

Creating and Managing Tables

EX_NO:1

DATE:

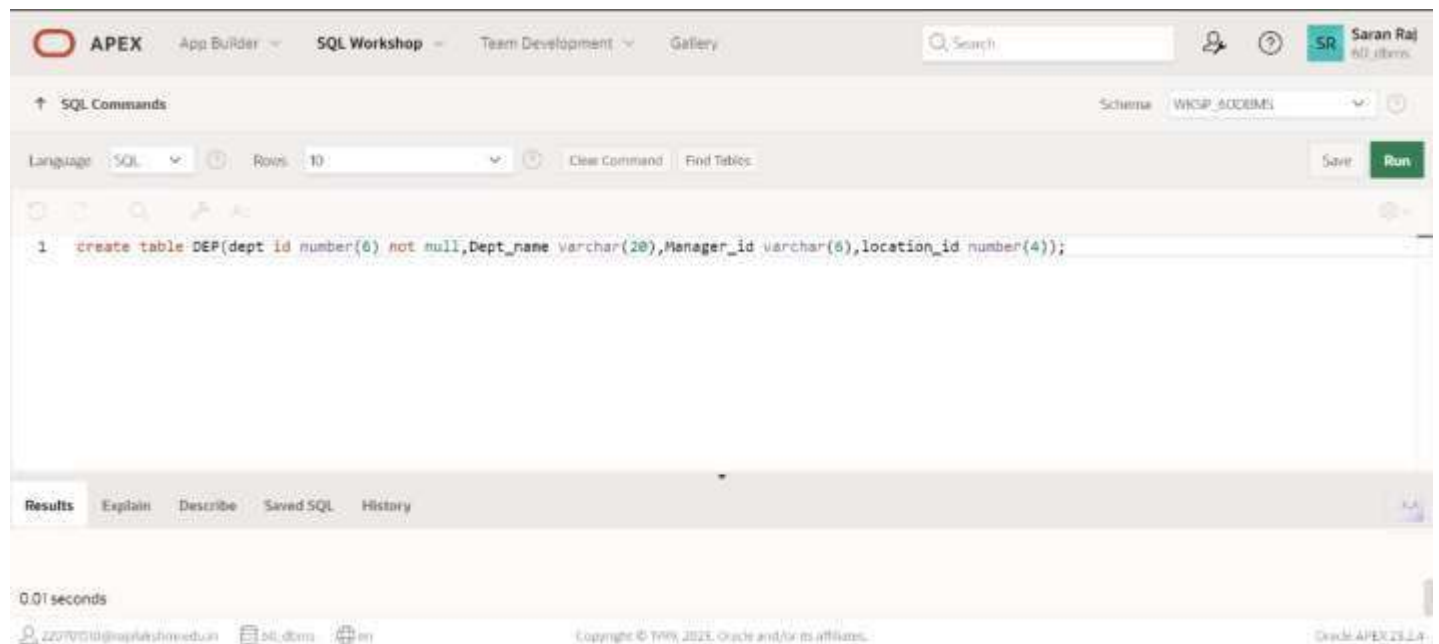
1.Create the DEPT table based on the DEPARTMENT following the table instance chart below. Confirm that the table is created.

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

QUERY:

Create table dept(id number(7) not null, 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),lastname varchar(25),firstname varchar(25),deptid number(7));

OUTPUT:

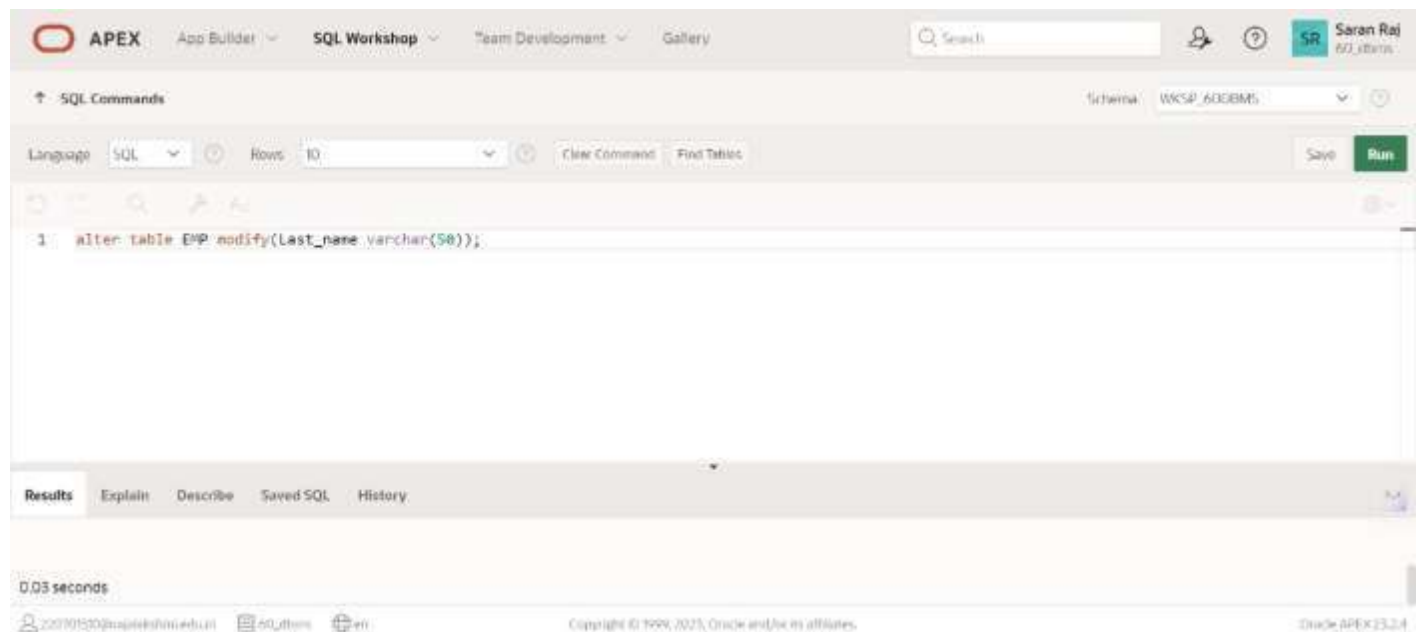


3.Modify the EMP table to allow for longer employee last names. Confirm the modification.(Hint: Increase the size to 50)

QUERY:

Alter table emp modify(lastname varchar(50));

OUTPUT:

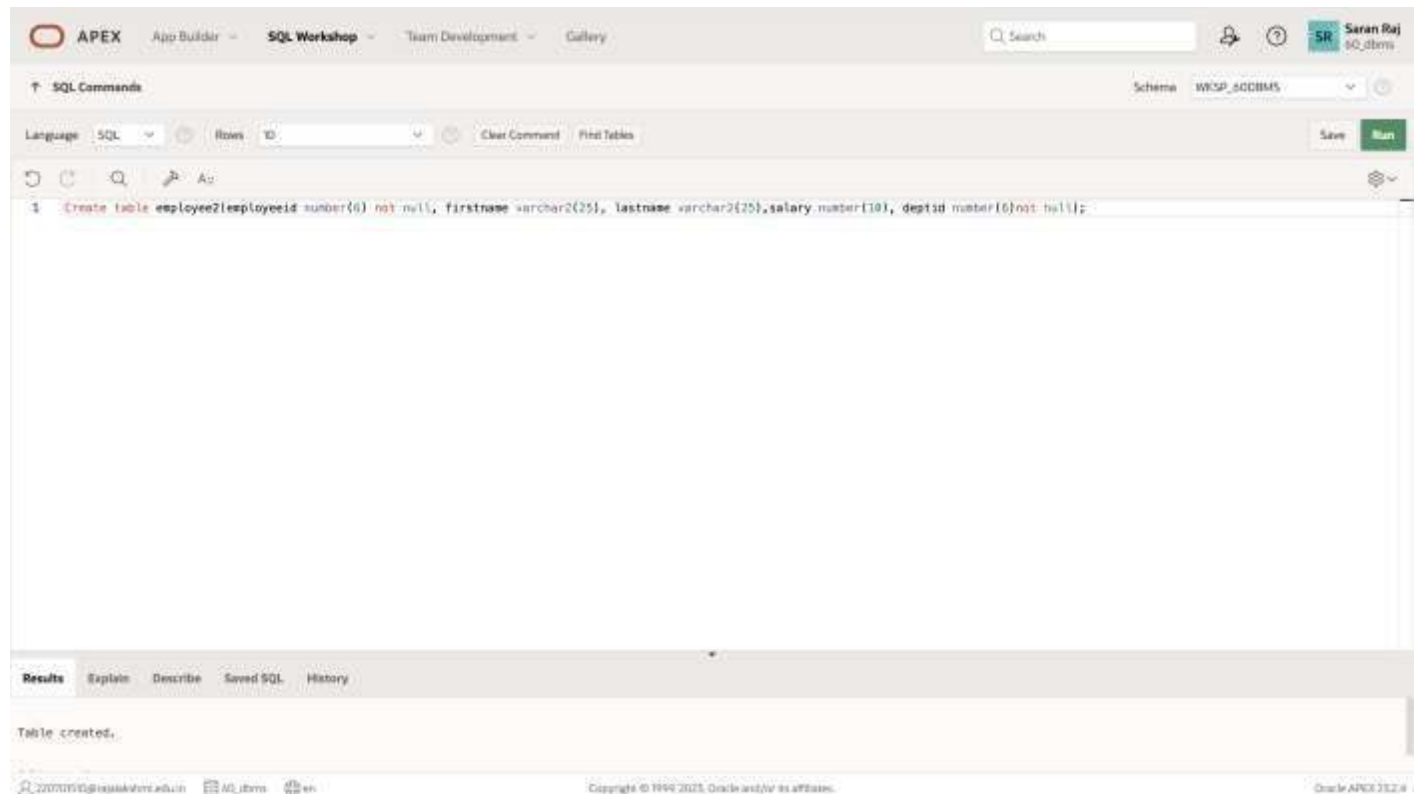


4.Create the EMPLOYEES2 table based on the structure of EMPLOYEES table. Include Only the Employee_id, First_name, Last_name, Salary and Dept_id coloumns. Name the columns Id, First_name, Last_name, salary and Dept_id respectively.

QUERY:

Create table employee2(employeeid number(6) not null, firstname varchar2(25), lastname varchar2(25),salary number(10), deptid number(6)not null);

OUTPUT:

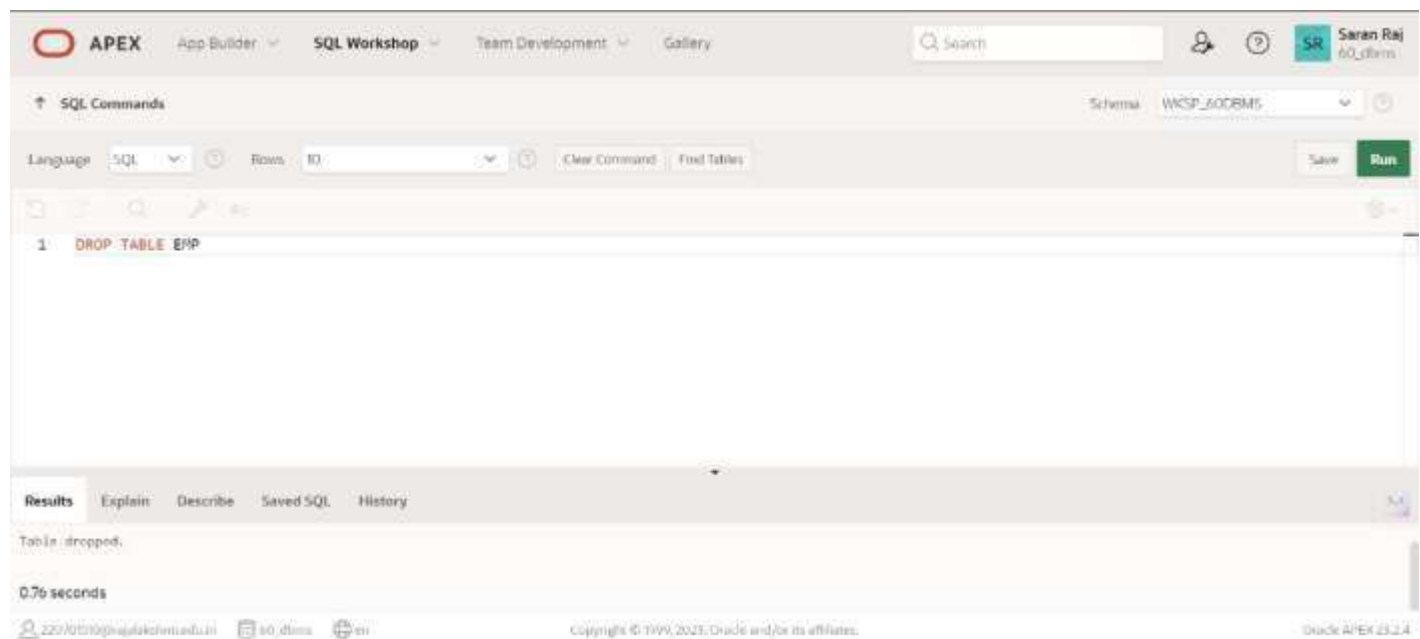


5.Drop the EMP table.

QUERY:

Drop table emp;

OUTPUT:



6.Rename the EMPLOYEES2 table as EMP.

QUERY:

Rename emp2 to emp;

OUTPUT:



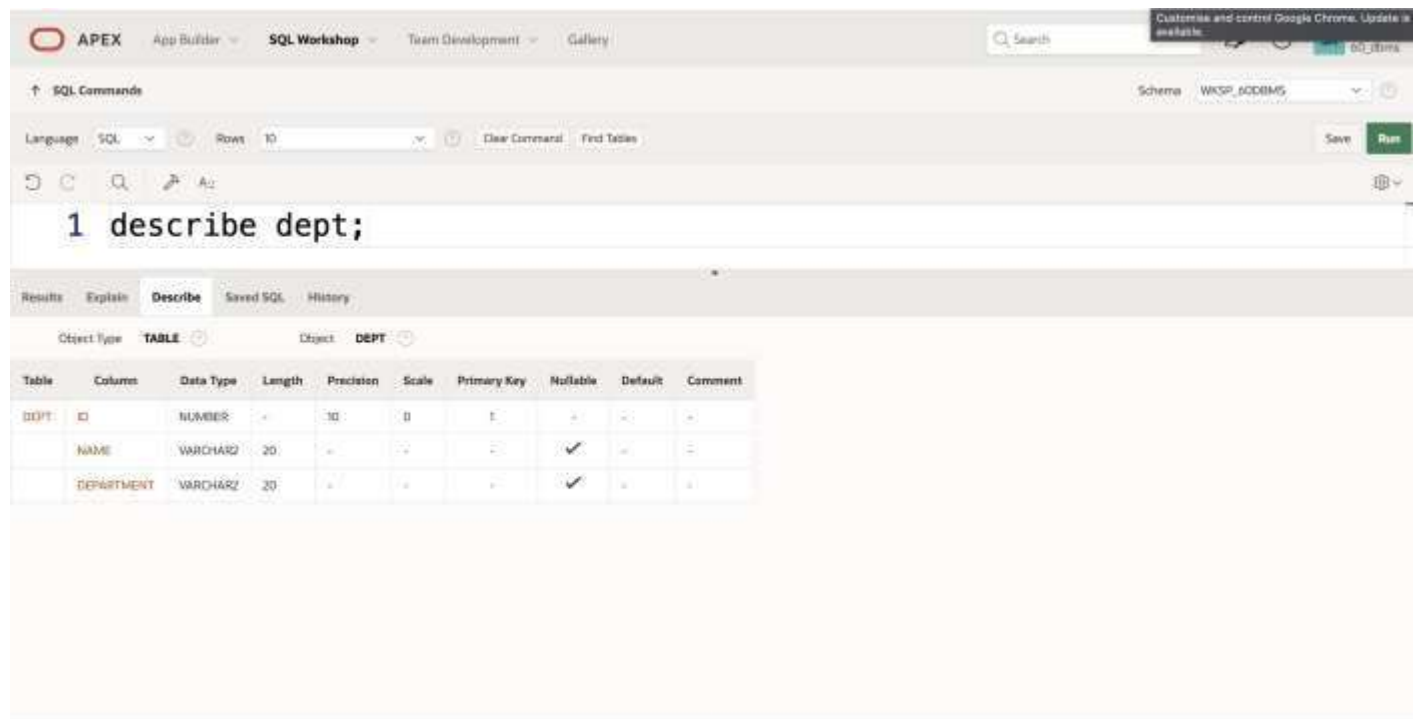
7. Add a comment on DEPT and EMP tables. Confirm the modification by describing the table.

QUERY:

comment on table dept is 'Department info';

comment on table emp is Employee info';

OUTPUT:

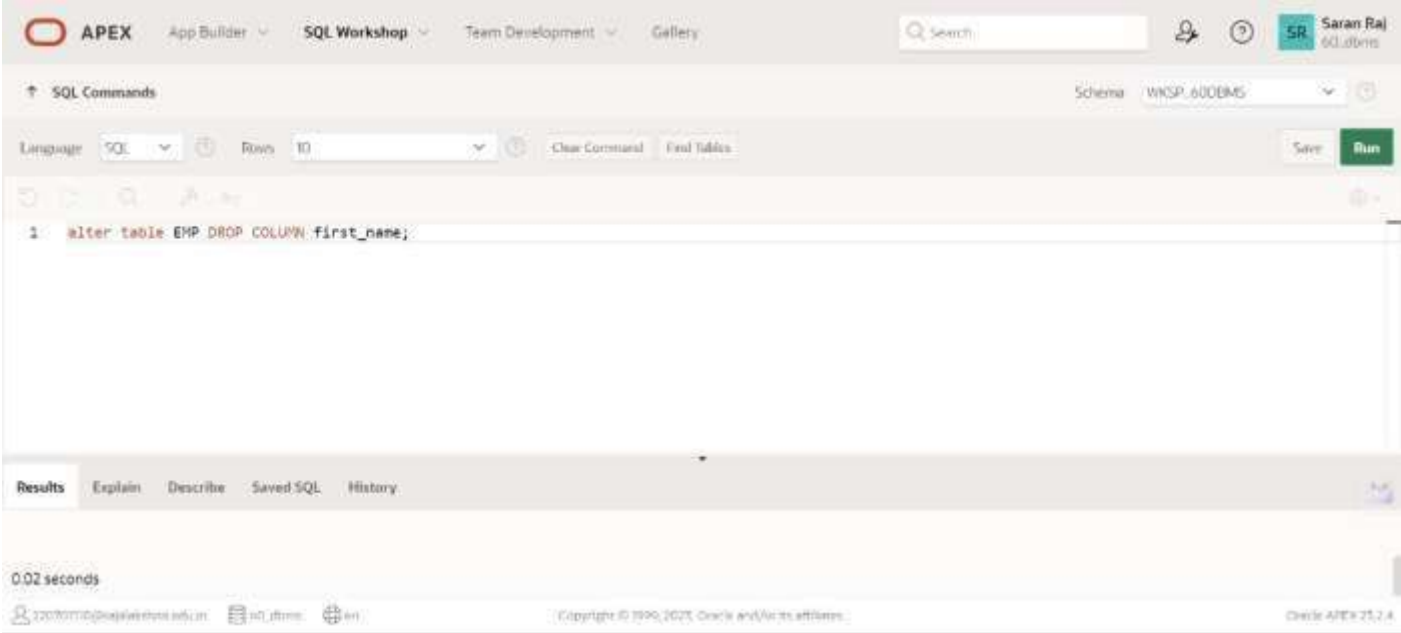


8. Drop the First_name column from the EMP table and confirm it.

QUERY:

Alter table emp drop column firstname;

OUTPUT:



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

RESULT:

MANIPULATING DATA

EX_NO:2

DATE:

1.Create MY_EMPLOYEE table with the following structure

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

QUERY:

Create table myemployee(id number(4) not null ,lastname varchar(25),firstname varchar(25),userid varchar(25),salary number(9,2);

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The main area displays the SQL Commands tab with the following command:

```
1 create table MY_employe(id number(4),lasty_name varchar(25),first_name varchar(25),
2 user_id varchar(25),salary number(9,2));
```

The Results tab is active, showing an error message: "Error at Line 1/14: ORA-00955: name is already used by an existing object". Below the error message, the command is repeated: "1. create table MY_employe(id number(4),lasty_name varchar(25),first_name varchar(25),user_id varchar(25),salary number(9,2));". The execution time is shown as 0.02 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 myemployee values(2,'dancs','betty','bdancs',860);

OUTPUT:

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

```
1 Insert into MY_EMPLOYEE  
2 values(2,'dancs','betty','bdancs',860);
```

The results pane shows:

1 row(s) inserted.
0.05 seconds

3.Display the table with values.

QUERY:

Select * from employee;

OUTPUT:

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

```
1 Select * from employee  
2 order by id ;
```

The results pane shows a table with the following data:

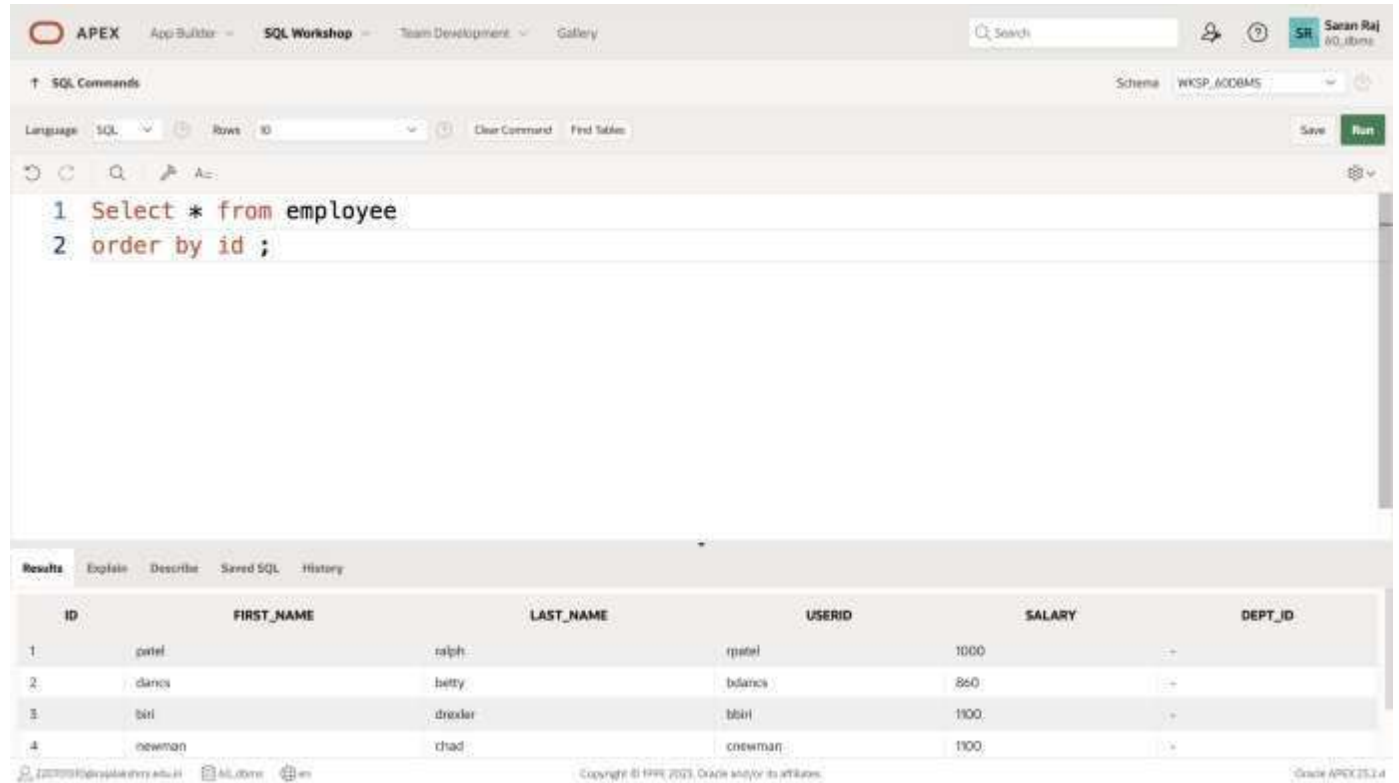
ID	FIRST_NAME	LAST_NAME	USERID	SALARY	DEPT_ID
1	patel	ralph	rpatel	1000	-
2	dancs	betty	bdancs	860	-

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 employee values (4,'newman','chad','cnewman',860);

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL Commands pane contains the query: `1 Select * from employee` and `2 order by id ;`. The Results pane displays a table with 4 rows and 6 columns: ID, FIRST_NAME, LAST_NAME, USERID, SALARY, and DEPT_ID.

ID	FIRST_NAME	LAST_NAME	USERID	SALARY	DEPT_ID
1	patel	ralph	rpatal	1000	10
2	dancy	betty	bdancy	850	10
3	bihi	drexler	bbih	1100	10
4	newman	chad	cnewman	1100	10

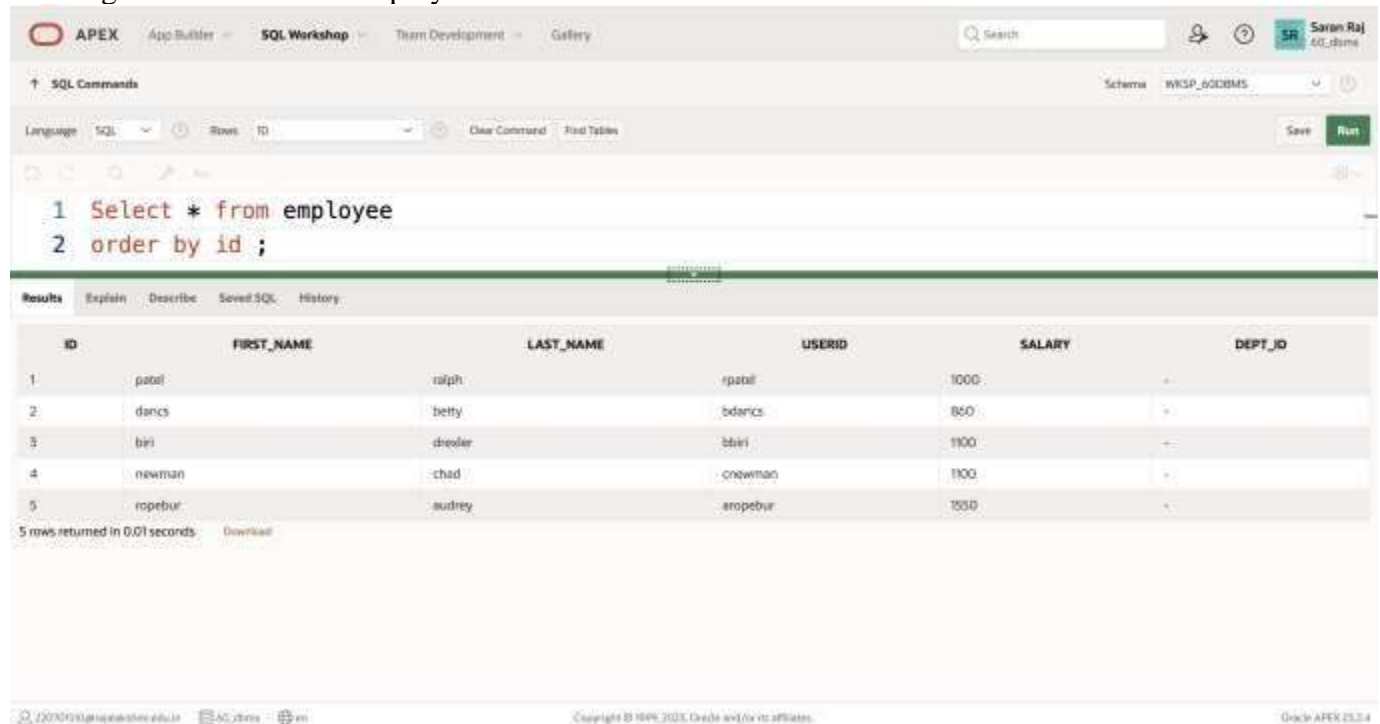
5. Make the data additions permanent.

QUERY:

Select * from employee;

OUTPUT:

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



The screenshot shows the APEX SQL Workshop interface. The SQL Commands pane contains the query: `1 Select * from employee` and `2 order by id ;`. The Results pane displays a table with 5 rows and 6 columns: ID, FIRST_NAME, LAST_NAME, USERID, SALARY, and DEPT_ID.

ID	FIRST_NAME	LAST_NAME	USERID	SALARY	DEPT_ID
1	patel	ralph	rpatal	1000	10
2	dancy	betty	bdancy	850	10
3	bihi	drexler	bbih	1100	10
4	newman	chad	cnewman	1100	10
5	ropebur	audrey	eropebur	1050	10

5 rows returned in 0.01 seconds

QUERY:

UPDATE employee SET last_name = 'drexler' where id=3;

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 'Saran Raj' are on the right. The 'SQL Commands' section is active, showing a query: `1 UPDATE employee`, `2 SET last_name = 'drexler'`, `3 where id=3;`. The 'Results' tab shows '1 row(s) updated.' and '0.01 seconds'. The bottom status bar indicates 'Copyright © 1996, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

QUERY:

UPDATE employee SET salary=1000 where salary<900;

OUTPUT:

This screenshot shows the Oracle APEX SQL Workshop interface with the same query as the previous screenshot: `1 UPDATE employee`, `2 SET salary=1000`, `3 where salary<900;`. The 'Results' tab shows '1 row(s) updated.' and '0.01 seconds'. The bottom status bar is identical to the first screenshot, showing 'Copyright © 1996, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

8.Delete Betty dancs from MY_EMPLOYEE table.

QUERY:

Delete from myemployee where firstname='Betty';

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 'Saran Raj' are on the right. The 'SQL Commands' tab is active, showing a query editor with the following SQL code:

```
1 Delete from employee
2 where first_name='Betty';
3
```

Below the editor, the 'Results' tab shows the execution output: '0 row(s) deleted.' and '0.02 seconds'. The bottom status bar indicates the user is logged in as '220700510@apexlakshmi.edu.in' and the schema is 'WKSP_60DBMS'.

9. Empty the fourth row of the emp table.

QUERY:

Delete from emp where employeeid=10004;

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 'Saran Raj' are on the right. The 'SQL Commands' tab is active, showing a query editor with the following SQL code:

```
1 Delete from employees
2 where employeeid=10004;
```

Below the editor, the 'Results' tab shows the execution output: '1 row(s) deleted.' and '0.02 seconds'. The bottom status bar indicates the user is logged in as '220700510@apexlakshmi.edu.in' and the schema is 'WKSP_60DBMS'.

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

RESULT:

INCLUDING CONSTRAINTS

EX_NO:3

DATE:

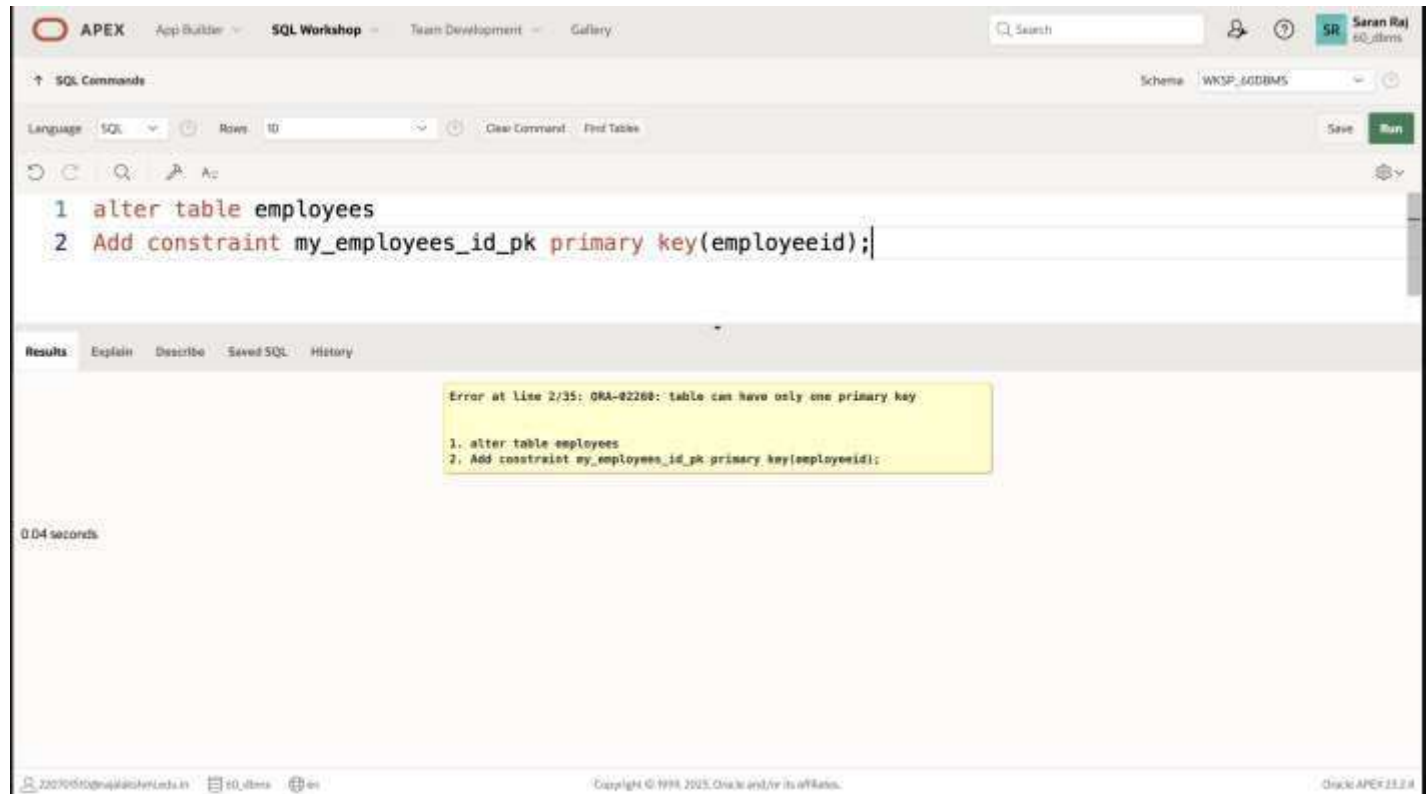
1. Add a table-level PRIMARY KEY constraint to the EMP table on the ID column. The constraint should be named at creation. Name the constraint my_emp_id_pk.

QUERY:

Alter table emp

Add constraint my_emp_id_pk primary key(employee id);

OUTPUT:



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

```
1 alter table employees
2 Add constraint my_employees_id_pk primary key(employeeid);
```

The Results pane shows an error message:

```
Error at line 2/35: ORA-02268: table can have only one primary key
1. alter table employees
2. Add constraint my_employees_id_pk primary key(employeeid);
```

The execution time is 0.04 seconds.

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

QUERY:

Alter table dept

Add constraint my_dept_id_pk primary key(deptid);

OUTPUT:

APEX

App Builder

SQL Workshop

Team Development

Gallery

Search

SQL Commands

SchemaWKSP_600BMS

LanguageSQL

Rows10

Clear Command

Find Tables

Aut

1Alter table dept

2Add constraint my_dept_id_pk primary key(id);

Results

Explain

Describe

Saved SQL

History

Error at Line 2/30: ORA-02260: table can have only one primary key

1. Alter table dept

2. Add constraint my_dept_id_pk primary key(id);

0.02 seconds

220701810@vjalkshmi.edu.in

80_18m1

en

Copyright © 1996, 2023, Oracle and/or its affiliates.

Dis

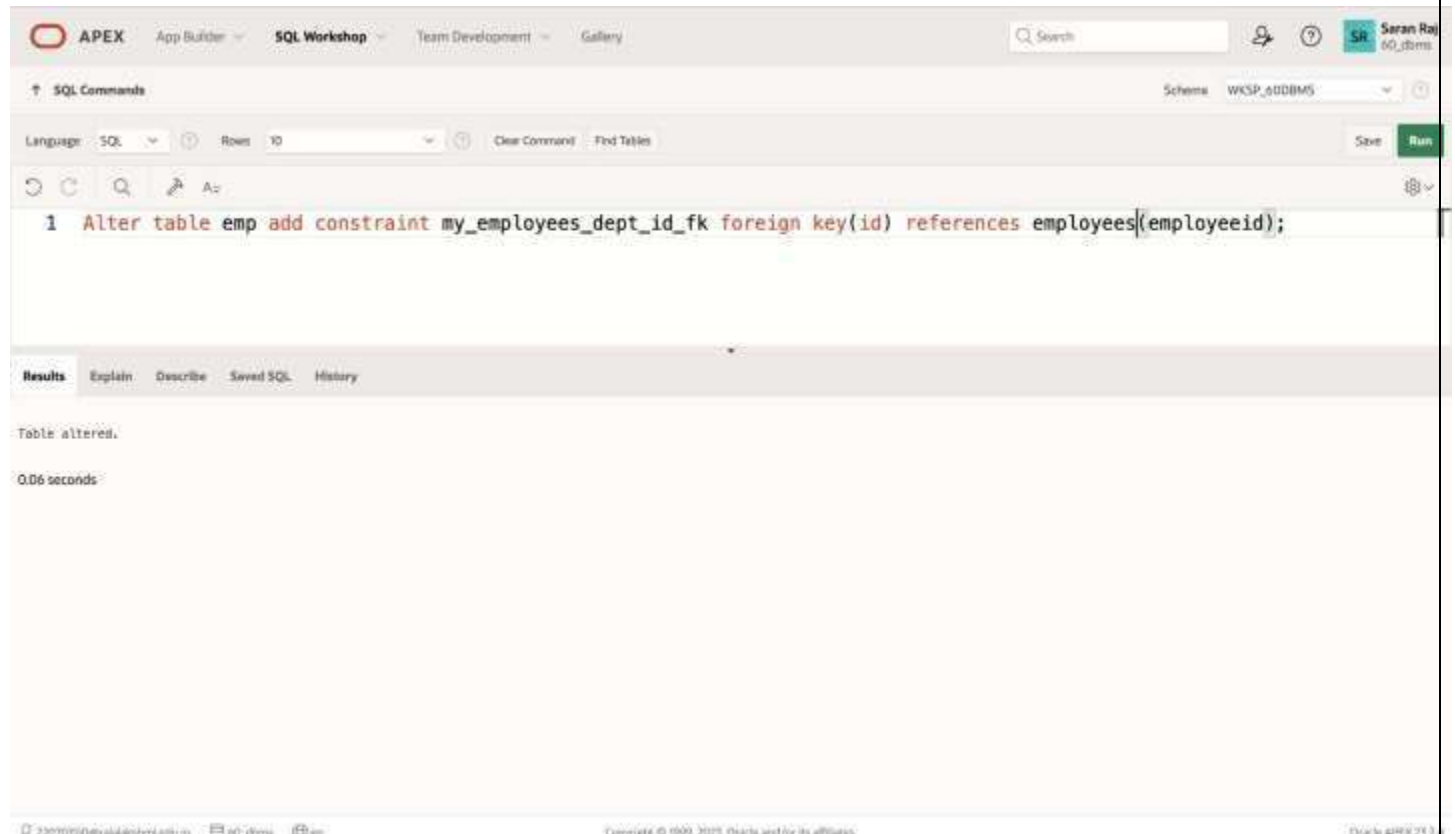
```
[T][T][T][T][T]
[SEP][SEP][SEP][SEP][SEP]
[T][T][T][T][T]
[SEP][SEP][SEP][SEP][SEP]
[T][T][T][T][T][T][T][T][T][T]
[SEP][SEP][SEP][SEP][SEP][SEP][SEP][SEP]
```

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

QUERY:

Alter table emp add constraint my_emp_dept_id_fk foreign key(deptid) references emp(employeeid);

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 emp

ADD COMMISSION NUMBER(2, 2) CONSTRAINT emp_comm_check CHECK (commission > 0);

OUTPUT:



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

RESULT:

Writing Basic SQL SELECT Statements

EX_NO:4

DATE:

1. The following statement executes successfully.

Identify the Errors

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

QUERY:

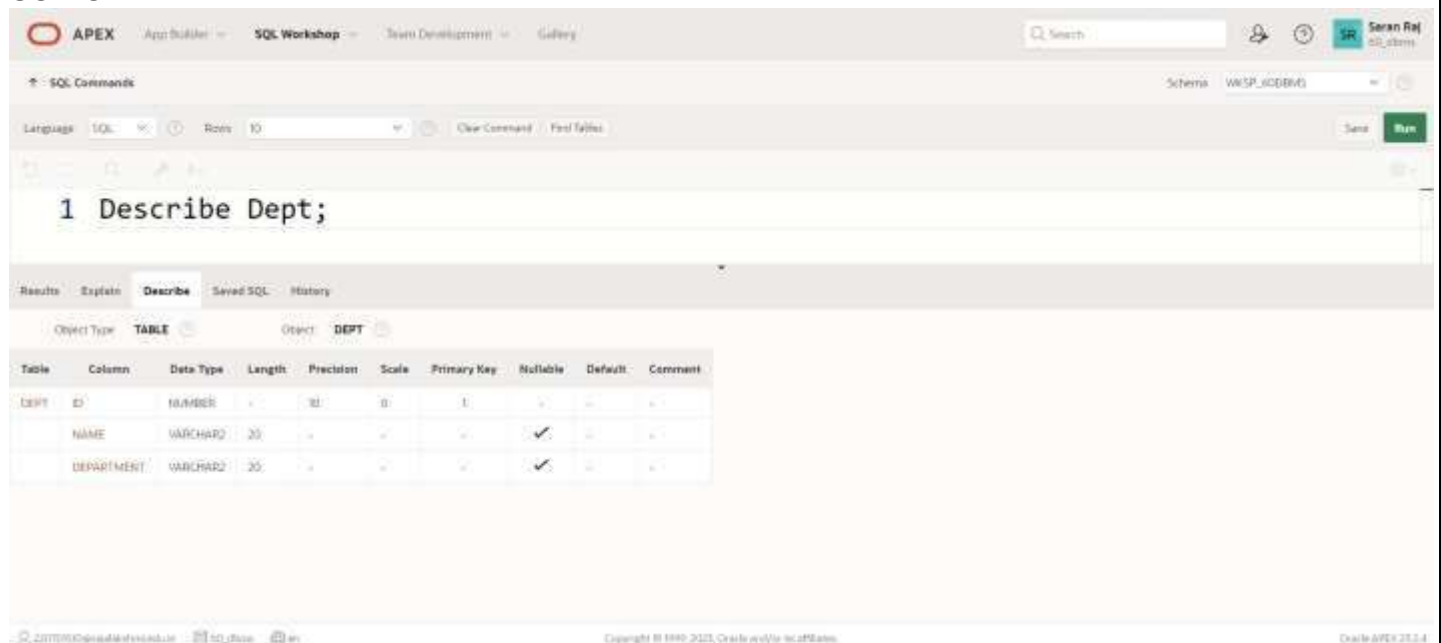
```
SELECT employeeid, lastname ,Salary*12 as Annual salary  
From employees;
```

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

QUERY:

Describe Dept;

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The 'SQL Commands' tab is active, and the command '1 Describe Dept;' is entered in the SQL editor. The 'Describe' tab is selected, showing the table structure for the 'DEPT' table. The table has three columns: ID (NUMBER, 10, Primary Key), NAME (VARCHAR2, 25, Nullable), and DEPARTMENT (VARCHAR2, 25, Nullable).

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPT	ID	NUMBER	10		0	✓	✓		
	NAME	VARCHAR2	25				✓		
	DEPARTMENT	VARCHAR2	25				✓		

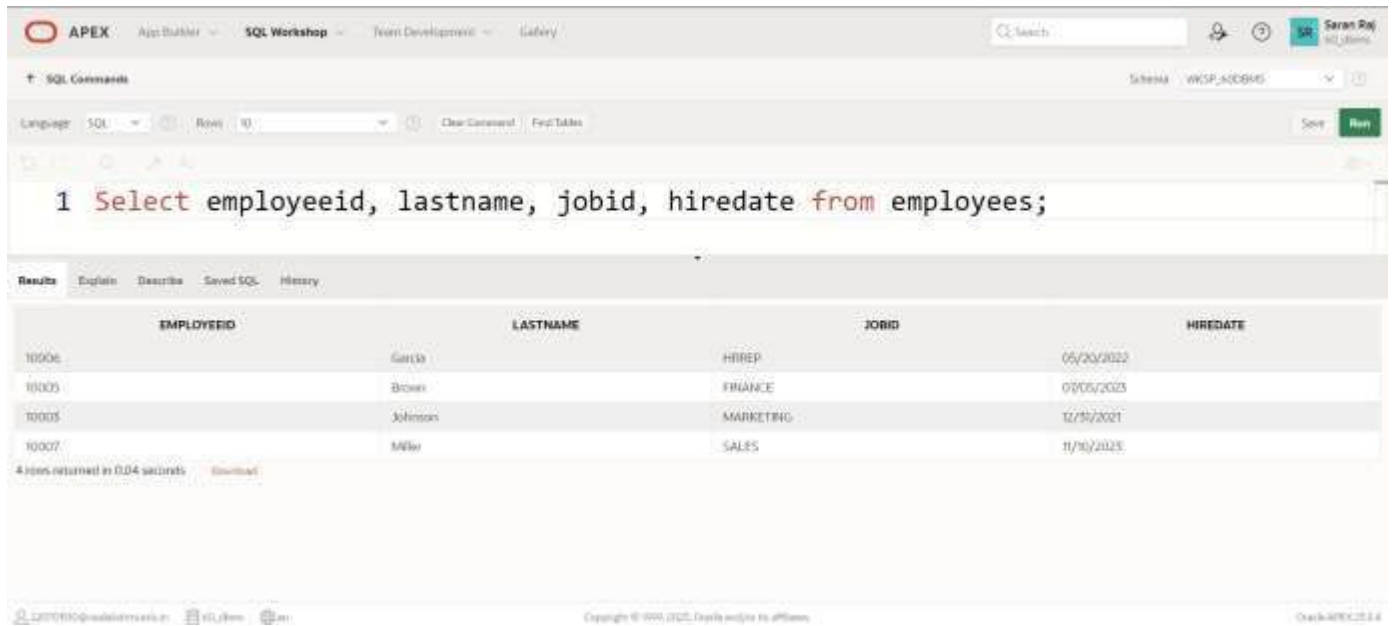
RESULT:

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 employeeid, lastname, jobid, hiredate from employees;

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command entered is: `1 Select employeeid, lastname, jobid, hiredate from employees;`. The results are displayed in a table with the following data:

EMPLOYEEID	LASTNAME	JOBID	HIREDATE
10006	Gutts	HRREP	05/20/2022
10005	Brown	FINANCE	07/05/2023
10005	Johnson	MARKETING	12/31/2021
10007	Miller	SALES	11/30/2023

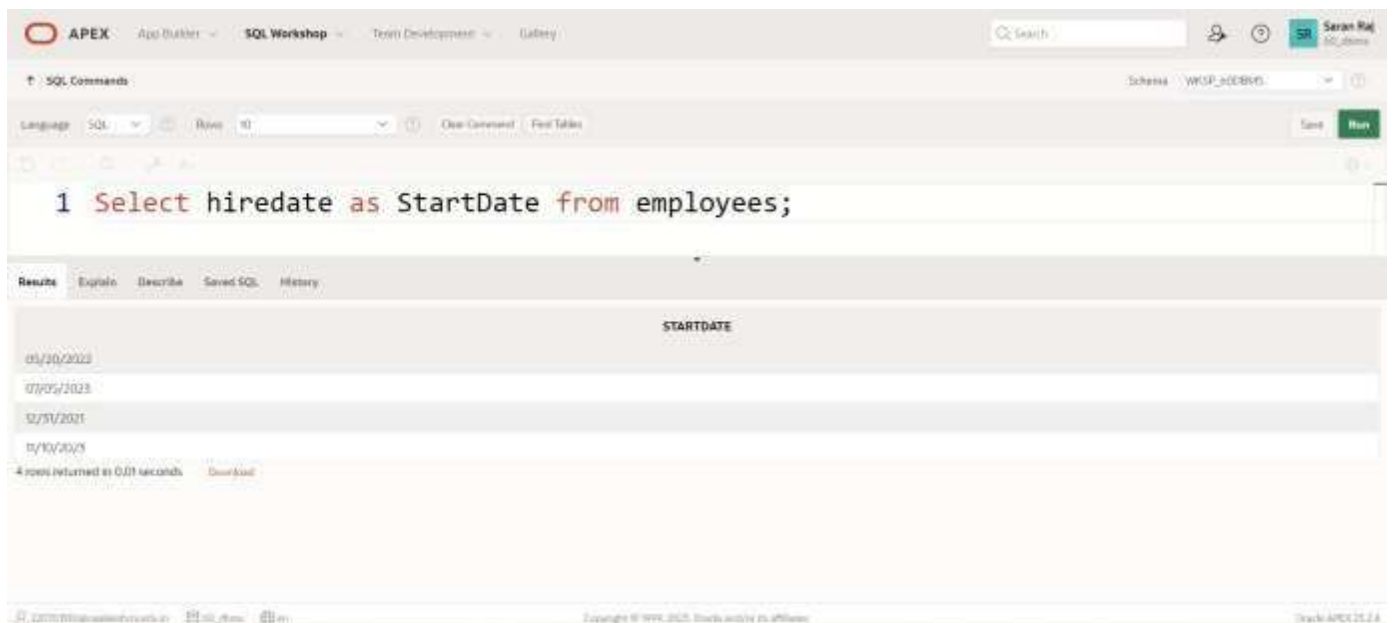
4 rows returned in 0.04 seconds

4. Provide an alias STARTDATE for the hire date.

QUERY:

Select hiredate as StartDate from employees;

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command entered is: `1 Select hiredate as StartDate from employees;`. The results are displayed in a table with the following data:

STARTDATE
05/20/2022
07/05/2023
12/31/2021
11/30/2023

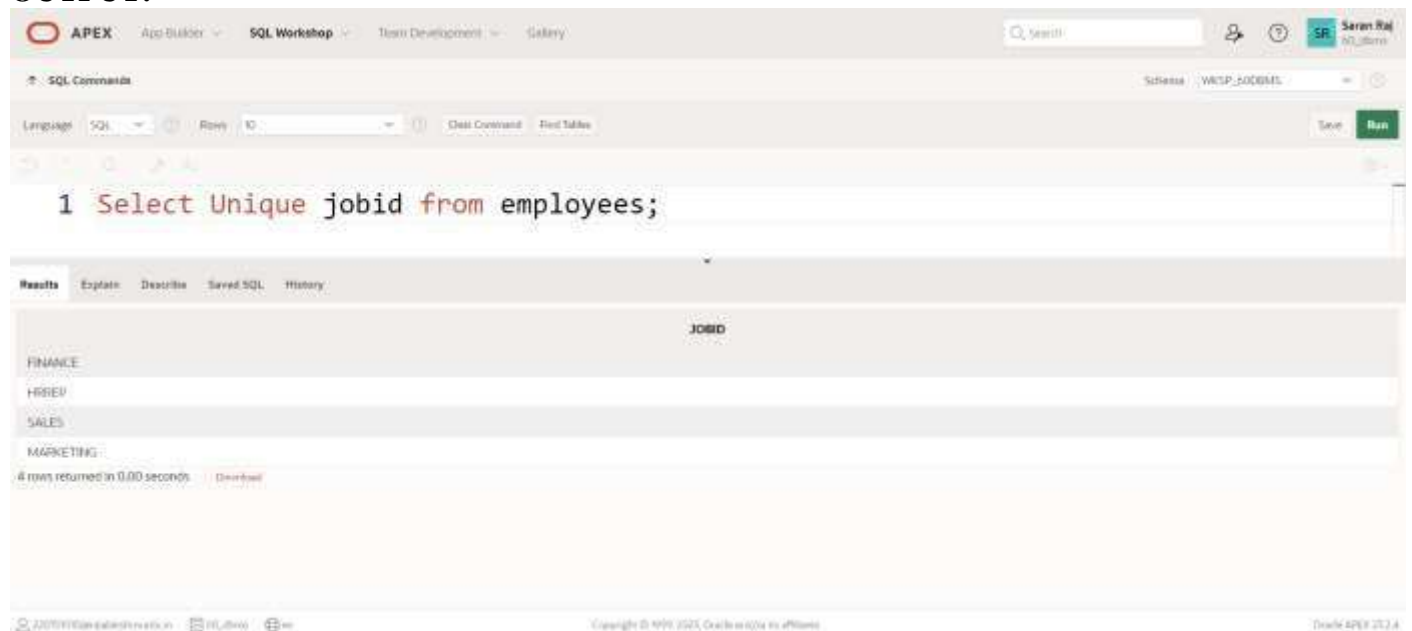
4 rows returned in 0.01 seconds

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

QUERY:

Select Unique jobid from employees;

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The query editor contains the text: `1 Select Unique jobid from employees;`. The results pane displays a table with the column header **JOBID** and four rows of data: `FINANCE`, `HRREP`, `SALES`, and `MARKETING`. Below the table, it states "4 rows returned in 0.00 seconds" and provides a "Download" link.

JOBID
FINANCE
HRREP
SALES
MARKETING

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 lastname || ', ' || jobid as Title from employees;

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The query editor contains the text: `1 Select lastname || ', ' || jobid as Title from employees;`. The results pane displays a table with the column header **TITLE** and four rows of data: `Garcia,HRREP`, `Brown,FINANCE`, `Johnson,MARKETING`, and `Mina,SALES`. Below the table, it states "4 rows returned in 0.01 seconds" and provides a "Download" link.

TITLE
Garcia,HRREP
Brown,FINANCE
Johnson,MARKETING
Mina,SALES

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 employeeid ||','|| lastname ||','|| jobid ||','|| email ||','|| salary ||','|| hiredate as "the_output" from employees;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is: `1 Select employeeid ||','|| lastname ||','|| jobid ||','|| email ||','|| salary ||','|| hiredate as "the_output" from emp`. The results are displayed in a table with the column header `the_output`. The results show four rows of employee data with their details concatenated into a single string per row. The status bar indicates 4 rows returned in 0.00 seconds.

the_output
10006,Gardai,HIREP,sarah.gardai@company.com,45000,05/20/2022
10005,Brown,FINANCE,michael.brown@company.com,50000,07/05/2023
10003,Johnson,MARKETING,alice.johnson@company.com,60000,02/31/2021
10007,Miller,SALES,william.miller@company.com,30000,11/10/2023

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

RESULT:

RESTRICTING AND SORTING DATA

EX_NO:5

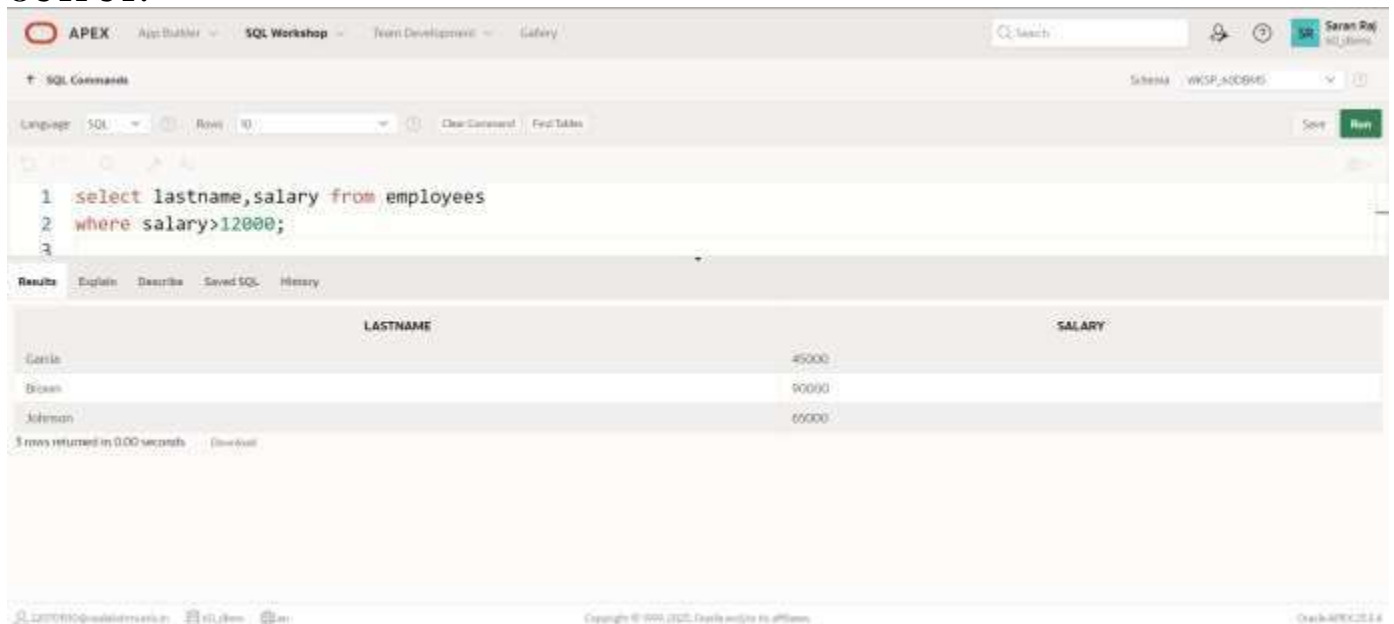
DATE:

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

QUERY:

```
select lastname,salary from employees
where salary>12000;
```

OUTPUT:



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

```
1 select lastname,salary from employees
2 where salary>12000;
3
```

The Results tab is selected, displaying a table with two columns: LASTNAME and SALARY. The table contains three rows of data:

LASTNAME	SALARY
Ganis	45000
Brown	90000
Johnson	60000

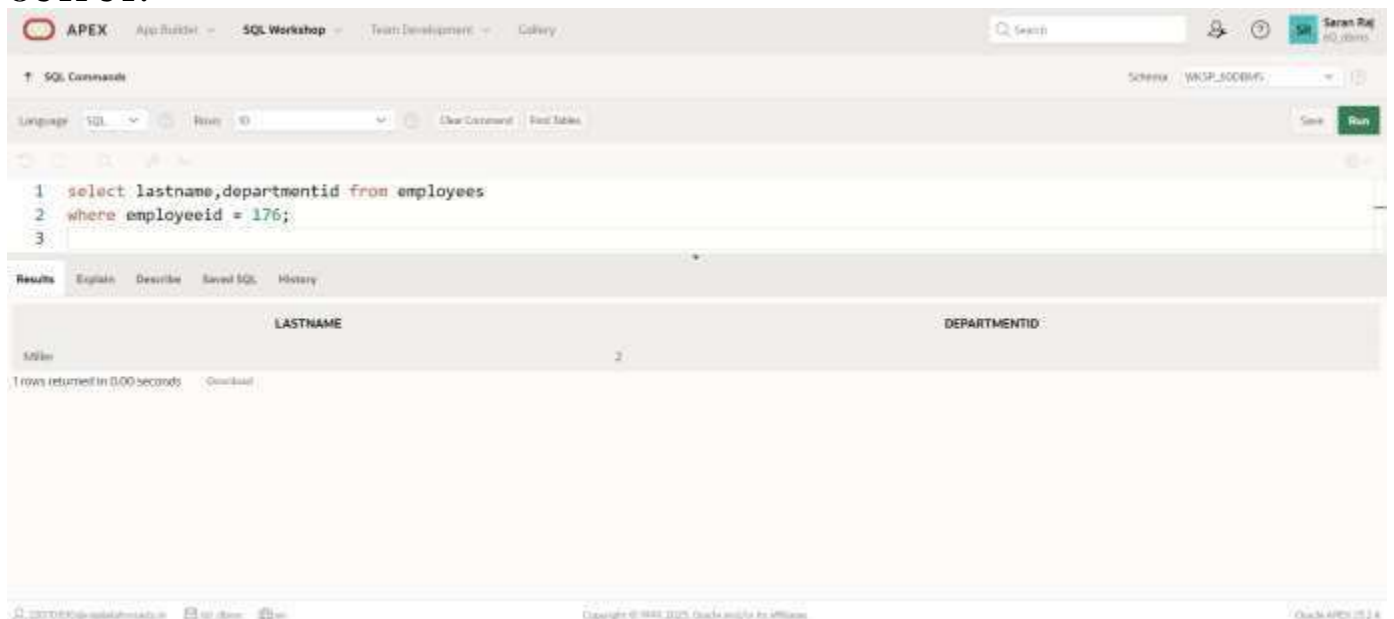
Below the table, it states "3 rows returned in 0.00 seconds" and provides a "Download" link.

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

QUERY:

```
select lastname,departmentid from employees
where employeeid = 176;
```

OUTPUT:



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

```
1 select lastname,departmentid from employees
2 where employeeid = 176;
3
```

The Results tab is selected, displaying a table with two columns: LASTNAME and DEPARTMENTID. The table contains one row of data:

LASTNAME	DEPARTMENTID
Miller	2

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

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 lastname, salary from employees
where salary not between 5000 and 12000;

OUTPUT:

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

```
1 select lastname, salary from employees
2 where salary not between 5000 and 12000;
3
```

The Results tab is selected, displaying a table with the following data:

LASTNAME	SALARY
Garcia	45000
Brown	90000
Johnson	65000

3 rows returned in 0.01 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 lastname,jobid,hiredate from employees where hiredate between '02-20-1998' and '05-01-1998' order by hiredate asc;

OUTPUT:

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

```
1 select lastname,jobid,hiredate from employees where hiredate between '02-20-1998' and '05-01-1998' order by hiredate asc;
```

The Results tab is selected, displaying a table with the following data:

LASTNAME	JOBID	HIREDATE
Brown	FINANCE	02/20/1998

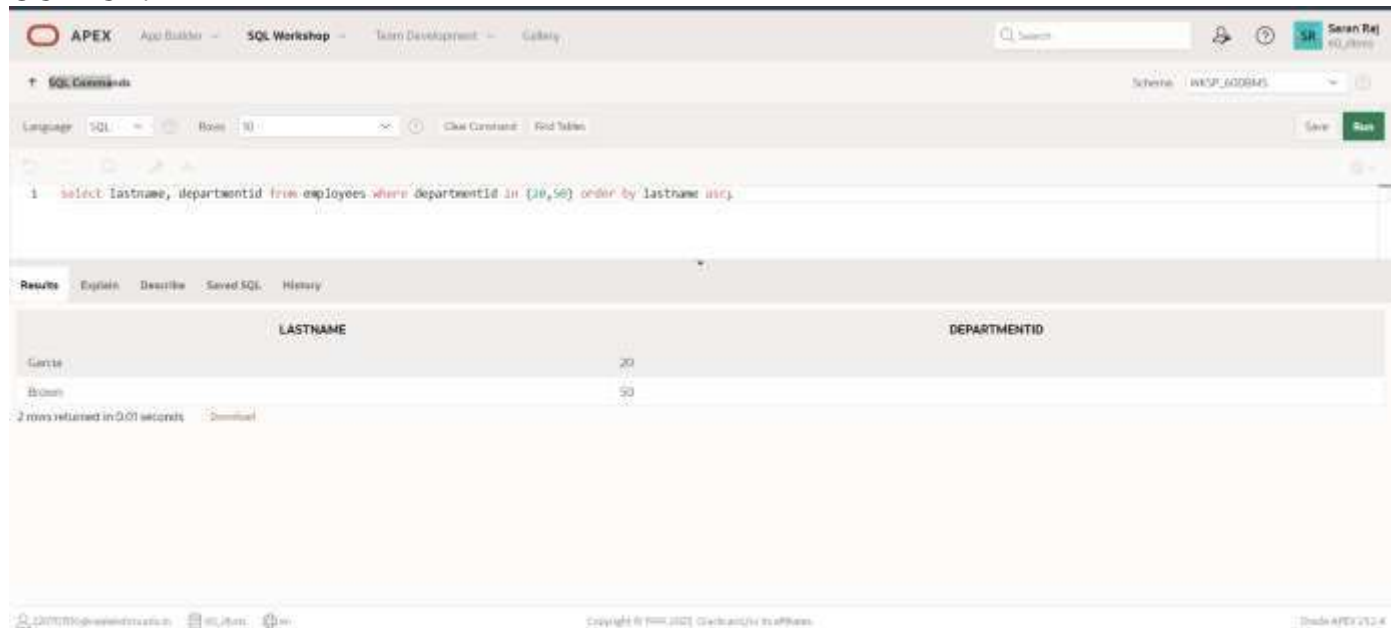
1 rows returned in 0.01 seconds

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 lastname, departmentid from employees where departmentid in (20,50) order by lastname asc;

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command is: `select lastname, departmentid from employees where departmentid in (20,50) order by lastname asc;`. The results are displayed in a table with two columns: LASTNAME and DEPARTMENTID. The results are: Gertz (20) and Brown (50). The status bar indicates "2 rows returned in 0.01 seconds".

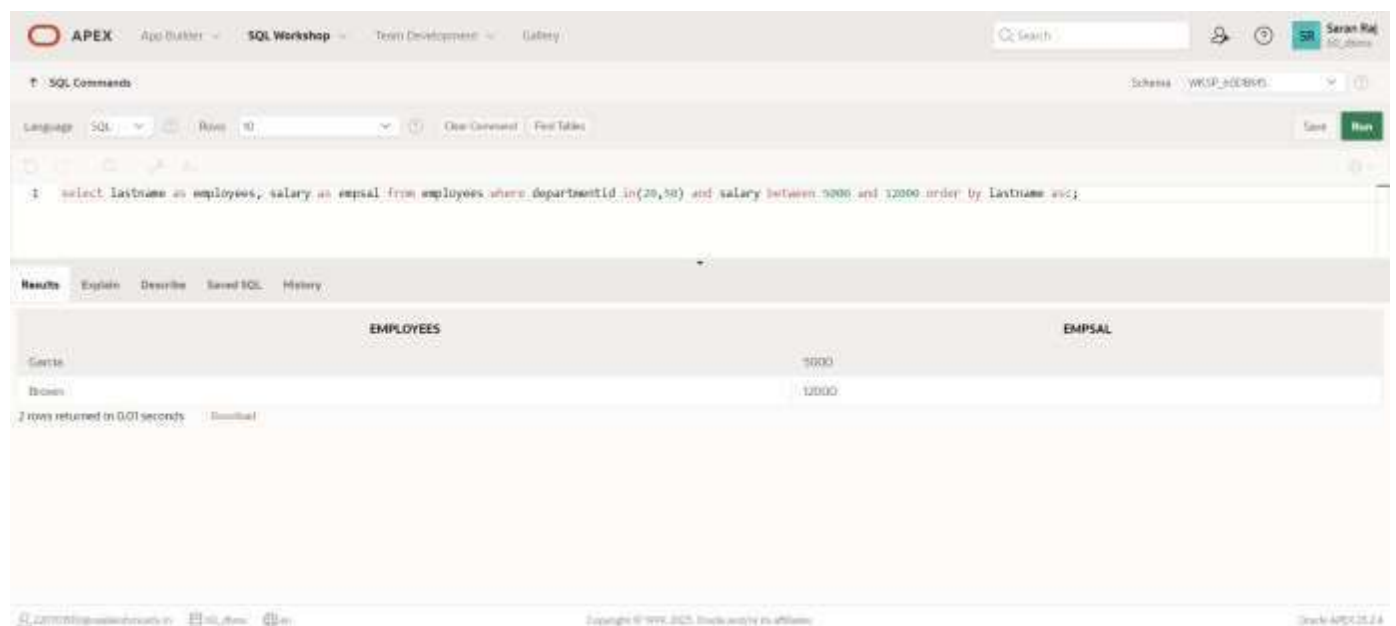
LASTNAME	DEPARTMENTID
Gertz	20
Brown	50

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 lastname as employees, salary as empsal from employees where departmentid in(20,50) and salary between 5000 and 12000 order by lastname asc;

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command is: `select lastname as employees, salary as empsal from employees where departmentid in(20,50) and salary between 5000 and 12000 order by lastname asc;`. The results are displayed in a table with two columns: EMPLOYEES and EMPSAL. The results are: Gertz (5000) and Brown (12000). The status bar indicates "2 rows returned in 0.01 seconds".

EMPLOYEES	EMPSAL
Gertz	5000
Brown	12000

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

QUERY:

select lastname, hiredate from employees where hiredate like '%1994';

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 'Saran Raj' are on the right. The 'SQL Commands' section shows the query: `select lastname, hiredate from employees where hiredate like '%1994';`. The 'Results' tab is active, displaying a table with two columns: 'LASTNAME' and 'HIREDATE'. The first row shows 'Johnson' and '12/31/1994'. Below the table, it states '1 rows returned in 0.00 seconds' and provides a 'Download' link. The footer includes 'Copyright © 1996, 2021 Oracle and/or its affiliates' and 'Oracle APEX 23.2.4'.

LASTNAME	HIREDATE
Johnson	12/31/1994

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

QUERY:

select lastname, jobid from employees where managerid is null;

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 'Saran Raj' are on the right. The 'SQL Commands' section shows the query: `select lastname, jobid from employees where managerid is null;`. The 'Results' tab is active, displaying a table with two columns: 'LASTNAME' and 'JOBID'. The first row shows 'Miles' and 'SALES'. Below the table, it states '1 rows returned in 0.01 seconds' and provides a 'Download' link. The footer includes 'Copyright © 1996, 2021 Oracle and/or its affiliates' and 'Oracle APEX 23.2.4'.

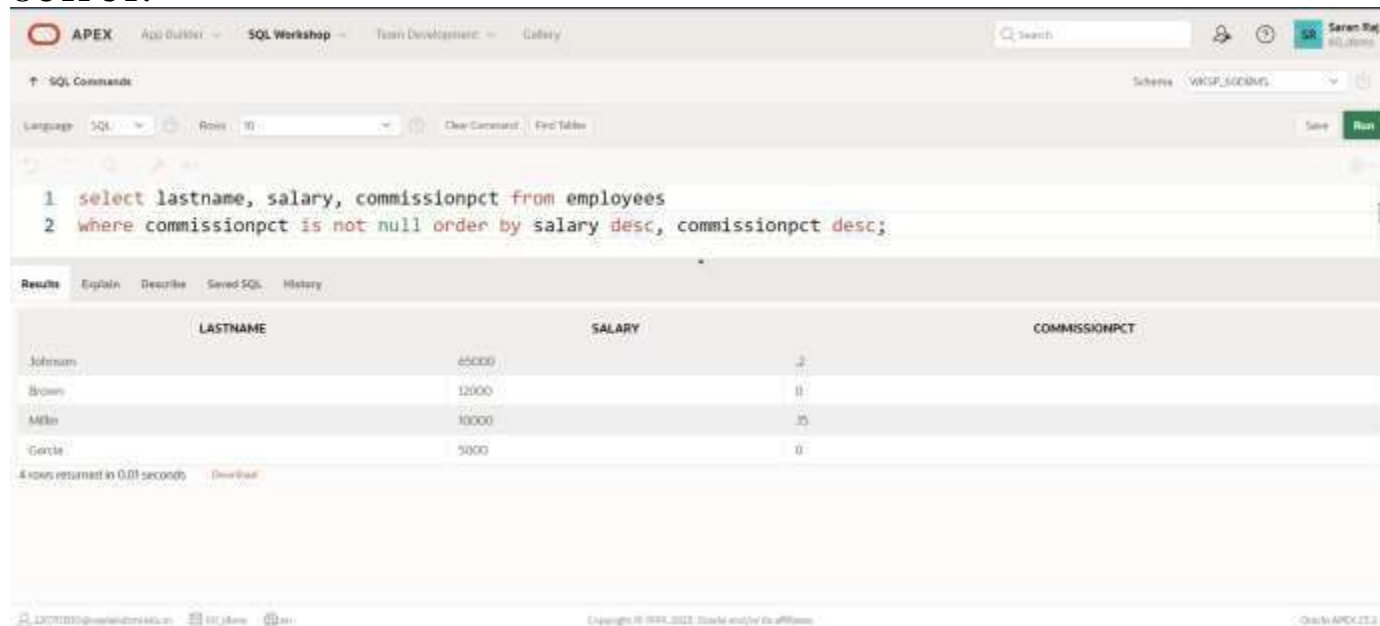
LASTNAME	JOBID
Miles	SALES

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 nul,orderby)

QUERY:

select lastname, salary, commissionpct from employees where commissionpct is not null order by salary desc, commissionpct desc;

OUTPUT:



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

```
1 select lastname, salary, commissionpct from employees
2 where commissionpct is not null order by salary desc, commissionpct desc;
```

The Results tab is selected, displaying a table with the following data:

LASTNAME	SALARY	COMMISSIONPCT
Johnson	65000	2
Brown	12000	0
Miller	10000	35
Garcia	5000	0

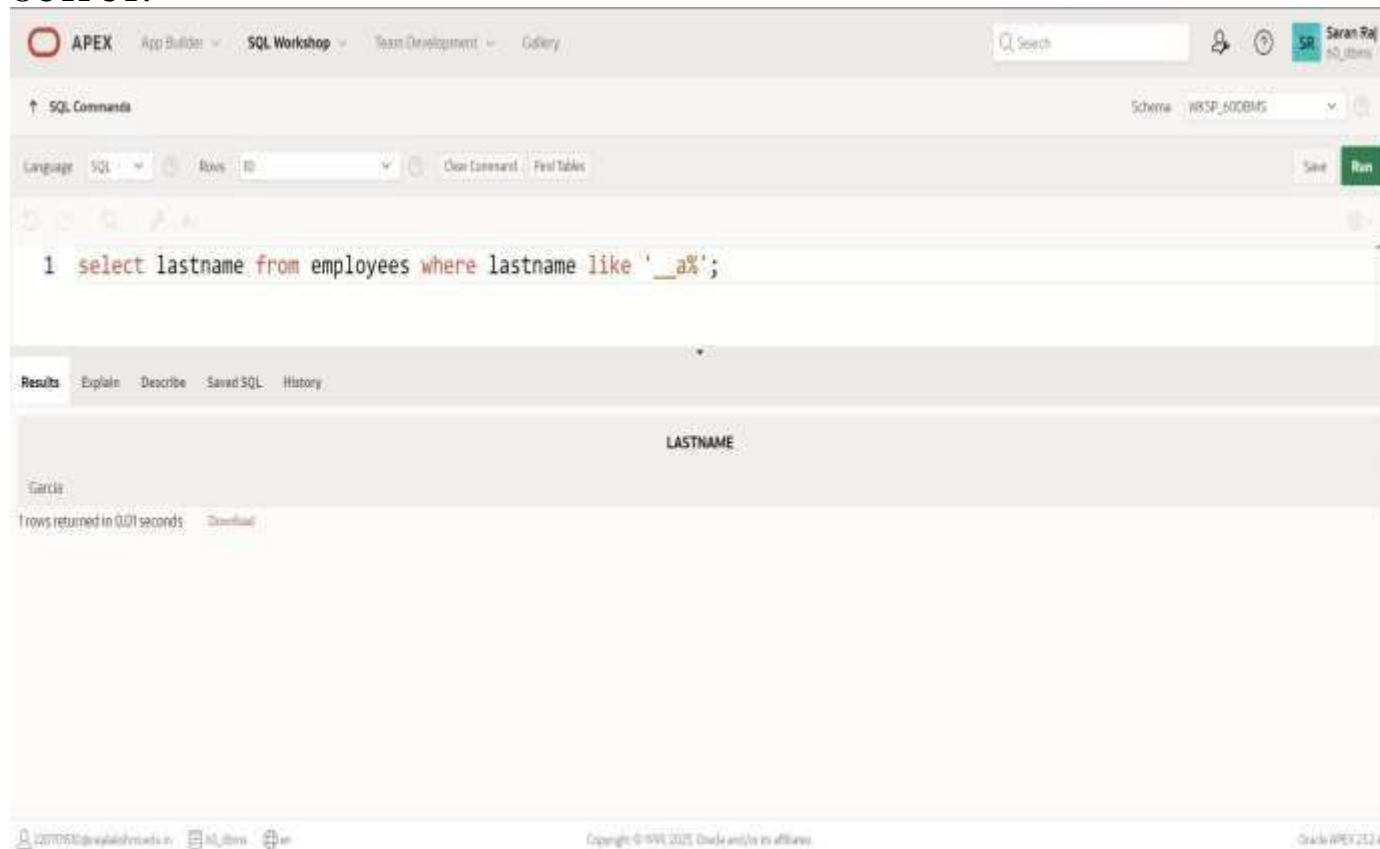
4 rows returned in 0.01 seconds. Download

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

QUERY:

select lastname from employees where lastname like '__a%';

OUTPUT:



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

```
1 select lastname from employees where lastname like '__a%';
```

The Results tab is selected, displaying a table with the following data:

LASTNAME
Garcia

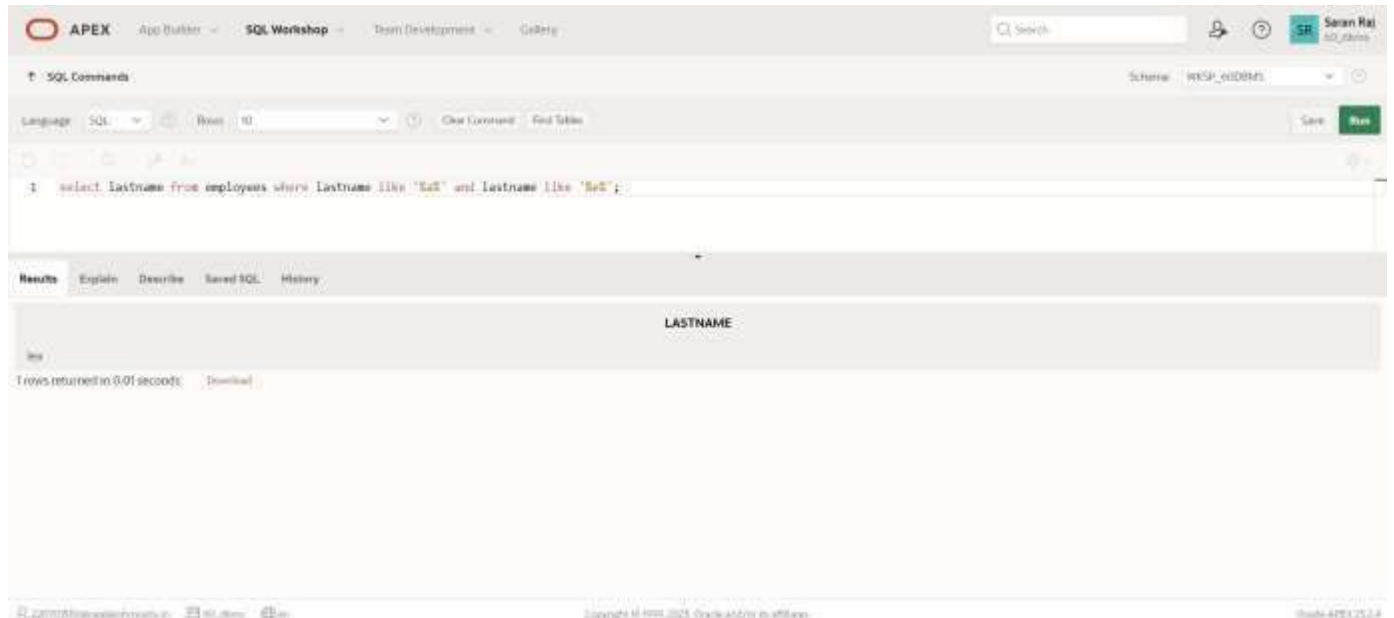
1 rows returned in 0.01 seconds. Download

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

QUERY:

select lastname from employees where lastname like '%a%' and lastname like '%e%';

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command entered is: `select lastname from employees where lastname like 'a%' and lastname like 'e%';`. The results are displayed in a table with the column header **LASTNAME**. The output shows the last name 'a'.

LASTNAME
a

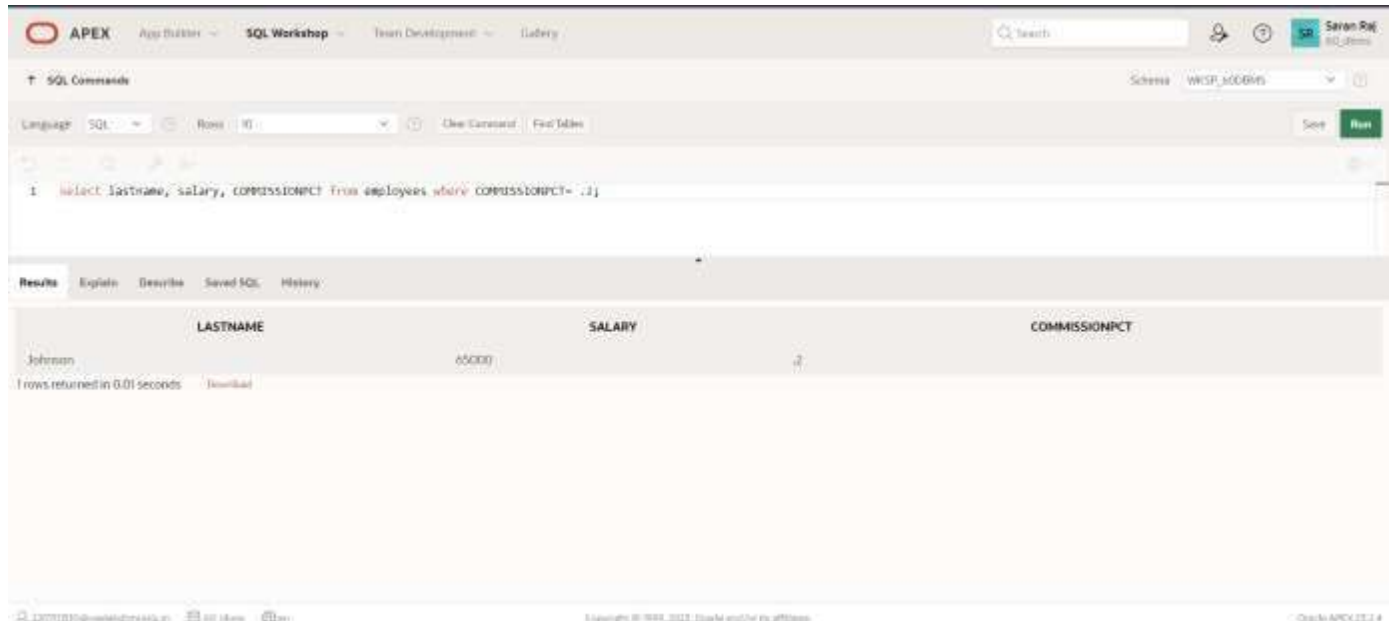
1 rows returned in 0.01 seconds

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

QUERY:

select lastname, salary, COMMISSIONPCT from employees where COMMISSIONPCT= .2;

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command entered is: `select lastname, salary, commissionpct from employees where commissionpct= .2;`. The results are displayed in a table with the column headers **LASTNAME**, **SALARY**, and **COMMISSIONPCT**. The output shows the last name 'Johnson', salary '65000', and commissionpct '0.2'.

LASTNAME	SALARY	COMMISSIONPCT
Johnson	65000	0.2

1 rows returned in 0.01 seconds

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

QUERY:

select lastname as employee, salary as empsal from employees where departmentid in (20,50) and salary between 5000 and 12000 order by lastname asc;

OUTPUT:

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

```
1 select lastname as employee, salary as empsal from employees where departmentid in (20,50) and salary between 5000 and 12000 order by lastname asc;
```

The Results tab is active, displaying the following data:

EMPLOYEE	EMPSAL
Brown	12000
Iba	5000

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

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

RESULT:

SINGLE ROW FUNCTIONS

EX.NO.6

DATE:

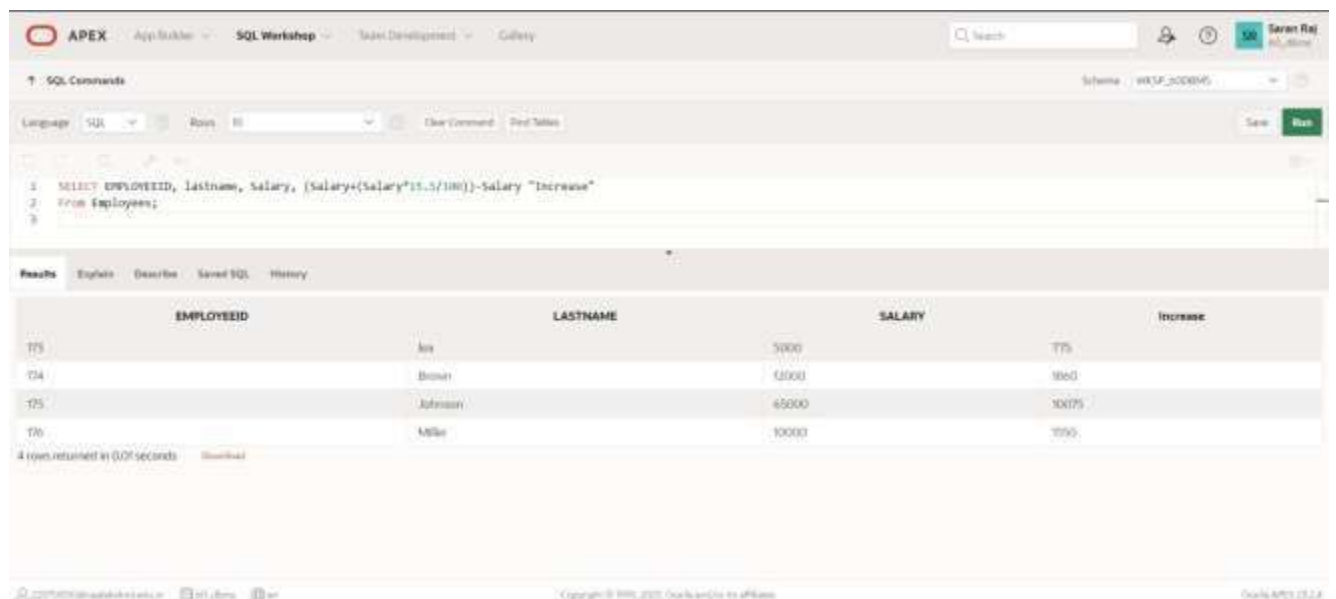
Find the Solution for the following:

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

QUERY:

SELECT SYSDATE AS "DATE" FROM DUAL;

OUTPUT:



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

```
1 SELECT EMPLOYEEID, lastname, salary, (Salary*(15.5/100))-Salary "Increase"
2 From Employees;
3
```

The results tab is active, displaying a table with 4 rows. The columns are EMPLOYEEID, LASTNAME, SALARY, and Increase. The data is as follows:

EMPLOYEEID	LASTNAME	SALARY	Increase
173	Kim	5000	775
174	Brown	12000	1860
175	Johnson	45000	70125
176	Miller	10000	1550

At the bottom of the results tab, it says "4 rows returned in 0.02 seconds" and there is a "Download" link.

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 EMPLOYEEID, LASTNAME, Salary, Salary+(15.5/100*Salary)
"NEW_SALARY"
From Employees;**

OUTPUT:

" ; " ; _____

Language: English

nw.t. 11ia'J. W...•J

© 2001 Blackwell Science Ltd
Journal of Internal Medicine 250: 111–117

88 88

Γ.

449

999

© 2000 Blackwell Science Ltd *Journal of Internal Medicine* 247: 399–405

1999, 2000).

2000

[illegible]

0000-0001-9100-0000

1120

1999

100

1999

[illegible]

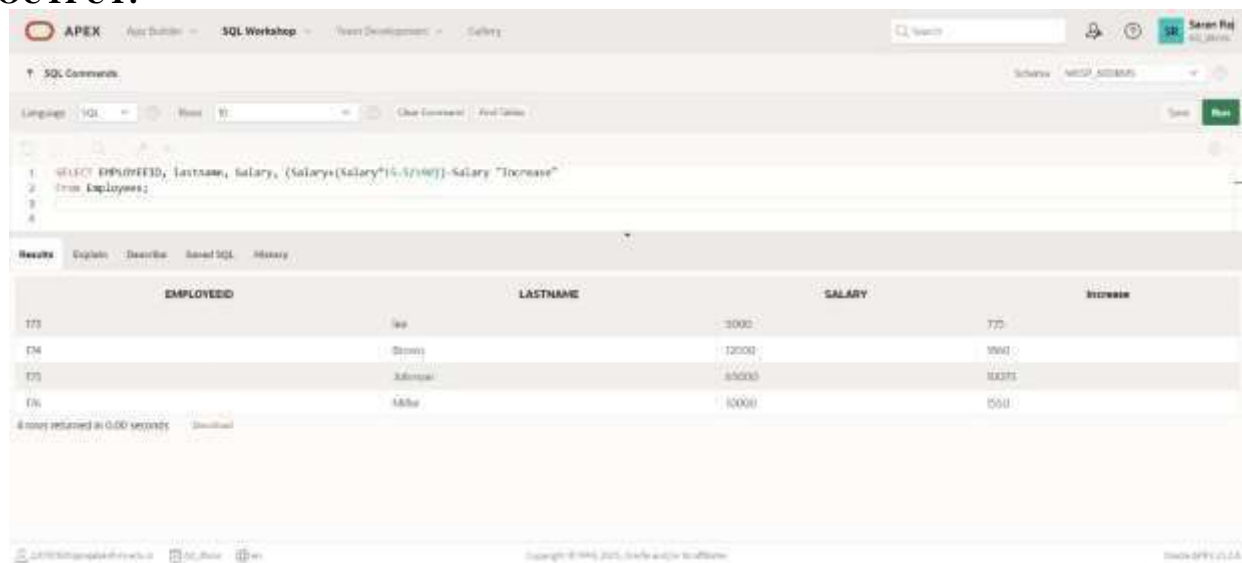
Copyright © 2001 John Wiley & Sons, Inc.

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 EMPLOYEEID, lastname, Salary, (Salary+(Salary*15.5/100))-Salary
"Increase"
From Employees;**

OUTPUT:



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

```
1. SELECT EMPLOYEEID, lastname, Salary, (Salary+(Salary*15.5/100))-Salary "Increase"
2. from Employees;
3.
4.
```

The Results tab is selected, displaying a table with the following data:

EMPLOYEEID	LASTNAME	SALARY	Increase
173	Deo	3000	775
174	Strom	12000	3960
175	Adams	8500	10075
176	Miller	10000	5500

4 rows returned in 0.00 seconds

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

QUERY:

**Select initcap(lastname) "Name", length(lastname) "Length of Name" from Employees
where lastname like 'J%' or lastname like 'A%' or lastname like 'M%' order by lastname;**

OUTPUT:



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

```
1. Select initcap(lastname) "Name", length(lastname) "Length of Name"
2. from Employees
3. where lastname like 'J%' or lastname like 'A%' or lastname like 'M%'
4. order by lastname;
```

The Results tab is selected, displaying a table with the following data:

Name	Length of Name
Joan	4
Miller	6

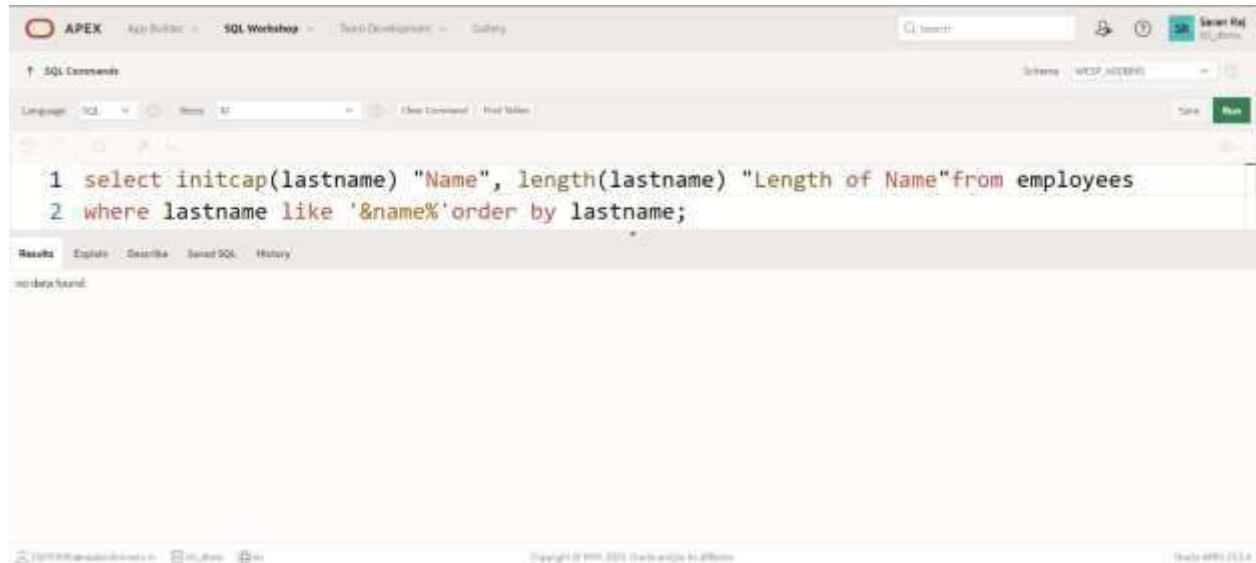
2 rows returned in 0.01 seconds

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(lastname) "Name", length(lastname) "Length of Name" from employees
where lastname like '&name%' order by lastname;**

OUTPUT:



**select initcap(lastname) "Name", length(lastname) "Length of Name" from employees
where lastname like 'H%' order by
lastname;**

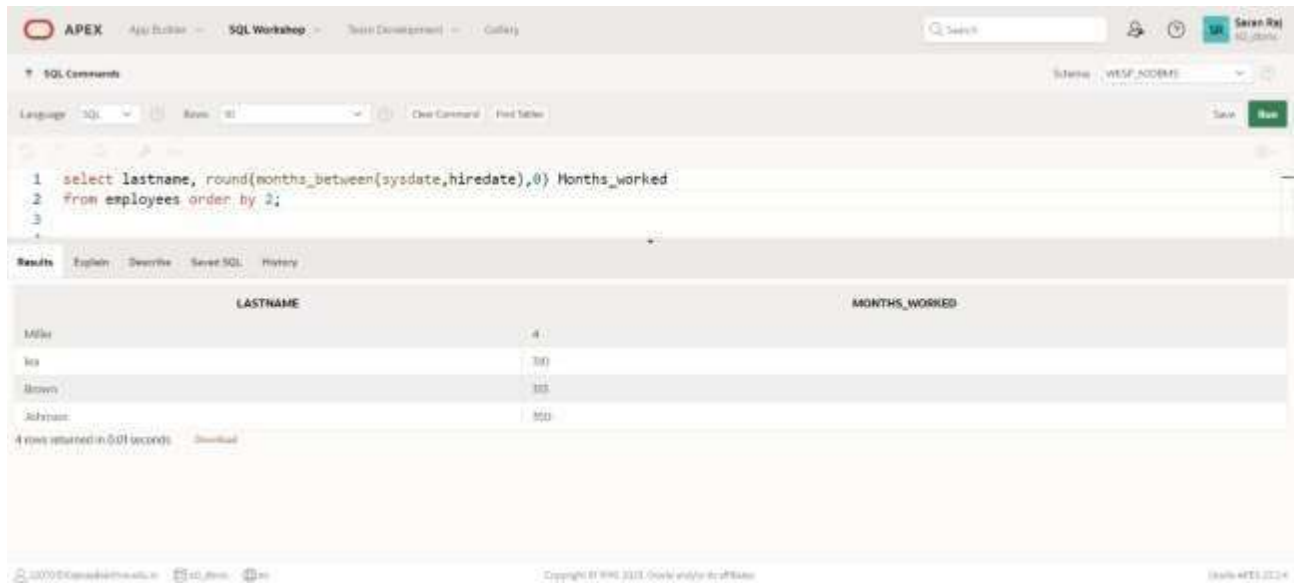


6. The HR department wants to find the length of employment for each employee. For each employee, display the last name and calculate the number of months between today and the date on which the employee was hired. Label the column MONTHS_WORKED. Order your results by the number of months employed. Round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

QUERY:

```
select lastname, round(months_between(sysdate,hiredate),0) Months_worked
from employees order by 2;
```

OUTPUT:



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

```
1 select lastname, round(months_between(sysdate,hiredate),0) Months_worked
2 from employees order by 2;
```

The results are shown in a table with two columns: LASTNAME and MONTHS_WORKED. The data is as follows:

LASTNAME	MONTHS_WORKED
Miller	4
Lee	10
Brown	15
Johnson	20

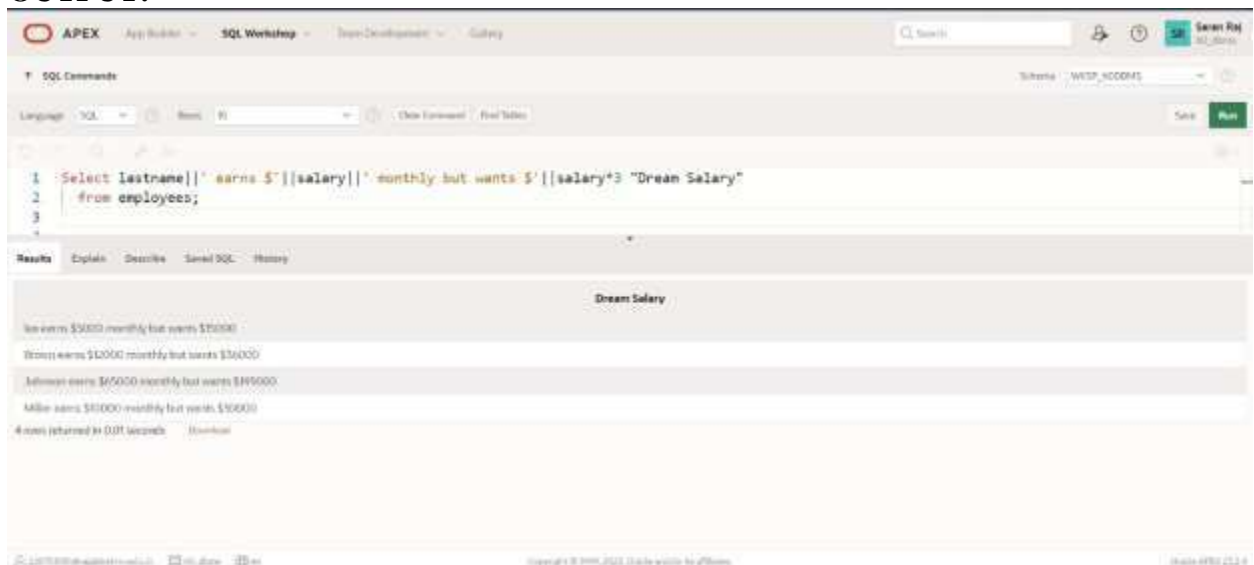
4 rows returned in 0.01 seconds. Download

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 lastname||' earns \$'||salary||' monthly but wants \$'||salary*3 'Dream Salary'
from employees;

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command entered is: `Select lastname||' earns $'||salary||' monthly but wants $'||salary*3 'Dream Salary' from employees;`. The results are displayed in a table with the column header "Dream Salary".

Dream Salary
lee earns \$5000 monthly but wants \$15000
Brown earns \$2000 monthly but wants \$6000
Johnson earns \$5000 monthly but wants \$15000
Miller earns \$10000 monthly but wants \$30000

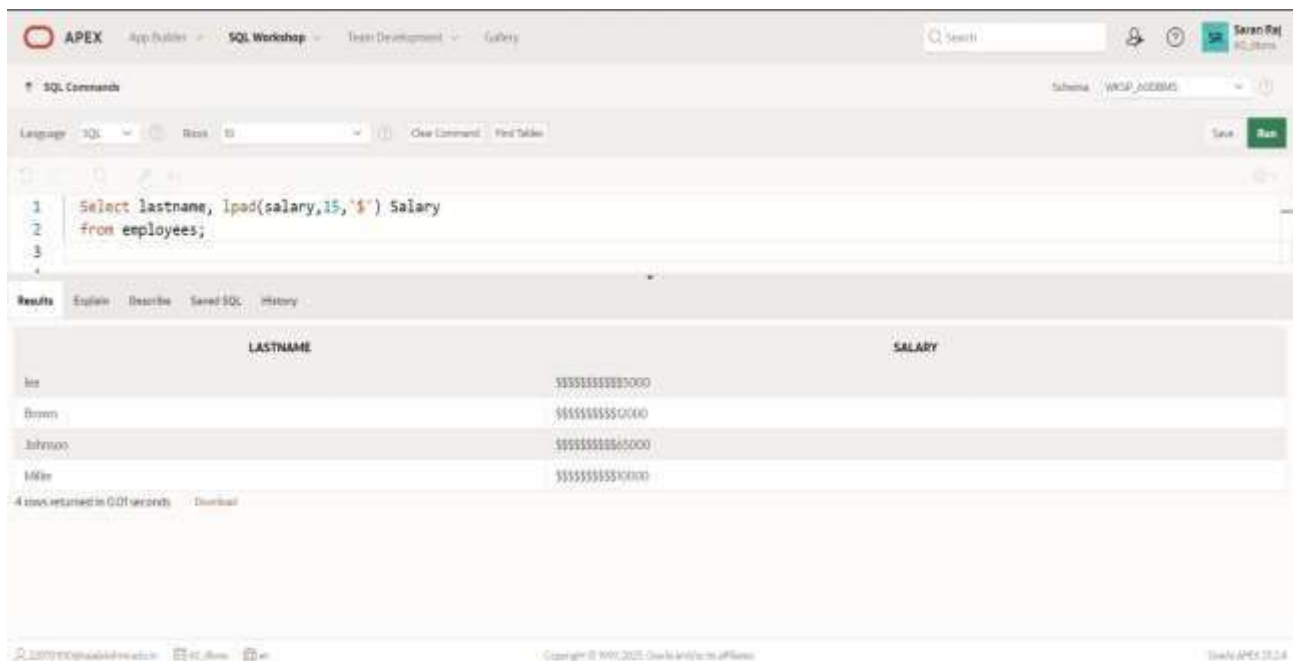
4 rows returned in 0.01 seconds

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 lastname, lpad(salary,15,'\$') Salary
from employees;

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command entered is: `Select lastname, lpad(salary,15,'$') Salary from employees;`. The results are displayed in a table with columns "LASTNAME" and "SALARY".

LASTNAME	SALARY
lee	\$500000000000000
Brown	\$200000000000000
Johnson	\$500000000000000
Miller	\$100000000000000

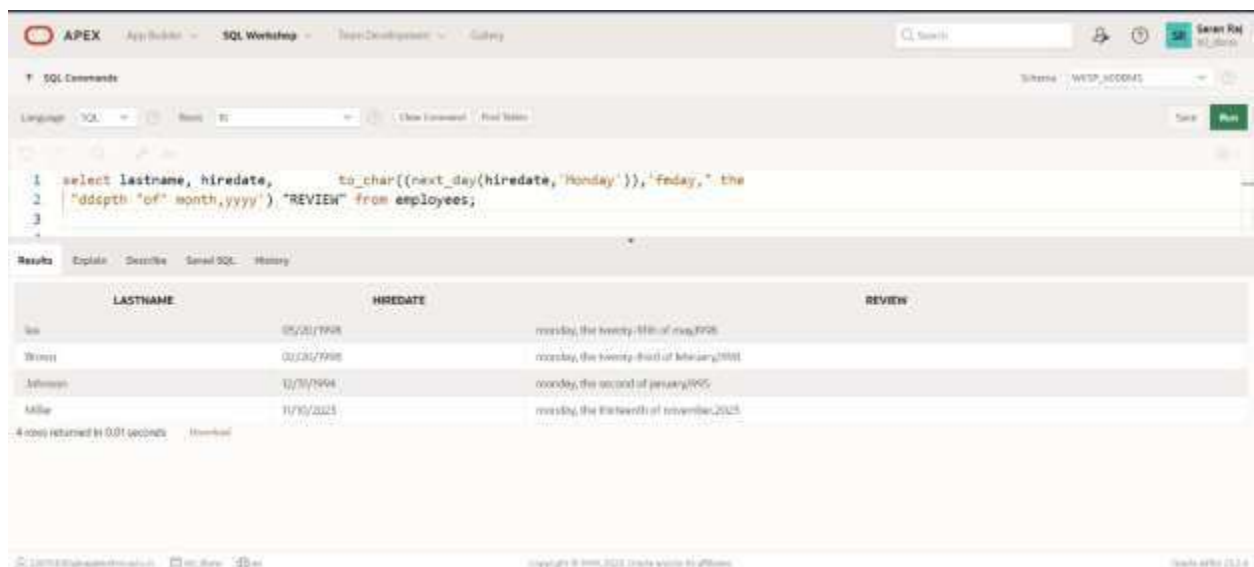
4 rows returned in 0.01 seconds

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 lastname, hiredate, to_char((next_day(hiredate,'Monday')), 'fmday, " the  
"ddspth "of" month,yyyy') "REVIEW" from employees;
```

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command is: `select lastname, hiredate, to_char((next_day(hiredate,'Monday')), 'fmday, " the "ddspth "of" month,yyyy') "REVIEW" from employees;` The results table has columns LASTNAME, HIREDATE, and REVIEW. It contains 4 rows of data.

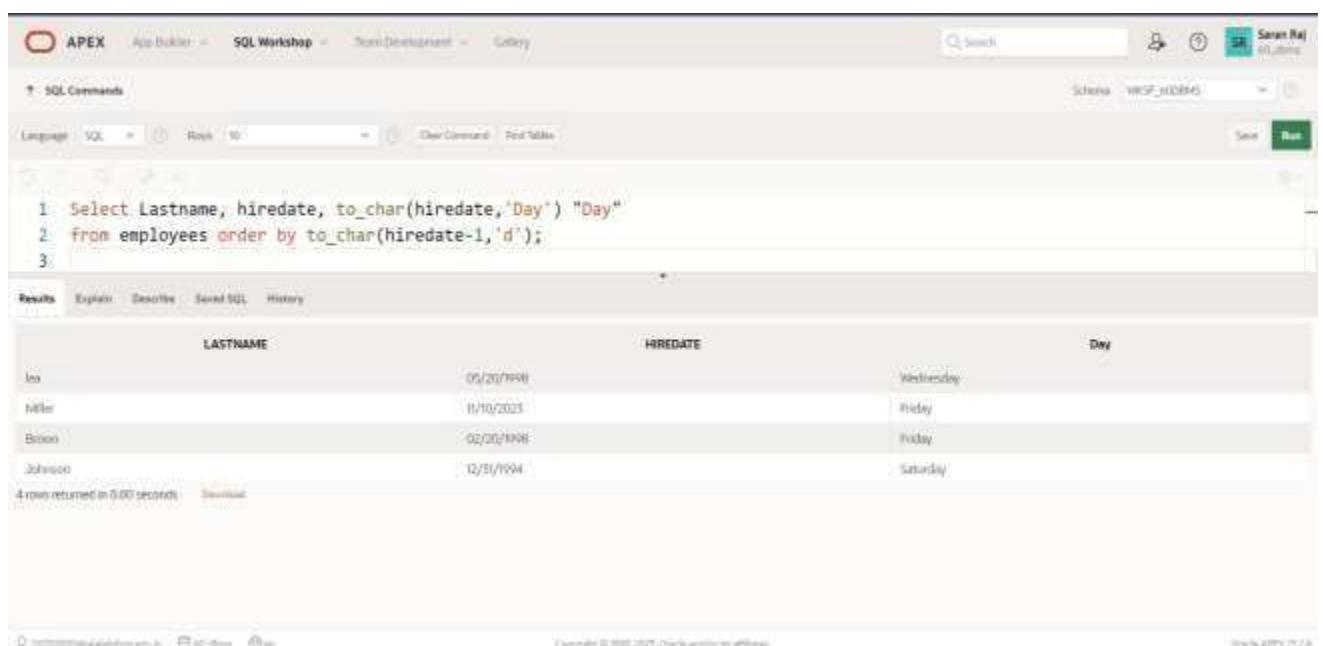
LASTNAME	HIREDATE	REVIEW
Isc	05/20/1998	monday, the twenty-fifth of may,1998
Brown	02/20/1998	monday, the twenty-third of february,1998
Johnson	12/31/1994	monday, the second of january,1995
Miller	11/10/2023	monday, the thirteenth of november,2023

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 Lastname, hiredate, to_char(hiredate,'Day') "Day"  
from employees order by to_char(hiredate-1,'d');
```

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command is: `Select Lastname, hiredate, to_char(hiredate,'Day') "Day" from employees order by to_char(hiredate-1,'d');` The results table has columns LASTNAME, HIREDATE, and Day. It contains 4 rows of data, ordered by the day of the week.

LASTNAME	HIREDATE	Day
Isc	05/20/1998	Wednesday
Miller	11/10/2023	Friday
Brown	02/20/1998	Friday
Johnson	12/31/1994	Saturday

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

RESULT:

DISPLAYING DATA FROM MULTIPLE TABLES

EX_NO:7

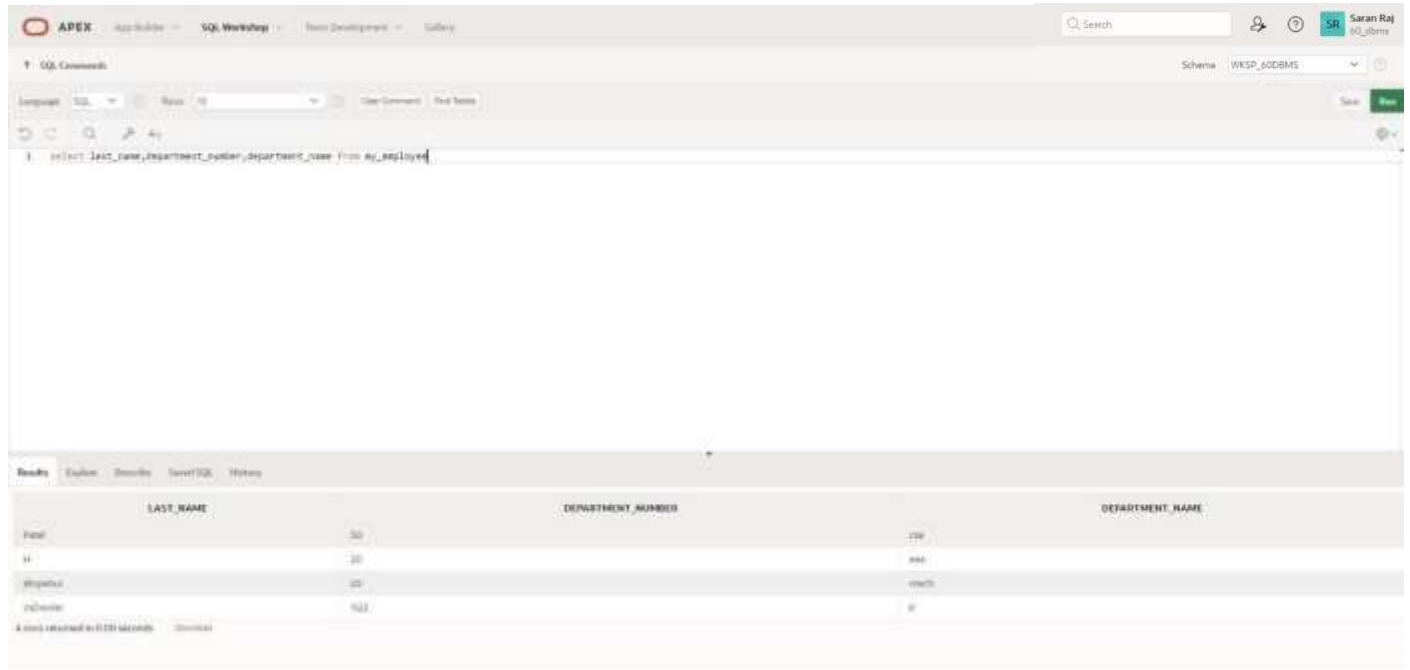
DATE:

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

QUERY:

Select last_name,department_name, department_number from my_employee

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command window contains the query: `select last_name,department_name, department_number from my_employee`. The results window displays a table with three columns: LAST_NAME, DEPARTMENT_NUMBER, and DEPARTMENT_NAME. The data is as follows:

LAST_NAME	DEPARTMENT_NUMBER	DEPARTMENT_NAME
Patel	50	ITP
H	20	HR
Wong	20	HR
Chen	50	ITP

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_code ,department_location_id from my_employees ,departments d where deptment _ number like '%80%';

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL command window contains the query: `select distinct job_code,department_location_id from my_employees ,departments d where department_number like '%80%'`. The results window displays a table with two columns: JOB_CODE and DEPARTMENT_LOCATION. The data is as follows:

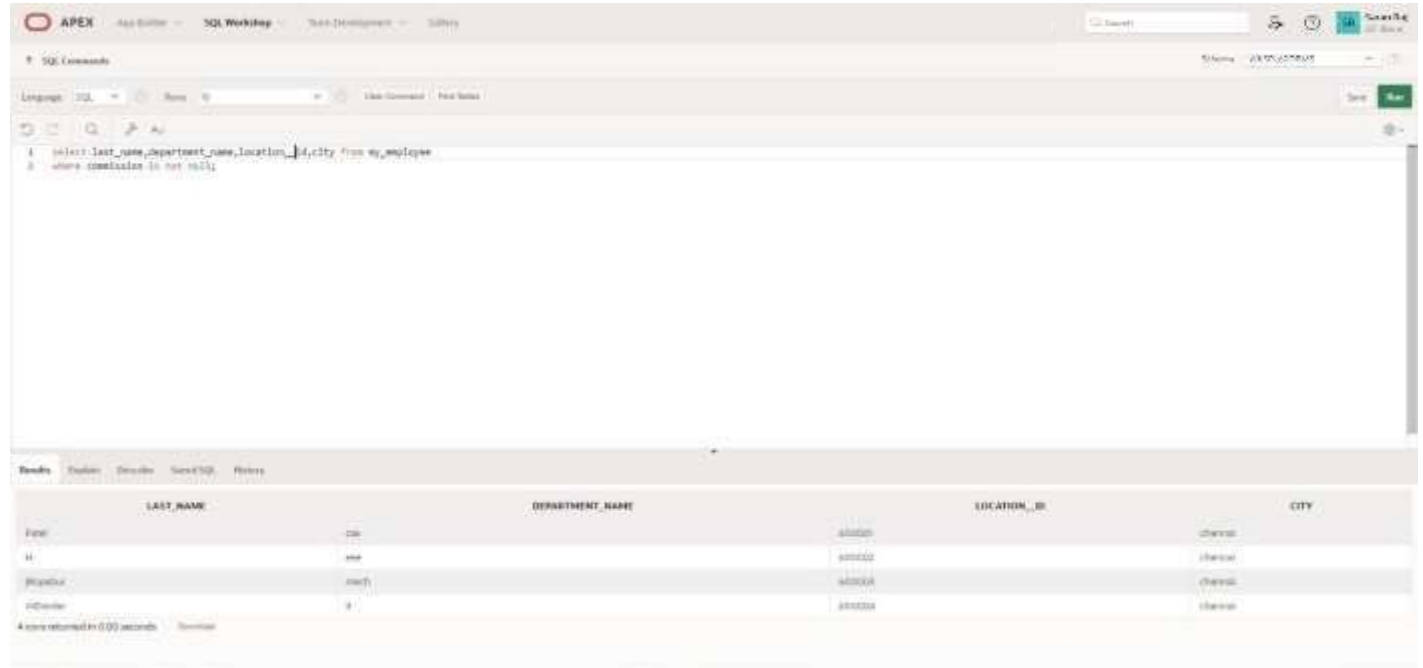
JOB_CODE	DEPARTMENT_LOCATION
job	dept
dept	job

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

QUERY:

Select e.last_name,d.dept_name,d.location_id,l.city from employees e,departments d,locations l
where e.dept_id = d.dept_id and d.location_id=location_id and e.commission_pct is not null;

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top bar includes the APEX logo and navigation tabs: App Builder, SQL Workshop (selected), Task Development, and Gallery. The SQL Commands tab is active, displaying a query in the editor:

```
1 select last_name,department_name,location_id,city from emp_employees
2 where commission_pct is not null;
```

Below the editor, the Results tab shows the query output as a table with 4 rows and 5 columns: LAST_NAME, DEPARTMENT_NAME, LOCATION_ID, and CITY. The data is as follows:

LAST_NAME	DEPARTMENT_NAME	LOCATION_ID	CITY
Patel	HR	10000	Chennai
H	HR	10002	Chennai
Popoju	HR	10003	Chennai
Chen	HR	10004	Chennai

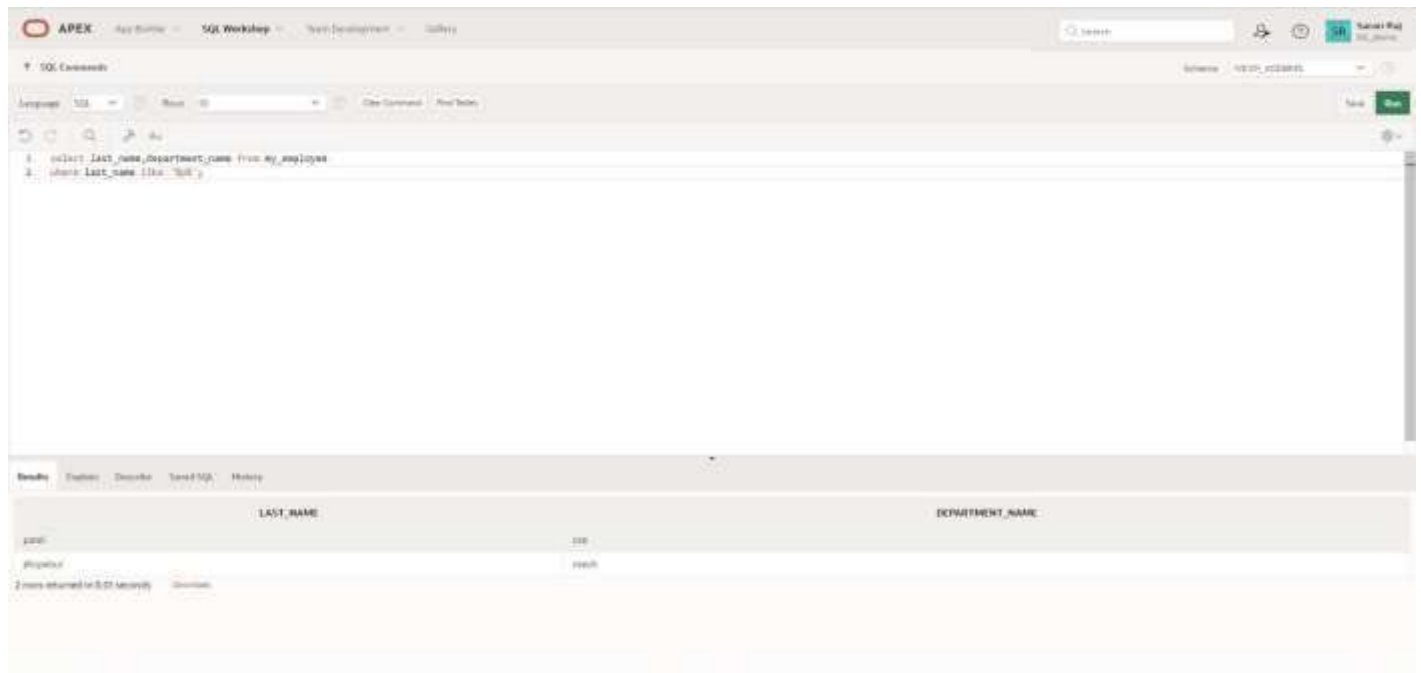
At the bottom of the results section, it states "4 rows returned in 0.00 seconds" with a "Download" link.

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

QUERY:

Select last_name,dept_name from employees,departments where employees.dept_id=departments.dept_id
And last_name like '%a%';

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Tools & Resources', and 'Gallery'. The 'SQL Commands' tab is active, displaying a query with two lines: '1. select last_name,department_name from employees' and '2. where last_name like '%a%''. The 'Results' tab is selected, showing a table with two columns: 'LAST_NAME' and 'DEPARTMENT_NAME'. The table contains two rows: 'paul' in the 'LAST_NAME' column and 'IT' in the 'DEPARTMENT_NAME' column. Below the table, it indicates '2 rows returned in 0.07 seconds' and provides a 'Download' link.

LAST_NAME	DEPARTMENT_NAME
paul	IT
phelpst	person

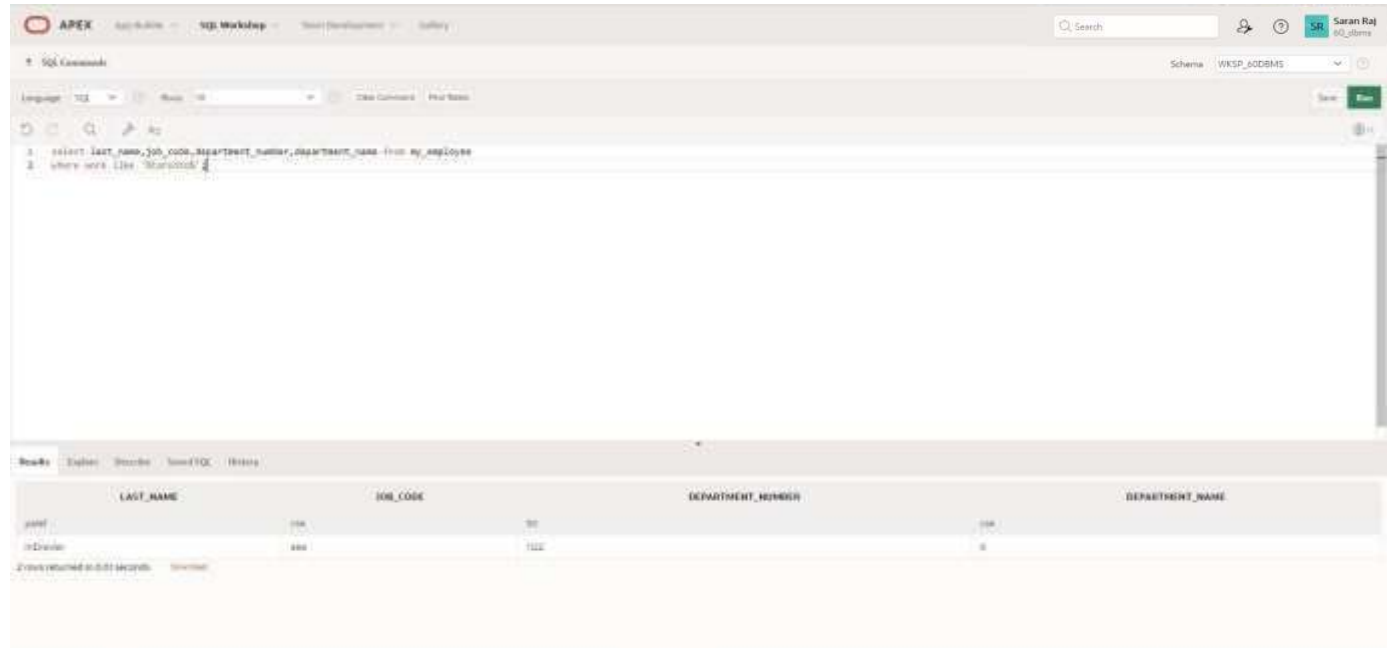
2 rows returned in 0.07 seconds [Download](#)

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

QUERY:

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

OUTPUT:



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

```
1 select last_name,job_code,department_number,department_name from emp_employee
2 where work_site = 'toronto';
```

The results window displays the following data:

LAST_NAME	JOB_CODE	DEPARTMENT_NUMBER	DEPARTMENT_NAME
jeff	ITA	90	IT
Neenan	ABA	100	HR

2 rows returned in 0.01 seconds.

APEX SQL Workshop interface showing a query and its results.

```

1 SELECT e.last_name "Employee", m.emp_id "EMP#",
2       m.last_name "Manager", m.emp_id "Mgr#",
3       FROM employees e
4       LEFT OUTER JOIN employees m
5       ON (m.manager_id = e.emp_id);
6
7

```

Results:

Employee	EMP#	Manager	Mgr#
Neena	5	Neena	2
Lex	6	Neena	2
Neena	2	Neena	1

3 rows returned in 0.05 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 dept_id department,e.last_name employee,c.last_name colleague from employees e join employees c
On (e.deot_id=c.dept_id) where e.emp_id<>c.emp_id order by e.dept_id,e.last_name,c.last.name;

OUTPUT:

APEX SQL Workshop interface showing a query and its results.

```

1 select e.dept_id department,e.last_name employee,c.last_name colleague from employees e join employees c
2 on (e.dept_id=c.dept_id) where e.emp_id<>c.emp_id order by e.dept_id,e.last_name,c.last.name;
3

```

Results:

DEPARTMENT_NUMBER	LAST_NAME	DEPARTMENT_NUMBER
100	Neena	100
100	Lex	100
100	Neena	100
100	Neena	100

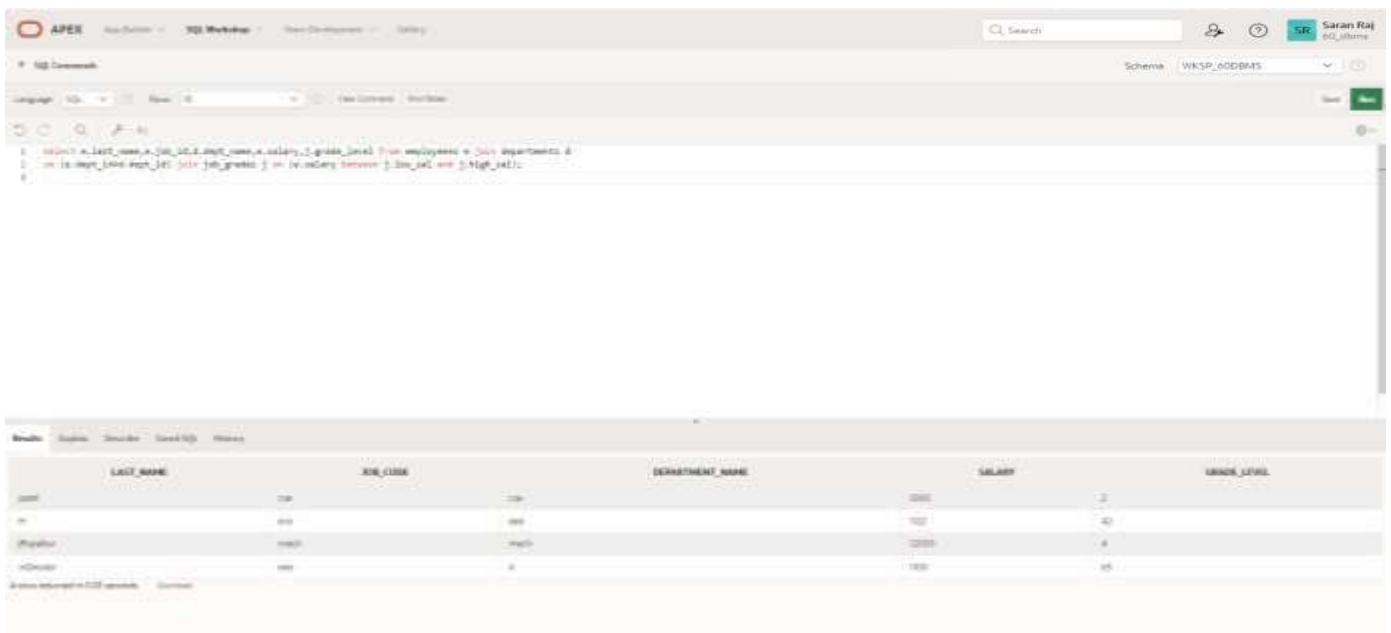
4 rows returned in 0.05 seconds

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 employees e join departments d on (e.dept_id=d.dept_id) join job_grades j on (e.salary between j.low_sal and j.high_sal);

OUTPUT:



The screenshot shows the SQL Developer interface. The top toolbar includes buttons for 'Run', 'Save', 'Commit', and 'Rollback'. The 'SQL Command' window contains the following query:

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

The 'Results' window displays the output of the query. The table has five columns: LAST_NAME, JOB_ID, DEPARTMENT_NAME, SALARY, and GRADE_LEVEL. The data is as follows:

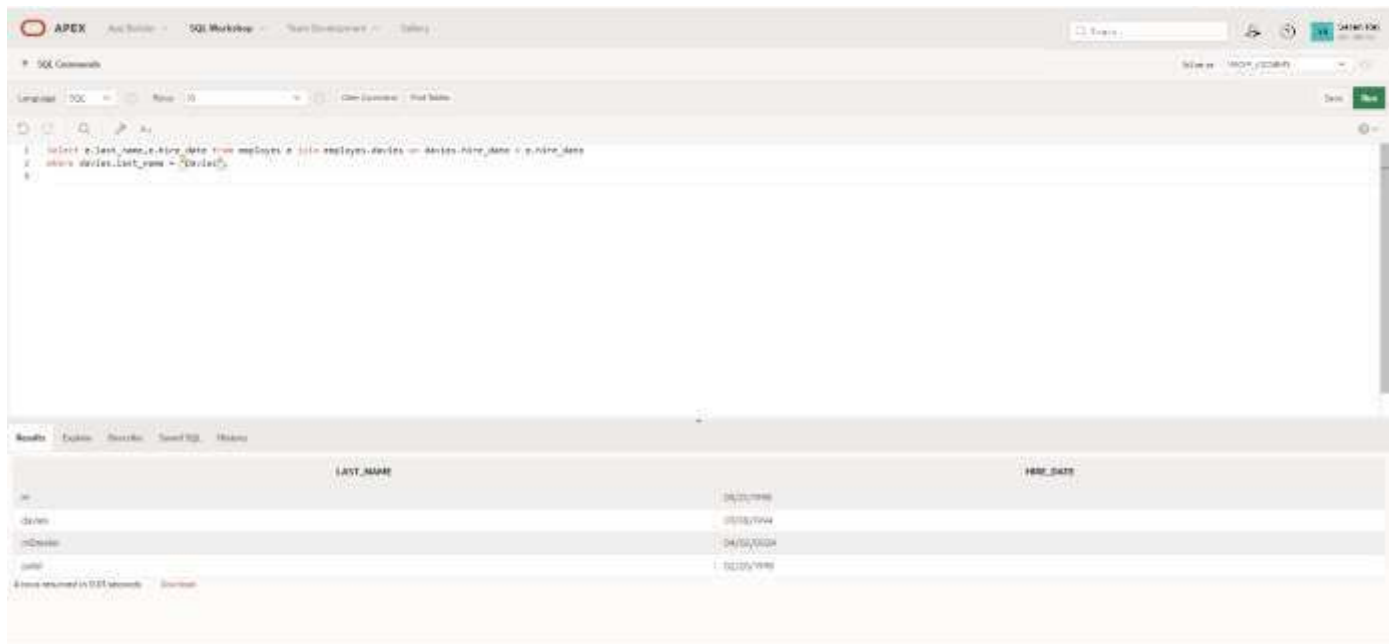
LAST_NAME	JOB_ID	DEPARTMENT_NAME	SALARY	GRADE_LEVEL
Scott	ANALYST	ACCOUNTING	3000	3
Kim	STOCKER	WAREHOUSE	1100	1
Phyllis	SALESREP	SALES	1200	2
Chen	CLERK	ADMINISTRATIVE	1900	2

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

QUERY:

Select e.last_name,e.hire_date from employees e join employees.davies on davies.hire_date < e.hire_date
Where davies.last_name = 'Davies';

OUTPUT:



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

```
1 select e.last_name,e.hire_date from employees e join employees.davies on davies.hire_date < e.hire_date
2 where davies.last_name = 'Davies';
3
```

The results are displayed in a table with two columns: LAST_NAME and HIRE_DATE. There are four rows of data.

LAST_NAME	HIRE_DATE
en	24/03/1990
enDavies	25/03/1994
enDavies	24/03/2024
enDavies	25/03/1992

At the bottom, it indicates "4 rows returned in 0.01 seconds" and provides a "Download" link.

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 last-name as Employee,e.hire_date as Emp_hired,e.manager_name as manager,m.hire_date as mgr_hired from my_employee 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

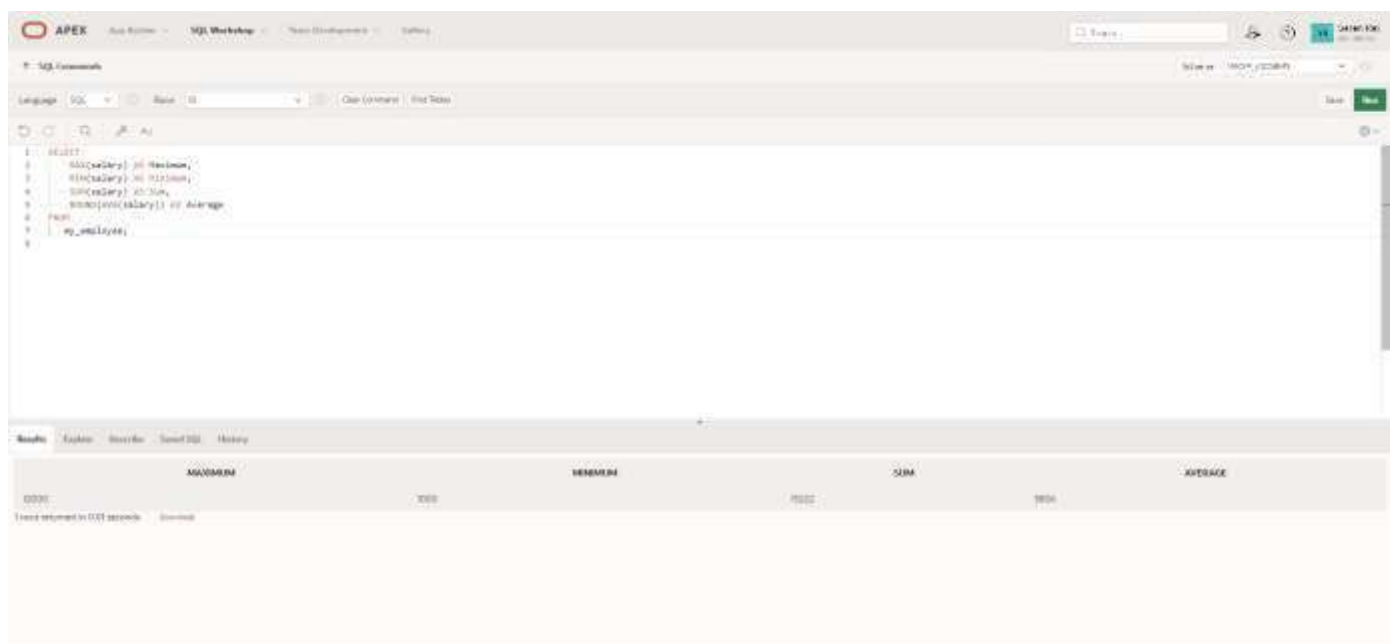
QUERY:

SELECT

MAX(salary) AS Maximum, MIN(salary) AS Minimum,

SUM(salary) AS Sum, ROUND(AVG(salary)) AS Average FROM my_employee;

OUTPUT:



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

```
1 SELECT
2   MAX(salary) AS Maximum,
3   MIN(salary) AS Minimum,
4   SUM(salary) AS Sum,
5   ROUND(AVG(salary)) AS Average
6 FROM
7   my_employee;
```

The output is displayed in a table with the following columns: MAXIMUM, MINIMUM, SUM, and AVERAGE. The data is as follows:

MAXIMUM	MINIMUM	SUM	AVERAGE
10000	1000	100000	10000

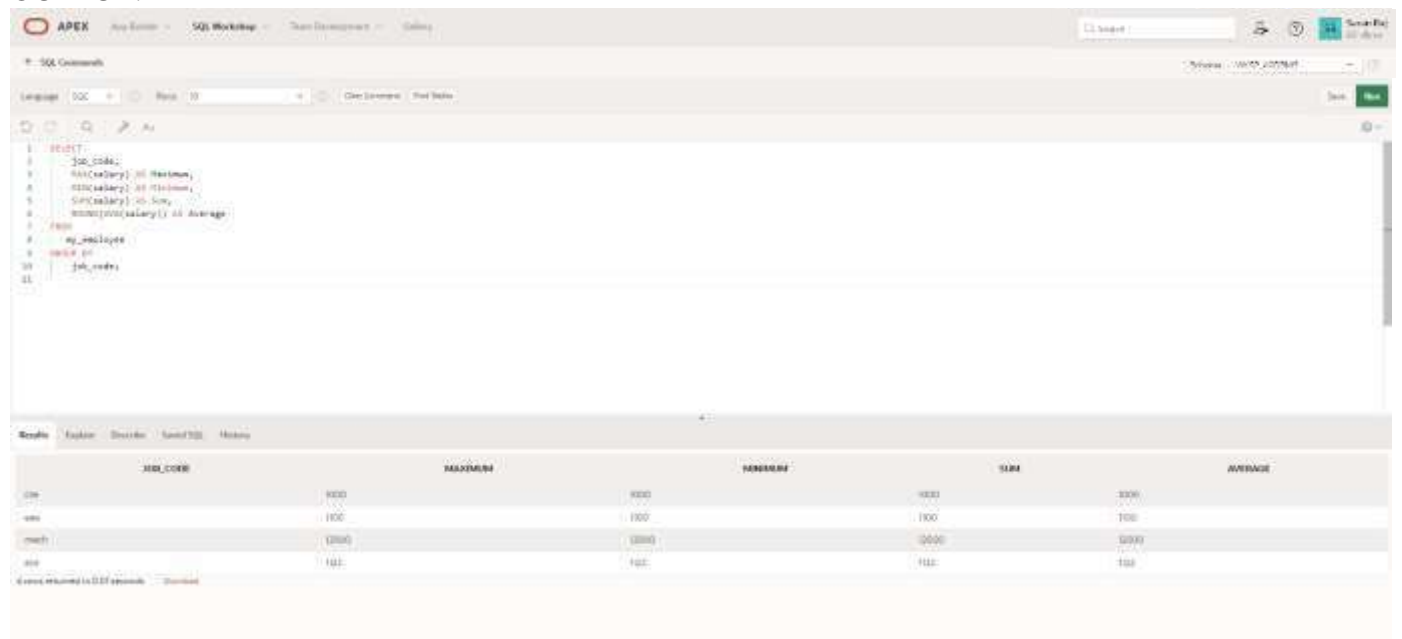
5. Modify the above query to display the minimum, maximum, sum, and average salary for each job type.

QUERY:

SELECT

```
job_code, MAX(salary) AS Maximum, MIN(salary) AS Minimum, SUM(salary) AS Sum  
ROUND(AVG(salary)) AS Average FROM my_employee GROUP BY job_code;
```

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The top section displays the SQL query being executed. The bottom section shows the results of the query in a table format.

```
1 SELECT  
2   job_code,  
3   MAX(salary) AS Maximum,  
4   MIN(salary) AS Minimum,  
5   SUM(salary) AS Sum,  
6   ROUND(AVG(salary)) AS Average  
7 FROM  
8   my_employee  
9 GROUP BY  
10  job_code;
```

JOB_CODE	MAXIMUM	MINIMUM	SUM	AVERAGE
clerk	9000	3000	30000	3000
anal	1000	500	1500	500
manag	12000	8000	20000	8000
pres	14000	1000	15000	1500

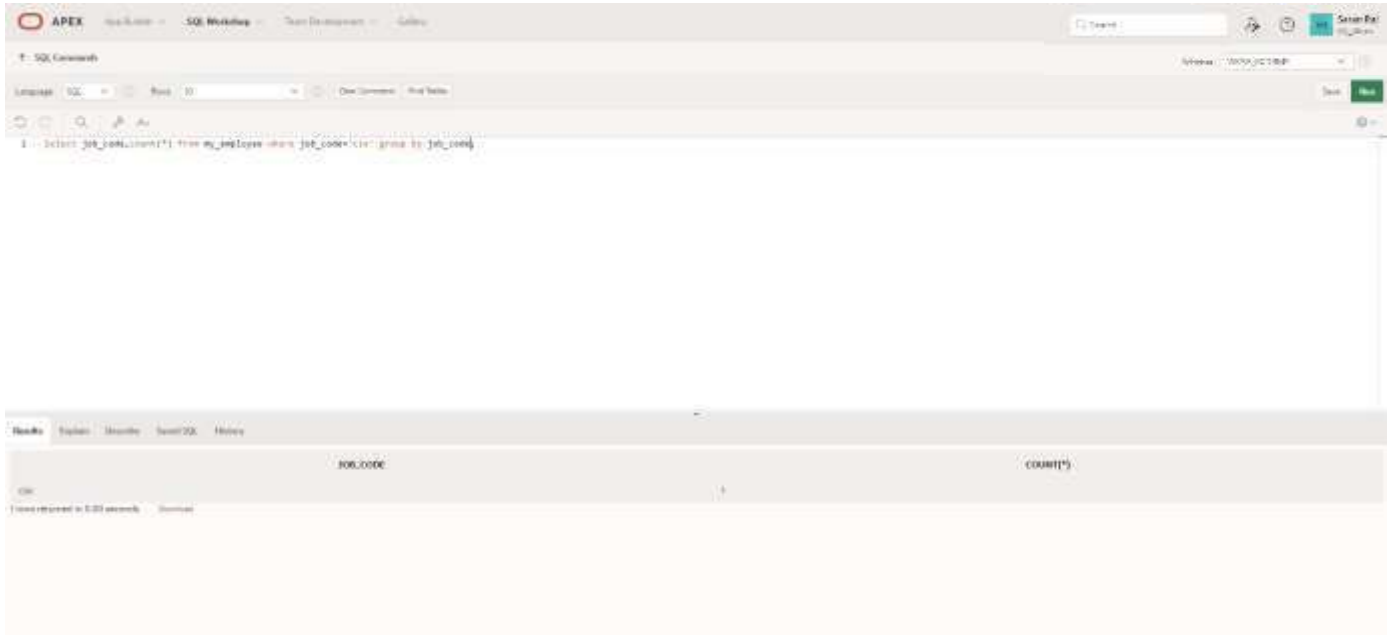
4 rows returned in 0.07 seconds

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_code, count(*) from my_employee where job_code='cse' group by job_code;
```

OUTPUT:



The screenshot shows the APEX SQL Worksheet interface. At the top, there's a navigation bar with 'APEX', 'SQL Worksheet', 'SQL', 'SQL Worksheet', 'SQL Worksheet', and 'SQL Worksheet'. Below this, there's a search bar and a 'Run' button. The main area contains a SQL query: `SELECT job_name, count(*) FROM my_employee WHERE job_code = 'CLERK' GROUP BY job_name;`. Below the query, there's a table with two columns: 'JOB_NAME' and 'COUNT(*)'. The table has one row with the value 'CLERK' in the first column and '1' in the second column. At the bottom, there's a status bar indicating '1 rows returned in 0.00 seconds' and a 'Download' button.

JOB_NAME	COUNT(*)
CLERK	1

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

QUERY:

SELECT COUNT(DISTINCT manager_id) AS "Number of managers" FROM my_employee;

OUTPUT:

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

```
1 SELECT COUNT(DISTINCT manager_id) AS "Number of managers" FROM my_employees;
```

The query is executed, and the results are displayed in a table with the following structure:

Number of managers
4

Below the table, it indicates "1 rows returned in 0.02 seconds".

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

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

OUTPUT:

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

```
1 select max(salary)-min(salary) difference from my_employees;
```

The query is executed, and the results are displayed in a table with the following structure:

DIFFERENCE
10000

Below the table, it indicates "1 rows returned in 0.02 seconds".

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 my_employee 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 query editor contains the following SQL code:

```
1 select manager_id,min(salary) from my_employee where manager_id is not null group by manager_id  
2 having min(salary)>6000 order by min(salary) desc  
3
```

The Results tab is active, displaying a table with two columns: **MANAGER_ID** and **MIN(SALARY)**. The table contains one row of data:

MANAGER_ID	MIN(SALARY)
100	11000

Below the table, it indicates "1 row returned in 0.03 seconds" and provides a "Download" link.

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

QUERY:

```
Select count(*) as  
total,sum(decode(to_char(hire_date,'YYYY'),1995,1,0))"1995",sum(decode(to_char(hire_date,'YYYY'),1996,1,0))
```


"1996"sum(decode(to_char (hire_date, 'YYYY'),1997,1,0))"1997" sum(decode(to_char (hire_date, 'YYYY'),1998,1,0))"1998" from emp;

OUTPUT:



The screenshot shows an SQL IDE interface. The top panel displays the following SQL query:

```
1 select count(*) as total,sum(decode(to_char(hire_date, 'YYYY'),1997,1,0))"1996",
2 sum(decode(to_char(hire_date, 'YYYY'),1996,1,0))"1997",
3 sum(decode(to_char(hire_date, 'YYYY'),1995,1,0))"1998",
4 sum(decode(to_char(hire_date, 'YYYY'),1998,1,0))"1999" from emp;
```

The bottom panel shows the results of the query in a table format:

TOTAL	1995	1996	1997	1998
5	1	1	1	1

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

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" from emp;

OUTPUT:

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

RESULT :

SUB QUERIES

EX_NO:9

DATE:

EX_NO:9

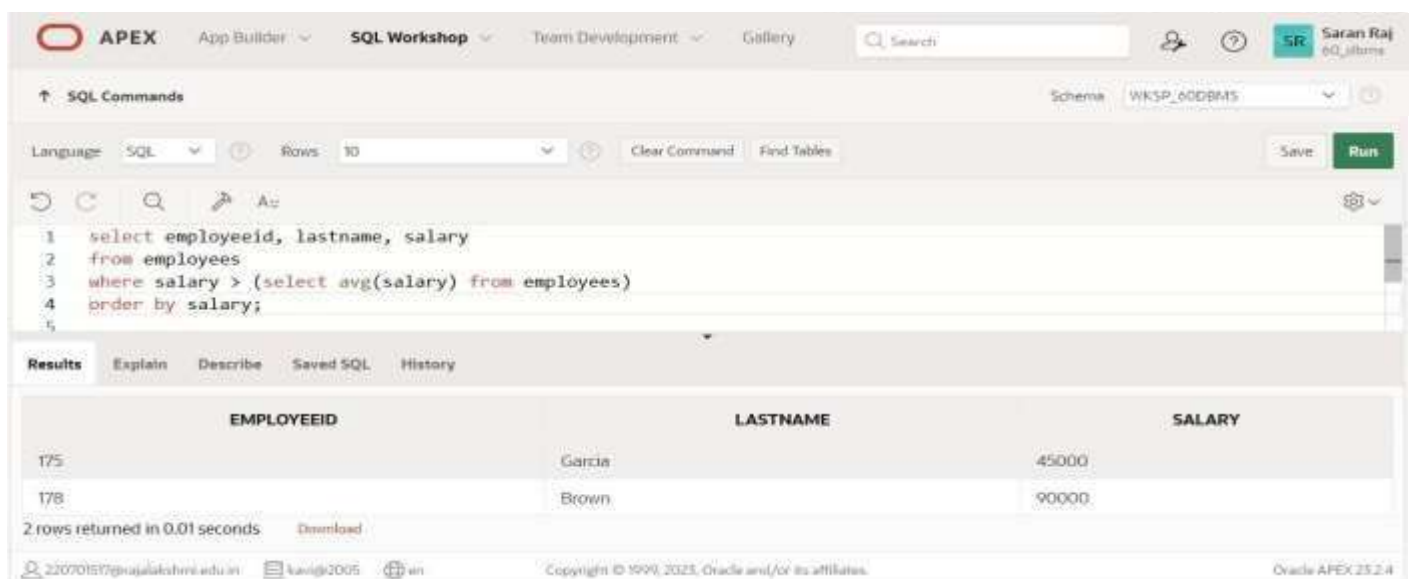
DATE:

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

QUERY:

select employeeid, lastname, salary from employees

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



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

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

The Results tab is selected, displaying the following data:

EMPLOYEEID	LASTNAME	SALARY
175	Garcia	45000
178	Brown	90000

2 rows returned in 0.01 seconds. The interface also shows the user 'Saran Raj' and the schema 'WKSP_60DBMS'.

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

QUERY:

select employeeid, lastname

from employeeswhere deptid in

(select deptid from employees where

lastname like '%u%');OUTPUT:

SQL Commands

W 1:1 &10=3t,s

1,1Jl

Clear Command Find Tables

Q.

1 !>el ct mploye: id lilli-tn,im

2 from mploye s

3 wher-e dept d in (lielac deptid -from emp, QYee5, w-heT'e ,H,tn.ime l-i k:e 'Sut.');

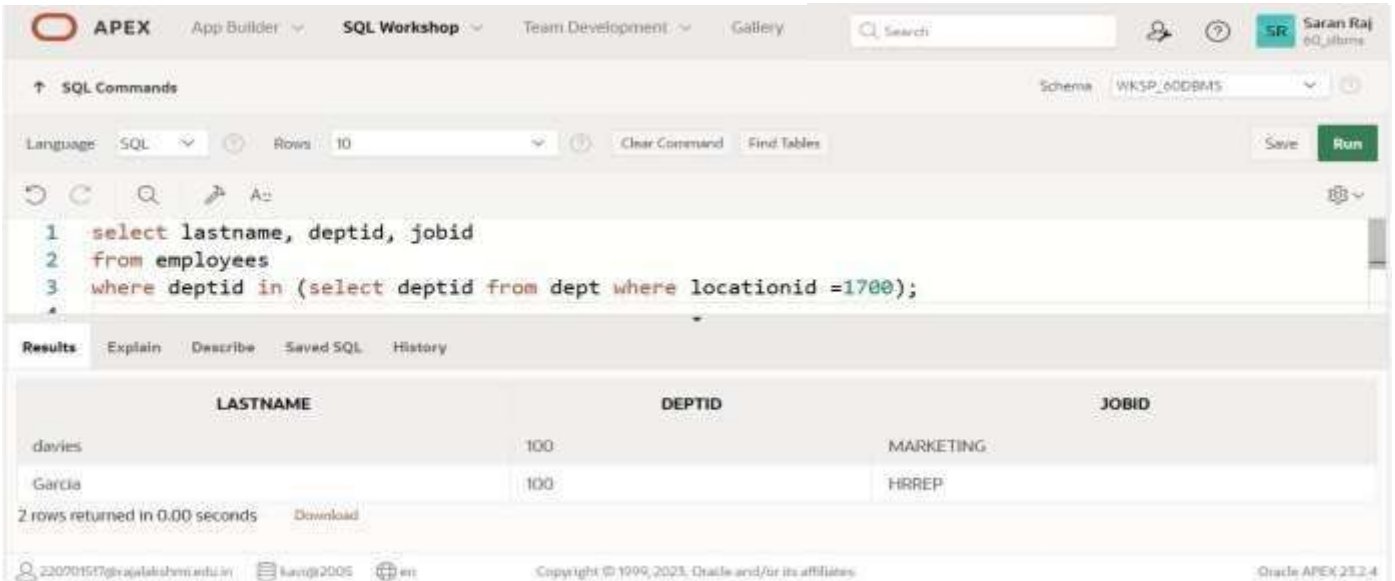
4

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

QUERY:

```
select lastname, deptid, jobid  
from employees
```

where deptid in (select deptid from dept where locationid =1700);



The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is: `select lastname, deptid, jobid from employees where deptid in (select deptid from dept where locationid =1700);`. The results are displayed in a table with columns LASTNAME, DEPTID, and JOBID. Two rows are returned: one for 'davis' in department 100 (MARKETING) and one for 'Garcia' in department 100 (HRREP).

LASTNAME	DEPTID	JOBID
davis	100	MARKETING
Garcia	100	HRREP

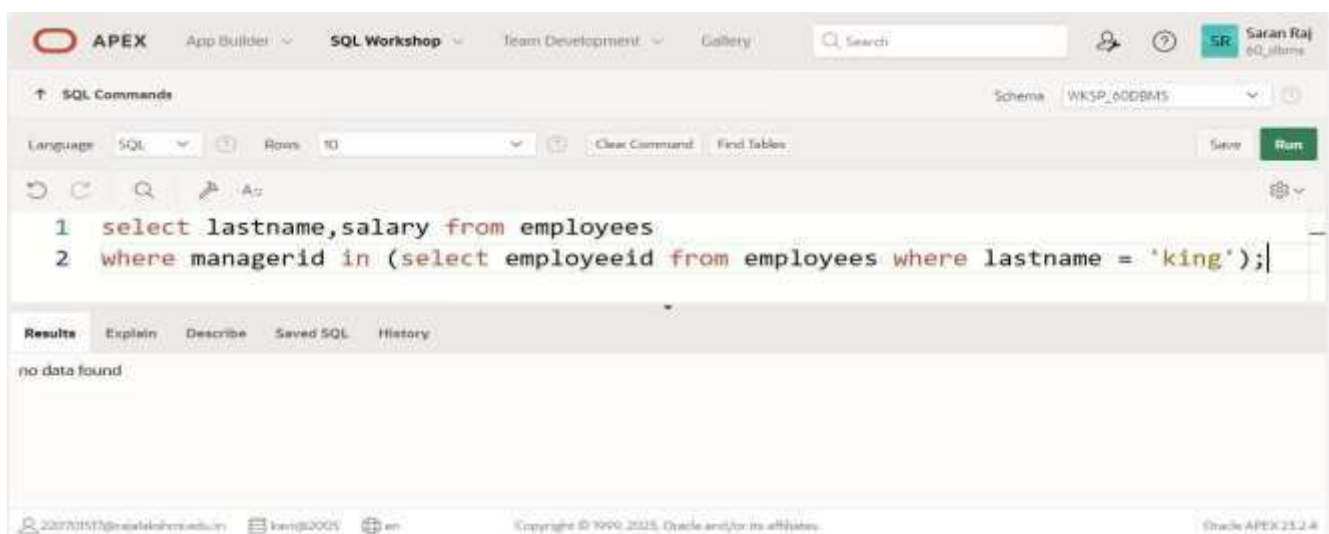
3. Create a report for HR that displays the last name and salary of every employee who reports to King.

QUERY:

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

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

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is: `select lastname, salary from employees where managerid in (select employeeid from employees where lastname = 'king');`. The results section shows "no data found".

LASTNAME	SALARY
----------	--------

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

QUERY:

```
select deptid, lastname, jobid  
from employees
```

where deptid in (select deptid from dept where deptname = 'executive');OUTPUT:

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

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

The results tab shows the following data:

DEPTID	LASTNAME	JOBID
100	davies	MARKETING
100	king	HRREP

2 rows returned in 0.01 seconds

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

QUERY:

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

where salary > (select avg(salary) from employees) and

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

OUTPUT:

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

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

The results tab shows the following data:

EMPLOYEEID	LASTNAME	SALARY
178	Brown	90000

1 rows returned in 0.01 seconds

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

RESULT:

USING THE SET OPERATORS

EX.NO:10

DATE:

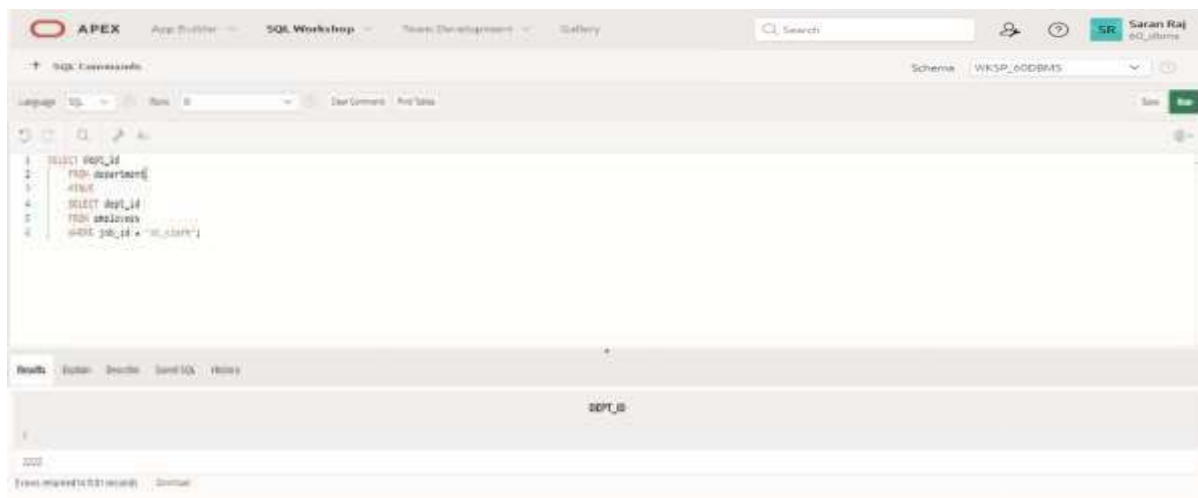
Find the Solution for the following:

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

OUTPUT:

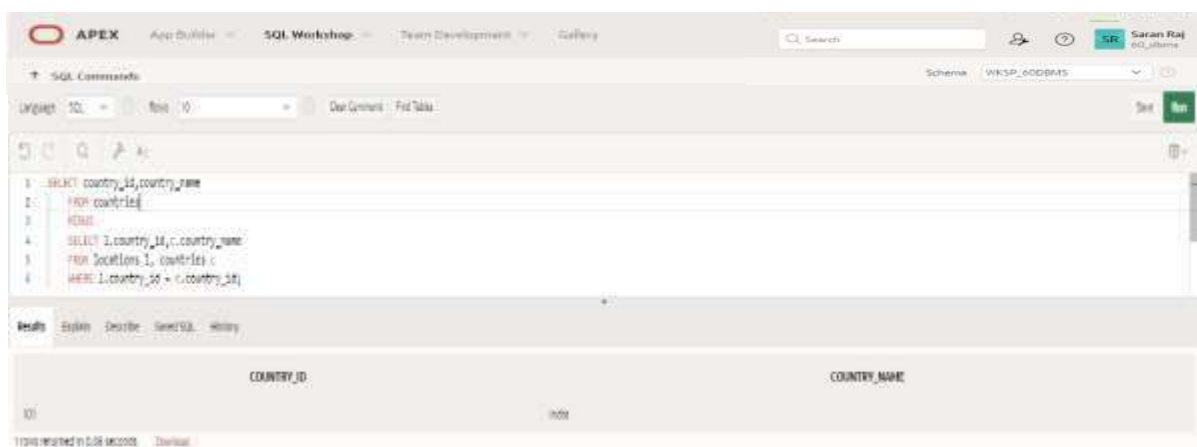


2.The HR department needs a list of countries that have no departments located in them. Display the country ID and the name of the countries. Use set operators to create this report.

QUERY:

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

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The SQL editor contains 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 query is executed, and the results are displayed in a table with two columns: COUNTRY_ID and COUNTRY_NAME. The table shows one row of data, which is '30' for COUNTRY_ID and 'India' for COUNTRY_NAME. The status bar at the bottom indicates that 1 rows were returned in 0.08 seconds.

COUNTRY_ID	COUNTRY_NAME
30	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_no, dept_id  
FROM employees  
WHERE dept_id = 10
```

UNION ALL

```
SELECT DISTINCT job_no, dept_id  
FROM employees  
WHERE dept_id = 50
```

UNION ALL

```
SELECT DISTINCT job_no, dept_id  
FROM employees  
WHERE dept_id = 20;
```

OUTPUT:



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

```
1. SELECT DISTINCT job_no, dept_id  
2. FROM employees  
3. WHERE dept_id = 10  
4.  
5. UNION ALL  
6.  
7. SELECT DISTINCT job_no, dept_id  
8. FROM employees  
9. WHERE dept_id = 50  
10.  
11. UNION ALL  
12.  
13. SELECT DISTINCT job_no, dept_id  
14. FROM employees  
15. WHERE dept_id = 20;
```

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

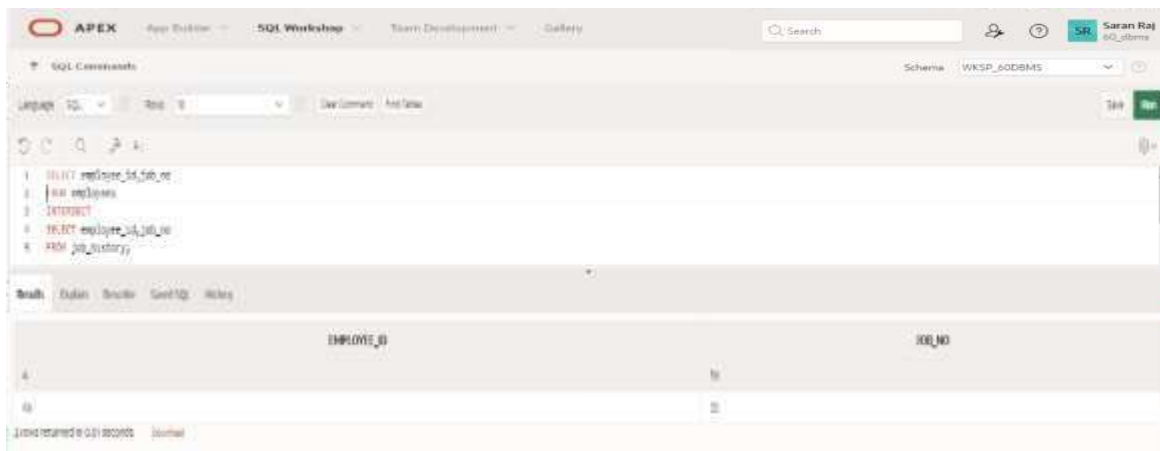
JOB_NO	DEPT_ID
1000	10
1001	10
1002	10

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

QUERY:

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

OUTPUT:



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

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

The Results pane shows the output of the query, which is a table with two columns: EMPLOYEE_ID and JOB_NO. The table contains two rows of data:

EMPLOYEE_ID	JOB_NO
4	10
60	20

At the bottom of the Results pane, it indicates that 2 rows were returned in 0.01 seconds.

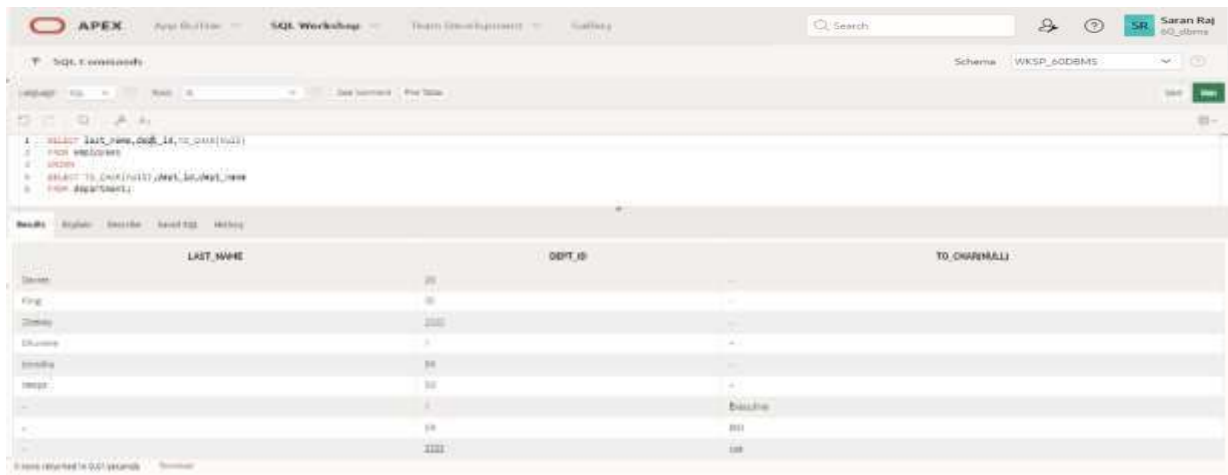
5. The HR department needs a report with the following specifications:

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

QUERY:

```
SELECT last_name,dept_id,TO_CHAR(null)
FROM employees
UNION
SELECT TO_CHAR(null),dept_id,dept_name
FROM department;
```

OUTPUT:



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

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

The results pane displays the output of the query. It has three columns: LAST_NAME, DEPT_ID, and TO_CHAR(NULL). The data is as follows:

LAST_NAME	DEPT_ID	TO_CHAR(NULL)
Clare	20	
King	10	
DeHaan	2100	
Chuneev	2	
Scott	10	
Neer	20	
		Baseline
	10	101
	2100	108

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

RESULT:

EXNO:11

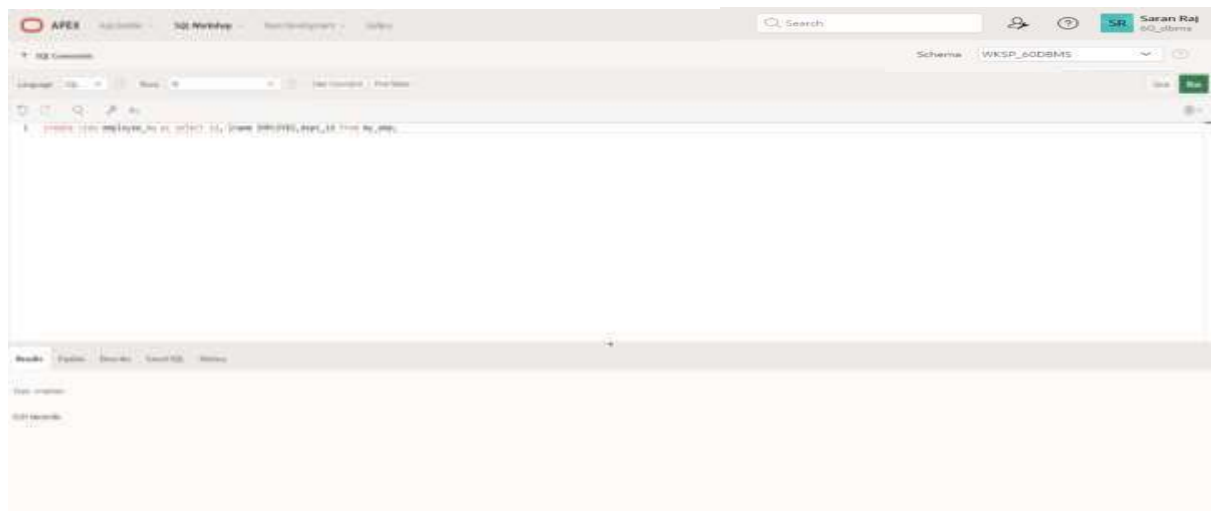
CREATING VIEWS

DATE:

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

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

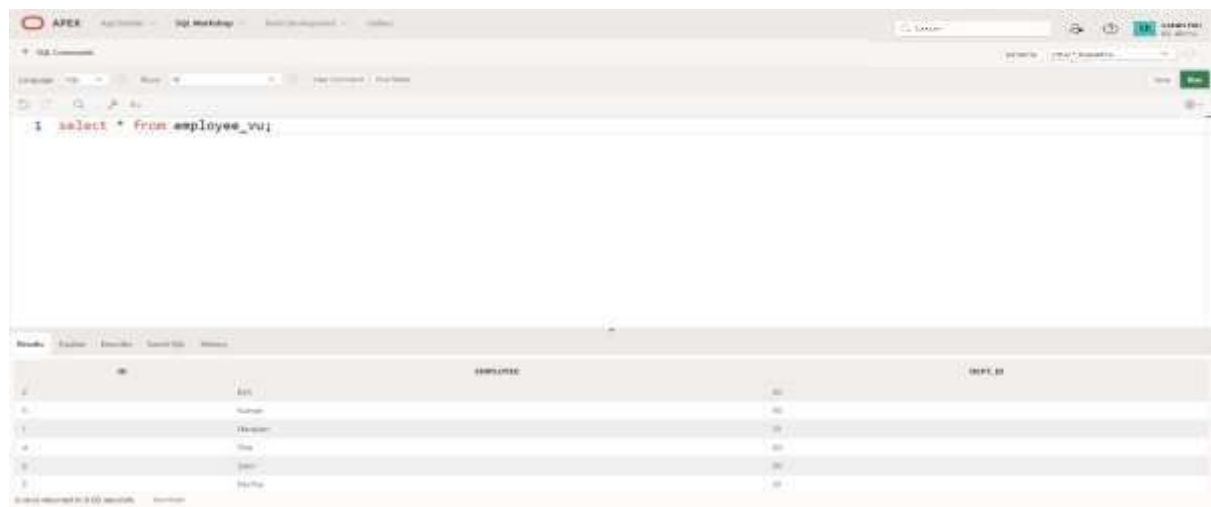
OUTPUT:



2. Display the contents of the EMPLOYEES_VU view.

QUERY: select * from employee_vu;

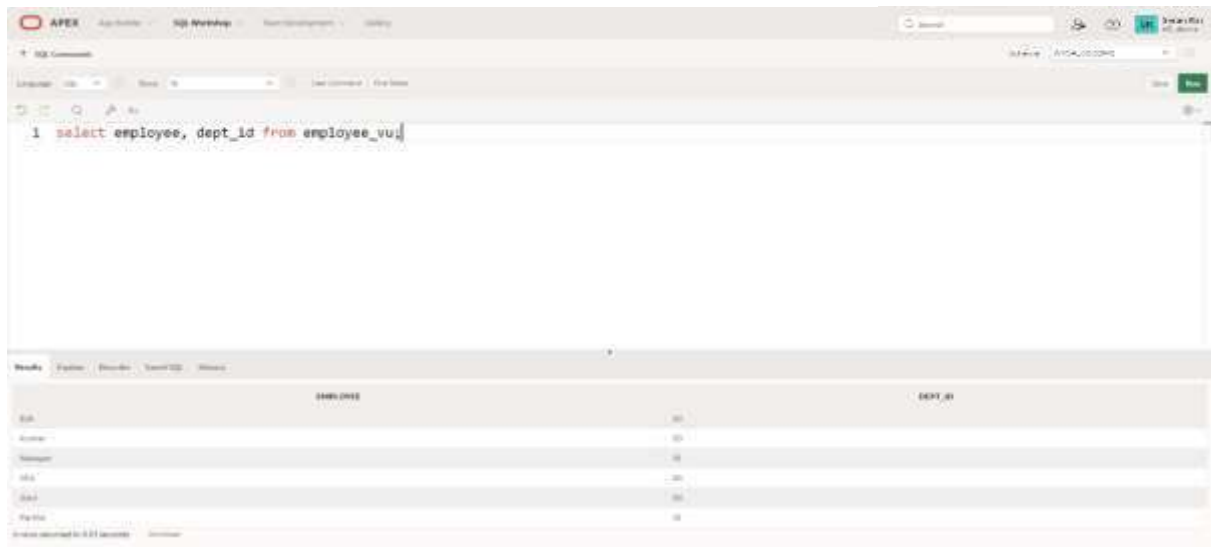
OUTPUT:



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

QUERY: select employee, dept_id from employee_vu;

OUTPUT:



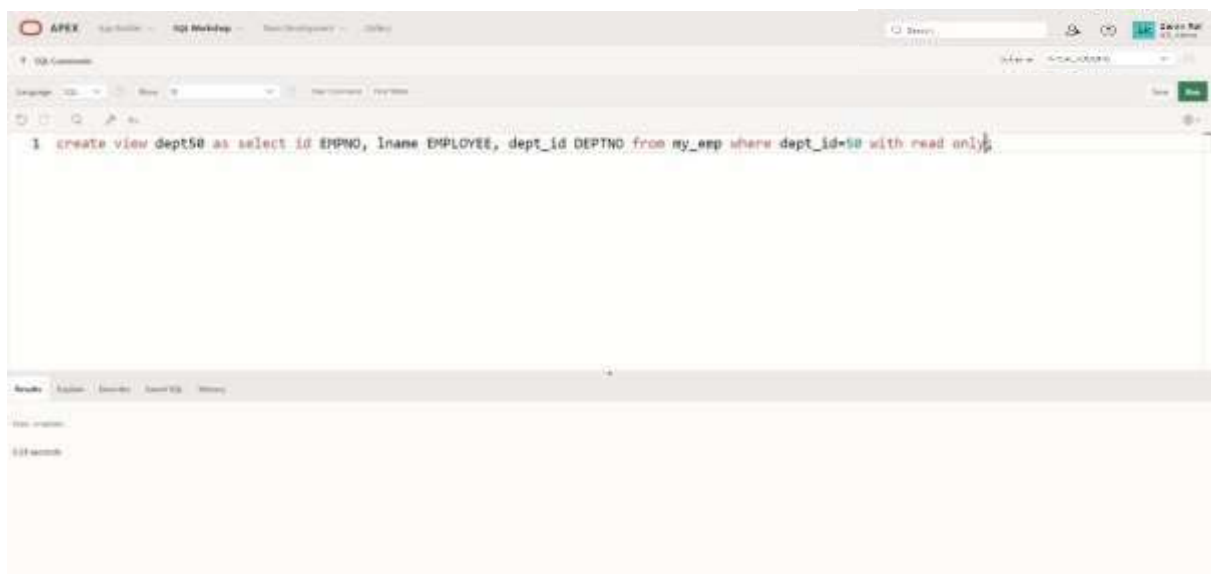
The screenshot shows the APEX SQL Workshop interface. The SQL command window contains the query: `1 select employee, dept_id from employee_vu;`. The Results tab is active, displaying a table with two columns: EMPLOYEE and DEPT_ID. The table contains six rows of data.

EMPLOYEE	DEPT_ID
Scott	50
Adams	50
Turner	50
Watt	50
Clark	50
Ford	50

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

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

OUTPUT:

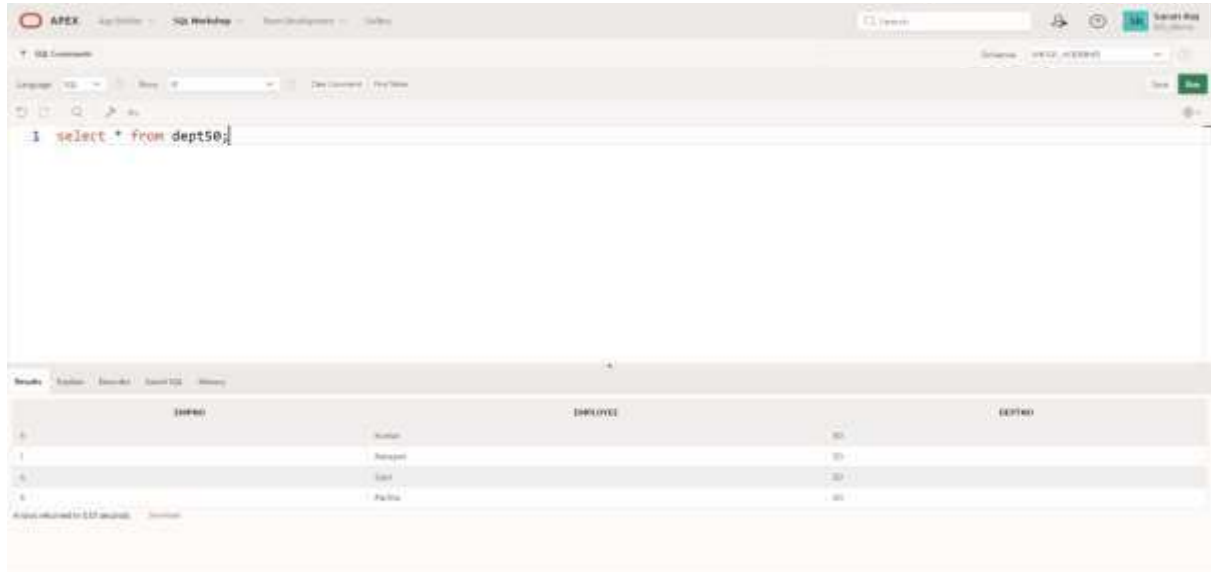


The screenshot shows the APEX SQL Workshop interface. The SQL command window contains the query: `1 create view dept50 as select id EMPNO, lname EMPLOYEE, dept_id DEPTNO from my_emp where dept_id=50 with read only;`. The Results tab is active, displaying a message: "View created." Below the message, it indicates the execution time: "0.01 seconds".

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

QUERY: select * from dept50;

OUTPUT:



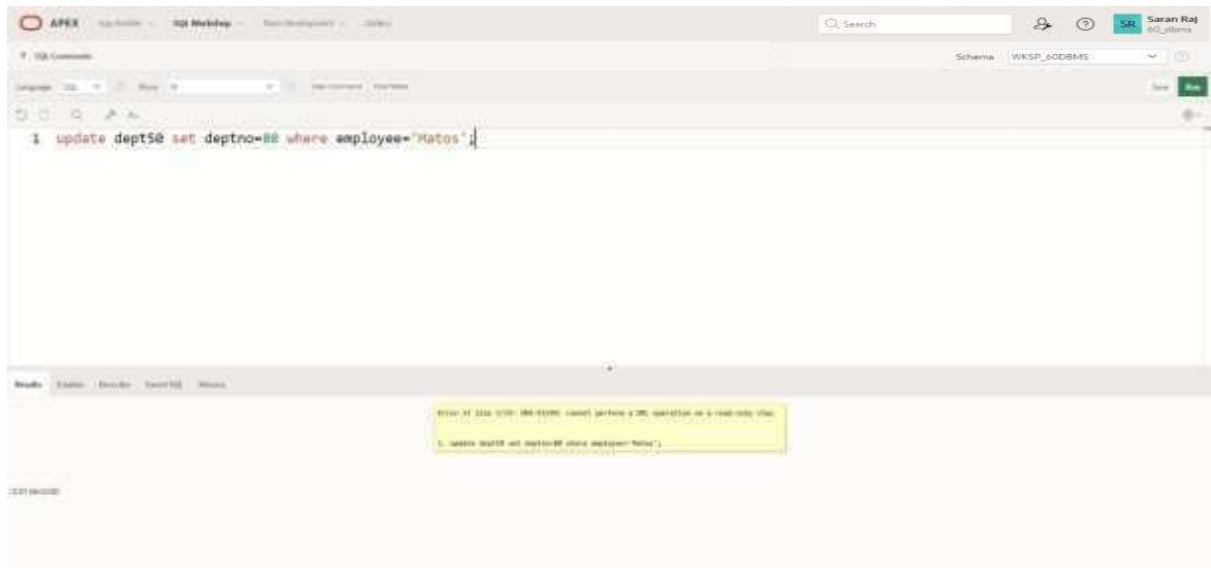
The screenshot shows the APEX SQL Workshop interface. The SQL statement `select * from dept50;` has been entered and executed. The results are displayed in a table with 4 columns: DEPTNO, EMPNO, EMPLOYEE, and DEPTNAME. The table contains 4 rows of data.

DEPTNO	EMPNO	EMPLOYEE	DEPTNAME
10	1	King	DEPT
10	2	DeMott	DEPT
10	3	Smith	DEPT
10	4	Patel	DEPT

5. Attempt to reassign Matos to department 80.

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

OUTPUT:



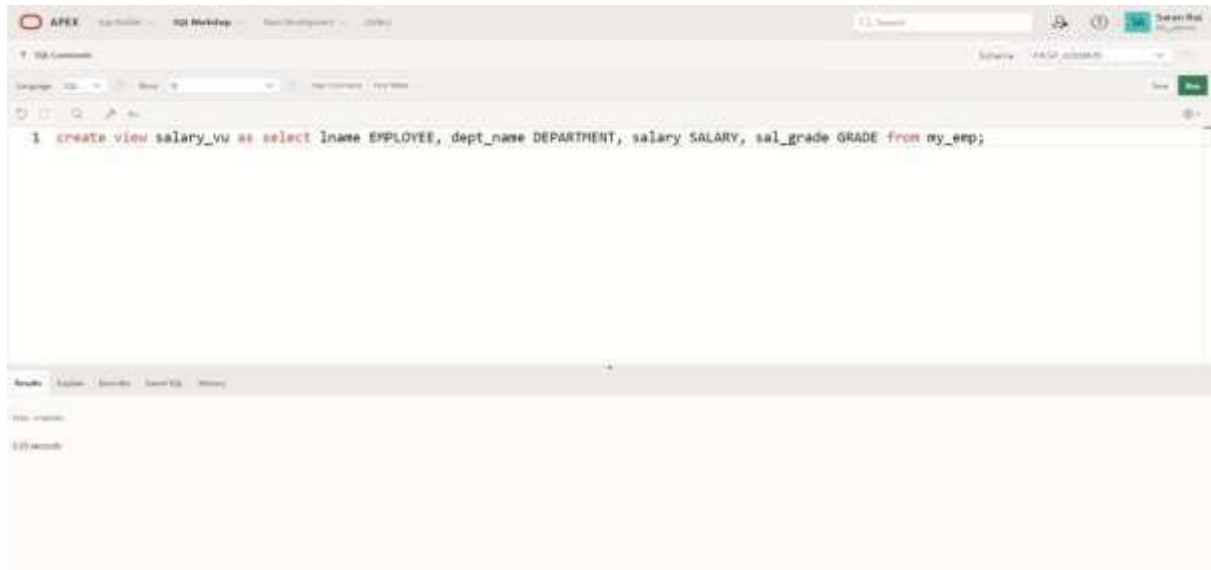
The screenshot shows the APEX SQL Workshop interface. The SQL statement `update dept50 set deptno=80 where employee='Matos';` has been entered. An error message is displayed: "Error at Line 1: ORA-01774: cannot partition a table, operation on a read-only table".

Error at Line 1: ORA-01774: cannot partition a table, operation on a read-only table
1. update dept50 set deptno=80 where employee='Matos';

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

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

OUTPUT:



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

RESULT:

EXNO:12 PRIMARY KEY, FOREIGN KEY AND CHECK CONSTRAINTS

1. What is the purpose of a

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

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

animal_id NUMBER(6) - **PRIMARY KEY**

name VARCHAR2(25)

license_tag_number NUMBER(10)- **UNIQUE**

admit_date DATE- **NOT NULL**

adoption_id NUMBER(5),

vaccination_date DATE- **NOT NULL**

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

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

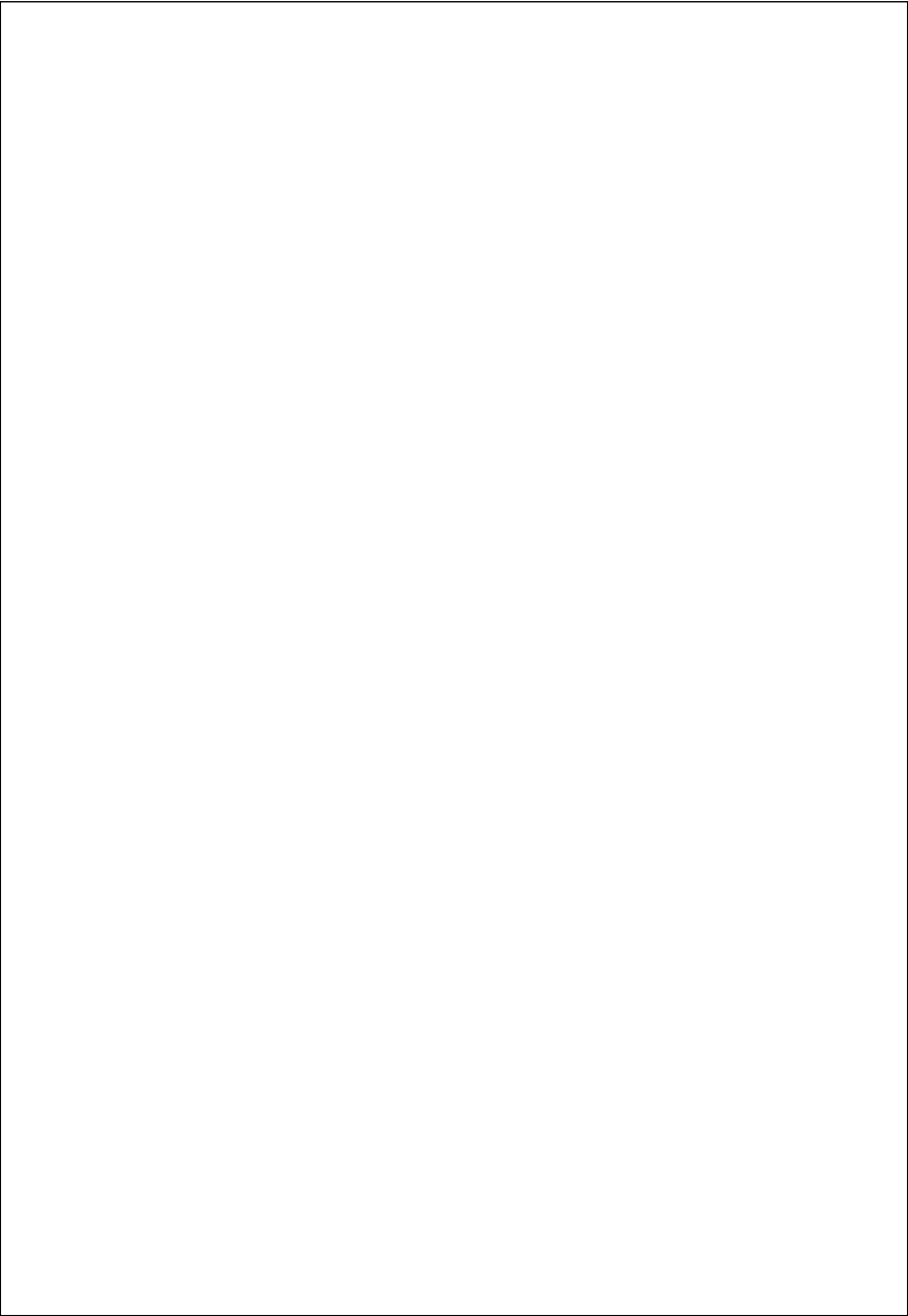
OUTPUT:



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

Messages

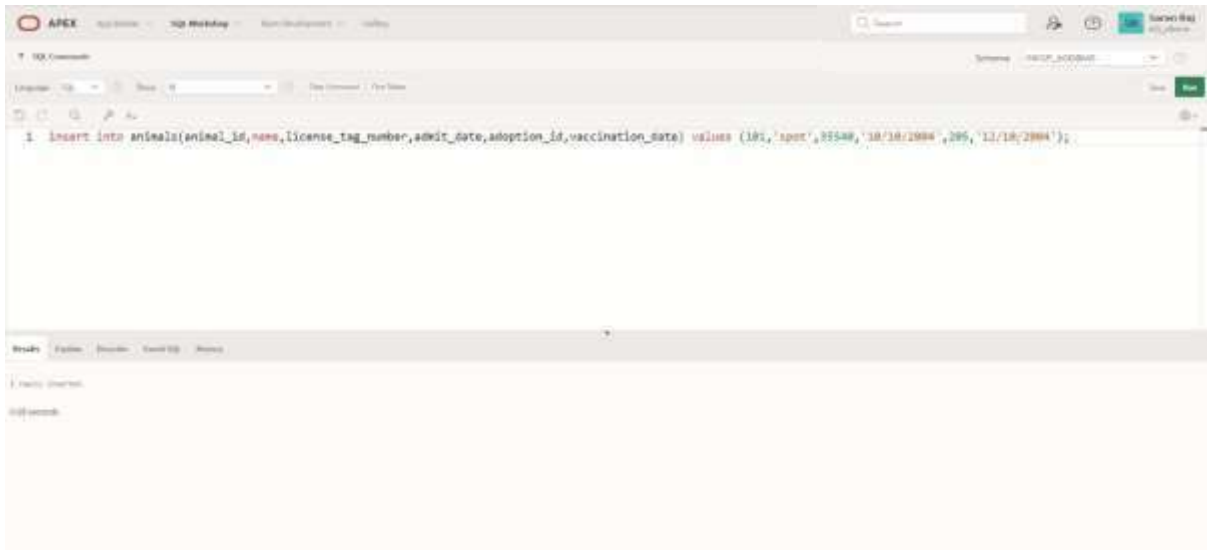
Table created.



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

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

OUTPUT:



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

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

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

RESULT:

EXERCISE 13

Creating Views

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

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

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

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

3. SELECT * FROM view_d_songs. What was returned?

Results	Explain	Describe	Saved SQL	History
ID	Song Title	ARTIST		
47	Hurrah for Today	The Jubilant Trio		
49	Lets Celebrate	The Celebrants		

2 rows returned in 0.00 seconds Download

4. REPLACE view_d_songs. Add type_code to the column list. Use aliases for all columns. Or use alias after the CREATE statement as shown.

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

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 averagesalaries, 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';
```

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

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

```
SELECT * FROM view_copy_d_songs;
```

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

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

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

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

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

```
SELECT * FROM read_copy_d_cds;
```

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

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

6. Use REPLACE to modify read_copy_d_cds. Replace the READ ONLY option with WITH CHECK

OPTION CONSTRAINT ck_read_copy_d_cds. Execute a SELECT * statement to verify that the view exists.

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

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

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

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

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

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

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

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

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

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

DELETE, INSERT, MODIFY restricted if it contains: Group

functions

GROUP BY CLAUSE

DISTINCT

pseudocolumn ROWNUM Keyword

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

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

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

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

Managing Views

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

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

```
SELECT * FROM view_copy_d_songs;
```

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, salaryFROM  
(SELECT * FROM f_staffs ORDER BY SALARY);
```

Indexes and Synonyms

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

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

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

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

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

3. When will an index be created automatically?

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

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

```
CREATE INDEX d_tlg_cd_number_fk_ion  
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 SYNONYM dj_tracks FOR d_track_listings;
```

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

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

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

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

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

10. Drop the synonym that you created in question

```
DROP SYNONYM dj_tracks2;
```

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

RESULT:

OTHER DATABASE OBJECTS

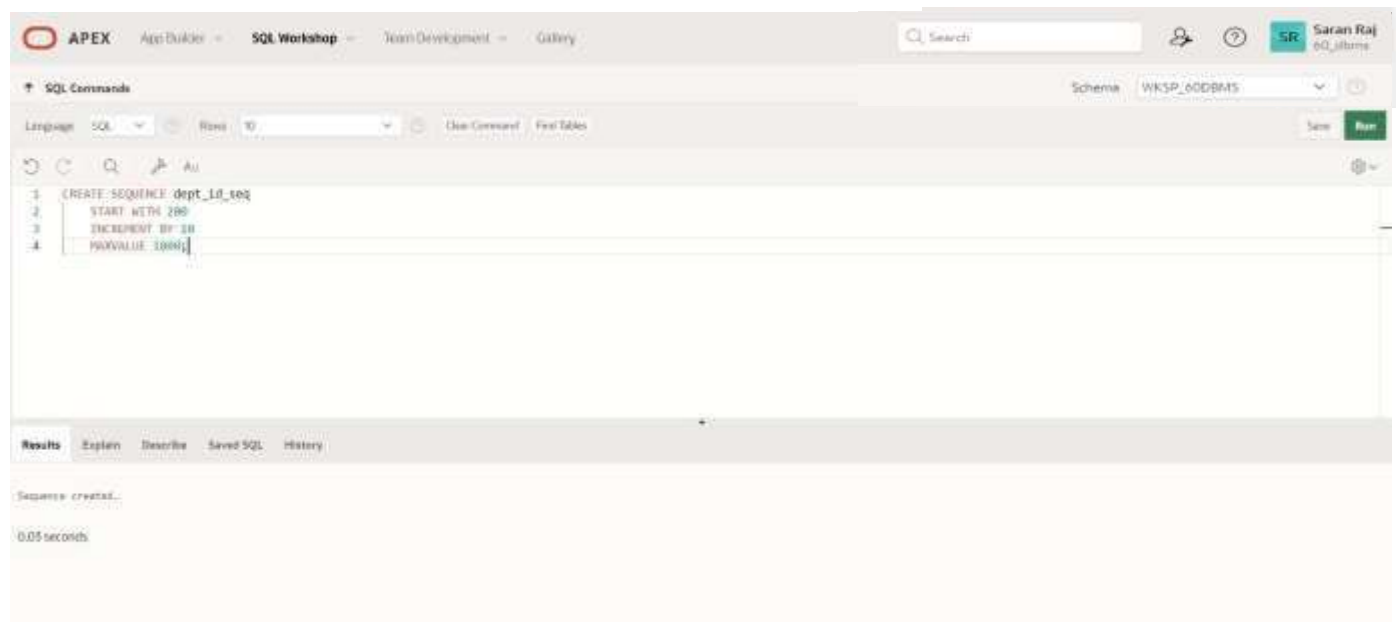
EX_NO:14

DATE:

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

QUERY:

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



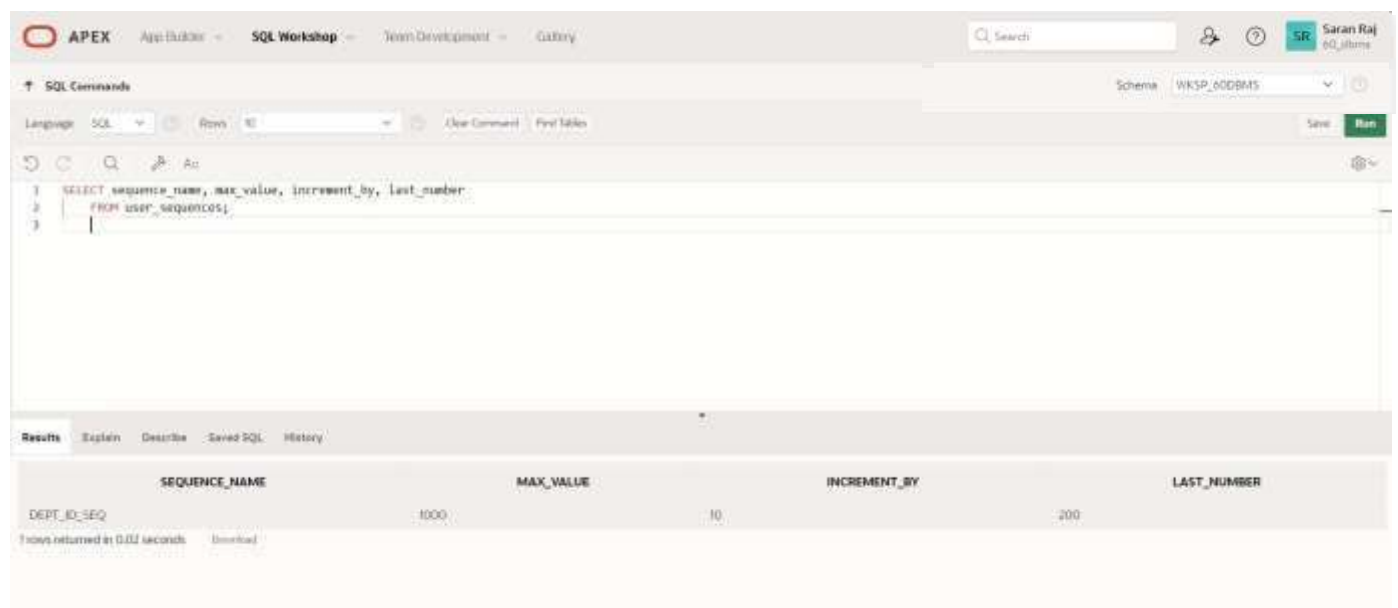
OUTPUT:

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:



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

QUERY:

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

OUTPUT:

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 'Saran Raj' are on the right. The 'SQL Commands' section is active, showing a script named 'asasci616' with a 'Complete' status. The command is 'INSERT INTO dept VALUES (dept_id_seq.nextval, 'Education');'. The execution results show '1 Statements Processed', '1 Successful', and '0 With Errors'. The feedback column shows '1 row(s) inserted.'.

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

QUERY:

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

OUTPUT:

The screenshot shows the APEX SQL Workshop interface. The top navigation bar is the same as the previous screenshot. The 'SQL Commands' section is active, showing a script named 'asasci616' with a 'Complete' status. The command is 'CREATE INDEX emp_dept_id_idx ON EMPLOYEES (department_id);'. The execution results show 'Index created.' and '0.02 seconds'.

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

QUERY:

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

OUTPUT:

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

```
1 SELECT index_name, table_name, uniqueness
2 FROM user_indexes
3 WHERE table_name = 'EMPLOYEES';
4
```

The Results tab is active, displaying the following output:

INDEX_NAME	TABLE_NAME	UNIQUENESS
EMP_DEPT_ID_IDX	EMPLOYEES	NONUNIQUE

1 rows returned in 0.05 seconds

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

RESULT:

CONTROLLING USER ACCESS

EX_NO:15

DATE:

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

The CREATE SESSION system privilege

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

The CREATE TABLE privilege

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

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

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

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

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

The ALTER USER statement

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

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

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

7. Query all the rows in your DEPARTMENTS table.

SELECT * FROM departments;

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

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

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

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

SELECT table_name FROM user_tables;

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

Team 1 revokes the privilege.

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

Team 2 revokes the privilege.

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

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

Team 1 executes this INSERT statement.

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

Team 2 executes this INSERT statement.

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

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

RESULT:

CONTROL STRUCTURES

EX_NO:**DATE:**

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

QUERY:

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

OUTPUT:

The screenshot displays 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 'Sourav Roy' are on the right. The 'SQL Commands' section is active, showing a list of commands with line numbers 1 through 9. The code being executed is the same PL/SQL block as shown in the query section. Below the code editor, the 'Results' tab is selected, showing the output: 'Incentive = 8480' and 'Statement processed.' in 0.00 seconds.

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

Results | Explain | Describe | Saved SQL | History

Incentive = 8480
Statement processed.
0.00 seconds

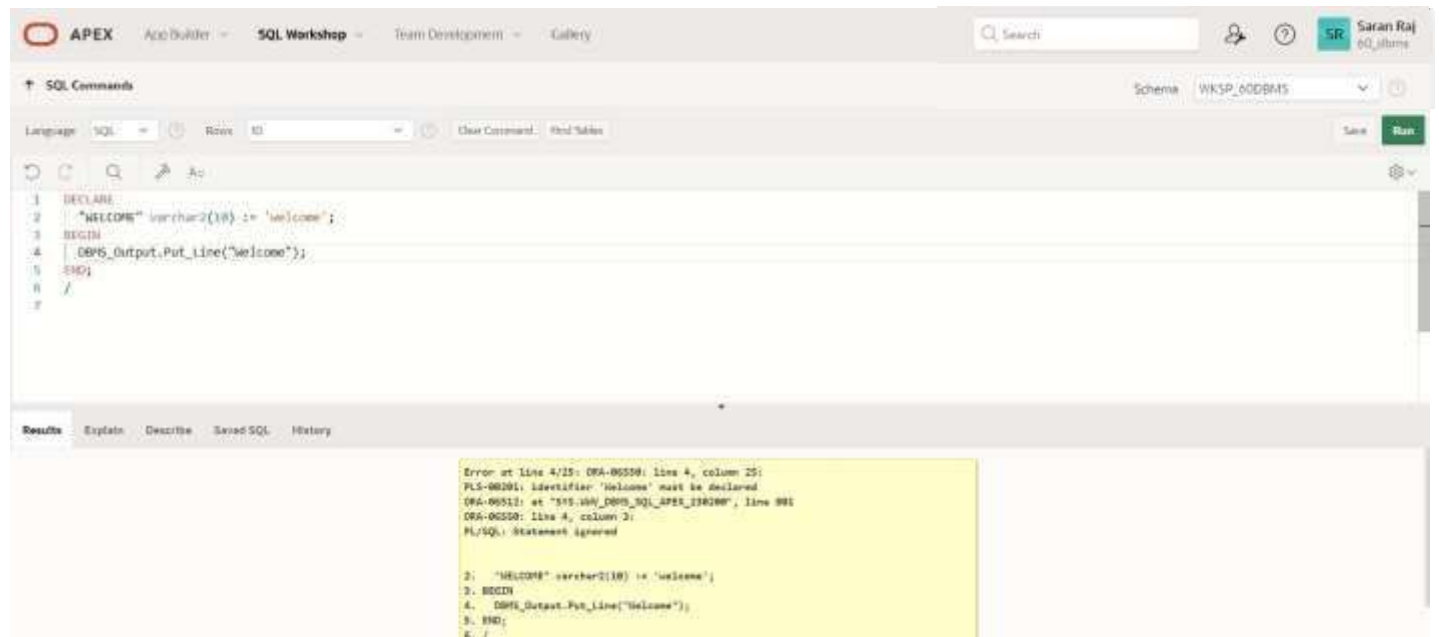
2.) Write a PL/SQL block to show an invalid case-insensitive reference to a quoted and without quoted user-defined identifier

QUERY:

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

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

OUTPUT:



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

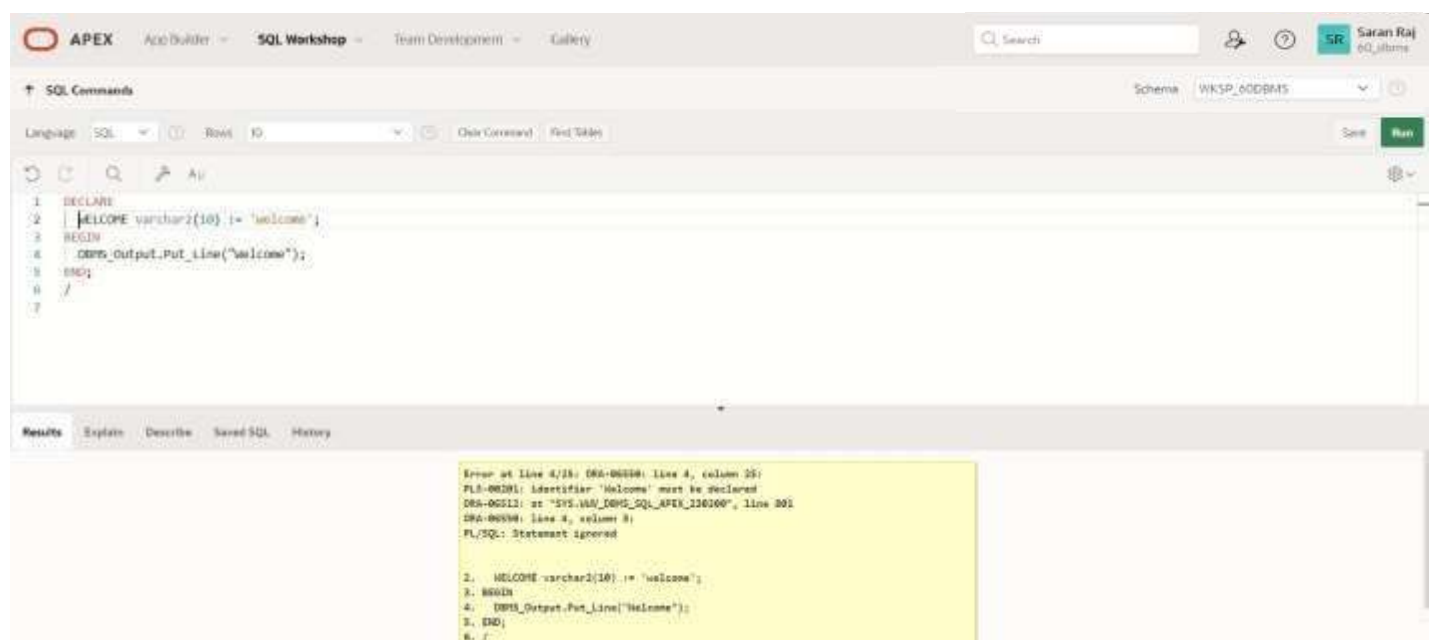
```
1 DECLARE
2 "WELCOME" varchar2(10) := 'welcome';
3 BEGIN
4 DBMS_Output.Put_Line("Welcome");
5 END;
6 /
```

The Results pane shows the following error message:

```
Error at line 4/25: ORA-00559: line 4, column 25:
PLS-00559: identifier 'Welcome' must be declared
ORA-00512: at 'SYS.WMV_DBMS_SQL_APEX_130200', line 991
ORA-00559: line 4, column 2:
PL/SQL: Statement ignored
```

The code in the Results pane is:

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



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

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

The Results pane shows the following error message:

```
Error at line 4/25: ORA-00559: line 4, column 25:
PLS-00559: identifier 'WELCOME' must be declared
ORA-00512: at 'SYS.WMV_DBMS_SQL_APEX_130200', line 991
ORA-00559: line 4, column 2:
PL/SQL: Statement ignored
```

The code in the Results pane is:

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

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

QUERY:

DECLARE

salary_of_emp NUMBER(8,2);

PROCEDURE approx_salary (

emp NUMBER,

empsal IN OUT NUMBER,

addless NUMBER

) IS

BEGIN

empsal := empsal + addless;

END;

BEGIN

SELECT salary INTO salary_of_emp

FROM employees

WHERE employee_id = 122;

DBMS_OUTPUT.PUT_LINE

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

approx_salary (100, salary_of_emp, 1000);

DBMS_OUTPUT.PUT_LINE

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

END;

/

OUTPUT:

The screenshot displays 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 'Saran Raj' are on the right. The 'SQL Commands' pane on the left shows the following code:

```
1 DECLARE
2   salary_of_emp NUMBER(8,2);
3
4   PROCEDURE approx_salary (
5     emp NUMBER,
6     empsal IN OUT NUMBER,
7     addless NUMBER
8   ) IS
9   BEGIN
10    empsal := empsal + addless;
11  END;
12
13 BEGIN
14   SELECT salary INTO salary_of_emp
15   FROM employees
16   WHERE employee_id = 122;
17
```

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

```
Before invoking procedure, salary_of_emp: 8900
After invoking procedure, salary_of_emp: 9900
Statement processed.
0.00 seconds
```

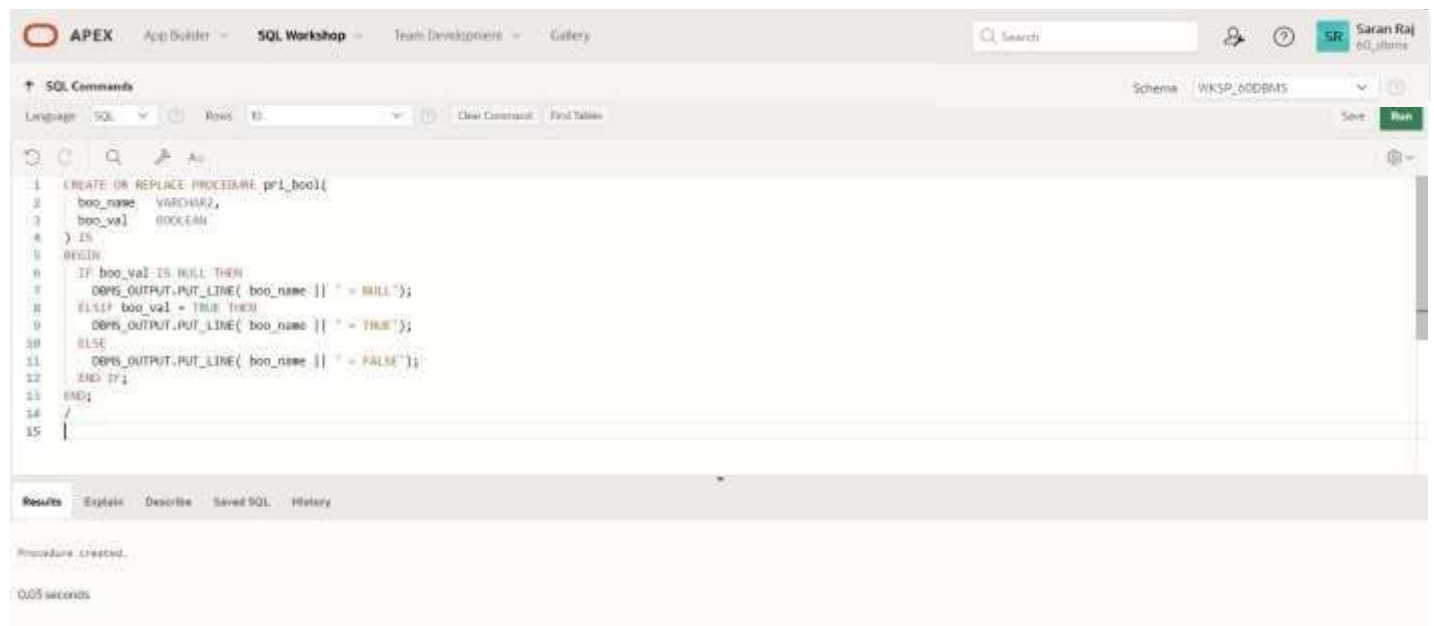
4.) Write a PL/SQL block to create a procedure using the "IS [NOT] NULL Operator" and show AND operator returns TRUE if and only if both operands are TRUE.

QUERY:

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

```

OUTPUT:



The screenshot displays 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 'Saran Raj' are on the right. The 'SQL Commands' panel shows the following code:

```
1 CREATE OR REPLACE PROCEDURE pri_bool(  
2     boo_name VARCHAR2,  
3     boo_val  BOOLEAN  
4 ) IS  
5 BEGIN  
6     IF boo_val IS NULL THEN  
7         DBMS_OUTPUT.PUT_LINE( boo_name || ' = NULL');  
8     ELSIF boo_val = TRUE THEN  
9         DBMS_OUTPUT.PUT_LINE( boo_name || ' = TRUE');  
10    ELSE  
11        DBMS_OUTPUT.PUT_LINE( boo_name || ' = FALSE');  
12    END IF;  
13 END;  
14 /  
15 |
```

The 'Results' panel at the bottom shows the message 'Procedure created.' and the execution time '0.05 seconds'.

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 displays 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 'Saran Raj' are on the right. The 'SQL Commands' panel is active, showing a PL/SQL block with line numbers 1 through 17. The block defines a procedure 'pat_match' and calls it with two test cases. The 'Results' panel at the bottom shows the output: 'TRUE' for the first call and 'FALSE' for the second. Below the results, it states 'Statement processed.' and '0.00 seconds'.

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

Results: Explain Describe Saved SQL History

TRUE
FALSE

Statement processed.

0.00 seconds

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

QUERY:

DECLARE

num_small NUMBER := 8;

num_large NUMBER := 5;

num_temp NUMBER;

BEGIN

IF num_small > num_large THEN

num_temp := num_small;

num_small := num_large;

num_large := num_temp;

END IF;

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

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

END;

/

OUTPUT:

The screenshot displays 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 'Saran Raj' are on the right. The 'SQL Commands' panel shows the following code:

```
1  num_small NUMBER := 8;
2  num_large NUMBER := 5;
3  num_temp NUMBER;
4  BEGIN
5
6
7  IF num_small > num_large THEN
8      num_temp := num_small;
9      num_small := num_large;
10     num_large := num_temp;
11 END IF;
12
13 DBMS_OUTPUT.PUT_LINE ('num_small = ' || num_small);
14 DBMS_OUTPUT.PUT_LINE ('num_large = ' || num_large);
15 END;
16 /
17
```

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

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

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

QUERY:

DECLARE

PROCEDURE test1 (

sal_achieve NUMBER,

target_qty NUMBER,

emp_id NUMBER

)

IS

incentive NUMBER := 0;

updated VARCHAR2(3) := 'No';

BEGIN

IF sal_achieve > (target_qty + 200) THEN

incentive := (sal_achieve - target_qty)/4;

UPDATE employees

SET salary = salary + incentive

WHERE employee_id = emp_id;

updated := 'Yes';

END IF;

DBMS_OUTPUT.PUT_LINE (

'Table updated? ' || updated || ', ' ||

'incentive = ' || incentive || '.'

);

END test1;

BEGIN

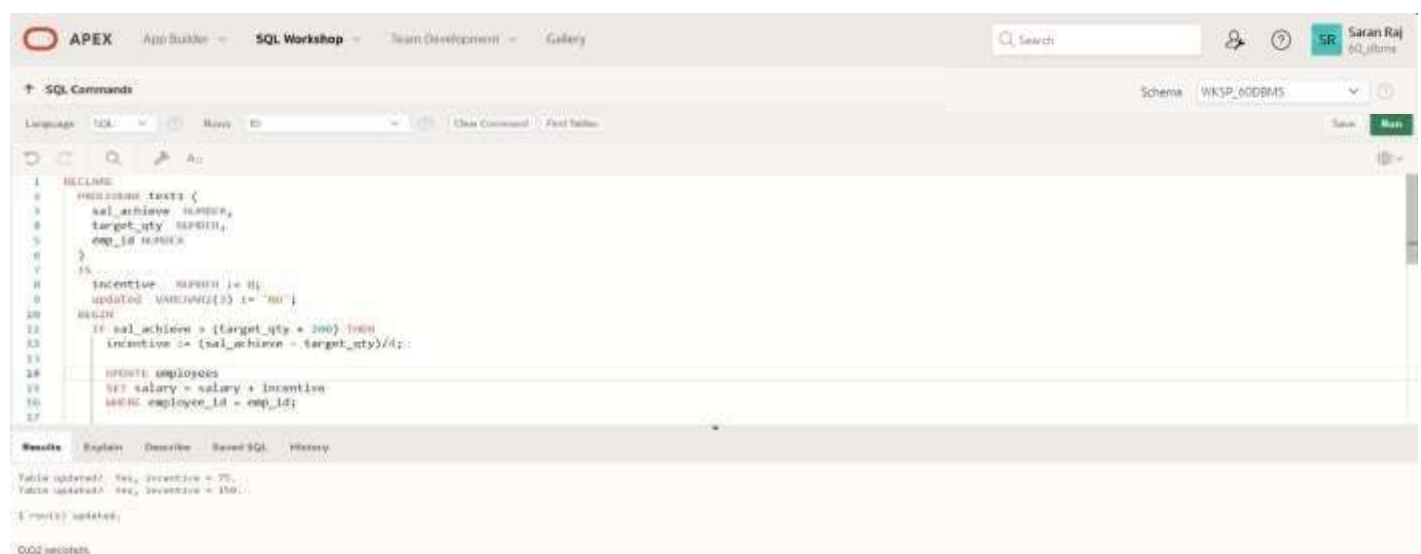
test1(2300, 2000, 144);

test1(3600, 3000, 145);

END;

/

OUTPUT:



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 'Saran Raj' are on the right. The 'SQL Commands' window is active, showing the PL/SQL code. The 'Results' window at the bottom displays the output of the execution.

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

Results

Text
Table updated? Yes, incentive = 75.
Table updated? Yes, incentive = 150.
1 row(s) updated.

OOZ widgets

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

QUERY:

DECLARE

PROCEDURE test1 (sal_achieve NUMBER)

IS

incentive NUMBER := 0;

BEGIN

IF sal_achieve > 44000 THEN

incentive := 1800;

ELSIF sal_achieve > 32000 THEN

incentive := 800;

ELSE

incentive := 500;

END IF;

DBMS_OUTPUT.NEW_LINE;

DBMS_OUTPUT.PUT_LINE (

'Sale achieved : ' || sal_achieve || ', incentive : ' || incentive || '

);

END test1;

BEGIN

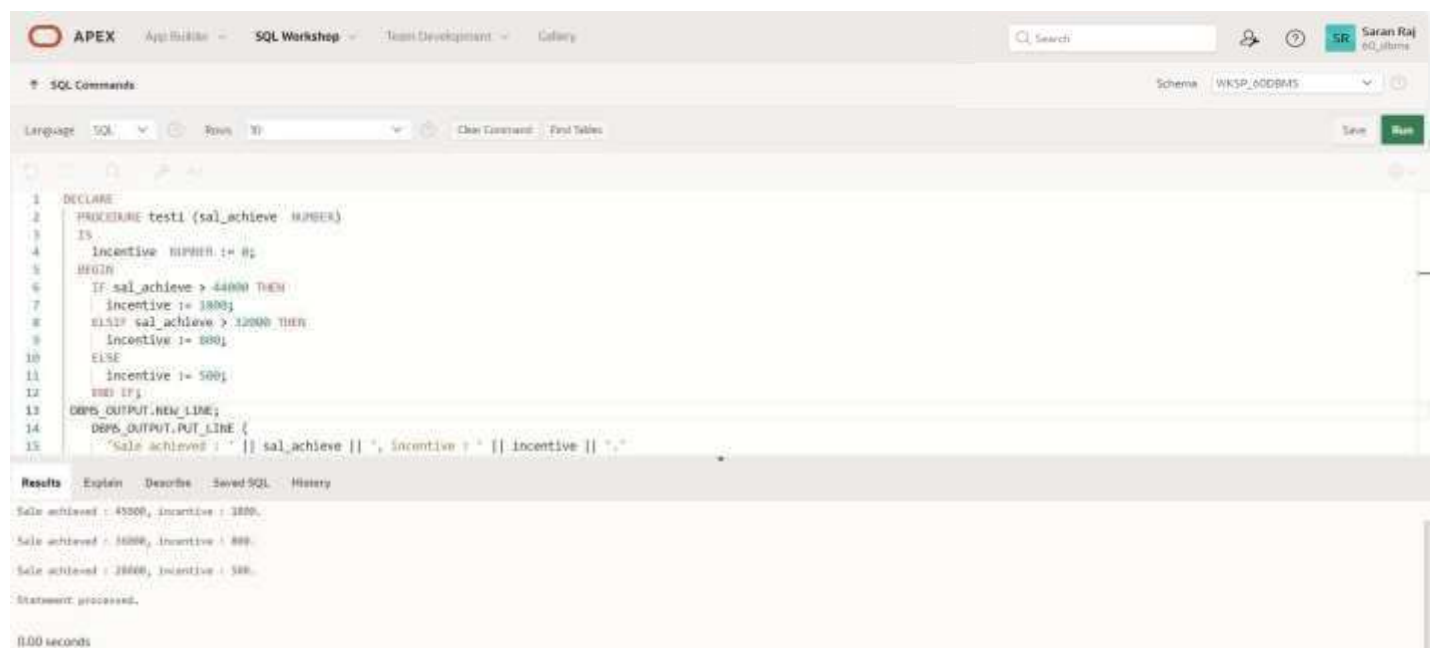
test1(45000);

test1(36000);

test1(28000);

END;

/



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 'Saran Raj' are on the right. The 'SQL Commands' tab is active, showing a list of commands. The main editor displays the PL/SQL code from the previous block. Below the editor, the 'Results' tab is selected, showing the output of the procedure execution. The output consists of three lines of text, each representing the result of a procedure call with different sales achievement values. The first line shows 'Sale achieved : 45000, incentive : 1800.', the second shows 'Sale achieved : 36000, incentive : 800.', and the third shows 'Sale achieved : 28000, incentive : 500.'. The status 'Statement processed.' and a duration of '0.00 seconds' are also visible.

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

```

Results Explain Describe Saved SQL History

Sale achieved : 45000, incentive : 1800.

Sale achieved : 36000, incentive : 800.

Sale achieved : 28000, incentive : 500.

Statement processed.

0.00 seconds

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

QUERY:

SET SERVEROUTPUT ON

DECLARE

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

BEGIN

```
get_dep_id := 80;
```

```
SELECT Count(*)
```

```
INTO tot_emp
```

```
FROM employees e
```

```
join departments d
```

```
ON e.department_id = d.department_id
```

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

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

```
IF tot_emp >= 45 THEN
```

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

```
ELSE
```

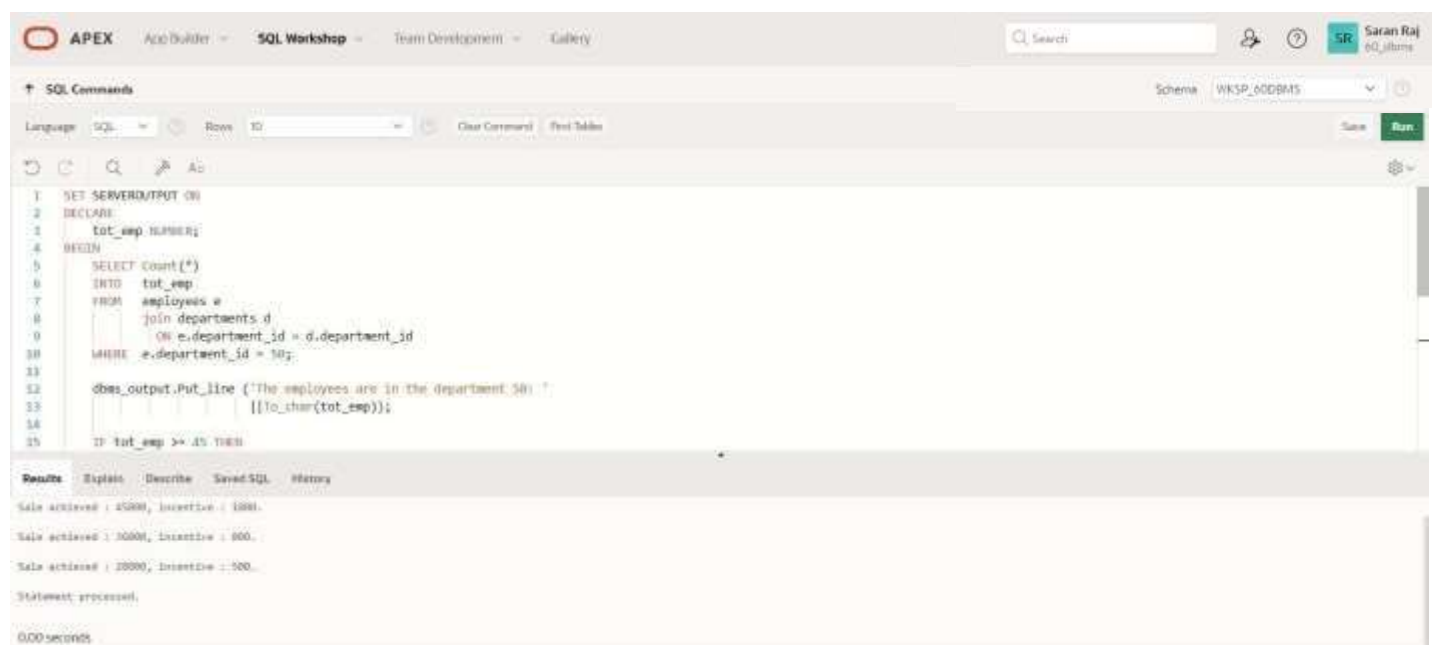
```
dbms_output.Put_line ('There are ' || to_char(45-tot_emp) || ' vacancies in department ' ||
get_dep_id );
```

```
END IF;
```

```
END;
```

```
/
```

OUTPUT:



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

```
1 SET SERVEROUTPUT ON
2 DECLARE
3   tot_emp NUMBER;
4   get_dep_id NUMBER;
5 BEGIN
6   SELECT Count(*)
7   INTO tot_emp
8   FROM employees e
9   join departments d
10  ON e.department_id = d.department_id
11  WHERE e.department_id = 50;
12
13  dbms_output.Put_line ('The employees are in the department 50: '
14  || To_char(tot_emp));
15
16  IF tot_emp >= 45 THEN
17    dbms_output.Put_line ('There are no vacancies in the department 50');
18  ELSE
19    dbms_output.Put_line ('There are ' || to_char(45-tot_emp) || ' vacancies in department 50');
20  END IF;
21 END;
```

The Results pane shows the output of the query:

```
Results: Explain Describe Saved SQL History
The employees are in the department 50: 45
There are 0 vacancies in the department 50
```

The status bar at the bottom indicates that the statement was processed successfully in 0.00 seconds.

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

QUERY:

DECLARE

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

BEGIN

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

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

```
IF tot_emp >= 45 THEN
```

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

```
ELSE
```

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

```
END IF;
```

```
END;
```

```
/
```

OUTPUT:

The screenshot displays 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 'SR Saran Raj' are also visible. The 'SQL Commands' tab is active, showing a PL/SQL script. The script declares variables, sets a department ID, counts employees, and outputs the results. The 'Run' button is highlighted. Below the script, the 'Results' tab shows the output: 'The employees are in the department 80 is: 42' and 'There are 3 vacancies in department 80'. The status 'Statement processed.' and execution time '0.03 seconds' are also displayed.

```
1 DECLARE  
2   tot_emp NUMBER;  
3   get_dep_id NUMBER;  
4  
5 BEGIN  
6   get_dep_id := 80;  
7   SELECT Count(*)  
8   INTO tot_emp  
9   FROM employees e  
10  join departments d  
11  ON e.department_id = d.dept_id  
12  WHERE e.department_id = get_dep_id;  
13  
14  dbms_output.Put_line ('The employees are in the department ' || get_dep_id || ' is: '  
15  || To_char(tot_emp));  
16  
17 IF tot_emp >= 45 THEN  
18   dbms_output.Put_line ('There are no vacancies in the department ' || get_dep_id);  
19 ELSE  
20   dbms_output.Put_line ('There are ' || to_char(45-tot_emp) || ' vacancies in department ' ||  
21   get_dep_id );  
22 END IF;  
23 END;  
24 /
```

Results Explain Describe Saved SQL History

The employees are in the department 80 is: 42
There are 3 vacancies in department 80

Statement processed.

0.03 seconds

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

QUERY:

DECLARE

v_employee_id employees.employee_id%TYPE;

v_full_name employees.first_name%TYPE;

v_job_id employees.job_id%TYPE;

v_hire_date employees.hire_date%TYPE;

v_salary employees.salary%TYPE;

CURSOR c_employees IS

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

BEGIN

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

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

OPEN c_employees;

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

WHILE c_employees%FOUND LOOP

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

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

END LOOP;

CLOSE c_employees;

END;

/

OUTPUT:

SQL Commands

```
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;
```

Results

Employee ID	Full Name	Job Title	Hire Date	Salary
120	John Smith	Sales Rep	01/08/1995	70000
121	Kelly Johnson	Tr Rep	04/26/1998	5000
122	Tamiya Tami	Ac Account	03/05/2024	4800
124	Ann Davies	St Clerk	02/03/1999	12000
142	Jane Davis	Ac Account	03/05/1997	10000
135	Vijaya Mohan	St Clerk	02/22/1999	4800

Statement processed.

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

QUERY:

DECLARE

CURSOR emp_cursor IS

SELECT e.employee_id, e.first_name, m.first_name AS manager_name

FROM employees e

LEFT JOIN employees m ON e.manager_id = m.employee_id;

emp_record emp_cursor%ROWTYPE;

BEGIN

OPEN emp_cursor;

FETCH emp_cursor INTO emp_record;

WHILE emp_cursor%FOUND LOOP

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

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

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

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

FETCH emp_cursor INTO emp_record;

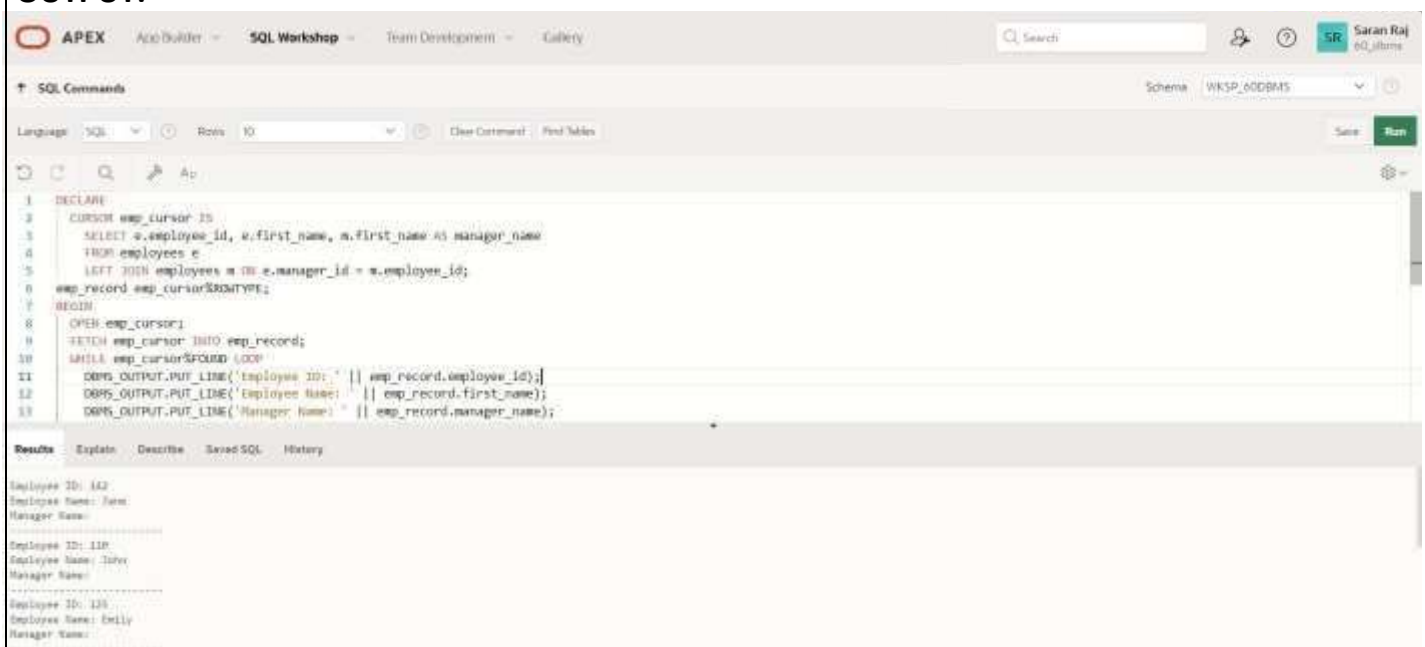
END LOOP;

CLOSE emp_cursor;

END;

/

OUTPUT:



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

```
1 DECLARE
2   CURSOR emp_cursor IS
3     SELECT e.employee_id, e.first_name, m.first_name AS manager_name
4     FROM employees e
5     LEFT JOIN employees m ON e.manager_id = m.employee_id;
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(' ----- ');
15    FETCH emp_cursor INTO emp_record;
16  END LOOP;
17  CLOSE emp_cursor;
18 END;
```

The Results window shows the output of the program:

```
Employee ID: 142
Employee Name: Ram
Manager Name:
-----
Employee ID: 119
Employee Name: Jony
Manager Name:
-----
Employee ID: 115
Employee Name: Emily
Manager Name:
-----
```

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

QUERY:

DECLARE

CURSOR job_cursor IS

SELECT e.job_id, j.lowest_sal

FROM job_grade j, employees e;

job_record job_cursor%ROWTYPE;

BEGIN

OPEN job_cursor;

FETCH job_cursor INTO job_record;

WHILE job_cursor%FOUND LOOP

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

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

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

FETCH job_cursor INTO job_record;

END LOOP;

CLOSE job_cursor;

END;

/

OUTPUT:

The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'Saran Raj' are on the right. The 'SQL Commands' window shows the PL/SQL code being executed. The 'Results' window displays the output of the program, which lists job IDs and their minimum salaries.

```
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;
```

Results

Job ID	Minimum Salary
sales_rep	2500
sr_rep	2500
sr_account	2500
sr_clerk	2500

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

QUERY:

DECLARE

CURSOR employees_cur IS

SELECT employee_id, last_name, job_id, start_date

FROM employees NATURAL join job_history;

emp_start_date DATE;

BEGIN

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

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

FOR emp_sal_rec IN employees_cur LOOP

-- find out most recent end_date in job_history

SELECT Max(end_date) + 1

INTO emp_start_date

FROM job_history

WHERE employee_id = emp_sal_rec.employee_id;

IF emp_start_date IS NULL THEN

emp_start_date := emp_sal_rec.start_date;

END IF;

dbms_output.Put_line(Rpad(emp_sal_rec.e
mployee_id, 15)

||Rpad(emp_sal_rec.last_name, 25)

|| Rpad(emp_sal_rec.job_id, 35)

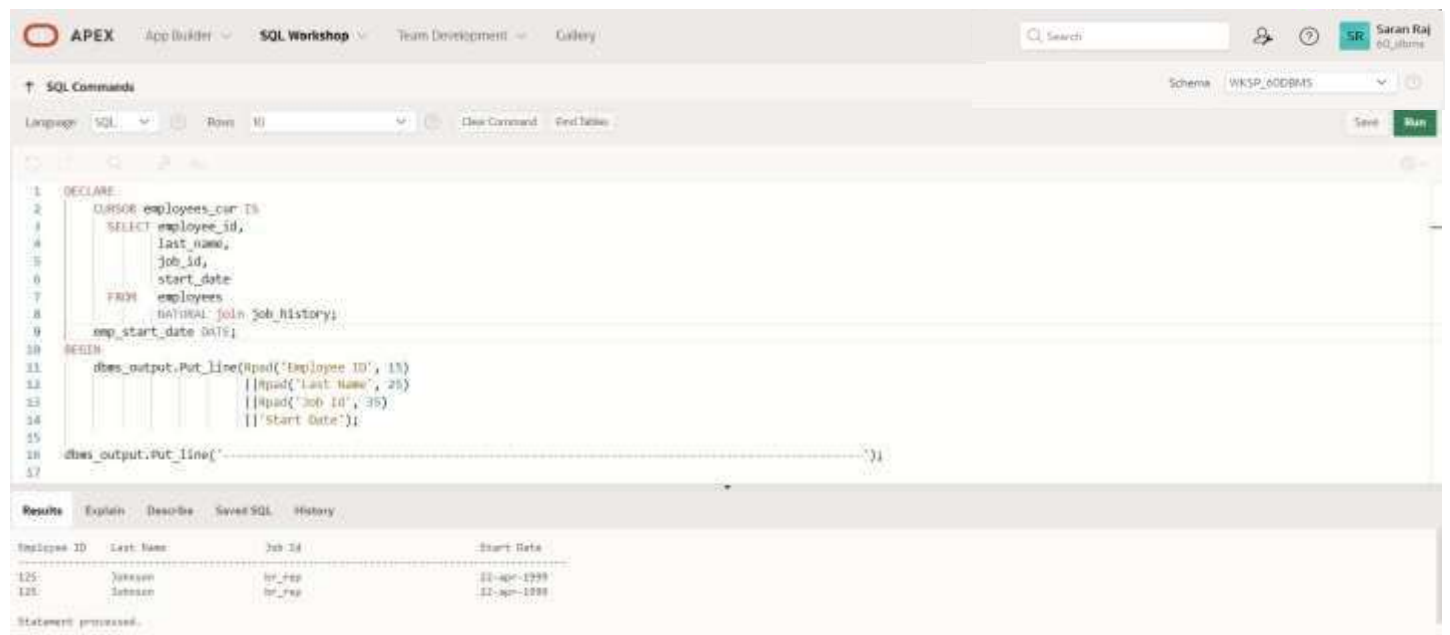
|| To_char(emp_start_date, 'dd-mon-
yyyy'));

END LOOP;

END;

/

OUTPUT:



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

```
1 DECLARE
2   CURSOR employees_cur IS
3     SELECT employee_id,
4            last_name,
5            job_id,
6            start_date
7     FROM   employees
8           NATURAL JOIN job_history;
9   emp_start_date DATE;
10
11 BEGIN
12   dbms_output.put_line('Employee ID: ' ||
13                        ||pad('Last Name', 25)
14                        ||pad('Job ID', 35)
15                        ||'Start Date:');
16   dbms_output.put_line('-----');
17
```

The Results window shows the output of the query:

Employee ID	Last Name	Job ID	Start Date
125	Jackson	hr_rep	22-Apr-1999
126	Johnson	hr_rep	22-Apr-1999

Statement processed.

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

QUERY:

DECLARE

v_employee_id employees.employee_id%TYPE;

v_first_name employees.last_name%TYPE;

v_end_date job_history.end_date%TYPE;

CURSOR c_employees IS

SELECT e.employee_id, e.first_name, jh.end_date

FROM employees e

JOIN job_history jh ON e.employee_id = jh.employee_id;

BEGIN

OPEN c_employees;

FETCH c_employees INTO v_employee_id, v_first_name, v_end_date;

WHILE c_employees%FOUND LOOP

DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_employee_id);

DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_first_name);

DBMS_OUTPUT.PUT_LINE('End Date: ' || v_end_date);

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

FETCH c_employees INTO v_employee_id, v_first_name, v_end_date;

END LOOP;

CLOSE c_employees;

END;

OUTPUT:

APEX

App Builder

SQL Workshop

Team Development

Gallery

Search

Saran Raj

60_jlhrs

SQL Commands

Schema: WKSP_60DBMS

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);
```

Results

Explain

Describe

Saved SQL

History

Employee ID: 125

Employee Name: Emily

End Date: 04/21/1999

Employee ID: 125

Employee Name: Emily

End Date: 05/21/1997

Statement processed.

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;

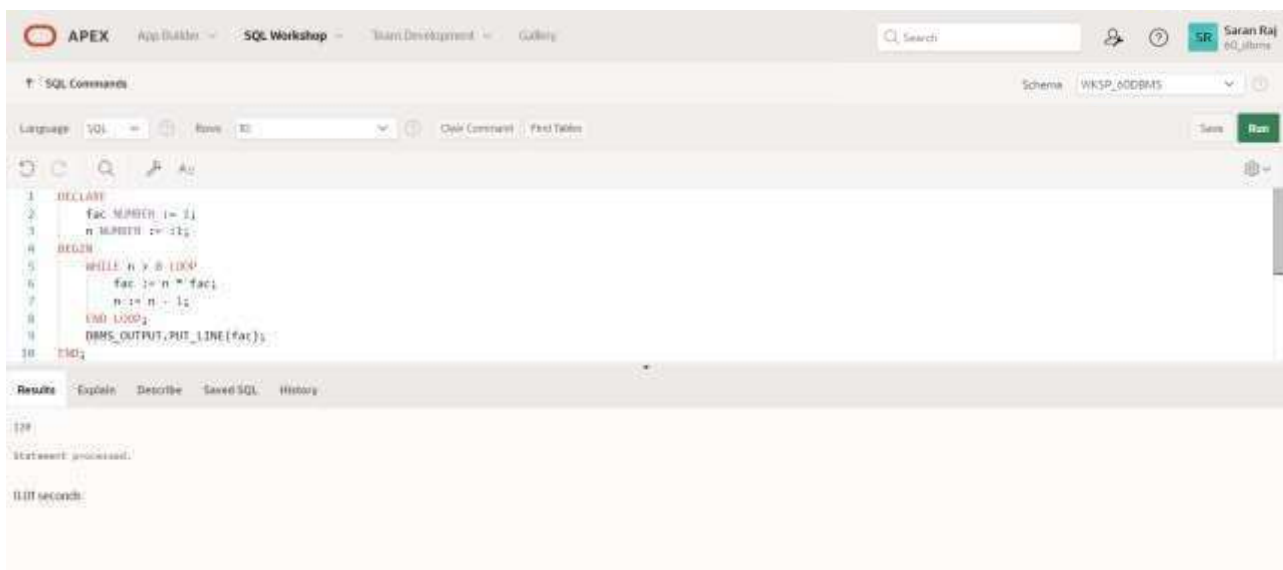
n := n - 1;

END LOOP;

DBMS_OUTPUT.PUT_LINE(fac);

END;

OUTPUT:



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

QUERY:

CREATE OR REPLACE PROCEDURE get_book_info (
p_book_id IN NUMBER,

```

    p_title IN OUT VARCHAR2,
    p_author OUT VARCHAR2,
    p_year_published OUT NUMBER
)
AS
BEGIN
    SELECT title, author, year_published INTO p_title, p_author, p_year_published
    FROM books
    WHERE book_id = p_book_id;

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

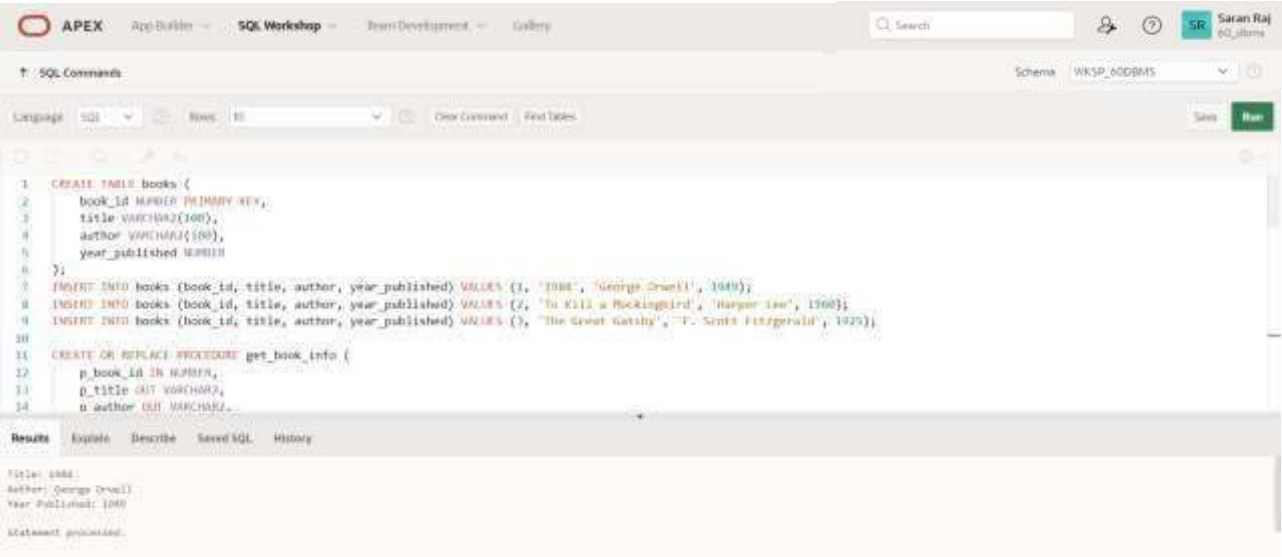
DECLARE
    v_book_id NUMBER := 1;
    v_title VARCHAR2(100);
    v_author VARCHAR2(100);
    v_year_published NUMBER;
BEGIN
    v_title := 'Initial Title';

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

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

```

OUTPUT:



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

TRIGGER

EX_NO: 18

DATE:

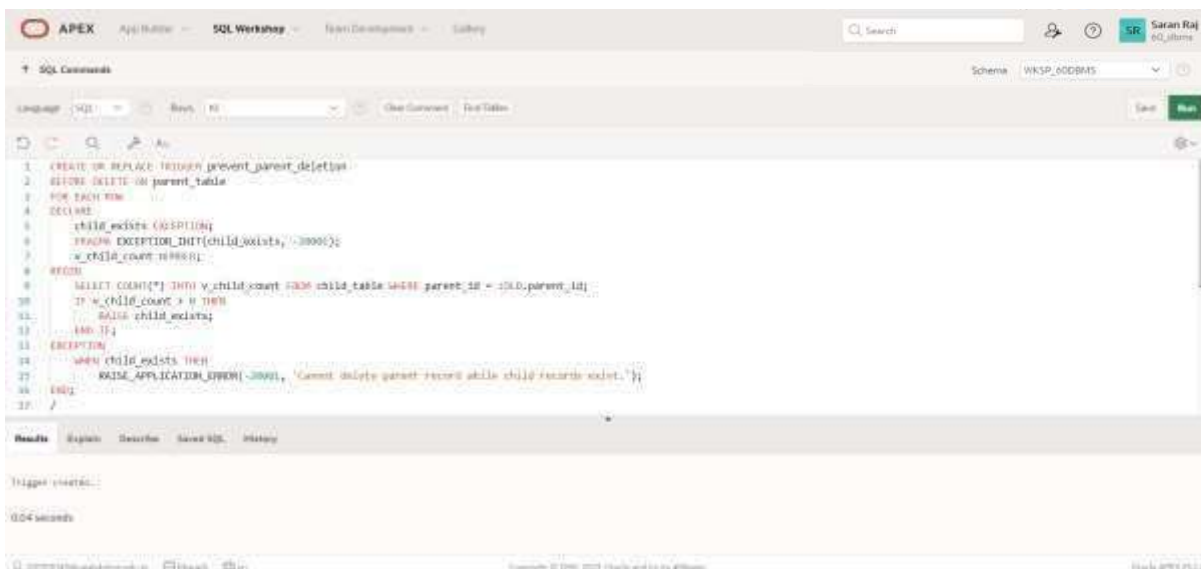
1.) Write a code in PL/SQL to develop a trigger that enforces referential integrity by preventing the deletion of a parent record if child records exist

QUERY:

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

END;

OUTPUT:



```
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

Trigger created.

0.04 seconds

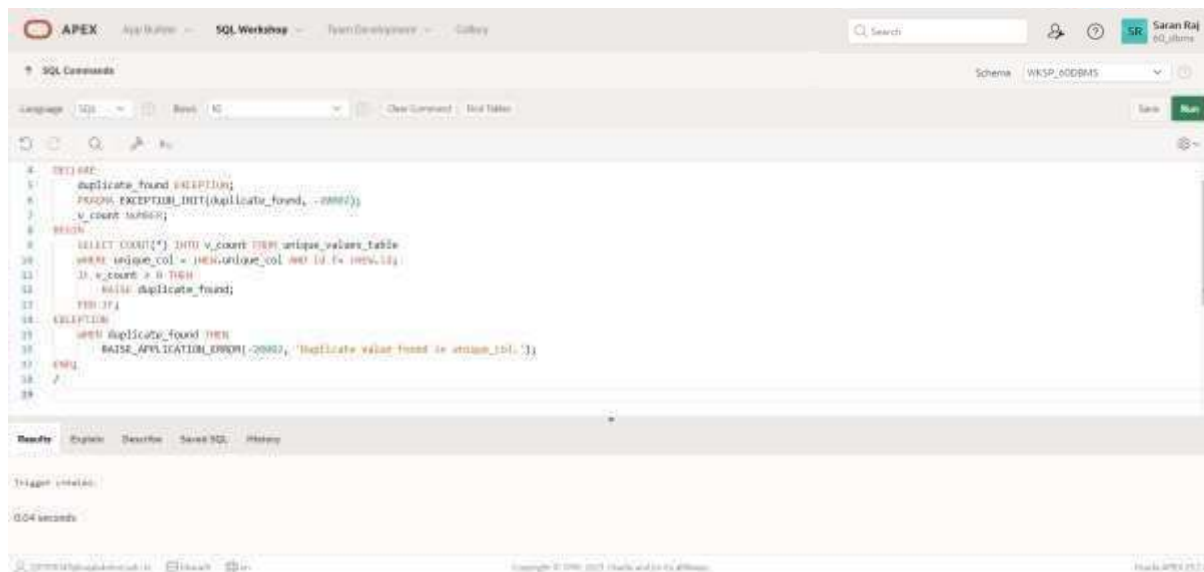
2.) Write a code in PL/SQL to create a trigger that checks for duplicate values in a specific column and raises an exception if found

QUERY:

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

END;

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'Saran Raj' are on the right. The 'SQL Commands' section is active, showing a list of commands with a search bar and 'Clear Command' and 'Find Tables' buttons. The command list includes the trigger creation query. The 'Results' section shows the output: 'Trigger created.' and '0.04 seconds'. The footer contains copyright information and version details.

```
1 11/11/2023 11:11:11 AM
2 duplicate_found EXCEPTION;
3 PRAGMA EXCEPTION_INIT(duplicate_found, -20002);
4 v_count NUMBER;
5 BEGIN
6     SELECT COUNT(*) INTO v_count FROM unique_values_table
7     WHERE unique_col = :NEW.unique_col AND id != :NEW.id;
8     IF v_count > 0 THEN
9         RAISE duplicate_found;
10    END IF;
11 EXCEPTION
12     WHEN duplicate_found THEN
13         RAISE_APPLICATION_ERROR(-20002, 'Duplicate value found in unique_col.');
```

Results

Trigger created.

0.04 seconds

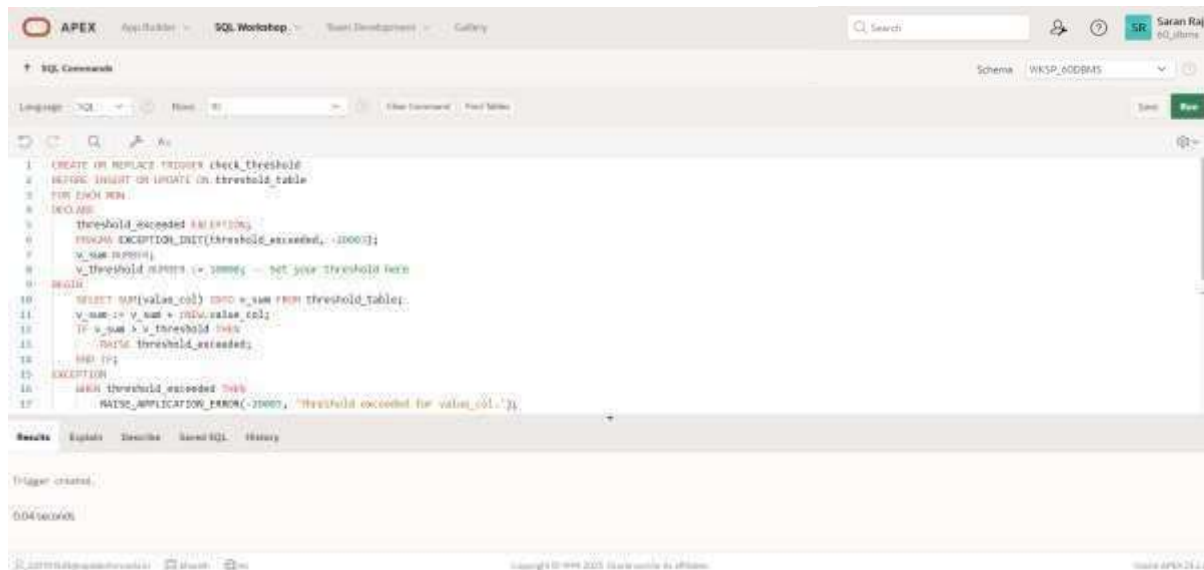
3.) Write a code in PL/SQL to create a trigger that restricts the insertion of new rows if the total of a column's values exceeds a certain threshold

QUERY:

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

END;

OUTPUT:



```
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 Source SQL History

Trigger created.

0.046 seconds

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

QUERY:

CREATE OR REPLACE TRIGGER log_changes

AFTER UPDATE ON main_table

FOR EACH ROW

BEGIN

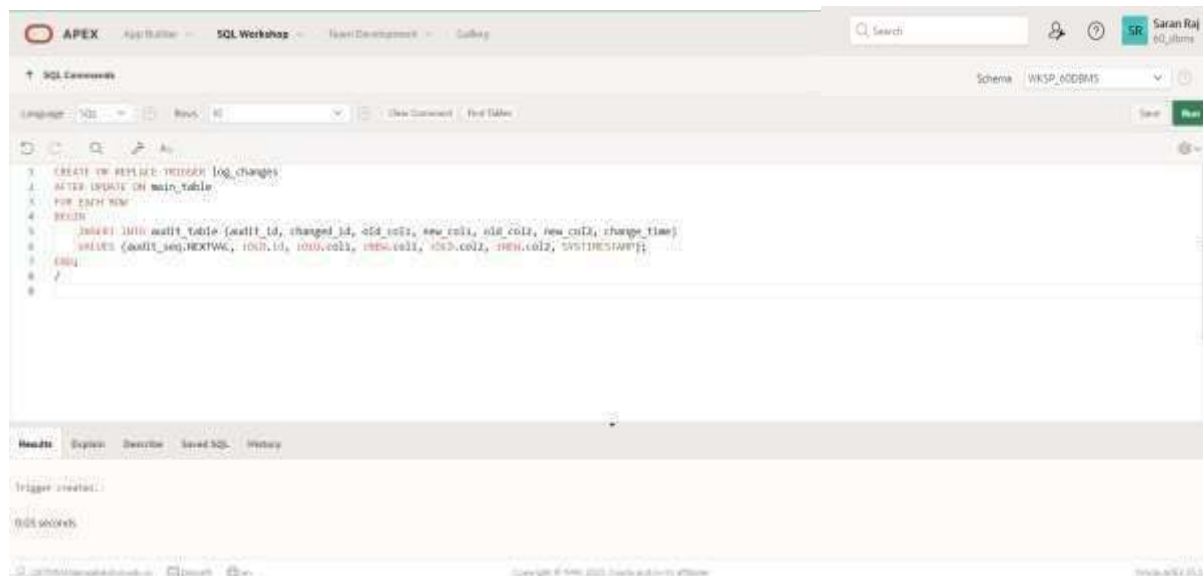
INSERT INTO audit_table (audit_id, changed_id, old_col1, new_col1, old_col2, new_col2, change_time)

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

SYSTIMESTAMP);

END;

OUTPUT:

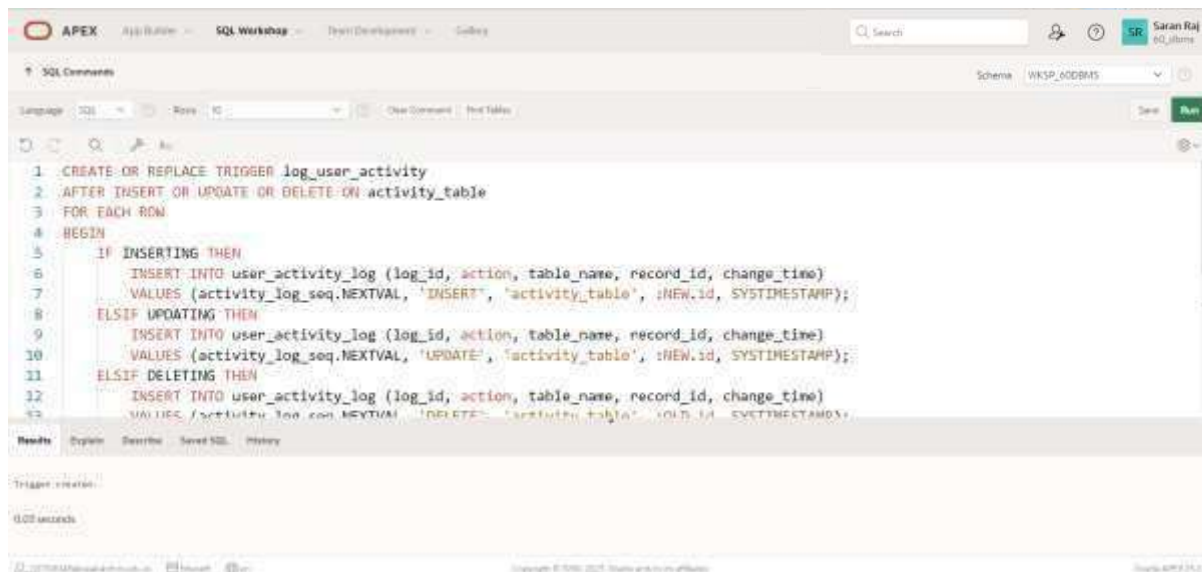


5.) Write a code in PL/SQL to implement a trigger that records user activity (inserts, updates, deletes) in an audit log for a given set of tables.

QUERY:

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

OUTPUT:



The screenshot shows the APEX SQL Workshop interface. The SQL Editor contains the PL/SQL code for the trigger. The Results pane at the bottom shows the output of the execution: 'Trigger created.' and '0.03 seconds'.

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

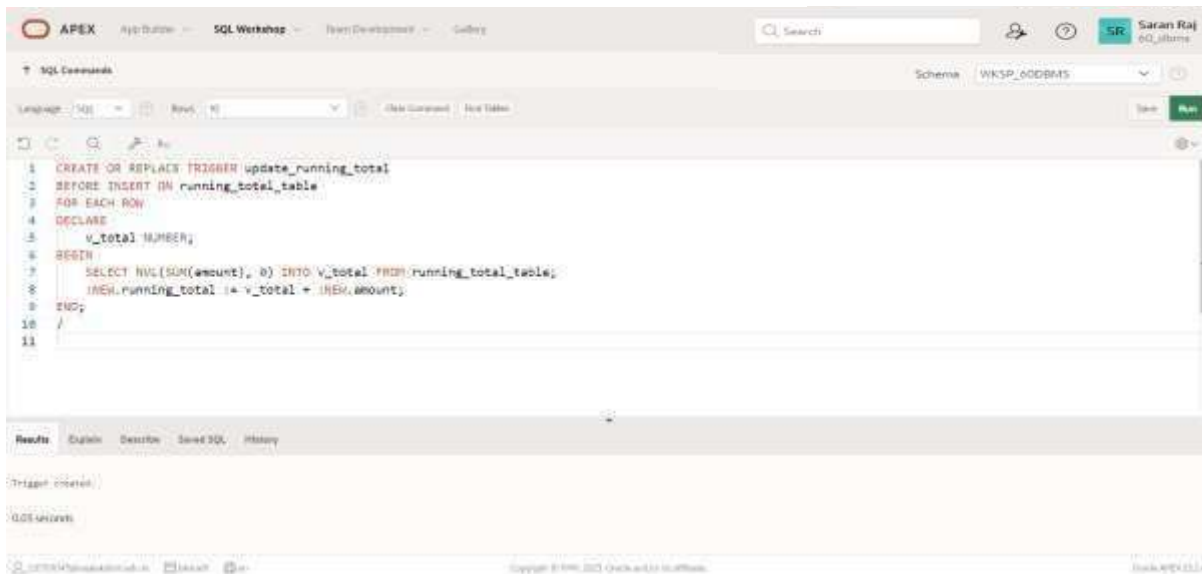
QUERY:

```

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

```

OUTPUT:



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

QUERY:

```

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

END;

OUTPUT:

The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes the APEX logo, 'App Builder', 'SQL Workshop', and 'Tools/Development'. The 'SQL Commands' tab is active, displaying the SQL script for creating the trigger. The script is as follows:

```

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
```

Below the script, the 'Results' tab shows the output: 'Trigger created.' and '0.05 seconds'. The bottom status bar indicates 'Copyright © 1996, 2021 Oracle and/or its affiliates. Oracle APEX 21.0.8'.

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

RESULT:

MONGO DB

EX_NO: 19

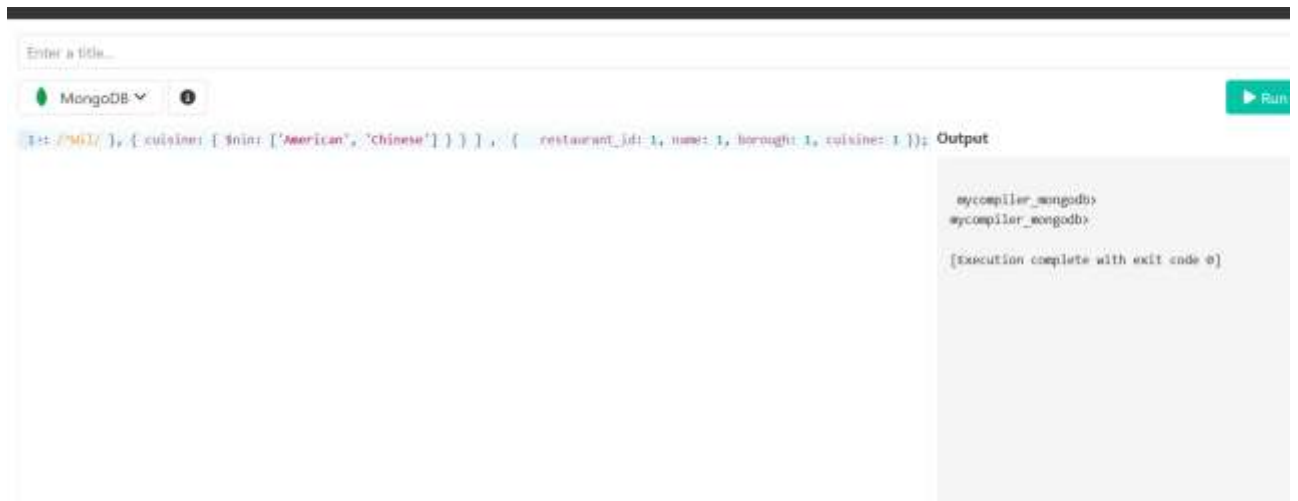
DATE:

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:



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:



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:

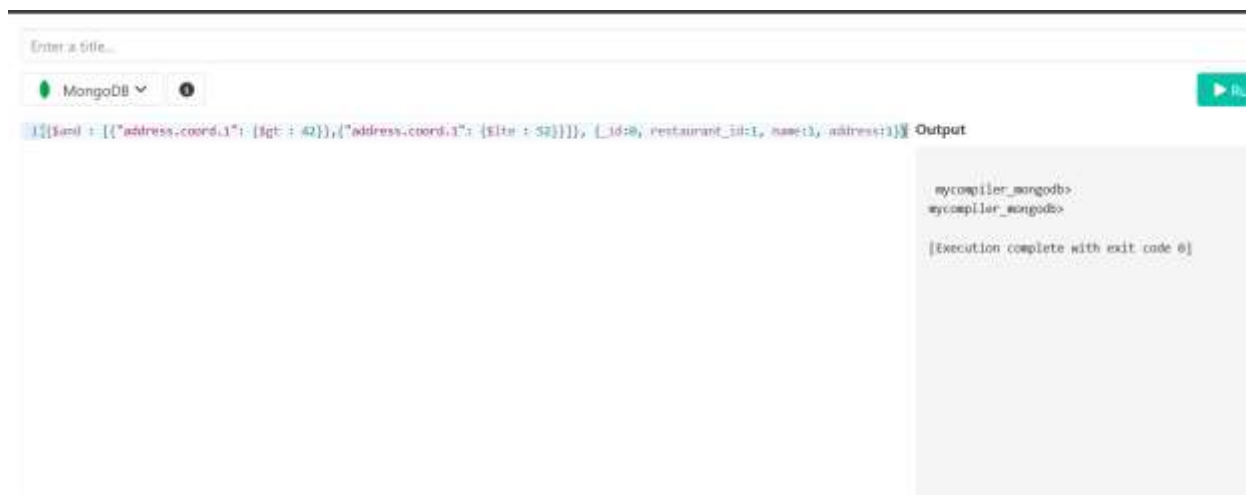


6.) Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.

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 a web-based MongoDB interface. At the top, there is a search bar labeled "Enter a title...". Below it, a dropdown menu shows "MongoDB" with a green leaf icon and a help icon. The main area contains a code editor with the following query: `1 db.restaurants.find({}, { _id: 0 }).sort({ cuisine: 1, borough: -1 })`. To the right of the code editor is an "Output" panel. The output panel shows the command prompt `mycompiler_mongodb>`, the command `mycompiler_mongodb>`, and the message `[Execution complete with exit code 0]`. A green "Run" button is located at the top right of the interface.

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

QUERY:

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

OUTPUT:



The screenshot shows a web-based MongoDB interface. At the top, there is a search bar labeled "Enter a title...". Below it, a dropdown menu shows "MongoDB" with a green leaf icon and a help icon. The main area contains a code editor with the following query: `1 db.restaurants.find({ "address.street": { $exists: true, $ne: "" } })`. To the right of the code editor is an "Output" panel. The output panel shows the command prompt `mycompiler_mongodb>`, the command `mycompiler_mongodb>`, and the message `[Execution complete with exit code 0]`. A green "Run" button is located at the top right of the interface.

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:



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:

Enter a title...

MongoDB ⓘ

Run

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

Output

mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]

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

QUERY:

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

OUTPUT:

Enter a title...

MongoDB ⓘ

Run

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

Output

mycompiler_mongodb>
mycompiler_mongodb>

[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:



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:

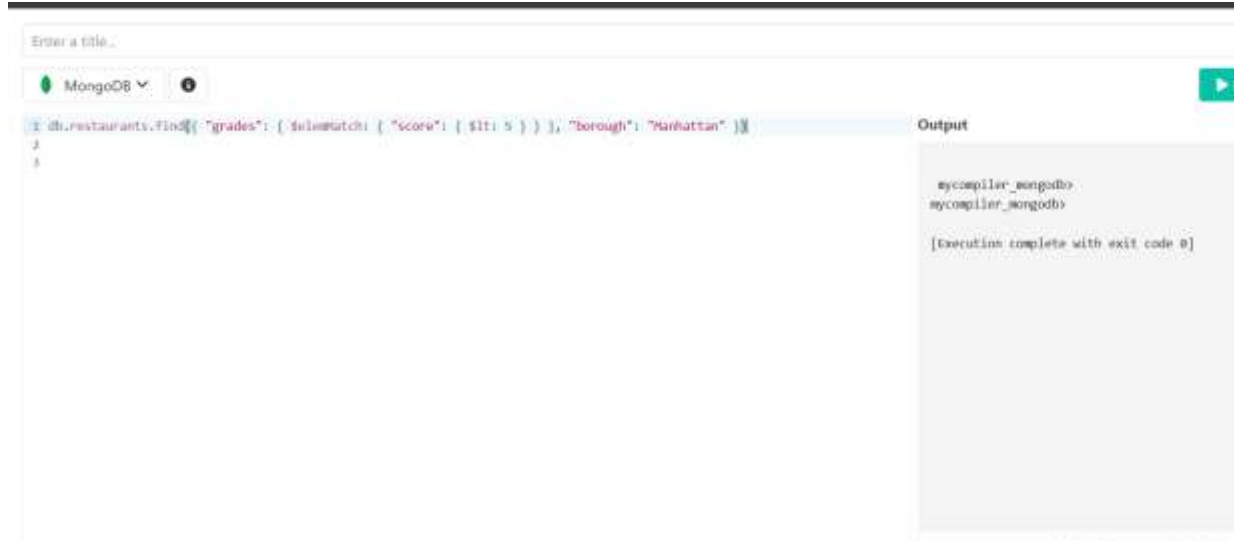


14. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan.

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:

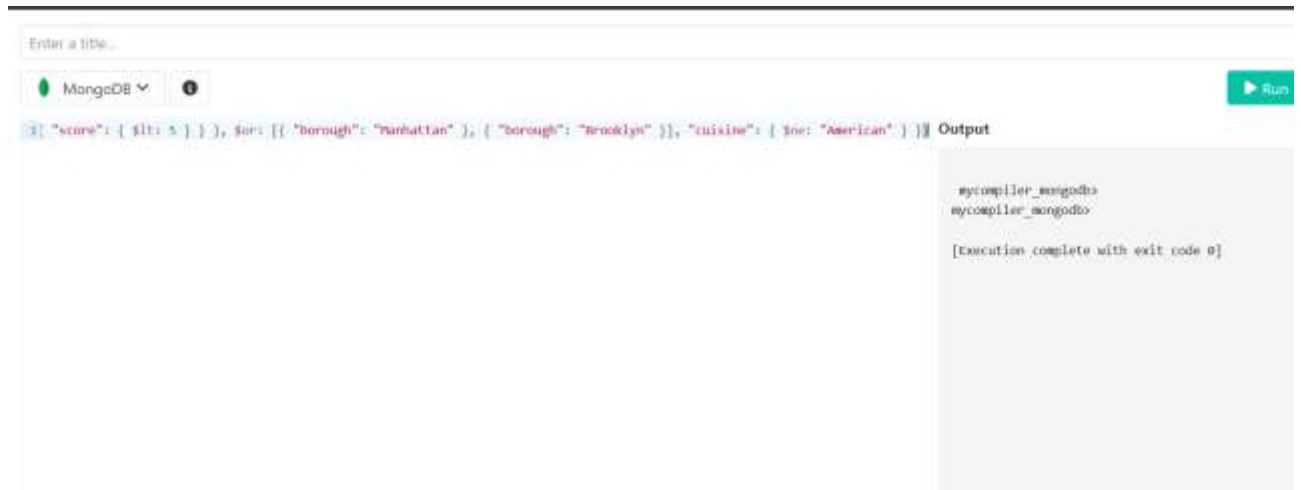


16. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn, and their cuisine is not American.

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:



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:



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 MongoDB Shell interface. At the top, there is a text input field labeled "Enter a title...". Below it, a dropdown menu shows "MongoDB" with a green leaf icon and a help icon. The main area contains a code editor with the query `db.movies.find({ year: 1893 })`. To the right of the code editor is an "Output" panel. The output panel shows the prompt `mycompiler_mongodb>` twice, followed by the message `[Execution complete with exit code 0]`. A green "Run" button is located in the top right corner of the interface.

2.) Find all movies with full information from the 'movies' collection that have a runtime greater than 120 minutes.

QUERY:

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

OUTPUT:



The screenshot shows the MongoDB Shell interface. At the top, there is a text input field labeled "Enter a title...". Below it, a dropdown menu shows "MongoDB" with a green leaf icon and a help icon. The main area contains a code editor with the query `db.movies.find({ runtime: { $gt: 120 } })`. To the right of the code editor is an "Output" panel. The output panel shows the prompt `mycompiler_mongodb>` twice, followed by the message `[Execution complete with exit code 0]`. A green "Run" button is located in the top right corner of the interface.

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:

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:

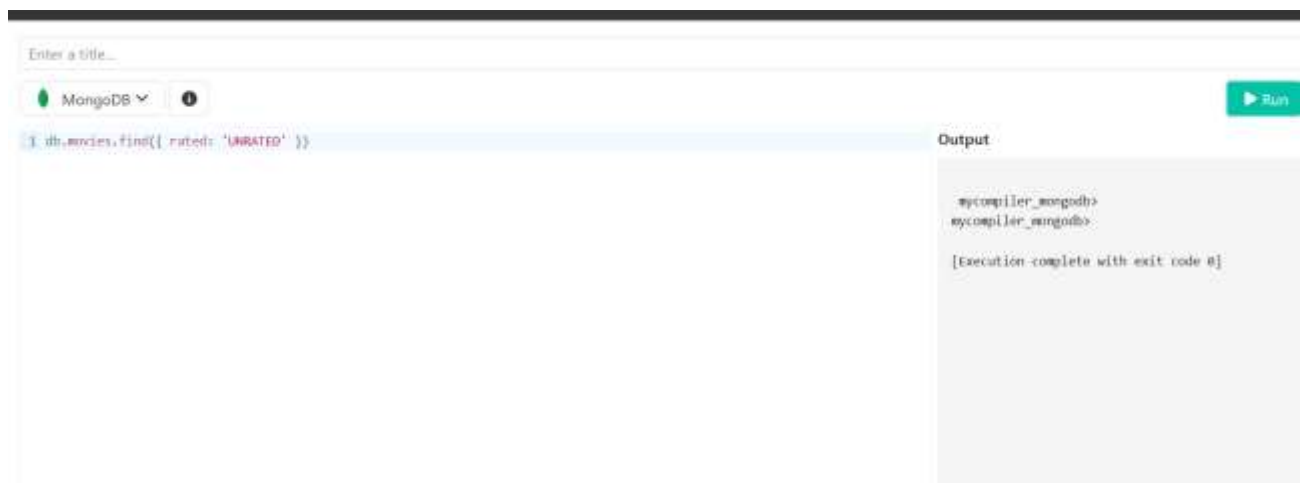


6.) Retrieve all movies from the 'movies' collection that have complete information and are rated as "UNRATED".

QUERY:

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

OUTPUT:



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:

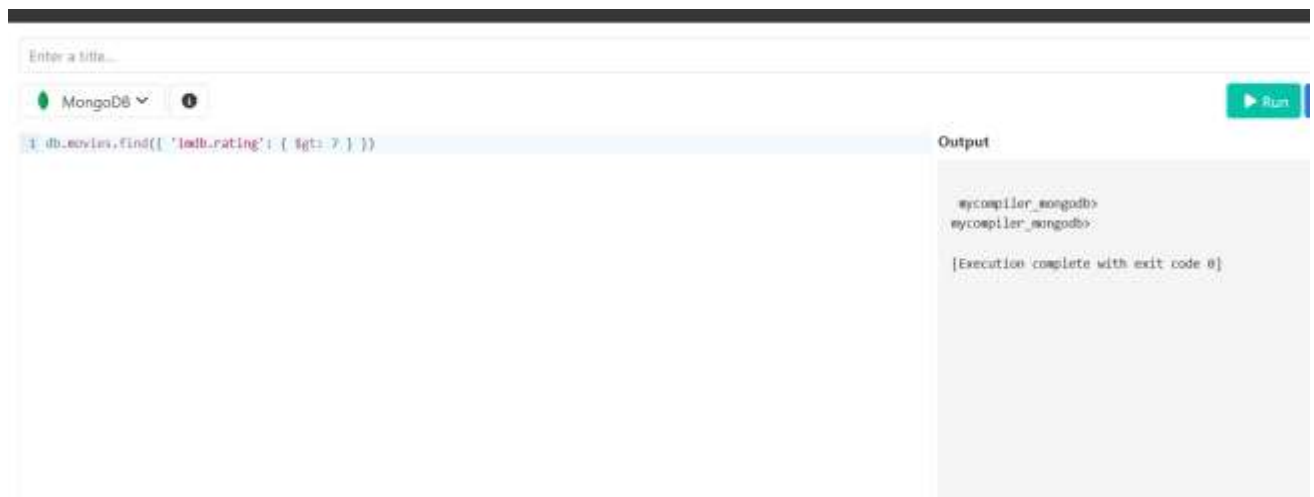


8.) Retrieve all movies from the 'movies' collection that have complete information and have an IMDb rating higher than 7.

QUERY:

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

OUTPUT:



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:



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

QUERY:

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

OUTPUT:



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:

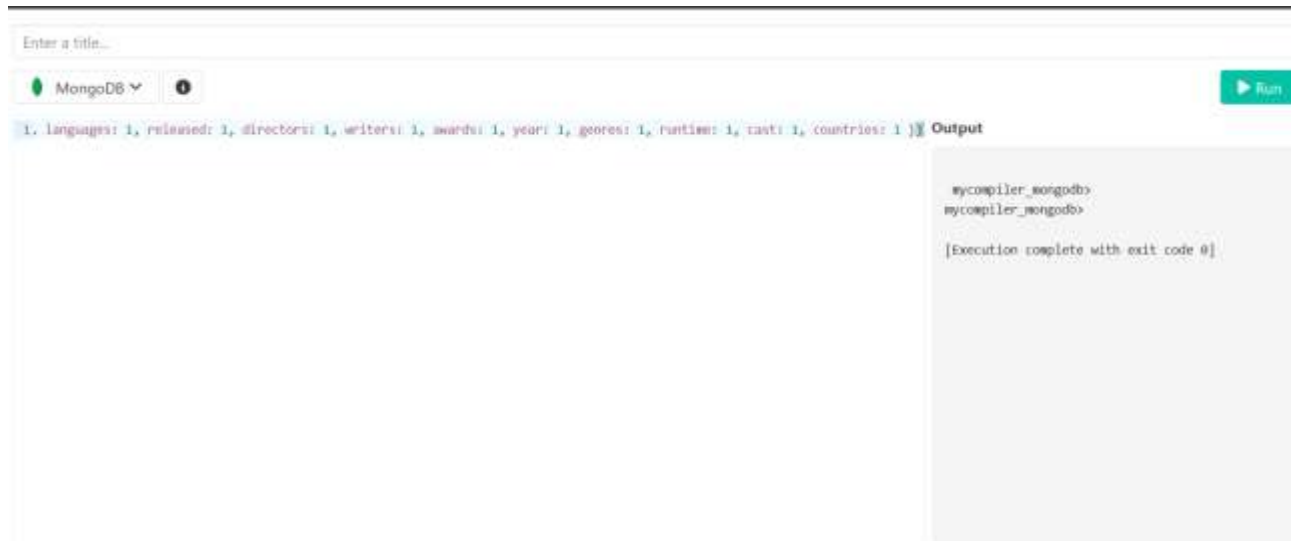


12.) Find all movies with title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries from the 'movies' collection in MongoDB with cast including "Charles Kayser".

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:



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:



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 a web-based MongoDB interface. At the top, there is a search bar labeled "Enter a title...". Below it, a dropdown menu shows "MongoDB" with a green icon. A "Run" button is visible on the right. The main area contains a code editor with the following query:

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

. To the right of the code editor, the "Output" section displays the command prompt output:

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

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

RESULT: