

# 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
<b>Key Type</b>		
<b>Nulls/Unique</b>		
<b>FK table</b>		
<b>FK column</b>		
<b>Data Type</b>	Number	Varchar2
<b>Length</b>	7	25

## QUERY:

```
Create table deptcsea3(ID int,Name varchar(20));
```

## OUTPUT:

The screenshot shows the Oracle APEX SQL Commands interface. In the SQL pane, the command `create table deptcsea3(ID int,Name varchar(20));` is entered. The Results pane displays the output: "Table created." and "0.05 seconds". The interface includes tabs for Results, Explain, Describe, SavedSQL, and History, along with various toolbar buttons.

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

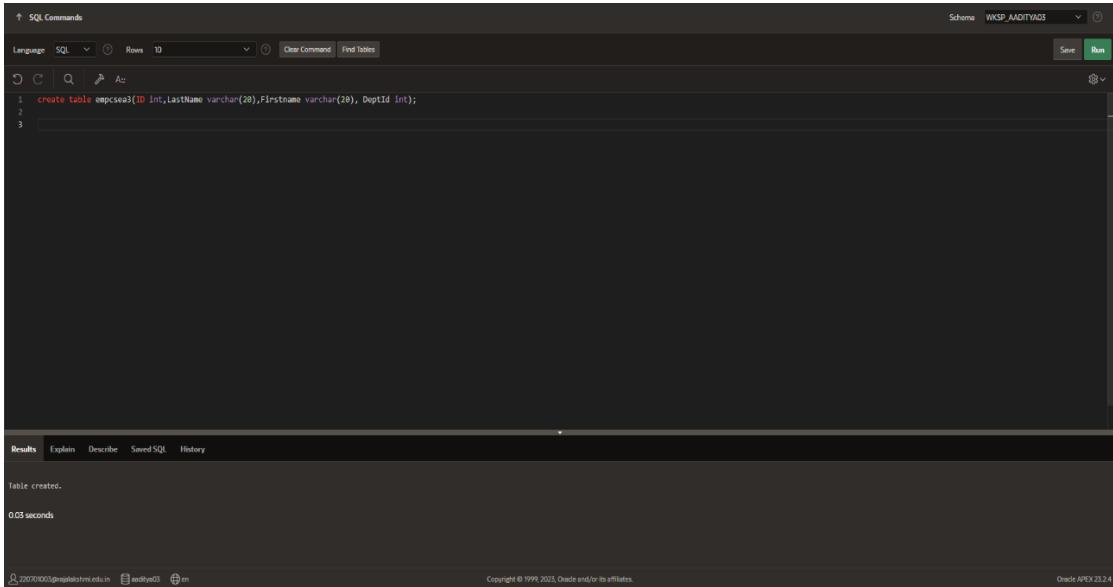
Column name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
<b>Key Type</b>				
<b>Nulls/Unique</b>				
<b>FK table</b>				
<b>FK column</b>				

<b>Data Type</b>	Number	Varchar2	Varchar2	Number
<b>Length</b>	7	25	25	7

### QUERY:

```
create table empcsea3(ID int,LastName varchar(20),FirstName varchar(20),DeptId int);
```

### OUTPUT:



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

```
1 create table empcsea3(ID int,LastName varchar(20),FirstName varchar(20),DeptId int);
2
3
```

In the Results pane, the output is:

```
Table created.

0.03 seconds
```

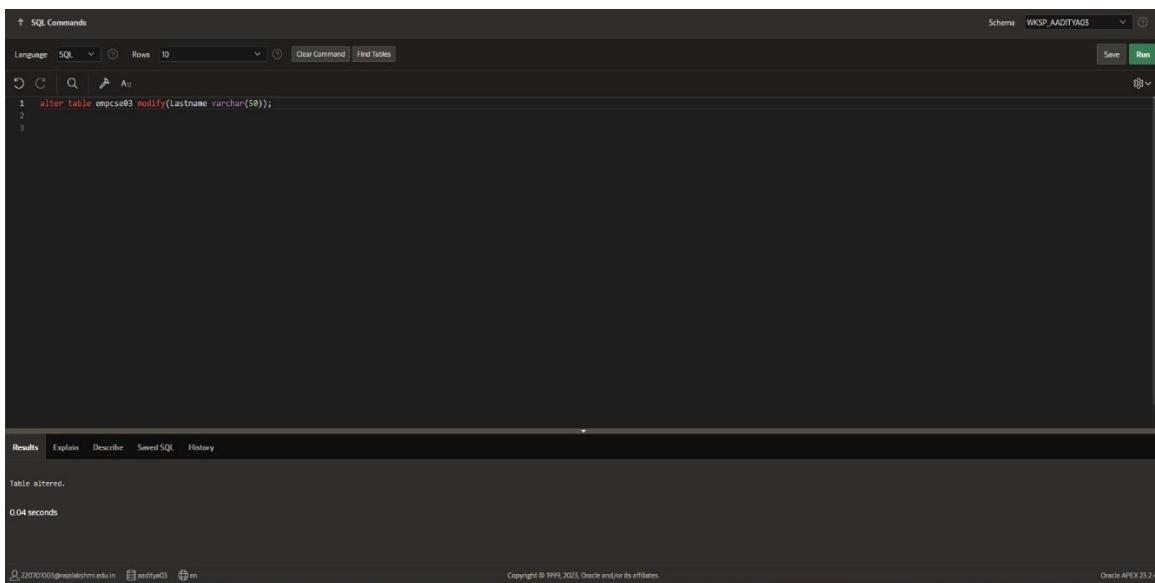
At the bottom, the URL is 220707003@oracleshri.in:5503 and the schema is WKSP\_AADITYA03.

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

### QUERY:

```
Alter table empcse03 modify(LastName varchar(50));
```

### OUTPUT:



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

```
1 alter table empcse03 modify(Lastname varchar(50));
2
3
```

In the Results pane, the output is:

```
Table altered.

0.04 seconds
```

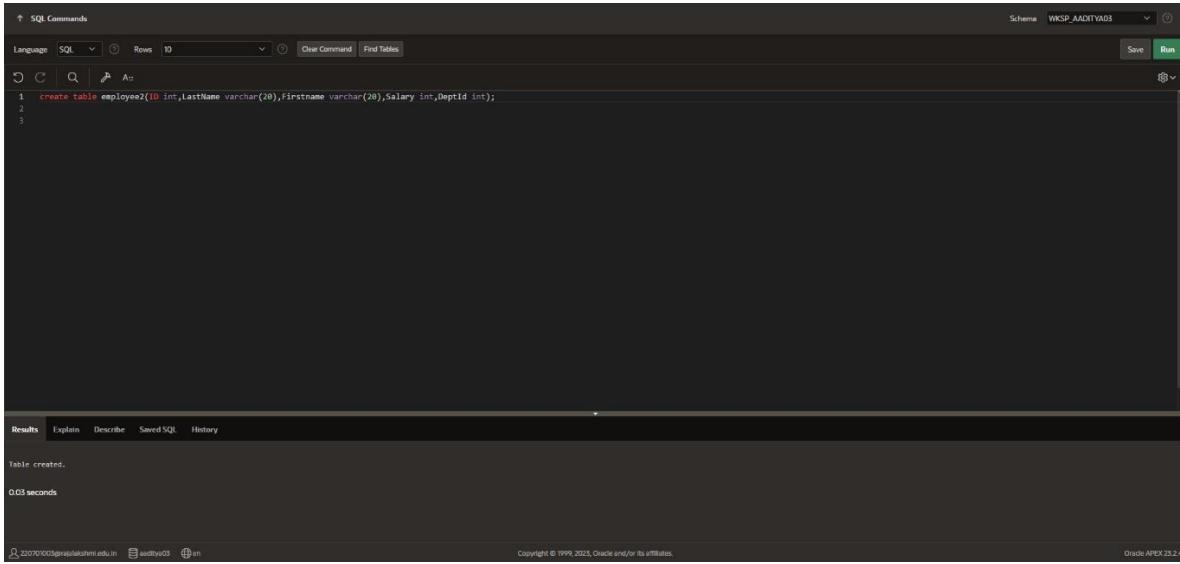
At the bottom, the URL is 220707003@oracleshri.in:5503 and the schema is WKSP\_AADITYA03.

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(ID int,LastName varchar(20),Firstname varchar(20),Salary int,DeptId int);
```

**OUTPUT:**



The screenshot shows the Oracle SQL Developer interface. In the top-left, there's a toolbar with icons for Undo, Redo, Find, and Save. Below it is a dropdown menu for Language (set to SQL) and a Row count selector (set to 10). To the right are buttons for Clear Command and Find Tables. The main area contains the SQL command:

```
1 create table employee2(ID int,LastName varchar(20),Firstname varchar(20),Salary int,DeptId int);  
2  
3
```

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

```
Table created.  
0.03 seconds
```

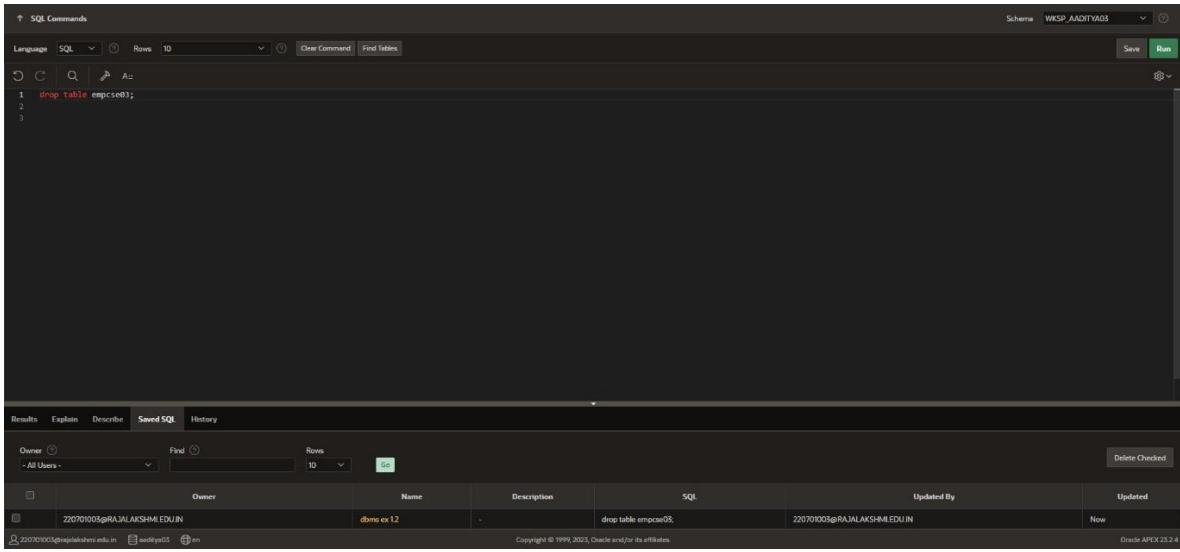
At the bottom, the footer includes the user information (220701003@rajalakshmi.edu.in, auditya03, en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the version (Oracle APEX 23.2.4).

5.Drop the EMP table.

**QUERY:**

```
Drop table emp;
```

**OUTPUT:**



The screenshot shows the Oracle SQL Developer interface. The top section is identical to the previous one, with the SQL language selected and the command:

```
1 drop table emp;
```

The results tab shows the output:

```
Table dropped.
```

The bottom section shows the history of the table drop. The table 'emp' was dropped by user '220701003@RAJALAKSHMI.EDU.IN' at '2023-09-12 10:45:12'. The history table has columns: Owner, Name, Description, SQL, Updated By, and Updated.

Owner	Name	Description	SQL	Updated By	Updated
- All Users -	dbms_xe_12	-	drop table emp;	220701003@RAJALAKSHMI.EDU.IN	Now

The footer includes the user information (220701003@rajalakshmi.edu.in, auditya03, en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the version (Oracle APEX 23.2.4).

6.Rename the EMPLOYEES2 table as EMP.

**QUERY:**

```
Rename employees2 to emp;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The SQL editor contains the following command:

```
1 rename employees2 to emp;
2
```

The results pane shows the output of the command:

Statement processed.  
0.04 seconds

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

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The SQL editor contains the following command:

```
1 comment on table emp is 'Employee details';
2
```

The results pane shows the output of the command:

Statement processed.  
0.05 seconds

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

**QUERY:**

Alter table emp drop column firstname;

**OUTPUT:**

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, there is a SQL Commands tab with the following code:

```
1 alter table emp drop column firstname;
2
```

The top-right pane shows the schema as "WKSP\_AADITYADS". There are "Save" and "Run" buttons. Below the code editor, there is a results grid with the following columns: Owner, Name, Description, SQL, Updated By, and Updated. One row is visible:

Owner	Name	Description	SQL	Updated By	Updated
2207003@RAJALAKSHMI.EDU.IN	dbms_ex_18	-	alter table emp drop column firstname;	2207003@RAJALAKSHMI.EDU.IN	1 seconds ago

At the bottom right of the interface, it says "Oracle APEX 23.2.4".

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

## RESULT:

# MANIPULATING DATA

EX\_NO:2

DATE:

1.Create MY\_EMPLOYEE table with the following structure

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

## QUERY:

```
create table empex02(Id number(4),Last_name varchar(20),First_Name varchar(20),Userid varchar(25),Salary number(9,2));
```

## OUTPUT:

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

```
1 create table empex02(Id number(4),Last_name varchar(20),First_Name varchar(20),Userid varchar(25),Salary number(9,2));
```

In the Results pane, the output is displayed:

```
Table created.  
0.02 seconds
```

At the bottom, the user information is shown as Z20709005@replabshmsk.edu.in and the session ID is 6ed7yeG5.

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

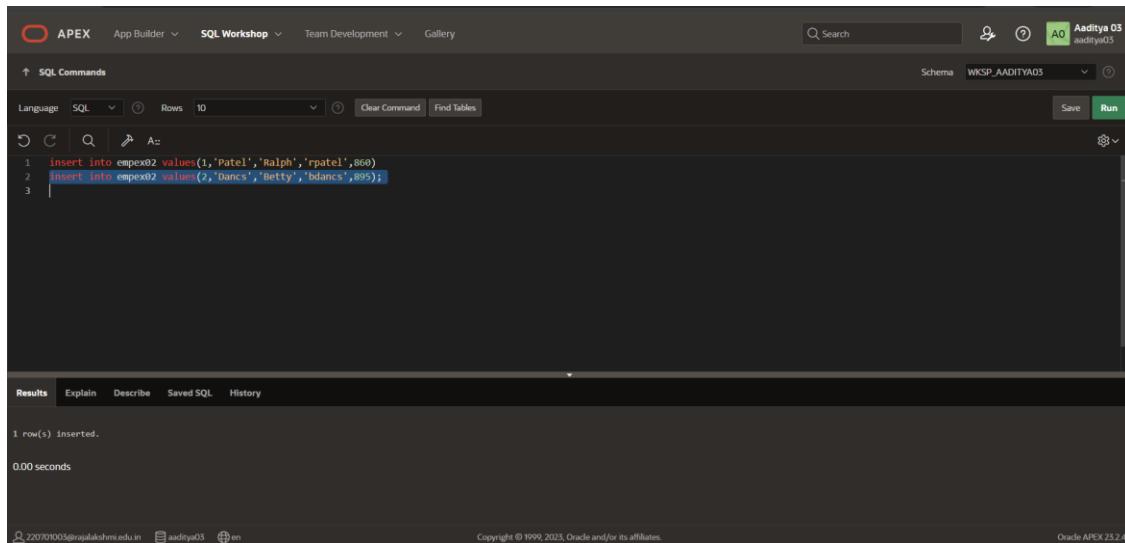
ID	Last_name	First_name	Userid	salary
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	Cnewman	750

5	Ropebur	Audrey	aropebur	1550
---	---------	--------	----------	------

### QUERY:

```
insert into empex02 values(1,'Patel','Ralph','rpatel',860);
insert into empex02 values(2,'Dancs','Betty','bdancs',895);
```

### OUTPUT:



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

```
1 insert into empex02 values(1,'Patel','Ralph','rpatel',860)
2 insert into empex02 values(2,'Dancs','Betty','bdancs',895);
```

The Results tab displays the output:

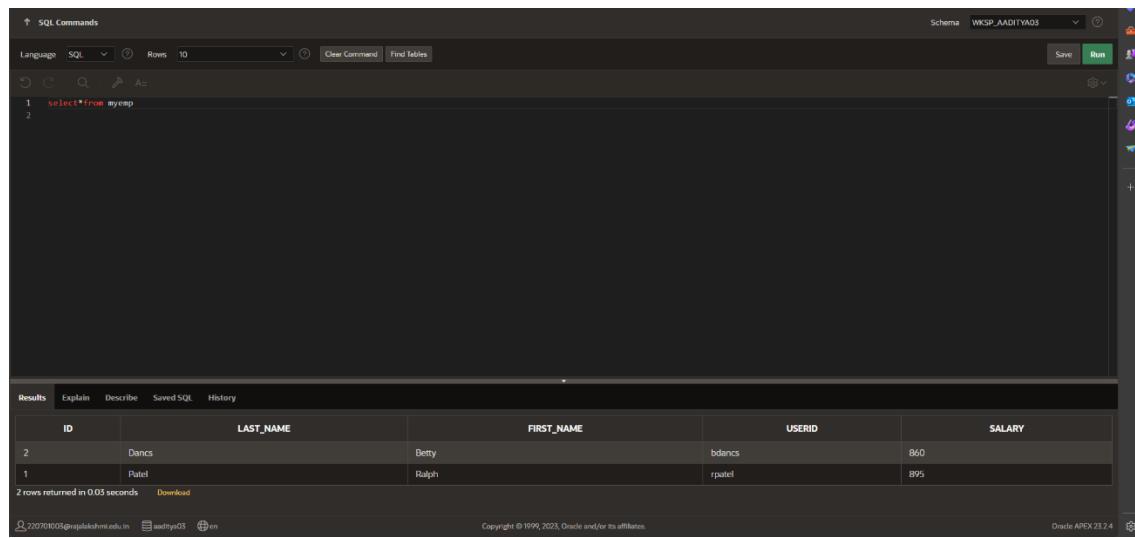
1 row(s) inserted.  
0.00 seconds

3.Display the table with values.

### QUERY:

```
Select*from empex02;
```

### OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. A SELECT\* query is executed against the myemp table:

```
1 select*from myemp
2
```

The Results tab displays the output:

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

2 rows returned in 0.03 seconds

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

**QUERY:** insert into empex02 values(1,'Biri','Ben','bbiri',1100);  
insert into empex02 values(2,'Newman','Chad','Cnewman',750);

**OUTPUT:**

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

```
1 insert into empex02 values(1,'Biri','Ben','bbiri',1100);
2 insert into empex02 values(2,'Newman','Chad','Cnewman',750);
```

Below the code, the Results tab shows the output:

1 row(s) inserted.  
0.01 seconds

At the bottom, the URL is 220701003@rajalakshmi.edu.in, the session is aaditya05, and the page is en. The copyright notice is Copyright © 1999, 2023, Oracle and/or its affiliates. The version is Oracle APEX 23.2.4.

5.Make the data additions permanent.

**QUERY:**  
select\*from empex02;

**OUTPUT:**

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

```
1 select*from empex02;
```

Below the code, the Results tab shows the output:

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
2	Dans	Betty	bdbans	895
1	Biri	Ben	bbiri	1100
2	Newman	Chad	Cnewman	750
1	Biri	Ben	bbiri	1100
1	Patel	Ralph	rpatel	895

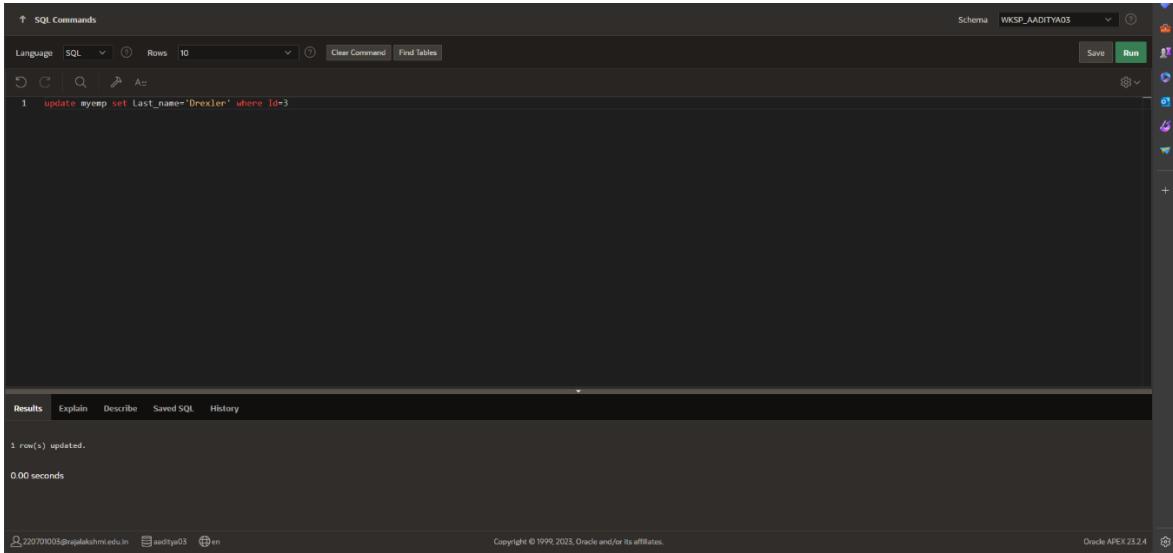
5 rows returned in 0.01 seconds. The URL is 220701003@rajalakshmi.edu.in, the session is aaditya05, and the page is en. The copyright notice is Copyright © 1999, 2023, Oracle and/or its affiliates. The version is Oracle APEX 23.2.4.

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

### QUERY:

Update myemp set lastname='Drexler' where id=3;

### OUTPUT:



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

```
update myemp set Last_name='Drexler' where id=3
```

The results section shows:

```
1 row(s) updated.  
0.00 seconds
```

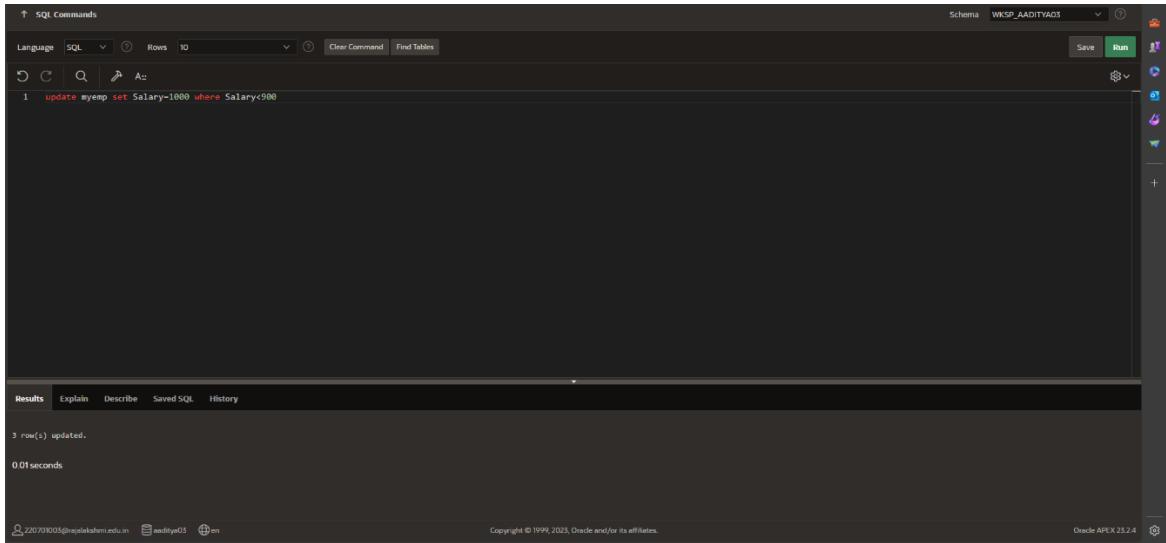
At the bottom, it displays the user information: 22070103@rajalakshmi.edu.in, aaditya03, and Oracle APEX 23.2.4.

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

### QUERY:

Update myemp set salary=1000 where salary<900;

### OUTPUT:



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

```
update myemp set Salary=1000 where Salary<900
```

The results section shows:

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

At the bottom, it displays the user information: 22070103@rajalakshmi.edu.in, aaditya03, and Oracle APEX 23.2.4.

8.Delete Betty dancs from MY \_EMPLOYEE table.

**QUERY:**

Delete from myemp where firstname='Betty';

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The SQL command entered is: `delete from myemp where First_name = 'Betty'`. The results section shows: `1 row(s) deleted.` and `0.01 seconds`. The interface includes tabs for Results, Explain, Describe, Saved SQL, and History. The schema is set to WKSP\_AADITYAOS. The bottom status bar indicates the user is 220701005@rejpalasheri.edu.in and the session is aedity03.

9.Empty the fourth row of the emp table.

**QUERY:**

Delete from myemp where id=4;

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Commands interface. The SQL command entered is: `delete from myemp where Id=4`. The results section shows: `1 row(s) deleted.` and `0.01 seconds`. The interface includes tabs for Results, Explain, Describe, Saved SQL, and History. The schema is set to WKSP\_AADITYAOS. The bottom status bar indicates the user is 220701005@rejpalasheri.edu.in and the session is aedity03.

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

**RESULT:**

# INCLUDING CONSTRAINTS

EX\_NO:3

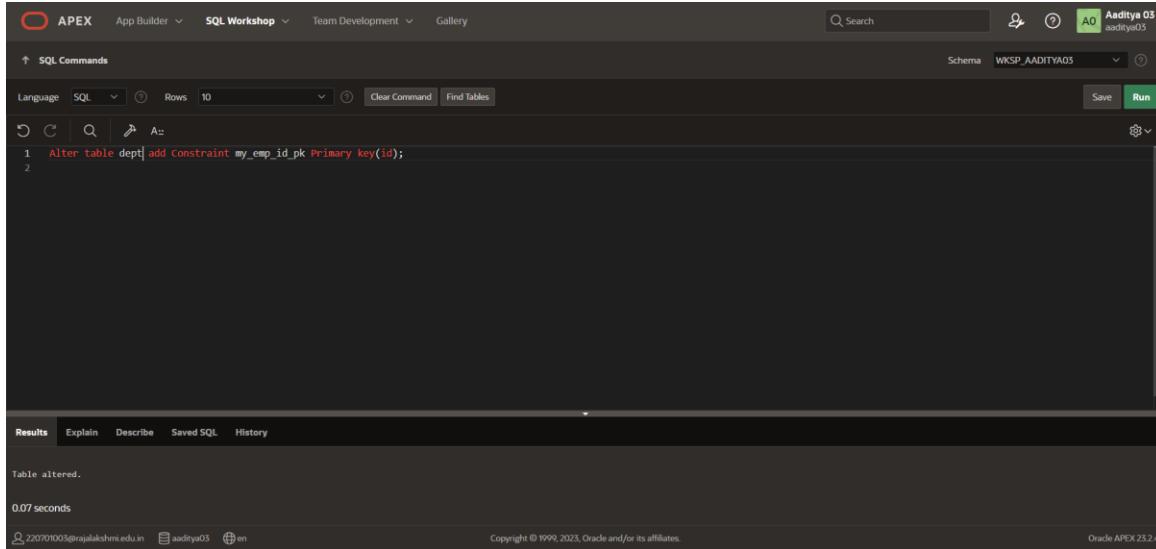
DATE:

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

## QUERY:

Alter table myemp add Constraint my\_emp\_id\_pk Primary key(id);

## OUTPUT:



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

```
1 Alter table dept| add Constraint my_emp_id_pk Primary key(id);  
2
```

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

```
Table altered.  
0.07 seconds
```

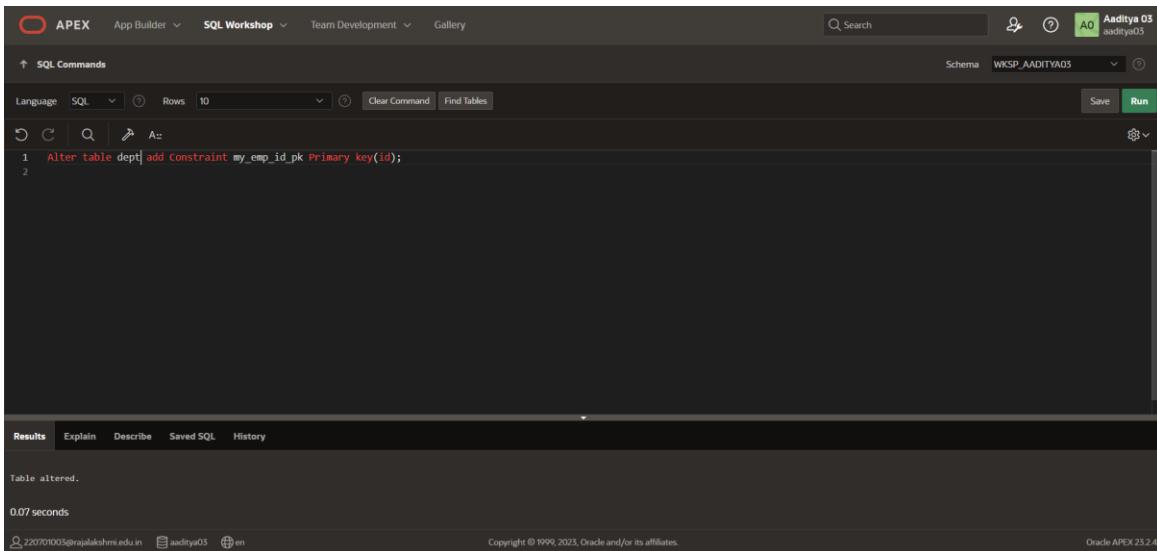
At the bottom, the footer includes the URL 220701003@rajalakshmi.edu.in, the user auditya03, and the session ID en. The copyright notice is Copyright © 1999, 2023, Oracle and/or its affiliates. The version is Oracle APEX 23.2.4.

2.Create a PRIMAY 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:

Alter table dept add Constraint my\_dept\_id\_pk Primary key(id);

## OUTPUT:



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

```
1 Alter table dept| add Constraint my_dept_id_pk Primary key(id);  
2
```

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

```
Table altered.  
0.07 seconds
```

At the bottom, the footer includes the URL 220701003@rajalakshmi.edu.in, the user auditya03, and the session ID en. The copyright notice is Copyright © 1999, 2023, Oracle and/or its affiliates. The version is Oracle APEX 23.2.4.

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 dept add Constraint my\_emp\_dept\_id\_fk Foreign key(DEPT\_ID) references emp(DEPT\_ID);

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows the user 'Aaditya 05' and the schema 'WKSP\_AADITYA05'. The main area is titled 'SQL Commands' with a sub-section 'Language: SQL'. The code entered is:

```
1 Alter table dept add Constraint my_emp_dept_id_fk Foreign key(DEPT_ID) references emp(DEPT_ID);
2
```

The results section shows the output:

```
Table altered.
0.06 seconds
```

At the bottom, it shows the URL '72070005@rajalakshmi.edu.in', the user 'aaditya05', and the session ID '1'. The footer indicates 'Copyright © 1999 2025, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

#### QUERY:

Alter table dept add (Commission number(2,2),Constraint Cn Check(Commission>0));

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows the user 'Aaditya 05' and the schema 'WKSP\_AADITYA05'. The main area is titled 'SQL Commands' with a sub-section 'Language: SQL'. The code entered is:

```
1 Alter table dept add (Commission number(2,2),Constraint Cn Check(Commission>0));
2
```

The results section shows the output:

```
Table altered.
0.06 seconds
```

At the bottom, it shows the URL '72070005@rajalakshmi.edu.in', the user 'aaditya05', and the session ID '1'. The footer indicates 'Copyright © 1999 2025, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

**RESULT:**

# Writing Basic SQL SELECT Statements

EX\_NO:4

DATE:

1. The following statement executes successfully.

## Identify the Errors

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

## QUERY:

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

## OUTPUT:

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

```
1 create table employee (employee_id int,First_name varchar(20),Last_name varchar(20),Email varchar(20),Phone_number number(10),Hire_date varchar(20), Job_id int, Salary int, Manager_id int, Department_id Int, Annual_Salary number(10,2))  
2 insert into employee values(1,'Surya','Kumar','abc@gmail.com',7904656343,'05/03/2024',35,15000,12,1,10000)  
3 select employee_id,Last_name,"SALARY"*12 "ANNUAL_SALARY" from employee
```

The Results tab displays the output of the third query:

EMPLOYEE_ID	LAST_NAME	ANNUAL_SALARY
1	Kumar	180000

1 rows returned in 0.03 seconds

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

Oracle APEX 23.2.4

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

#### QUERY:

Select \* from employees;

#### OUTPUT:

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

```
1 create table employee (employee_id int,First_name varchar(20),Last_name varchar(20),Email varchar(20),Phone_number number(10),Hire_date varchar(20),Job_id int,Salary int,Manager_id int,Department_id Int,Annual_Salary number(10,2))
2 insert into employee values(1,'Surya','Kumar','abc@gmail.com',7904656343,'05/03/2024',35,15000,12,1,10000)
3 select employee_id,Last_name,"SALARY"*12 "ANNUAL_SALARY" from employee
4 select*from employee
```

The Results tab displays the following table:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	salary	MANAGER_ID	DEPARTMENT_ID	ANNUAL_SALARY
1	Surya	Kumar	abc@gmail.com	7904656345	05/03/2024	35	15000	12	1	10000

1 rows returned in 0.02 seconds

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

#### QUERY:

Select employee\_id as employee\_number, last\_name, job\_id from employees;

#### OUTPUT:

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

```
1 create table employee (employee_id int,First_name varchar(20),Last_name varchar(20),Email varchar(20),Phone_number number(10),Hire_date varchar(20),Job_id int,Salary int,Manager_id int,Department_id Int,Annual_Salary number(10,2))
2 insert into employee values(1,'Surya','Kumar','abc@gmail.com',7904656343,'05/03/2024',35,15000,12,1,10000)
3 select employee_id,Last_name,"SALARY"*12 "ANNUAL_SALARY" from employee
4 select*from employee
5 select employee_id as employee_number,Last_name,Job_id,Hire_date from employee;
```

The Results tab displays the following table:

EMPLOYEE_NUMBER	LAST_NAME	JOB_ID	HIRE_DATE
1	Kumar	35	05/03/2024

1 rows returned in 0.00 seconds

4. Provide an alias STARTDATE for the hire date.

#### QUERY:

Select hire\_date as "Startdate" from employees;

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Commands interface. The SQL code entered is:

```
1 create table employee (employee_id int,First_name varchar(20),Last_name varchar(20),Email varchar(20),Phone_number number(10),Hire_date varchar(20), Job_id int, Salary int, Manager_id int, Department_id Int, Annual_Salary number(10,2));
2 insert into employee values(1,'Surya','Kumar','abc@gmail.com',7904656343,'05/03/2024',35,15000,12,1,10000)
3 select employee_id,Last_name,"SALARY"*12 "ANNUAL_SALARY" from employee
4 select*from employee
5 select employee_id as employee_number,Last_name,Job_id,Hire_date from employee;
6 select Hire_date as Star_date from employee
```

The results section shows the output:

STAR_DATE
05/03/2024

1 rows returned in 0.00 seconds

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

#### QUERY:

Select distinct job\_id from employees;

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Commands interface. The SQL code entered is:

```
1 create table employee (employee_id int,First_name varchar(20),Last_name varchar(20),Email varchar(20),Phone_number number(10),Hire_date varchar(20), Job_id int, Salary int, Manager_id int, Department_id Int, Annual_Salary number(10,2));
2 insert into employee values(1,'Surya','Kumar','abc@gmail.com',7904656343,'05/03/2024',35,15000,12,1,10000)
3 select employee_id,Last_name,"SALARY"*12 "ANNUAL_SALARY" from employee
4 select*from employee
5 select employee_id as employee_number,Last_name,Job_id,Hire_date from employee;
6 select Hire_date as Star_date from employee
7 select distinct job_id from employee
```

The results section shows the output:

JOB_ID
35

1 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||', '||job_id as "Employee_and_title" from employees;
```

### OUTPUT:

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

```
1 create table employee (employee_id int,First_name varchar(20),Last_name varchar(20),Email varchar(20),Phone_number number(10),Hire_date varchar(20), Job_id int, Salary int, Manager_id int, Department_id Int, Annual_Salary number(10,2));
2 insert into employee values(1,'Surya','Kumar','abc@gmail.com',7904656343,'05/03/2024',35,15000,12,1,10000)
3 select employee_id,Last_name,"SALARY"*12 "ANNUAL_SALARY" from employee
4 select*from employee
5 select employee_id as employee_number,Last_name,Job_id,Hire_date from employee;
6 select Hire_date as Start_date from employee
7 select distinct job_id from employee
8 select Last_name ||','||Job_id as "Employee_and_title" from employee
```

The results pane shows the output of the query:

Employee_and_title
Kumar,35

Copyright © 1999, 2025, Oracle and/or its affiliates.

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

### OUTPUT:

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

```
1 create table employee (employee_id int,First_name varchar(20),Last_name varchar(20),Email varchar(20),Phone_number number(10),Hire_date varchar(20), Job_id int, Salary int, Manager_id int, Department_id Int, Annual_Salary number(10,2));
2 insert into employee values(1,'Surya','Kumar','abc@gmail.com',7904656343,'05/03/2024',35,15000,12,1,10000)
3 select employee_id,Last_name,"SALARY"*12 "ANNUAL_SALARY" from employee
4 select*from employee
5 select employee_id as employee_number,Last_name,Job_id,Hire_date from employee;
6 select Hire_date as Start_date from employee
7 select distinct job_id from employee
8 select Last_name ||','||Job_id as "Employee_and_title" from employee
9 select employee_id||','||First_name||','||Last_name||','||Email||','||Phone_number||','||Hire_date||','||Job_id||','||Salary||','||Manager_id||','||Department_id as "the_output" from employee
```

The results pane shows the output of the query:

the_output
1,Surya,Kumar,abc@gmail.com,7904656343,05/03/2024,35,15000,12,1,10000

Copyright © 1999, 2025, Oracle and/or its affiliates.

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

**RESULT:**

# RESTRICTING AND SORTING DATA

EX\_NO:5

DATE:

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

QUERY:

```
select last_name,salary from employee where salary>12000
```

OUTPUT:

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

```
1 insert into employee values(5,'Rohit','Sharma','sharma@gmail.com',9526149757,'05/11/2024',39,22000,15,6,390000)
2 select last_name,salary from employee where salary>12000
```

The results window displays the following data:

LAST_NAME	SALARY
Abhijith	20000
Kumar	15000
Sharma	22000
Kumar	15000
Kohli	21000

5 rows returned in 0.00 seconds

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

QUERY:

```
select last_name,department_id from employee where employee_id=176
```

OUTPUT:

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

```
1 select last_name,department_id from employee where employee_id=176
```

The results window displays the following data:

LAST_NAME	DEPARTMENT_ID
Kohli	8

1 rows returned in 0.01 seconds

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

#### QUERY:

```
select last_name,salary from employee where salary<5000 or salary>12000
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Worksheet interface. The query `select last\_name,salary from employee where salary<5000 or salary>12000` is entered in the command field. The results are displayed in a table with columns `LAST\_NAME` and `SALARY`. The output shows five rows of data.

LAST_NAME	SALARY
Abhijith	20000
Kumar	15000
Sharma	22000
Kumar	15000
Kohli	21000

5 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 last_name,job_id,Hire_date from employee where hire_date between '2/20/1998' and '5/01/1998'  
order by hire_date
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Worksheet interface. The query `select last\_name,job\_id,Hire\_date from employee where hire\_date between '2/20/1998' and '5/01/1998' order by hire\_date` is entered in the command field. The results are displayed in a table with columns `LAST\_NAME`, `JOB\_ID`, and `HIRE\_DATE`. The output shows two rows of data.

LAST_NAME	JOB_ID	HIRE_DATE
Kumar	36	03/24/1998
Kumar	35	04/05/1998

2 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 last_name,department_id from employee where department_id between 20 and 50 order by first_name
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top bar includes tabs for 'SQL Commands' and 'Results'. The 'Results' tab is selected. The SQL command entered is: `select last_name,department_id from employee where department_id between 20 and 50 order by first_name`. The results table has two columns: 'LAST\_NAME' and 'DEPARTMENT\_ID'. The data returned is:

LAST_NAME	DEPARTMENT_ID
Abhijith	21
Sharma	45
Kumar	34

Below the table, it says '3 rows returned in 0.01 seconds'.

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

**QUERY:**

```
select last_name as "EMPLOYEE",salary as "MONTHLY SALARY" from employee where department_id between 20 and 50 and salary between 5000 and 12000 order by last_name
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top bar includes tabs for 'SQL Commands' and 'Results'. The 'Results' tab is selected. The SQL command entered is: `select last_name as "EMPLOYEE",salary as "MONTHLY SALARY" from employee where department_id between 20 and 50 and salary between 5000 and 12000 order by last_name`. The results table has two columns: 'EMPLOYEE' and 'MONTHLY SALARY'. The data returned is:

EMPLOYEE	MONTHLY SALARY
Abhijith	6000
Kumar	11000

Below the table, it says '2 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 last_name,salary from employee where hire_date like '%1994'
```

#### OUTPUT:

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

```
1 select last_name,salary from employee where hire_date like '%1994'
```

The results table displays one row:

LAST_NAME	SALARY
Sharma	22000

1 rows returned in 0.00 seconds

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

#### QUERY:

```
select last_name,job_id from employee where manager_id is null
```

#### OUTPUT:

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

```
1 select last_name,job_id from employee where manager_id is null
```

The results table displays one row:

LAST_NAME	JOB_ID
Kumar	36

1 rows returned in 0.01 seconds

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

### QUERY:

```
select last_name,salary,commission from employee where commission is not null order by salary,commission desc
```

### OUTPUT:

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

```
1 select last_name,salary,commission from employee where commission is not null order by salary,commission desc
```

The results section displays the following data:

LAST_NAME	SALARY	COMMISSION
Abhijith	6000	1000
Kumar	11000	420
Kumar	15000	3400
Kohli	21000	1800

4 rows returned in 0.01 seconds

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

### QUERY:

```
select last_name from employee where last_name like '%__a'
```

### OUTPUT:

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

```
1 select last_name from employee where last_name like '%__a'
```

The results section displays the following data:

LAST_NAME
Sharma

1 rows returned in 0.01 seconds

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

#### QUERY:

```
select last_name from employee where last_name like '%a%' and last_name like '%e%'
```

#### OUTPUT:

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

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

The results table shows one row returned:

LAST_NAME
abhijeeth

1 rows returned in 0.00 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 last_name,job_name,salary from employee where job_name in ('sales executive','stock clerk') and salary not in(2500,3500,7000)
```

#### OUTPUT:

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

```
1 select last_name,job_name,salary from employee where job_name in ('sales executive','stock clerk') and salary not in(2500,3500,7000)
```

The results table shows two rows returned:

LAST_NAME	JOB_NAME	SALARY
abhijeeth	sales executive	6000
Kumar	stock clerk	11000

2 rows returned in 0.01 seconds

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

### QUERY:

```
select last_name,salary,commission from employee where commission=salary*0.20
```

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The query `select last_name,salary,commission from employee where commission=salary*0.20` is entered in the command line. The results pane displays a single row: Sharma, 22000, 4400. The bottom status bar indicates "1 rows returned in 0.01 seconds".

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

### RESULT :

# SINGLE ROW FUNCTIONS

EX\_NO:6

DATE:

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

QUERY:

```
select sysdate from dual;
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Commands interface. The query `select sysdate from dual` is entered in the command field. The results show a single row with the column name 'SYSDATE' and the value '03/12/2024'. The interface includes tabs for Results, Explain, Describe, Saved SQL, and History. The bottom status bar shows the URL, session ID, and locale information.

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

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

OUTPUT:

The screenshot shows the Oracle APEX SQL Commands interface. The query `alter table employee add New\_salary number(10)` followed by `update employee set New\_salary=salary+salary\*0.155` and `select employee\_id, last\_name, salary, New\_salary from employee` is entered in the command field. The results show a report with columns 'EMPLOYEE\_ID', 'LAST\_NAME', 'SALARY', and 'NEW\_SALARY'. The data includes rows for employees with IDs 3, 2, 5, 1, and 176. The bottom status bar shows the URL, session ID, and locale information.

**3.** Modify your query lab\_03\_02.sql to add a column that subtracts the old salary from the new salary. Label the column Increase.

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL developer interface. The top bar includes 'Language' set to 'SQL', 'Rows' set to '10', and 'Run' button. The SQL command entered is: `1 select (New_Salary - salary)"Increase" from employee`. The results section shows a single column named 'Increase' with five rows of data: 930, 2325, 3410, 1705, and 3255. Below the results, it says '5 rows returned in 0.02 seconds'. The bottom status bar shows the connection information '220701003@rajalakshmi.edu.in seditya03 en' and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' The version 'Oracle APEX 23.2.4' is also visible.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL developer interface. The top bar includes 'Language' set to 'SQL', 'Rows' set to '10', and 'Run' button. The SQL command entered is: `1 select initcap(last_name), length(last_name) as "Length_of_last_name" from employees where last_name 2 like 'J%' or last_name like 'A%' or last_name like 'M%' order by last_name asc;`. The results section shows two columns: 'INITCAP(LAST\_NAME)' and 'Length\_of\_last\_name'. The data rows are Abhijeeth (length 9) and Mohan (length 5). Below the results, it says '2 rows returned in 0.01 seconds'. The bottom status bar shows the connection information '220701003@rajalakshmi.edu.in seditya03 en' and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' The version 'Oracle APEX 23.2.4' is also visible.

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 last\_name from employee where last\_name like 'H%'

**OUTPUT:**

The screenshot shows the Oracle APEX SQL workspace interface. The top navigation bar includes 'Language' set to 'SQL', 'Rows' set to '10', and buttons for 'Clear Command' and 'Find Tables'. On the right are 'Save' and 'Run' buttons. Below the toolbar is a search bar with icons for refresh, search, and table. The main area contains the SQL command:

```
1 select last_name from employee where last_name like 'H%'
```

Below the command is a results grid with the following data:

LAST_NAME
Harsha
Harris

Text at the bottom of the results pane indicates '2 rows returned in 0.00 seconds' and a 'Download' link. The footer of the page shows the user information '220701003@rejalekshmi.edu.in aaditya03 en', the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.', and the version 'Oracle APEX 23.2.4'.

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL workspace interface. The top navigation bar includes 'Language' set to 'SQL', 'Rows' set to '10', and buttons for 'Clear Command' and 'Find Tables'. On the right are 'Save' and 'Run' buttons. Below the toolbar is a search bar with icons for refresh, search, and table. The main area contains the SQL command:

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

Below the command is a results grid with the following data:

LAST_NAME	MONTHS_WORKED
Kohli	291
Mohan	316
Harris	316
Abhijeeth	318
Harsha	364

Text at the bottom of the results pane indicates '5 rows returned in 0.01 seconds' and a 'Download' link. The footer of the page shows the user information '220701003@rejalekshmi.edu.in aaditya03 en', the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.', and the version 'Oracle APEX 23.2.4'.

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

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL developer interface. The top bar includes 'Language' (SQL), 'Rows' (10), 'Clear Command', 'Find Tables', 'Save', and 'Run' buttons. The SQL editor contains the query:

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

The results tab is selected, showing the output:

DREAM_SALARIES	
Abhijeeth	earns 6000 monthly but wants 18000
Mohan	earns 15000 monthly but wants 45000
Harsha	earns 22000 monthly but wants 66000
Harris	earns 11000 monthly but wants 33000
Kohli	earns 21000 monthly but wants 63000

Below the results, it says '5 rows returned in 0.01 seconds'. The bottom status bar shows the user '220701003@rajalakshmi.edu.in' and 'saditya03', along with copyright information 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

**8.** Create a query to display the last name and salary for all employees. Format the salary to be

15 characters long, left-padded with the \$ symbol. Label the column SALARY.

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL developer interface. The top bar includes 'Language' (SQL), 'Rows' (10), 'Clear Command', 'Find Tables', 'Save', and 'Run' buttons. The SQL editor contains the query:

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

The results tab is selected, showing the output:

LAST_NAME	SALARY
Abhijeeth	\$\$\$\$\$\$\$\$\$\$6000
Mohan	\$\$\$\$\$\$\$\$\$\$15000
Harsha	\$\$\$\$\$\$\$\$\$\$22000
Harris	\$\$\$\$\$\$\$\$\$\$11000
Kohli	\$\$\$\$\$\$\$\$\$\$21000

Below the results, it says '5 rows returned in 0.01 seconds'. The bottom status bar shows the user '220701003@rajalakshmi.edu.in' and 'saditya03', along with copyright information 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL workspace interface. The SQL command is displayed in the top pane:

```
1 SELECT last_name,hire_date,TO_CHAR(hire_date,'Day') as Day from employee order by  
2 TO_CHAR(hire_date,'Day');
```

The results are shown in the bottom pane, titled 'Results'. The table has three columns: LAST\_NAME, HIRE\_DATE, and DAY. The data is as follows:

LAST_NAME	HIRE_DATE	DAY
Abhijeeth	02/02/1998	Monday
Kohli	04/03/2000	Monday
Harris	04/05/1998	Sunday
Mohan	03/24/1998	Tuesday
Harsha	05/03/1994	Tuesday

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

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL workspace interface. The SQL command is displayed in the top pane:

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

The results are shown in the bottom pane, titled 'Results'. The table has three columns: LAST\_NAME, HIRE\_DATE, and REVIEW. The data is as follows:

LAST_NAME	HIRE_DATE	REVIEW
Abhijeeth	02/02/1998	Monday, the 03 of August, 1998
Mohan	03/24/1998	Monday, the 28 of September, 1998
Harsha	05/03/1994	Monday, the 07 of November, 1994
Harris	04/05/1998	Monday, the 12 of October, 1998
Kohli	04/03/2000	Monday, the 09 of October, 2000

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

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

**RESULT:**

# DISPLAYING DATA FROM MULTIPLE TABLES

**EX\_NO:7**

**DATE:**

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

**QUERY:**

```
Select e.last_name,e.department_id,d.deptid from employee e,mydept d where e.department_id=d.deptid;
```

**OUTPUT:**

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

```
1 Select e.last_name,e.department_id,d.deptid from employee e,mydept d where e.department_id=d.deptid;
```

The results table displays the following data:

LAST_NAME	DEPARTMENT_ID	DEPTID
Abhijeeth	80	80
Pardip	78	78
Harsha	80	80
Kohli	80	80

4 rows returned in 0.01 seconds

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

**QUERY:** select distinct job\_id,location\_id from employee e,mydept d where e.department\_id=d.deptid and e.department\_id=80;

**OUTPUT:**

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

```
1 select distinct job_id,location_id from employee e,mydept d where e.department_id=d.deptid and e.department_id=80;
```

The results table displays the following data:

JOB_ID	LOCATION_ID
36	2
58	7
24	6

3 rows returned in 0.01 seconds

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

#### QUERY:

```
Select e.last_name,e.department_id,d.dept_name,d.loc_id,l.city from employee e,mydept d,location l  
where e.department_id=d.deptid and d.loc_id=l.locationid and e.commission is not null;
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile 'A0 Aaditya 03 aaditya03'. The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the query is displayed:

```
1 Select e.last_name,e.department_id,d.dept_name,d.loc_id,l.city from employee e,mydept d,location l where  
2 |e.department_id=d.deptid and d.loc_id=l.locationid and e.commission is not null;
```

Under 'Results', the output is a table:

LAST_NAME	DEPARTMENT_ID	DEPT_NAME	LOC_ID	CITY
Abhijeeth	80	CSE	1	Chennai
Abhijeeth	80	CSE	1	Chennai
Pardip	78	CSBS	3	Bengaluru
Harsha	80	CSE	1	Chennai
Harsha	80	CSE	1	Chennai
Kohli	80	CSE	1	Chennai

At the bottom, it shows the URL '220701003@rajalakshmi.edu.in', the schema 'aaditya03', and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' The status bar indicates 'Oracle APEX 23.2.4'.

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

#### QUERY:

```
Select employee.last_name,mydept.dept_name from employee,mydept where  
employee.department_id=mydept.deptid and last_name like '%a%'
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile 'A0 Aaditya 03 aaditya03'. The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the query is displayed:

```
1 Select employee.last_name,mydept.dept_name from employee,mydept where employee.department_id=mydept.deptid and last_name like '%a%'
```

Under 'Results', the output is a table:

LAST_NAME	DEPT_NAME
Pardip	CSBS
Harsha	CSE

At the bottom, it shows the URL '220701003@rajalakshmi.edu.in', the schema 'aaditya03', and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' The status bar indicates 'Oracle APEX 23.2.4'.

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

### QUERY:

```
Select e.last_name,e.department_id,e.job_id,d.dept_name from employee e join mydept d  
on(e.department_id=d.deptid) join location on (d.loc_id=location.locationid) where  
lower(location.city)='toronto';
```

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile for 'Aaditya 03' (aaditya03). The main area has tabs for 'SQL Commands' and 'Results'. Under 'Results', there are tabs for 'Explain', 'Describe', 'Saved SQL', and 'History'. The results table has columns: LAST\_NAME, DEPARTMENT\_ID, JOB\_ID, and DEPT\_NAME. The data shows three rows for employee Pardip, each with DEPARTMENT\_ID 78, JOB\_ID 56, and DEPT\_NAME CSBS. At the bottom, it says '3 rows returned in 0.01 seconds' and has 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 last_name "Employee",employee_id "emp#",manager_name "manager",manager_id "Mgr#"  
from employee
```

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile for 'Aaditya 03' (aaditya03). The main area has tabs for 'SQL Commands' and 'Results'. Under 'Results', there are tabs for 'Explain', 'Describe', 'Saved SQL', and 'History'. The results table has columns: Employee, emp#, manager, and Mgr#. The data shows five rows with varying manager information. At the bottom, it says '5 rows returned in 0.01 seconds' and has a 'Download' link.

7. Modify lab4\_6.sql to display all employees including King, who has no manager. Order the results by the employee number.

#### QUERY:

```
select e.last_name,e.manager_name,e.department_name,d.loc_id,d.loc,e.salary from employee e join mydept d on e.employee_id=d.empid order by e.employee_id
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile 'Aaditya 03' (aaditya03). The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_AADITYA03'. The SQL editor contains the query: 'select e.last\_name,e.manager\_name,e.department\_name,d.loc\_id,d.loc,e.salary from employee e join mydept d on e.employee\_id=d.empid order by e.employee\_id'. The results section displays a single row of data:

LAST_NAME	MANAGER_NAME	DEPARTMENT_NAME	LOC_ID	LOC	SALARY
Abhijeeth	Aadhitya	Commerce	3	Bangaluru	6000

Below the table, it says '1 rows returned in 0.00 seconds' and there is a 'Download' link. The bottom status bar includes JavaScript code, a copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.', and the version 'Oracle APEX 23.2.4'.

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

**QUERY:** select e.department\_id "Dept",e.last\_name "colleague" from employee e join employee c on (e.department\_id=c.department\_id) where e.employee\_id <> c.employee\_id order by e.department\_id,e.last\_name,c.last\_name;

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile 'Aaditya 03' (aaditya03). The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_AADITYA03'. The SQL editor contains the query: 'select e.department\_id "Dept",e.last\_name "colleague" from employee e join employee c on (e.department\_id=c.department\_id) where e.employee\_id <> c.employee\_id order by e.department\_id,e.last\_name,c.last\_name;'. The results section displays a table with columns 'Dept' and 'colleague':

Dept	colleague
80	Abhijeeth
80	Abhijeeth
80	Harsha
80	Harsha
80	Kohli
80	Kohli

The bottom status bar includes JavaScript code, a copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.', and the version 'Oracle APEX 23.2.4'.

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

**QUERY:** `SELECT e.last_name, e.job_id, d.dept_name, e.salary, j.grade_level FROM employee e JOIN mydept d ON (e.department_id = d.deptid) JOIN job_grade j ON (e.salary BETWEEN j.lowest_sal AND j.highest_sal);`

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The top right shows the user 'Aaditya 03' (aaditya03). The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_AADITYA03'. The SQL editor contains the following code:

```
1 SELECT e.last_name, e.job_id, d.dept_name, e.salary, j.grade_level FROM employee e JOIN mydept d ON (e.department_id = d.deptid)
2 JOIN job_grade j ON (e.salary BETWEEN j.lowest_sal AND j.highest_sal);
```

The results section shows a table with the following data:

LAST_NAME	JOB_ID	DEPT_NAME	SALARY	GRADE_LEVEL
Harsha	58	CSE	22000	2nd
Kohli	36	CSE	21000	2nd
Abhijeeth	24	CSE	6000	5th
Pardip	56	CSBS	15000	4th

Below the table, it says '4 rows returned in 0.02 seconds' and has a 'Download' link. The bottom status bar includes 'javascript:setValue('P1003\_SQL\_COMMAND1',');\$x('P1003\_SQL\_COMMAND1').foc...' and 'Copyright © 1999, 2023, Oracle and/or its affiliates.' The version 'Oracle APEX 23.2.4' is also visible.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The top right shows the user 'Aaditya 03' (aaditya03). The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_AADITYA03'. The SQL editor contains the following code:

```
1 SELECT e.last_name, e.hire_date FROM employee e, employee davies WHERE davies.last_name = 'Davies' AND davies.hire_date < e.hire_date;
```

The results section shows a table with the following data:

LAST_NAME	HIRE_DATE
Pardip	03/24/1998
Kohli	04/03/2000

Below the table, it says '2 rows returned in 0.01 seconds' and has a 'Download' link. The bottom status bar includes 'javascript:setValue('P1003\_SQL\_COMMAND1',');\$x('P1003\_SQL\_COMMAND1').foc...' and 'Copyright © 1999, 2023, Oracle and/or its affiliates.' The version 'Oracle APEX 23.2.4' is also visible.

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

## QUERY:

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

## 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. A search bar is present, along with user information for 'Aaditya 03' (aaditya03). The main workspace displays the following SQL code:

```
1 SELECT e.last_name AS Employee, e.hire_date AS Emp_Hired,e.manager_name AS Manager,
2 m.hire_date AS Mgr_Hired FROM employee e JOIN employee m ON e.manager_name =
m.manager_name WHERE e.hire_date < m.hire_date;
```

The results section shows a single row of data:

EMPLOYEE	EMP_HIRED	MANAGER	MGR_HIRED
parker	04/05/1996	Balaji	05/03/1997

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

At the bottom of the page, there is some JavaScript code: `javascript:setValue('P1003_SQL_COMMAND1','');$x('P1003_SQL_COMMAND1').foc...`. The page footer indicates 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

## RESULT

# AGGREGATING DATA USING GROUP FUNCTIONS

EX\_NO : 8

DATE:

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

True/False

**TRUE**

2. Group functions include nulls in calculations.

True/False

**FALSE**

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

True/False

**FALSE**

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

QUERY:

```
Select round(max(salary),0)"Maximum",round(min(salary),0)"Minimum",round(sum(salary),0)  
"sum",round(avg(salary),0)"average" from employee
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is logged in as 'Aaditya 03' (aaditya03). The SQL Commands tab is active, displaying the following SQL query:

```
1 select round(max(salary),0)"Maximum",round(min(salary),0)"Minimum",round(sum(salary),0)"sum",round(avg(salary),0)"average" from employee
```

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

Maximum	Minimum	sum	average
22000	6000	75000	15000

Below the table, it says '1 rows returned in 0.05 seconds'. At the bottom, there are links for 'javascript:setValue('P1003\_SQL\_COMMAND1',');\$x('P1003\_SQL\_COMMAND11).focus();' and 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

### QUERY:

```
Select job_id,round(max(salary),0)"Maximum",round(min(salary),0)"Minimum",round(sum(salary),0)"sum",round(avg(salary),0)"average" from employee group by job_id;
```

### OUTPUT:

JOB_ID	Maximum	Minimum	sum	average
58	22000	22000	22000	22000
36	21000	21000	21000	21000
56	15000	15000	15000	15000
24	6000	6000	6000	6000
47	11000	11000	11000	11000

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

### QUERY:

```
Select job_id,count(*) from employee where job_id='47' group by job_id
```

### OUTPUT:

JOB_ID	COUNT(*)
47	1

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

### QUERY:

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

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is logged in as Aaditya 03 (aaditya03). The SQL Commands tab is active, showing the query: `select count(distinct manager_id )"Number of managers" from employee`. The Results tab displays the output: `3`, indicating there are three distinct managers. The bottom status bar shows the copyright notice "Copyright © 1999, 2023, Oracle and/or its affiliates." and the version "Oracle APEX 23.2.4".

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

### QUERY:

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

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is logged in as Aaditya 03 (aaditya03). The SQL Commands tab is active, showing the query: `select max(salary)-min(salary)"difference" from employee`. The Results tab displays the output: `16000`, indicating the difference between the highest and lowest salaries. The bottom status bar shows the copyright notice "Copyright © 1999, 2023, Oracle and/or its affiliates." and the version "Oracle APEX 23.2.4".

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

**QUERY:** select manager\_id ,MIN(salary) from employee where manager\_id is not null group by manager\_id having min(salary) >6000 order by min(salary) desc;

**OUTPUT:**

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

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

The results table has two rows:

MANAGER_ID	MIN(SALARY)
17	21000
15	11000

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

**OUTPUT:**

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

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

The results table has one row:

TOTAL	1995	1996	1997	1998
5	0	1	2	1

1 rows returned in 0.01 seconds

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

### QUERY:

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

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile 'Aaditya 03' and the schema 'WKSP\_AADITYA03'. The main area is titled 'SQL Commands' with a 'Language' dropdown set to 'SQL'. Below it, there are buttons for 'Clear Command' and 'Find Tables'. The SQL code entered is:

```
1 select job_id "job", sum(decode(department_id,20,salary))"Dept20",sum (decode(department_id ,50, salary)) "dept50",
2   sum (decode(department_id ,80, salary)) "dept80",sum (decode(department_id ,90, salary)) "dept90",sum(salary) "TOTAL" from employee group by job_id
```

The results are displayed in a grid format:

job	Dept20	dept50	dept80	dept90	TOTAL
58	-	-	22000	-	22000
36	-	-	21000	-	21000
56	-	-	-	-	15000
24	-	-	6000	-	6000
47	-	-	-	-	11000

At the bottom, the footer includes user information '220701003@rajalakshmi.edu.in', session details 'aaditya03', and language 'en'. It also states 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

### QUERY:

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

### 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. On the right, a user profile for 'Aaditya 03' is shown. The main area contains a SQL command editor with the following code:

```
1  t(*) "Number of people",round(avg(salary),2) "salary" from mydept d inner join employee e on(d.deptid =e.department_id ) group by d.dept_name ,d.loc;
```

Below the editor, the results are displayed in a table:

dept_name	department location	Number of people	salary
CSBS	Bangaluru	1	15000
CSE	Chennai	3	16333.33

Text at the bottom left indicates 2 rows returned in 0.05 seconds. The bottom right shows copyright information: Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4.

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

**RESULT:**

# SUB QUERIES

EX\_NO:9

DATE:

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

**QUERY:**

```
select last_name,hire_date from employee where department_id=(select department_id from employee where last_name='Zlotkey') and last_name not in('Zlotkey')
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a user profile for 'Aaditya 03' (aaditya03). The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_AADITYA03'. The SQL editor contains the following code:

```
1 select last_name,hire_date from employee where department_id=(select department_id from employee where last_name='Zlotkey')
2 and last_name not in('Zlotkey')
```

Below the editor, the results tab is selected, showing the output of the query:

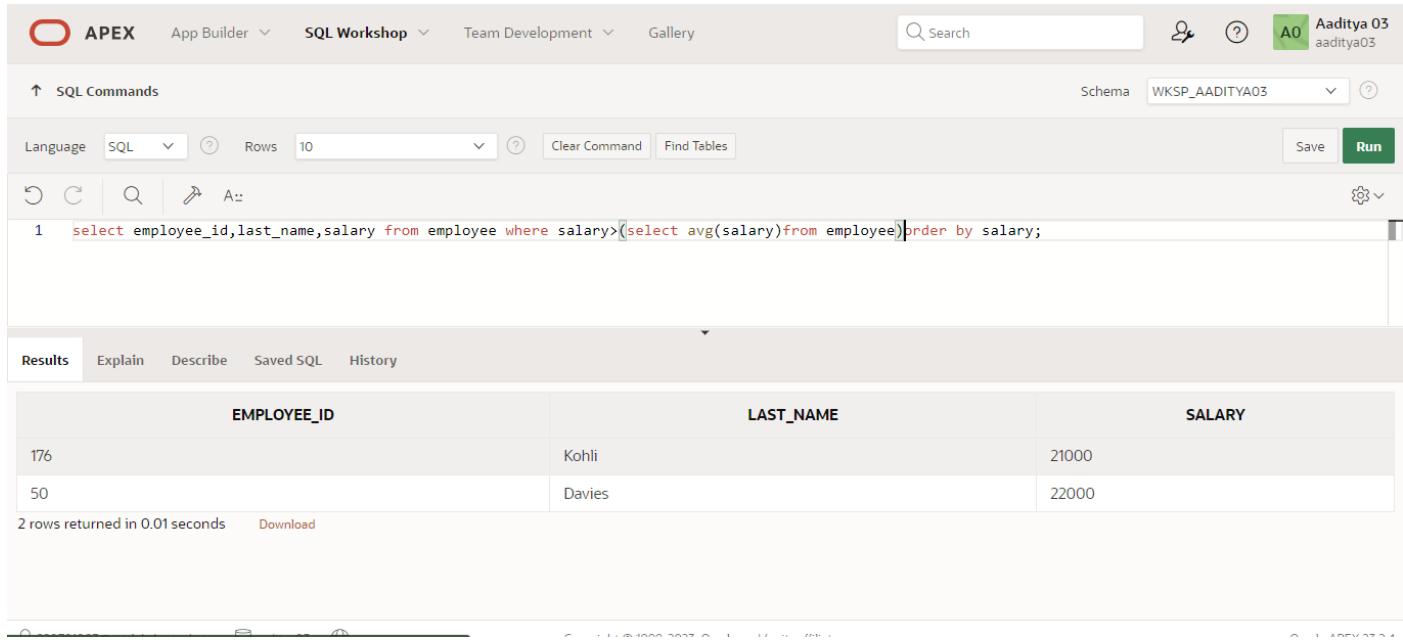
LAST_NAME	HIRE_DATE
Davies	05/03/1997
Kohli	04/03/2000

At the bottom, it says '2 rows returned in 0.00 seconds' and has a 'Download' link. The footer includes user information (220701003@rajalakshmi.edu.in, aaditya03, en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and version information (Oracle APEX 23.2.4).

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

**QUERY:** select employee\_id, last\_name, salary from employee where salary > (select avg(salary) from employee) order by salary;

**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 'Aaditya 03'. The SQL Commands panel shows the following query:

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

The Results tab displays the output:

EMPLOYEE_ID	LAST_NAME	SALARY
176	Kohli	21000
50	Davies	22000

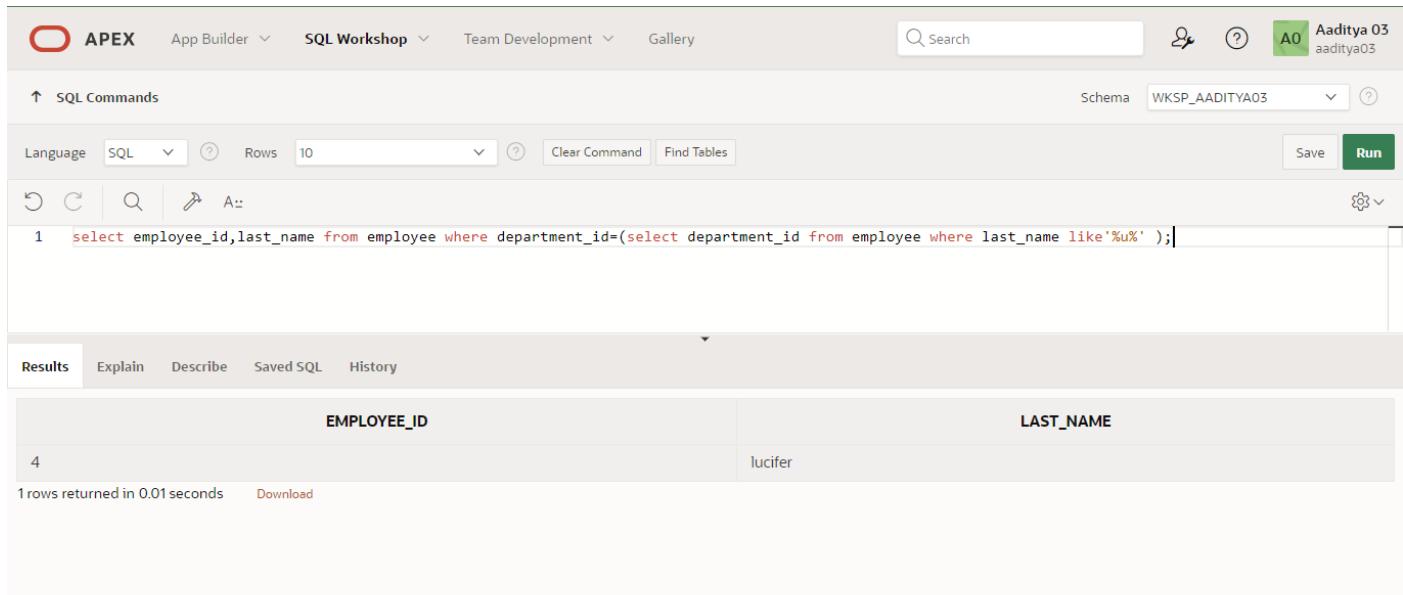
Below the table, it says '2 rows returned in 0.01 seconds' and has a 'Download' link.

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

**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 'Aaditya 03'. The SQL Commands panel shows the following query:

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

The Results tab displays the output:

EMPLOYEE_ID	LAST_NAME
4	lucifer

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 last_name,department_id,job_id from employee where department_id=(select deptid from mydept where location_id=1700);
```

#### OUTPUT:

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

```
1 select last_name,department_id,job_id from employee where department_id=(select deptid from mydept where location_id=1700);
```

The results section shows "no data found".

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

#### QUERY:

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

#### OUTPUT:

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

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

The results section shows a table with two rows:

LAST_NAME	SALARY
Davies	22000
lucifer	11000

2 rows returned in 0.01 seconds    Download

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

#### QUERY:

```
select department_id, last_name, job_id from employee where department_id in (select deptid from mydept where dept_name='Executive');
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes tabs for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for Aaditya 03 (aaditya03). The main area has a search bar and a toolbar with options like Language (SQL selected), Rows (set to 10), Clear Command, Find Tables, Save, and Run. The SQL command entered is:

```
1 select department_id, last_name, job_id from employee where department_id in (select deptid from mydept where dept_name='Executive');
```

The results section shows a table with three rows:

DEPARTMENT_ID	LAST_NAME	JOB_ID
78	Zlotkey	56
78	Davies	58
78	Kohli	36

Below the table, it says "3 rows returned in 0.00 seconds" and there is a "Download" link. At the bottom, it shows the user's email (220701003@rajalakshmi.edu.in) and name (aaditya03), along with copyright information (Copyright © 1999, 2023, Oracle and/or its affiliates) and the version (Oracle APEX 23.2.4).

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

#### QUERY:

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

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface, similar to the previous one. The top navigation bar and user profile are the same. The SQL command entered is:

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

The results section shows a table with three rows:

EMPLOYEE_ID	LAST_NAME	SALARY
3	Zlotkey	15000
50	Davies	22000
176	Kohli	21000

Below the table, it says "3 rows returned in 0.01 seconds" and there is a "Download" link. At the bottom, it shows the user's email (220701003@rajalakshmi.edu.in) and name (aaditya03), along with copyright information (Copyright © 1999, 2023, Oracle and/or its affiliates) and the version (Oracle APEX 23.2.4).

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

**RESULT:**



## USING THE SET OPERATORS

EX.NO:10

DATE:

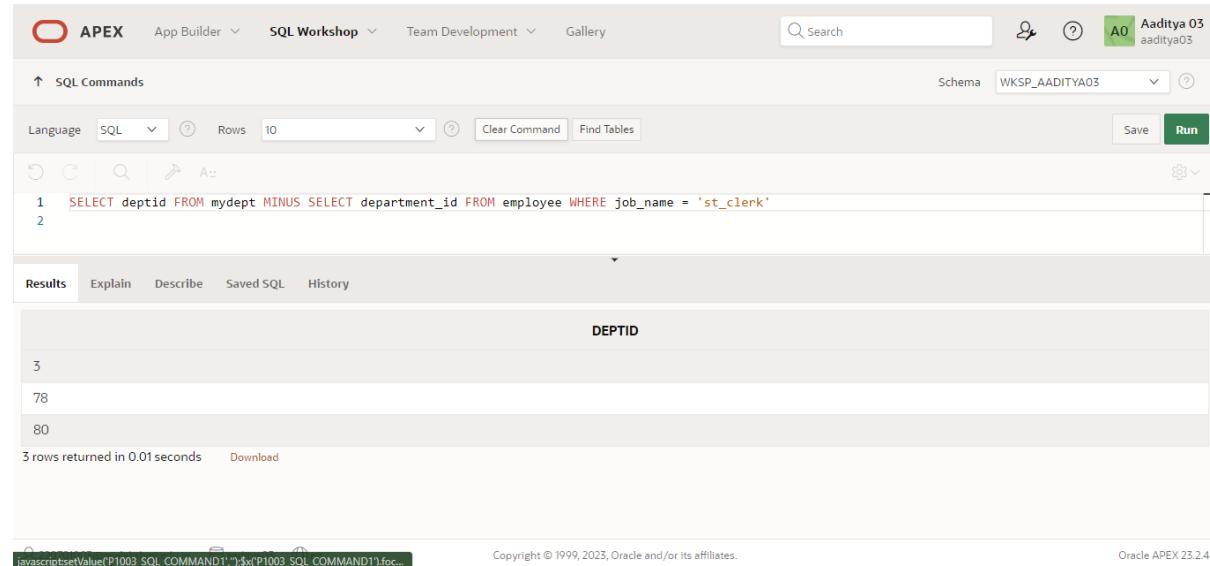
Find the Solution for the following:

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 deptid FROM mydept MINUS SELECT department_id FROM employee WHERE job_name = 'st_clerk';
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is logged in as Aaditya 03 (aaditya03). The main workspace is titled "SQL Commands". The query entered is:

```
1  SELECT deptid FROM mydept MINUS SELECT department_id FROM employee WHERE job_name = 'st_clerk'
2
```

The results section displays the output:

DEPTID
3
78
80

Below the results, it says "3 rows returned in 0.01 seconds" and provides a "Download" link. The bottom footer of the page includes copyright information for Oracle and the text "Oracle APEX 23.2.4".

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

**QUERY:** SELECT country\_id,country\_name FROM country MINUS SELECT l.country\_id,c.country\_name FROM location l, country c WHERE l.country\_id =c.country\_id;

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'Aaditya 03' (aaditya03). The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_AADITYA03'. The command input field contains the following SQL query:

```
1 SELECT country_id,country_name FROM country MINUS SELECT l.country_id,c.country_name FROM location l, country c WHERE l.country_id =c.country_id;
```

The results section displays the output of the query:

COUNTRY_ID	COUNTRY_NAME
19	New zealand

Below the table, it says '1 rows returned in 0.02 seconds' and provides a 'Download' link. The bottom of the page includes copyright information and a footer for 'Oracle APEX 23.2.4'.

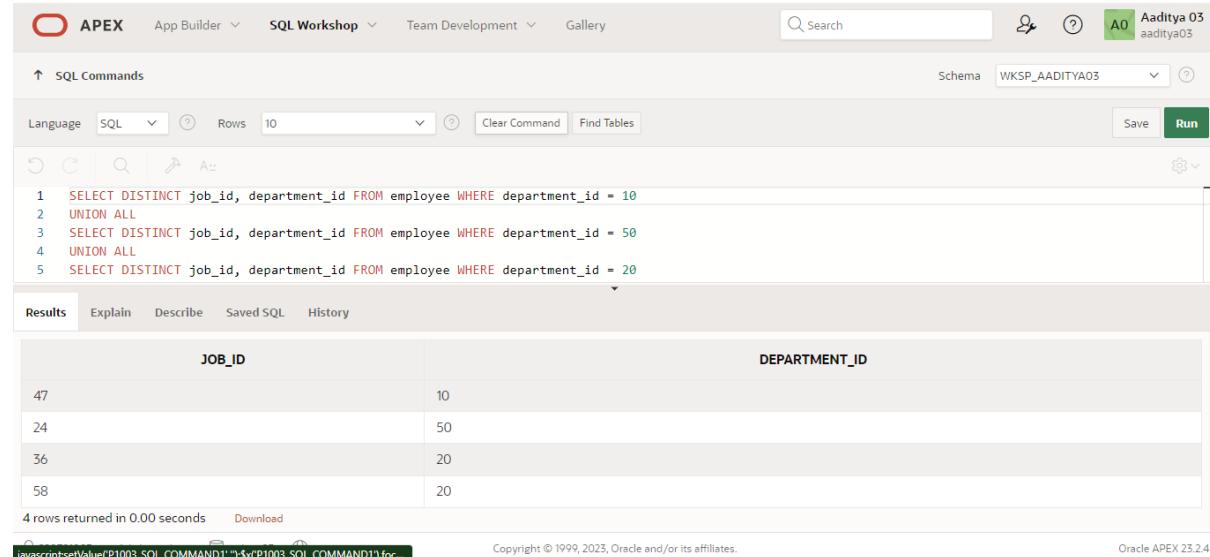
3.Produce a list of jobs for departments 10, 50, and 20, in that order. Display job ID and department ID using set operators.

**QUERY:**SELECT DISTINCT job\_id, department\_id FROM employee WHERE department\_id = 10 UNION ALL

SELECT DISTINCT job\_id, department\_id FROM employee WHERE department\_id = 50 UNION ALL

SELECT DISTINCT job\_id, department\_id FROM employee WHERE department\_id = 20

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for Aaditya 03. The main area has tabs for SQL Commands, Results (selected), Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the following code:

```
1 SELECT DISTINCT job_id, department_id FROM employee WHERE department_id = 10
2 UNION ALL
3 SELECT DISTINCT job_id, department_id FROM employee WHERE department_id = 50
4 UNION ALL
5 SELECT DISTINCT job_id, department_id FROM employee WHERE department_id = 20
```

The Results tab displays the output in a table:

JOB_ID	DEPARTMENT_ID
47	10
24	50
36	20
58	20

Below the table, it says "4 rows returned in 0.00 seconds".

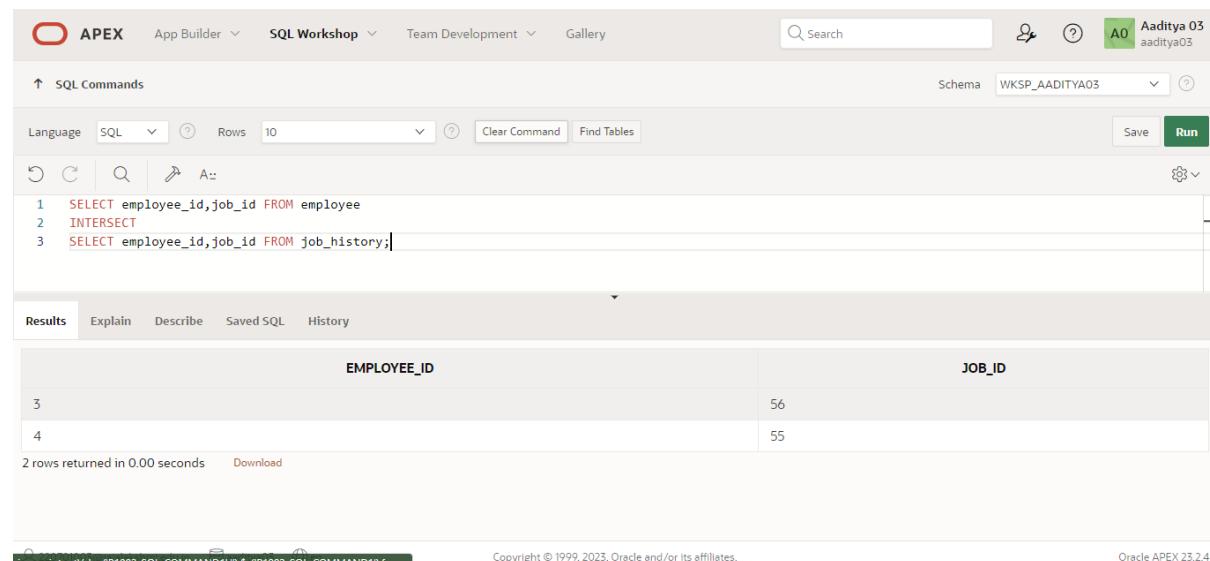
4. Create a report that lists the employee IDs and job IDs of those employees who currently have a job title that is the same as their job title when they were initially hired by the company (that is, they changed jobs but have now gone back to doing their original job).

**QUERY:** SELECT employee\_id,job\_id FROM employee

INTERSECT

SELECT employee\_id,job\_id FROM job\_history;

## OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The top right shows a user profile for 'Aaditya 03' (aaditya03). The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_AADITYA03'. The SQL editor contains the following code:

```
1 SELECT employee_id,job_id FROM employee
2 INTERSECT
3 SELECT employee_id,job_id FROM job_history;
```

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

EMPLOYEE_ID	JOB_ID
3	56
4	55

Below the table, it says '2 rows returned in 0.00 seconds' and has a 'Download' link. The bottom footer includes copyright information for Oracle and the text 'Oracle APEX 23.2.4'.

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

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

#### QUERY:

```
SELECT last_name,department_id,TO_CHAR(null) FROM employee  
UNION SELECT TO_CHAR(null),deptid,dept_name FROM mydept
```

#### 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. A search bar and a user profile for 'Aaditya 03' are also present. The main workspace has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the following code:

```
1: SELECT last_name,department_id,TO_CHAR(null) FROM employee  
2: UNION SELECT TO_CHAR(null),deptid,dept_name FROM mydept;
```

The Results tab displays the output of the query:

LAST_NAME	DEPARTMENT_ID	TO_CHAR(NULL)
Abhijeeth	50	-
Davies	20	-
Kohli	20	-
Pardip	78	-
parker	10	-
-	3	Mech
-	78	CSBS

At the bottom of the interface, there are footer links for 'Copyright © 1999, 2023, Oracle and/or its affiliates.', 'Oracle APEX 23.2.4', and user information including email (220701003@rajalakshmi.edu.in) and session details (aaditya03, en).

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

**RESULT:**

# CREATING VIEWS

EX\_NO:11

DATE:

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. The schema is set to 'WKSP\_AADITYA03'. The code entered is:

```
1 create or replace view employee_vu as select employee_id, last_name, department_id from employees
```

In the results section, it shows:

```
View created.
```

0.04 seconds

At the bottom, the footer includes the URL '220701005@rajalakshmi.edu.in', session 'aaditya03', and language 'en'. It also states 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

QUERY: select \* from employees\_vu;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. The schema is set to 'WKSP\_AADITYA03'. The code entered is:

```
1 select * from employee_vu
```

In the results section, it displays a table with the following data:

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
46	Abhijeeth	3
3	Zlotkey	78
50	Davies	78
4	lucifer	3
176	Kohli	78

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

At the bottom, the footer includes the URL '220701003@rajalakshmi.edu.in', session 'aaditya03', and language 'en'. It also states 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

3.)Select the view name and text from the USER\_VIEWS data dictionary views

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows a user profile for 'Aaditya 03'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, showing the query: 'select view\_name, text from user\_views;'. The Results tab displays the output:

VIEW_NAME	TEXT
EMPLOYEE_VU	select employee_id,last_name,department_id from employee

1 rows returned in 0.03 seconds. The bottom of the screen shows copyright information and the Oracle APEX version.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows a user profile for 'Aaditya 03'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, showing the query: 'select last\_name,department\_id from employee\_vu;'. The Results tab displays the output:

LAST_NAME	DEPARTMENT_ID
Abhijeeth	3
Zlotkey	78
Davies	78
lucifer	3
Kohli	78

5 rows returned in 0.01 seconds. The bottom of the screen shows copyright information and the Oracle APEX version.

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

#### QUERY:

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

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile 'Aaditya 03'. The main area has tabs for 'SQL Commands', 'Language' (set to SQL), 'Rows' (set to 10), and buttons for 'Clear Command', 'Find Tables', 'Save', and 'Run'. The command entered is:

```
1 create view dept50 as select employee_id EMP_NO,last_name EMPLOYEE,department_id DEPTNO from employee where department_id=50
```

Below the command, the results tab is selected, showing the message 'View created.' and a execution time of '0.05 seconds'. The bottom footer includes the user's email '220701005@rajalakshmi.edu.in', the schema 'aaditya03', and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.'

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

#### QUERY:

```
Describe dept50;
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface with the 'Describe' tab selected. The top navigation bar and right-side user information are identical to the previous screenshot. The command entered is:

```
1 describe dept50
2 
```

The results tab displays the structure of the view 'DEPT50' with three columns: 'EMP\_NO', 'EMPLOYEE', and 'DEPTNO'. The table below shows the detailed column definitions:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPT50	EMP_NO	NUMBER	22	-	0	-	✓	-	-
	EMPLOYEE	VARCHAR2	20	-	-	-	✓	-	-
	DEPTNO	NUMBER	22	-	0	-	✓	-	-

The bottom footer includes the user's email '220701005@rajalakshmi.edu.in', the schema 'aaditya03', and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.'

## 7.) Attempt to reassign Matos to department 80

### QUERY:

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

### OUTPUT:

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

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

Below the command, the results section shows:

0 row(s) updated.  
0.05 seconds

At the bottom, the footer includes user information and copyright details.

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

### QUERY:

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

### OUTPUT:

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

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

Below the command, the results section shows a table with the following data:

Employee	Department	Salary	Grades
Davies	Executive	22000	2nd
Kohli	Executive	21000	2nd
Zlotkey	Executive	15000	4th

At the bottom, the footer includes user information and copyright details.

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.

```
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

ANIMAL_ID	NAM E	LICENSE_TAG_NUMBE R	ADMIT_DATE	ADOPTION_ID	VACCINATION_DATE
101	Spot	35540	10-Oct-2004	205	12-Oct-2004

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

## 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 ASSELECT title, artistFROM
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_idFROM(SELECT dpt.department_id,
MAX(NVL(emp.salary,0)) max_dpt_salFROM departments dpt LEFT OUTER JOIN employees emp ON
dpt.department_id = emp.department_idGROUP BY dpt.department_id) dptmx LEFT OUTER JOIN
employees empm ON dptmx.department_id = empm.department_idWHERE NVL(empm.salary,0) =
dptmx.max_dpt_sal;
```

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

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

# **Indexes and Synonyms**

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

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

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

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

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

3. When will an index be created automatically?

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

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

**CREATE INDEX d\_tlg\_cd\_number\_fk\_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_idxON 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

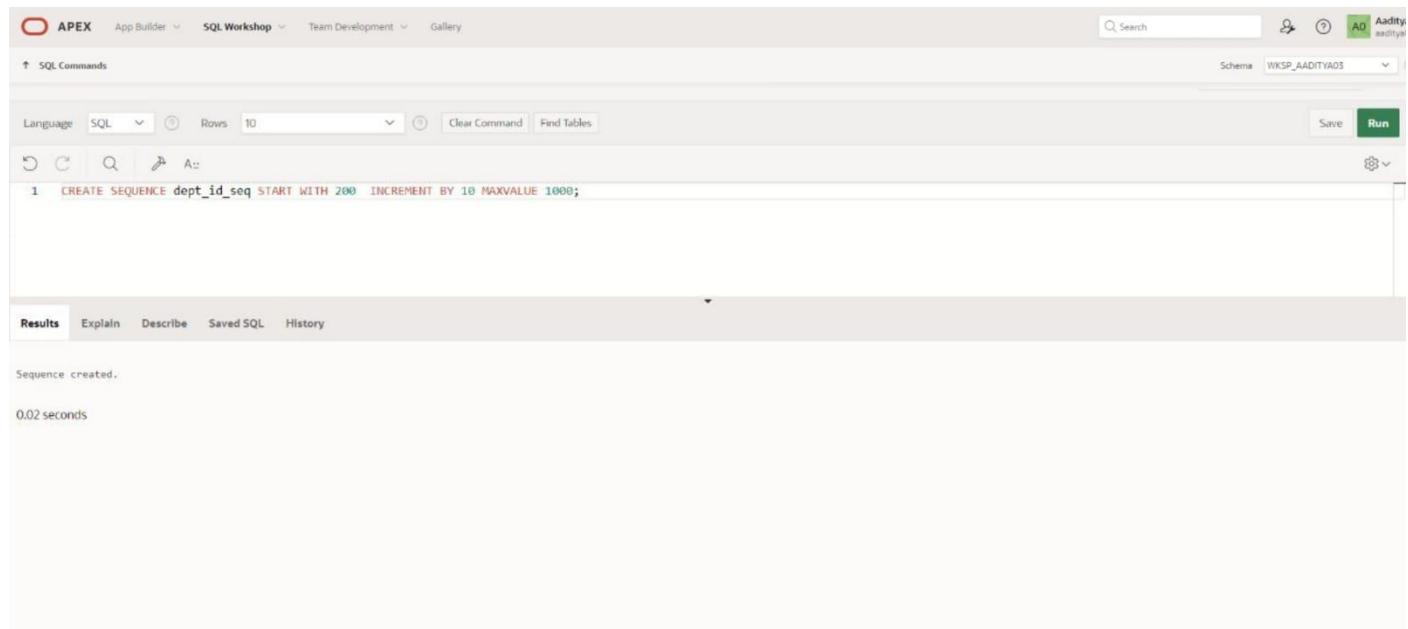
DATE:

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:**



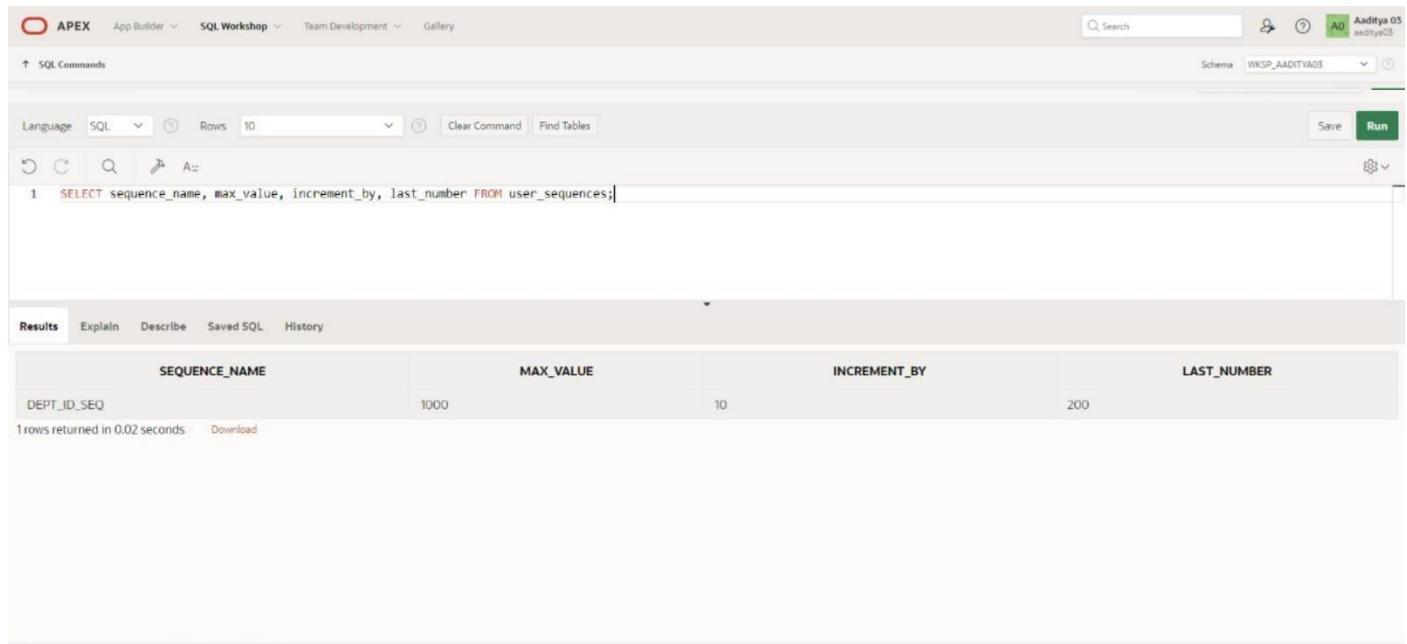
The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area displays the SQL command: 'CREATE SEQUENCE dept\_id\_seq START WITH 200 INCREMENT BY 10 MAXVALUE 1000;'. Below the command, the 'Results' tab is active, showing the output: 'Sequence created.' and '0.02 seconds'. The schema is set to 'WKSP\_AADITYA03'.

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

**QUERY:**

SELECT sequence\_name, max\_value, increment\_by, last\_number FROM user\_sequences;

### OUTPUT:



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

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

Below the command window, there's a results grid:

SEQUENCE_NAME	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
DEPT_ID_SEQ	1000	10	200

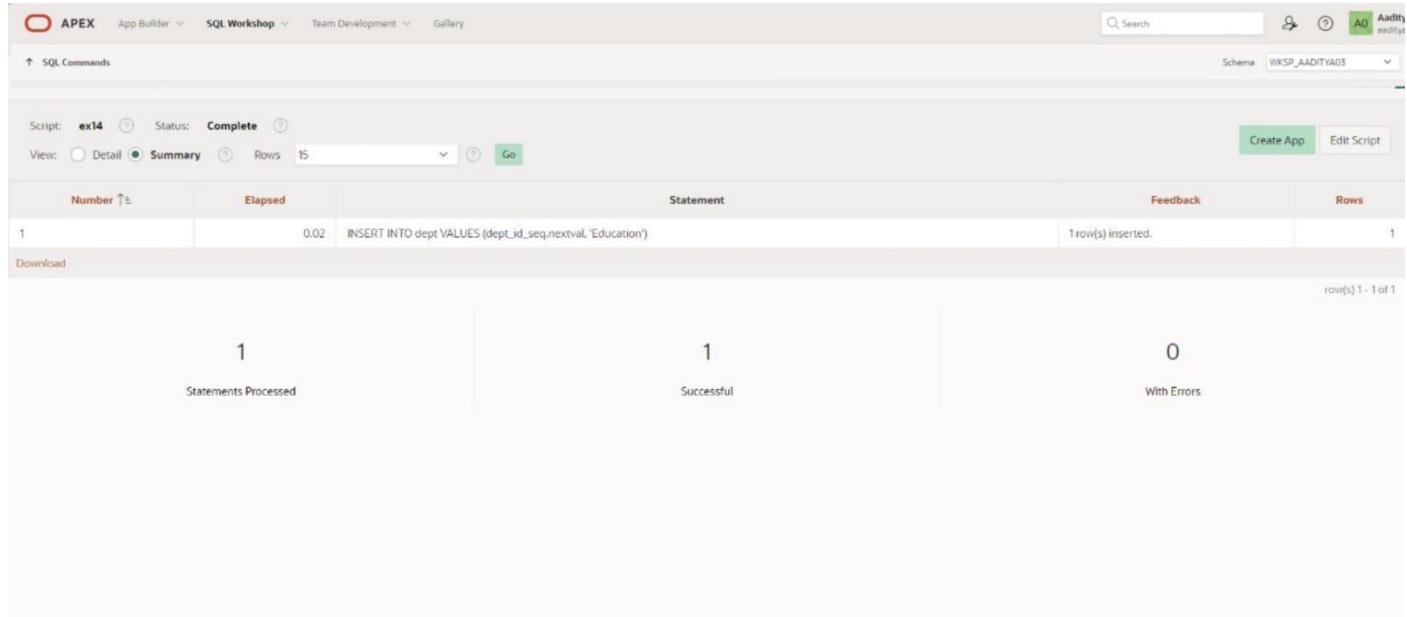
At the bottom left, it says "1 rows returned in 0.02 seconds". At the bottom right, there are "Save" and "Run" buttons.

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

### QUERY:

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

### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is active. The main area displays an execution summary for a script named 'ex14':

Script: ex14 Status: Complete

View:  Detail  Summary Rows: 15 Go

Number	Elapsed	Statement	Feedback	Rows
1	0.02	INSERT INTO dept VALUES (dept_id_seq.nextval, 'Education')	1 row(s) inserted.	1

At the bottom, there are three status indicators: "Statements Processed" (1), "Successful" (1), and "With Errors" (0).

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. In the top navigation bar, 'APEX' is selected. The main area contains the following SQL command:

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

Below the command, the results section displays the output:

Index created.  
0.03 seconds

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

**QUERY:**

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

**OUTPUT:**

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

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

Below the command, the results section displays the output:

INDEX_NAME	TABLE_NAME	UNIQUENESS
EMP_DEPT_ID_IDX	EMPLOYEES	NONUNIQUE

1 rows returned in 0.04 seconds    Download

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  
the other team.

Team 1 revokes the

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

Evaluation Procedure	Marks awarded
Query(5)	

privilege on your table from  
privilege.

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;
```

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

**RESULT:**

# PL/SQL CONTROL STRUCTURES

**EX NO:16**

**DATE:**

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

**QUERY:**

DECLARE

    incentive NUMBER(8,2);

BEGIN

    SELECT salary\*0.12 INTO incentive

    FROM employees

    WHERE employee\_id = 110;

    DBMS\_OUTPUT.PUT\_LINE('Incentive = ' || TO\_CHAR(incentive));

END;

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The top right shows the user 'Aaditya 03' and schema 'WKSP\_AADITYA03'. The main area has tabs for 'SQL Commands' (selected), 'Explain', 'Describe', 'Saved SQL', and 'History'. The SQL editor contains the following PL/SQL code:

```
1 DECLARE
2     PROCEDURE test1 (sal_achieve NUMBER)
3     IS
4         incentive NUMBER := 0;
5     BEGIN
6         IF sal_achieve > 44000 THEN
7             incentive := 1800;
8         ELSIF sal_achieve > 32000 THEN
9             incentive := 800;
10        ELSE
11            incentive := 500;
12        END IF;
13        DBMS_OUTPUT.NEW_LINE;
14        DBMS_OUTPUT.PUT_LINE (
15            'Sale achieved : ' || sal_achieve || ', incentive : ' || incentive || ',');
16    );
17    END test1;
18    BEGIN
19        test1(45000);
20        test1(36000);
21        test1(28000);
22    END;
23 /
```

The results pane shows the output of the executed code:

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

At the bottom, it says '0.01 seconds' and includes copyright information: 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

### QUERY:

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

```

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

```

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a session named 'Aaditya05' with schema 'WKSP\_AADITYA05'. The main area is a SQL editor with the following code:

```
1 DECLARE
2   v_emp_count NUMBER;
3   v_vacancies NUMBER := 45;
4 BEGIN
5   -- Count the number of employees in department 50
6   SELECT COUNT(*)
7     INTO v_emp_count
8    FROM employees
9   WHERE dept_id = 50;
10  -- Display the number of employees in department 50
11  DBMS_OUTPUT.PUT_LINE('Number of employees in department 50: ' || v_emp_count);
12
13  -- Check if there are any vacancies
14  IF v_emp_count < v_vacancies THEN
15    DBMS_OUTPUT.PUT_LINE('There are vacancies in department 50.');
16  ELSE
17    DBMS_OUTPUT.PUT_LINE('There are no vacancies in department 50.');
18  END IF;
19
20 END;
21 /

```

The results pane shows the output of the query:

```
Number of employees in department 50: 0
There are vacancies in department 50.

Statement processed.

0.02 seconds
```

At the bottom, it shows the URL '220701005@relakshmi.edu.in', the session name 'aaditya05', and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' The footer also indicates 'Oracle APEX 23.2.4'.

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

### QUERY:

DECLARE

    salary\_of\_emp NUMBER(8,2);

PROCEDURE approx\_salary (

    emp     NUMBER,

    empsal IN OUT NUMBER,

    addless   NUMBER

) IS

BEGIN

    empsal := empsal + addless;

END;

BEGIN

    SELECT salary INTO salary\_of\_emp

    FROM employees

    WHERE employee\_id = 122;

    DBMS\_OUTPUT.PUT\_LINE

    ('Before invoking procedure, salary\_of\_emp: ' || salary\_of\_emp);

    approx\_salary (100, salary\_of\_emp, 1000);

    DBMS\_OUTPUT.PUT\_LINE

    ('After invoking procedure, salary\_of\_emp: ' || salary\_of\_emp);

END;

/

### OUTPUT:

```
1  DECLARE
2      v_dept_id employees.dept_id%TYPE := 50; -- Change this to the desired department ID
3      v_dept_count NUMBER;
4      v_vacancies NUMBER := 45; -- Change this to the number of vacancies in the department
5  BEGIN
6      -- Count the number of employees in the specified department
7      SELECT COUNT(*)
8          INTO v_dept_count
9          FROM employees
10         WHERE dept_id = v_dept_id;
11
12     -- Display the count
13     DBMS_OUTPUT.PUT_LINE('Number of employees in department ' || v_dept_id || ': ' || v_dept_count);
14
15     -- Check for vacancies
16     IF v_dept_count < v_vacancies THEN
17         DBMS_OUTPUT.PUT_LINE('There are vacancies in department ' || v_dept_id || '.');
18         DBMS_OUTPUT.PUT_LINE('Number of vacancies: ' || (v_vacancies - v_dept_count));
19     ELSE
20         DBMS_OUTPUT.PUT_LINE('There are no vacancies in department ' || v_dept_id || '.');
21     END IF;
22  END;
23 /
```

Results Explain Describe Saved SQL History

Number of employees in department 50: 2  
There are vacancies in department 50.  
Number of vacancies: 43

Statement processed.

0.01 seconds

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

### QUERY:

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

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The right side shows a workspace named 'Aaditya03' with schema 'WKSP\_AADITYA03'. The main area has tabs for 'SQL Commands', 'Language' (set to SQL), 'Rows' (set to 10), and 'Clear Command'. Below these are icons for search, refresh, and run. The code editor contains a PL/SQL block:

```
1  DECLARE
2      CURSOR employee_cursor IS
3          SELECT employee_id, first_name || ' ' || last_name AS full_name, job_id, hire_date, salary
4      FROM employee;
5  BEGIN
6      -- Loop through the cursor and display employee information
7      FOR employee_rec IN employee_cursor LOOP
8          DBMS_OUTPUT.PUT_LINE('Employee ID: ' || employee_rec.employee_id);
9          DBMS_OUTPUT.PUT_LINE('Employee Name: ' || employee_rec.full_name);
10         DBMS_OUTPUT.PUT_LINE('Job Title: ' || employee_rec.job_id);
11         DBMS_OUTPUT.PUT_LINE('Hire Date: ' || TO_CHAR(employee_rec.hire_date, 'DD-MON-YYYY'));
12         DBMS_OUTPUT.PUT_LINE('Salary: ' || employee_rec.salary);
13     DBMS_OUTPUT.PUT_LINE('-----');
14  END LOOP;
15 END;
```

The results tab displays the output of the PL/SQL block, listing three employees with their details:

```
Employee ID: 50
Employee Name: Rohit Davies
Job Title: 58
Hire Date: 03-MAY-1997
Salary: 22000
-----
Employee ID: 4
Employee Name: Surya lucifer
Job Title: 55
Hire Date: 05-APR-1998
Salary: 11000
-----
Employee ID: 176
Employee Name: Virat Kohli
Job Title: 36
Hire Date: 03-APR-2000
Salary: 21000
-----
```

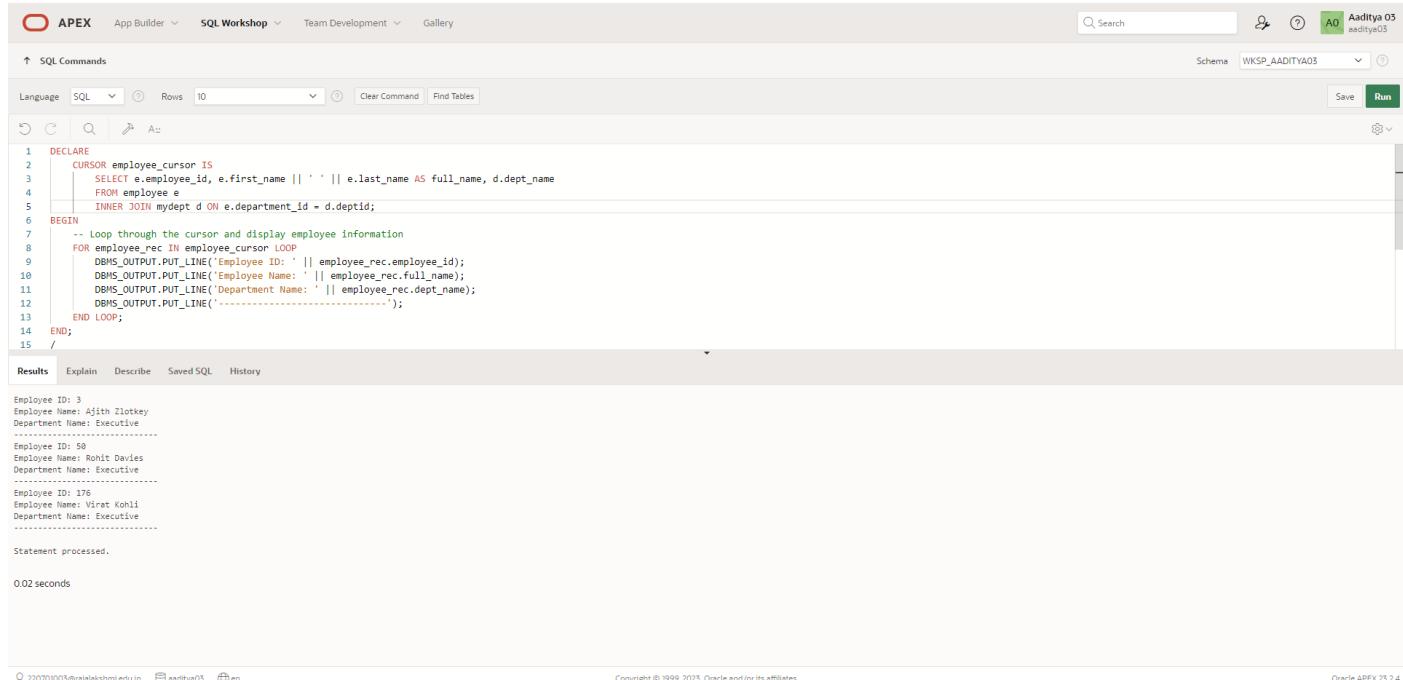
At the bottom, it says 'Statement processed.' and '0.02 seconds'.

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The top right shows the schema 'WKSP\_AADITYA03' and a user icon. The main area has tabs for 'SQL Commands' (selected), 'Explain', 'Describe', 'Saved SQL', and 'History'. The SQL editor contains the provided PL/SQL code. The results tab shows the output of the executed code, which displays employee information for three employees: Ajith Zlotkey, Rohit Davies, and Virat Kohli, each with their Employee ID, First Name, Last Name, and Department Name.

```
Employee ID: 3
Employee Name: Ajith Zlotkey
Department Name: Executive
-----
Employee ID: 50
Employee Name: Rohit Davies
Department Name: Executive
-----
Employee ID: 175
Employee Name: Virat Kohli
Department Name: Executive
-----
Statement processed.

0.02 seconds
```

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

#### QUERY:

DECLARE

```
num_small NUMBER := 8;
```

```
num_large NUMBER := 5;
```

```
num_temp NUMBER;
```

```
BEGIN
```

```
IF num_small > num_large THEN
```

```
    num_temp := num_small;
```

```
    num_small := num_large;
```

```
    num_large := num_temp;
```

```
END IF;
```

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

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

```
END;
```

```
/
```

#### OUTPUT:

```
CURSOR job_cursor IS
  SELECT job_id, MIN(salary) AS min_salary
  FROM employee
  GROUP BY job_id;
BEGIN
  -- Loop through the cursor and display job information
  FOR job_rec IN job_cursor LOOP
    DBMS_OUTPUT.PUT_LINE('Job ID: ' || job_rec.job_id);
    DBMS_OUTPUT.PUT_LINE('Minimum Salary: ' || job_rec.min_salary);
    DBMS_OUTPUT.PUT_LINE('-----');
  END LOOP;
END;
/
15
```

Results

```
Job ID: 58
Minimum Salary: 22000
-----
Job ID: 36
Minimum Salary: 21000
-----
Job ID: 56
Minimum Salary: 15000
-----
Job ID: 55
Minimum Salary: 11000
-----
Job ID: 24
Minimum Salary: 6000
-----
Statement processed.

0.02 seconds
```

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

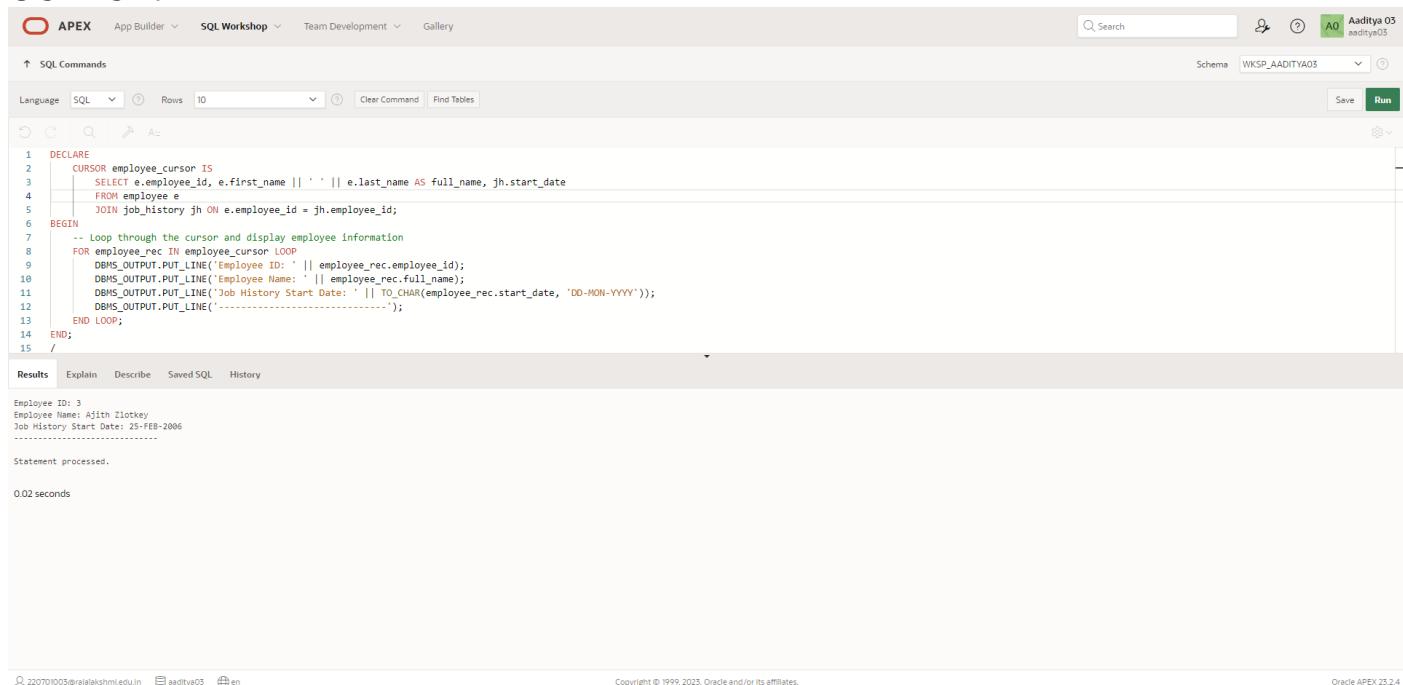
Oracle APEX 23.2.4

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

## QUERY:

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

## OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown is set to 'WKSP\_AADITYA03'. The main area displays the PL/SQL code for the 'test1' procedure. The code uses a cursor to select employee information from the 'employee' and 'job\_history' tables, then loops through the results to print each employee's ID, name, and start date. The output pane at the bottom shows the results of the execution, which include the employee details for employee IDs 144 and 145.

```
1  DECLARE
2      CURSOR employee_cursor IS
3          SELECT e.employee_id, e.first_name || ' ' || e.last_name AS full_name, jh.start_date
4          FROM employee e
5              JOIN job_history jh ON e.employee_id = jh.employee_id;
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.employee_id);
10         DBMS_OUTPUT.PUT_LINE('Employee Name: ' || employee_rec.full_name);
11         DBMS_OUTPUT.PUT_LINE('Job History Start Date: ' || TO_CHAR(employee_rec.start_date, 'DD-MON-YYYY'));
12         DBMS_OUTPUT.PUT_LINE('-----');
13     END LOOP;
14  END;
15 /
```

Employee ID: 3  
Employee Name: Ajith Zlotkey  
Job History Start Date: 25-FEB-2006  
-----  
Statement processed.  
0.02 seconds

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

**QUERY:**

DECLARE

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

IS

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

BEGIN

```
    IF sal_achieve > 44000 THEN
```

```
        incentive := 1800;
```

```
    ELSIF sal_achieve > 32000 THEN
```

```
        incentive := 800;
```

ELSE

```
        incentive := 500;
```

END IF;

```
    DBMS_OUTPUT.NEW_LINE;
```

```
    DBMS_OUTPUT.PUT_LINE (
```

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

```
    );
```

END test1;

BEGIN

```
    test1(45000);
```

```
    test1(36000);
```

```
    test1(28000);
```

END;

```
/
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there are buttons for 'Search', 'Save', 'Run', and a user profile 'Aaditya 03'. The main workspace displays the PL/SQL code for the 'test1' procedure. The code uses a cursor to select employee information from the 'employee' and 'job\_history' tables, then loops through the results to print employee details and their job history end date. The output pane shows the results of running the procedure with input values 45000, 36000, and 28000. The output includes employee IDs, names, and job history end dates. At the bottom, it shows the statement was processed in 0.01 seconds.

```
1  DECLARE
2      CURSOR employee_cursor IS
3          SELECT e.employee_id, e.first_name || ' ' || e.last_name AS full_name, jh.end_date
4          FROM employee e
5              JOIN job_history jh ON e.employee_id = jh.employee_id;
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.employee_id);
10         DBMS_OUTPUT.PUT_LINE('Employee Name: ' || employee_rec.full_name);
11
12         -- Check if the end date is NULL (meaning the employee is currently in the job)
13         IF employee_rec.end_date IS NULL THEN
14             DBMS_OUTPUT.PUT_LINE('Job History End Date: (Still Employed)');
15         ELSE
Employee ID: 3
Employee Name: Ajith Zlotkey
Job History End Date: 23-FEB-2014
-----
Statement processed.

0.01 seconds
```

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

**QUERY:**

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
    tot_emp NUMBER;
```

```
    get_dep_id NUMBER;
```

```
BEGIN
```

```
    get_dep_id := 80;
```

```
    SELECT Count(*)
```

```
    INTO tot_emp
```

```
    FROM employees e
```

```
        join departments d
```

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

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

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

```
        ||To_char(tot_emp));
```

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

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

```
    ELSE
```

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

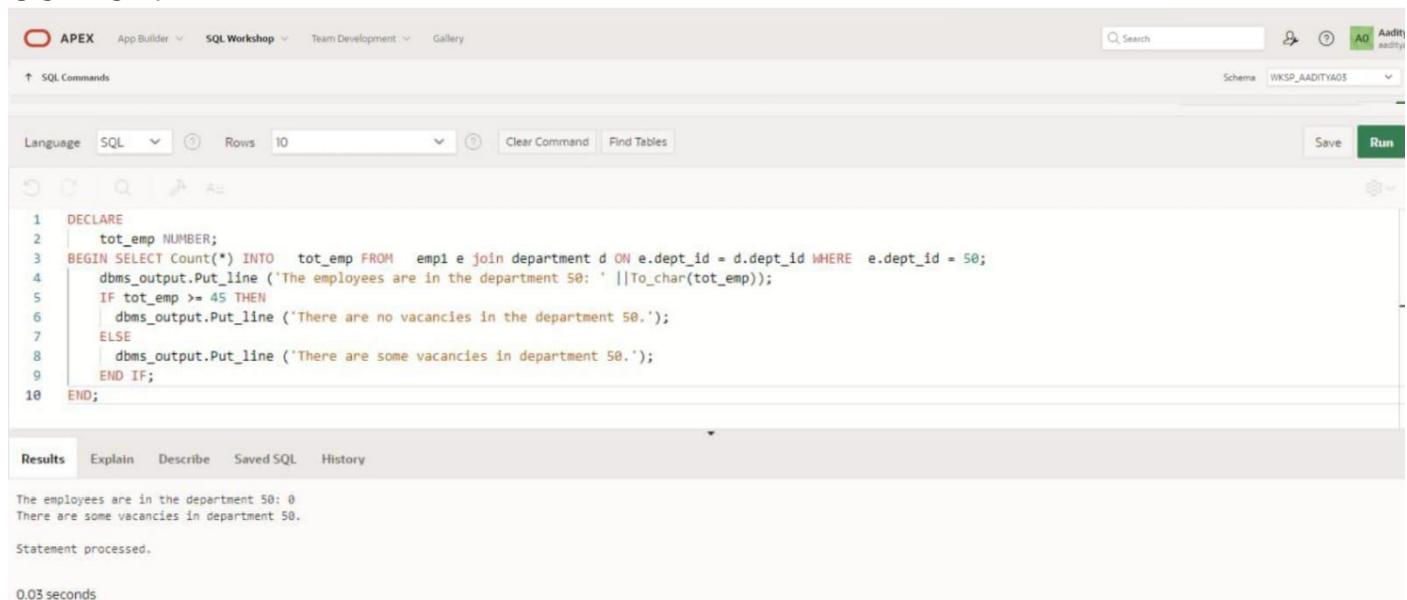
```
);
```

```
    END IF;
```

```
END;
```

```
/
```

**OUTPUT:**



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

The employees are in the department 50: 0  
There are some vacancies in department 50.  
Statement processed.  
0.03 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:**

```
11  dbms_output.Put_line ('The employees are in the department'||get_dep_id||' is: '  
12  ||To_char(tot_emp));  
13  IF tot_emp >= 45 THEN  
14  dbms_output.Put_line ('There are no vacancies in the department'||get_dep_id);  
15  ELSE  
16  dbms_output.Put_line ('There are'||to_char(45-tot_emp)||' vacancies in department'||  
17  get_dep_id );  
18  END IF;  
19  END;  
20 /
```

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

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

**QUERY:**

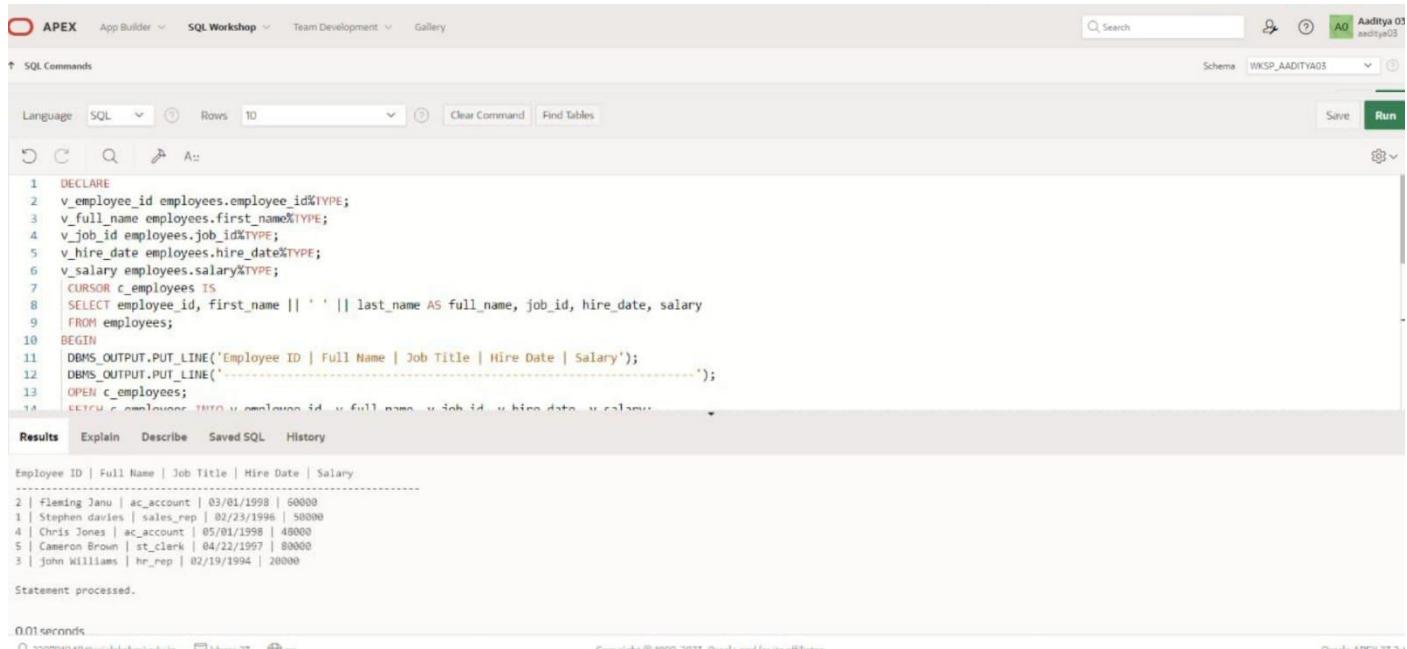
DECLARE

```
v_employee_id employees.employee_id%TYPE;
v_full_name employees.first_name%TYPE;
v_job_id employees.job_id%TYPE;
v_hire_date employees.hire_date%TYPE;
v_salary employees.salary%TYPE;

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

BEGIN
  DBMS_OUTPUT.PUT_LINE('Employee ID | Full Name | Job Title | Hire Date | Salary');
  DBMS_OUTPUT.PUT_LINE('-----');
  OPEN c_employees;
  FETCH c_employees INTO v_employee_id, v_full_name, v_job_id, v_hire_date, v_salary;
  WHILE c_employees%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(v_employee_id || ' ' || v_full_name || ' ' || v_job_id || ' ' ||
v_hire_date || ' ' || v_salary);
    FETCH c_employees INTO v_employee_id, v_full_name, v_job_id, v_hire_date, v_salary;
  END LOOP;
  CLOSE c_employees;
END;
/
```

**OUTPUT:**



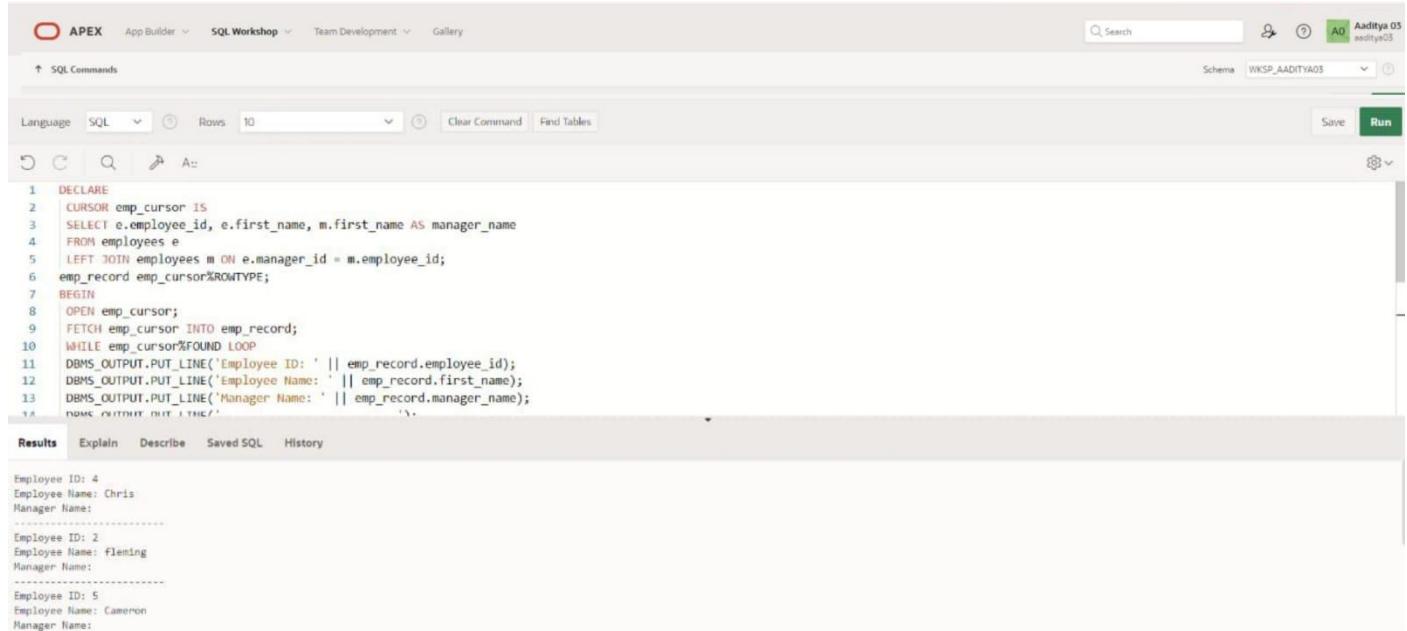
```
Employee ID | Full Name | Job Title | Hire Date | Salary
-----+
2 | Fleming Janu | ac_account | 03/01/1998 | 60000
1 | Stephen davies | sales_rep | 02/23/1998 | 50000
4 | Chris Jones | ac_account | 05/01/1998 | 40000
5 | Cameron Brown | st_clerk | 04/22/1997 | 80000
3 | John Williams | hr_rep | 02/19/1994 | 20000
```

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, Team Development, and Gallery. The SQL Commands tab is selected. The schema is set to WKSP\_AADITYA03. The code area contains the PL/SQL block from the previous step. The results pane shows the output of the DBMS\_OUTPUT.PUT\_LINE statements for three employees: Chris, Fleming, and Cameron, each with their respective Employee ID, Employee Name, and Manager Name.

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

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

**QUERY:**

DECLARE

  CURSOR job\_cursor IS

    SELECT e.job\_id, j.lowest\_sal

    FROM job\_grade j,employees e;

job\_record job\_cursor%ROWTYPE;

BEGIN

  OPEN job\_cursor;

  FETCH job\_cursor INTO job\_record;

  WHILE job\_cursor%FOUND LOOP

    DBMS\_OUTPUT.PUT\_LINE('Job ID: ' || job\_record.job\_id);

    DBMS\_OUTPUT.PUT\_LINE('Minimum Salary: ' || job\_record.lowest\_sal);

    DBMS\_OUTPUT.PUT\_LINE('-----');

    FETCH job\_cursor INTO job\_record;

  END LOOP;

  CLOSE job\_cursor;

END;

/

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. The code entered is a PL/SQL block that retrieves job IDs and their minimum salaries from the 'job\_grade' and 'employees' tables, using a cursor and DBMS\_OUTPUT.PUT\_LINE to display the results. The output pane shows the results of the execution.

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

Results

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

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

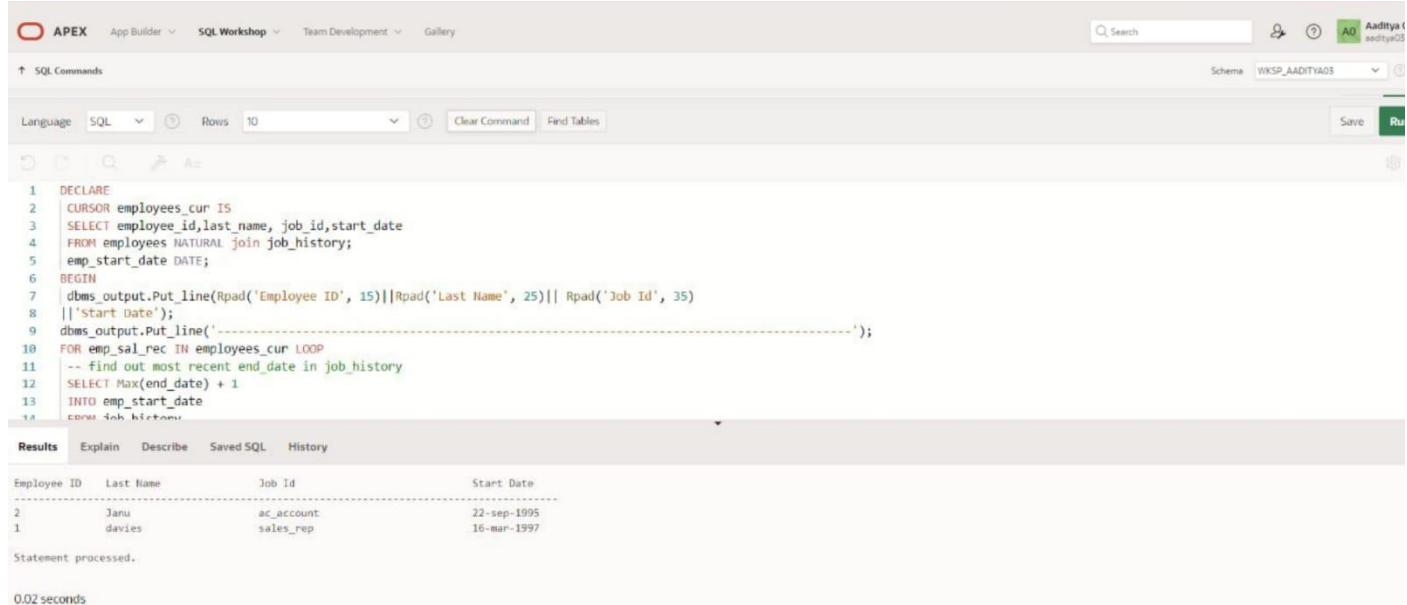
**QUERY:**

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:**



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

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

Statement processed.  
0.02 seconds

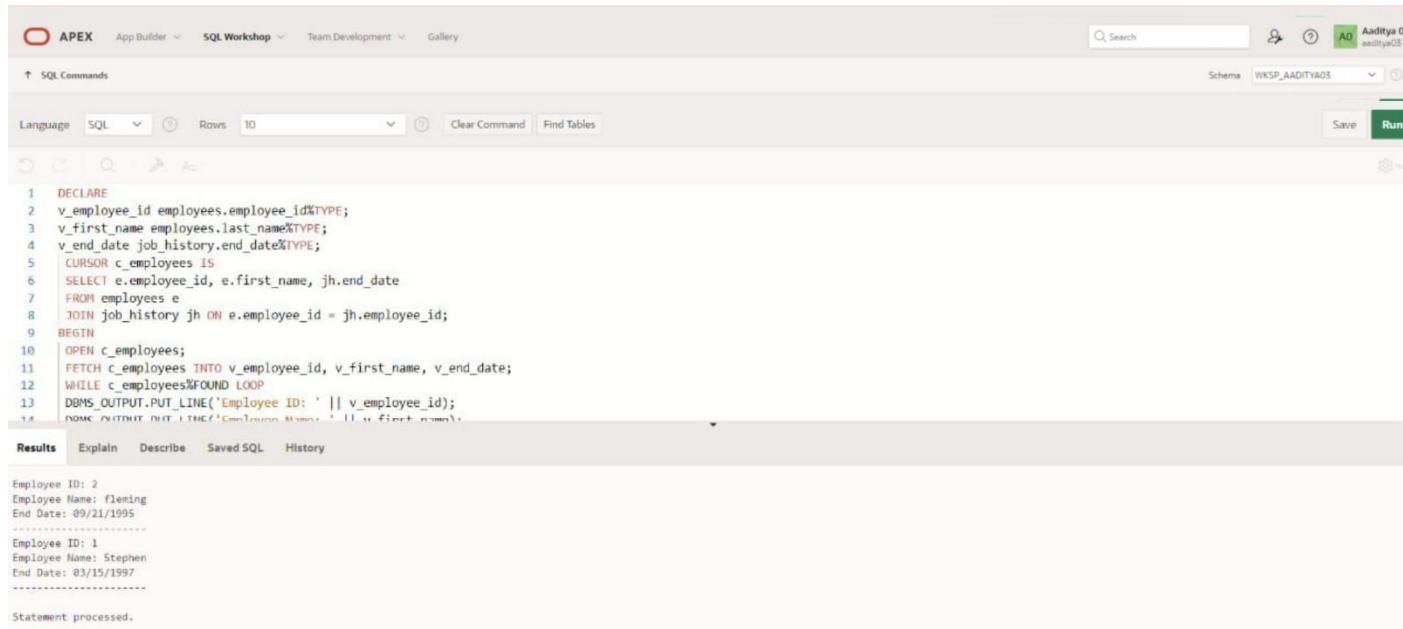
**15.)** Write a PL/SQL program to display the employee IDs, names, and job history end dates of all employees.

**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, Team Development, and Gallery. The right side shows a user profile for Aaditya Oo (aadityao3). The main area has tabs for SQL Commands, Language, SQL, Rows, and Run. The SQL tab is active, displaying the PL/SQL code. The code is executed, and the results are shown in the Results tab, which displays the output of the DBMS\_OUTPUT.PUT\_LINE statements.

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

**Results**

```
Employee ID: 2
Employee Name: Fleming
End Date: 09/21/1995
-----
Employee ID: 1
Employee Name: Stephen
End Date: 03/15/1997
-----
Statement processed.
```

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

**RESULT:**

# PROCEDURES AND FUNCTIONS

EX\_NO: 17

DATE:

## 1.) Factorial of a number using function.

QUERY:

DECLARE

    fac NUMBER := 1;

    n NUMBER := :1;

BEGIN

    WHILE n > 0 LOOP

        fac := n \* fac;

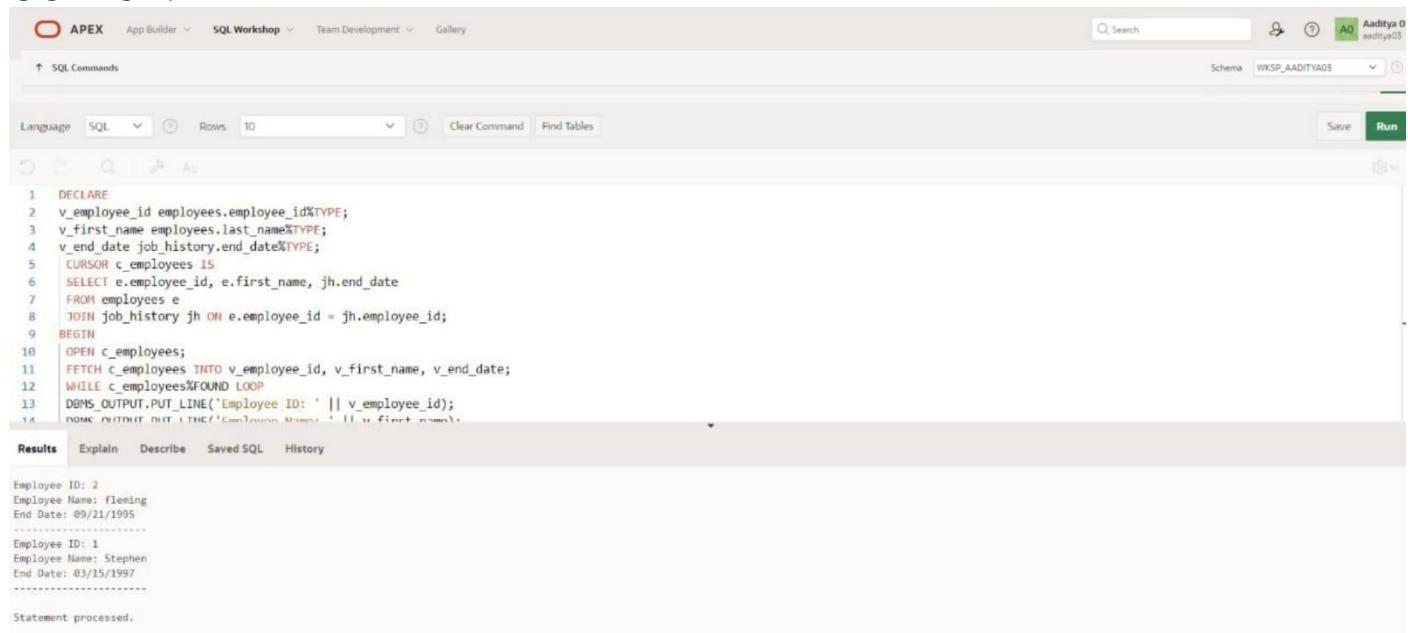
        n := n - 1;

    END LOOP;

    DBMS\_OUTPUT.PUT\_LINE(fac);

END;

## OUTPUT:



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

Employee ID: 2  
Employee Name: Fleming  
End Date: 09/21/1995  
-----  
Employee ID: 1  
Employee Name: Stephen  
End Date: 03/15/1997

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**

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

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

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected, followed by 'App Builder' and 'SQL Workshop'. The main area is titled 'SQL Commands'. The language is set to 'SQL' and the number of rows is set to 10. The code editor contains the following PL/SQL code:

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

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

Procedure created.

0.04 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;
```

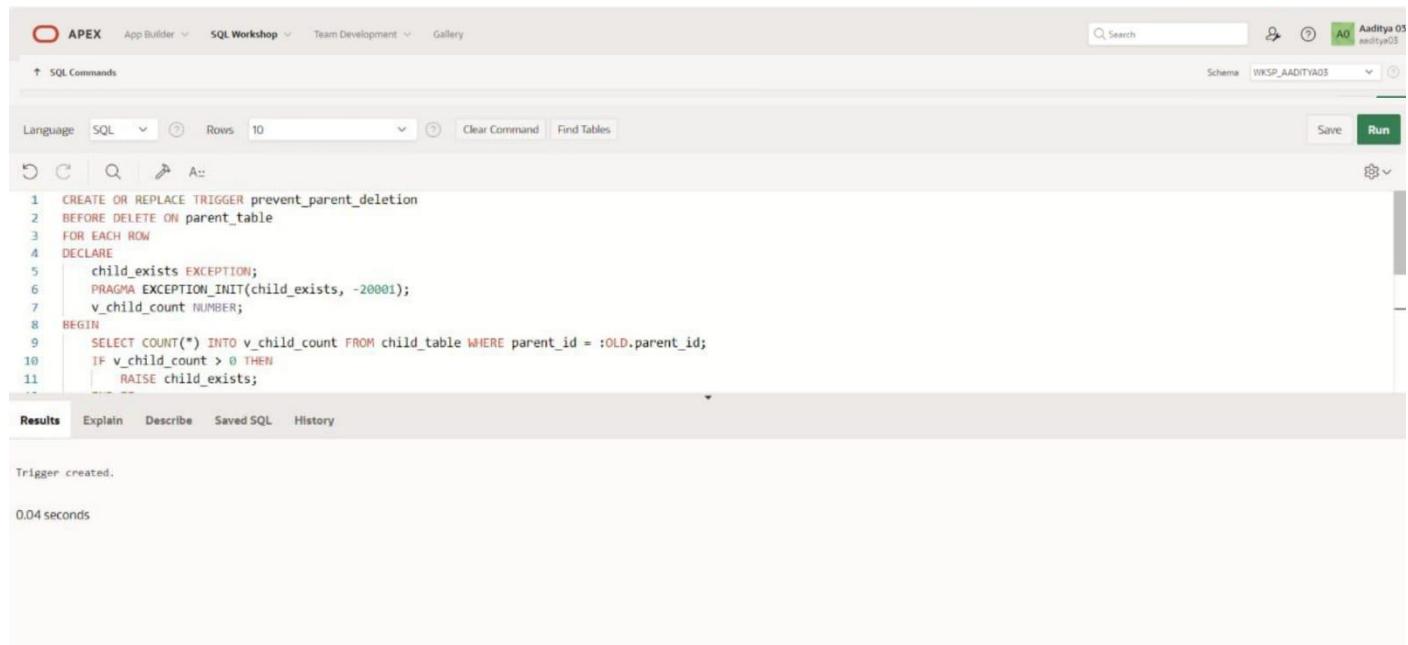
```
EXCEPTION
```

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

```
1 CREATE OR REPLACE TRIGGER prevent_parent_deletion
2 BEFORE DELETE ON parent_table
3 FOR EACH ROW
4 DECLARE
5     child_exists EXCEPTION;
6     PRAGMA EXCEPTION_INIT(child_exists, -20001);
7     v_child_count NUMBER;
8 BEGIN
9     SELECT COUNT(*) INTO v_child_count FROM child_table WHERE parent_id = :OLD.parent_id;
10    IF v_child_count > 0 THEN
11        RAISE child_exists;
```

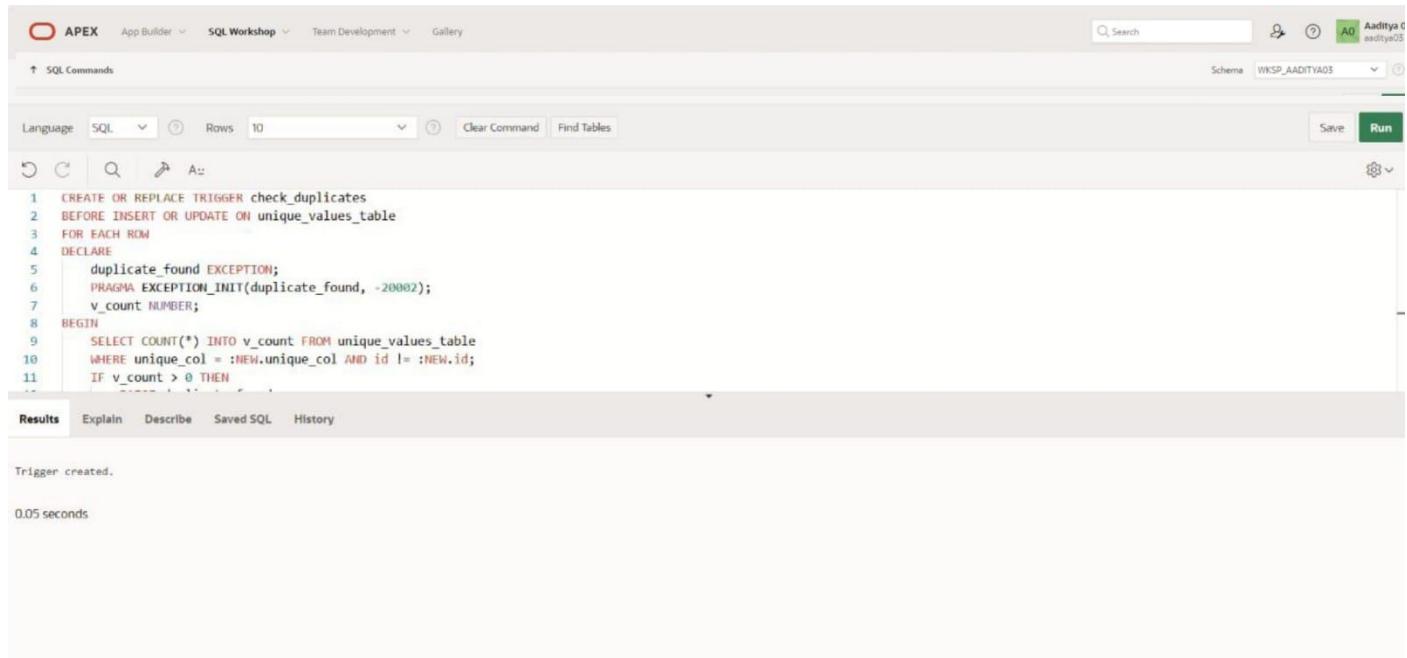
Below the SQL editor, the 'Results' tab is active, showing the output: "Trigger created." and "0.04 seconds".

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user information (Aaditya03), and a schema dropdown set to WKSP\_AADITYA03. The main area is a SQL editor with the following content:

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

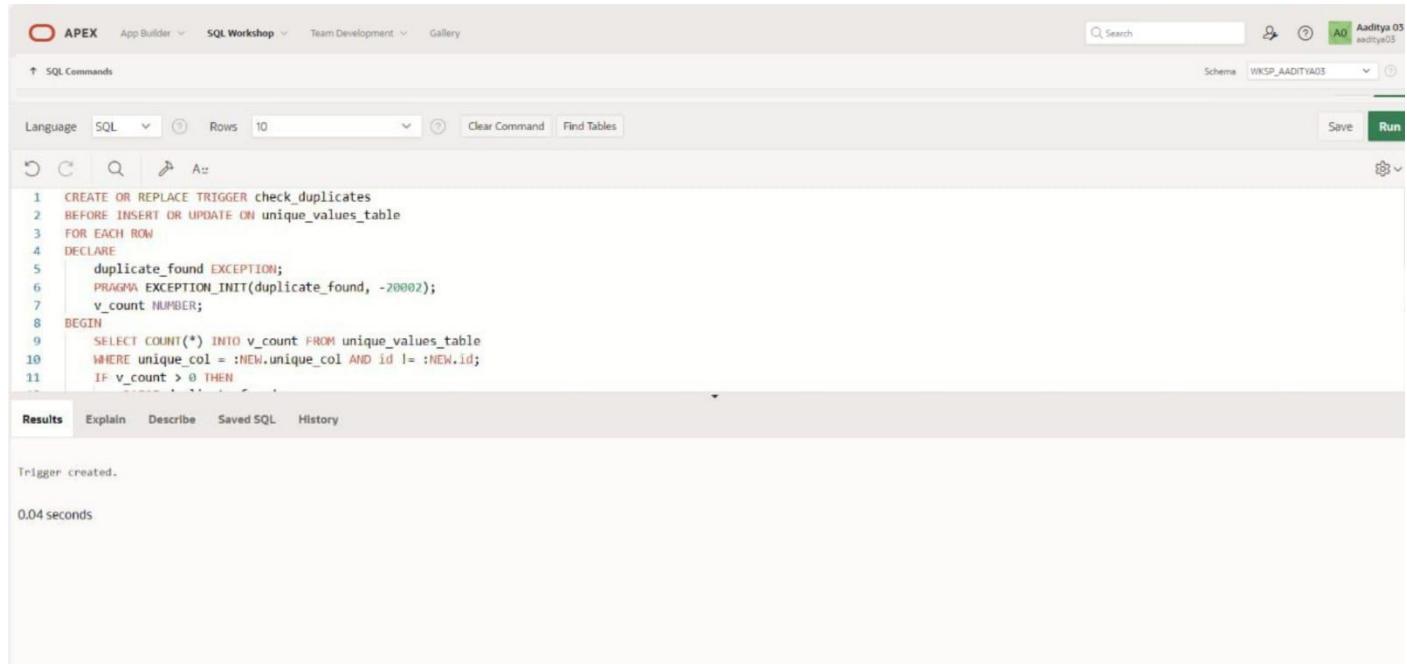
The 'Results' tab is selected at the bottom, showing the message "Trigger created." and a execution time of "0.05 seconds".

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

**QUERY:**

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

**OUTPUT:**



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 a trigger:

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

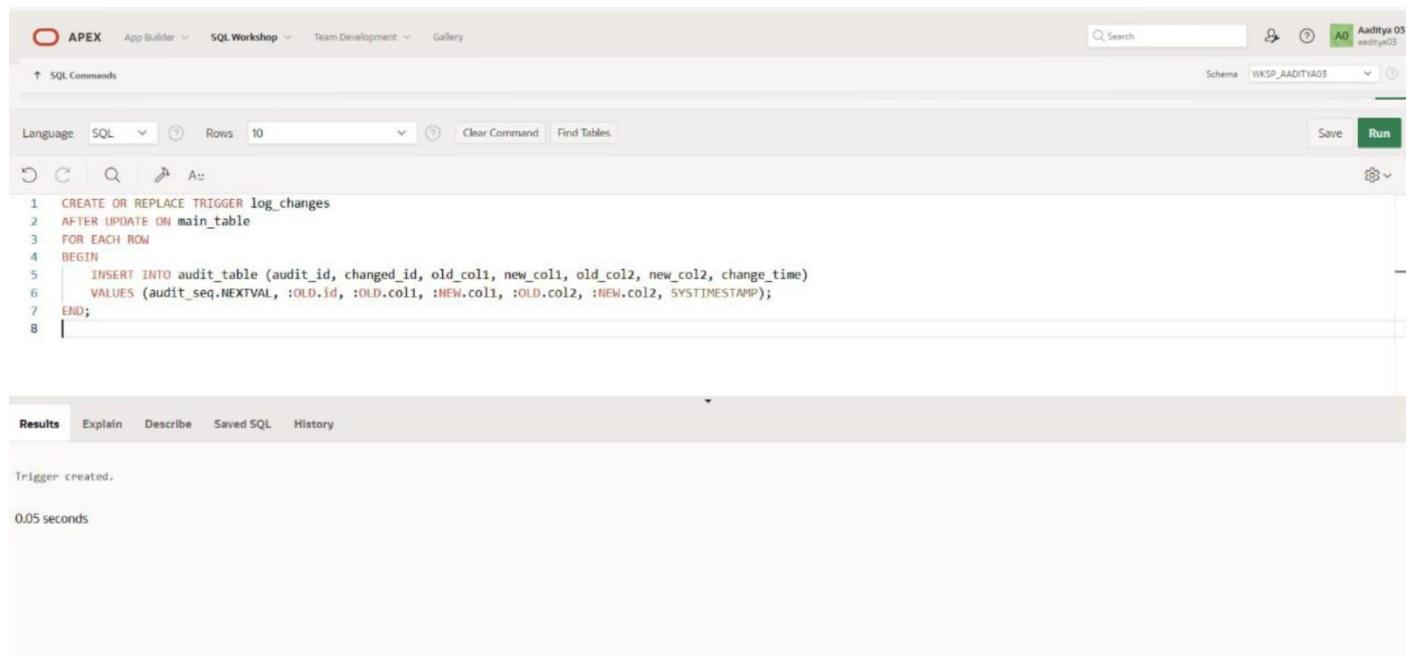
The 'Results' tab is active at the bottom, showing the output: "Trigger created." and "0.04 seconds".

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

**QUERY:**

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

**OUTPUT:**



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 'log\_changes' trigger. The command is as follows:

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

Below the SQL editor, the 'Results' tab is active. The output shows:

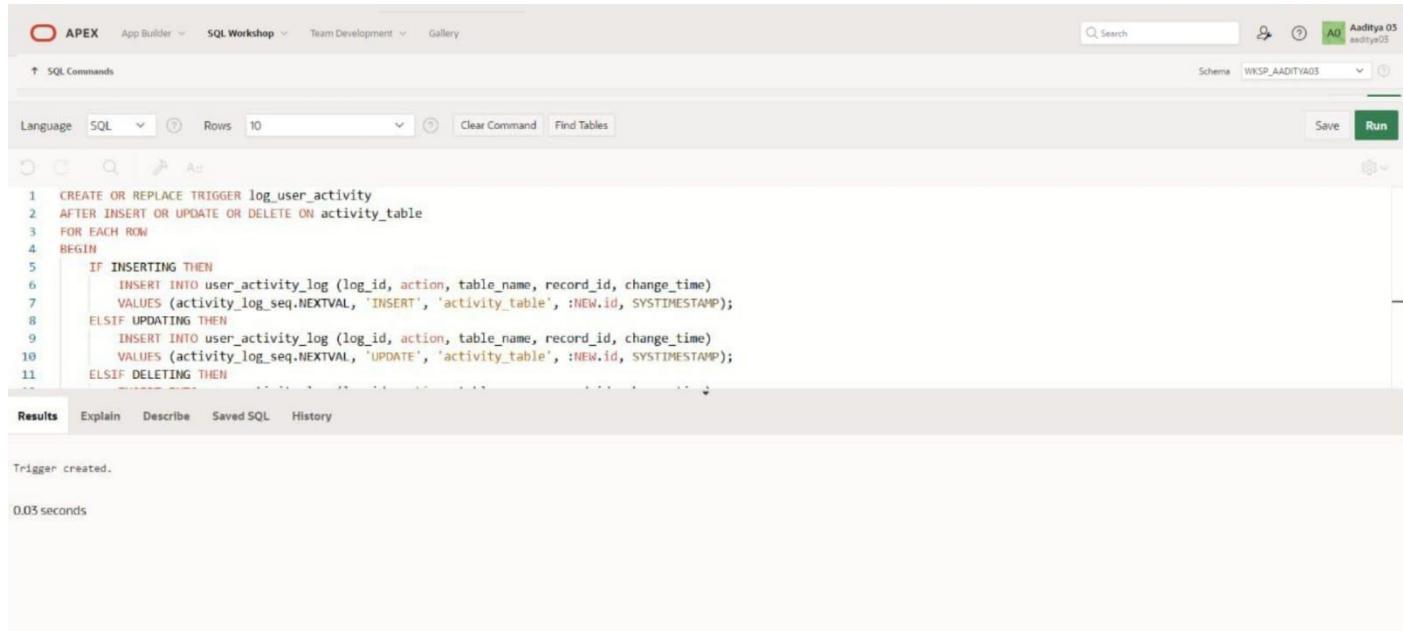
```
Trigger created.
0.05 seconds
```

**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:

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

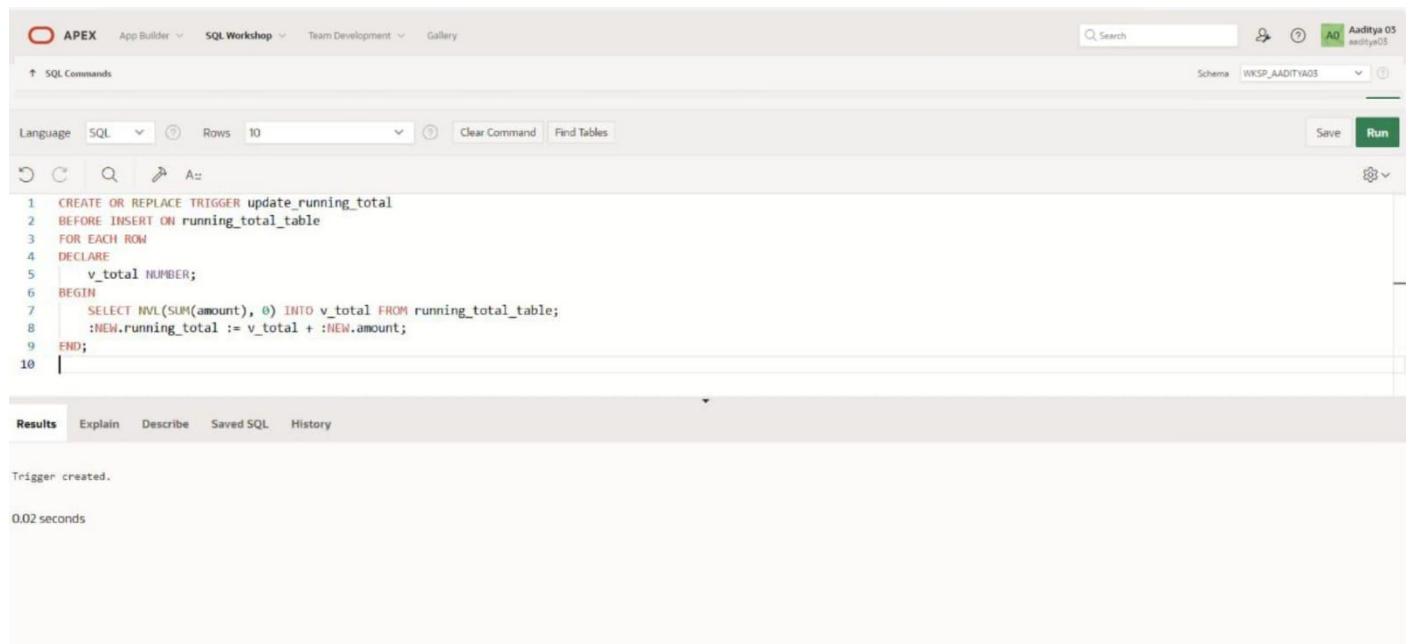
The 'Results' tab is active, showing the output: "Trigger created." Below it, the execution time is listed as "0.05 seconds".

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

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

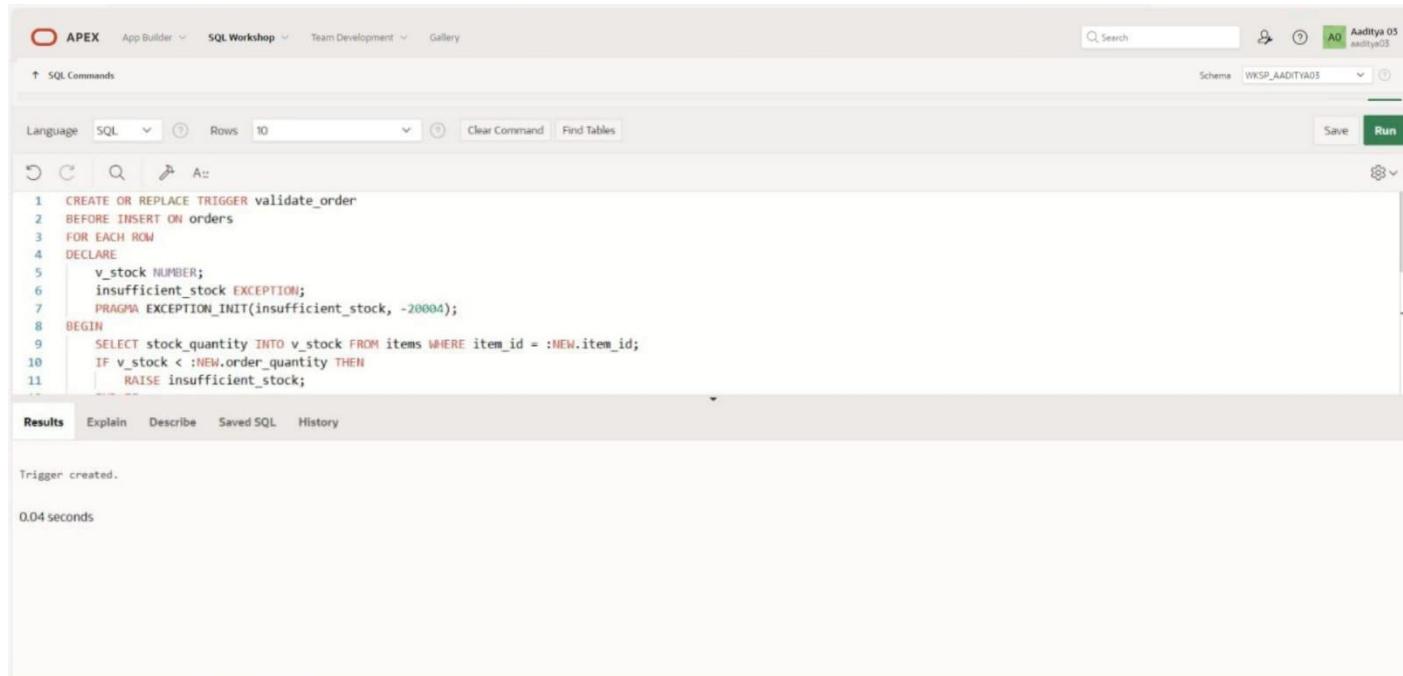
Below the command, the 'Results' tab is active, showing the output: "Trigger created." and "0.02 seconds".

**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. The main area displays the SQL command for creating the trigger. The command is as follows:

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

```

Below the command, the 'Results' tab is active, showing the message "Trigger created." and a execution time of "0.04 seconds".

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

**RESULT:**

# MONGO DB

**EX\_NO: 19**

**DATE:**

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

**QUERY:**

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

**OUTPUT:**

```
Bhanupriya_40> db.restaurants.find({$and:[{ $or: [{cuisine: { $nin: ["American", "Chinese"] }}, {name: /"Wil/}]}], {restaurant_id: 1, name: 1, borough: 1, cuisine: 1})
[
  {
    _id: ObjectId('6650a544c5dcab95e3cdcdf6'),
    borough: 'Bronx',
    cuisine: 'Bakery',
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
Bhanupriya_40> |
```

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:**

```
Bhanupriya_40> db.restaurants.find({ "grades": { $elemMatch: { "grade": "A", "score": 11, "date": ISODate("2014-08-11T00:00:00Z") } } }, { restaurant_id: 1, name: 1, grades: 1 })
Bhanupriya_40>
```

**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:**

```
Bhanupriya_40> 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 } )
```

```
Bhanupriya_40>
```

**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:**

```
Bhanupriya_40> db.restaurants.find({$and : [{"address.coord.1": {$gt : 42}}, {"address.coord.1": {$lte : 52}}]}, {_id:0, restaurant_id:1, name:1, address:1})
```

```
Bhanupriya_40>
```

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:**

```
Bhanupriya_40> 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'
}
]
Bhanupriya_40>
```

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:**

```
Bhanupriya_40> 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'
}
]
Bhanupriya_40>
```

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:

```
Bhanupriya_40> db.restaurants.find({ "address.street": { $exists: true, $ne: "" } })
[
  {
    _id: ObjectId('6650a544c5dcab95e3cdcdf6'),
    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'
  }
]
Bhanupriya_40>
```

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:

```
Bhanupriya_40> db.restaurants.find({ "address.street": { $exists: true, $ne: "" } })
[
  {
    _id: ObjectId('6650a544c5dcab95e3cdcdf6'),
    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'
  }
]
Bhanupriya_40>
```

**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:**

```
Bhanupriya_40> db.restaurants.find({ "address.coord": { $elemMatch: { $type: "double" } } })
[ {
  _id: ObjectId('6650a544c5dcab95e3cdcdf6'),
  address: {
    building: '1007',
    coord: [ -73.856077, 40.848447 ],
    street: 'Morris Park Ave',
    zipcode: '18462'
  },
  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'
}
]
Bhanupriya_40>
```

**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:**

```
Bhanupriya_40> db.restaurants.find({ "grades.score": { $mod: [7, 0] } }, { restaurant_id: 1, name: 1, grades: 1 })
[ {
  _id: ObjectId('6650a544c5dcab95e3cdcdf6'),
  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'
}
]
Bhanupriya_40>
```

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:**

```
Bhanupriya_40> db.restaurants.find({ name: /mon/i }, { name: 1, borough: 1, "address.coord": 1, cuisine: 1 })  
Bhanupriya_40>
```

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:**

```
Bhanupriya_40> db.restaurants.find({ name: /mon/i }, { name: 1, borough: 1, "address.coord": 1, cuisine: 1 })  
Bhanupriya_40>
```

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:**

```
Bhanupriya_40> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } } })
[ {
    _id: ObjectId('6650a544c5dcab95e3cdcdf6'),
    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'
}
]
Bhanupriya_40>
```

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:**

```
Bhanupriya_40> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, "borough": "Manhattan" })
Bhanupriya_40>
```

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:**

```
Bhanupriya_40> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }] })
```

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:**

```
Bhanupriya_40> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $ne: "American" } })
```

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:**

```
Bhanupriya_40> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $nin: ["American", "Chinese"] } })  
Bhanupriya_40>
```

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:**

```
Bhanupriya_40> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }] })  
[  
  {  
    _id: ObjectId('6650a544c5dcab95e3cdcdf6'),  
    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'  
  }  
]  
Bhanupriya_40>
```

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:**

```
Bhanupriya_40> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }] })
```

```
Bhanupriya_40>
```

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:**

```
Bhanupriya_40> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }] })
```

```
Bhanupriya_40>
```

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:**

```
Bhanupriya_40> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $ne: "American" } })
```

```
Bhanupriya_40>
```

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:**

```
Bhanupriya_40> 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"] } })
```

```
Bhanupriya_40>
```

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:**

```
Bhanupriya_40> db.restaurants.find({ $or: [{ "grades.score": 2 }, { "grades.score": 6 }] })
[ {
  _id: ObjectId('6650a544c5dcab95e3cdcdf6'),
  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'
}
]
Bhanupriya_40>
```

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 year.

```
Bhanupriya_40> db.movies.find({ year: 1893 })  
Bhanupriya_40>
```

- 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:**

```
Bhanupriya_40> db.movies.find({ runtime: { $gt: 120 } })  
Bhanupriya_40>
```

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

**QUERY:**

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

**OUTPUT:**

```
Bhanupriya_40> db.movies.find({ genres: 'Short' })
[
  {
    _id: ObjectId('573a1390f29313caabcd42e8'),
    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/MVSBMTU3NjE5NzYtYTYYN500MDVmLWIwYjgtMmYwYWIxZDYyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._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')
    }
  }
]
Bhanupriya_40>
```

### 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:**

```
Bhanupriya_40> db.movies.find({ genres: 'Short' })
[
  {
    _id: ObjectId('573a1390f29313caabcd42e8'),
    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/MVSBMTU3NjE5NzYtYTYYN500MDVmLWIwYjgtMmYwYWIxZDYyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._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')
    }
  }
]
Bhanupriya_40>
```

**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:**

```
Bhanupriya_40> db.movies.find({ directors: 'William K.L. Dickson' })  
Bhanupriya_40>
```

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

**QUERY:**

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

**OUTPUT:**

```
Bhanupriya_40> db.movies.find({ directors: 'William K.L. Dickson' })  
Bhanupriya_40>
```

**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:**

```
Bhanupriya_40> db.movies.find({ directors: 'William K.L. Dickson' })  
Bhanupriya_40>
```

**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**

```
Bhanupriya_40> db.movies.find({ 'imdb.rating': { $gt: 7 } })  
[  
  {  
    _id: ObjectId('573a1390f29313caabcd42e8'),  
    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/MV5BMHU3NjE5NzYtYTYYNS00MDVmlWlTwYjgtMmYwYWIxZDYyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._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('2015-08-08T19:16:18.000Z')  
    }  
  }  
]  
Bhanupriya_40>
```

**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:**

```
Bhanupriya_40> db.movies.find({ 'tomatoes.viewer.rating': { $gt: 4 } })  
Bhanupriya_40>
```

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

**QUERY:**

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

**OUTPUT:**

```
Bhanupriya_40> db.movies.find({ 'awards.wins': { $gt: 0 } })  
[  
  {  
    _id: ObjectId('573a1390f29313caabcd42e8'),  
    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/MVSBMTU3NjE5NzYtYTYYNS00MDVnLWIwYjgtMmYwYWIXZDYyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._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')  
    }  
  }  
]  
Bhanupriya_40> |
```

**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:**

```
Bhanupriya_40> 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 }
... )
```

Bhanupriya\_40>

**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:**

```
Bhanupriya_40> 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 }
... )
```

Bhanupriya\_40>

**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:**

```
Bhanupriya_40> db.movies.find(
...   { released: ISODate("1893-05-09T00:00:00.000Z") },
...   { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 }
... )
```

Bhanupriya\_40>

**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:**

```
Bhanupriya_40> db.movies.find(
...   { title: /scene/i },
...   { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 }
... )
```

Bhanupriya\_40>

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

**RESULT:**