salary from My_emp where salary > (select avg(salary) from My_emp) order by salary asc;

OUTPUT:



A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar shows 'APEX' and 'SQL Workshop'. The SQL command window contains the following code:

```
1 select id, lname, salary from My_emp where salary > (select avg(salary) from My_emp) order by salary ;
```

The results window shows two rows of data:

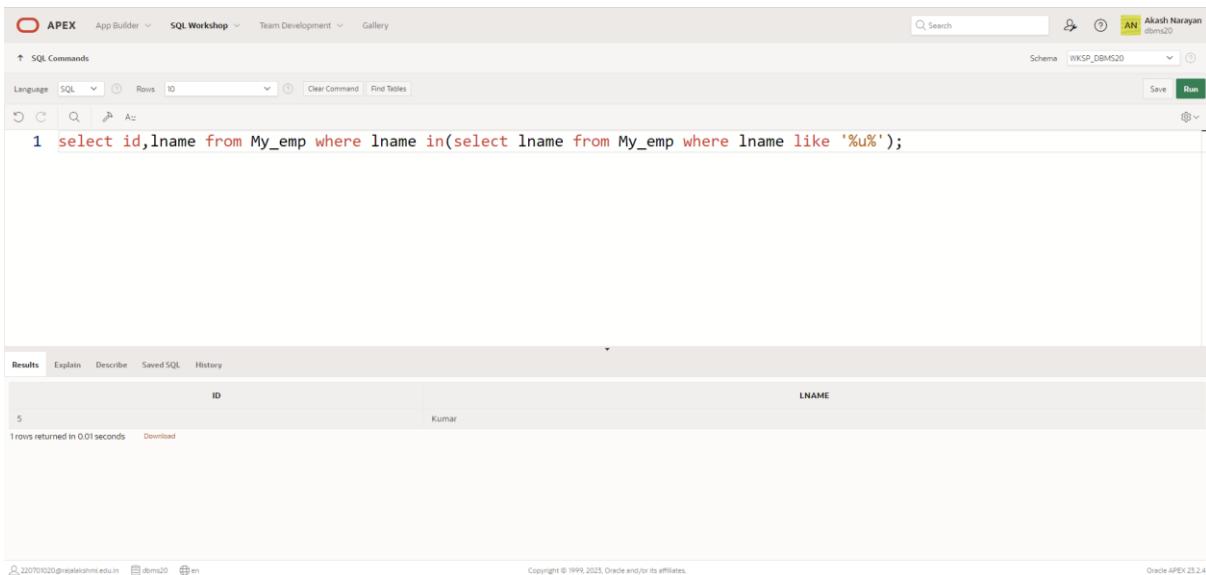
ID	LNAME	SALARY
5	Kumar	1200000
1	Narayan	2000000

2 rows returned in 0.01 seconds

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

QUERY: select emp_id, lname from My_emp where id_dept in(select id_dept from My_emp where lname like '%u%');

OUTPUT:



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

```
1 select id, lname from My_emp where lname in(select lname from My_emp where lname like '%u%');
```

The results section shows a single row:

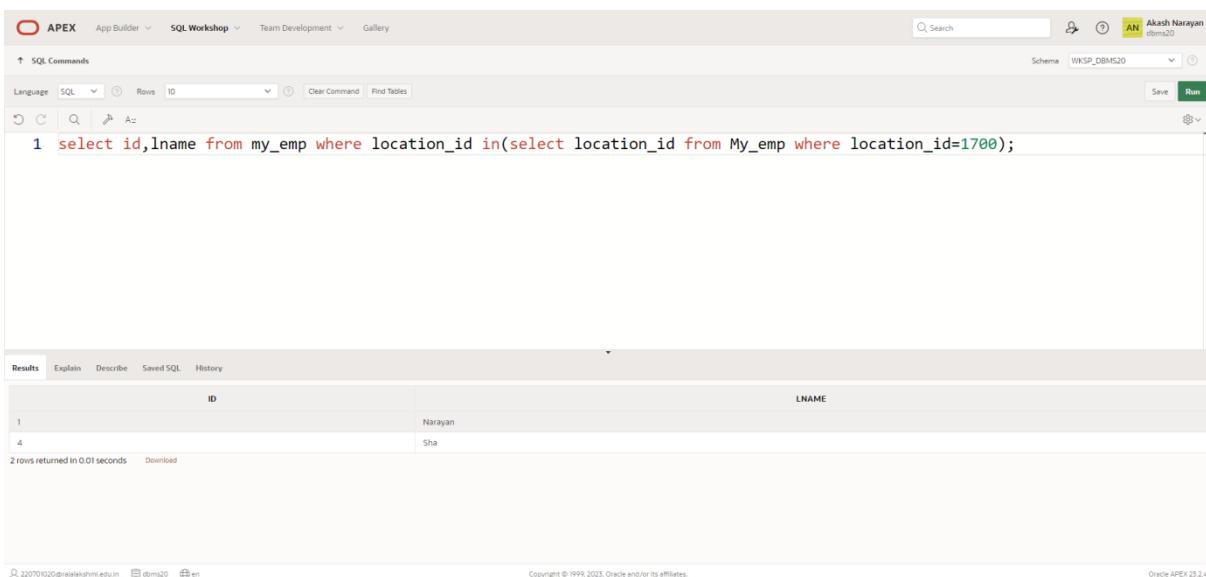
ID	LNAME
5	Kumar

1 rows returned in 0.01 seconds

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

QUERY: select emp_id, lname from my_emp where loc_id in(select loc_id from My_emp where loc_id=1700);

OUTPUT:



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

```
1 select id, lname from my_emp where location_id in(select location_id from My_emp where location_id=1700);
```

The results section shows two rows:

ID	LNAME
1	Narayan
4	Sha

2 rows returned in 0.01 seconds

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

QUERY: select emp_id, lname, id_dept from My_emp where id_dept in (select id_dept from My_emp where job_title = 'ex');

OUTPUT:

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

```
1 select dept_id, lname, job_code from my_emp where dept_name='Executive';
```

The results section displays the following data:

DEPT_ID	LNAME	JOB_CODE
80	Esh	3
10	Narayan	3
50	Gavil	9

3 rows returned in 0.01 seconds

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

QUERY: select emp_id, lname, salary from My_emp where salary > (select avg(salary) from my_emp) and id_dept in (select id_dept from My_emp where lname like '%u%');

OUTPUT:

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

```
1 from My_emp where salary > (select avg(salary) from my_emp) and lname in (select lname from My_emp where lname like '%u%');
```

The results section displays the following data:

ID	LNAME	SALARY
5	Kumar	1200000

1 rows returned in 0.02 seconds

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

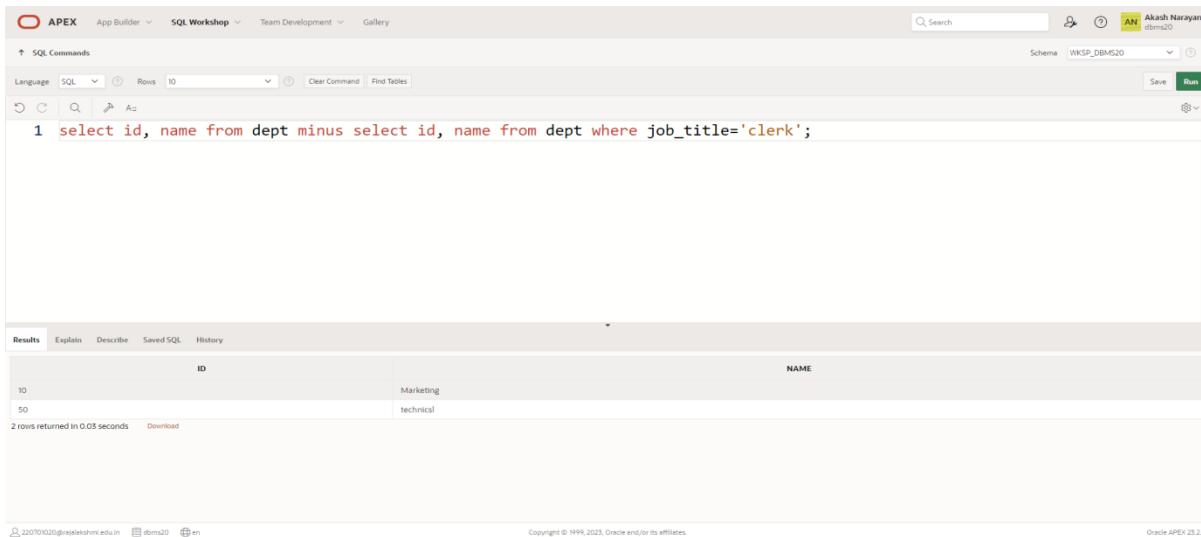
RESULT:

EXNO:10 USING THE SET OPERATORS DATE:

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

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

OUTPUT:



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

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

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

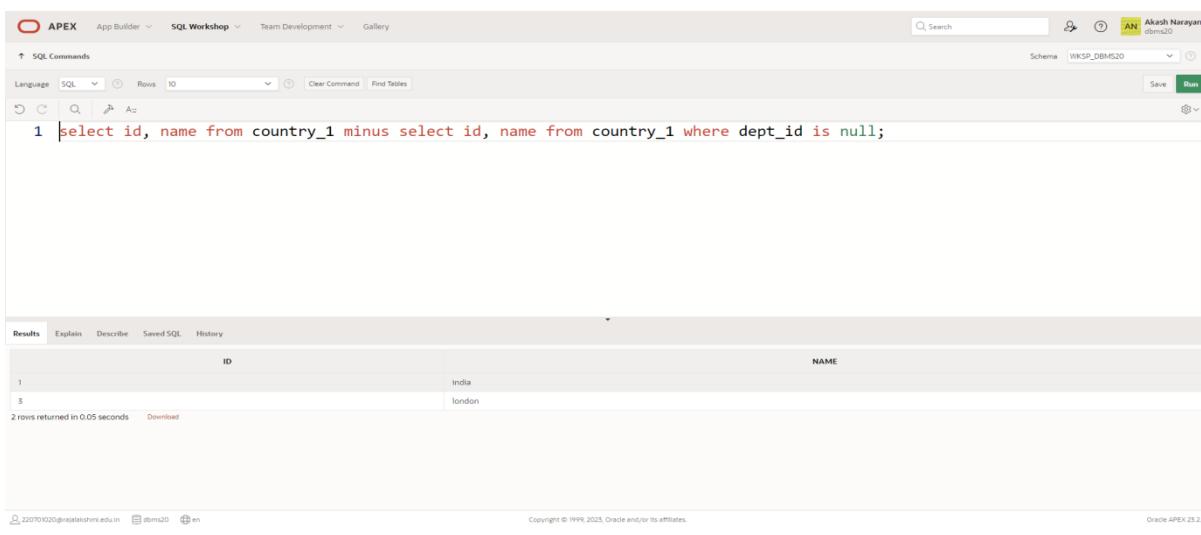
ID	NAME
10	Marketing
50	technicsl

Below the table, it says "2 rows returned in 0.03 seconds".

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

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

OUTPUT:



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

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

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

ID	NAME
1	India
3	London

Below the table, it says "2 rows returned in 0.05 seconds".

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

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

OUTPUT:

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

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

The results section displays the output:

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

6 rows returned in 0.02 seconds

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:

OUTPUT:

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

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

The results section displays the output:

ID	JOB_CODE
2	3
3	5
4	5
6	9

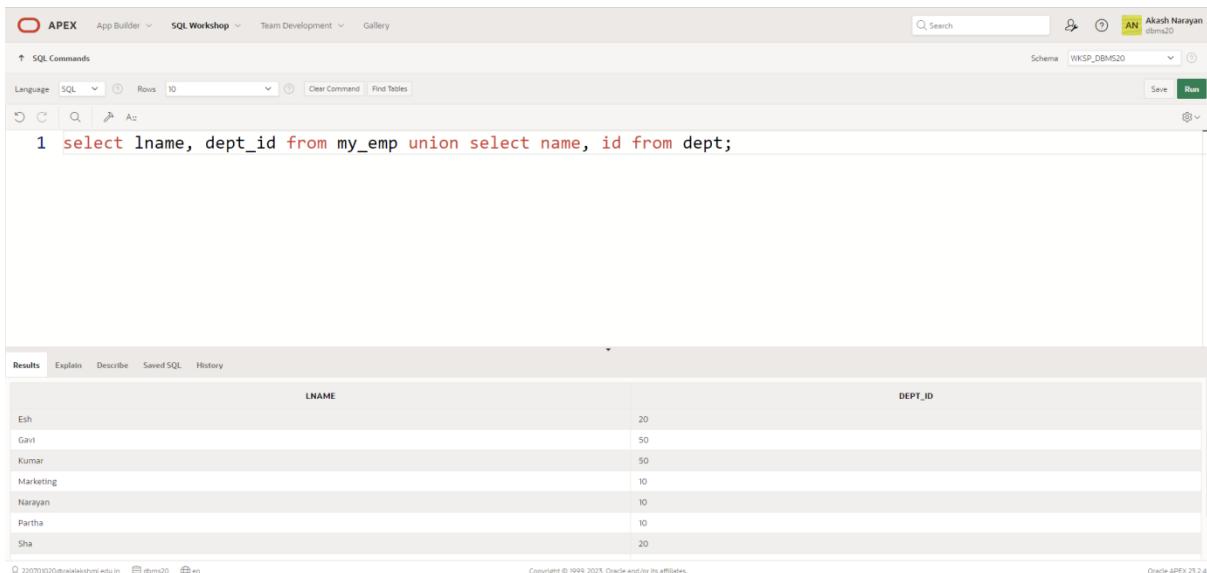
4 rows returned in 0.01 seconds

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

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

QUERY: select lname, dept_id from my_emp union select name, id from dept;

OUTPUT:



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

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

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

LNAME	DEPT_ID
Esh	20
Gavil	50
Kumar	50
Marketing	10
Narayan	10
Partha	10
Sha	20

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

RESULT:

EXNO:11

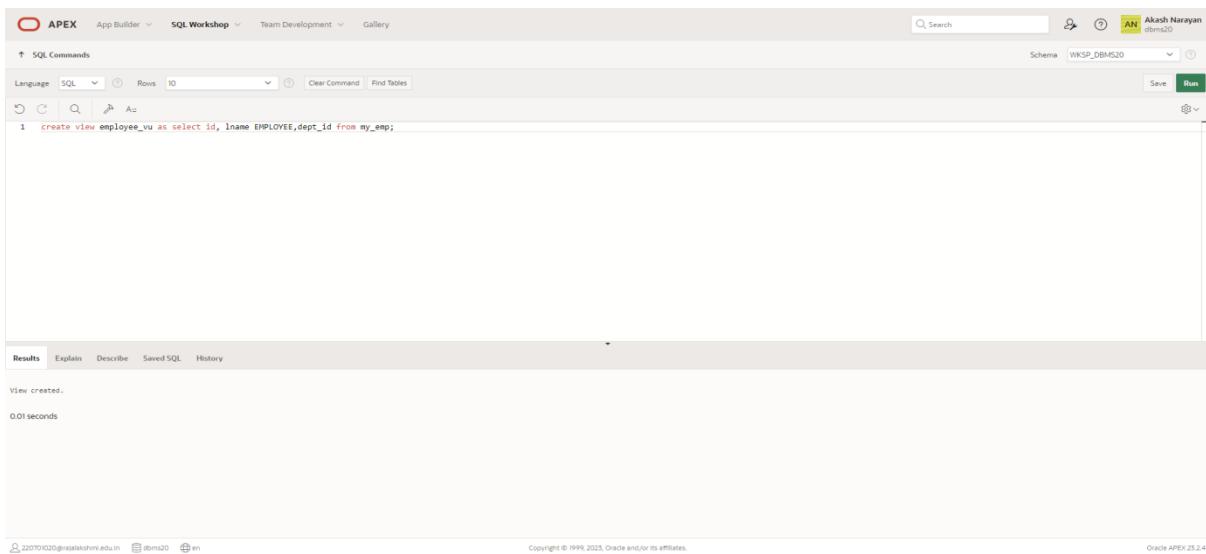
CREATING VIEWS

DATE:

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

QUERY: create view employee_vu as select id, lname EMPLOYEE,dept_id from my_emp;

OUTPUT:



A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar shows 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side of the header includes a user profile for 'Akash Narayan dbms20' and a search bar. The main workspace has a toolbar with icons for undo, redo, search, and run. Below the toolbar is a dropdown menu for 'Language' set to 'SQL', a 'Rows' dropdown set to '10', and buttons for 'Clear Command' and 'Find Tables'. The SQL command area contains the following code:

```
1 create view employee_vu as select id, lname EMPLOYEE,dept_id from my_emp;
```

The results section below shows the output of the command:

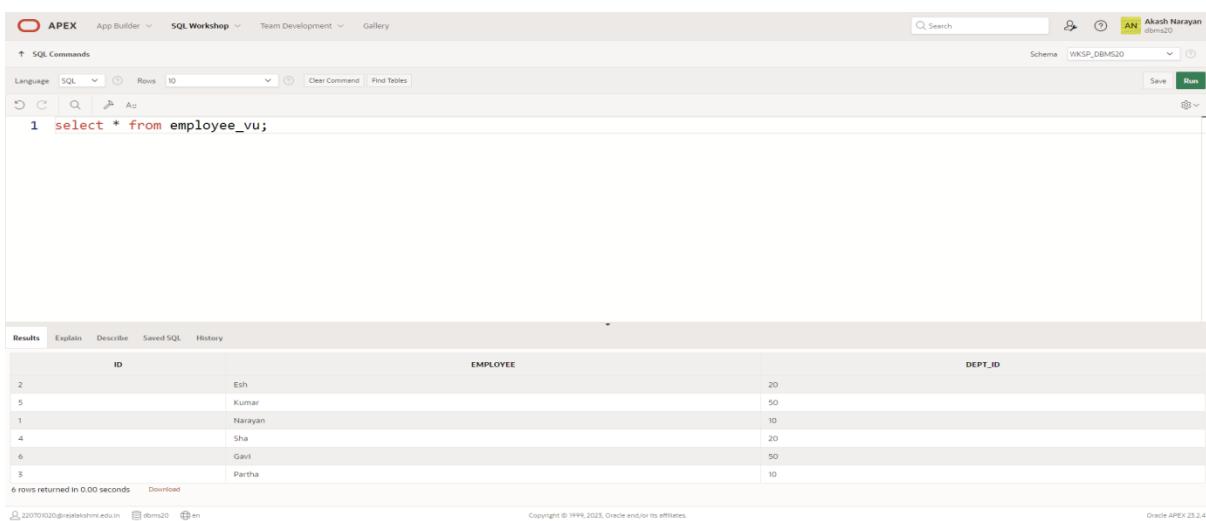
```
View created.  
0.01 seconds
```

At the bottom, the footer includes the URL '220701020@rajalakshmi.edu.in', the schema 'dbms20', and the language 'en'. It also states 'Copyright © 1999-2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

2.Display the contents of the EMPLOYEES_VU view.

QUERY: select * from employee_vu;

OUTPUT:



A screenshot of the Oracle APEX SQL Workshop interface, similar to the previous one but with a different SQL command. The top navigation bar and right sidebar are identical. The SQL command area contains the following code:

```
1 select * from employee_vu;
```

The results section shows the data returned by the query:

ID	EMPLOYEE	DEPT_ID
2	Esh	20
5	Kumar	50
1	Narayan	10
4	Sha	20
6	Gavi	50
3	Partha	10

Below the table, it says '6 rows returned in 0.00 seconds'. The footer information is identical to the first screenshot.

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

QUERY: select employee, dept_id from employee_vu;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL Commands tab contains the query: "select employee, dept_id from employee_vu;". The Results tab displays the output:

EMPLOYEE	DEPT_ID
Esh	20
Kumar	50
Narayan	10
Sha	20
Gavi	50
Partha	10

6 rows returned in 0.01 seconds

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

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

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL Commands tab contains the query: "create view dept50 as select id EMPNO, lname EMPLOYEE, dept_id DEPTNO from my_emp where dept_id=50 with read only;". The Results tab displays the output:

View created.

0.03 seconds

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

QUERY: select * from dept50;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows the user 'Akash Narayan' and the schema 'WKSP_DBMS20'. The main area has tabs for 'SQL Commands' and 'Results'. The SQL command entered is 'select * from dept50;'. The results section displays a table with four rows:

EMPNO	EMPLOYEE	DEPTNO
5	Kumar	50
1	Narayan	50
6	Gavi	50
3	Partha	50

Below the table, it says '4 rows returned in 0.01 seconds' and there is a 'Download' link. At the bottom, it shows the URL '220701020@rajaikshmi.edu.in', the schema 'dbms20', and the language 'en'. Copyright information and the text 'Oracle APEX 23.2.4' are also present.

6. Attempt to reassign Matos to department 80.

QUERY: update dept50 set deptno=80 where employee='Matos';

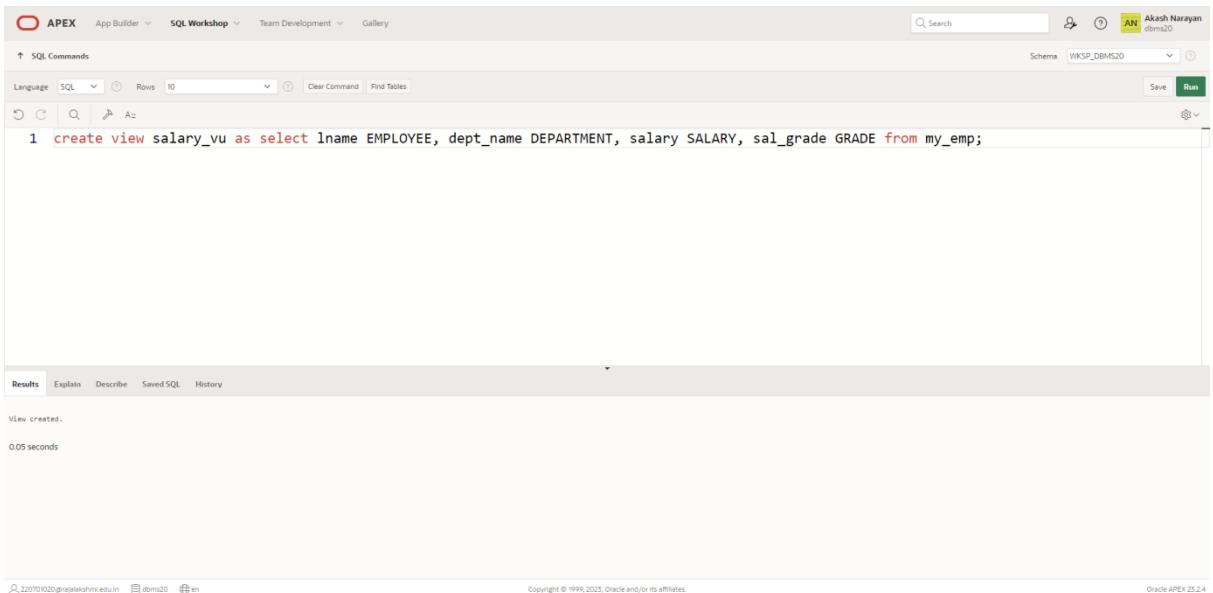
OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface, similar to the previous one. The top navigation bar and user information are the same. The main area has tabs for 'SQL Commands' and 'Results'. The SQL command entered is 'update dept50 set deptno=80 where employee='Matos';'. In the results section, a yellow box highlights the error message: 'Error at line 1/19: ORA-42399: cannot perform a DML operation on a read-only view'. Below the error message, the command is repeated: '1. update dept50 set deptno=80 where employee='Matos';'. At the bottom, it shows the URL '220701020@rajaikshmi.edu.in', the schema 'dbms20', and the language 'en'. Copyright information and the text 'Oracle APEX 23.2.4' are also present.

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

QUERY: create view salary_vu as select lname EMPLOYEE, dept_name DEPARTMENT, salary SALARY, sal_grade GRADE from my_emp;

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is chosen from a dropdown menu. The main area is titled 'SQL Commands'. A single line of SQL code is entered in the command input field: 'create view salary_vu as select lname EMPLOYEE, dept_name DEPARTMENT, salary SALARY, sal_grade GRADE from my_emp;'. The 'Run' button is visible at the bottom right of the command input field. Below the command input, there is a 'Results' tab which is currently selected. The results pane displays the message 'View created.' and '0.05 seconds' execution time. At the bottom of the page, there is footer information including the URL '220701020@rajalakshmi.edu.in', the schema 'dbms20', and the language 'en'. The copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 23.2.4' are also present.

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

RESULT:

INTRODUCTION TO CONSTRAINTS: NOT NULL AND UNIQUE CONSTRAINTS

EXNO:12

DATE:

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

Ans: A constraint is a criteria or a condition based on which a certain action is done.

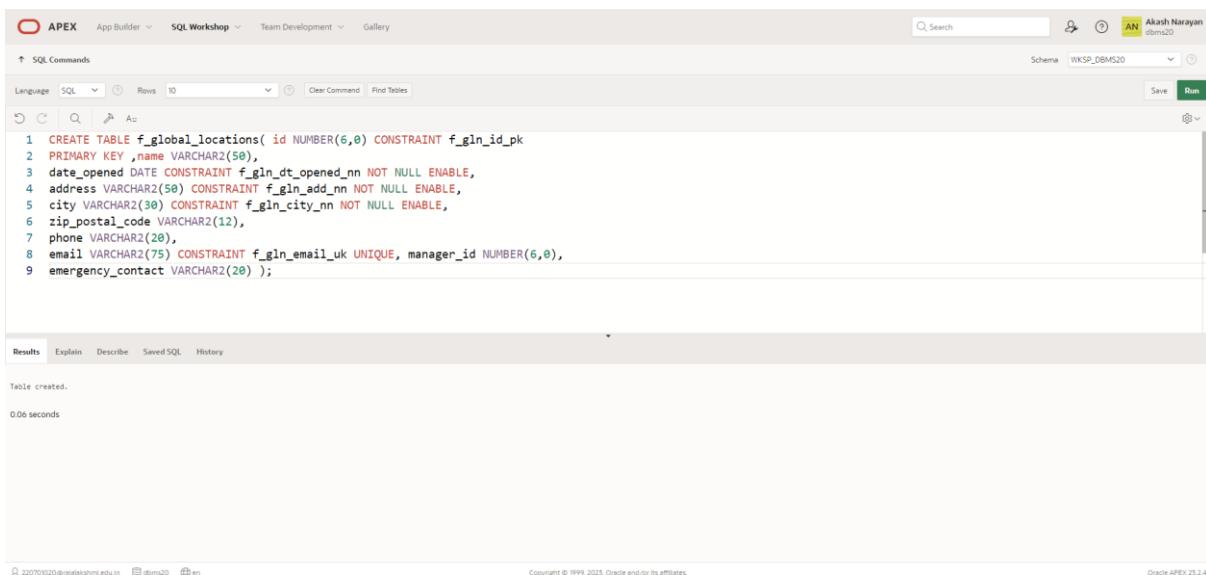
2. Why is it important to give meaningful names to constraints?

Ans: To understand the data easily and access it faster.

3. Write the create table statement for the Global fast foods locations table to define the constraints at the column level.

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below the bar, there are tabs for 'SQL Commands' and 'Results'. The 'Results' tab is active, showing the output of the SQL command. The command itself is displayed in the SQL Commands pane. The output pane shows the message 'Table created.' and '0.00 seconds'. The bottom of the screen displays the Oracle APEX footer with copyright information.

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

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

QUERY: describe f_global_locations;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. At the top, there are tabs for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. Below the tabs, there's a search bar and user information for 'Akash Narayan' and 'Schema WKSP_DBMS20'. The main area has tabs for SQL Commands, Language (SQL selected), Rows (10), Clear Command, and Find Tables. A 'Run' button is also present. The SQL editor contains the command: '1 describe f_global_locations; 2 |'. Below the editor, the results tab is selected, showing the table structure for 'F_GLOBAL_LOCATIONS'. The columns listed are ID, NAME, DATE_OPENED, ADDRESS, CITY, ZIP_POSTAL_CODE, and PHONE. The table structure is as follows:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
F_GLOBAL_LOCATIONS	ID	NUMBER	6	0	1	✓	+	+	+
	NAME	VARCHAR2	50	-	-	✓	+	+	+
	DATE_OPENED	DATE	7	-	-	+	+	+	+
	ADDRESS	VARCHAR2	50	-	-	+	+	+	+
	CITY	VARCHAR2	30	-	-	+	+	+	+
	ZIP_POSTAL_CODE	VARCHAR2	12	-	-	+	✓	+	+
	PHONE	VARCHAR2	20	-	-	+	✓	+	+

5. 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.

QUERY:

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

PRIMARY KEY, FOREIGN KEY AND CHECK CONSTRAINTS

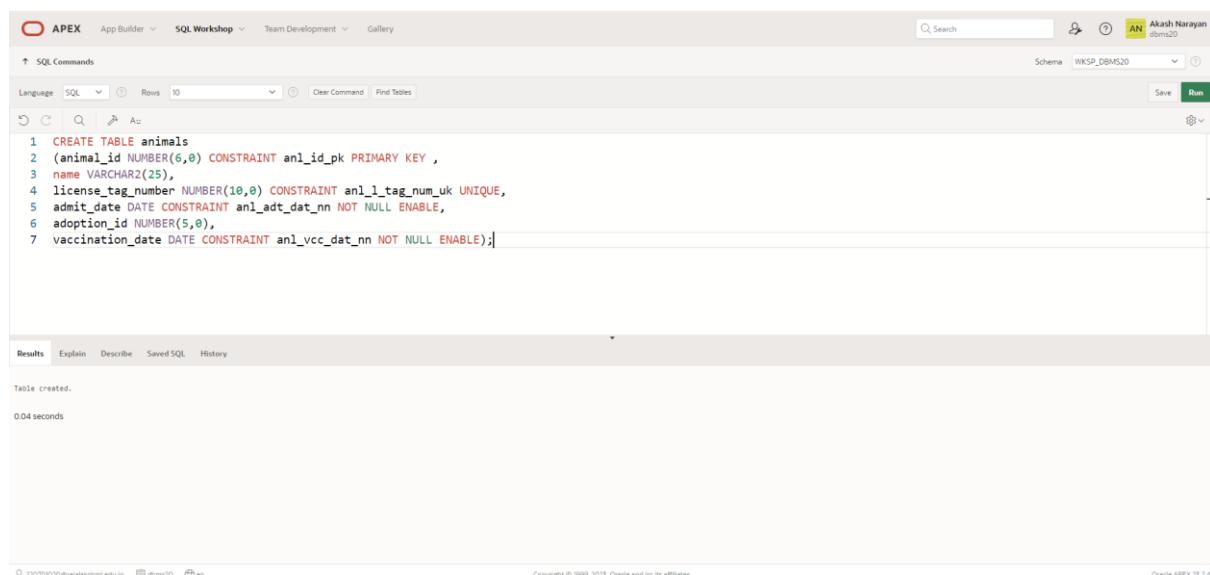
1. What is the purpose of a

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

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

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

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is chosen from a dropdown menu. The main area is titled 'SQL Commands'. A code editor window contains the following SQL code:

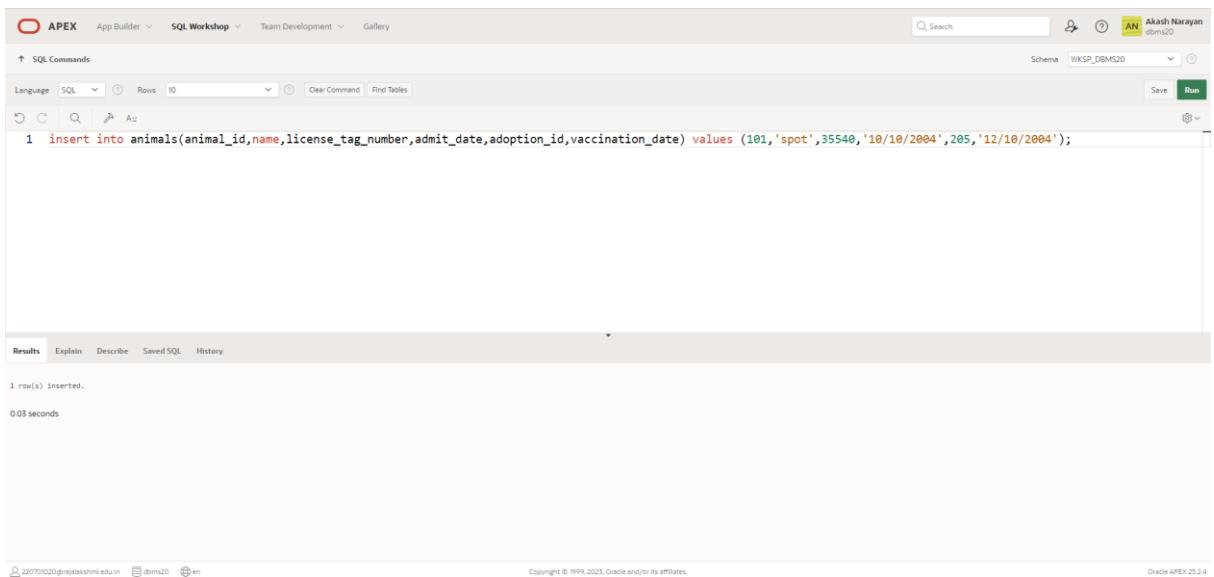
```
1 CREATE TABLE animals
2 (animal_id NUMBER(6,0) CONSTRAINT anl_id_pk PRIMARY KEY ,
3 name VARCHAR2(25),
4 license_tag_number NUMBER(10,0) CONSTRAINT anl_l_tag_num_uk UNIQUE,
5 admit_date DATE CONSTRAINT anl_adt_dat_nn NOT NULL ENABLE,
6 adoption_id NUMBER(5,0),
7 vaccination_date DATE CONSTRAINT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

Below the code editor, the 'Results' tab is selected. The output pane displays the message 'Table created.' and '0.04 seconds'. At the bottom of the page, there are footer links for 'Copyright © 1999, 2023, Oracle and/or its affiliates.', 'Oracle APEX 23.2.4', and a URL '22070020@nptekshnm.edu.in'.

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

QUERY:insert into
animals(animal_id,name,license_tag_number,admit_date,adoption_id,
vaccination_date) values (101,'spot',35540,'10/10/2004',205,'12/10/2004');

OUTPUT:



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

```
1 insert into animals(animal_id,name,license_tag_number,admit_date,adoption_id,vaccination_date) values (101,'spot',35540,'10/10/2004',205,'12/10/2004');
```

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

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

Execution time is listed as 0.03 seconds. At the bottom of the results pane, there are copyright notices for Oracle and APEX.

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

Ans: If you define a CHECK constraint on a column it will allow only certain values for this column. If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

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

RESULT:

CREATING VIEWS

EXNO:13

DATE:

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

Ans: 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.

**QUERY: CREATE OR REPLACE 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';**

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 top right shows the user 'Akash Narayan dbms20' and a save button. The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the code for creating the view is entered:

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

Under the 'Results' tab, the output shows:

```
View created.
0.04 seconds
```

At the bottom, footer information includes the URL '22070102@relaxshrm.edu.in', the schema 'dbms20', the language 'en', and copyright notice 'Copyright © 1999-2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

3. SELECT * FROM view_d_songs. What was returned?

QUERY: SELECT * FROM view_d_songs

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL Commands tab is active, displaying the query: "SELECT * FROM view_d_songs;". The Results tab shows the output of the query:

ID	Song Title	ARTIST
3	rockonharris	harris(jayara)
1	marakkuma	ar rahman

2 rows returned in 0.01 seconds

4. REPLACE view_d_songs. Add type_code to the column list. Use aliases for all columns.

QUERY: 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 APEX SQL Workshop interface. The SQL Commands tab is active, displaying the query:

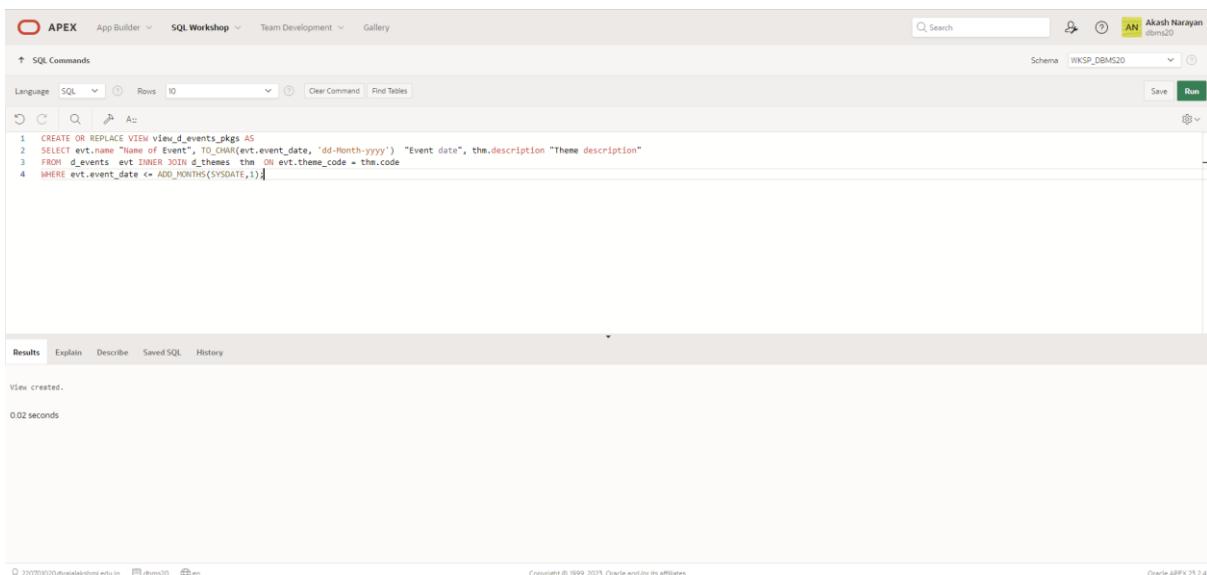
```
1 CREATE OR REPLACE VIEW view_d_songs AS
2  SELECT d_songs.id, d_songs.title "Song Title", d_songs.artist, d_songs.type_code
3  from d_songs INNER JOIN d_types ON d_songs.type_code = d_types.code
4  where d_types.description = 'New Age';
```

The Results tab shows the output: "View created." and "0.03 seconds".

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

QUERY: 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. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is active. The main area is titled 'SQL Commands'. The code entered is:

```
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);
```

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

```
View created.
0.02 seconds
```

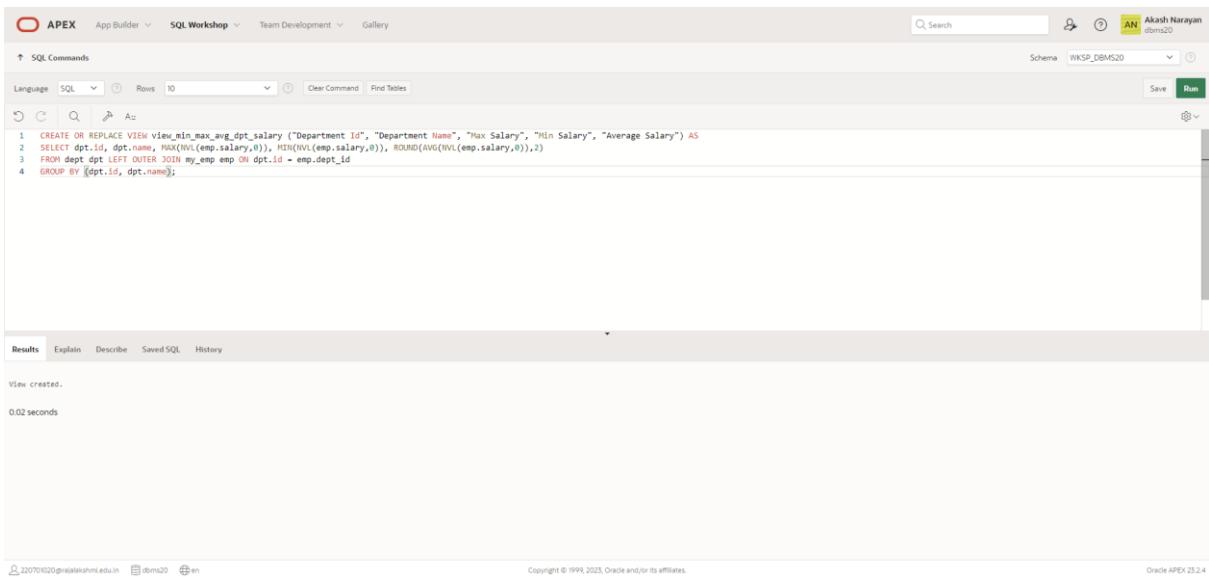
At the bottom of the page, footer information includes:

Copyright © 1999-2023, Oracle and/or its affiliates.
Oracle APEX 23.2.4
220705@relakshmi.edu.in dbms20 en

6. It is company policy that only upper-level management be allowed access 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.

QUERY: CREATE OR REPLACE VIEW view_min_max_avg_dpt_salary ("Department Id", "Department Name", "Max Salary", "Min Salary", "Average Salary") AS
SELECT dpt.id, dpt.name, MAX(NVL(emp.salary,0)), MIN(NVL(emp.salary,0)),
ROUND(AVG(NVL(emp.salary,0)),2)
FROM dept dpt LEFT OUTER JOIN my_emp emp ON dpt.id = emp.dept_id
GROUP BY (dpt.id, dpt.name);

OUTPUT:



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

```
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.id, dpt.name, MAX(NVL(emp.salary,0)), MIN(NVL(emp.salary,0)), ROUND(AVG(NVL(emp.salary,0)),2)
3 FROM dept dpt LEFT OUTER JOIN my_emp emp ON dpt.id = emp.dept_id
4 GROUP BY (dpt.id, dpt.name);
```

Below the code, the 'Results' tab is selected. The output message 'View created.' is displayed, along with a timestamp '0.02 seconds'. At the bottom of the page, there are footer links for '22070020@rajalakshmi.edu.in', 'dbms20', 'en', 'Copyright © 1999, 2023, Oracle and/or its affiliates.', and 'Oracle APEX 23.2.4'.

DML OPERATIONS AND VIEWS

1. 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.

**QUERY: CREATE OR REPLACE VIEW view_copy_d_songs AS
SELECT *
FROM copy_d_songs;**

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is chosen from the dropdown menu. The main area is titled 'SQL Commands'. The schema is set to 'WKSP_DBMS20'. The code entered is:

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

After running the command, the results show:

View created.
0.03 seconds

At the bottom, the copyright notice reads: Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4.

2. 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.

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

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is chosen from the dropdown menu. The main area is titled 'SQL Commands'. The schema is set to 'WKSP_DBMS20'. The code entered is:

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

After running the command, the results show:

1 row(s) inserted.
0.01 seconds

At the bottom, the copyright notice reads: Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4.

3.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.

**QUERY: 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;

OUTPUT:

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

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

Below the code, the 'Results' tab is active, showing the message 'View created.' and a execution time of '0.03 seconds'. The bottom status bar indicates the session is connected to '220701020@readishm.edu.in' on 'dbms20' with language 'en'.

4. Using the read_copy_d_cds view, execute a DELETE FROM read_copy_d_cds WHERE cd_number = 90;

QUERY: DELETE FROM read_copy_d_cds WHERE cd_number = 90;

OUTPUT:

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

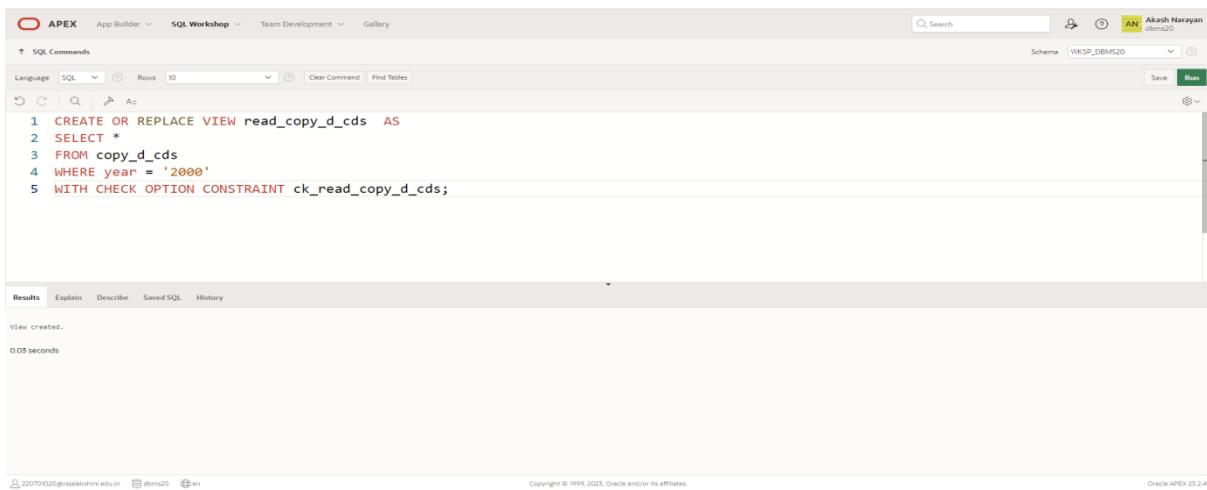
```
DELETE FROM read_copy_d_cds WHERE cd_number = 90;
```

A yellow box highlights the error message: 'Error at line 1/13: ORA-42399: cannot perform a DML operation on a read-only view'. The bottom status bar indicates the session is connected to '220701020@readishm.edu.in' on 'dbms20' with language 'en'.

5. 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.

QUERY: 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;

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Commands' is chosen from the dropdown menu. The main area contains the following SQL code:

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

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

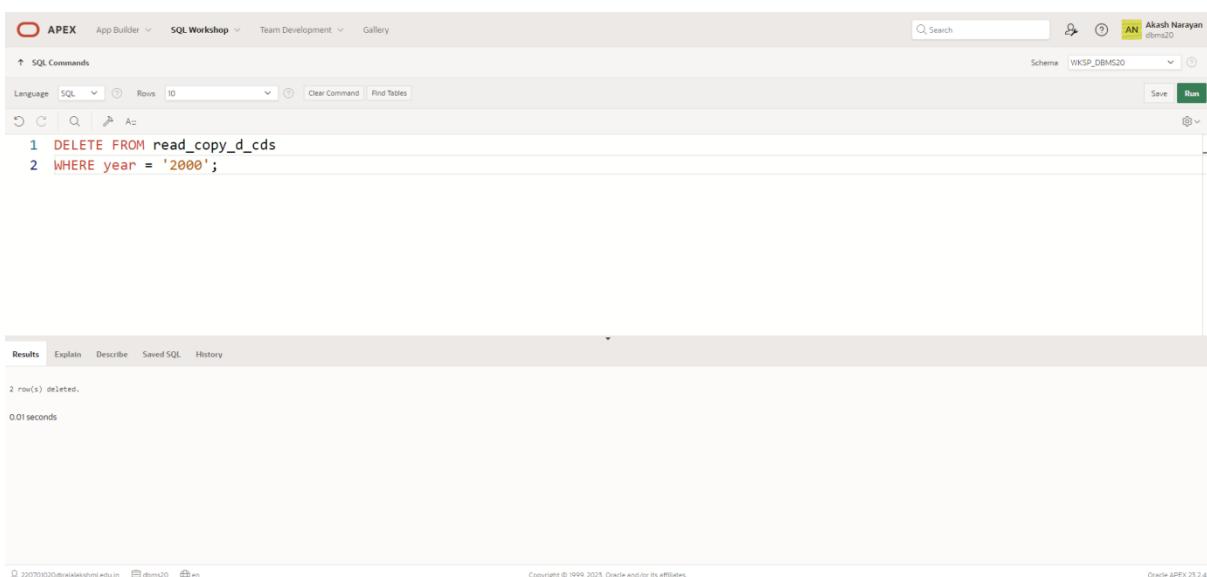
View created.
0.05 seconds

At the bottom of the interface, the footer includes the user's email (220701020@rajalakshmi.edu.in), the schema (dbms20), and the language (en). It also displays the copyright information (Copyright © 1999-2023, Oracle and/or its affiliates) and the version (Oracle APEX 23.2.4).

6. Use the read_copy_d_cds view to delete any CD of year 2000 from the underlying copy_d_cds.

QUERY: DELETE FROM read_copy_d_cds
WHERE year = '2000';

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Commands' is chosen from the dropdown menu. The main area contains the following SQL code:

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

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

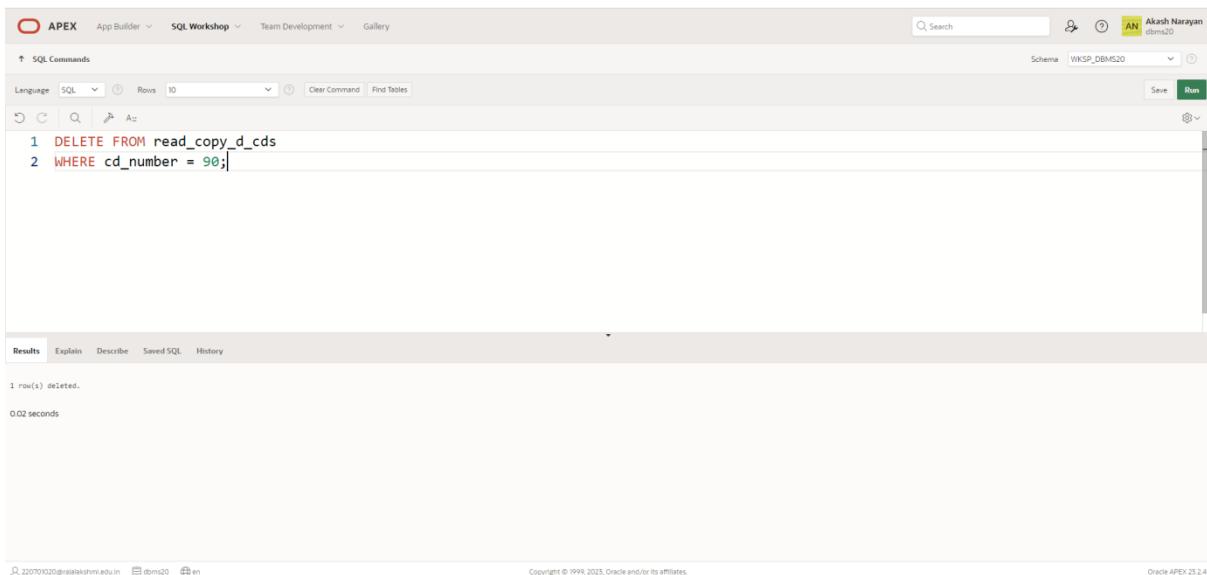
2 row(s) deleted.
0.01 seconds

At the bottom of the interface, the footer includes the user's email (220701020@rajalakshmi.edu.in), the schema (dbms20), and the language (en). It also displays the copyright information (Copyright © 1999-2023, Oracle and/or its affiliates) and the version (Oracle APEX 23.2.4).

7. Use the read_copy_d_cds view to delete cd_number 90 from the underlying copy_d_cds table.

**QUERY: DELETE FROM read_copy_d_cds
WHERE cd_number = 90;**

OUTPUT:



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

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

The Results tab shows the output of the query:

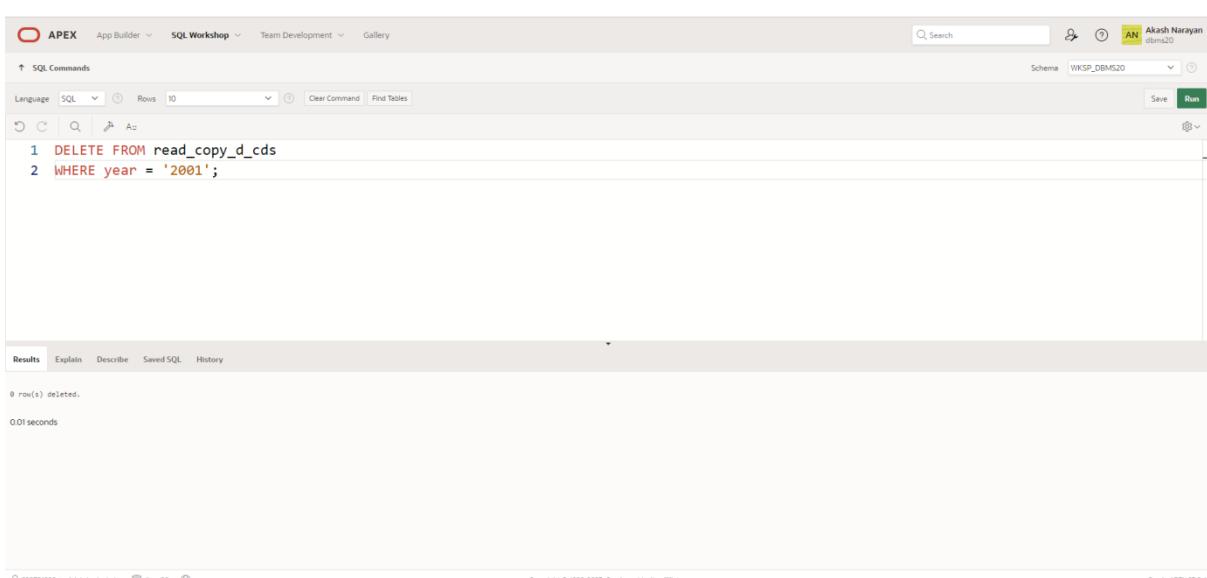
```
1 row(s) deleted.
0.02 seconds
```

At the bottom, the footer includes the URL 220701020@rajalakshmi.edu.in, the schema dbms20, the language en, and the copyright notice Copyright © 1999-2023, Oracle and/or its affiliates. Oracle APEX 23.2.4.

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

**QUERY: DELETE FROM read_copy_d_cds
WHERE year = '2001';**

OUTPUT:



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

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

The Results tab shows the output of the query:

```
0 row(s) deleted.
0.01 seconds
```

At the bottom, the footer includes the URL 220701020@rajalakshmi.edu.in, the schema dbms20, the language en, and the copyright notice Copyright © 1999-2023, Oracle and/or its affiliates. Oracle APEX 23.2.4.

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.

**QUERY: CREATE OR REPLACE VIEW view_copy_d_songs AS
SELECT title, artist
FROM copy_d_songs;**

OUTPUT:

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

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

Below the command, the results pane displays the output of the query:

TITLE	ARTIST
highonull	yuvan
rockonharris	harrispayalaj
marakkuma	ar rahman
Mello Jello	The What

Text at the bottom of the results pane: "4 rows returned in 0.01 seconds" and "Download".

2. Issue a DROP view_copy_d_songs. Execute a SELECT * statement to verify that the view has been deleted.

QUERY: DROP VIEW view_copy_d_songs;

OUTPUT:

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

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

Below the command, the results pane displays an error message:

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

Text at the bottom of the results pane: "Copyright © 1999, 2023, Oracle and/or its affiliates." and "Oracle APEX 23.2.4".

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

**QUERY: SELECT * FROM
(SELECT lname, salary FROM my_emp ORDER BY salary DESC)
WHERE ROWNUM <= 3;**

OUTPUT:

```
1 SELECT * FROM
2 (SELECT lname, salary FROM my_emp ORDER BY salary DESC)
3 WHERE ROWNUM <= 3;
```

LNAME	SALARY
Narayan	2000000
Kumar	1200000
Partha	700000

3 rows returned in 0.01 seconds Download

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.

**QUERY: SELECT empm.lname, empm.salary, dptmx.id
FROM
(SELECT dpt.id, MAX(NVL(emp.salary,0)) max_dpt_sal
FROM dept dpt LEFT OUTER JOIN my_emp emp ON dpt.id = emp.dept_id
GROUP BY dpt.id) dptmx LEFT OUTER JOIN my_emp empm ON dptmx.id =
empm.dept_id
WHERE NVL(empm.salary,0) = dptmx.max_dpt_sal;**

OUTPUT:

```
1 SELECT empm.lname, empm.salary, dptmx.id
2 FROM
3 (SELECT dpt.id, MAX(NVL(emp.salary,0)) max_dpt_sal
4 FROM dept dpt LEFT OUTER JOIN my_emp emp ON dpt.id = emp.dept_id
5 GROUP BY dpt.id) dptmx LEFT OUTER JOIN my_emp empm ON dptmx.id = empm.dept_id
6 WHERE NVL(empm.salary,0) = dptmx.max_dpt_sal;
```

LNAME	SALARY	ID
Narayan	2000000	50
Sha	600000	20
Partha	700000	10

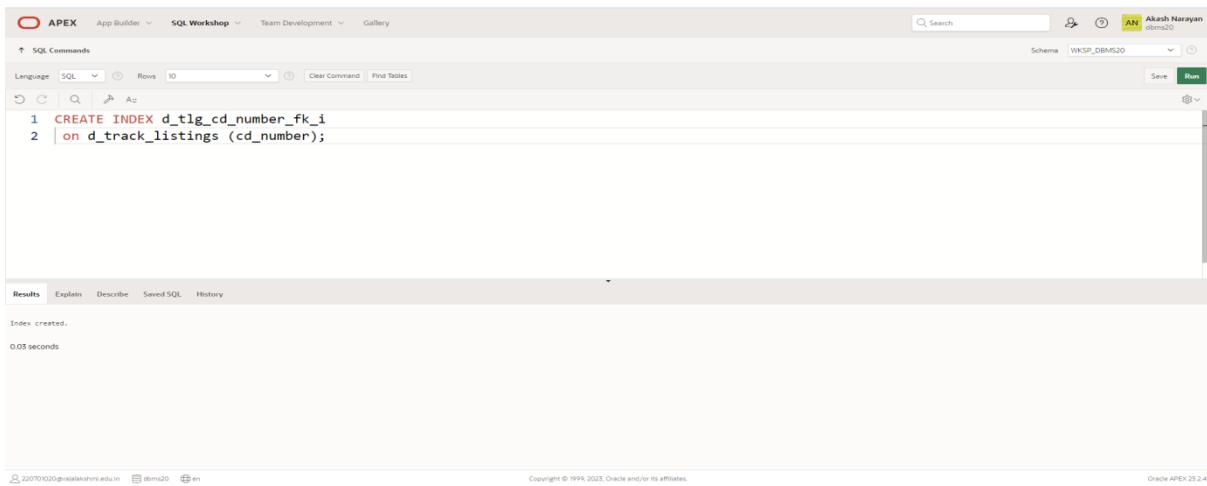
3 rows returned in 0.01 seconds Download

INDEXES AND SYNONYMS

1. 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.

**QUERY: CREATE INDEX d_tlg_cd_number_fk_i
on d_track_listings (cd_number);**

OUTPUT:



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

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

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

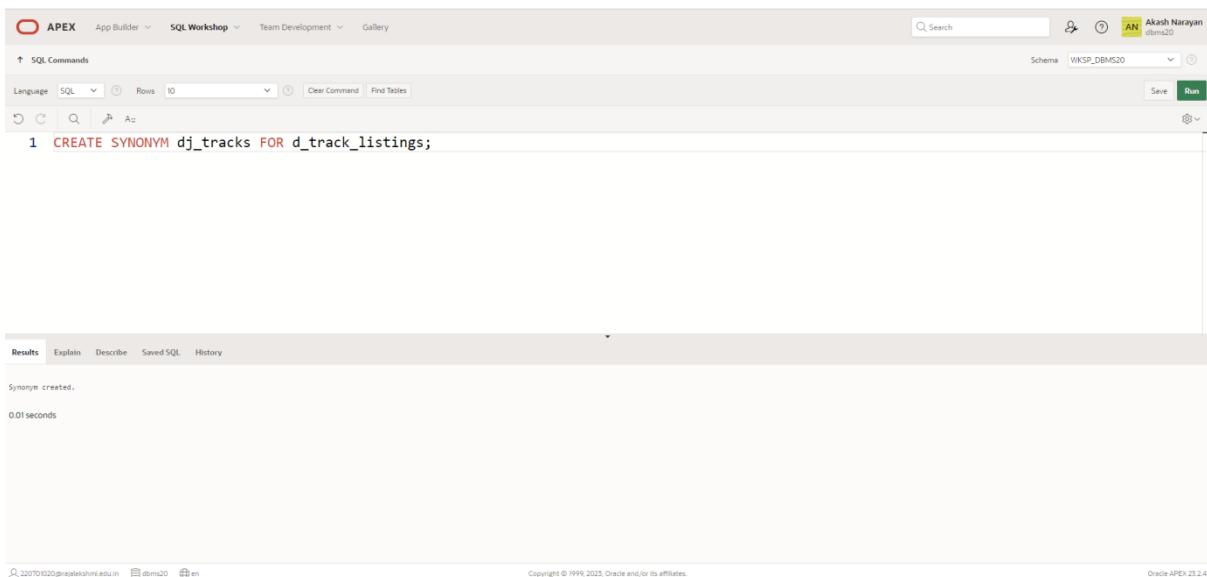
```
Index created.
0.03 seconds
```

At the bottom, the status bar shows the user's email (220701020@rejaliakshmi.edu.in), schema (dbms20), and language (en).

2. Write a query to create a synonym called dj_tracks for the DJs on Demand d_track_listings table.

QUERY: CREATE SYNONYM dj_tracks FOR d_track_listings;

OUTPUT:



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

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

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

```
Synonym created.
0.01 seconds
```

At the bottom, the status bar shows the user's email (220701020@rejaliakshmi.edu.in), schema (dbms20), and language (en).

3. Create a synonym for the D_TRACK_LISTINGS table. Confirm that it has been created by querying the data dictionary.

QUERY: CREATE SYNONYM dj_tracks2 FOR d_track_listings;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top right corner, the user 'Akash Narayan' is logged in. The schema dropdown is set to 'WKSP_DBMS20'. The SQL command window contains the following SQL statement:

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

Below the command window, the results tab is selected. The output shows:

```
Synonym created.
```

Execution time: 0.02 seconds.

At the bottom of the interface, the URL '220701020@rajalakshmi.edu.in', the schema 'dbms20', and the language 'en' are displayed. The footer also includes the copyright notice 'Copyright © 1994-2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 23.2.4'.

4. Drop the synonym that you created in question

QUERY: DROP SYNONYM dj_tracks2;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top right corner, the user 'Akash Narayan' is logged in. The schema dropdown is set to 'WKSP_DBMS20'. The SQL command window contains the following SQL statement:

```
1 DROP SYNONYM dj_tracks2;
```

Below the command window, the results tab is selected. The output shows:

```
Synonym dropped.
```

Execution time: 0.02 seconds.

At the bottom of the interface, the URL '220701020@rajalakshmi.edu.in', the schema 'dbms20', and the language 'en' are displayed. The footer also includes the copyright notice 'Copyright © 1994-2023, 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:

OTHER DATABASE OBJECTS

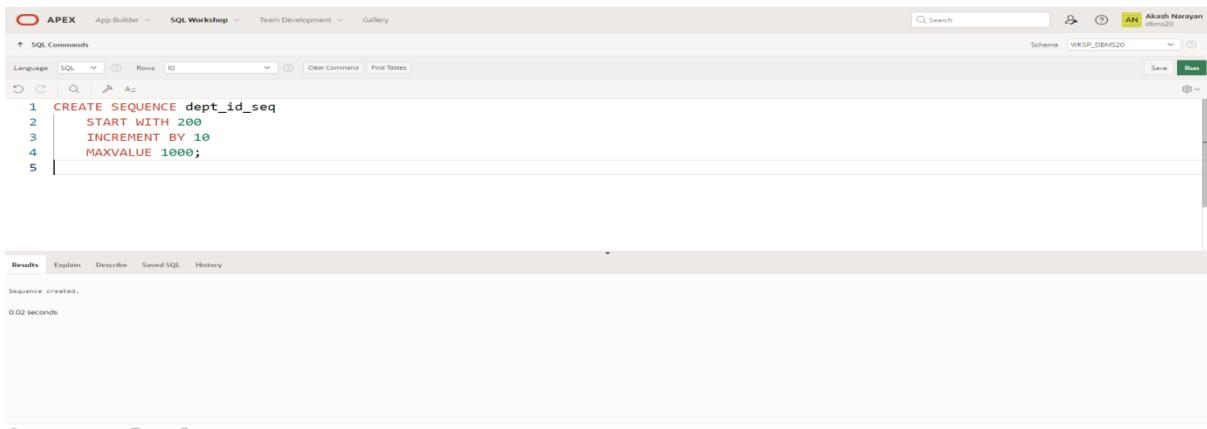
EXNO:14

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. In the SQL Commands tab, the following SQL code is entered:

```
1 CREATE SEQUENCE dept_id_seq
2   START WITH 200
3   INCREMENT BY 10
4   MAXVALUE 1000;
5
```

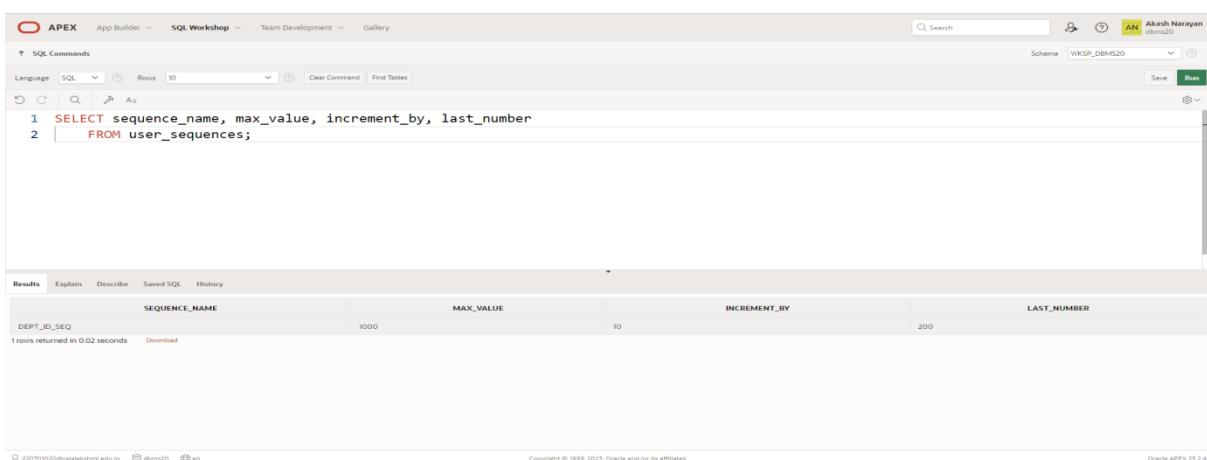
After running the command, the Results tab displays the output:

```
Sequence created.
0.02 seconds
```

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

**QUERY: SELECT sequence_name, max_value, increment_by, last_number
FROM user_sequences;**

OUTPUT:



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

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

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

SEQUENCE_NAME	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
DEPT_ID_SEQ	1000	10	200

1 rows returned in 0.02 seconds

3. Write a script to insert two rows into the DEPT table. Name your script lab12_3.sql. Be sure to use the sequence that you created for the ID column. Add two departments named Education and Administration. Confirm your additions. Run the commands in your script.

**QUERY: INSERT INTO dept
VALUES (dept_id_seq.nextval, 'Education');
INSERT INTO dept
VALUES (dept_id_seq.nextval, 'Administration');**

OUTPUT:

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

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

An error message is displayed in a yellow box:

Error at line 1/13: ORA-00947: not enough values
1. INSERT INTO dept
2. VALUES (dept_id_seq.nextval, 'Education');

The Results pane shows the execution time as 0.00 seconds.

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

QUERY: CREATE INDEX emp_dept_id_idx ON my_emp (dept_id);

OUTPUT:

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

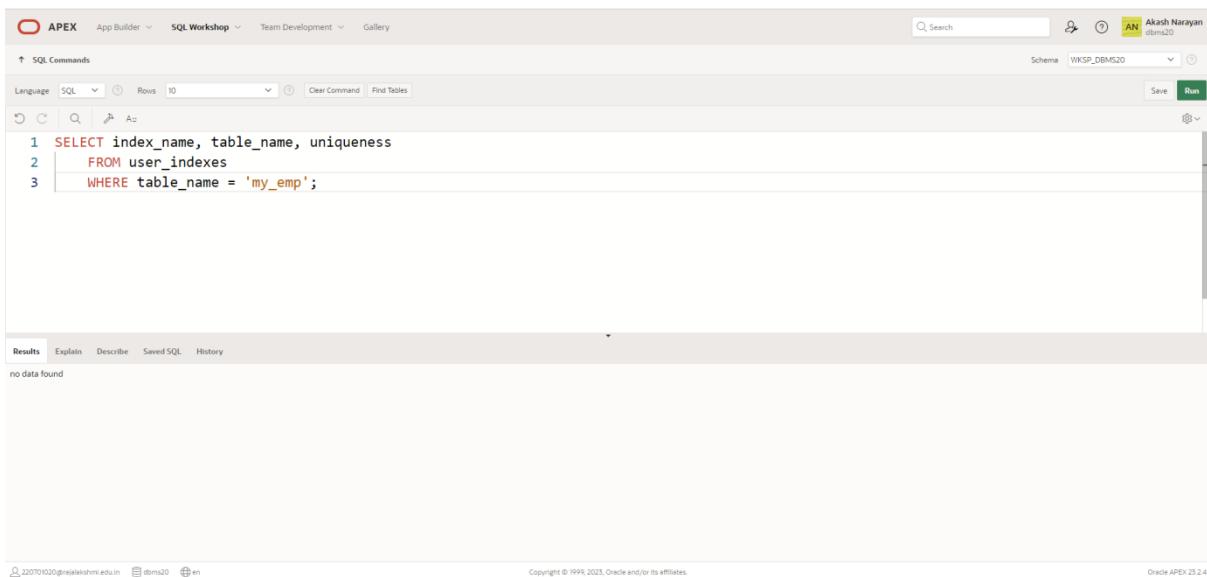
```
1 CREATE INDEX emp_dept_id_idx ON my_emp (dept_id);
```

The Results pane shows the message "Index created." and the execution time as 0.03 seconds.

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

QUERY: `SELECT index_name, table_name, uniqueness
FROM user_indexes
WHERE table_name = 'my_emp';`

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows the user profile 'Akash Narayan dbms20'. The main area has tabs for SQL Commands, SQL (selected), Rows (10), Clear Command, and Find Tables. The SQL editor contains the following code:

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

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

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

RESULT:

CONTROLLING USER ACCESS

EXNO:15

DATE:

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

QUERY: GRANT select
ON departments
TO <adh04____>;

OUTPUT:

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

```
1 GRANT select
2   ON dept
3   TO adh04_____;
```

An error message is displayed in a yellow box:

```
Error at line 3/5: ORA-01917: user or role 'ADH04_____' does not exist
```

Below the command, the results section shows the command was run in 0.02 seconds. The URL at the bottom is <https://220701020@rajalakshmi.edu.in>.

2. Query all the rows in your DEPARTMENTS table.

QUERY: select * from dept;

OUTPUT:

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

```
1 select * from dept;
```

The results section displays the data from the DEPARTMENTS table:

ID	NAME	JOB_TITLE	CITY_NAME	CITY_ID	NO_OF_EMP	Avg_Salary
20	sales	clerk	chennai	1	34	150000
50	technical	supervisor	bangalore	2	7	1000000
10	Marketing	manager	hyderabad	3	3	2000000

3 rows returned in 0.02 seconds. The URL at the bottom is <https://220701020@rajalakshmi.edu.in>.

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

RESULT:

PL/SQL **CONTROL STRUCTURES**

EXNO:16

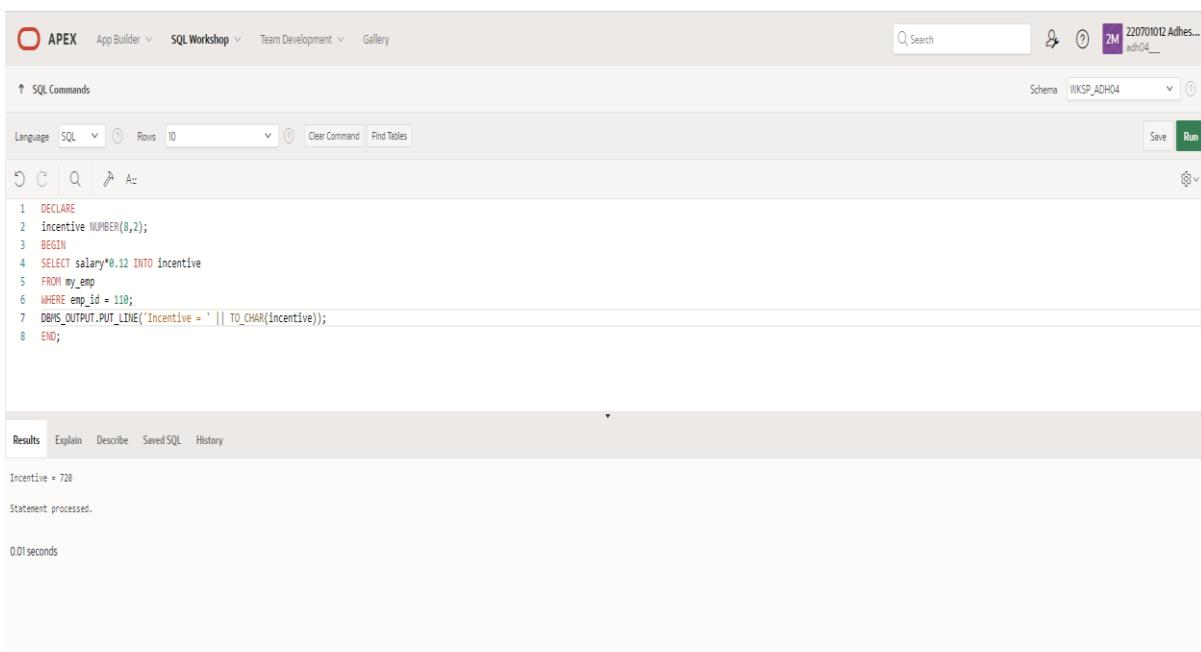
DATE:

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, a PL/SQL block is written and executed. The output pane shows the result of the DBMS_OUTPUT.PUT_LINE statement.

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

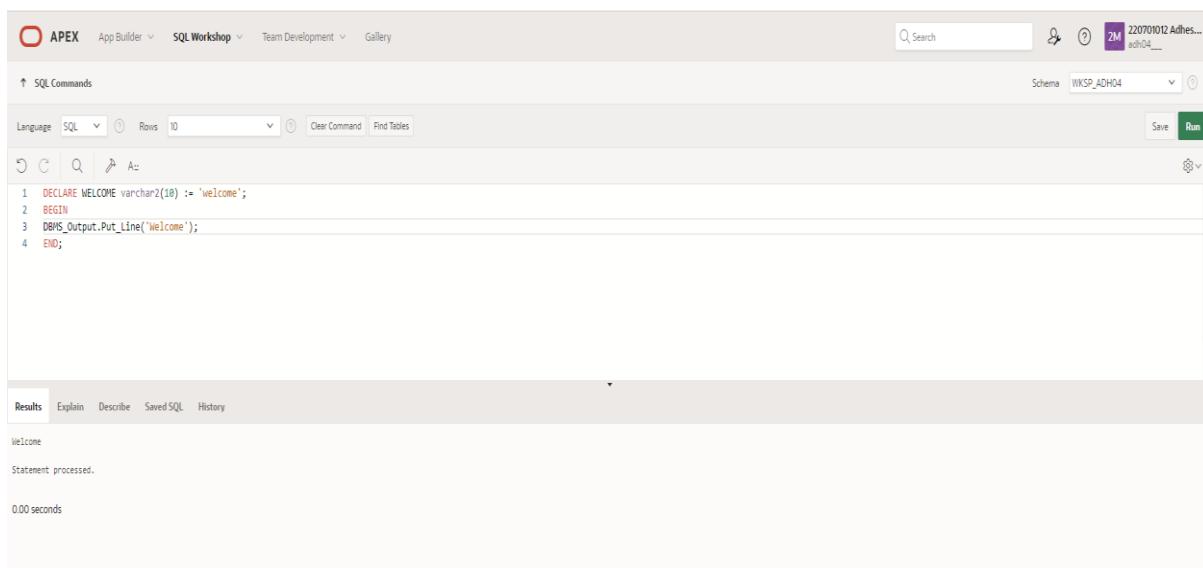
Incentive = 720
Statement processed.
0.01 seconds

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, a PL/SQL block is entered:

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

In the Results pane, the output is displayed:

```
Welcome
Statement processed.
```

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

QUERY: DECLARE

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

BEGIN

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

OUTPUT:

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

Results Explain Describe Saved SQL History

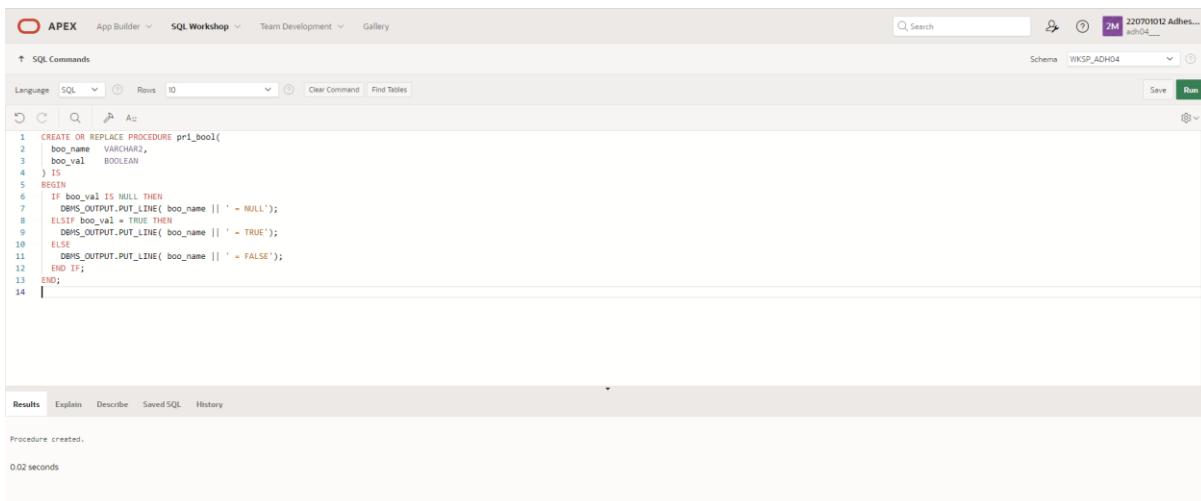
```
Before invoking procedure, salary_of_emp: 54642
After invoking procedure, salary_of_emp: 55642
Statement processed.

0.01 seconds
```

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

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, tabs for 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery' are visible. The main area is titled 'SQL Commands'. A toolbar with various icons is at the top, followed by a search bar and a schema dropdown set to 'WKSP_ADH04'. The SQL editor contains the PL/SQL code provided in the previous section. The code is highlighted in red for syntax errors. The bottom pane is titled 'Results' and displays the output: 'Procedure created.' and '0.02 seconds'.

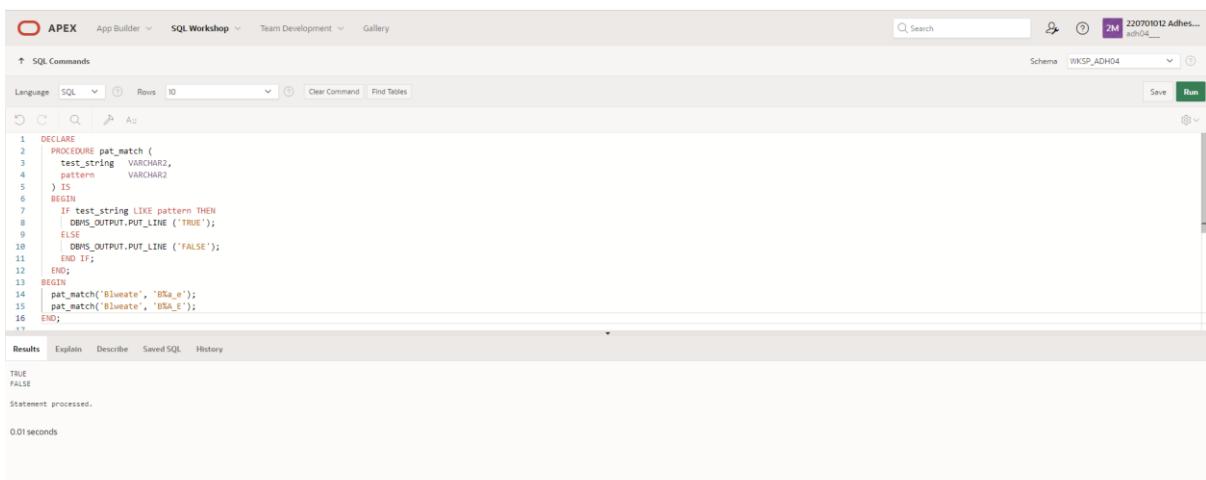
```
CREATE OR REPLACE PROCEDURE pri_bool(
boo_name VARCHAR2,
boo_val BOOLEAN
) IS
BEGIN
IF boo_val IS NULL THEN
DBMS_OUTPUT.PUT_LINE( boo_name || ' = NULL');
ELSIF boo_val = TRUE THEN
DBMS_OUTPUT.PUT_LINE( boo_name || ' = TRUE');
ELSE
DBMS_OUTPUT.PUT_LINE( boo_name || ' = FALSE');
END IF;
END;
```

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

QUERY: DECLARE

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area displays the PL/SQL code provided in the question. Below the code, the 'Results' tab is active, showing the output of the execution. The output consists of two lines: 'TRUE' and 'FALSE'. A note below the results states 'Statement processed.' and '0.01 seconds'.

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

Results Explain Describe Saved SQL History

TRUE
FALSE

Statement processed.
0.01 seconds

6. Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num_small variable and large number will store in num_large variable

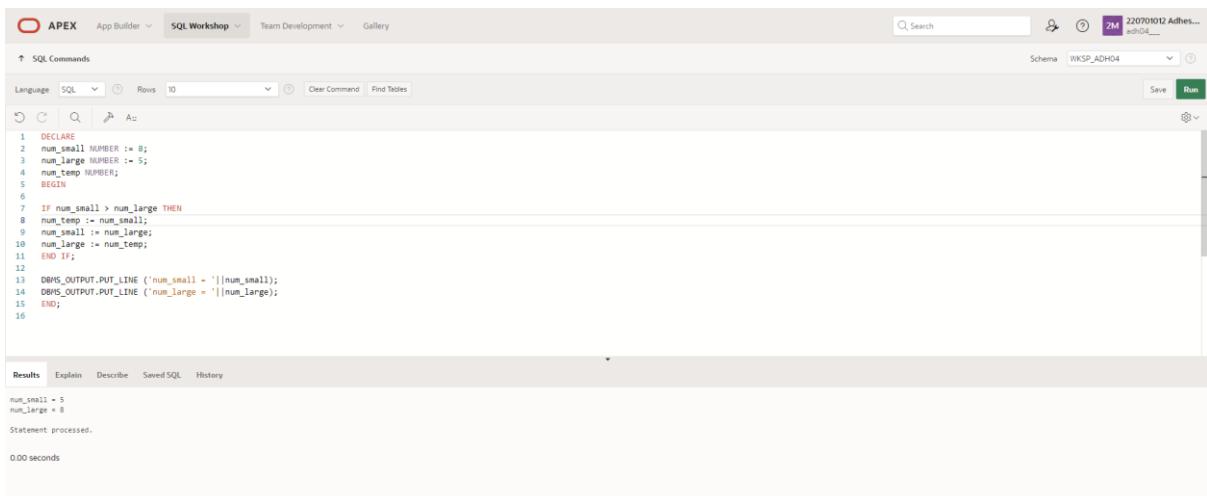
QUERY: DECLARE

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

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

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands tab, a PL/SQL block is written and executed. The code declares three variables (num_small, num_large, num_temp), initializes num_small to 8 and num_large to 5, and then swaps their values if num_small is greater than num_large. It uses DBMS_OUTPUT.PUT_LINE to print the values of num_small and num_large. The results tab shows the output: num_small = 5 and num_large = 8.

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

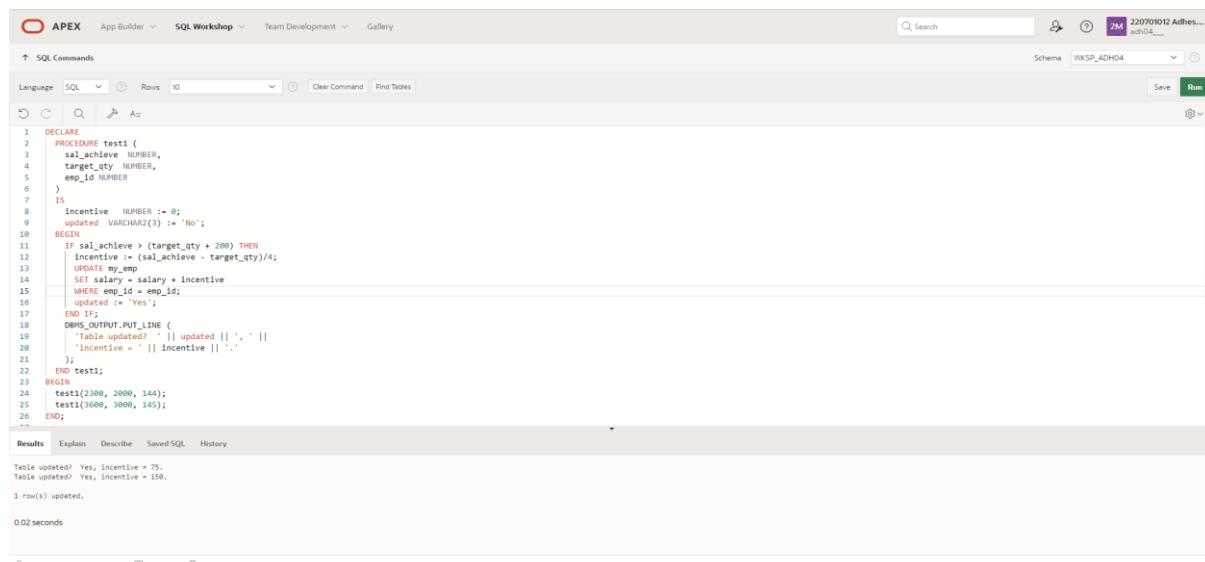
Results
num_small = 5 num_large = 8 Statement processed. 0.00 seconds

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

QUERY: DECLARE

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

OUTPUT:



```
1 DECLARE
2 PROCEDURE test1 (
3     sal_achieve NUMBER,
4     target_qty NUMBER,
5     emp_id NUMBER
6 )
7 IS
8     incentive NUMBER := 0;
9     updated VARCHAR2(3) := 'No';
10 BEGIN
11     IF sal_achieve > (target_qty + 200) THEN
12         incentive := (sal_achieve - target_qty)/4;
13         UPDATE employees
14         SET salary = salary + incentive
15         WHERE emp_id = emp_id;
16         updated := 'Yes';
17     END IF;
18     DBMS_OUTPUT.PUT_LINE (
19         'Table updated? ' || updated || ','
20         'incentive = ' || incentive || !
21     );
22 END test1;
23 BEGIN
24     test1(2300, 2000, 144);
25     test1(3600, 3000, 145);
26 END;
27
```

Results

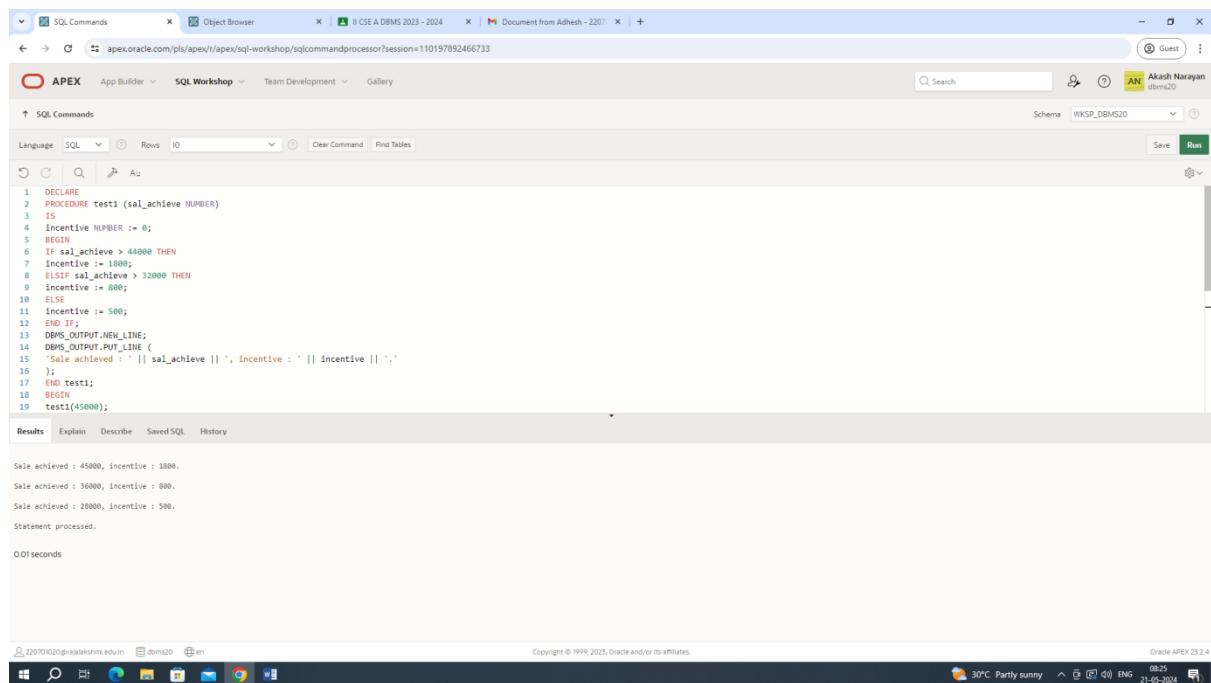
```
Table updated? Yes, Incentive = 75.
Table updated? Yes, Incentive = 150.
1 row(s) updated.

0.02 seconds
```

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

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The code area contains the provided PL/SQL procedure. The results pane displays the output of three executions of the procedure with input values 45000, 36000, and 28000, showing incentives of 1800, 800, and 500 respectively. The status bar at the bottom indicates the session ID, schema, and Oracle version.

```
1  DECLARE
2  PROCEDURE test1 (sal_achieve NUMBER)
3  IS
4    incentive NUMBER := 0;
5  BEGIN
6    IF sal_achieve > 44000 THEN
7      incentive := 1800;
8    ELSIF sal_achieve > 32000 THEN
9      incentive := 800;
10   ELSE
11     incentive := 500;
12   END IF;
13   DBMS_OUTPUT.NEW_LINE;
14   DBMS_OUTPUT.PUT_LINE (
15     'Sale achieved : ' || sal_achieve || ', incentive : ' || incentive || '.'
16   );
17 END test1;
18 BEGIN
19   test1(45000);

```

Sale achieved : 45000, incentive : 1800.
Sale achieved : 36000, incentive : 800.
Sale achieved : 28000, incentive : 500.
Statement processed.
0.01 seconds

Copyright © 1996-2023, Oracle and/or its affiliates. Oracle APEX 23.2.4
30°C Partly sunny 08:25 ENG 21-05-2024

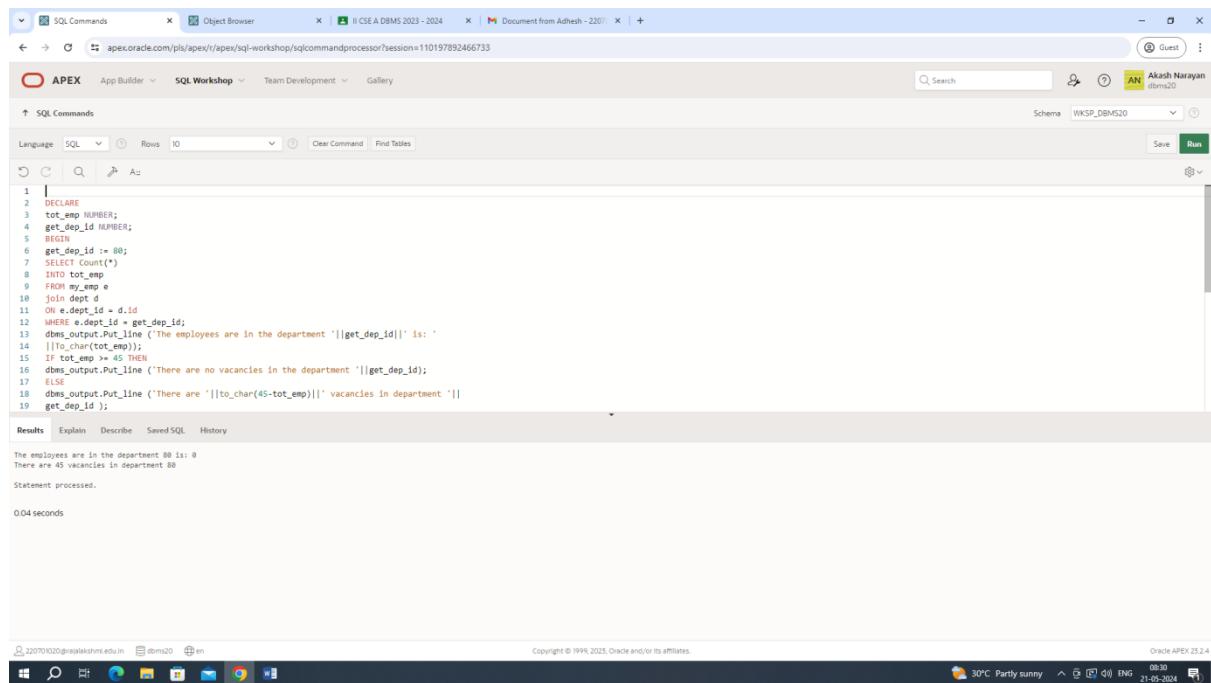
9. Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

QUERY:

```
DECLARE
    tot_emp NUMBER;
    get_dep_id NUMBER;

BEGIN
    get_dep_id := 80;
    SELECT Count(*)
    INTO tot_emp
    FROM employees e
        join departments d
            ON e.department_id = d.department_id
    WHERE e.department_id = get_dep_id;
    dbms_output.Put_line ('The employees are in the department'||get_dep_id||' is: '
                           ||To_char(tot_emp));
    IF tot_emp >= 45 THEN
        dbms_output.Put_line ('There are no vacancies in the department'||get_dep_id);
    ELSE
        dbms_output.Put_line ('There are'||to_char(45-tot_emp)||' vacancies in
department'|| get_dep_id );
    END IF;
END;
/
```

OUTPUT:



```
1 | 
2 | DECLARE
3 |     tot_emp NUMBER;
4 |     get_dep_id NUMBER;
5 |     BEGIN
6 |         get_dep_id := 80;
7 |         SELECT Count(*)
8 |         INTO tot_emp
9 |         FROM employees e
10 |             join departments d
11 |                 ON e.department_id = d.department_id
12 |             WHERE e.department_id = get_dep_id;
13 |             dbms_output.Put_line ('The employees are in the department'||get_dep_id||' is: '
14 |                           ||To_char(tot_emp));
15 |             IF tot_emp >= 45 THEN
16 |                 dbms_output.Put_line ('There are no vacancies in the department'||get_dep_id);
17 |             ELSE
18 |                 dbms_output.Put_line ('There are'||to_char(45-tot_emp)||' vacancies in department'|| get_dep_id );
19 |             END IF;
20 |         END;
21 | /
```

The employees are in the department 80 is: 0
There are 45 vacancies in department 80
Statement processed.
0.04 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

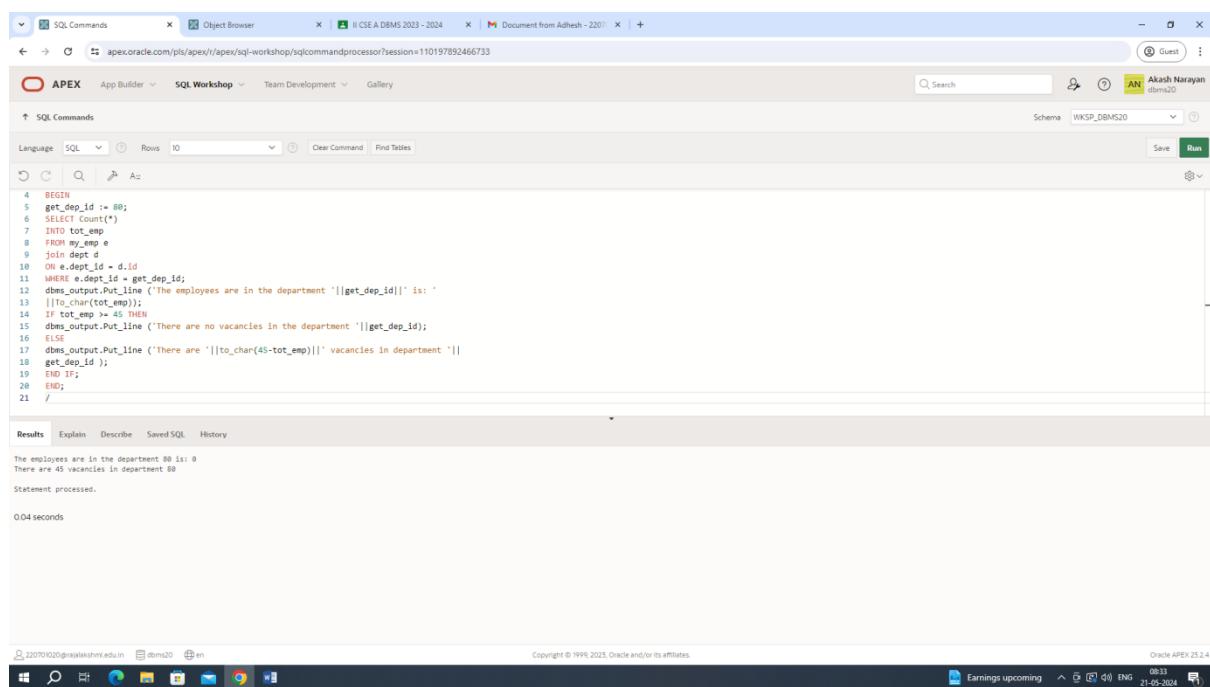
```
dbms_output.Put_line ('There are'||to_char(45-tot_emp)||' vacancies in
department'|| get_dep_id );
```

END IF;

END;

/

OUTPUT:



```
4 BEGIN
5 get_dep_id := 80;
6 SELECT Count(*)
7 INTO tot_emp
8 FROM employees e
9 join departments d
10 ON e.department_id = d.id
11 WHERE e.department_id = get_dep_id;
12 dbms_output.Put_line ('The employees are in the department'||get_dep_id||' is: '
13 ||To_char(tot_emp));
14 IF tot_emp >= 45 THEN
15 dbms_output.Put_line ('There are no vacancies in the department'||get_dep_id);
16 ELSE
17 dbms_output.Put_line ('There are'||to_char(45-tot_emp)||' vacancies in department'|| 
18 get_dep_id );
19 END IF;
20 END;
21 /
```

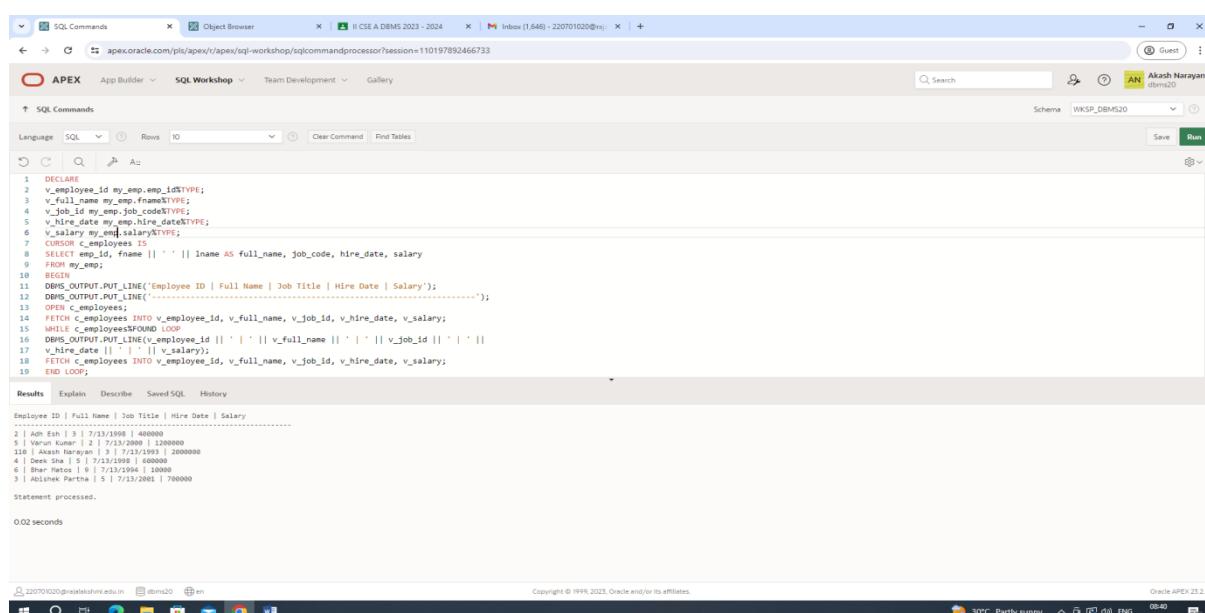
The employees are in the department 80 is: 0
There are 45 vacancies in department 80
Statement processed.
0.04 seconds

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

QUERY: DECLARE

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

OUTPUT:



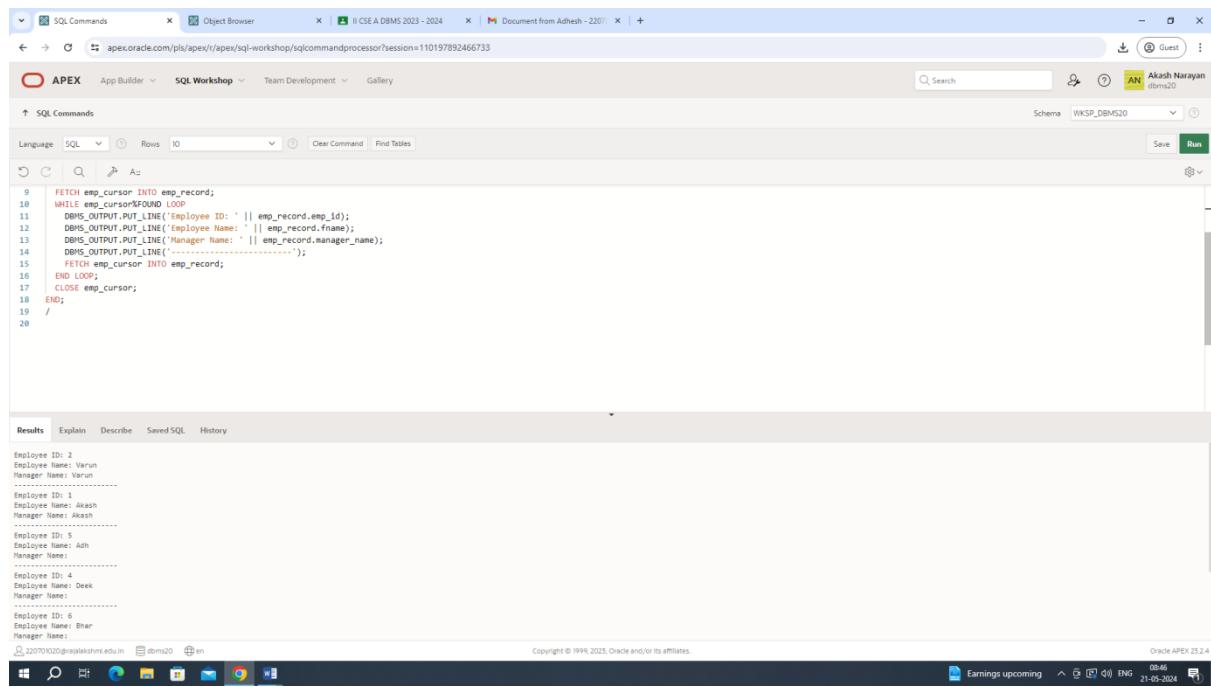
Employee ID	Full Name	Job Title	Hire Date	Salary
2	Ash Esh	3 7/13/1990 400000		
5	Varun Kumar	2 7/13/2000 120000		
6	Shivam Patel	4 7/13/1998 200000		
4	Deeksha Sha	5 7/13/1998 600000		
8	Bhav Patel	9 7/13/1994 100000		
7	Umesh Patel	6 7/13/2001 700000		

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. In the top navigation bar, the 'SQL Commands' tab is selected. Below the tabs, there are dropdowns for 'Language' (set to SQL), 'Rows' (set to 10), and buttons for 'Clear Command' and 'Find Tables'. On the right side, there are buttons for 'Save' and 'Run'. The main workspace contains the PL/SQL code provided in the question. The results pane at the bottom displays the output of the code execution, which lists employee details with their respective manager names. The output is as follows:

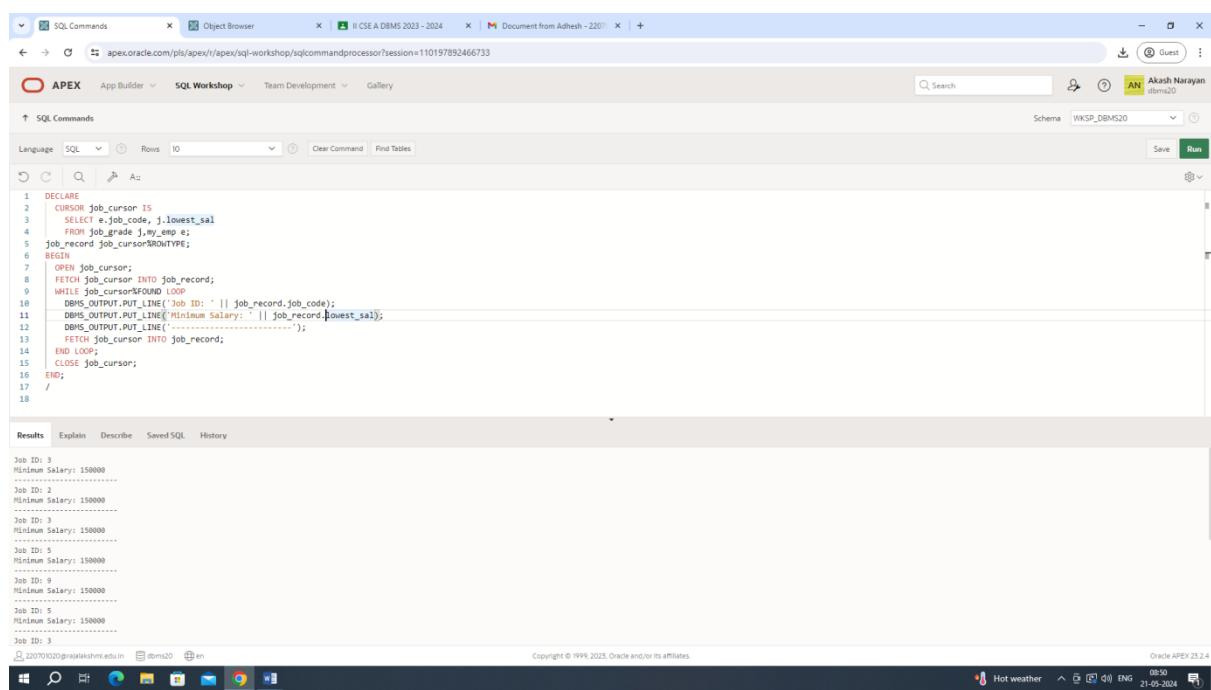
Employee ID	Employee Name	Manager Name
2	Varun	Varun
3	Akash	Akash
5	Ath	Ath
4	Deek	Deek
6	Bhar	Bhar

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:



```
1  DECLARE
2  CURSOR job_cursor IS
3    SELECT e.job_code, j.lowest_sal
4      FROM job_grade j,emp e;
5    job_record job_cursor%ROWTYPE;
6  BEGIN
7    OPEN job_cursor;
8    FETCH job_cursor INTO job_record;
9    WHILE job_cursor%FOUND LOOP
10      DBMS_OUTPUT.PUT_LINE('Job ID: ' || job_record.job_code);
11      DBMS_OUTPUT.PUT_LINE('Minimum Salary: ' || job_record.lowest_sal);
12      DBMS_OUTPUT.PUT_LINE('-----');
13      FETCH job_cursor INTO job_record;
14    END LOOP;
15    CLOSE job_cursor;
16  END;
17 /
18
```

Results

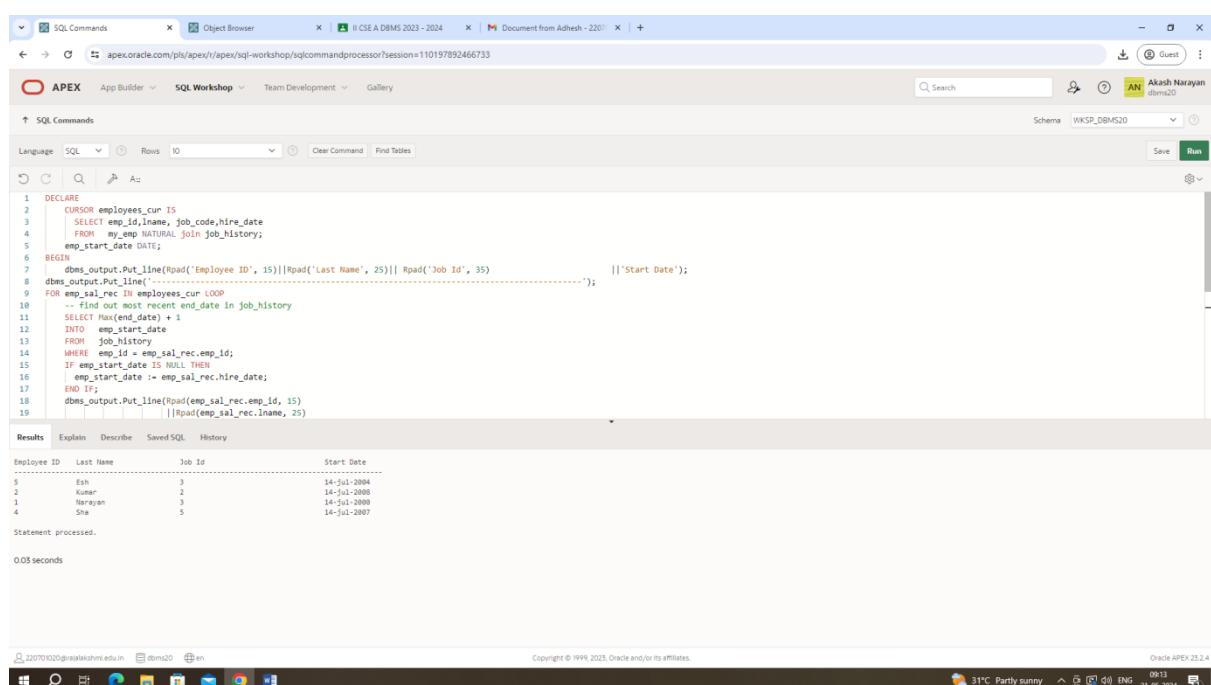
Job ID	Minimum Salary
1	150000
2	150000
3	150000
4	150000
5	150000
6	150000
7	150000
8	150000
9	150000
10	150000
11	150000
12	150000
13	150000
14	150000
15	150000
16	150000
17	150000
18	150000

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:



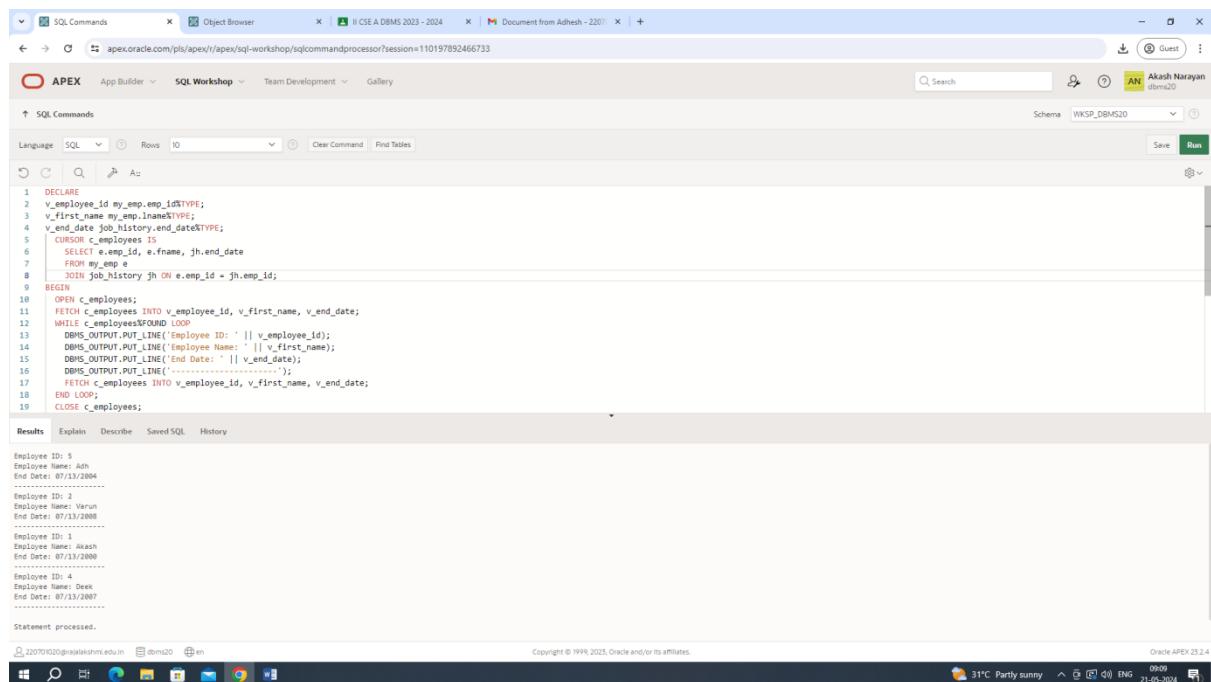
Employee ID	Last Name	Job Id	Start Date
5	Ali	3	14-Jul-2000
2	Kumar	2	14-Jul-2000
1	Narayan	3	14-Jul-2000
4	Sha	5	14-Jul-2007

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:



```
Employee ID: 5
Employee Name: Adh
End Date: 07/13/2004
-----
Employee ID: 2
Employee Name: Varun
End Date: 07/13/2008
-----
Employee ID: 1
Employee Name: Akash
End Date: 07/13/2008
-----
Employee ID: 4
Employee Name: Deepak
End Date: 07/13/2007
-----
Statement processed.
```

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

RESULT:

PROCEDURES AND FUNCTIONS

EXNO: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. The code area contains a PL/SQL block to calculate the factorial of a number. The results tab shows the output: 120, indicating the factorial of 5. The bottom status bar includes copyright information and the Oracle APEX version.

```
1 DECLARE
2     fac NUMBER := 1;
3     n NUMBER := :1;
4 BEGIN
5     WHILE n > 0 LOOP
6         fac := n * fac;
7         n := n - 1;
8     END LOOP;
9     DBMS_OUTPUT.PUT_LINE(fac);
10 END;
11 
```

Results

120
Statement processed.
0.00 seconds

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

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

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

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

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

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

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

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the code editor, a PL/SQL block is written to retrieve book information. The code includes declarations for variables like v_book_id, v_title, v_author, and v_year_published, and a BEGIN block that calls a procedure get_book_info with parameters p_book_id, p_title, p_author, and p_year_published. It then uses DBMS_OUTPUT.PUT_LINE to print the results. The results tab shows the output: Title: ISBNAN - Retrieved, Author: HARI, Year Published: 2009. The status bar at the bottom indicates the statement was processed in 0.01 seconds.

```
1 DECLARE
2   v_book_id NUMBER := 1;
3   v_title VARCHAR2(100);
4   v_author VARCHAR2(100);
5   v_year_published NUMBER;
6 BEGIN
7   v_title := 'Initial Title';
8
9   get_book_info(p_book_id => v_book_id, p_title => v_title, p_author => v_author, p_year_published => v_year_published);
10
11  DBMS_OUTPUT.PUT_LINE('Title: ' || v_title);
12  DBMS_OUTPUT.PUT_LINE('Author: ' || v_author);
13  DBMS_OUTPUT.PUT_LINE('Year Published: ' || v_year_published);
14 END;
15
```

Results Explain Describe Saved SQL History

Title: ISBNAN - Retrieved
Author: HARI
Year Published: 2009
Statement processed.
0.01seconds

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

RESULT:

TRIGGER

EXNO: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. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The user is logged in as 'Akash Narayan doms20'. The schema is set to 'WKSP_DBMS20'. The main area is titled 'SQL Commands' with tabs for 'Language', 'SQL', 'Rows', and 'Clear Command'. The command entered is the PL/SQL code for the trigger 'prevent_parent_deletion'. The code implements a BEFORE DELETE trigger on the 'parent_table' that prevents rows from being deleted if any child records exist for that parent. The trigger uses a local variable 'v_child_count' to count child records and raises the 'child_exists' exception if the count is greater than zero. It also handles the 'child_exists' exception by raising the application error '-20001' with the message 'Cannot delete parent record while child records exist.'. The results tab shows the output: 'Trigger created.' and '0.05 seconds'. The bottom footer includes copyright information for Oracle and the APEX version.

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

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.');

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 top right shows the user 'Akash Narayan' and the schema 'WKSP_DBMS20'. The main workspace displays the PL/SQL code for the 'check_duplicates' trigger. The code is identical to the one provided in the question. Below the code, the 'Results' tab shows the output: 'Trigger created.' and '0.04 seconds'. The bottom footer includes copyright information: 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

```
1 CREATE OR REPLACE TRIGGER check_duplicates
2 BEFORE INSERT OR UPDATE ON unique_values_table
3 FOR EACH ROW
4 DECLARE
5     duplicate_found EXCEPTION;
6     PRAGMA EXCEPTION_INIT(duplicate_found, -20002);
7     v_count NUMBER;
8 BEGIN
9     SELECT COUNT(*) INTO v_count FROM unique_values_table
10    WHERE unique_col = :NEW.unique_col AND id != :NEW.id;
11    IF v_count > 0 THEN
12        RAISE duplicate_found;
13    END IF;
14 EXCEPTION
15    WHEN duplicate_found THEN
16        RAISE_APPLICATION_ERROR(-20002, 'Duplicate value found in unique_col.');
17 END;
```

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.');

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 top right shows the user 'Akash Narayan dbms20' and the schema 'WKSP_DBMS20'. The main area is titled 'SQL Commands' with tabs for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. Below these are buttons for 'Save' and 'Run'. The SQL command area contains the PL/SQL code provided in the question. The results tab at the bottom shows the output: 'Trigger created.' and '0.04 seconds'. The bottom footer includes copyright information: 'Copyright © 1999-2023, Oracle and/or its affiliates.' and 'Oracle APEX 25.2.4'.

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

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

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

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, tabs for 'App Builder', 'SQL Workshop', and 'Team Development' are visible. The 'SQL Workshop' tab is active. On the right side, the user 'Akash Narayan dbms20' is logged in. The schema dropdown shows 'WKSP_DBMS20'. The main area is titled 'SQL Commands' and contains the following PL/SQL code:

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

Below the code, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected. The output pane displays the message 'Trigger created.' and '0.04 seconds'. At the bottom of the page, the URL '220701020@rajalakshmi.edu.in', the schema 'dbms20', and the session ID 'en' are shown. The footer also includes the copyright notice 'Copyright © 1999-2023, Oracle and/or its affiliates.' and the 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 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a user profile for 'Akash Narayan dbms20'. The main area is titled 'SQL Commands' with tabs for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. The 'Run' button is highlighted in green. Below the tabs, there's a search bar and a schema dropdown set to 'WKSP_DBMS20'. The SQL editor contains the PL/SQL code for the trigger, which is then run. The results show the message 'Trigger created.' and a execution time of '0.04 seconds'. At the bottom, there are links for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The footer includes copyright information for Oracle and the URL '220701020@rajalakshmi.edu.in'.

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

Trigger created.
0.04 seconds

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

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

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, the tabs 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery' are visible. On the right side, the user 'Akash Narayan dbms20' is logged in. The main area is titled 'SQL Commands'. The 'Language' dropdown is set to 'SQL'. The 'Rows' dropdown is set to '10'. The 'Schema' dropdown is set to 'WKSP_DBMS20'. There are 'Save' and 'Run' buttons on the right. The SQL command entered is:

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

Below the command, the 'Results' tab is selected. The output shows the message 'Trigger created.' and '0.04 seconds'. At the bottom, the footer includes the URL '220701020@nejalakshmi.edu.in', the schema 'dbms20', the language 'en', copyright information 'Copyright © 1999, 2023, Oracle and/or its affiliates.', and the version 'Oracle APEX 25.2.4'.

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. In the top navigation bar, 'APEX' is selected. Below it, the tabs 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery' are visible. On the right side, there's a user profile for 'Akash Narayan' and a schema dropdown set to 'WKSP_DBMS20'. The main workspace is a code editor with syntax highlighting for PL/SQL. The code listed is the trigger definition provided in the previous text block. At the bottom of the code editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The status bar at the bottom of the screen displays the URL '220701020@rejalaakshmi.edu.in', the session ID 'd0m20', and the language 'en'. It also shows copyright information '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:

MONGO DB

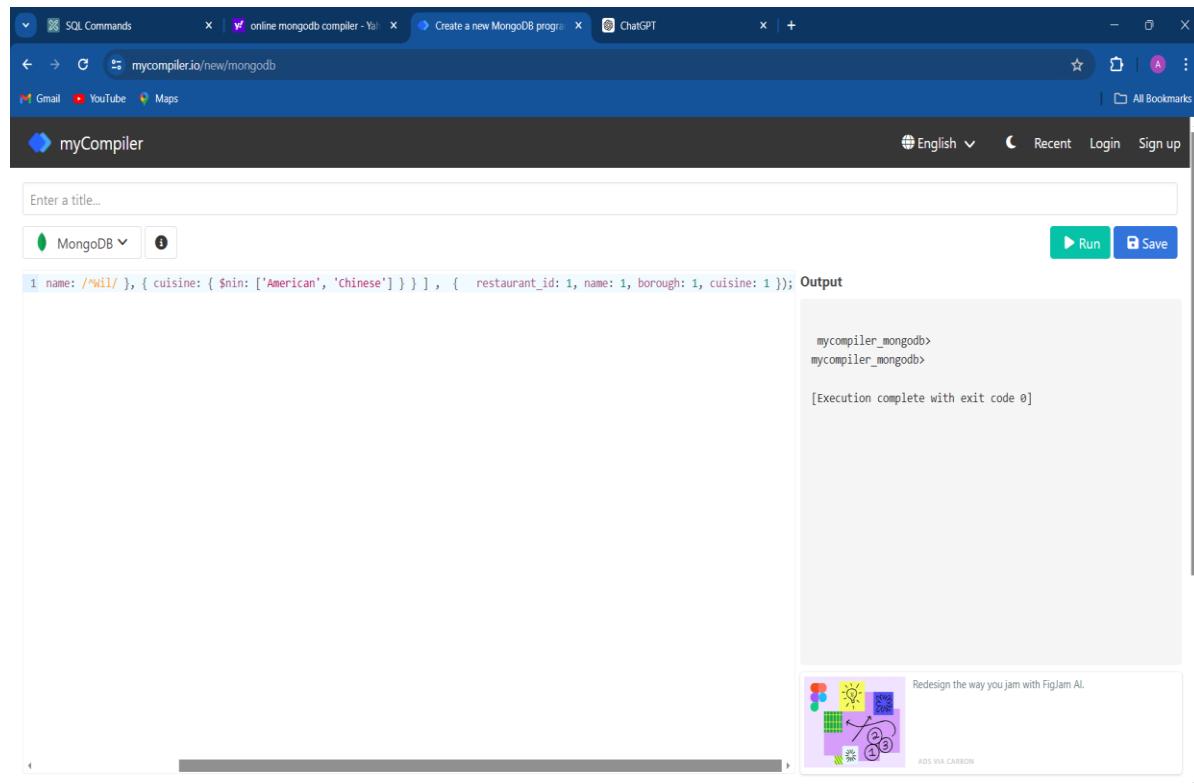
EXNO: 19

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 a web browser window with the URL mycompiler.io/new/mongodb. The page has a dark theme with a header bar containing tabs for "SQL Commands", "Create a new MongoDB program", and "ChatGPT". Below the header is a toolbar with icons for "Gmail", "YouTube", "Maps", "myCompiler", "English", "Recent", "Login", and "Sign up". A search bar says "Enter a title...". A dropdown menu shows "MongoDB" and a help icon. At the bottom are "Run" and "Save" buttons. The main area contains a code editor with the following MongoDB query:

```
1 name: /Wil/, { cuisine: { $nin: ['American', 'Chinese'] } } ] , { restaurant_id: 1, name: 1, borough: 1, cuisine: 1 } ); Output
```

To the right is a terminal window titled "mycompiler_mongodb" showing the command "mycompiler_mongodb" and the message "[Execution complete with exit code 0]". At the bottom of the terminal is an 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 a web browser window with the URL mycompiler.io/new/mongodb. The page has a dark header with the title "myCompiler". Below the header is a search bar with placeholder text "Enter a title...". A dropdown menu is open, showing "MongoDB" and other options like "SQL Commands" and "JavaScript". To the right of the dropdown are "Run" and "Save" buttons. The main area contains a code editor with the following MongoDB query:

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

Below the code editor is a results panel titled "Output" which displays the command being run:

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

At the bottom of the results panel is an 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("2014-08-11T00:00:00Z") },{ restaurant_id: 1, name: 1, grades: 1 });

OUTPUT:

The screenshot shows the same web browser setup as the first one, but the code editor now contains the query from the previous step:

```
1 { grades: { $elemMatch: { grade: "A", score: 9, "date": ISODate("2014-08-11T00:00:00Z") } }, { restaurant_id: 1, name: 1, grades: 1 } }; Output
```

The results panel shows the command running again:

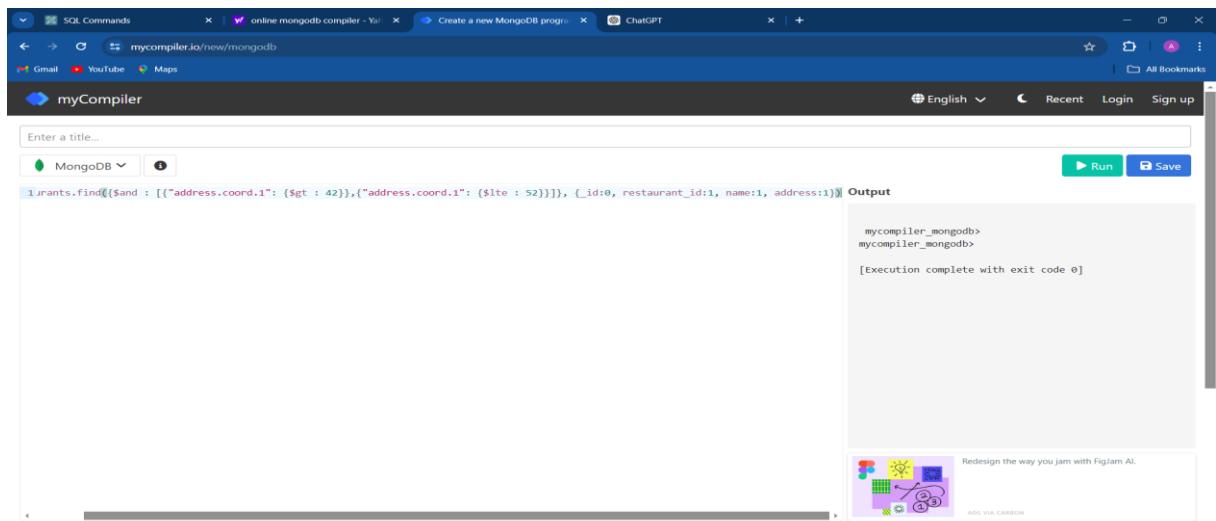
```
mycompiler_mongodb>
mycompiler_mongodb>
[Execution complete with exit code 0]
```

At the bottom of the results panel is an advertisement for FigJam AI.

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:

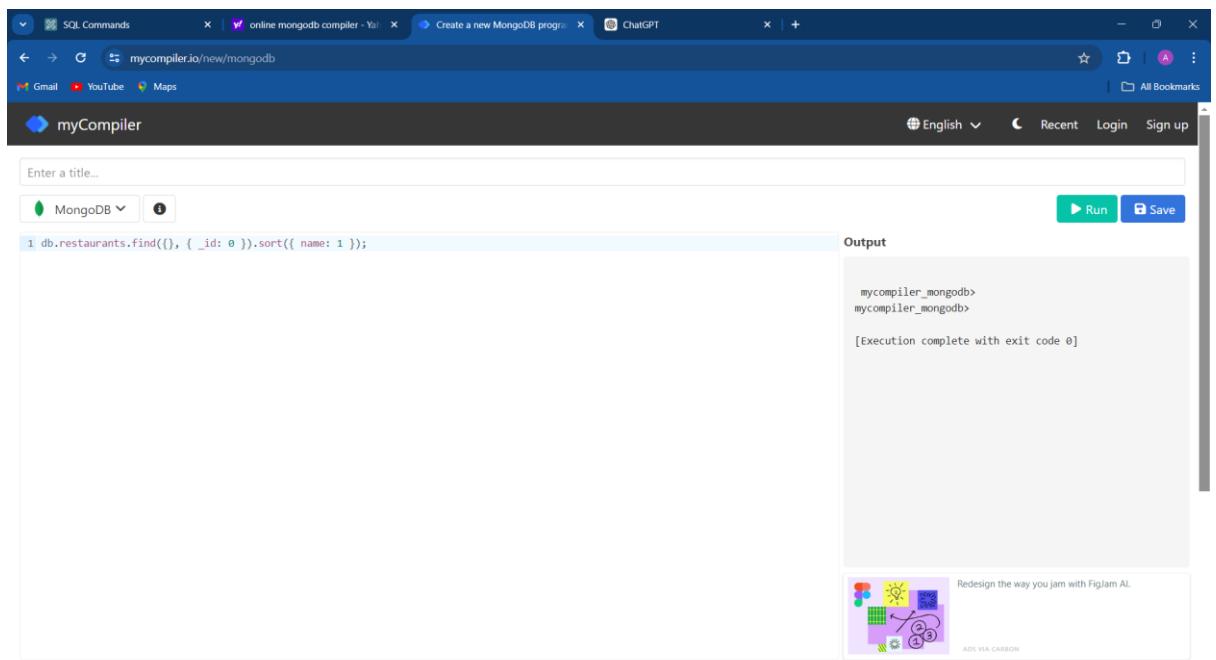


The screenshot shows a web-based MongoDB query compiler. The top navigation bar includes tabs for "SQL Commands", "online mongodb compiler - Yale", "Create a new MongoDB program", and "ChatGPT". Below the navigation is a toolbar with "Enter a title...", a MongoDB dropdown, and "Run" and "Save" buttons. The main area contains the MongoDB query: db.restaurants.find({\$and : [{"address.coord.1": {\$gt : 42}}, {"address.coord.1": {\$lte : 52}}]}, {_id:0, restaurant_id:1, name:1, address:1}). To the right is an "Output" panel showing the command being run and the response: "mycompiler_mongodb> mycompiler_mongodb> [Execution complete with exit code 0]". A small advertisement for FigJam AI is visible at the bottom right.

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:

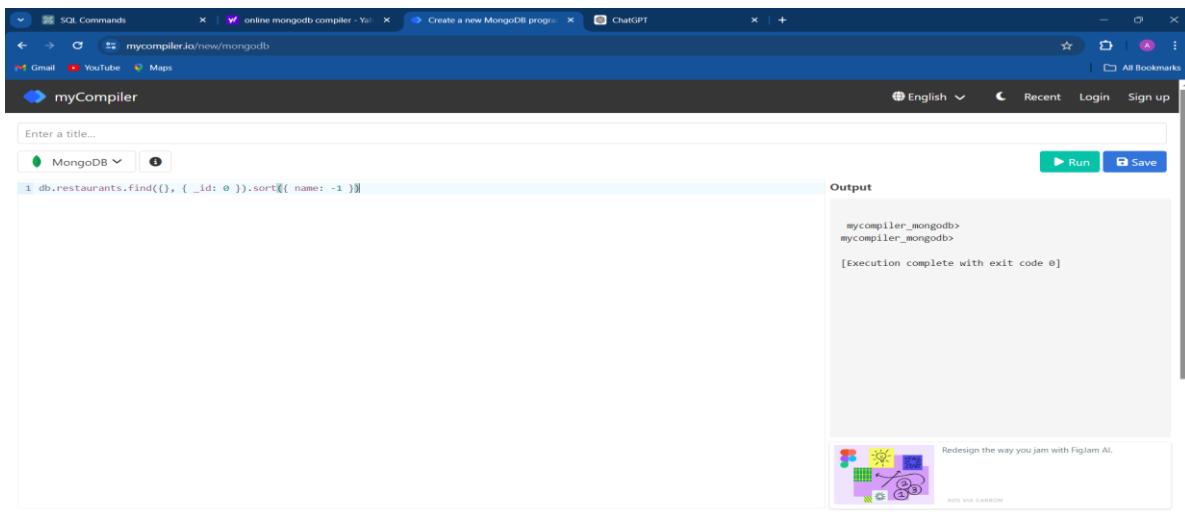


The screenshot shows a web-based MongoDB query compiler. The top navigation bar includes tabs for "SQL Commands", "online mongodb compiler - Yale", "Create a new MongoDB program", and "ChatGPT". Below the navigation is a toolbar with "Enter a title...", a MongoDB dropdown, and "Run" and "Save" buttons. The main area contains the MongoDB query: db.restaurants.find({}, { _id: 0 }).sort({ name: 1 });. To the right is an "Output" panel showing the command being run and the response: "mycompiler_mongodb> mycompiler_mongodb> [Execution complete with exit code 0]". A small advertisement for FigJam AI is visible at the bottom right.

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:



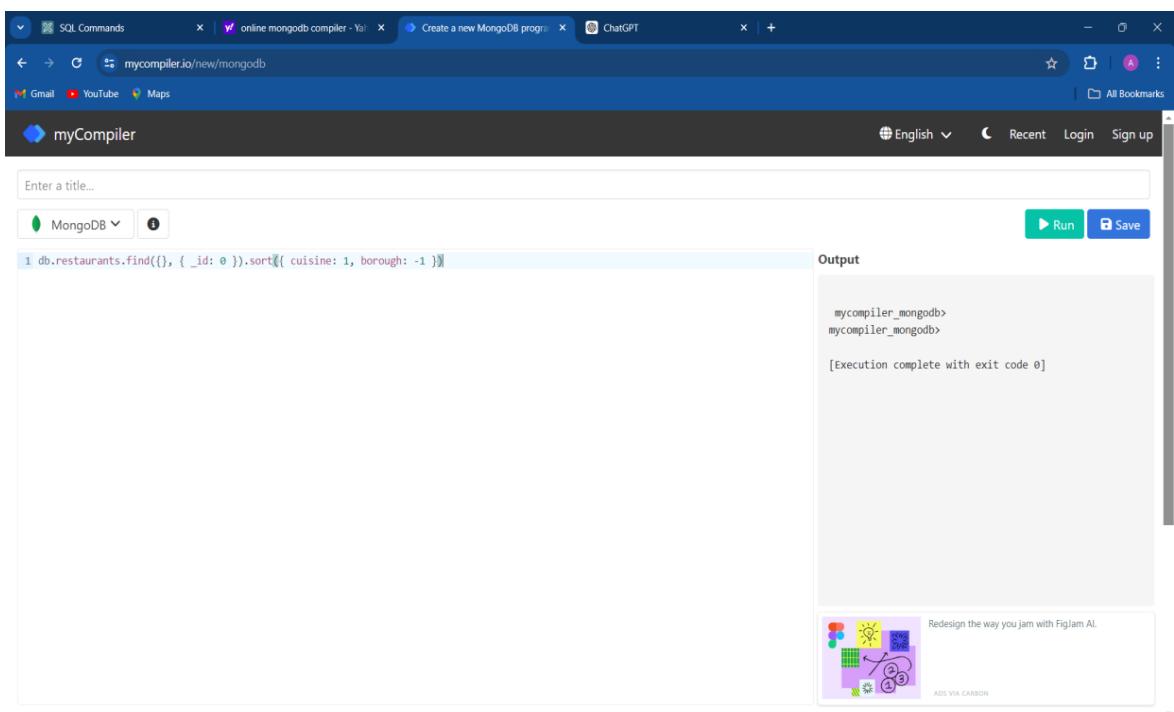
The screenshot shows a browser window with the URL mycompiler.io/new/mongodb. The title bar says "myCompiler". The main area has a text input field with the query: "db.restaurants.find({}, { _id: 0 }).sort({ name: -1 })". Below the input field are "Run" and "Save" buttons. To the right is an "Output" panel displaying the results of the query execution:

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

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:



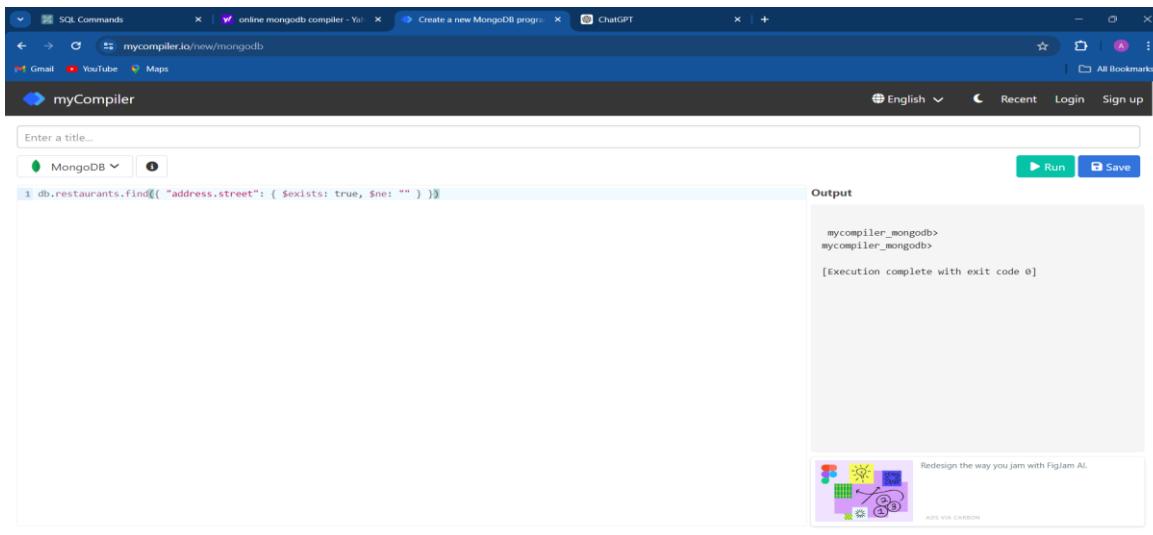
The screenshot shows a browser window with the URL mycompiler.io/new/mongodb. The title bar says "myCompiler". The main area has a text input field with the query: "db.restaurants.find({}, { _id: 0 }).sort({ cuisine: 1, borough: -1 })". Below the input field are "Run" and "Save" buttons. To the right is an "Output" panel displaying 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:

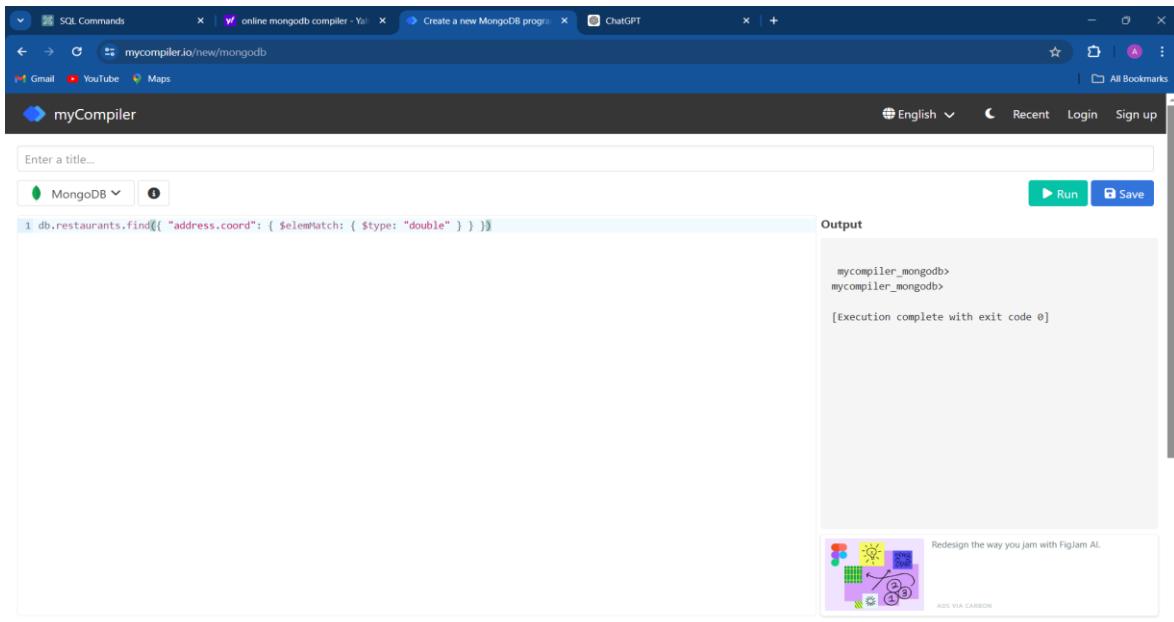


The screenshot shows a browser window with the URL mycompiler.io/new/mongodb. The title bar includes tabs for "SQL Commands", "online mongodb compiler - Yal...", "Create a new MongoDB program", and "ChatGPT". Below the tabs, there are links for "Gmail", "YouTube", and "Maps". The main area has a dark header with "myCompiler" and a "Run" button. The code input field contains the query: db.restaurants.find({ "address.street": { \$exists: true, \$ne: "" } }). The output panel shows the results: mycompiler_mongodb> mycompiler_mongodb> [Execution complete with exit code 0]. At the bottom right of the output panel is an advertisement for FigJam AI.

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:

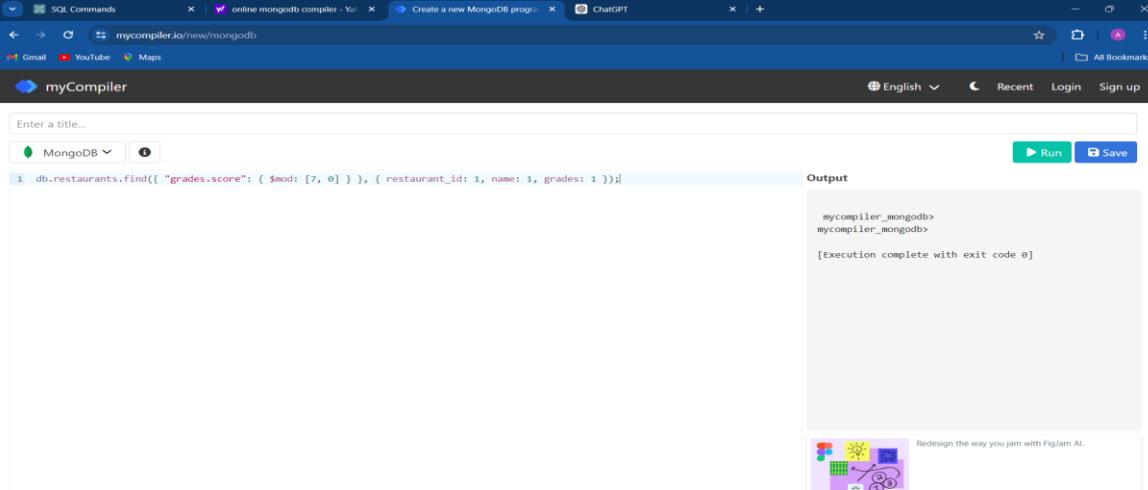


The screenshot shows a browser window with the URL mycompiler.io/new/mongodb. The title bar includes tabs for "SQL Commands", "online mongodb compiler - Yal...", "Create a new MongoDB program", and "ChatGPT". Below the tabs, there are links for "Gmail", "YouTube", and "Maps". The main area has a dark header with "myCompiler" and a "Run" button. The code input field contains the query: db.restaurants.find({ "address.coord": { \$elemMatch: { \$type: "double" } } }). The output panel shows the results: mycompiler_mongodb> mycompiler_mongodb> [Execution complete with exit code 0]. At the bottom right of the output panel is an advertisement for FigJam AI.

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 a web-based MongoDB query editor. The title bar says "myCompiler". The left panel contains a code editor with the following query:

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

The right panel is titled "Output" and shows the results of the query execution:

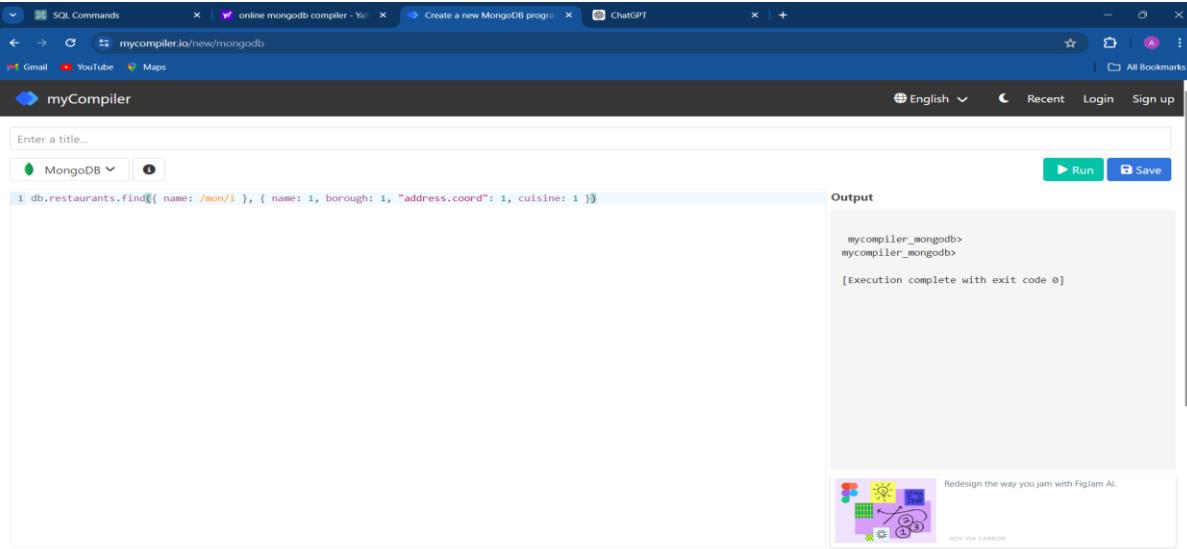
```
mycompiler_mongodb>
mycompiler_mongodb>
[Execution complete with exit code 0]
```

Below the output window, there is an advertisement for FigJam AI.

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

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

OUTPUT:



The screenshot shows a web-based MongoDB query editor. The title bar says "myCompiler". The left panel contains a code editor with the following query:

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

The right panel is titled "Output" and shows the results of the query execution:

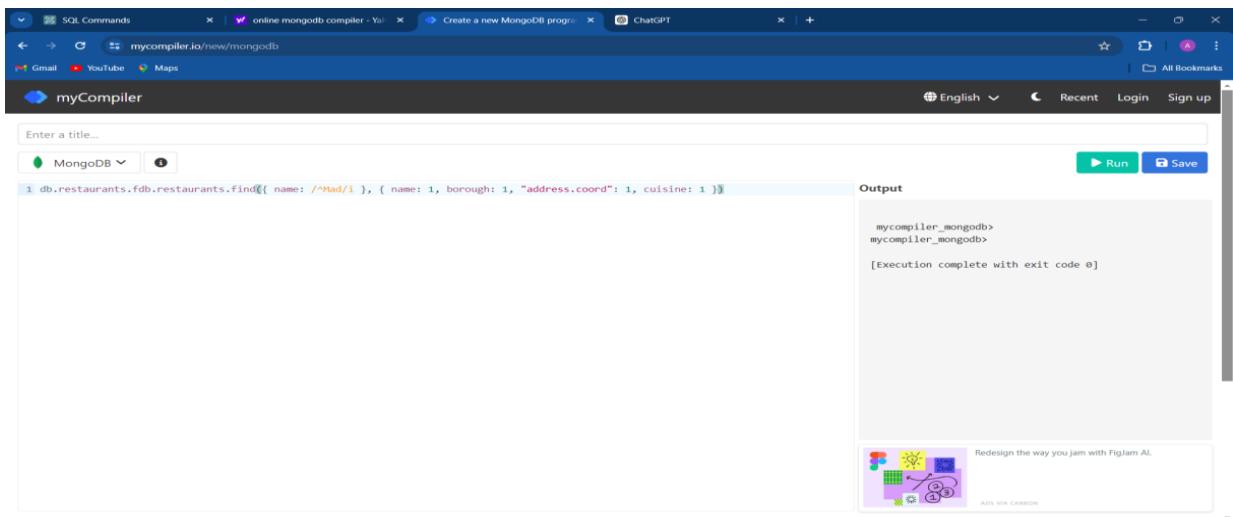
```
mycompiler_mongodb>
mycompiler_mongodb>
[Execution complete with exit code 0]
```

Below the output window, there is an advertisement for 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:

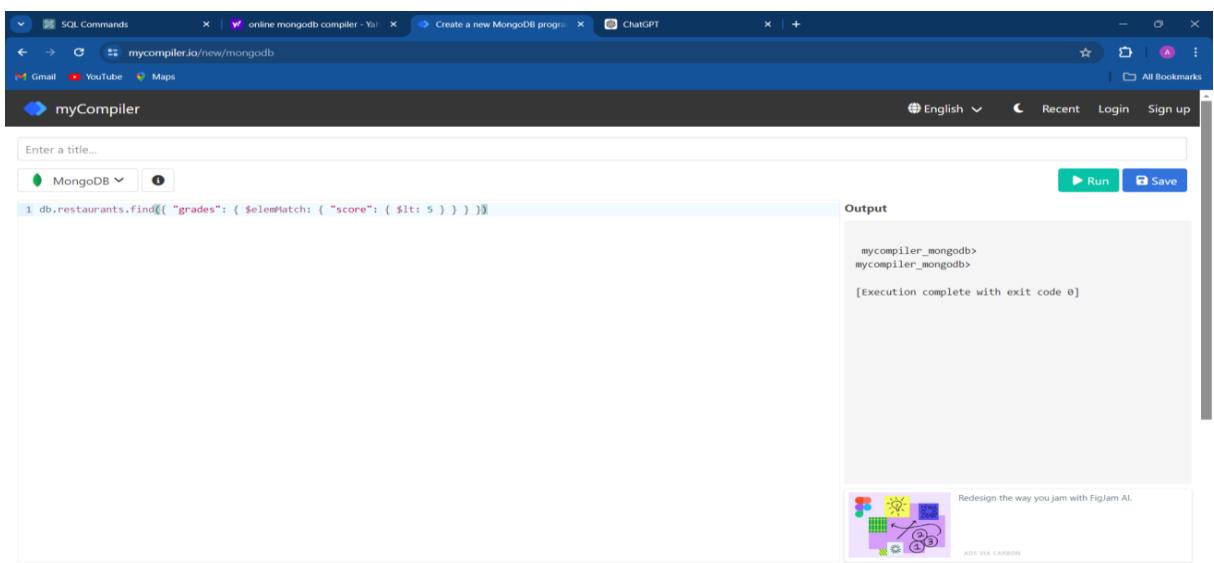


The screenshot shows a browser window with the URL mycompiler.io/new/mongodb. The main area contains a MongoDB query:db.restaurants.find({ name: /^Mad/i }, { name: 1, borough: 1, "address.coord": 1, cuisine: 1 })The output panel on the right shows the results of the query execution:mycompiler_mongodb>
mycompiler_mongodb>
[Execution complete with exit code 0]A small advertisement for FigJam AI is visible at the bottom of the page.

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:

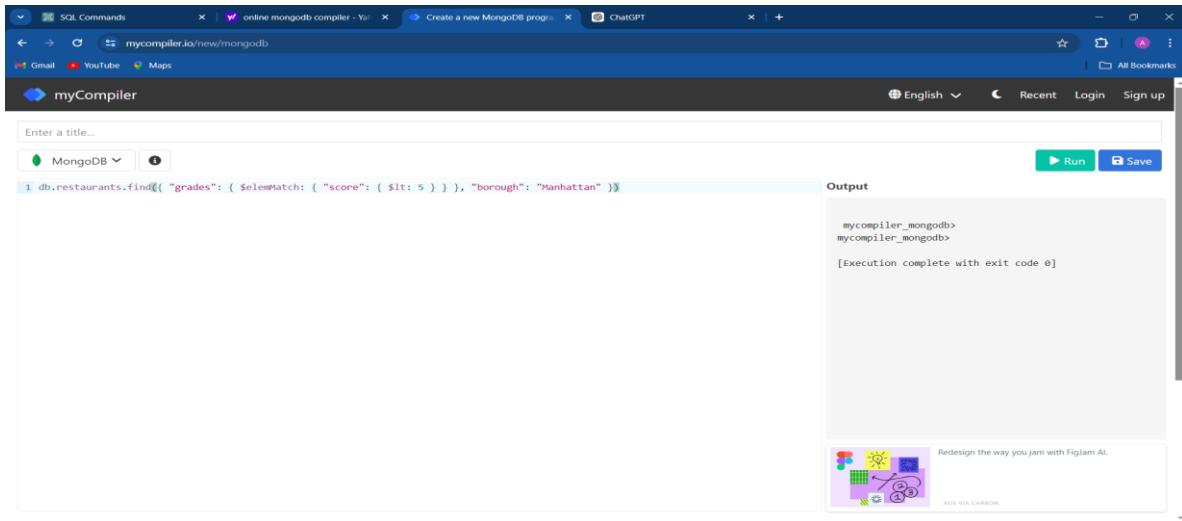


The screenshot shows a browser window with the URL mycompiler.io/new/mongodb. The main area contains a MongoDB query:db.restaurants.find({ "grades": { \$elemMatch: { "score": { \$lt: 5 } } } })The output panel on the right shows the results of the query execution:mycompiler_mongodb>
mycompiler_mongodb>
[Execution complete with exit code 0]A small advertisement for FigJam AI is visible at the bottom of the page.

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:

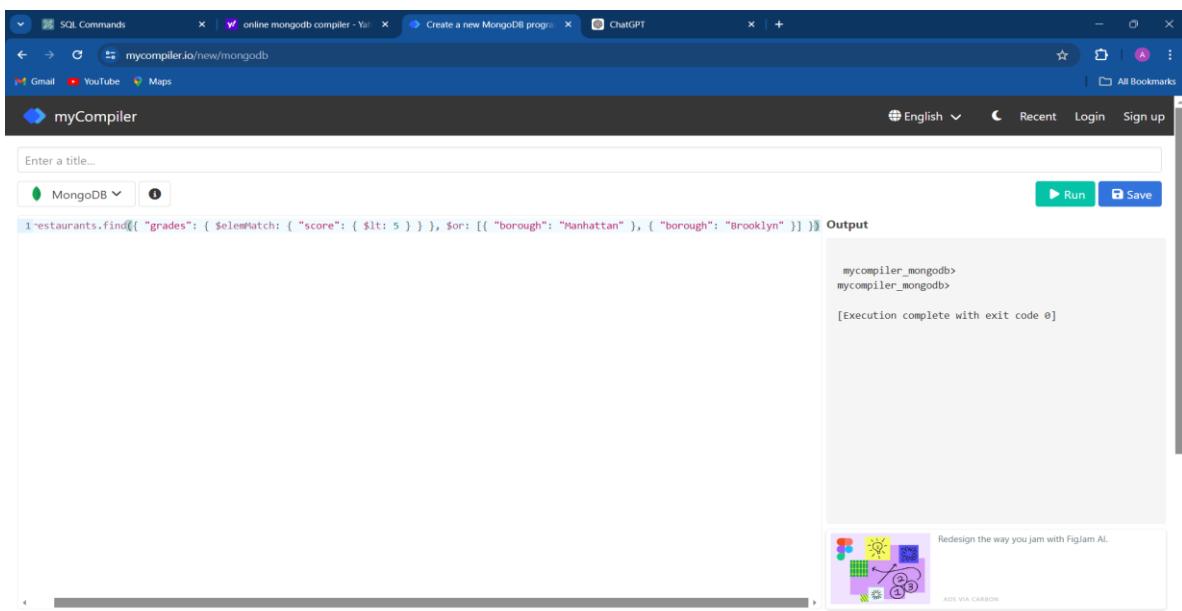


The screenshot shows a browser window with several tabs open. The active tab is titled 'myCompiler' and contains a MongoDB query. The query is: db.restaurants.find({ "grades": { \$elemMatch: { "score": { \$lt: 5 } } }, "borough": "Manhattan" }). Below the query, there is an 'Output' section. The output shows the command being run: mycompiler_mongodb> mycompiler_mongodb>. It also indicates that the execution was complete with exit code 0. A small advertisement for FigJam AI is visible at the bottom right of the output area.

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:

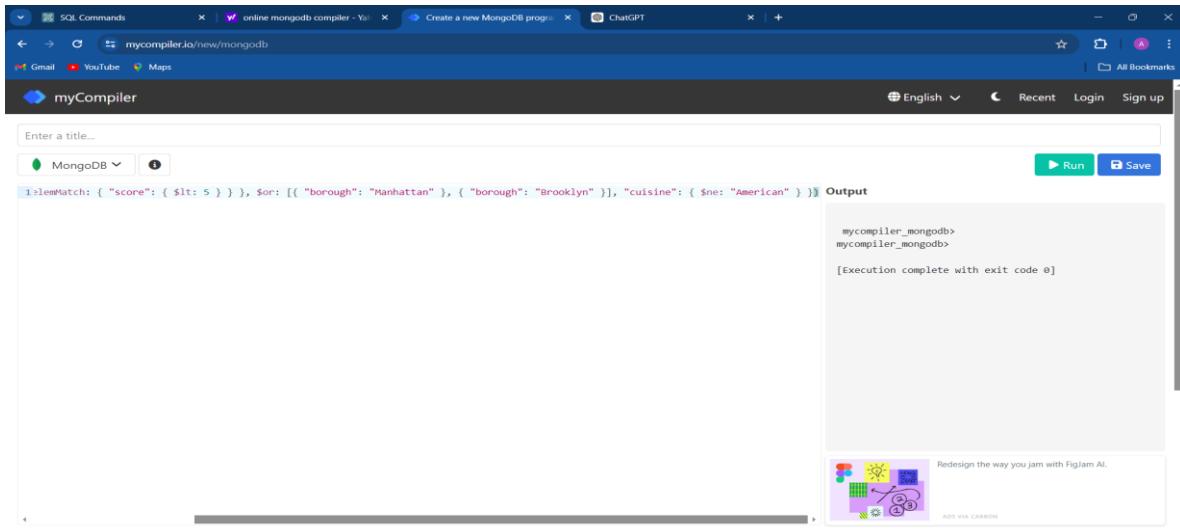


The screenshot shows a browser window with several tabs open. The active tab is titled 'myCompiler' and contains a MongoDB query. The query is: db.restaurants.find({ "grades": { \$elemMatch: { "score": { \$lt: 5 } } } }, { \$or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }] }). Below the query, there is an 'Output' section. The output shows the command being run: mycompiler_mongodb> mycompiler_mongodb>. It also indicates that the execution was complete with exit code 0. A small advertisement for FigJam AI is visible at the bottom right of the output area.

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:

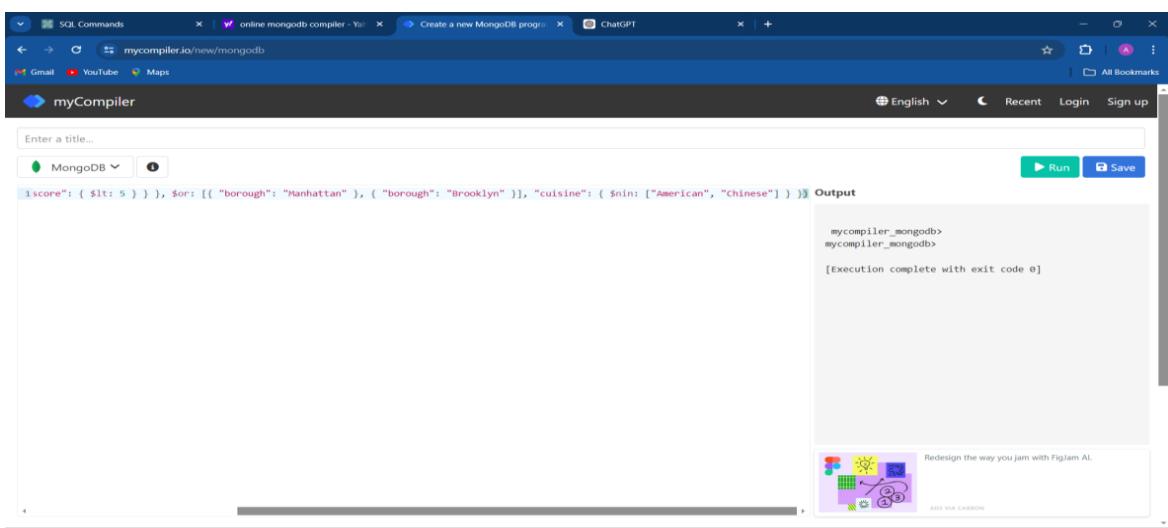


```
mycompiler_mongodb>
mycompiler_mongodb>
[Execution complete with exit code 0]
```

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

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

OUTPUT:

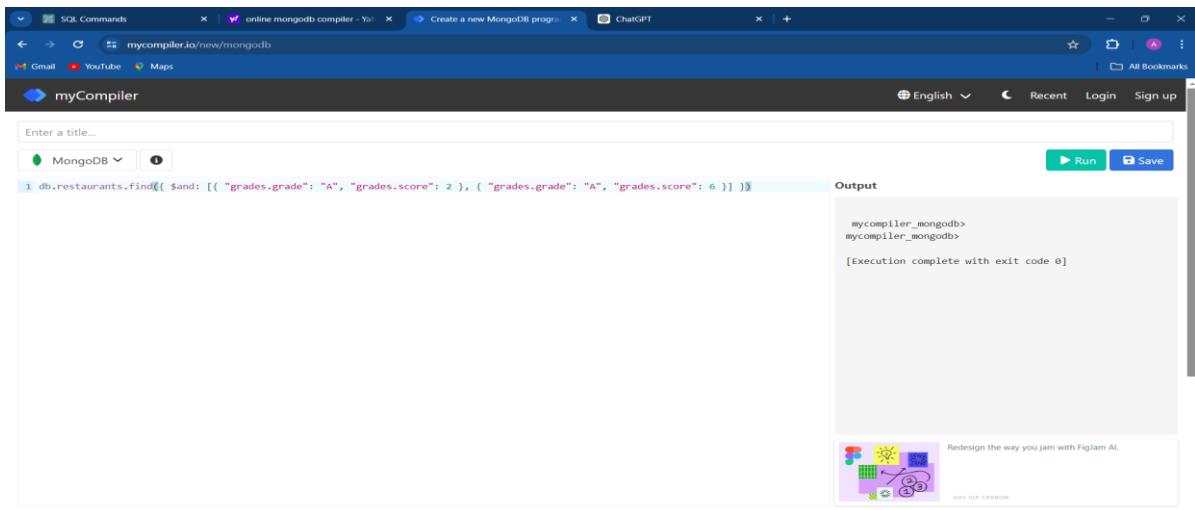


```
mycompiler_mongodb>
mycompiler_mongodb>
[Execution complete with exit code 0]
```

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 a browser window with the URL mycompiler.io/new/mongodb. The title bar says "myCompiler". The main area has a text input field with placeholder "Enter a title..." and a dropdown menu set to "MongoDB". Below the input field is a code editor containing the following MongoDB query:

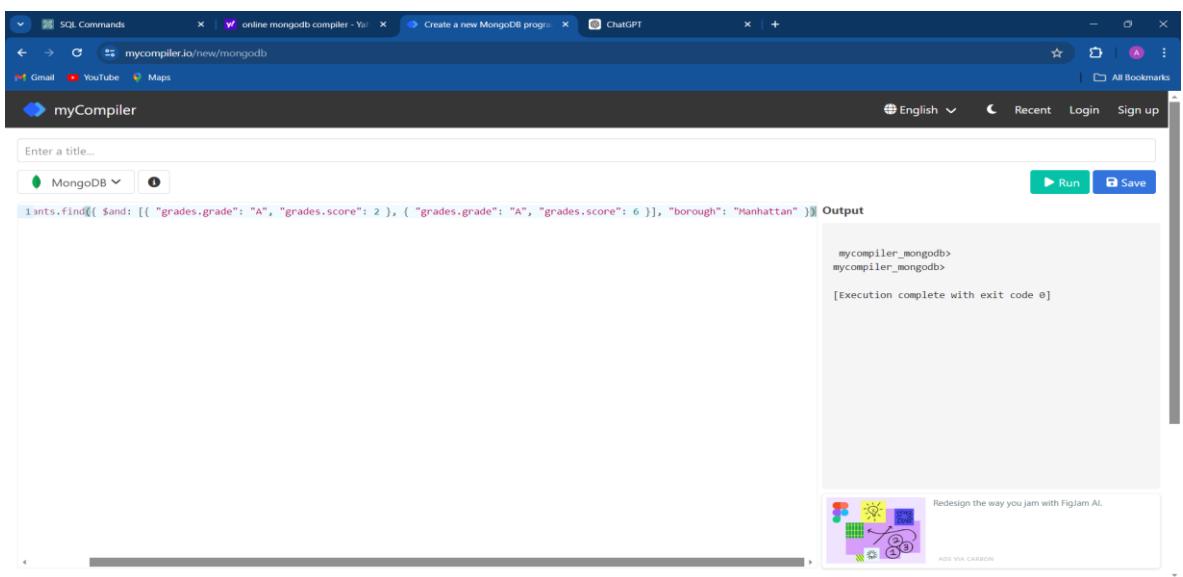
```
1 db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 } ] })
```

To the right of the code editor is an "Output" panel. It displays the command prompt "mycompiler_mongodb>" followed by "[Execution complete with exit code 0]". There is also a small advertisement for FigJam AI at the bottom right of the output panel.

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:



The screenshot shows a browser window with the URL mycompiler.io/new/mongodb. The title bar says "myCompiler". The main area has a text input field with placeholder "Enter a title..." and a dropdown menu set to "MongoDB". Below the input field is a code editor containing the following MongoDB query:

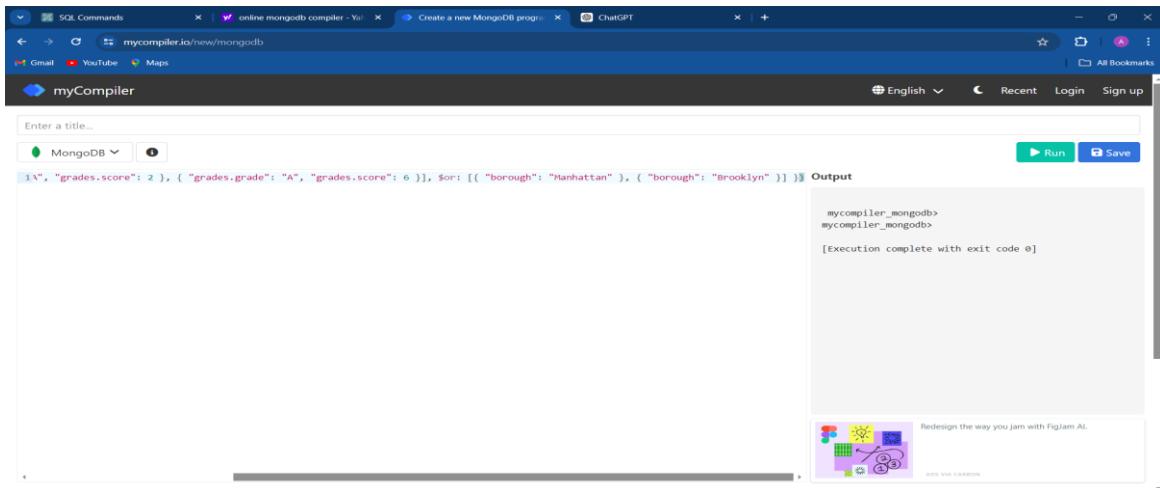
```
1 db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], "borough": "Manhattan" })
```

To the right of the code editor is an "Output" panel. It displays the command prompt "mycompiler_mongodb>" followed by "[Execution complete with exit code 0]". There is also a small advertisement for FigJam AI at the bottom right of the output panel.

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:



The screenshot shows a browser window with several tabs open. The active tab is titled 'myCompiler' and contains a MongoDB query. The query is: db.restaurants.find({ \$and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], \$or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }] }). Below the query, there is an 'Output' section. The output shows the command being run: mycompiler_mongodb> mycompiler_mongodb>. It also includes a message: [Execution complete with exit code 0]. There is a small advertisement for FigJam AI at the bottom right.

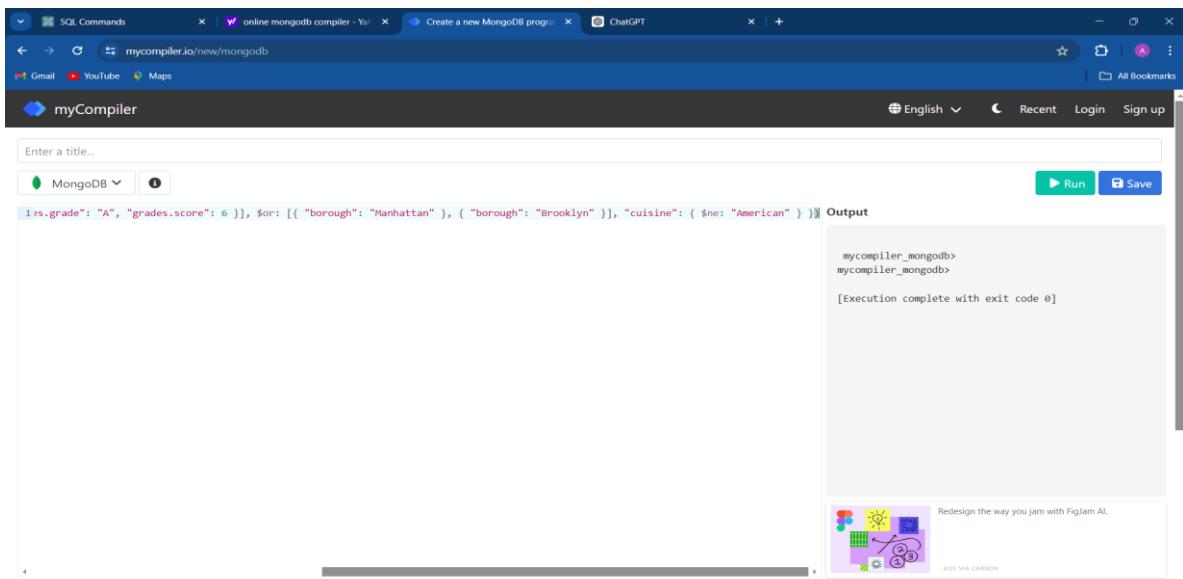
```
db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }] })
```

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

21. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American.

QUERY: db.restaurants.find({ \$and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], \$or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { \$ne: "American" } })

OUTPUT:



The screenshot shows a browser window with several tabs open. The active tab is titled 'myCompiler' and contains a MongoDB query. The query is: db.restaurants.find({ \$and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], \$or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { \$ne: "American" } }). Below the query, there is an 'Output' section. The output shows the command being run: mycompiler_mongodb> mycompiler_mongodb>. It also includes a message: [Execution complete with exit code 0]. There is a small advertisement for FigJam AI at the bottom right.

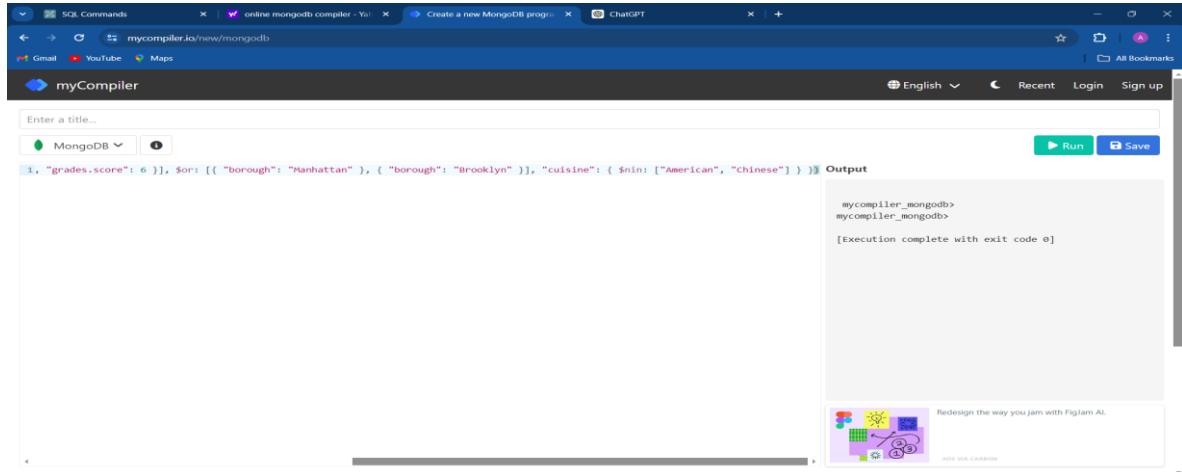
```
db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $ne: "American" } })
```

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

22. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American or Chinese.

QUERY: db.restaurants.find({ \$and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], \$or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { \$nin: ["American", "Chinese"] } })

OUTPUT:



The screenshot shows a browser window titled 'myCompiler' with a MongoDB query editor. The query is: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"] } })

```
Below the query, the output window shows the results of the execution:
```

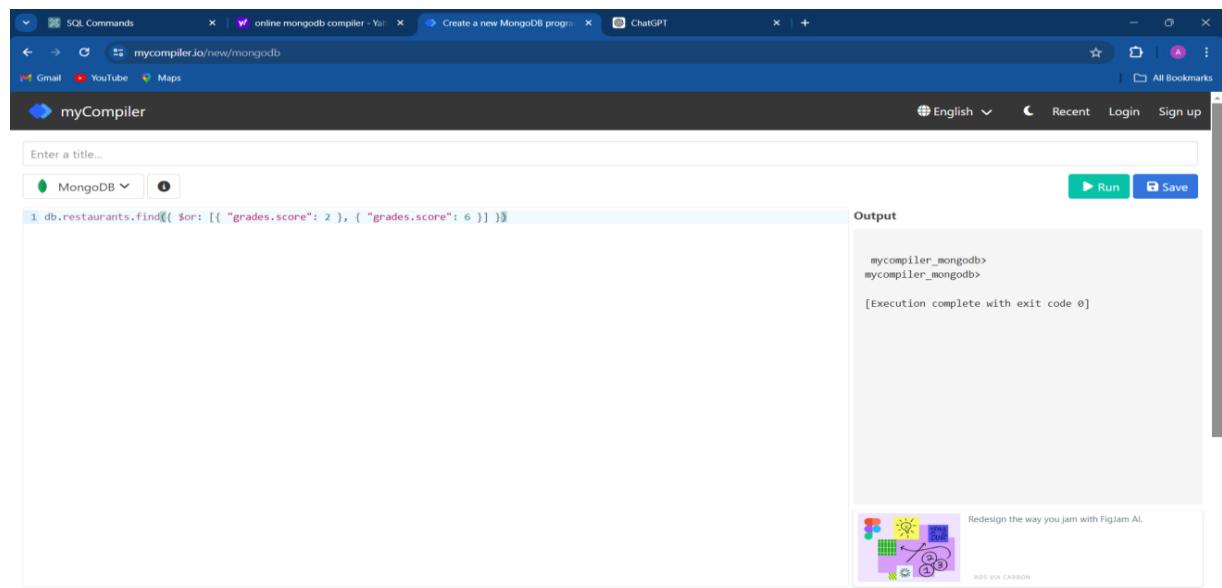
mycompiler_mongodb>
mycompiler_mongodb>
[Execution complete with exit code 0]

The browser also has tabs for 'SQL Commands', 'Create a new MongoDB program', and 'ChatGPT'. The address bar shows 'mycompiler.io/new/mongodb'. The bottom right corner features 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:



The screenshot shows a browser window titled 'myCompiler' with a MongoDB query editor. The query is:db.restaurants.find({ \$or: [{ "grades.score": 2 }, { "grades.score": 6 }] })

```
Below the query, the output window shows the results of the execution:
```

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

The browser also has tabs for 'SQL Commands', 'Create a new MongoDB program', and 'ChatGPT'. The address bar shows 'mycompiler.io/new/mongodb'. The bottom right corner features an advertisement for FigJam AI.

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

RESULT:

MONGO DB

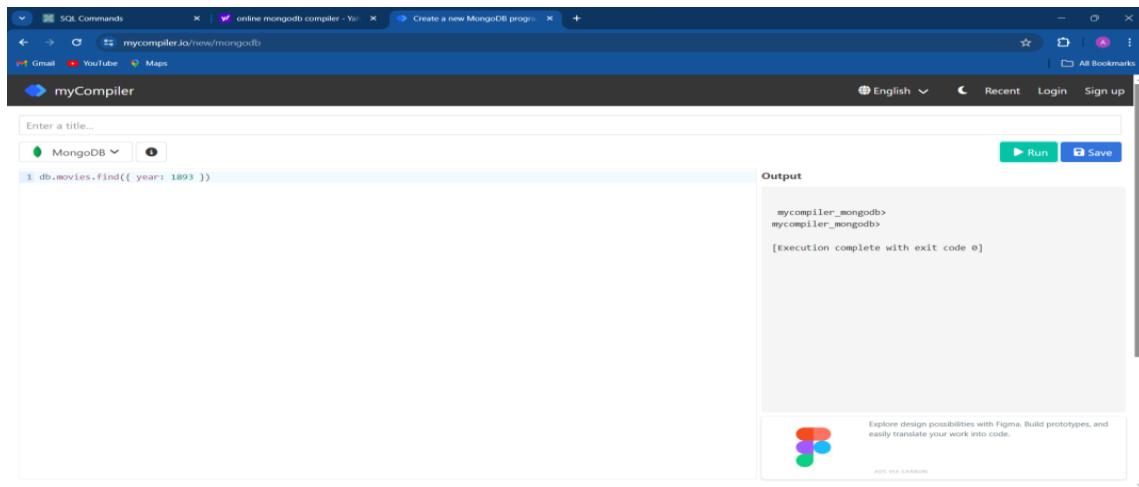
EXNO:20

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:



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

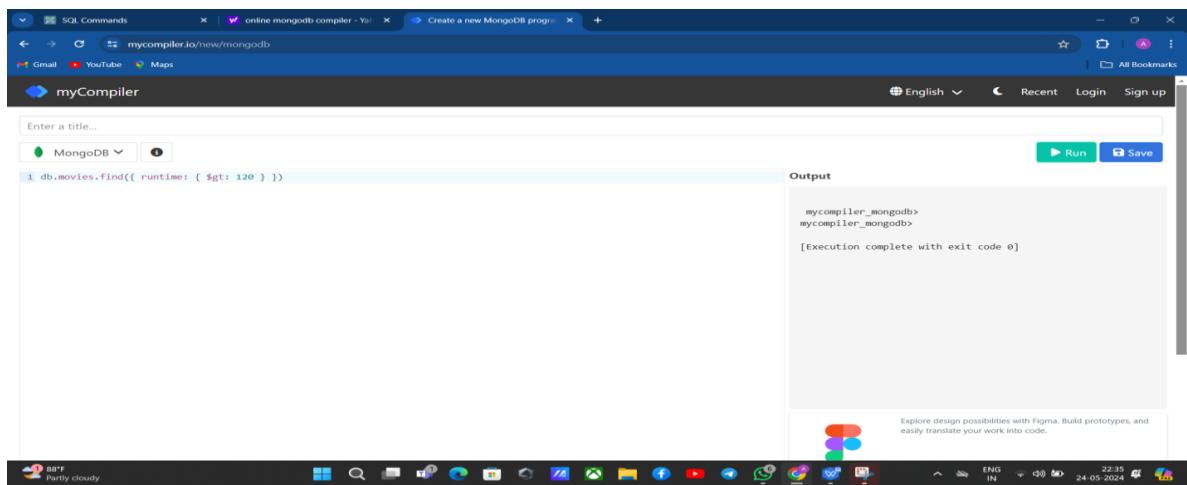
Output

```
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:



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

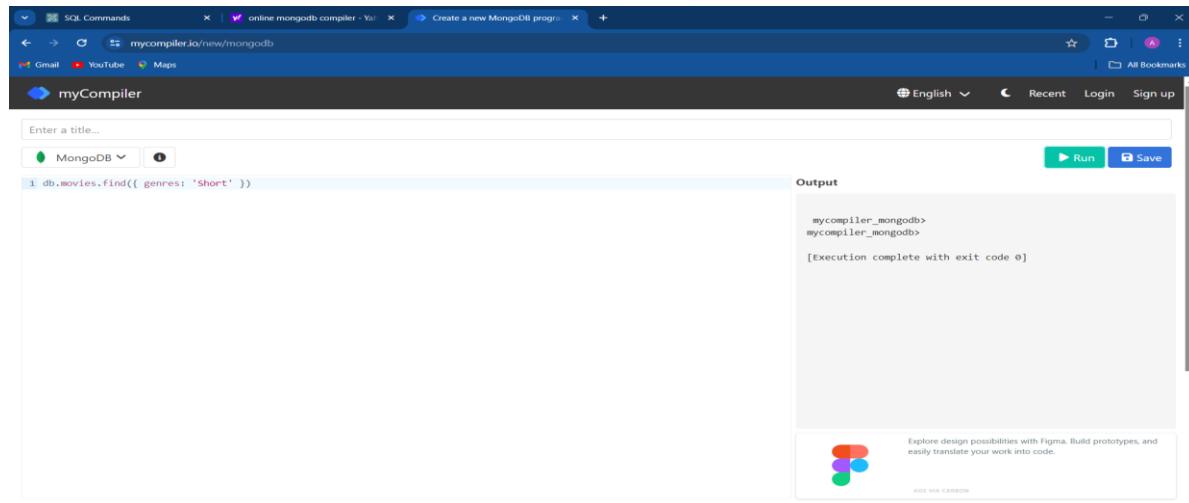
Output

```
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:

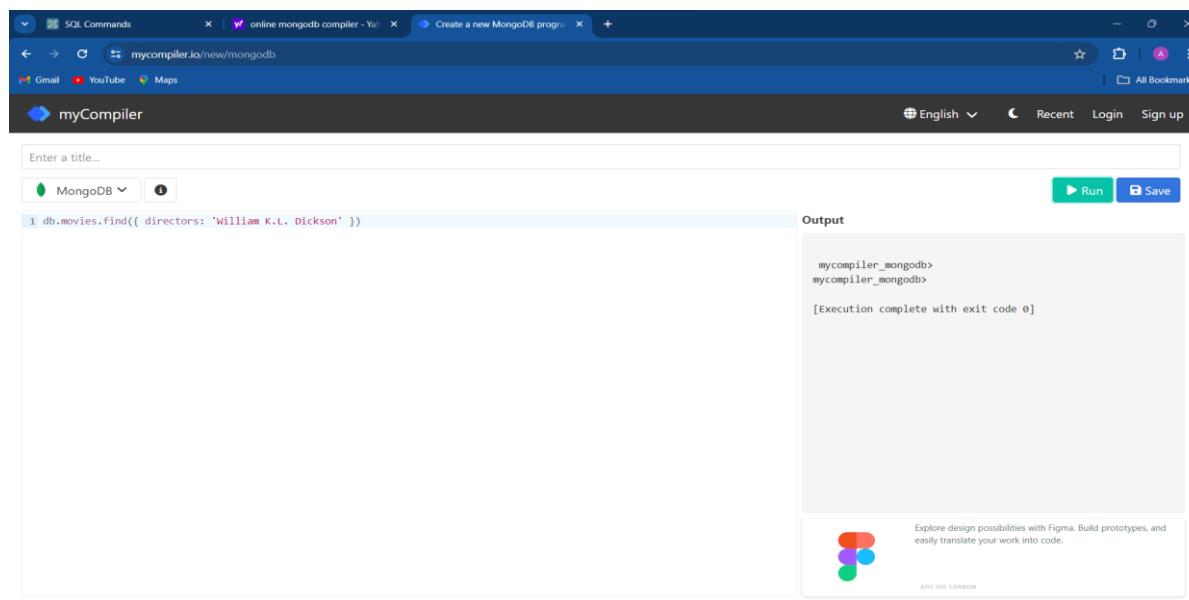


The screenshot shows a browser window with the URL mycompiler.io/new/mongodb. The title bar says "myCompiler". The main area has a text input field with placeholder "Enter a title..." and a dropdown menu set to "MongoDB". Below the input field is a code editor containing the MongoDB query: "db.movies.find({ genres: 'Short' })". To the right of the code editor is an "Output" panel. The output shows the command "mycompiler_mongodb>" followed by "mycompiler_mongodb>". Below this is the message "[Execution complete with exit code 0]". At the bottom of the output panel, there is an advertisement for Figma with the text "Explore design possibilities with Figma. Build prototypes, and easily translate your work into code." and a "ADS VIA CARBON" link.

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:

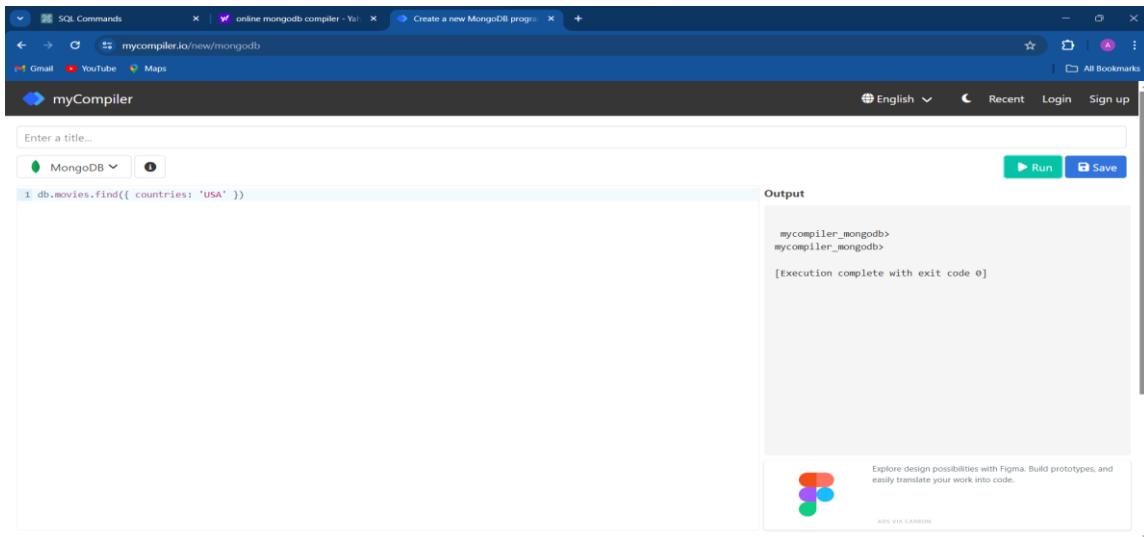


The screenshot shows a browser window with the URL mycompiler.io/new/mongodb. The title bar says "myCompiler". The main area has a text input field with placeholder "Enter a title..." and a dropdown menu set to "MongoDB". Below the input field is a code editor containing the MongoDB query: "db.movies.find({ directors: 'William K.L. Dickson' })". To the right of the code editor is an "Output" panel. The output shows the command "mycompiler_mongodb>" followed by "mycompiler_mongodb>". Below this is the message "[Execution complete with exit code 0]". At the bottom of the output panel, there is an advertisement for Figma with the text "Explore design possibilities with Figma. Build prototypes, and easily translate your work into code." and a "ADS VIA CARBON" link.

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:

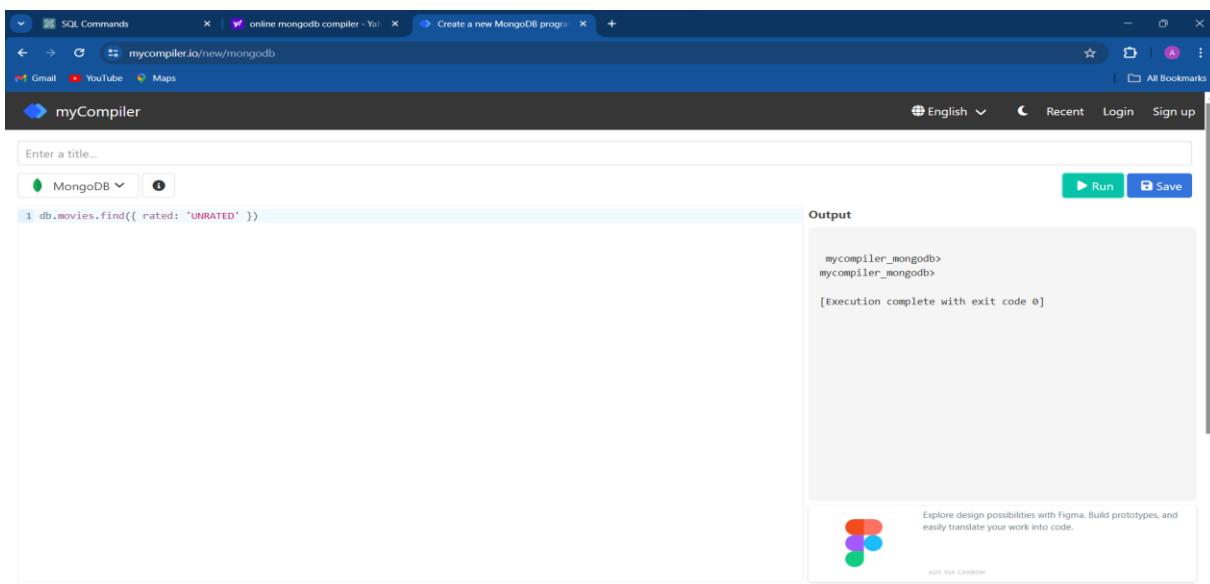


The screenshot shows a web-based MongoDB interface. In the top navigation bar, there are tabs for "SQL Commands", "online mongodb compiler - YouTube", and "Create a new MongoDB program". Below the navigation is a search bar labeled "Enter a title..." and a dropdown menu set to "MongoDB". On the left, a code editor window contains the MongoDB query: "db.movies.find({ countries: 'USA' })". To the right of the code editor is an "Output" window showing the results of the query execution. The output text is: "mycompiler_mongodb> mycompiler_mongodb> [Execution complete with exit code 0]". At the bottom of the interface, there is an advertisement for Figma and a note about ads via Carbon.

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

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

OUTPUT:

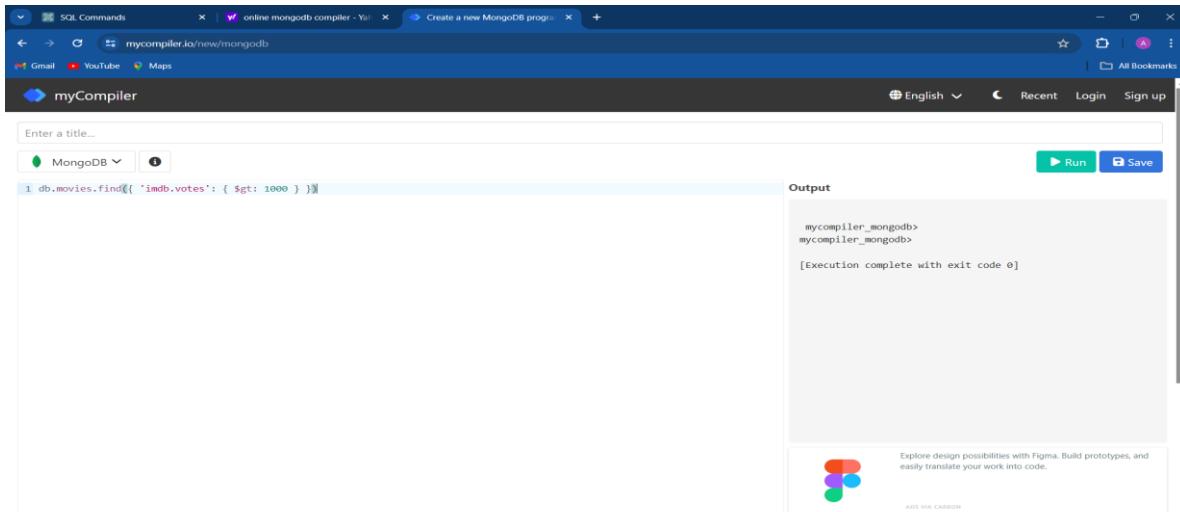


The screenshot shows a web-based MongoDB interface, identical to the one above. The top navigation bar includes "SQL Commands", "online mongodb compiler - YouTube", and "Create a new MongoDB program". The search bar and MongoDB dropdown are also present. The code editor window contains the query "db.movies.find({ rated: 'UNRATED' })". The "Output" window shows the results: "mycompiler_mongodb> mycompiler_mongodb> [Execution complete with exit code 0]". An advertisement for Figma is at the bottom.

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

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

OUTPUT:

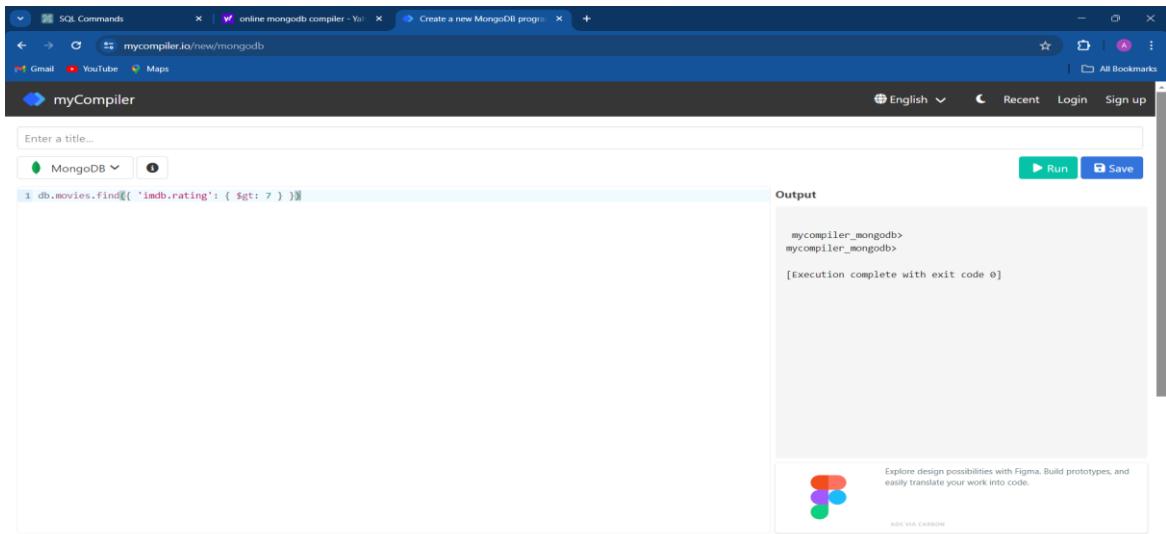


The screenshot shows a web-based MongoDB interface. In the top navigation bar, there are tabs for "SQL Commands", "online mongodb compiler - Yol", and "Create a new MongoDB program". Below the tabs, there are links for "Gmail", "YouTube", and "Maps". The main area has a title "myCompiler" and a sub-section "myCompiler_mongodb". A search bar says "Enter a title...". A dropdown menu shows "MongoDB" is selected. On the right, there are "Run" and "Save" buttons. The code input field contains the query: "db.movies.find({ 'imdb.votes': { \$gt: 1000 } })". To the right, under "Output", the results are displayed: "mycompiler_mongodb> mycompiler_mongodb> [Execution complete with exit code 0]". At the bottom right of the interface, there is an advertisement for Figma.

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:

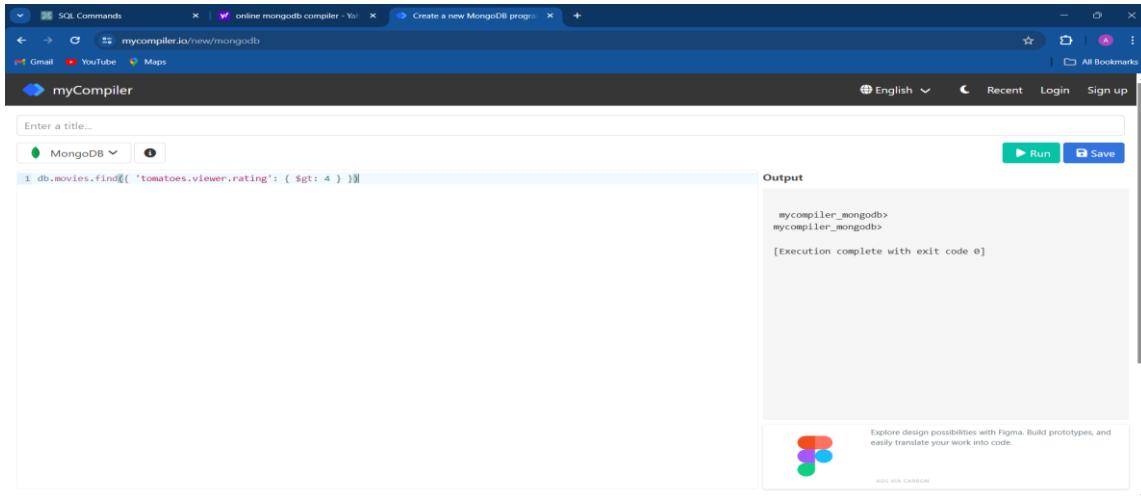


This screenshot is identical to the one above, showing the same interface and the execution of the query "db.movies.find({ 'imdb.rating': { \$gt: 7 } })". The output shows the same results: "mycompiler_mongodb> mycompiler_mongodb> [Execution complete with exit code 0]". The Figma advertisement at the bottom right is also present.

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:

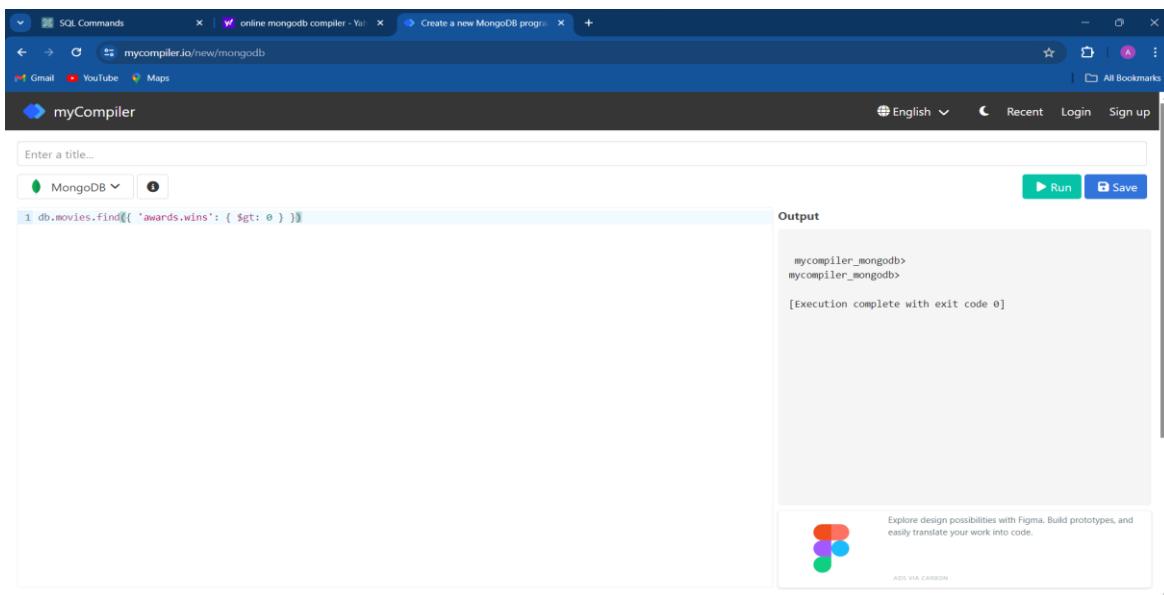


The screenshot shows a web browser window with the URL "mycompiler.io/new/mongodb". The title bar says "myCompiler". The main area has a text input field containing the MongoDB query: "db.movies.find({ 'tomatoes.viewer.rating': { \$gt: 4 } })". Below the input field are "Run" and "Save" buttons. To the right is an "Output" panel showing the results of the query execution. The output text is: "mycompiler_mongodb> mycompiler_mongodb> [Execution complete with exit code 0]". There is also a small advertisement for Figma at the bottom right.

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

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

OUTPUT:

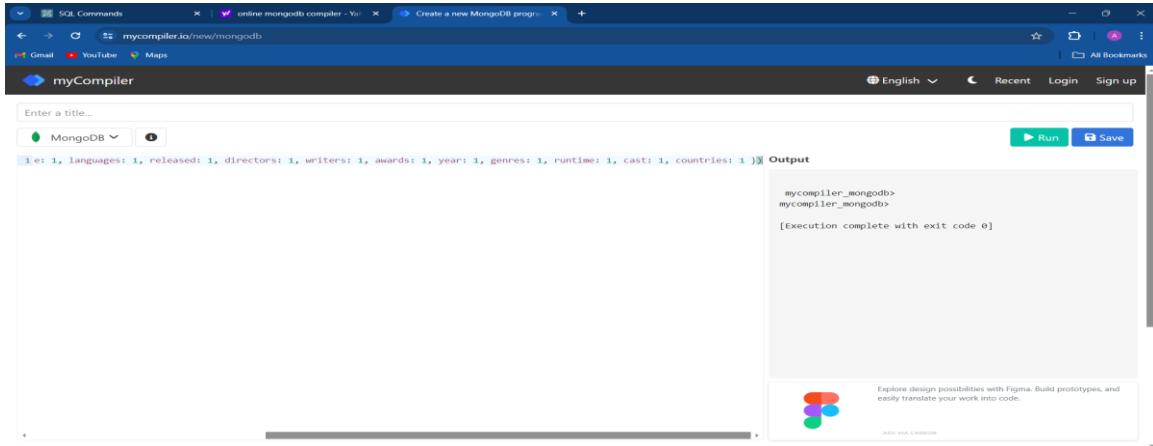


The screenshot shows a web browser window with the URL "mycompiler.io/new/mongodb". The title bar says "myCompiler". The main area has a text input field containing the MongoDB query: "db.movies.find({ 'awards.wins': { \$gt: 0 } })". Below the input field are "Run" and "Save" buttons. To the right is an "Output" panel showing the results of the query execution. The output text is: "mycompiler_mongodb> mycompiler_mongodb> [Execution complete with exit code 0]". There is also a small advertisement for Figma at the bottom right.

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 a web-based MongoDB interface titled "myCompiler". In the top navigation bar, there are links for "SQL Commands", "mycompiler.io/new/mongodb", and "Create a new MongoDB program". Below the navigation is a search bar labeled "Enter a title...". A dropdown menu is open, showing "MongoDB" as the selected option. To the right of the dropdown are "Run" and "Save" buttons. The main area contains a code editor with the following query:

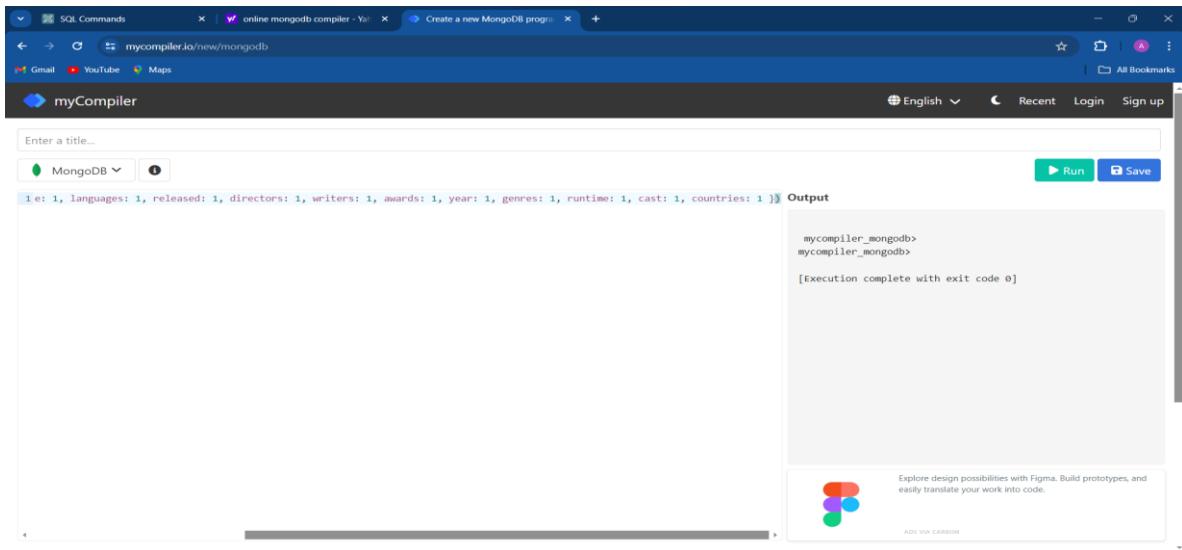
```
1 e: 1, languages: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 })
```

Below the code editor is an "Output" panel. The output shows the command "mycompiler_mongodb>" followed by "[Execution complete with exit code 0]". There is also a small advertisement for Figma at the bottom of the panel.

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:

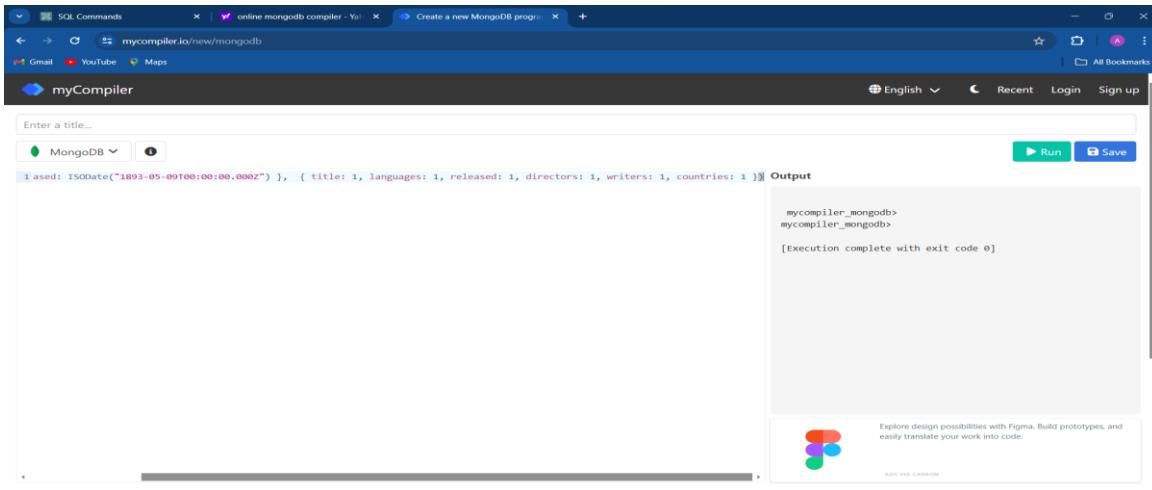


This screenshot is identical to the previous one, showing the same interface and the execution of the same query. The only difference is the output message, which now includes the name "Charles Kayser" in the list of cast members.

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:

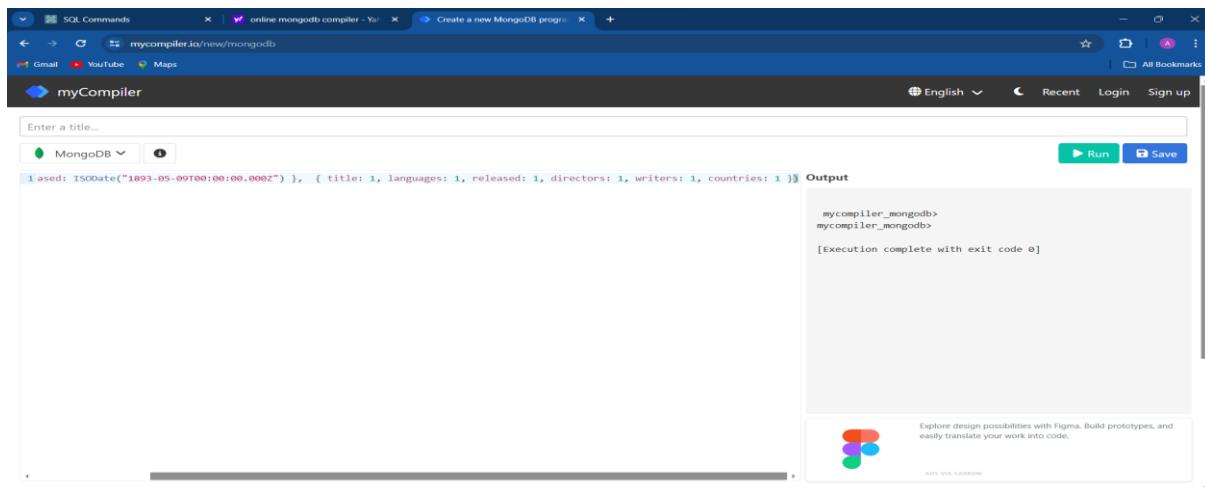


The screenshot shows a web-based MongoDB interface. In the top-left, there's a search bar labeled "Enter a title...". Below it, a dropdown menu is set to "MongoDB". On the right side of the interface, there are two buttons: "Run" and "Save". The main area contains the MongoDB command: "db.movies.find({ released: ISODate('1893-05-09T00:00:00.000Z') }, { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 })". To the right of the command, there's an "Output" section. The output shows the command being run: "mycompiler_mongodb> mycompiler_mongodb> [Execution complete with exit code 0]". At the bottom right of the interface, there's an advertisement for Figma.

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:



This screenshot is identical to the one above, showing the same MongoDB interface and command. The difference is in the output. The output now shows the results of the query, which include movies like "The Story of the Kelly Gang" and "A Trip to the Moon", both of which contain the word "scene" in their titles.

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

RESULT: