

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),name varchar2(25));
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. At the top, there's a navigation bar with APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there are search, help, and user profile icons. Below the navigation bar, the schema is set to WKSP_BHARGAVI44. The main area is titled 'SQL Commands' and contains a toolbar with icons for Undo, Redo, Find, Replace, and Run. The command input field shows the SQL statement: 'CREATE TABLE dept(id Number(7),name varchar2(25));'. The results tab at the bottom shows the output: 'Table created.' and a execution time of '0.04 seconds'.

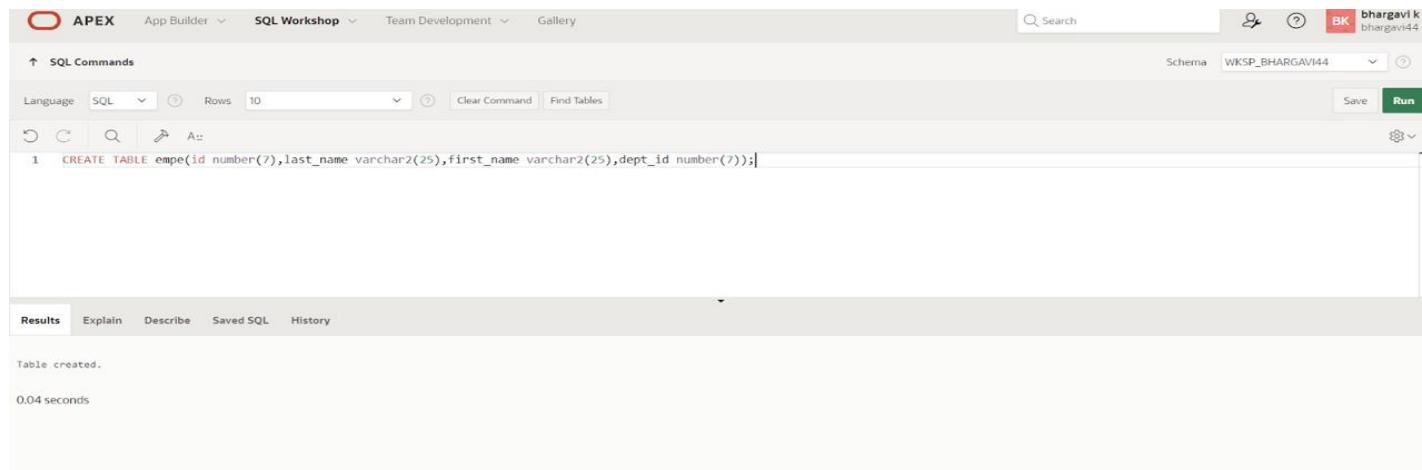
2.Create the EMP table based on the following instance chart. Confirm that the table is created.

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

QUERY:

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

OUTPUT:



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

```
CREATE TABLE emp(id number(7),last_name varchar2(25),first_name varchar2(25),dept_id number(7));
```

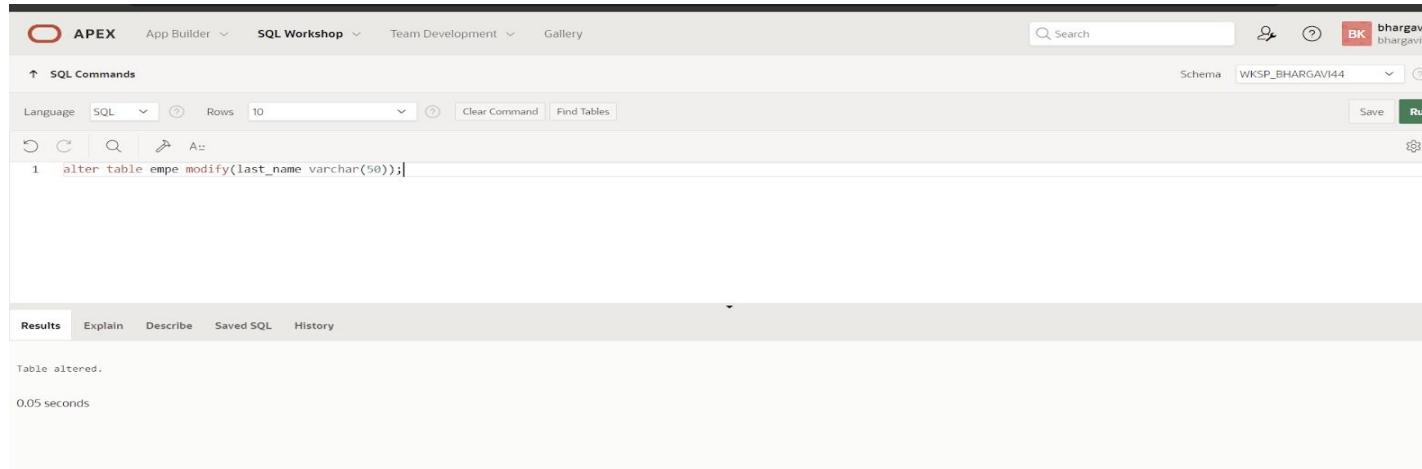
The results pane displays the message "Table created." and a execution time of "0.04 seconds".

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

QUERY:

```
Alter table emp modify(Last_Name varchar2(50));
```

OUTPUT:



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

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

The results pane displays the message "Table altered." and a execution time of "0.05 seconds".

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area displays the SQL command: 'CREATE TABLE empe2(id number(7),last_name varchar2(25),first_name varchar2(25),salary number(9),dept_id number(7));'. Below the command, the 'Results' tab is active, showing the output: 'Table created.' and a execution time of '0.03 seconds'.

5.Drop the EMP table.

QUERY:

Drop table emp;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area displays the SQL command: 'drop table emp;'. Below the command, the 'Results' tab is active, showing the output: 'Table dropped.' and a execution time of '0.08 seconds'.

6.Rename the EMPLOYEES2 table as EMP.

QUERY:

Rename employees2 to emp;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'bhargavi k bhargavi44', and various tool icons. The main area is titled 'SQL Commands' with a 'Schema' dropdown set to 'WKSP_BHARGAVI44'. Below this are tabs for Language (set to SQL), Rows (set to 10), Clear Command, Find Tables, and a Run button. The SQL editor contains the command: '1 rename employees2 to emp;'. The results section below shows the output: 'Statement processed.' and '0.05 seconds'.

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

QUERY:

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

OUTPUT:

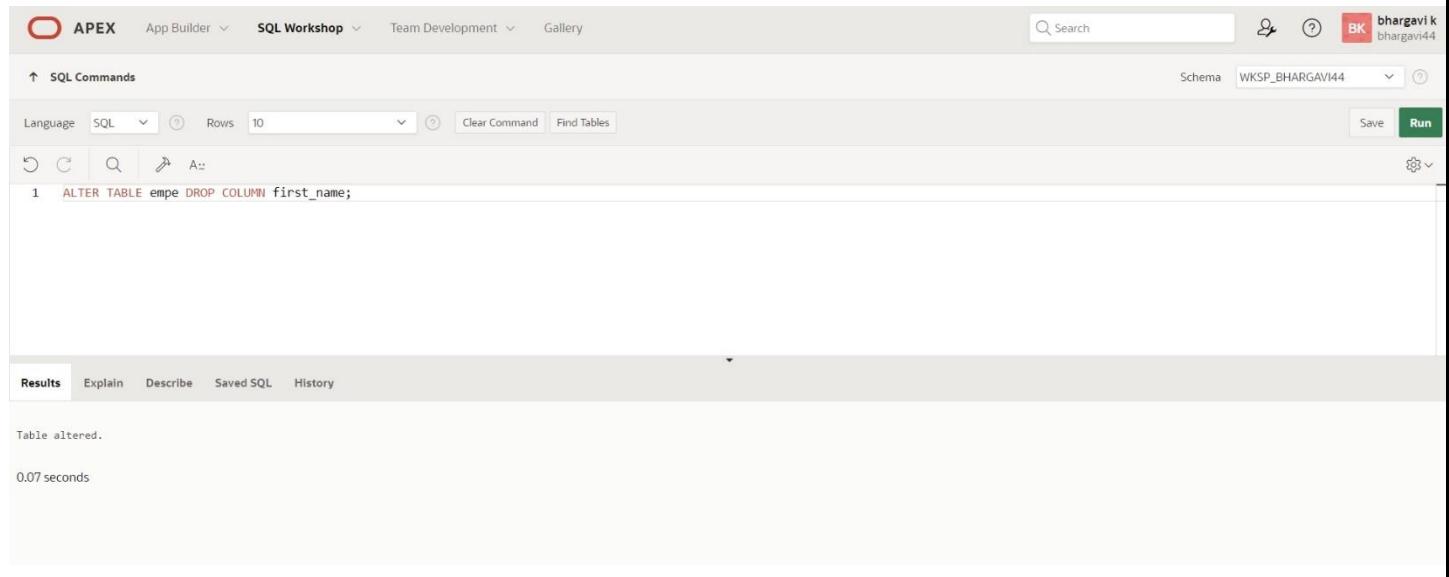
The screenshot shows the Oracle SQL Workshop interface, identical to the previous one but with different SQL commands. The SQL editor contains: '1 comment on table dept is 'Department info'' and '1 comment on table emp is Employee info''. The results section shows 'Statement processed.' and '0.05 seconds'.

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

QUERY:

Alter table emp drop column first_name;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area displays the SQL command: 'ALTER TABLE emp DROP COLUMN first_name;'. Below the command, the output shows 'Table altered.' and a execution time of '0.07 seconds'. The results tab is active.

```
1 ALTER TABLE emp DROP COLUMN first_name;
```

Table altered.
0.07 seconds

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

RESULT:

MANIPULATING DATA

EX_NO:2

DATE:

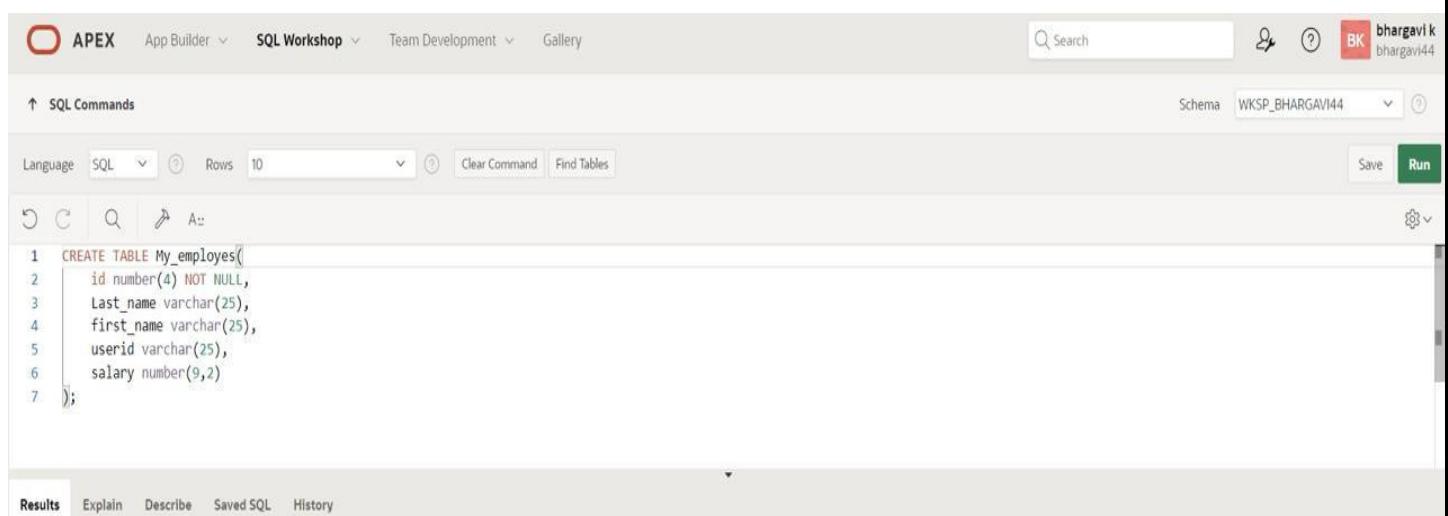
1.Create MY_EMPLOYEE table with the following structure

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

QUERY:

```
Create table my_employee(id number(4),Last_Name varchar2(25),First_Name varchar2(25),Userid varchar2(25),Salary number(9,2));
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, tabs for 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery' are visible. A search bar and user information ('bhargavi k bhargavi44') are on the right. The main area is titled 'SQL Commands'. It shows a code editor with the following SQL command:

```
1 CREATE TABLE My_employees(
2     id number(4) NOT NULL,
3     Last_name varchar(25),
4     first_name varchar(25),
5     userid varchar(25),
6     salary number(9,2)
7 );
```

Below the code editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, displaying the message 'Table created.' and a timestamp '0.03 seconds'.

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

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

QUERY:

```
Insert into my_employee values(1,'Patel','Ralph','rpatel',895);
```

```
Insert into my_employee values(1,'Dancs','Betty','bdancs',860);
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands tab, two INSERT statements are executed:

```
1 INSERT INTO My_employees (id ,last_name,first_name,userid,salary)VALUES(1,'Patel','Ralph','rpatel',895)
2
```

In the Results tab, the output is shown:

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

```
0.04 seconds
```

3.Display the table with values.

QUERY:

```
Select*from my_employee;
```

OUTPUT

The screenshot shows the Oracle SQL Workshop interface. A SELECT statement is executed:

```
1 SELECT*FROM My_employees|order by(id);
```

In the Results tab, the output is shown:

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860

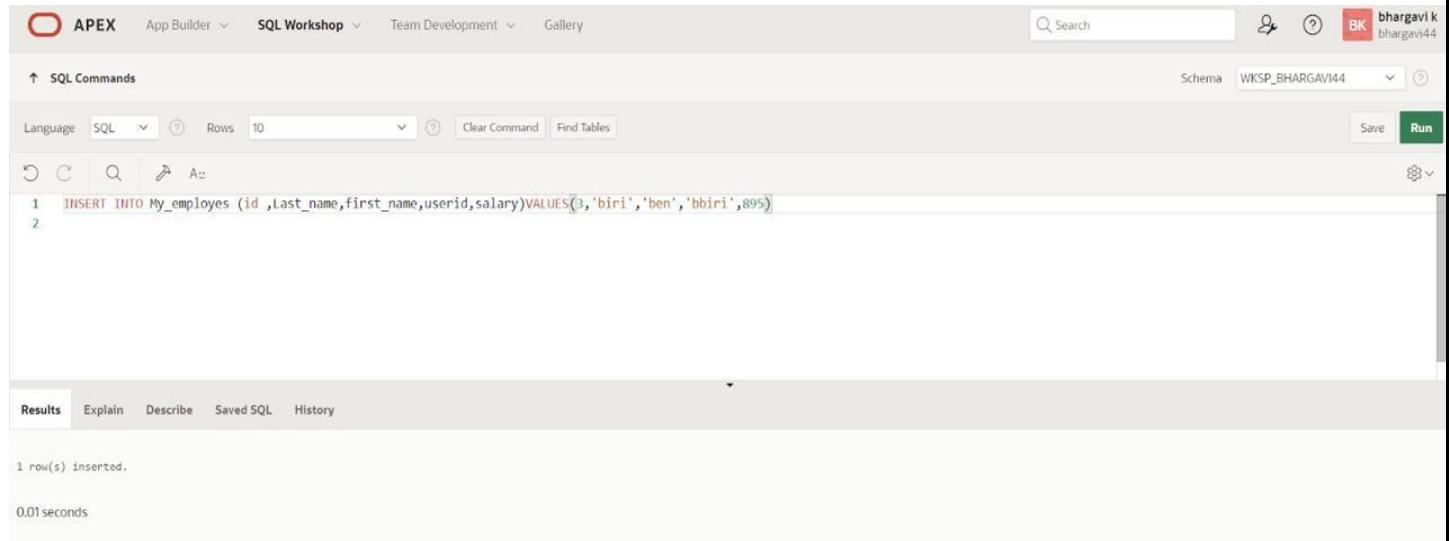
2 rows returned in 0.01 seconds

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

QUERY:

```
Insert into my_employee values(3,'Biri','Ben','bbiril',1100);  
Insert into my_employee values(4,'Newman','Chad','cnewman',750);
```

OUTPUT:



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

```
1 INSERT INTO My_employees (id ,last_name,first_name,userid,salary)VALUES(3,'biri','ben','bbiri',895)  
2
```

Below the code, the results section shows:

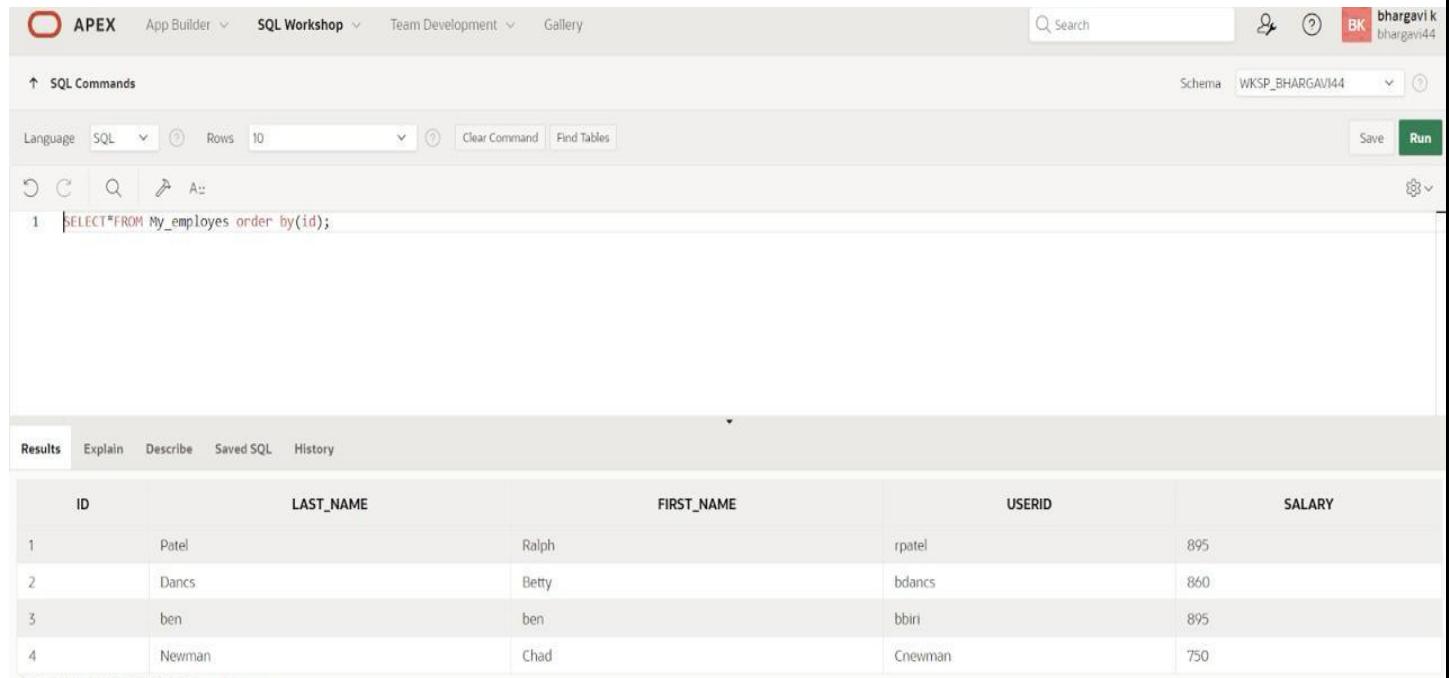
```
1 row(s) inserted.  
0.01 seconds
```

5.Make the data additions permanent.

QUERY:

```
Select*from my_employee;
```

OUTPUT:



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

```
1 |SELECT*FROM My_employees order by(id);
```

Below the code, the results section displays a table with the following data:

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	ben	ben	bbiri	895
4	Newman	Chad	Cnewman	750

At the bottom of the results section, it says "4 rows returned in 0.00 seconds".

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

QUERY:

Update my_employee set last_name='Drexler' where id=3;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user information ('bhargavi k bhargavi44') are on the right. The main area is titled 'SQL Commands' with tabs for 'Language' (set to 'SQL'), 'Rows' (set to 10), and buttons for 'Clear Command' and 'Find Tables'. Below this is a toolbar with icons for copy, paste, search, and refresh. The SQL command input field contains the following code:

```
1 UPDATE My_employees SET Last_name='Drexler' WHERE id=3;
```

Below the command, the results section shows the output:

1 row(s) updated.

0.00 seconds

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

QUERY:

update my_employee set salary=1000 where salary

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user information ('bhargavi k bhargavi44') are on the right. The main area is titled 'SQL Commands' with tabs for 'Language' (set to 'SQL'), 'Rows' (set to 10), and buttons for 'Clear Command' and 'Find Tables'. Below this is a toolbar with icons for copy, paste, search, and refresh. The SQL command input field contains the following code:

```
1 UPDATE My_employees SET salary=1000 WHERE salary<900;
```

Below the command, the results section shows the output:

4 row(s) updated.

0.01 seconds

8.Delete Betty dancs from MY_EMPLOYEE table.

QUERY:

```
delete from my_employee where last_name='Dancs';
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'bhargavi k', and a schema dropdown set to 'WKSP_BHARGAVI44'. The main area is titled 'SQL Commands' with a 'Run' button. Below it, the command 'DELETE FROM My_employees WHERE id=2;' is entered. The results section shows '1 row(s) deleted.' and a execution time of '0.00 seconds'.

```
1 DELETE FROM My_employees WHERE id=2;
```

1 row(s) deleted.
0.00 seconds

9.Empty the fourth row of the emp table.

QUERY:

```
Delete from emp where id=4;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'bhargavi k', and a schema dropdown set to 'WKSP_BHARGAVI44'. The main area is titled 'SQL Commands' with a 'Run' button. Below it, the command 'DELETE FROM My_employees WHERE id=4;' is entered. The results section shows '1 row(s) deleted.' and a execution time of '0.00 seconds'.

```
1 DELETE FROM My_employees WHERE id=4;
```

1 row(s) deleted.
0.00 seconds

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

RESULT:

INCLUDING CONSTRAINTS

EX_NO:3

DATE:

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

QUERY:

```
alter table emp add constraint my_emp_id_pk primary key(id);
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is active. The main area is titled 'SQL Commands'. The query entered is: `1 alter table emp add constraint my_emp_id_pk primary key(id);`. The results section shows the output: 'Table altered.' and '0.08 seconds'. The bottom status bar includes the user information '220701044@rajalakshmi.edu.in bhargavi44 en' and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' The version 'Oracle APEX 23.2.4' is also visible.

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 emp add constraint my_dept_id_pk primary key(id);
```

OUTPUT

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is active. The main area is titled 'SQL Commands'. The query entered is: `1 alter table dept add constraint my_dept_id_pk primary key(id);`. The results section shows the output: 'Table altered.' and '0.07 seconds'. The bottom status bar includes the user information '220701044@rajalakshmi.edu.in bhargavi44 en' and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' The version 'Oracle APEX 23.2.4' is also visible.

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

QUERY:

```
alter table emp add constraint my_emp_dept_id_fk foreign key(dept_id) references emp(id);
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, a user icon, and a session identifier 'bhargavi44'. The main area is titled 'SQL Commands' with tabs for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. Below this is a toolbar with icons for copy, paste, search, and run. The SQL command entered is: `1 alter table emp add constraint my_emp_dept_id_fk foreign key(dept_id) references emp(id);`. The results section shows the output: 'Table altered.', '0.08 seconds', and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' The bottom right corner indicates 'Oracle APEX 23.2.4'.

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

QUERY:

```
Alter table emp add constraint number(2,2) check(commission);
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, a user icon, and a session identifier 'bhargavi44'. The main area is titled 'SQL Commands' with tabs for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. Below this is a toolbar with icons for copy, paste, search, and run. The SQL command entered is: `1 alter table emp add commission number(2,2) check(commission>0);`. The results section shows the output: 'Table altered.', '0.06 seconds', and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' The bottom right corner indicates 'Oracle APEX 23.2.4'.

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 employee_id, last_name, sal*12 as "ANNUAL SALARY" from employees;

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

QUERY:

Desc dept;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user profile 'bhargavi k bhargavi44', and a schema dropdown set to 'WKSP_BHARGAVI44'. Below the toolbar, the SQL Commands section is active, with Language set to SQL, Rows to 10, and a Run button. The SQL editor contains the command '1 desc dept;'. The results tab is selected, showing the description of the 'DEPT' table with four columns: ID, NAME, MANAGER_ID, and LOCATION_ID. The table details are as follows:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPT	ID	NUMBER	-	7	0	1	-	-	-
	NAME	VARCHAR2	25	-	-	-	✓	-	-
	MANAGER_ID	NUMBER	22	-	-	-	✓	-	-
	LOCATION_ID	NUMBER	22	-	-	-	✓	-	-

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

QUERY:

Select employee_id, last_name, job_id, hire_date from employees;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'bhargavi44', and a 'Run' button. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab shows the query: 'select emp_id, last_name, job_code, hire_date from emp2 order by emp_id;'. The Results tab displays a table with three rows:

EMP_ID	LAST_NAME	JOB_CODE	HIRE_DATE
1	ram	12	01/12/2002
2	rem	11	03/15/2004
3	riya	18	10/13/2003

Below the table, it says '3 rows returned in 0.01 seconds' and has a 'Download' link.

4.Provide an alias STARTDATE for the hire date.

QUERY:

select hire_date as "STARTDATE" from employees;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface, similar to the previous one. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'bhargavi44', and a 'Run' button. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab shows the query: 'select hire_date as start_date from emp2;'. The Results tab displays a table with three rows:

START_DATE
01/12/2002
10/13/2003
03/15/2004

Below the table, it says '3 rows returned in 0.01 seconds' and has a 'Download' link.

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

QUERY:

select distinct job_id from employees;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user information ('bhargavi k bhargavi44') are also present. The main area has tabs for 'SQL Commands' and 'Results'. The SQL command entered is 'select distinct job_code from emp2;'. The results section displays the column 'JOB_CODE' with three rows: 12, 11, and 18. A note at the bottom says '3 rows returned in 0.01 seconds'.

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

QUERY:

Select last_name||','||job_id as "EMPLOYEE_AND_TITLE" from employees;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user information ('bhargavi k bhargavi44') are also present. The main area has tabs for 'SQL Commands' and 'Results'. The SQL command entered is 'select last_name||','||job_code as "employee_and_title" from emp2;'. The results section displays the column 'employee_and_title' with three rows: ram,12; riya,18; rem,11. A note at the bottom says '3 rows returned in 0.00 seconds'.

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

```
employee_id||'||first_name||'||last_name||'||email||'||phone_no||'||hire_date||'||job_id||'||salary||'||commission  
_pct||'||manager_id||'||department_id as THE_OUTPUT from employees;
```

OUTPUT:

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

```
1 Select emp_id||'||first_name||'||last_name||'||hire_date||'||job_code||'||salary||' as THE_OUTPUT from emp2 ;
```

In the results section, the output is displayed under the heading 'THE_OUTPUT':

THE_OUTPUT
1,reena,ram,01/12/2002,12,5654,
3,ram,riya,10/15/2005,18,56467,
2,riya,rem,03/15/2004,11,76569,

Below the table, it says '3 rows returned in 0.01 seconds'. There is a 'Download' link next to the table.

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

RESULT:

RESTRICTING AND SORTING DATA

EX_NO:5

DATE:

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

QUERY:

Select last_name from emp2 where salary>12000

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'bhargavi k' (bhargavi44). The main area has a search bar and a schema dropdown set to 'WKSP_BHARGAVI44'. The SQL Commands panel shows the query: 'Select last_name from emp2 where salary>12000'. The Results tab displays the output:

LAST_NAME
riya
rem

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

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

QUERY:

Select last_name,department_id from emp2 where emp_id=176;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'bhargavi k' (bhargavi44). The main area has a search bar and a schema dropdown set to 'WKSP_BHARGAVI44'. The SQL Commands panel shows the query: 'Select last_name,department_id from emp2 where emp_id=176;'. The Results tab displays the output:

LAST_NAME	DEPARTMENT_ID
no data found	

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

QUERY:

```
select last_name,salary from emp2 where salary not between 5000 and 12000
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, a user icon for 'bhargavi k bhargav144', and a schema dropdown set to 'WKSP_BHARGAVI144'. The main area is titled 'SQL Commands' with a language dropdown set to 'SQL'. It shows a command line with two rows of code: '1 select last_name,salary from emp2 where salary not between 5000 and 12000' and '2'. Below the command line is a results grid with columns 'LAST_NAME' and 'SALARY'. Two rows are returned: 'riya' with a salary of 56467 and 'rem' with a salary of 76569. At the bottom left, it says '2 rows returned in 0.00 seconds' and there's a 'Download' link.

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

QUERY:

```
Select last_name,job_code,hire_date from emp2 where hire_date between 'February,20,1998' and 'May,1,1998';
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, a user icon for 'bhargavi k bhargav144', and a schema dropdown set to 'WKSP_BHARGAVI144'. The main area is titled 'SQL Commands' with a language dropdown set to 'SQL'. It shows a command line with one row of code: '1 Select last_name,job_code,hire_date from emp2 where hire_date between 'February,20,1998' and 'May,1,1998'';. Below the command line is a results grid. At the bottom left, it says 'no data found'.

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

QUERY:

```
select last_name,department_id from emp2 where department_id in(20,50) order by last_name
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown is set to WKSP_BHARGAVI44. The main area displays the following SQL command:

```
1 select last_name,department_id from emp2 where department_id in(20,50) order by last_name;
```

The results section shows the output:

LAST_NAME	DEPARTMENT_ID
riya	20

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

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

QUERY:

```
select last_name as "EMPLOYEE",salary as "MONTHLY SALARY" from emp2 where (salary between 5000  
and 12000) and (department_id in(20,50)) order by last_name asc;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown is set to WKSP_BHARGAVI44. The main area displays the following SQL command:

```
1 select last_name as "EMPLOYEE",salary as "MONTHLY SALARY" from emp2 where (salary between 5000  
and 12000) and (department_id in(20,50)) order by last_name asc;
```

The results section shows the output:

EMPLOYEE	MONTHLY SALARY

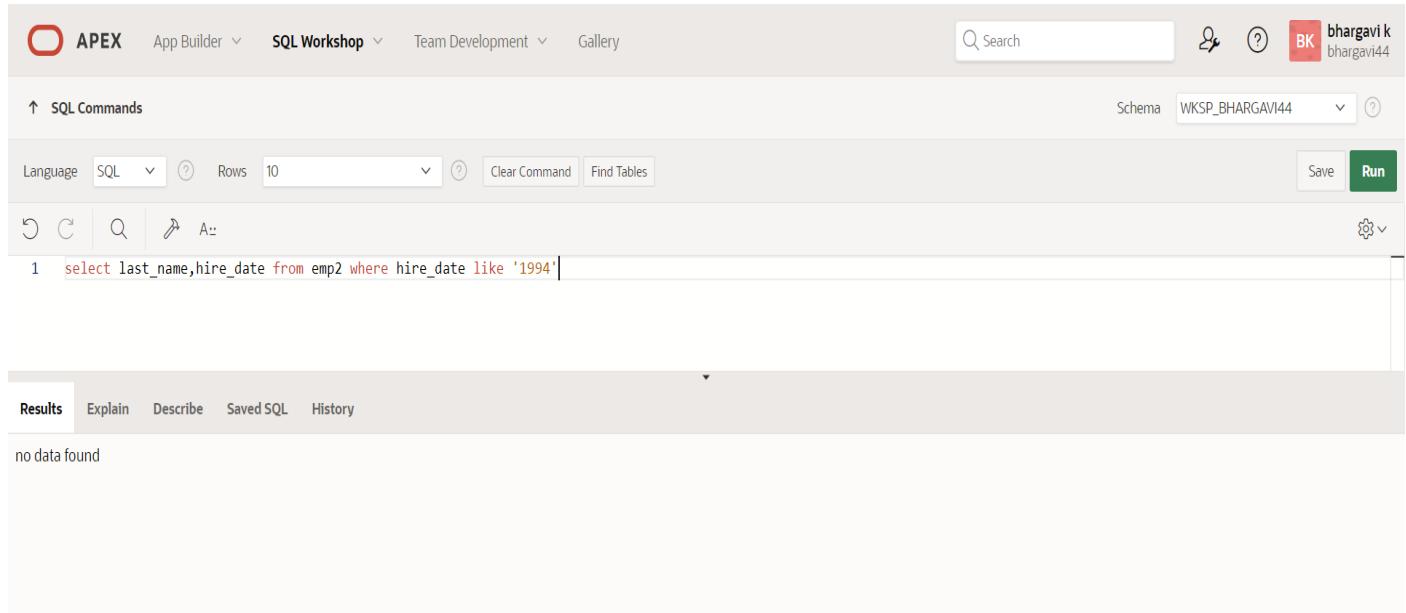
no data found

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

QUERY:

```
select last_name,hire_date from emp2 where hire_date like '1994'
```

OUTPUT:



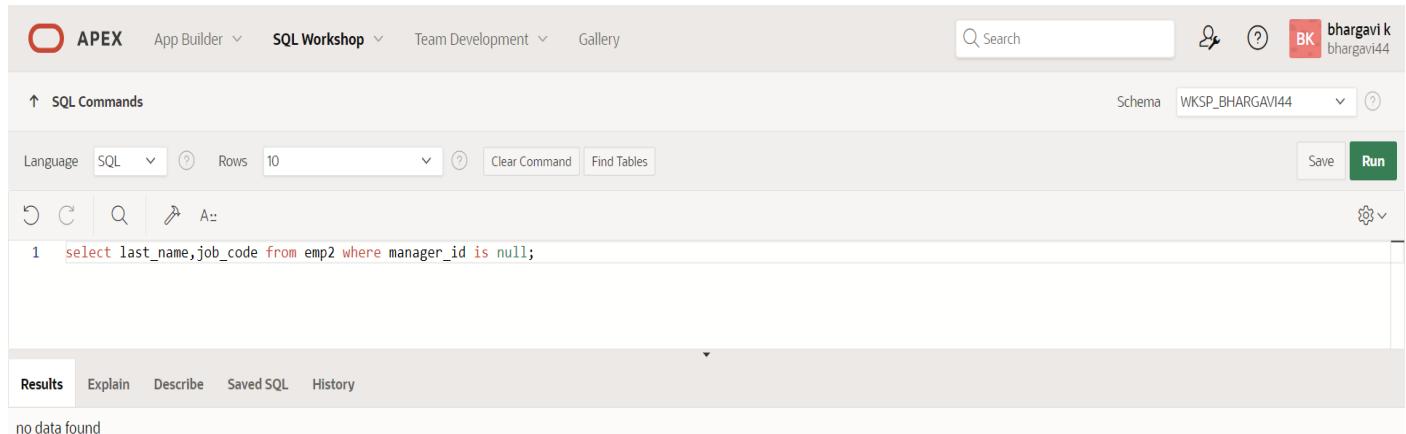
The screenshot shows the Oracle SQL Workshop interface. At the top, there are tabs for 'APEX', 'App Builder', 'SQL Workshop' (which is selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, a user icon for 'bhargavi k bhargavi44', and a 'Run' button. Below the toolbar, the schema is set to 'WKSP_BHARGAVI44'. The main area has a 'Language' dropdown set to 'SQL', a 'Rows' dropdown set to '10', and buttons for 'Clear Command' and 'Find Tables'. The SQL command 'select last_name,hire_date from emp2 where hire_date like '1994'' is entered in the command line. The results section below shows the message 'no data found'.

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

QUERY:

```
select last_name,job_code from emp2 where manager_id is null;
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface, identical to the previous one but with a different query. The schema is still 'WKSP_BHARGAVI44'. The SQL command 'select last_name,job_code from emp2 where manager_id is null;' is entered. The results section shows the message 'no data found'.

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

QUERY:

```
select last_name,salary,commission_pct from emp2 where commission_pct is not null order by salary desc;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'bhargavi k bhargavi44', and a 'Run' button. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the query: 'select last_name,salary,commission_pct from emp2 where commission_pct is not null order by salary desc;'. The Results tab displays a table with three rows:

LAST_NAME	SALARY	COMMISSION_PCT
rem	76569	355
riya	56467	3444
ram	5654	4435

Below the table, it says '3 rows returned in 0.01 seconds' and has a 'Download' link.

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

QUERY:

```
select last_name from emp2 where last_name like '_a%';
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface, identical to the previous one but with a different query. The SQL Commands tab contains the query: 'select last_name from emp2 where last_name like '_a%';'. The Results tab shows the message 'no data found'.

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

QUERY:

```
select last_name from emp2 where last_name like '%a%' and last_name like '%e%';
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows a user profile for 'bhargavi k bhargavi44'. The main area is titled 'SQL Commands' with a 'Schema' dropdown set to 'WKSP_BHARGAVI44'. The command input field contains the query: 'select last_name from emp2 where last_name like '%a%' and last_name like '%e%';'. Below the input field, there are buttons for Save and Run. The results section is currently empty, displaying 'no data found'.

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

QUERY:

```
select last_name,job_code,salary from emp2 where job_code in ('sales representative','stock clerk') and salary not in(2500,3500,7000);
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows a user profile for 'bhargavi k bhargavi44'. The main area is titled 'SQL Commands' with a 'Schema' dropdown set to 'WKSP_BHARGAVI44'. The command input field contains the query: 'select last_name,job_code,salary from emp2 where job_code in ('sales representative','stock clerk') and salary not in(2500,3500,7000);'. Below the input field, there are buttons for Save and Run. The results section shows a table with three columns: LAST_NAME, JOB_CODE, and SALARY. The single row returned is: ram, sales representative, 5654. At the bottom, it says '1rows returned in 0.00 seconds' and has a 'Download' link.

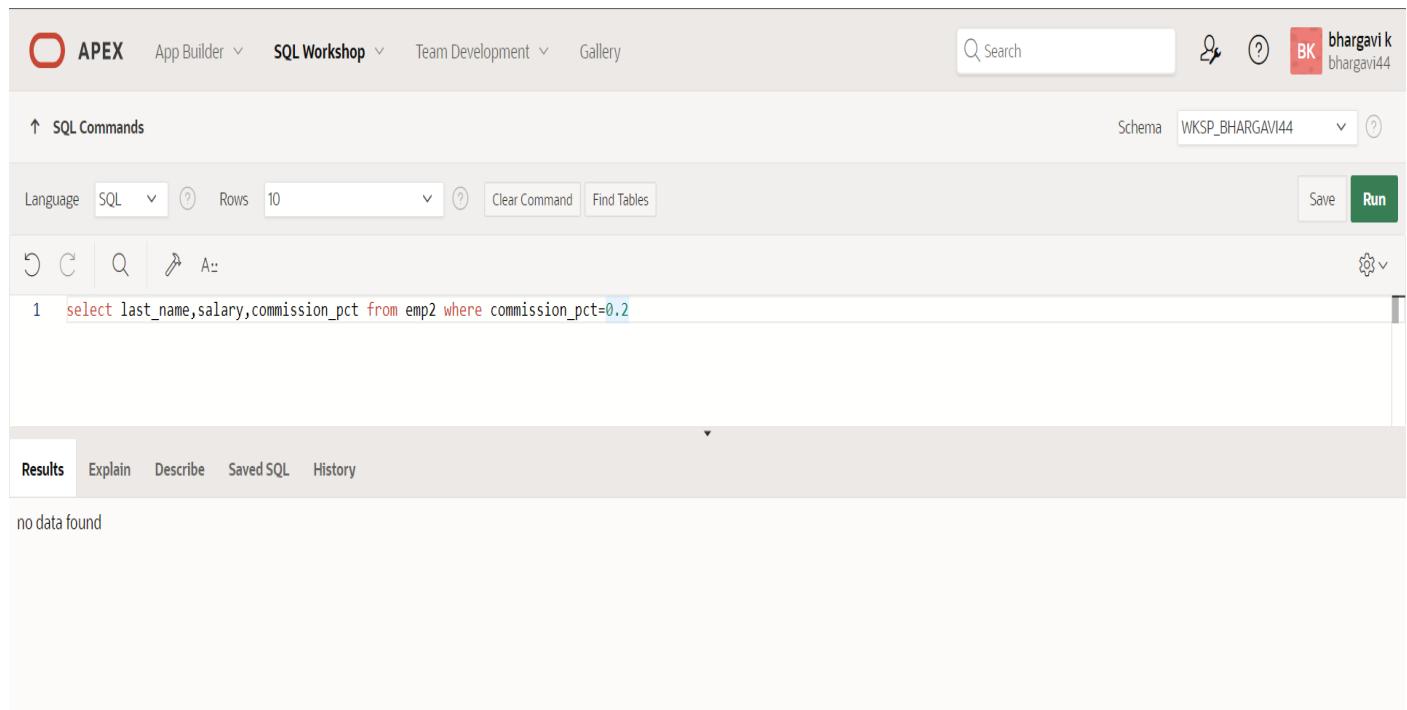
LAST_NAME	JOB_CODE	SALARY
ram	sales representative	5654

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

QUERY:

```
select last_name,salary,commission_pct from emp2 where commission_pct=0.2;
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, a user icon for 'bhargavi k bhargav44', and a 'Run' button. Below the navigation is a toolbar with icons for Undo, Redo, Copy, Paste, Find, and Save. The main area shows the SQL command: 'select last_name,salary,commission_pct from emp2 where commission_pct=0.2'. The results section below shows the message 'no data found'.

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

RESULT:

SINGLE ROW FUNCTIONS

EX_NO:6

DATE

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

QUERY:

```
select sysdate from dual;
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands pane, the query `select sysdate from dual;` is entered. The Results pane displays the output:

SYSDATE
03/13/2024

1 rows returned in 0.02 seconds

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands pane, the query `select employee_id, last_name, salary, salary+(15.5/100*salary) "new_salary" from employees;` is entered. The Results pane displays the output:

EMP_ID	LAST_NAME	SALARY	new_salary
1	Sravan	5654	6530.37
3	Harish	56467	65219.385
2	Mohan	76569	88437.195

3 rows returned in 0.01 seconds

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

QUERY

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

OUTPUT

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

```
1 select emp_id, last_name, salary, salary+(15.5/100*salary) "new_salary", salary+(15.5/100*salary)-salary as "Increase" from emp2;
```

The results section displays the following data:

EMP_ID	LAST_NAME	SALARY	new_salary	Increase
1	Sravan	5654	6530.37	876.37
3	Harish	56467	65219.385	8752.385
2	Mohan	76569	88437.195	11868.195

3 rows returned in 0.01 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(last_name), length(last_name) as "Length_of_last_name" from employees where last_name like 'J%' or last_name like 'A%' or last_name like 'M%' order by last_name asc;
```

OUTPUT

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

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

The results section displays the following data:

INITCAP(LAST_NAME)	Length_of_last_name
Mohan	5

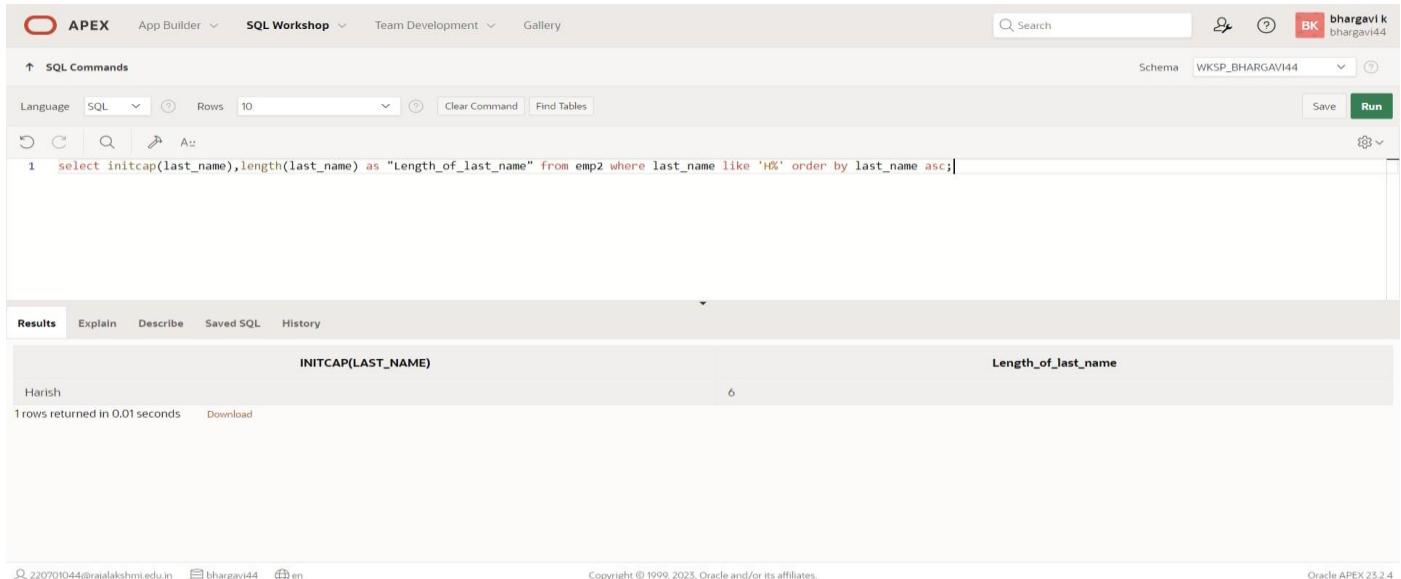
1 rows returned in 0.01 seconds

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(last_name),length(last_name) as "Length_of_last_name" from emp2 where last_name like 'H%' order by last_name asc;
```

OUTPUT:



INITCAP(LAST_NAME)	Length_of_last_name
Harish	6

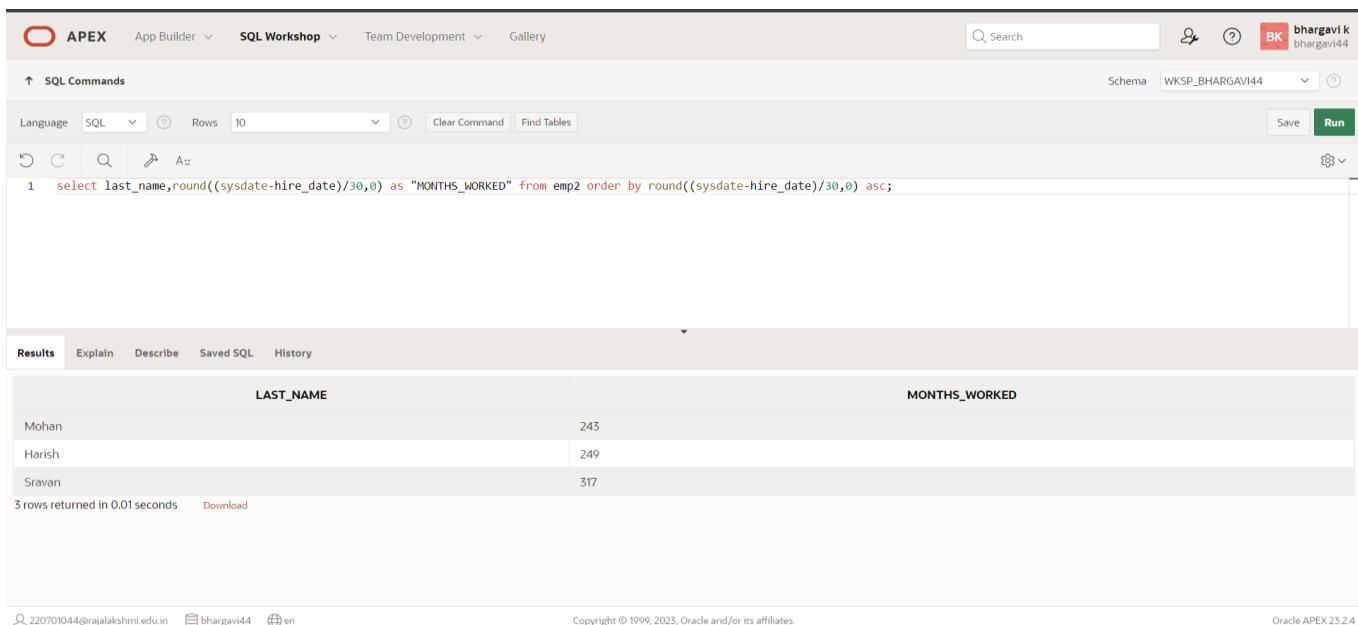
1 rows returned in 0.01 seconds [Download](#)

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

QUERY:

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

OUTPUT:



LAST_NAME	MONTHS_WORKED
Mohan	243
Harish	249
Sravan	317

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

OUTPUT:

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

```
1 select last_name||' earns'||salary||' monthly but wants'||salary*3 as "DREAM_SALARIES" from emp2;
2
3
```

The results are displayed in a table titled "DREAM_SALARIES".

	DREAM_SALARIES
1	Sravan earns 5654 monthly but wants 16962
2	Harish earns 56467 monthly but wants 169401
3	Mohan earns 76569 monthly but wants 229707

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

At the bottom, it shows the user information: 220701044@rajalakshmi.edu.in, bhargavi44, en. Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

QUERY:

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

OUTPUT:

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

```
1 select last_name, lpad(salary,15,'$') as "SALARY" from emp2;
2
```

The results are displayed in a table with columns "LAST_NAME" and "SALARY".

LAST_NAME	SALARY
Sravan	\$\$\$\$\$\$\$\$\$\$5654
Harish	\$\$\$\$\$\$\$\$\$\$56467
Mohan	\$\$\$\$\$\$\$\$\$\$76569

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

At the bottom, it shows the user information: 220701044@rajalakshmi.edu.in, bhargavi44, en. Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile for 'bhargavi k bhargavi44'. The main area has tabs for 'SQL Commands' and 'Results'. Under 'Results', there are tabs for 'Explain', 'Describe', 'Saved SQL', and 'History'. The results table has columns: LAST_NAME, HIRE_DATE, and REVIEW. The data shows three rows: Sravan (02/25/1998, Monday, the 31 of August, 1998), Harish (10/13/2003, Monday, the 19 of April, 2004), and Mohan (03/15/2004, Monday, the 20 of September, 2004). A note at the bottom says '3 rows returned in 0.00 seconds'. The bottom of the page includes copyright information and links for '220701044@rajalakshmi.edu.in', 'bhargavi44', and 'en'.

LAST_NAME	HIRE_DATE	REVIEW
Sravan	02/25/1998	Monday, the 31 of August, 1998
Harish	10/13/2003	Monday, the 19 of April, 2004
Mohan	03/15/2004	Monday, the 20 of September, 2004

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile for 'bhargavi k bhargavi44'. The main area has tabs for 'SQL Commands' and 'Results'. Under 'Results', there are tabs for 'Explain', 'Describe', 'Saved SQL', and 'History'. The results table has columns: LAST_NAME, HIRE_DATE, and DAY. The data shows three rows: Harish (10/13/2003, Monday), Mohan (03/15/2004, Monday), and Sravan (02/25/1998, Wednesday). A note at the bottom says '3 rows returned in 0.01 seconds'. The bottom of the page includes copyright information and links for '220701044@rajalakshmi.edu.in', 'bhargavi44', and 'en'.

LAST_NAME	HIRE_DATE	DAY
Harish	10/13/2003	Monday
Mohan	03/15/2004	Monday
Sravan	02/25/1998	Wednesday

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

RESULT:

DISPLAYING DATA FROM MULTIPLE TABLES

EX_NO:7

DATE:

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

QUERY:

```
Select e.last_name,e.department_number,d.dept_id from emp2 e,dept d where e.department_number=d.dept_id;
```

OUTPUT:



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

```
1 Select e.last_name,e.department_number,d.dept_id from emp2 e,dept d where e.department_number=d.dept_id;
```

The results section displays the following data:

LAST_NAME	DEPARTMENT_NUMBER	DEPT_ID
Sravan	54	54
Harish	55	55
Kappor	56	56

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

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

QUERY:

```
select distinct e.job_id ,d.loc_id from emp2 e,dept d where e.department_number=d.dept_id and e.department_number=80;
```

OUTPUT:



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

```
1 select distinct e.job_id ,d.loc_id from emp2 e,dept d where e.department_number=d.dept_id and e.department_number=80;
```

The results section displays the following data:

JOB_ID	LOC_ID
74676	453

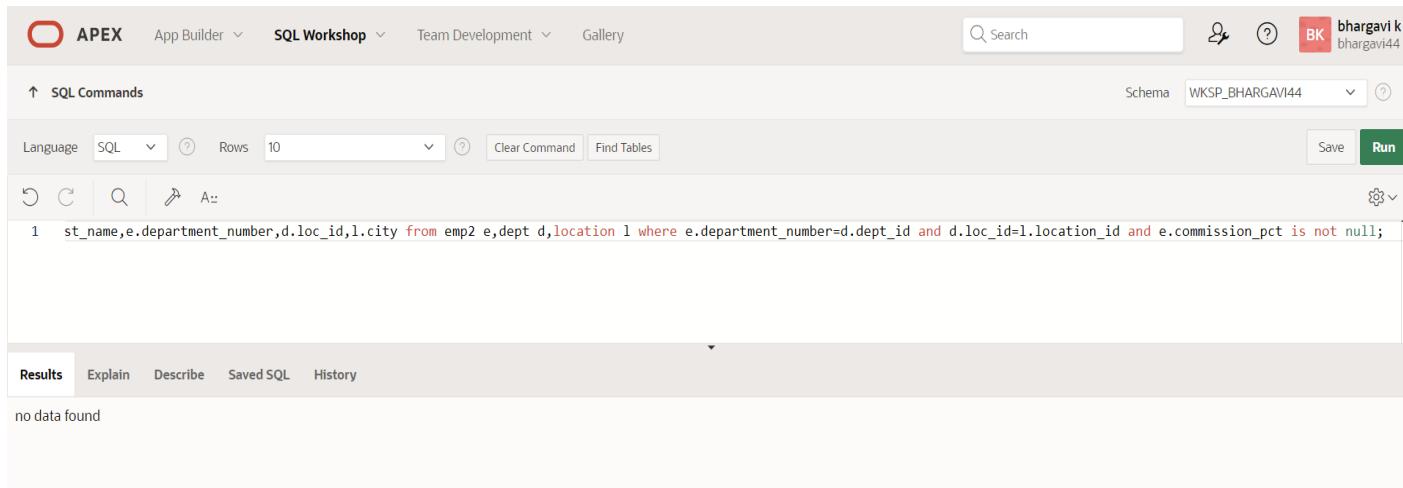
1 rows returned in 0.02 seconds [Download](#)

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

QUERY:

```
Select e.last_name,e.department_number,d.dept_name,d.loc_id,l.city from emp2 e,dept d,location l where e.department_number=d.dept_id and d.loc_id=l.location_id and e.commission_pct is not null;
```

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile 'bhargavi k bhargavi44' are on the right. The main area shows a SQL command window with the following content:

```
↑ SQL Commands
Language: SQL Rows: 10 Clear Command Find Tables
1 st_name,e.department_number,d.loc_id,l.city from emp2 e,dept d,location l where e.department_number=d.dept_id and d.loc_id=l.location_id and e.commission_pct is not null;
```

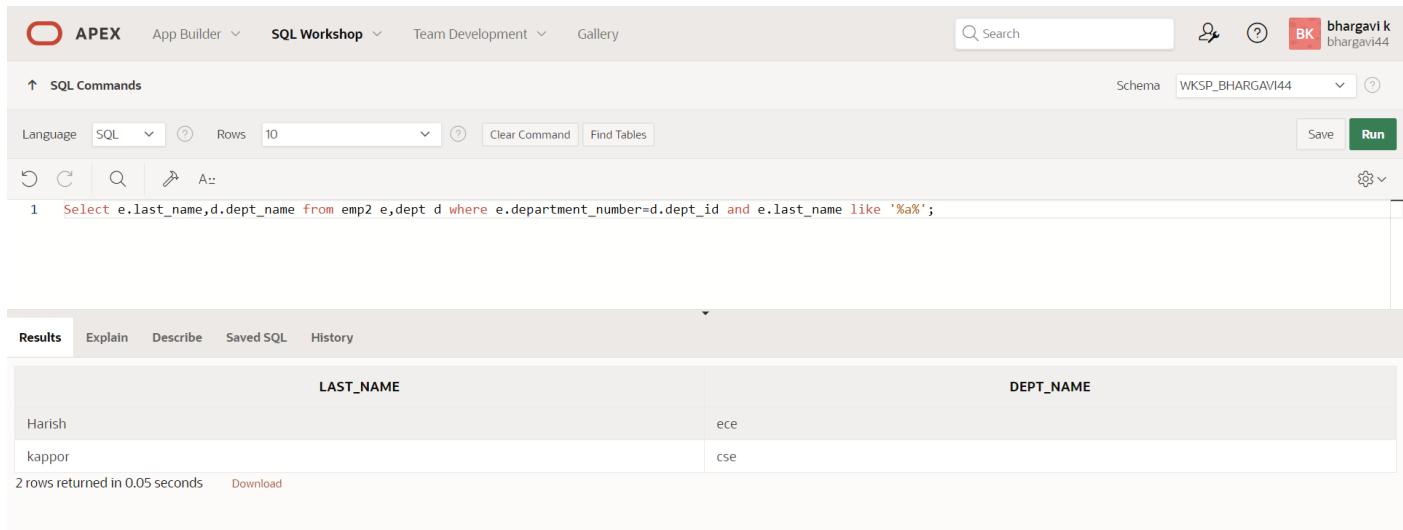
The results tab is selected, showing the message "no data found".

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

QUERY:

```
Select e.last_name,d.dept_name from emp2 e,dept d where e.department_number=d.dept_id and e.last_name like '%a%';
```

OUTPUT:



A screenshot of the Oracle SQL Workshop interface, similar to the previous one. The top navigation bar and search bar are visible. The main area shows a SQL command window with the following content:

```
↑ SQL Commands
Language: SQL Rows: 10 Clear Command Find Tables
1 Select e.last_name,d.dept_name from emp2 e,dept d where e.department_number=d.dept_id and e.last_name like '%a%';
```

The results tab is selected, displaying the following table:

LAST_NAME	DEPT_NAME
Harish	ece
Kapoor	cse

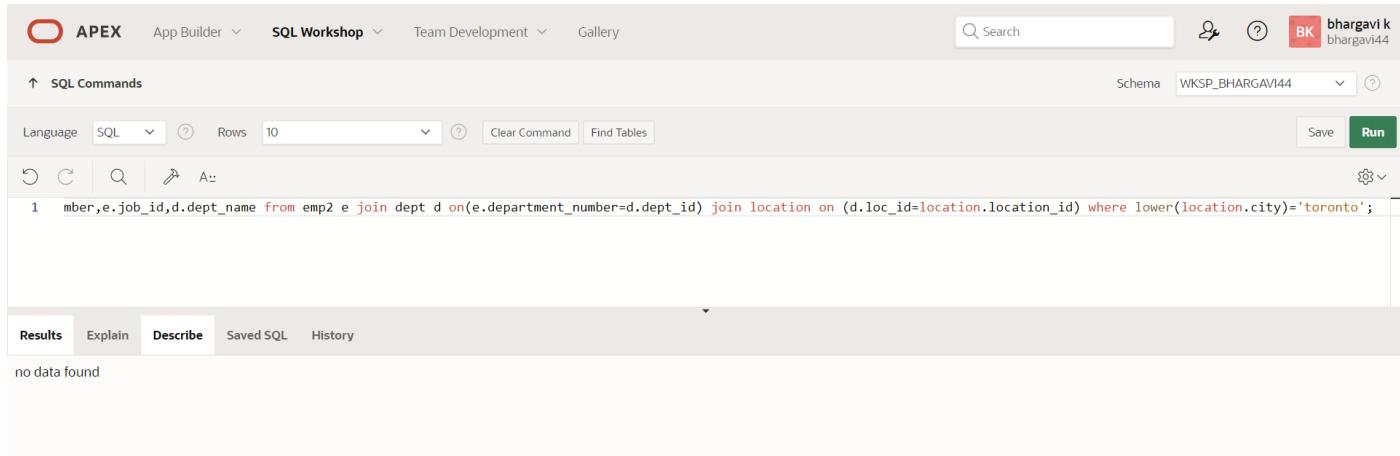
Below the table, it says "2 rows returned in 0.05 seconds" and has a "Download" link.

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.department_number,e.job_id,d.dept_name from emp2 e join dept d  
on(e.department_number=d.dept_id) join location on (d.loc_id=location.location_id) where  
lower(location.city)='toronto';
```

OUTPUT:



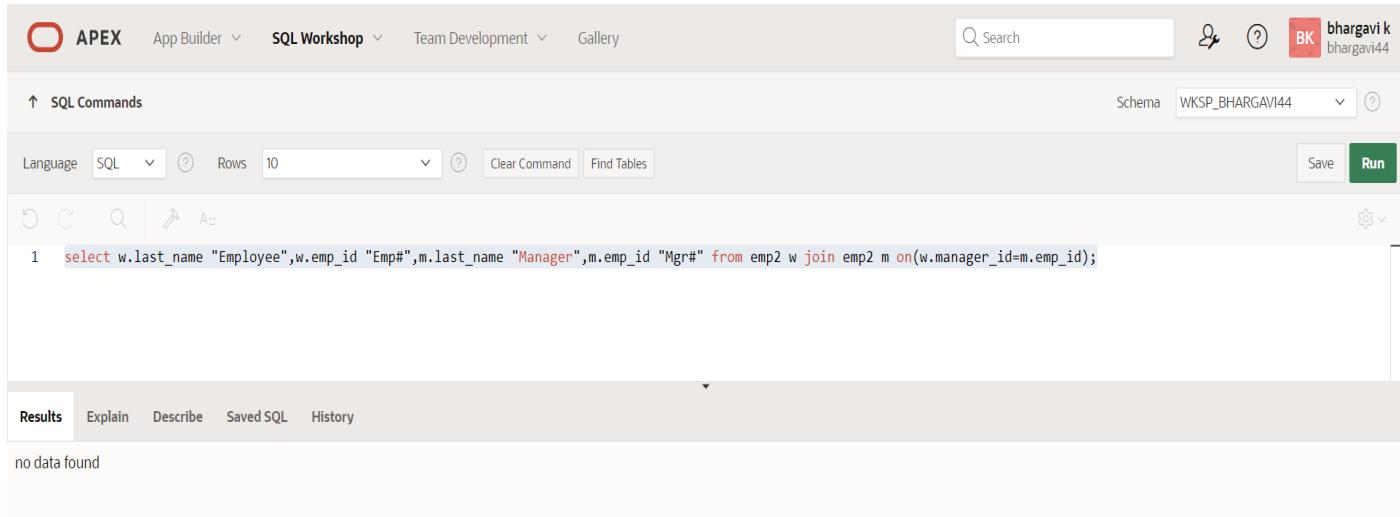
The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user information (bhargavi k bhargavi44), and a Run button. The main area is titled 'SQL Commands' with a sub-section '1'. The query is displayed in red: `1 select e.last_name,e.department_number,e.job_id,d.dept_name from emp2 e join dept d on(e.department_number=d.dept_id) join location on (d.loc_id=location.location_id) where lower(location.city)='toronto';`. Below the query, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, showing the message 'no data found'.

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

QUERY:

```
select w.last_name "Employee",w.emp_id "Emp#",m.last_name "Manager",m.emp_id "Mgr#" from emp2 w  
join emp2 m on(w.manager_id=m.emp_id);
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface, identical to the previous one but with a different query. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user information (bhargavi k bhargavi44), and a Run button. The main area is titled 'SQL Commands' with a sub-section '1'. The query is displayed in red: `1 select w.last_name "Employee",w.emp_id "Emp#",m.last_name "Manager",m.emp_id "Mgr#" from emp2 w join emp2 m on(w.manager_id=m.emp_id);`. Below the query, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, showing the message 'no data found'.

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

QUERY:

```
select w.last_name "Employee",w.emp_id "Emp#",m.last_name "Manager",m.emp_id "Mgr#" from emp2 w  
left outer join emp2 m on(w.manager_id=m.emp_id);
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'bhargavi k bhargavi44', and a 'Run' button. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP_BHARGAVI44'. The code entered is:

```
1 select w.last_name "Employee",w.emp_id "Emp#",m.last_name "Manager",m.emp_id "Mgr#" from emp2 w left outer join emp2 m on(w.manager_id=m.emp_id);
```

The results section displays a table with four columns: Employee, Emp#, Manager, and Mgr#. The data is as follows:

Employee	Emp#	Manager	Mgr#
Sravan	1	-	-
kappor	4	-	-
Harish	3	-	-
Mohan	2	-	-

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

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'bhargavi k bhargavi44', and a 'Run' button. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP_BHARGAVI44'. The code entered is:

```
1 select e.department_number departments,e.last_name colleague from employees e join employees c on (e.department_number=c.department_number) where e.emp_id <> c.emp_id order by e.department_number,e.last_name,c.last_name;
```

The results section displays a table with three columns: departments, colleague, and last_name. The data is as follows:

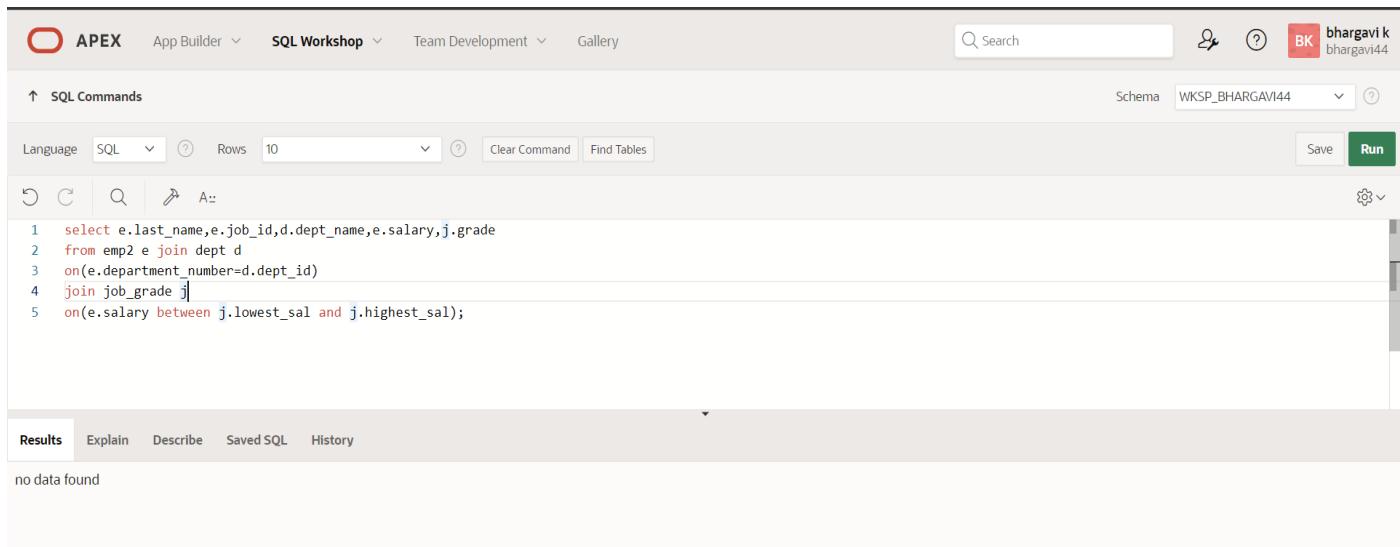
departments	colleague	last_name
no data found		

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  
from emp2 e join dept d  
on(e.department_number=d.dept_id)  
join job_grade j  
on(e.salary between j.lowest_sal and j.highest_sal);
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'bhargavi k bhargavi44', and a 'Run' button. The main area is titled 'SQL Commands' with a language dropdown set to 'SQL'. It shows the following query:

```
1 select e.last_name,e.job_id,d.dept_name,e.salary,j.grade  
2 from emp2 e join dept d  
3 on(e.department_number=d.dept_id)  
4 join job_grade j  
5 on(e.salary between j.lowest_sal and j.highest_sal);
```

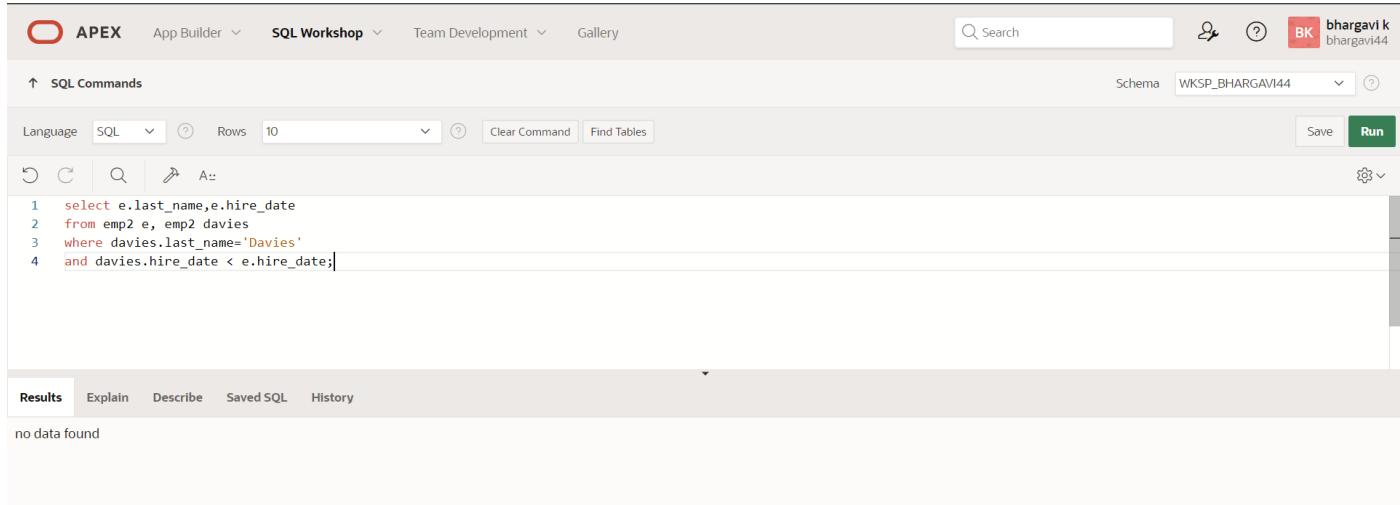
Below the code, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, showing the message 'no data found'.

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 emp2 e, emp2 davies  
where davies.last_name='Davies'  
and davies.hire_date < e.hire_date;
```

OUTPUT:



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

```
1 select e.last_name,e.hire_date  
2 from emp2 e, emp2 davies  
3 where davies.last_name='Davies'  
4 and davies.hire_date < e.hire_date;
```

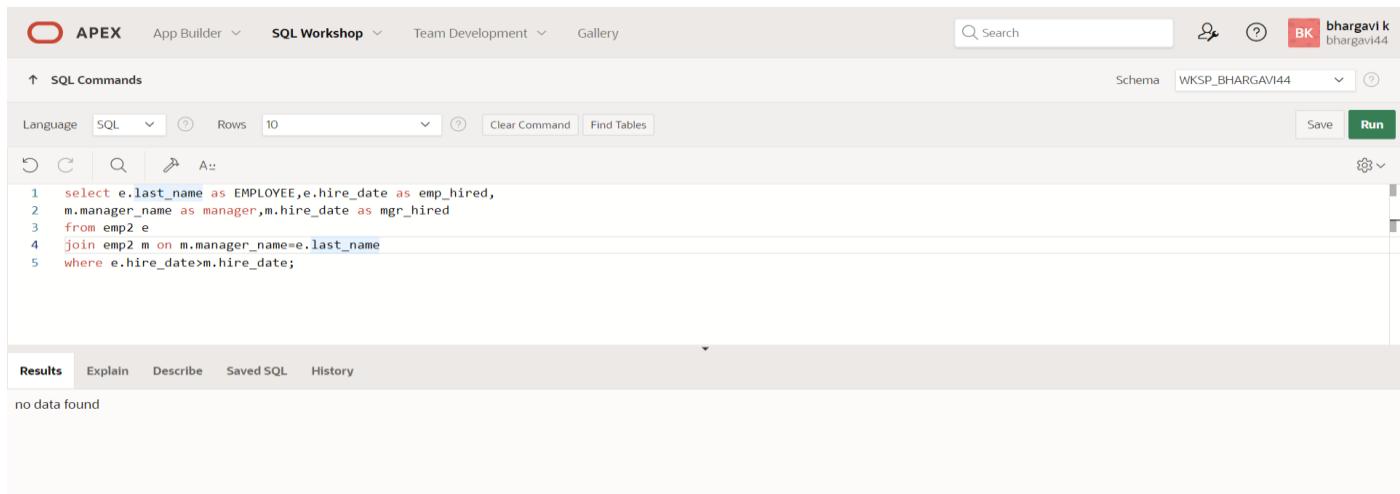
Below the code, the Results tab is selected, showing the message 'no data found'.

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

QUERY:

```
select e.last_name as EMPLOYEE,e.hire_date as emp_hired,
m.manager_name as manager,m.hire_date as mgr_hired
from emp2 e
join emp2 m on m.manager_name=e.last_name
where e.hire_date>m.hire_date;
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there is a search bar, a user icon for 'bhargavi k bhargavi44', and a 'Run' button. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the query code. The Results tab shows the output: 'no data found'.

```
1 select e.last_name as EMPLOYEE,e.hire_date as emp_hired,
2 m.manager_name as manager,m.hire_date as mgr_hired
3 from emp2 e
4 join emp2 m on m.manager_name=e.last_name
5 where e.hire_date>m.hire_date;
```

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

RESULT:

AGGREGATING DATA USING GROUP FUNCTIONS

EX_NO:8

DATE:

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

True/False

TRUE

2. Group functions include nulls in calculations.

True/False

FALSE

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

True/False

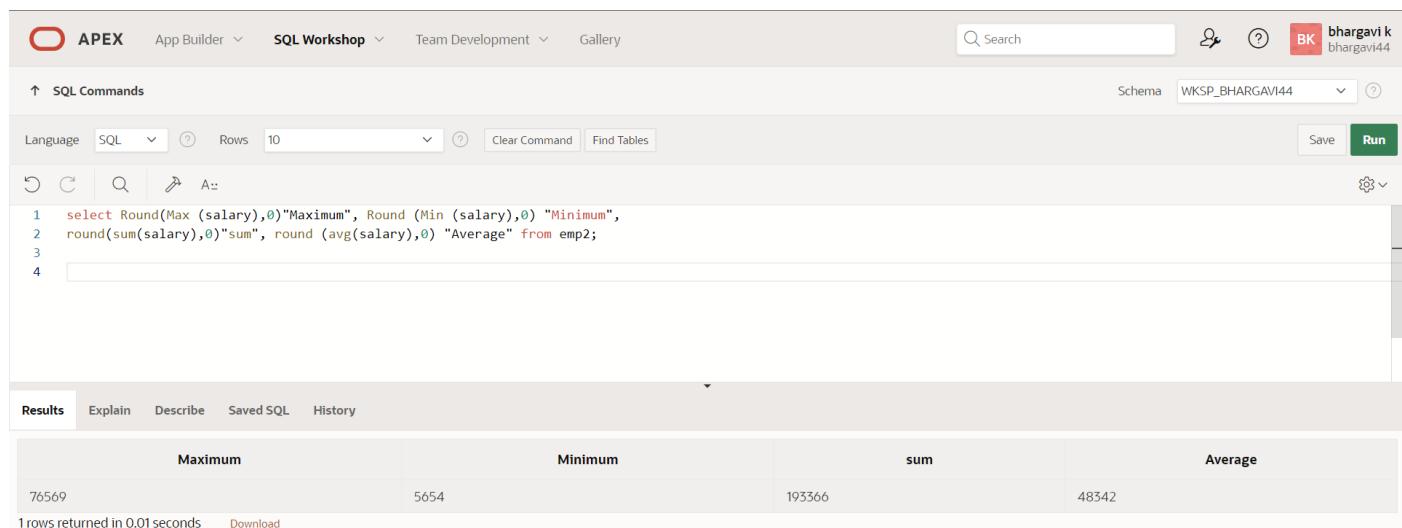
FALSE

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

QUERY:

```
select Round(Max (salary),0)"Maximum", Round (Min (salary),0) "Minimum",
round(sum(salary),0)"sum", round (avg(salary),0) "Average" from emp2;
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a user icon for 'bhargavi k' and a session identifier 'WKSP_BHARGAVI44'. The main area has tabs for SQL Commands, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, displaying the following SQL code:

```
1 select Round(Max (salary),0)"Maximum", Round (Min (salary),0) "Minimum",
2 round(sum(salary),0)"sum", round (avg(salary),0) "Average" from emp2;
3
4
```

Below the code, the Results tab is selected, showing the output:

Maximum	Minimum	sum	Average
76569	5654	193366	48342

At the bottom left, it says '1 rows returned in 0.01 seconds' and there's a 'Download' link.

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

QUERY:

```
select job_id ,Round(MAX(salary),0) "MAXIMUM",Round (Min(salary),0)"Minimum",Round (SUM(Salary),0)"sum" ,Round (AVG (salary),0)"average" from emp2 group by job_id;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. On the right, there's a search bar, a user icon for 'bhargavi k bhargavi44', and a 'Run' button. The main area is titled 'SQL Commands' with a 'Language' dropdown set to 'SQL'. Below it, the query is displayed:

```
1 select job_id ,Round(MAX(salary),0) "MAXIMUM",Round (Min(salary),0)"Minimum",Round (SUM(Salary),0)"sum" ,Round (AVG (salary),0)"average" from emp2 group by job_id;
```

The results tab is selected, showing the following table:

JOB_ID	MAXIMUM	MINIMUM	sum	average
4798	76569	76569	76569	76569
76578	54676	54676	54676	54676
466	5654	5654	5654	5654
74676	56467	56467	56467	56467

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

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

QUERY:

```
select job_id, count(*) from EMPLOYEES group by job_id ;
```

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

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. On the right, there's a search bar, a user icon for 'bhargavi k bhargavi44', and a 'Run' button. The main area is titled 'SQL Commands' with a 'Language' dropdown set to 'SQL'. Below it, the query is displayed:

```
1 select job_id, count(*) from emp2 group by job_id ;
```

The results tab is selected, showing the following table:

JOB_ID	COUNT(*)
4798	1
76578	1
466	1
74676	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 )"Number of managers" from employees;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'bhargavi k', and a schema dropdown set to 'WKSP_BHARGAVI44'. The main workspace is titled 'SQL Commands' and contains the following SQL code:

```
1 select count(distinct manager_id )"Number of managers" from emp2;
2
3
```

Below the code, the results tab is selected, showing the output:

Number of managers
2

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

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'bhargavi k', and a schema dropdown set to 'WKSP_BHARGAVI44'. The main workspace is titled 'SQL Commands' and contains the following SQL code:

```
1 select max(salary)-min(salary) difference from emp2;
```

Below the code, the results tab is selected, showing the output:

DIFFERENCE
70915

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

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 employees where manager_id is not null group by manager_id having min(salary)>6000 order by min(salary) desc;
```

OUTPUT:

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

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

The results table has two columns: MANAGER_ID and MIN(SALARY). The single row returned is:

MANAGER_ID	MIN(SALARY)
67576547	54676

1 rows returned in 0.01 seconds

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

QUERY:

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

OUTPUT:



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

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

The results table has five columns: TOTAL, 1995, 1996, 1997, and 1998. The single row returned is:

TOTAL	1995	1996	1997	1998
4	0	0	0	1

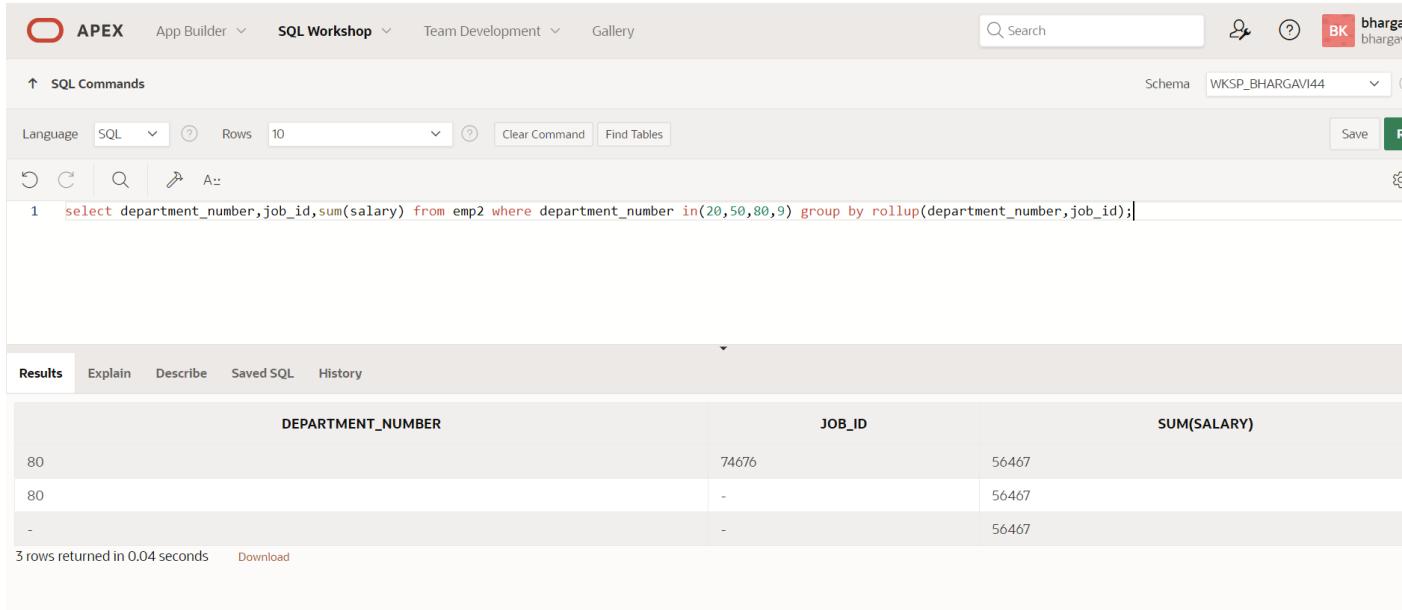
1 rows returned in 0.01 seconds

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 department_number,job_id,sum(salary) from emp2 where department_number in(20,50,80,9) group by rollup(department_number,job_id);
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'bhargav'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL tab is selected, showing the query: `select department_number,job_id,sum(salary) from emp2 where department_number in(20,50,80,9) group by rollup(department_number,job_id);`. The Results tab displays the output:

DEPARTMENT_NUMBER	JOB_ID	SUM(SALARY)
80	74676	56467
80	-	56467
-	-	56467

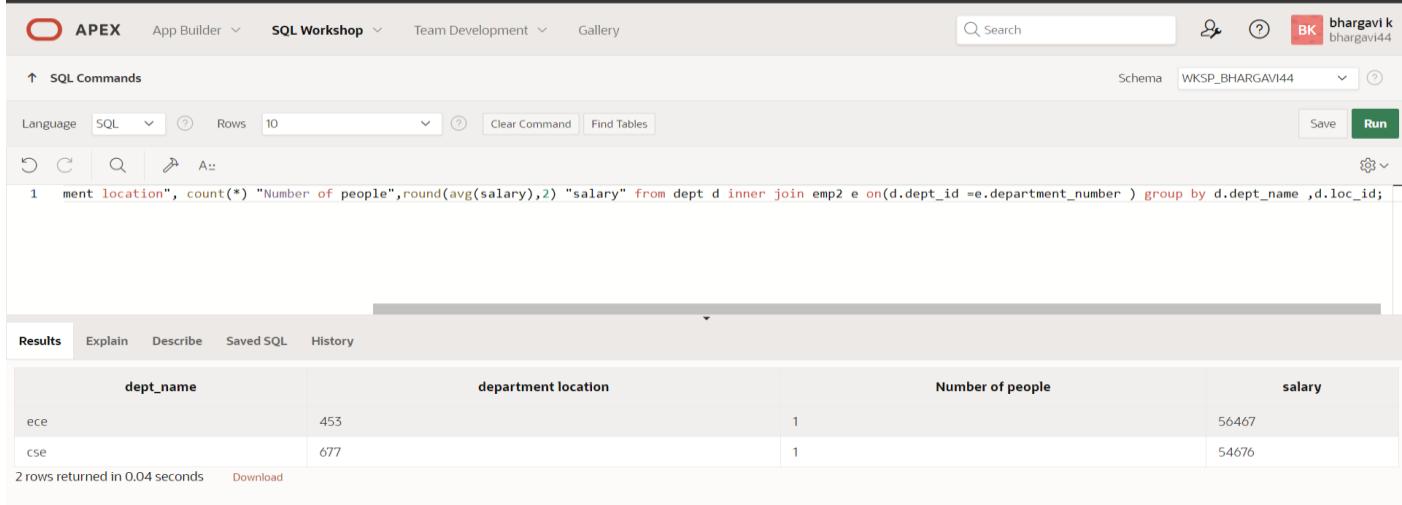
3 rows returned in 0.04 seconds. There is a 'Download' link.

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'bhargav'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL tab is selected, showing the query: `select d.dept_name as "dept_name",d.loc as "department location", count(*) "Number of people",round(avg(salary),2)
"salary" from departments d inner join employees e on(d.dpt_id =e.department_id) group by d.dept_name ,d.loc;`. The Results tab displays the output:

dept_name	department location	Number of people	salary
ece	453	1	56467
cse	677	1	56467

2 rows returned in 0.04 seconds. There is a 'Download' link.

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

RESULT:

SUB QUERIES

EX_NO:9

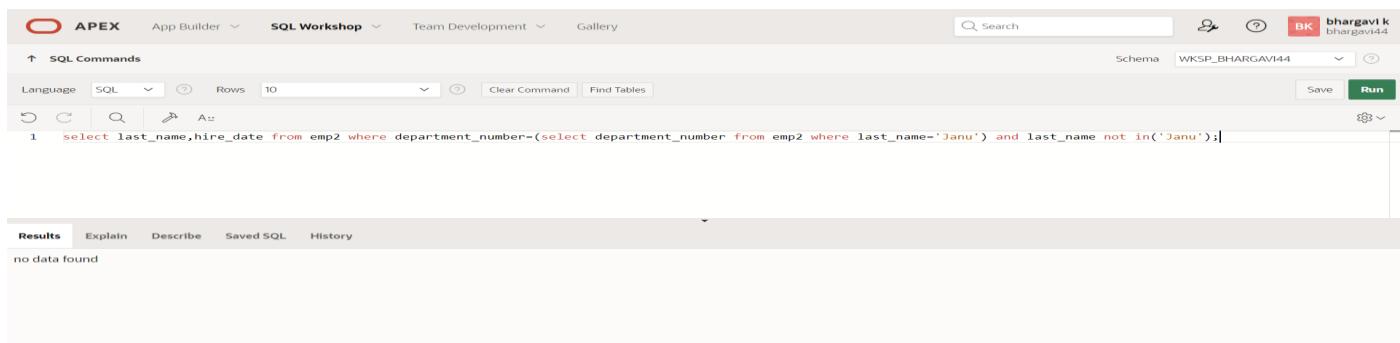
DATE:

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

QUERY:

```
select last_name,hire_date from employees where department_id=(select department_id from employees where last_name='Janu') and last_name not in('Janu');
```

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a user icon for 'bhargavik' and a schema dropdown set to 'WKSP_BHARGAVI44'. The main area shows a SQL command line with the following code:

```
1 select last_name,hire_date from emp2 where department_number=(select department_number from emp2 where last_name='Janu') and last_name not in('Janu');
```

The results section below shows the message 'no data found'.

2.)

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

QUERY

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

OUTPUT:



A screenshot of the Oracle SQL Workshop interface, similar to the previous one. The top navigation bar and schema are identical. The SQL command line contains the following query:

```
1 select emp_id,last_name,salary from emp2 where salary>(select avg(salary) from emp2) order by salary;
```

The results section displays a table with three rows of data:

EMP_ID	LAST_NAME	SALARY
4	kappor	54676
3	Harish	56467
2	Mohan	76569

At the bottom, it says '3 rows returned in 0.01 seconds'.

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

QUERY:

```
select employee_id,last_name from employees where department_id=(select department_id from employees  
where last_name like'%u%');
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, a user icon for 'bhargavi k', and a schema dropdown set to 'WKSP_BHARGAVI44'. The main area is titled 'SQL Commands' with a 'Run' button. The SQL editor contains the following query:

```
1 select emp_id,last_name from emp2 where department_number=(select department_number from emp2 where last_name like'%u%');
```

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

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

QUERY:

```
select last_name,department_id,job_id from employees where department_id=(select dept_id from  
departments where location_id=1700);
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, a user icon for 'bhargavi k', and a schema dropdown set to 'WKSP_BHARGAVI44'. The main area is titled 'SQL Commands' with a 'Run' button. The SQL editor contains the following query:

```
1 select last_name,department_number,job_id from emp2 where department_number=(select dept_id from dept where loc_id=1700);
```

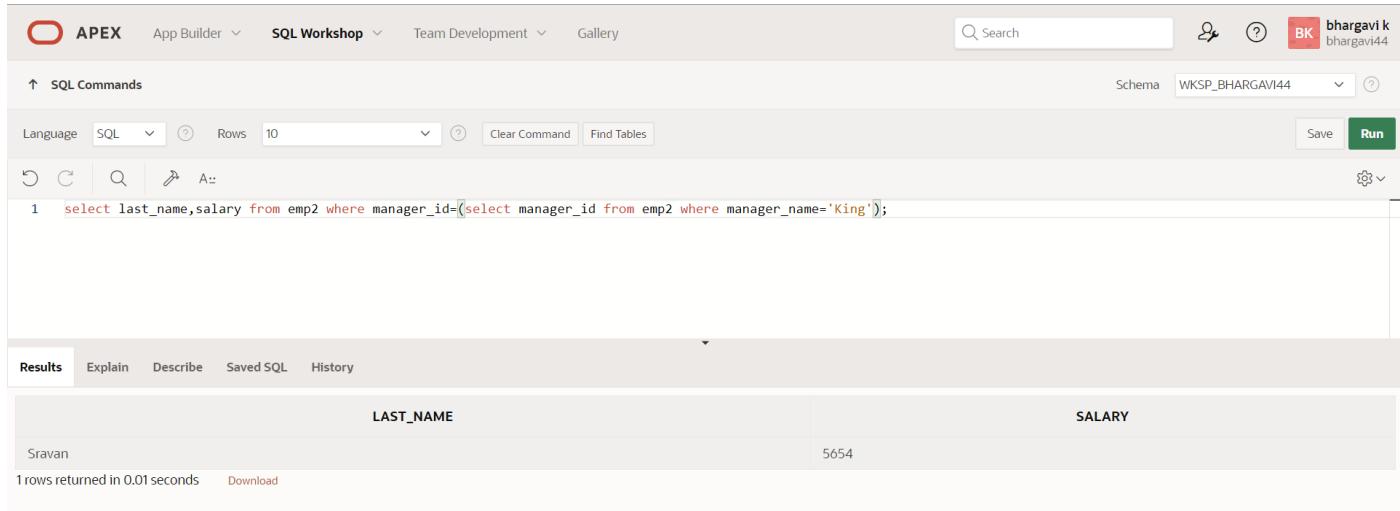
The results tab is selected, showing the message 'no data found'.

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

QUERY:

```
select last_name,salary from employees where manager_id=(select manager_id from employees where manager_name='King');
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'bhargavi k' (bhargavi44). The main area is titled 'SQL Commands' with a search bar and a 'Run' button. The schema is set to 'WKSP_BHARGAVI44'. The code entered is:

```
1 select last_name,salary from emp2 where manager_id=(select manager_id from emp2 where manager_name='King'));
```

The results section shows the output:

LAST_NAME	SALARY
Sravan	5654

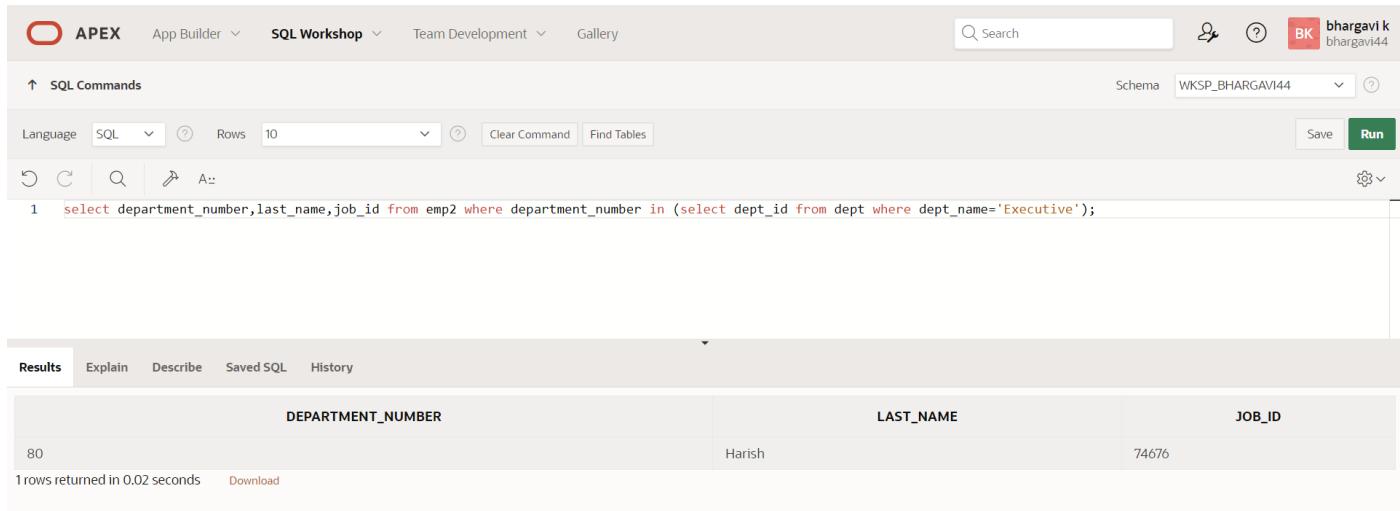
1 rows returned in 0.01 seconds [Download](#)

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'bhargavi k' (bhargavi44). The main area is titled 'SQL Commands' with a search bar and a 'Run' button. The schema is set to 'WKSP_BHARGAVI44'. The code entered is:

```
1 select department_number,last_name,job_id from emp2 where department_number in (select dept_id from dept where dept_name='Executive');
```

The results section shows the output:

DEPARTMENT_NUMBER	LAST_NAME	JOB_ID
80	Harish	74676

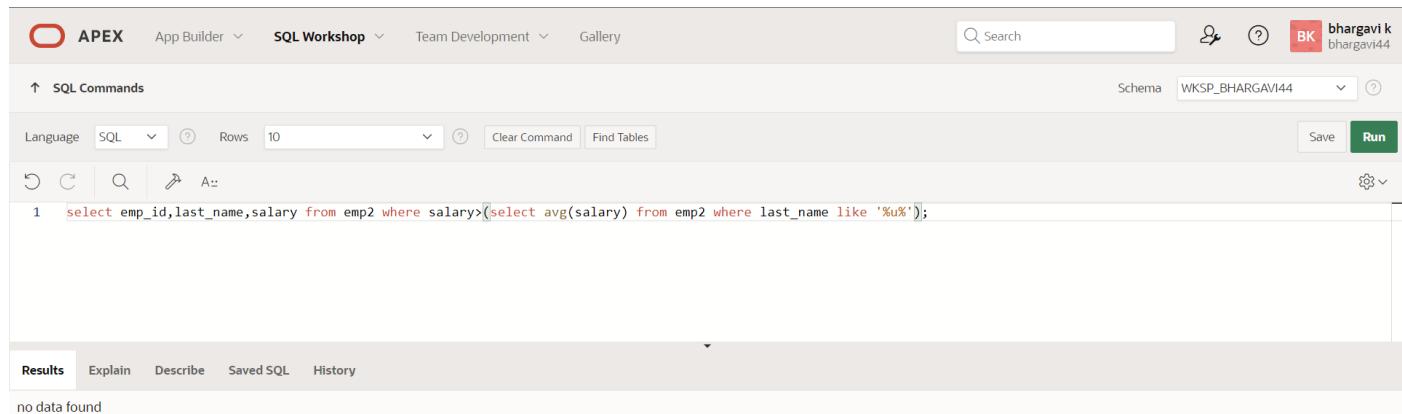
1 rows returned in 0.02 seconds [Download](#)

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. At the top, there are tabs for APEX, App Builder, SQL Workshop (which is selected), and Team Development. On the right, there's a user profile for 'bhargavi k' (bhargavi44). Below the tabs, there's a search bar and a 'Run' button. The main area is titled 'SQL Commands' and contains a code editor with the following query:

```
1 select emp_id,last_name,salary from emp2 where salary>(select avg(salary) from emp2 where last_name like '%u%');
```

Below the code editor, there are buttons for Refresh, Undo, Redo, Find, and Paste. The results tab is selected at the bottom, showing the message 'no data found'.

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

RESULT:

USING THE SET OPERATORS

EX_NO:10

DATE:

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

QUERY:

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

OUTPUT:

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

```
1 select department_number from emp2 minus select department_number from emp2 where job_id='st_clerk';
```

In the Results pane, the output is displayed as:

DEPARTMENT_NUMBER
56
565675

2 rows returned in 0.01 seconds

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

QUERY:

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

OUTPUT:

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

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

In the Results pane, the output is displayed as:

COUNTRY_ID	STATE_PROVINCE
34	andra
43	tn
45	tn

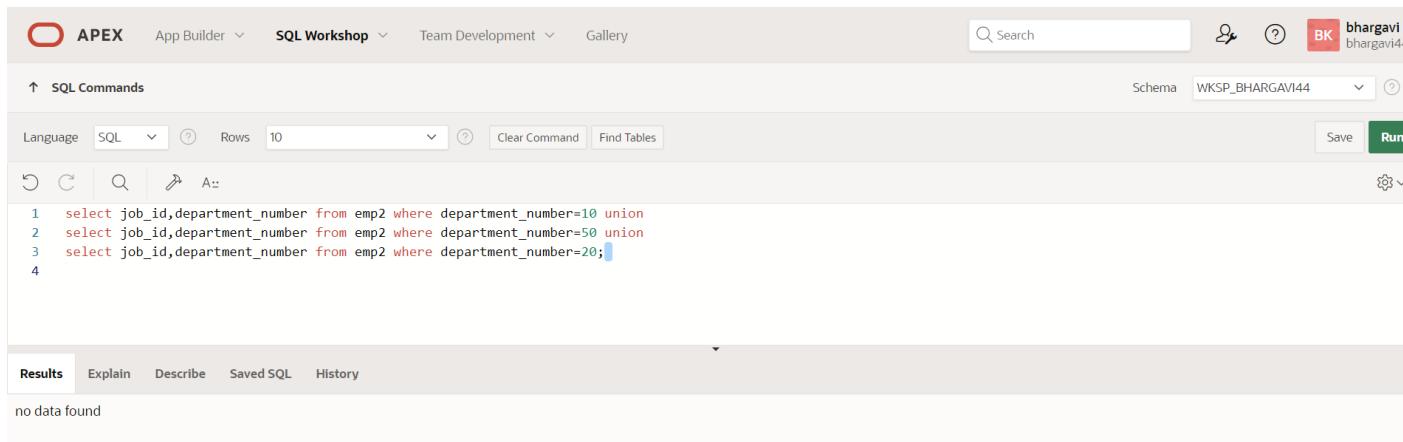
3 rows returned in 0.03 seconds

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, a user icon, and the session information 'bhargavi bhargavi44'. The main area is titled 'SQL Commands' with tabs for 'Language' (set to 'SQL'), 'Rows' (set to 10), and buttons for 'Clear Command' and 'Find Tables'. A 'Run' button is on the far right. Below this is a toolbar with icons for refresh, undo, redo, search, and a dropdown menu. The code editor contains the following SQL query:

```
1 select job_id,department_number from emp2 where department_number=10 union
2 select job_id,department_number from emp2 where department_number=50 union
3 select job_id,department_number from emp2 where department_number=20;
4
```

Below the code editor is a results pane with tabs for 'Results' (selected), 'Explain', 'Describe', 'Saved SQL', and 'History'. The results section displays the message 'no data found'.

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface, identical to the previous one but with a different query. The top navigation bar, code editor, and results pane are all the same. The code editor contains the following SQL query:

```
1 select job_id,emp_id from emp2 intersect select e.job_id,e.emp_id from emp2 e,job_history j where e.job_id=j.old_job_id;
2
3
```

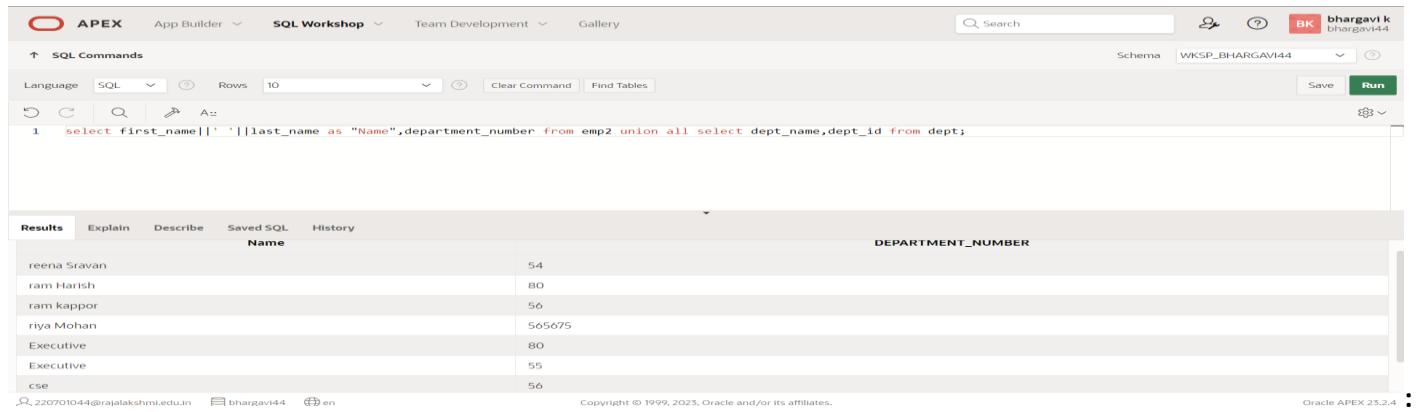
The results pane again displays the message 'no data found'.

5.)The HR department needs a report with the following specifications: - Last name and department ID of all the employees from the EMPLOYEES table, regardless of whether or not they belong to a department. - Department ID and department name of all the departments from the DEPARTMENTS table, regardless of whether or not they have employees working in them Write a compound query to accomplish this.

QUERY:

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

OUTPUT



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

```
1 select first_name||' '|last_name as "Name",department_number from emp2 union all select dept_name,dept_id from dept;
```

The results are displayed in a table:

Name	DEPARTMENT_NUMBER
reena Sravan	54
ram Harish	80
ram Kapoor	56
riya Mohan	565675
Executive	80
Executive	55
cse	56

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

RESULT:

CREATING VIEWS

EX_NO:11

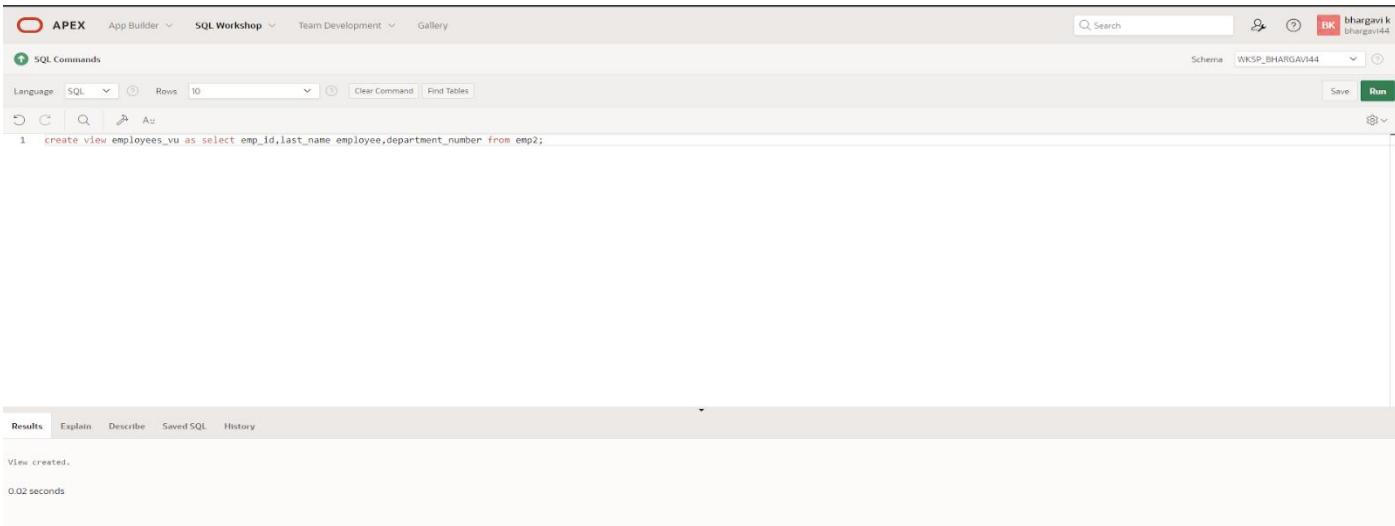
DATE:

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

QUERY:

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

OUTPUT:



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

```
1 create view employees_vu as select emp_id,last_name employee,department_number from emp2;
```

In the Results tab, the output shows:

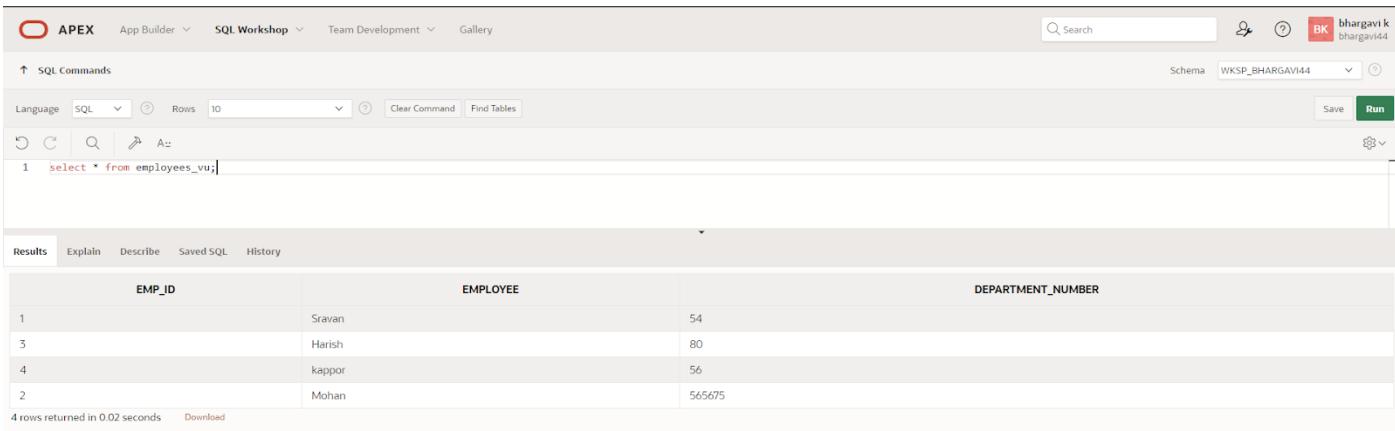
```
View created.  
0.02 seconds
```

2.) Display the contents of the EMPLOYEES_VU view.

QUERY:

```
select * from employees_vu;
```

OUTPUT:



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

```
1 select * from employees_vu;
```

In the Results tab, the output shows the following table:

EMP_ID	EMPLOYEE	DEPARTMENT_NUMBER
1	Sravan	54
3	Harish	80
4	kappor	56
2	Mohan	565675

4 rows returned in 0.02 seconds

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows a user profile for 'bhargavi k' (bhargavi44). The main area has tabs for SQL Commands, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, showing the query: 'select view_name, text from user_views;'. The Results tab is selected, displaying the output:

VIEW_NAME	TEXT
EMPLOYEES_VU	select emp_id, last_name, employee.department_number from emp2

1 rows returned in 0.03 seconds [Download](#)

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface, identical to the previous one but with a different query. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows a user profile for 'bhargavi k' (bhargavi44). The main area has tabs for SQL Commands, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, showing the query: 'select employee, department_number from employees_vu;'. The Results tab is selected, displaying the output:

EMPLOYEE	DEPARTMENT_NUMBER
Sravan	54
Harish	80
kappor	56
Mohan	565675

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

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

QUERY:

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

OUTPUT:

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

```
1 create view dept_50 as select emp_id emp_no, last_name employee, department_number from emp2 where department_number=50 with check option constraint emp_dept_50;
```

Below the command, the results show:

View created.
0.02 seconds

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

QUERY:

```
Describe dept50;
```

OUTPUT:

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

```
1 describe dept_50;
```

Below the command, the results show the structure of the view:

Object Type: VIEW
Object: DEPT_50

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPT_50	EMP_NO	NUMBER	-	8	0	-	✓	-	-
	EMPLOYEE	VARCHAR2	12	-	-	-	✓	-	-
	DEPARTMENT_NUMBER	NUMBER	22	-	-	-	✓	-	-

7.) Attempt to reassign Matos to department 80

QUERY:

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

OUTPUT:

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

```
1 update dept_50 set department_number=80 where employee='matos';
```

Below the command, the results show:

0 row(s) updated.
0.01 seconds

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

QUERY:

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

OUTPUT:

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

```
1 create or replace view salary_vu as select e.last_name "employee",d.dept_name "department",e.salary "salary",j.grade "grades" from emp2 e, dept d,job_grade j
2 where e.department_number=d.dept_id and e.salary between j.lowest_sal and j.highest_sal;
```

Below the command, the results show:

View created.
0.02 seconds

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

RESULT

PRIMARY KEY, FOREIGN KEY, and CHECK Constraints

EX 12

1. What is the purpose of a

- PRIMARY KEY
- FOREIGN KEY
- CHECK CONSTRAINT

a. PRIMARY KEY

Uniquely identify each row in table.

b. FOREIGN KEY

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

c. CHECK CONSTRAINT

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

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

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

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

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

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

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

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

SELECT * FROM animals;

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

COLUMN LEVEL STATEMENT:

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

TABLE LEVEL STATEMENT:

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

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

a.ON DELETE CASCADE

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

b.ON DELETE SET NULL

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

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

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

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

CREATING VIEWS

EX:13

1What 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.**

1. 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';
```

2. SELECT * FROM view_d_songs. What was returned?

The screenshot shows the Oracle SQL Workshop interface. At the top, there are tabs for APEX, App Builder, SQL Workshop (which is selected), Team Development, and Gallery. On the right, there are user profile icons for 'bhargavi k' and 'bhargavi44'. Below the tabs, there are buttons for Search, Help, and Run. The main area has sections for SQL Commands, Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run. The SQL command entered is:

```
1 SELECT * FROM view_d_songs;
2 
```

Below the command window, there is a preview pane showing the results of the query. The results are displayed in a table with columns: ID, Song Title, and ARTIST. One row is visible:

ID	Song Title	ARTIST
4	Song D	Artist D

At the bottom of the preview pane, there are tabs for Results (selected), Explain, Describe, Saved SQL, and History. The results table below shows two rows:

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

At the very bottom left, it says '2 rows returned in 0.00 seconds' and there is a 'Download' link.

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

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

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'bhargavi' with a session ID 'bhargavi4'. The main area is titled 'SQL Commands' with a 'Language' dropdown set to 'SQL'. Below it is a code editor containing the SQL query for creating the view. The code is numbered from 1 to 9. The results tab is selected at the bottom, showing the message 'View created'.

```
1 CREATE OR REPLACE VIEW view_d_songs AS
2 SELECT d_songs.id AS "Song ID",
3        d_songs.title AS "Song Title",
4        d_songs.artist AS "Artist",
5        d_songs.type_code AS "Type Code"
6   FROM d_songs
7  INNER JOIN d_types ON d_songs.type_code = d_types.code
8 WHERE d_types.description = 'New Age';
9
10 |
```

Results Explain Describe Saved SQL History

View created

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

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

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'bhargavi' with a session ID 'bhargavi4'. The main area is titled 'SQL Commands' with a 'Language' dropdown set to 'SQL'. Below it is a code editor containing the SQL query for creating the view. The code is numbered from 1 to 6. The results tab is selected at the bottom, showing the message 'View created'.

```
1 CREATE OR REPLACE VIEW view_d_events_pkgs AS
2 SELECT evt.name "Name of Event", TO_CHAR(evt.event_date, 'dd-Month-yyyy') "Event date", thm.description "Theme description"
3 FROM d_events evt INNER JOIN d_themes thm ON evt.theme_code = thm.code
4 WHERE evt.event_date <= ADD_MONTHS(SYSDATE,1);
5
6 |
```

Results Explain Describe Saved SQL History

View created

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

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

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. The code entered is the CREATE OR REPLACE VIEW statement provided above. The 'Run' button is highlighted in green at the bottom right of the code editor. Below the editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. A message 'View created.' is displayed in the results area.

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';
```

The screenshot shows the Oracle SQL Workshop interface. The code entered is the SELECT statement provided above. The 'Run' button is highlighted in green at the bottom right of the code editor. Below the editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The results table shows the following data:

OWNER	TABLE_NAME	COLUMN_NAME	UPDATABLE	INSERTABLE	DELETABLE
WKSP_BHARGAVI44	COPY_D_SONGS	ID	YES	YES	YES
WKSP_BHARGAVI44	COPY_D_SONGS	TITLE	YES	YES	YES

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

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. On the right, there's a search bar, a user icon for 'bhargavi k', and a schema dropdown set to 'WKSP_BHARGAVI44'. The main area is titled 'SQL Commands' with a 'Run' button. The code entered is:

```
1 SELECT owner, table_name, column_name, updatable,insertable, deletable  
2 FROM user_updatable_columns WHERE LOWER(table_name) = 'copy_d_events';  
3  
4  
5
```

The results section shows a table with the following data:

OWNER	TABLE_NAME	COLUMN_NAME	UPDATABLE	INSERTABLE	DELETABLE
WKSP_BHARGAVI44	COPY_D_EVENTS	EVENT_ID	YES	YES	YES
WKSP_BHARGAVI44	COPY_D_EVENTS	NAME	YES	YES	YES

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

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. On the right, there's a search bar, a user icon for 'bhargavi k', and a schema dropdown set to 'WKSP_BHARGAVI44'. The main area is titled 'SQL Commands' with a 'Run' button. The code entered is:

```
1 SELECT owner, table_name, column_name, updatable,insertable, deletable  
2 FROM user_updatable_columns WHERE LOWER(table_name) = 'copy_d_cds';  
3  
4  
5  
6
```

The results section shows a table with the following data:

OWNER	TABLE_NAME	COLUMN_NAME	UPDATABLE	INSERTABLE	DELETABLE
WKSP_BHARGAVI44	COPY_D_CDS	CD_NUMBER	YES	YES	YES
WKSP_BHARGAVI44	COPY_D_CDS	TITLE	YES	YES	YES
WKSP_BHARGAVI44	COPY_D_CDS	ARTIST	YES	YES	YES

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;
```

This screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, a user icon for 'bhargavi k', and a schema dropdown set to 'WKSP_BHARGAVI44'. The main area is titled 'SQL Commands' with a 'Language' dropdown set to 'SQL'. It displays the following code:

```
1 CREATE OR REPLACE VIEW view_copy_d_songs AS  
2 SELECT *  
3 FROM copy_d_songs;  
4  
5  
6
```

The 'Results' tab is selected at the bottom, showing the message 'View created.' and a execution time of '0.02 seconds'.

This screenshot shows the Oracle SQL Workshop interface again. The top navigation bar and schema selection are identical. The main area now contains the following code:

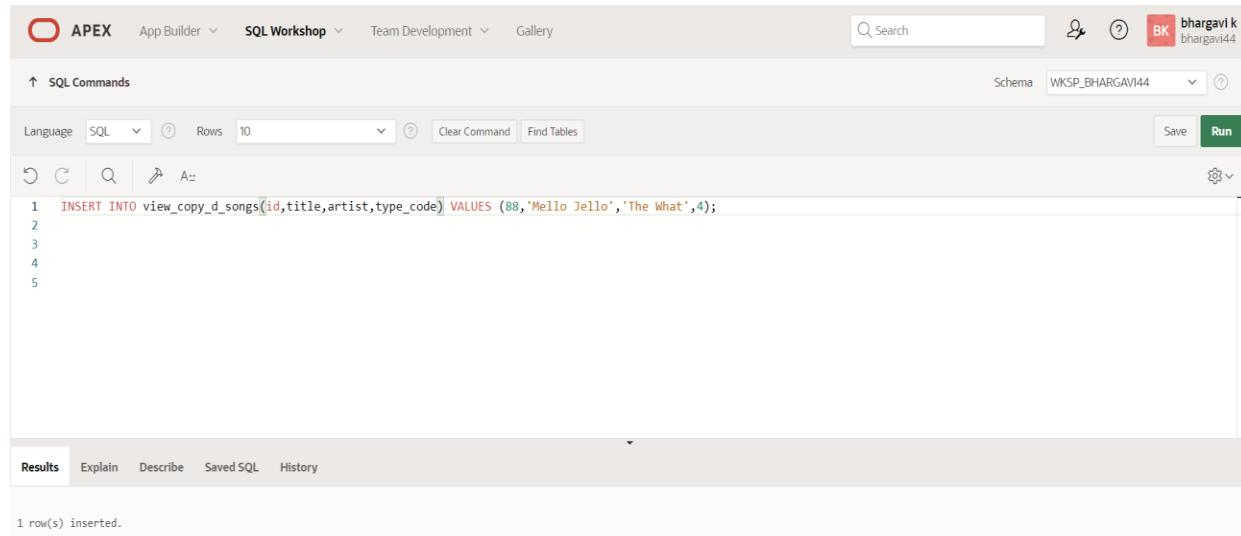
```
1 SELECT * FROM view_copy_d_songs;
```

The 'Results' tab is selected at the bottom, showing the message 'no data found'.

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

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);



A screenshot of the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile 'bhargavi k bhargavi44' are on the right. The main area shows a code editor with the following SQL command:

```

1 INSERT INTO view_copy_d_songs(id,title,artist,type_code) VALUES (88,'Mello Jello','The What',4);
2
3
4
5

```

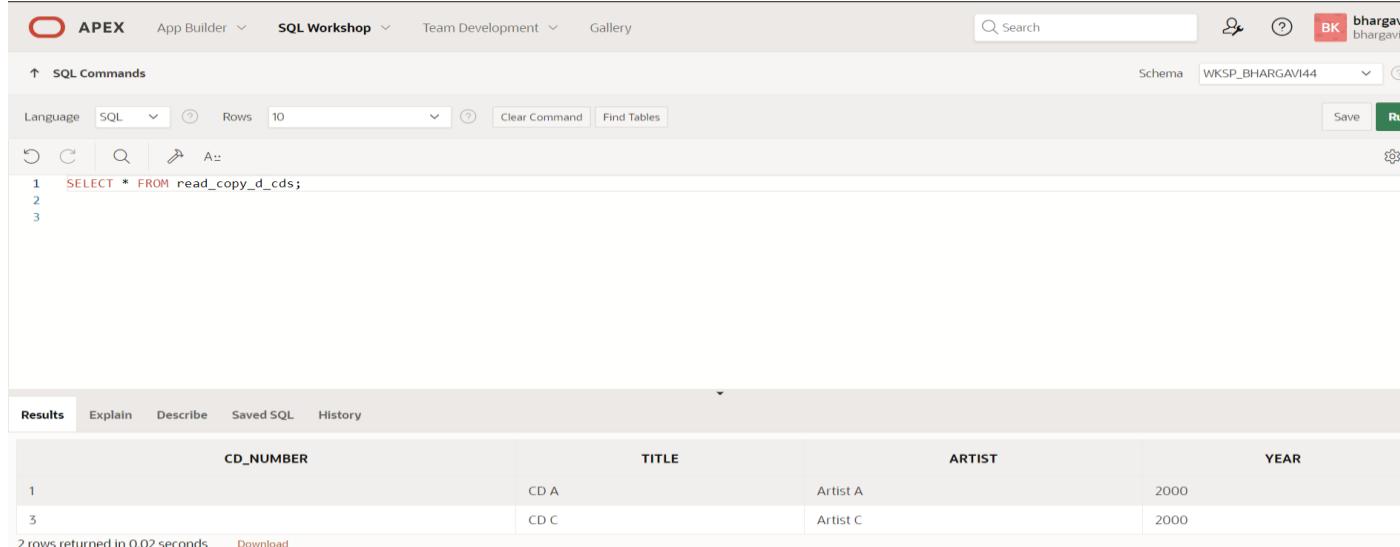
Below the code editor, the 'Results' tab is selected. It displays the message '1 row(s) inserted.' indicating the success of the insert operation.

2. 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;

```



A screenshot of the Oracle SQL Workshop interface, similar to the previous one. The top navigation bar and user profile are identical. The code editor contains the following SQL statements:

```

1 SELECT * FROM read_copy_d_cds;
2
3

```

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

CD_NUMBER	TITLE	ARTIST	YEAR
1	CD A	Artist A	2000
3	CD C	Artist C	2000

At the bottom left, it says '2 rows returned in 0.02 seconds'.

3. 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

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

CREATE OR REPLACE VIEW read_copy_d_cds AS

```
SELECT *
FROM copy_d_cds
WHERE year = '2000'
WITH CHECK OPTION CONSTRAINT ck_read_copy_d_cds;
```

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. A search bar and user information (bhargavi) are also present. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP_BHARGAVI44'. Below the title are buttons for Language (SQL), Rows (10), Clear Command, and Find Tables. The code editor contains the following SQL:

```
1 CREATE OR REPLACE VIEW read_copy_d_cds AS
2 SELECT *
3 FROM copy_d_cds
4 WHERE year = '2000'
5 WITH CHECK OPTION CONSTRAINT ck_read_copy_d_cds;
6
7
8
```

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

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

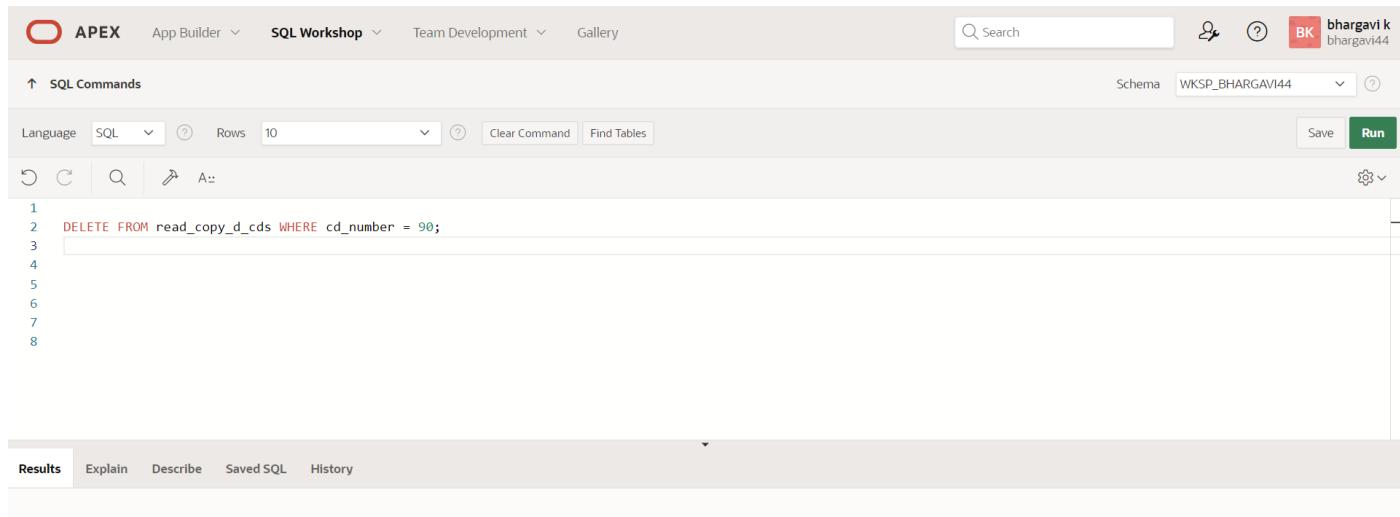
The screenshot shows the Oracle SQL Workshop interface, identical to the previous one but with a different command. The top navigation bar, schema dropdown, and code editor are the same. The code editor now contains a delete statement:

```
1 DELETE FROM read_copy_d_cds WHERE year = '2000';
2
3
4
5
6
```

The results panel at the bottom shows the output: "2 row(s) deleted."

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

DELETE FROM read_copy_d_cds
WHERE cd_number = 90;



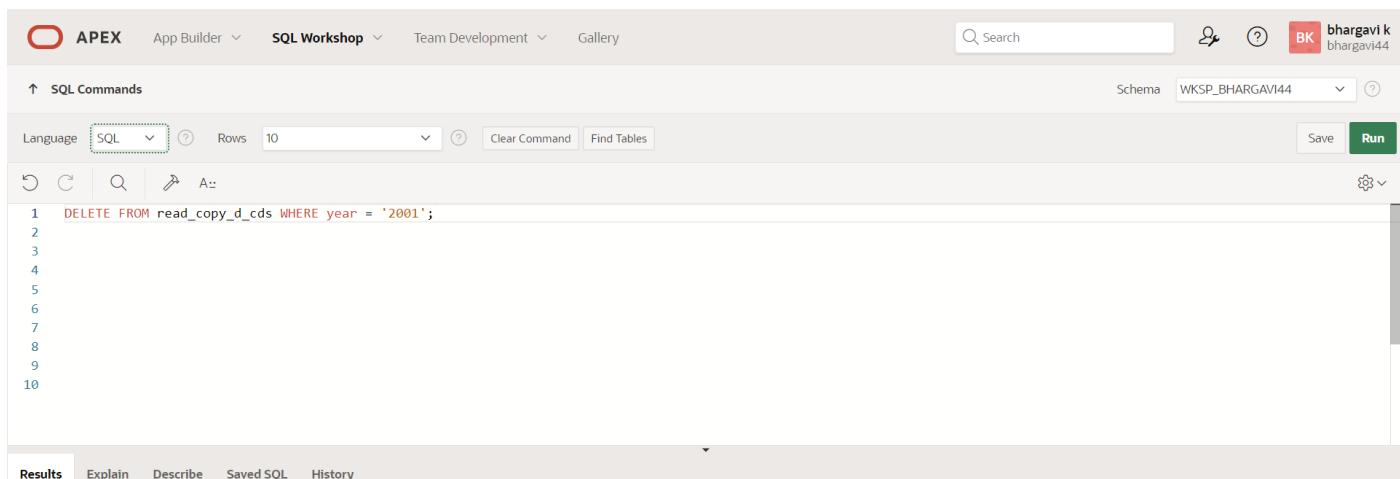
The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'bhargavi k' with the schema 'WKSP_BHARGAVI44'. The main area is titled 'SQL Commands' with a search bar and a 'Run' button. The code editor contains the following SQL command:

```
1 DELETE FROM read_copy_d_cds WHERE cd_number = 90;
2
3
4
5
6
7
8
```

The results section below shows the output: '0 row(s) deleted.'

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

DELETE FROM read_copy_d_cds
WHERE year = '2001';



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'bhargavi k' with the schema 'WKSP_BHARGAVI44'. The main area is titled 'SQL Commands' with a search bar and a 'Run' button. The code editor contains the following SQL command:

```
1 DELETE FROM read_copy_d_cds WHERE year = '2001';
2
3
4
5
6
7
8
9
10
```

The results section below shows the output: '0 row(s) deleted.'

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

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

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

DELETE, INSERT, MODIFY restricted if it contains:

Group functions
GROUP BY CLAUSE
DISTINCT
pseudocolumn ROWNUM Keyword

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

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

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

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

Managing Views

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

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

```
SELECT * FROM view_copy_d_songs;
```

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there is a search bar, a user icon for 'bhargavi k bhargavi44', and a 'Run' button. The main workspace is titled 'SQL Commands'. It features a toolbar with icons for Undo, Redo, Find, and Save. Below the toolbar, the schema is set to 'WKSP_BHARGAVI44'. The SQL editor contains the following code:

```
1 SELECT * FROM view_copy_d_songs;
2
3
4
5
6
7
8
9
```

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

TITLE	ARTIST
Mello Jello	The What

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;
```

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, a user icon for 'bhargavi k bhargavi44', and a 'Run' button. Below the navigation is a toolbar with icons for Undo, Redo, Find, and others. The main area is titled 'SQL Commands' and contains a code editor with the following SQL:

```
1 SELECT * FROM (SELECT last_name, salary FROM emp2 ORDER BY salary DESC) WHERE ROWNUM <= 3;
2
3
4
5
6
7
8
9
```

Below the code editor is a results grid with columns 'LAST_NAME' and 'SALARY'. The data is as follows:

LAST_NAME	SALARY
Mohan	76569
Harish	56467

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

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

The screenshot shows the Oracle SQL Workshop interface, identical to the previous one but with a different query. The top navigation bar and toolbar are the same. The code editor contains the SQL from step 4:

```
1 .dept_id = emp2.department_number GROUP BY dept.dept_id) dept LEFT OUTER JOIN emp2 ON dept.dept_id = emp2.department_number WHERE NVL(emp2.salary,0) = dept.max_dpt_sal;
```

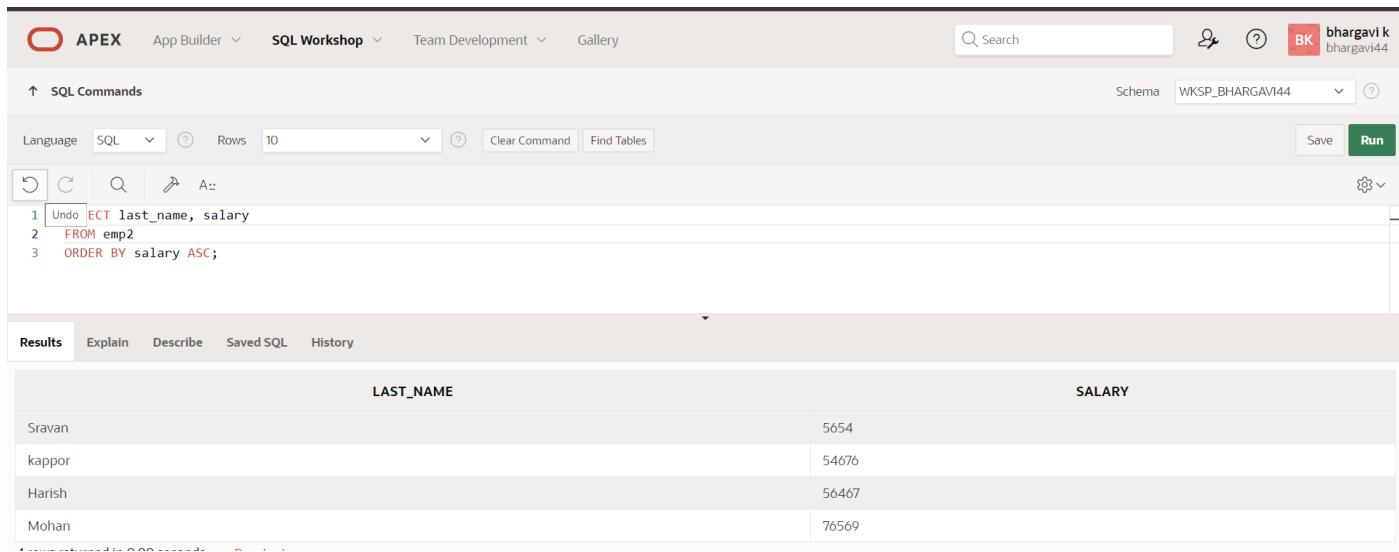
Below the code editor is a results grid with columns 'LAST_NAME', 'SALARY', and 'DEPT_ID'. The data is as follows:

LAST_NAME	SALARY	DEPT_ID
Harish	56467	80
kappor	54676	56
-	-	545
-	-	55

At the bottom left, it says '4 rows returned in 0.02 seconds' and there's a 'Download' link.

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

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



The screenshot shows the Oracle SQL Workshop interface. At the top, there are tabs for APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, a user icon for 'bhargavi k bhargav144', and a 'Run' button. Below the tabs, there's a toolbar with icons for Undo, Redo, Find, and Copy. The main area shows the SQL command entered:

```
1 SELECT last_name, salary  
2 FROM emp2  
3 ORDER BY salary ASC;
```

Below the command, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, displaying a table with two columns: LAST_NAME and SALARY. The data is as follows:

LAST_NAME	SALARY
Sravan	5654
kappor	54676
Harish	56467
Mohan	76569

Indexes and Synonyms

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

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

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

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

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

3. When will an index be created automatically?

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

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

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

The screenshot shows the Oracle Application Express SQL Workshop Data Browser interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, a user icon for 'bhargavi k bhargav144', and a 'Run' button. The main area is titled 'SQL Commands' with a 'Language' dropdown set to 'SQL'. Below it, there are buttons for 'Clear Command' and 'Find Tables'. The SQL editor contains the following code:

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

Below the editor, tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History' are visible. The results section shows the message 'no data found'.

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

5. 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';

6. 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;

7. 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));**

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

CREATE SYNONYM dj_tracks2 FOR d_track_listings;

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

10. Drop the synonym that you created in question

DROP SYNONYM dj_tracks2;

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

RESULT:

OTHER DATABASE OBJECTS

EX_NO:14

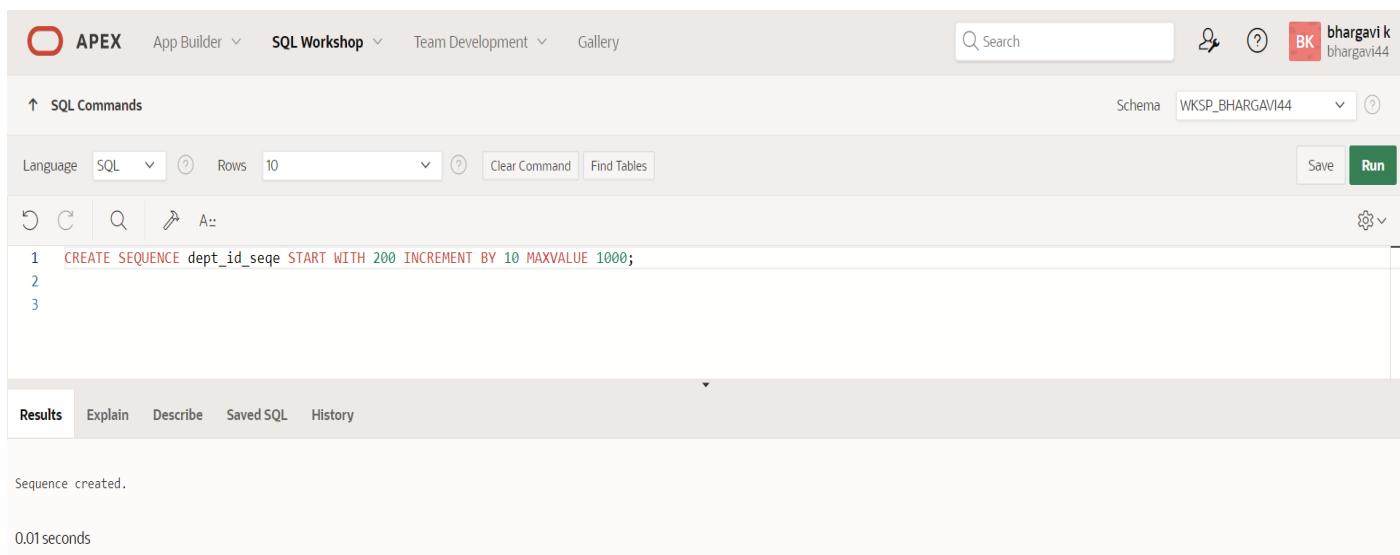
DATE:

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. The schema dropdown shows 'WKSP_BHARGAVI44'. The code editor contains the following SQL command:

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

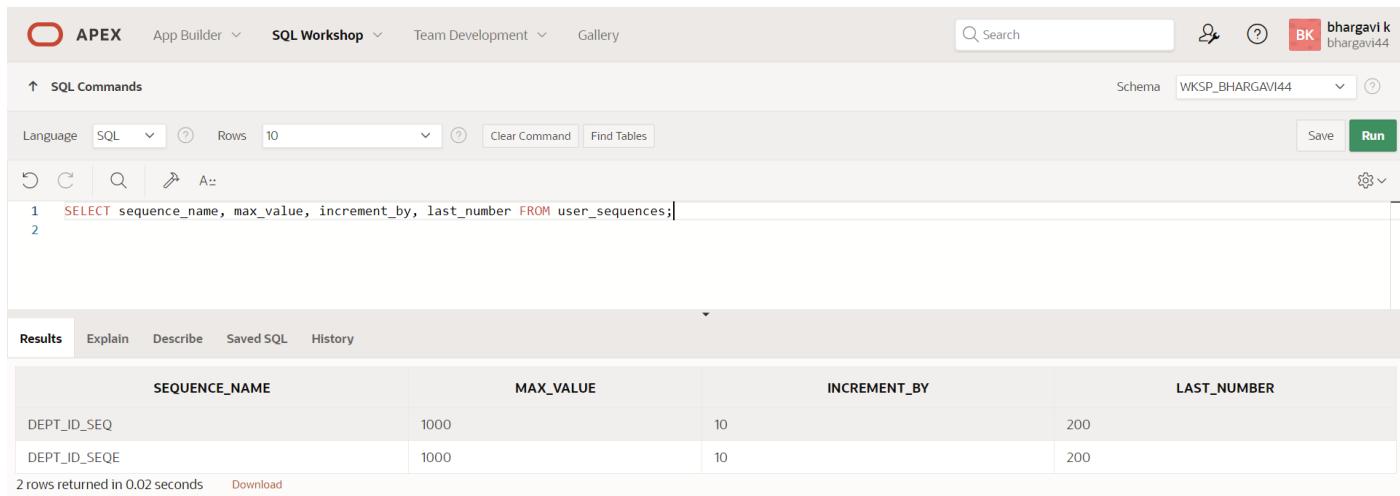
The 'Results' tab is active, showing the output: "Sequence created." Below the results, it says "0.01 seconds".

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. The schema dropdown shows 'WKSP_BHARGAVI44'. The code editor contains the following SQL command:

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

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

SEQUENCE_NAME	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
DEPT_ID_SEQ	1000	10	200
DEPT_ID_SEQE	1000	10	200

At the bottom, it says "2 rows returned in 0.02 seconds" and has a "Download" link.

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 Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. A search bar and user profile are also present. The main area is titled 'SQL Commands' with a 'Schema' dropdown set to 'WKSP_BHARGAV'. The SQL editor contains the following code:

```
1
2  INSERT INTO dept1|VALUES (dept_id_seq.nextval, 'Education');
3
4
```

Below the editor, the 'Results' tab is selected, showing the output:

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

Execution time: 0.03 seconds.

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. A search bar and user profile are also present. The main area is titled 'SQL Commands' with a 'Schema' dropdown set to 'WKSP_BHARGAVI44'. The SQL editor contains the following code:

```
1  CREATE INDEX emp_dept_id_idx ON emp2| (department_number);
2
```

Below the editor, the 'Results' tab is selected, showing the output:

```
Index created.
```

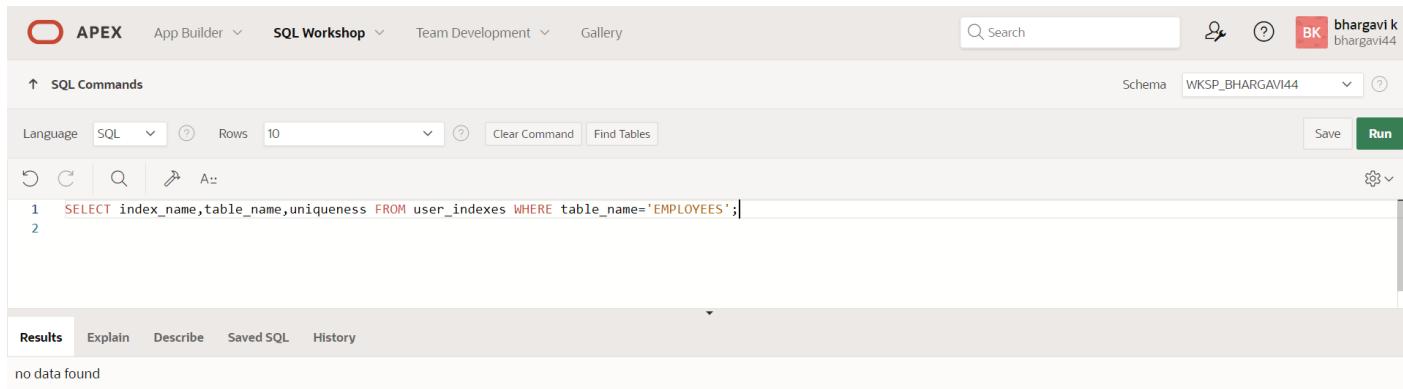
Execution time: 0.04 seconds.

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. At the top, there are tabs for 'APEX', 'App Builder', 'SQL Workshop' (which is selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, a user icon for 'bhargavi k bhargavi44', and a 'Run' button. Below the tabs, there's a toolbar with icons for 'SQL Commands', 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. The main area contains the SQL query:

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

Below the query, there are tabs for 'Results' (selected), 'Explain', 'Describe', 'Saved SQL', and 'History'. The results section displays the message 'no data found'.

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 department 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;`

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

RESULT:

PL/SQL

CONTROL STRUCTURES

EX.NO:16

DATE:

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'SQL Commands' and a schema dropdown set to 'WKSP_SHRAVANTHI902CSI'. Below the bar are buttons for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', 'Find Tables', 'Save', and a large green 'Run' button. The main workspace displays the PL/SQL code. The code is numbered from 1 to 8. Lines 1-7 represent the PL/SQL block, and line 8 ends the block with 'END;'. The code is syntax-highlighted. Below the code, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, showing the output: 'Incentive = 660' and 'Statement processed.' At the bottom, there are footer links for user information ('220701902@rajalakshmi.edu.in', 'shravanti902cse'), copyright notice ('Copyright © 1999, 2023, Oracle and/or its affiliates.'), and version information ('Oracle APEX 23.2.4').

```
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;
```

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

QUERY:

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

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

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the code editor, there are two nearly identical PL/SQL blocks. The first block uses quoted identifiers ('WELCOME') and the second uses unquoted identifiers ('welcome'). Both blocks include a DBMS_OUTPUT.PUT_LINE statement. The code editor has syntax highlighting and line numbers (1 through 6). Below the code editor is a results panel. The results panel displays an error message in a yellow box: "Error at line 4/23: ORA-06550: line 4, column 23: PLS-00201: identifier 'Welcome' must be declared ORA-06512: at "SYS.OWW_DBMS_SQL_APEX_230200", line 801 ORA-06550: line 4, column 1: PL/SQL: Statement ignored". At the bottom of the results panel, the line "2. WELCOME varchar2(10) := 'welcome';" is visible.

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

Error at line 4/23: ORA-06550: line 4, column 23:
PLS-00201: identifier 'Welcome' must be declared
ORA-06512: at "SYS.OWW_DBMS_SQL_APEX_230200", line 801
ORA-06550: line 4, column 1:
PL/SQL: Statement ignored

2. WELCOME varchar2(10) := 'welcome';

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

QUERY:

```
DECLARE
    salary_of_emp NUMBER(8,2);
PROCEDURE approx_salary (
    emp      NUMBER,
    empsal IN OUT NUMBER,
    addless  NUMBER
) IS
BEGIN
    empsal := empsal + addless;
END;
/
BEGIN
    SELECT salary INTO salary_of_emp
    FROM employees
    WHERE employee_id = 122;
    DBMS_OUTPUT.PUT_LINE
    ('Before invoking procedure, salary_of_emp: ' || salary_of_emp);
    approx_salary (100, salary_of_emp, 1000);
    DBMS_OUTPUT.PUT_LINE
    ('After invoking procedure, salary_of_emp: ' || salary_of_emp);
END;
/
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Commands interface. The code area contains the PL/SQL block provided above. The results pane shows the output of the `DBMS_OUTPUT.PUT_LINE` statements. The bottom status bar indicates "Statement processed."

```
1  DECLARE
2      salary_of_emp NUMBER(8,2);
3  PROCEDURE approx_salary (
4      emp      NUMBER,
5      empsal IN OUT NUMBER,
```

Results

```
Before invoking procedure, salary_of_emp: 20000
After invoking procedure, salary_of_emp: 21000

Statement processed.
```

220701902@rajalakshmi.edu.in shrawanthi902cse Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'SQL Commands' and a schema dropdown set to 'WKSP_SHRAVANTHI902CSI'. The main toolbar has buttons for Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run. Below the toolbar is a toolbar with icons for Undo, Redo, Search, Insert, and others. The code editor displays the PL/SQL procedure definition. The bottom section shows the results of the execution, indicating the procedure was created successfully in 0.03 seconds. The footer includes user information (email: 220701902@rajalakshmi.edu.in, workspace: shrawanthi902cse), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the version (Oracle APEX 25.2.4).

```
1 CREATE OR REPLACE PROCEDURE pri_bool(
2     boo_name  VARCHAR2,
3     boo_val   BOOLEAN
4 ) IS
5 BEGIN
```

Procedure created.

0.03 seconds

220701902@rajalakshmi.edu.in shrawanthi902cse Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 25.2.4

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle APEX SQL workspace interface. The top navigation bar includes 'SQL Commands', 'Schema' set to 'WKSP_SHRAVANTHI902CSI', and a 'Run' button. Below the toolbar are icons for Undo, Redo, Find, and Sort. The main area displays the PL/SQL code for the 'pat_match' procedure. The results tab shows the output of the executed code, displaying 'TRUE' and 'FALSE' as expected. The bottom footer includes user information, copyright notice, and the version 'Oracle APEX 23.2.4'.

```
1  DECLARE
2  |  PROCEDURE pat_match (
3  |    test_string  VARCHAR2,
4  |    pattern      VARCHAR2
5  |  ) IS
```

Results

```
TRUE
FALSE

Statement processed.
```

220701902@rajalakshmi.edu.in shrawanthi902cse Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

QUERY:

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

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

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

OUTPUT:

The screenshot shows the Oracle APEX SQL workspace interface. The top navigation bar includes 'SQL Commands' and a schema dropdown set to 'WKSP_SHRAVANTHI902CSI'. Below the bar are buttons for Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run. The main area displays the PL/SQL code. The 'Results' tab is selected, showing the output of the DBMS_OUTPUT.PUT_LINE statements: 'num_small = 5' and 'num_large = 8'. At the bottom, it says 'Statement processed.' and includes copyright information for Oracle APEX 23.2.4.

```
1  DECLARE
2  num_small NUMBER := 8;
3  num_large NUMBER := 5;
4  num_temp NUMBER;
5  BEGIN

num_small = 5
num_large = 8

Statement processed.

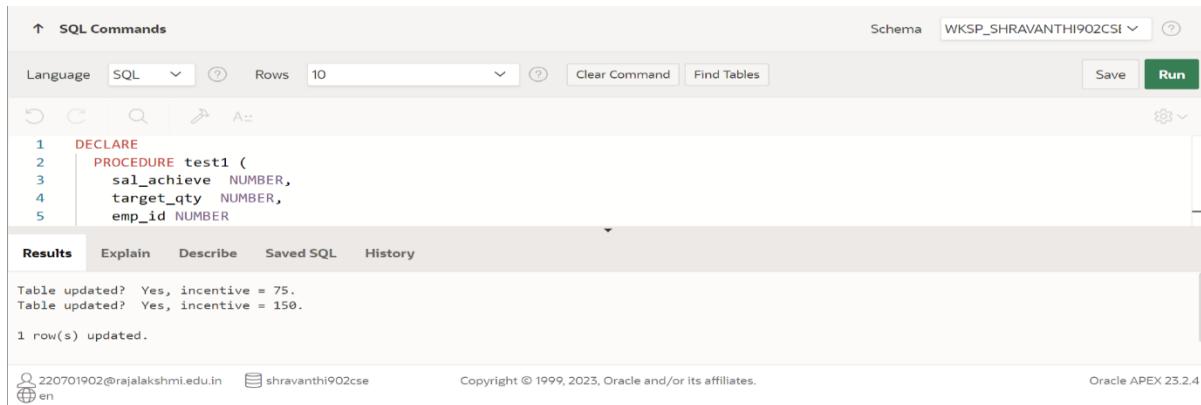
Copyright © 1999, 2023, Oracle and/or its affiliates.
Oracle APEX 23.2.4
```

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle APEX interface with the SQL Commands page open. The code has been executed, and the results are displayed in the Results tab. The output shows two rows of data, each indicating that the table was updated and specifying the incentive amount.

Result	Table updated?	Incentive
1	Yes	75.
2	Yes	150.

Below the results, it says "1 row(s) updated."

At the bottom of the page, there are footer links for "220701902@rajalakshmi.edu.in", "shravanthi902cse", "Copyright © 1999, 2023, Oracle and/or its affiliates.", and "Oracle APEX 23.2.4".

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

QUERY:

DECLARE

```
PROCEDURE test1 (sal_achieve NUMBER)
```

IS

```
    incentive NUMBER := 0;
```

BEGIN

```
    IF sal_achieve > 44000 THEN
```

```
        incentive := 1800;
```

```
    ELSIF sal_achieve > 32000 THEN
```

```
        incentive := 800;
```

ELSE

```
        incentive := 500;
```

END IF;

```
    DBMS_OUTPUT.NEW_LINE;
```

```
    DBMS_OUTPUT.PUT_LINE (
```

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

```
    );
```

```
END test1;
```

BEGIN

```
    test1(45000);
```

```
    test1(36000);
```

```
    test1(28000);
```

END;

OUTPUT:

The screenshot shows the Oracle APEX interface with the following details:

- Toolbar:** SQL Commands, Language (SQL), Rows (10), Clear Command, Find Tables, Save, Run.
- Code Area:** The code block above is pasted here, showing the declaration of a procedure named test1 that takes a parameter sal_achieve and calculates an incentive based on its value. It uses DBMS_OUTPUT to print the results.
- Results Tab:** The tab is selected, showing the output of the procedure execution.
- Output Content:**

```
Sale achieved : 45000, incentive : 1800.  
Sale achieved : 36000, incentive : 800.  
Sale achieved : 28000, incentive : 500.  
Statement processed.
```
- Page Footer:** Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4.

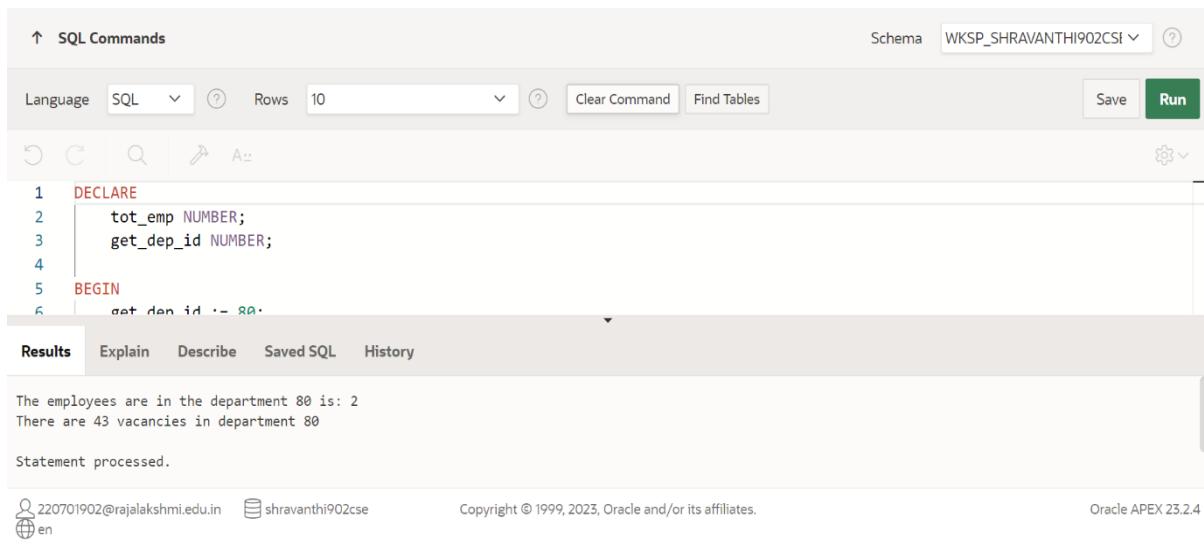
9.) Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

QUERY:

```
SET SERVEROUTPUT ON
DECLARE
    tot_emp NUMBER;
    get_dep_id NUMBER;

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

OUTPUT:



The screenshot shows the Oracle APEX SQL Commands interface. The code entered is a PL/SQL block that counts employees in department 80 and checks for vacancies. The output window shows the results of the execution.

```
1  DECLARE
2      tot_emp NUMBER;
3      get_dep_id NUMBER;
4
5  BEGIN
6      get_dep_id := 80;
7
8      SELECT Count(*)
9      INTO tot_emp
10     FROM employees e
11         join departments d
12             ON e.department_id = d.department_id
13    WHERE e.department_id = get_dep_id;
14
15      dbms_output.Put_line ('The employees are in the department '| |get_dep_id| | is: '
16          ||To_char(tot_emp));
17
18      IF tot_emp >= 45 THEN
19          dbms_output.Put_line ('There are no vacancies in the department '| |get_dep_id);
20      ELSE
21          dbms_output.Put_line ('There are '| |to_char(45-tot_emp)| | vacancies in department '| | get_dep_id );
22      END IF;
23
24  END;
25
```

The output window displays the following results:

```
The employees are in the department 80 is: 2
There are 43 vacancies in department 80

Statement processed.
```

At the bottom, the footer includes:

- 220701902@rajalakshmi.edu.in
- shravanthi902cse
- Copyright © 1999, 2023, Oracle and/or its affiliates.
- Oracle APEX 23.2.4

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

QUERY:

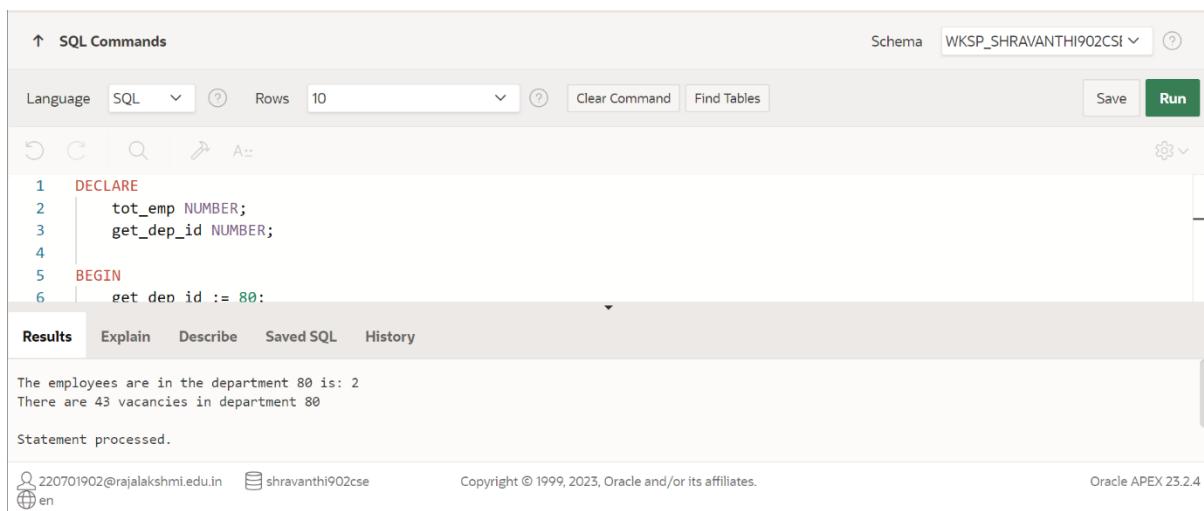
```
DECLARE
    tot_emp NUMBER;
    get_dep_id NUMBER;

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

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

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

OUTPUT:



The screenshot shows the Oracle APEX SQL workspace interface. The top navigation bar includes 'SQL Commands', 'Schema' set to 'WKSP_SHRAVANTHI902CSI', and a 'Run' button. Below the toolbar are standard database navigation icons. The main area contains the PL/SQL code from the previous section. The 'Results' tab is selected, displaying the output of the executed code. The output shows two lines of text: 'The employees are in the department 80 is: 2' and 'There are 43 vacancies in department 80'. At the bottom of the results pane, it says 'Statement processed.' The footer of the page includes copyright information for Oracle and links to user profiles and the Oracle APEX version.

```
1  DECLARE
2      tot_emp NUMBER;
3      get_dep_id NUMBER;
4
5  BEGIN
6      get dep id := 80;
7
8      dbms_output.Put_line ('The employees are in the department '| |get_dep_id| |' is: '
9          ||To_char(tot_emp));
10
11     IF tot_emp >= 45 THEN
12         dbms_output.Put_line ('There are no vacancies in the department '| |get_dep_id);
13     ELSE
14         dbms_output.Put_line ('There are '| |to_char(45-tot_emp)| |' vacancies in department '| | get_dep_id );
15     END IF;
16
17 END;
18 /
```

The employees are in the department 80 is: 2
There are 43 vacancies in department 80
Statement processed.

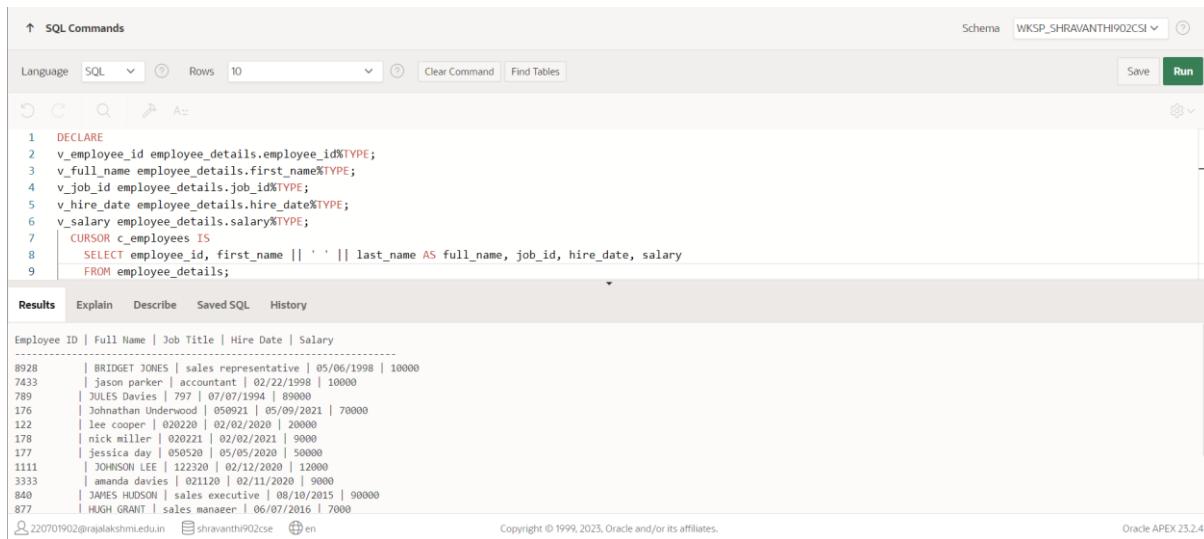
220701902@rajalakshmi.edu.in shravanti902cse Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle APEX SQL Commands interface. The code area contains the PL/SQL block provided above. The results section displays the output of the program, which is a tabular report of employee details:

Employee ID	Full Name	Job Title	Hire Date	Salary
8928	BRIDGET JONES	sales representative	05/06/1998	10000
7433	jason parker	accountant	02/22/1998	10000
789	JULES Davies	797	07/07/1994	89000
176	Johnathan Underwood	050921	05/09/2021	70000
122	lee cooper	020220	02/02/2020	20000
178	nick miller	020221	02/02/2021	9000
177	jessica day	050520	05/05/2020	50000
1111	JOHNSON LEE	123230	02/12/2020	12000
3333	amanda davies	021120	02/11/2020	9000
840	JAMES HUDSON	sales executive	08/10/2015	90000
877	HUGH GRANT	sales manager	06/07/2016	7000

At the bottom of the interface, there are footer links: Copyright © 1999, 2023, Oracle and/or its affiliates., Oracle APEX 23.2.4, and a session identifier.

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:

```
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;
```

Employee ID	Employee Name	Manager Name
8928	BRIDGET	
178	nick	
789	JULES	
122	lee	
877		

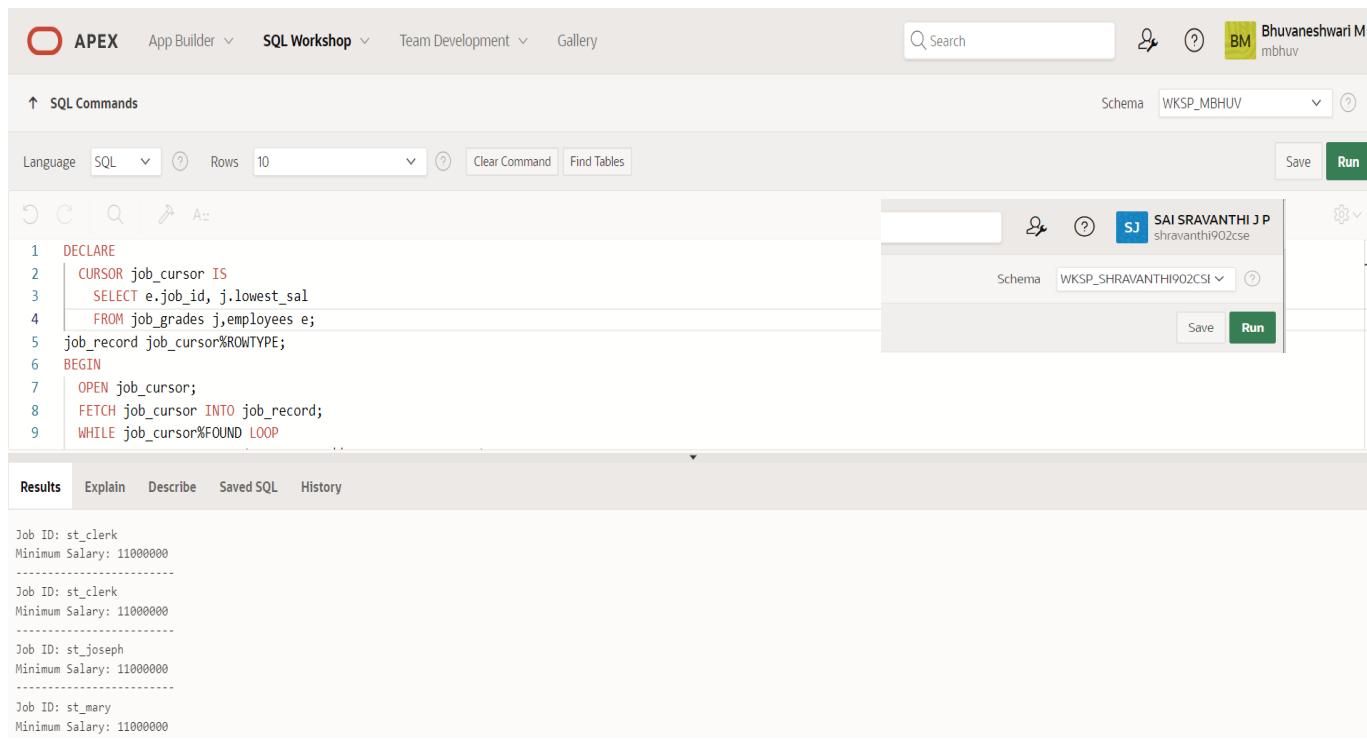
13.) Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all jobs

QUERY:

DECLARE

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there are user profile icons for Bhuvaneshwari M (mbhuv) and SAI SRAVANTHI J P (shravanthi902cse). The main workspace has two tabs open: one for the SQL Commands editor containing the PL/SQL code, and another for the Results viewer displaying the output.

SQL Commands Tab:

```
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
```

Results Tab:

Job ID	Minimum Salary
st_clerk	11000000
st_clerk	11000000
st_joseph	11000000
st_mary	11000000

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

QUERY:

DECLARE

```
CURSOR employees_cur IS
  SELECT employee_id, last_name, job_id, start_date
  FROM employees NATURAL JOIN job_history;
  emp_start_date DATE;

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

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema is set to WKSP_SHRAVANTHI902CSI. The code in the SQL Commands pane is identical to the one provided above. The Results pane displays the output of the PL/SQL code, which includes the employee ID, last name, job ID, and start date for each employee, formatted as specified in the code.

Employee ID	Last Name	Job Id	Start Date
101	Aasper	AD_VP	2000-01-01
102	Buchanan	AD_ASST	2000-01-01
103	Fochester	AD_ASST	2000-01-01
104	Morgane	AD_ASST	2000-01-01
105	Snavely	AD_ASST	2000-01-01
106	Fuller	AD_PRES	2000-01-01
107	King	AD_PRES	2000-01-01
108	Plumbe	AD_VP	2000-01-01
109	Abrahams	AD_ASST	2000-01-01
110	Andersen	AD_ASST	2000-01-01
111	Frederiksen	AD_ASST	2000-01-01
112	Knudsen	AD_ASST	2000-01-01
113	Christensen	AD_ASST	2000-01-01
114	Willeke	AD_ASST	2000-01-01
115	Hammer	AD_ASST	2000-01-01
116	Elisabeth	AD_ASST	2000-01-01
117	Andreas	AD_ASST	2000-01-01
118	Stephanie	AD_ASST	2000-01-01
119	Markus	AD_ASST	2000-01-01
120	Thomas	AD_ASST	2000-01-01
121	Mathias	AD_ASST	2000-01-01
122	Elisabeth	AD_ASST	2000-01-01
123	Stephanie	AD_ASST	2000-01-01
124	Markus	AD_ASST	2000-01-01
125	Thomas	AD_ASST	2000-01-01
126	Mathias	AD_ASST	2000-01-01
127	Elisabeth	AD_ASST	2000-01-01
128	Stephanie	AD_ASST	2000-01-01
129	Markus	AD_ASST	2000-01-01
130	Thomas	AD_ASST	2000-01-01
131	Mathias	AD_ASST	2000-01-01
132	Elisabeth	AD_ASST	2000-01-01
133	Stephanie	AD_ASST	2000-01-01
134	Markus	AD_ASST	2000-01-01
135	Thomas	AD_ASST	2000-01-01
136	Mathias	AD_ASST	2000-01-01
137	Elisabeth	AD_ASST	2000-01-01
138	Stephanie	AD_ASST	2000-01-01
139	Markus	AD_ASST	2000-01-01
140	Thomas	AD_ASST	2000-01-01
141	Mathias	AD_ASST	2000-01-01
142	Elisabeth	AD_ASST	2000-01-01
143	Stephanie	AD_ASST	2000-01-01
144	Markus	AD_ASST	2000-01-01
145	Thomas	AD_ASST	2000-01-01
146	Mathias	AD_ASST	2000-01-01
147	Elisabeth	AD_ASST	2000-01-01
148	Stephanie	AD_ASST	2000-01-01
149	Markus	AD_ASST	2000-01-01
150	Thomas	AD_ASST	2000-01-01
151	Mathias	AD_ASST	2000-01-01
152	Elisabeth	AD_ASST	2000-01-01
153	Stephanie	AD_ASST	2000-01-01
154	Markus	AD_ASST	2000-01-01
155	Thomas	AD_ASST	2000-01-01
156	Mathias	AD_ASST	2000-01-01
157	Elisabeth	AD_ASST	2000-01-01
158	Stephanie	AD_ASST	2000-01-01
159	Markus	AD_ASST	2000-01-01
160	Thomas	AD_ASST	2000-01-01
161	Mathias	AD_ASST	2000-01-01
162	Elisabeth	AD_ASST	2000-01-01
163	Stephanie	AD_ASST	2000-01-01
164	Markus	AD_ASST	2000-01-01
165	Thomas	AD_ASST	2000-01-01
166	Mathias	AD_ASST	2000-01-01
167	Elisabeth	AD_ASST	2000-01-01
168	Stephanie	AD_ASST	2000-01-01
169	Markus	AD_ASST	2000-01-01
170	Thomas	AD_ASST	2000-01-01
171	Mathias	AD_ASST	2000-01-01
172	Elisabeth	AD_ASST	2000-01-01
173	Stephanie	AD_ASST	2000-01-01
174	Markus	AD_ASST	2000-01-01
175	Thomas	AD_ASST	2000-01-01
176	Mathias	AD_ASST	2000-01-01
177	Elisabeth	AD_ASST	2000-01-01
178	Stephanie	AD_ASST	2000-01-01
179	Markus	AD_ASST	2000-01-01
180	Thomas	AD_ASST	2000-01-01
181	Mathias	AD_ASST	2000-01-01
182	Elisabeth	AD_ASST	2000-01-01
183	Stephanie	AD_ASST	2000-01-01
184	Markus	AD_ASST	2000-01-01
185	Thomas	AD_ASST	2000-01-01
186	Mathias	AD_ASST	2000-01-01
187	Elisabeth	AD_ASST	2000-01-01
188	Stephanie	AD_ASST	2000-01-01
189	Markus	AD_ASST	2000-01-01
190	Thomas	AD_ASST	2000-01-01
191	Mathias	AD_ASST	2000-01-01
192	Elisabeth	AD_ASST	2000-01-01
193	Stephanie	AD_ASST	2000-01-01
194	Markus	AD_ASST	2000-01-01
195	Thomas	AD_ASST	2000-01-01
196	Mathias	AD_ASST	2000-01-01
197	Elisabeth	AD_ASST	2000-01-01
198	Stephanie	AD_ASST	2000-01-01
199	Markus	AD_ASST	2000-01-01
200	Thomas	AD_ASST	2000-01-01
201	Mathias	AD_ASST	2000-01-01
202	Elisabeth	AD_ASST	2000-01-01
203	Stephanie	AD_ASST	2000-01-01
204	Markus	AD_ASST	2000-01-01
205	Thomas	AD_ASST	2000-01-01
206	Mathias	AD_ASST	2000-01-01
207	Elisabeth	AD_ASST	2000-01-01
208	Stephanie	AD_ASST	2000-01-01
209	Markus	AD_ASST	2000-01-01
210	Thomas	AD_ASST	2000-01-01
211	Mathias	AD_ASST	2000-01-01
212	Elisabeth	AD_ASST	2000-01-01
213	Stephanie	AD_ASST	2000-01-01
214	Markus	AD_ASST	2000-01-01
215	Thomas	AD_ASST	2000-01-01
216	Mathias	AD_ASST	2000-01-01
217	Elisabeth	AD_ASST	2000-01-01
218	Stephanie	AD_ASST	2000-01-01
219	Markus	AD_ASST	2000-01-01
220	Thomas	AD_ASST	2000-01-01
221	Mathias	AD_ASST	2000-01-01
222	Elisabeth	AD_ASST	2000-01-01
223	Stephanie	AD_ASST	2000-01-01
224	Markus	AD_ASST	2000-01-01
225	Thomas	AD_ASST	2000-01-01
226	Mathias	AD_ASST	2000-01-01
227	Elisabeth	AD_ASST	2000-01-01
228	Stephanie	AD_ASST	2000-01-01
229	Markus	AD_ASST	2000-01-01
230	Thomas	AD_ASST	2000-01-01
231	Mathias	AD_ASST	2000-01-01
232	Elisabeth	AD_ASST	2000-01-01
233	Stephanie	AD_ASST	2000-01-01
234	Markus	AD_ASST	2000-01-01
235	Thomas	AD_ASST	2000-01-01
236	Mathias	AD_ASST	2000-01-01
237	Elisabeth	AD_ASST	2000-01-01
238	Stephanie	AD_ASST	2000-01-01
239	Markus	AD_ASST	2000-01-01
240	Thomas	AD_ASST	2000-01-01
241	Mathias	AD_ASST	2000-01-01
242	Elisabeth	AD_ASST	2000-01-01
243	Stephanie	AD_ASST	2000-01-01
244	Markus	AD_ASST	2000-01-01
245	Thomas	AD_ASST	2000-01-01
246	Mathias	AD_ASST	2000-01-01
247	Elisabeth	AD_ASST	2000-01-01
248	Stephanie	AD_ASST	2000-01-01
249	Markus	AD_ASST	2000-01-01
250	Thomas	AD_ASST	2000-01-01
251	Mathias	AD_ASST	2000-01-01
252	Elisabeth	AD_ASST	2000-01-01
253	Stephanie	AD_ASST	2000-01-01
254	Markus	AD_ASST	2000-01-01
255	Thomas	AD_ASST	2000-01-01
256	Mathias	AD_ASST	2000-01-01
257	Elisabeth	AD_ASST	2000-01-01
258	Stephanie	AD_ASST	2000-01-01
259	Markus	AD_ASST	2000-01-01
260	Thomas	AD_ASST	2000-01-01
261	Mathias	AD_ASST	2000-01-01
262	Elisabeth	AD_ASST	2000-01-01
263	Stephanie	AD_ASST	2000-01-01
264	Markus	AD_ASST	2000-01-01
265	Thomas	AD_ASST	2000-01-01
266	Mathias	AD_ASST	2000-01-01
267	Elisabeth	AD_ASST	2000-01-01
268	Stephanie	AD_ASST	2000-01-01
269	Markus	AD_ASST	2000-01-01
270	Thomas	AD_ASST	2000-01-01
271	Mathias	AD_ASST	2000-01-01
272	Elisabeth	AD_ASST	2000-01-01
273	Stephanie	AD_ASST	2000-01-01
274	Markus	AD_ASST	2000-01-01
275	Thomas	AD_ASST	2000-01-01
276	Mathias	AD_ASST	2000-01-01
277	Elisabeth	AD_ASST	2000-01-01
278	Stephanie	AD_ASST	2000-01-01
279	Markus	AD_ASST	2000-01-01
280	Thomas	AD_ASST	2000-01-01
281	Mathias	AD_ASST	2000-01-01
282	Elisabeth	AD_ASST	2000-01-01
283	Stephanie	AD_ASST	2000-01-01
284	Markus	AD_ASST	2000-01-01
285	Thomas	AD_ASST	2000-01-01
286	Mathias	AD_ASST	2000-01-01
287	Elisabeth	AD_ASST	2000-01-01
288	Stephanie	AD_ASST	2000-01-01
289	Markus	AD_ASST	2000-01-01
290	Thomas	AD_ASST	2000-01-01
291	Mathias	AD_ASST	2000-01-01
292	Elisabeth	AD_ASST	2000-01-01
293	Stephanie	AD_ASST	2000-01-01
294	Markus	AD_ASST	2000-01-01
295	Thomas	AD_ASST	2000-01-01
296	Mathias	AD_ASST	2000-01-01
297	Elisabeth	AD_ASST	2000-01-01
298	Stephanie	AD_ASST	2000-01-01
299	Markus	AD_ASST	2000-01-01
300	Thomas	AD_ASST	2000-01-01
301	Mathias	AD_ASST	2000-01-01
302	Elisabeth	AD_ASST	2000-01-01
303	Stephanie	AD_ASST	2000-01-01
304	Markus	AD_ASST	2000-01-01
305	Thomas	AD_ASST	2000-01-01
306	Mathias	AD_ASST	2000-01-01
307	Elisabeth	AD_ASST	2000-01-01
308	Stephanie	AD_ASST	2000-01-01
309	Markus	AD_ASST	2000-01-01
310	Thomas	AD_ASST	2000-01-01
311	Mathias	AD_ASST	2000-01-01
312	Elisabeth	AD_ASST	2000-01-01
313	Stephanie	AD_ASST	2000-01-01
314	Markus	AD_ASST	2000-01-01
315	Thomas	AD_ASST	2000-01-01
316	Mathias	AD_ASST	2000-01-01
317	Elisabeth	AD_ASST	2000-01-01
318	Stephanie	AD_ASST	2000-01-01
319	Markus	AD_ASST	2000-01-01
320	Thomas	AD_ASST	2000-01-01
321	Mathias	AD_ASST	2000-01-01
322	Elisabeth	AD_ASST	2000-01-01
323	Stephanie	AD_ASST	2000-01-01
324	Markus	AD_ASST	2000-01-01
325	Thomas	AD_ASST	2000-01-01
326	Mathias	AD_ASST	2000-01-01
327	Elisabeth	AD_ASST	2000-01-01
328	Stephanie	AD_ASST	2000-01-01
329	Markus	AD_ASST	2000-01-01
330	Thomas	AD_ASST	2000-01-01
331	Mathias	AD_ASST	2000-01-01
332	Elisabeth	AD_ASST	2000-01-01
333	Stephanie	AD_ASST	2000-01-01
334	Markus	AD_ASST	2000-01-01
335	Thomas	AD_ASST	2000-01-01
336	Mathias	AD_ASST	2000-01-01
337	Elisabeth	AD_ASST	2000-01-01
338	Stephanie	AD_ASST	2000-01-01
339	Markus	AD_ASST	2000-01-01
340	Thomas	AD_ASST	2000-01-01
341	Mathias	AD_ASST	2000-01-01
342	Elisabeth	AD_ASST	2000-01-01
343	Stephanie	AD_ASST	2000-01-01
344	Markus	AD_ASST	2000-01-01
345	Thomas	AD_ASST	2000-01-01
346	Mathias	AD_ASST	2000-01-01
347	Elisabeth	AD_ASST	2000-01-01
348	Stephanie	AD_ASST	2000-01-01
349	Markus	AD_ASST	2000-01-01
350	Thomas	AD_ASST	2000-01-01
351	Mathias	AD_ASST	2000-01-01
352	Elisabeth	AD_ASST	2000-01-01
353	Stephanie	AD_ASST	2000-01-01
354	Markus	AD_ASST	2000-01-01
355	Thomas	AD	

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

QUERY:

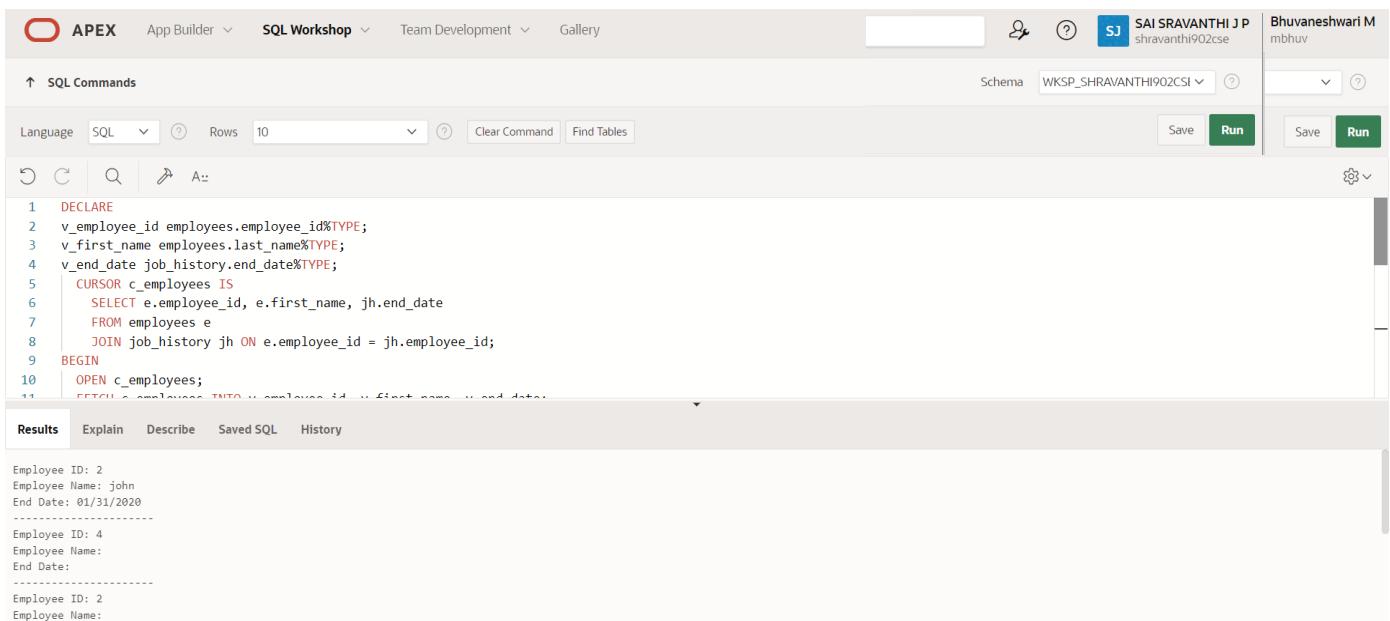
DECLARE

```
v_employee_id employees.employee_id%TYPE;
v_first_name employees.last_name%TYPE;
v_end_date job_history.end_date%TYPE;

CURSOR c_employees IS
SELECT e.employee_id, e.first_name, jh.end_date
FROM employees e
JOIN job_history jh ON e.employee_id = jh.employee_id;

BEGIN
OPEN c_employees;
FETCH c_employees INTO v_employee_id, v_first_name, v_end_date;
WHILE c_employees%FOUND LOOP
DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_employee_id);
DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_first_name);
DBMS_OUTPUT.PUT_LINE('End Date: ' || v_end_date);
DBMS_OUTPUT.PUT_LINE('-----');
FETCH c_employees INTO v_employee_id, v_first_name, v_end_date;
END LOOP;
CLOSE c_employees;
END;
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The user information shows SAI SRAVANTHI J P (shravanthi902cse) and Bhuvaneshwari M (mbhuv). The SQL Commands tab is active, displaying the PL/SQL code. The code declares variables for employee ID, first name, and end date, defines a cursor for employees, and performs a SELECT query. It then begins a loop to fetch rows from the cursor, outputting the employee ID, name, and end date using DBMS_OUTPUT.PUT_LINE, and separating each record with a dashed line. The results pane at the bottom shows the output for three employees: one record for employee ID 2, and two records for employee ID 4 (one with end date 01/31/2020 and one with null end date).

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

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:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'bhargavi k', and a connection name 'WKSP_BHARGAVI44'. The main area is titled 'SQL Commands' with tabs for Language (set to SQL), Rows (set to 10), and buttons for Clear Command and Find Tables. Below this is a code editor window containing the following PL/SQL block:

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

Below the code editor, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, showing the output: "Statement processed." and "0.00 seconds".

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

QUERY:

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

    p_title := p_title || ' - Retrieved';

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            p_title := NULL;
            p_author := NULL;
            p_year_published := NULL;
END;

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

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

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

OUTPUT:

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

```
1 CREATE OR REPLACE PROCEDURE get_book_info (
2     p_book_id IN NUMBER,
3     p_title IN OUT VARCHAR2,
4     p_author OUT VARCHAR2,
5     p_year_published OUT NUMBER
6 )
7 AS
8 BEGIN
9     SELECT title, author, year_published INTO p_title, p_author, p_year_published
10    FROM books
11   WHERE book_id = p_book_id;
```

The code is highlighted in red and blue, indicating syntax. Below the code, the 'Results' tab is selected, showing the message "Procedure created.".

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

RESULT:

TRIGGER

EX_NO: 18

DATE:

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface with the SQL Commands tab selected. The code for the trigger is entered in the command window. The output pane at the bottom displays the message "Trigger created.".

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

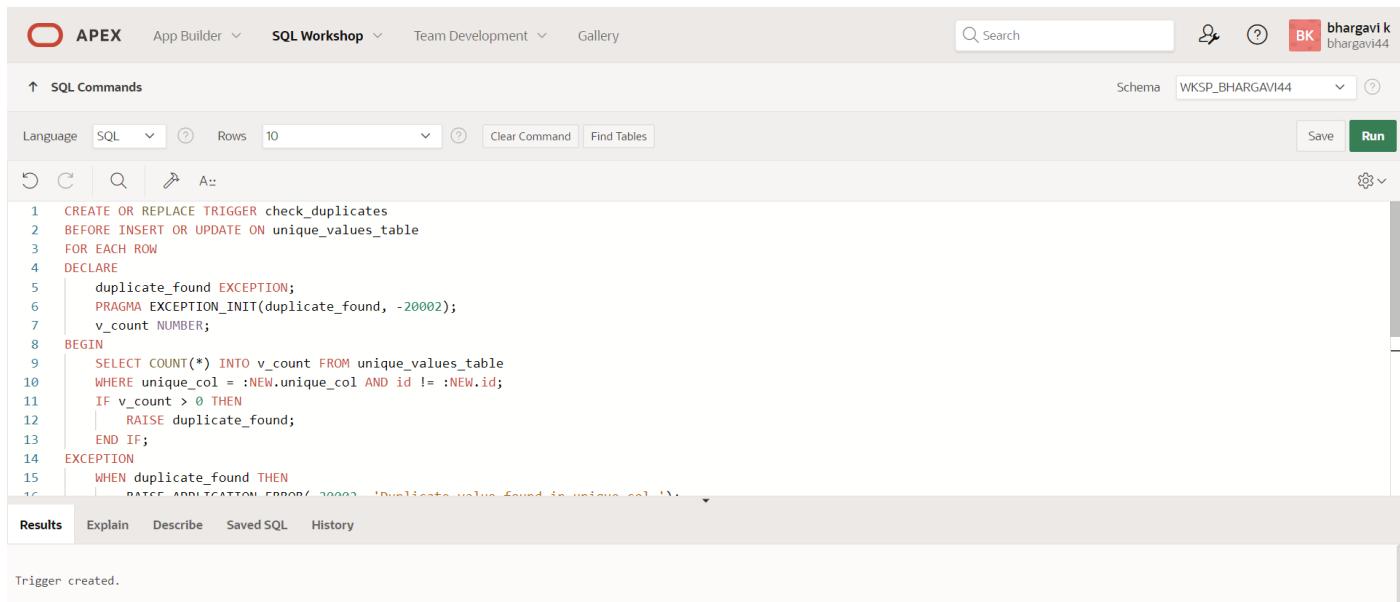
Trigger created.

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, a user icon, and a 'Run' button. The main area is titled 'SQL Commands' and contains the PL/SQL code for the trigger. Below the code, the 'Results' tab is selected, showing the message 'Trigger created.'.

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

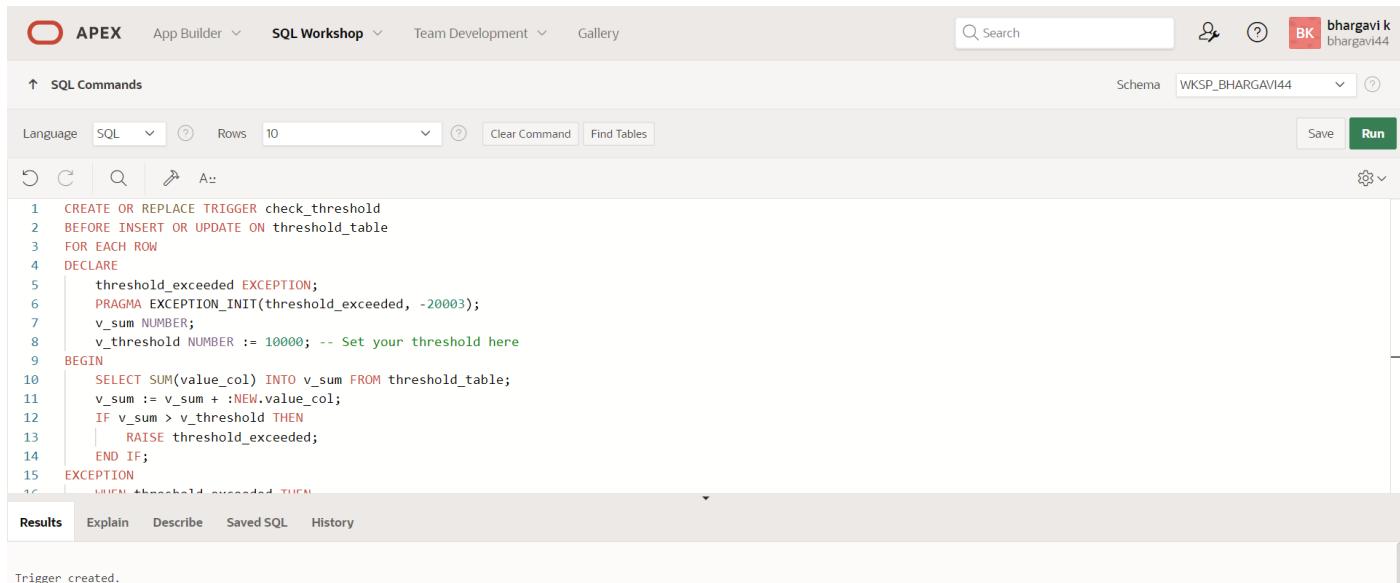
```

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

QUERY:

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

OUTPUT:



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

```
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.');
18 END;
```

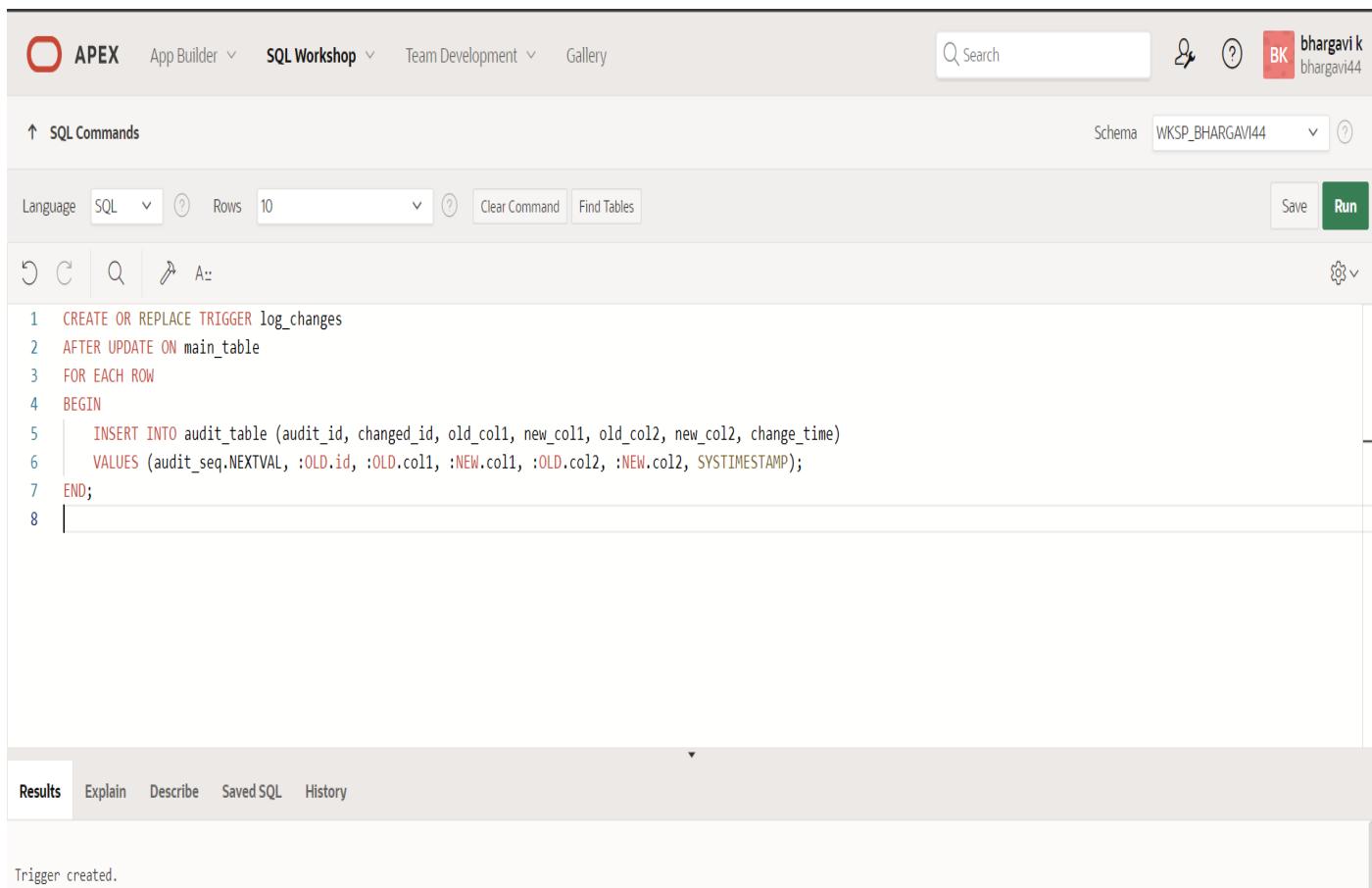
The status bar at the bottom of the interface shows the message "Trigger created."

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, a user icon for 'bhargavi k bhargavi44', and a 'Schema' dropdown set to 'WKSP_BHARGAVI44'. The main workspace is titled 'SQL Commands' and contains a code editor. The code editor shows the PL/SQL trigger definition with syntax highlighting for keywords like CREATE, OR REPLACE, TRIGGER, AFTER, UPDATE, FOR, EACH, BEGIN, and END. The code is as follows:

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

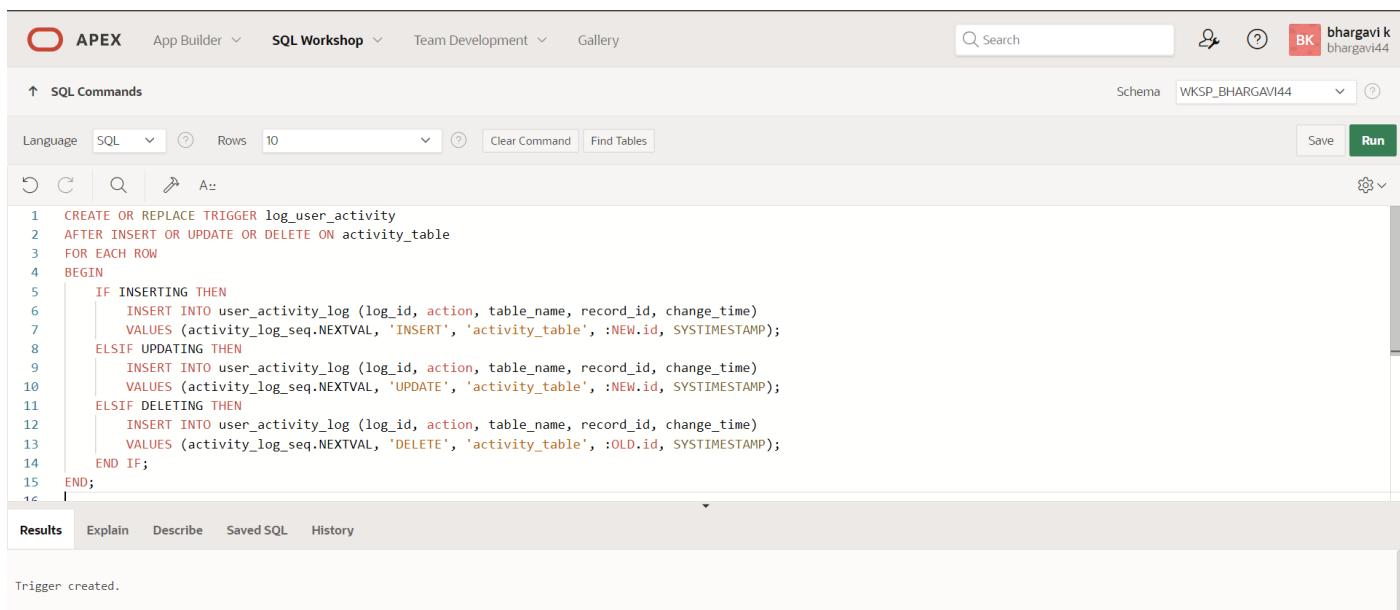
Below the code editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, displaying the message "Trigger created.".

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a user icon for 'bhargavi k' and a session identifier 'WKSP_BHARGAVI44'. The main area is titled 'SQL Commands' with a 'Run' button. Below that, there are filters for 'Language' (SQL), 'Rows' (10), and buttons for 'Clear Command' and 'Find Tables'. The SQL code for the trigger is pasted into the command window. At the bottom, tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History' are visible. The results pane displays the message 'Trigger created.'

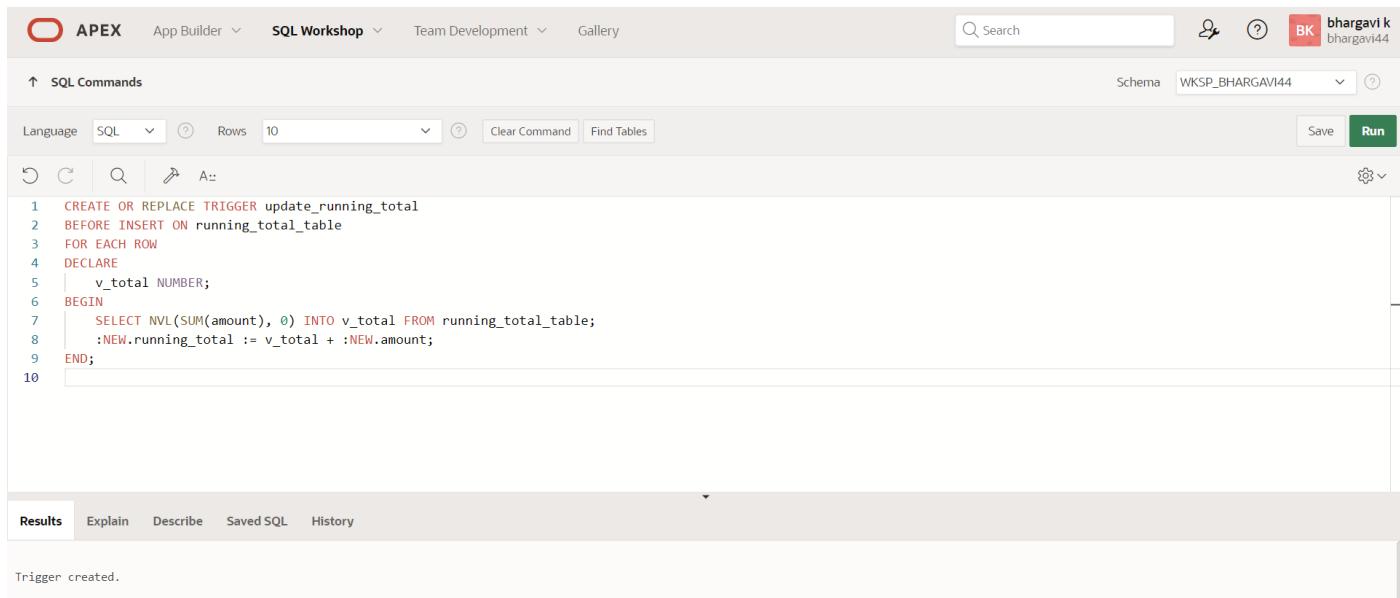
```
1 CREATE OR REPLACE TRIGGER log_user_activity
2 AFTER INSERT OR UPDATE OR DELETE ON activity_table
3 FOR EACH ROW
4 BEGIN
5   IF INSERTING THEN
6     INSERT INTO user_activity_log (log_id, action, table_name, record_id, change_time)
7     VALUES (activity_log_seq.NEXTVAL, 'INSERT', 'activity_table', :NEW.id, SYSTIMESTAMP);
8   ELSIF UPDATING THEN
9     INSERT INTO user_activity_log (log_id, action, table_name, record_id, change_time)
10    VALUES (activity_log_seq.NEXTVAL, 'UPDATE', 'activity_table', :NEW.id, SYSTIMESTAMP);
11   ELSIF DELETING THEN
12     INSERT INTO user_activity_log (log_id, action, table_name, record_id, change_time)
13     VALUES (activity_log_seq.NEXTVAL, 'DELETE', 'activity_table', :OLD.id, SYSTIMESTAMP);
14   END IF;
15 END;
16 
```

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'bhargavi k bhargavi44', and a 'Run' button. The main workspace is titled 'SQL Commands'. It has tabs for Language (SQL selected), Rows (10), Clear Command, Find Tables, Save, and Run. Below these are standard toolbar icons for Undo, Redo, Search, and Paste. The SQL editor contains the trigger creation code. The results tab at the bottom shows the message 'Trigger created.'

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

Results Explain Describe Saved SQL History

Trigger created.

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected, followed by 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. On the right side, there is a user profile for 'bhargavi k' (bhargav44). The main area is titled 'SQL Commands' and contains the PL/SQL code for the trigger. The code is highlighted in red and black, indicating syntax. Below the code, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, displaying the message 'Trigger created.'

```
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
        RAISE_APPLICATION_ERROR(-20004, 'Insufficient stock for the item.');
16
```

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

RESULT:

MONGO DB

EX_NO: 19

DATE:

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

QUERY:

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

OUTPUT:

The screenshot shows a MongoDB shell interface. At the top, it displays the user information: 220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K. Below this is a toolbar with a MongoDB icon, dropdown menus, and buttons for Run and Save. The main area contains a code editor with the following query:

```
1 db.restaurants.find(
2   { $or: [
3     { name: /^Wil/ },
4     { cuisine: { $nin: ['American', 'Chinese'] } }
5   ] },
6   { restaurant_id: 1, name: 1, borough: 1, cuisine: 1 }
7 );
8
```

To the right of the code editor is an "Output" window. It shows the command being run: "mycompiler_mongodb> ...". Below that, it shows the response: "mycompiler_mongodb> ...". At the bottom of the output window, it says "[Execution complete with exit code 0]".

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

QUERY:

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

OUTPUT:

The screenshot shows a MongoDB shell interface. At the top, it displays the user information: 220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K. Below this is a toolbar with a MongoDB icon, dropdown menus, and buttons for Run and Save. The main area contains a code editor with the following query:

```
1 db.restaurants.find(
2   { grades: { $elemMatch: { grade: "A", score: 11, date: I
3   { restaurant_id: 1, name: 1, grades: 1 }
4 };
5
```

To the right of the code editor is an "Output" window. It shows the command being run: "mycompiler_mongodb> ...". Below that, it shows the response: "mycompiler_mongodb> ...". At the bottom of the output window, it says "[Execution complete with exit code 0]".

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

QUERY:

```
db.restaurants.find( {"grades.1.grade": "A", "grades.1.score": 9, "grades.1.date": ISODate("2014-08-11T00:00:00Z") }, { restaurant_id: 1, name: 1, grades: 1 } );
```

OUTPUT:

The screenshot shows the MongoDB shell interface. At the top, there is a header bar with the text "220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K". Below the header are two buttons: "MongoDB" with a dropdown arrow and a help icon. To the right are "Run" and "Save" buttons. The main area is divided into two sections: "Output" on the left and "Logs" on the right. In the "Output" section, the command is displayed in red:
1 db.restaurants.find(
2 { "grades.1.grade": "A", "grades.1.score": 9, "grades.1.date": ISODate("2014-08-11T00:00:00Z") }, { restaurant_id: 1, name: 1, grades: 1 }
3);
4
5

In the "Logs" section, the logs are shown in black:
mycompiler_mongodb> ...
mycompiler_mongodb>
[Execution complete with exit code 0]

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. At the top, there is a header bar with the text "220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K". Below the header are two buttons: "MongoDB" with a dropdown arrow and a help icon. To the right are "Run" and "Save" buttons. The main area is divided into two sections: "Output" on the left and "Logs" on the right. In the "Output" section, the command is displayed in red:
1 db.restaurants.find(
2 { \$and: [
3 { "address.coord.1": { \$gt: 42 } },
4 { "address.coord.1": { \$lte: 52 } }
5] },
6 { _id: 0, restaurant_id: 1, name: 1, address: 1 }
7);
8

In the "Logs" section, the logs are shown in black:
mycompiler_mongodb> ...
mycompiler_mongodb>
[Execution complete with exit code 0]

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. The top bar displays the user's details: "220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K". On the right side of the interface are buttons for "Ctrl+Enter", "Run", and "Save". In the main area, the query `db.restaurants.find({}, { _id: 0 }).sort({ name: 1 })` is entered in the command line. The output window shows the results of the query: "mycompiler_mongodb> mycompiler_mongodb> [Execution complete with exit code 0]".

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. The top bar displays the user's details: "220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K". On the right side of the interface are buttons for "Run" and "Save". In the main area, the query `db.restaurants.find({}, { _id: 0 }).sort({ name: -1 })` is entered in the command line. The output window shows the results of the query: "mycompiler_mongodb> mycompiler_mongodb> [Execution complete with exit code 0]".

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

QUERY:

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

OUTPUT:

The screenshot shows a MongoDB query being run on the mycompiler.io platform. The query is: `db.restaurants.find({}, { _id: 0 }).sort({ cuisine: 1, borough: -1 })`. The output window shows the command prompt: `mycompiler_mongodb>`, followed by three blank lines, and the message: `[Execution complete with exit code 0]`. The URL `https://www.mycompiler.io` is visible at the bottom left.

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 MongoDB query being run on the mycompiler.io platform. The query is: `db.restaurants.find({ "address.street": { $exists: true, $ne: "" } })`. The output window shows the command prompt: `mycompiler_mongodb>`, followed by three blank lines, and the message: `[Execution complete with exit code 0]`. The URL `https://www.mycompiler.io` is visible at the bottom left.

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. The top bar includes a dropdown for 'MongoDB' and a save icon. On the right are 'Run' and 'Save' buttons. The code input area contains the following command:

```
1 db.restaurants.find({ "address.coord": { $elemMatch: { $type: "double" } } })  
2 |
```

The output window shows the results of the command:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. The top bar includes a dropdown for 'MongoDB' and a save icon. On the right are 'Run' and 'Save' buttons. The code input area contains the following command:

```
1 db.restaurants.find(  
2   { "grades.score": { $mod: [7, 0] } },  
3   { restaurant_id: 1, name: 1, grades: 1 }  
4 );  
5 |
```

The output window shows the results of the command:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. At the top, there is a header bar with the user's name '220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K'. Below the header, there are buttons for 'MongoDB ▾', 'Run', and 'Save'. The main area has two sections: 'Output' and 'Input'. The 'Input' section contains the MongoDB command: db.restaurants.find({ name: /mon/i }, { name: 1, borough: 1, "address.coord": 1, cuisine: 1 }). The 'Output' section shows the command being run and the response from the database: 'mycompiler_mongodb> ' followed by '[Execution complete with exit code 0]'. There is also a 'Ctrl+Enter' button at the top right of the output area.

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. At the top, there is a header bar with the user's name '220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K'. Below the header, there are buttons for 'MongoDB ▾', 'Run', and 'Save'. The main area has two sections: 'Output' and 'Input'. The 'Input' section contains the MongoDB command: db.restaurants.find({ name: /^Mad/i }, { name: 1, borough: 1, "address.coord": 1, cuisine: 1 }). The 'Output' section shows the command being run and the response from the database: 'mycompiler_mongodb> ' followed by '[Execution complete with exit code 0]'. There is also a 'Ctrl+Enter' button at the top right of the output area.

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:

```
220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K
```

MongoDB ▾ 

▶ Run

Save

```
1 db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } } })  
2
```

```
mycompiler_mongodb> ... . . . .  
mycompiler_mongodb>
```

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

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

QUERY:

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

OUTPUT:

```
220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K
```

MongoDB ▾ 

▶ Run

Save

```
1 db.restaurants.find(  
2   { "grades": { $elemMatch: { "score": { $lt: 5 } } }, "bo  
3 };  
4
```

Output

```
mycompiler_mongodb> ... . . . .  
mycompiler_mongodb>
```

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

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

QUERY:

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

OUTPUT:

220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K

MongoDB ▾



▶ Run

Save

```
1 db.restaurants.find(  
2   { "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or  
3 };  
4
```

Output

```
mycompiler_mongodb> ... . . . .  
mycompiler_mongodb>
```

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

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:

220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K

MongoDB ▾



▶ Run

Save

```
1 db.restaurants.find(  
2   { "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or  
3 };  
4
```

Output

```
mycompiler_mongodb> ... . . . .  
mycompiler_mongodb>
```

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. At the top, it displays the user's details: 220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K. Below this is a toolbar with a MongoDB icon, dropdown menus, and buttons for Run and Save. The main area contains a code editor with the following command:

```
1 db.restaurants.find(
2   { "grades": { $elemMatch: { "score": { $lt: 5 } } }, "bo
3 );
4
```

To the right of the code editor is an "Output" window titled "mycompiler_mongodb>". It shows the command being run and the response:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. At the top, it displays the user's details: 220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K. Below this is a toolbar with a MongoDB icon, dropdown menus, and buttons for Run and Save. The main area contains a code editor with the following command:

```
1 db.restaurants.find(
2   { "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or
3 );
4
```

To the right of the code editor is an "Output" window titled "mycompiler_mongodb>". It shows the command being run and the response:

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

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

QUERY:

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

OUTPUT:

```
220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K
```

MongoDB ▾ 

Run

Save

```
1 db.restaurants.find(  
2   { "grades": { $elemMatch: { "score": { $lt: 5 } } } }, $or  
3 );  
4
```

Output

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

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

QUERY:

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

OUTPUT:

```
220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K
```

MongoDB ▾ 

Run

Save

```
1 db.restaurants.find(  
2   { $and: [{ "grades.score": 2 }, { "grades.score": 6 }], $or: [{ "borough"  
3 }];  
4
```

Output

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. The top bar displays the user's name and ID: 220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K. Below the bar are two buttons: 'MongoDB' with a dropdown arrow and a small info icon. To the right are 'Run' and 'Save' buttons. The main area has a light gray background with a dark blue header containing the command. The command itself is:

```
1 db.restaurants.find(  
2   { $and: [{ "grades.score": 2 }, { "grades.score": 6 }], $or: [{ "borough"  
3 }];  
4 |
```

To the right of the command is a 'Output' section with a light gray background. It shows the command being run and the response:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. The top bar displays the user's name and ID: 220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K. Below the bar are two buttons: 'MongoDB' with a dropdown arrow and a small info icon. To the right are 'Run' and 'Save' buttons. The main area has a light gray background with a dark blue header containing the command. The command itself is:

```
1 db.restaurants.find(  
2   { $or: [{ "grades.score": 2 }, { "grades.score": 6 }] }  
3 );  
4 |
```

To the right of the command is a 'Output' section with a light gray background. It shows the command being run and the response:

```
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:

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 a MongoDB shell interface. At the top, it displays the user's details: 220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K. Below this is a toolbar with a MongoDB icon, a dropdown menu, and two buttons: 'Run' and 'Save'. The code input area contains the command: db.movies.find({ year: 1893 }). The output panel shows the results of the query execution, which is empty in this case. The status message at the bottom right indicates: [Execution complete with exit code 0].

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

QUERY:

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

OUTPUT:

The screenshot shows a MongoDB shell interface. At the top, it displays the user's details: 220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K. Below this is a toolbar with a MongoDB icon, a dropdown menu, and two buttons: 'Run' and 'Save'. The code input area contains the command: db.movies.find({ runtime: { \$gt: 120 } }). The output panel shows the results of the query execution, which is empty in this case. The status message at the bottom right indicates: [Execution complete with exit code 0].

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

QUERY:

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

OUTPUT:

The screenshot shows a MongoDB shell interface. At the top, there is a header bar with the text "220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K". Below the header are two buttons: "MongoDB" with a dropdown arrow and a small info icon. To the right are "Run" and "Save" buttons. The main area has a light gray background. On the left, a code editor window displays the command: "1 db.movies.find({ genres: 'Short' })". The number "2" is partially visible below it, and "3" is at the bottom. On the right, a "Output" window titled "Output" shows the results of the command. It starts with "mycompiler_mongodb>" followed by "mycompiler_mongodb>". Below that is the message "[Execution complete with exit code 0]".

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

QUERY:

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

OUTPUT:

The screenshot shows a MongoDB shell interface. At the top, there is a header bar with the text "220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K". Below the header are two buttons: "MongoDB" with a dropdown arrow and a small info icon. To the right are "Run" and "Save" buttons. The main area has a light gray background. On the left, a code editor window displays the command: "1 db.movies.find({ directors: 'William K.L. Dickson' })". The number "2" is partially visible below it, and "3" and "4" are at the bottom. On the right, a "Output" window titled "Output" shows the results of the command. It starts with "mycompiler_mongodb>" followed by "mycompiler_mongodb>". Below that is the message "[Execution complete with exit code 0]".

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. At the top, there is a header bar with the text "220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K". Below the header are two buttons: "MongoDB" with a dropdown arrow and a small info icon. To the right are "Run" and "Save" buttons. The main area has a light blue background. On the left, a code editor window displays the command: "1 db.movies.find({ countries: 'USA' })". The number "2" is highlighted in blue, indicating the current line of execution. On the right, a "Output" window shows the results of the command. It starts with "mycompiler_mongodb>" followed by "[Execution complete with exit code 0]".

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. At the top, there is a header bar with the text "220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K". Below the header are two buttons: "MongoDB" with a dropdown arrow and a small info icon. To the right are "Run" and "Save" buttons. The "Ctrl+Enter" keybinding is also visible. The main area has a light blue background. On the left, a code editor window displays the command: "1 db.movies.find({ rated: 'UNRATED' })". The number "2" is highlighted in blue. On the right, a "Output" window shows the results of the command. It starts with "mycompiler_mongodb>" followed by "[Execution complete with exit code 0]".

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB Compass interface. At the top, there are fields for 'MongoDB' and 'Run'. Below the code input, the output window displays the results of the query. The results are empty, indicating no movies were found that meet the criteria.

```
220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K
```

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

```
2
```

```
3
```

```
4
```

```
5
```

```
6
```

```
7
```

Output

```
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>
```

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

<https://www.mycompiler.io>

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB Compass interface. At the top, there are fields for 'MongoDB' and 'Run'. Below the code input, the output window displays the results of the query. The results are empty, indicating no movies were found that meet the criteria.

```
220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K
```

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

```
2
```

Output

```
mycompiler_mongodb>
mycompiler_mongodb>
```

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. At the top, there is a header bar with the text "220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K". Below the header are two buttons: "MongoDB" with a dropdown arrow and "Run" with a play icon. To the right of the "Run" button is a "Save" button with a disk icon. The main area is divided into two sections: "Output" on the left and "Output" on the right. The left "Output" section contains the command: `1 db.movies.find({ 'tomatoes.viewer.rating': { $gt: 4 } })`. The right "Output" section shows the results of the command: `mycompiler_mongodb>`, followed by four blank lines, and then `[Execution complete with exit code 0]`.

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. At the top, there is a header bar with the text "220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K". Below the header are two buttons: "MongoDB" with a dropdown arrow and "Run" with a play icon. To the right of the "Run" button is a "Save" button with a disk icon. The main area is divided into two sections: "Output" on the left and "Output" on the right. The left "Output" section contains the command: `1 db.movies.find({ 'awards.wins': { $gt: 0 } })`. The right "Output" section shows the results of the command: `mycompiler_mongodb>`, followed by four blank lines, and then `[Execution complete with exit code 0]`.

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. The top bar displays the user's name and ID: 220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K. Below the bar are two buttons: 'MongoDB ▾' and 'Run'. The main area contains the following code:

```
1 db.movies.find(
2   { 'awards.nominations': { $gt: 0 } },
3   {
4     title: 1,
5     languages: 1,
6     released: 1,
7     directors: 1,
8     writers: 1,
9     awards: 1,
10    year: 1,
11    genres: 1,
12    runtime: 1,
13    cast: 1,
14    countries: 1
15  }
16 )
17
```

The 'Output' panel on the right shows the results of the query:

```
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. The top bar displays the user's name and ID: 220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K. Below the bar are two buttons: 'MongoDB ▾' and 'Run'. The main area contains the following code:

```
1 db.movies.find(
2   { cast: 'Charles Kayser' },
3   {
4     title: 1,
5     languages: 1,
6     released: 1,
7     directors: 1,
8     writers: 1,
9     awards: 1,
10    year: 1,
11    genres: 1,
12    runtime: 1,
13    cast: 1,
14    countries: 1
15  }
16 )
17
```

The 'Output' panel on the right shows the results of the query:

```
mycompiler_mongodb> ....
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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

QUERY:

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

OUTPUT:

220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K

MongoDB ▾ ⓘRunSave

```
1 db.movies.find(
2   { released: ISODate("1893-05-09T00:00:00.000Z") },
3   {
4     title: 1,
5     languages: 1,
6     released: 1,
7     directors: 1,
8     writers: 1,
9     countries: 1
10    }
11  )
12
13
14
15
16
```

Output

```
mycompiler_mongodb> . . . . .
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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

QUERY:

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

OUTPUT:

220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K

The screenshot shows the MongoDB Compass interface. In the left panel, a query is written in the MongoDB shell syntax:

```
1 db.movies.find(  
2   { title: /scene/i },  
3   {  
4     title: 1,  
5     languages: 1,  
6     released: 1,  
7     directors: 1,  
8     writers: 1,  
9     countries: 1  
10    }  
11 )  
12  
13  
14  
15  
16
```

In the right panel, the results of the query are displayed under the "Output" tab. The results show multiple documents from the "movies" collection, each containing the specified fields (title, languages, released, directors, writers, and countries) with a value of 1, indicating that all matching documents are being returned.

Run Save

Output

```
mycompiler_mongodb> ...  
mycompiler_mongodb>  
mycompiler_mongodb>  
mycompiler_mongodb>  
mycompiler_mongodb>  
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: