

Creating and Managing Tables

EX_NO:1

DATE:

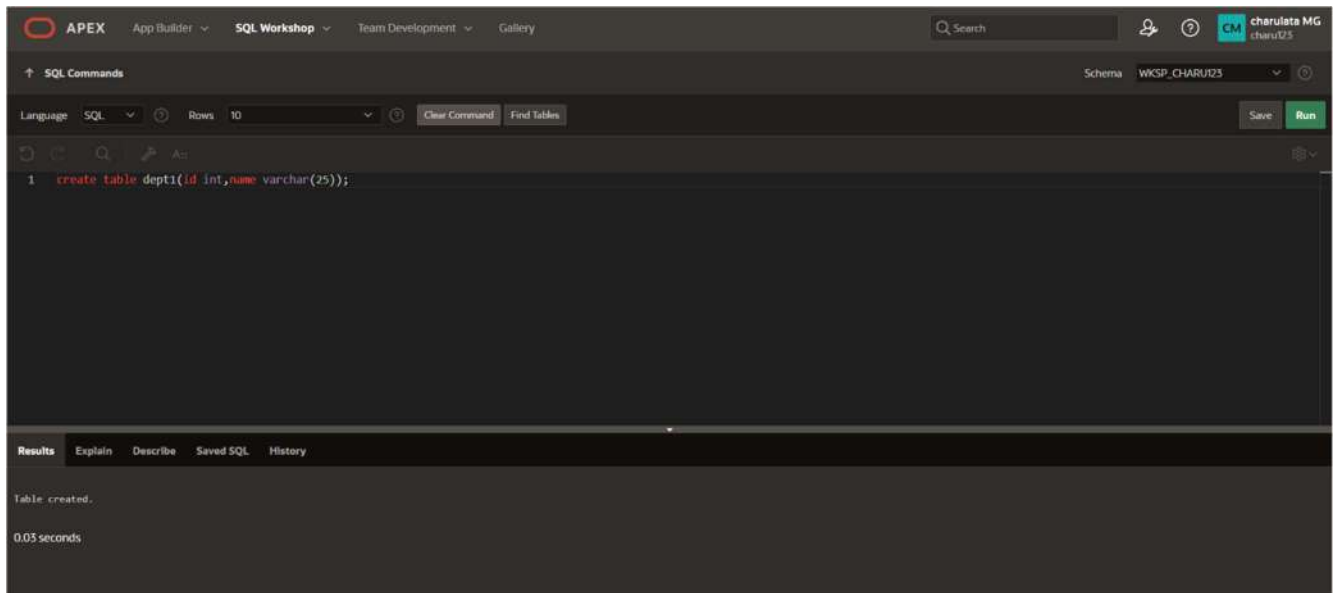
1.Create the DEPT table based on the DEPARTMENT following the table instance chart below. Confirm that the table is created.

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

QUERY:

Create table dept1(id number(7),name varchar2(25));

OUTPUT:



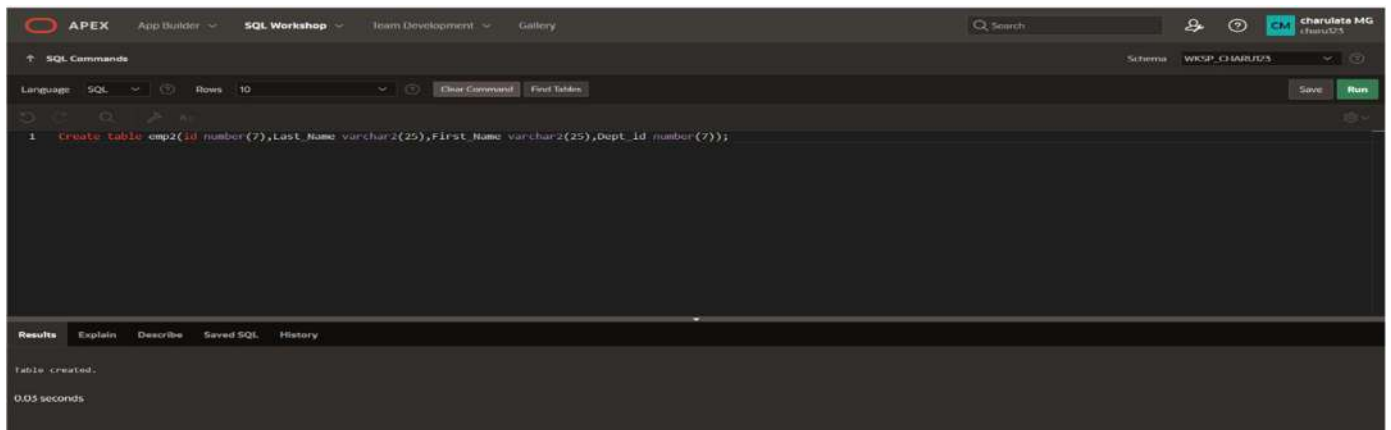
2..Create the EMP table based on the following instance chart. Confirm that the table is created.

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

QUERY:

Create table emp(id number(7),Last_Name varchar2(25),First_Name varchar2(25),Dept_id number(7));

OUTPUT:



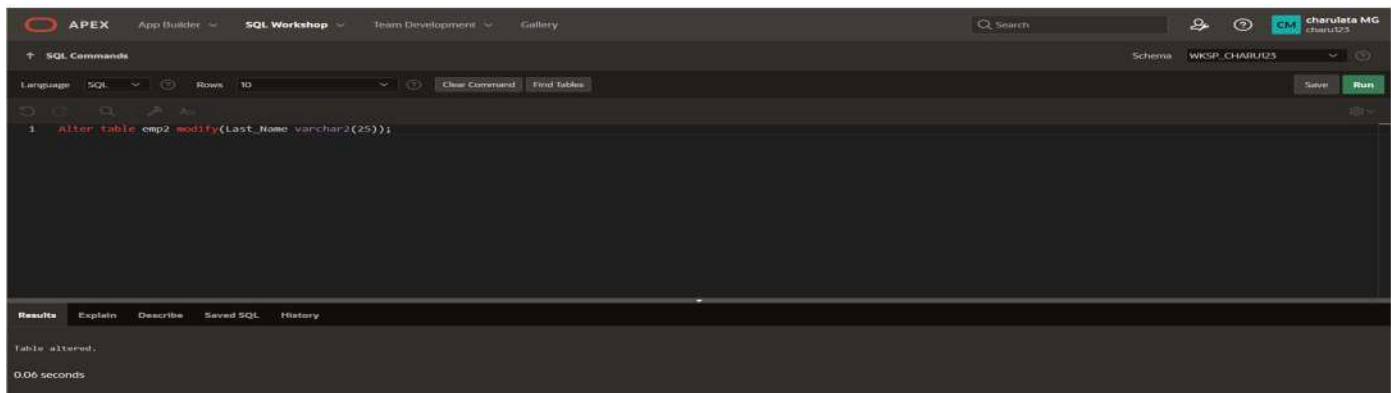
The screenshot shows the APEX SQL Workshop interface. The SQL Commands pane contains the following statement: `1 Create table emp2(id number(7),Last_Name varchar2(25),First_Name varchar2(25),Dept_id number(7));`. The Results pane shows the message "Table created." and the execution time "0.05 seconds".

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

QUERY:

Alter table emp modify(Last_Name varchar2(25));

OUTPUT:



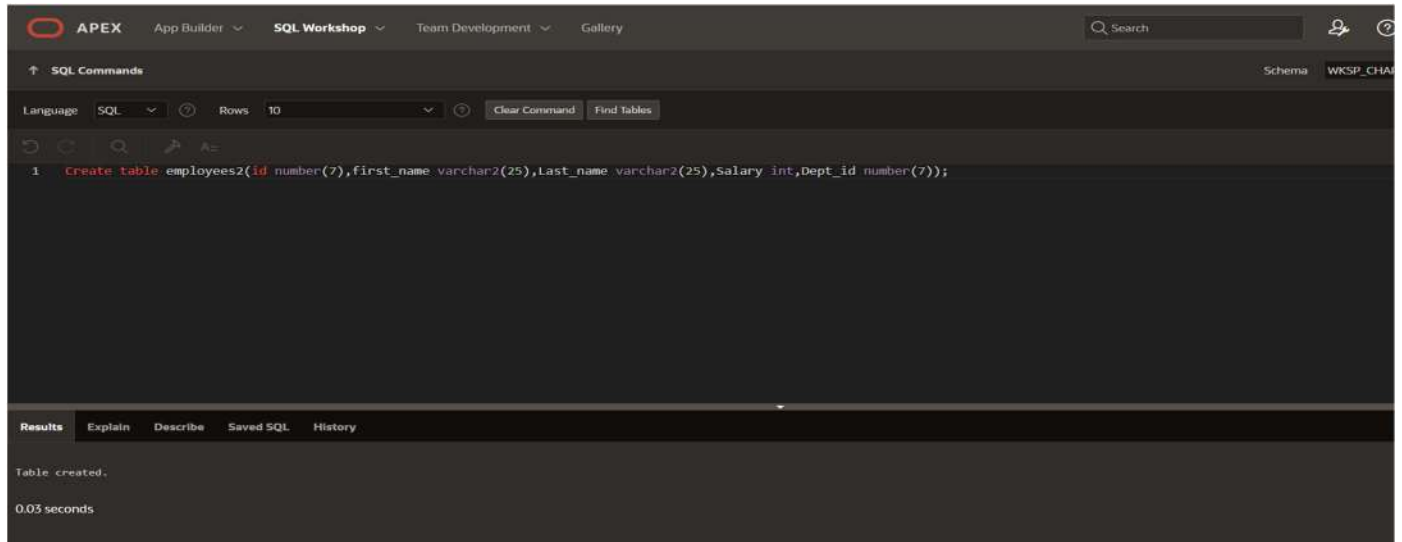
The screenshot shows the APEX SQL Workshop interface. The SQL Commands pane contains the following statement: `1 Alter table emp2 modify(Last_Name varchar2(25));`. The Results pane shows the message "Table altered." and the execution time "0.06 seconds".

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

QUERY:

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

OUTPUT:

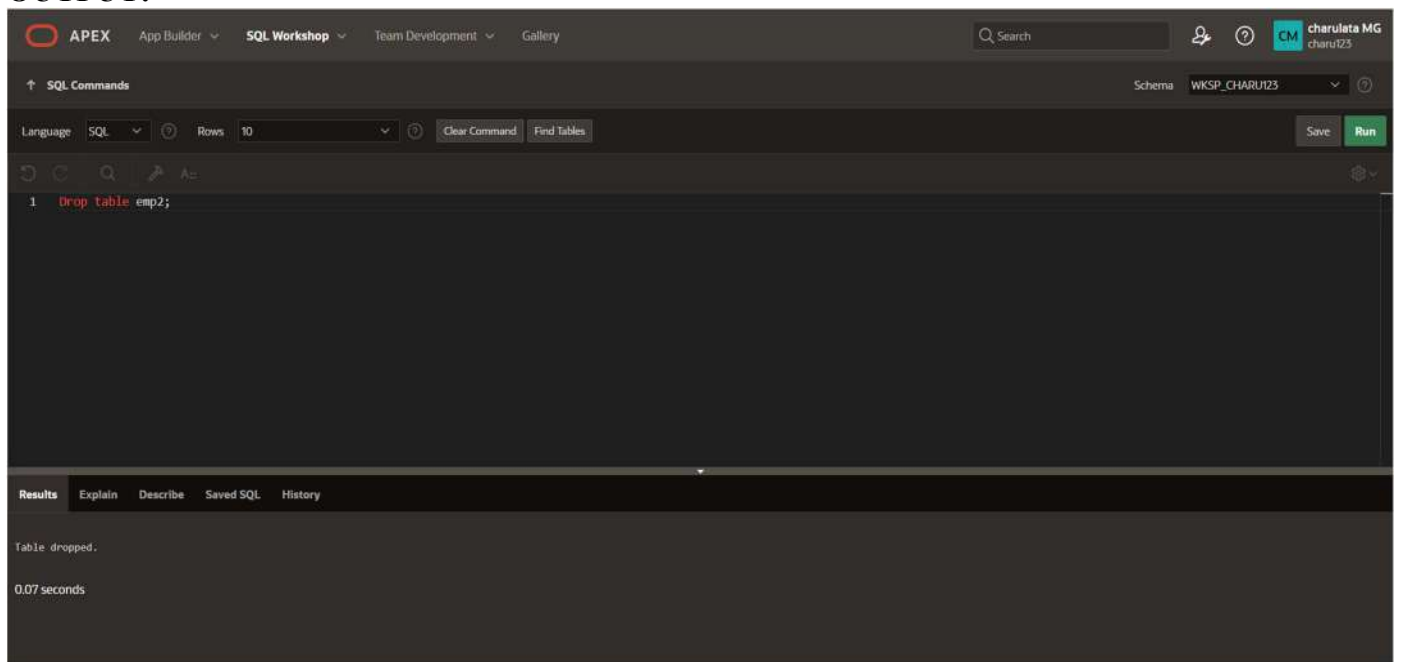


5. Drop the EMP table.

QUERY:

Drop table emp;

OUTPUT:

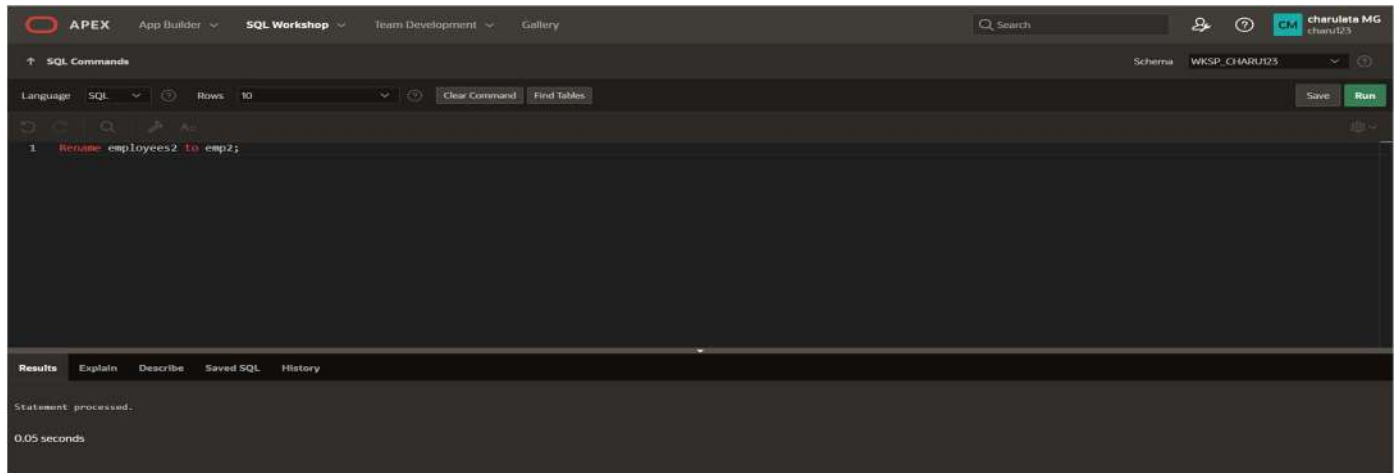


6.Rename the EMPLOYEES2 table as EMP.

QUERY:

Rename employees2 to emp;

OUTPUT:



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

QUERY:

comment on table dept is 'Department info';

comment on table emp is Employee info';

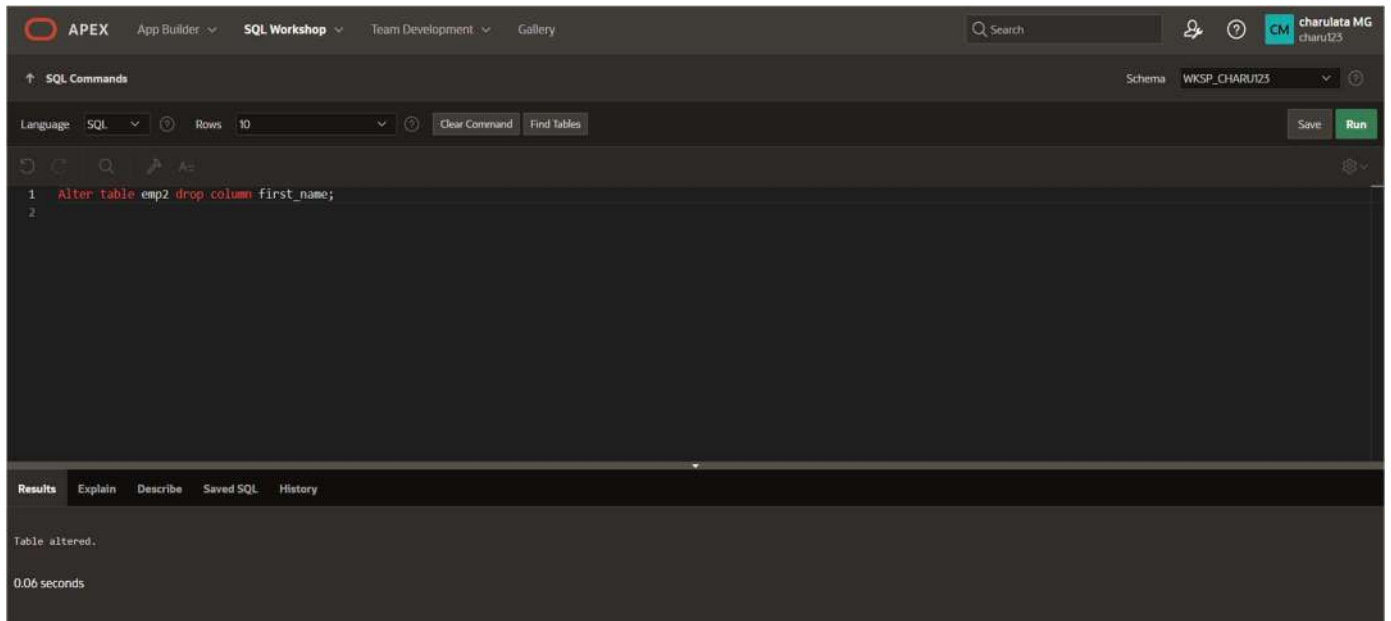
OUTPUT:

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 APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'charulata MG' are on the right. The 'SQL Commands' panel is active, showing the command 'Alter table emp2 drop column first_name;'. The 'Results' tab at the bottom displays the message 'Table altered.' and the execution time '0.06 seconds'.

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

RESULT:

MANIPULATING DATA

EX_NO:2

DATE:

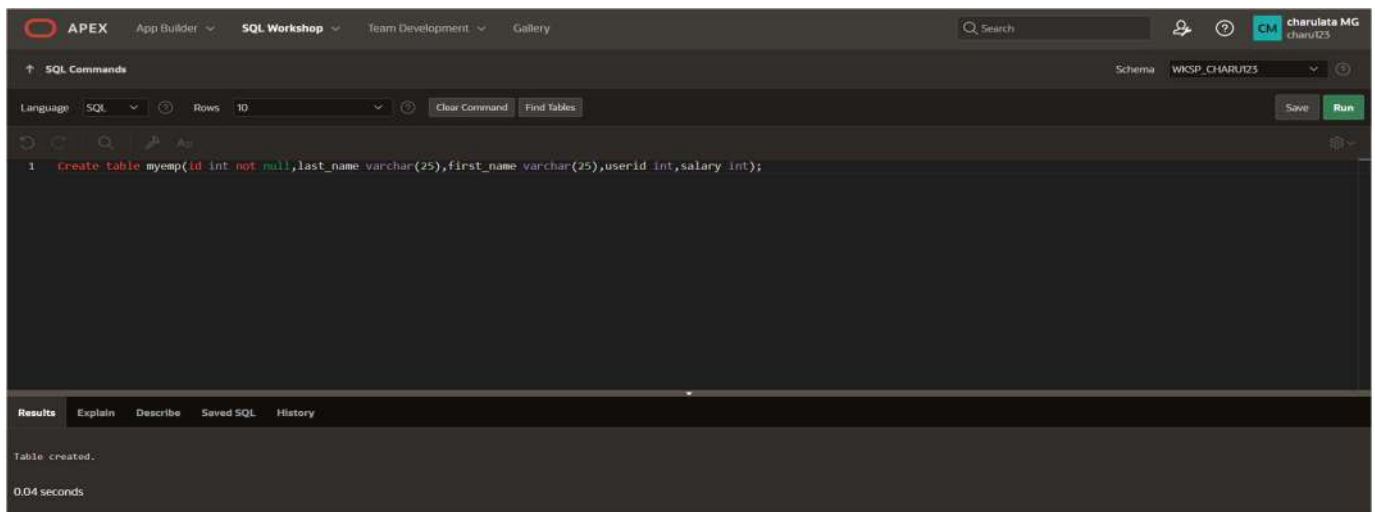
1.Create MY EMPLOYEE table with the following structure

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

QUERY:

Create table myemp(id int not null,last_name varchar(25),first_name varchar(25),userid int,salary int);

OUTPUT:



The screenshot displays the Oracle APEX SQL Workshop interface. At the top, the navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'charulata MG' are also visible. The main panel is titled 'SQL Commands' and shows the 'Schema' as 'WKSP_CHARUT23'. The 'Language' is set to 'SQL' and 'Rows' to '10'. The SQL command entered is: `1 create table myemp(id int not null,last_name varchar(25),first_name varchar(25),userid int,salary int);`. Below the command, the 'Results' tab is active, displaying the message 'Table created.' and the execution time '0.04 seconds'.

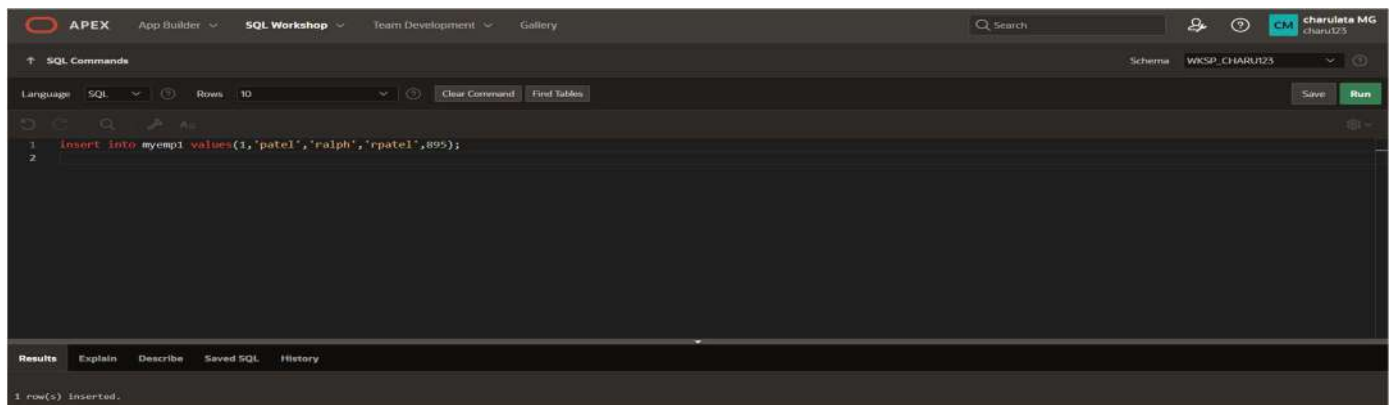
2.Add the first and second rows data to MY_EMPLOYEE table from the following sample data.

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 myemp1 values(1,'patel','ralph','rpatel',895);  
insert into myemp1 values(2,'dancs','betty','bdancs',860);
```

OUTPUT:



3.Display the table with values.

QUERY:

```
select*from myemp1  
order by id;
```

OUTPUT:

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

```
1 select*from myemp1  
2 order by id;
```


The Results pane shows the output as a table with columns: ID, LAST_NAME, FIRST_NAME, USERID, and SALARY. The data is as follows:

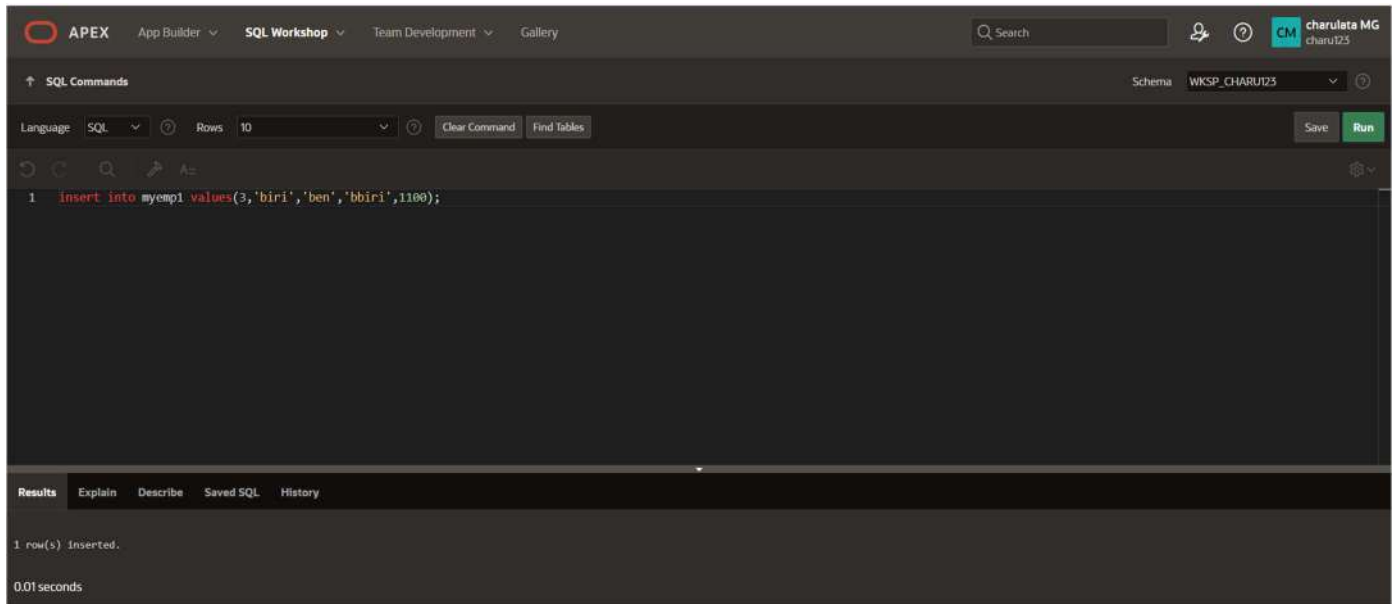
ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	patel	ralph	rpatel	895
2	dancs	betty	bdancs	860

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

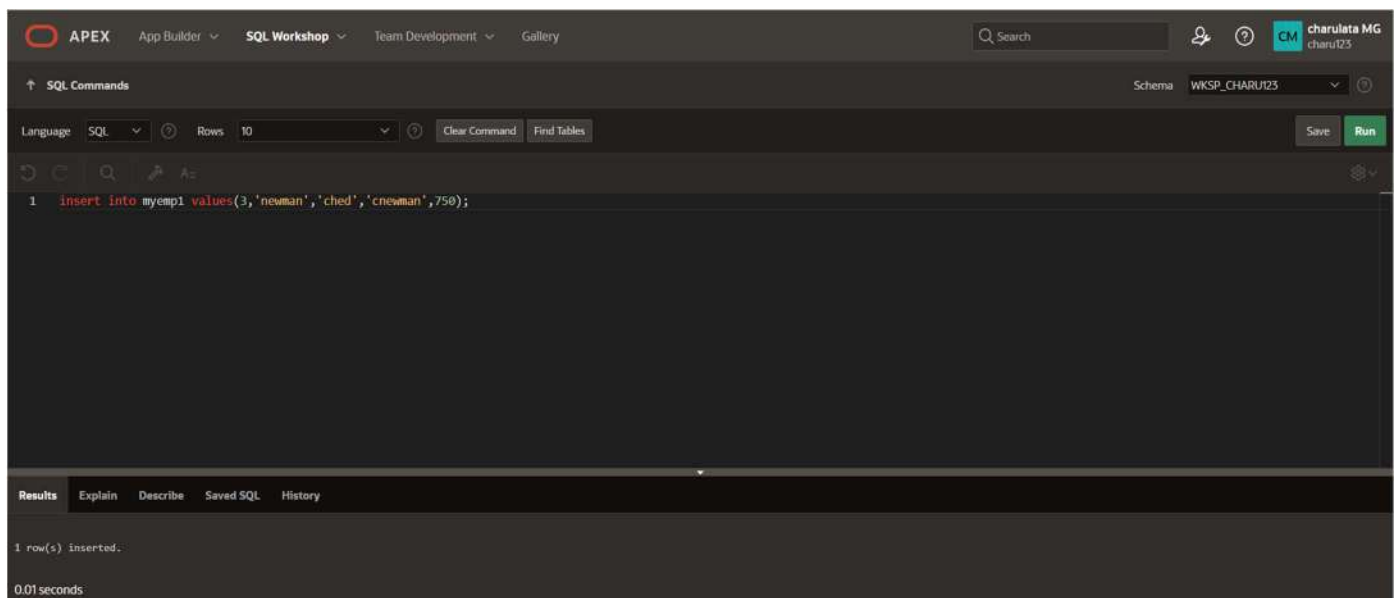
QUERY:

```
insert into myemp1 values(3,'biri','ben','bbiri',1100);  
insert into myemp1 values(3,'newman','ched','cnewman',750);
```

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'charulata MG charu123' are on the right. The 'SQL Commands' tab is active, showing a schema dropdown set to 'WKSP_CHARU123'. The command area contains the SQL statement: `1 insert into myemp1 values(3,'biri','ben','bbiri',1100);`. The 'Run' button is highlighted in green. Below the command area, the 'Results' tab is selected, displaying the output: '1 row(s) inserted.' and '0.01 seconds'.



This screenshot is similar to the one above, showing the APEX SQL Workshop interface. The 'SQL Commands' tab is active, and the command area contains the SQL statement: `1 insert into myemp1 values(3,'newman','ched','cnewman',750);`. The 'Run' button is highlighted in green. The 'Results' tab below shows the output: '1 row(s) inserted.' and '0.01 seconds'.

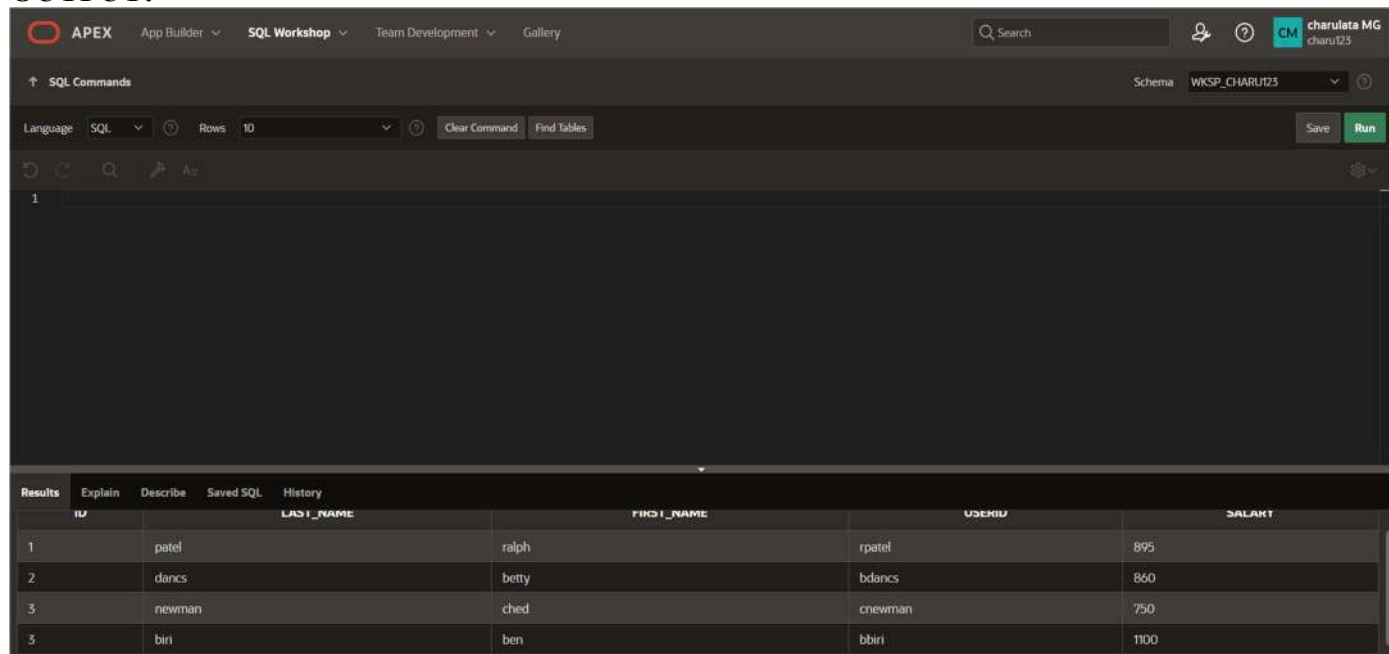
5. Make the data additions permanent.

QUERY:

```
select * from myemp1
```

order by id;

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'charulata MG' are on the right. The 'SQL Commands' section shows the schema 'WKSP_CHARUT23'. The command area contains a single line: 'select * from myemp1'. Below the command area, the 'Results' tab is active, displaying a table with 5 rows and 6 columns: ID, LAST_NAME, FIRST_NAME, USERID, and SALARY. The data is as follows:

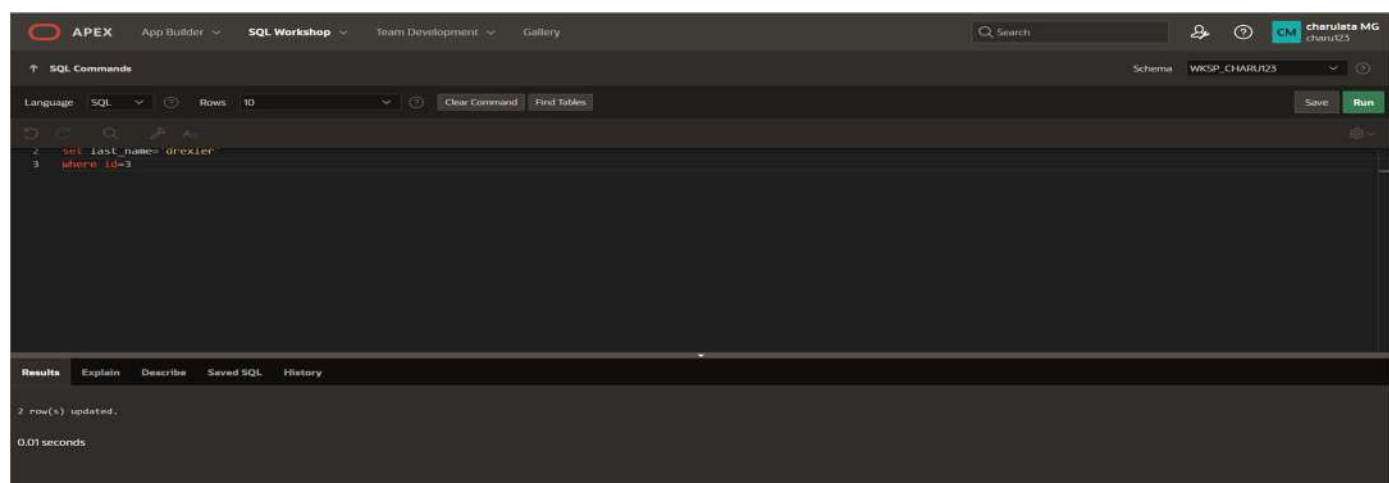
ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	patel	ralph	rpatel	895
2	dancs	betty	bdancs	860
3	newman	ched	cnewman	750
3	biri	ben	bbiri	1100

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

QUERY:

```
update myemp1 set last_name='drexler' where id=3
```

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The top navigation bar is the same as the previous screenshot. The 'SQL Commands' section shows the schema 'WKSP_CHARUT23'. The command area contains two lines: 'update myemp1 set last_name='drexler'' and 'where id=3'. Below the command area, the 'Results' tab is active, displaying the message '2 row(s) updated.' and '0.01 seconds'.

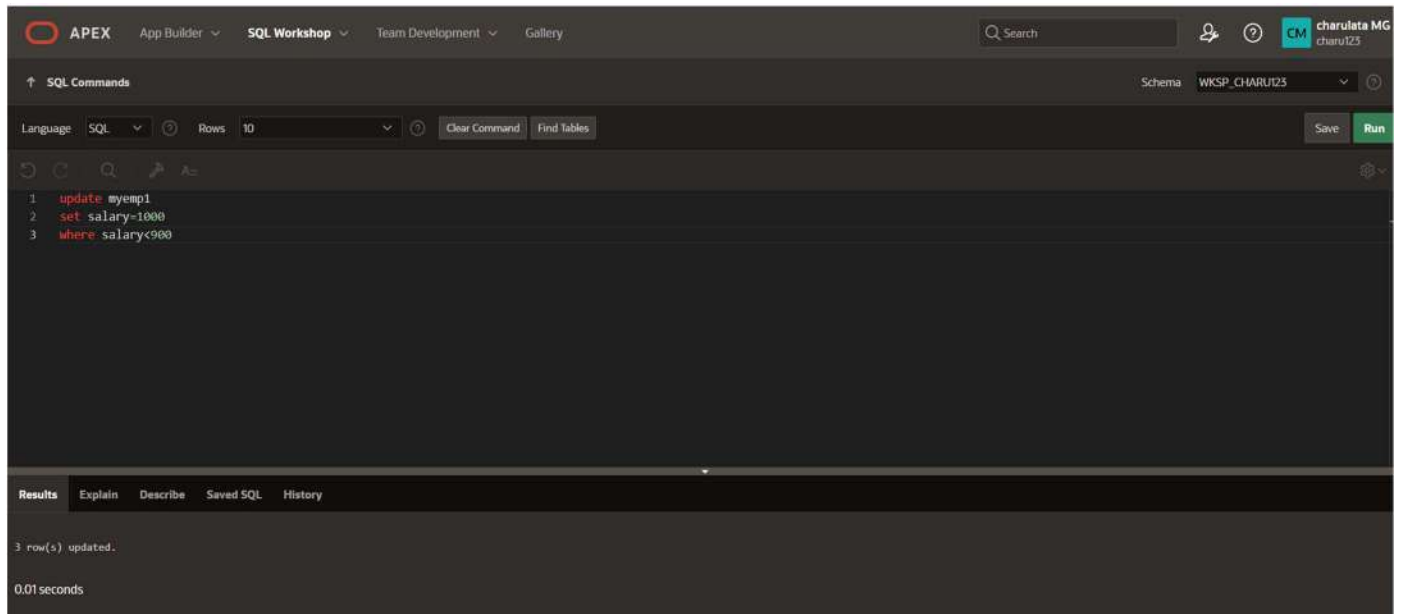
Results	Explain	Describe	Saved SQL	History
2 row(s) updated.				
0.01 seconds				

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

QUERY:

update myemp1 set salary=1000 where salary<900

OUTPUT:

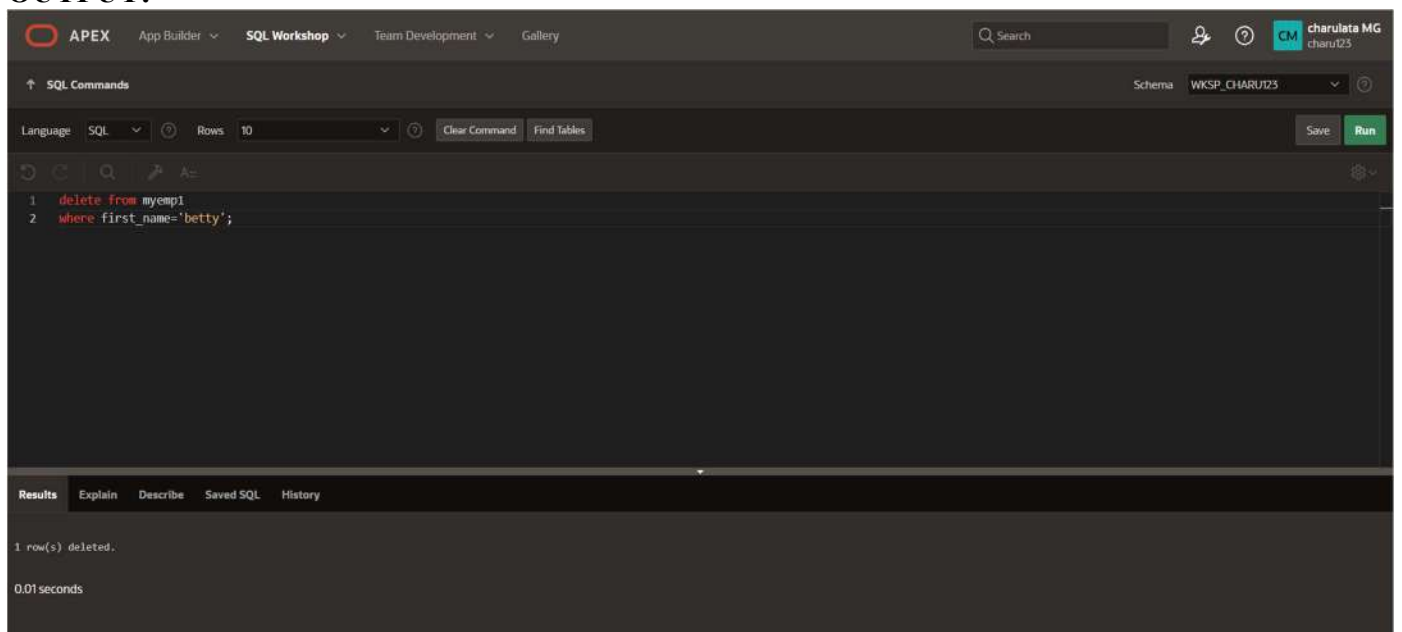


8.Delete Betty dancs from MY _EMPLOYEE table.

QUERY:

delete from myemp1 where first_name='betty';

OUTPUT:

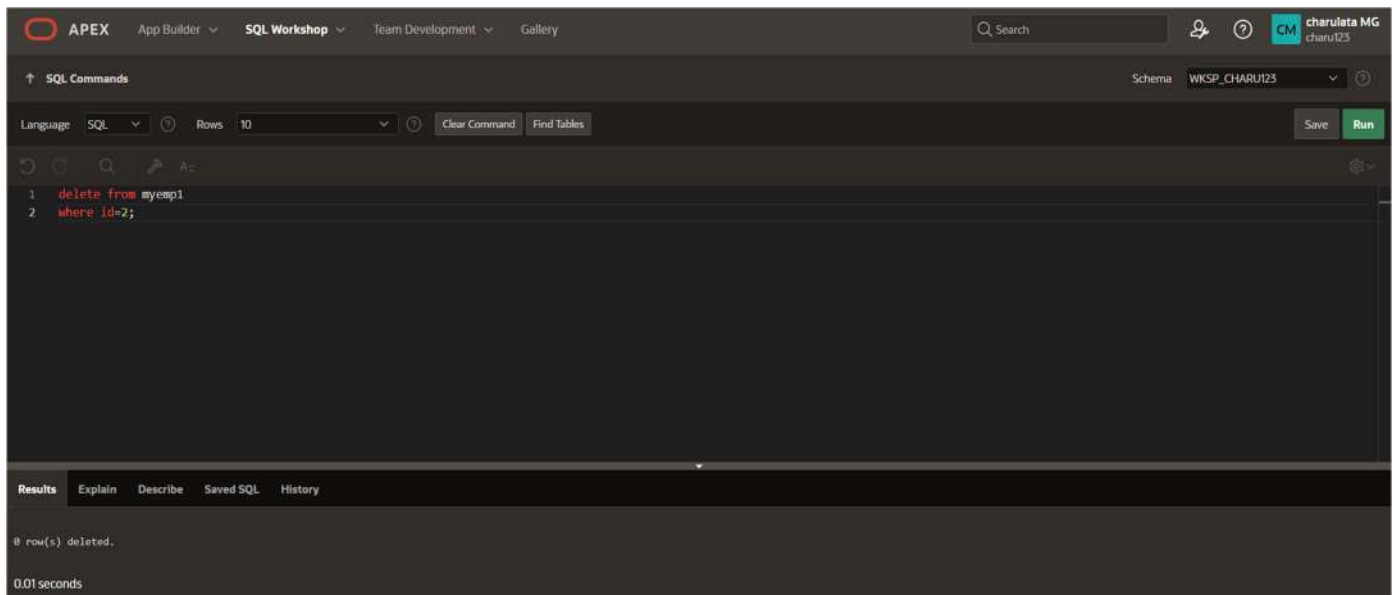


9.Empty the fourth row of the emp table.

QUERY:

delete from myemp1 where id=2;

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'charulata MG' are on the right. The 'SQL Commands' panel is active, showing a query: `1 delete from myemp1` and `2 where id=2;`. The 'Run' button is highlighted. Below the query editor, the 'Results' tab is selected, displaying the message: `0 row(s) deleted.` and the execution time: `0.01 seconds`.

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

RESULT:

INCLUDING CONSTRAINTS

EX_NO:3

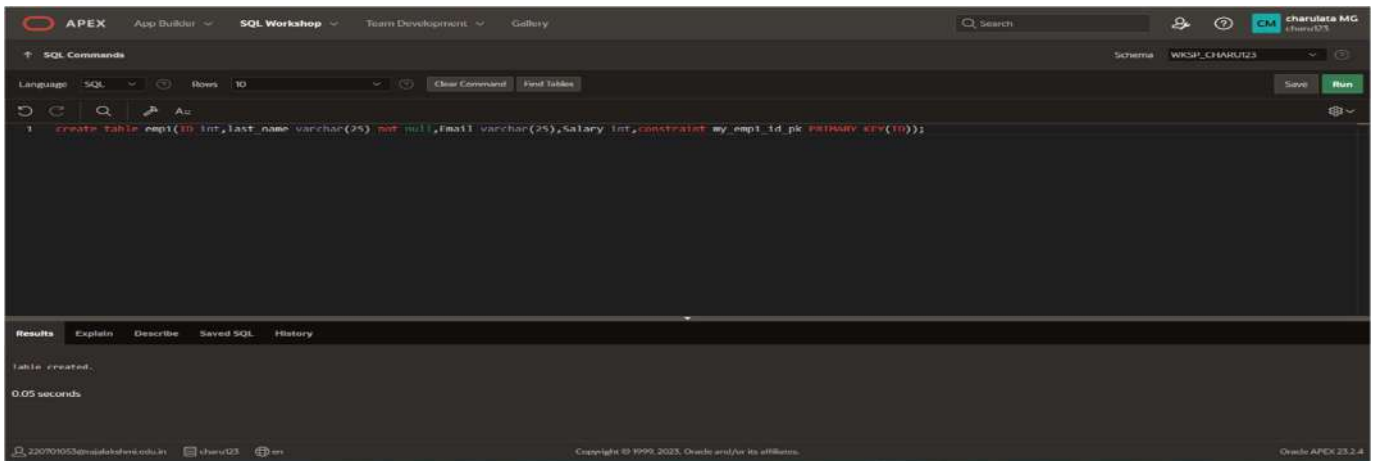
DATE:

1.Add a table-level PRIMARY KEY constraint to the EMP table on the ID column.The constraint should be named at creation. Name the constraint my_emp_id_pk.

QUERY:

Create table emp1(id int,last_name varchar(25) not null,email varchar(25),salary int,constraint my_emp_id_pk PRIMARY KEY(id));

OUTPUT:

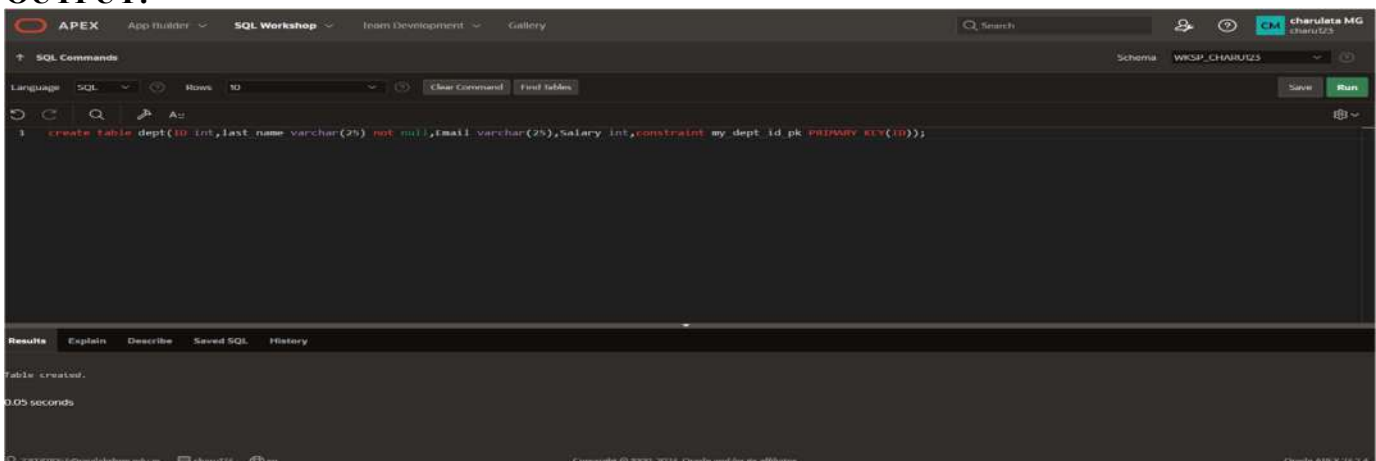


2.Create a PRIMAY KEY constraint to the DEPT table using the ID column. The constraint should be named at creation. Name the constraint my_dept_id_pk.

QUERY:

Create table dept(id int,last_name varchar(25) not null, email varchar(25),salary int, constraint my_dept_id_pk PRIMARY KEY(id));

OUTPUT:

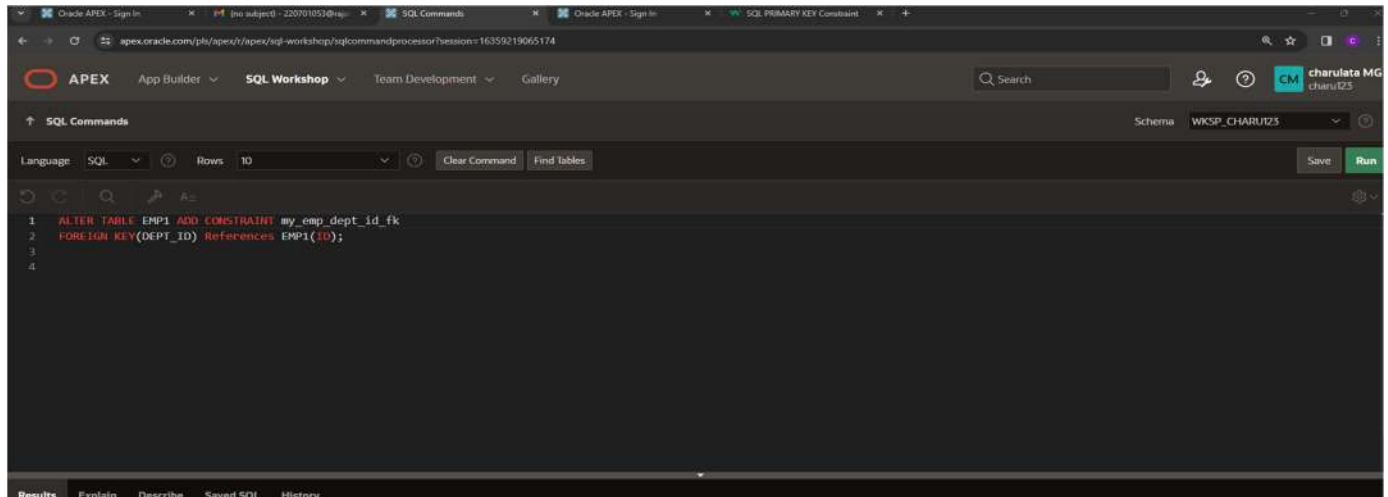


3. Add a column DEPT_ID to the EMP table. Add a foreign key reference on the EMP table that ensures that the employee is not assigned to nonexistent department. Name the constraint my_emp_dept_id_fk.

QUERY:

```
ALTER TABLE emp1 add constraint my_emp_dept_id_fk  
FOREIGN KEY(DEPT_ID) References emp1(id);
```

OUTPUT:

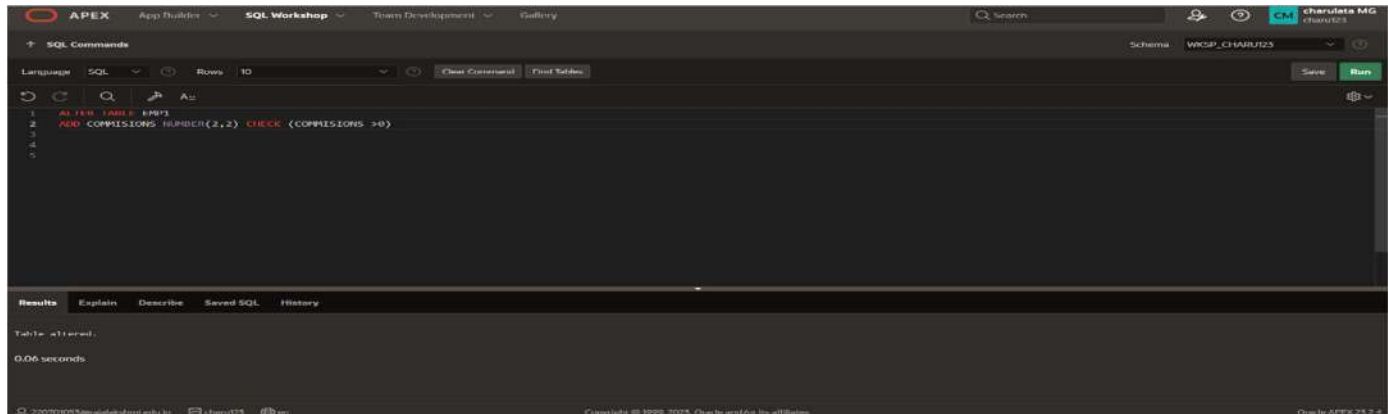


4. Modify the EMP table. Add a COMMISSION column of NUMBER data type, precision 2, scale 2. Add a constraint to the commission column that ensures that a commission value is greater than zero.

QUERY:

```
ALTER TABLE emp1  
ADD commissions number(2,2) check (commissions>0)
```

OUTPUT:



Evaluation Procedure Marks

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

RESULT:

Writing Basic SQL SELECT Statements

EX_NO:4

DATE:

1.The following statement executes successfully.

Identify the Errors

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

QUERY:

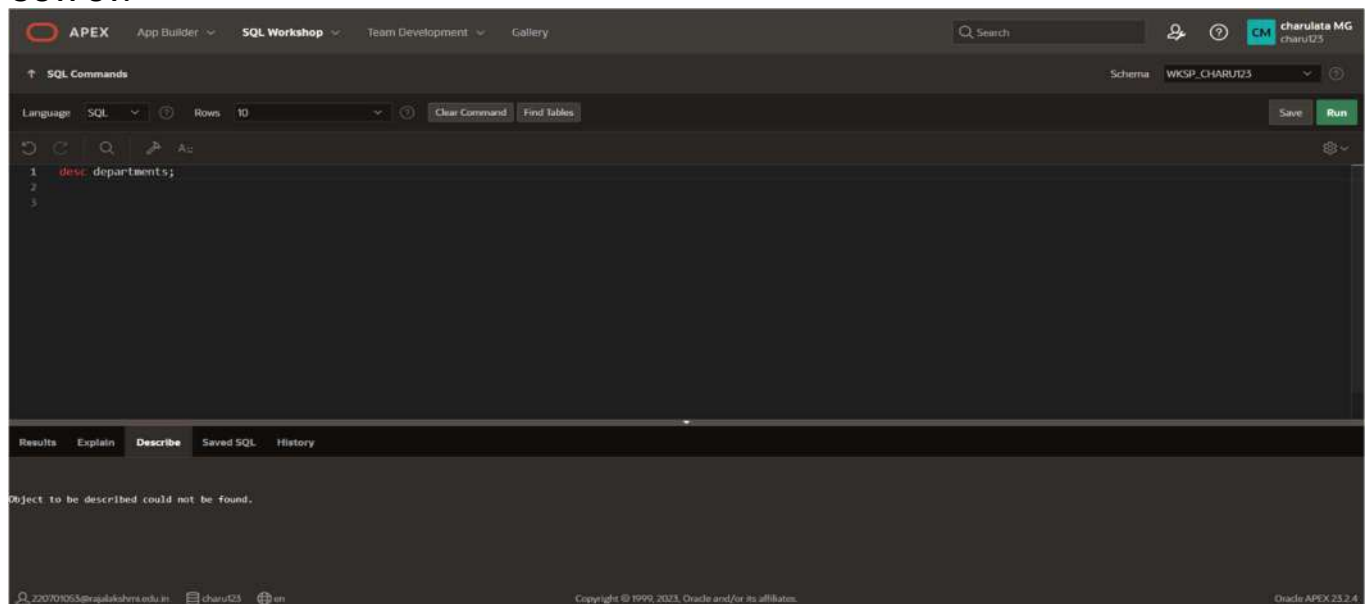
```
SELECT employee_id, last_name  
salary*12 ANNUAL SALARY  
FROM employees;
```

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

QUERY:

```
desc departments;
```

OUTPUT:

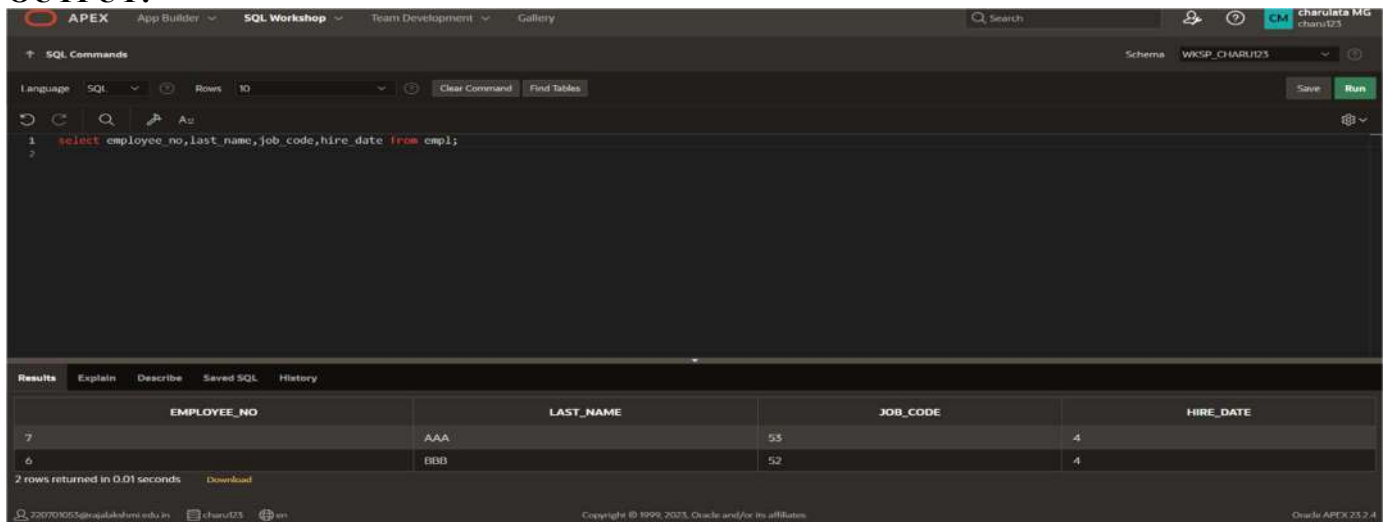


3.Create a query to display the last name, job code, hire date, and employee number for each employee, with employee number appearing first.

QUERY:

Select employee_no,last_name,job_code,hire_date from emp1;

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is `select employee_no,last_name,job_code,hire_date from emp1;`. The results are displayed in a table with 2 rows.

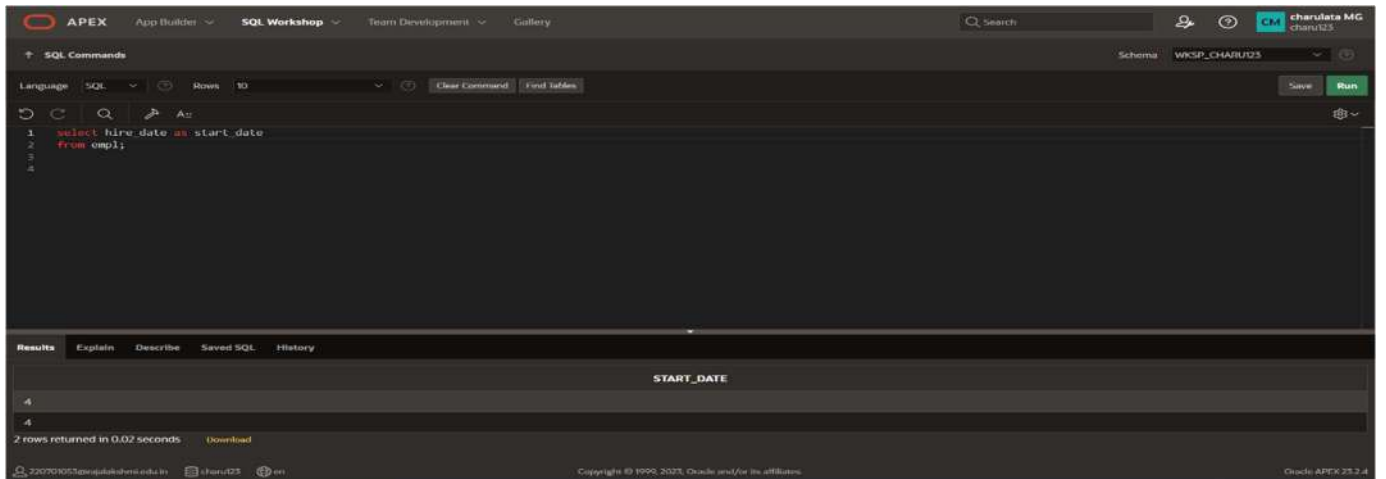
EMPLOYEE_NO	LAST_NAME	JOB_CODE	HIRE_DATE
7	AAA	55	4
6	BBB	52	4

2 rows returned in 0.01 seconds

4.Provide an alias STARTDATE for the hire date.

QUERY:

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is `select hire_date as start_date from emp1;`. The results are displayed in a table with 2 rows.

START_DATE
4
4

2 rows returned in 0.02 seconds

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

QUERY:

```
Select distinct job_code  
From emp1
```

OUTPUT:

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

```
1 select distinct job_code  
2 from emp1  
3  
4
```

The Results pane shows the output of the query:

JOB_CODE
52
53

2 rows returned in 0.00 seconds. Download

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

QUERY:

```
Select last_name||', '||job_code as "EMPLOYEE AND TITLE" from emp1;
```

OUTPUT:

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

```
1 select last_name||', '||job_code as "EMPLOYEE AND TITLE" from emp1;  
2  
3
```

The Results pane shows the output of the query:

EMPLOYEE AND TITLE
AAA, 53
BBB, 52

2 rows returned in 0.01 seconds. Download

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

QUERY:

Select last_name||','||job_code||','||hire_date||','||employee_no as THE_OUTPUT from emp1;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'charulata MG charu123' are on the right. The 'SQL Commands' panel shows a query: `select last_name||','||job_code||','||hire_date||','||employee_no as THE_OUTPUT from emp1;`. The 'Results' panel shows the output of the query, which is a table with one column 'THE_OUTPUT' and two rows of data: 'AAA,53,4,7' and 'BBB,52,4,6'. The status bar at the bottom indicates '2 rows returned in 0.01 seconds' and provides a 'Download' link.

Evaluation Procedure Marks

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

RESULT:

RESTRICTING AND SORTING DATA

EX_NO:5

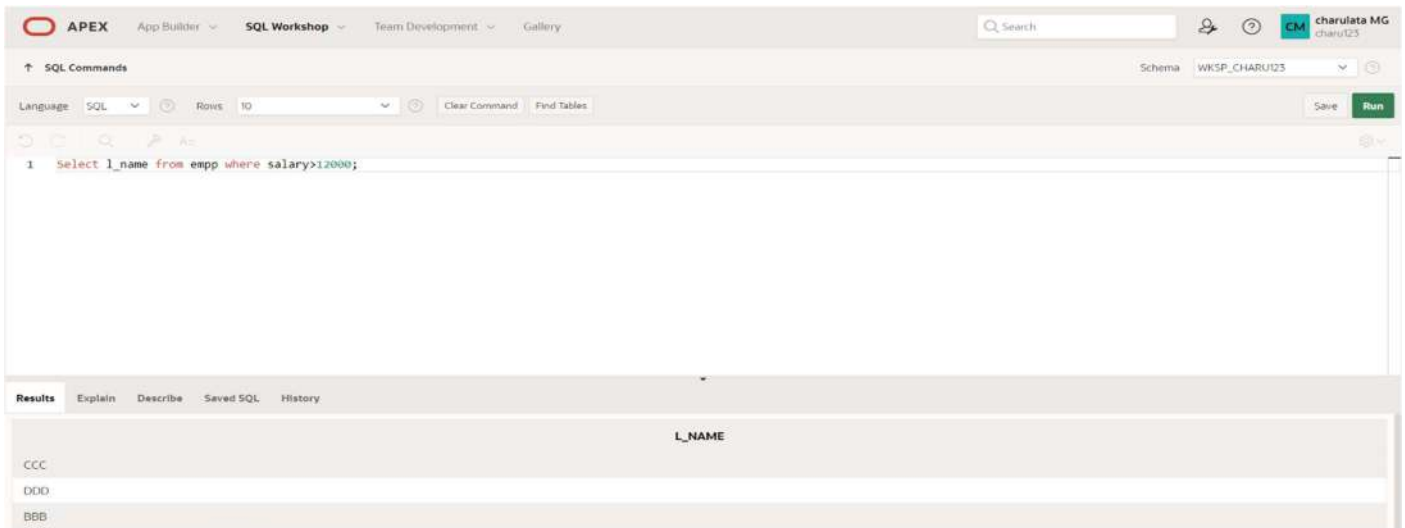
DATE: 07-03-2024

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

QUERY:

Select l_name from empp where salary>12000;

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The query entered is `Select l_name from empp where salary>12000;`. The results are displayed in a table with the column `L_NAME`. The results are:

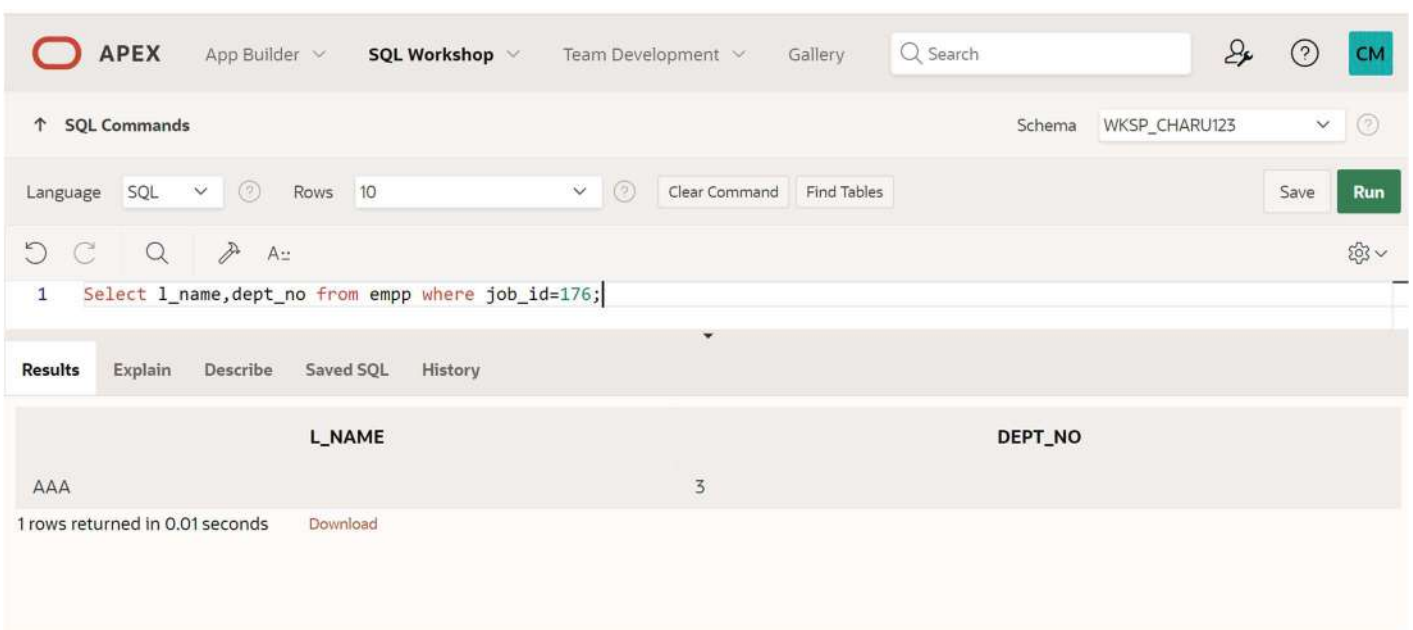
L_NAME
CCC
DDD
BBB

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

QUERY:

Select l_name,dept_id from employees where job_id=176;

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The query entered is `Select l_name,dept_no from empp where job_id=176;`. The results are displayed in a table with the columns `L_NAME` and `DEPT_NO`. The results are:

L_NAME	DEPT_NO
AAA	3

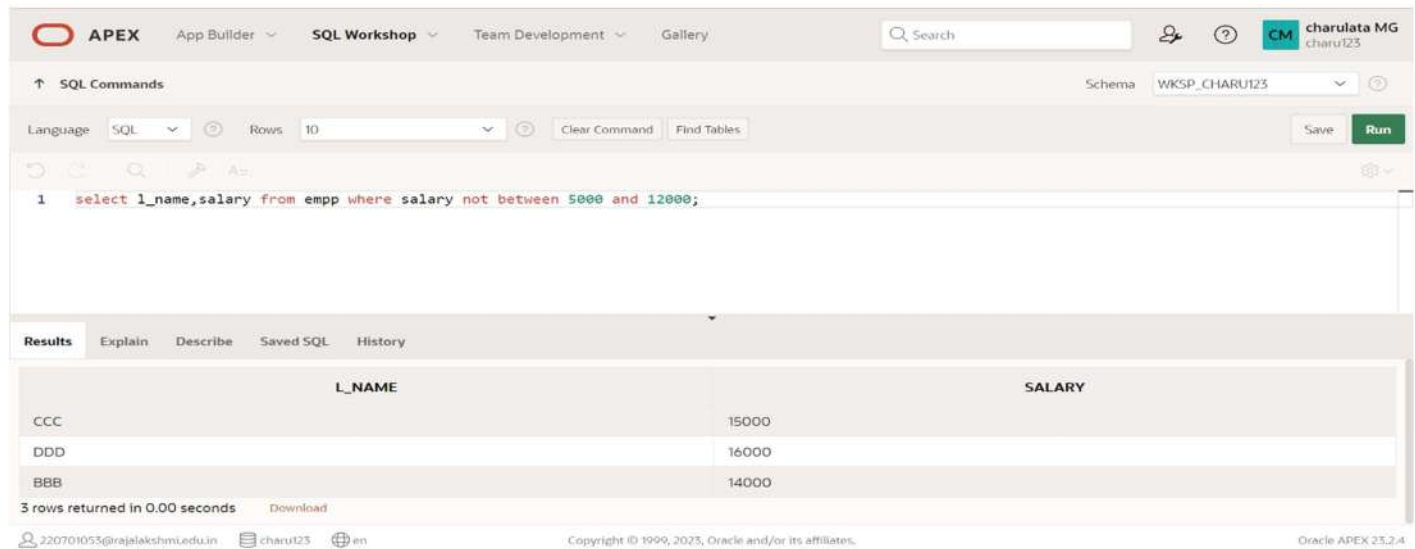
1 rows returned in 0.01 seconds

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 l_name,salary from empp where salary not between 5000 and 12000;

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is: `select l_name,salary from empp where salary not between 5000 and 12000;`. The results are displayed in a table with two columns: L_NAME and SALARY. The results are:

L_NAME	SALARY
CCC	15000
DDD	16000
BBB	14000

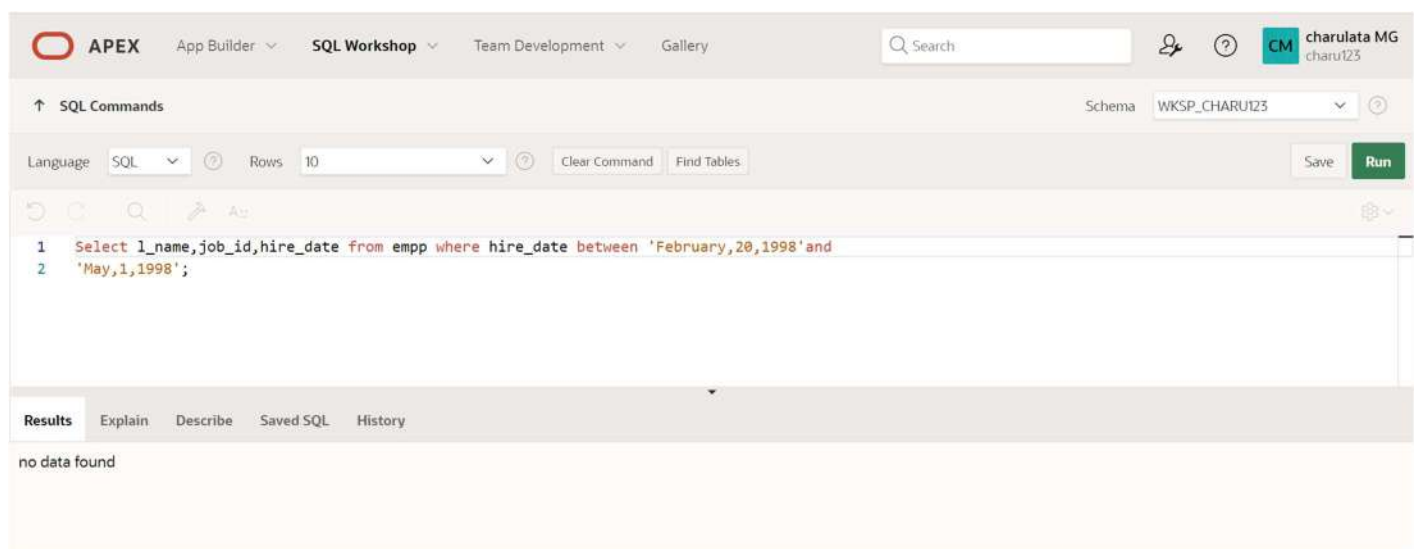
3 rows returned in 0.00 seconds

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

QUERY:

Select l_name,job_id,hire_date from empp where hire_date between 'February,20,1998'and 'May,1,1998'; 'May,1,1998';

OUTPUT:

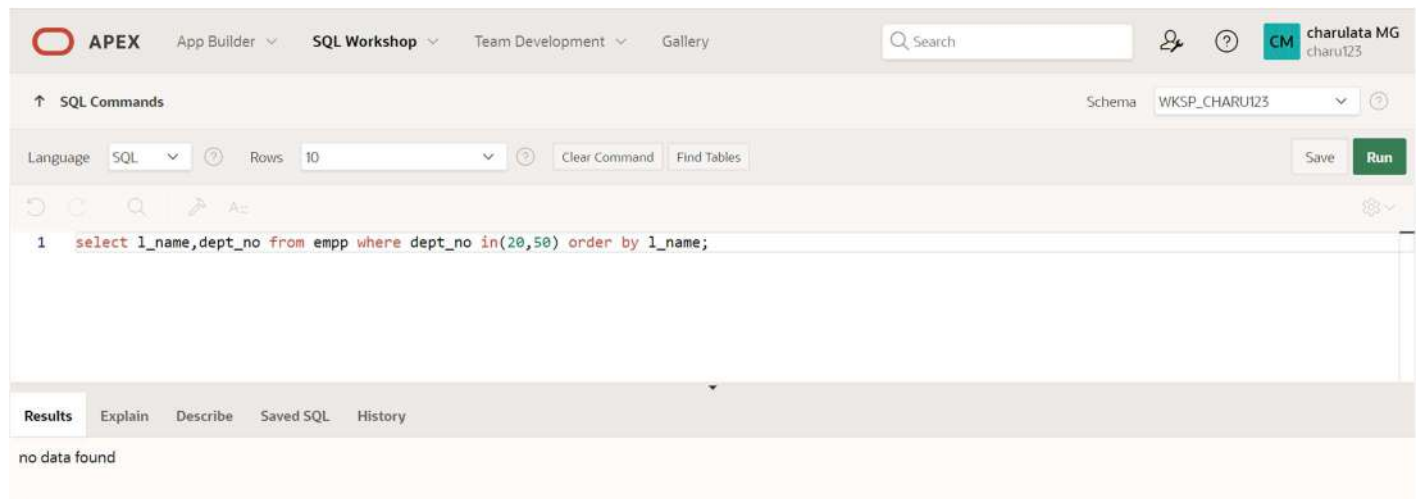


The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is: `Select l_name,job_id,hire_date from empp where hire_date between 'February,20,1998'and 'May,1,1998'; 'May,1,1998';`. The results are displayed as "no data found".

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

QUERY: select l_name,dept_no from empp where dept_no in(20,50) order by l_name;

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'charulata MG' are on the right. The 'SQL Commands' section shows the schema 'WKSP_CHARU123'. The query editor contains the SQL statement: `1 select l_name,dept_no from empp where dept_no in(20,50) order by l_name;`. The 'Results' tab is selected, displaying 'no data found'.

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

QUERY:

select l_name as "EMPLOYEE",salary as "MONTHLY SALARY" from empp where (salary between 5000 and 12000) and (dept_no in(20,50)) order by l_name asc;

OUTPUT:

APEX App Builder SQL Workshop Team Development Gallery

Search

Schema WKSP_CHARU123

Language SQL Rows 10 Clear Command Find Tables Save Run

```

1 select l_name as "EMPLOYEE", salary as "MONTHLY SALARY" from emp where (salary between 5000
2 and 12000) and (dept_no in(20,50)) order by l_name asc;

```

Results Explain Describe Saved SQL History

EMPLOYEE	MONTHLY SALARY
AAA	11000

1 rows returned in 0.01 seconds Download

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

QUERY:

select l_name,hire_date from emp where hire_date like '1994';

OUTPUT:

APEX App Builder SQL Workshop Team Development Gallery

Search

Schema WKSP_CHARU123

Language SQL Rows 10 Clear Command Find Tables Save Run

```

1 select l_name,hire_date from emp where hire_date like '1994';

```

Results Explain Describe Saved SQL History

no data found

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

QUERY:

select l_name,job_id from emp where dept_no is null;

OUTPUT:

The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'charulata MG charu123' are on the right. The 'SQL Commands' section shows a query: `select l_name, job_id from emp where dept_no is null;`. The 'Results' tab is selected, displaying 'no data found'.

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

QUERY:

```
select l_name, salary, commision_pet from empp where commision_pet is not null order by salary desc
```

OUTPUT:

The screenshot shows the APEX SQL Workshop interface with a query: `select l_name, salary, commision_pet from empp where commision_pet is not null order by salary desc;`. The 'Results' tab displays a table with the following data:

L_NAME	SALARY	COMMISION_PET
DDD	16000	53
CCC	15000	50
BBB	14000	55
AAA	11000	56

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

QUERY:

```
select l_name from empp where l_name like '__a%';
```

OUTPUT:

The screenshot shows the APEX SQL Workshop interface. At the top, there's a navigation bar with 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'charulata MG' are also present. Below the navigation bar, the 'SQL Commands' section is active, showing a schema dropdown set to 'WKSP_CHARU123'. The 'Language' is set to 'SQL' and 'Rows' is set to '10'. The query entered is: `1 select l_name from empp where l_name like ' _a%';`. The 'Run' button is highlighted. Below the query editor, the 'Results' tab is selected, displaying 'no data found'.

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

QUERY:

select l_name from empp where l_name like '%a%' and l_name like '%e%';

OUTPUT:

The screenshot shows the APEX SQL Workshop interface. At the top, there's a navigation bar with 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'charulata MG' are also present. Below the navigation bar, the 'SQL Commands' section is active, showing a schema dropdown set to 'WKSP_CHARU123'. The 'Language' is set to 'SQL' and 'Rows' is set to '10'. The query entered is: `1 select l_name from empp where l_name like '%a%' and l_name like '%e%';`. The 'Run' button is highlighted. Below the query editor, the 'Results' tab is selected, displaying 'no data found'.

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

QUERY:

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

OUTPUT:

The screenshot shows the APEX SQL Workshop interface. The SQL command entered is: `select last_name,job_id,salary from employees where job_id in ('sales representative','stock clerk') and salary not in(2500,3500,7000);`. The results tab shows "no data found".

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

QUERY:

```
select l_name,salary,commision_pet from empp where commision_pet=0.2
```

OUTPUT:

The screenshot shows the APEX SQL Workshop interface. The SQL command entered is: `select l_name,salary,commision_pet from empp where commision_pet=0.2`. The results tab shows a table with 1 row returned.

L_NAME	SALARY	COMMISION_PET
CCC	15000	.2

1 rows returned in 0.01 seconds

Evaluation Procedure Marks

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

RESULT:

SINGLE ROW FUNCTIONS

EX.NO.6

DATE:

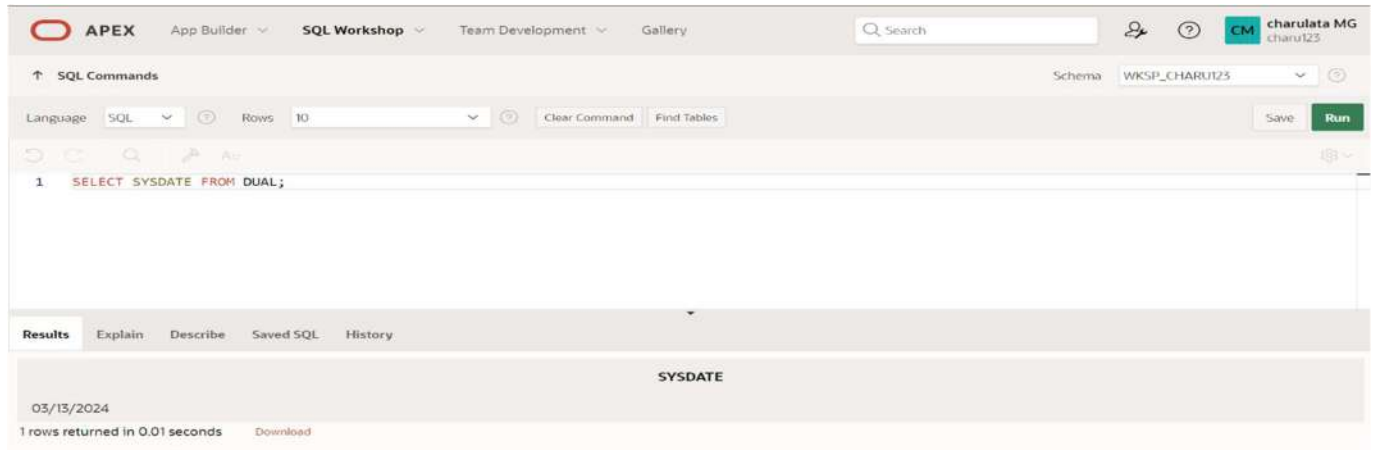
Find the Solution for the following:

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

QUERY:

SELECT SYSDATE AS "DATE" FROM DUAL;

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL Commands tab is active, displaying the query `SELECT SYSDATE FROM DUAL;`. The Results tab shows a single row with the value `03/13/2024` under the column header `SYSDATE`. The status bar indicates '1 rows returned in 0.01 seconds'.

SYSDATE
03/13/2024

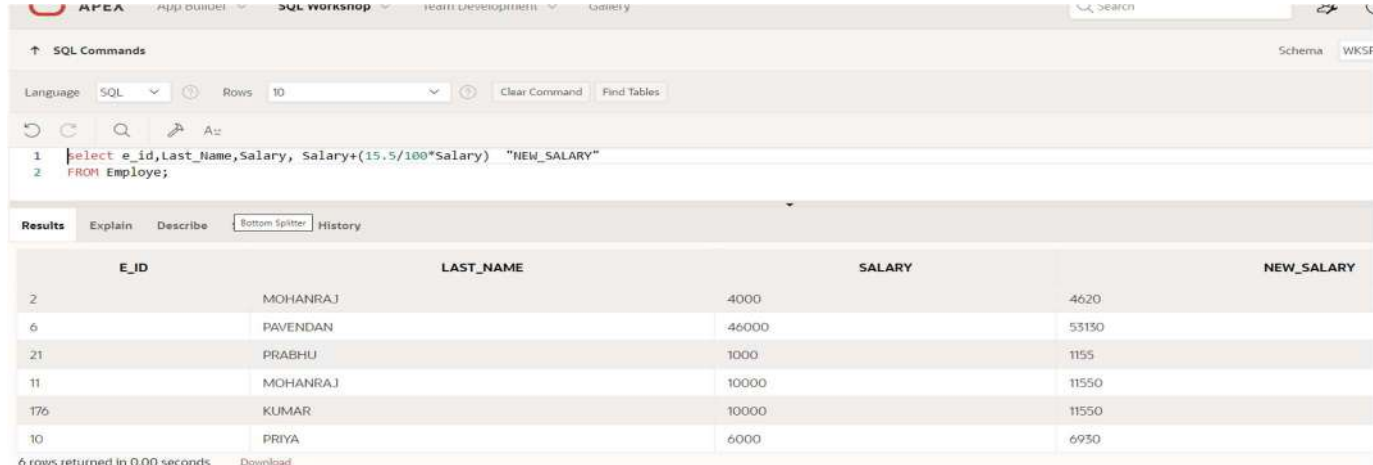
2. The HR department needs a report to display the employee number, last name, salary, and increased by 15.5% (expressed as a whole number) for each employee. Label the column New Salary.

QUERY:

SELECT e_id, last_name, Salary, Salary+(15.5/100*Salary) "NEW_SALARY"

From Employee;

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL Commands tab is active, displaying the query `SELECT e_id, last_name, Salary, Salary+(15.5/100*Salary) "NEW_SALARY" FROM Employee;`. The Results tab shows a table with 6 rows. The status bar indicates '6 rows returned in 0.00 seconds'.

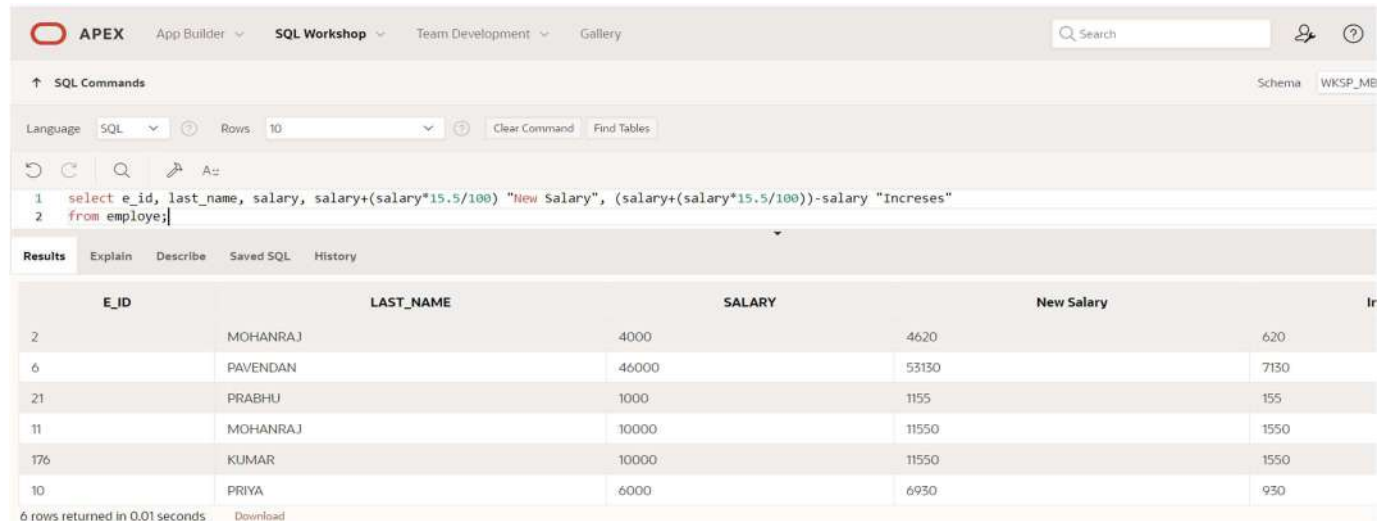
E_ID	LAST_NAME	SALARY	NEW_SALARY
2	MOHANRAJ	4000	4620
6	PAVENDAN	46000	53130
21	PRABHU	1000	1155
11	MOHANRAJ	10000	11550
176	KUMAR	10000	11550
10	PRIYA	6000	6930

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

QUERY:

```
SELECT e_id, last_name, Salary, (Salary+(Salary*15.5/100))-Salary "Increase"
From Employee;
```

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command entered is: `select e_id, last_name, salary, salary+(salary*15.5/100) "New Salary", (salary+(salary*15.5/100))-salary "Increases" from employee;`. The results are displayed in a table with 6 rows and 5 columns: E_ID, LAST_NAME, SALARY, New Salary, and Increases.

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

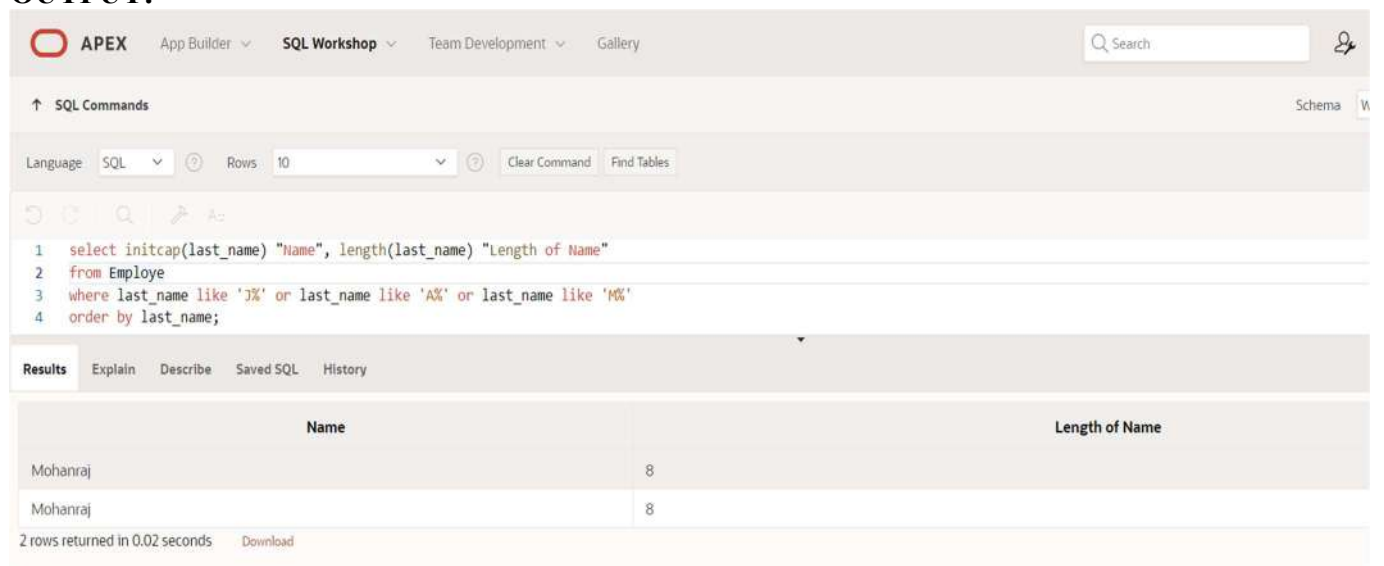
6 rows returned in 0.01 seconds

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

QUERY:

```
Select initcap(last_name) "Name", length(last_name) "Length of Name"
from Employee
where last_name like 'J%' or last_name like 'A%' or last_name like 'M%'
order by last_name;
```

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command entered is: `select initcap(last_name) "Name", length(last_name) "Length of Name" from Employee where last_name like 'J%' or last_name like 'A%' or last_name like 'M%' order by last_name;`. The results are displayed in a table with 2 rows and 2 columns: Name and Length of Name.

Name	Length of Name
Mohanraj	8
Mohanraj	8

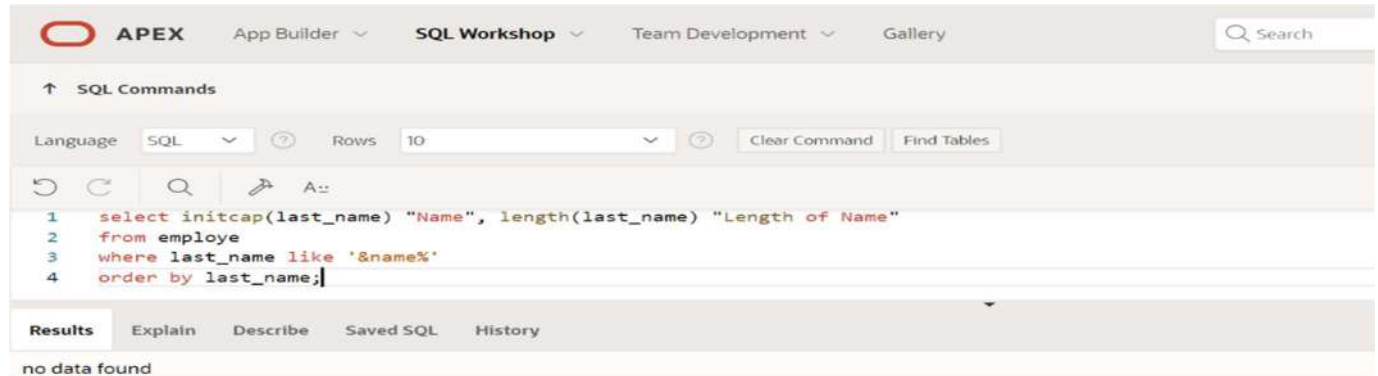
2 rows returned in 0.02 seconds

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

QUERY:

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

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes the APEX logo and links to App Builder, SQL Workshop, Team Development, and Gallery. A search bar is on the right. Below the navigation bar, the 'SQL Commands' section is active. It shows a language dropdown set to 'SQL', a rows limit of '10', and buttons for 'Clear Command' and 'Find Tables'. The SQL editor contains the following query:

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

Below the editor, the 'Results' tab is selected, showing 'no data found'.



The screenshot shows the APEX SQL Workshop interface with the same query as above, but the 'where' clause is updated to 'where last_name like 'H%''. The 'Results' tab is selected, showing a table with one row of data.

Name	Length of Name
Harini	6

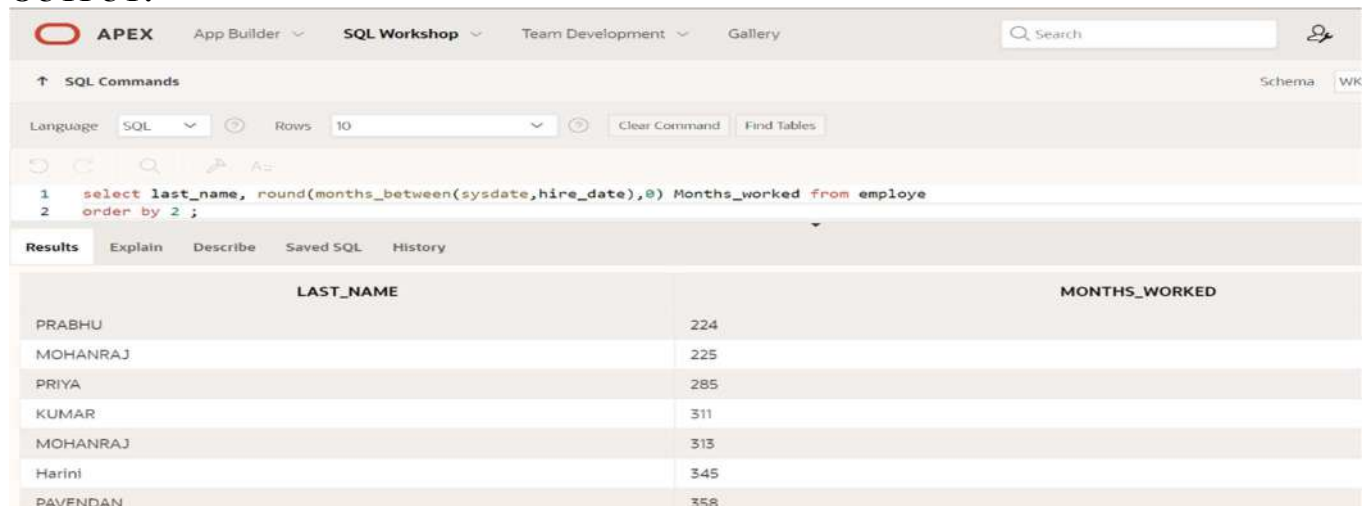
1 rows returned in 0.01 seconds [Download](#)

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

QUERY:

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

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The query is entered in the SQL Commands pane, and the Results pane displays the output of the query. The results are as follows:

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

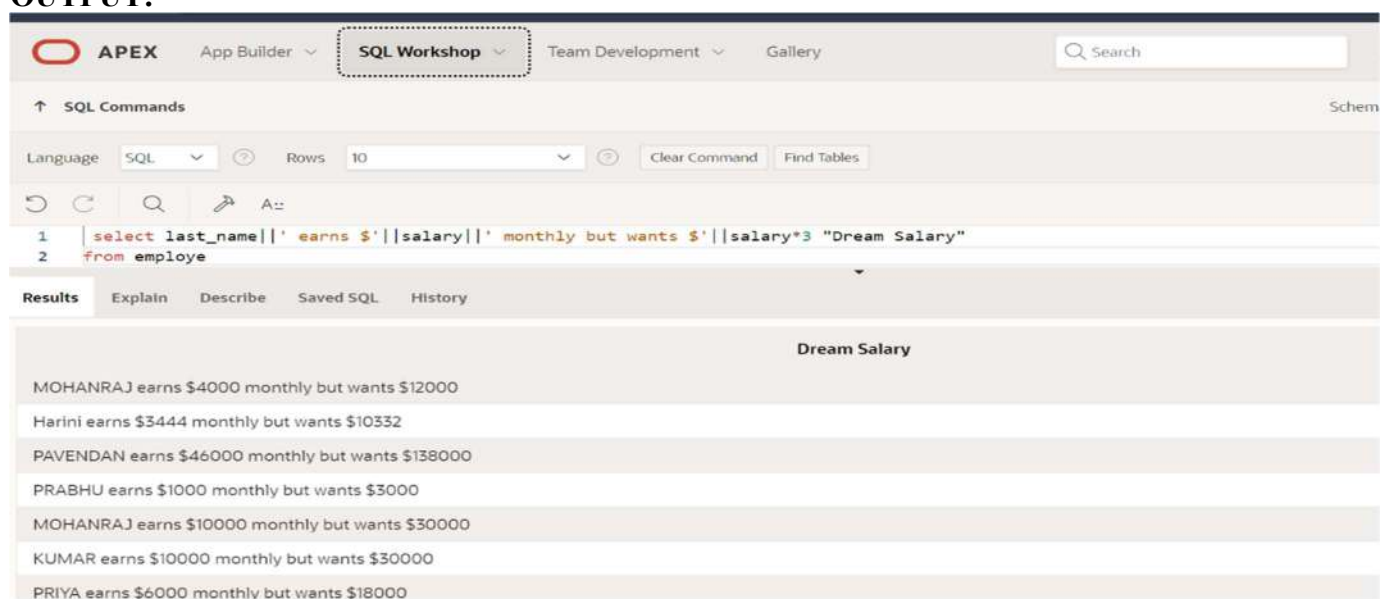
7.Create a report that produces the following for each employee:

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

QUERY:

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

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The query is entered in the SQL Commands pane, and the Results pane displays the output of the query. The results are as follows:

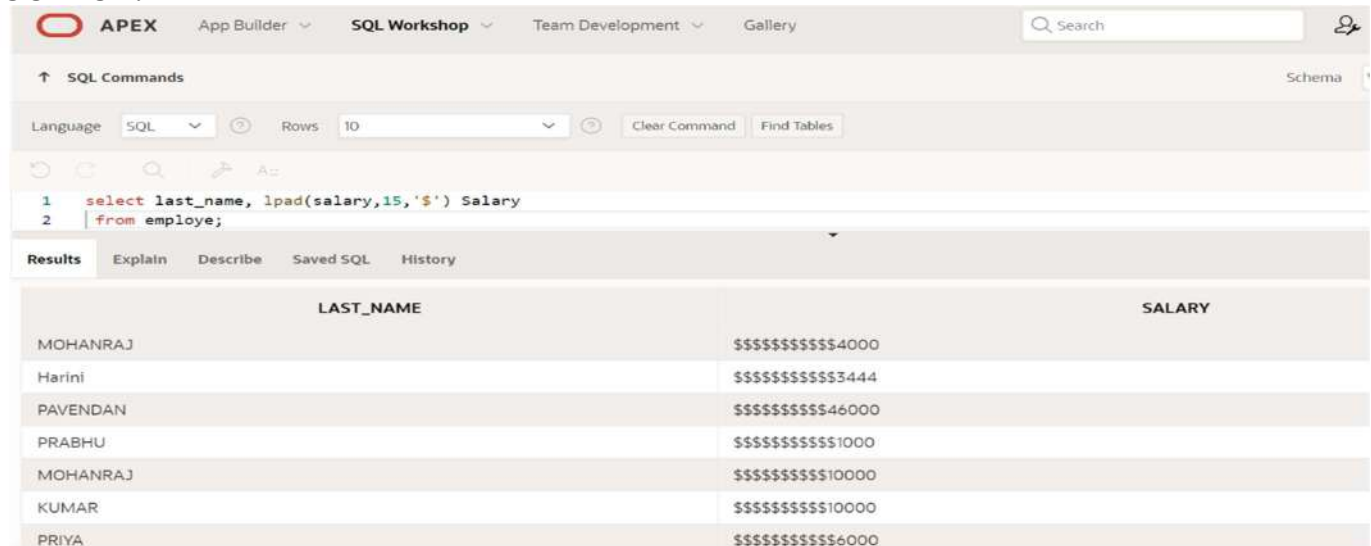
Dream Salary
MOHANRAJ earns \$4000 monthly but wants \$12000
Harini earns \$3444 monthly but wants \$10332
PAVENDAN earns \$46000 monthly but wants \$138000
PRABHU earns \$1000 monthly but wants \$3000
MOHANRAJ earns \$10000 monthly but wants \$30000
KUMAR earns \$10000 monthly but wants \$30000
PRIYA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

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

QUERY:

Select last_name, lpad(salary,15,'\$') Salary
from employee;

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command entered is: `1 select last_name, lpad(salary,15,'$') Salary` and `2 from employee;`. The results are displayed in a table with two columns: LAST_NAME and SALARY. The results are as follows:

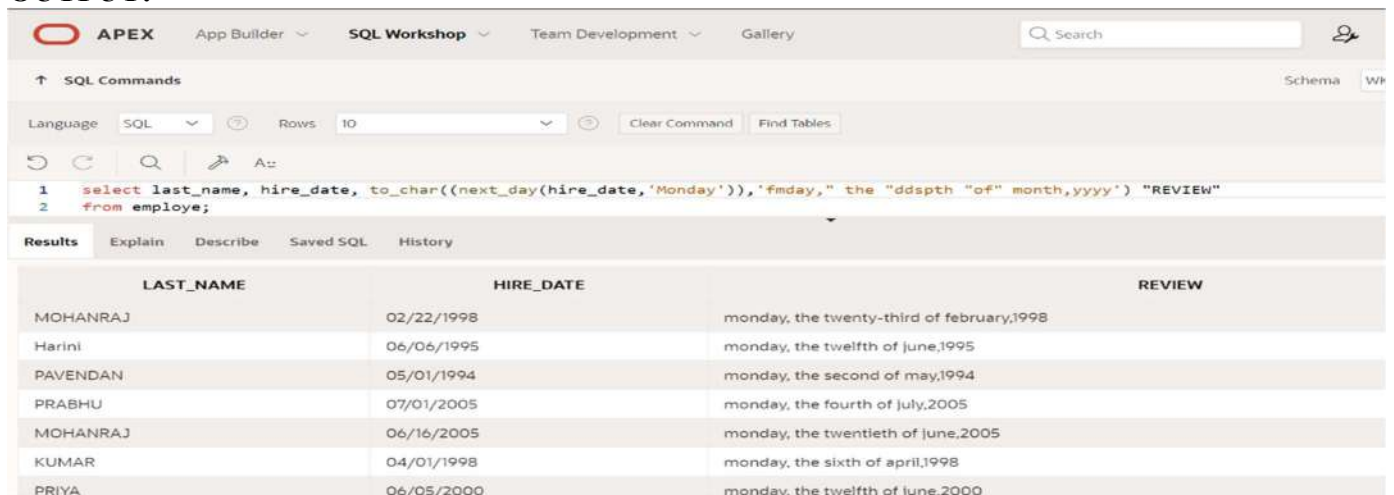
LAST_NAME	SALARY
MOHANRAJ	\$\$\$\$\$\$\$\$\$\$\$4000
Harini	\$\$\$\$\$\$\$\$\$\$\$3444
PAVENDAN	\$\$\$\$\$\$\$\$\$\$\$46000
PRABHU	\$\$\$\$\$\$\$\$\$\$\$1000
MOHANRAJ	\$\$\$\$\$\$\$\$\$\$\$10000
KUMAR	\$\$\$\$\$\$\$\$\$\$\$10000
PRIYA	\$\$\$\$\$\$\$\$\$\$\$6000

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

QUERY:

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

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command entered is: `1 select last_name, hire_date, to_char((next_day(hire_date,'Monday')),'fmday," the "ddspth "of" month,yyyy') "REVIEW"` and `2 from employee;`. The results are displayed in a table with three columns: LAST_NAME, HIRE_DATE, and REVIEW. The results are as follows:

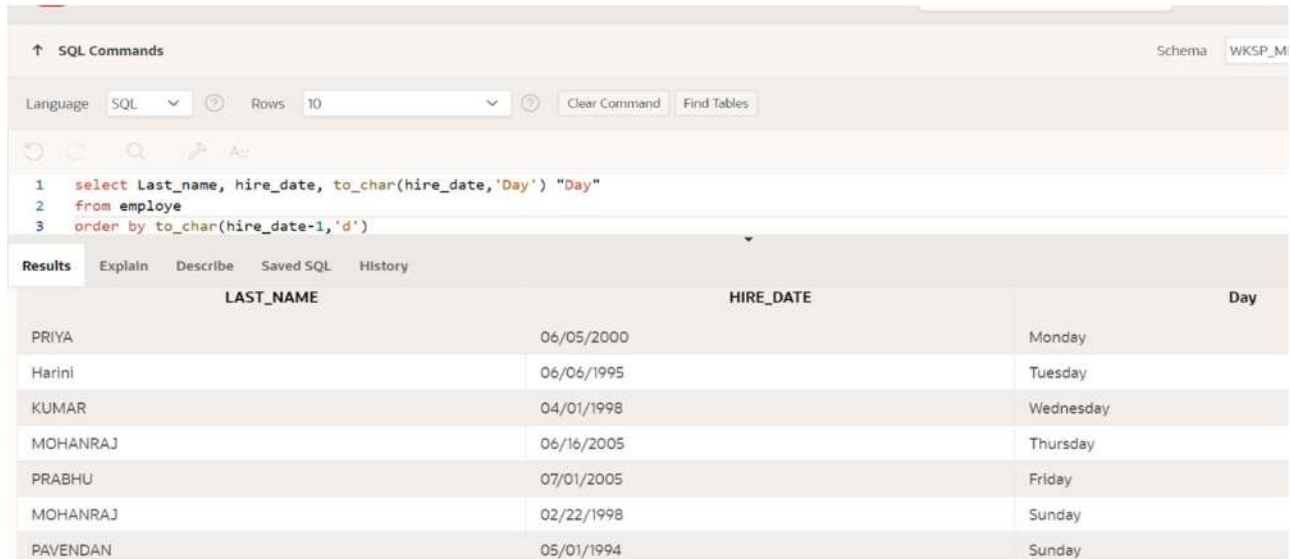
LAST_NAME	HIRE_DATE	REVIEW
MOHANRAJ	02/22/1998	monday, the twenty-third of february,1998
Harini	06/06/1995	monday, the twelfth of june,1995
PAVENDAN	05/01/1994	monday, the second of may,1994
PRABHU	07/01/2005	monday, the fourth of july,2005
MOHANRAJ	06/16/2005	monday, the twentieth of june,2005
KUMAR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

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

QUERY:

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

OUTPUT:



The screenshot shows the SQL Developer interface. The 'SQL Commands' tab is active, displaying the following query:

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

The 'Results' tab is selected, showing the output of the query. The results are as follows:

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

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

RESULT:

DISPLAYING DATA FROM MULTIPLE TABLES

EX_NO:7

DATE:

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

QUERY:

Select e.l_name,e.dept_id,d.dept_id from emp1 e,dept d where e.dept_id=d.dept_id;

OUTPUT:



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

```
1. Select e.l_name,e.dept_id,d.dept_id from emp1 e,dept d where e.dept_id=d.dept_id;
```

The results pane displays the following data:

L_NAME	DEPT_ID	DEPT_ID
davies	80	80
dans	80	80

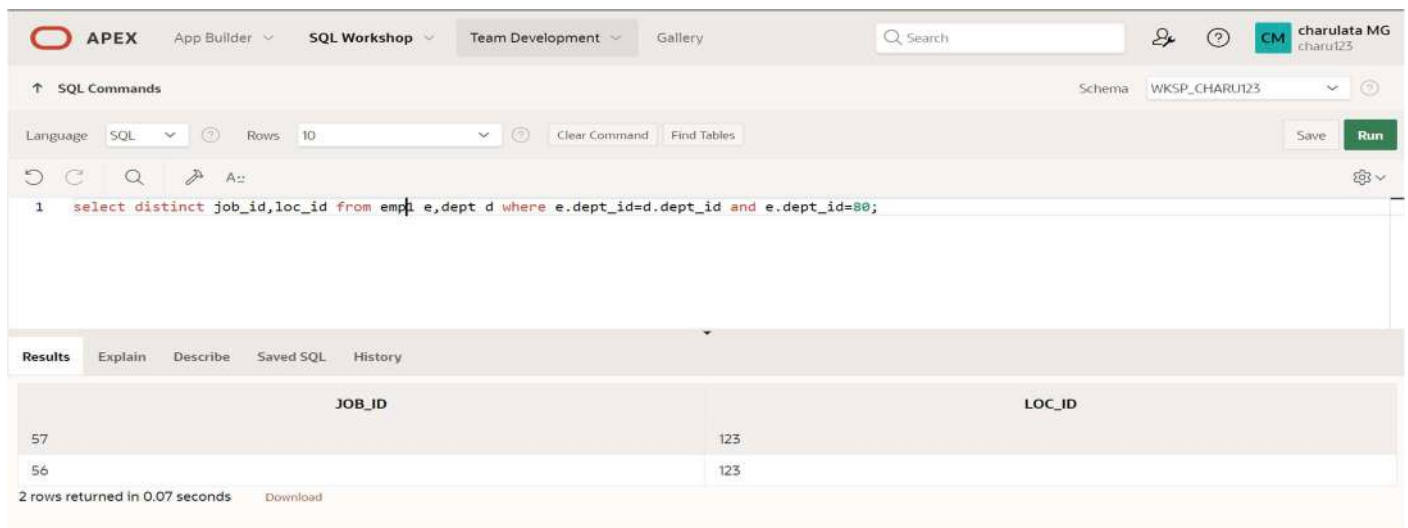
2 rows returned in 0.02 seconds

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 emp1 e,dept d where e.dept_id=d.dept_id and e.dept_id=80;

OUTPUT:



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

```
1. select distinct job_id,loc_id from emp1 e,dept d where e.dept_id=d.dept_id and e.dept_id=80;
```

The results pane displays the following data:

JOB_ID	LOC_ID
57	123
56	123

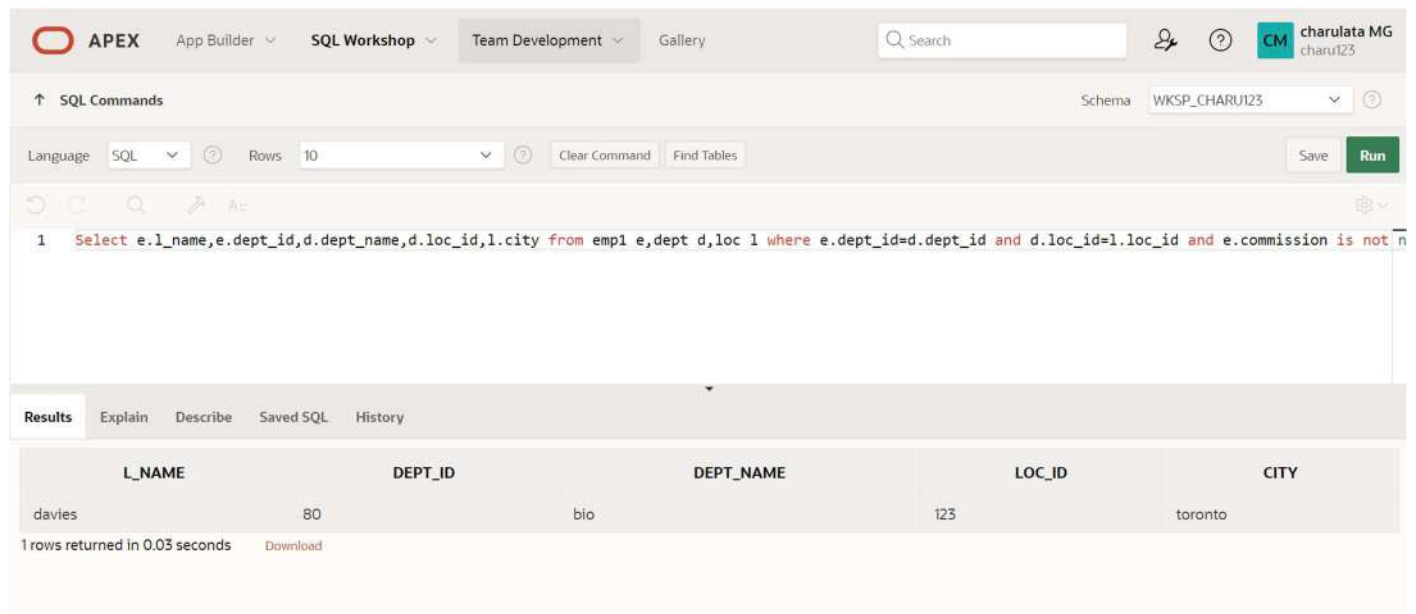
2 rows returned in 0.07 seconds

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

QUERY:

Select e.l_name,e.dept_id,d.dept_name,d.loc_id,l.city from emp1 e,dept d,loc l where e.dept_id=d.dept_id and d.loc_id=l.loc_id and e.commission is not null;

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command entered is: `Select e.l_name,e.dept_id,d.dept_name,d.loc_id,l.city from emp1 e,dept d,loc l where e.dept_id=d.dept_id and d.loc_id=l.loc_id and e.commission is not null;`. The results tab is active, displaying a table with 5 columns: L_NAME, DEPT_ID, DEPT_NAME, LOC_ID, and CITY. One row is returned: davies, 80, bio, 123, toronto.

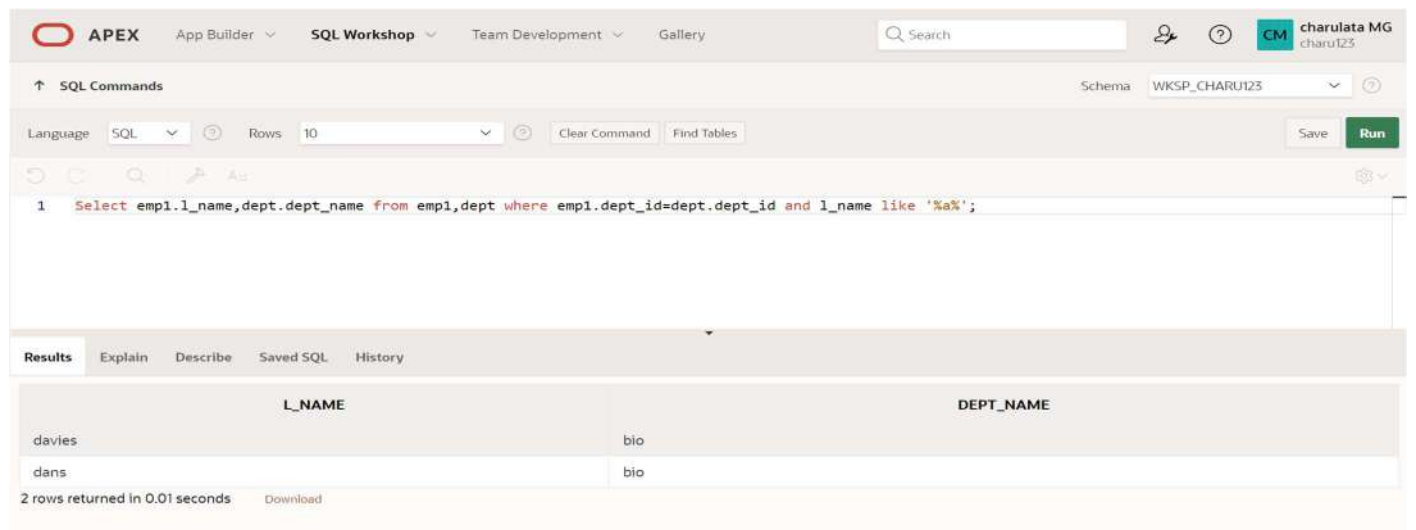
L_NAME	DEPT_ID	DEPT_NAME	LOC_ID	CITY
davies	80	bio	123	toronto

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

QUERY:

Select emp1.l_name,dept.dept_name from emp1,dept where emp1.dept_id=dept.dept_id and l_name like '%a%';

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command entered is: `Select emp1.l_name,dept.dept_name from emp1,dept where emp1.dept_id=dept.dept_id and l_name like '%a%';`. The results tab is active, displaying a table with 2 columns: L_NAME and DEPT_NAME. Two rows are returned: davies, bio and dans, bio.

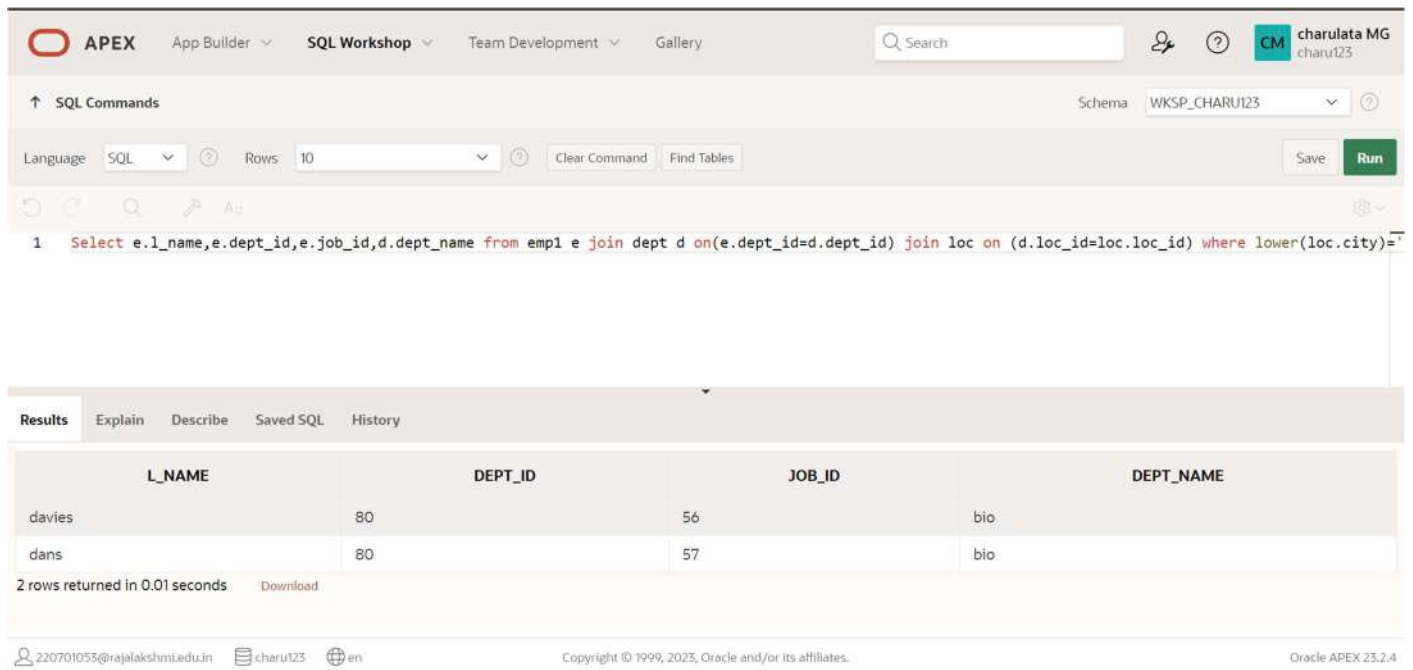
L_NAME	DEPT_NAME
davies	bio
dans	bio

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

QUERY:

Select e.l_name,e.dept_id,e.job_id,d.dept_name from emp1 e join dept d on(e.dept_id=d.dept_id) join loc on (d.loc_id=loc.loc_id) where lower(loc.city)='toronto';

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The query entered is: `1 Select e.l_name,e.dept_id,e.job_id,d.dept_name from emp1 e join dept d on(e.dept_id=d.dept_id) join loc on (d.loc_id=loc.loc_id) where lower(loc.city)='toronto';` The results tab shows two rows of data.

L_NAME	DEPT_ID	JOB_ID	DEPT_NAME
davies	80	56	bio
dans	80	57	bio

2 rows returned in 0.01 seconds

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

QUERY:

Select w.last_name “Employee”,w.emp_id “emp#”,m.last_name ‘manager”,m.emp_id “Mgr#” from emp21 m on (w.manager_id=m.emp_id);

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The query entered is: `1 Select w.last_name “Employee”,w.emp_id “emp#”,m.last_name ‘manager”,m.emp_id “Mgr#” from emp21 m on (w.manager_id=m.emp_id);` The results tab shows two rows of data.

Employee	emp#	manager	Mgr#
...
...

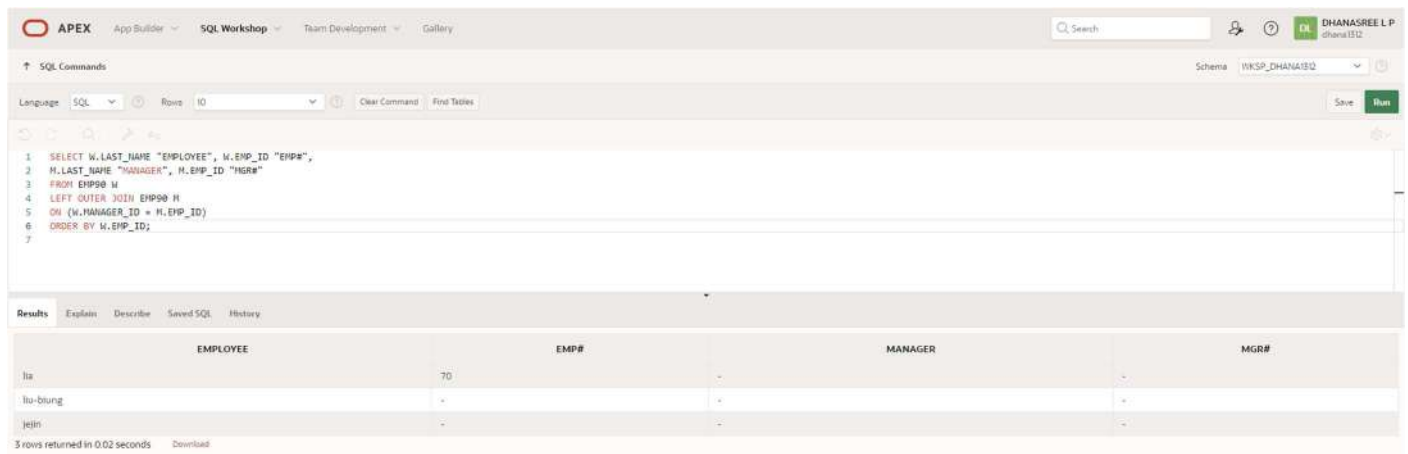
2 rows returned in 0.01 seconds

7. Modify lab4_6.sql to display all employees including King, who has no manager. Order the results by the employee number.

QUERY:

Select w.last_name “Employee”,w.emp_id “emp#”,m.last_name ‘manager’,m.emp_id “Mgr#” from emp021 w left outer join emp021 m on (w.manager_id=m.emp_id);

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command is as follows:

```
1 SELECT W.LAST_NAME "EMPLOYEE", W.EMP_ID "EMP#",
2       M.LAST_NAME "MANAGER", M.EMP_ID "MGR#"
3 FROM EMP021 W
4 LEFT OUTER JOIN EMP021 M
5 ON (W.MANAGER_ID = M.EMP_ID)
6 ORDER BY W.EMP_ID;
```

The results table displays the following data:

EMPLOYEE	EMP#	MANAGER	MGR#
Ita	70	-	-
Ilu-bung	-	-	-
jein	-	-	-

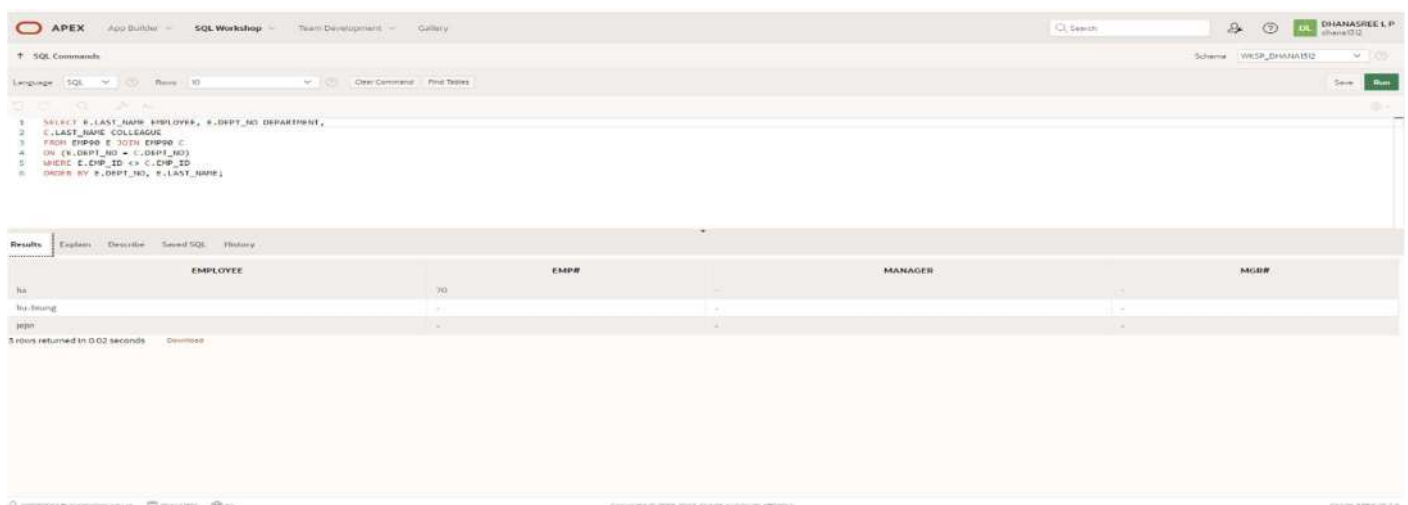
3 rows returned in 0.02 seconds

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

QUERY:

select e.department_number dept23,e.last_name colleague from emp021 e join emp021 c on (e.department_number=c.department_number) where e.emp_id <> c.emp_id order by e.department_number,e.last_name,c.last_name;

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command is as follows:

```
1 SELECT E.LAST_NAME EMPLOYEE, E.DEPT_NO DEPARTMENT,
2       C.LAST_NAME COLLEAGUE
3 FROM EMP021 E JOIN EMP021 C
4 ON (E.DEPT_NO = C.DEPT_NO)
5 WHERE E.EMP_ID <> C.EMP_ID
6 ORDER BY E.DEPT_NO, E.LAST_NAME;
```

The results table displays the following data:

EMPLOYEE	EMP#	MANAGER	MGR#
Ita	70	-	-
Ilu-bung	-	-	-
jein	-	-	-

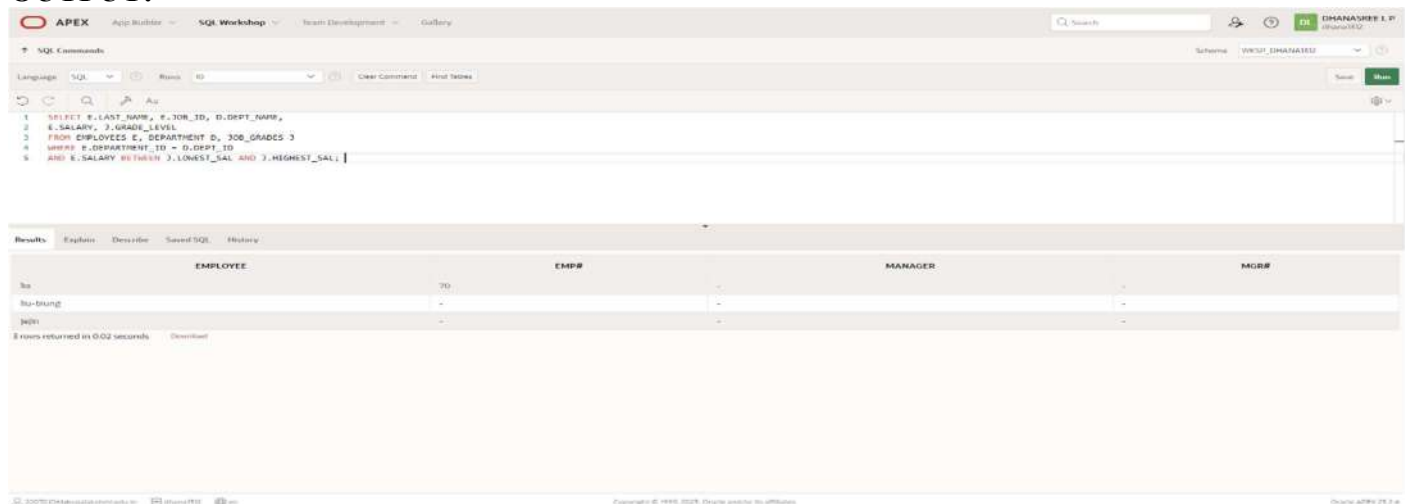
3 rows returned in 0.02 seconds

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

QUERY:

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

OUTPUT:



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

```
1 SELECT e.last_name, e.job_id, d.dept_name,
2 e.salary, j.grade_level
3 FROM EMPLOYEES e, DEPARTMENT d, JOB_GRADES j
4 WHERE e.department_id = d.dept_id
5 AND e.salary BETWEEN j.lowest_sal AND j.highest_sal;
```

The Results pane shows the output of the query. The table has 5 columns: LAST_NAME, JOB_ID, DEPT_NAME, SALARY, and GRADE_LEVEL. The results are as follows:

LAST_NAME	JOB_ID	DEPT_NAME	SALARY	GRADE_LEVEL
Isa	70			
Isa-brung				
Isa				

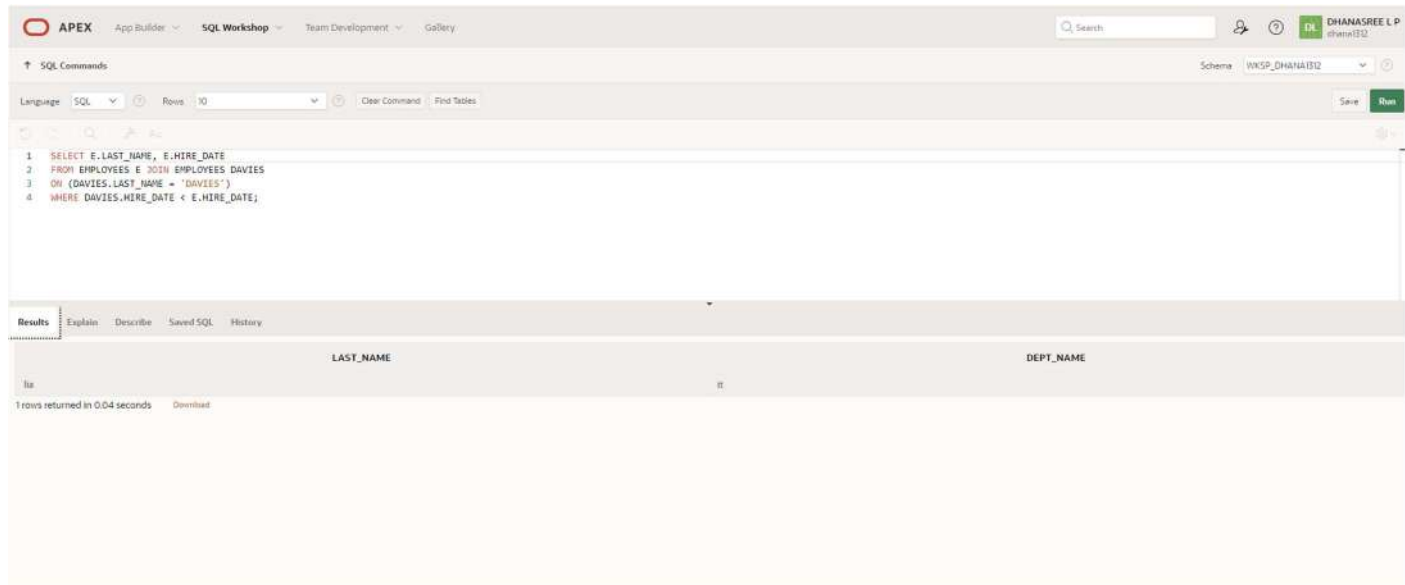
3 rows returned in 0.02 seconds

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

QUERY:

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

OUTPUT:



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

```
1 SELECT e.last_name, e.hire_date
2 FROM EMPLOYEES e JOIN EMPLOYEES davies
3 ON (davies.last_name = 'Davies')
4 WHERE davies.hire_date < e.hire_date;
```

The Results pane shows the output of the query. The table has 2 columns: LAST_NAME and HIRE_DATE. The results are as follows:

LAST_NAME	HIRE_DATE
Isa	

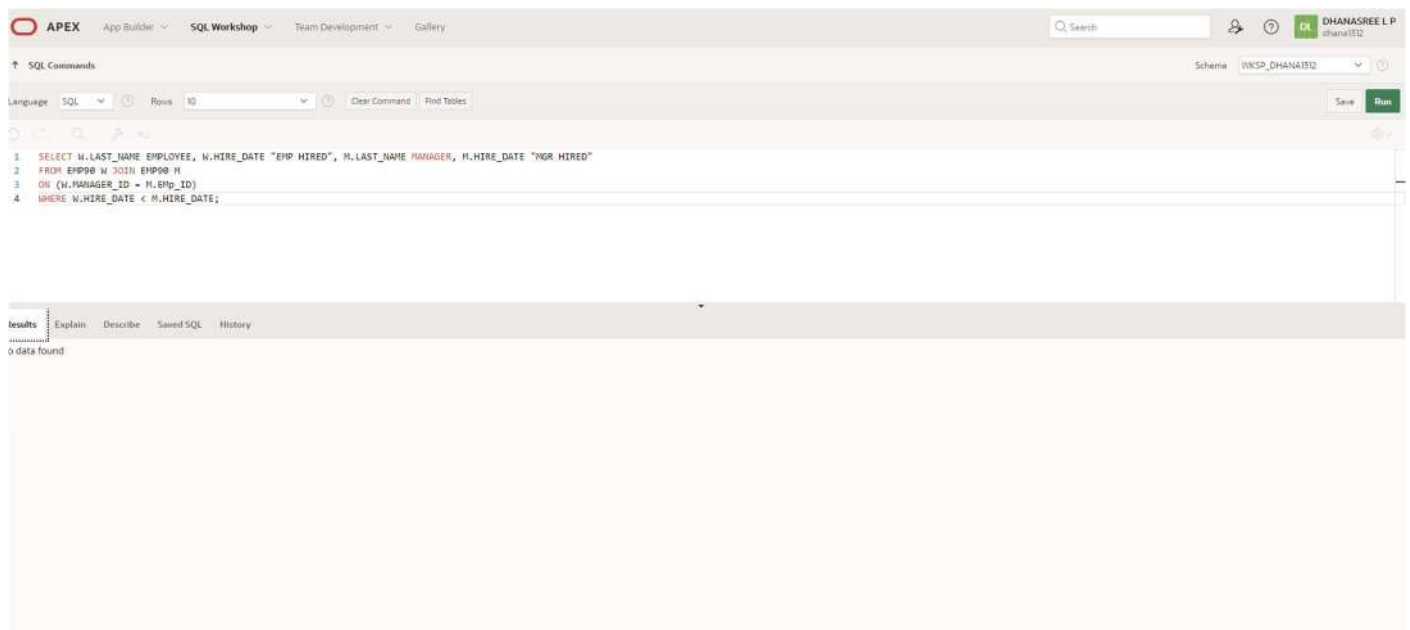
1 rows returned in 0.04 seconds

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

QUERY:

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

OUTPUT:



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

RESULT

AGGREGATING DATA USING GROUP FUNCTIONS

EX_NO : 8

DATE:

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

True/False

TRUE

2.Group functions include nulls in calculations.

True/False

FALSE

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

True/False

FALSE

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

QUERY:

```
select Round(Max (salary),0)"Maximum", Round (Min (salary),0) "Minimum",  
round(sum(salary),0)"sum", round (avg(salary),0) "Average" from EMPB;
```

OUTPUT:

The screenshot displays the Oracle APEX SQL Workshop interface. At the top, the navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'BHANU PRIYA bhannu23' are on the right. The 'SQL Commands' section shows the query: `select Round(Max (salary),0)"Maximum", Round (Min (salary),0) "Minimum", round(sum(salary),0)"sum", round (avg(salary),0) "Average" from EMPB;`. Below the query editor, the 'Results' tab is active, showing a table with four columns: 'Maximum', 'Minimum', 'sum', and 'Average'. The values are 90000, 60000, 500000, and 75000 respectively. A status bar at the bottom indicates '1 rows returned in 0.01 seconds' and provides a 'Download' link.

Maximum	Minimum	sum	Average
90000	60000	500000	75000

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is: `select job_id ,Round(MAX(salary),0) "MAXIMUM",Round (Min(salary),0)"Minimum",Round (SUM(Salary),0)"sum" ,Round (AVg (salary),0)"average" from EMPB group by job_id;` The results are displayed in a table with 5 columns: JOB_ID, MAXIMUM, Minimum, sum, and average. There are 4 rows of data.

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

4 rows returned in 0.01 seconds. Download

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is: `select job_id, count(*) from EMPB where job_id='47' group by job_id ;` The results are displayed in a table with 2 columns: JOB_ID and COUNT(*). There is 1 row of data.

JOB_ID	COUNT(*)
47	1

1 rows returned in 0.01 seconds. Download

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

QUERY:

```
select count(distinct manager_id )"Number of managers" from empb;
```

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command is: `select count(distinct manager_id)"Number of managers" from empb;`. The results pane shows a single row with the value 8, labeled "Number of managers".

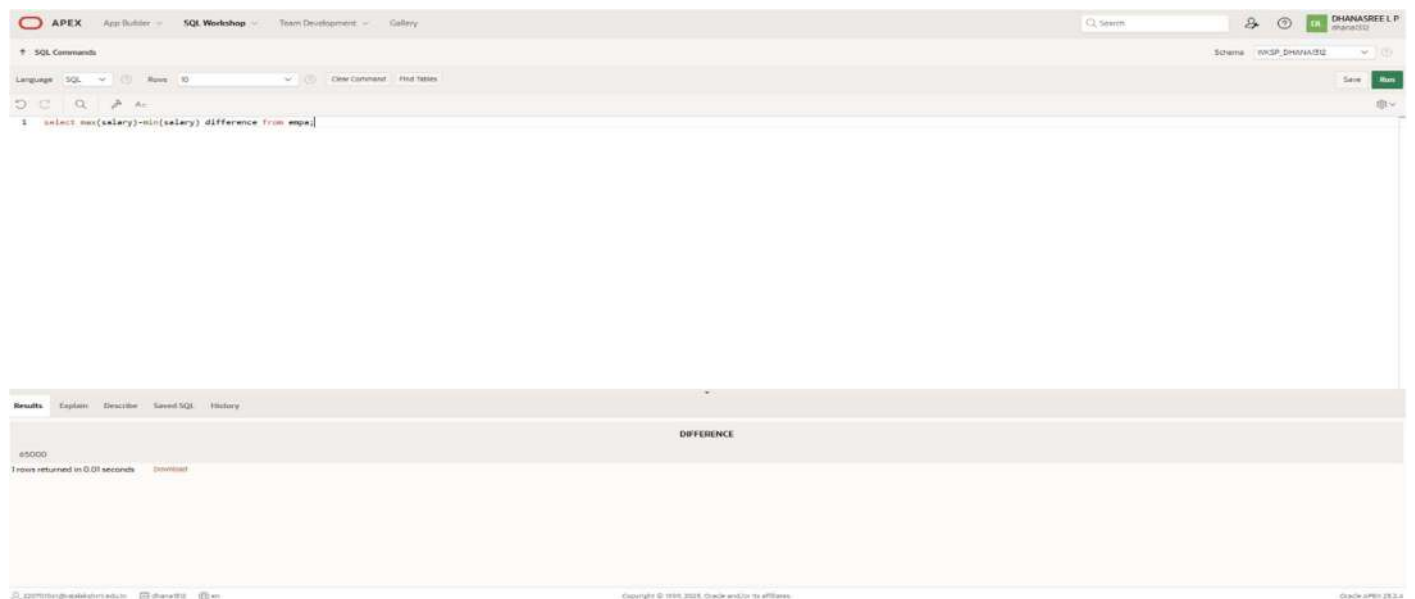
Number of managers
8

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

QUERY:

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

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command is: `select max(salary)-min(salary) difference from empb;`. The results pane shows a single row with the value 45000, labeled "DIFFERENCE".

DIFFERENCE
45000

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

QUERY:

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

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command is: `select manager_id ,min(salary) from empb where manager_id is not null group by manager_id having min(salary) >6000 order by min(salary) desc;`. The results are displayed in a table with the following data:

manager_id	min(salary)
55000	DIFFERENCE

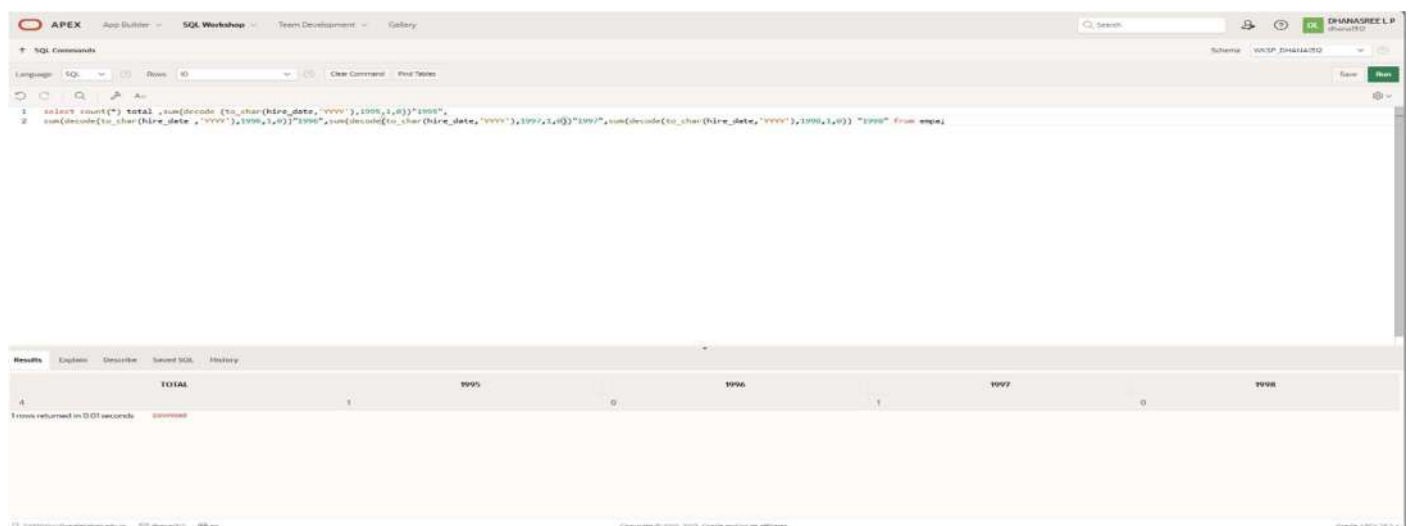
Results returned in 0.01 seconds. The interface also shows tabs for Results, Explain, Describe, Saved SQL, and History.

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

QUERY:

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

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command is: `select count(*) total, sum(decode(to_char(hire_date,'YYYY'),1995,1,0)) "1995", sum(decode(to_char(hire_date,'YYYY'),1996,1,0)) "1996", sum(decode(to_char(hire_date,'YYYY'),1997,1,0)) "1997", sum(decode(to_char(hire_date,'YYYY'),1998,1,0)) "1998" from empb;`. The results are displayed in a table with the following data:

TOTAL	1995	1996	1997	1998
4	1	0	1	0

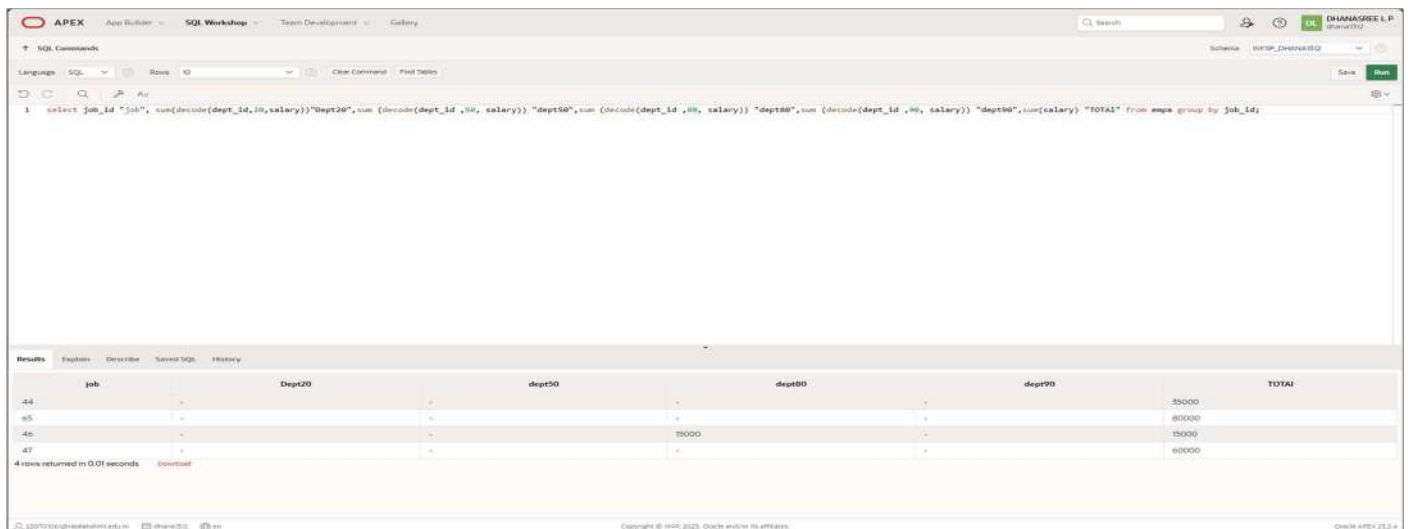
Results returned in 0.01 seconds. The interface also shows tabs for Results, Explain, Describe, Saved SQL, and History.

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

QUERY:

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

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command window contains the query for job-based salary aggregation. The Results tab displays a table with 7 columns: job, Dept20, dept50, dept80, dept90, and TOTAL. There are 4 rows of data corresponding to job IDs 44, 45, 46, and 47.

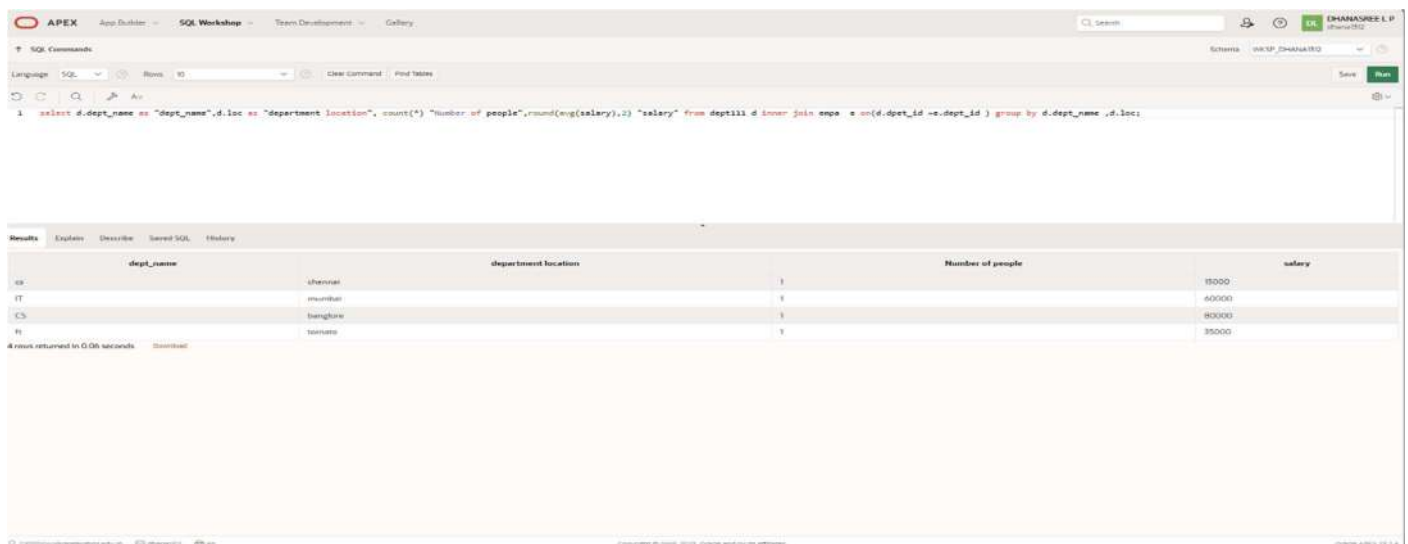
job	Dept20	dept50	dept80	dept90	TOTAL
44					55000
45					60000
46			15000		15000
47					60000

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

QUERY:

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

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command window contains the query for department-level statistics. The Results tab displays a table with 5 columns: dept_name, department location, Number of people, and salary. There are 4 rows of data for departments 44, 45, 46, and 47.

dept_name	department location	Number of people	salary
44	chennai	1	15000
45	mumbai	1	60000
46	bangalore	1	80000
47	toronto	1	35000

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

RESULT:

SUB-QUERIES

EX.NO:9

DATE:

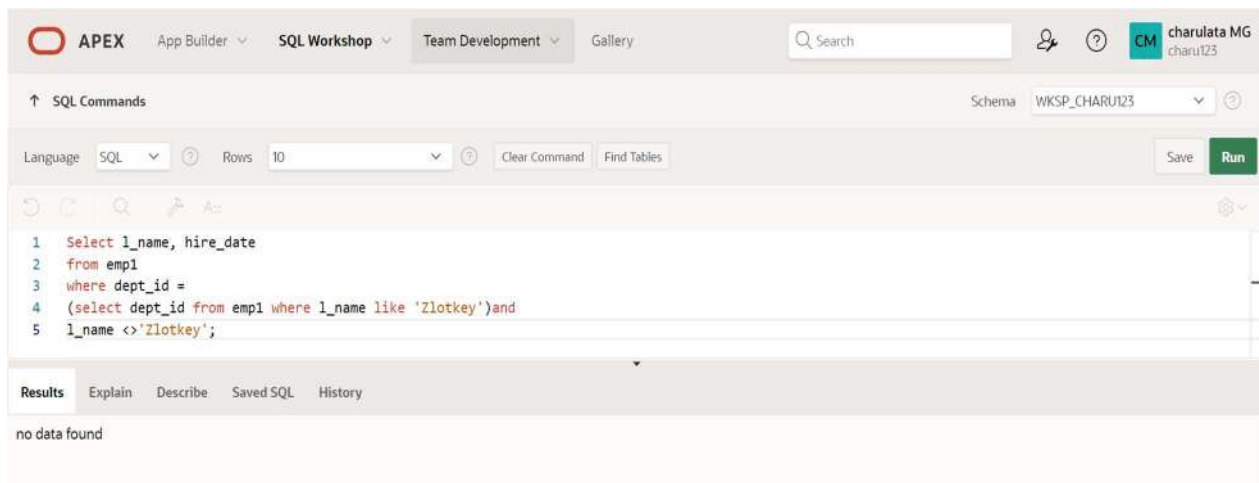
Find the Solution for the following:

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

QUERY:

```
Select l_name, hire_date
from emp1
where dept_id =
(select dept_id from emp1 where l_name like 'Zlotkey')and
l_name <>'Zlotkey';
```

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'charulata MG charu123' are also visible. The 'SQL Commands' section shows the query:

```
1 Select l_name, hire_date
2 from emp1
3 where dept_id =
4 (select dept_id from emp1 where l_name like 'Zlotkey')and
5 l_name <>'Zlotkey';
```

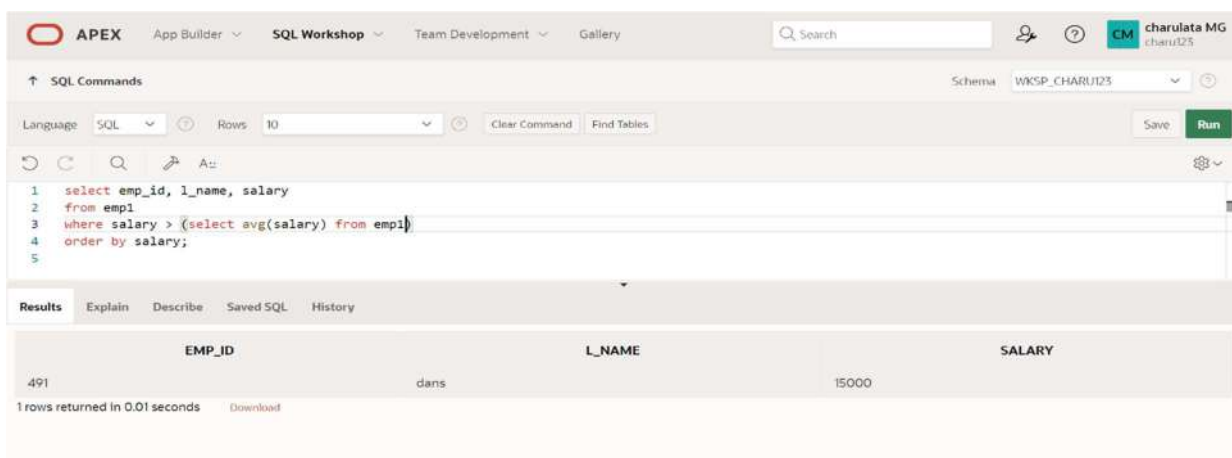
 The 'Results' tab is selected, displaying 'no data found'.

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

QUERY:

```
select emp_id, l_name, salary
from emp1
where salary > (select avg(salary) from emp1)
order by salary;
```

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command is: `select emp_id, l_name, salary from emp1 where salary > (select avg(salary) from emp1) order by salary;`. The results are displayed in a table with columns EMP_ID, L_NAME, and SALARY. One row is returned: emp_id 491, l_name dans, salary 15000.

EMP_ID	L_NAME	SALARY
491	dans	15000

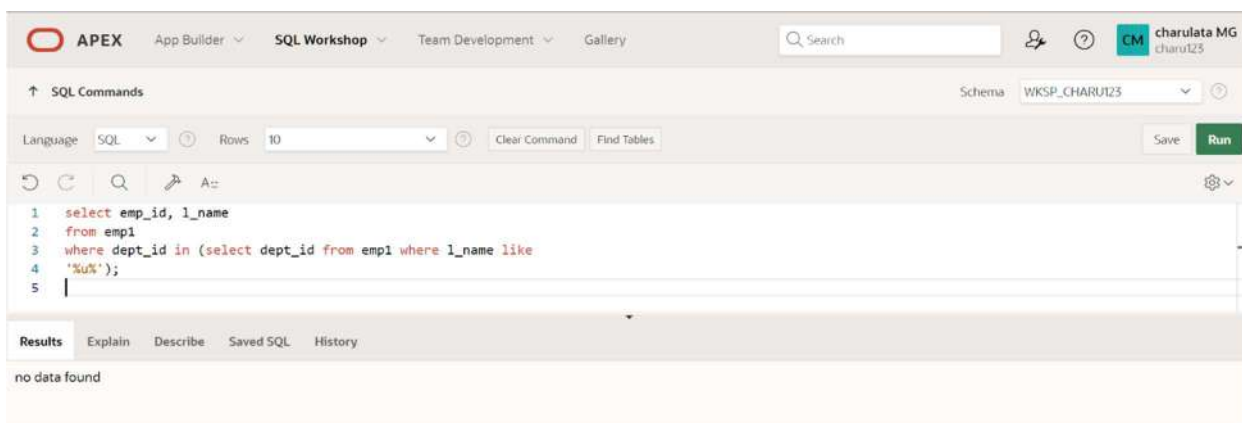
1 rows returned in 0.01 seconds

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, l_name
from emp1
where dept_id in (select dept_id from emp1 where l_name like '%u%');
```

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command is: `select emp_id, l_name from emp1 where dept_id in (select dept_id from emp1 where l_name like '%u%');`. The results section shows "no data found".

EMP_ID	L_NAME
--------	--------

no data found

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 l_name, dept_id, job_id
from emp1
where dept_id in (select dept_id from dept where loc_id =1700);
```

OUTPUT:

The screenshot shows the APEX SQL Workshop interface. The SQL command is: `select l_name, dept_id, job_id from emp1 where dept_id in (select dept_id from dept where loc_id =1700);`. The results tab shows "no data found".

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

QUERY:

```
select l_name, salary
from emp1
where manager_id in (select emp_id from emp1 where
l_name='King');
```

OUTPUT:

The screenshot shows the APEX SQL Workshop interface. The SQL command is: `select l_name, salary from emp1 where manager_id in (select emp_id from emp1 where l_name='King');`. The results tab shows a table with two columns: L_NAME and SALARY. The data row shows King with a salary of 6544. Below the table, it says "1 rows returned in 0.01 seconds" and there is a "Download" link.

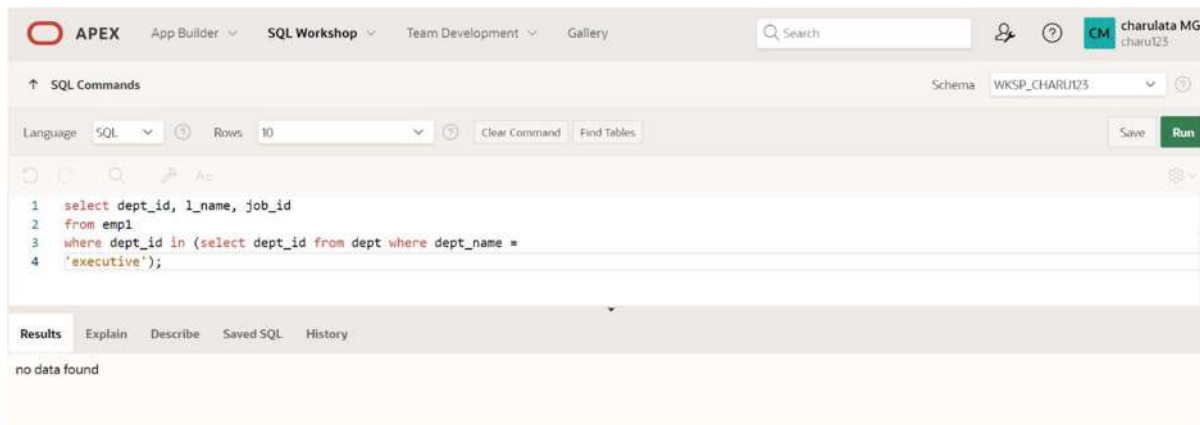
L_NAME	SALARY
King	6544

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

QUERY:

```
select dept_id, l_name, job_id
from emp1
where dept_id in (select dept_id from dept where dept_name =
'executive');
```

OUTPUT:

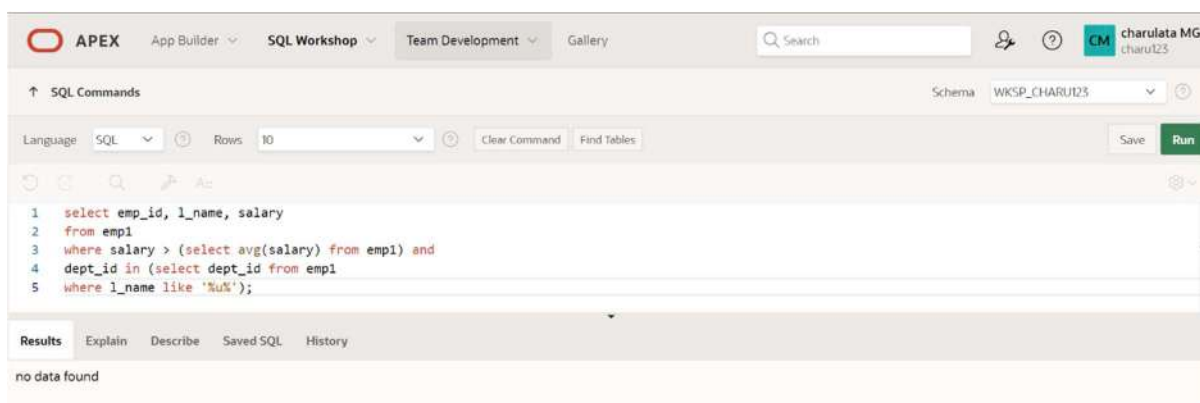


7. Modify the query 3 to display the employee number, last name, and salary of all employees who earn more than the average salary and who work in a department with any employee whose last name contains a u.

QUERY:

```
select emp_id, l_name, salary
from emp1
where salary > (select avg(salary) from emp1) and
dept_id in (select dept_id from emp1
where l_name like '%u%');
```

OUTPUT:



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

RESULT:

USING THE SET OPERATORS

EX.NO:10

DATE:

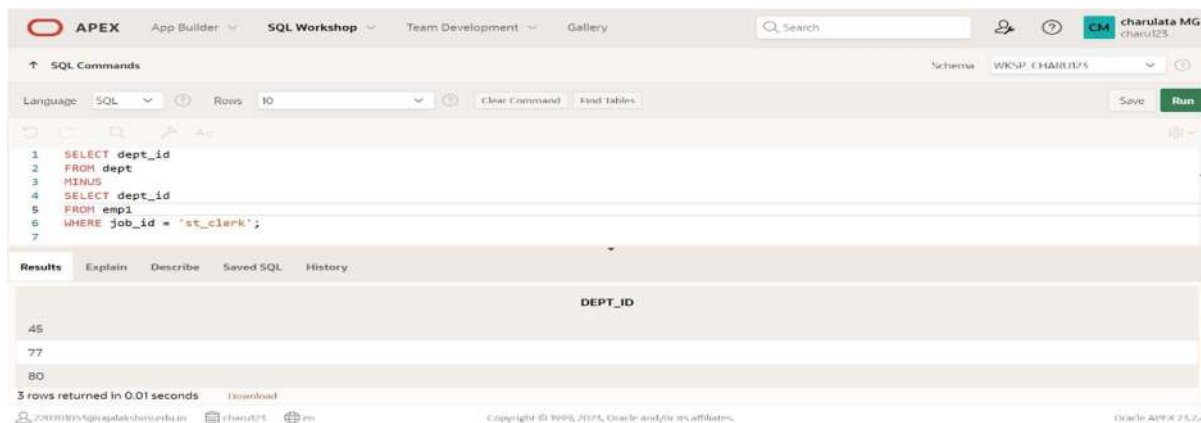
Find the Solution for the following:

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

QUERY:

```
SELECT dept_id
FROM dept
MINUS
SELECT dept_id
FROM emp1
WHERE job_id = 'st_clerk';
```

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL Commands tab is active, displaying the following query:

```
1 SELECT dept_id
2 FROM dept
3 MINUS
4 SELECT dept_id
5 FROM emp1
6 WHERE job_id = 'st_clerk';
7
```

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

DEPT_ID
45
77
80

3 rows returned in 0.01 seconds. Download button is visible.

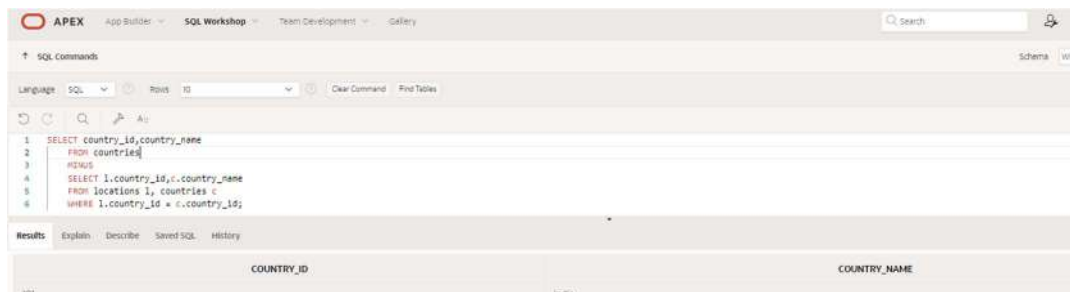
2.HR

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

QUERY:

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

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL Commands tab is active, displaying the following query:

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

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

COUNTRY_ID	COUNTRY_NAME
101	India

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

QUERY:

```
SELECT DISTINCT job_id, dept_id
FROM emp1
WHERE dept_id = 597
```

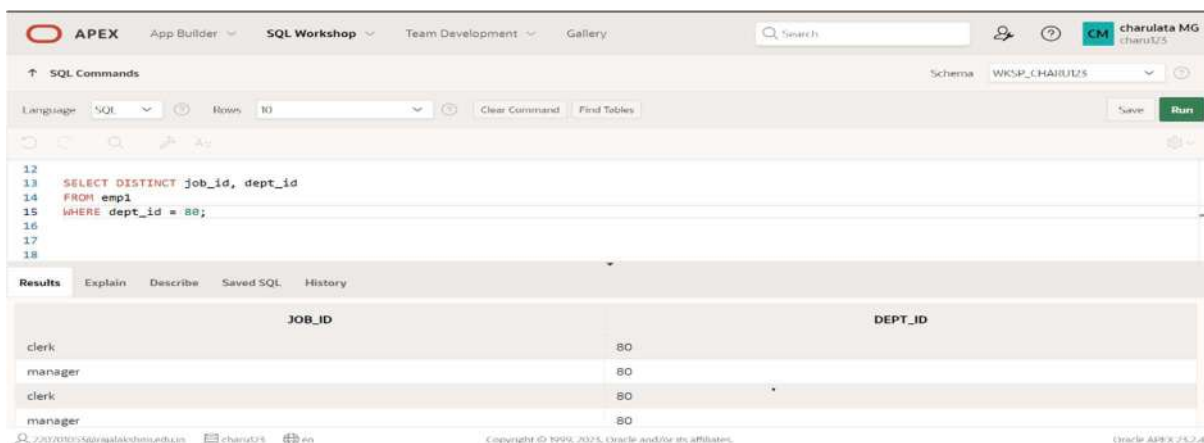
UNION ALL

```
SELECT DISTINCT job_id, dept_id
FROM emp1
WHERE dept_id = 80
```

UNION ALL

```
SELECT DISTINCT job_id, dept_id
FROM emp1
WHERE dept_id = 80;
```

OUTPUT:



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

```
12
13 SELECT DISTINCT job_id, dept_id
14 FROM emp1
15 WHERE dept_id = 80;
16
17
18
```

The results are displayed in a table with two columns: JOB_ID and DEPT_ID. The table contains four rows of data:

JOB_ID	DEPT_ID
clerk	80
manager	80
clerk	80
manager	80

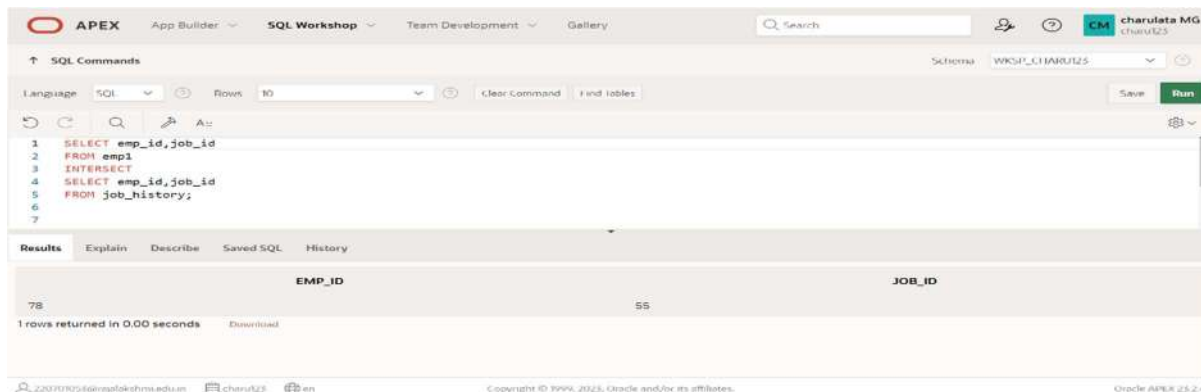
The footer of the screenshot includes the text: "Copyright © 1996, 2021, Oracle and/or its affiliates. Oracle APEX 21.2.4".

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

QUERY:

```
SELECT emp_id,job_id
FROM emp1
INTERSECT
SELECT emp_id,job_id
FROM job_history;
```

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL Commands tab is active, displaying the query: `SELECT emp_id,job_id FROM emp1 INTERSECT SELECT emp_id,job_id FROM job_history;`. The Results tab shows the output with columns EMP_ID and JOB_ID. The first row has values 78 and 55. A message at the bottom indicates '1 rows returned in 0.00 seconds'.

EMP_ID	JOB_ID
78	55

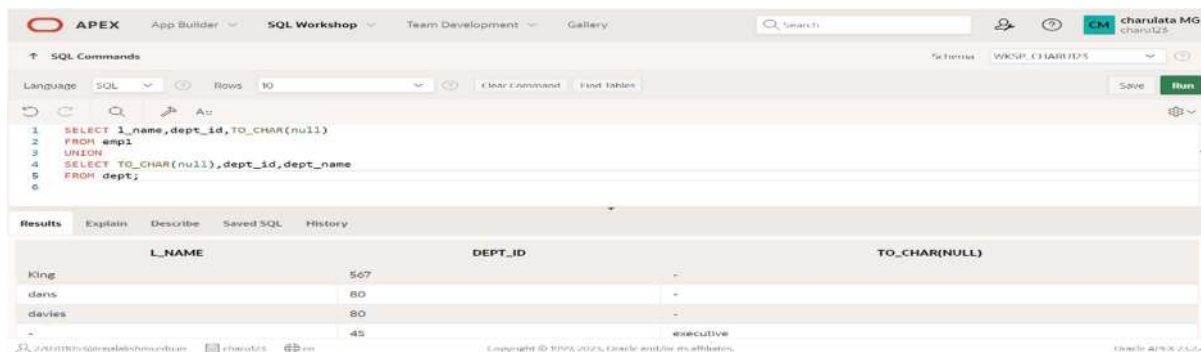
5. The HR department needs a report with the following specifications:

- 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 l_name,dept_id,TO_CHAR(null)
FROM emp1
UNION
SELECT TO_CHAR(null),dept_id,dept_name
FROM dept;
```

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL Commands tab is active, displaying the query: `SELECT l_name,dept_id,TO_CHAR(null) FROM emp1 UNION SELECT TO_CHAR(null),dept_id,dept_name FROM dept;`. The Results tab shows the output with columns L_NAME, DEPT_ID, and TO_CHAR(NULL). The first row has values King, 567, and -. The second row has values davis, 80, and -. The third row has values davis, 80, and -. The fourth row has values -, 45, and executive.

L_NAME	DEPT_ID	TO_CHAR(NULL)
King	567	-
davis	80	-
davis	80	-
-	45	executive

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

RESULT:

EXERCISE-11 CREATING VIEWS

EX.NO:10

DATE:

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

QUERY:

create or replace view employees_vu as select emp_id,l_name employee,dept_id from emp1;

OUTPUT:



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

```
1 create or replace view employees_vu as select emp_id,l_name employee,dept_id from emp1;
```

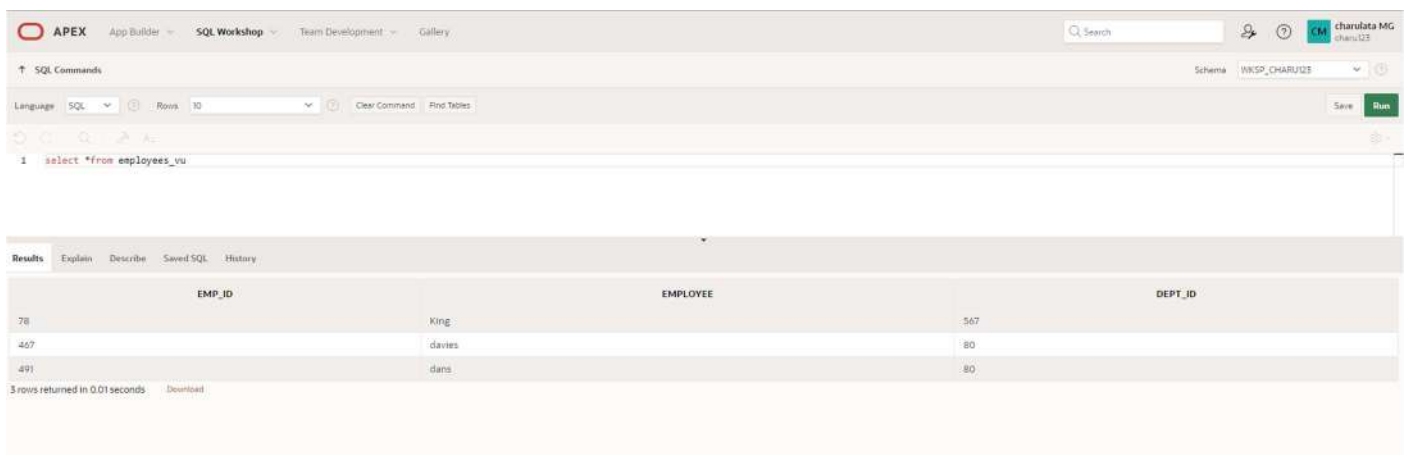
The Results pane shows the message "View created." and the execution time "0.06 seconds".

2.Display the contents of the EMPLOYEES_VU view.

QUERY:

select *from employees_vu;

OUTPUT:



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

```
1 select *from employees_vu
```

The Results pane shows a table with 3 rows and 3 columns: EMP_ID, EMPLOYEE, and DEPT_ID.

EMP_ID	EMPLOYEE	DEPT_ID
78	King	567
407	davies	80
491	dams	80

3 rows returned in 0.01 seconds

3. Select the view name and text from the USER_VIEWS data dictionary views.

QUERY:

```
select view_name, text from user_views;
```

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'charulata MG' are on the right. The 'SQL Commands' section shows the query 'select view_name, text from user_views;'. Below the query editor, the 'Results' tab is active, displaying a table with two columns: 'VIEW_NAME' and 'TEXT'. The table contains one row for 'EMPLOYEES_VU' with the text 'select emp_id, name employee, dept_id from emp1'. The status bar indicates '1 rows returned in 0.03 seconds'.

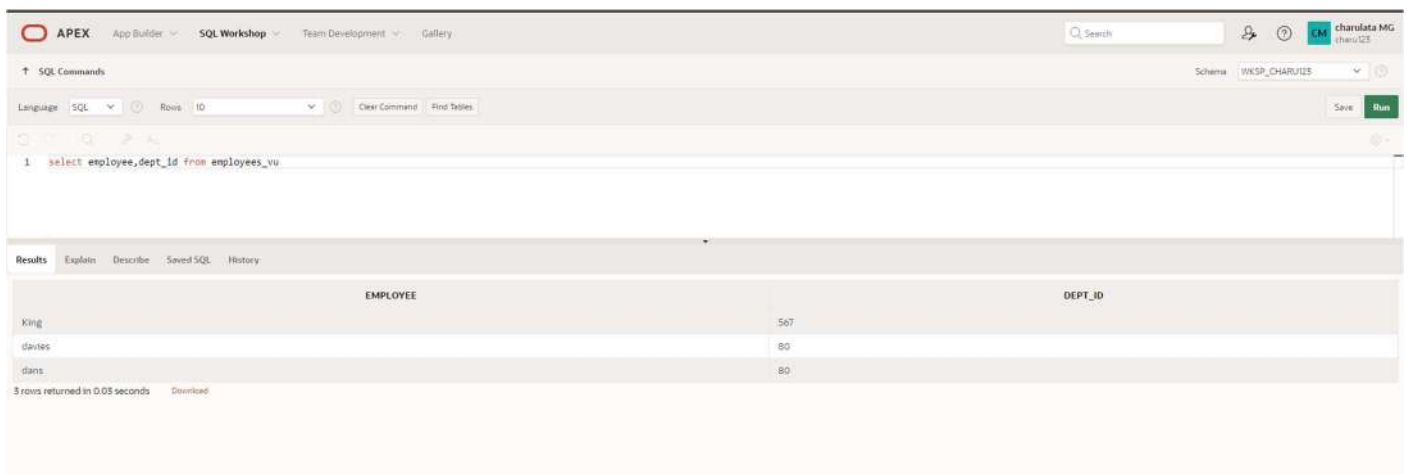
VIEW_NAME	TEXT
EMPLOYEES_VU	select emp_id, name employee, dept_id from emp1

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

QUERY:

```
select employee, dept_id from employees_vu;
```

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'charulata MG' are on the right. The 'SQL Commands' section shows the query 'select employee, dept_id from employees_vu;'. Below the query editor, the 'Results' tab is active, displaying a table with two columns: 'EMPLOYEE' and 'DEPT_ID'. The table contains three rows: 'King' (507), 'davis' (80), and 'davis' (80). The status bar indicates '3 rows returned in 0.05 seconds'.

EMPLOYEE	DEPT_ID
King	507
davis	80
davis	80

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

QUERY:

create view dept50 as select emp_id empno,l_name employee,dept_id dept_no from emp1 where dept_id=50 with check option constraint emp_dept_50;

OUTPUT:

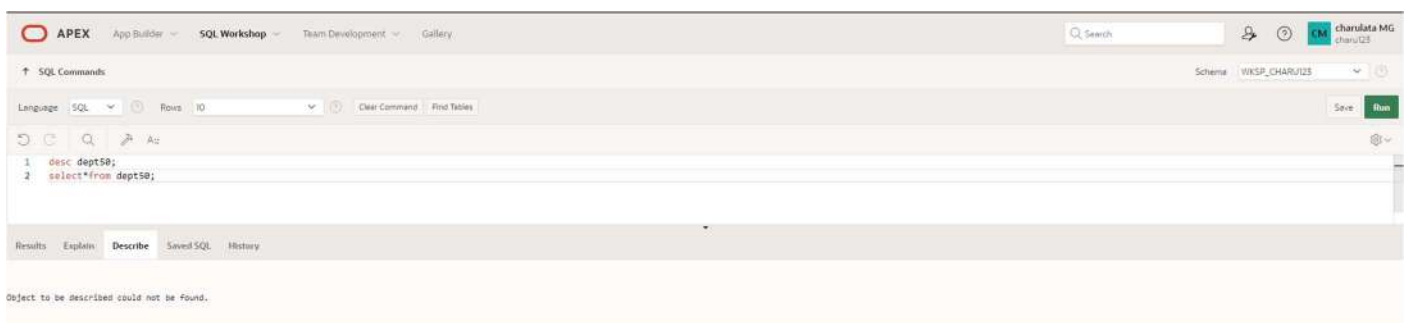


6. Display the structure and contents of the DEPT50 view.

QUERY:

desc dept50;
select*from dept50;

OUTPUT:



7. Attempt to reassign Matos to department 80.

QUERY:

```
update dept50 set dept_no=80 where employee='Matos';
```

OUTPUT:



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

QUERY:

```
create or replace view salary_vu as select e.l_name "Employee",d.dept_name"Department",e.salary "salary",j.grade_level "Grades" from emp1 e,dept d,jb_grade j where e.dept_id=d.dept_id and e.salary between j.low_sal and j.high_sal;
```

OUTPUT:



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

RESULT:

EXERCISE 12

Intro to Constraints; NOT NULL and UNIQUE Constraints

Exp no:

Date:

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

Global Fast Foods global_locations Table						
NAME	TYPE	LENGTH	PRECISION	SCALE	NULLABLE	DEFAULT
id	pk				No	
name						
date_opened					No	
address					No	
city					No	
zip_postal_code						
phone						
email	uk					
manager_id						
emergency_contact						

TYPE is key type

Nullable targets optionality

pk – primary key, uk-Unique key, fk- foreign key

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

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

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

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

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

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

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

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

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

email	uk	VARCHAR2	75			
manager_id		NUMBER	6	0		
emergency_contact		VARCHAR2	20			

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

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

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

```
CREATE TABLE f_global_locations
```

```
( id NUMBER(6,0) CONSTRAINT f_gln_id_pk PRIMARY KEY ,
```

```
name VARCHAR2(50),
```

```
date_opened DATE CONSTRAINT f_gln_dt_opened_nn NOT NULL ENABLE,
```

```
address VARCHAR2(50) CONSTRAINT f_gln_add_nn NOT NULL ENABLE,
```

```
city VARCHAR2(30) CONSTRAINT f_gln_city_nn NOT NULL ENABLE,
```

```
zip_postal_code VARCHAR2(12),
```

```
phone VARCHAR2(20),
```

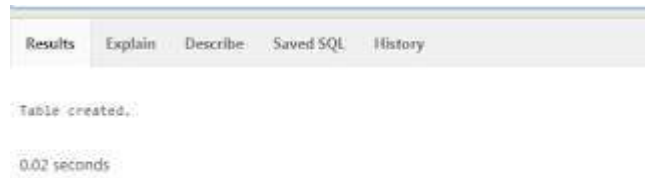
```
email VARCHAR2(75) CONSTRAINT f_gln_email_uk UNIQUE,

manager_id NUMBER(6,0),

emergency_contact VARCHAR2(20)

);
```

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



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

DESCRIBE f_global_locations;

Table	Columns	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
F_GLOBAL_LOCATIONS	ID	NUMBER	6	0		Y	N		
	NAME	VARCHAR2	50				Y		
	DATE_OPENED	DATE	8				Y		
	ADDRESS	VARCHAR2	50				Y		
	CITY	VARCHAR2	30				Y		
	ZIP_POSTAL_CODE	VARCHAR2	12				Y		
	PHONE	VARCHAR2	20				Y		
	EMAIL	VARCHAR2	75				Y		
	MANAGER_ID	NUMBER	6	0			Y		
	EMERGENCY_CONTACT	VARCHAR2	20				Y		

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

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

a. PRIMARY KEY

Uniquely identify each row in table.

b. FOREIGN KEY

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

c. CHECK CONSTRAINT

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

2. Using the column information for the animals table below, name constraints where applicable at the table level, otherwise name them at the column level. Define the primary key (animal_id). The license_tag_number must be unique. The admit_date and vaccination_date columns cannot contain null values.

animal_id NUMBER(6) - PRIMARY KEY

name VARCHAR2(25)

license_tag_number NUMBER(10)- UNIQUE

admit_date DATE- NOT NULL

adoption_id NUMBER(5),

vaccination_date DATE- NOT NULL

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

CREATE TABLE animals

(animal_id NUMBER(6,0) CONSTRAINT anl_anl_id_pk PRIMARY KEY ,

name VARCHAR2(25),

license_tag_number NUMBER(10,0) CONSTRAINT anl_l_tag_num_uk UNIQUE,

admit_date DATE CONSTRAINT anl_adt_dat_nn NOT NULL ENABLE,

```
adoption_id NUMBER(5,0),
vaccination_date DATE CONSTRAINT anl_vcc_dat_nn NOT NULL ENABLE

);
```

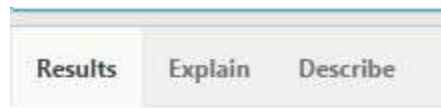


Table created.

0.02 seconds

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

ANIMAL_ID	NAME	LICENSE_TAG_NUMBER	ADMIT_DATE	ADOPTION_ID	VACCINATION_DATE
101	Spot	35540	10-Oct-2004	205	12-Oct-2004

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

```
SELECT * FROM animals;
```

ANIMAL_ID	NAME	LICENSE_TAG_NUMBER	ADMIT_DATE	ADOPTION_ID	VACCINATION_DATE
101	Spot	35540	10-Oct-2004	205	12-Oct-2004

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

```
ALTER TABLE animals
```



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

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

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

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

Gives:

```
SELECT delete_rule  
FROM user_constraints  
WHERE LOWER(table_name) = 'animals' AND constraint_type = 'R';
```

Results	Explain	Describe	Saved SQL	History
DELETE_RULE				
NO ACTION				
1 rows returned in 0.55 seconds Download				

b. ON DELETE SET NULL

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

Results	Explain	Describe	Saved SQL	History
DELETE_RULE				
SET NULL				
1 rows returned in 0.69 seconds Download				

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

1 row(s) deleted.

0.01 seconds

SELECT * FROM animals;

Value in animals.adoption_id where 500 adoptions.id from parent was referred is now set to NULL;

Describe	Saved SQL	History
----------	-----------	---------

ADMIT_DATE	ADOPTION_ID	VACCINATION
10-Oct-2004	-	12-Oct-2004

a. ON DELETE CASCADE

ALTER TABLE animals

ADD CONSTRAINT anl_adopt_id_fk FOREIGN KEY (adoption_id)

REFERENCES adoptions(id) ON DELETE CASCADE ENABLE ;

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

DELETE_RULE

CASCADE

1 rows returned in 0.56 seconds: [Download](#)

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

1 row(s) deleted.

0.01 seconds

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

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

· I cannot call functions that are not deterministic e.g. CURRENT_DATE, CURRENT_TIMESTAMP, DBTIMEZONE, LOCALTIMESTAMP, SESSIONTIMEZONE, SYSDATE, SYSTIMESTAMP, UID, USER, and USERENV

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

RESULT:

EXERCISE 13

Creating Views

Exp no:

Date

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

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

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

```
CREATE VIEW view_d_songs AS
```

```
SELECT d_songs.id, d_songs.title "Song Title", d_songs.artist
```

```
from d_songs INNER JOIN d_types ON d_songs.type_code = d_types.code
```

```
where d_types.description = 'New Age';
```

3.SELECT * FROM view_d_songs. What was returned?

```
SELECT * FROM view_d_songs ;
```

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

Or use alias after the CREATE statement as shown.

```
CREATE OR REPLACE VIEW view_d_songs AS
```

```
SELECT d_songs.id, d_songs.title "Song Title", d_songs.artist, d_songs.type_code
```

```
from d_songs INNER JOIN d_types ON d_songs.type_code = d_types.code
```

```
where d_types.description = 'New Age';
```

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

```
CREATE OR REPLACE VIEW view_d_events_pkgs AS

SELECT evt.name "Name of Event", TO_CHAR(evt.event_date, 'dd-Month-yyyy') "Event date",
thm.description "Theme description"

FROM d_events evt INNER JOIN d_themes thm ON evt.theme_code = thm.code

WHERE evt.event_date <= ADD_MONTHS(SYSDATE,1);
```

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

```
CREATE OR REPLACE VIEW view_min_max_avg_dpt_salary ("Department Id", "Department Name", "Max Salary", "Min Salary", "Average Salary") AS

SELECT dpt.department_id, dpt.department_name, MAX(NVL(emp.salary,0)), MIN(NVL(emp.salary,0)),
ROUND(AVG(NVL(emp.salary,0)),2)

FROM departments dpt LEFT OUTER JOIN employees emp ON dpt.department_id = emp.department_id

GROUP BY (dpt.department_id, dpt.department_name);
```

DML Operations and Views

Use the DESCRIBE statement to verify that you have tables named copy_d_songs, copy_d_events, copy_d_cds, and copy_d_clients in your schema. If you don't, write a query to create a copy of each.

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

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

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

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

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

Use the same syntax but change table_name of the other tables.

2. Use the CREATE or REPLACE option to create a view of all the columns in the copy_d_songs table called view_copy_d_songs.

```
CREATE OR REPLACE VIEW view_copy_d_songs AS  
SELECT *
```

```
FROM copy_d_songs;
```

```
SELECT * FROM view_copy_d_songs;
```

3. Use view_copy_d_songs to INSERT the following data into the underlying copy_d_songs table. Execute a SELECT * from copy_d_songs to verify your DML command. See the graphic.

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

```
INSERT INTO view_copy_d_songs(id,title,duration,artist,type_code)
```

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

4. Create a view based on the DJs on Demand COPY_D_CDS table. Name the view read_copy_d_cds. Select all columns to be included in the view. Add a WHERE clause to restrict the year to 2000. Add the WITH READ ONLY option.

```
CREATE OR REPLACE VIEW read_copy_d_cds AS
```

```
SELECT *
```

```
FROM copy_d_cds
```

```
WHERE year = '2000'
```

```
WITH READ ONLY ;
```

```
SELECT * FROM read_copy_d_cds;
```

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

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

6. Use REPLACE to modify read_copy_d_cds. Replace the READ ONLY option with WITH CHECK OPTION CONSTRAINT ck_read_copy_d_cds. Execute a SELECT * statement to verify that the view exists.

```
CREATE OR REPLACE VIEW read_copy_d_cds AS
```

```
SELECT *
```

```
FROM copy_d_cds
```

```
WHERE year = '2000'
```

```
WITH CHECK OPTION CONSTRAINT ck_read_copy_d_cds;
```

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

```
DELETE FROM read_copy_d_cds
```

```
WHERE year = '2000';
```

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

```
DELETE FROM read_copy_d_cds
```

```
WHERE cd_number = 90;
```

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

```
DELETE FROM read_copy_d_cds
```

```
WHERE year = '2001';
```

10. Execute a SELECT * statement for the base table copy_d_cds. What rows were deleted?

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

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

Delete restricted if it contains

Modify restricted if it contains

INSERT restricted if it contains

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

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

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

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

3 Reasons To Believe The Singularity Is Near as per Greg Satell on Forbes:

- We're Going Beyond Moore's Law
- Robots Are Doing Human Jobs
- We're Editing Genes

Managing Views

1.Create a view from the copy_d_songs table called view_copy_d_songs that includes only the title and artist. Execute a SELECT * statement to verify that the view exists.

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

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

```
DROP VIEW view_copy_d_songs;  
  
SELECT * FROM view_copy_d_songs;  
  
ORA-00942: table or view does not exist
```

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

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

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

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

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

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

Indexes and Synonyms

What is an index and what is it used for?

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

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

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

3.When will an index be created automatically?

For primary/unique keys: Although unique index can be created manually, but preferred should be by using unique/primary constraint in the table. So, it means that primary key/unique key use already existing unique index but if index is not present already, it is created while applying unique/primary key constraint.

4.Create a nonunique index (foreign key) for the DJs on Demand column (cd_number) in the D_TRACK_LISTINGS table. Use the Oracle Application Express SQL Workshop Data Browser to confirm that the index was created.

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

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

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

6.Use a SELECT statement to display the index_name, table_name, and uniqueness from the data dictionary USER_INDEXES for the DJs on Demand D_EVENTS table.

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

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

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

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

```
CREATE INDEX d_ptr_last_name_idx
```

```
ON d_partners(LOWER(last_name));
```

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

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

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

10.Drop the synonym that you created in question

```
DROP SYNONYM dj_tracks2;
```

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

RESULT:

EXERCISE-14

OTHER DATABASE OBJECTS

Ex no:

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 APEX SQL Workshop interface. The SQL command `CREATE SEQUENCE dept_id_seq START WITH 200 INCREMENT BY 10 MAXVALUE 1000;` has been entered and executed. The results pane shows the message "Sequence created." and the execution time "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 APEX SQL Workshop interface with the query `SELECT sequence_name, max_value, increment_by, last_number FROM user_sequences;` executed. The results pane displays a table with the following data:

SEQUENCE_NAME	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
DEPT_ID_SEQ	1000	10	200

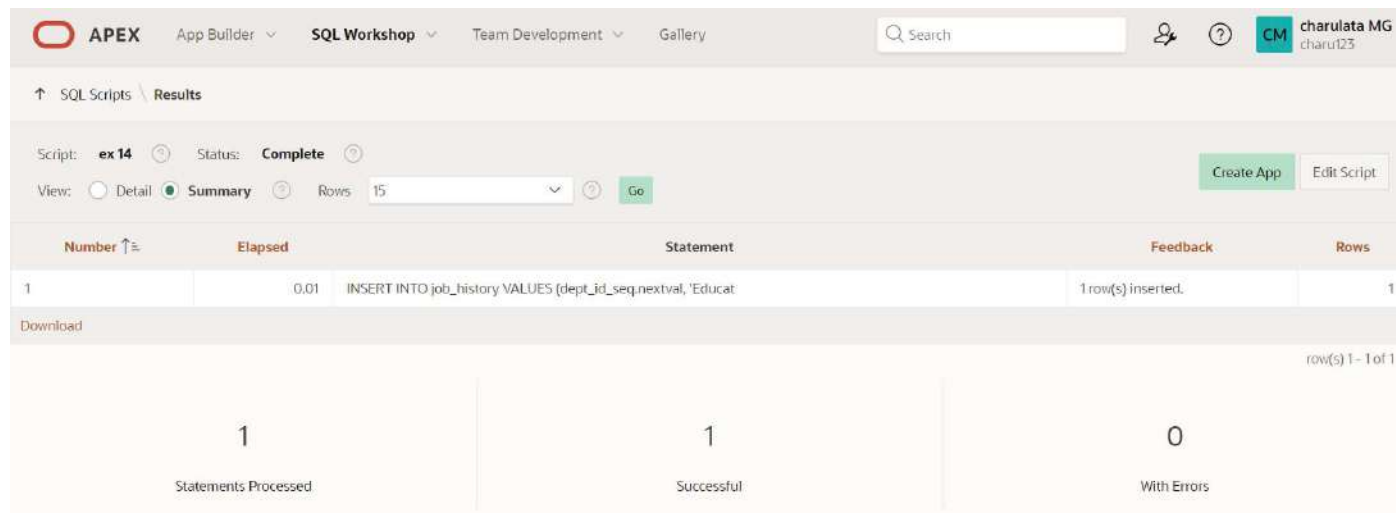
1 rows returned in 0.01 seconds

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

Query:

```
INSERT INTO dept VALUES (dept_id_seq.nextval, '&#39;Education&#39;);
```

Output:



The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'charulata MG charu123' are on the right. The main area is titled 'SQL Scripts \ Results'. It shows a script named 'ex 14' with a status of 'Complete'. The view is set to 'Summary' and 'Rows' is 15. A 'Go' button is present. Below this is a table with columns: Number, Elapsed, Statement, Feedback, and Rows. The table contains one row: 1, 0.01, INSERT INTO job_history VALUES (dept_id_seq.nextval, 'Educat', 1 row(s) inserted., 1. A 'Download' button is on the left. At the bottom, a summary bar shows '1 Statements Processed', '1 Successful', and '0 With Errors'.

Number	Elapsed	Statement	Feedback	Rows
1	0.01	INSERT INTO job_history VALUES (dept_id_seq.nextval, 'Educat	1 row(s) inserted.	1

Download

row(s) 1 - 1 of 1

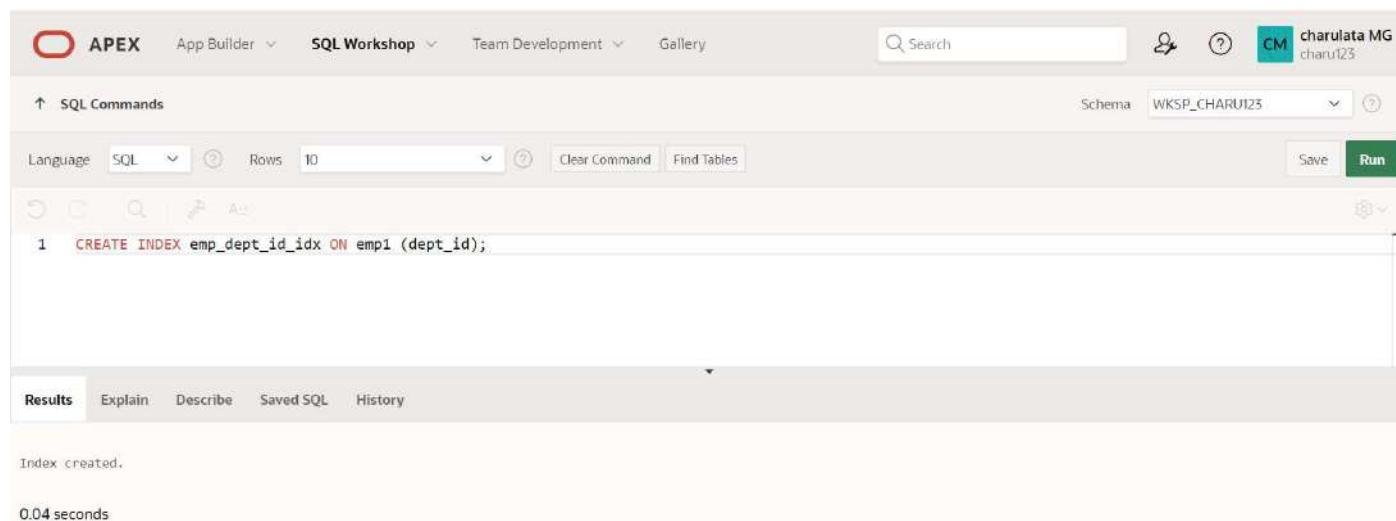
1 Statements Processed | 1 Successful | 0 With Errors

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

Query:

```
CREATE INDEX emp_dept_id_idx ON emp1 (dept_id);
```

Output:



The screenshot shows the APEX SQL Workshop interface. The top navigation bar is the same as the previous screenshot. The main area is titled 'SQL Commands'. The schema is set to 'WKSP_CHARU123'. The language is 'SQL' and 'Rows' is 10. There are buttons for 'Clear Command', 'Find Tables', 'Save', and 'Run'. The SQL command entered is 'CREATE INDEX emp_dept_id_idx ON emp1 (dept_id);'. Below the command area, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, showing 'Index created.' and '0.04 seconds'.

Schema: WKSP_CHARU123

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

1 CREATE INDEX emp_dept_id_idx ON emp1 (dept_id);

Results Explain Describe Saved SQL History

Index created.

0.04 seconds

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

Query:

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

Output:



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

RESULT:

EXERCISE-15 Controlling User Access

Ex no:

Date:

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

Query:

The CREATE SESSION system privilege

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

Query:

The CREATE TABLE privilege

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

Query:

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

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

Query:

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

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

Query:

The ALTER USER statement

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

Query:

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

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

7. Query all the rows in your DEPARTMENTS table.

Query:

SELECT * FROM departments;

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

Query:

Team 1 executes this INSERT statement. INSERT INTO departments(department_id, department_name) VALUES (500, 'Education'); COMMIT;

Team 2 executes this INSERT statement. INSERT INTO departments(department_id, department_name) VALUES (510, 'Administration'); COMMIT;

9. Query the USER_TABLES data dictionary to see information about the tables that you own.

Query :

SELECT table_name FROM user_tables;

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

Query:

Team 1 revokes the privilege.

REVOKE select

ON departments

FROM user2;

Team 2 revokes the privilege.

REVOKE select

ON departments

FROM user1;

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

Query:

Team 1 executes this INSERT statement.

DELETE FROM departments

WHERE department_id = 500;

COMMIT;

Team 2 executes this INSERT statement.

DELETE FROM departments

WHERE department_id = 510;

Evaluation Procedure

Marks

awarded

Practice Evaluation

(5)

Viva(5)

Total (10)

Faculty Signature

COMMIT;

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

RESULT:

PL/SQL

Ex no:

Date:

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

Query:

```
DECLARE
    incentive NUMBER(8,2);
BEGIN
    SELECT salary * 0.12 INTO incentive
    FROM emp1
    WHERE emp_id = 110;
    DBMS_OUTPUT.PUT_LINE('Incentive = ' || TO_CHAR(incentive));
END;
/
```

Output:

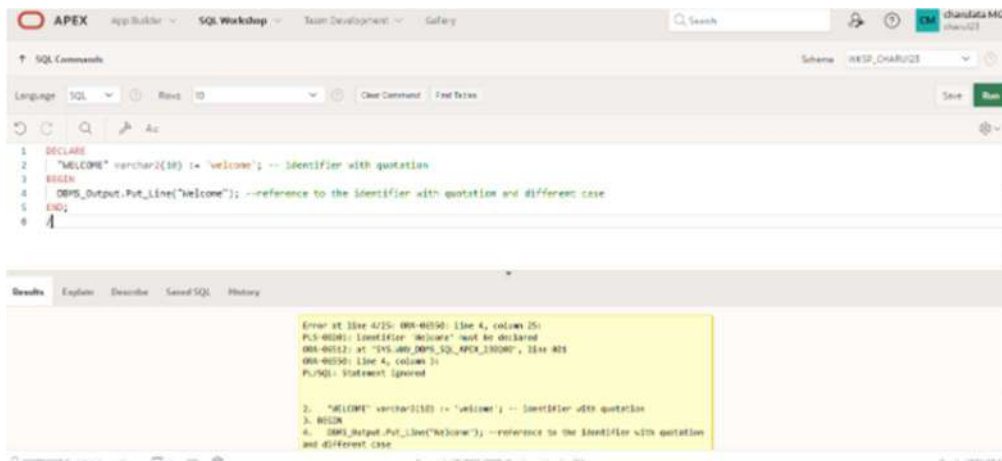


2. Write a PL/SQL block to show an invalid case-insensitive reference to a quoted and without quoted user-defined identifier.

Query:

```
DECLARE
    "WELCOME" varchar2(10) := 'welcome'; -- identifier with quotation
BEGIN
    DBMS_Output.Put_Line("Welcome"); --reference to the identifier with quotation and different case
END;
/
```

Output:



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

Query:

```
DECLARE
  salary_of_emp NUMBER(8,2);
PROCEDURE approx_salary (
  emp      NUMBER,
  empsal IN OUT NUMBER,
  address  NUMBER
) IS
BEGIN empsal := empsal + address;
END;
BEGIN
  SELECT salary INTO salary_of_emp FROM emp1 WHERE emp_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     address      NUMBER
7   ) IS
8   BEGIN empsal := empsal + address;
9   END;
10 BEGIN
11   SELECT salary INTO salary_of_emp FROM emp1 WHERE emp_id = 122;
12   DBMS_OUTPUT.PUT_LINE ('Before invoking procedure, salary_of_emp: ' || salary_of_emp);
13   approx_salary (100, salary_of_emp, 1000);
14   DBMS_OUTPUT.PUT_LINE('After invoking procedure, salary_of_emp: ' || salary_of_emp);
15 END;
```

Results Explain Describe Saved SQL History

Before invoking procedure, salary_of_emp: 8165
After invoking procedure, salary_of_emp: 9165

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:

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

Results Explain Describe Saved SQL History

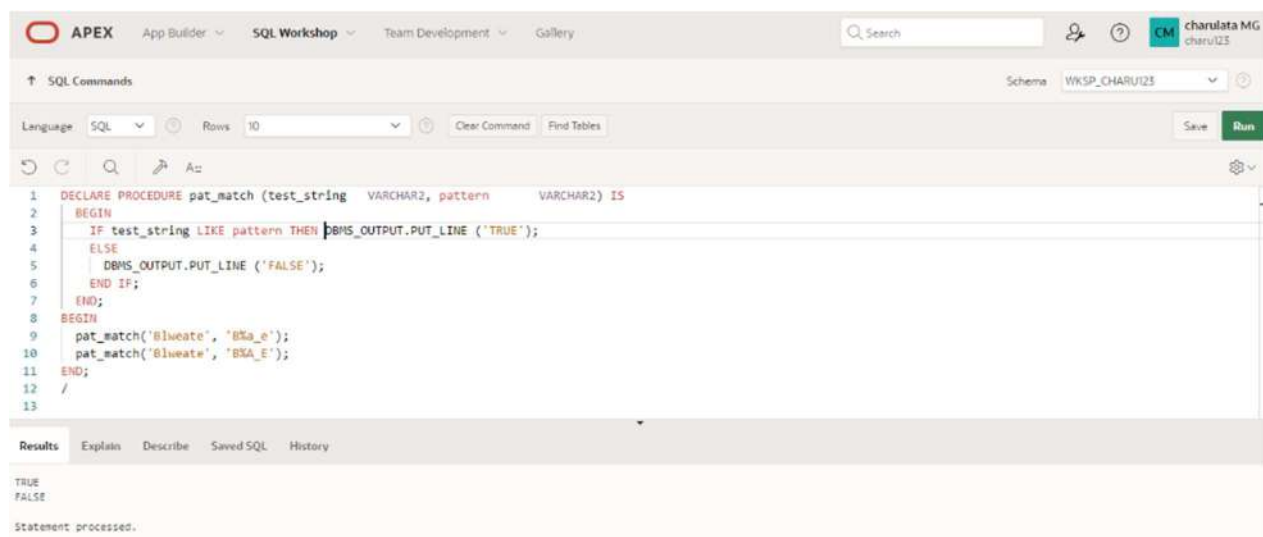
Procedure created.

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

Query:

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

Output:



The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'SQL Commands' window is active, showing a PL/SQL block with the following code:

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

The 'Results' window at the bottom shows the output of the execution:

```
TRUE
FALSE
Statement processed.
```

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

Query:

```
SET SERVEROUTPUT ON DECLARE tot_emp NUMBER;

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

```

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

END IF;

END;

```

Output:

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL Commands pane contains the following PL/SQL code:

```

3  get_dep_id NUMBER;
4  BEGIN
5  get_dep_id := 80;
6  SELECT Count(*)
7  INTO tot_emp
8  FROM employees e
9  JOIN departments d
10 ON e.department_id = d.dept_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;

```

The Results pane shows the output of the query:

```

The employees are in the department 80 is: 6
There are 39 vacancies in department 80
Statement processed.
0.03 seconds

```

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

Query:

```

DECLARE tot_emp NUMBER;

get_dep_id NUMBER; BEGIN get_dep_id := 80;

SELECT Count(*) INTO tot_emp FROM employees e join departments d ON e.department_id = d.dept_id

WHERE e.department_id = get_dep_id;

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

IF tot_emp >= 45 THEN

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

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

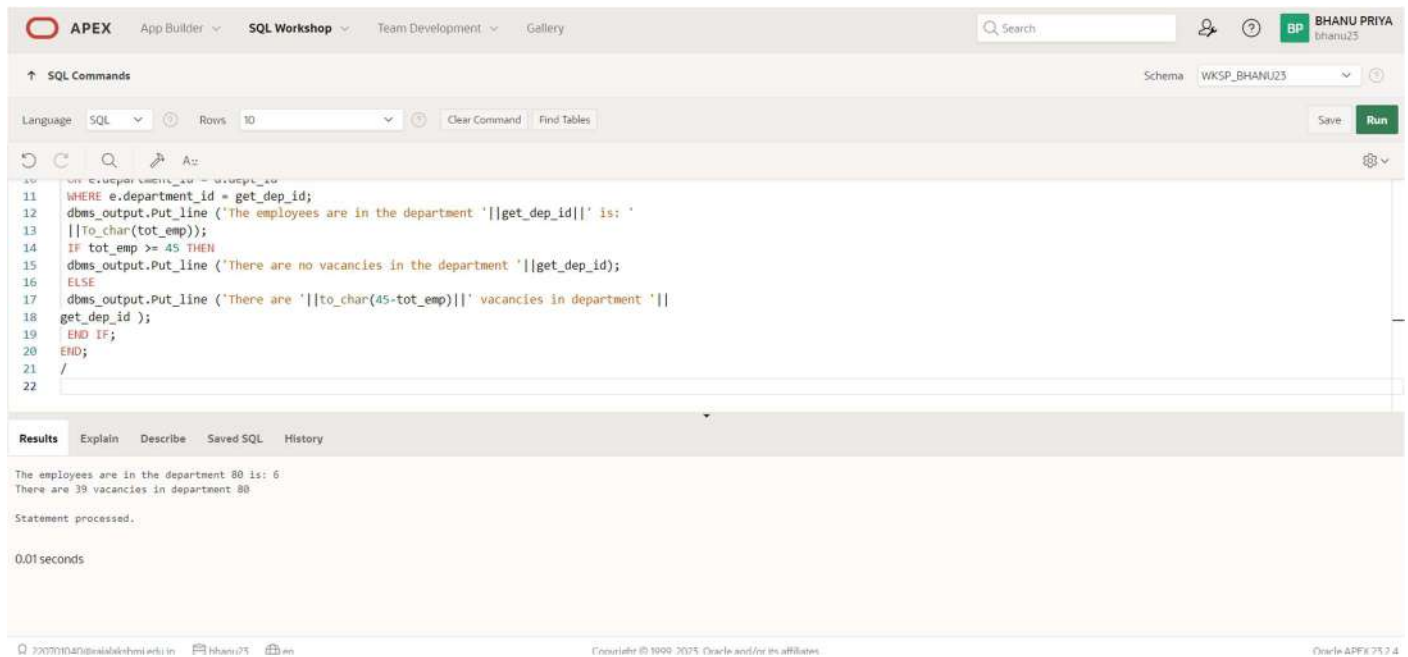
END IF;

END;

/

```

Output:



The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'BHANU PRIYA' are on the right. The 'SQL Commands' section is active, showing a PL/SQL program with line numbers 11 to 22. The program logic checks the number of employees in a department against a total of 45. The 'Results' tab at the bottom shows the output of the program execution.

```
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 /
22
```

Results Explain Describe Saved SQL History

The employees are in the department 80 is: 6
There are 39 vacancies in department 80.

Statement processed.

0.01 seconds

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

Query:

```
DECLARE v_employee_id employees.employee_id%TYPE;
```

```
v_full_name employees.first_name%TYPE;
```

```
v_job_id employees.job_id%TYPE;
```

```
v_hire_date employees.hire_date%TYPE;
```

```
v_salary employees.salary%TYPE;
```

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

```
BEGIN
```

```
DBMS_OUTPUT.PUT_LINE('Employee ID | Full Name | Job Title | Hire Date | Salary');
```

```
DBMS_OUTPUT.PUT_LINE('-----');
```

```
OPEN c_employees;
```

```
FETCH c_employees INTO v_employee_id, v_full_name, v_job_id, v_hire_date, v_salary;
```

```
WHILE c_employees%FOUND LOOP
```

```

DBMS_OUTPUT.PUT_LINE(v_employee_id || ' ' || v_full_name || ' ' || v_job_id || ' ' || v_hire_date
|| ' ' || v_salary);

FETCH c_employees INTO v_employee_id, v_full_name, v_job_id, v_hire_date, v_salary;

END LOOP;

CLOSE c_employees;

END;

/

```

Output:

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL Commands window contains the following PL/SQL code:

```

1 DECLARE
2   v_employee_id employees.employee_id%TYPE;
3   v_full_name employees.first_name%TYPE;
4   v_job_id employees.job_id%TYPE;
5   v_hire_date employees.hire_date%TYPE;
6   v_salary employees.salary%TYPE;
7   CURSOR c_employees IS
8     SELECT employee_id, first_name || ' ' || last_name AS full_name, job_id, hire_date, salary
9     FROM employees;
10 BEGIN
11   DBMS_OUTPUT.PUT_LINE('Employee ID | Full Name | Job Title | Hire Date | Salary');
12   DBMS_OUTPUT.PUT_LINE('-----');
13   OPEN c_employees;
14   FETCH c_employees INTO v_employee_id, v_full_name, v_job_id, v_hire_date, v_salary;

```

The Results window shows the output of the program:

Employee ID	Full Name	Job Title	Hire Date	Salary
2	Fleming Janu	ac_account	02/01/1998	60000
1	Stephen davis	sales_rep	02/23/1998	50000
4	Chris Jones	ac_account	05/01/1998	48000
5	Cameron Brown	st_clerk	04/22/1997	30000
3	John Williams	hr_rep	02/19/1994	20000

Statement processed.

0.01 seconds

Copyright © 1996, 2025, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

Query:

```

DECLARE CURSOR emp_cursor IS SELECT e.employee_id, e.first_name, m.first_name AS manager_name
FROM employees e LEFT JOIN employees m ON e.manager_id = m.employee_id;

emp_record emp_cursor%ROWTYPE;

BEGIN

OPEN emp_cursor;

FETCH

emp_cursor INTO emp_record;

WHILE emp_cursor%FOUND LOOP

DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp_record.employee_id);

```



```

DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_record.first_name);

DBMS_OUTPUT.PUT_LINE('Manager Name: ' || emp_record.manager_name);

DBMS_OUTPUT.PUT_LINE('-----');

FETCH emp_cursor INTO emp_record;

END LOOP;

CLOSE emp_cursor;

END;

/

```

Output:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'BHANU PRIYA bhanu23' are on the right. The 'SQL Commands' tab is active, showing a PL/SQL program with line numbers 1 through 14. The program declares a cursor, selects employee and manager data, and loops through the results, printing employee ID, name, and manager name. The 'Results' tab at the bottom displays the output of the program, showing three rows of data for employees with IDs 4, 2, and 5.

```

1 DECLARE
2   CURSOR emp_cursor IS
3     SELECT e.employee_id, e.first_name, m.first_name AS manager_name
4     FROM employees e
5     LEFT JOIN employees m ON e.manager_id = m.employee_id;
6   emp_record emp_cursor%ROWTYPE;
7 BEGIN
8   OPEN emp_cursor;
9   FETCH emp_cursor INTO emp_record;
10  WHILE emp_cursor%FOUND LOOP
11    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp_record.employee_id);
12    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_record.first_name);
13    DBMS_OUTPUT.PUT_LINE('Manager Name: ' || emp_record.manager_name);
14    DBMS_OUTPUT.PUT_LINE('-----');

```

Results

```

Employee ID: 4
Employee Name: Chris
Manager Name:
-----
Employee ID: 2
Employee Name: Fleming
Manager Name:
-----
Employee ID: 5
Employee Name: Cameron
Manager Name:

```

10. 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; j
ob_record job_cursor%ROWTYPE;

BEGIN

OPEN job_cursor;

FETCH job_cursor INTO job_record;

WHILE job_cursor%FOUND LOOP

DBMS_OUTPUT.PUT_LINE('Job ID: ' || job_record.job_id);

DBMS_OUTPUT.PUT_LINE('Minimum Salary: ' || job_record.lowest_sal);

DBMS_OUTPUT.PUT_LINE('-----'); FETCH job_cursor INTO job_record;

```

```
END LOOP;

CLOSE job_cursor;

END;
```

Output:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user information 'BHANU PRIYA bhannu23' are on the right. The 'SQL Commands' tab is active, showing a PL/SQL program with line numbers 1 through 13. The program declares a cursor, opens it, and loops through job records, outputting job ID and minimum salary. The 'Results' tab at the bottom displays the output of the program, showing three rows of data for job IDs 'ac_account', 'sales_rep', and 'st_clerk', each with a minimum salary of 40000.

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

Results

```
Job ID: ac_account
Minimum Salary: 40000
-----
Job ID: sales_rep
Minimum Salary: 40000
-----
Job ID: ac_account
Minimum Salary: 40000
-----
Job ID: st_clerk
Minimum Salary: 40000
-----
```

11. 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;

END;

/
```

Output:

The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar is on the right. Below the navigation bar, the 'SQL Commands' tab is active, showing a PL/SQL query. The query is as follows:

```
1 DECLARE
2   CURSOR employees_cur IS
3     SELECT employee_id, last_name, job_id, start_date
4     FROM employees NATURAL JOIN job_history;
5   emp_start_date DATE;
6 BEGIN
7   dbms_output.put_line(Rpad('Employee ID', 15) || Rpad('Last Name', 25) || Rpad('Job Id', 35)
8   || 'Start Date');
9   dbms_output.put_line('-----');
10  FOR emp_sal_rec IN employees_cur LOOP
11    -- find out most recent end_date in job_history
12    SELECT Max(end_date) + 1
13    INTO emp_start_date
14    FROM job_history;
```

Below the query editor, the 'Results' tab is active, displaying the output of the query. The output is a table with four columns: 'Employee ID', 'Last Name', 'Job Id', and 'Start Date'. The data is as follows:

Employee ID	Last Name	Job Id	Start Date
2	Janu	ac_account	22-sep-1995
1	davies	sales_rep	16-mar-1997

Below the table, the text 'Statement processed.' and '0.02 seconds' are displayed. The bottom of the interface shows the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 23.2'.

12. 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:


APEX
App Builder
SQL Workshop
Team Development
Gallery


BHANU PRIYA
bhanu25

SQL Commands
Schema WKSP_BHANU25

Language SQL Rows 10
Clear Command
Find Tables
Save
Run

```




1 DECLARE
2   v_employee_id employees.employee_id%TYPE;
3   v_first_name employees.last_name%TYPE;
4   v_end_date job_history.end_date%TYPE;
5   CURSOR c_employees IS
6     SELECT e.employee_id, e.first_name, jh.end_date
7     FROM employees e
8     JOIN job_history jh ON e.employee_id = jh.employee_id;
9 BEGIN
10  OPEN c_employees;
11  FETCH c_employees INTO v_employee_id, v_first_name, v_end_date;
12  WHILE c_employees%FOUND LOOP
13    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_employee_id);
14    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_first_name);
15  END LOOP;
16  CLOSE c_employees;
17 END;
```

Results
Explain
Describe
Saved SQL
History

Employee ID: 2
Employee Name: Fleming
End Date: 09/21/1995

Employee ID: 1
Employee Name: Stephen
End Date: 03/15/1997

Statement processed.

 220701040@prajalakshmi.edu.in
 bhanu25
 en

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

Oracle APEX 23.2.4

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

RESULT:

PROCEDURES AND FUNCTIONS

EX_NO: 17

DATE:

1.Factorial of a number using function.

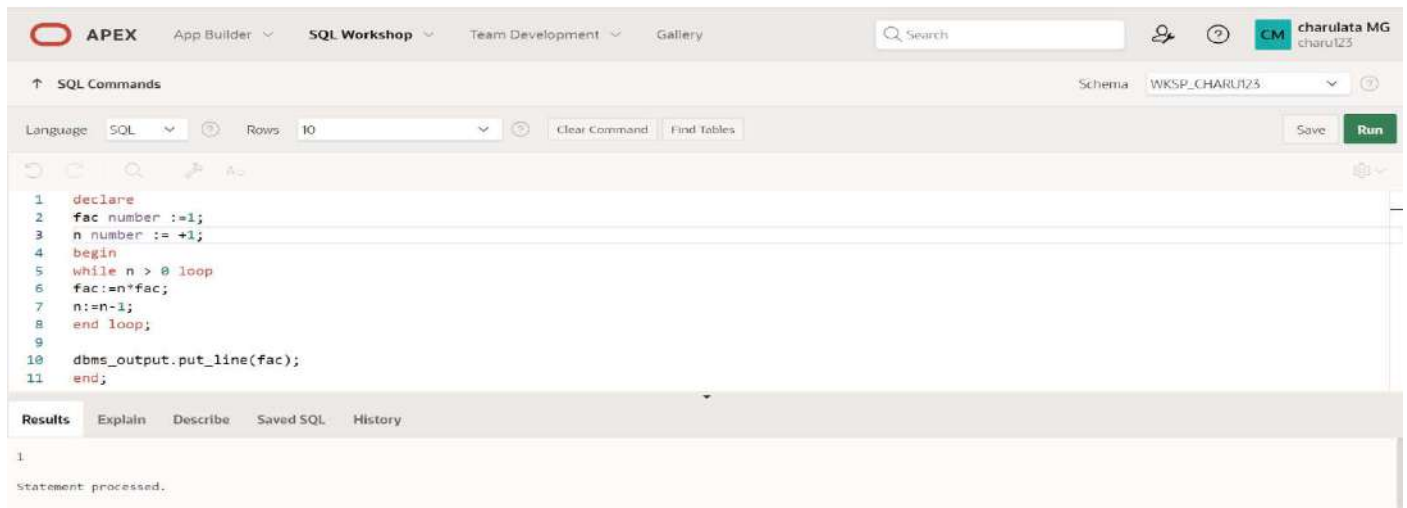
QUERY:

```
declare
fac number :=1;
n number := +1;
begin
while n > 0 loop
fac:=n*fac;
```

```
n:=n-1;  
end loop;
```

```
dbms_output.put_line(fac);  
end;
```

OUTPUT:



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);

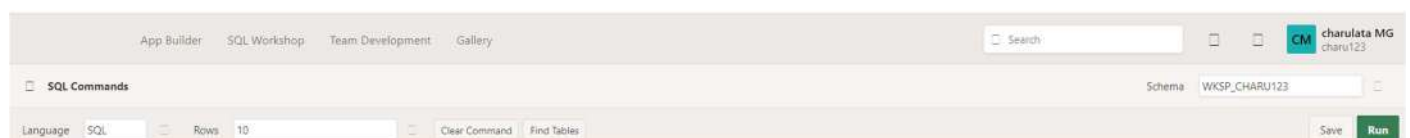
DBMS_OUTPUT.PUT_LINE('Title: ' || v_title);

DBMS_OUTPUT.PUT_LINE('Author: ' || v_author);

DBMS_OUTPUT.PUT_LINE('Year Published: ' || v_year_published);

END;

OUTPUT:



```

1 CREATE OR REPLACE PROCEDURE get_book_info (
2     p_book_id IN NUMBER,
3     p_title IN OUT VARCHAR2,
4     p_author OUT VARCHAR2,
5     p_year_published OUT NUMBER
6 )
7 AS
8 BEGIN
9     SELECT title, author, year_published INTO p_title, p_author, p_year_published
10    FROM books
11   WHERE book_id = p_book_id;
12
13     p_title := p_title || ' - Retrieved';
14 EXCEPTION
15     WHEN NO_DATA_FOUND THEN
16         p_title := NULL;
17         p_author := NULL;
18         p_year_published := NULL;
19 END;

```

Results Explain Describe Saved SQL History

Procedure created.

0.04 seconds

220701049@brajalakshmi.edu.in chandar en

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

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

RESULT:

TRIGGER

EX_NO: 18

DATE:

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

QUERY:

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

OUTPUT:

SQL Commands

Schema WKSP_CHARUT23

Language SQL

Rows 10

Clear Command

Find Tables

Save

Run

A:

```
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.');
```

Results Explain Describe Saved SQL History

Trigger created.

0.04 seconds

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

QUERY:

```
CREATE OR REPLACE TRIGGER check_duplicates
BEFORE INSERT OR UPDATE ON unique_values_table
FOR EACH ROW
DECLARE
duplicate_found EXCEPTION;
PRAGMA EXCEPTION_INIT(duplicate_found, -20002);
v_count NUMBER;
BEGIN
SELECT COUNT(*) INTO v_count FROM unique_values_table
WHERE unique_col = :NEW.unique_col AND id != :NEW.id;
IF v_count > 0 THEN
RAISE duplicate_found;
END IF;
EXCEPTION
WHEN duplicate_found THEN
RAISE_APPLICATION_ERROR(-20002, '&#39;Duplicate value found in unique_col.&#39;);
END;
```

OUTPUT:

APEX

App Builder

SQL Workshop

Team Development

Gallery

Search

CM

charulata MG

charu123

SQL Commands

Schema

WKSP_CHARUT23

Language

SQL

Rows

10

Clear Command

Find Tables

Save

Run

A::

4

DECLARE

5

duplicate_found EXCEPTION;

6

PRAGMA EXCEPTION_INIT(duplicate_found, -20002);

7

v_count NUMBER;

8

BEGIN

9

SELECT COUNT(*) INTO v_count FROM unique_values_table

10

WHERE unique_col = :NEW.unique_col AND id != :NEW.id;

11

IF v_count > 0 THEN

12

RAISE duplicate_found;

13

END IF;

14

EXCEPTION

15

WHEN duplicate_found THEN

16

RAISE_APPLICATION_ERROR(-20002, 'Duplicate value found in unique_col.');

17

END;

18

/

19

Results

Explain

Describe

Saved SQL

History

Trigger created.

0.04 seconds

220701043@rajalakshmi.edu.in

bharath

en

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

Oracle APEX 23.2.4

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

QUERY:

```
CREATE OR REPLACE TRIGGER check_threshold
BEFORE INSERT OR UPDATE ON threshold_table
FOR EACH ROW

DECLARE

threshold_exceeded EXCEPTION;

PRAGMA EXCEPTION_INIT(threshold_exceeded, -20003);

v_sum NUMBER;

v_threshold NUMBER := 10000; -- Set your threshold here

BEGIN

SELECT SUM(value_col) INTO v_sum FROM threshold_table;

v_sum := v_sum + :NEW.value_col;

IF v_sum > v_threshold THEN

RAISE threshold_exceeded;

END IF;

EXCEPTION

WHEN threshold_exceeded THEN

RAISE_APPLICATION_ERROR(-20003, 'Threshold exceeded for value_col.');
```

END;

OUTPUT:

APEX

App Builder

SQL Workshop

Team Development

Gallery

Search

CM

charulata MG

charu123

SQL Commands

Schema

WKSP_CHARU123

Language

SQL

Rows

10

Clear Command

Find Tables

Save

Run

A-Z

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.');

Results

Explain

Describe

Saved SQL

History

Trigger created.

0.04 seconds

20701045@prajalakshmi.edu.in

bharath

en

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

Oracle APEX 23.2.4

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

QUERY:

```
CREATE OR REPLACE TRIGGER log_changes
```

```
AFTER UPDATE ON main_table
```

```
FOR EACH ROW
```

```
BEGIN
```

```
INSERT INTO audit_table (audit_id, changed_id, old_col1, new_col1, old_col2, new_col2,  
change_time)
```

```
VALUES (audit_seq.NEXTVAL, :OLD.id, :OLD.col1, :NEW.col1, :OLD.col2, :NEW.col2,
```

```
SYSTIMESTAMP);
```

```
END;
```

OUTPUT:

The screenshot displays the Oracle APEX SQL Workshop interface. At the top, the navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'charulata MG' are also visible. The 'SQL Commands' section shows the schema 'WKSP_CHARU123'. Below this, the 'Language' is set to 'SQL' and 'Rows' to '10'. The main editor 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, new_col2, change_time)
6     VALUES (audit_seq.NEXTVAL, :OLD.id, :OLD.col1, :NEW.col1, :OLD.col2, :NEW.col2, SYSTIMESTAMP);
7 END;
8 /
9
```

The 'Results' tab at the bottom shows the output: 'Trigger created.' and '0.05 seconds'.


Footer information includes the email '220701043@rajalakshmi.edu.in', the name 'bharath', the language 'en', 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, '&#39;INSERT&#39;, '&#39;activity_table&#39;, :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, '&#39;UPDATE&#39;, '&#39;activity_table&#39;, :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, '&#39;DELETE&#39;, '&#39;activity_table&#39;, :OLD.id,
        SYSTIMESTAMP);
    END IF;
END;
```


OUTPUT:

 **APEX** App Builder ▾ SQL Workshop ▾ Team Development ▾ Gallery






Search

charulata MG
charu123

SQL Commands

Schema WKSP_CHARU123

Language SQL Rows 10 Clear Command Find Tables Save Run

    A: 

```
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);
```

Results Explain Describe Saved SQL History

Trigger created.

0.03 seconds

220701043@rajalakshmi.edu.in bharath en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

QUERY:

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

OUTPUT:

The screenshot displays the Oracle APEX SQL Workshop interface. At the top, the navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'charulata MG charu123' are also visible. The 'SQL Commands' section shows the 'Schema' set to 'WKSP_CHARUT23'. Below this, the 'Language' is set to 'SQL' and 'Rows' to '10'. The main editor contains the following PL/SQL code:

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

The 'Results' tab at the bottom shows the message 'Trigger created.' and the execution time '0.03 seconds'. The footer includes the email '220701043@rajalakshmi.edu.in', the name 'bharath', and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4'.

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

QUERY:

```
CREATE OR REPLACE TRIGGER validate_order

BEFORE INSERT ON orders

FOR EACH ROW

DECLARE

v_stock NUMBER;

insufficient_stock EXCEPTION;

PRAGMA EXCEPTION_INIT(insufficient_stock, -20004);

BEGIN

SELECT stock_quantity INTO v_stock FROM items WHERE item_id = :NEW.item_id;

IF v_stock < :NEW.order_quantity THEN

RAISE insufficient_stock;

END IF;

UPDATE items SET stock_quantity = stock_quantity - :NEW.order_quantity WHERE

item_id = :NEW.item_id;

EXCEPTION

WHEN insufficient_stock THEN

RAISE_APPLICATION_ERROR(-20004, 'Insufficient stock for the item.');
```

END;

OUTPUT:

APEX

App Builder

SQL Workshop

Team Development

Gallery

Search

CM

charulata MG

charu123

SQL Commands

SchemaWKSP_CHARU123

LanguageSQLRows10Clear CommandFind TablesSaveRun

Az

```
1 CREATE OR REPLACE TRIGGER validate_order
2 BEFORE INSERT ON orders
3 FOR EACH ROW
4 DECLARE
5     v_stock NUMBER;
6     insufficient_stock EXCEPTION;
7     PRAGMA EXCEPTION_INIT(insufficient_stock, -20004);
8 BEGIN
9     SELECT stock_quantity INTO v_stock FROM items WHERE item_id = :NEW.item_id;
10    IF v_stock < :NEW.order_quantity THEN
11        RAISE insufficient_stock;
12    END IF;
13    UPDATE items SET stock_quantity = stock_quantity - :NEW.order_quantity WHERE item_id = :NEW.item_id;
14 EXCEPTION
15     WHEN insufficient_stock THEN
```

Results

Explain

Describe

Saved SQL

History

Trigger created.

0.05 seconds

220701043@rajalakshmi.edu.in

bhasath

en

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

Oracle APEX 25.2.4

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

RESULT:

MONGO DB

EX_NO: 19

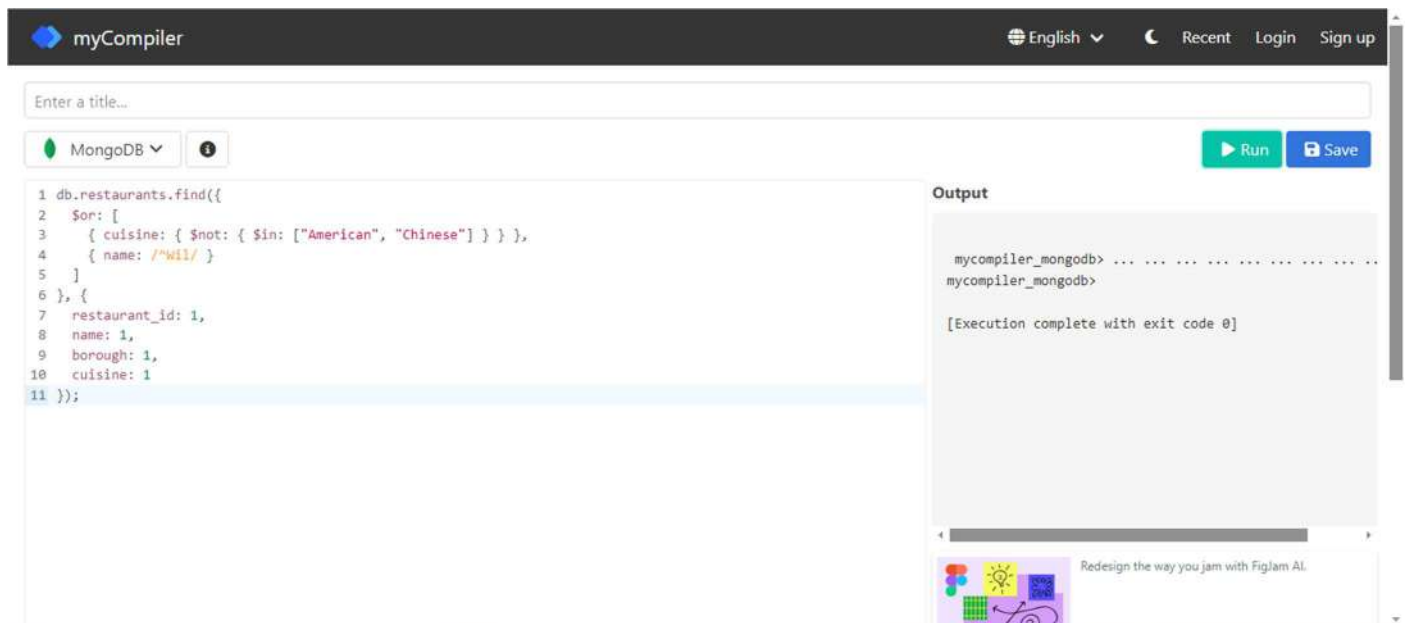
DATE:

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

QUERY:

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

OUTPUT:

The screenshot shows the myCompiler web interface. At the top, there's a navigation bar with 'myCompiler', 'English', 'Recent', 'Login', and 'Sign up'. Below this is a search bar 'Enter a title...'. A dropdown menu shows 'MongoDB' selected. To the right are 'Run' and 'Save' buttons. The main area contains a code editor with the following MongoDB query:

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

The 'Output' panel on the right shows the command prompt:

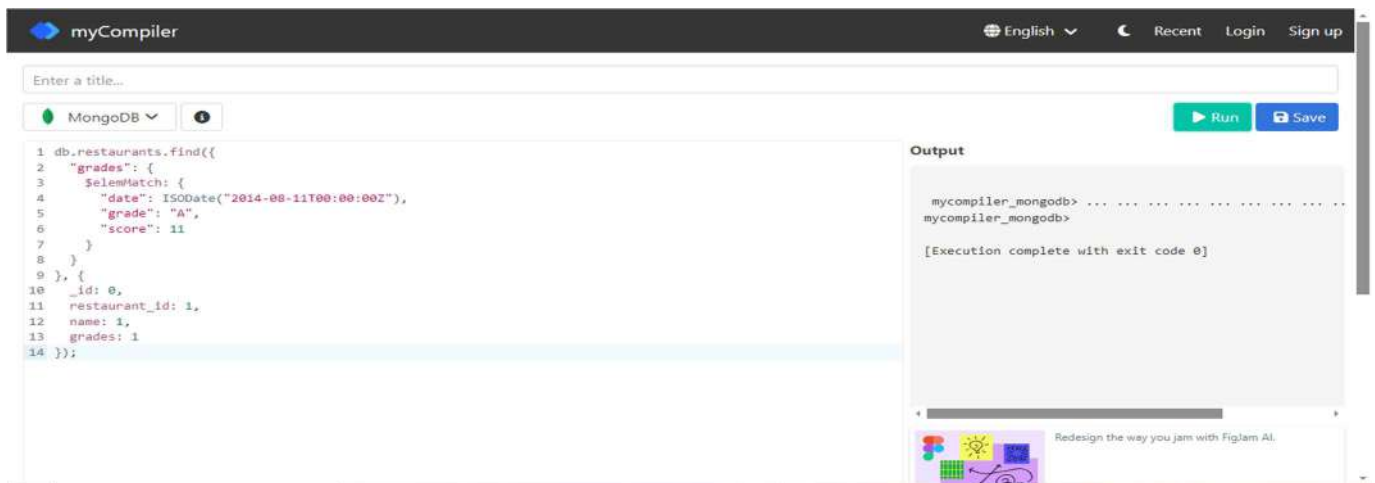
```
mycompiler_mongodb> ...  
mycompiler_mongodb>  
  
[Execution complete with exit code 0]
```

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

QUERY:

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

OUTPUT:

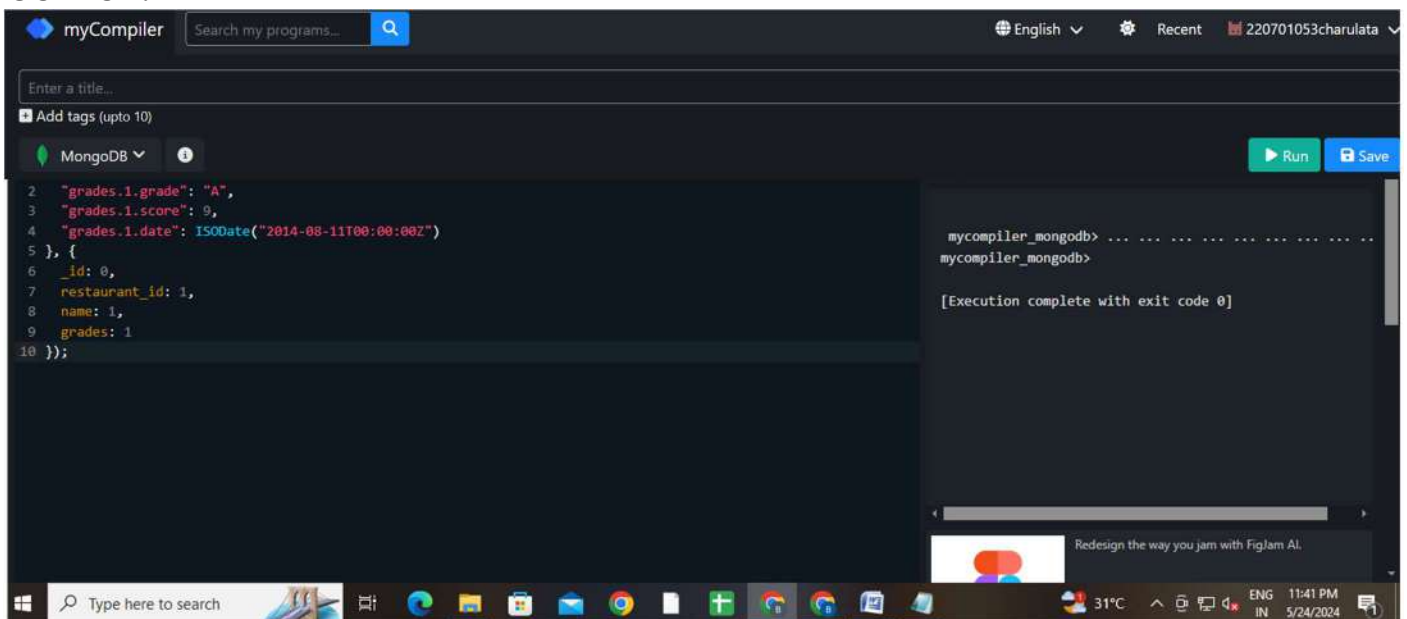


3.) Write a MongoDB query to find the restaurant Id, name and grades for those restaurants where the 2nd element of grades array contains a grade of "A" and score 9 on an ISODate "2014-08-11T00:00:00Z".

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:

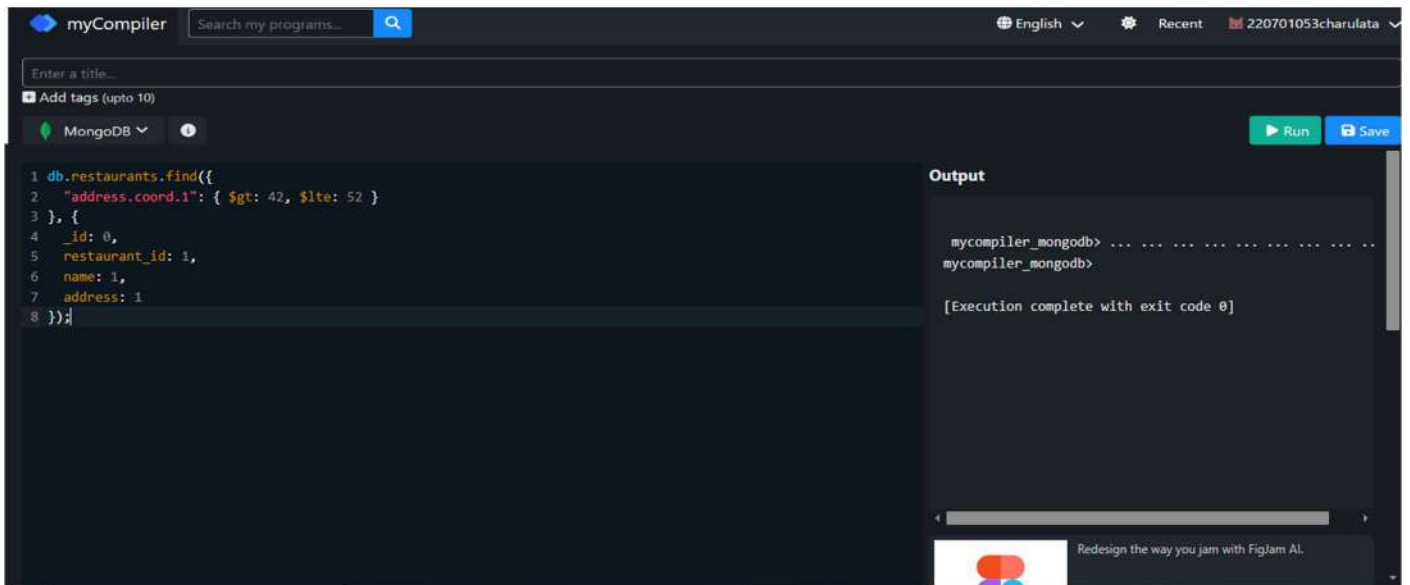


4.) Write a MongoDB query to find the restaurant Id, name, address and geographical location for those restaurants where 2nd element of coord array contains a value which is more than 42 and upto 52

QUERY:

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

OUTPUT:



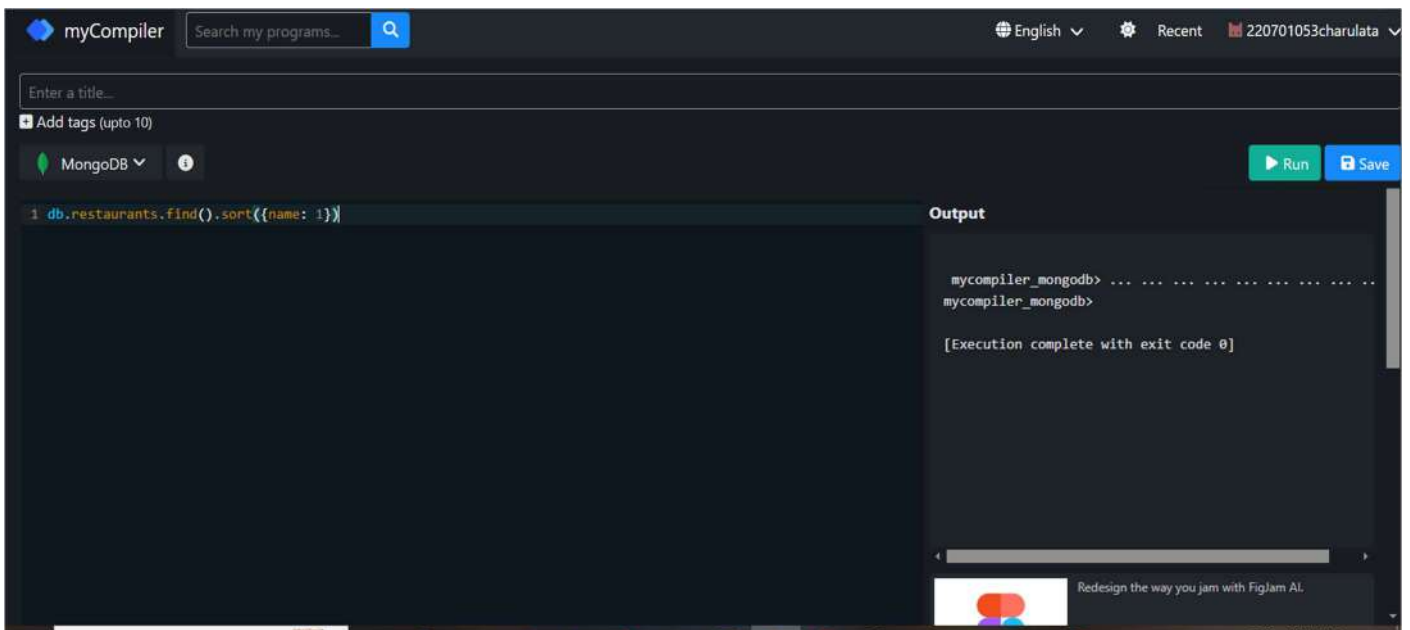
The screenshot shows the myCompiler MongoDB interface. The code editor on the left contains a MongoDB query: `1 db.restaurants.find({`
`2 "address.coord.1": { $gt: 42, $lte: 52 }`
`3 }, {`
`4 _id: 0,`
`5 restaurant_id: 1,`
`6 name: 1,`
`7 address: 1`
`8 })`. The right panel, titled "Output", shows the command prompt `mycompiler_mongodb>` followed by `mycompiler_mongodb>` and the message `[Execution complete with exit code 0]`. The interface includes a search bar, a language dropdown set to "English", and a user profile icon.

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

QUERY:

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

OUTPUT:



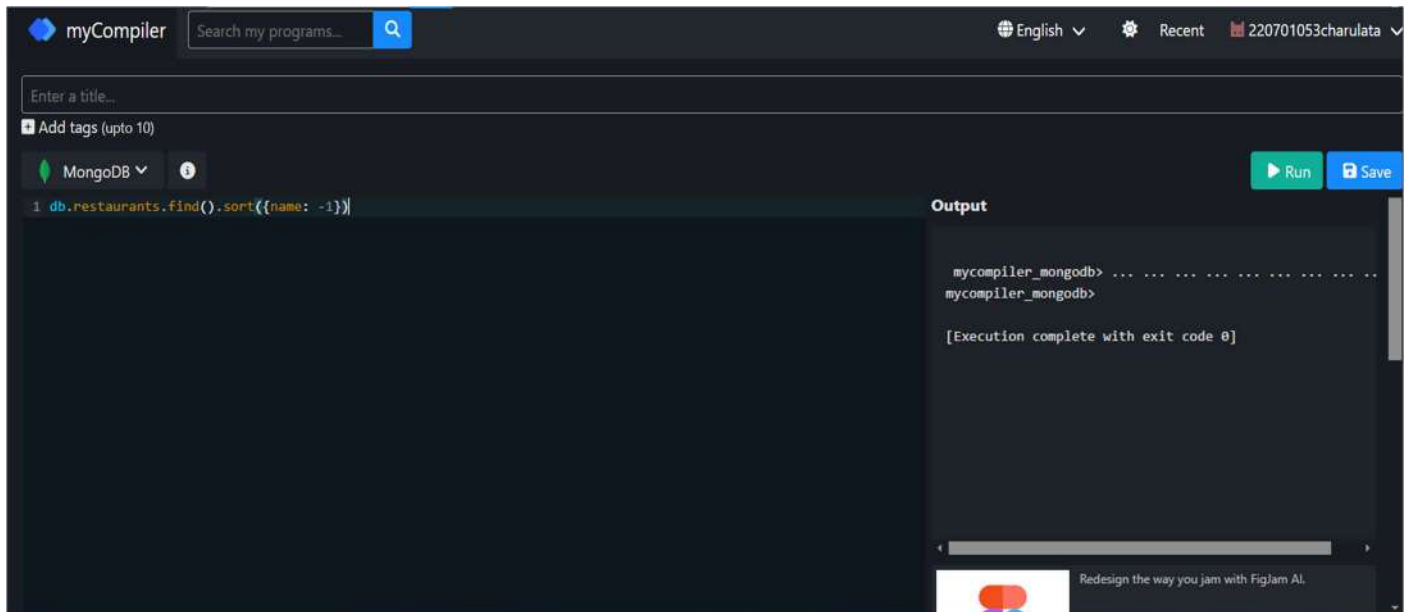
The screenshot shows the myCompiler MongoDB interface. The code editor on the left contains a MongoDB query: `1 db.restaurants.find().sort({name: 1})`. The right panel, titled "Output", shows the command prompt `mycompiler_mongodb>` followed by `mycompiler_mongodb>` and the message `[Execution complete with exit code 0]`. The interface includes a search bar, a language dropdown set to "English", and a user profile icon.

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:

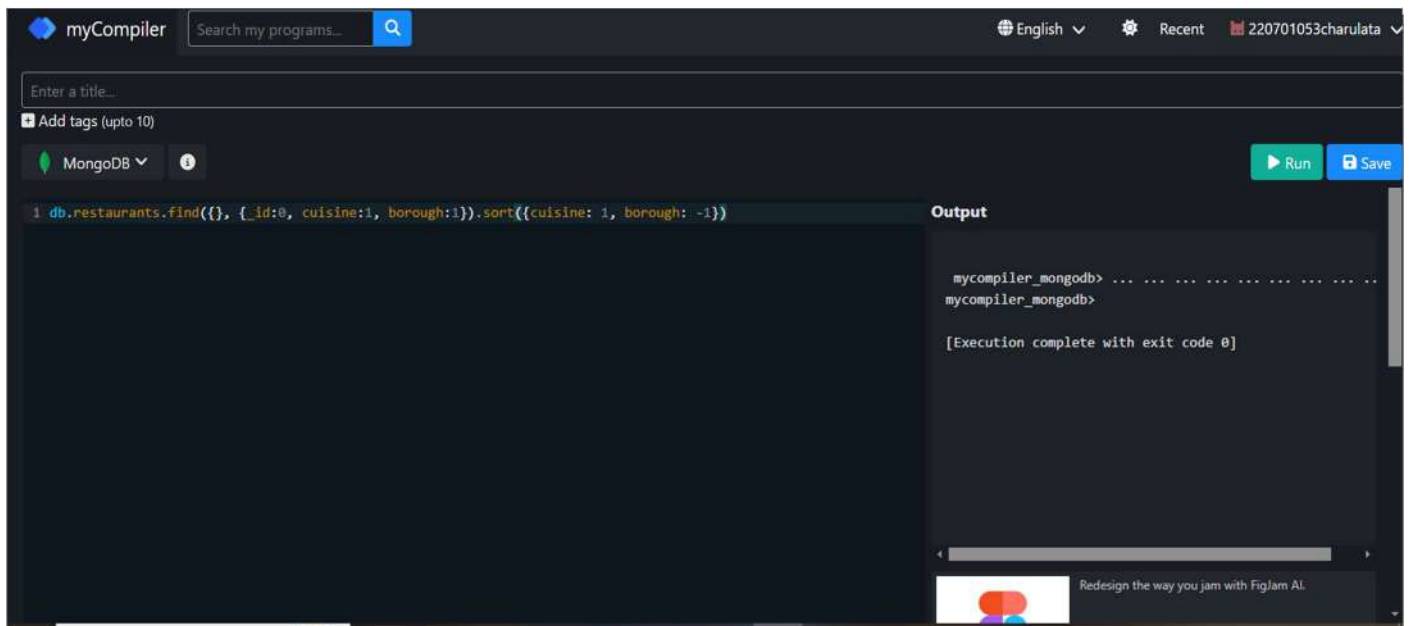


7.)Write a MongoDB query to arranged the name of the cuisine in ascending order and for that same cuisine borough should be in descending order.

QUERY:

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

OUTPUT:



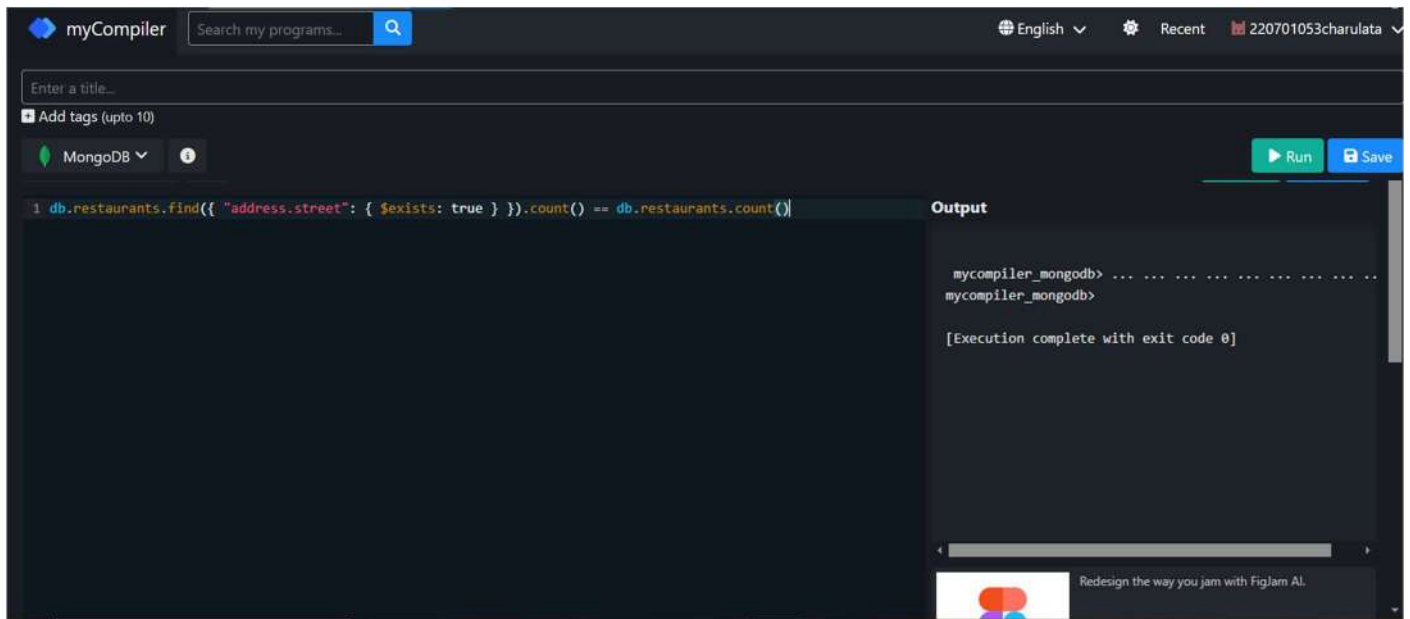
The screenshot shows the myCompiler web interface. The top bar includes the myCompiler logo, a search bar, and user information. The main editor area has a title field, a tag field, and a language dropdown set to MongoDB. The code editor contains the query: `1 db.restaurants.find({}, { _id:0, cuisine:1, borough:1}).sort({cuisine: 1, borough: -1})`. To the right of the editor is an 'Output' console. The output shows the command prompt `mycompiler_mongodb>` followed by a series of dots, then another `mycompiler_mongodb>` prompt, and finally the message `[Execution complete with exit code 0]`. At the bottom right, there is a small advertisement for FigJam AI.

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

QUERY:

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

OUTPUT:



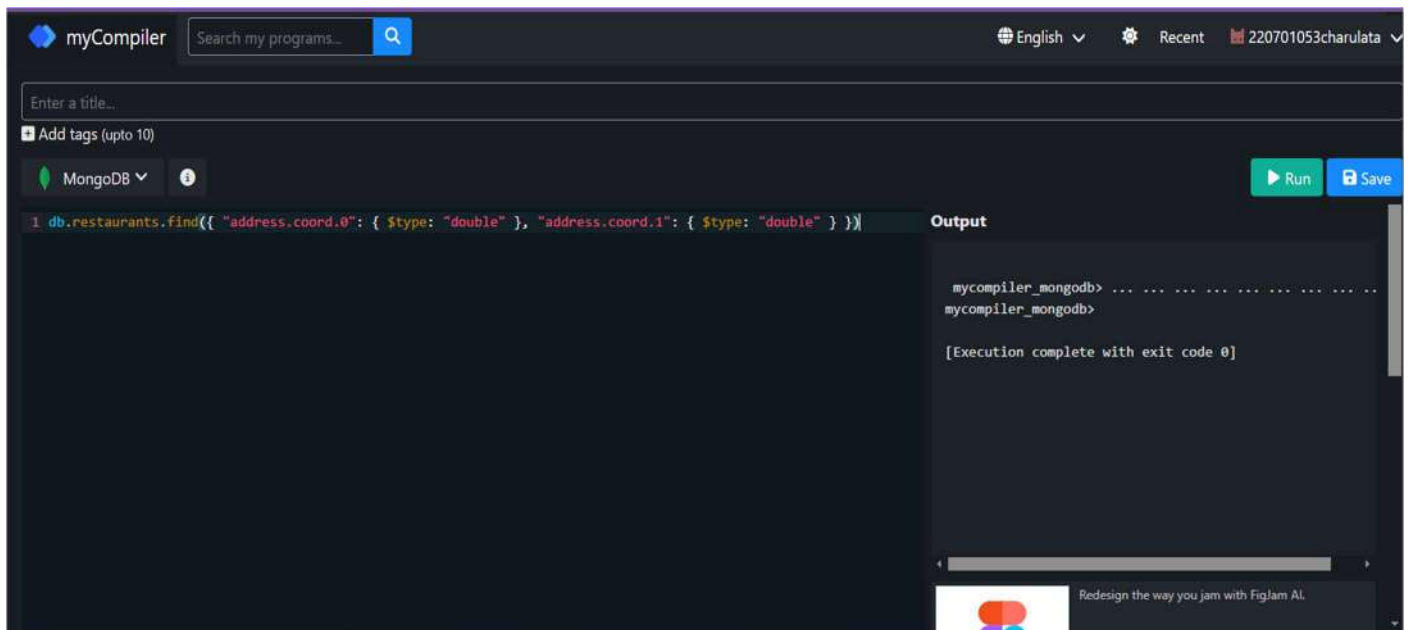
The screenshot shows the myCompiler web interface. At the top, there's a search bar and a language dropdown set to English. Below that, a title field and a tag management section are visible. The main editor area is titled 'MongoDB' and contains a single line of code: `1 db.restaurants.find({ "address.street": { $exists: true } }).count() == db.restaurants.count()`. To the right of the code editor is an 'Output' pane. It shows the command prompt `mycompiler_mongodb>` followed by another `mycompiler_mongodb>` line, and then the message `[Execution complete with exit code 0]`. At the bottom right, there's a small 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:



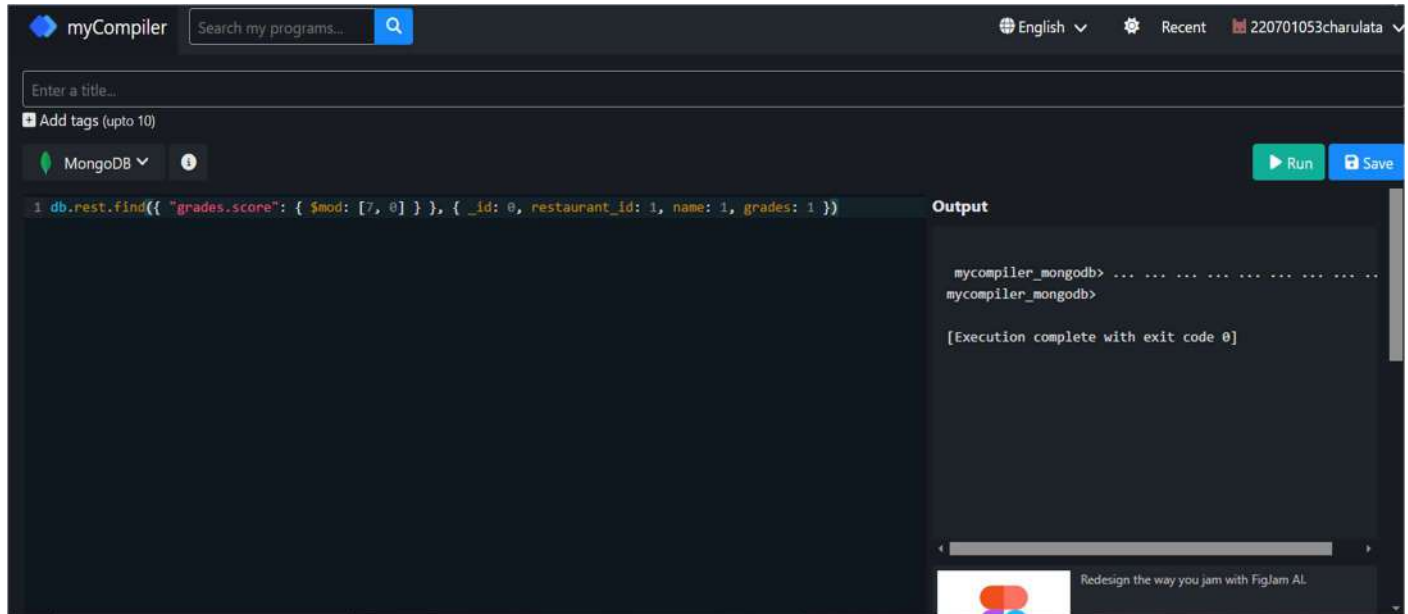
The screenshot shows the myCompiler web interface. At the top, there's a search bar and user information. Below, a code editor displays a MongoDB query: `1 db.restaurants.find({ "address.coord.0": { $type: "double" }, "address.coord.1": { $type: "double" } })`. To the right of the editor are 'Run' and 'Save' buttons. The 'Output' pane on the right shows the command prompt output: `mycompiler_mongodb> ... mycompiler_mongodb>` followed by `[Execution complete with exit code 0]`. At the bottom, there's a small 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:



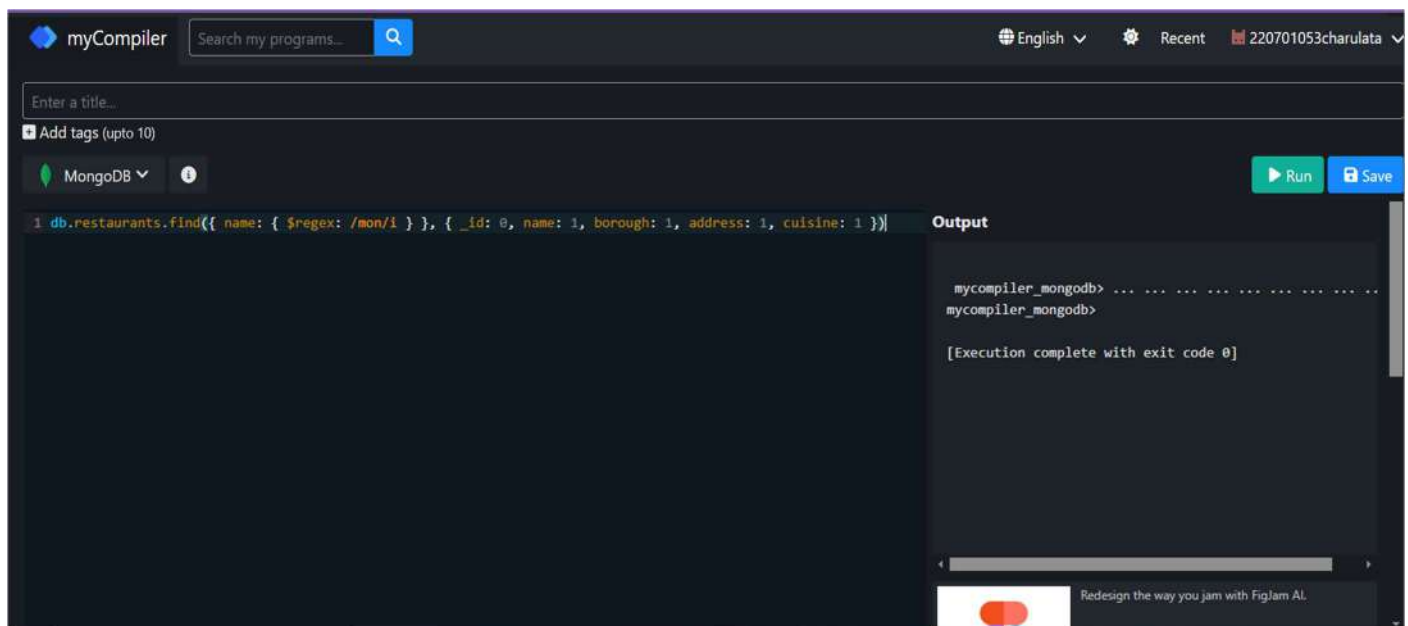
The screenshot shows the myCompiler interface with the MongoDB tab selected. The query entered is `1 db.rest.find({ "grades.score": { $mod: [7, 0] } }, { _id: 0, restaurant_id: 1, name: 1, grades: 1 })`. The output panel shows the command prompt `mycompiler_mongodb>` and the message `[Execution complete with exit code 0]`.

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

QUERY:

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

OUTPUT:



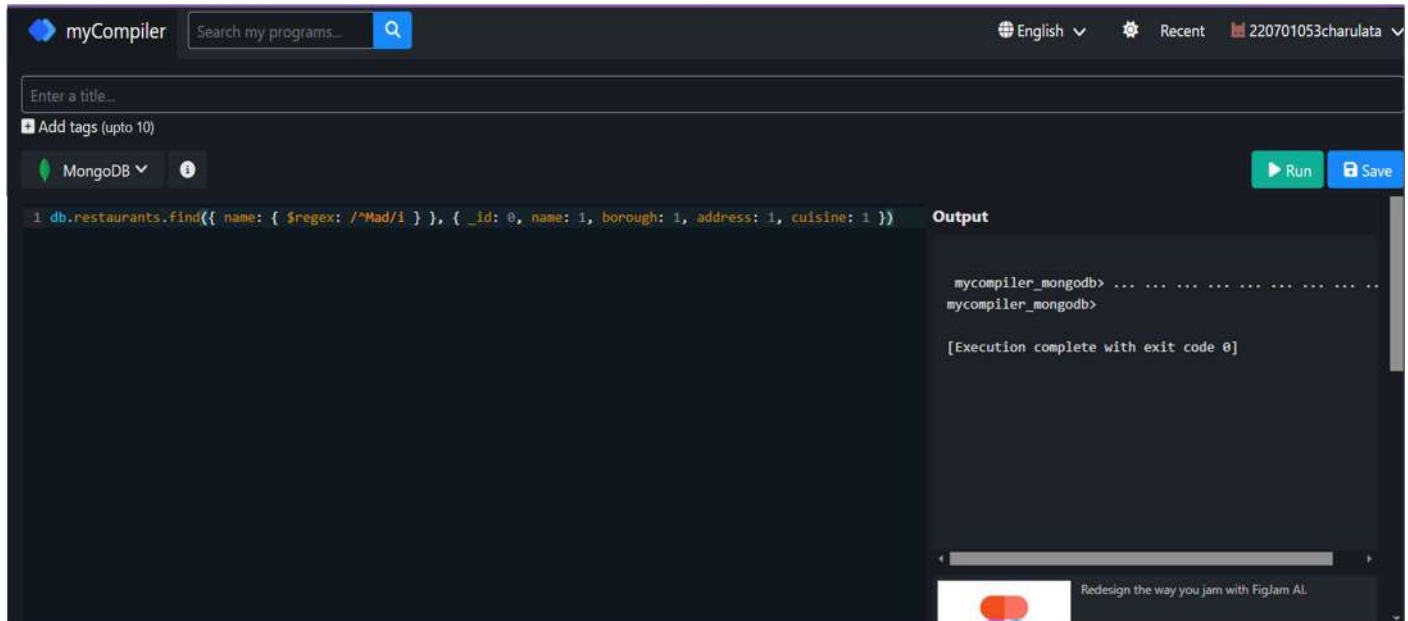
The screenshot shows the myCompiler interface with the MongoDB tab selected. The query entered is `1 db.restaurants.find({ name: { $regex: /mon/i } }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 })`. The output panel shows the command prompt `mycompiler_mongodb>` and the message `[Execution complete with exit code 0]`.

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

QUERY:

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

OUTPUT:

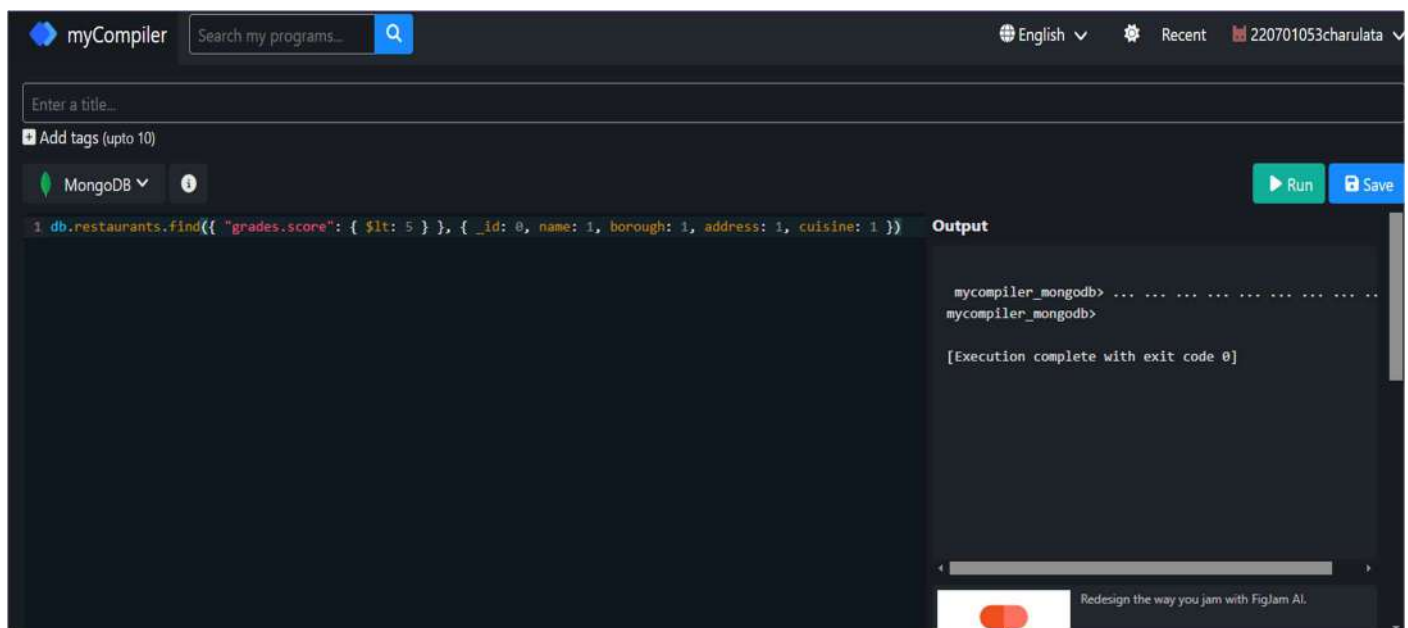
A screenshot of the myCompiler web application interface. The top bar includes the 'myCompiler' logo, a search bar, and user information. The main area shows a MongoDB query editor with the query: `1 db.restaurants.find({ name: { $regex: /^Mad/i } }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 })`. To the right of the editor are 'Run' and 'Save' buttons. The 'Output' pane on the right shows the command prompt output: `mycompiler_mongodb> ... mycompiler_mongodb> [Execution complete with exit code 0]`. At the bottom right, there is a small advertisement for FigJam AI.

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:

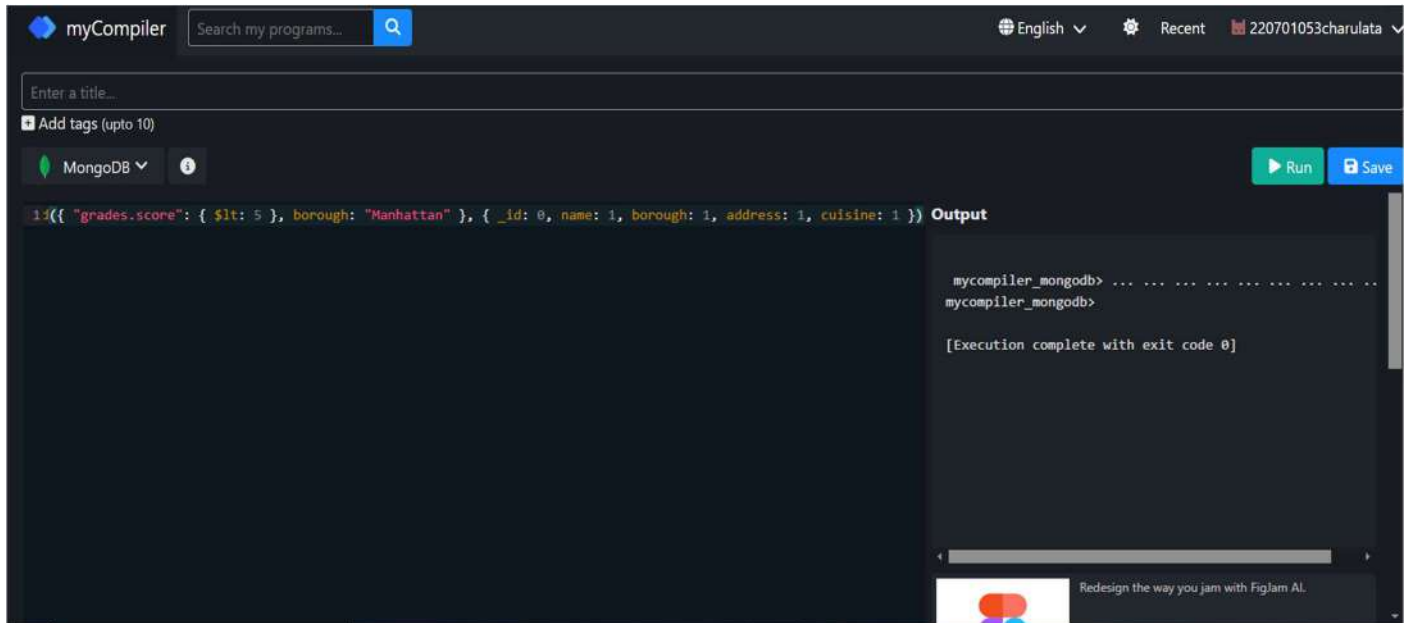
A screenshot of the myCompiler web application interface, similar to the one above. The MongoDB query editor contains the query: `1 db.restaurants.find({ "grades.score": { $lt: 5 } }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 })`. The 'Output' pane shows the same command prompt output: `mycompiler_mongodb> ... mycompiler_mongodb> [Execution complete with exit code 0]`. The interface elements and bottom advertisement are consistent with the previous screenshot.

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:

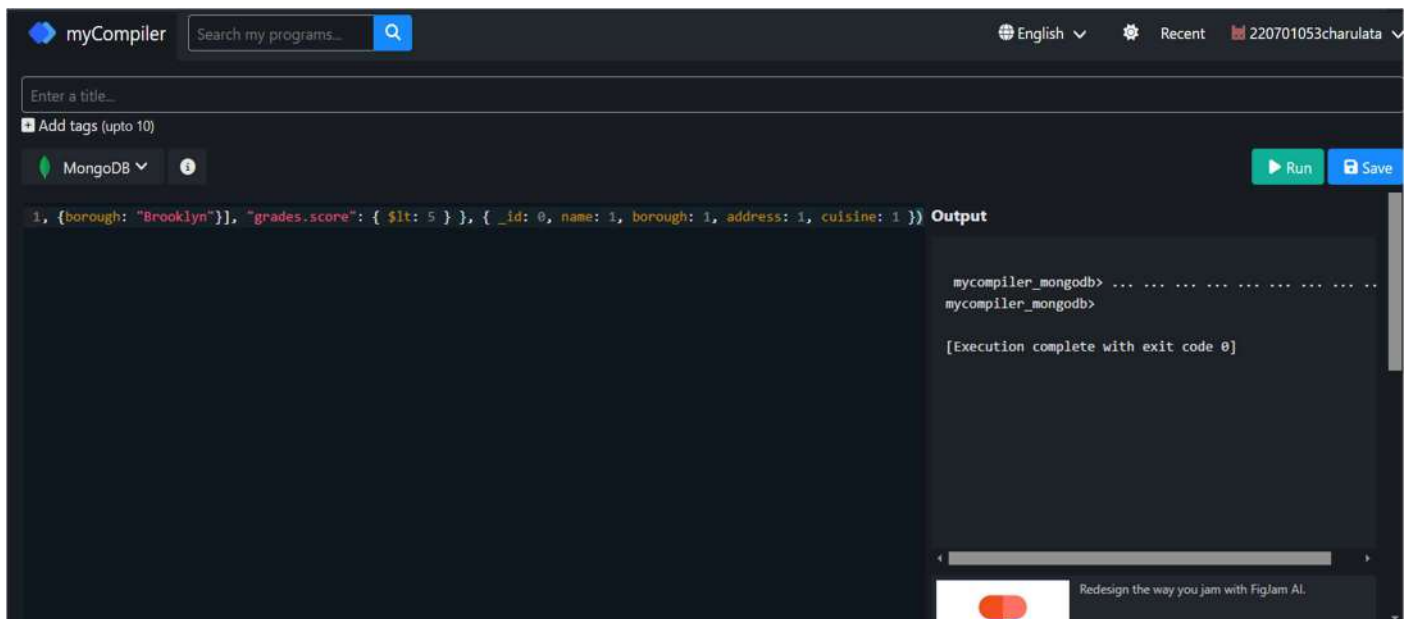


15. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn.

QUERY:

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

OUTPUT:

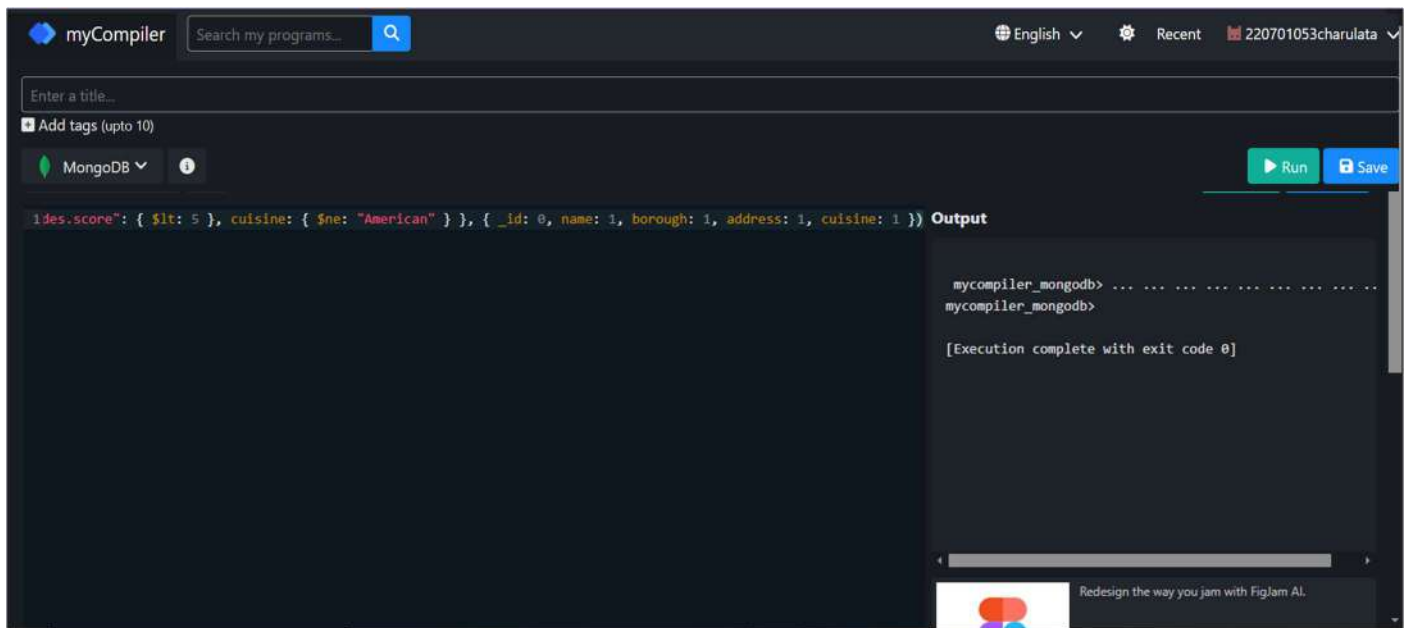


16. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn, and their cuisine is not American.

QUERY:

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

OUTPUT:

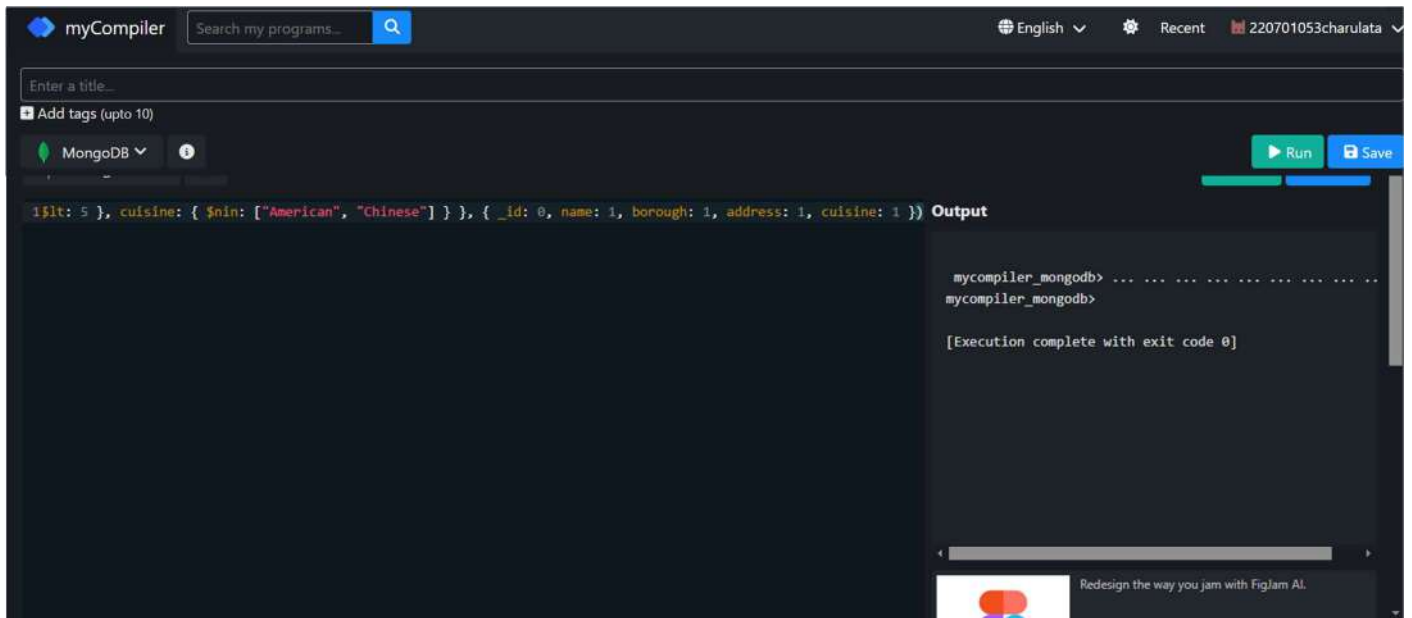


17. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn, and their cuisine is not American or Chinese.

QUERY:

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

OUTPUT:

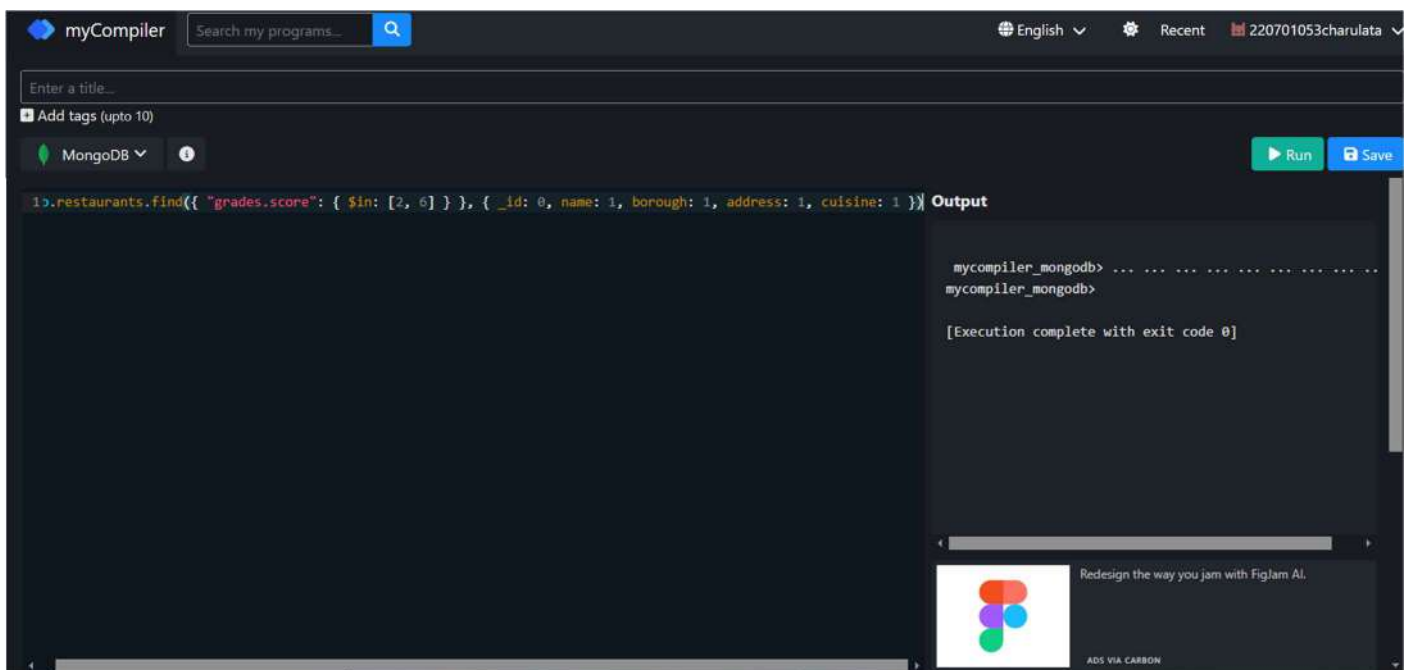


18. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6.

QUERY:

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

OUTPUT:

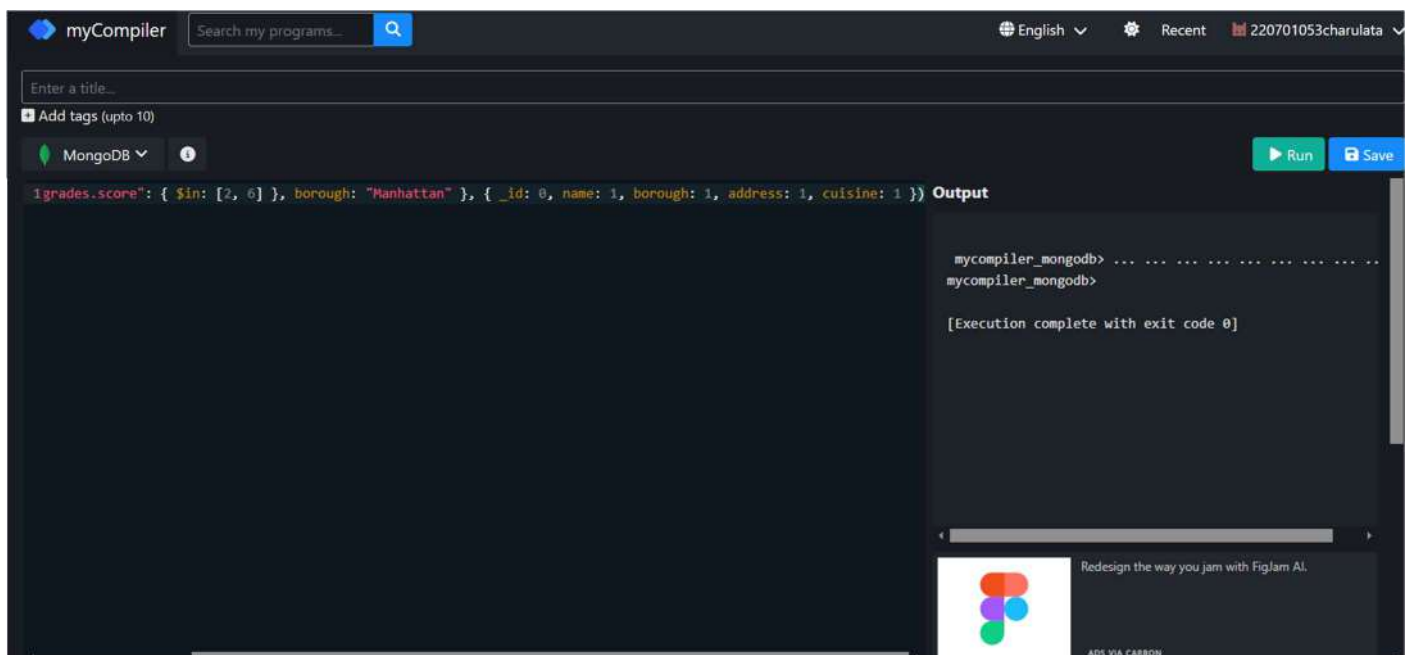


19. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan.

QUERY:

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

OUTPUT:

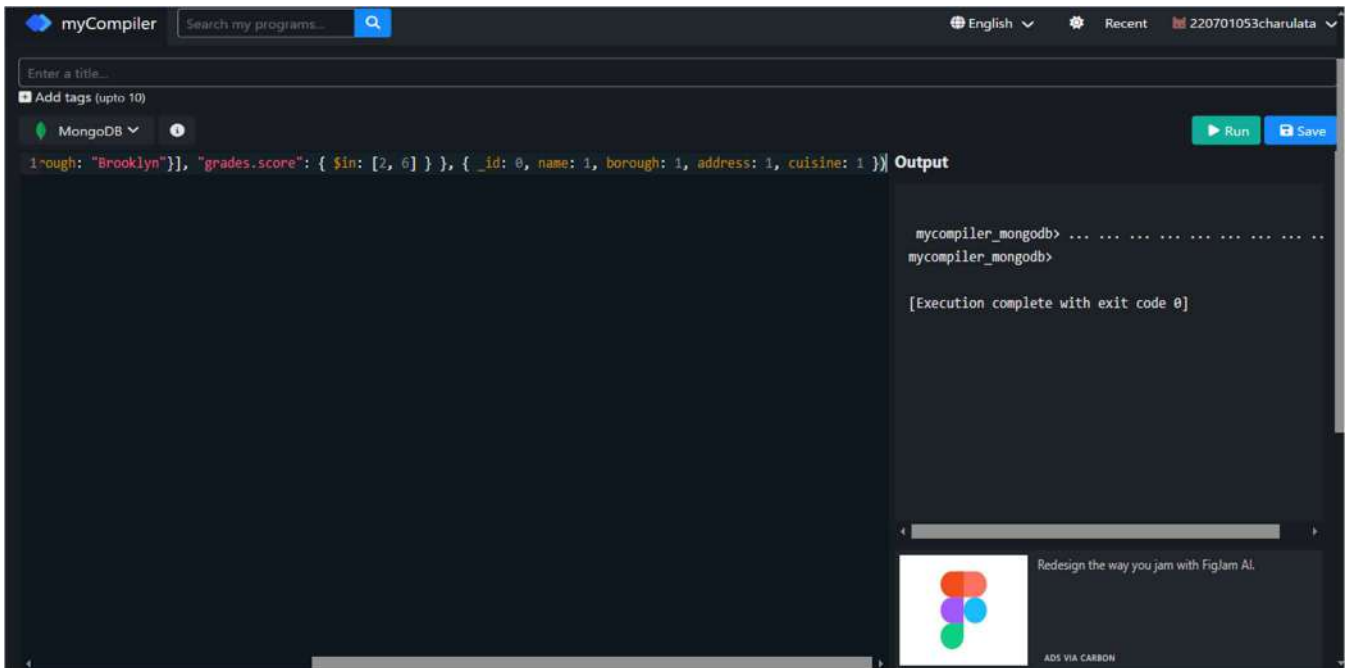


20. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn.

QUERY:

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

OUTPUT:



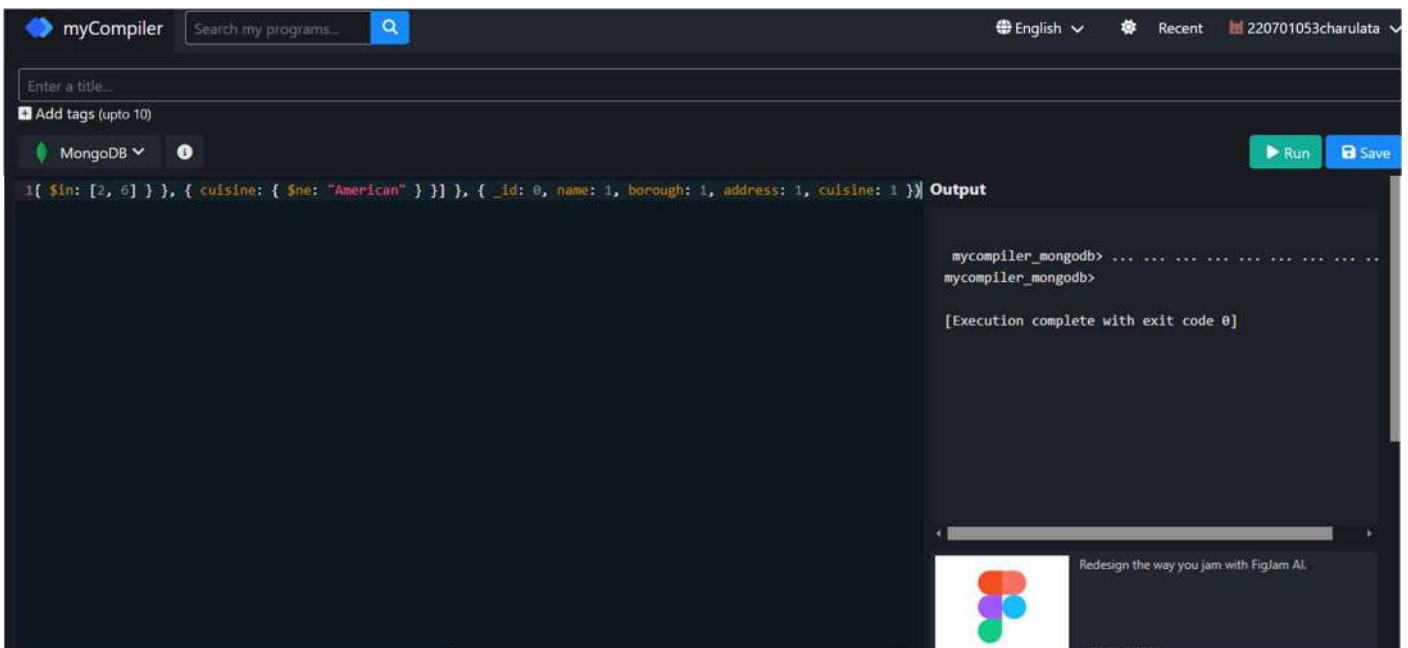
The screenshot shows the myCompiler MongoDB interface. The top bar includes the myCompiler logo, a search bar, and user information. The main area has a title input, a tag input, and a MongoDB dropdown. The code editor contains a MongoDB query: `1{borough: "Brooklyn"}], "grades.score": { $in: [2, 6] } }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 }}`. The output panel shows the command `mycompiler_mongodb>` and the message `[Execution complete with exit code 0]`. A small advertisement for FigJam AI is visible at the bottom right.

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:



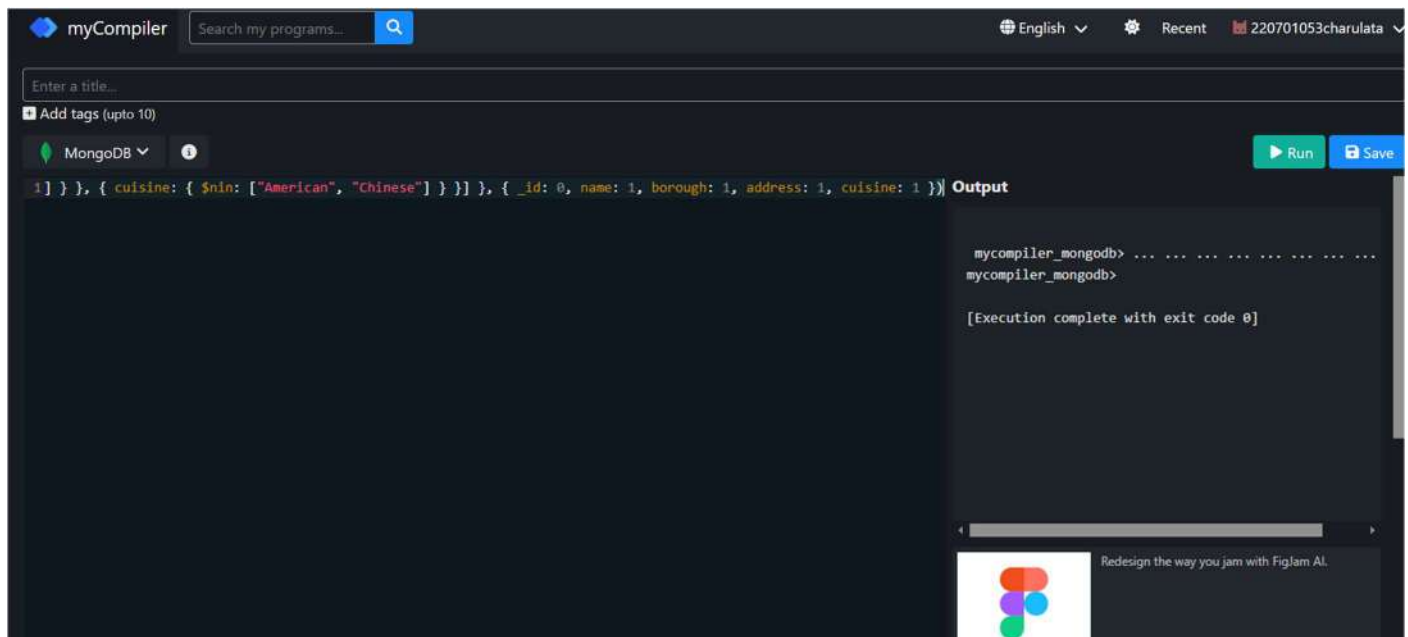
The screenshot shows the myCompiler MongoDB interface. The top bar includes the myCompiler logo, a search bar, and user information. The main area has a title input, a tag input, and a MongoDB dropdown. The code editor contains a MongoDB query: `1{ $in: [2, 6] } }, { cuisine: { $ne: "American" } }] }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 }}`. The output panel shows the command `mycompiler_mongodb>` and the message `[Execution complete with exit code 0]`. A small advertisement for FigJam AI is visible at the bottom right.

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:

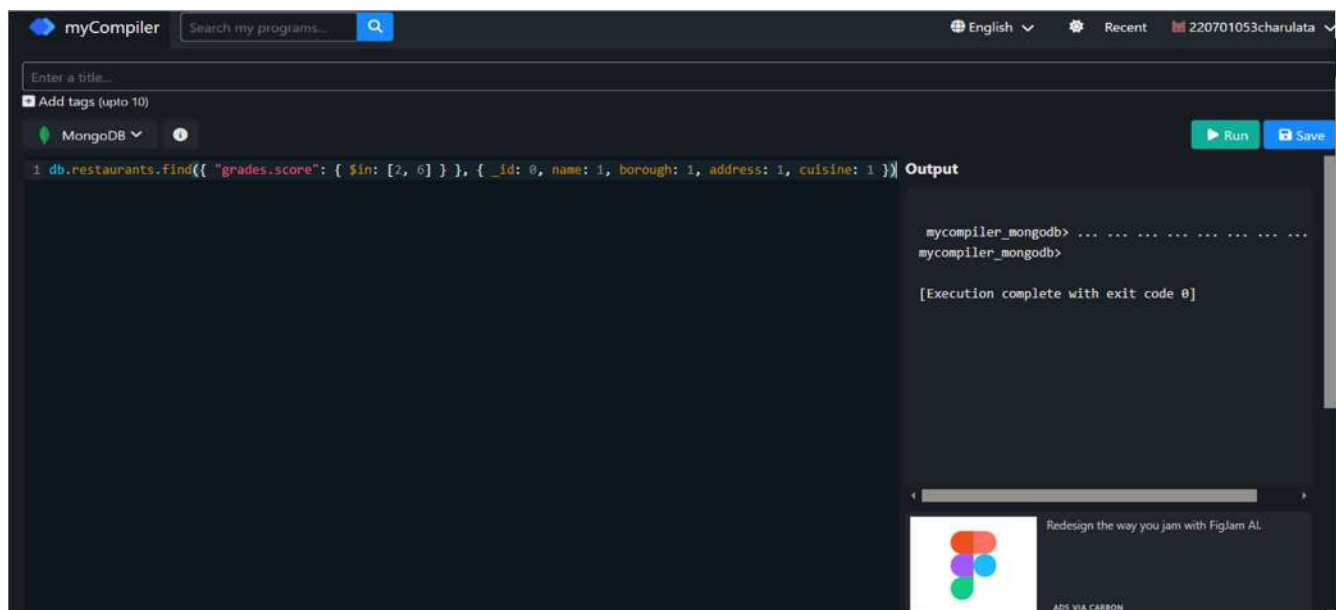


23. Write a MongoDB query to find the restaurants that have a grade with a score of 2 or a grade with a score of 6.

QUERY:

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

OUTPUT:



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

RESULT:

MONGO DB

EX_NO: 20

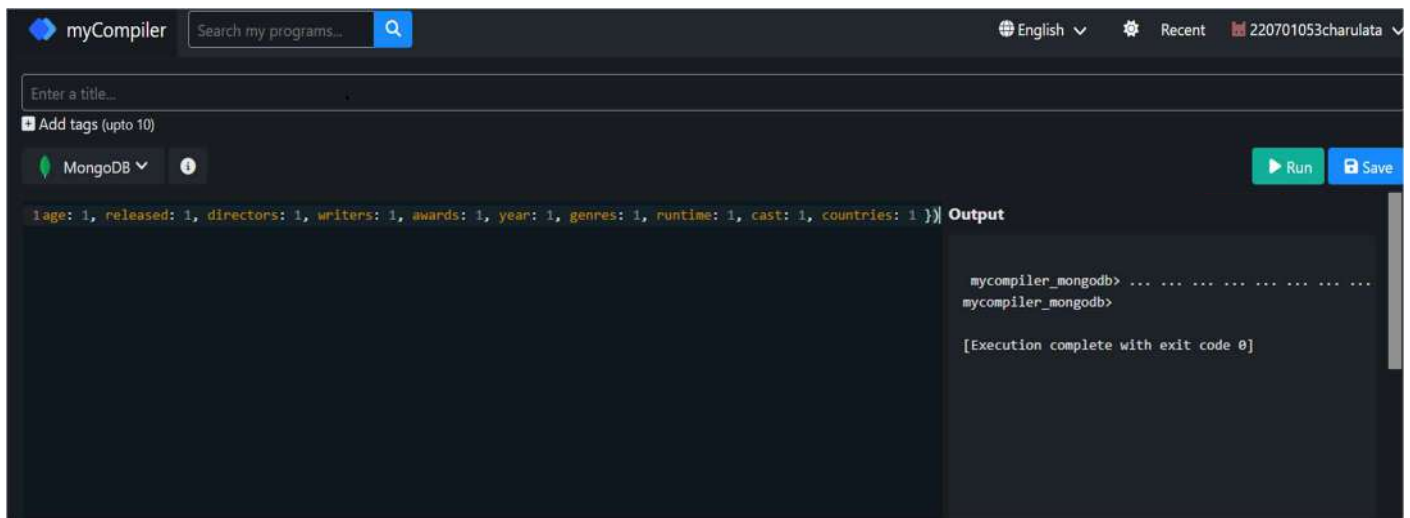
DATE:

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

QUERY:

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

OUTPUT:



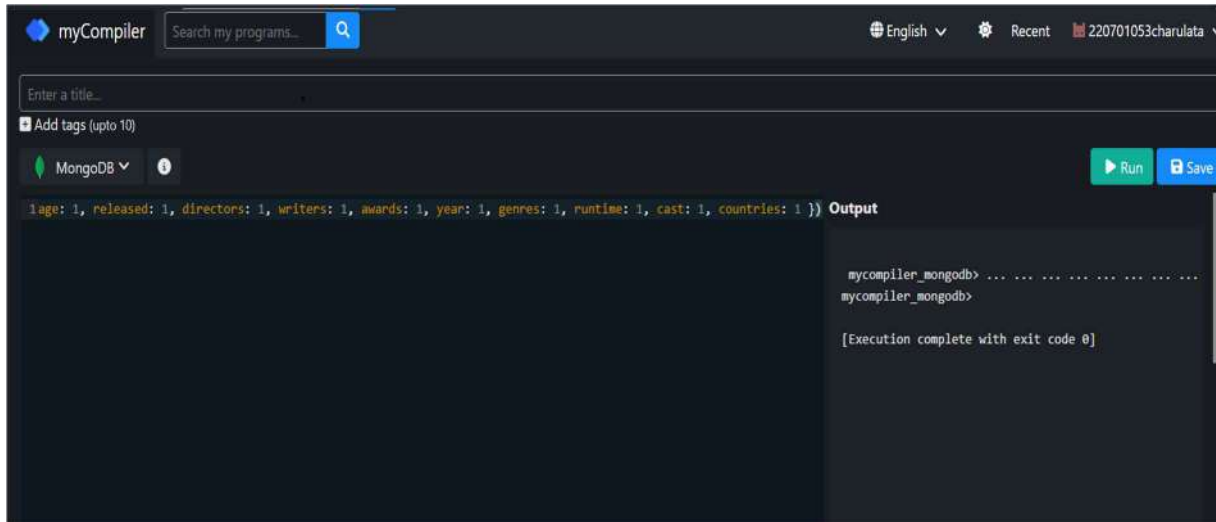
The screenshot shows the myCompiler web interface. At the top, there's a search bar and a language dropdown set to English. Below that, a text input field contains the query: `age: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 } }`. To the right of the input are 'Run' and 'Save' buttons. Below the input, the output is displayed in a terminal-like window. The output shows the prompt `mycompiler_mongodb>` followed by a blank line and then `[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:

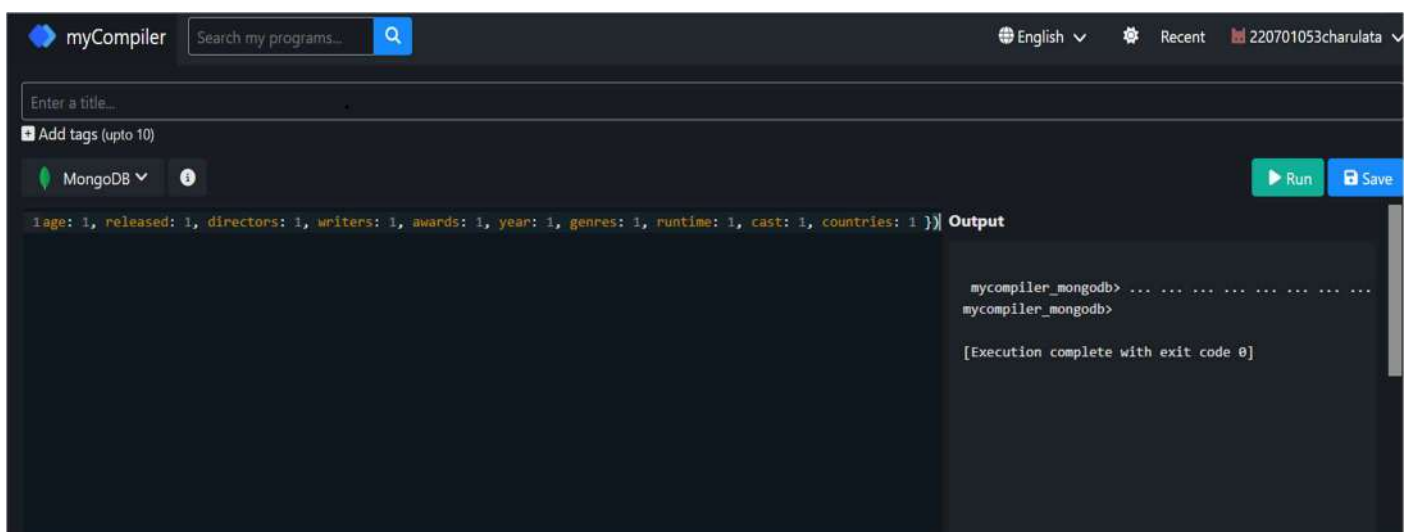


3.) Find all movies with full information from the 'movies' collection that have "Short" genre.

QUERY:

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

OUTPUT:

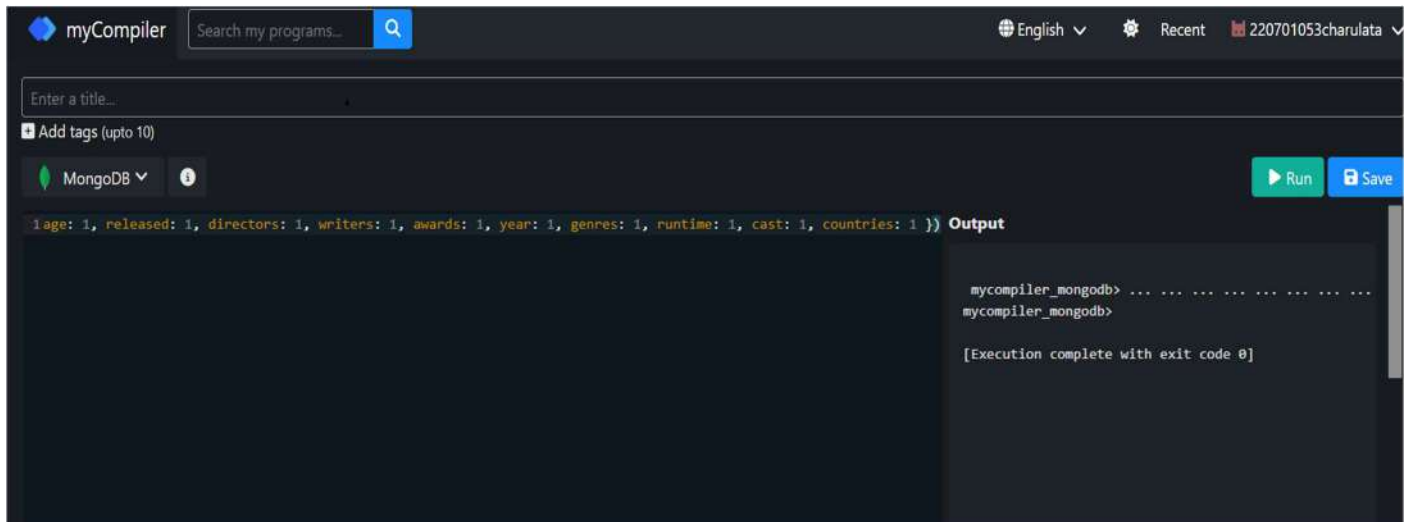


4.)Retrieve all movies from the 'movies' collection that were directed by "William K.L. Dickson" and include complete information for each movie.

QUERY:

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

OUTPUT:



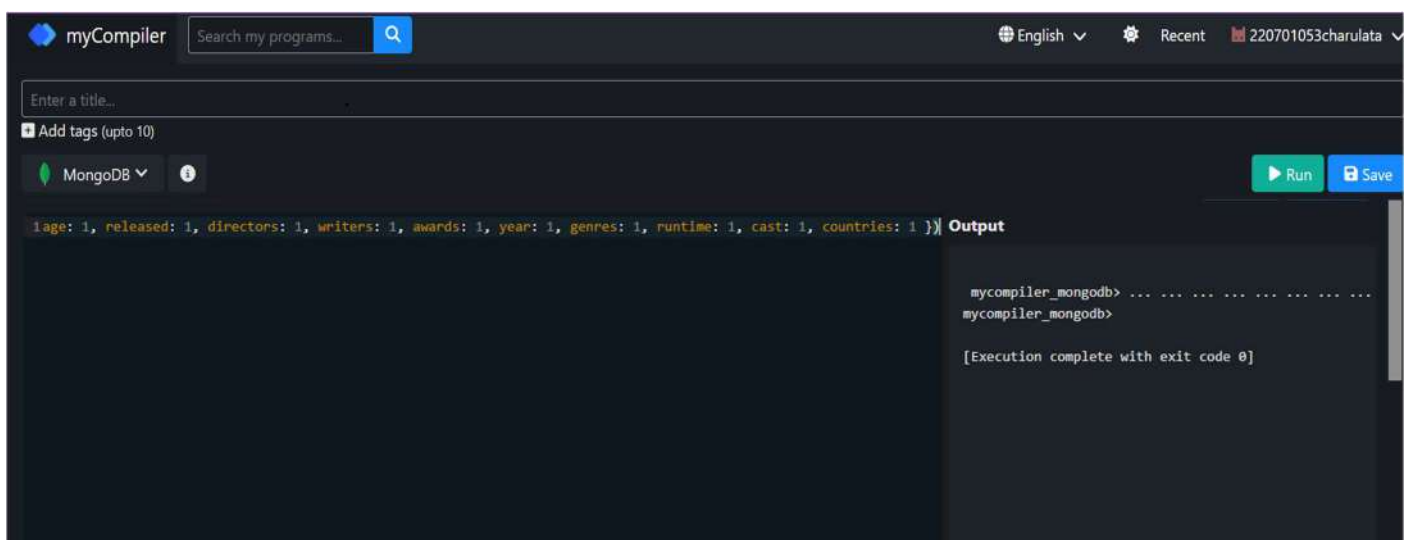
The screenshot shows the myCompiler MongoDB interface. The top bar includes the myCompiler logo, a search bar, and user information. The main area has a text input for a title, a tag input, and a MongoDB dropdown menu. The code editor contains a query: `1age: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 })`. The output panel on the right shows the command `mycompiler_mongodb> ...` and the message `[Execution complete with exit code 0]`.

5.)Retrieve all movies from the 'movies' collection that were released in the USA and include complete information for each movie.

QUERY:

```
db.movies.find({ countries: 'USA' })
```

OUTPUT:



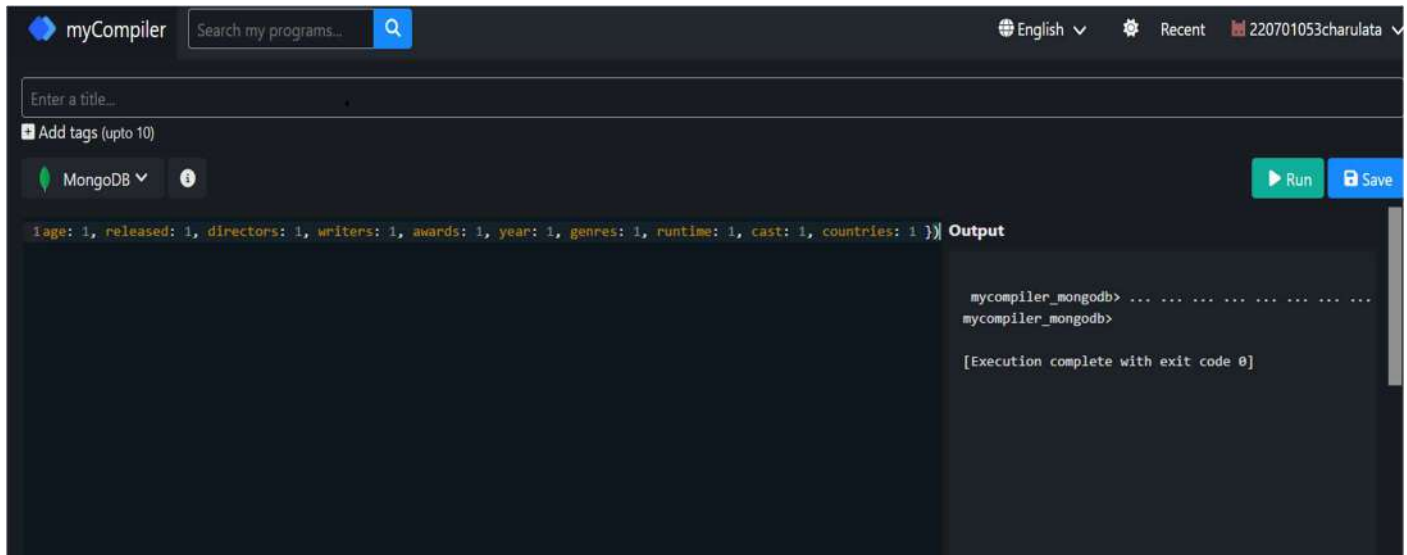
The screenshot shows the myCompiler MongoDB interface. The top bar includes the myCompiler logo, a search bar, and user information. The main area has a text input for a title, a tag input, and a MongoDB dropdown menu. The code editor contains a query: `1age: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 })`. The output panel on the right shows the command `mycompiler_mongodb> ...` and the message `[Execution complete with exit code 0]`.

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

QUERY:

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

OUTPUT:



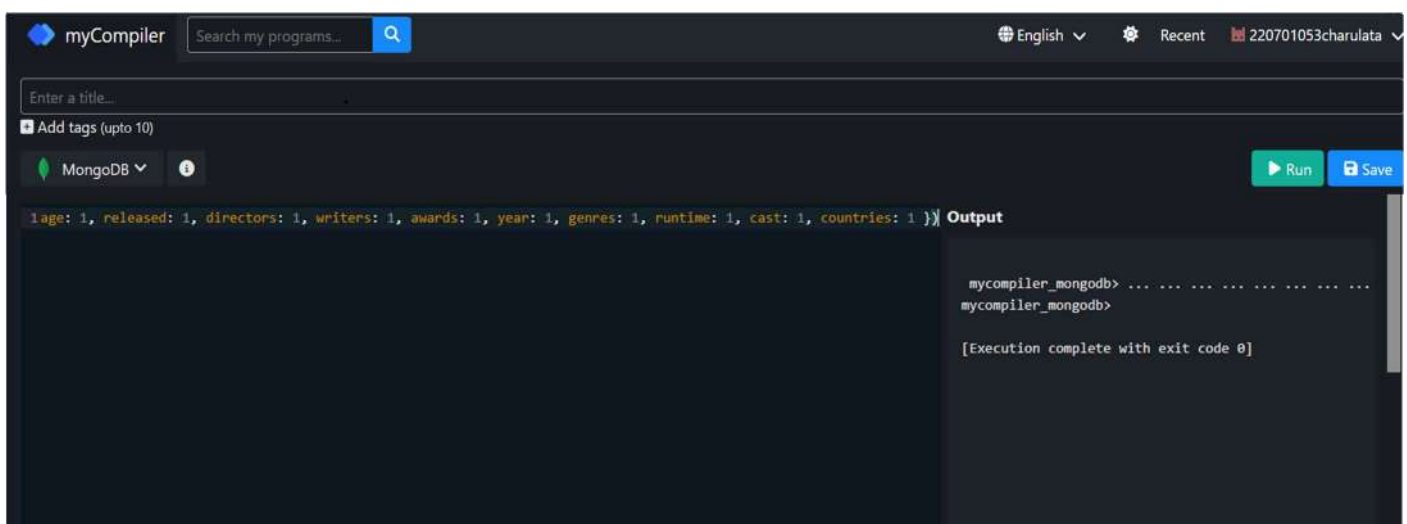
The screenshot shows the myCompiler MongoDB interface. The top bar includes the myCompiler logo, a search bar, and user information. The main area has a text input for a title, a tag input, and a MongoDB dropdown. The code editor contains the query `1age: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 } }`. The output panel on the right shows the command prompt `mycompiler_mongodb>` and the message `[Execution complete with exit code 0]`.

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

QUERY:

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

OUTPUT:



The screenshot shows the myCompiler MongoDB interface. The top bar includes the myCompiler logo, a search bar, and user information. The main area has a text input for a title, a tag input, and a MongoDB dropdown. The code editor contains the query `1age: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 } }`. The output panel on the right shows the command prompt `mycompiler_mongodb>` and the message `[Execution complete with exit code 0]`.

8.)Retrieve all movies from the 'movies' collection that have complete information and have an IMDb rating higher than 7.

QUERY:

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

OUTPUT:



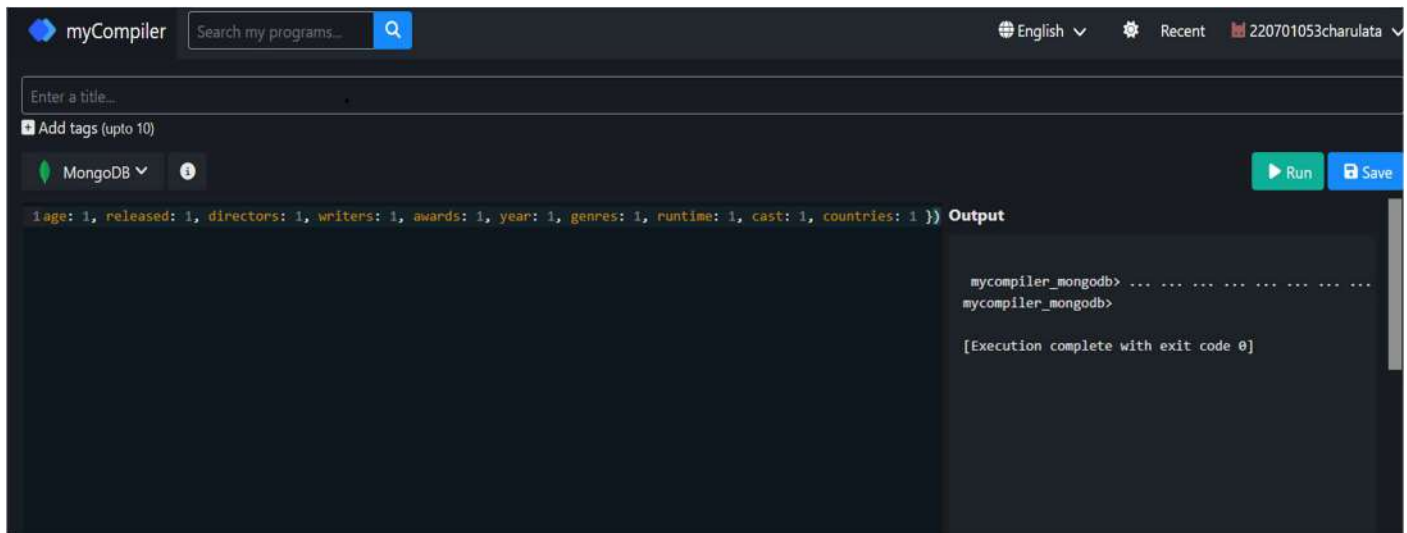
The screenshot shows the myCompiler web interface. At the top, there's a search bar for programs and a language dropdown set to English. Below this is a section for adding tags (up to 10). The main area is a code editor with a MongoDB icon in the top left. The code entered is a MongoDB find query: `1age: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 })`. To the right of the code editor are 'Run' and 'Save' buttons. The output panel on the right shows the command `mycompiler_mongodb>` followed by a separator line of dots, another `mycompiler_mongodb>` prompt, and the message `[Execution complete with exit code 0]`.

9.)Retrieve all movies from the 'movies' collection that have complete information and have a viewer rating higher than 4 on Tomatoes.

QUERY:

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

OUTPUT:



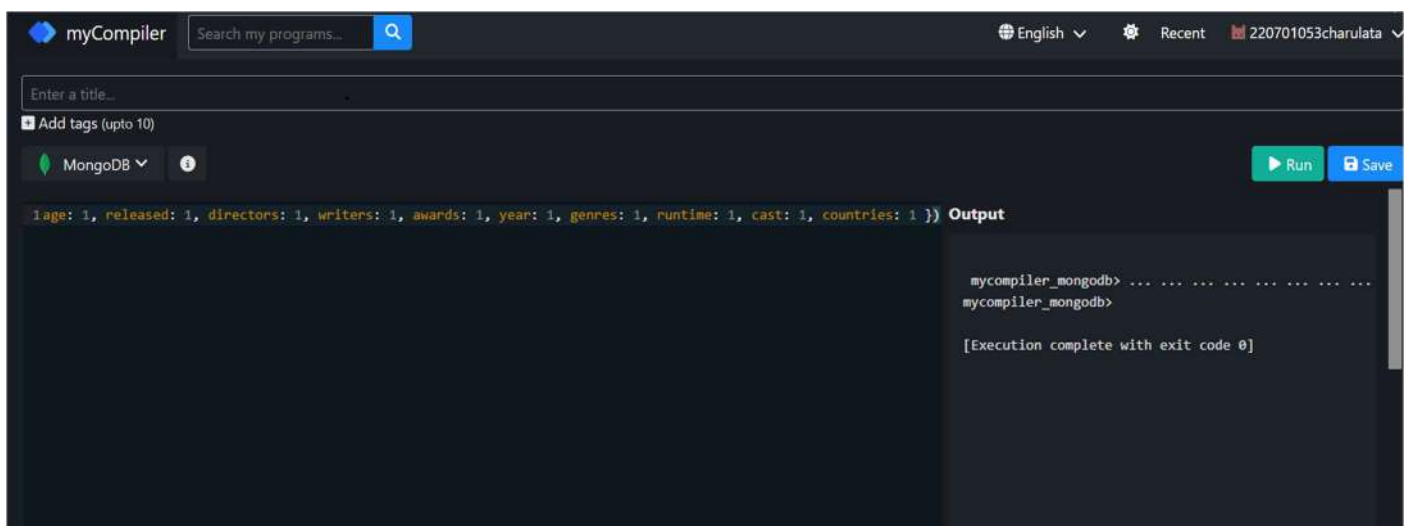
The screenshot shows the myCompiler MongoDB interface. The top bar includes the myCompiler logo, a search bar, and user information. The main area has a title input, tag input, and a MongoDB dropdown. The code editor contains a query: `1age: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 })`. The output panel shows the command `mycompiler_mongodb> ...` and the message `[Execution complete with exit code 0]`.

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

QUERY:

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

OUTPUT:



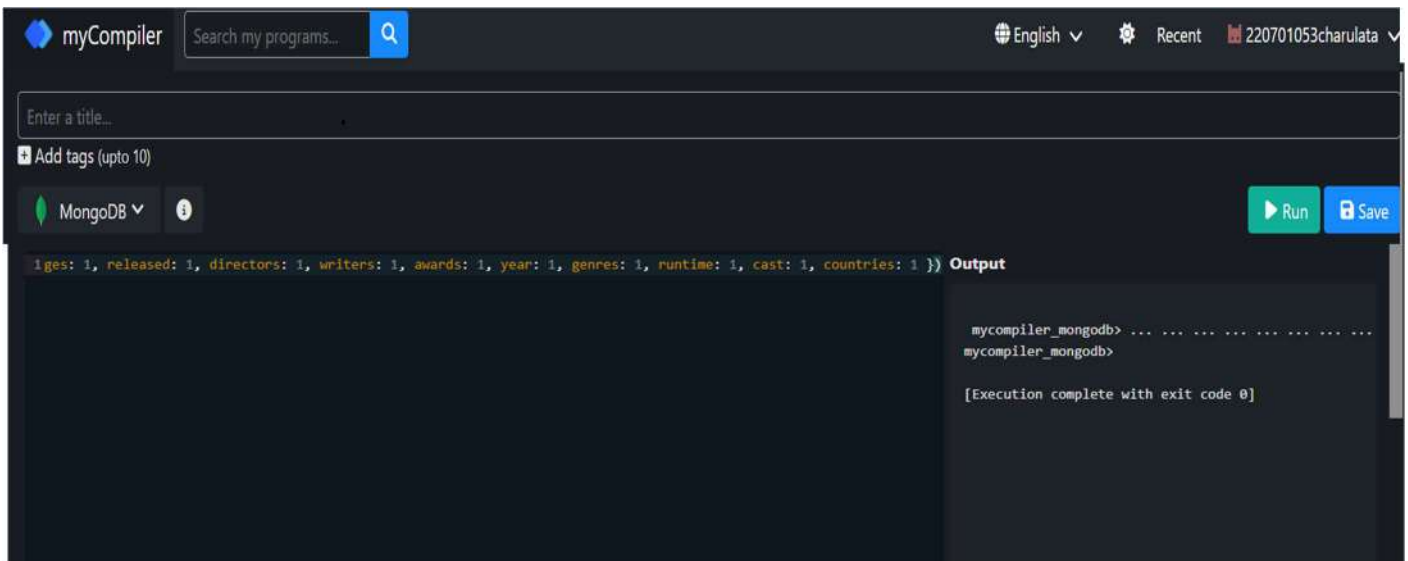
The screenshot shows the myCompiler MongoDB interface. The top bar includes the myCompiler logo, a search bar, and user information. The main area has a title input, tag input, and a MongoDB dropdown. The code editor contains a query: `1age: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 })`. The output panel shows the command `mycompiler_mongodb> ...` and the message `[Execution complete with exit code 0]`.

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

QUERY:

```
db.movies.find( { 'awards.nominations': { $gt: 0 } }, { title: 1, languages: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 })
```

OUTPUT:



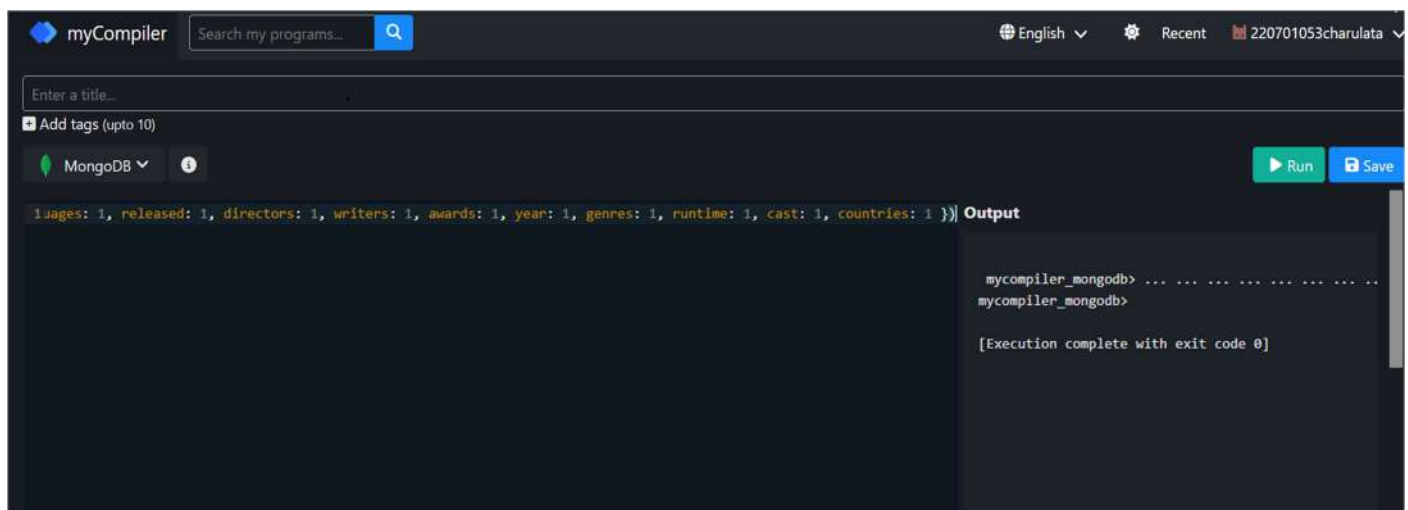
The screenshot shows the myCompiler MongoDB interface. The query entered is: `ges: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 })`. The output panel shows the command `mycompiler_mongodb> ...` and the message `[Execution complete with exit code 0]`.

12.)Find all movies with title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries from the 'movies' collection in MongoDB with cast including "Charles Kayser".

QUERY:

```
db.movies.find( { cast: 'Charles Kayser' }, { title: 1, languages: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 })
```

OUTPUT:



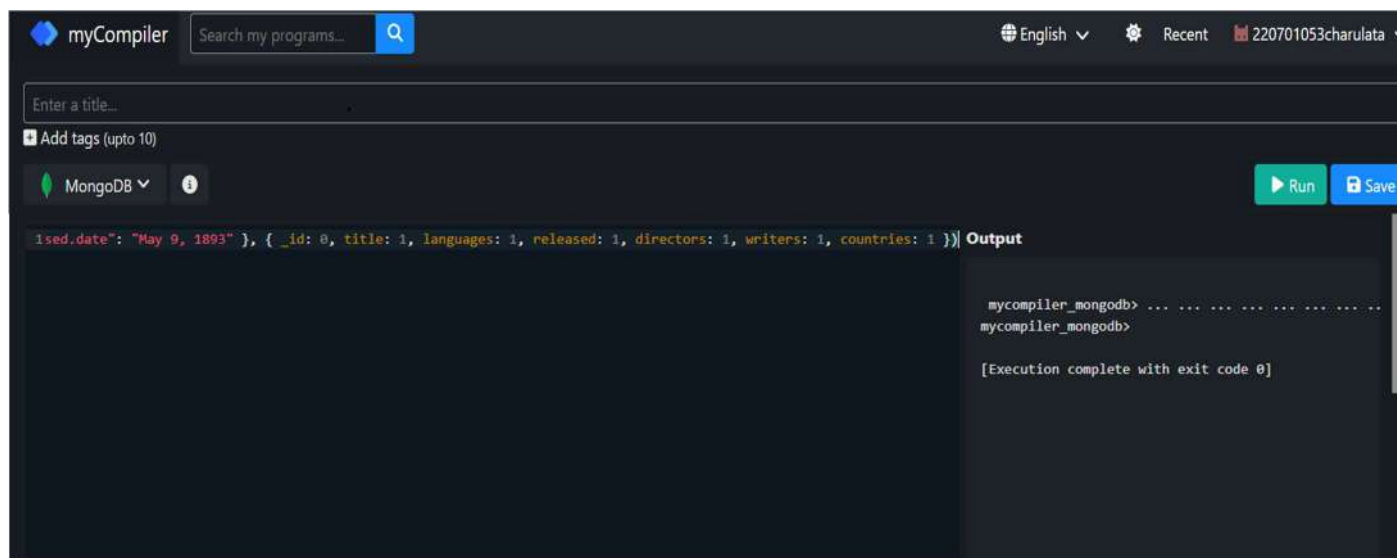
The screenshot shows the myCompiler MongoDB interface. The query entered is: `languages: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 })`. The output panel shows the command `mycompiler_mongodb> ...` and the message `[Execution complete with exit code 0]`.

13.)Retrieve all movies with title, languages, released, directors, writers, countries from the 'movies' collection in MongoDB that released on May 9, 1893.

QUERY:

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

OUTPUT:



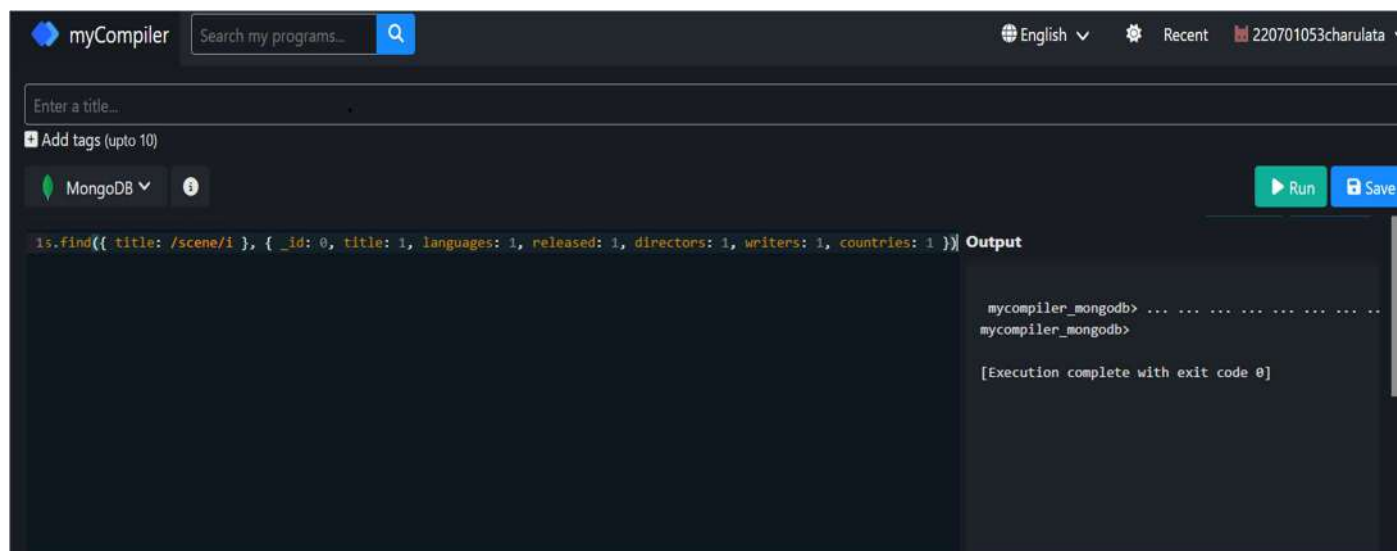
The screenshot shows the myCompiler interface with the MongoDB tab selected. The query entered is `db.movies.find({ released: ISODate("1893-05-09T00:00:00.000Z") }, { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 })`. The output pane on the right shows the command prompt `mycompiler_mongodb>` and the message `[Execution complete with exit code 0]`.

14.)Retrieve all movies with title, languages, released, directors, writers, countries from the 'movies' collection in MongoDB that have a word "scene" in the title.

QUERY:

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

OUTPUT:



The screenshot shows the myCompiler interface with the MongoDB tab selected. The query entered is `db.movies.find({ title: /scene/i }, { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 })`. The output pane on the right shows the command prompt `mycompiler_mongodb>` and the message `[Execution complete with exit code 0]`.

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

RESULT:

