

# 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 department(dept number(6),name varchar(20));

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, Gallery, and a search bar. The user is logged in as agish raj with session ID AR. The SQL Workshop tab is selected. In the main area, the schema is set to WKSP\_ARKHAM16. The query editor contains the command: `create table department(dept number(7),name varchar(25));`. Below the editor, the results pane shows the output: "Table created." and "0.03 seconds". Other tabs in the results pane include Explain, Describe, Saved SQL, and History.

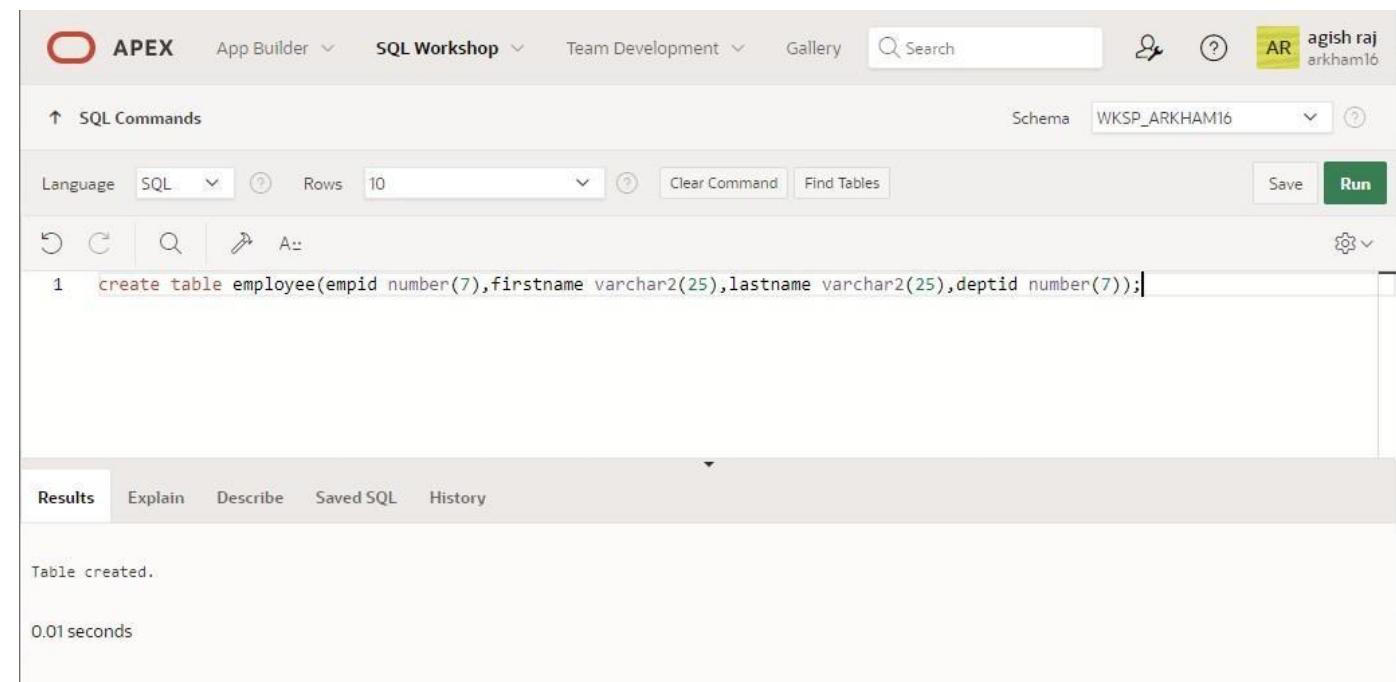
2. Create the EMP table based on the following instance chart. Confirm that the table is created.

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 employee(empid number(7),Firstname Varchar(25),Lastname Varchar(25),Email varChar(25),deptid number(7));
```

#### OUTPUT:



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

```
1 create table employee(empid number(7),firstname varchar2(25),lastname varchar2(25),deptid number(7));
```

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

Table created.  
0.01 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 employee modify (lastname varchar(50));
```

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A single line of SQL code is entered: 'alter table employee modify (lastname varchar2(50));'. Below the code, the 'Results' tab is active, showing the output: 'Table altered.' and '0.06 seconds'. The schema dropdown at the top right is set to 'WKSP\_ARKHAM16'.

4. Create the EMPLOYEES2 table based on the structure of EMPLOYEES table. Include Only the Employee\_id, First\_name, Last\_name, Salary and Dept\_id coloumns. Name the columns Id, First\_name, Last\_name, salary and Dept\_id respectively.

**QUERY:**

```
create table employee2(empid number(6),Firstname Varchar(20),Lastname  
Varchar(20),Salary number(8,2),deptid number(4));
```

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A single line of SQL code is entered: 'create table employee2(empid number(6),firstname varchar2(20),lastname varchar2(25),salary number(8,2),deptid number(4));'. Below the code, the 'Results' tab is active, showing the output: 'Table created.' and '0.03 seconds'. The schema dropdown at the top right is set to 'WKSP\_ARKHAM16'.

5. Drop the EMP table.

**QUERY:**

**Drop table employee;**

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A single line of SQL code is entered: '1 drop table employee;'. Below the code, the results pane displays the message 'Table dropped.' and a execution time of '0.08 seconds'.

```
drop table employee;
```

Table dropped.  
0.08 seconds

6. Rename the EMPLOYEES2 table as EMP.

**QUERY:**

**alter table employee2 rename to employee;**

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A single line of SQL code is entered: '1 alter table employee2 rename to employee;'. Below the code, the results pane displays the message 'Table altered.' and a execution time of '0.06 seconds'.

```
alter table employee2 rename to employee;
```

Table altered.  
0.06 seconds

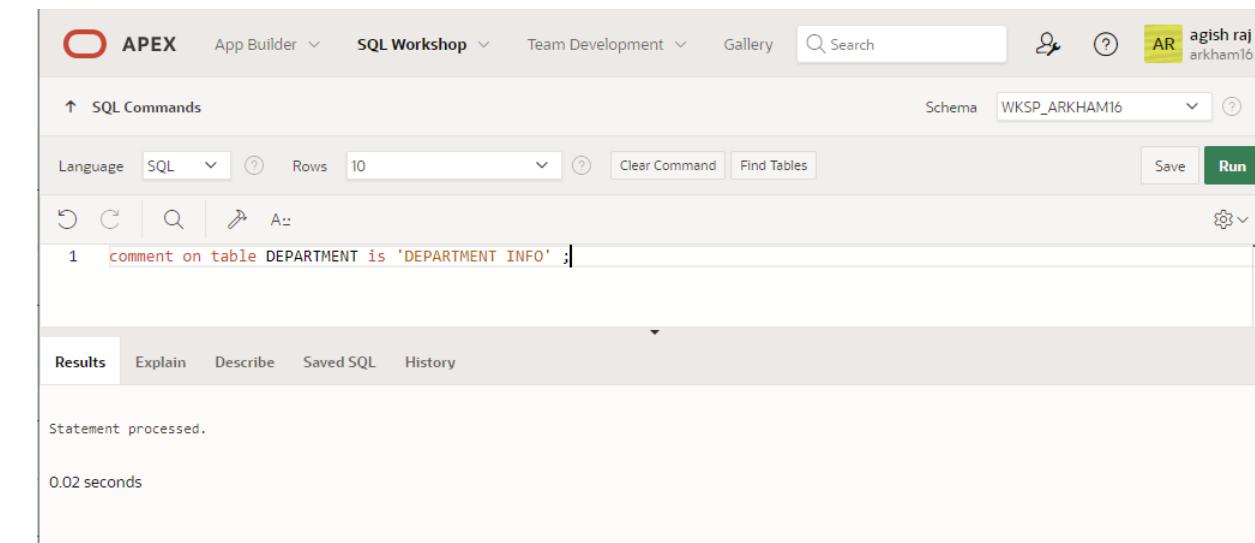
7. Add a comment on DEPT and EMP tables. Confirm the modification by describing the table.

**QUERY:**

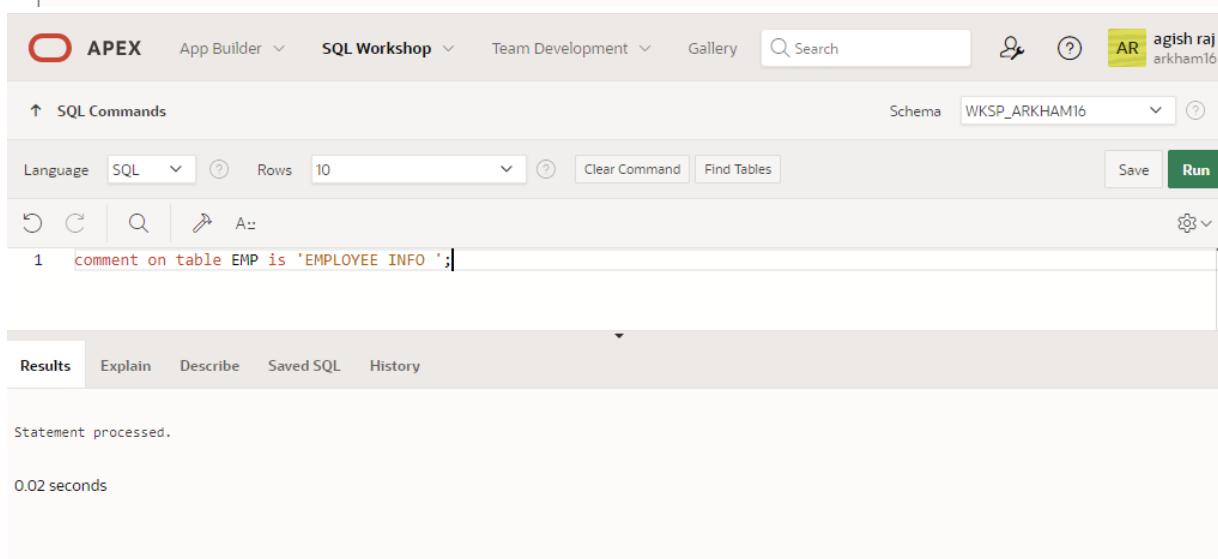
**comment on table DEPARTMENT is 'DEPARTMENT INFO'**

**comment on table EMP is 'EMPLOYEE INFO '**

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (which is selected), 'Team Development', 'Gallery', a search bar, and user information 'agish raj arkham16'. The main area is titled 'SQL Commands'. It has a toolbar with icons for undo, redo, search, and execute. Below the toolbar, the command 'comment on table DEPARTMENT is 'DEPARTMENT INFO'' is entered in the text area. The results section at the bottom shows the message 'Statement processed.' and a time of '0.02 seconds'.



This screenshot shows the Oracle SQL Workshop interface again. The top navigation bar is identical to the previous one. The main area is titled 'SQL Commands'. The command 'comment on table EMP is 'EMPLOYEE INFO '' is entered in the text area. The results section at the bottom shows the message 'Statement processed.' and a time of '0.02 seconds'.

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

**QUERY:**

```
alter table employee drop column  
Firstname; describe employee;
```

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A single line of SQL code is entered: 'alter table employee drop column firstname;'. Below the command, the output shows the message 'Table altered.' and a execution time of '0.07 seconds'.

```
1 alter table employee drop column firstname;
```

Table altered.  
0.07 seconds

The screenshot shows the Oracle SQL Workshop interface with the 'Describe' tab selected. The command entered is 'describe employee;'. The results pane displays the structure of the EMPLOYEE table, including columns: EMPID, LASTNAME, SALARY, and DEPTID. The table has four columns: EMPID (NUMBER), LASTNAME (VARCHAR2), SALARY (NUMBER), and DEPTID (NUMBER). The primary key is EMPID, and both LASTNAME and DEPTID are nullable.

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMPLOYEE	EMPID	NUMBER	-	6	0	-	✓	-	-
	LASTNAME	VARCHAR2	25	-	-	-	✓	-	-
	SALARY	NUMBER	-	8	2	-	✓	-	-
	DEPTID	NUMBER	-	4	0	-	✓	-	-

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

**RESULT:**

# MANIPULATING DATA

EX.NO.2

DATE:

## Find the Solution for the following:

1. Create MY\_EMPLOYEE table with the following structure

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

## QUERY:

Create table myemployee (empid Number(4) not null ,Lastname varchar(25), Firstname varchar(25),Userid varchar2(25), Salary Number(9,2));

## OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, and a search bar. The schema dropdown is set to WKSP\_ARKHAM16. The main workspace displays the following SQL command:

```
1 create table myemployee(empid number(4) not null,lastname varchar2(25),firstname varchar2(25),
2 userid varchar(25),salary number(9,2));
```

The results tab shows the output of the query:

```
Table created.
```

Execution time: 0.04 seconds.

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

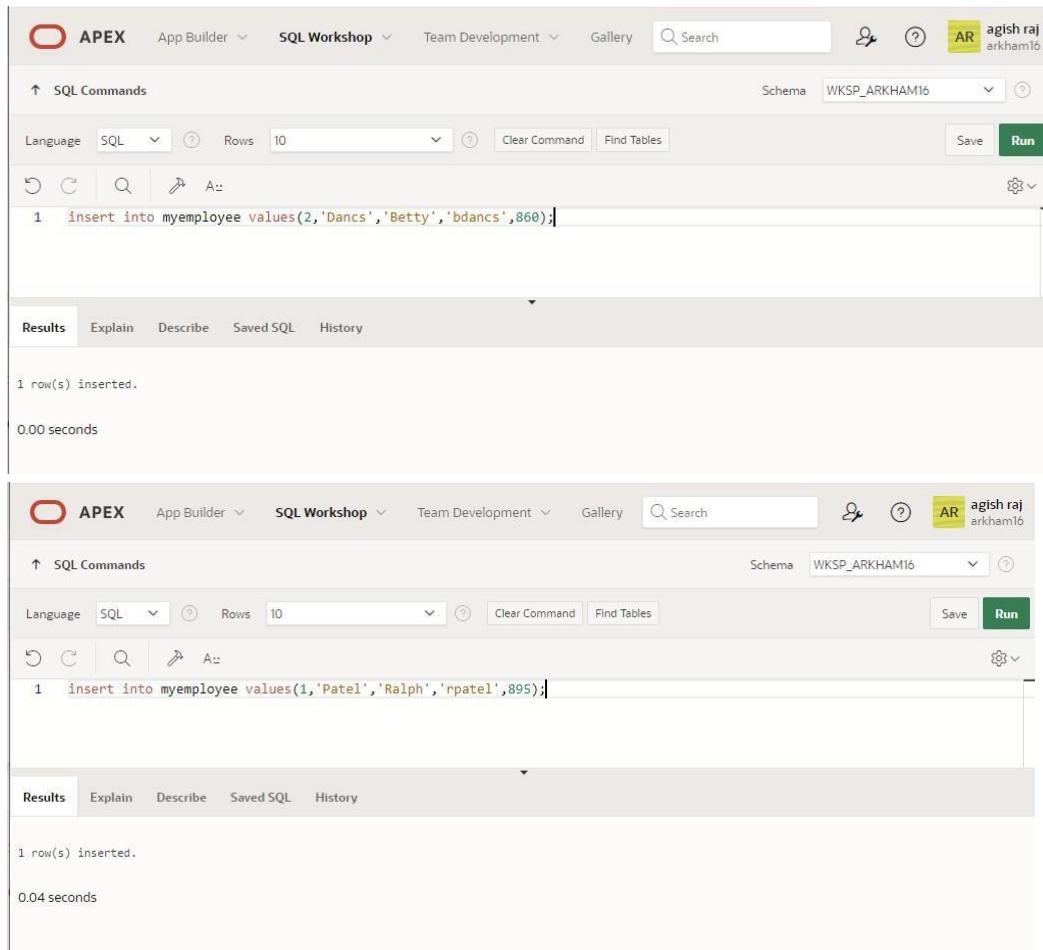
ID	Last_name	First_name	Userid	salary
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	Cnewman	750
5	Ropebur	Audrey	aropebur	1550

#### QUERY:

```
INSERT INTO Myemployee values(1, 'patel','Ralph','rpatel',895);
```

```
INSERT INTO Myemployee values(2, 'Dancs','Betty','bdancs',860);
```

#### OUTPUT:



The screenshot shows two separate sessions in the Oracle SQL Workshop interface. Both sessions are connected to the schema 'WKSP\_ARKHAM16'.

**Session 1 (Top):**

- SQL Command: `insert into myemployee values(2, 'Dancs','Betty','bdancs',860);`
- Results: 1 row(s) inserted.
- Time: 0.00 seconds

**Session 2 (Bottom):**

- SQL Command: `insert into myemployee values(1, 'Patel','Ralph','rpatel',895);`
- Results: 1 row(s) inserted.
- Time: 0.04 seconds

3. Display the table with values.

**QUERY:**

**SELECT \* from**

**myemployee;**

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and a user profile 'agish raj arkham16'. The SQL Commands panel shows the query 'select \* from myemployee;'. The Results panel displays the following data:

EMPID	LASTNAME	FIRSTNAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860

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

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\_EMPLO values(3, 'Biri', 'Ben', 'bbiri', 1100);**

**INSERT INTO MY\_EMPLO values(4, 'Newman', 'Chad', 'Cnewman', 750);**

**OUTPUT**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and a user profile 'agish raj arkham16'. The SQL Commands panel shows the query 'insert into myemployee values(3, 'Biri', 'Ben', 'bbiri', 1100);'. The Results panel displays the message '1 row(s) inserted.' and '0.01 seconds'.

5. Make the data additions permanent.

**QUERY:**

**SELECT \* from myemployee;**

**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, along with a search bar and user profile information. The main workspace is titled "SQL Commands". The query "select \* from myemployee;" is entered in the command input field. The results tab is selected, displaying a table with four rows of data. The columns are labeled EMPID, LASTNAME, FIRSTNAME, USERID, and SALARY. The data is as follows:

EMPID	LASTNAME	FIRSTNAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
3	Biri	Ben	bbiri	1100
2	Dancs	Betty	bdancs	860
4	Newman	Chad	Cnewman	750

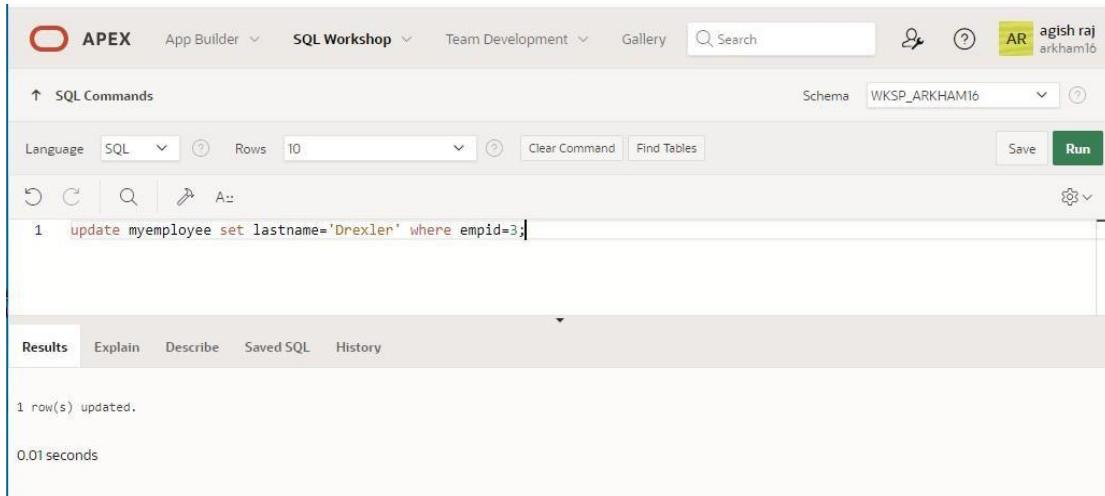
Below the table, a message indicates "4 rows returned in 0.00 seconds" and provides a "Download" link.

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

**QUERY:**

**UPDATE myemployee SET LastName='Drexler' where id=3;**

**OUTPUT:**



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

```
1 update myemployee set lastname='Drexler' where empid=3;
```

Below the command, the results section shows:

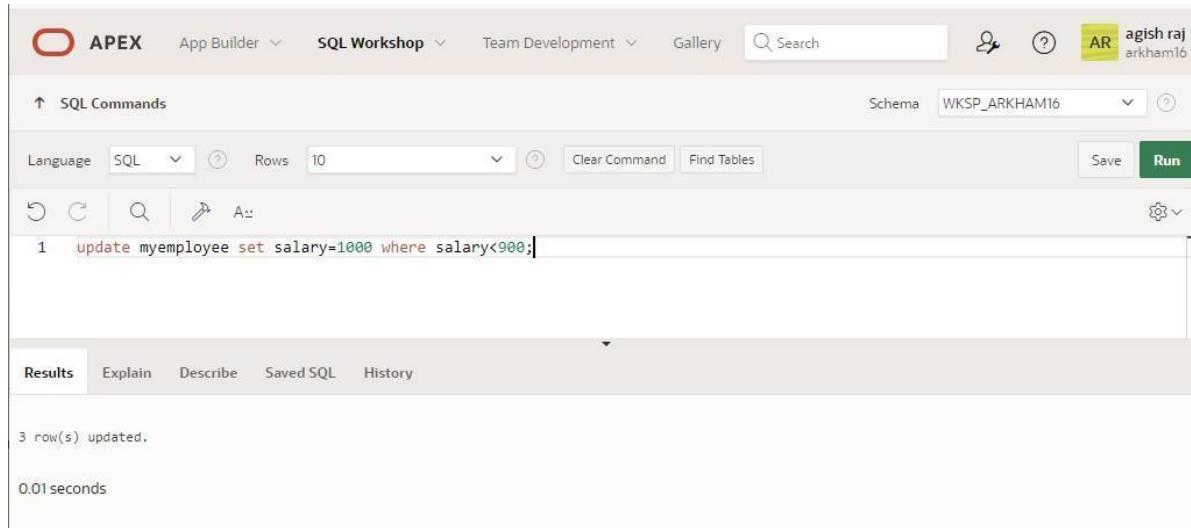
1 row(s) updated.  
0.01 seconds

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

**QUERY:**

**UPDATE myemployee SET Salary=1000 where Salary<900;**

**OUTPUT:**



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

```
1 update myemployee set salary=1000 where salary<900;
```

Below the command, the results section shows:

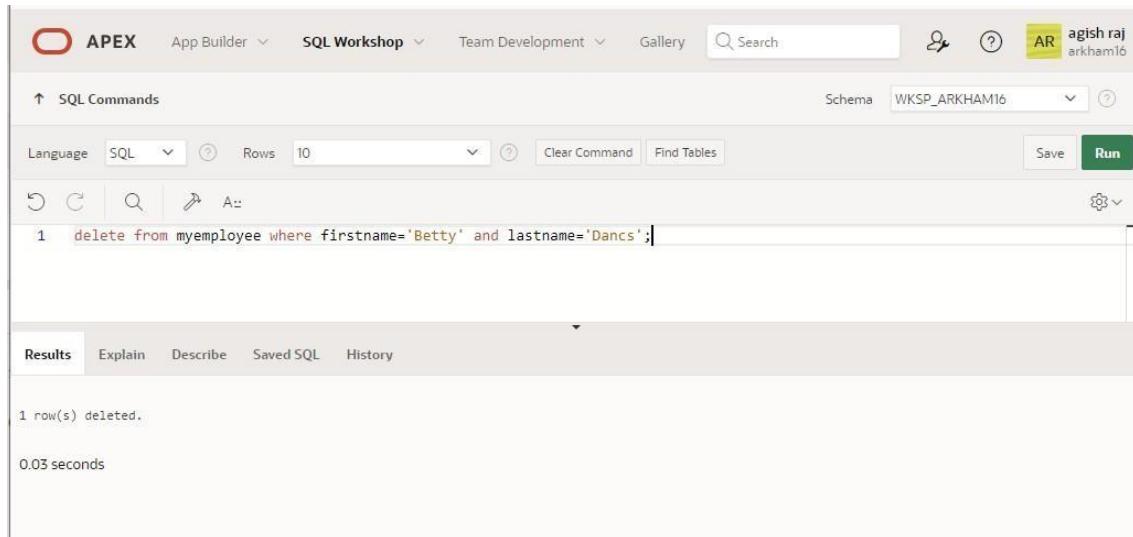
3 row(s) updated.  
0.01 seconds

8. Delete Betty dancs from MY\_EMPLOYEE table.

**QUERY:**

**DELETE from myemployee where Firstname ='Betty' and  
lastName='Dancs';**

**OUTPUT:**



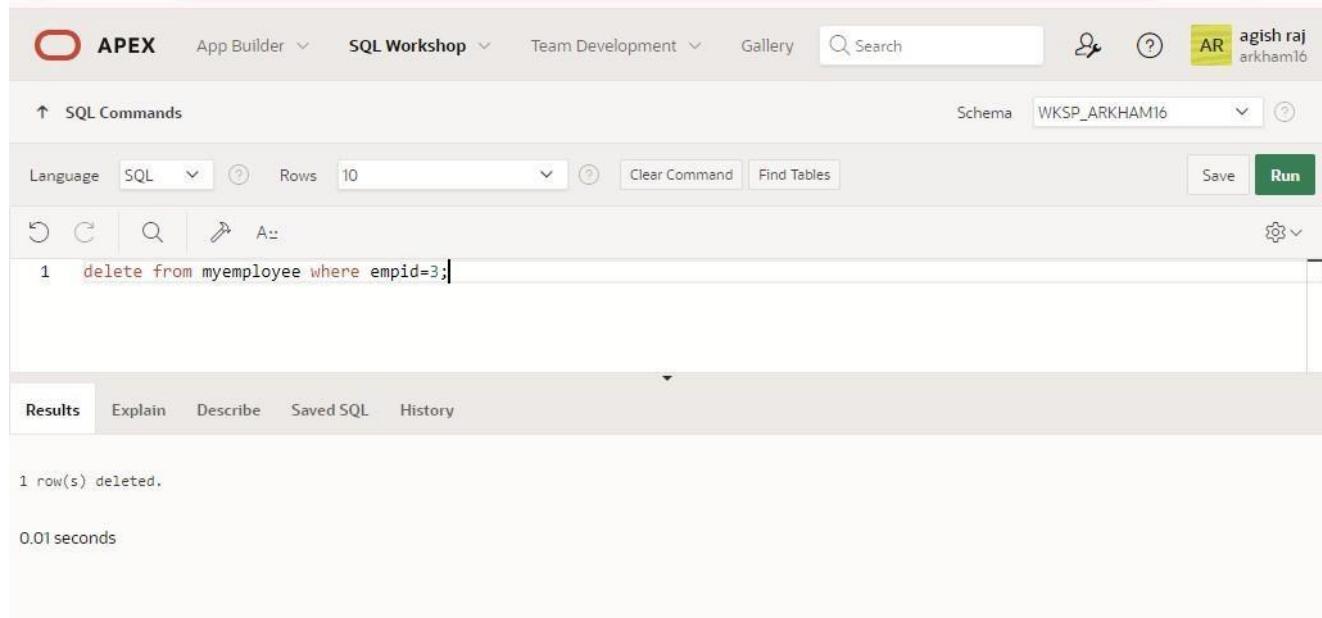
The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A single line of SQL code is entered: 'delete from myemployee where firstname='Betty' and lastname='Dancs';'. Below the command, the results section displays the output: '1 row(s) deleted.' and '0.03 seconds'.

9. Empty the fourth row of the emp table.

**QUERY:**

**Delete from MY\_EMP where id=4;**

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A single line of SQL code is entered: 'delete from myemployee where empid=3;'. Below the command, the results section displays the output: '1 row(s) deleted.' and '0.01 seconds'.

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

**RESULT:**

# INCLUDING CONSTRAINTS

EX.NO.3

DATE:

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

**QUERY:**

```
Create table Emp(employeeid number(6), lastname varchar(25), firstname varchar(25), Salary number(8,2),CONSTRAINT my_emp_id_pk PRIMARY KEY(employeeid));
```

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, the tabs 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery' are visible. A search bar is present, along with user information 'agish raj arkham16'. The main area is titled 'SQL Commands' with a 'Schema' dropdown set to 'WKSP\_ARKHAM16'. The language is set to 'SQL'. The code entered is:

```
1 create table emp(employeeid number(6),firstname varchar2(25),lastname varchar2(25),salary number(8,2),
2 constraint my_emp_id_pk primary key(employeeid));
```

Below the code, the results section shows the message 'Table created.' and a execution time of '0.10 seconds'.

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

**QUERY:**

```
Create table dept(deptid number(6),firstname varchar(25),lastname varchar(25),Salary number(8,2),CONSTRAINT my_dept_id_pk PRIMARY KEY(deptid));
```

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar and schema selection are identical to the previous screenshot. The main area is titled 'SQL Commands' with a 'Schema' dropdown set to 'WKSP\_ARKHAM16'. The language is set to 'SQL'. The code entered is:

```
1 create table dept(deptid number(6),firstname varchar2(25),lastname varchar2(25),salary number(8,2),
2 constraint my_dept_id_pk primary key(deptid));
```

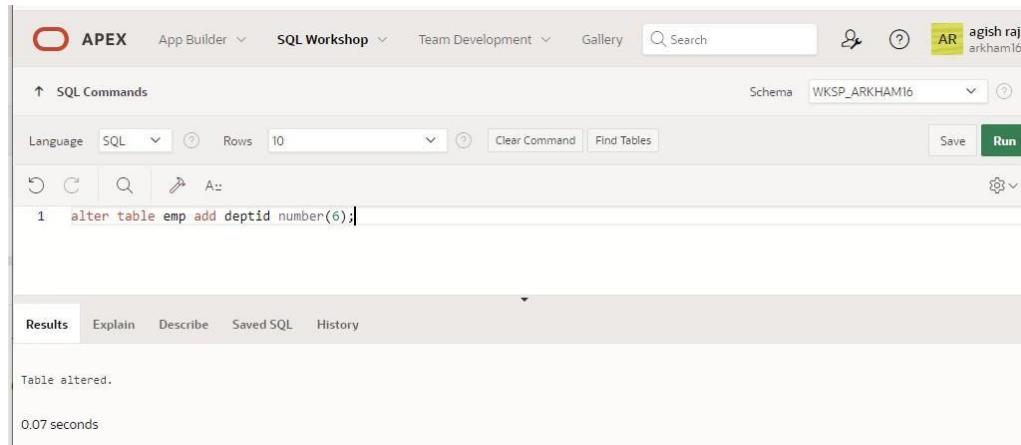
Below the code, the results section shows the message 'Table created.' and a execution time of '0.05 seconds'.

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 DEPTID number(6);
alter table emp ADD CONSTRAINT my_emp_dept_id_fk FOREIGN KEY(dept1_id)
REFERENCES Employee1(emp_id);
```

**OUTPUT:**

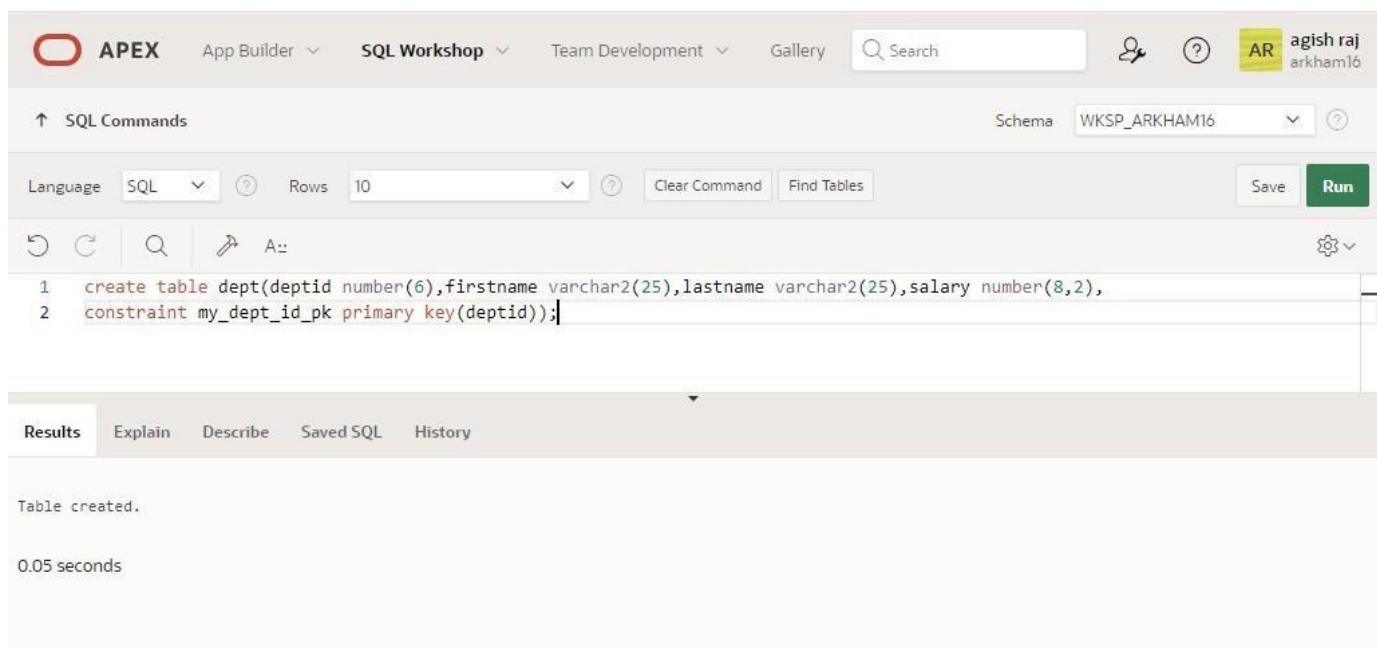


The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the following command is entered:

```
1 alter table emp add deptid number(6);
```

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

```
Table altered.  
0.07 seconds
```



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

```
1 create table dept(deptid number(6),firstname varchar2(25),lastname varchar2(25),salary number(8,2),
2 constraint my_dept_id_pk primary key(deptid));
```

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

```
Table created.  
0.05 seconds
```

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

**QUERY:**

```
alter table emp ADD COMMISSION number(2,2) check (COMMISSION > 0);
```

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, and a search bar. The user is signed in as agish raj (arkham16). The SQL Commands tab is active, showing the following SQL code:

```
1 alter table emp add commission number(2,2) check (commission>0);
2 |
```

The Run button is highlighted in green at the top right of the editor area. Below the editor, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, displaying the output:

Table altered.  
0.06 seconds

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. Identify the Errors

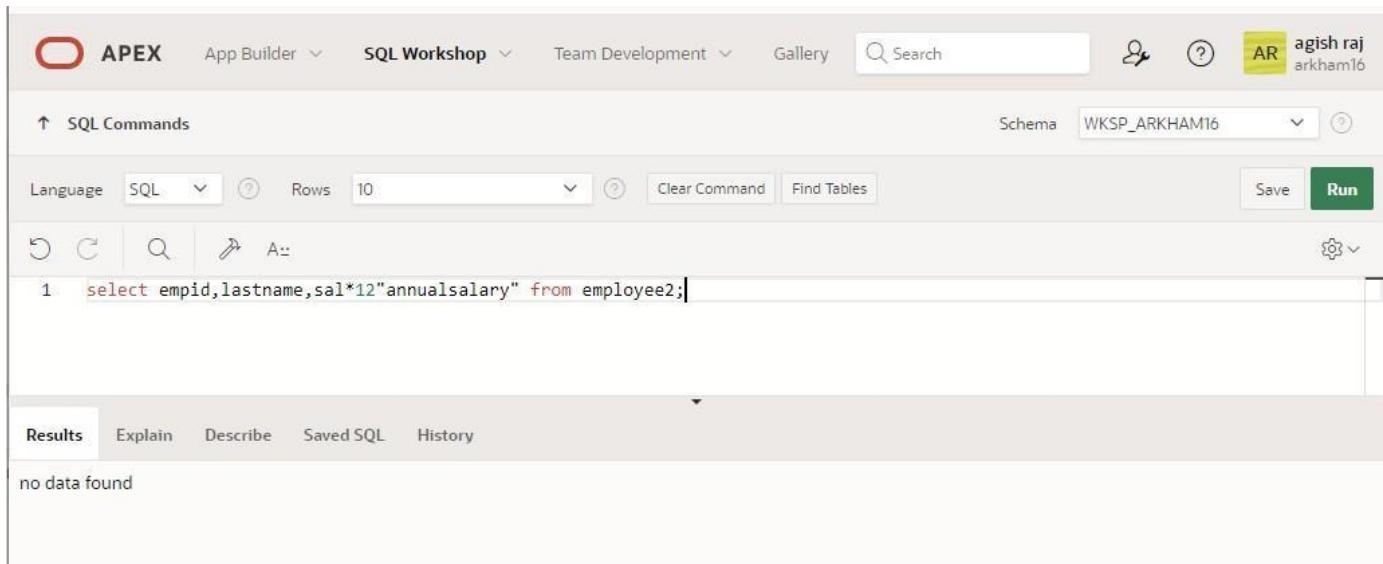
```
SELECT employee_id, last_name  
sal*12 ANNUAL SALARY
```

FROM employees;

**Correct Query:**

```
Select emp_id , last_name,Salary*12 as ANNUAL_SALARY from Employee1;
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and user information 'agish raj arkham16'. Below the navigation is a toolbar with icons for undo, redo, search, and refresh, followed by a dropdown menu. The main area is titled 'SQL Commands' and contains a text input field with the following SQL code:

```
1 select empid,lastname,sal*12"annualsalary" from employee2;
```

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

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. In the SQL Commands pane, the command `desc dept;` is entered. Below the results pane, the "Describe" tab is selected. The results show the structure of the DEPT table:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPT	DEPTID	NUMBER	-	6	0	1	-	-	-
	FIRSTNAME	VARCHAR2	25	-	-	-	✓	-	-
	LASTNAME	VARCHAR2	25	-	-	-	✓	-	-
	SALARY	NUMBER	-	8	2	-	✓	-	-

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 empid, lastname, jobcode, hiredate, employeenum from Employee2;**

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, Gallery, a search bar, and user information (agish raj, arkhamb16). The main area is titled "SQL Commands". The schema is set to "WKSP\_ARKHAMB16". The query entered is:

```
1 select empid, lastname, jobcode, hiredate, employeenum from employee2;
2 |
```

The results section displays the following data:

EMPID	LASTNAME	JOBCODE	HIREDATE	EMPLOYEENUM
453	bala	h64	09/05/2020	123
234	raj	a346	08/09/2023	876
321	john	g645	03/09/2022	654

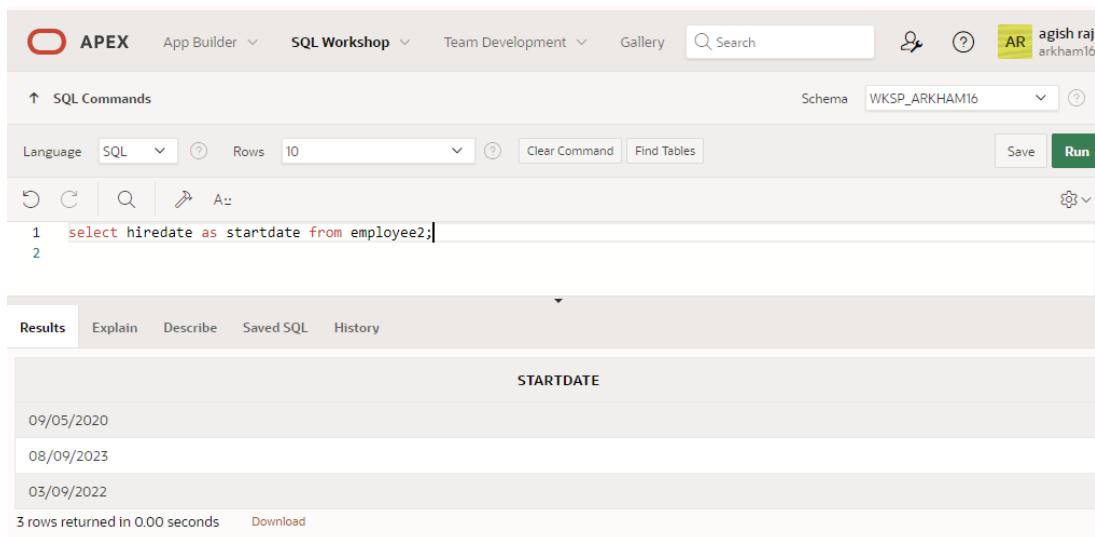
Below the results, it says "3 rows returned in 0.01 seconds" and there is a "Download" link.

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. In the SQL Commands pane, the query `select hiredate as startdate from employee2;` is entered. The Results pane displays the output:

STARTDATE
09/05/2020
08/09/2023
03/09/2022

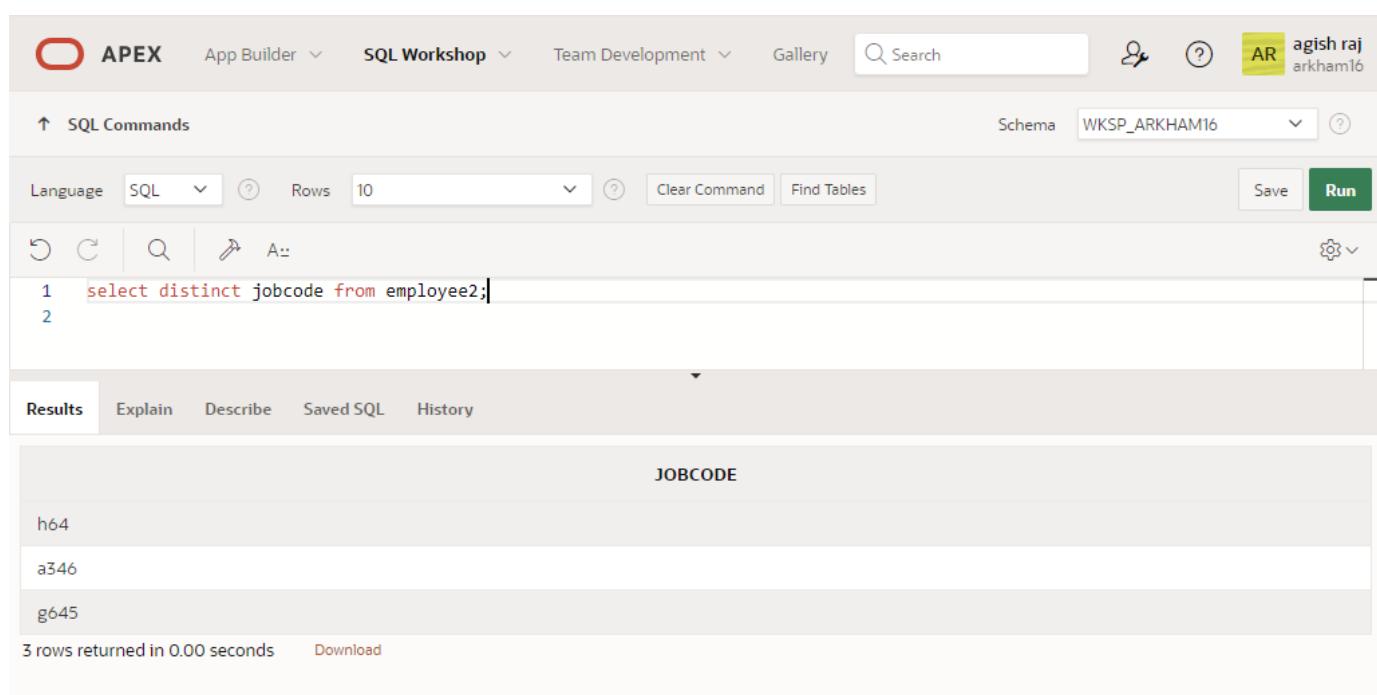
3 rows returned in 0.00 seconds

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

**QUERY:**

**SELECT DISTINCT jobcode from Employee2;**

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the query `select distinct jobcode from employee2;` is entered. The Results pane displays the output:

JOBCODE
h64
a346
g645

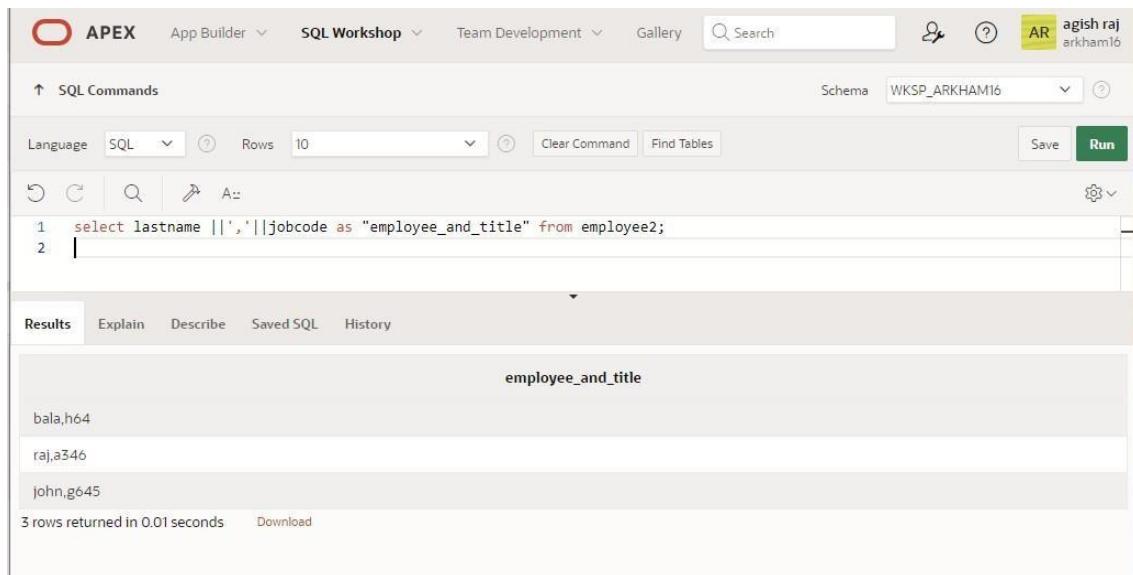
3 rows returned in 0.00 seconds

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||','||jobcode AS "EMPLOYEE and TITLE" FROM Employee2;
```

**OUTPUT:**



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

```
1 select lastname ||','||jobcode as "employee_and_title" from employee2;
2
```

In the Results pane, the output is displayed under the heading "employee\_and\_title". The results are:

employee_and_title
bala,h64
raj,a346
john,g645

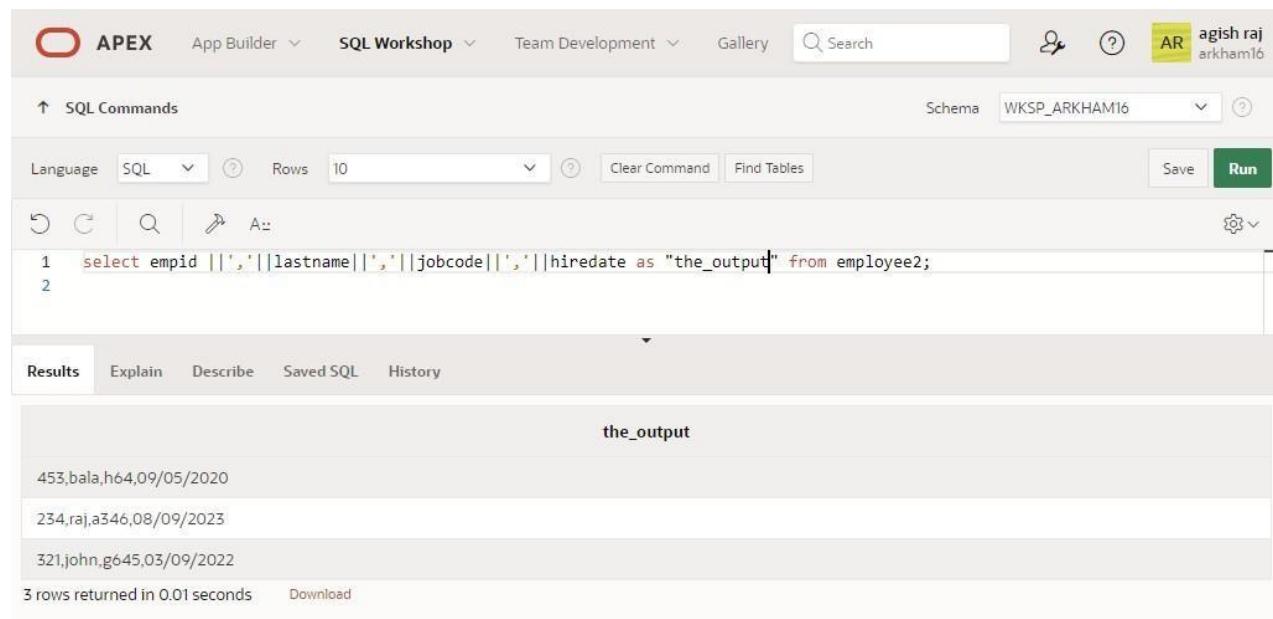
3 rows returned in 0.01 seconds

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 empid||','|| lastname||','||jobcode||','||Salary||','|| Hiredate AS "THE OUTPUT" FROM Employee2;
```

**OUTPUT:**



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

```
1 select empid ||','|| lastname||','||jobcode||','||Salary||','|| Hiredate as "the_output" from employee2;
2
```

In the Results pane, the output is displayed under the heading "the\_output". The results are:

the_output
453,bala,h64,09/05/2020
234,raj,a346,08/09/2023
321,john,g645,03/09/2022

3 rows returned in 0.01 seconds

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

**RESULT:**

## **RESTRICTING AND SORTING DATA**

**EX.NO:5**

**DATE:**

Find the Solution for the following:

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

**QUERY:**

**select lastname,salary from employe where salary>12000;**

**OUTPUT:**

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

```
1 select lastname,salary from employe where salary>12000;
2 |
```

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

LASTNAME	SALARY
kumar	50000
ethan	60000

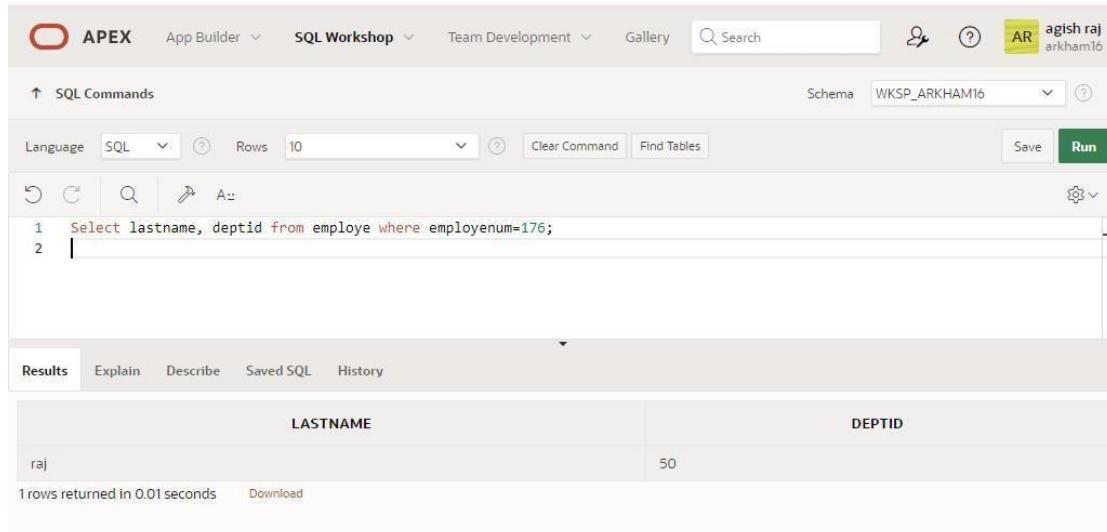
Below the table, it says "2 rows returned in 0.01 seconds".

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

### QUERY:

**Select lastname, deptid from employe where employenum=176;**

### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and user information 'agish raj arkham16'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_ARKHAM16'. Below this are buttons for Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run. The command input field contains the query: '1 Select lastname, deptid from employe where employenum=176;'. The results tab is selected, showing a single row with columns 'LASTNAME' and 'DEPTID'. The data is: LASTNAME = 'raj' and DEPTID = '50'. A note at the bottom says '1 rows returned in 0.01 seconds'.

LASTNAME	DEPTID
raj	50

1 rows returned in 0.01 seconds    Download

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

### QUERY:

**select lastname , salary from employe where salary not between 5000 and 12000;**

### OUTPUT:

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

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

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

LASTNAME	SALARY
kumar	50000
ethan	60000

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

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

### QUERY:

**Select lastname, jobid, hiredate,startdate from employe where hiredate between 'February 20,1998' and 'May 1,1998' order by startdate;**

### OUTPUT:

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

```
1 Select lastname, jobid, hiredate,startdate  from employe where hiredate between 'February 20,1998'
2 and 'May 1,1998' order by startdate;
3
4
5
```

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

LASTNAME	JOBID	HIREDATE	STARTDATE
raj	srg34	03/20/1998	04/01/1998

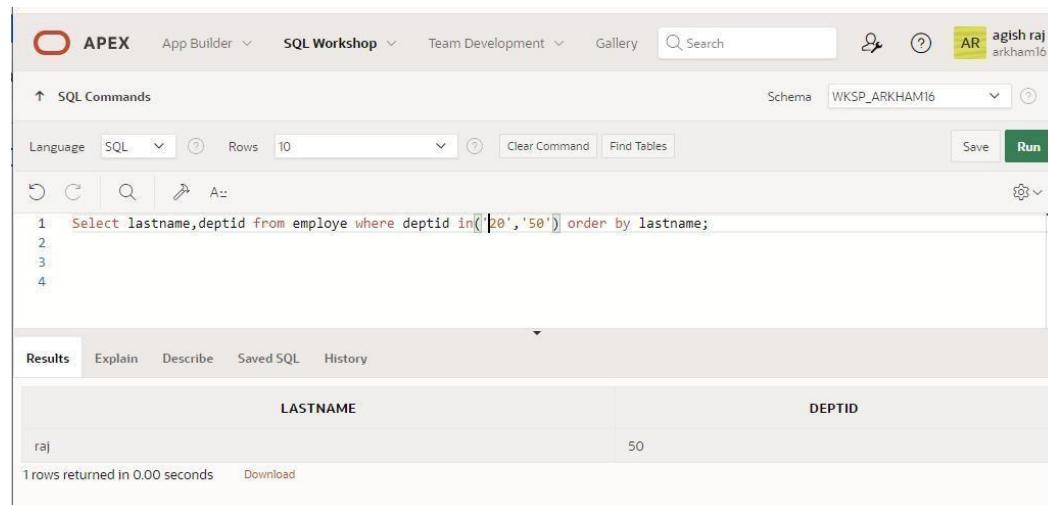
Below the table, it says "1 rows returned in 0.01 seconds".

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

### QUERY:

Select lastname,deptid from employe where deptid in('20','50') order by lastname;

### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and user information 'agish raj' and 'arkham16'. Below the navigation is a toolbar with icons for Undo, Redo, Search, and others. The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the schema is set to 'WKSP\_ARKHAM16'. The SQL command entered is:

```
1 Select lastname,deptid from employe where deptid in('20','50') order by lastname;
2
3
4
```

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

LASTNAME	DEPTID
raj	50

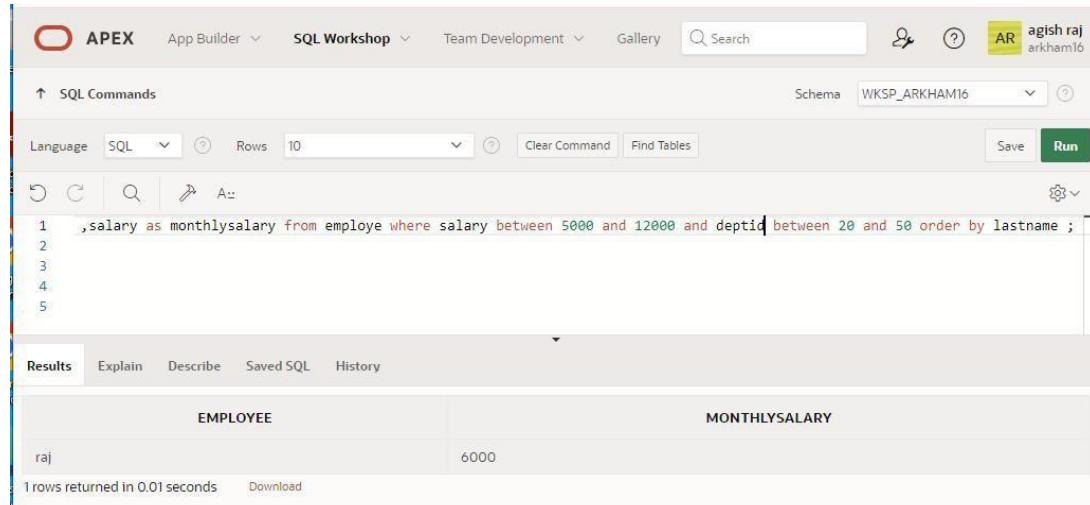
Below the table, it says '1 rows returned in 0.00 seconds' and there is a 'Download' link.

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

### QUERY:

Select lastname as employee ,salary as monthly salary from employe where salary between 5000 and 12000 and dept between 20 and 50 order by lastname ;

### OUTPUT:



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

```
1 ,salary as monthlysalary from employe where salary between 5000 and 12000 and deptid between 20 and 50 order by lastname ;
```

The results table shows one row:

EMPLOYEE	MONTHLYSALARY
raj	6000

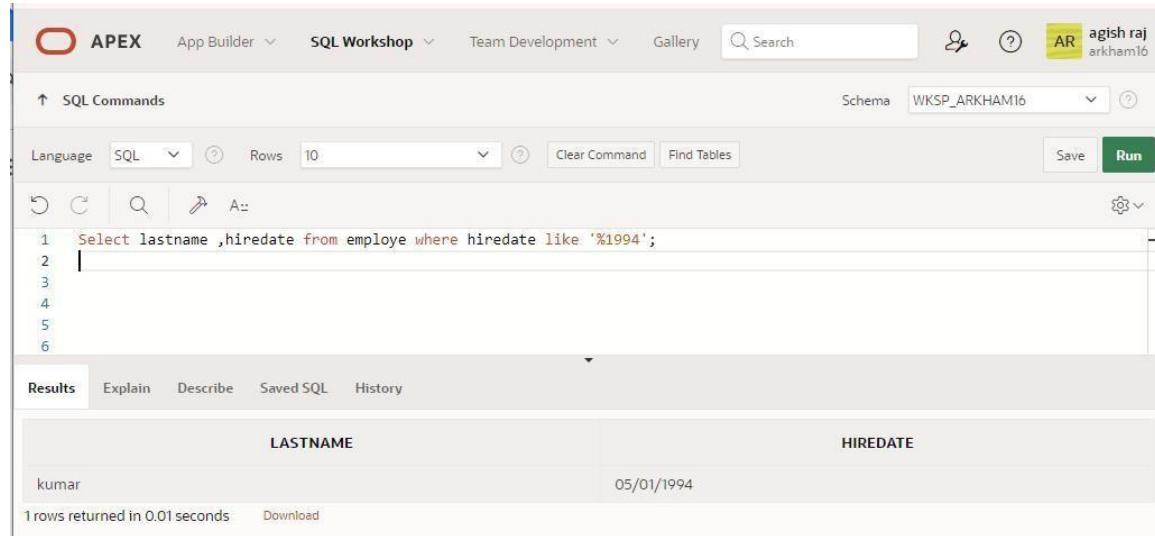
1 rows returned in 0.01 seconds

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

### QUERY:

Select lastname ,hiredate from employe where hiredate like '%1994';

### OUTPUT:



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

```
1 Select lastname ,hiredate from employe where hiredate like '%1994';
```

The results table shows one row:

LASTNAME	HIREDATE
kumar	05/01/1994

1 rows returned in 0.01 seconds

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

### QUERY:

Select lastname ,jobtitle from employe where manager is null;

### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is: `1 Select lastname ,jobtitle from employe where manager is null;`. The results table has two rows: one for 'kumar' with 'salesrep' as the job title, and one for 'ethan' with 'manager' as the job title.

LASTNAME	JOBTITLE
kumar	salesrep
ethan	manager

2 rows returned in 0.01 seconds    Download

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

### QUERY:

Select lastname ,salary,commission from employe where commission is not null order by salary,commission desc;

### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is: `1 Select lastname ,salary,commission from employe where commission is not null order by salary,commission desc;`. The results table has three rows: 'raj' with salary 6000 and commission 1000, 'kumar' with salary 50000 and commission 5000, and 'ethan' with salary 60000 and commission 6000.

LASTNAME	SALARY	COMMISSION
raj	6000	1000
kumar	50000	5000
ethan	60000	6000

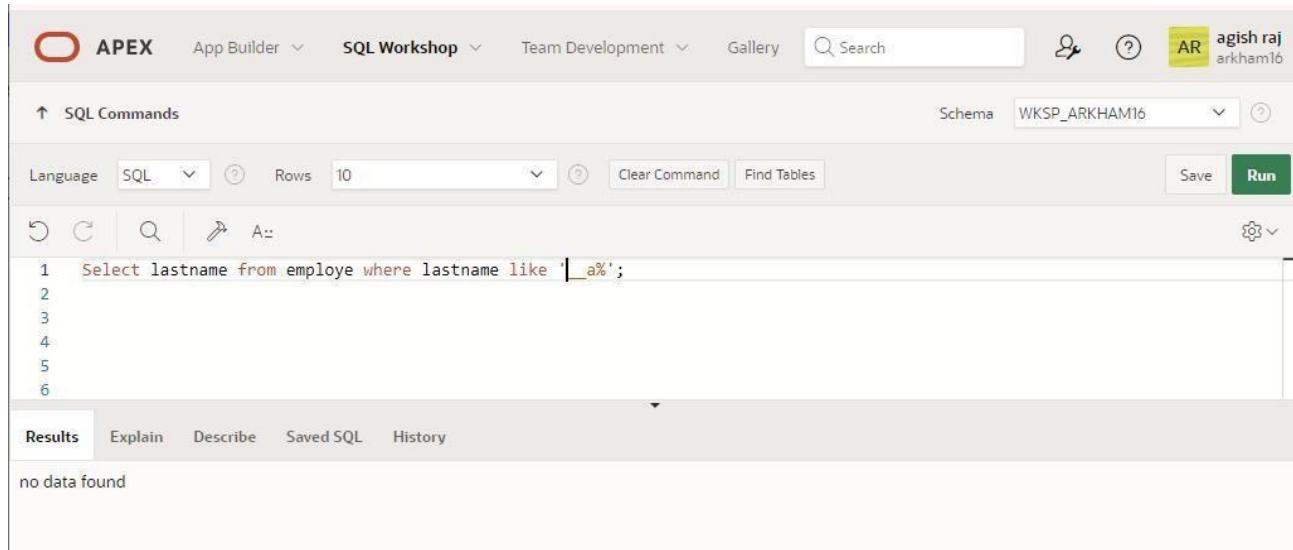
3 rows returned in 0.00 seconds    Download

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

### QUERY:

Select lastname from employe where lastname like '\_a%';

### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, Gallery, a search bar, and user information (agish raj, AR, arkham16). The main area is titled "SQL Commands". The schema is set to "WKSP\_ARKHAM16". The query entered is:

```
1 Select lastname from employe where lastname like '_a%';  
2  
3  
4  
5  
6
```

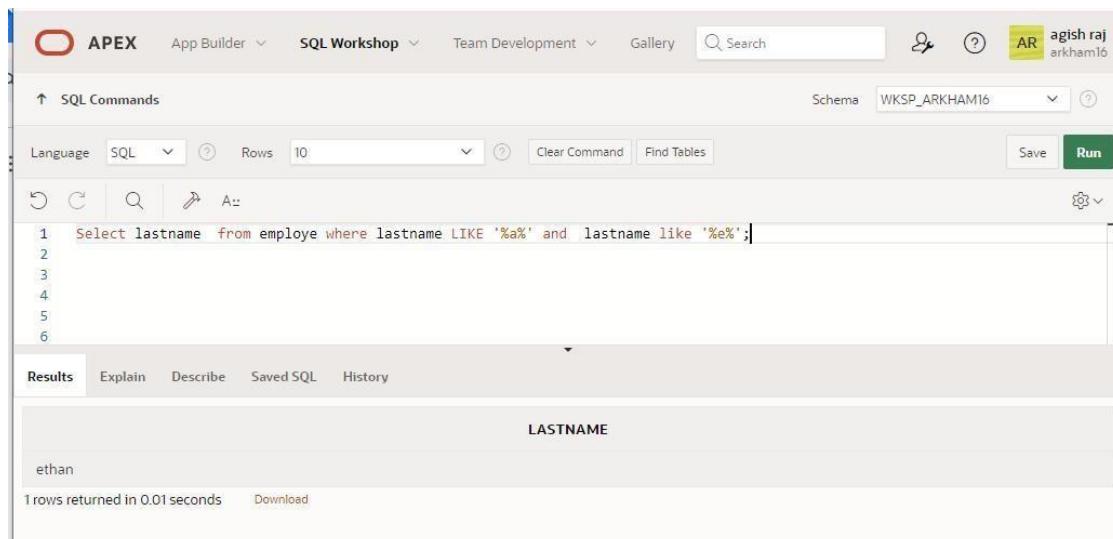
The results tab is selected, showing the message "no data found".

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

### QUERY:

Select lastname from employe where lastname LIKE '%a%' and lastname like '%e%';

### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, Gallery, a search bar, and user information (agish raj, AR, arkham16). The main area is titled "SQL Commands". The schema is set to "WKSP\_ARKHAM16". The query entered is:

```
1 Select lastname from employe where lastname LIKE '%a%' and lastname like '%e%';  
2  
3  
4  
5  
6
```

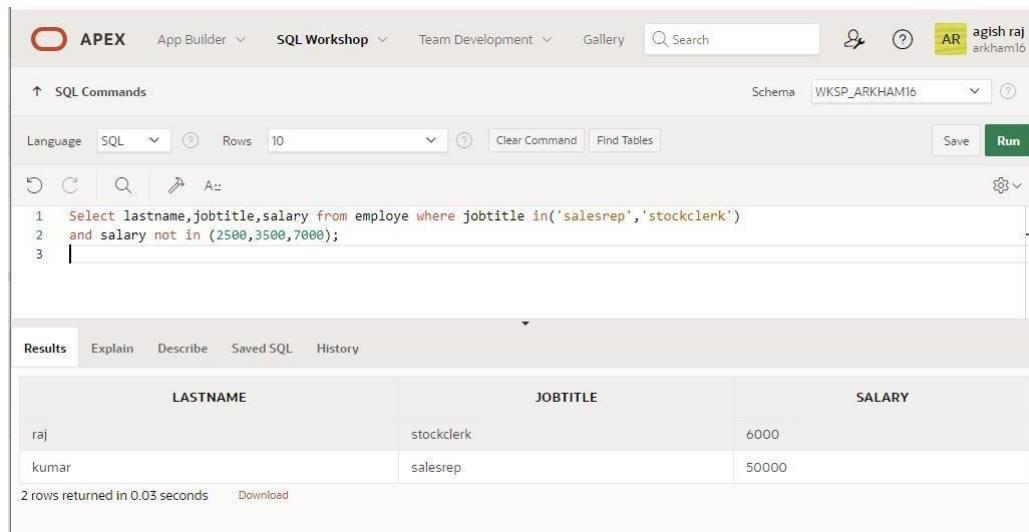
The results tab is selected, displaying a single row with the column "LASTNAME" containing "ethan". Below the table, it says "1 rows returned in 0.01 seconds".

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

### QUERY:

Select lastname,jobtitle,salary from employe where jobtitle in('salesrep','stockclerk') and salary not in (2500,3500,7000);

### OUTPUT:



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

```
1 Select lastname,jobtitle,salary from employe where jobtitle in('salesrep','stockclerk')
2 and salary not in (2500,3500,7000);
3
```

The Results pane displays the output of the query:

LASTNAME	JOBTITLE	SALARY
raj	stockclerk	6000
kumar	salesrep	50000

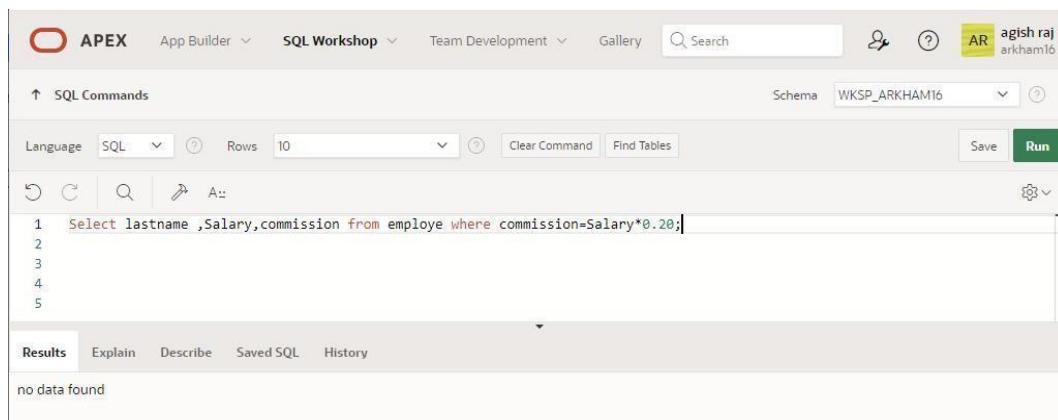
2 rows returned in 0.03 seconds

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

### QUERY:

Select lastname ,Salary,commission from employe where commission=Salary\*0.20;

### OUTPUT:



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

```
1 Select lastname ,Salary,commission from employe where commission=Salary*0.20;
2
3
4
5
```

The Results pane displays the output:

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:**

**Find the Solution for the following:**

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

**QUERY:**

**select sysdate as "date" from employe;**

**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` with the value `03/12/2024`. Below the results, it says `1 rows returned in 0.02 seconds`.

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

**QUERY:**

**select employenum, lastname, Salary, Salary+(15.5/100\*Salary) "new salary" From Employe;**

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands pane, the query `select employenum, lastname, salary, salary+(15.5/100*salary) "new_salary" from employe;` is entered. The Results pane displays the output for three employees: Raj, Kumar, and Ethan. The columns are `EMPLOYENUM`, `LASTNAME`, `SALARY`, and `new_salary`. The data is as follows:

EMPLOYENUM	LASTNAME	SALARY	new_salary
175	Raj	6000	6930
125	Kumar	50000	57750
255	Ethan	60000	69200

Below the results, it says `3 rows returned in 0.05 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 employenum, lastname, Salary, (Salary+(Salary\*15.5/100))-Salary  
“Increase” From Employe;

### OUTPUT:

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

```
1 select employenum, lastname, salary, salary+(15.5/100*salary) "new_salary", (salary+(15.5/100*salary))-salary as "increase" from employee;
```

The results section displays the following data:

EMPLOYENUM	LASTNAME	SALARY	new_salary	increase
126	Aaditya	6000	6930	930
125	kumar	50000	57750	7750
235	ethan	60000	69300	9300

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

### QUERY:

Select initcap(lastname) "Name", length(lastname) "Length of Name" from Employe where lastname like 'j%' or lastname like 'a%' or lastname like 'm%' order by lastname;

### OUTPUT:

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

```
1 select initcap(lastname),length(lastname) as "length_of_last_name" from employee where lastname like 'j%' or lastname like 'a%' or lastname like 'm%' order by lastname asc;
```

The results section displays the following data:

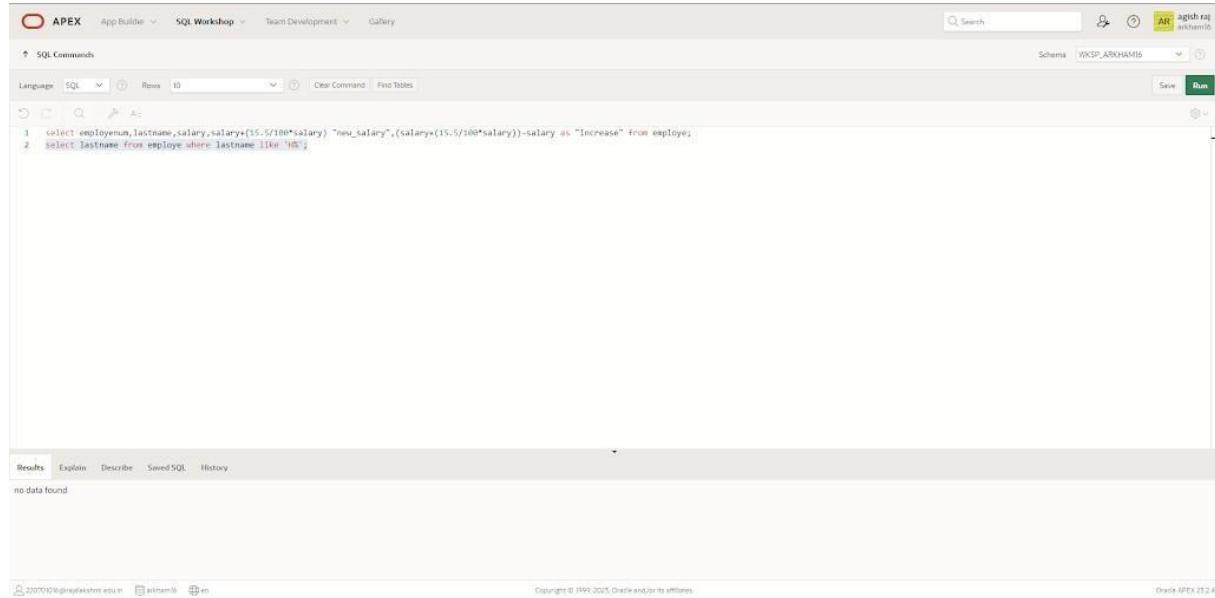
INITCAP(LASTNAME)	length_of_last_name
Aaditya	8

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

### QUERY:

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

### OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is active. The schema dropdown is set to WKSP\_ARKHAM16. The main workspace contains two SQL statements:

```
1 select ename,lastname,salary,salary+(15.5/100*salary) "new_salary",(salary+(15.5/100*salary))-salary as "Increase" from employee;
2 select lastname from employee where lastname like '&name%'
```

The results pane below shows the message "no data found". At the bottom of the page, there are footer links for copyright information and Oracle APEX 21.2.4.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, a search bar, and user information (agish raj, AR, arkham10). The main area is titled 'SQL Commands' with a schema dropdown set to WKSP\_ARKHAM16. The SQL editor contains the following query:

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

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

LASTNAME	MONTHS_WORKED
Aadhitya	312
kumar	358
ethan	409
bats	24026

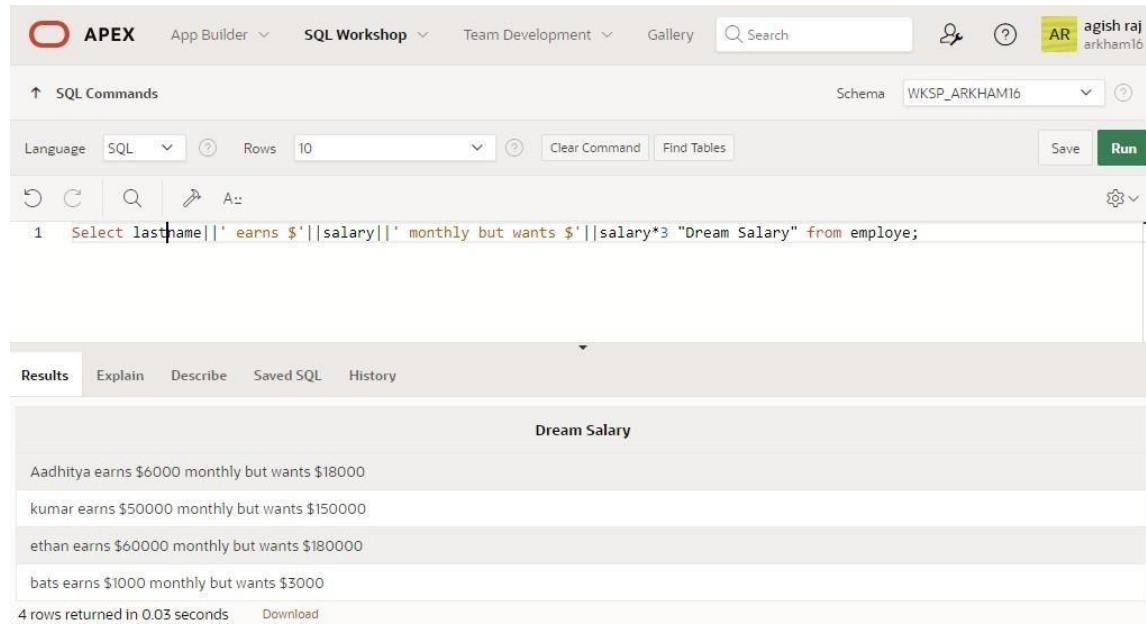
Below the table, a message indicates "4 rows returned in 0.03 seconds" and there is a "Download" link.

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

**QUERY:**

Select last\_name||' earns \$'||salary||' monthly but wants \$'||salary\*3 "Dream Salary"  
from employe;

**OUTPUT:**



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

```
1 Select lastname||' earns $'||salary||' monthly but wants $'||salary*3 "Dream Salary" from employe;
```

The Results pane displays the output:

Dream Salary
Aadhitya earns \$6000 monthly but wants \$18000
kumar earns \$50000 monthly but wants \$150000
ethan earns \$60000 monthly but wants \$180000
bats earns \$1000 monthly but wants \$3000

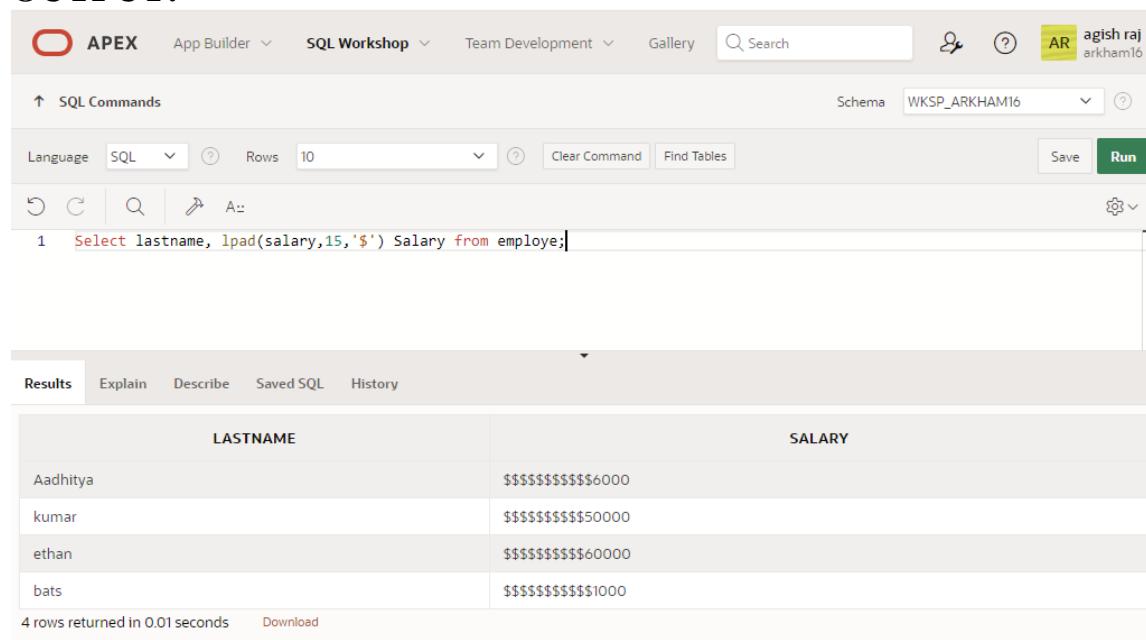
4 rows returned in 0.03 seconds

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

**QUERY:**

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

**OUTPUT:**



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

```
1 Select lastname, lpad(salary,15,'$') Salary from employe;
```

The Results pane displays the output:

LASTNAME	SALARY
Aadhitya	\$\$\$\$\$\$\$\$\$\$6000
kumar	\$\$\$\$\$\$\$\$\$\$50000
ethan	\$\$\$\$\$\$\$\$\$\$60000
bats	\$\$\$\$\$\$\$\$\$\$1000

4 rows returned in 0.01 seconds

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

**QUERY:**

```
select lastname, hiredate, to_char((next_day(hiredate,'Monday')),'fmday," the  
"ddspth "of" month,yyyy') "review" from employe;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and a user profile 'agish raj arkham16'. The 'SQL Commands' tab is active. The schema is set to 'WKSP\_ARKHAM16'. The SQL editor contains the following code:

```
1 select lastname, hiredate, to_char((next_day(hiredate,'Monday')),  
2 |'fmday," the "ddspth "of" month,yyyy') "review" from employe;
```

The results section displays the output of the query:

LASTNAME	HIREDATE	review
Aadhitya	03/20/1998	monday, the twenty-third of march,1998
kumar	05/01/1994	monday, the second of may,1994
ethan	02/20/1990	monday, the twenty-sixth of february,1990
bats	01/12/0022	monday, the nineteenth of january,22

4 rows returned in 0.00 seconds

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and a user profile 'agish raj arkham16'. The 'SQL Commands' tab is active. The schema is set to 'WKSP\_ARKHAM16'. The SQL editor contains the following code:

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

The results section displays the output of the query:

LASTNAME	HIREDATE	Day
bats	01/12/0022	Monday
ethan	02/20/1990	Tuesday
Aadhitya	03/20/1998	Friday
kumar	05/01/1994	Sunday

4 rows returned in 0.01 seconds

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

**RESULT:**

# DISPLAYING DATA FROM MULTIPLE TABLES

EXP.NO:7

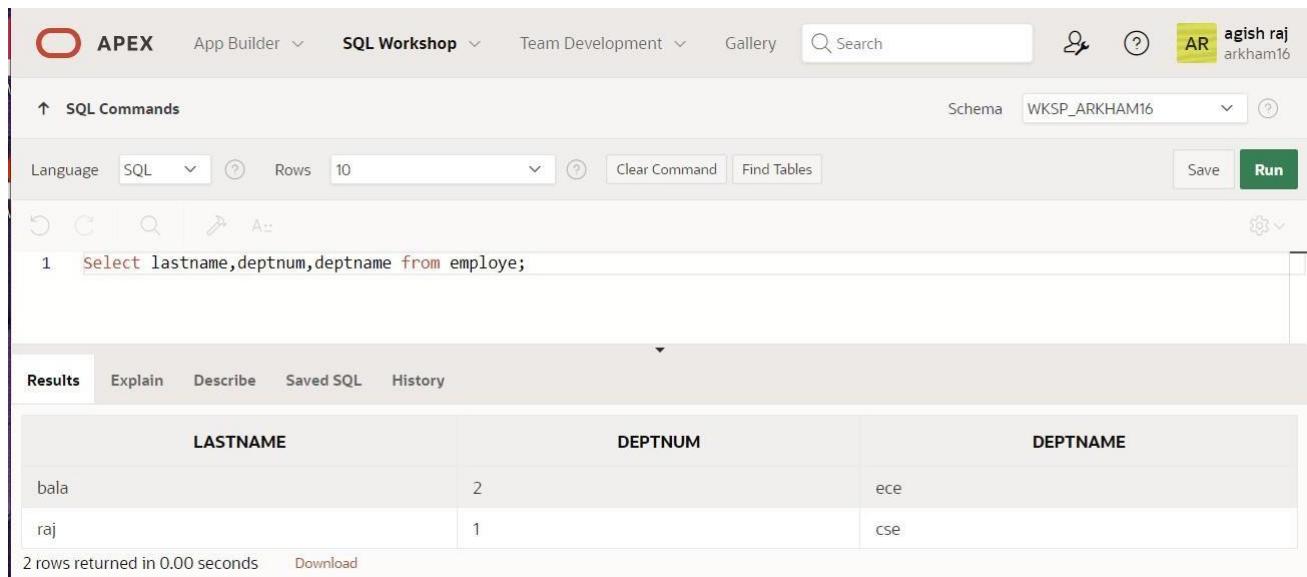
DATE:

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

**QUERY:**

Select lastname,deptnum,deptname from employe;

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and user information 'agish raj arkham16'. Below the navigation is a toolbar with 'SQL Commands', language selection ('SQL'), row limit ('10'), and 'Run' button. The main area contains the SQL command: '1 Select lastname,deptnum,deptname from employe;'. The results section displays the output in a grid format:

LASTNAME	DEPTNUM	DEPTNAME
bala	2	ece
raj	1	cse

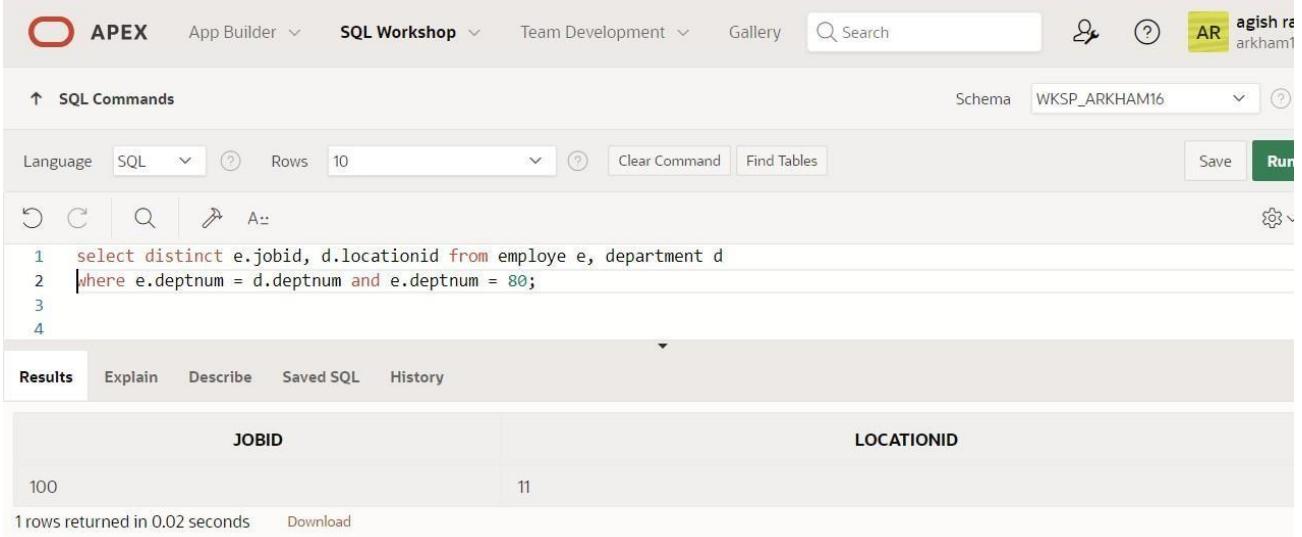
Below the results, it says '2 rows returned in 0.00 seconds' and has a 'Download' link.

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.jobid, d.locationid from employe e, department d  
where e.deptnum = d.deptnum and e.deptnum = 80;
```

### 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. A search bar and user profile are also present. The main area is titled "SQL Commands". The schema is set to "WKSP\_ARKHAM16". The SQL editor contains the following code:

```
1 select distinct e.jobid, d.locationid from employe e, department d  
2 where e.deptnum = d.deptnum and e.deptnum = 80;  
3  
4
```

The results tab is selected, displaying the output:

JOBID	LOCATIONID
100	11

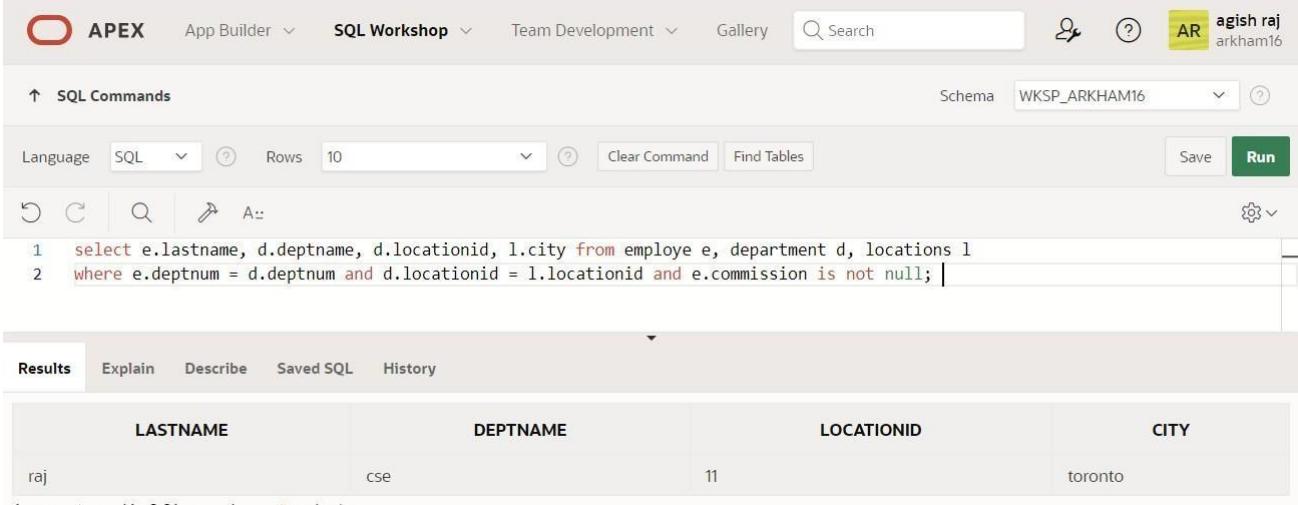
Below the table, it says "1 rows returned in 0.02 seconds" and has a "Download" link.

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.lastname, d.deptname, d.locationid, l.city  
from employe e, department d, locations l  
where e.deptnum = d.deptnum and d.locationid = l.locationid  
and e.comission is not null;
```

### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', and a search bar. On the right, there's a user profile for 'agish raj' and a schema dropdown set to 'WKSP\_ARKHAM16'. The main area is titled 'SQL Commands' with a language selector set to 'SQL'. The query is entered in the command window:

```
1 select e.lastname, d.deptname, d.locationid, l.city from employe e, department d, locations l  
2 where e.deptnum = d.deptnum and d.locationid = l.locationid and e.comission is not null; |
```

Below the command window, the results tab is selected. The output table has columns: LASTNAME, DEPTNAME, LOCATIONID, and CITY. One row is returned:

LASTNAME	DEPTNAME	LOCATIONID	CITY
raj	cse	11	toronto

At the bottom left, it says '1 rows returned in 0.01 seconds'. There are also 'Download' and 'Run' buttons.

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

### QUERY:

```
select lastname, deptname from employe e, department d  
where e.deptnum= d.deptnum and lastname like '%a%';
```

### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, a search bar, and user information (agish raj, arkham16). The main area is titled 'SQL Commands'. The schema is set to WKSP\_ARKHAM16. The SQL command entered is:

```
1 select e.lastname, e.deptname from employe e, department d  
2 where e.deptnum= d.deptnum and lastname like '%a%';  
3
```

The results section displays the output:

LASTNAME	DEPTNAME
raj	cse
bala	ece

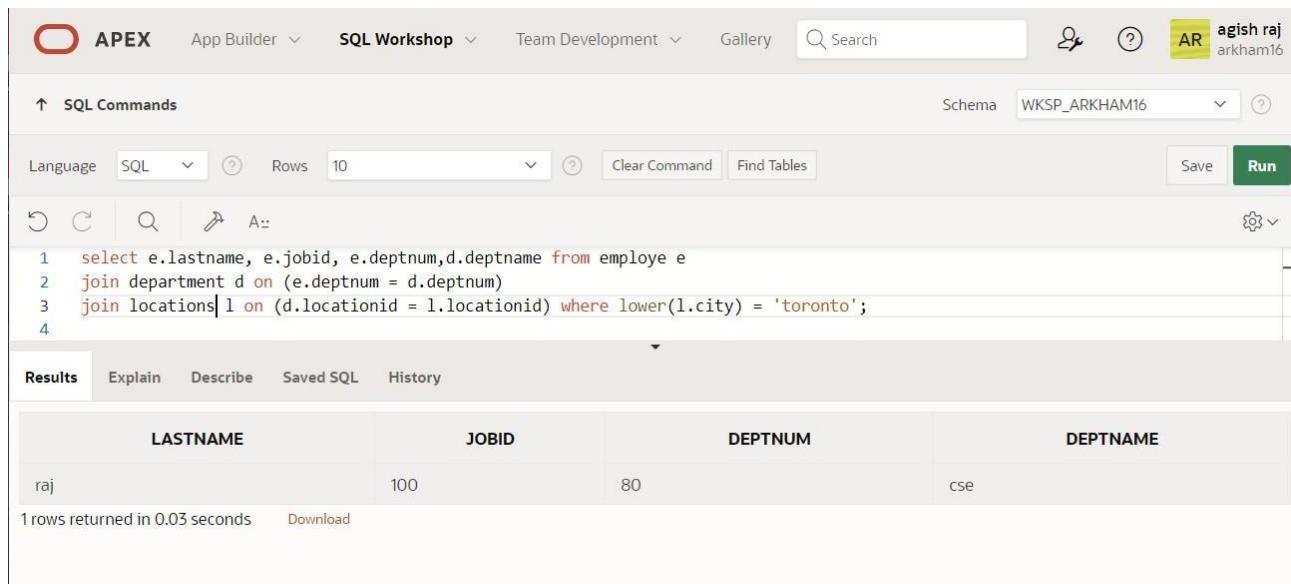
2 rows returned in 0.01 seconds [Download](#)

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

### QUERY:

```
select e.lastname, e.jobid, e.deptnum,d.deptname from employe e  
join department d on (e.deptnum = d.deptnum)  
join location l on (d.locationid = l.locationid) where lower(l.city) = 'toronto';
```

### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and user information 'agish raj' and 'AR arkham16'. Below the navigation is a toolbar with icons for copy, paste, search, and refresh, followed by a 'Run' button. The main area is titled 'SQL Commands' and contains the following SQL code:

```
1 select e.lastname, e.jobid, e.deptnum,d.deptname from employe e  
2 join department d on (e.deptnum = d.deptnum)  
3 join locations|l on (d.locationid = l.locationid) where lower(l.city) = 'toronto';  
4
```

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

LASTNAME	JOBID	DEPTNUM	DEPTNAME
raj	100	80	cse

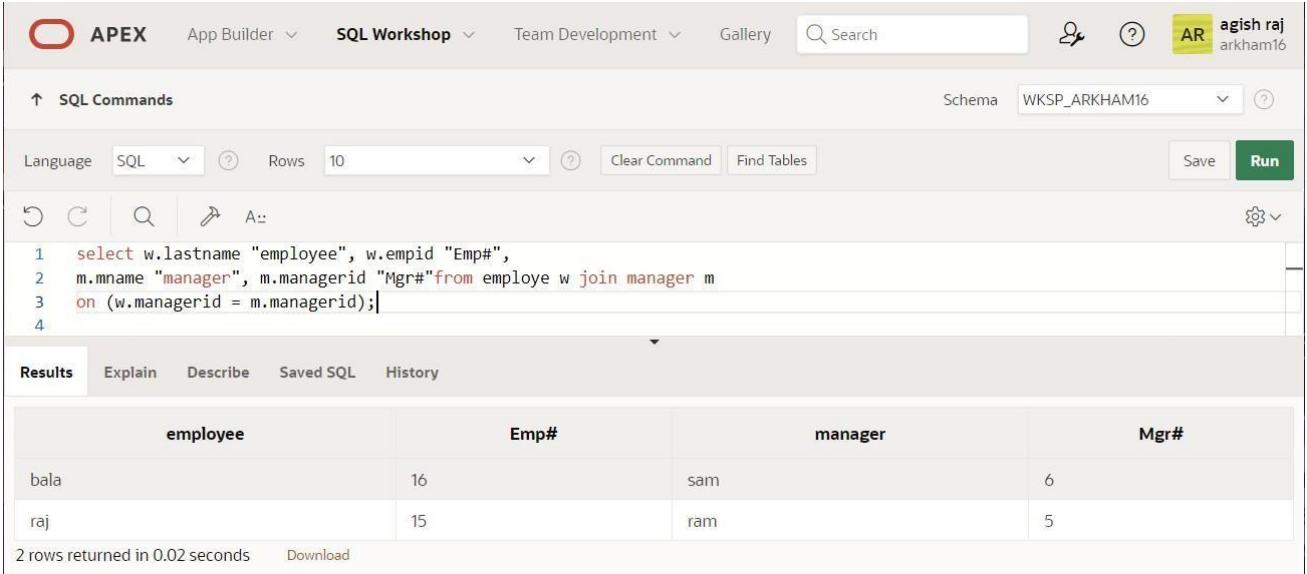
At the bottom left, it says '1 rows returned in 0.03 seconds'. There is also a 'Download' link.

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.lastname "employee", w.empid "Emp#",  
m.mname "manager", m.managerid "Mgr#" from employee w join manager m  
on (w.managerid = m.managerid);
```

### 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, along with a search bar and user information for 'agish raj' in 'arkham16'. The main area is titled 'SQL Commands' and contains a code editor with the following SQL query:

```
1 select w.lastname "employee",  
2      w.empid "Emp#",  
3      m.mname "manager",  
4      m.managerid "Mgr#" from employee w join manager m  
5    on (w.managerid = m.managerid);
```

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

employee	Emp#	manager	Mgr#
bala	16	sam	6
raj	15	ram	5

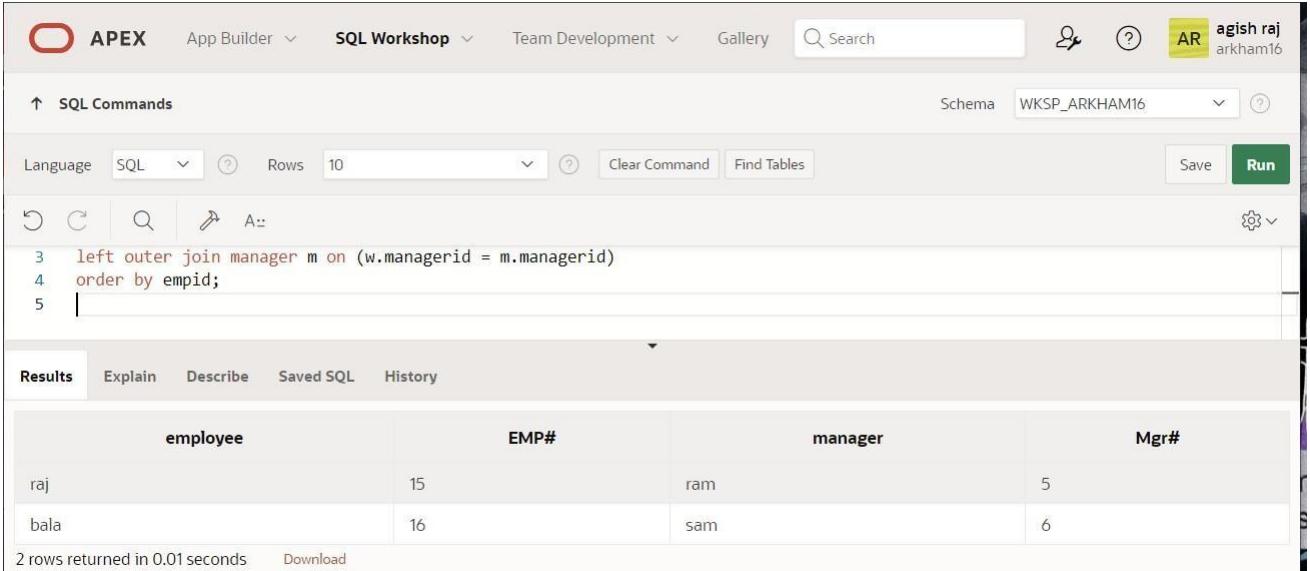
The message '2 rows returned in 0.02 seconds' is displayed at the bottom left, and a 'Download' link is available at the bottom right.

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.lastname "employee", w.empid "EMP#",  
m.mname "manager", m.managerid "Mgr#" from employee w  
left outer join manager m on (w.managerid = m.managerid)  
order by empid;
```

### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and user information 'agish raj arkham16'. 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 with the following SQL query:

```
3 left outer join manager m on (w.managerid = m.managerid)  
4 order by empid;  
5
```

Below the code editor is a 'Results' tab. The results table has four columns: 'employee', 'EMP#', 'manager', and 'Mgr#'. The data returned is:

employee	EMP#	manager	Mgr#
raj	15	ram	5
bala	16	sam	6

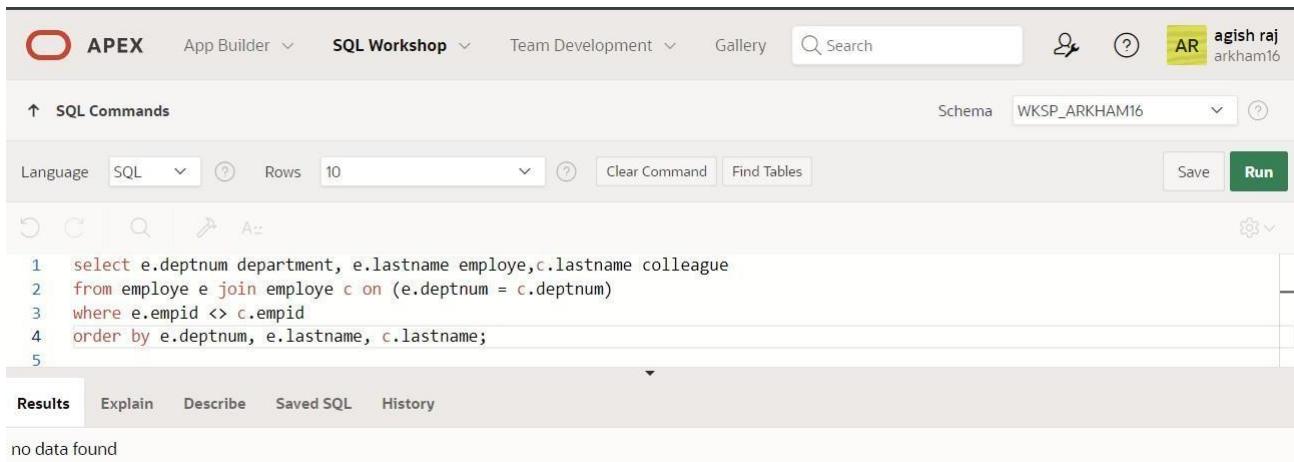
At the bottom of the results table, it says '2 rows returned in 0.01 seconds' and has a 'Download' link.

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.deptnum department, e.lastname employe,c.lastname colleague  
from employe e join employe c on (e.deptnum = c.deptnum)  
where e.empid <> c.empid  
order by e.deptnum, e.lastname, c.lastname;
```

### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, and a search bar. On the right, there's a user profile for 'agish raj' with the identifier 'ARKHAM16'. Below the navigation is a toolbar with icons for Undo, Redo, Search, and others. The main area is titled 'SQL Commands' and shows the following SQL code:

```
1 select e.deptnum department, e.lastname employe,c.lastname colleague  
2 from employe e join employe c on (e.deptnum = c.deptnum)  
3 where e.empid <> c.empid  
4 order by e.deptnum, e.lastname, c.lastname;  
5
```

Below the code, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, displaying the message "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.lastname, e.jobid, d.deptname, e.salary, j.grade  
from employe e join department d on (e.deptnum = d.deptnum)  
join jobgrade j on (e.salary between j.lowestsal and j.highestsal);
```

### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and user information 'agish raj arkham16'. The main area is titled 'SQL Commands' with tabs for 'Language' (set to 'SQL'), 'Rows' (set to 10), and buttons for 'Clear Command', 'Find Tables', 'Save', and 'Run'. The code editor contains the following SQL query:

```
1 select e.lastname, e.jobid, d.deptname, e.salary, j.grade  
2 from employe e join department d on (e.deptnum = d.deptnum)  
3 join jobgrade j on (e.salary between j.lowestsal and j.highestsal);  
4
```

The results section shows a table with the following data:

LASTNAME	JOBID	DEPTNAME	SALARY	GRADE
raj	100	cse	90000	A+
bala	99	ece	25000	B

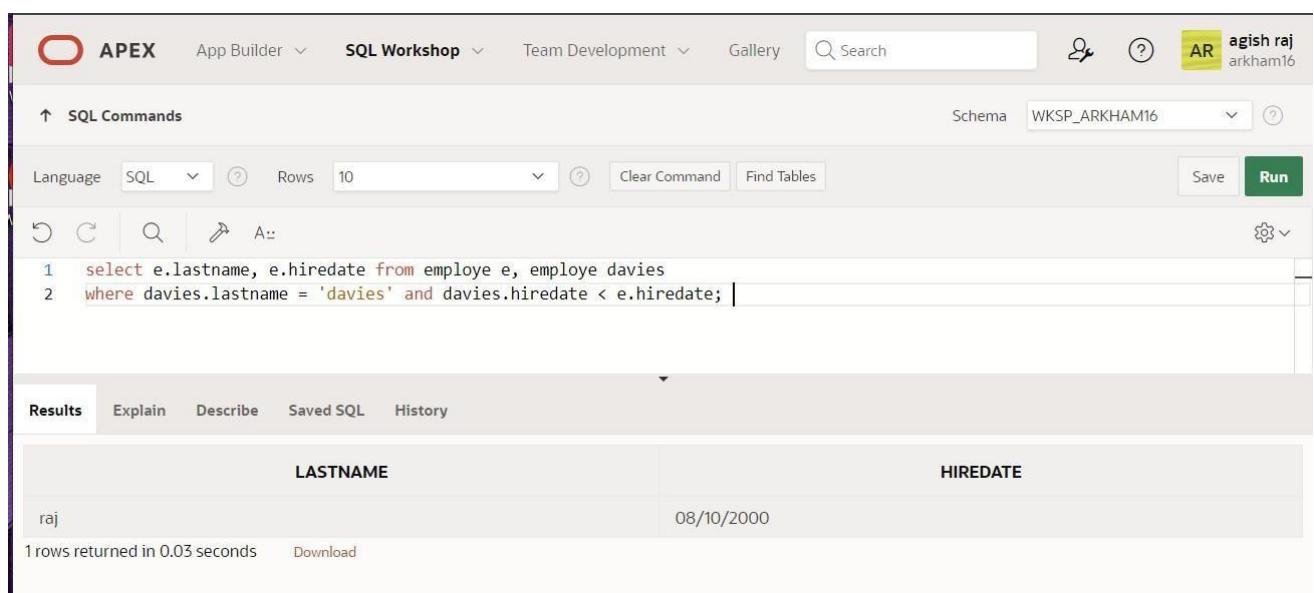
Below the table, it says '2 rows returned in 0.04 seconds' and has a 'Download' link.

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

### QUERY:

```
select e.lastname, e.hiredate from  
employe e, employe davies where  
davies.lastname = 'davies' and  
davies.hiredate < e.hiredate;
```

### OUTPUT:



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

```
1 select e.lastname, e.hiredate from employe e, employe davies  
2 where davies.lastname = 'davies' and davies.hiredate < e.hiredate;
```

The results table displays one row:

LASTNAME	HIREDATE
raj	08/10/2000

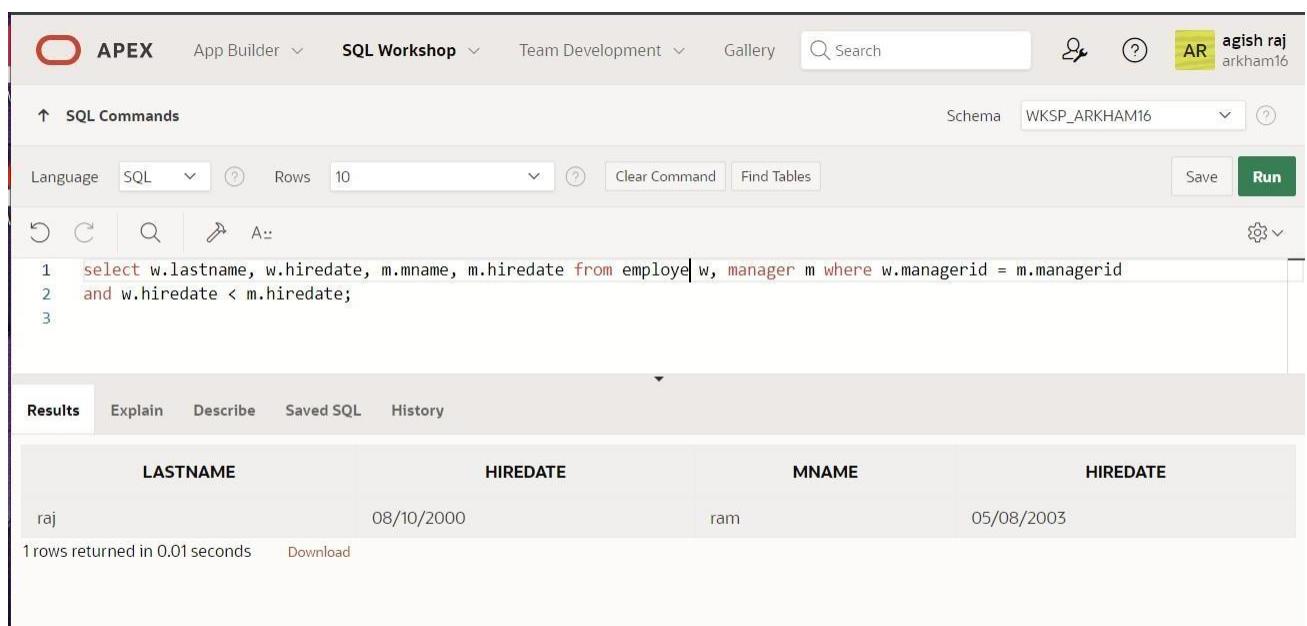
1 rows returned in 0.03 seconds

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

**QUERY:**

```
select w.lastname, w.hiredate, m.mname,  
m.hiredate from employee w, manager m  
where w.managerid = m.managerid  
and w.hiredate < m.hiredate;
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, a search bar, and a user profile for 'agish raj' with the ID 'arkham16'. The main area is titled 'SQL Commands'. The command entered is:

```
1 select w.lastname, w.hiredate, m.mname, m.hiredate from employee w, manager m where w.managerid = m.managerid  
2 and w.hiredate < m.hiredate;  
3
```

The results section shows a single row of data:

LASTNAME	HIREDATE	MNAME	HIREDATE
raj	08/10/2000	ram	05/08/2003

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

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

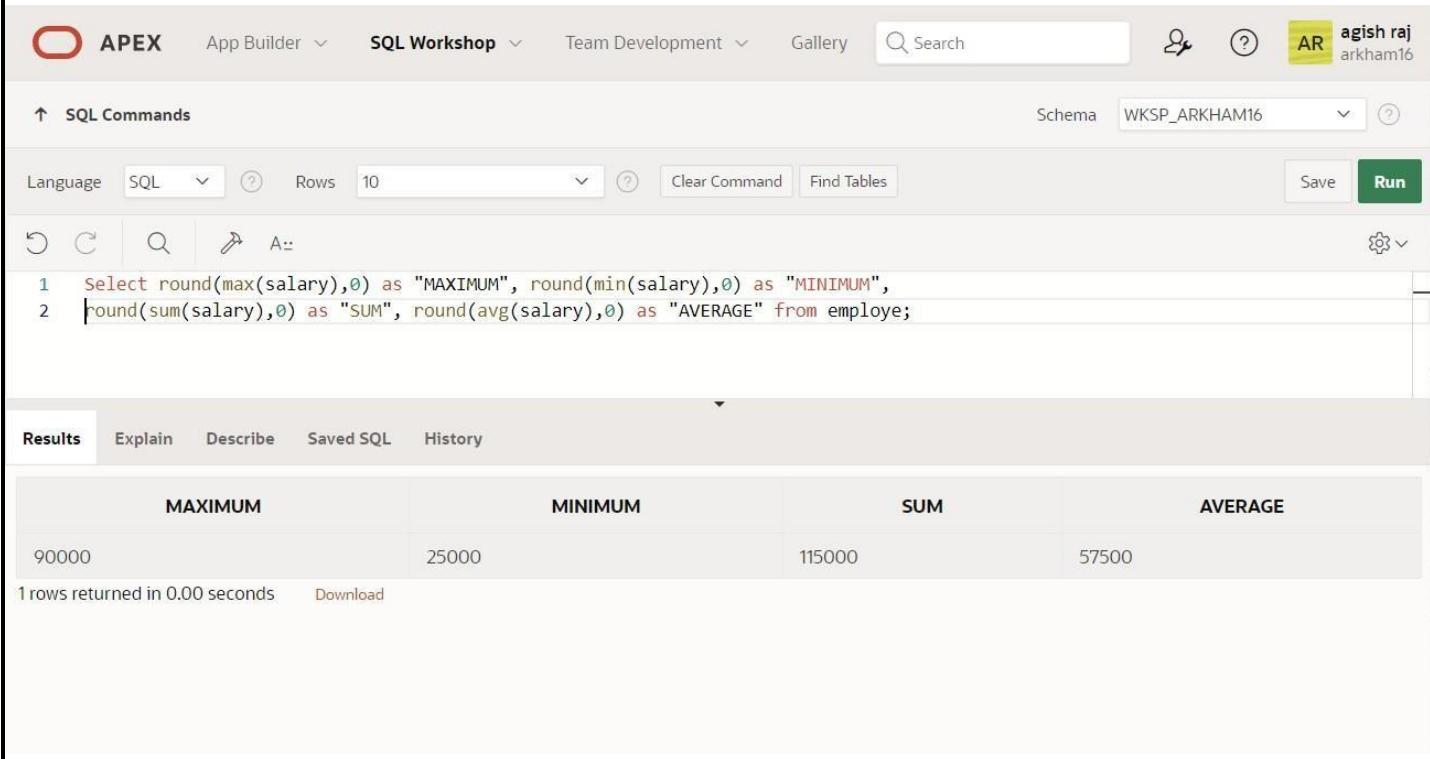
**FALSE**

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) as "MAXIMUM", round(min(salary),0) as "MINIMUM",
round(sum(salary),0) as "SUM", round(avg(salary),0) as "AVERAGE" from employe;
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, Gallery, and a user profile for 'arkham16'. The SQL Workshop tab is selected. Below the toolbar, there are buttons for SQL Commands, Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run. The main area contains the SQL query:

```
1 Select round(max(salary),0) as "MAXIMUM", round(min(salary),0) as "MINIMUM",
2 |round(sum(salary),0) as "SUM", round(avg(salary),0) as "AVERAGE" from employe;
```

The Results tab is active, displaying the output:

MAXIMUM	MINIMUM	SUM	AVERAGE
90000	25000	115000	57500

Below the table, it says '1 rows returned in 0.00 seconds' and has a 'Download' link.

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

**QUERY:**

```
select jobid,round(max(salary),0) as "MAXIMUM", round(min(salary),0) as "MINIMUM",
round(sum(salary),0) as "SUM", round(avg(salary),0) AS "AVERAGE"
from employe group by jobid;
```

**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, along with a search bar and user profile information for 'agish raj'.

In the SQL Workshop tab, the schema is set to 'WKSP\_ARKHAM16'. The SQL command entered is:

```
1 select jobid,round(max(salary),0) as "MAXIMUM", round(min(salary),0) as "MINIMUM",
2 round(sum(salary),0) as "SUM", round(avg(salary),0) AS "AVERAGE"
3 from employe group by jobid;
4
```

The results section displays the output in a table:

JOBID	MAXIMUM	MINIMUM	SUM	AVERAGE
99	25000	25000	25000	25000
100	90000	90000	90000	90000

Below the table, it says '2 rows returned in 0.00 seconds' and there is 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 jobid, count(jobid) from  
employe group by jobid;
```

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and a user profile for 'agish raj arkham16'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_ARKHAM16'. The SQL editor contains the query: 'select jobid, count(jobid) from employe group by jobid;'. Below the editor, the 'Results' tab is selected, displaying a table with two rows. The table has columns 'JOBID' and 'COUNT(JOBID)'. The data is as follows:

JOBID	COUNT(JOBID)
99	1
100	1

Below the table, it says '2 rows returned in 0.00 seconds' and there is a 'Download' link.

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 managerid) "NUMBER OF MANAGERS" from employee;**

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and user information 'agish raj arkham16'. The main area is titled 'SQL Commands' with tabs for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', 'Find Tables', 'Save', and 'Run'. The command entered is 'select count(distinct managerid) "NUMBER OF MANAGERS" from employee;'. Below the command, the results tab is selected, showing the output: 'NUMBER OF MANAGERS' with a value of '2'. A note indicates '1 rows returned in 0.00 seconds'.

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

**QUERY:**

**select max(salary)-min(salary) as "DIFFERENCE" from employee;**

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and user information 'agish raj arkham16'. The main area is titled 'SQL Commands' with tabs for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', 'Find Tables', 'Save', and 'Run'. The command entered is 'select max(salary)-min(salary) as "DIFFERENCE" from employee;'. Below the command, the results tab is selected, showing the output: 'DIFFERENCE' with a value of '65000'. A note indicates '1 rows returned in 0.01 seconds'.

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

#### QUERY:

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

#### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, Gallery, a search bar, and user information (agish raj, arkham16). The SQL Commands tab is selected, showing the following SQL code:

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

The Results tab displays the output:

MANAGERID	MIN(SALARY)
5	90000
6	25000

Below the table, it says "2 rows returned in 0.00 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(*) as TOTAL, sum(decode(extract(year from hiredate), 1995, 1, 0)) as "1995",  
sum(decode(extract(year from hiredate), 1996, 1, 0)) as "1996",  
sum (decode(extract(year from hiredate), 1997, 1, 0)) as "1997",  
sum (decode(extract(year from hiredate), 1998, 1, 0)) as "1998" from employe;
```

#### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, Gallery, a search bar, and user information (agish raj, arkham16). The SQL Commands tab is selected, showing the following SQL code:

```
1 select count(*) as TOTAL, sum(decode(extract(year from hiredate), 1995, 1, 0)) as "1995",  
2 sum(decode(extract(year from hiredate), 1996, 1, 0)) as "1996",  
3 sum (decode(extract(year from hiredate), 1997, 1, 0)) as "1997",  
4 sum (decode(extract(year from hiredate), 1998, 1, 0)) as "1998" from employe;  
5
```

The Results tab displays the output:

TOTAL	1995	1996	1997	1998
2	0	0	0	1

Below the table, it says "1 rows returned in 0.00 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 jobid "JOB",
sum(decode(deptnum , 20, salary)) "DEPT 20",
sum(decode(deptnum , 50, salary)) "DEPT 50",
sum(decode(deptnum , 80, salary)) "DEPT 80",
sum(decode(deptnum , 90, salary)) "DEPT 90",
sum(salary) "TOTAL" from employe group by jobid;
```

#### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, a search bar, and a user profile for 'arkish raj'.

The SQL Commands section displays the following SQL code:

```
1 select jobid "JOB",
2 sum(decode(deptnum , 20, salary)) "DEPT 20",
3 sum(decode(deptnum , 50, salary)) "DEPT 50",
4 sum(decode(deptnum , 80, salary)) "DEPT 80",
5 sum(decode(deptnum , 90, salary)) "DEPT 90", sum(salary) "TOTAL" from employe group by jobid;
```

The Results section shows the output of the query:

JOB	DEPT 20	DEPT 50	DEPT 80	DEPT 90	TOTAL
99	-	-	-	25000	25000
100	-	-	90000	-	90000

Below the table, it says '2 rows returned in 0.01 seconds' and has 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.deptname "NAME",d.locationid "LOCATION ",count(*) "NUMBER OF PEOPLE",
round(avg(salary),2) "SALARY" from employe e,department d
where e.deptnum = d.deptnum group by d.deptname, d.locationid;
```

**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, along with a search bar and user profile information (agish raj, ar, arkham16). Below the navigation is a toolbar with various icons for SQL operations like insert, update, delete, and search. The main workspace displays the SQL command entered:

```
1 select d.deptname "NAME",d.locationid "LOCATION ",count(*) "NUMBER OF PEOPLE",
2 |From employe e,department d
3 where e.deptnum = d.deptnum group by d.deptname, d.locationid;
4
5
6
```

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

NAME	LOCATION	NUMBER OF PEOPLE	SALARY
cse	11	1	90000
ece	12	1	25000

Below the table, a message indicates "2 rows returned in 0.06 seconds" and a "Download" link.

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

**RESULT:**

## SUB-QUERIES

EXP.NO:9

DATE:

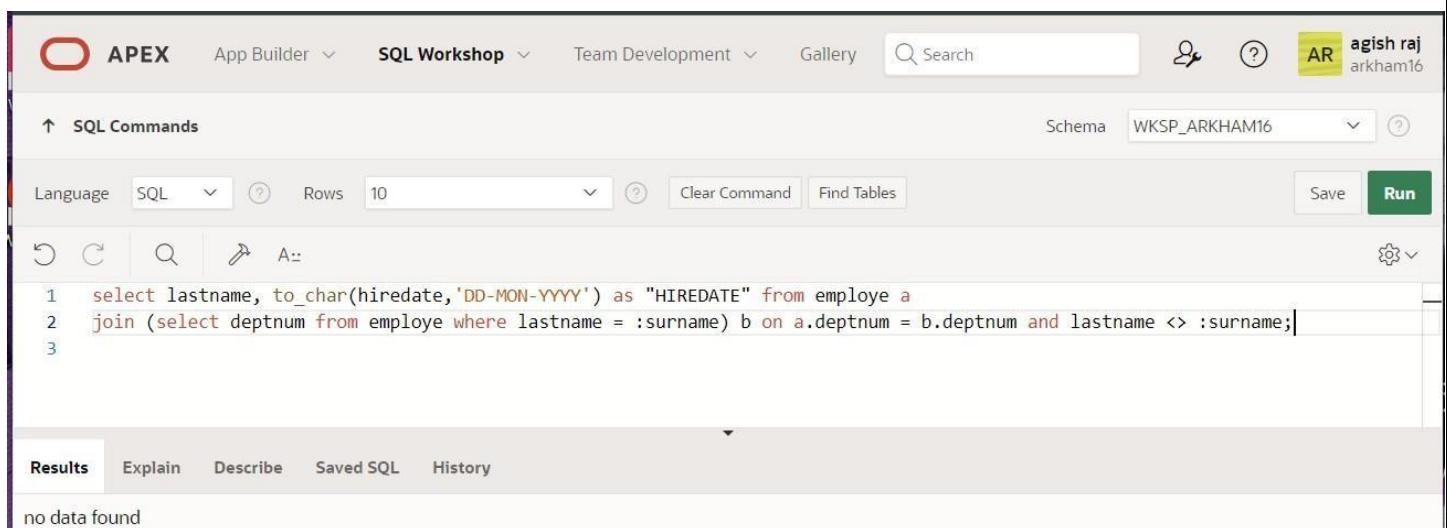
Find the Solution for the following:

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

### **QUERY:**

```
select lastname, to_char(hiredate,'DD-MON-YYYY') as "HIREDATE" from employe a  
join (select deptnum from employe where lastname = :surname) b on a.deptnum = b.deptnum  
and lastname <> :surname;
```

### **OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, and a search bar. On the right, there's a user profile for 'agish raj' and the schema 'WKSP\_ARKHAM16'. The main workspace is titled 'SQL Commands'. It features a toolbar with icons for Undo, Redo, Find, Replace, and a dropdown menu. Below the toolbar, the command input area contains the following SQL code:

```
1 select lastname, to_char(hiredate,'DD-MON-YYYY') as "HIREDATE" from employe a  
2 join (select deptnum from employe where lastname = :surname) b on a.deptnum = b.deptnum and lastname <> :surname;  
3
```

At the bottom of the workspace, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, and the message 'no data found' is displayed.

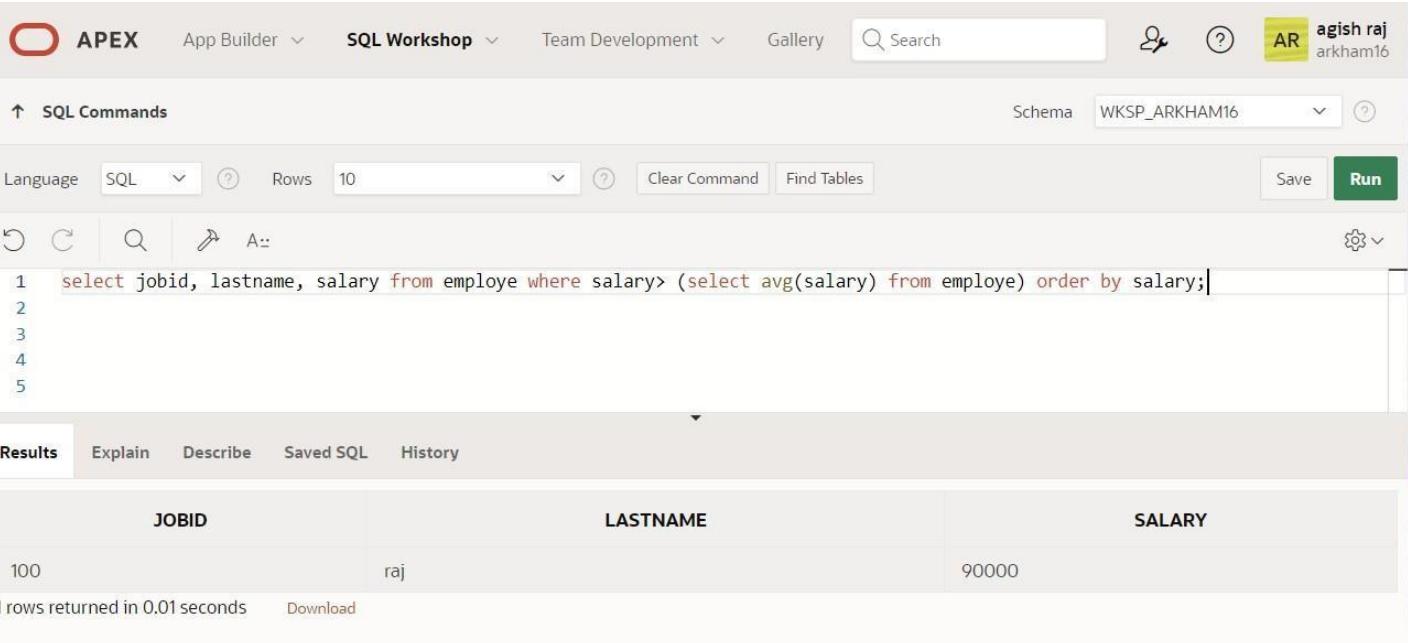
2. Create a report that displays the employee number, last name, and salary of all employees who earn more than the average salary. Sort the results in order of ascending salary.

### QUERY:

```
select jobid, lastname, salary from employe where salary>
```

```
(select avg(salary) from employe) order by salary;
```

### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, and a search bar. The right side shows a user profile for 'agish raj' with the ID 'AR'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_ARKHAM16'. The toolbar below has options for Language (SQL selected), Rows (set to 10), Clear Command, Find Tables, Save, and Run. The code editor contains the following SQL query:

```
1 select jobid, lastname, salary from employe where salary > (select avg(salary) from employe) order by salary;
2
3
4
5
```

The results tab is active, displaying the output of the query:

JOBID	LASTNAME	SALARY
100	raj	90000

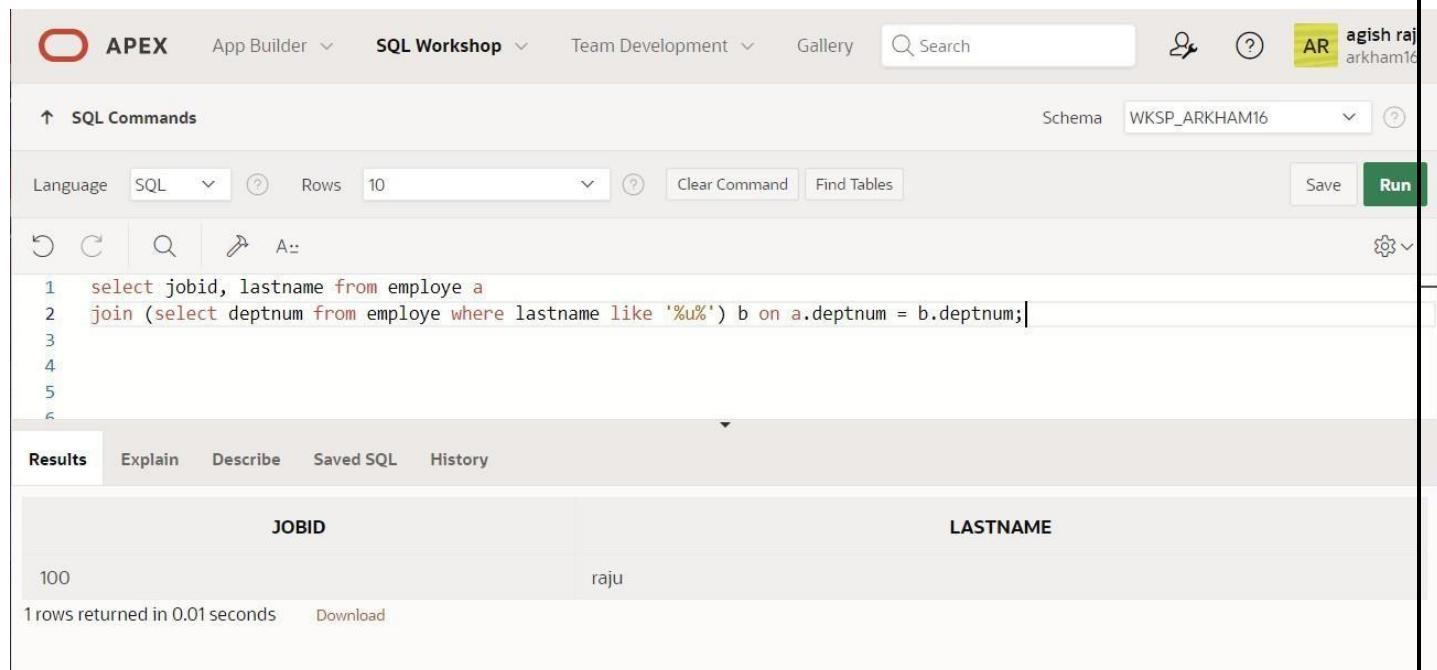
Below the table, it says '1 rows returned in 0.01 seconds' and there is a 'Download' link.

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 jobid, lastname from employe a  
join (select deptnum from employe where lastname like '%u%') b on a.deptnum =  
b.deptnum;
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and a user profile for 'agish raju'. The 'Schema' dropdown is set to 'WKSP\_ARKHAM16'. The main area is titled 'SQL Commands' with tabs for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', 'Find Tables', 'Save', and 'Run' (highlighted in green). Below these are icons for Undo, Redo, Find, and Paste. The SQL command is listed in the code editor:

```
1 select jobid, lastname from employe a  
2 join (select deptnum from employe where lastname like '%u%') b on a.deptnum = b.deptnum;  
3  
4  
5  
6
```

The results tab is selected, showing a single row of data:

JOBID	LASTNAME
100	raju

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

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

### QUERY:

```
select lastname, deptnum, jobid from employee a  
join (select deptno from department where locationid=1700)b on  
a.deptnum=b.deptno;
```

### 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, along with a search bar and user profile information for 'agish raj'.

The main area is titled 'SQL Commands'. It features a code editor with the following SQL query:

```
1 select lastname, deptnum, jobid from employee a  
2 join (select deptno from department where locationid=1700)b on a.deptnum=b.deptno;  
3  
4  
5  
6
```

Below the code editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, displaying the query's output in a table format:

LASTNAME	DEPTNUM	JOBID
davies	90	99

At the bottom left, it says '1 rows returned in 0.02 seconds'. There is also a 'Download' link.

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

**QUERY:**

```
select lastname, salary from employe  
where managerid in (select empid from employe where managerid is null);
```

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and user information 'agish raj arkham16'. Below the navigation is a toolbar with icons for undo, redo, search, and other functions. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_ARKHAM16'. The SQL editor contains the following code:

```
1 select lastname, salary from employe  
2 where managerid in (select empid from employe where managerid is null);  
3
```

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

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 deptnum, lastname, jobid from employe  
where deptnum = (select deptnum from department where deptname='executive');
```

**OUTPUT:**

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

```
1 select deptnum, lastname, jobid from employe  
2 where deptnum = (select deptnum from department where deptname='executive');  
3
```

The results section displays the message 'no data found'.

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 empid, lastname, salary from employe where salary > (select avg(salary) from employe) and deptnum in (select deptnum from employe where lastname like '%u%');
```

### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, a search bar, and a user profile for 'agish raj' (arkham16). The main area is titled 'SQL Commands'. The schema is set to 'WKSP\_ARKHAM16'. The SQL editor contains the following code:

```
1 select empid, lastname, salary from employe where salary > (select avg(salary) from employe) and
2 deptnum in (select deptnum from employe where lastname like '%u%');
3
4
```

Below the editor, the 'Results' tab is selected. The output table has columns: EMPID, LASTNAME, and SALARY. One row is returned:

EMPID	LASTNAME	SALARY
-	raju	90000

At the bottom left, it says '1 rows returned in 0.01 seconds'. There is also a 'Download' link.

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

**RESULT:**

## USING THE SET OPERATORS

EX.NO:10

DATE:

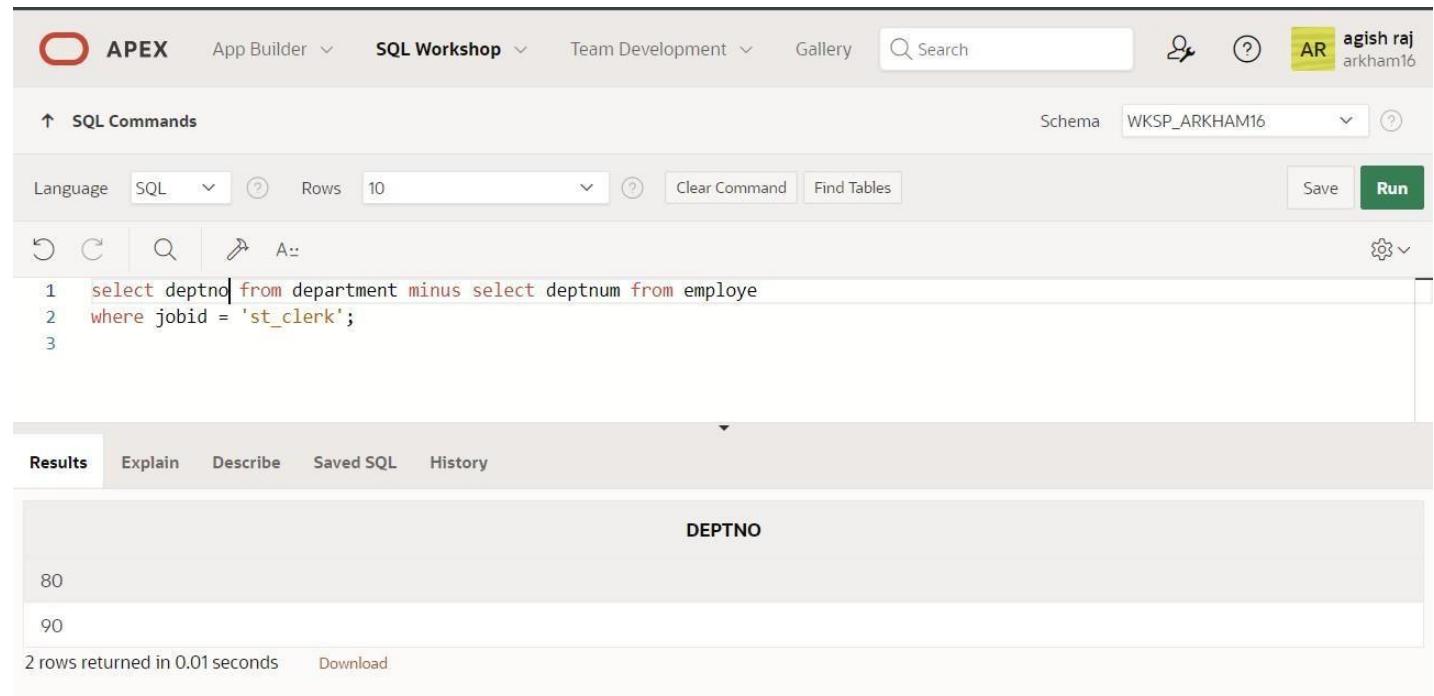
Find the Solution for the following:

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

**QUERY:**

```
select deptnum from department minus  
select deptnum from employe  
where jobid = 'st_clerk';
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, and a search bar. The user is identified as agish raj. The main workspace is titled 'SQL Commands' and shows the following query:

```
1 select deptno from department minus select deptnum from employe  
2 where jobid = 'st_clerk';  
3
```

The results tab is selected, displaying the output:

DEPTNO
80
90

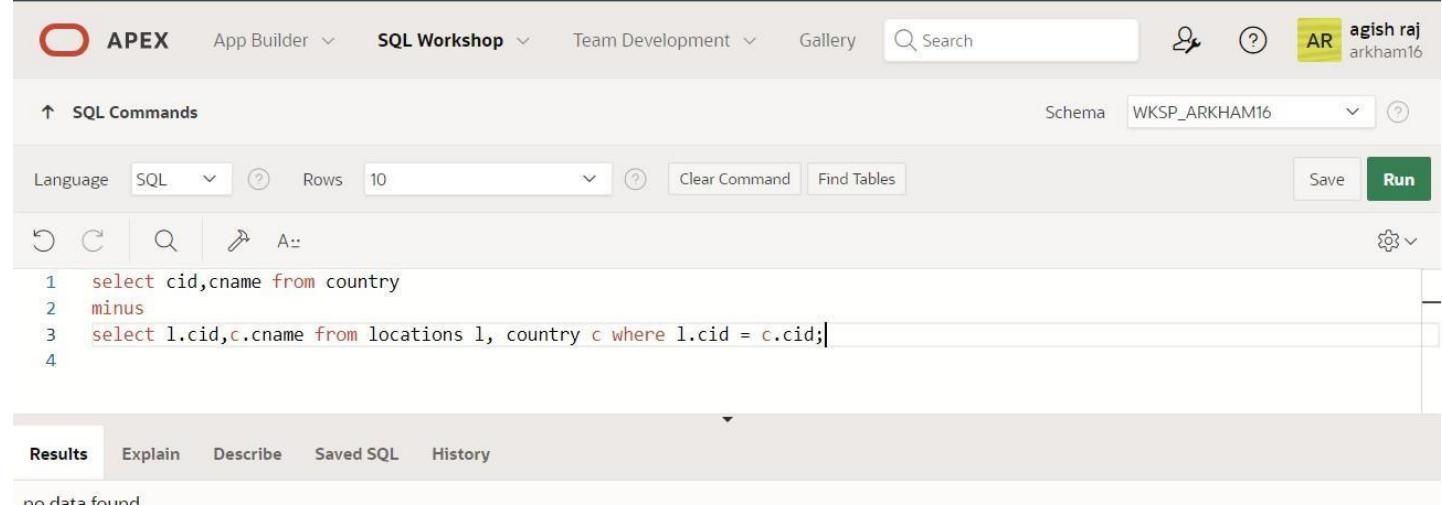
Below the table, it says '2 rows returned in 0.01 seconds' and has a 'Download' link.

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 cid,cname from
country
minus
select l.cid,c.cname from
location l, country c where
l.cid = c.cid;
```

**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. A search bar and user profile information ('agish raj' and 'arkham16') are also at the top. Below the tabs, the schema is set to 'WKSP\_ARKHAM16'. The main area is titled 'SQL Commands' and contains the following SQL code:

```
1 select cid,cname from country
2 minus
3 select l.cid,c.cname from locations l, country c where l.cid = c.cid;
4
```

Below the code, there are buttons for Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run. The 'Results' tab is selected, showing the message 'no data found'.

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

**QUERY:**

```
select distinct jobid, deptnum from employe
where deptnum = 10 union all
select distinct jobid, deptnum from employe
where deptnum = 50 union all
select distinct jobid, deptnum from employe
where deptnum = 20;
```

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, a search bar, and a user profile for agish raj. The main area is titled "SQL Commands". The schema dropdown is set to WKSP\_ARKHAM16. The command input field contains the following SQL code:

```
1 select distinct jobid, deptnum from employe where deptnum = 10 union all
2 select distinct jobid, deptnum from employe where deptnum = 50 union all
3 select distinct jobid, deptnum from employe where deptnum = 20;
4
```

Below the command input, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, displaying the following table:

JOBID	DEPTNUM
67	10
99	20

At the bottom left, it says "2 rows returned in 0.01 seconds".

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

#### QUERY:

```
select empid,jobno from
employe intersect select
empid,jobno from
jobhistory;
```

#### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, Gallery, a search bar, and a user profile for 'agish raj arkham16'. The SQL Workshop tab is selected. Below the toolbar, the schema is set to 'WKSP\_ARKHAM16'. The main area shows the SQL command entered:

```
1 select empid,jobno from employe intersect select empid,jobno from jobhistory;
```

The results section displays the output of the query:

EMPID	JOBNO
50	70

Below the table, it says '1 rows returned in 0.01 seconds' and there is a 'Download' link.

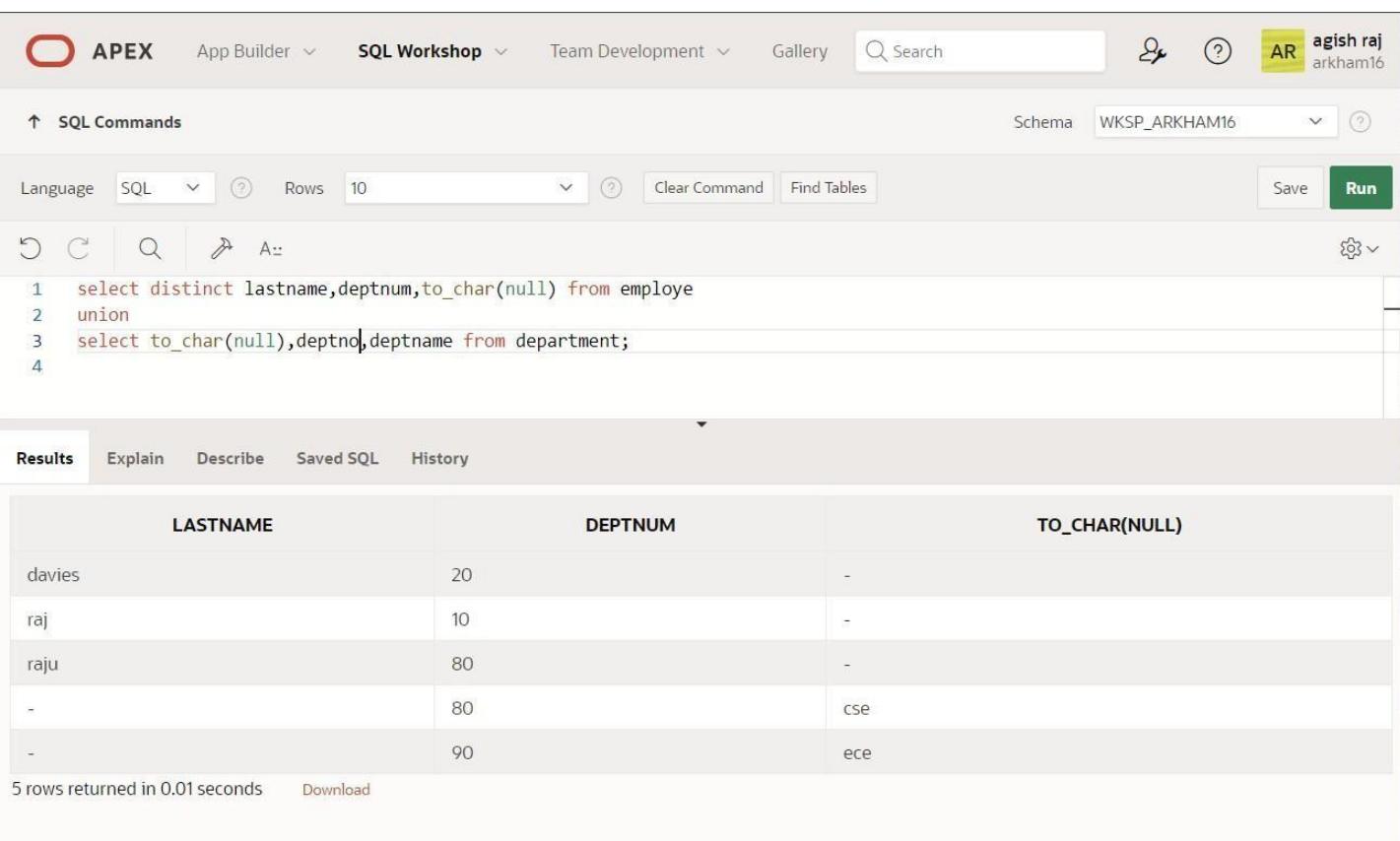
5. The HR department needs a report with the following specifications:

- 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 distinct
lastname,deptnum,to_char(null) from
employe
union
select to_char(null),deptnum,deptname
from department;
```

#### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, a search bar, and user information (agish raj, arkham16). The main area is titled 'SQL Commands' with a schema dropdown set to WKSP\_ARKHAM16. The SQL editor contains the following code:

```
1 select distinct lastname,deptnum,to_char(null) from employe
2 union
3 select to_char(null),deptno,deptname from department;
4
```

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

LASTNAME	DEPTNUM	TO_CHAR(NULL)
davies	20	-
raj	10	-
raju	80	-
-	80	cse
-	90	ece

At the bottom, it says '5 rows returned in 0.01 seconds' and has a 'Download' link.

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

## **RESULT:**

# CREATING VIEWS

EXP NO: 11

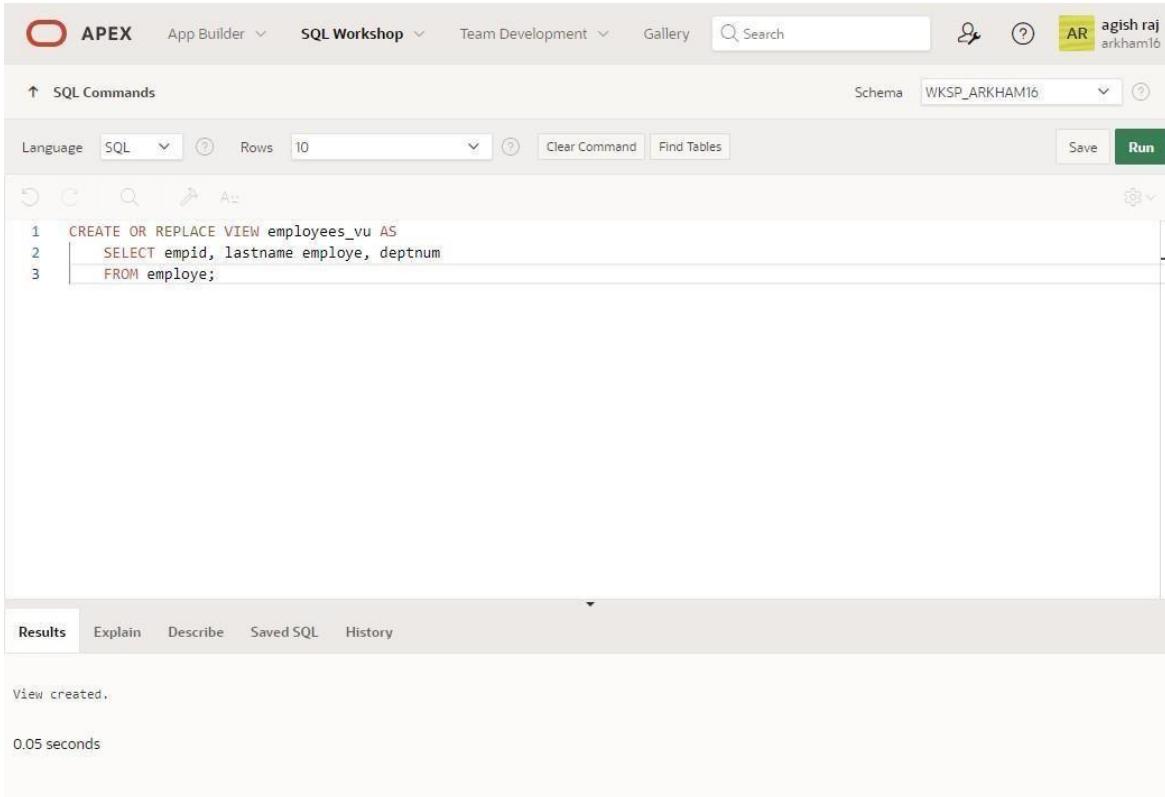
DATE:

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 view employees_vu as select empid, lastname employe,deptnum from employe;
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, a search bar, and user information (agish raj, arkham16). The main area is titled 'SQL Commands' with a schema dropdown set to WKSP\_ARKHAM16. The command input field contains the following SQL code:

```
1 CREATE OR REPLACE VIEW employees_vu AS
2   SELECT empid, lastname employe, deptnum
3   FROM employe;
```

The 'Run' button is highlighted in green. Below the command area, there are tabs for Results, Explain, Describe, Saved SQL, and History. The results tab displays the message "View created." and "0.05 seconds".

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

**QUERY:**

```
select * from employees_vu;
```

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, a search bar, and a user profile for agish raj. The main area is titled "SQL Commands" with a schema dropdown set to WKSP\_ARKHAM16. The command input field contains:

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

The results section below shows the output of the query:

VIEW_NAME	TEXT
EMPLOYEES_VU	SELECT empid, lastname employee, deptnum FROM employee

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

The screenshot shows the Oracle SQL Workshop interface again. The top navigation bar is identical. The main area has the command input field containing:

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

The results section shows the output of the query:

EMPID	EMPLOYEE	DEPTNUM
50	davies	20
6	raj	10
70	raju	80

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

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

**QUERY:**

```
select employee, deptnum from employees_vu;
```

**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', 'Gallery', a search bar, and user information 'agish raj arkham16'. Below the bar, the 'SQL Commands' tab is active. The command entered is:

```
1  SELECT employee, deptnum
2  |   FROM employees_vu;
```

At the bottom of the command area, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, displaying the query output:

EMPLOYEE	DEPTNUM
davies	20
raj	10
raju	80

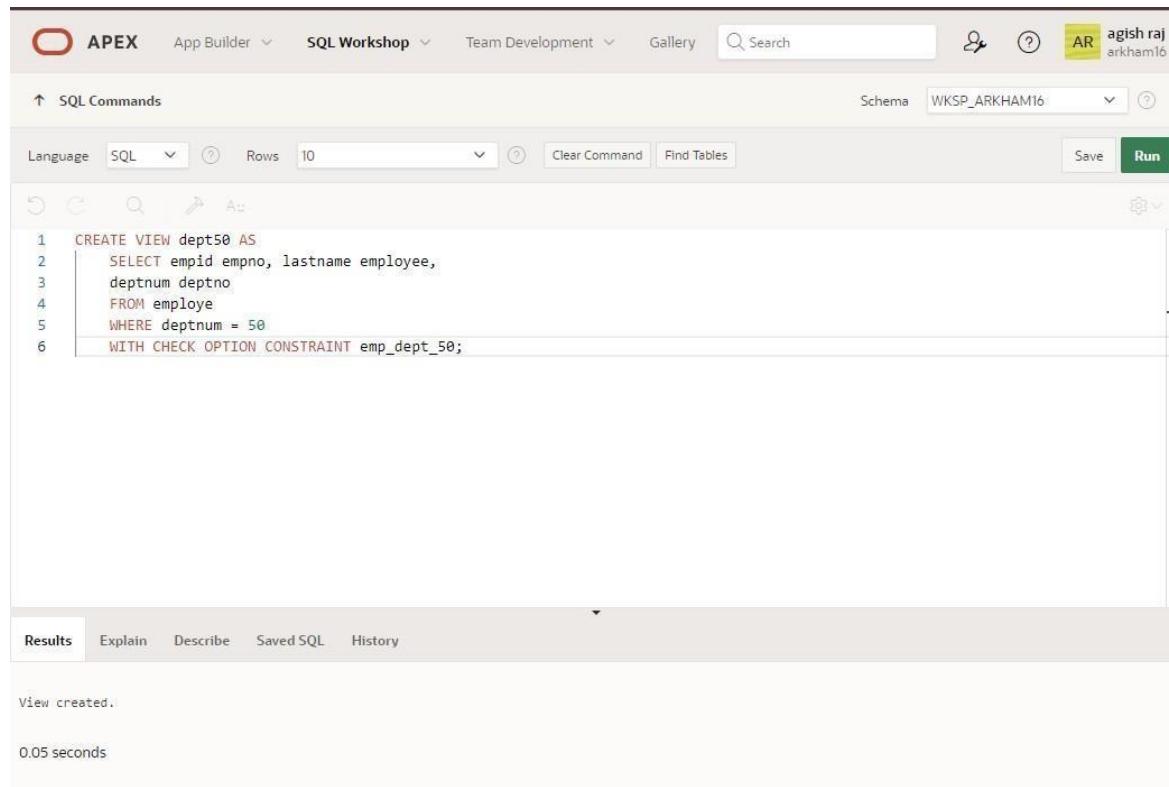
Below the table, it says '3 rows returned in 0.00 seconds' and has a 'Download' link.

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

**QUERY:**

```
create view dept50 as select empid empno, lastname employee,deptnum deptno from
employee where deptnum= 50 with check option constraint emp_dept_50;
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and user information 'agish raj arkham16'. Below the toolbar, the schema is set to 'WKSP\_ARKHAM16'. The main area is a SQL editor with the following code:

```
1 CREATE VIEW dept50 AS
2   SELECT empid empno, lastname employee,
3         deptnum deptno
4   FROM employee
5   WHERE deptnum = 50
6   WITH CHECK OPTION CONSTRAINT emp_dept_50;
```

Below the code, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, displaying the message 'View created.' and '0.05 seconds' execution time.

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

**QUERY:      Describe**

**dept50; select \* from**

**dept50;**

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. At the top, there's a toolbar with icons for App Builder, SQL Workshop, Team Development, and Gallery. A search bar is also present. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_ARKHAM16'. Below this, there are buttons for Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run. The command line shows '1 DESCRIBE dept50;'. The results pane is open, showing the 'Describe' tab selected. It displays the structure of the DEPT50 view with three columns: EMPNO (NUMBER), EMPLOYEE (VARCHAR2), and DEPTNO (NUMBER). The table structure is as follows:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPT50	EMPNO	NUMBER	22	-	-	-	✓	-	-
	EMPLOYEE	VARCHAR2	25	-	-	-	✓	-	-
	DEPTNO	NUMBER	22	-	-	-	✓	-	-

The screenshot shows the Oracle SQL Workshop interface again. The command line now contains '1 SELECT \* FROM dept50;'. The results pane shows the 'Results' tab selected, but it displays 'no data found'.

6. 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. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, a search bar, and a user profile for 'agish raj' (arkham16). The main workspace is titled 'SQL Commands'. It features a toolbar with icons for undo, redo, search, and refresh. Below the toolbar, the schema is set to 'WKSP\_ARKHAM16'. The SQL editor contains the following code:

```
1 update dept50 set deptno = 80
2 where employee = 'Matos';
3
```

The 'Run' button is highlighted in green. The results section at the bottom displays the output of the query:

0 row(s) updated.  
0.07 seconds

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

#### QUERY:

```
create view salary_vu as select e.lastname 'Employee', d.deptname 'Department', e.salary 'Salary', j.grade  
'Grade' from employee e ,department d ,grade j where e.deptnum=d.deptnum and e.salary between  
j.lowestsal and j.highestsal;
```

#### 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 is at the top right. The main area is titled "SQL Commands". The schema dropdown is set to "WKSP\_ARKHAM16". The code editor contains the following SQL command:

```
1 CREATE OR REPLACE VIEW salary_vu  
2 AS  
3 SELECT e.lastname "Employee",  
4       d.deptname "Department",  
5       e.salary "Salary",  
6       j.grade "Grades"  
7  FROM employee e,  
8       department d,  
9       grade j  
10 WHERE e.deptnum = d.deptno  
11   AND e.salary BETWEEN j.lowestsal and j.highestsal;
```

Below the code editor, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, displaying the message "View created." and "0.03 seconds".

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

**RESULT:**

# EXERCISE 12

## PRACTICE QUESTIONS

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Table Created.

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

DESCRIBE f\_global\_locations;

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

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

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

# PRIMARY KEY, FOREIGN KEY, and CHECK Constraints

1. What is the purpose of a

- PRIMARY KEY
- 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	

# PRACTICE PROBLEM

## Managing Constraints

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

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

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

- ADD
- DROP
- ENABLE
- DISABLE

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

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

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

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

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

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

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

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

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

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

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

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

INSERT INTO

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

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

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

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

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

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

**1 row(s) inserted.**

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

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

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

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

**1 row(s) deleted.**

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

**Table altered.**

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

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

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

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

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

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

# EXERCISE 13

## PRACTICE QUESTIONS

### Creating Views

1. What are three uses for a view from a DBA's perspective?
  - Restrict access and display selective columns
  - Reduce complexity of queries from other internal systems. So, providing a way to view same data in a different manner.
  - Let the app code rely on views and allow the internal implementation of tables to be modified later.
2. Create a simple view called view\_d\_songs that contains the ID, title and artist from the DJs on Demand table for each "New Age" type code. In the subquery, use the alias "Song Title" for the title column.

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

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

Results	Explain	Describe	Saved SQL	History
ID	Song Title		ARTIST	
47	Hurrah for Today		The Jubilant Trio	
49	Lets Celebrate		The Celebrants	
2 rows returned in 0.00 seconds			Download	

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

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

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

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

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

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

# DML Operations and Views

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

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

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

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

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

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

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

```
SELECT * FROM view_copy_d_songs;
```

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

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

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

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

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

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

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

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

**CREATE OR REPLACE VIEW read\_copy\_d\_cds AS**

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

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

**DELETE FROM read\_copy\_d\_cds**

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

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

**DELETE FROM read\_copy\_d\_cds**

```
WHERE cd_number = 90;
```

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

**DELETE FROM read\_copy\_d\_cds**

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

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

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

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

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

**Group functions**

**GROUP BY CLAUSE**

**DISTINCT**

**pseudocolumn ROWNUM Keyword**

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

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

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

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

# Managing Views

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

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

```
SELECT * FROM view_copy_d_songs;
```

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

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

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

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

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

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

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

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

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

# Indexes and Synonyms

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

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

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

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

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

3. When will an index be created automatically?

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

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

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

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

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

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

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

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

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

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

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

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

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

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

10. Drop the synonym that you created in question

**DROP SYNONYM dj\_tracks2;**

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

**RESULT:**

# OTHER DATABASE OBJECTS

**EX\_NO:14**

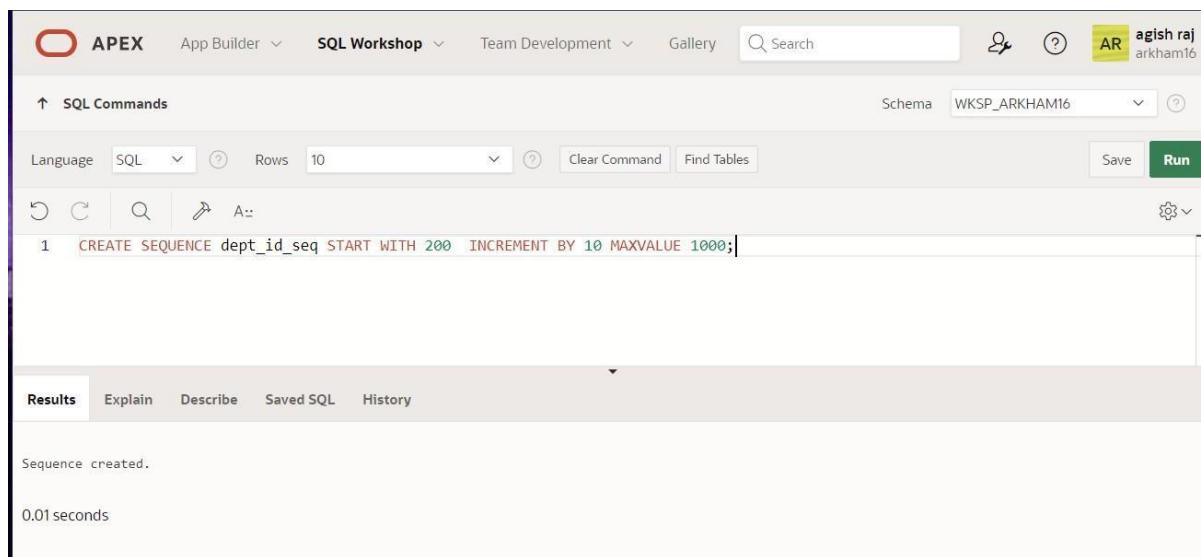
**DATE:**

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

**QUERY:**

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

**OUTPUT:**



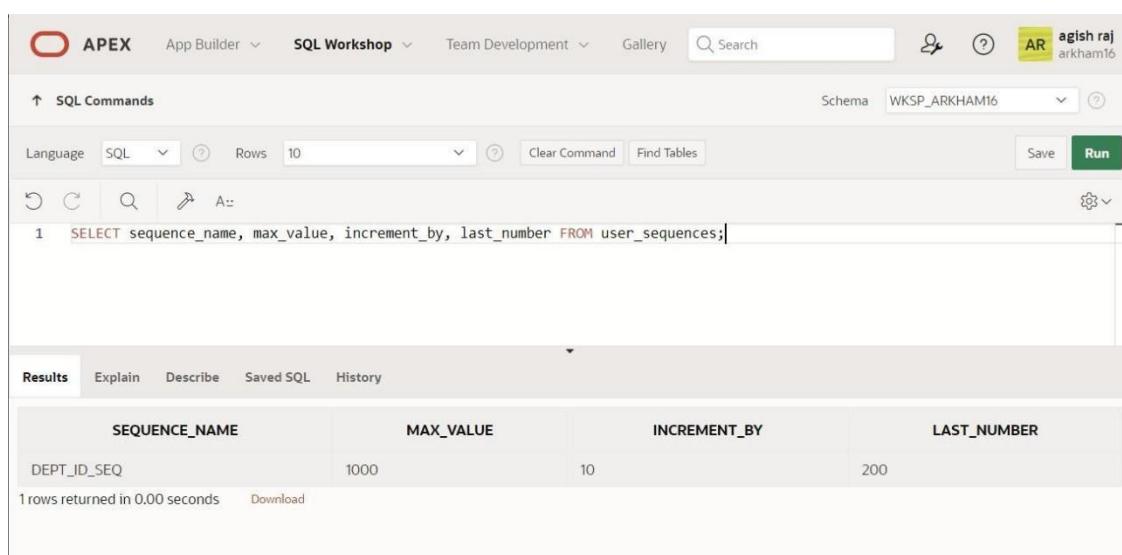
The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and a user profile 'agish raj arkham16'. The main area is titled 'SQL Commands'. It has tabs for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', 'Find Tables', 'Save', and a 'Run' button. Below these is a toolbar with icons for refresh, undo, redo, search, and edit. The command input field contains the SQL code: 'CREATE SEQUENCE dept\_id\_seq START WITH 200 INCREMENT BY 10 MAXVALUE 1000;'. The results section is titled 'Results' and displays the message 'Sequence created.' Below it, 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. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and a user profile 'agish raj arkham16'. The main area is titled 'SQL Commands'. It has tabs for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', 'Find Tables', 'Save', and a 'Run' button. Below these is a toolbar with icons for refresh, undo, redo, search, and edit. The command input field contains the SQL code: 'SELECT sequence\_name, max\_value, increment\_by, last\_number FROM user\_sequences;'. The results section is titled 'Results' and displays a table with four columns: 'SEQUENCE\_NAME', 'MAX\_VALUE', 'INCREMENT\_BY', and 'LAST\_NUMBER'. A single row is shown: 'DEPT\_ID\_SEQ' with values '1000', '10', and '200'. Below the table, it says '1 rows returned in 0.00 seconds'.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, Gallery, and a search bar. The user is signed in as 'agish raj' with the workspace 'arkham16'. The SQL Commands tab is active, showing the following command in the editor:

```
1 insert into dept values (dept_id_seq.nextval, 'Education');
```

Below the editor, the Results tab is selected, displaying the output:

```
1 row(s) inserted.  
0.02 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, Team Development, Gallery, and a search bar. The user is signed in as 'agish raj' with the workspace 'arkham16'. The SQL Commands tab is active, showing the following command in the editor:

```
1 CREATE INDEX emp_dept_id_idx ON EMPLOYEE (department_id);
```

Below the editor, the Results tab is selected, displaying the output:

```
Index created.  
0.07 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. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and a user profile 'agish raj arkham16'. The main workspace has a toolbar with icons for undo, redo, search, and find. Below the toolbar is a command input field containing the SQL query: 'SELECT index\_name,table\_name,uniqueness FROM user\_indexes WHERE table\_name='EMPLOYEE';'. The results tab is selected, displaying a table with three columns: INDEX\_NAME, TABLE\_NAME, and UNIQUENESS. The single row returned is 'EMP\_DEPT\_ID\_IDX', 'EMPLOYEE', and 'NONUNIQUE'. The status bar at the bottom indicates '1 rows returned in 0.08 seconds' and a 'Download' link.

INDEX_NAME	TABLE_NAME	UNIQUENESS
EMP_DEPT_ID_IDX	EMPLOYEE	NONUNIQUE

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

**RESULT:**

# **CONTROLLING USER ACCESS**

**EX\_NO:15**

**DATE:**

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

The CREATE SESSION system privilege

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

The CREATE TABLE privilege

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

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

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

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

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

The ALTER USER statement

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

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

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

7. Query all the rows in your DEPARTMENTS table.

SELECT \* FROM departments;

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

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

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

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

SELECT table\_name FROM user\_tables;

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

Team 1 revokes the privilege.

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

Team 2 revokes the privilege.

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

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

Team 1 executes this INSERT statement.

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

Team 2 executes this INSERT statement.

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

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

**RESULT:**

# PL/SQL

## CONTROL STRUCTURES

EXP.NO:16

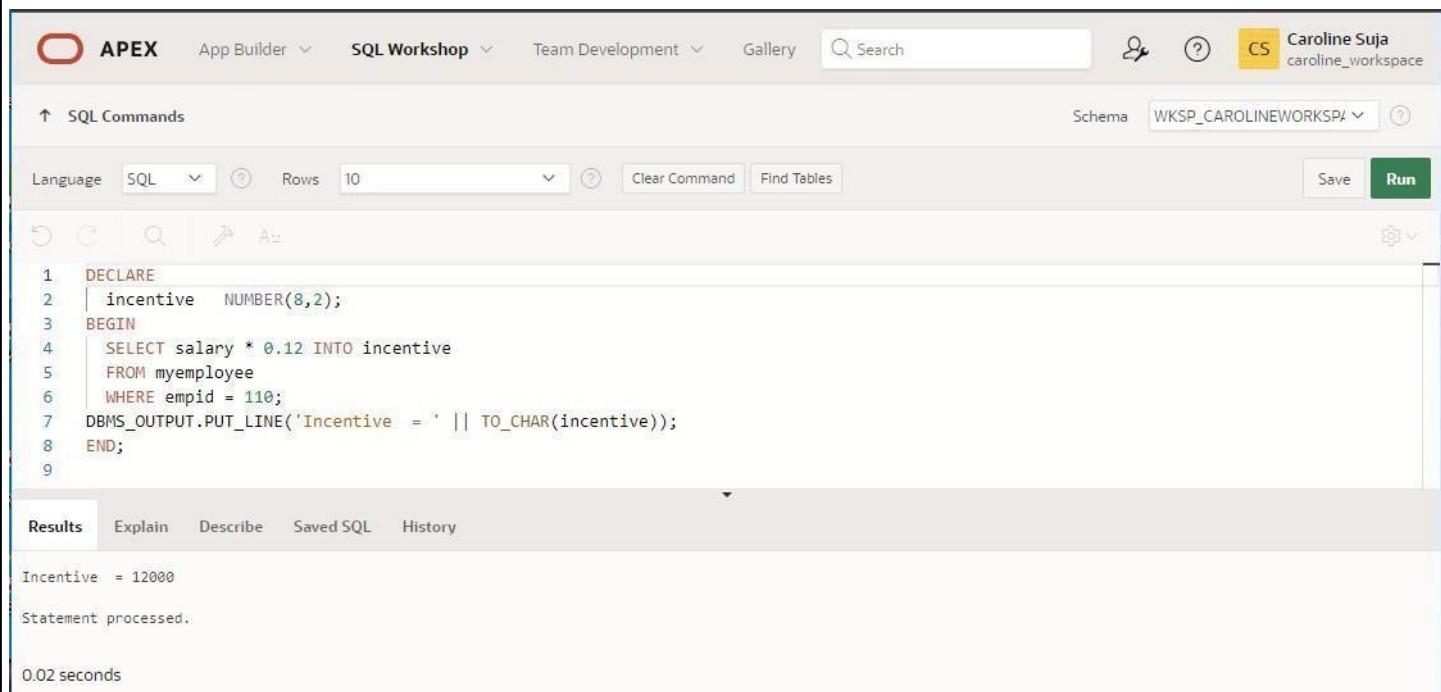
DATE:

- 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 SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, and a search bar. On the right, there's a user profile for 'Caroline Suja' and a workspace dropdown set to 'WKSP\_CAROLINEWORKSP4'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_CAROLINEWORKSP4'. Below the title are buttons for Language (SQL selected), Rows (10), Clear Command, Find Tables, Save, and Run. The code editor contains a PL/SQL block with line numbers 1 through 9. The code is as follows:

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

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

```
Incentive = 12000
Statement processed.

0.02 seconds
```

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

**QUERY:**

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

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

**OUTPUT:**

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

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

An error message is displayed in a yellow box:

```
Error at line 4/25: ORA-06550: line 4, column 25:
PLS-00201: identifier 'Welcome' must be declared
ORA-06512: at "SYS.WMV_DBMS_SQL_APEX_230200", line 801
ORA-06550: line 4, column 3:
PL/SQL: Statement ignored
```

Below the error message, the problematic lines of code are highlighted:

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

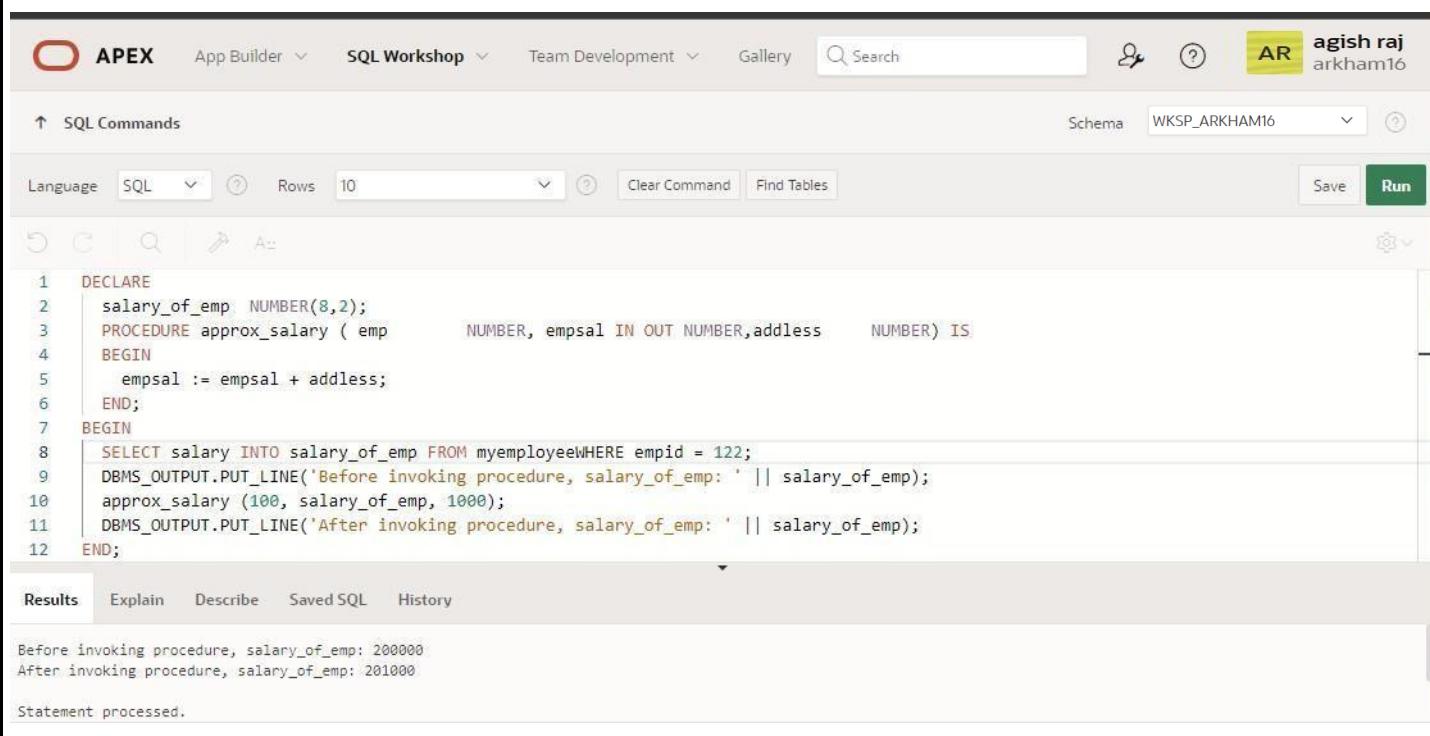
3.) Write a PL/SQL block to adjust the salary of the employee whose ID 122.

**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 Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, Gallery, and a search bar. The user's profile 'agish raj' and workspace 'arkham16' are also visible. The main workspace is titled 'SQL Commands'. The code area contains the PL/SQL block provided above. The results pane at the bottom displays the output of the DBMS\_OUTPUT.PUT\_LINE statements, showing the salary before and after the procedure call.

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

Results

```
Before invoking procedure, salary_of_emp: 200000
After invoking procedure, salary_of_emp: 201000

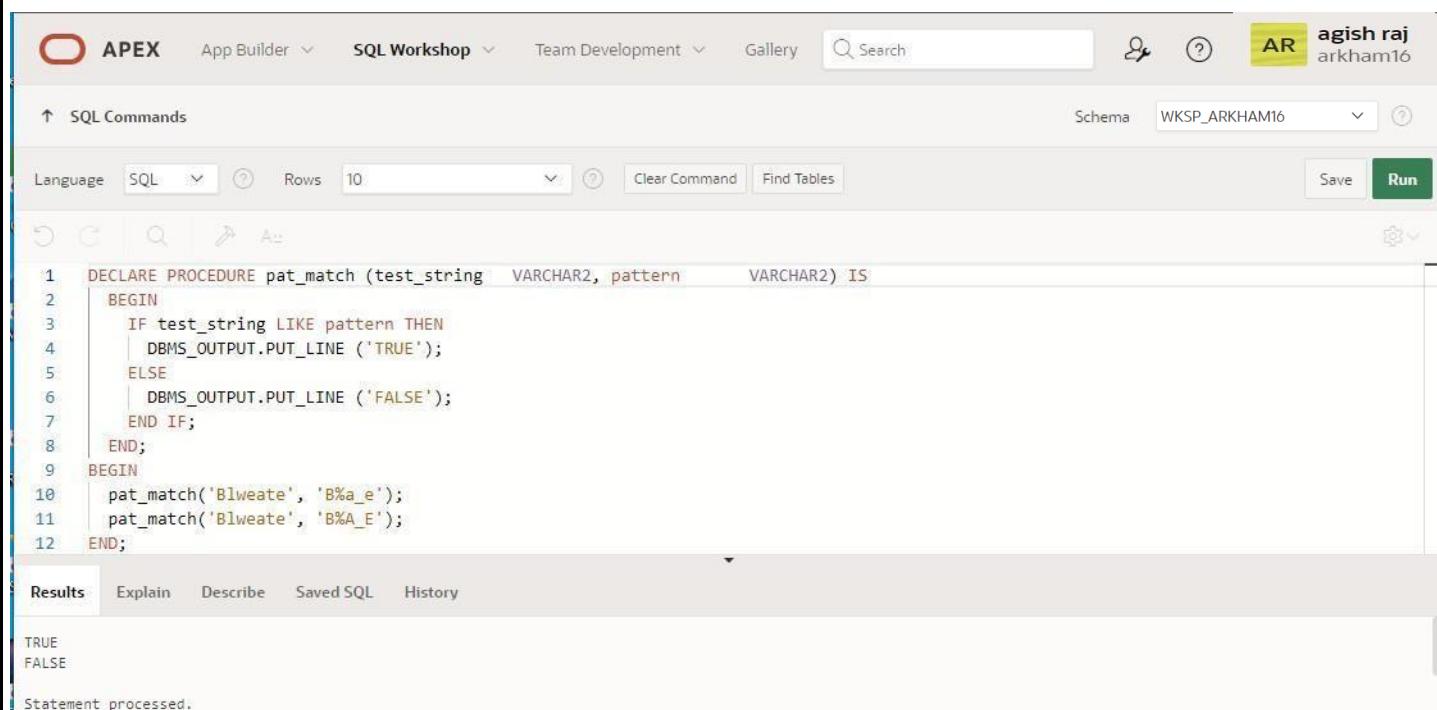
Statement processed.
```

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

#### QUERY:

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

#### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, and a search bar. On the right, there's a user profile for 'agish raj' and a schema dropdown set to 'WKSP\_ARKHAM16'. The main area is titled 'SQL Commands' and contains a code editor with the following PL/SQL code:

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

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

```
TRUE
FALSE
```

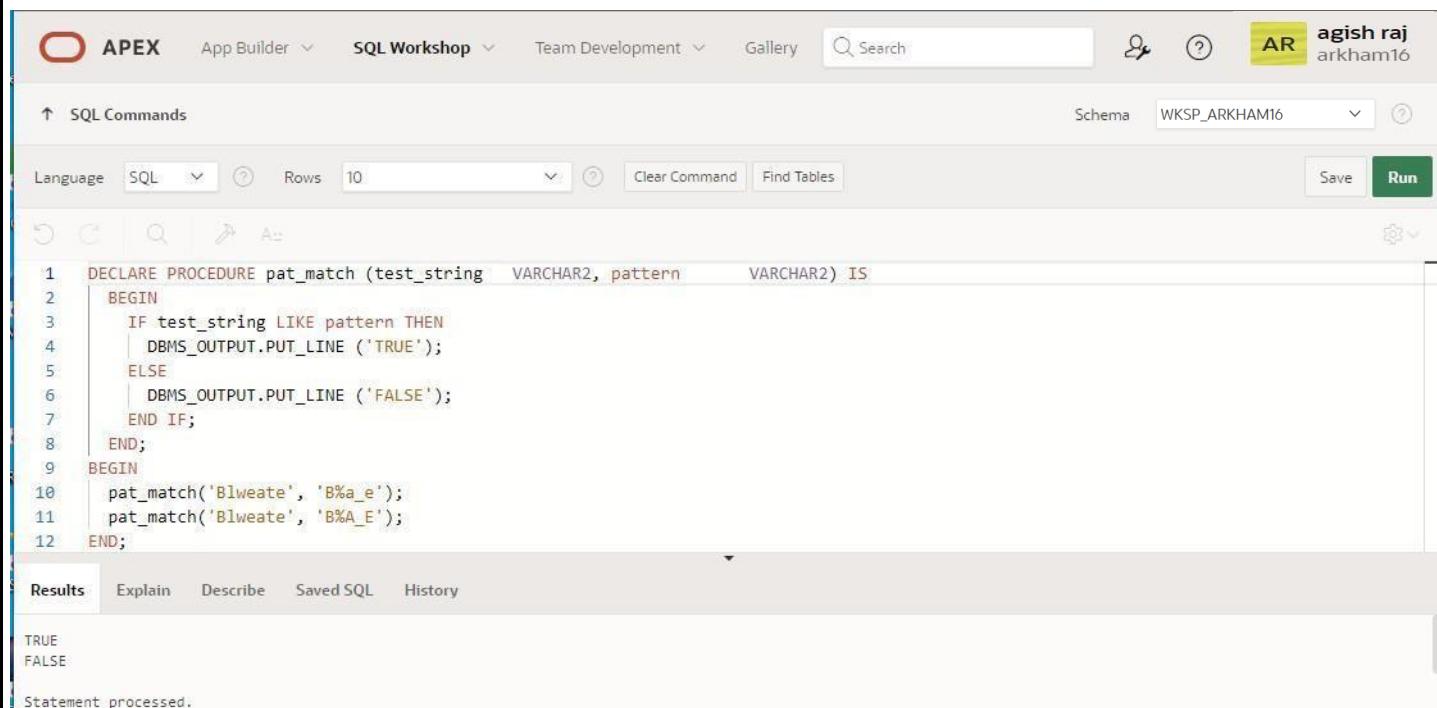
A status message at the bottom says 'Statement processed.'

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 SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, and a search bar. The right side shows a user profile for 'agish raj' with the identifier 'AR' and the schema 'WKSP\_ARKHAM16'. The main workspace displays the PL/SQL code in the SQL Commands editor. The code declares a procedure 'pat\_match' that checks if a 'test\_string' matches a 'pattern' using the LIKE operator. It uses DBMS\_OUTPUT.PUT\_LINE to print 'TRUE' or 'FALSE'. The code then calls this procedure with two patterns: 'B%a\_e' and 'B%A\_E'. The results are shown in the Results tab, where 'TRUE' is printed for the first call and 'FALSE' for the second. The status bar at the bottom indicates 'Statement processed.'

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

Results Explain Describe Saved SQL History

TRUE  
FALSE

Statement processed.

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

#### QUERY:

DECLARE

num\_small NUMBER := 8;

num\_large NUMBER := 5;

num\_temp NUMBER;

BEGIN

IF num\_small > num\_large THEN

num\_temp := num\_small;

num\_small := num\_large;

num\_large := num\_temp;

END IF;

DBMS\_OUTPUT.PUT\_LINE ('num\_small = '||num\_small);

DBMS\_OUTPUT.PUT\_LINE ('num\_large = '||num\_large);

END;

/

#### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, a search bar, and a user profile for agish raj arkham16.

In the SQL Commands tab, the code is displayed:

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

The Results tab shows the output:

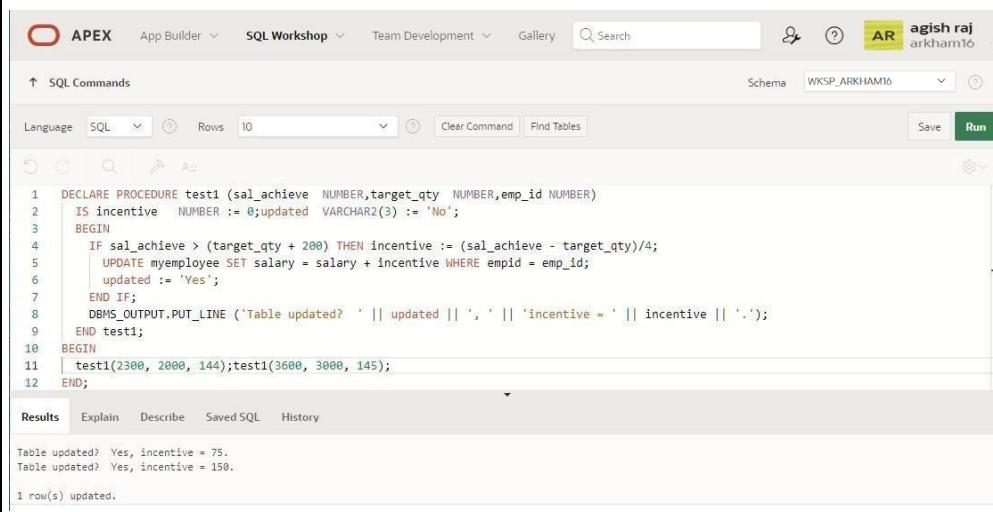
```
num_small = 5
num_large = 8
Statement processed.
```

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 SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', 'Gallery', a search bar, and a user icon 'agish raj arkham16'. The main workspace is titled 'SQL Commands' and contains the PL/SQL code from the previous step. The code is numbered from 1 to 12. Lines 10 through 12 show the execution of the 'test1' procedure with two sets of parameters. The results pane at the bottom displays the output of the DBMS\_OUTPUT.PUT\_LINE statements, showing 'Table updated? Yes, incentive = 75.' and 'Table updated? Yes, incentive = 150.'. A note at the bottom indicates '1 row(s) updated.'

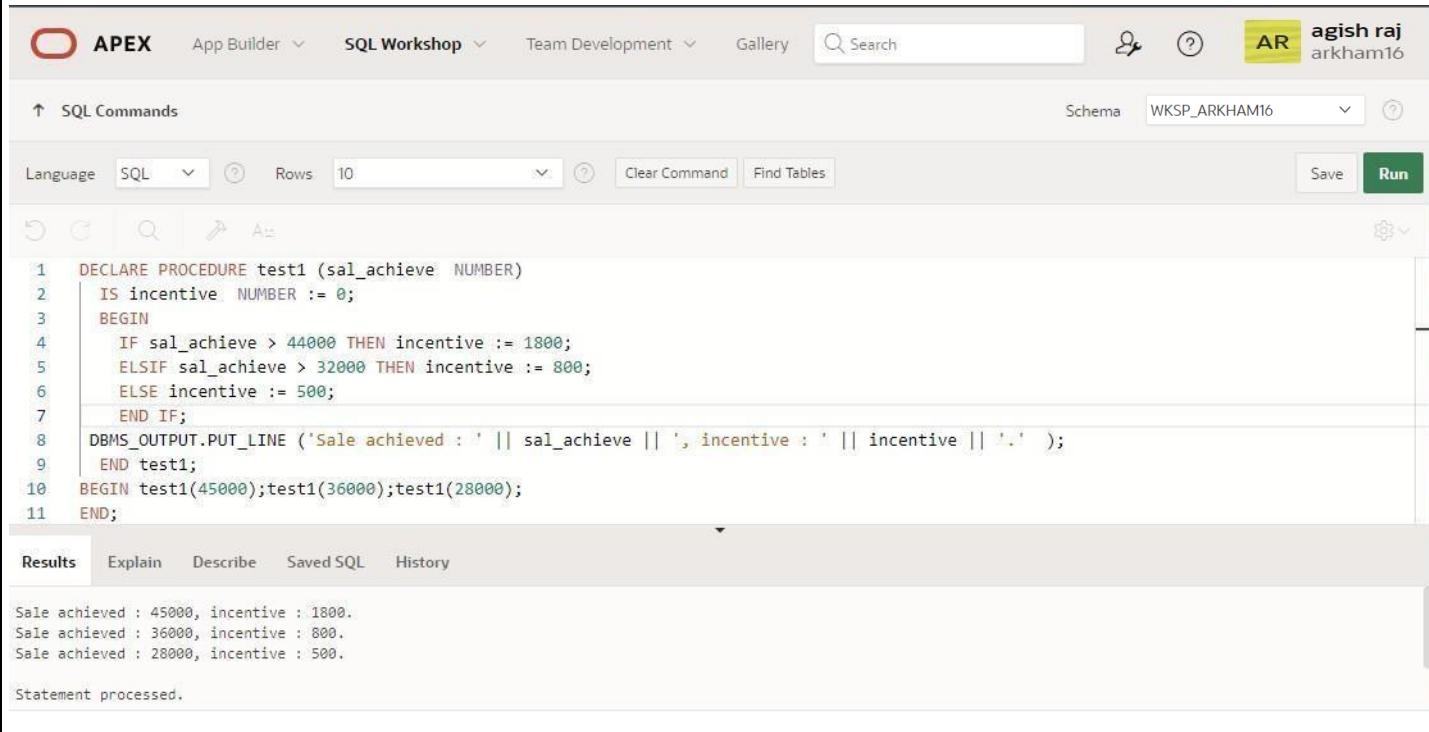
8.) Write a PL/SQL procedure to calculate incentive achieved according to the specific sale limit

**QUERY:**

DECLARE

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, a search bar, and user information for agish raj (arkham16). The main workspace is titled 'SQL Commands' and contains the PL/SQL code from the previous block. The code is numbered 1 to 11. The 'Run' button is visible at the bottom right of the command input area. Below the code, the 'Results' tab is selected, displaying the output of the executed procedure calls. The output shows three rows of results: 'Sale achieved : 45000, incentive : 1800.', 'Sale achieved : 36000, incentive : 800.', and 'Sale achieved : 28000, incentive : 500.'. A message 'Statement processed.' is also present at the bottom.

```
1  DECLARE PROCEDURE test1 (sal_achieve NUMBER)
2  IS incentive NUMBER := 0;
3  BEGIN
4      IF sal_achieve > 44000 THEN incentive := 1800;
5      ELSIF sal_achieve > 32000 THEN incentive := 800;
6      ELSE incentive := 500;
7      END IF;
8      DBMS_OUTPUT.PUT_LINE ('Sale achieved : ' || sal_achieve || ', incentive : ' || incentive || '.');
9      END test1;
10 BEGIN test1(45000);test1(36000);test1(28000);
11 END;
```

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

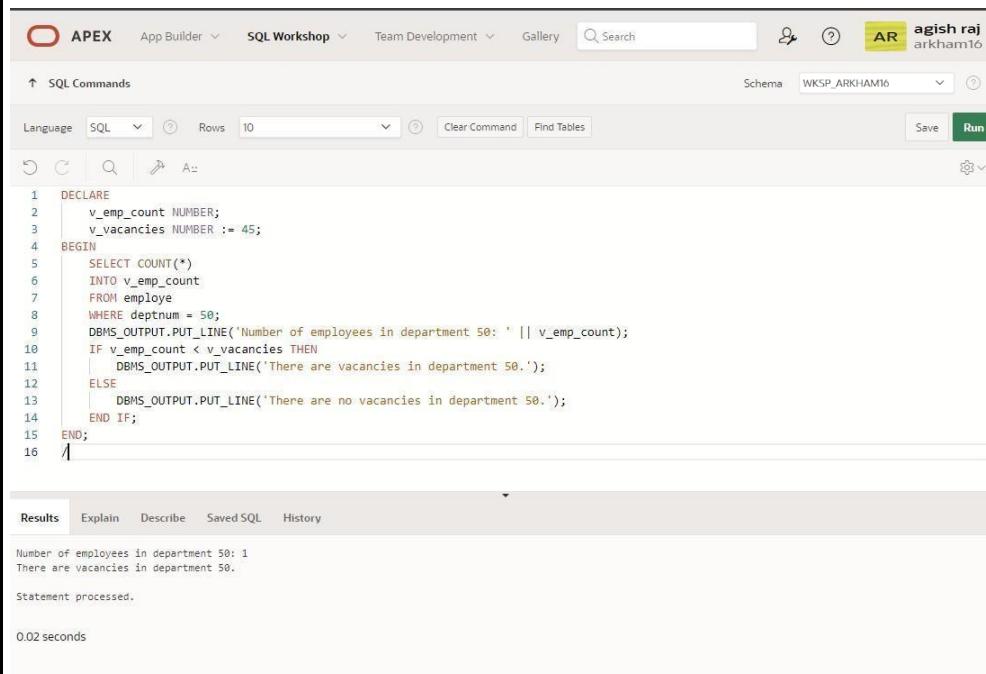
9.) Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

#### 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 SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, Gallery, and a search bar. The user is signed in as 'agish raj' with the schema 'WKSP\_ARKHAM16'. The main workspace is titled 'SQL Commands' and contains the following PL/SQL code:

```
1  DECLARE
2      v_emp_count NUMBER;
3      v_vacancies NUMBER := 45;
4  BEGIN
5      SELECT COUNT(*)
6          INTO v_emp_count
7          FROM employee
8          WHERE department = 50;
9      DBMS_OUTPUT.PUT_LINE('Number of employees in department 50: '|| v_emp_count);
10     IF v_emp_count < v_vacancies THEN
11         DBMS_OUTPUT.PUT_LINE('There are vacancies in department 50.');
12     ELSE
13         DBMS_OUTPUT.PUT_LINE('There are no vacancies in department 50.');
14     END IF;
15  END;
16  /
```

The results tab at the bottom displays the output of the program:

```
Number of employees in department 50: 1
There are vacancies in department 50.

Statement processed.

0.02 seconds
```

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

**QUERY:**

DECLARE

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

BEGIN

```
get_dep_id := 80;
```

```
SELECT Count(*)
```

```
INTO tot_emp
```

```
FROM employees e
```

```
join departments d
```

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

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

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

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

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

```
ELSE
```

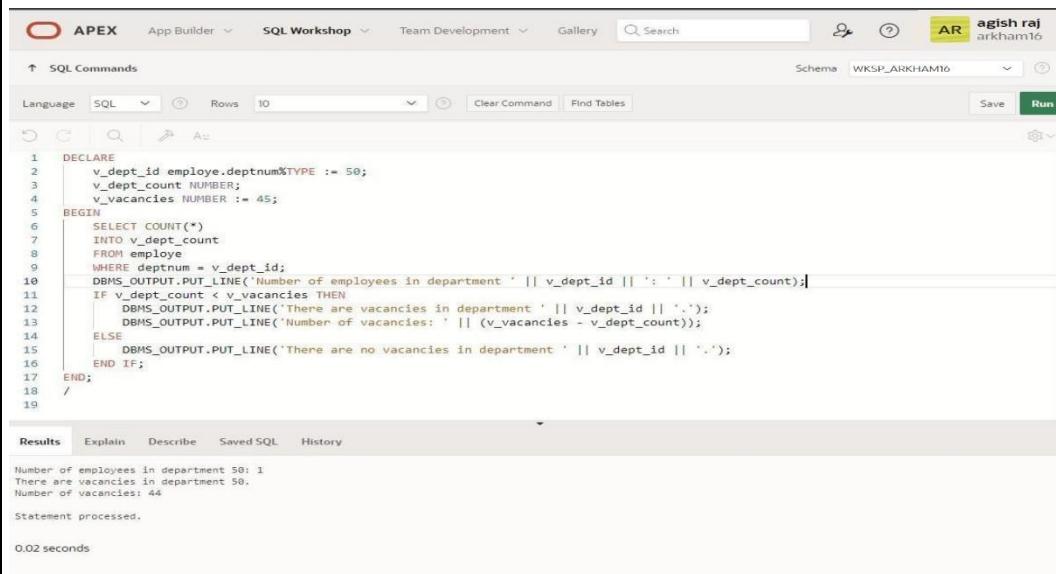
```
dbms_output.Put_line ('There are '| |to_char(45-tot_emp)| |' vacancies in department '| |
get_dep_id );
```

```
END IF;
```

```
END;
```

```
/
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface with the following details:

- Toolbar:** APEX, App Builder, SQL Workshop, Team Development, Gallery, Search, Schema (WKSP\_ARKHAM16), Run.
- Language:** SQL, Rows: 10.
- Code Area:** The code block is pasted above. It declares variables for department ID, count, and vacancies, then performs a SELECT INTO statement to get employee counts for department 80. It then uses DBMS\_OUTPUT.PUT\_LINE to print the employee count and a conditional block to determine if there are vacancies. The output shows the employee count and a message indicating there are no vacancies.
- Results Tab:** Shows the output of the executed code.

```
1  DECLARE
2      v_dept_id employees.deptnum%TYPE := 50;
3      v_dept_count NUMBER;
4      v_vacancies NUMBER := 45;
5  BEGIN
6      SELECT COUNT(*)
7          INTO v_dept_count
8         FROM employee
9        WHERE deptnum = v_dept_id;
10     DBMS_OUTPUT.PUT_LINE('Number of employees in department '| | v_dept_id | | ': '| | v_dept_count);
11     IF v_dept_count < v_vacancies THEN
12         DBMS_OUTPUT.PUT_LINE('There are vacancies in department '| | v_dept_id | | '.');
13         DBMS_OUTPUT.PUT_LINE('Number of vacancies: '| | (v_vacancies - v_dept_count));
14     ELSE
15         DBMS_OUTPUT.PUT_LINE('There are no vacancies in department '| | v_dept_id | | '.');
16     END IF;
17  END;
18  /
```

Results:

```
Number of employees in department 50: 1
There are vacancies in department 50.
Number of vacancies: 44
Statement processed.
0.02 seconds
```

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

**QUERY:**

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

**OUTPUT:**

```
APEX App Builder SQL Workshop Team Development Gallery Search AR agish raj arkham16
SQL Commands Language SQL Rows 10 Clear Command Find Tables Schema WKSP_ARKHAM16 Save Run
1  DECLARE
2    CURSOR employee_cursor IS
3      SELECT empid, firstname || ' ' || lastname AS full_name, jobid, hiredate, salary
4      FROM employee;
5  BEGIN
6    FOR employee_rec IN employee_cursor LOOP
7      DBMS_OUTPUT.PUT_LINE('Employee ID: ' || employee_rec.empid);
8      DBMS_OUTPUT.PUT_LINE('Employee Name: ' || employee_rec.full_name);
9      DBMS_OUTPUT.PUT_LINE('Job Title: ' || employee_rec.jobid);
10     DBMS_OUTPUT.PUT_LINE('Hire Date: ' || TO_CHAR(employee_rec.hiredate, 'DD-MON-YYYY'));
11     DBMS_OUTPUT.PUT_LINE('Salary: ' || employee_rec.salary);
12     DBMS_OUTPUT.PUT_LINE('-----');
13   END LOOP;
14 END;
15 /
16

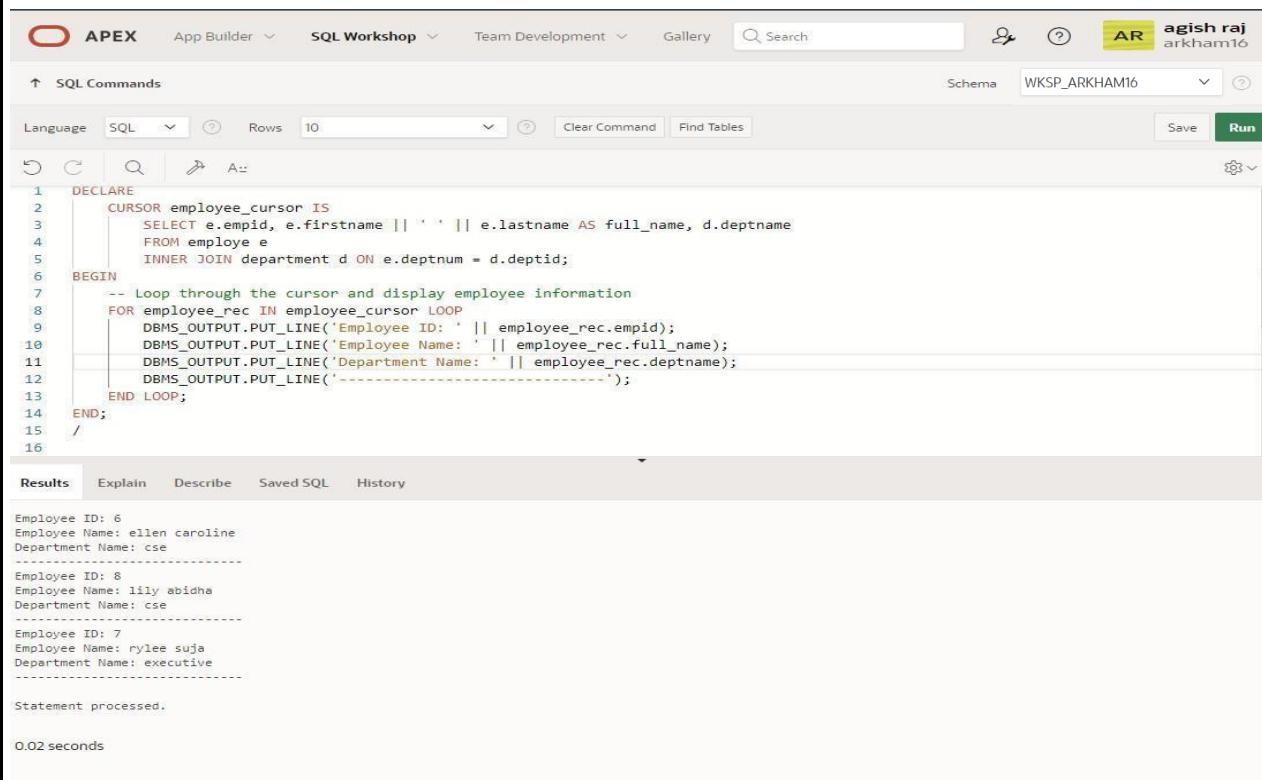
Results Explain Describe Saved SQL History
Employee ID: 6
Employee Name: elen caroline
Job Title: st_clerk
Hire Date: 03-APR-2004
Salary: 70000
-----
Employee ID: 7
Employee Name: lily abidha
Job Title: st_clerk
Hire Date: 04-MAY-1998
Salary: 50000
-----
Employee ID: 7
Employee Name: rylee suja
Job Title: ast_clerk
Hire Date: 17-MAY-2002
Salary: 75000
-----
Statement processed.
```

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

#### QUERY:

```
DECLARE
CURSOR emp_cursor IS
  SELECT e.employee_id, e.first_name, m.first_name AS manager_name
  FROM employees e
  LEFT JOIN employees m ON e.manager_id = m.employee_id;
emp_record emp_cursor%ROWTYPE;
BEGIN
  OPEN emp_cursor;
  FETCH emp_cursor INTO emp_record;
  WHILE emp_cursor%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp_record.employee_id);
    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_record.first_name);
    DBMS_OUTPUT.PUT_LINE('Manager Name: ' || emp_record.manager_name);
    DBMS_OUTPUT.PUT_LINE('-----');
    FETCH emp_cursor INTO emp_record;
  END LOOP;
  CLOSE emp_cursor;
END;
/
```

#### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, and a search bar. The right side shows a user profile for 'agish raj' and the schema 'WKSP\_ARKHAM16'. The main area displays a PL/SQL block with numbered lines. Lines 1 through 15 show the declaration of a cursor, opening it, and performing a loop to output employee information. Line 16 ends the block with a slash. Below the code, the 'Results' tab is selected, showing the output of the executed query. The output lists three employees with their details: Ellen Caroline, Iily Abidna, and Rylee Suja, each followed by a separator line. At the bottom, it says 'Statement processed.' and '0.02 seconds'.

```
1  DECLARE
2    CURSOR employee_cursor IS
3      SELECT e.empid, e.firstname || ' ' || e.lastname AS full_name, d.deptname
4      FROM employee e
5      INNER JOIN department d ON e.deptnum = d.deptid;
6  BEGIN
7    -- Loop through the cursor and display employee information
8    FOR employee_rec IN employee_cursor LOOP
9      DBMS_OUTPUT.PUT_LINE('Employee ID: ' || employee_rec.empid);
10     DBMS_OUTPUT.PUT_LINE('Employee Name: ' || employee_rec.full_name);
11     DBMS_OUTPUT.PUT_LINE('Department Name: ' || employee_rec.deptname);
12     DBMS_OUTPUT.PUT_LINE('-----');
13   END LOOP;
14 END;
15 /
16
```

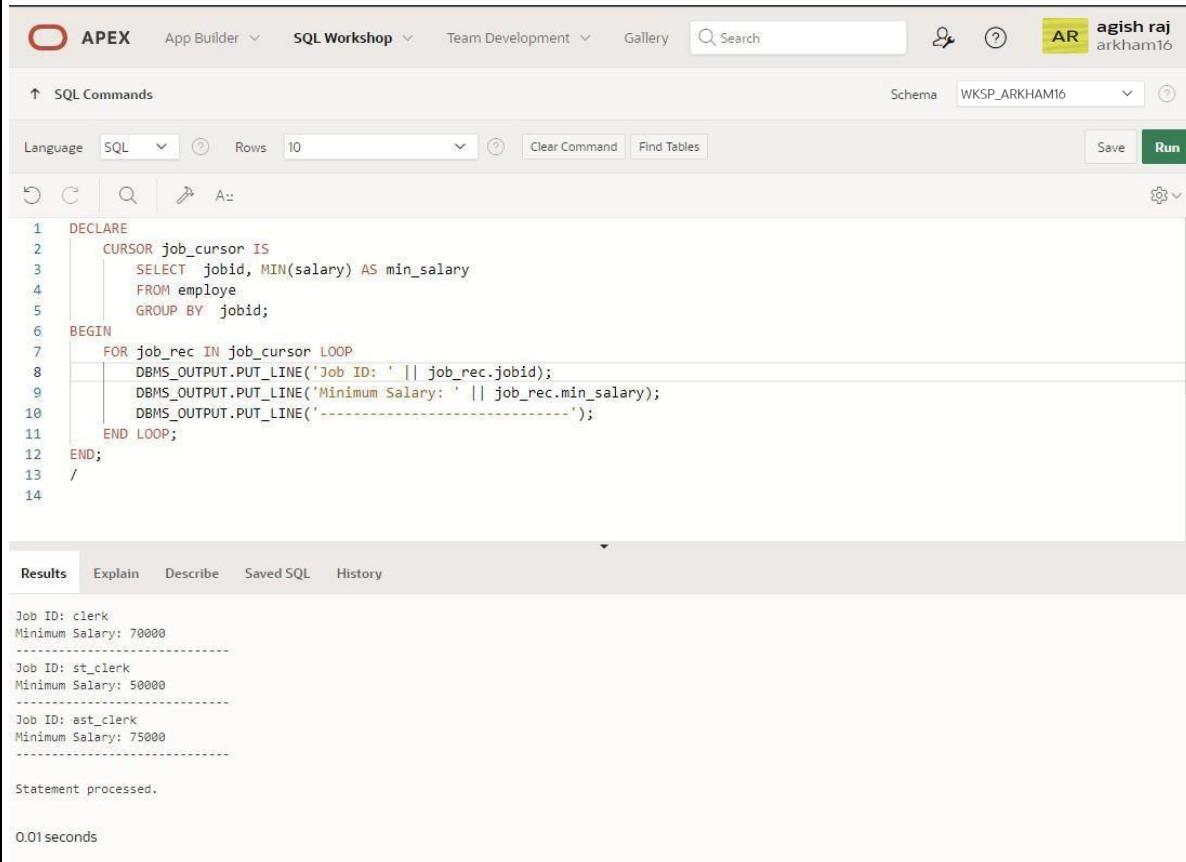
Employee ID: 6  
Employee Name: ellen caroline  
Department Name: cse  
-----  
Employee ID: 8  
Employee Name: iily abidna  
Department Name: cse  
-----  
Employee ID: 7  
Employee Name: rylee suja  
Department Name: executive  
-----  
Statement processed.  
0.02 seconds

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. In the top navigation bar, 'APEX' is selected. The main area displays the PL/SQL code from the previous step. The code uses a cursor to select job IDs and their minimum salaries from the employees and job\_grade tables, then loops through the results using DBMS\_OUTPUT.PUT\_LINE to print the job ID and minimum salary for each job. The output window at the bottom shows the resulting data.

```
1  DECLARE
2  CURSOR job_cursor IS
3    SELECT jobid, MIN(salary) AS min_salary
4      FROM employee
5     GROUP BY jobid;
6  BEGIN
7    FOR job_rec IN job_cursor LOOP
8      DBMS_OUTPUT.PUT_LINE('Job ID: ' || job_rec.jobid);
9      DBMS_OUTPUT.PUT_LINE('Minimum Salary: ' || job_rec.min_salary);
10     DBMS_OUTPUT.PUT_LINE('-----');
11   END LOOP;
12 END;
13 /
```

**Results**

```
Job ID: clerk
Minimum Salary: 70000
-----
Job ID: st_clerk
Minimum Salary: 50000
-----
Job ID: ast_clerk
Minimum Salary: 75000
-----
Statement processed.

0.01 seconds
```

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

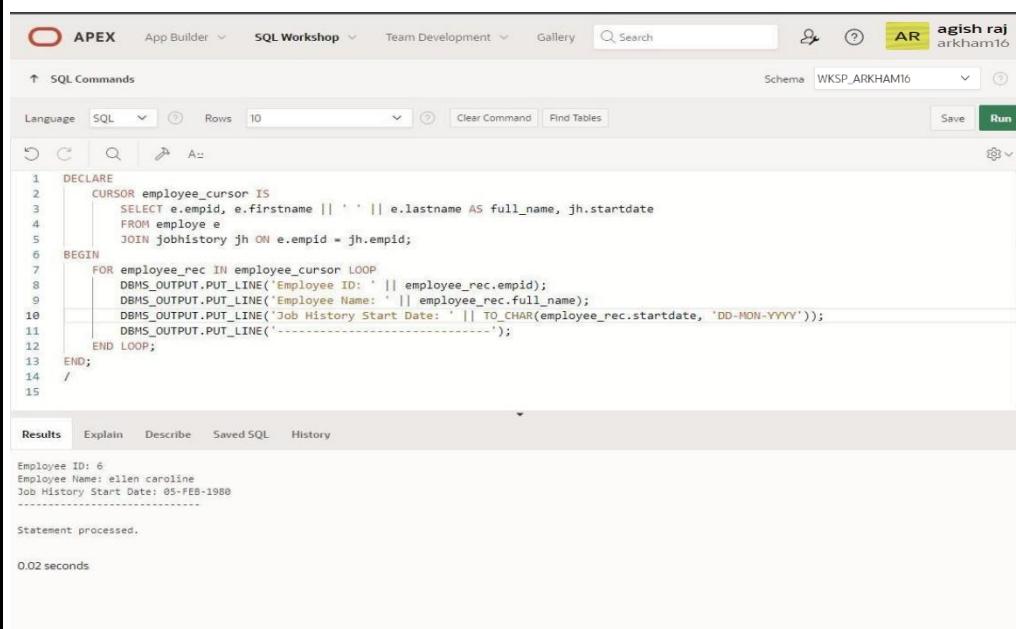
**QUERY:**

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', 'Gallery', 'Search', and a user profile 'agish raj'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_ARKHAM16'. The code editor contains a PL/SQL block with numbered lines from 1 to 15. Lines 1-5 define a cursor 'employees\_cursor' that joins 'employees' and 'job\_history' tables. Lines 6-14 form a loop for each record in the cursor. Inside the loop, lines 7-11 output the employee ID, name, and job history start date using DBMS\_OUTPUT.PUT\_LINE. Line 12 ends the loop, line 13 ends the block, and line 14 ends the declaration. Line 15 is a blank line. Below the code editor is a results pane showing the output: 'Employee ID: 6 Employee Name: ellen caroline Job History Start Date: 05-FEB-1988'. At the bottom, it says 'Statement processed.' and '0.02 seconds'.

```
1  DECLARE
2    CURSOR employee_cursor IS
3      SELECT e.empid, e.firstname || ' ' || e.lastname AS full_name, jh.startdate
4      FROM employee e
5      JOIN jobhistory jh ON e.empid = jh.empid;
6  BEGIN
7    FOR employee_rec IN employee_cursor LOOP
8      DBMS_OUTPUT.PUT_LINE('Employee ID: ' || employee_rec.empid);
9      DBMS_OUTPUT.PUT_LINE('Employee Name: ' || employee_rec.full_name);
10     DBMS_OUTPUT.PUT_LINE('Job History Start Date: ' || TO_CHAR(employee_rec.startdate, 'DD-MON-YYYY'));
11     DBMS_OUTPUT.PUT_LINE('-----');
12   END LOOP;
13 END;
14 /
15
```

Results Explain Describe Saved SQL History

Employee ID: 6  
Employee Name: ellen caroline  
Job History Start Date: 05-FEB-1988  
-----  
Statement processed.  
0.02 seconds.

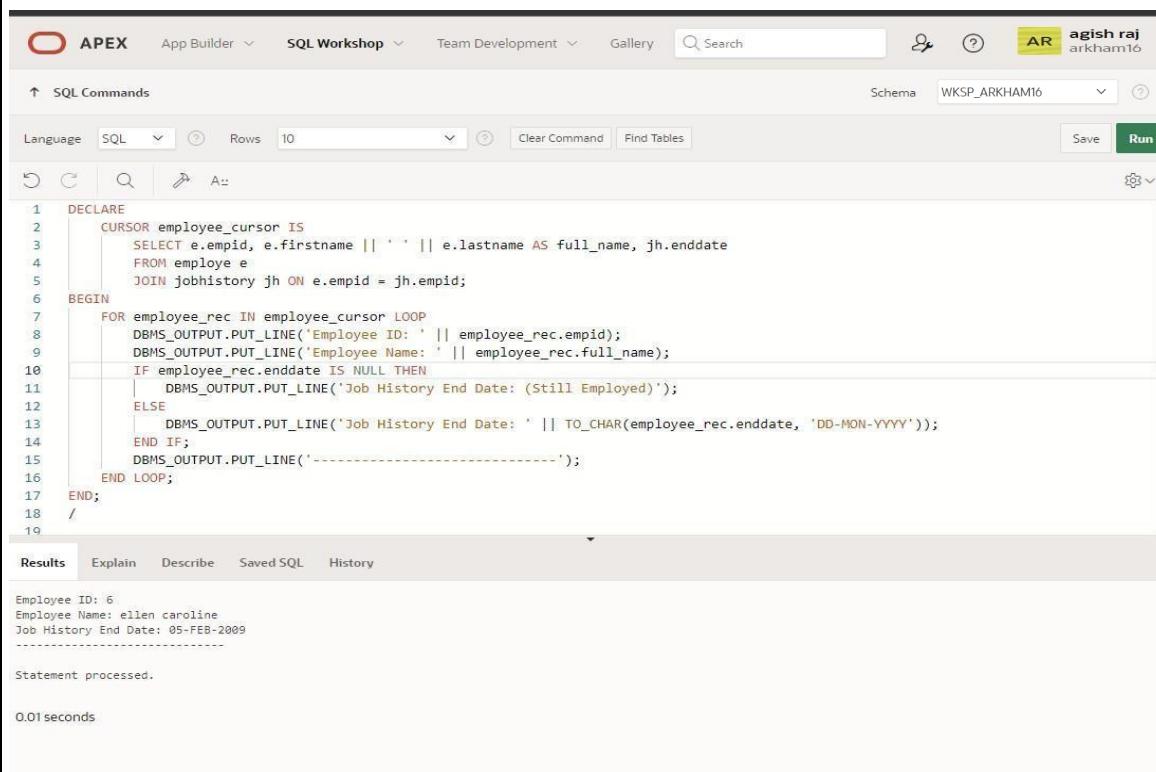
15.) Write a PL/SQL program to display the employee IDs, names, and job history end dates of all employees.

**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', 'Gallery', 'Search', and a user profile 'agish raj arkham16'. The main workspace is titled 'SQL Commands' and contains the following PL/SQL code:

```
1  DECLARE
2    CURSOR employee_cursor IS
3      SELECT e.empid, e.firstname || ' ' || e.lastname AS full_name, jh.enddate
4        FROM employee e
5       JOIN jobhistory jh ON e.empid = jh.empid;
6  BEGIN
7    FOR employee_rec IN employee_cursor LOOP
8      DBMS_OUTPUT.PUT_LINE('Employee ID: ' || employee_rec.empid);
9      DBMS_OUTPUT.PUT_LINE('Employee Name: ' || employee_rec.full_name);
10     IF employee_rec.enddate IS NULL THEN
11       DBMS_OUTPUT.PUT_LINE('Job History End Date: (Still Employed)');
12     ELSE
13       DBMS_OUTPUT.PUT_LINE('Job History End Date: ' || TO_CHAR(employee_rec.enddate, 'DD-MON-YYYY'));
14     END IF;
15     DBMS_OUTPUT.PUT_LINE('-----');
16   END LOOP;
17 END;
18 /
19
```

The 'Results' tab at the bottom shows the output of the executed code:

```
Employee ID: 6
Employee Name: ellen caroline
Job History End Date: 05-FEB-2009
-----
Statement processed.

0.01 seconds
```

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, Gallery, a search bar, and a user profile for 'agish raj arkham16'. The main workspace is titled 'SQL Commands' and shows the following PL/SQL code:

```
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;
```

The code is executed, and the results pane at the bottom displays the output:

```
120
Statement processed.
```

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. Below it, tabs for 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery' are visible. On the right side of the interface, there are buttons for 'Search', 'Save', 'Run', and a schema dropdown set to 'INVENTORY'. The main area is titled 'SQL Commands' and contains the following PL/SQL code:

```
56  DECLARE
57      v_book_id NUMBER := 1; -- Example book ID
58      v_title VARCHAR2(100);
59      v_author VARCHAR2(100);
60      v_year_published NUMBER;
61  BEGIN
62      -- Call the procedure
63      get_book_info(v_book_id, v_title, v_author, v_year_published);
64
65      -- Display the retrieved information
66      IF v_title IS NOT NULL THEN
67          DBMS_OUTPUT.PUT_LINE('Title: ' || v_title);
68          DBMS_OUTPUT.PUT_LINE('Author: ' || v_author);
69          DBMS_OUTPUT.PUT_LINE('Year Published: ' || v_year_published);
70      ELSE
71          DBMS_OUTPUT.PUT_LINE('Book information could not be retrieved.');
72      END IF;
73  END;
```

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

```
Title: 1984
Author: George Orwell
Year Published: 1940
Statement processed.
```

At the bottom left, it says '0.02 seconds'.

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

## RESULT:

# TRIGGER

EX\_NO: 18

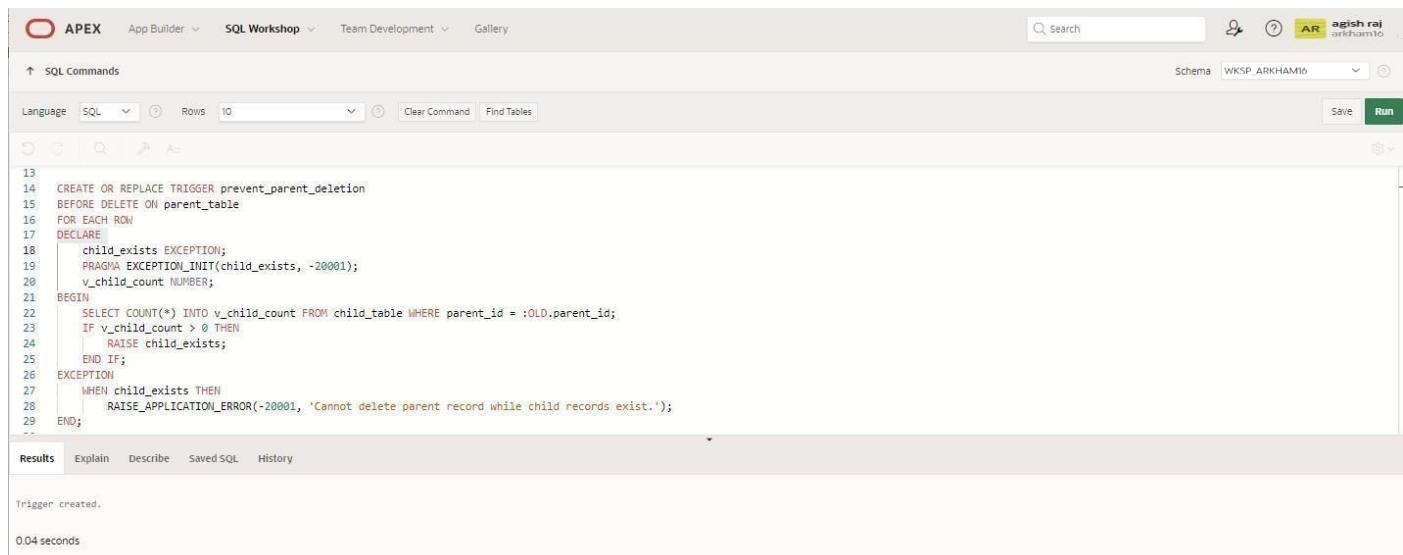
DATE:

- 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. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The main area displays the PL/SQL code for the trigger. The code is identical to the one provided in the question, including the exception handling and the raise\_application\_error call. Below the code, the results pane shows the message "Trigger created." and a execution time of "0.04 seconds". The schema is set to WKSP.ARKHAM16.

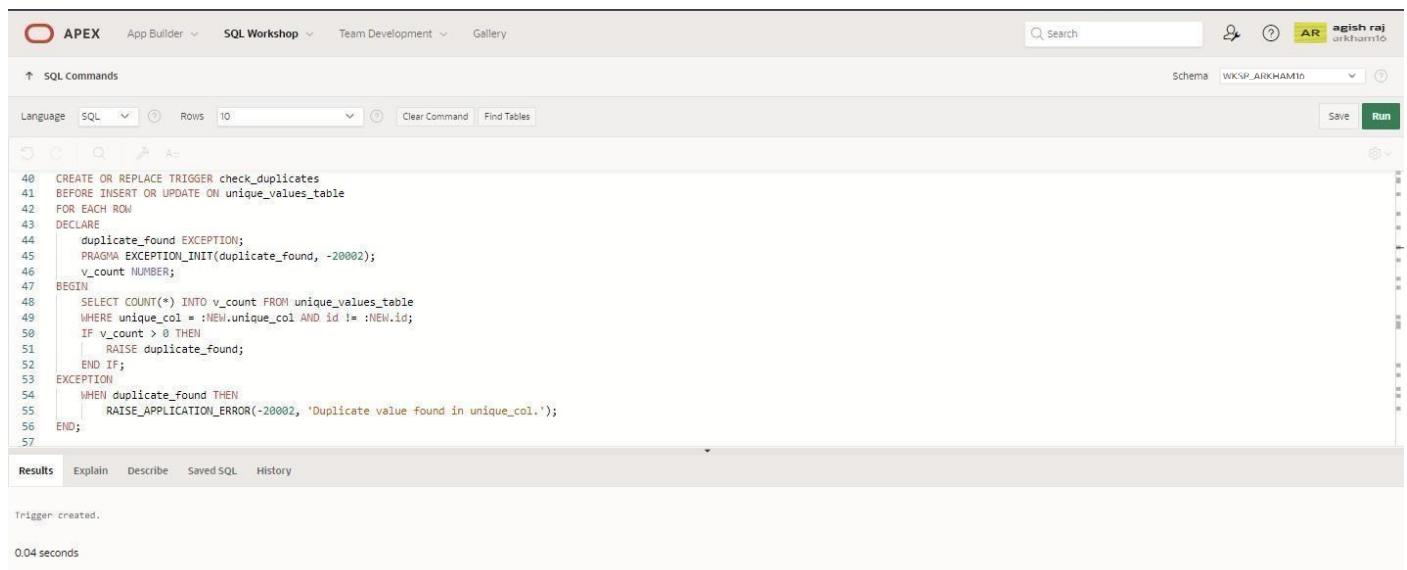
```
13
14 CREATE OR REPLACE TRIGGER prevent_parent_deletion
15 BEFORE DELETE ON parent_table
16 FOR EACH ROW
17 DECLARE
18     child_exists EXCEPTION;
19     PRAGMA EXCEPTION_INIT(child_exists, -20001);
20     v_child_count NUMBER;
21 BEGIN
22     SELECT COUNT(*) INTO v_child_count FROM child_table WHERE parent_id =
:OLD.parent_id;
23     IF v_child_count > 0 THEN
24         RAISE child_exists;
25     END IF;
26 EXCEPTION
27     WHEN child_exists THEN
28         RAISE_APPLICATION_ERROR(-20001, 'Cannot delete parent record while child
records exist.');
29 END;
```

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, Team Development, and Gallery. The user is signed in as agish raj with the schema WKSP.ARKHAM10. The main area is titled "SQL Commands" and contains the PL/SQL code for the trigger. The code is highlighted in blue and black. The bottom section shows the "Results" tab with the message "Trigger created." and a timestamp of "0.04 seconds".

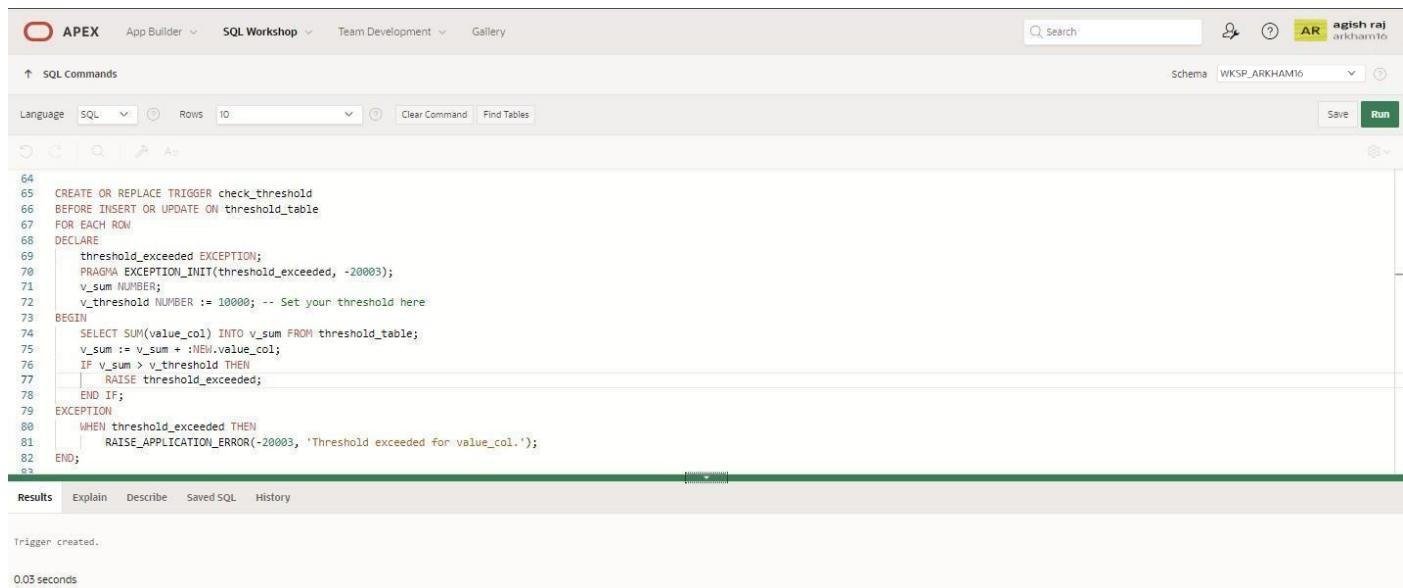
```
40 CREATE OR REPLACE TRIGGER check_duplicates
41 BEFORE INSERT OR UPDATE ON unique_values_table
42 FOR EACH ROW
43 DECLARE
44     duplicate_found EXCEPTION;
45     PRAGMA EXCEPTION_INIT(duplicate_found, -20002);
46     v_count NUMBER;
47 BEGIN
48     SELECT COUNT(*) INTO v_count FROM unique_values_table
49     WHERE unique_col = :NEW.unique_col AND id != :NEW.id;
50     IF v_count > 0 THEN
51         RAISE duplicate_found;
52     END IF;
53 EXCEPTION
54     WHEN duplicate_found THEN
55         RAISE_APPLICATION_ERROR(-20002, 'Duplicate value found in unique_col.');
56 END;
57
```

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 PL/SQL code for the trigger. The code is highlighted in red and blue, indicating syntax. The schema is set to 'WKSP.ARKHAM16'. The bottom status bar shows 'Trigger created.' and '0.03 seconds'.

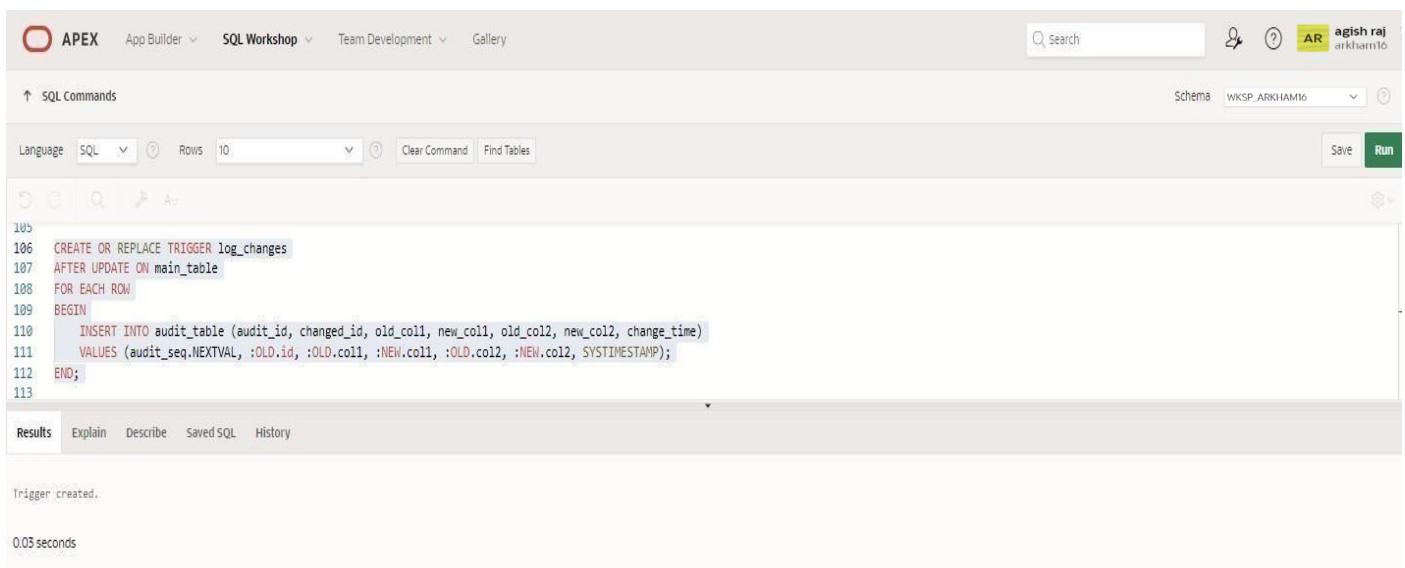
```
64 CREATE OR REPLACE TRIGGER check_threshold
65   BEFORE INSERT OR UPDATE ON threshold_table
66   FOR EACH ROW
67   DECLARE
68     threshold_exceeded EXCEPTION;
69     PRAGMA EXCEPTION_INIT(threshold_exceeded, -20003);
70     v_sum NUMBER;
71     v_threshold NUMBER := 10000; -- Set your threshold here
72 BEGIN
73   SELECT SUM(value_col) INTO v_sum FROM threshold_table;
74   v_sum := v_sum + :NEW.value_col;
75   IF v_sum > v_threshold THEN
76     RAISE threshold_exceeded;
77   END IF;
78 EXCEPTION
79   WHEN threshold_exceeded THEN
80     RAISE_APPLICATION_ERROR(-20003, 'Threshold exceeded for value_col.');
81 END;
82
```

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. In the top navigation bar, 'APEX' is selected. The main area displays the SQL command for creating the trigger:

```
105
106 CREATE OR REPLACE TRIGGER log_changes
107 AFTER UPDATE ON main_table
108 FOR EACH ROW
109 BEGIN
110     INSERT INTO audit_table (audit_id, changed_id, old_col1, new_col1, old_col2, new_col2, change_time)
111     VALUES (audit_seq.NEXTVAL, :OLD.id, :OLD.col1, :NEW.col1, :OLD.col2, :NEW.col2, SYSTIMESTAMP);
112 END;
113
```

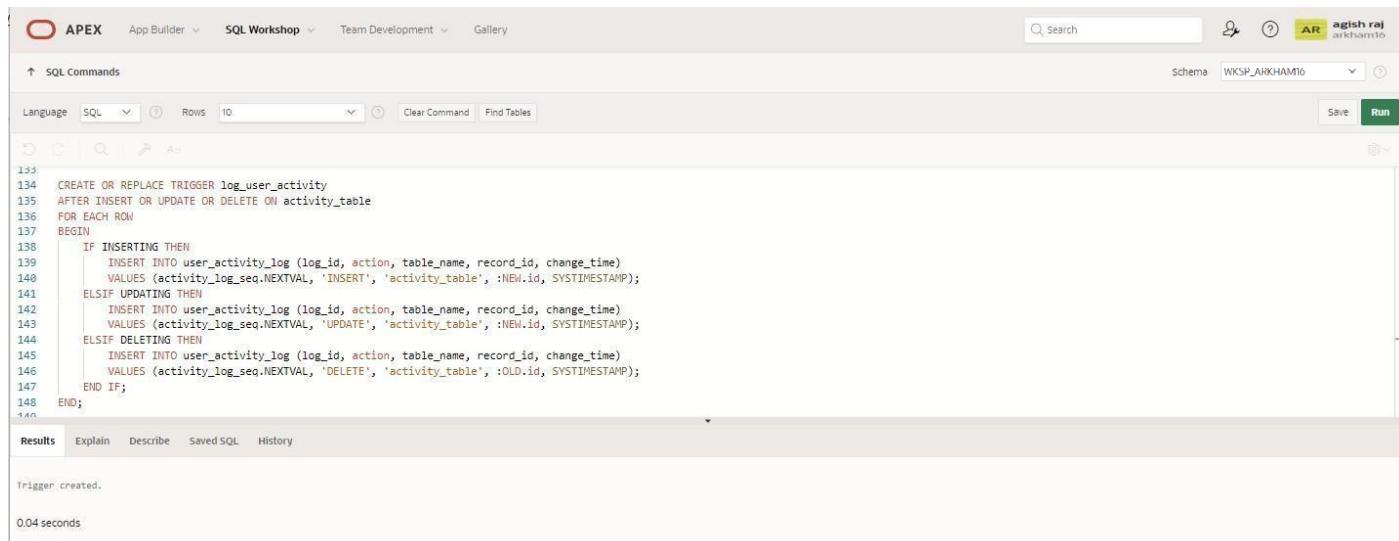
The 'Results' tab is active, showing the message "Trigger created." and a execution time of "0.03 seconds".

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. In the top navigation bar, 'APEX' is selected. The main area displays the SQL command for creating the trigger:

```
133
134 CREATE OR REPLACE TRIGGER log_user_activity
135 AFTER INSERT OR UPDATE OR DELETE ON activity_table
136 FOR EACH ROW
137 BEGIN
138   IF INSERTING THEN
139     INSERT INTO user_activity_log (log_id, action, table_name, record_id, change_time)
140       VALUES (activity_log_seq.NEXTVAL, 'INSERT', 'activity_table', :NEW.id, SYSTIMESTAMP);
141   ELSIF UPDATING THEN
142     INSERT INTO user_activity_log (log_id, action, table_name, record_id, change_time)
143       VALUES (activity_log_seq.NEXTVAL, 'UPDATE', 'activity_table', :NEW.id, SYSTIMESTAMP);
144   ELSIF DELETING THEN
145     INSERT INTO user_activity_log (log_id, action, table_name, record_id, change_time)
146       VALUES (activity_log_seq.NEXTVAL, 'DELETE', 'activity_table', :OLD.id, SYSTIMESTAMP);
147   END IF;
148 END;
```

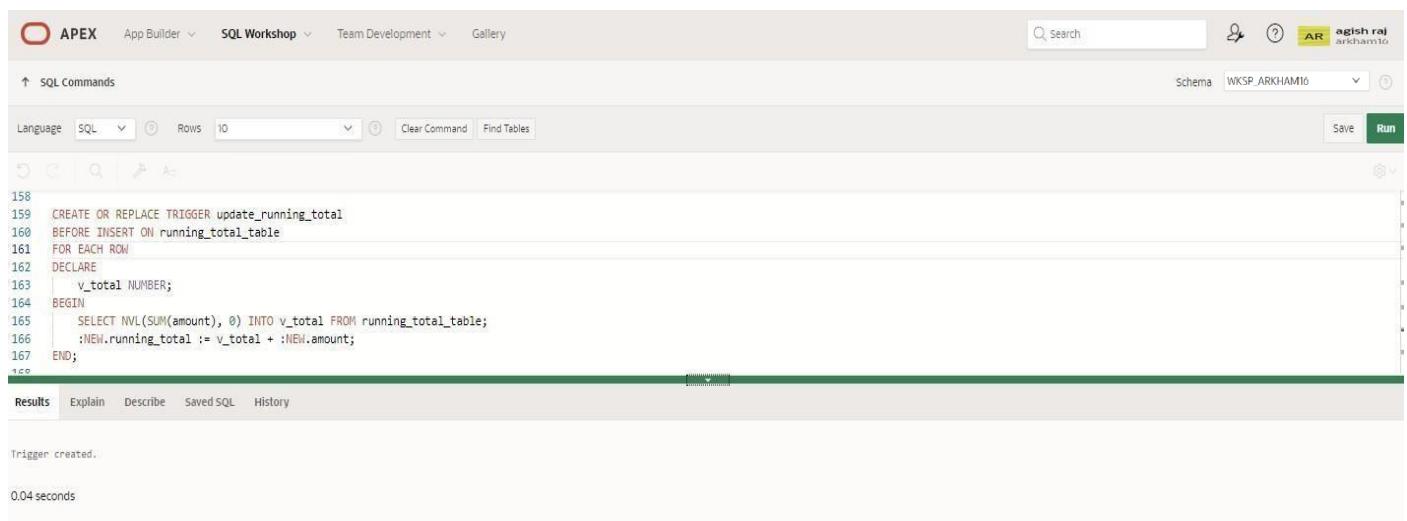
The 'Results' tab is selected at the bottom, showing the message "Trigger created." and a execution time of "0.04 seconds".

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

## QUERY:

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

## OUTPUT:



The screenshot shows the Oracle 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 'agish raj' with the schema 'WKSP\_ARKHAM16'. The main area is a SQL editor with the following content:

```
158 CREATE OR REPLACE TRIGGER update_running_total
159   BEFORE INSERT ON running_total_table
160   FOR EACH ROW
161   DECLARE
162     v_total NUMBER;
163   BEGIN
164     SELECT NVL(SUM(amount), 0) INTO v_total FROM running_total_table;
165     :NEW.running_total := v_total + :NEW.amount;
166   END;
167
168
```

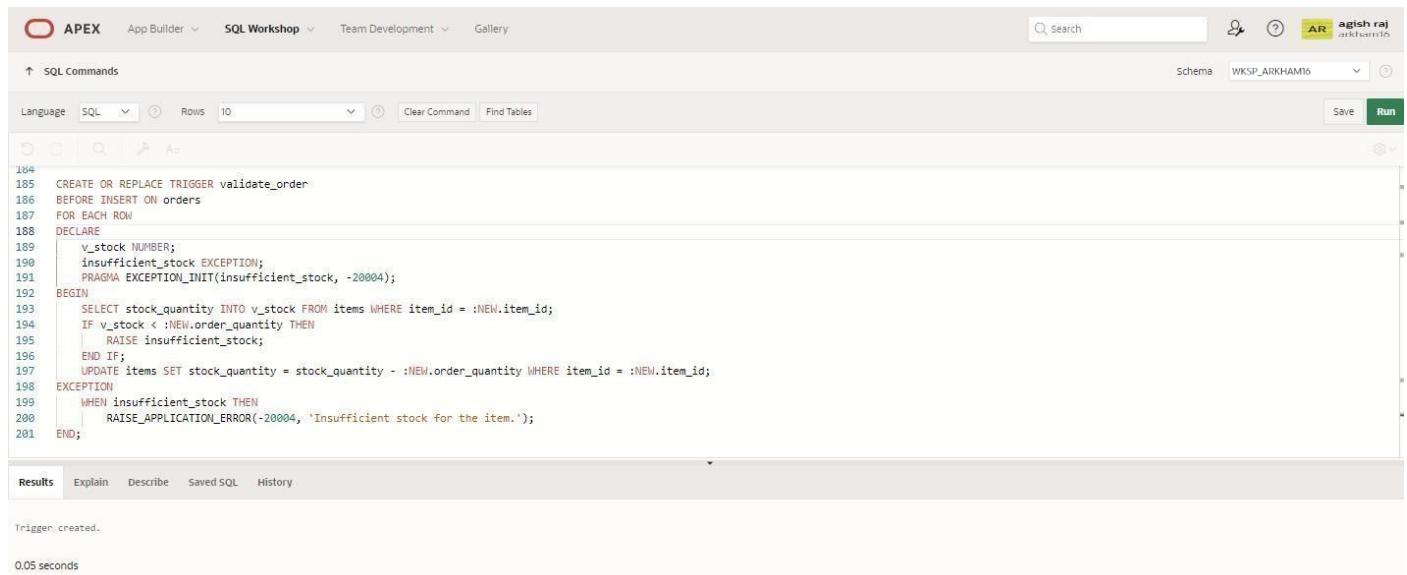
Below the code, the status bar shows 'Trigger created.' and '0.04 seconds'.

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. Below it, tabs for 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery' are visible. On the right side, there's a user profile icon for 'agish raj' and a workspace name 'WKSP\_ARKHAM10'. The main area is titled 'SQL Commands'. A toolbar below the title includes buttons for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', 'Find Tables', 'Save', and 'Run'. The SQL editor contains the PL/SQL code for the trigger. The code is numbered from 184 to 201. The output pane at the bottom shows the message 'Trigger created.' and a execution time of '0.05 seconds'.

```
184: CREATE OR REPLACE TRIGGER validate_order
185: BEFORE INSERT ON orders
186: FOR EACH ROW
187: DECLARE
188:     v_stock NUMBER;
189:     insufficient_stock EXCEPTION;
190:     PRAGMA EXCEPTION_INIT(insufficient_stock, -20004);
191: BEGIN
192:     SELECT stock_quantity INTO v_stock FROM items WHERE item_id = :NEW.item_id;
193:     IF v_stock < :NEW.order_quantity THEN
194:         RAISE insufficient_stock;
195:     END IF;
196:     UPDATE items SET stock_quantity = stock_quantity - :NEW.order_quantity WHERE item_id = :NEW.item_id;
197: EXCEPTION
198:     WHEN insufficient_stock THEN
199:         RAISE_APPLICATION_ERROR(-20004, 'Insufficient stock for the item.');
200: END;
```

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

## **RESULT:**

# MONGO DB

**EX\_NO: 19**

**DATE:24/05/2024**

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:**

```
agish_16> db.restaurants.find( { $or: [ { name: /^Wil/ }, { cuisine: { $nin: ['American', 'Chinese'] } } ] } , { restaurant_id: 1, name: 1, borough: 1, cuisine: 1 } );
[
  {
    _id: ObjectId('665434a8e97a41578cccdcf6'),
    borough: 'Bronx',
    cuisine: 'Bakery',
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
agish_16>
```

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:**

```
agish_16> db.restaurants.find( { grades: { $elemMatch: { grade: "A", score: 11, date: ISODate("2014-08-11T00:00:00Z") } } }, { restaurant_id: 1, name: 1, grades: 1 } );
agish_16>
agish_16>
agish_16>
agish_16>
agish_16>
```

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:**

```
agish_16> 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 } );
agish_16>
agish_16>
agish_16>
agish_16>
```

**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:**

```
agish_16> db.restaurants.find({$and : [{"address.coord.1": {$gt : 42}}, {"address.coord.1": {$lte : 52}}]}, {_id:0, restaurant_id:1, name:1, address:1});
[{"name": "Morris Park Bake Shop", "address": {"building": "3007", "coord": [-73.856077, 40.848447], "street": "Morris Park Ave", "zipcode": "10462"}, "borough": "Bronx", "cuisine": "Bakery", "grades": [{"date": ISODate("2014-03-03T00:00:00.000Z"), "grade": "A", "score": 2}, {"date": ISODate("2013-09-11T00:00:00.000Z"), "grade": "A", "score": 6}, {"date": ISODate("2013-01-24T00:00:00.000Z"), "grade": "A", "score": 10}, {"date": ISODate("2011-11-23T00:00:00.000Z"), "grade": "A", "score": 9}, {"date": ISODate("2011-03-10T00:00:00.000Z"), "grade": "B", "score": 14}], "restaurant_id": 30075445"}]
```

**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:**

```
agish_16> db.restaurants.find({}, { _id: 0 }).sort({ name: 1 });
[{"name": "Morris Park Bake Shop", "address": {"building": "3007", "coord": [-73.856077, 40.848447], "street": "Morris Park Ave", "zipcode": "10462"}, "borough": "Bronx", "cuisine": "Bakery", "grades": [{"date": ISODate("2014-03-03T00:00:00.000Z"), "grade": "A", "score": 2}, {"date": ISODate("2013-09-11T00:00:00.000Z"), "grade": "A", "score": 6}, {"date": ISODate("2013-01-24T00:00:00.000Z"), "grade": "A", "score": 10}, {"date": ISODate("2011-11-23T00:00:00.000Z"), "grade": "A", "score": 9}, {"date": ISODate("2011-03-10T00:00:00.000Z"), "grade": "B", "score": 14}], "restaurant_id": 30075445"}]
```

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:

```
ogish_16> db.restaurants.find({}, { _id: 0 }).sort({ name: -1 })
[
  {
    address: {
      building: '1007',
      coord: [ -73.856077, 40.848447 ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: ISODate('2014-03-03T00:00:00.000Z'),
        grade: 'A',
        score: 2
      },
      {
        date: ISODate('2013-09-11T00:00:00.000Z'),
        grade: 'A',
        score: 6
      },
      {
        date: ISODate('2013-01-24T00:00:00.000Z'),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate('2011-11-23T00:00:00.000Z'),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate('2011-03-10T00:00:00.000Z'),
        grade: 'B',
        score: 14
      }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
ogish_16>
```

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:

```
ogish_16> db.restaurants.find({}, { _id: 0 }).sort({ cuisine: 1, borough: -1 })
[
  {
    address: {
      building: '1007',
      coord: [ -73.856077, 40.848447 ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: ISODate('2014-03-03T00:00:00.000Z'),
        grade: 'A',
        score: 2
      },
      {
        date: ISODate('2013-09-11T00:00:00.000Z'),
        grade: 'A',
        score: 6
      },
      {
        date: ISODate('2013-01-24T00:00:00.000Z'),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate('2011-11-23T00:00:00.000Z'),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate('2011-03-10T00:00:00.000Z'),
        grade: 'B',
        score: 14
      }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
ogish_16>
```

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:

```
egish_16> db.restaurants.find({ "address.street": { $exists: true, $ne: "" } });
[ {
  _id: ObjectId('665434a8e97a41578cccdcf6'),
  address: {
    building: '1007',
    coord: [ -73.856077, 40.848447 ],
    street: 'Morris Park Ave',
    zipcode: '10462'
  },
  borough: 'Bronx',
  cuisine: 'Bakery',
  grades: [
    {
      date: ISODate('2014-03-03T00:00:00.000Z'),
      grade: 'A',
      score: 2
    },
    {
      date: ISODate('2013-09-11T00:00:00.000Z'),
      grade: 'A',
      score: 6
    },
    {
      date: ISODate('2013-01-24T00:00:00.000Z'),
      grade: 'A',
      score: 10
    },
    {
      date: ISODate('2011-11-23T00:00:00.000Z'),
      grade: 'A',
      score: 9
    },
    {
      date: ISODate('2011-03-10T00:00:00.000Z'),
      grade: 'B',
      score: 14
    }
  ],
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445'
}
]
```

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:

```
egish_16> db.restaurants.find({ "address.coord": { $elemMatch: { $type: "double" } } });
[ {
  _id: ObjectId('665434a8e97a41578cccdcf6'),
  address: {
    building: '1007',
    coord: [ -73.856077, 40.848447 ],
    street: 'Morris Park Ave',
    zipcode: '10462'
  },
  borough: 'Bronx',
  cuisine: 'Bakery',
  grades: [
    {
      date: ISODate('2014-03-03T00:00:00.000Z'),
      grade: 'A',
      score: 2
    },
    {
      date: ISODate('2013-09-11T00:00:00.000Z'),
      grade: 'A',
      score: 6
    },
    {
      date: ISODate('2013-01-24T00:00:00.000Z'),
      grade: 'A',
      score: 10
    },
    {
      date: ISODate('2011-11-23T00:00:00.000Z'),
      grade: 'A',
      score: 9
    },
    {
      date: ISODate('2011-03-10T00:00:00.000Z'),
      grade: 'B',
      score: 14
    }
  ],
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445'
}
]
```

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:

```
agish_16> db.restaurants.find({ "grades.score": { $mod: [7, 0] } }, { restaurant_id: 1, name: 1, grades: 1 });
[
  {
    _id: ObjectId('665434a8e97a41578cccdcf6'),
    grades: [
      {
        date: ISODate('2014-03-03T00:00:00.000Z'),
        grade: 'A',
        score: 2
      },
      {
        date: ISODate('2013-09-11T00:00:00.000Z'),
        grade: 'A',
        score: 6
      },
      {
        date: ISODate('2013-01-24T00:00:00.000Z'),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate('2011-11-23T00:00:00.000Z'),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate('2011-03-10T00:00:00.000Z'),
        grade: 'B',
        score: 14
      }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
agish_16>
```

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:

```
agish_16> db.restaurants.find({ name: /mon/i }, { name: 1, borough: 1, "address.coord": 1, cuisine: 1 });
agish_16>
agish_16>
agish_16>
agish_16>
```

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:

```
agish_16> db.restaurants.find({ name: /^Mad/i }, { name: 1, borough: 1, "address.coord": 1, cuisine: 1 });
agish_16>
agish_16>
agish_16>
agish_16>
```

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:

```
agish_16> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } } });
{
  "_id": ObjectId('605434a8e97a41578cccdcf6'),
  "address": {
    "building": "1007",
    "coord": [ -73.856077, 40.848447 ],
    "street": "Morris Park Ave",
    "zipcode": "10462"
  },
  "borough": "Bronx",
  "cuisine": "Bakery",
  "grades": [
    {
      "date": ISODate('2014-03-03T00:00:00.000Z'),
      "grade": 'A',
      "score": 2
    },
    {
      "date": ISODate('2013-09-11T00:00:00.000Z'),
      "grade": 'A',
      "score": 6
    },
    {
      "date": ISODate('2013-01-24T00:00:00.000Z'),
      "grade": 'A',
      "score": 10
    },
    {
      "date": ISODate('2011-11-23T00:00:00.000Z'),
      "grade": 'A',
      "score": 9
    },
    {
      "date": ISODate('2011-03-10T00:00:00.000Z'),
      "grade": 'B',
      "score": 14
    }
  ],
  "name": "Morris Park Bake Shop",
  "restaurant_id": "30075445"
}
agish_16>
```

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:

```
agish_16> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, "borough": "Manhattan" });
agish_16>
agish_16>
agish_16>
```

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:

```
agish_16> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }] });
agish_16>
agish_16>
agish_16>
```

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:

```
agish_16> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $ne: "American" } });

agish_16>

agish_16>

agish_16>
```

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:

```
agish_16> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $nin: ["American", "Chinese"] } });

agish_16>

agish_16>

agish_16>
```

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:

```
agish_16> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }] });

{
  _id: ObjectId('665434a8e97a41578cccdcf6'),
  address: {
    building: '1007',
    coord: [-73.856077, 40.848447],
    street: 'Morris Park Ave',
    zipcode: '10462'
  },
  borough: 'Bronx',
  cuisine: 'Bakery',
  grades: [
    {
      date: ISODate('2014-03-03T00:00:00.000Z'),
      grade: 'A',
      score: 2
    },
    {
      date: ISODate('2013-09-11T00:00:00.000Z'),
      grade: 'A',
      score: 6
    },
    {
      date: ISODate('2013-01-24T00:00:00.000Z'),
      grade: 'A',
      score: 10
    },
    {
      date: ISODate('2011-11-23T00:00:00.000Z'),
      grade: 'A',
      score: 9
    },
    {
      date: ISODate('2011-03-10T00:00:00.000Z'),
      grade: 'B',
      score: 14
    }
  ],
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445'
}
agish_16>
```

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:**

```
agish_16> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], "borough": "Manhattan" });
agish_16>
agish_16>
agish_16>
```

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

**QUERY:**

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

**OUTPUT:**

```
agish_16> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }] });
agish_16>
agish_16>
agish_16>
```

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:**

```
agish_16> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $ne: "American" } });
agish_16>
agish_16>
agish_16>
agish_16>
```

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:

```
ogish_16> 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"] } });
ogish_16>
ogish_16>
ogish_16>
```

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:

```
ogish_16> db.restaurants.find({ $or: [{ "grades.score": 2 }, { "grades.score": 6 }] });
[
  {
    _id: ObjectId('665434a8e97a41578cccdcf6'),
    address: {
      building: '1007',
      coord: [-73.856077, 40.848447],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: ISODate('2014-03-03T00:00:00.000Z'),
        grade: 'A',
        score: 2
      },
      {
        date: ISODate('2013-09-11T00:00:00.000Z'),
        grade: 'A',
        score: 6
      },
      {
        date: ISODate('2013-01-24T00:00:00.000Z'),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate('2011-11-23T00:00:00.000Z'),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate('2011-03-10T00:00:00.000Z'),
        grade: 'B',
        score: 14
      }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
ogish_16>
```

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:

```
ogish_16> db.movies.find({ year: 1893 });
ogish_16>
ogish_16>
ogish_16>
```

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:

```
ogish_16> db.movies.find({ runtime: { $gt: 120 } });
ogish_16>
ogish_16>
ogish_16>
```

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

QUERY:

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

OUTPUT:

```
ogish_16> db.movies.find({ genres: 'Short' });
[
  {
    _id: ObjectId('573a1398f29313cabcd42e8'),
    plot: 'A group of bandits stage a brazen train hold-up, only to find a determined posse hot on their heels.',
    genres: [ 'Short', 'Western' ],
    runtime: 11,
    cast: [
      'A.C. Abadie',
      'Gilbert M. 'Broncho Billy' Anderson',
      'George Barnes',
      'Justus D. Barnes'
    ],
    poster: 'https://m.media-amazon.com/images/M/MV5BMTU3NjE5NzYtYTYYNS00MDVmLWIwYjgtMmYwYjIxZDYyNzQzXkEyXkFqcGdeQXVxNzQzNzI@._V1_SX677_AL_.jpg',
    title: 'The Great Train Robbery',
    fullplot: 'Among the earliest existing films in American cinema - notable as the first film that presented a narrative story to tell - it depicts a group of cowboy outlaws who hold up a train and rob the passengers. They are then pursued by a Sheriff's posse. Several scenes have color included - all hand tinted.',
    languages: [ 'English' ],
    released: ISODate('1903-12-01T00:00:00.000Z'),
    directors: [ 'Edwin S. Porter' ],
    rated: 'TV-G',
    awards: { wins: 1, nominations: 0, text: '1 win.' },
    lastupdated: '2015-08-13 00:27:59.177000000',
    year: 1903,
    imbd: { rating: 7.4, votes: 9847, id: 439 },
    countries: [ 'USA' ],
    type: 'movie',
    tomatoes: {
      viewer: { rating: 3.7, numReviews: 2559, meter: 75 },
      fresh: 6,
      critic: { rating: 7.6, numReviews: 6, meter: 100 },
      rotten: 0,
      lastupdated: ISODate('2015-08-08T19:16:10.000Z')
    }
  }
]
ogish_16>
```

**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:**

```
agish_16> db.movies.find({ directors: 'William K.L. Dickson' });
agish_16>
agish_16>
agish_16>
```

**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:**

```
agish_16> db.movies.find({ countries: 'USA' });
[
  {
    _id: ObjectId('573a1390f29313caabed42e8'),
    plot: 'A group of bandits stage a brazen train hold-up, only to find a determined posse hot on their heels.',
    genres: [ 'Short', 'Western' ],
    runtime: 11,
    cast: [
      'A.C. Abadie',
      'Gilbert M. "Broncho Billy" Anderson',
      'George Barnes',
      'Justus D. Barnes'
    ],
    poster: 'https://m.media-amazon.com/images/M/MN5BMTU3NjE5NzYtYTYYNS80MDVmLWIwYjgtUmYvVlIxZDVyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzT@_V1_SV1000_SZ677_AL_.jpg',
    title: 'The Great Train Robbery',
    fullplot: "Among the earliest existing films in American cinema - notable as the first film that presented a narrative story to tell - it depicts a group of cowboy outlaws who hold up a train and rob the passengers. They are then pursued by a Sheriff's posse. Several scenes have color included - all hand tinted.",
    languages: [ 'English' ],
    released: ISODate('1903-12-01T00:00:00.000Z'),
    directors: [ 'Edwin S. Porter' ],
    rated: 'TV-G',
    awards: { wins: 1, nominations: 0, text: '1 win.' },
    lastupdated: '2015-08-13 00:27:59.177000000',
    year: 1903,
    imbd: { rating: 7.4, votes: 9847, id: 439 },
    countries: [ 'USA' ],
    type: 'movie',
    tomatoes: {
      viewer: { rating: 3.7, numReviews: 2559, meter: 75 },
      fresh: 6,
      critic: { rating: 7.6, numReviews: 6, meter: 100 },
      rotten: 0,
      lastUpdated: ISODate('2015-08-08T19:16:10.000Z')
    }
  }
]
agish_16>
```

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

**QUERY:**

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

**OUTPUT:**

```
agish_16> db.movies.find({ rated: 'UNRATED' });
agish_16>
agish_16>
agish_16>
```

**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:**

```
agish_16> db.movies.find({ 'imdb.votes': { $gt: 1000 } });
[
  {
    _id: ObjectId('573a1390f29313caababcd42e8'),
    plot: 'A group of bandits stage a brazen train hold-up, only to find a determined posse hot on their heels.',
    genres: [ 'Short', 'Western' ],
    runtime: 11,
    cast: [
      'A.C. Abadie',
      'Gilbert M. "Broncho Billy" Anderson',
      'George Barnes',
      'Justus D. Barnes'
    ],
    poster: 'https://m.media-amazon.com/images/M/MV5BMTU3NjE5NzYtYTYYNS00MDVmLWIwYjgtNmYvYIxZDYyNzU2XkEyXkFqcGdeQXVvNzQzQxNzI@._V1_SV1000_SX677_AL_.jpg',
    title: 'The Great Train Robbery',
    fullplot: 'Among the earliest existing films in American cinema - notable as the first film that presented a narrative story to tell - it depicts a group of cowboy outlaws who hold up a train and rob the passengers. They are then pursued by a Sheriff's posse. Several scenes have color included - all hand tinted.',
    languages: [ 'English' ],
    released: ISODate('1903-12-01T00:00:00.000Z'),
    directors: [ 'Edwin S. Porter' ],
    rated: 'TV-G',
    awards: { wins: 1, nominations: 0, text: '1 win.' },
    lastupdated: '2015-08-13 00:27:59.177000000',
    year: 1903,
    imdb: { rating: 7.4, votes: 9847, id: 439 },
    countries: [ 'USA' ],
    type: 'movie',
    tomatoes: {
      viewer: { rating: 3.7, numReviews: 2559, meter: 75 },
      fresh: 6,
      critic: { rating: 7.6, numReviews: 6, meter: 100 },
      rotten: 0,
      lastUpdated: ISODate('2015-08-08T19:16:10.000Z')
    }
  }
]
agish_16>
```

**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:**

```
agish_16> db.movies.find({ 'imdb.rating': { $gt: 7 } });
[
  {
    _id: ObjectId('573a1390f29313caababcd42e8'),
    plot: 'A group of bandits stage a brazen train hold-up, only to find a determined posse hot on their heels.',
    genres: [ 'Short', 'Western' ],
    runtime: 11,
    cast: [
      'A.C. Abadie',
      'Gilbert M. "Broncho Billy" Anderson',
      'George Barnes',
      'Justus D. Barnes'
    ],
    poster: 'https://m.media-amazon.com/images/M/MV5BMTU3NjE5NzYtYTYYNS00MDVmLWIwYjgtNmYvYIxZDYyNzU2XkEyXkFqcGdeQXVvNzQzQxNzI@._V1_SV1000_SX677_AL_.jpg',
    title: 'The Great Train Robbery',
    fullplot: 'Among the earliest existing films in American cinema - notable as the first film that presented a narrative story to tell - it depicts a group of cowboy outlaws who hold up a train and rob the passengers. They are then pursued by a Sheriff's posse. Several scenes have color included - all hand tinted.',
    languages: [ 'English' ],
    released: ISODate('1903-12-01T00:00:00.000Z'),
    directors: [ 'Edwin S. Porter' ],
    rated: 'TV-G',
    awards: { wins: 1, nominations: 0, text: '1 win.' },
    lastupdated: '2015-08-13 00:27:59.177000000',
    year: 1903,
    imdb: { rating: 7.4, votes: 9847, id: 439 },
    countries: [ 'USA' ],
    type: 'movie',
    tomatoes: {
      viewer: { rating: 3.7, numReviews: 2559, meter: 75 },
      fresh: 6,
      critic: { rating: 7.6, numReviews: 6, meter: 100 },
      rotten: 0,
      lastUpdated: ISODate('2015-08-08T19:16:10.000Z')
    }
  }
]
agish_16>
```

**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:**

```
agish_16> db.movies.find({ 'tomatoes.viewer.rating': { $gt: 4 } });  
agish_16>  
agish_16>  
agish_16>
```

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

**QUERY:**

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

**OUTPUT:**

```
agish_16> db.movies.find({ 'awards.wins': { $gt: 0 } });  
[  
  {  
    _id: ObjectId('573a1390f29313caabacd42e8'),  
    plot: 'A group of bandits stage a brazen train hold-up, only to find a determined posse hot on their heels.',  
    genres: [ 'Short', 'Western' ],  
    runtime: 11,  
    cast: [  
      'A.C. Abadie',  
      'Gilbert M. 'Broncho Billy' Anderson',  
      'George Barnes',  
      'Justus D. Barnes'  
    ],  
    poster: 'https://m.media-amazon.com/images/M/MV5MTU3NjE5NzYtYTtyNS00MDVmLWIVYjgtUmYvMlxZDYYNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._V1_Sy1000_Sx677_Al_.jpg',  
    title: 'The Great Train Robbery',  
    fullplot: "Among the earliest existing films in American cinema - notable as the first film that presented a narrative story to tell - it depicts a group of cowboy outlaws who hold up a train and rob the passengers. They are then pursued by a Sheriff's posse. Several scenes have color included - all hand tinted.",  
    languages: [ 'English' ],  
    released: ISODate('1903-12-01T00:00:00.000Z'),  
    directors: [ 'Edwin S. Porter' ],  
    rated: 'TV-G',  
    awards: { wins: 1, nominations: 0, text: '1 win.' },  
    lastupdated: '2015-08-13 00:27:59.177000000',  
    year: 1903,  
    imdb: { rating: 7.4, votes: 9847, id: 439 },  
    countries: [ 'USA' ],  
    type: 'movie',  
    tomatoes: {  
      viewer: { rating: 3.7, numReviews: 2559, meter: 75 },  
      fresh: 6,  
      critic: { rating: 7.6, numReviews: 6, meter: 100 },  
      rotten: 0,  
      lastUpdated: ISODate('2015-08-08T19:16:10.000Z')  
    }  
  }  
]  
agish_16>
```

**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:**

```
agish_16> 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 } );  
agish_16>  
agish_16>  
agish_16>
```

**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:**

```
ogish_16> 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 } );
ogish_16>
ogish_16>
ogish_16>
```

**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:**

```
ogish_16> db.movies.find( { released: ISODate("1893-05-09T00:00:00.000Z") }, { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 } );
ogish_16>
ogish_16>
ogish_16>
```

**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:**

```
ogish_16> db.movies.find( { title: /scene/i }, { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 } );
ogish_16>
ogish_16>
ogish_16>
```

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

**RESULT:**