

# Creating and Managing Tables

EX\_NO:1

DATE:

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

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

## QUERY:

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

## OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'Home', 'SQL Workshop', 'SQL Commands', 'Schema ASH32', and 'Help'. Below the toolbar, there are buttons for 'Autocommit' (checked), 'Rows' (set to 10), 'Save', and 'Run'. The main area contains the SQL command to create the 'DEPARTMENT32' table:

```
create table DEPARTMENT32(
ID number(7),
NAME varchar(25)
);
```

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

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

Length	7	25	25	7
--------	---	----	----	---

### QUERY:

Create table emp(id number(7),Last\_Name varchar2(25),First\_Name varchar2(25),Dept\_id number(7));

### OUTPUT:



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

```
create table emp32(
ID number(7),
LAST_NAME varchar(25),
FIRST_NAME varchar(25),
DEPT_ID number(7)
);
```

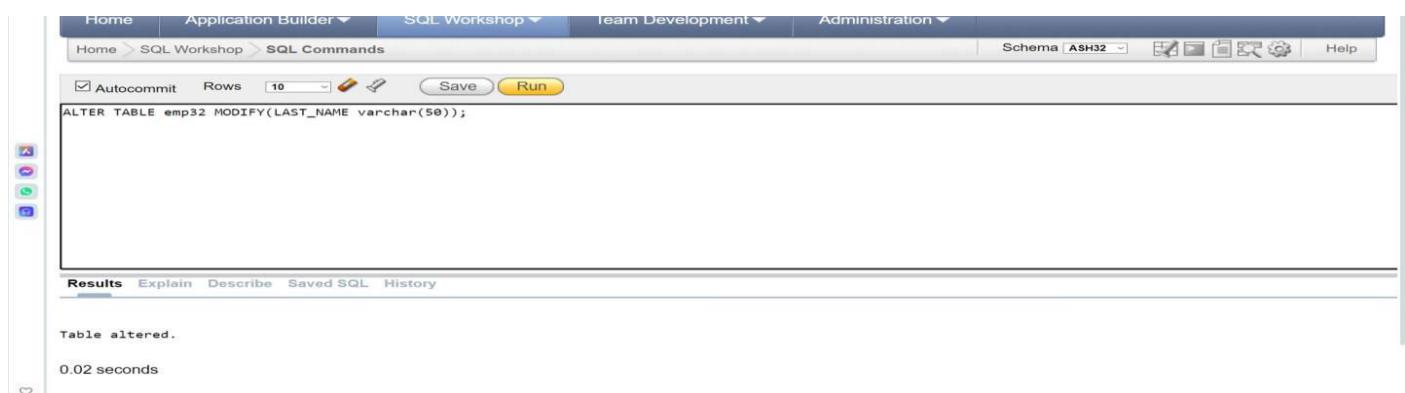
Below the code, there are several small icons representing different database objects like tables, views, and triggers.

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

### QUERY:

Alter table emp modify(Last\_Name varchar2(25));

### OUTPUT:



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

```
ALTER TABLE emp32 MODIFY(LAST_NAME varchar(50));
```

Below the code, the results pane shows the output:

```
Table altered.
0.02 seconds
```

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

### QUERY:

Create table employees2(id number(7),first\_name varchar2(25),Last\_name varchar2(25),Salary int,Dept\_id number(7));

### OUTPUT:

Home Application Builder SQL Workshop Team Development Administration

Home > SQL Workshop > SQL Commands Schema ASH32 Help

Autocommit Rows 10 Save Run

```
create TABLE emp2032(
id number(7),
First_name varchar(25),
Last_name varchar(50),
salary number(10),
Dept_id number(7)
);
```

Results Explain Describe Saved SQL History

Table created.  
0.00 seconds

5.Drop the EMP table.

**QUERY:**

```
Drop table emp;
```

**OUTPUT:**

Home > SQL Workshop > SQL Commands Schema ASH32 Help

Autocommit Rows 10 Save Run

```
drop table emp32;
```

Results Explain Describe Saved SQL History

Table dropped.  
0.04 seconds

6.Rename the EMPLOYEES2 table as EMP.

**QUERY:**

```
Rename employees2 to emp;
```

**OUTPUT:**

The screenshot shows a SQL workshop interface with the following details:

- Toolbar:** Home > SQL Workshop > SQL Commands. Includes Autocommit checked, Rows set to 10, Save, and Run buttons.
- Query Editor:** Contains the command: `rename emp2032 to empnew`.
- Status Bar:** Shows Statement processed. and 0.01 seconds.
- Bottom Navigation:** Results (selected), Explain, Describe, Saved SQL, History.

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

**QUERY:**

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

**OUTPUT:**

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

**QUERY:**

Alter table emp drop column first\_name;

## OUTPUT:

The screenshot shows a SQL Workshop interface. In the top navigation bar, it says "Home > SQL Workshop > SQL Commands". On the right, there are buttons for "Schema ASH32", "Save", "Run", and "Help". Below the toolbar, there are buttons for "Autocommit" (checked), "Rows" (set to 10), and "Save". The main area contains the following SQL code:

```
alter table empnew
drop column First_name;
```

Below the code, there are several small icons. At the bottom of the interface, there is a navigation bar with tabs: "Results" (which is selected), "Explain", "Describe", "Saved SQL", and "History".

The results section displays the output of the executed query:

```
Table altered.  
0.09 seconds
```

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

## RESULT:

---

---

# MANIPULATING DATA

EX\_NO:2

DATE:

1.Create MY\_EMPLOYEE table with the following structure

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

The screenshot shows a SQL command window with the following details:

- Header: SQL Commands, Schema: WKSP\_ASHB2
- Toolbar: Language (SQL), Rows (10), Clear Command, Find Tables, Save, Run.
- Text area:

```
1 create table MY.emp32 (
2     id number(4),
3     Last_name varchar(25),
4     First_name varchar(25),
5     Userid varchar(25),
6     Salary number(9,2)
7 );
8
9 INSERT INTO MY.emp32(HI(LC, 4, 'salih', 'cenkci', 90));
10
```

QUERY:

## OUTPUT:

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:

```
7 );
8
9 INSERT INTO MY_emp32 VALUES( 1,'patel','ralph','rpatel',895);
10 INSERT INTO MY_emp32 VALUES( 2,'danes','betty7','bdancs',860);
11 INSERT INTO MY_emp32 VALUES( 3,'biri','ben','bdiri',1100);
12 INSERT INTO MY_emp32 VALUES( 4,'newman','audrey','cnewman',750);
13 INSERT INTO MY_emp32 VALUES( 5,'ropebur','audrey','arpobeur',1550);
14
```

## OUTPUT :

3.Display the table with values.

**QUERY:**

```
select * from MY_emp32
```

**OUTPUT:**

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:**

**OUTPUT:**

5.Make the data additions permanent.

**QUERY:**

**OUTPUT:**

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

**QUERY:**

```
update MY_emp32 /*6th */
set Last_name='Drexler'
where ID=3;
```

**OUTPUT:**

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

**QUERY:**

```
23
24 Update MY_emp32 /*7th */
25 set salary=1000
26 where salary<900;
27
```

**OUTPUT :**

**8.**Delete Betty dances from MY\_EMPLOYEE table.

**QUERY:**

```
27  
28  delete from MY_emp32 where First_name='betty7';  
29  
30  
31  
**
```

**OUTPUT:**

**9.**Empty the fourth row of the emp table.

**QUERY:**

**OUTPUT:**

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:

```
1 create table EMP1 (
2 |   id number(5) not null primary key, last_name varchar(20), first_name varchar(20),dept_id number(6));
3 
```

OUTPUT:

Results Explain Describe Saved SQL History

Table created.  
0.03 seconds

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

QUERY:

```
1 create table dept (
2 |   id number(6) not null,
3 |   name varchar(20),
4 |   constraint my_dept_id_pk primary key(id)
5 );
```

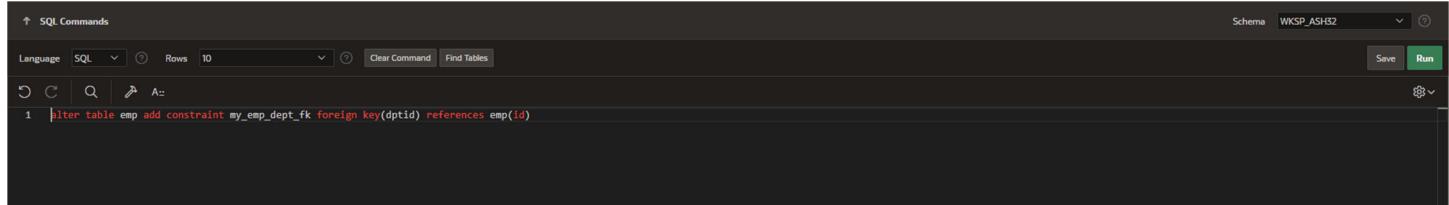
OUTPUT:

Results Explain Describe Saved SQL History

Table created.  
0.03 seconds

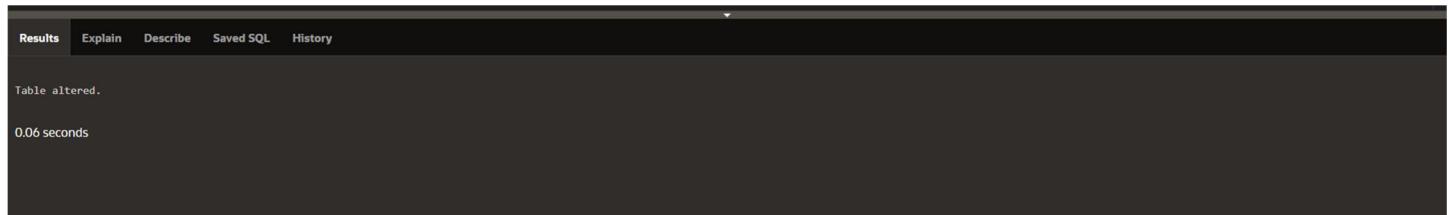
3.Add a column DEPT\_ID to the EMP table. Add a foreign key reference on the EMP table that ensures that the employee is not assigned to nonexistent department. Name the constraint my\_emp\_dept\_id\_fk.

#### QUERY:



A screenshot of a SQL editor interface. The top bar shows "SQL Commands" and "Schema WKSP\_ASH32". The main area has tabs for "Language" (set to "SQL"), "Rows" (set to 10), "Clear Command", and "Find Tables". Below these are icons for Undo, Redo, Cut, Copy, Paste, Find, and Replace. A search bar and a dropdown menu are also present. The code input field contains the command: `alter table emp add constraint my_emp_dept_id_fk foreign key(dptid) references emp(id)`. To the right of the input field are "Save" and "Run" buttons, along with a refresh icon.

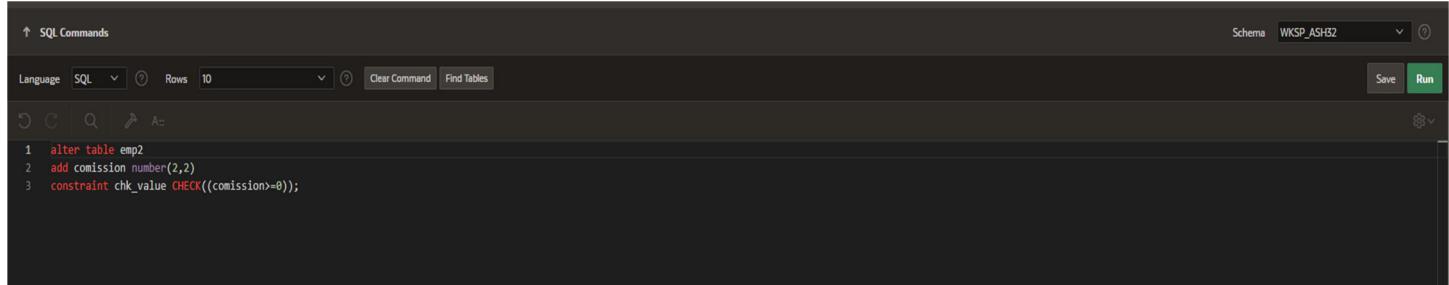
#### OUTPUT:



A screenshot of the SQL editor showing the results of the executed query. The results tab is selected. The output shows the message "Table altered." and a execution time of "0.06 seconds".

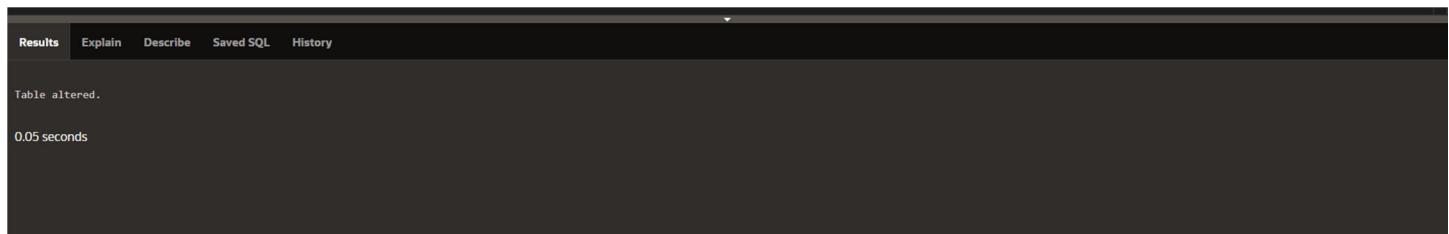
4.Modify the EMP table. Add a COMMISSION column of NUMBER data type, precision 2, scale 2. Add a constraint to the commission column that ensures that a commission value is greater than zero.

#### QUERY:



A screenshot of a SQL editor interface. The top bar shows "SQL Commands" and "Schema WKSP\_ASH32". The main area has tabs for "Language" (set to "SQL"), "Rows" (set to 10), "Clear Command", and "Find Tables". Below these are icons for Undo, Redo, Cut, Copy, Paste, Find, and Replace. A search bar and a dropdown menu are also present. The code input field contains the command: `1 alter table emp2  
2 add comission number(2,2)  
3 constraint chk_value CHECK((comission>=0));`. To the right of the input field are "Save" and "Run" buttons, along with a refresh icon.

#### OUTPUT:



A screenshot of the SQL editor showing the results of the executed query. The results tab is selected. The output shows the message "Table altered." and a execution time of "0.05 seconds".

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

**RESULT:**

# Writing Basic SQL SELECT Statements

EX\_NO:4

DATE:

- 1.The following statement executes successfully.

## Identify the Errors

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

## QUERY:

```
1 select employee_id, last_name, "SALARY"*12"ANNUAL_SALARY" from employees;  
2  
3  
4 select * from employees;  
5
```

## OUTPUT:

EMPLOYEE_ID	LAST_NAME	ANNUAL_SALARY
18	WAN	840000

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

## QUERY:

```
1 select employee_id, last_name, "SALARY"*12"ANNUAL_SALARY" from employees;  
2  
3  
4 select * from employees;  
5  
6  
7 Select employee_id as employee_number, last_name, job_id, hire_date from employees;  
8  
9
```

## OUTPUT:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	MANAGER_ID	DEPARTMENT_ID
18	ASH	WAN	ash@gmail.com	9867539787	01/05/2005	33	70000	9	5

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:**

```
3
4 select * from employees;
5
6
7 Select employee_id as employee_number, last_name, job_id, hire_date from employees;
8
9
10 Select hire_date as startdate from employees;
11
12
13 select distinct job_id from employees;
```

**OUTPUT:**

Results			
Explain    Describe    Saved SQL    History			
EMPLOYEE_NUMBER	LAST_NAME	JOB_ID	HIRE_DATE
18	WAN	53	01/05/2005
1 rows returned in 0.00 seconds    Download			

4.Provide an alias STARTDATE for the hire date.

**QUERY:**

```
6
7 Select employee_id as employee_number, last_name, job_id, hire_date from employees;
8
9
10 Select hire_date as startdate from employees;
11
12
13 select distinct job_id from employees;
14
15
16
```

**OUTPUT:**

Results			
Explain    Describe    Saved SQL    History			
STARTDATE			
01/05/2005			
1 rows returned in 0.01 seconds    Download			

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

**QUERY:**

```
9
10 Select hire_date as startdate from employees;
11
12
13 select distinct job_id from employees;
14
15 select last_name||','|| job_id as "employee_and_title" from employees;
16
```

## OUTPUT:

JOB_ID
33

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

## QUERY:

```
13 select distinct job_id from employees;
14
15 select last_name||','|| job_id as "employee_and_title" from employees;
16
17 select employee_id||','||first_name||','||last_name||','||email||','||phone_number||','||hire_date||','||job_id||',
18 ||salary||','||manager_id||','||department_id as "the_output" from employees;
19
```

## OUTPUT:

employee_and_title
WAN_33

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

## QUERY:

```
11
12
13 select distinct job_id from employees;
14
15 select last_name||','|| job_id as "employee_and_title" from employees;
16
17 select employee_id||','||first_name||','||last_name||','||email||','||phone_number||','||hire_date||','||job_id||',
18 ||salary||','||manager_id||','||department_id as "the_output" from employees;
19
```

## OUTPUT:

```
17 select employee_id||'|'||first_name||'|'||last_name||'|'||email||'|'||phone_number||'|'||hire_date||'|'||job_id||'|'||  
18 salary||'|'||manager_id||'|'||department_id as "the_output" from employees;  
19
```

Results	Explain	Describe	Saved SQL	History
the_output				
18,ASH,WAN,ash@gmail.com,9867539787,01/05/2005,33,70000,9.5				
1 rows returned in 0.01 seconds <a href="#">Download</a>				

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

## RESULT:

# RESTRICTING AND SORTING DATA

EX\_NO:5

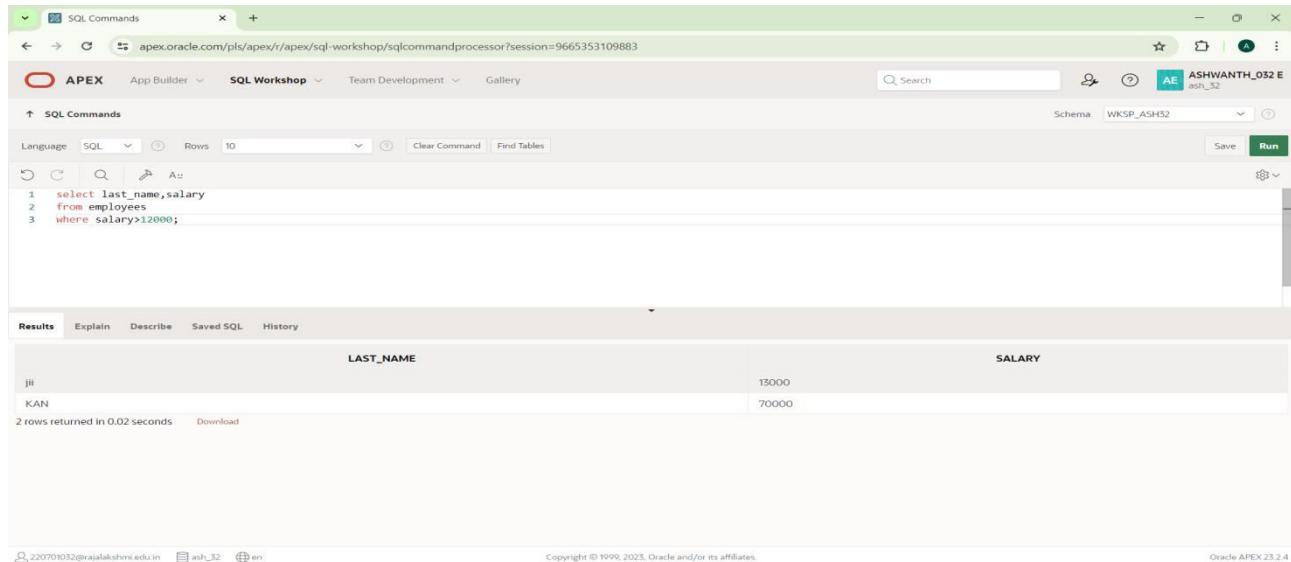
DATE:

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

**QUERY:**

Select last\_name,salary from employees where salary>12000;

**OUTPUT:**



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

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

The Results tab displays the output:

LAST_NAME	SALARY
jii	13000
KAN	70000

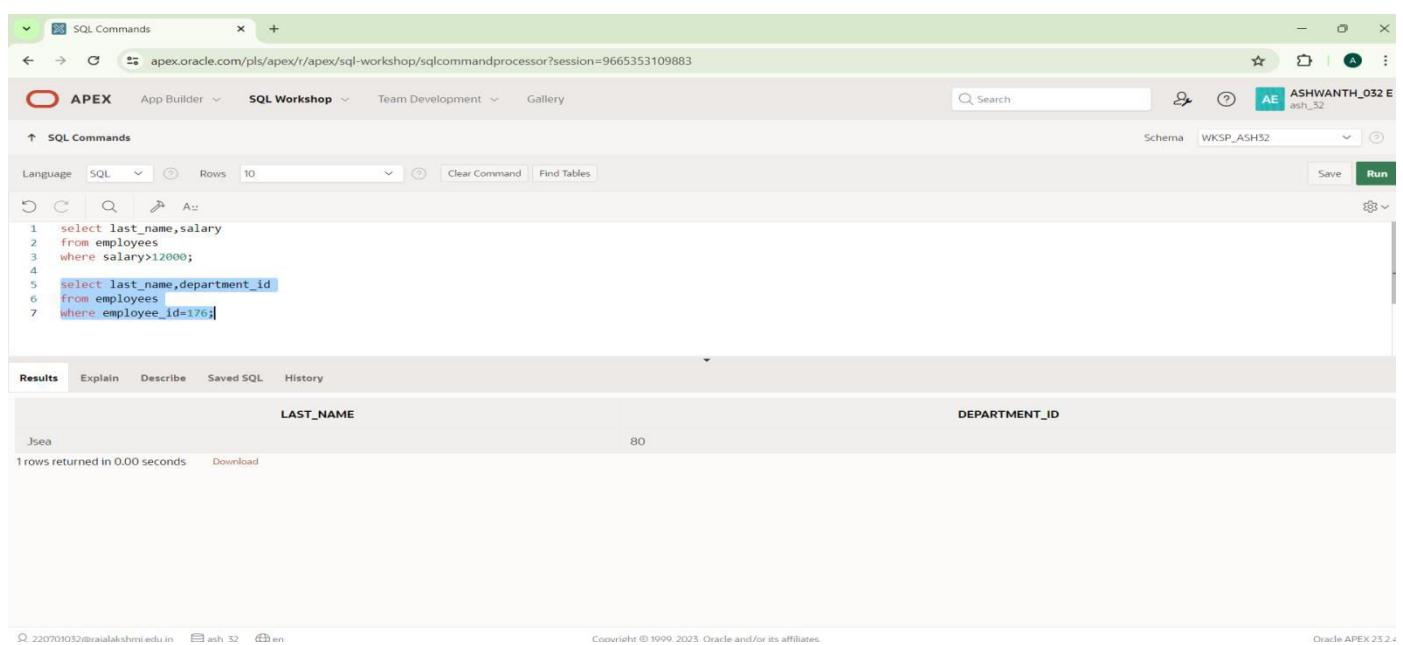
2 rows returned in 0.02 seconds

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

**QUERY:**

Select last\_name,department\_id from employees where employee\_id=176;

**OUTPUT:**



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

```
1 select last_name,salary
2 from employees
3 where salary>12000;
4
5 select last_name,department_id
6 from employees
7 where employee_id=176;
```

The Results tab displays the output:

LAST_NAME	DEPARTMENT_ID
Jsea	80

1 rows returned in 0.00 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 employees where salary not between 5000 and 12000;
```

### 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 main area is titled 'SQL Commands' with a sub-tab 'Schema'. Below the tabs are 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. The code editor contains two queries:

```
1 select last_name,salary
2   from employees
3  where salary>12000;
4
5 select last_name,salary
6   from employees
7  where salary not between 5000 and 12000;
```

The results tab shows the output of the second query:

LAST_NAME	SALARY
jii	13000
KAN	70000

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

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 employees where hire_date between '02/20/1998' and '05/01/1998';
```

### 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 main area is titled 'SQL Commands' with a sub-tab 'Schema'. Below the tabs are 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. The code editor contains three queries:

```
1 select last_name,salary
2   from employees
3  where salary>12000;
4
5 select last_name,job_id,hire_date
6   from employees
7  where hire_date between '02/20/1998' and '05/01/1998';
```

The results tab shows the output of the third query:

LAST_NAME	JOB_ID	HIRE_DATE
namaho	36	02/20/1998
rat	88	05/01/1998

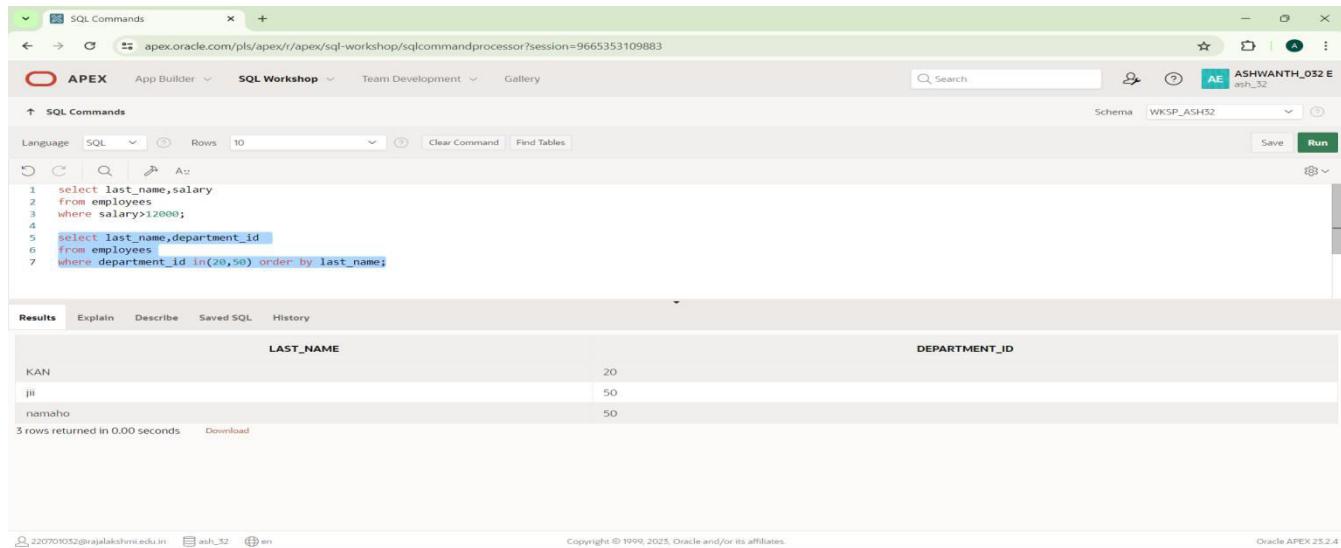
Below the table, it says '2 rows returned in 0.01 seconds' and has 'Download' and 'Run' buttons.

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 employees where department_id in(20,50) order by last_name;
```

#### OUTPUT:



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

```
1 select last_name,salary
2   from employees
3  where salary>12000;
4
5 select last_name,department_id
6   from employees
7  where department_id in(20,50) order by last_name;
```

The Results tab displays the output:

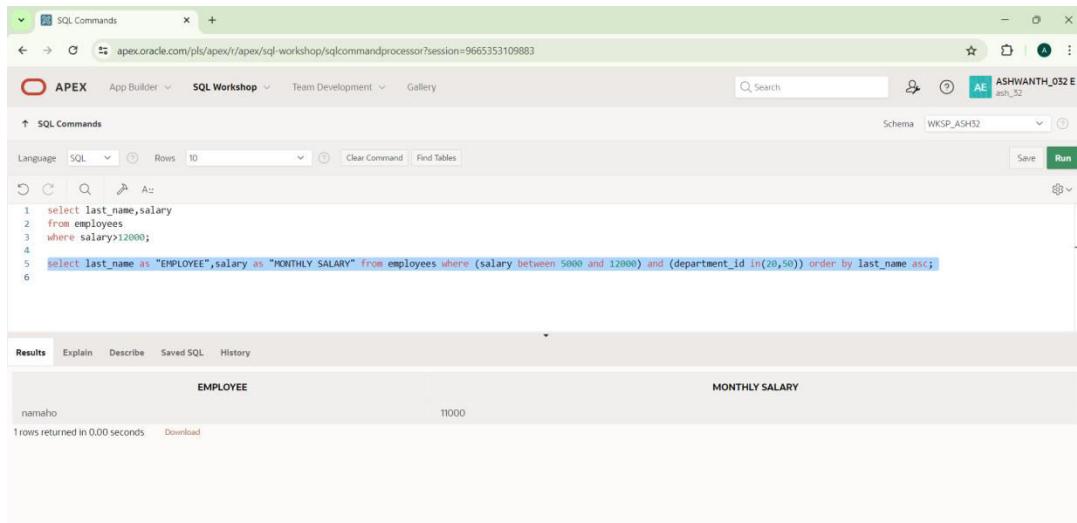
LAST_NAME	DEPARTMENT_ID
KAN	20
jii	50
namaho	50

3 rows returned in 0.00 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 employees where (salary between 5000 and 12000) and (department_id in(20,50)) order by last_name asc;
```



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

```
1 select last_name,salary
2   from employees
3  where salary>12000;
4
5 select last_name as "EMPLOYEE",salary as "MONTHLY SALARY" from employees where (salary between 5000 and 12000) and (department_id in(20,50)) order by last_name asc;
6
```

The Results tab displays the output:

EMPLOYEE	MONTHLY SALARY
namaho	11000

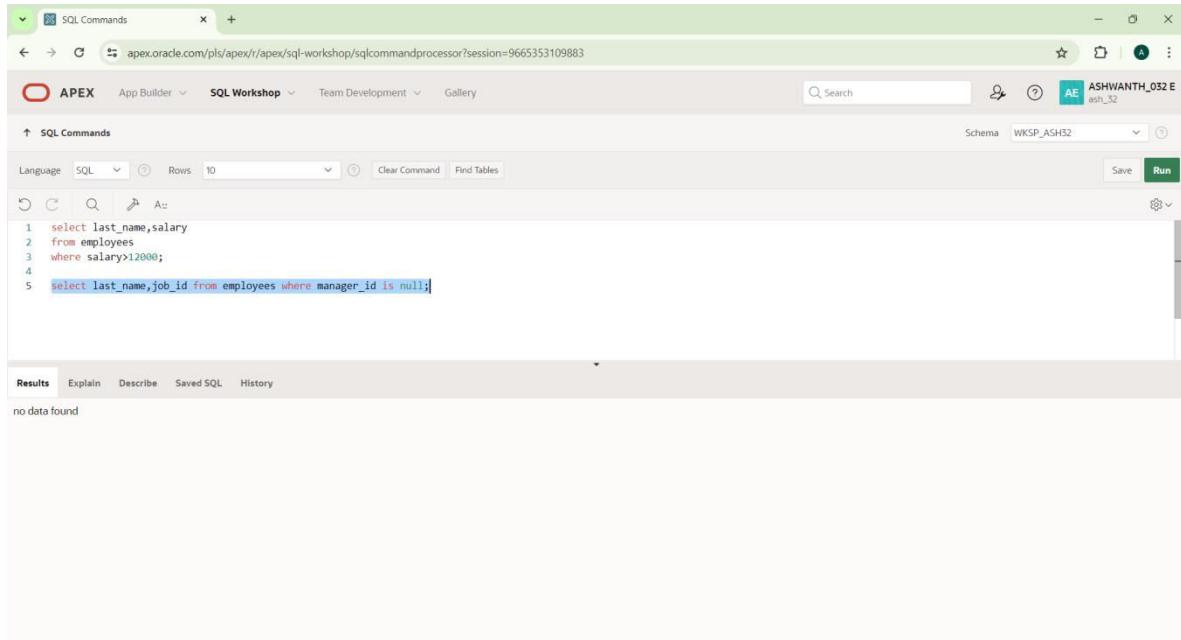
1 rows returned in 0.00 seconds

#### OUTPUT:

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

#### QUERY:

```
select last_name,hire_date from employees where hire_date like '%1994';
```



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 main area displays the following SQL code:

```
1 select last_name,salary
2   from employees
3  where salary>12000;
4
5 select last_name,job_id from employees where manager_id is null;
```

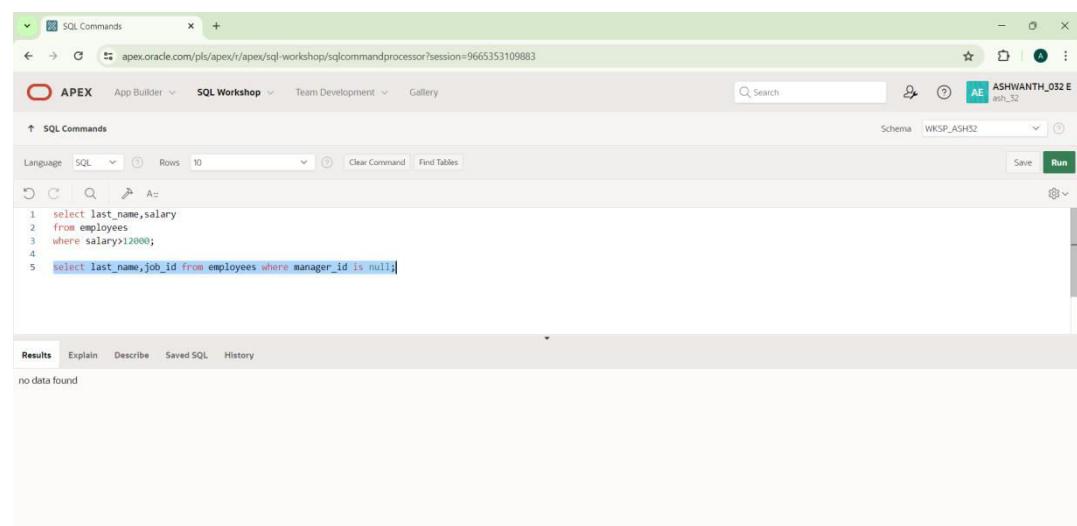
The results tab is active, showing the message "no data found".

#### OUTPUT:

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 employees where manager_id is null;
```



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 main area displays the following SQL code:

```
1 select last_name,salary
2   from employees
3  where salary>12000;
4
5 select last_name,job_id from employees where manager_id is null;
```

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

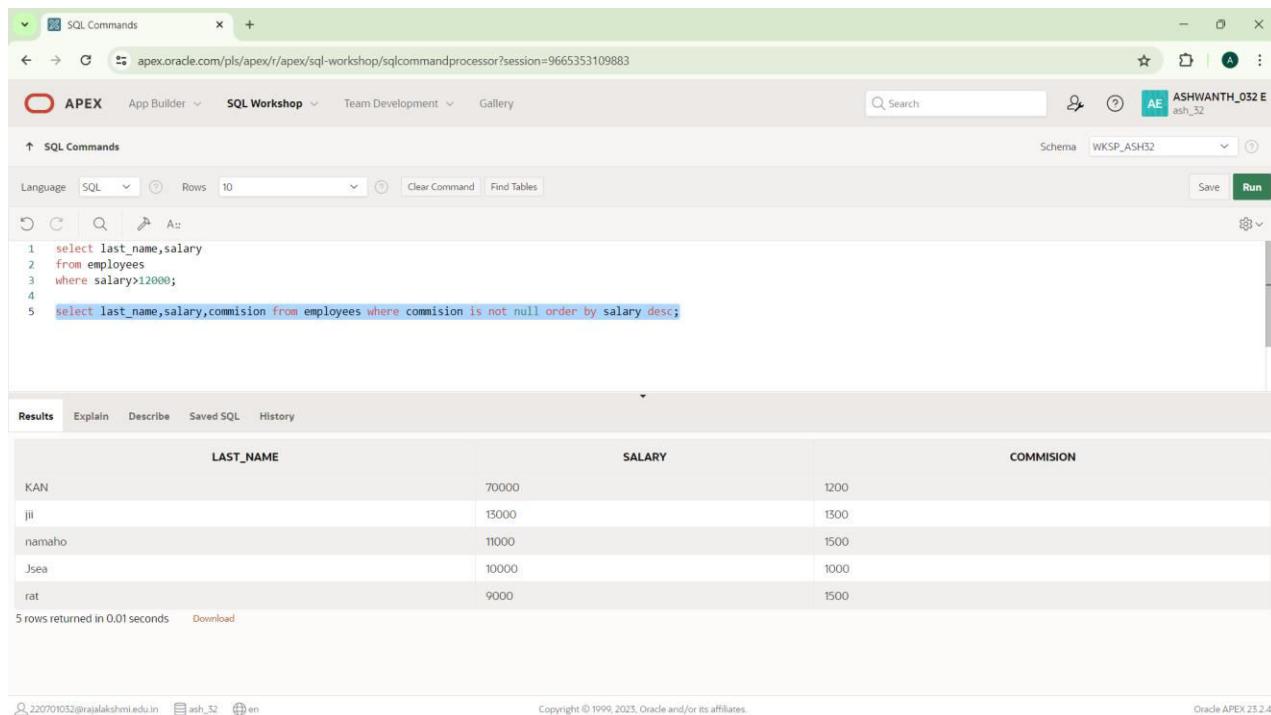
#### OUTPUT:

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

### QUERY:

```
select last_name,salary,commission_no from employees where commission_no is not null order by salary desc;
```

### OUTPUT:



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

```
1 select last_name,salary
2   from employees
3  where salary>12000;
4
5 select last_name,salary,commission from employees where commission is not null order by salary desc;
```

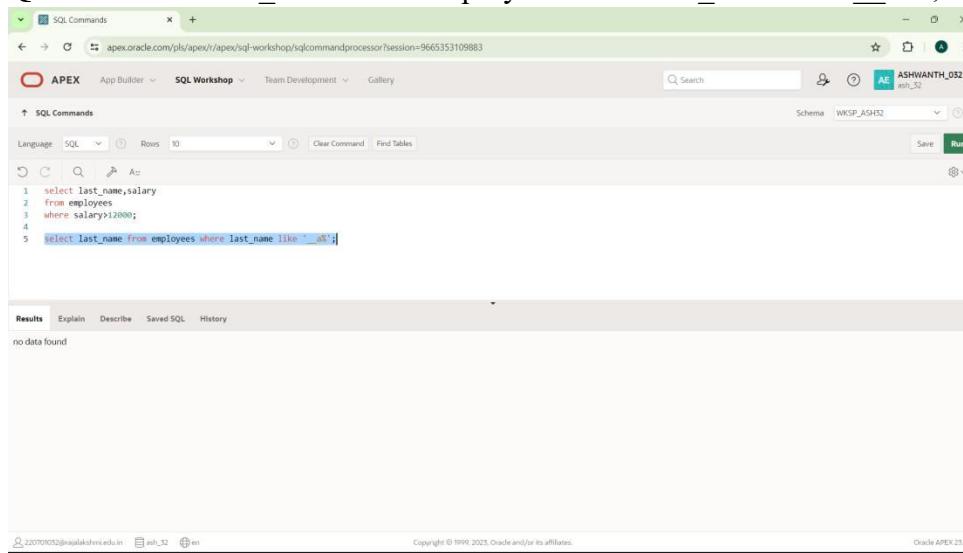
The Results tab shows the output of the query:

LAST_NAME	SALARY	COMMISSION
KAN	70000	1200
jii	13000	1300
namaho	11000	1500
Jsea	10000	1000
rat	9000	1500

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

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

### QUERY:select last\_name from employees where last\_name like '\_\_a%';



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

```
1 select last_name,salary
2   from employees
3  where salary>12000;
4
5 select last_name from employees where last_name like '__a%';
```

The Results tab shows the output of the query:

LAST_NAME
no data found

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

## **OUTPUT:**

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 employees where last_name like '%a%' and last_name like '%e%';
```

## OUTPUT:

SQL Commands

apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=9665353109883

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language SQL Rows 10 Clear Command Find Tables

Save Run

1 select last\_name,salary  
2 from employees  
3 where salary>12000;  
4  
5 select last\_name from employees where last\_name like '%J%' and last\_name like '%S%';

Results Explain Describe Saved SQL History

LAST_NAME
Jsea

1 rows returned in 0.00 seconds Download

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

## **QUERY:**

```
select last_name,job_id,salary from employees where job_id in ('SA_REP','ST_CL') and salary not in(2500,3500,7000);
```

### **OUTPUT:**

```
1 select last_name,salary
2 from employees
3 where salary>12000;
4
5 select last_name,job_id,salary from employees where job_id in ('SA_REP','ST_CL') and salary not in (2500,3500,7000);
```

Results Explain Describe Saved SQL History

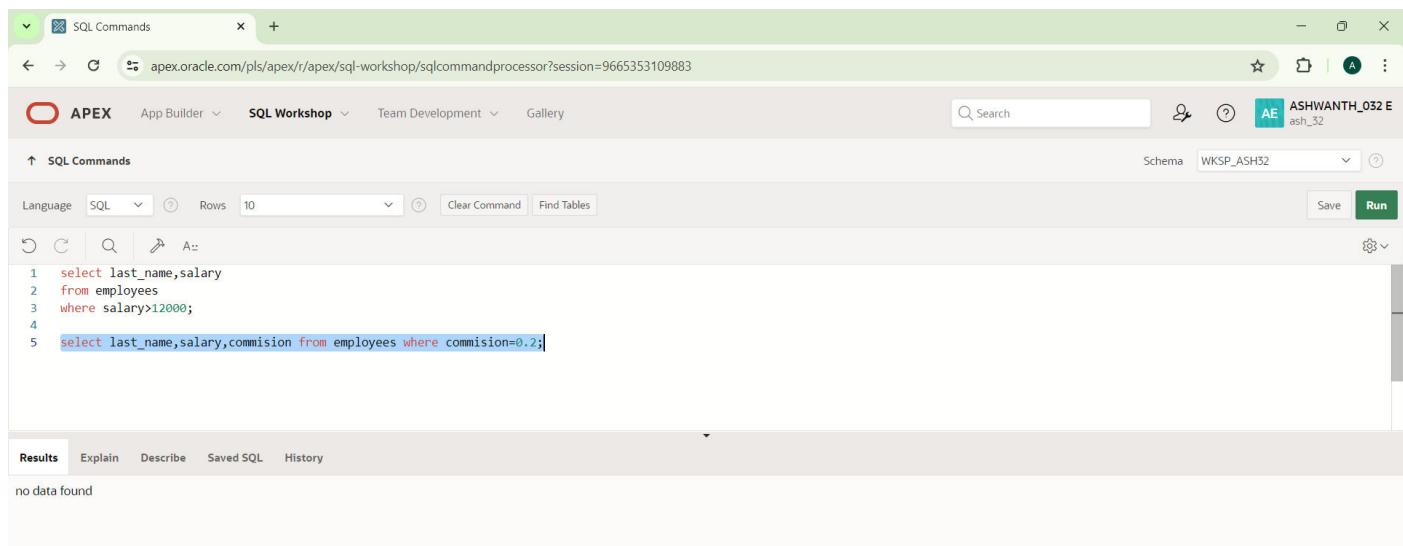
no data found

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

**QUERY:**

```
select last_name,salary,commission_no from employees where commission_no=0.2;
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (which is selected), Team Development, and Gallery. The right side of the header shows the user's name (ASHWANTH\_032 E) and session ID (ash\_32). The main area is titled "SQL Commands". The language is set to SQL, and the results page size is 10 rows. The command input field contains the following SQL code:

```
1 select last_name,salary
2   from employees
3  where salary>12000;
4
5 select last_name,salary,commission from employees where commission=0.2;
```

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

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

**RESULT:**

# DISPLAYING DATA FROM MULTIPLE TABLES

**EX\_NO:7**

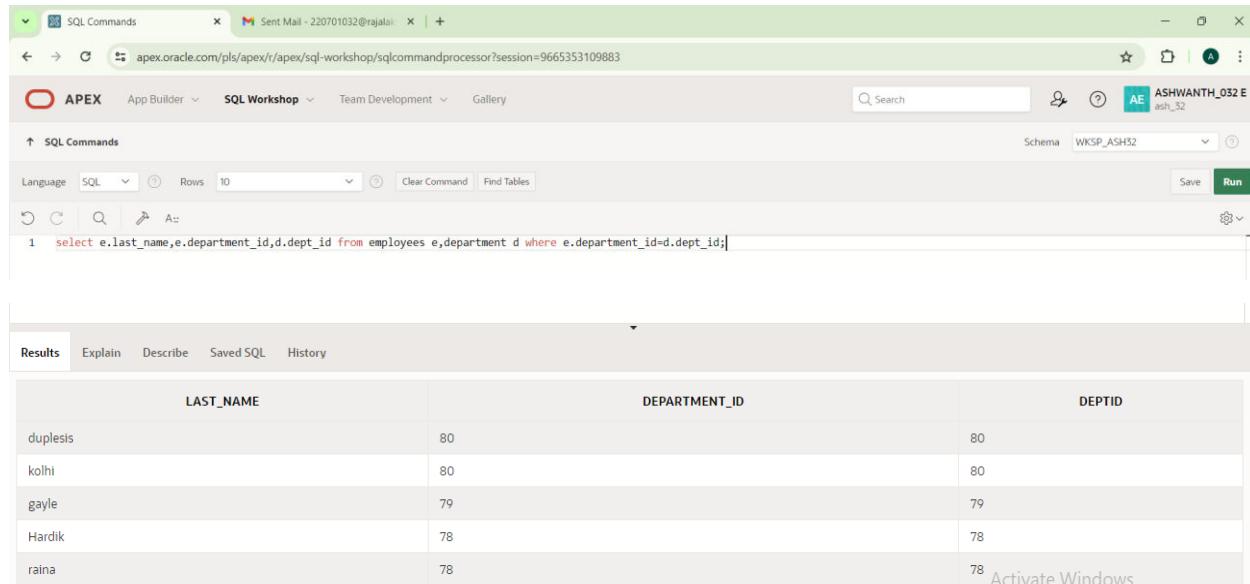
**DATE:**

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 employees e,MYDEPT d where  
e.department_id=d.deptid;
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The SQL Commands tab is active, displaying the query: "select e.last\_name,e.department\_id,d.deptid from employees e,MYDEPT d where e.department\_id=d.deptid;". The Results tab shows the output:

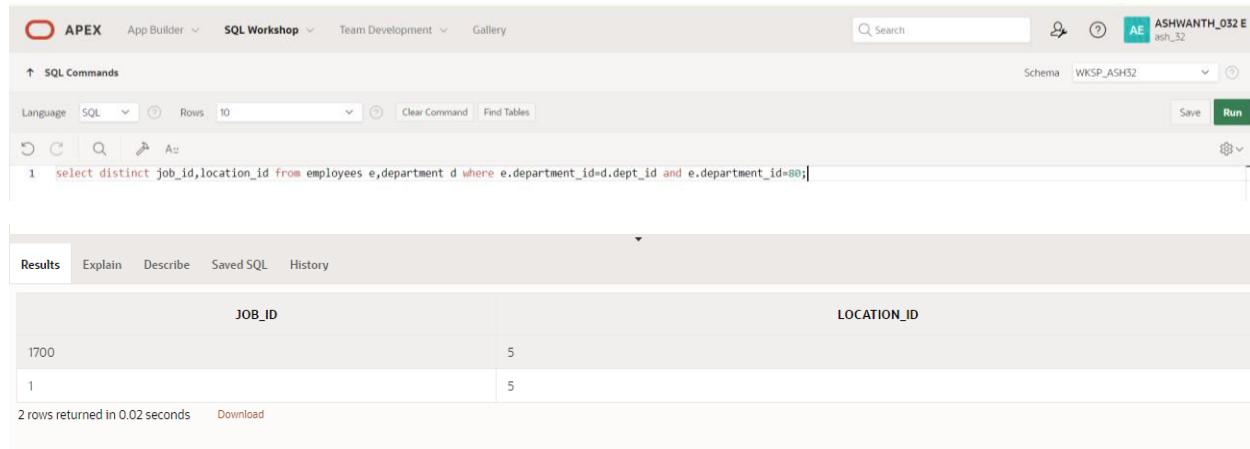
LAST_NAME	DEPARTMENT_ID	DEPTID
duplesis	80	80
kolhi	80	80
gayle	79	79
Hardik	78	78
raina	78	78

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 employees e,mydept d where e.department_id=d.deptid and  
e.department_id=80;
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The SQL Commands tab is active, displaying the query: "select distinct job\_id,location\_id from employees e,mydept d where e.department\_id=d.deptid and e.department\_id=80;". The Results tab shows the output:

JOB_ID	LOCATION_ID
1700	5
1	5

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

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

**QUERY:**

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

**OUTPUT:**

LAST_NAME	DEPARTMENT_ID	DEPT_NAME	LOC_ID	CITY
gayle	79	CSBS	3	toronto
duplesis	80	CSE	1	Chennai
kolhi	80	CSE	1	Chennai
duplesis	80	CSE	1	Chennai
kolhi	80	CSE	1	Chennai
gayle	79	CSBS	3	toronto

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

**QUERY:**

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

**OUTPUT:**

LAST_NAME	DEPT_NAME
gayle	CSBS
Hardik	Executive
raina	Executive

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 employees e join mydept d on(e.department\_id=d.deptid) join location on (d.loc\_id=location.locationid) where lower(location.city)='toronto';

**OUTPUT:**

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

LAST_NAME	DEPARTMENT_ID	JOB_ID	DEPT_NAME
gayle	79	3	CSBS

1 rows returned in 0.06 seconds    Download

Activate Windows  
Go to Settings to activate Windows.

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 employees

**OUTPUT:**

```
1 select last_name "Employee",employee_id "emp#",manager_name "manager",manager_id "Mgr#"
2 from employees
```

Employee	emp#	manager	Mgr#
Dhoni	30	King	8
duplesis	35	PARDIP	-
kolhi	18	VISHWA	6
smith	176	-	89
gayle	333	-	77
Hardik	43	-	Activate Windows Go to Settings to activate Windows.

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 SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema is set to WKSP\_ASH52. The command window contains the following SQL code:

```
1 select last_name "employee",employee_id "emp#",manager_name "maanager",manager_id "mgr#"
2  from employees
```

The results section displays the output of the query:

LAST_NAME	MANAGER_NAME	DEPT_NAME	LOC_ID	LOC	SALARY
smith	king	eee	3	Bangaluru	11000

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

A watermark at the bottom right of the interface says "Activate Windows Go to Settings to activate Windows."

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 employees e join employees 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 SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema is set to WKSP\_ASH52. The command window contains the following SQL code:

```
1 select e.department_id "Dept",e.last_name "colleague" from employees e join employees c on
(e.department_id=c.department_id) where e.employee_id <> c.employee_id order by
```

The results section displays the output of the query:

Dept	colleague
78	Hardik
78	raina
80	duplesis
80	kolhi

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

A watermark at the bottom right of the interface says "Activate Windows Go to Settings to activate Windows."

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

**QUERY:**

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

**OUTPUT:**

↑ SQL Commands

Language SQL Rows 10 Clear Command Find Tables Schema WKSP\_ARVINDH30 Save Run

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

Results Explain Describe Saved SQL History

LAST_NAME	JOB_ID	DEPT_NAME	SALARY	GRADE_LEVEL
duplesis	1	CSE	8999	5th
kolhi	1700	CSE	14000	4th
raina	43	Executive	15000	4th
Hardik	9	Executive	80200	1st
gayle	3	CSBS	12000	4th

5 rows returned in 0.01 seconds Download Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

**QUERY:**

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

**OUTPUT:**

↑ SQL Commands

Language SQL Rows 10 Clear Command Find Tables Schema WKSP\_ASH32 Save Run

```
1 select e.last_name, e.hire_date from employees e ,employees davies where davies.last_name='davies' and davies.hire_date < e.hire_date;
```

Results Explain Describe Saved SQL History

LAST_NAME	HIRE_DATE
Dhoni	09/07/2004
duplesis	09/08/2000
kolhi	12/07/2007
gayle	03/21/1998
Hardik	07/07/2001
Guptil	07/03/2003

Activate Windows Go to Settings to activate Windows.

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 employees e JOIN employee m ON e.manager_name = m.manager_name WHERE e.hire_date < m.hire_date;
```

**OUTPUT:**

↑ SQL Commands

Language SQL Rows 10 Clear Command Find Tables Schema WKSP\_ASH32 Save Run

```
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 employees e JOIN employees m ON e.manager_name =
3 m.manager_name WHERE e.hire_date < m.hire_date;
```

Results Explain Describe Saved SQL History

EMPLOYEE	EMP_HIRED	MANAGER	MGR_HIRED
gayle	04/05/1996	King	09/07/2004

1 rows returned in 0.01 seconds Download

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

# 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

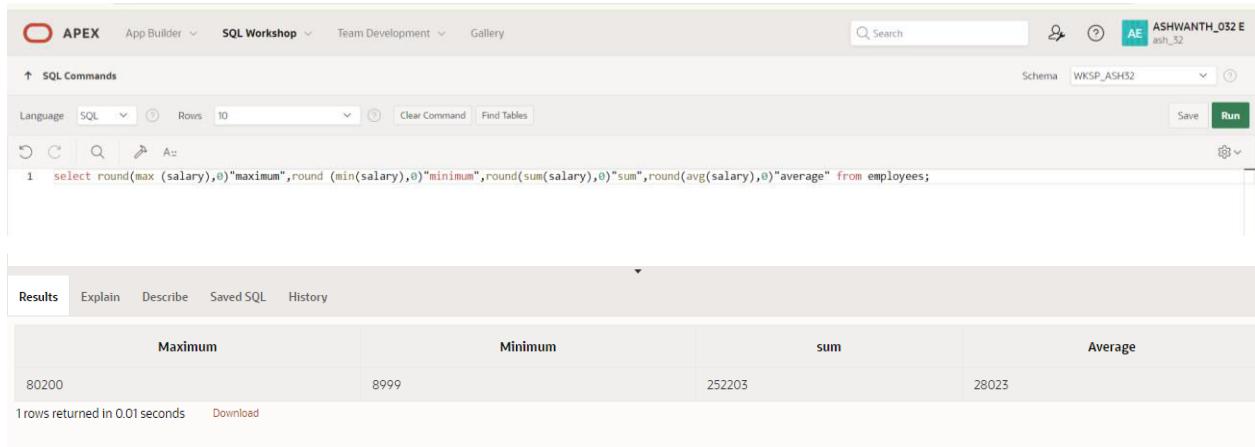
**FALSE**

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 EMPLOYEES;
```

**OUTPUT:**



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

```
1 select round(max(salary),0)"maximum",round(min(salary),0)"minimum",round(sum(salary),0)"sum",round(avg(salary),0)"average" from employees;
```

The Results tab displays the output:

	Maximum	Minimum	Sum	Average
	80200	8999	252203	28023

1 rows returned in 0.01 seconds

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 EMPLOYEES group by job_id;
```

**OUTPUT:**

```

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

```

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKSP\_ARVINDH30. The results table displays the following data:

JOB_ID	MAXIMUM	MINIMUM	sum	average
244	12000	12000	12000	12000
63	60000	60000	60000	60000
99	11000	11000	11000	11000
8	14000	14000	14000	14000
6	8999	8999	8999	8999
07	20000	20000	20000	20000

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

### QUERY:

Select job\_id, count(\*) from employees where job\_id='47' group by job\_id

### OUTPUT:

```

1 select job_id, count(*) from employees where job_id='47' group by job_id

```

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKSP\_ASH52. The results table displays the following data:

JOB_ID	COUNT(*)
47	2

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

[Activate Windows](#)

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

### QUERY:

select count(distinct manager\_id )"Number of managers" from employees;

```

1 select count(distinct manager_id )"number od managers" from employees;

```

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to WKSP\_ASH52. The results table displays the following data:

number od managers
4

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

### OUTPUT:

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

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 employees;

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The command entered is: `select max(salary)-min(salary) difference from employees;`. The results section displays a single row with the value 71201 under the heading 'DIFFERENCE'. Below the results, it says '1 rows returned in 0.01 seconds'.

9.Create a

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The command entered is: `select manager_id,min(salary) from employees where manager_id is not null group by manager_id having min(salary)>6000 order by min(salary) desc;`. The results section displays a table with two columns: 'MANAGER\_ID' and 'MIN(SALARY)'. The data is as follows:

MANAGER_ID	MIN(SALARY)
98	60000
24	31004
8	20000
2	15000
6	14000
77	12000

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

**QUERY:**

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

decode(to\_char(hire\_date,'YYYY'),1998,1,0)) "1998" from employees;

## OUTPUT:

The screenshot shows the SQL Commands tab with the following SQL code:

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

The Results tab displays the following data:

	TOTAL	1995	1996	1997	1998
9	0	0	0	0	1

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

Activate Windows  
Go to Settings to activate Windows.

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 employees group by job_id
```

## OUTPUT:

The screenshot shows the SQL Commands tab with the following SQL code:

```
1 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 employees group by job_id
```

The Results tab displays the following data:

Job	Dept20	dept50	dept80	dept90	TOTAL
244	-	12000	-	-	12000
63	-	-	-	-	60000
99	-	-	-	-	11000
8	-	-	-	-	14000
6	-	-	8999	-	8999
07	-	-	-	-	20000

Activate Windows  
Go to Settings to activate Windows.

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 employees e on(d.deptid =e.department_id ) group by d.dept_name ,d.loc;
```

## OUTPUT:

The screenshot shows a browser window for Oracle SQL Workshop. The URL is apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=9665353109883. The session user is ASHWANTH\_032 E. The SQL command executed is:

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

The results table has four columns: dept\_name, department location, Number of people, and salary. One row is returned:

dept_name	department location	Number of people	salary
CSE	Chennai	1	8999

1 rows returned in 0.06 seconds [Download](#)

**Evaluation Procedure**

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

**RESULT:**

# SUB QUERIES

**EX\_NO:9**

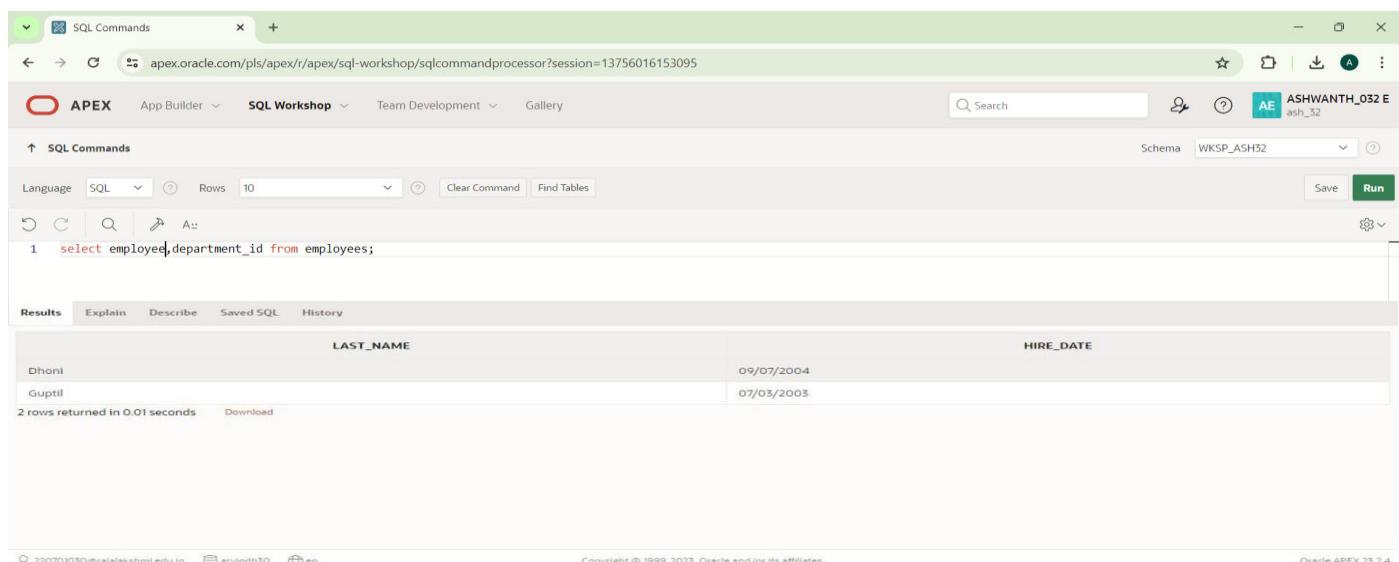
**DATE:**

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

**QUERY:**

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

**OUTPUT:**



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

```
1 select employee.department_id from employees;
```

The results window displays the following data:

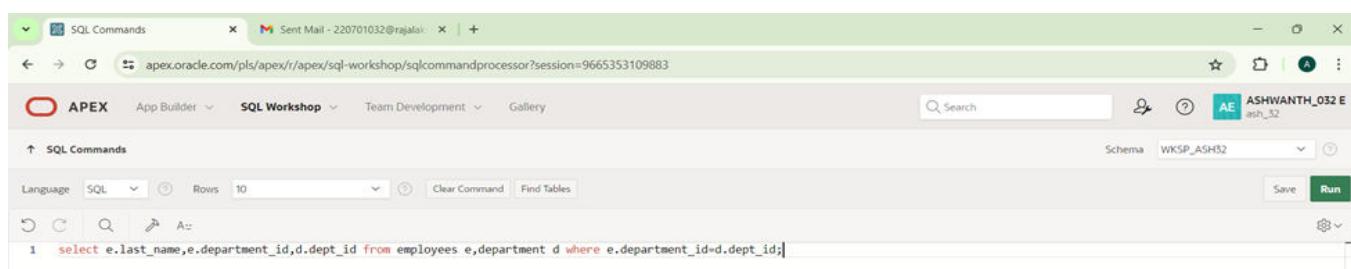
LAST_NAME	HIRE_DATE
Dhoni	09/07/2004
Guptil	07/03/2003

2 rows returned in 0.01 seconds

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

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

**OUTPUT:**



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

```
1 select e.last_name,e.department_id,d.dept_id from employees e,department d where e.department_id=d.dept_ids;
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	LAST_NAME	SALARY
899	Jadeja	31004
309	Guptil	60000
43	khan	80200

3 rows returned in 0.01 seconds Download

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

#### QUERY:

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

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (which is selected), 'Team Development', and 'Gallery'. The right side shows the user 'ASHWANTH\_032 E' and schema 'WKSP\_ASH32'. The main area is titled 'SQL Commands' with a 'Run' button. The code entered is:

```
1 select e.last_name, e.department_id, d.dept_id from employees e,department d where e.department_id=d.dept_id;
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	LAST_NAME
4	guptil

1 rows returned in 0.00 seconds Download

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

#### QUERY:

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

#### 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 'ARVINDH30' and schema 'WKSP\_ARVINDH30'. The main area is titled 'SQL Commands' with a 'Run' button. The code entered is:

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

The results table shows:

LAST_NAME	DEPARTMENT_ID	JOB_ID
Hardik	78	47

1 rows returned in 0.01 seconds Download

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user's profile: ASHWANTH\_032 E and ash\_52. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the following code:

```
1 select last_name "employee",employee_id "emp#",manager_name "maanager",manager_id "mgr#"  
2 from employees;
```

The Results tab displays the output:

LAST_NAME	SALARY
Dhoni	20000

Below the table, it says "1 rows returned in 0.01 seconds" and there is a "Download" link. The status bar at the bottom right says "Activate Windows".

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user's profile: ASHWANTH\_032 E and ash\_52. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the following code:

```
1 select last_name "employee",employee_id "emp#",manager_name "maanager",manager_id "mgr#"  
2 from employees;
```

Results Explain Describe Saved SQL History

DEPARTMENT_ID	LAST_NAME	JOB_ID
78	Hardik	47

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

Activate Windows  
Go to Settings to activate Windows.

220701030@rejalakshmi.edu.in arvindh30 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 25.2.4

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

### OUTPUT:

APEX App Builder SQL Workshop Team Development Gallery Search

↑ SQL Commands Schema WKSP\_ASH32

Language SQL Rows 10 Save Run

Clear Command Find Tables

```
1 select last_name "employee", employee_id "emp#", manager_name "manager", manager_id "mgr#"
2 from employees
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	LAST_NAME	SALARY
43	Hardik	80200
309	Guptil	60000

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

Activate Windows

]

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	

Total (15)	
Faculty Signature	

## RESULT:

# USING THE SET OPERATORS

**EX\_NO:10**

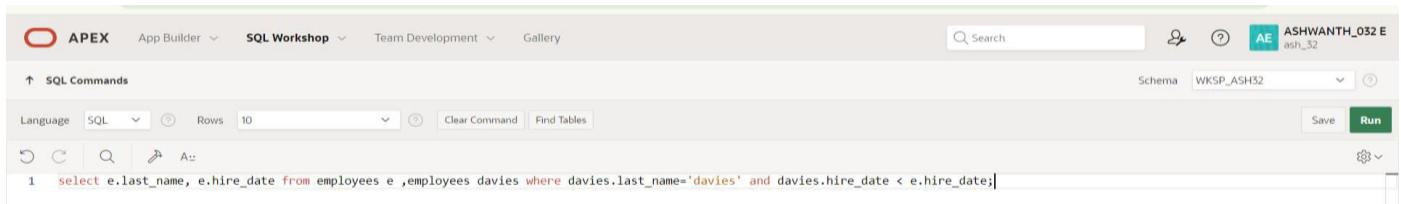
**DATE:**

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there are user profile icons and session information (ASHWANTH\_032 E, ash\_32). The main area is titled 'SQL Commands' with a sub-header 'Language: SQL'. It features a toolbar with icons for Undo, Redo, Find, Replace, and Run. Below the toolbar, there are dropdown menus for Language (SQL), Rows (10), and a Clear Command button. The SQL editor contains the following query:

```
1  select e.last_name, e.hire_date from employees e ,employees davies where davies.last_name='davies' and davies.hire_date < e.hire_date;
```

Results	Explain	Describe	Saved SQL	History
DEPARTMENT_ID				
4				
7				
11				
23				
45				
50				

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

#### QUERY:

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

#### OUTPUT:

Language	SQL	Rows	10	Clear Command	Find Tables	Save	Run
					Aw		
1 <code>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;</code>							
Results	Explain	Describe	Saved SQL	History			

COUNTRY_ID	COUNTRY_NAME
19	New Zealand

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

Activate Windows  
Go to Settings to activate Windows.

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

#### QUERY:

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

#### OUTPUT:

Language	SQL	Rows	10	Clear Command	Find Tables	Save	Run
					Aw		
1 2 <code>select job_id,department_id from employees where department_id=10 union</code> 3 <code>select job_id,department_id from employees where department_id=50 union</code> 4 <code>select job_id,department_id from employees where department_id=20;</code>							
Results	Explain	Describe	Saved SQL	History			

JOB_ID	DEPARTMENT_ID
244	50

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

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 employees
```

INTERSECT

```
SELECT employee_id,job_id FROM job_history;
```

### OUTPUT:



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

```
1 select e.department_id "dept",e.last_name "colleague" from employees e join employees e on (e.department_id=c.department_id) where e.employee_id <> c.employee_id order by
```

The results table has two columns: EMPLOYEE\_ID and JOB\_ID. The data is:

EMPLOYEE_ID	JOB_ID
18	55
30	47

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

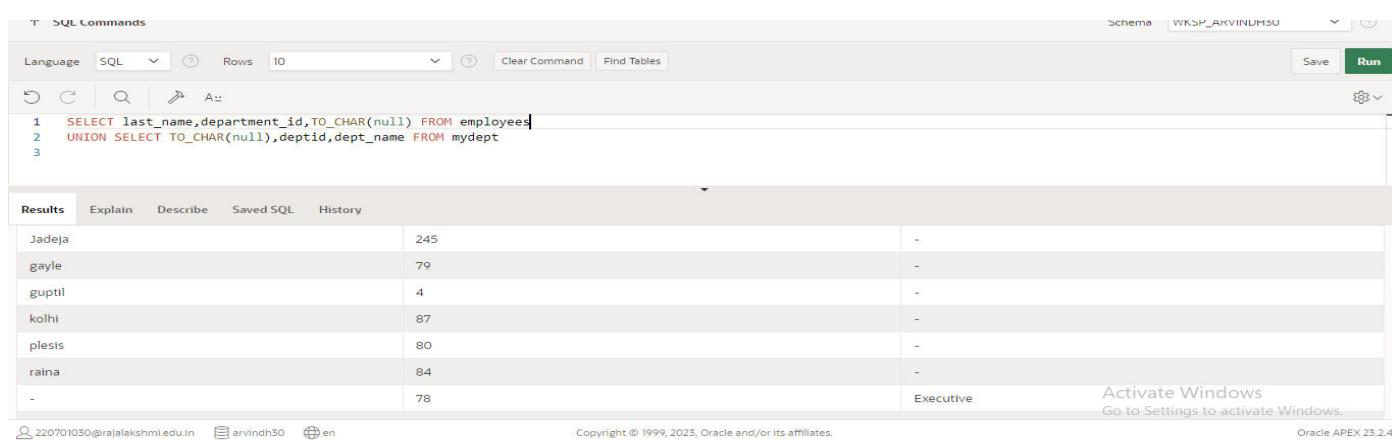
5.)The HR department needs a report with the following specifications: - Last name and department ID of all the employees from the EMPLOYEES table, regardless of whether or not they belong to a department. - Department ID and department name of all the departments from the DEPARTMENTS table, regardless 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 employees
```

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

### OUTPUT:



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

```
1 SELECT last_name,department_id,TO_CHAR(null) FROM employees
2 UNION SELECT TO_CHAR(null),deptid,dept_name FROM mydept
3
```

The results table has four columns: last\_name, department\_id, dept\_name, and a column with a blank value. The data is:

last_name	department_id	dept_name	
Jadeja	245		-
gayle	79		-
guptil	4		-
kolhi	87		-
plesis	80		-
raina	84		-
-	78		Executive

Activate Windows  
Go to Settings to activate Windows.

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:**

# CREATING VIEWS

EX\_NO:11

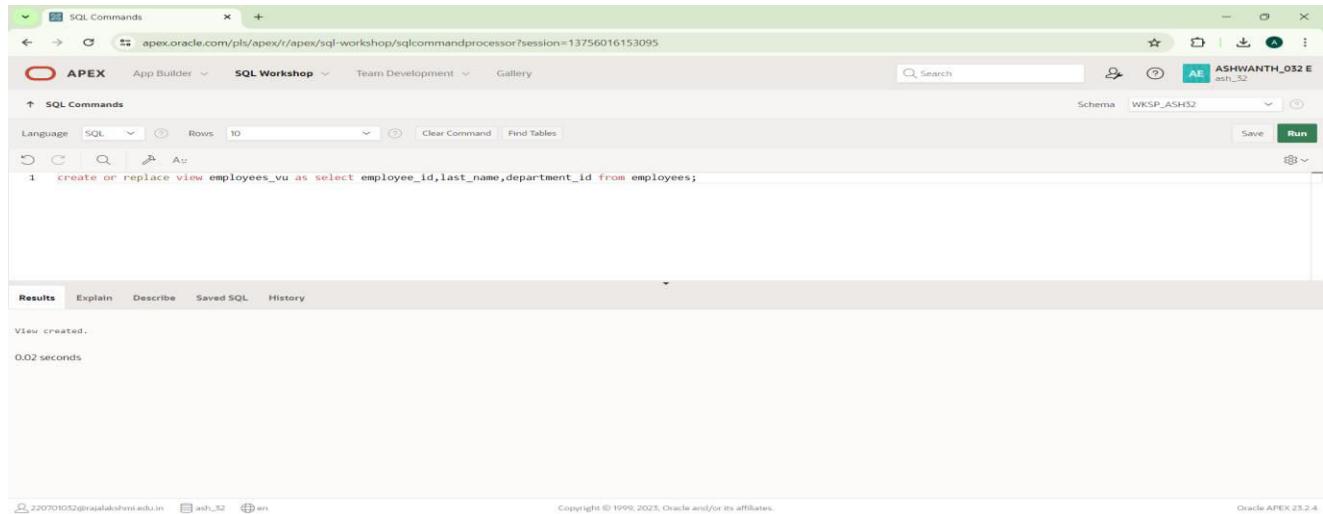
DATE:

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:**



A screenshot of the Oracle APEX SQL Workshop interface. The title bar shows 'SQL Commands' and the URL 'apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=13756016153095'. The main area contains the SQL command:

```
create or replace view employees_vu as select employee_id, last_name employee,
department_id FROM employees;
```

The results pane shows the output:

```
View created.
0.02 seconds
```

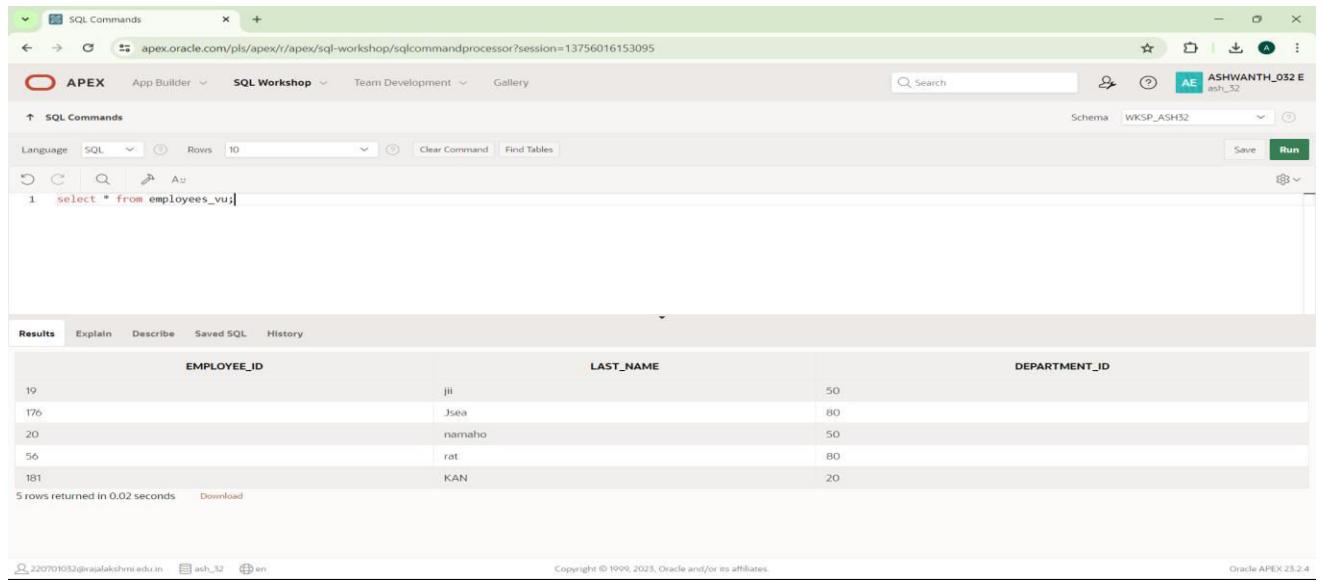
At the bottom, it says 'Copyright © 1999, 2025, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

**QUERY:**

```
select * from employees_vu;
```

**OUTPUT:**



A screenshot of the Oracle APEX SQL Workshop interface. The title bar shows 'SQL Commands' and the URL 'apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=13756016153095'. The main area contains the SQL command:

```
select * from employees_vu;
```

The results pane displays the data from the view:

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
19	jii	50
176	Jsea	80
20	namaho	50
56	rat	80
181	KAN	20

Below the table, it says '5 rows returned in 0.02 seconds' and 'Download'. At the bottom, it says 'Copyright © 1999, 2025, 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 SQL Commands tab is selected, displaying the query: `select view_name, text from user_views;`. The Results tab is selected, showing the output:

VIEW_NAME	TEXT
DEPT50	SELECT employee_id empno, last_name employee, department_id deptno FROM employees WHERE department_id = 50 WITH CHECK OPTION
EMPLOYEES_VU	SELECT employee_id, last_name employee, department_id FROM employees

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

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

**QUERY:**

SELECT employee, department\_id FROM employees\_vu;

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL Commands tab is selected, displaying the query: `select employee, department_id from employees_vu;`. The Results tab is selected, showing the output:

EMPLOYEE	DEPARTMENT_ID
Dhoni	7
plesis	80
kolhi	87
Davies	97
gayle	79
Hardik	78

Activate Windows  
Go to Settings to activate Windows.

220701030@rajalakshmi.edu.in arvindh30 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

#### QUERY:

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

#### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'SQL Commands' is selected. Below it, the command window displays the SQL code for creating the view:

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

Below the command window, the results tab is active, showing the message "View created." and a execution time of "0.05 seconds".

At the bottom of the interface, there are user and system status indicators, and a copyright notice: "Copyright © 1999, 2023, Oracle and/or its affiliates." and "Oracle APEX 23.2.4".

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

#### QUERY:

```
Describe dept50;
```

#### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface with the 'Describe' tab selected in the results panel. The object being described is 'DEPT50'. The results panel displays the structure of the view:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPT50	EMPNO	NUMBER	-	6	0	-	✓	-	-
	EMPLOYEE	VARCHAR2	25	-	-	-	✓	-	-
	DEPTNO	NUMBER	-	4	0	-	✓	-	-
	FIRST_NAME	VARCHAR2	20	-	-	-	✓	-	-

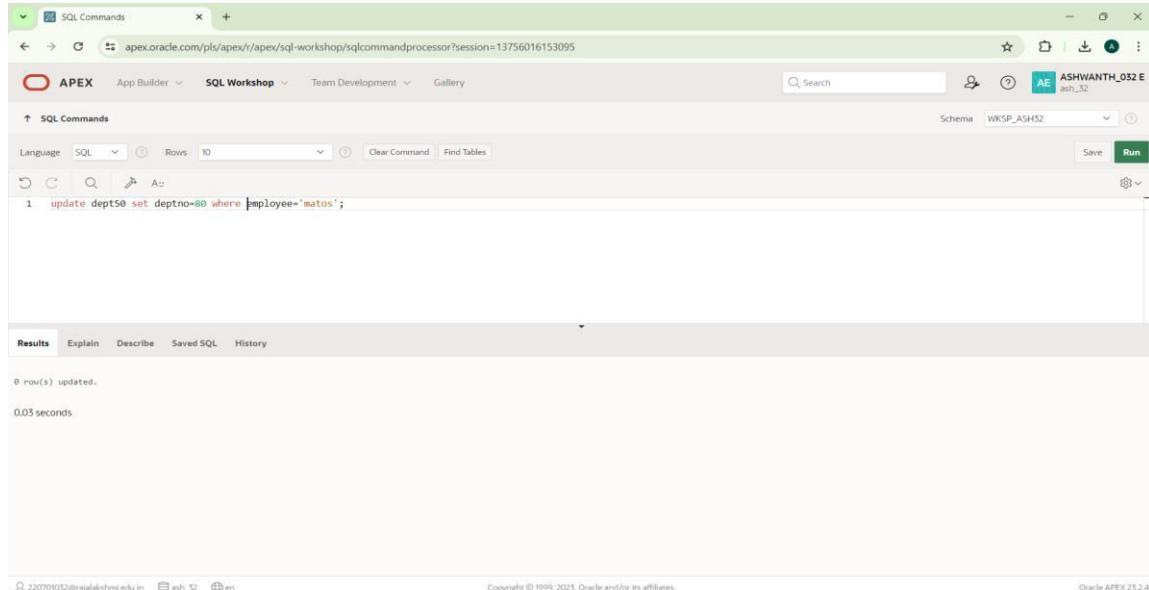
At the bottom of the interface, there are user and system status indicators, and a copyright notice: "Copyright © 1999, 2023, Oracle and/or its affiliates." and "Oracle APEX 23.2.4".

7.) Attempt to reassign Matos to department 80

**QUERY:**

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

**OUTPUT:**



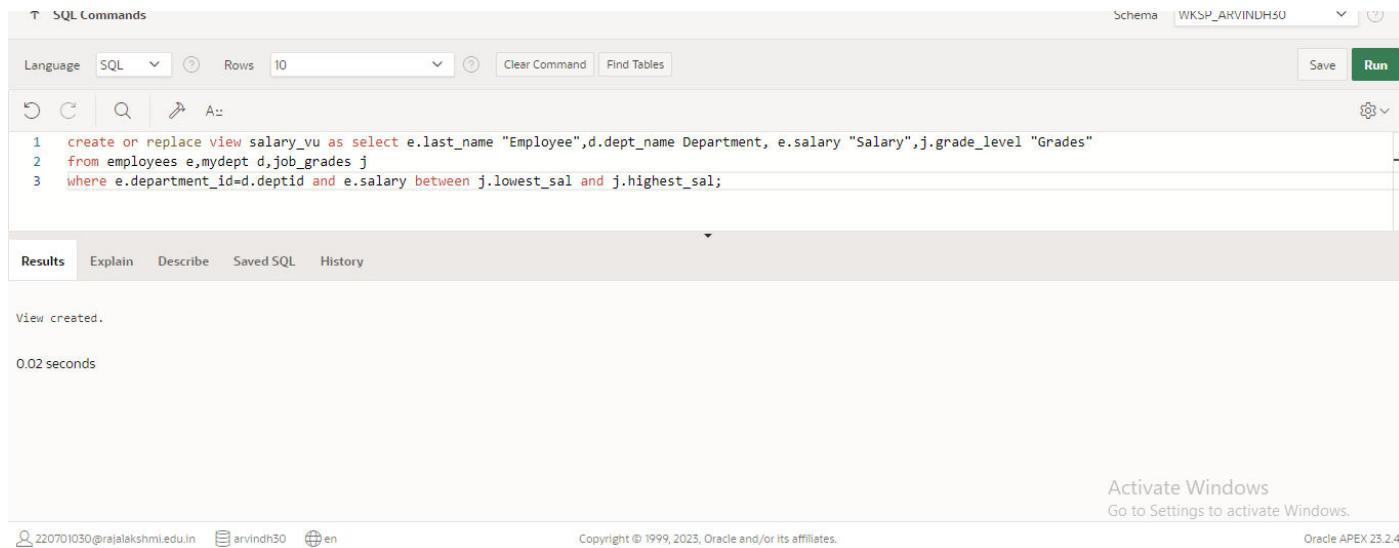
The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, the command `update dept50 set deptno=80 where employee='matos';` is entered. The results section shows "0 row(s) updated." and a execution time of "0.03 seconds". The bottom status bar indicates the user is 220701052@rajalakshmi.edu.in and the session ID is ash\_32.

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 SQL Commands tab, the command to create the view is entered. The results section shows "View created." and an execution time of "0.02 seconds". The bottom status bar indicates the user is 220701030@rajalakshmi.edu.in and the session ID is arvindh30.

Activate Windows  
Go to Settings to activate Windows.

Oracle APEX 23.2.4

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

**RESULT:**

# EXERCISE 12

## PRACTICE QUESTIONS

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Table Created.

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

DESCRIBE f\_global\_locations;

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

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

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

# PRIMARY KEY, FOREIGN KEY, and CHECK Constraints

1. What is the purpose of a

- PRIMARY KEY
- FOREIGN KEY
- CHECK CONSTRAINT

**a. PRIMARY KEY**

Uniquely identify each row in table.

**b. FOREIGN KEY**

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

**c. CHECK CONSTRAINT**

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

2. Using the column information for the animals table below, name constraints where applicable at the table level, otherwise name them at the column level. Define the primary key (animal\_id). The license\_tag\_number must be unique. The admit\_date and vaccination\_date columns cannot contain null values.

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

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

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

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

ANIMAL_ID	NAME	LICENSE_TAG_NUMBE R	ADMIT_DAT E	ADOPTION_I D	VACCINATION_DAT E
101	Spot	35540	10-Oct-2004	205	12-Oct-2004

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

```
SELECT * FROM animals;
```

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

**COLUMN LEVEL STATEMENT:**

```
ALTER TABLE animals
```

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

**TABLE LEVEL STATEMENT:**

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

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

a. ON DELETE CASCADE

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

b. ON DELETE SET NULL

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

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

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

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

# PRACTICE PROBLEM

## Managing Constraints

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

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

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

- ADD
- DROP
- ENABLE
- DISABLE

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**1 row(s) inserted.**

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

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

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

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

**1 row(s) deleted.**

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

**Table altered.**

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

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

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

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

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

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

# EXERCISE 13

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

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

4. REPLACE view\_d\_songs. Add type\_code to the column list. Use aliases for all columns.

Or use alias after the CREATE statement as shown.

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

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

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

6. It is company policy that only upper-level management be allowed access to individual employee salaries. The department managers, however, need to know the minimum, maximum, and average salaries, grouped by

department. Use the Oracle database to prepare a view that displays the needed information for department managers.

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

# DML Operations and Views

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

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

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

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

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

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

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

```
SELECT * FROM view_copy_d_songs;
```

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

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

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

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

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

```
SELECT * FROM read_copy_d_cds;
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**Group functions**  
**GROUP BY CLAUSE**  
**DISTINCT**  
**pseudocolumn ROWNUM Keyword**

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

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

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

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

# Managing Views

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

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

```
SELECT * FROM view_copy_d_songs;
```

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

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

ORA-00942: table or view does not exist

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

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

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

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

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

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

# **Indexes and Synonyms**

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

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

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

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

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

3. When will an index be created automatically?

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

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

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

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

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

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

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

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

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

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

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

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

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

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

10. Drop the synonym that you created in question

**DROP SYNONYM dj\_tracks2;**

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

**RESULT:**

# CONTROLLING USER ACCESS

**EX\_NO:15**

**DATE:**

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

The CREATE SESSION system privilege

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

The CREATE TABLE privilege

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

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

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

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

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

The ALTER USER statement

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

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

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

7. Query all the rows in your DEPARTMENTS table.

SELECT \* FROM departments;

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

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

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

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

SELECT table\_name FROM user\_tables;

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

Team 1 revokes the privilege.

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

Team 2 revokes the privilege.

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

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

Team 1 executes this INSERT statement.

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

Team 2 executes this INSERT statement.

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

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

**RESULT:**

# PL/SQL

## CONTROL STRUCTURES

**EX\_NO:**

**DATE:**

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

**QUERY:**

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

**OUTPUT:**

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

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

```
Incentive = 8400
Statement processed.

0.00 seconds
```

At the bottom of the page, there are footer links for user information and copyright notice.

**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;
/
```

**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 'ASHWANTH\_032 E' and schema 'WKSP\_ASH52'. The main area is titled 'SQL Commands' with tabs for 'Language', 'SQL', 'Rows (10)', 'Clear Command', and 'Find Tables'. Below these are icons for 'Copy', 'Run', and 'Save'. The code editor contains the following PL/SQL block:

```
1  DECLARE
2  WELCOME varchar2(10) := 'welcome';
3  BEGIN
4  DBMS_Output.Put_Line("Welcome");
5  END;
6  /
7
8
```

Below the code editor is a 'Results' tab. A yellow box highlights the error message from line 4:

```
Error at line 4/23: ORA-06550: line 4, column 23:
PLS-00201: identifier 'Welcome' must be declared
ORA-06512: at "SYS.0000_DBMS_SQL_APEX_230200", line 861
ORA-06501: line 4, column 1
PL/SQL: Statement ignored
```

The code block is also shown within the results area:

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

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

**QUERY:**

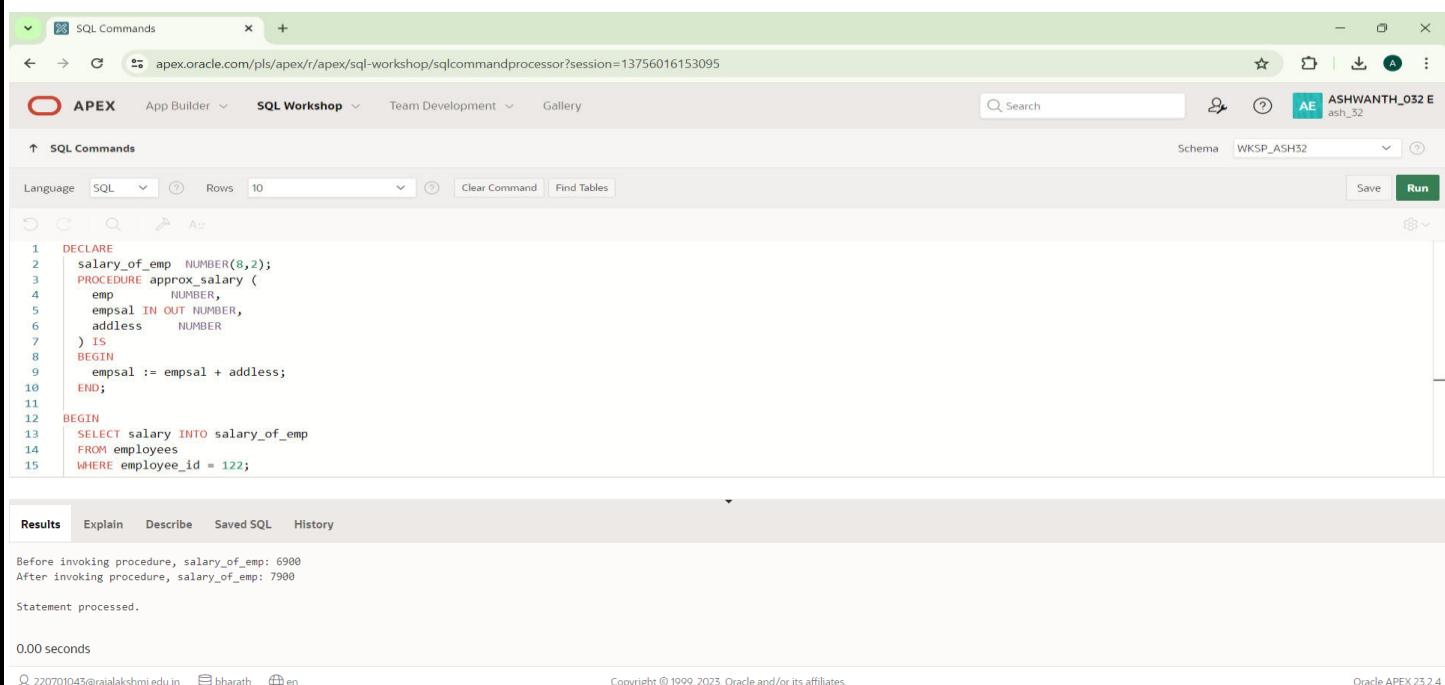
```
DECLARE
salary_of_emp NUMBER(8,2);
PROCEDURE approx_salary (
    emp      NUMBER,
    empsal IN OUT NUMBER,
    adddress NUMBER
) IS
BEGIN
    empsal := empsal + adddress;
END;
```

```

BEGIN
SELECT salary INTO salary_of_emp
FROM employees
WHERE employee_id = 122;
DBMS_OUTPUT.PUT_LINE
('Before invoking procedure, salary_of_emp: ' || salary_of_emp);
approx_salary (100, salary_of_emp, 1000);
DBMS_OUTPUT.PUT_LINE
('After invoking procedure, salary_of_emp: ' || salary_of_emp);
END;
/

```

#### OUTPUT:



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

```

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

```

The results pane displays the output of the query:

```

Before invoking procedure, salary_of_emp: 6900
After invoking procedure, salary_of_emp: 7900
Statement processed.

0.00 seconds

```

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

#### QUERY:

```

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

```

```

DBMS_OUTPUT.PUT_LINE( boo_name || ' = FALSE');
END IF;
END;
/

```

## OUTPUT:

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

```

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

```

Below the code editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected. The output pane shows the message 'Procedure created.' and a execution time of '0.03 seconds'. At the bottom of the page, there is footer information including the user's email (220701052@rajalakshmi.edu.in), session ID (ash\_32), and copyright notice (Copyright © 1999-2023, Oracle and/or its affiliates).

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 SQL Commands tab is active, displaying the following PL/SQL code:

```

6   BEGIN
7     IF test_string LIKE pattern THEN
8       DBMS_OUTPUT.PUT_LINE ('TRUE');
9     ELSE
10       DBMS_OUTPUT.PUT_LINE ('FALSE');
11     END IF;
12   BEGIN
13     pat_match('Blweate', 'B%a_e');
14     pat_match('Blweate', 'B%A_E');
15   END;
16
17 /

```

The results pane shows the output of the code execution:

```

TRUE
FALSE
Statement processed.
0.00 seconds

```

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

## QUERY:

```

DECLARE
num_small NUMBER := 8;
num_large NUMBER := 5;
num_temp NUMBER;
BEGIN
IF num_small > num_large THEN

```

```

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

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

```

## OUTPUT:

The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, 'SQL Commands' is selected. The main area contains the following PL/SQL code:

```

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

```

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

```

num_small = 5
num_large = 8
Statement processed.

0.00 seconds

```

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

## QUERY:

```

DECLARE
PROCEDURE test1 (
  sal_achieve NUMBER,
  target_qty NUMBER,
  emp_id NUMBER
)
IS
  incentive NUMBER := 0;
  updated VARCHAR2(3) := 'No';
BEGIN
  IF sal_achieve > (target_qty + 200) THEN
    incentive := (sal_achieve - target_qty)/4;
    UPDATE employees

```

```

SET salary = salary + incentive
WHERE employee_id = emp_id;
updated := 'Yes';
END IF;
DBMS_OUTPUT.PUT_LINE (
'Table updated? ' || updated || ',' ||
'incentive = ' || incentive || ''
);
END test1;
BEGIN
test1(2300, 2000, 144);
test1(3600, 3000, 145);
END;
/

```

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, the following PL/SQL code is run:

```

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;
/

```

In the Results tab, the output is:

```

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

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;
/

```

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

```

15
16  );
17 END test1;
18 BEGIN
19   test1(45000);
20   test1(36000);
21   test1(28000);
22 END;
23 /
24

```

In the Results tab, the output is displayed:

```

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

The screenshot shows the Oracle APEX SQL Workshop interface. The code entered is a PL/SQL block that joins the employees and departments tables, counts employees by department, and prints messages about the count. The output pane shows the results for three departments: Sales, Marketing, and Research.

```

10  CO_L_CMP
11  FROM employees e
12  join departments d
13  ON e.department_id = d.department_id
14  WHERE e.department_id = get_dep_id;
15  dbms_output.Put_line ('The employees are in the department'||get_dep_id||' is: '
16  ||To_char(tot_emp));
17  IF tot_emp >= 45 THEN
18      dbms_output.Put_line ('There are no vacancies in the department'||get_dep_id);
19  ELSE
20      dbms_output.Put_line ('There are'||to_char(45-tot_emp)||' vacancies in department'||get_dep_id );
21  END IF;
22
23

```

**Results**

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

Copyright © 1999, 2025, Oracle and/or its affiliates. Oracle APEX 23.2

**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:

```

SQL Commands
apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=13756016153095
APEX App Builder SQL Workshop Team Development Gallery Search ASHWANTH_032 E ash_32
Schema WKSP_ASH32 Save Run
SQL Commands
Language SQL Rows 10 Clear Command Find Tables
16   sale_achieved := || SQL_achieve || , incentive := || incentive || ;
17 END test1;
18 BEGIN
19   test1(45000);
20   test1(36000);
21   test1(28000);
22 END;
23 /
24

```

**Results**

```

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

0.01 seconds

```

220701052@rajalakshmi.edu.in ash\_32 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

#### QUERY:

```

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

```

```

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

```

## OUTPUT:

```

11  DBMS_OUTPUT.PUT_LINE('Employee ID | Full Name | Job Title | Hire Date | Salary');
12  DBMS_OUTPUT.PUT_LINE('-----');
13  OPEN c_employees;
14  FETCH c_employees INTO v_employee_id, v_full_name, v_job_id, v_hire_date, v_salary;
15  WHILE c_employees%FOUND LOOP
16    DBMS_OUTPUT.PUT_LINE(v_employee_id || ' ' || v_full_name || ' ' || v_job_id || ' ' || v_hire_date || ' ' || v_salary);
17    FETCH c_employees INTO v_employee_id, v_full_name, v_job_id, v_hire_date, v_salary;
18  END LOOP;
19  CLOSE c_employees;
20
21
22

```

Employee ID	Full Name	Job Title	Hire Date	Salary
19	Fazil Jili	34	02/07/1995	13000
176	Dani Jsea	78	03/08/2000	10000
20	Matos Namaho	36	02/20/1998	11000
56	Koli Rat	88	05/01/1998	9000
181	ASH KAN	33	01/05/1994	7000

Statement processed.

0.02 seconds

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

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 APEX SQL Workshop interface. The code area contains a PL/SQL block that prints employee details using DBMS\_OUTPUT.PUT\_LINE. The output pane shows the results for four employees.

```

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('-----');

```

Employee ID	Full Name	Job Title	Hire Date	Salary
176	Fazil Jiji	SA	02/07/1995	13000
20	Dani Jsee	SA	03/08/2000	10000
56	Matos Namaho	SA	02/20/1998	11000
181	Koli Rat	SA	05/01/1998	9000
	ASH KAN	SA	01/05/1994	70000

**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;

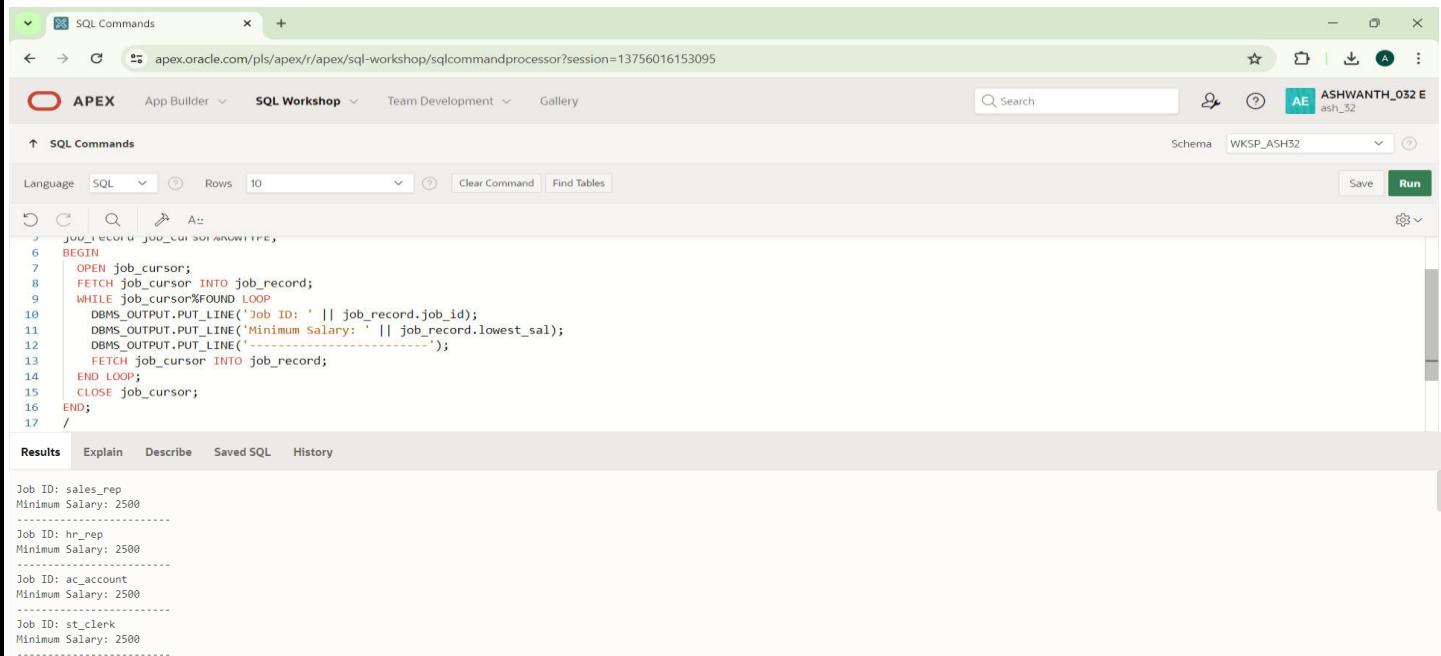
```

```

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 SQL Commands tab, a PL/SQL block is run. The output pane shows the results of the DBMS\_OUTPUT.PUT\_LINE statements, listing job IDs and their minimum salaries.

```

Job ID: sales_rep
Minimum Salary: 2500
-----
Job ID: hr_rep
Minimum Salary: 2500
-----
Job ID: ac_account
Minimum Salary: 2500
-----
Job ID: st_clerk
Minimum Salary: 2500
-----
```

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

#### QUERY:

```

DECLARE
  CURSOR employees_cur IS
    SELECT employee_id, last_name, job_id, start_date

```

```

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

```

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, the following PL/SQL block is executed:

```

12  INTO emp_start_date
13  FROM job_history
14  WHERE employee_id = emp_sal_rec.employee_id;
15  IF emp_start_date IS NULL THEN
16    | emp_start_date := emp_sal_rec.start_date;
17  END IF;
18  dbms_output.Put_line(Rpad(emp_sal_rec.employee_id, 15)
19    || Rpad(emp_sal_rec.last_name, 25)
20    || Rpad(emp_sal_rec.job_id, 35)
21    || To_char(emp_start_date, 'dd-mon-yyyy'));
22 END LOOP;
23 END;
24 /

```

In the Results tab, the output is:

Employee ID	Last Name	Job Id	Start Date
125	Johnson	hr_rep	22-apr-1999
125	Johnson	hr_rep	22-apr-1999

Statement processed.

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

## QUERY:

DECLARE

```

v_employee_id employees.employee_id%TYPE;
v_first_name employees.last_name%TYPE;

```

```

v_end_date job_history.end_date%TYPE;
CURSOR c_employees IS
  SELECT e.employee_id, e.first_name, jh.end_date
  FROM employees e
  JOIN job_history jh ON e.employee_id = jh.employee_id;
BEGIN
  OPEN c_employees;
  FETCH c_employees INTO v_employee_id, v_first_name, v_end_date;
  WHILE c_employees%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_employee_id);
    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_first_name);
    DBMS_OUTPUT.PUT_LINE('End Date: ' || v_end_date);
    DBMS_OUTPUT.PUT_LINE('-----');
    FETCH c_employees INTO v_employee_id, v_first_name, v_end_date;
  END LOOP;
  CLOSE c_employees;
END;

```

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, a PL/SQL block is run. The code uses DBMS\_OUTPUT.PUT\_LINE to print employee details. The output shows two records for employees with IDs 125 and 126, with their names and end dates.

```

Employee ID: 125
Employee Name: Emily
End Date: 04/21/1999
-----
Employee ID: 126
Employee Name: Emily
End Date: 03/21/1997
-----
Statement processed.

```

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

**RESULT:**

# PROCEDURES AND FUNCTIONS

EX\_NO: 17

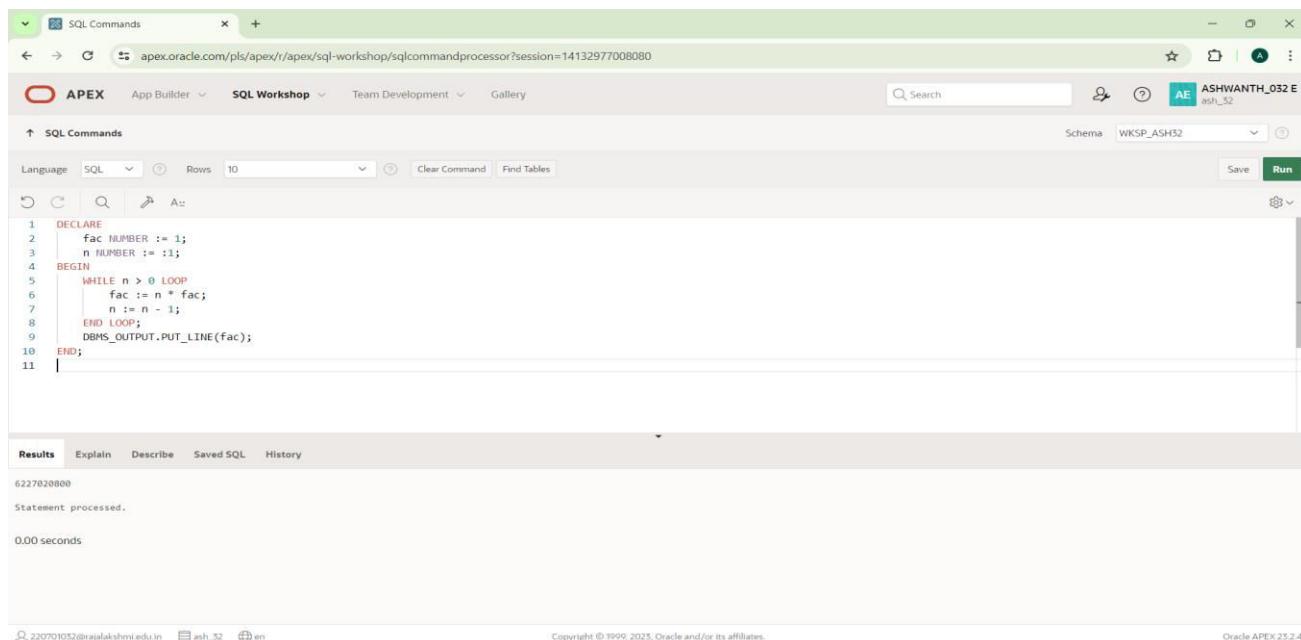
DATE:

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

QUERY:

```
DECLARE
    fac NUMBER := 1;
    n NUMBER := :1;
BEGIN
    WHILE n > 0 LOOP
        fac := n * fac;
        n := n - 1;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE(fac);
END;
```

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, a PL/SQL block is written to calculate the factorial of a number. The code uses a loop to multiply the current value of 'fac' by 'n' until 'n' reaches 0. The result is then output using DBMS\_OUTPUT.PUT\_LINE. The code is as follows:

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

In the Results tab, the output of the query is shown. It displays the message "Statement processed." and "0.00 seconds". At the bottom, there is copyright information: "Copyright © 1999, 2025, Oracle and/or its affiliates." and "Oracle APEX 23.2.4".

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

**QUERY:**

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

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

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

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

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

```

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

```

## OUTPUT:

The screenshot shows the Oracle Apex SQL Workshop interface. In the SQL Commands tab, a PL/SQL block is run. The code declares variables (v\_book\_id, v\_title, v\_author, v\_year\_published) and sets v\_title to 'Initial Title'. It then calls a procedure get\_book\_info with p\_book\_id set to 1. Finally, it uses DBMS\_OUTPUT.PUT\_LINE to print the title, author, and year published. The results panel shows the output: Title: 1984, Author: George Orwell, Year Published: 1949.

```

21
22  DECLARE
23      v_book_id NUMBER := 1;
24      v_title VARCHAR2(100);
25      v_author VARCHAR2(100);
26      v_year_published NUMBER;
27  BEGIN
28      v_title := 'Initial Title';
29
30      get_book_info(p_book_id => v_book_id, p_title => v_title, p_author => v_author, p_year_published => v_year_published);
31
32      DBMS_OUTPUT.PUT_LINE('Title: ' || v_title);
33      DBMS_OUTPUT.PUT_LINE('Author: ' || v_author);
34      DBMS_OUTPUT.PUT_LINE('Year Published: ' || v_year_published);
35  END;

```

Results Explain Describe Saved SQL History

Title: 1984  
Author: George Orwell  
Year Published: 1949

Statement processed.  
0.02 seconds

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

**RESULT:** Thus the above code has been executed successfully.

# TRIGGER

EX\_NO: 18

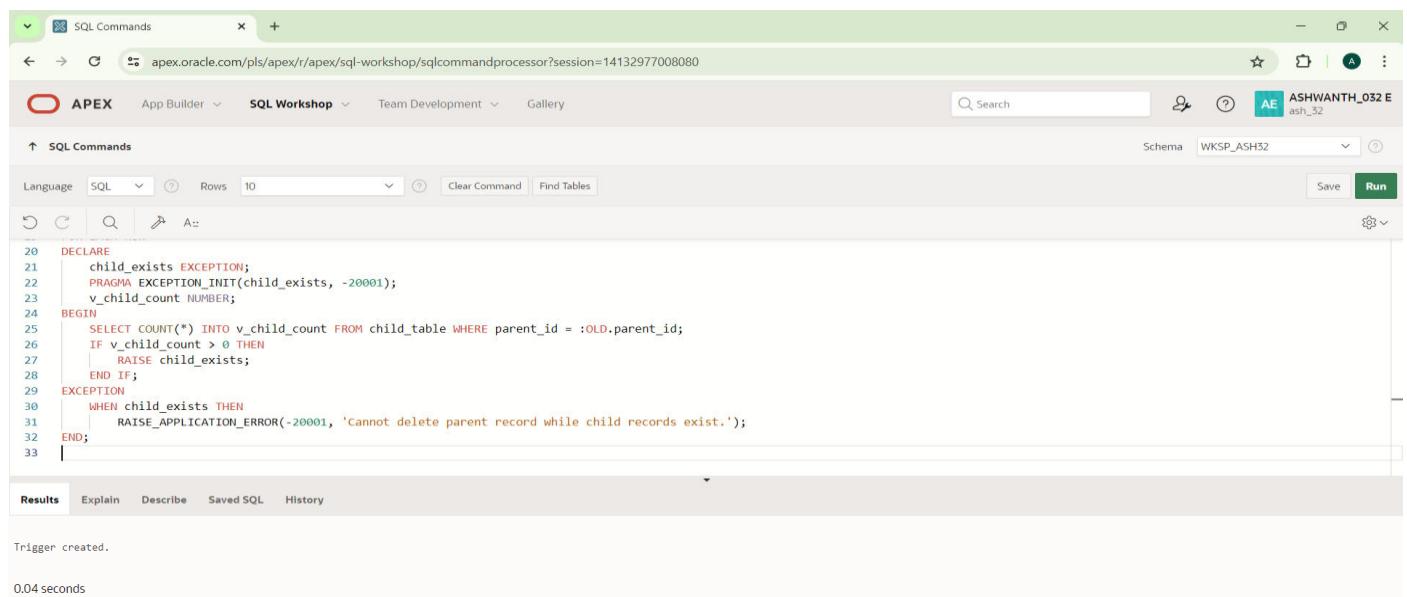
DATE:

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. The command is identical to the one provided in the text above. At the bottom of the SQL editor, the 'Run' button is visible. Below the editor, the results pane shows the message 'Trigger created.' and a execution time of '0.04 seconds'.

```
20 DECLARE
21     child_exists EXCEPTION;
22     PRAGMA EXCEPTION_INIT(child_exists, -20001);
23     v_child_count NUMBER;
24 BEGIN
25     SELECT COUNT(*) INTO v_child_count FROM child_table WHERE parent_id = :OLD.parent_id;
26     IF v_child_count > 0 THEN
27         RAISE child_exists;
28     END IF;
29 EXCEPTION
30     WHEN child_exists THEN
31         RAISE_APPLICATION_ERROR(-20001, 'Cannot delete parent record while child records
exist.');
32 END;
33 |
```

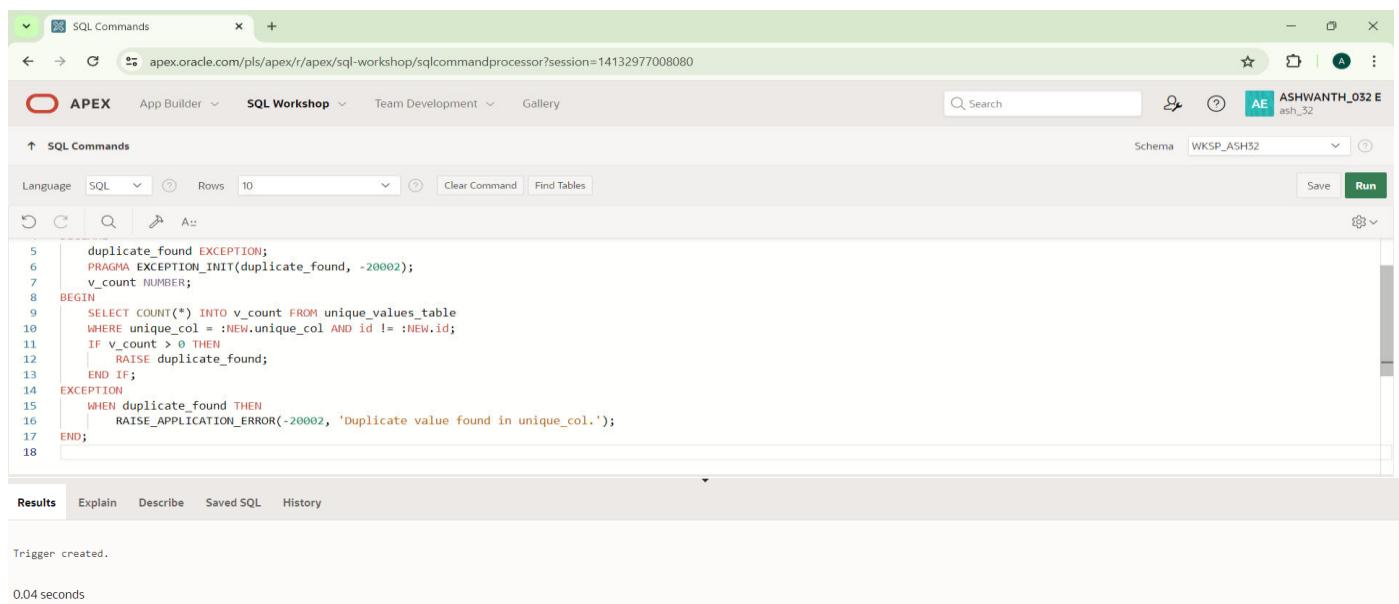
Trigger created.  
0.04 seconds

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

**QUERY:**

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

**OUTPUT:**



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

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

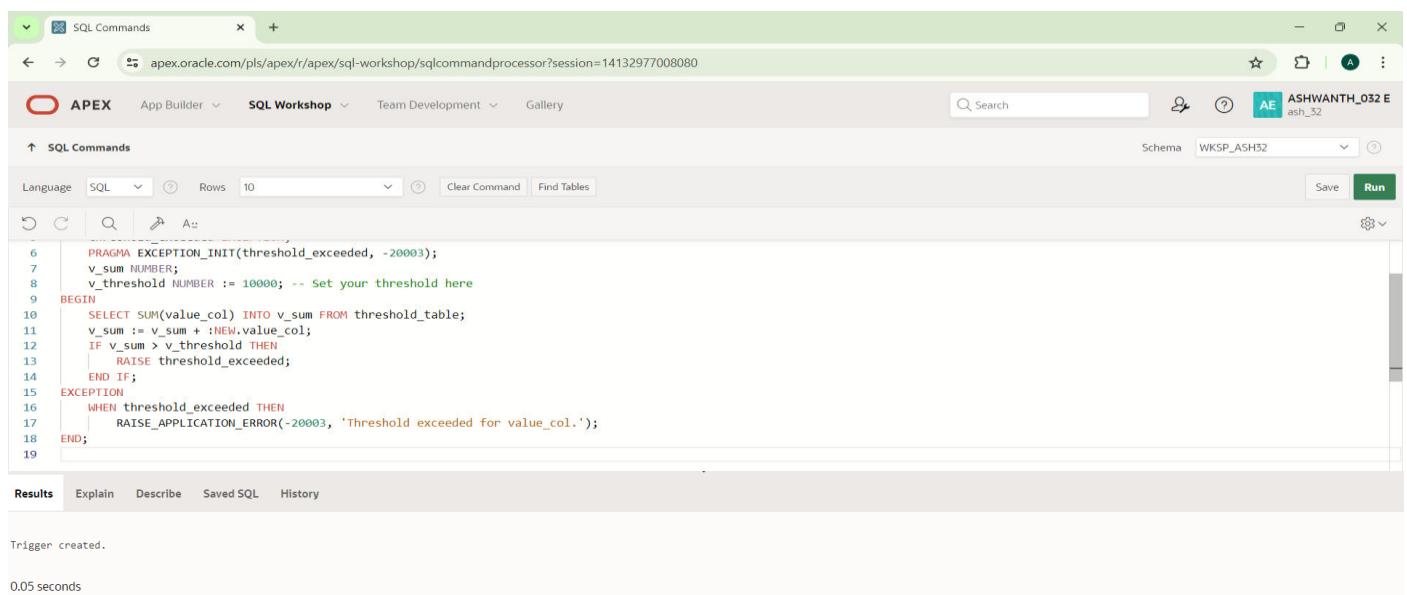
At the bottom of the interface, the 'Results' tab is active, showing the message "Trigger created." and a execution time of "0.04 seconds".

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

**QUERY:**

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

**OUTPUT:**



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 ASHWANTH\_032 E with session ID ash\_32. The main workspace is titled "SQL Commands". The SQL editor contains the PL/SQL code for the trigger. The code defines a trigger named "check\_threshold" that fires before inserting or updating rows in the "threshold\_table". It declares an exception "threshold\_exceeded" and initializes it with the value -20003. It also declares variables "v\_sum" and "v\_threshold" with initial values of 0 and 10000 respectively. The trigger body performs a SELECT statement to get the current sum of values in the table, adds the new value from the ":NEW.value\_col" bind variable to it, and then checks if the total sum exceeds the threshold. If it does, it raises the "threshold\_exceeded" exception. An exception block handles this raised exception by calling the "RAISE\_APPLICATION\_ERROR" procedure with parameters -20003 and a descriptive message. The "Run" button at the bottom right of the editor is highlighted.

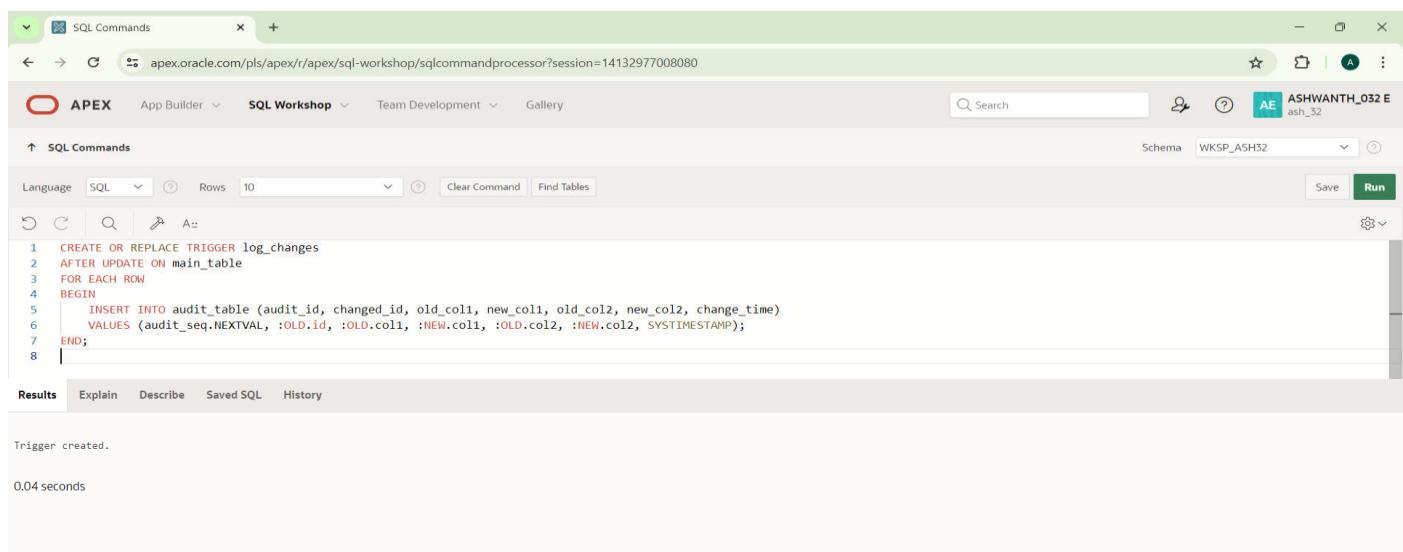
```
PRAGMA EXCEPTION_INIT(threshold_exceeded, -20003);
v_sum NUMBER;
v_threshold NUMBER := 10000; -- Set your threshold here
BEGIN
    SELECT SUM(value_col) INTO v_sum FROM threshold_table;
    v_sum := v_sum + :NEW.value_col;
    IF v_sum > v_threshold THEN
        RAISE threshold_exceeded;
    END IF;
EXCEPTION
    WHEN threshold_exceeded THEN
        RAISE_APPLICATION_ERROR(-20003, 'Threshold exceeded for value_col.');
END;
```

**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 APEX SQL Workshop interface. The top navigation bar includes links for SQL Commands, apex.oracle.com, App Builder, SQL Workshop (which is selected), Team Development, and Gallery. The right side of the header shows the user's name, ASHWANTH\_032 E, and session information, WKSP\_ASH52. The main workspace is titled "SQL Commands". The SQL editor contains the PL/SQL code for the trigger. The code is as follows:

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

Below the code, the "Results" tab is active, showing the output: "Trigger created." and "0.04 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 Developer interface with the SQL tab selected. The code for the trigger is entered in the command window. The output pane at the bottom shows the message "Trigger created." and a execution time of "0.05 seconds".

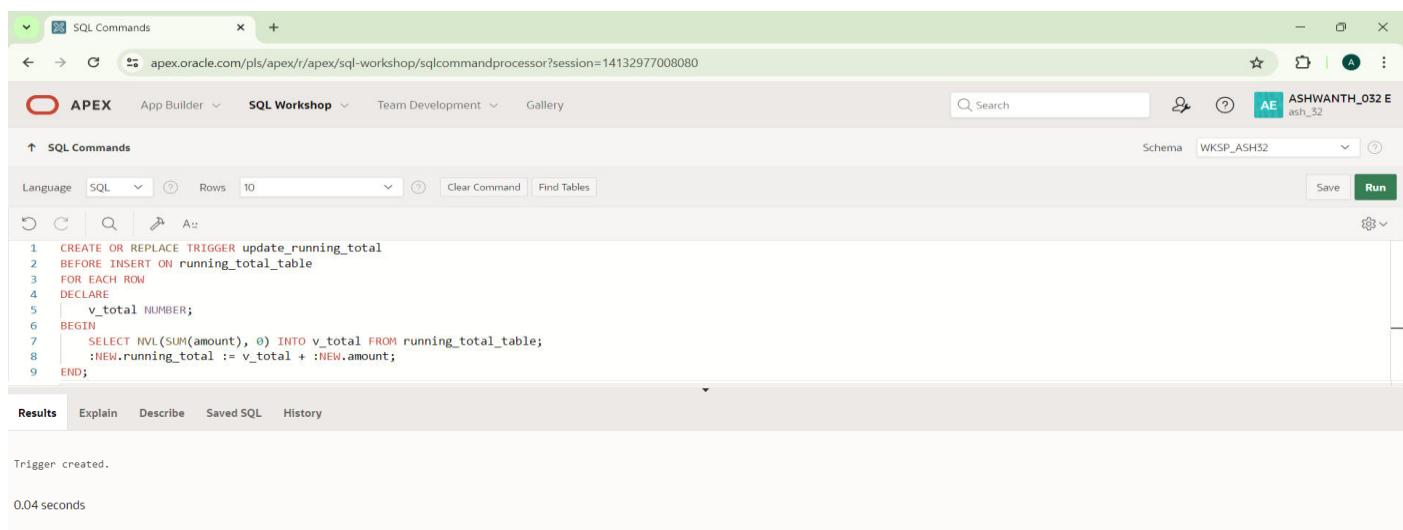
```
1 CREATE OR REPLACE TRIGGER log_user_activity
2 AFTER INSERT OR UPDATE OR DELETE ON activity_table
3 FOR EACH ROW
4 BEGIN
5     IF INSERTING THEN
6         INSERT INTO user_activity_log (log_id, action, table_name, record_id, change_time)
7         VALUES (activity_log_seq.NEXTVAL, 'INSERT', 'activity_table',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

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (which is selected), 'Team Development', and 'Gallery'. The main area is titled 'SQL Commands' and contains the PL/SQL code for the trigger. The code 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;
```

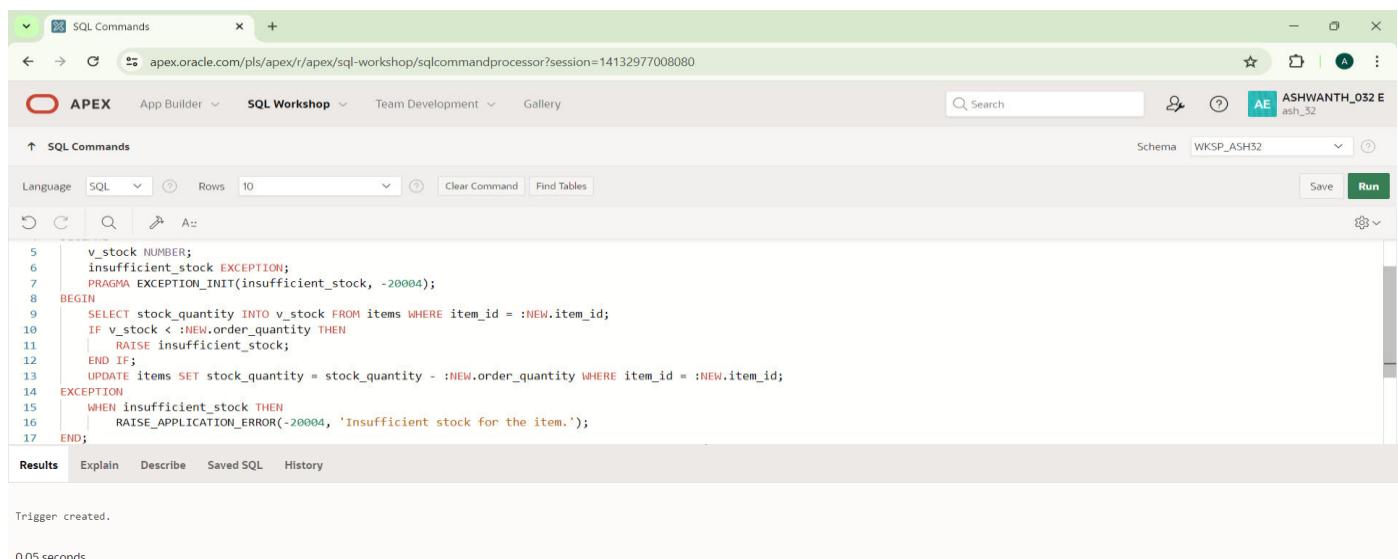
Below the code, the 'Results' tab is active, displaying the message 'Trigger created.' and a execution time of '0.04 seconds'.

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area displays the PL/SQL code for the 'validate\_order' trigger. The code checks if there is sufficient stock for a new order. If not, it raises an exception. It then updates the stock quantity. Finally, it handles the 'insufficient\_stock' exception by raising a specific application error. The code is as follows:

```
5  v_stock NUMBER;
6  insufficient_stock EXCEPTION;
7  PRAGMA EXCEPTION_INIT(insufficient_stock, -20004);
8  BEGIN
9    SELECT stock_quantity INTO v_stock FROM items WHERE item_id = :NEW.item_id;
10   IF v_stock < :NEW.order_quantity THEN
11     RAISE insufficient_stock;
12   END IF;
13   UPDATE items SET stock_quantity = stock_quantity - :NEW.order_quantity WHERE item_id = :NEW.item_id;
14 EXCEPTION
15   WHEN insufficient_stock THEN
16     RAISE_APPLICATION_ERROR(-20004, 'Insufficient stock for the item.');
17 END;
```

At the bottom of the interface, the 'Results' tab is active, showing the message 'Trigger created.' and a execution time of '0.05 seconds'.

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

**RESULT:** Thus the above sql queries has been executed successfully.

# MONGO DB

**EX\_NO: 19**

**DATE:**

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB shell interface. On the left, there is an input field with placeholder text "Enter a title...". Below it are two dropdown menus: one set to "MongoDB" and another with a question mark icon. To the right is a "Run" button with a play icon. The main area contains the MongoDB command and its output. The command is:

```
1: { $or: [ { name: /^Wil/ }, { cuisine: { $nin: ['American', 'Chinese'] } } ] , { restaurant_id: 1, name: 1, borough: 1, cuisine: 1 } } ); Output
```

The output shows the results of the query execution:

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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

**QUERY:**

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

**OUTPUT:**

myCompiler

English Recent

Enter a title...

MongoDB

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

mycompiler\_mongodb>  
mycompiler\_mongodb>

[Execution complete with exit code 0]

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

**QUERY:**

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

**OUTPUT:**

Enter a title...

MongoDB

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

mycompiler\_mongodb>  
mycompiler\_mongodb>

[Execution complete with exit code 0]

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB query editor interface. At the top, there is a search bar labeled "Enter a title..." and a dropdown menu set to "MongoDB". Below the search bar are two buttons: a blue one with a gear icon and a red one with a refresh icon. To the right is a green "Run" button with a play icon. The main area has a light gray background. On the left, a code editor window displays the MongoDB query. On the right, a terminal-like window titled "Output" shows the command "mycompiler\_mongodb> mycompiler\_mongodb>" followed by "[Execution complete with exit code 0]".

```
1{$and : [{"address.coord.1": {$gt : 42}}, {"address.coord.1": {$lte : 52}}]}, {_id:0, restaurant_id:1, name:1, address:1})| Output
```

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB shell interface. On the left, there is a text input field labeled "Enter a title...". Below it are two buttons: one with a green icon and the text "MongoDB", and another with a blue icon. To the right of these buttons is a small "i" button. At the top right, there is a "Ctrl+" button and a green play button icon. In the main area, a query is typed into a text input field:

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

To the right of the input field is a "Output" section. It contains the following text:

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB shell interface. On the left, there is a text input field labeled "Enter a title...". Below it are two buttons: one with a green icon and the text "MongoDB", and another with a blue icon. To the right of these buttons is a small "i" button. At the top right, there is a "Ctrl+" button and a green play button icon. In the main area, a query is typed into a text input field:

```
1{{ $and : [ {"address.coord.1": { $gt : 42 }}, {"address.coord.1": { $lte : 52 }} ]}, { _id:0, restaurant_id:1, name:1, address:1 }}| Output
```

To the right of the input field is a "Output" section. It contains the following text:

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB shell interface. On the left, there is a text input field with placeholder text "Enter a title...". Below it are two buttons: "MongoDB" with a dropdown arrow and a small info icon. To the right is a "Run" button with a play icon. The main area contains a command line with the following text:  
1 db.restaurants.find( {}, { \_id: 0 }).sort({ cuisine: 1, borough: -1 })  
The output window on the right shows the results of the command:  
mycompiler\_mongodb>  
mycompiler\_mongodb>  
[Execution complete with exit code 0]

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB shell interface. On the left, there is a text input field with placeholder text "Enter a title...". Below it are two buttons: "MongoDB" with a dropdown arrow and a small info icon. To the right is a "Run" button with a play icon. The main area contains a command line with the following text:  
1 db.restaurants.find( { "address.street": { \$exists: true, \$ne: "" } } )  
The output window on the right shows the results of the command:  
mycompiler\_mongodb>  
mycompiler\_mongodb>  
[Execution complete with exit code 0]

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows a MongoDB shell interface. On the left, there is a search bar labeled 'Enter a title...' and a dropdown menu set to 'MongoDB'. Below the search bar is a code input field containing the query: 'db.restaurants.find({ "address.coord": { \$elemMatch: { \$type: "double" } } })'. To the right of the code input is a green 'Run' button. To the right of the code input is an 'Output' panel. The output panel displays the command 'mycompiler\_mongodb>' followed by 'mycompiler\_mongodb>' on the next line. At the bottom of the output panel, it says '[Execution complete with exit code 0]'. The entire interface is enclosed in a dark border.

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

**QUERY:**

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

**OUTPUT:**

Enter a title...

MongoDB Run

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

Output

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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

**QUERY:**

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

**OUTPUT:**

Enter a title...

MongoDB Run

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

Output

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB shell interface. On the left, there is a text input field with placeholder text 'Enter a title...'. Below it are two buttons: 'MongoDB' with a dropdown arrow and a small info icon. To the right is a green 'Run' button with a play icon. The main area contains a code editor with the following content:

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

Below the code editor, there are three lines of numbers: '1', '2', and '3'. To the right, there is a 'Output' panel with the following text:

```
mycompiler_mongodb>
mycompiler_mongodb>
```

At the bottom of the output panel, there is a message: '[Execution complete with exit code 0]

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:**

Enter a title...

MongoDB ▾



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

2

3

#### Output

```
mycompiler_mongodb>
mycompiler_mongodb>
```

[Execution complete with exit code 0]

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

**QUERY:**

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

**OUTPUT:**

Enter a title...

MongoDB ▾



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

2

3

#### Output

```
mycompiler_mongodb>
mycompiler_mongodb>
```

[Execution complete with exit code 0]

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB shell interface. At the top, there is a search bar labeled "Enter a title...". Below it, a dropdown menu says "MongoDB" with a "▼" icon, and a small "i" icon. On the right, there is a green "Run" button with a play icon. The main area contains the following text:

```
1.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }] }) Output
2
3
```

On the right side of the interface, there is a terminal window with the following output:

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

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

**QUERY:**

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

**OUTPUT:**

```
Enter a title...
MongoDB ▾ ⓘ Run
1{ "score": { $lt: 5 } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $ne: "American" }}] Output
mycompiler_mongodb>
mycompiler_mongodb>
[Execution complete with exit code 0]
```

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

**QUERY:**

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

**OUTPUT:**

```
Enter a title...
MongoDB ▾ ⓘ Run
1{ $lt: 5 } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $nin: ["American", "Chinese"] })] Output
mycompiler_mongodb>
mycompiler_mongodb>
[Execution complete with exit code 0]
```

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows a MongoDB shell interface. On the left, there is a search bar labeled "Enter a title..." and a dropdown menu set to "MongoDB". Below the search bar is a toolbar with a green play button icon and a red "Run" button. In the main area, a code block contains the following command:

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

To the right of the code block is a "Output" section. It displays the command prompt "mycompiler\_mongodb>" followed by the response "mycompiler\_mongodb>". At the bottom of the output section, it says "[Execution complete with exit code 0]".

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

**QUERY:**

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

**OUTPUT:**

Enter a title...

MongoDB ▾



Run

```
1 $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], "borough": "Manhattan" }]} Output
```

```
mycompiler_mongodb>
```

```
mycompiler_mongodb>
```

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

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:**

Enter a title...

MongoDB ▾



Run

```
1.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }] }]} Output
```

```
mycompiler_mongodb>
```

```
mycompiler_mongodb>
```

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

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

grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American.

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB query editor interface. At the top, there is a search bar labeled "Enter a title...". Below it, a dropdown menu is set to "MongoDB" and a "Run" button is visible. The main area contains the MongoDB query:1 'A", "grades.score": 6 }], \$or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { \$ne: "American" } })

Below the query, the output window shows the command prompt "mycompiler\_mongodb>" followed by the command "mycompiler\_mongodb>". The message "[Execution complete with exit code 0]" is displayed at the bottom right of the output window.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB query editor interface. At the top, there is a search bar labeled "Enter a title...". Below it, a dropdown menu is set to "MongoDB" and a "Run" button is visible. The main area contains the MongoDB query:1 "core": 6 }], \$or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { \$nin: ["American", "Chinese"] } })

Below the query, the output window shows the command prompt "mycompiler\_mongodb>" followed by the command "mycompiler\_mongodb>". The message "[Execution complete with exit code 0]" is displayed at the bottom right of the output window.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB shell interface. On the left, there is a text input field with placeholder text 'Enter a title...'. Below it are two buttons: one with a green icon labeled 'MongoDB' and another with a blue icon. To the right are two buttons: a teal 'Run' button and a blue 'Save' button. The main area contains a code editor with the following content:

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

To the right of the code editor is a light gray panel titled 'Output' containing the following text:

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

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

**RESULT:**

# MONGO DB

**EX\_NO: 20**

**DATE:**

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB shell interface. On the left, there is an input field labeled "Enter a title...". Below it, a dropdown menu says "MongoDB" with a "Run" button next to it. The code input area contains the command: `1 db.movies.find({ year: 1893 })`. To the right, under the heading "Output", the results are shown: `mycompiler_mongodb>`, `mycompiler_mongodb>`, and "[Execution complete with exit code 0]".

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB shell interface. On the left, there is an input field labeled "Enter a title...". Below it, a dropdown menu says "MongoDB" with a "Run" button next to it. The code input area contains the command: `1 db.movies.find({ runtime: { $gt: 120 } })`. To the right, under the heading "Output", the results are shown: `mycompiler_mongodb>`, `mycompiler_mongodb>`, and "[Execution complete with exit code 0]".

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB shell interface. On the left, there is a search bar labeled "Enter a title...". Below it are two buttons: "MongoDB" with a dropdown arrow and a small info icon. To the right is a green "Run" button with a play icon and the text "Run". In the main area, a code input field contains the command: "1 db.movies.find({ genres: 'short' })". To the right, under the heading "Output", the results are displayed: "mycompiler\_mongodb>" followed by a blank line, another "mycompiler\_mongodb>" line, and "[Execution complete with exit code 0]" at the bottom.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB shell interface. On the left, there is a search bar labeled "Enter a title...". Below it are two buttons: "MongoDB" with a dropdown arrow and a small info icon. To the right is a green "Run" button with a play icon and the text "Run". In the main area, a code input field contains the command: "1 db.movies.find({ directors: 'William K.L. Dickson' })". To the right, under the heading "Output", the results are displayed: "mycompiler\_mongodb>" followed by a blank line, another "mycompiler\_mongodb>" line, and "[Execution complete with exit code 0]" at the bottom.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB shell interface. On the left, there is a text input field with placeholder text "Enter a title...". Below it are two buttons: one with a green leaf icon labeled "MongoDB" and another with an info icon. To the right is a "Run" button with a play icon. The main area contains the command `1 db.movies.find({ countries: 'USA' })`. To the right, under the heading "Output", is the response from the database: `mycompiler_mongodb>`, `mycompiler_mongodb>`, and [Execution complete with exit code 0].

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB shell interface. On the left, there is a text input field with placeholder text "Enter a title...". Below it are two buttons: one with a green leaf icon labeled "MongoDB" and another with an info icon. To the right is a "Run" button with a play icon. The main area contains the command `1 db.movies.find({ rated: 'UNRATED' })`. To the right, under the heading "Output", is the response from the database: `mycompiler_mongodb>`, `mycompiler_mongodb>`, and [Execution complete with exit code 0].

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

**QUERY:**

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

**OUTPUT:**

Enter a title...

MongoDB ▾

Ctrl+Enter

Run

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

Output

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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

**QUERY:**

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

**OUTPUT:**

Enter a title...

MongoDB ▾

Run

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

Output

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB shell interface. On the left, there is a search bar labeled "Enter a title...". Below it are two buttons: "MongoDB" with a dropdown arrow and a small info icon. To the right is a green "Run" button with a play icon. In the main area, a command is entered: `db.movies.find({ 'tomatoes.viewer.rating': { $gt: 4 } })`. The output window on the right shows the results of the query:

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

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB shell interface. On the left, there is a search bar labeled "Enter a title...". Below it are two buttons: "MongoDB" with a dropdown arrow and a small info icon. To the right is a green "Run" button with a play icon. In the main area, a command is entered: `db.movies.find({ 'awards.wins': { $gt: 0 } })`. The output window on the right shows the results of the query:

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

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

**least one nomination.**

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB shell interface. On the left, there is a search bar labeled "Enter a title..." and a dropdown menu set to "MongoDB". Below the search bar are two buttons: a green one with a gear icon and a blue one with a question mark icon. To the right of these buttons is a teal "Run" button. The main area contains a command line and its output. The command is:

```
1, languages: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 })
```

The output shows the results of the query:

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

At the bottom right of the interface, there is a small advertisement for Figma.

**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:**

Enter a title...

MongoDB ▾



Run

```
1, languages: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 })|| Output
```

```
mycompiler_mongodb>
mycompiler_mongodb>
```

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

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

**QUERY:**

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

**OUTPUT:**

Enter a title...

Ctrl+Enter

MongoDB ▾



Run

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

```
mycompiler_mongodb>
mycompiler_mongodb>
```

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

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

**QUERY:**

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

### OUTPUT:



The screenshot shows a MongoDB shell interface. On the left, there is a search bar with placeholder text "Enter a title...". Below it are two buttons: "MongoDB" with a dropdown arrow and a small info icon. To the right is a "Run" button with a play icon. In the main area, a query is typed into the command line:

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

To the right of the command line is a "Output" panel. It displays the results of the query execution:

```
mycompiler_mongodb>
mycompiler_mongodb>
```

[Execution complete with exit code 0]

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

### RESULT: