

EX.NO:1

REG.NO:220701024

DATE:

## CREATING AND MANAGING TABLES

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 varchar(25));

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

```
CREATE TABLE DEPT(id number(7),Name varchar(25));
```

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

Table created.  
0.01 seconds

In the bottom right corner, the footer indicates:

Application Express 4.0.2.00.09  
Workspace: AMALLESH24 User: AMALLESH24  
Language: en | Copyright © 1999, 2010, Oracle. All rights reserved.

### OUTPUT:

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

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

### QUERY:

```
create table emp1(id number(7), last_name varchar(25), first_name varchar(25), dept_id number(7));
```

### OUTPUT:

The screenshot shows the Oracle Application Express interface. In the top navigation bar, 'SQL Workshop' is selected. The main area contains the SQL command for creating the 'emp1' table:

```
CREATE TABLE emp1(id number(7),Last Name varchar(25),First Name varchar(25),Dept Id number(7));
```

Below the command, the results show:

Table created.  
0.00 seconds

At the bottom, the footer indicates the application version and copyright information:

Application Express 4.0.2.00.09  
Language: en | Copyright © 1999, 2010, Oracle. All rights reserved.

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

### QUERY:

```
Alter table emp1 modify(last_name varchar(50);
```

## OUTPUT:

ORACLE Application Express

Welcome AMALLESH24 ( Logout )

Home Application Builder SQL Workshop Team Development Administration

Schema AMALLESH24 Help

Home > SQL Workshop > SQL Commands

ALTER TABLE emp1 MODIFY(last\_name varchar(50));

Autocommit Rows 10 Save Run

Results Explain Describe Saved SQL History

Table altered.

0.02 seconds

Application Express 4.0.2.00.09

Workspace: AMALLESH24 User: AMALLESH24 Language: en | Copyright © 1999, 2010, Oracle. All rights reserved.

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 employees(id number(10), first\_name varchar(20), last\_name varchar(20), salary number(8,2), dept\_id number(4));

## OUTPUT

ORACLE Application Express

Welcome AMALLESH24 ( Logout )

Home Application Builder ▾ SQL Workshop ▾ Team Development ▾ Administration ▾

Home > SQL Workshop > SQL Commands

Schema AMALLESH24 ▾ Help

Autocommit Rows 10

```
CREATE TABLE Employees(Id number(10),first_name varchar(20),last_name varchar(20),salary number(20),Dept_Id number(4));
```

Results Explain Describe Saved SQL History

Table created.

0.00 seconds

Application Express 4.0.2.00.09

Workspace: AMALLESH24 User: AMALLESH24 Language: en | Copyright © 1999, 2010, Oracle. All rights reserved.

5.Drop the EMP table.

### QUERY:

```
Drop table emp1;
```

### OUTPUT:

Home Application Builder ▾ SQL Workshop ▾ Team Development ▾ Administration ▾

Home > SQL Workshop > SQL Commands

Schema AMALLESH24 Help

Autocommit Rows 10 Save Run

```
DROP TABLE emp;
```

Results Explain Describe Saved SQL History

Table dropped.

0.02 seconds

Application Express 4.0.2.00.09

Workspace: AMALLESH24 User: AMALLESH24 Language: en | Copyright © 1999, 2010, Oracle. All rights reserved.

6.Rename the EMPLOYEES2 table as EMP.

### QUERY:

ALTER TABLE employee rename to emp;

### OUTPUT:

oracle apex - Search × SQL Commands × +

https://apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=5478616924335

Import favorites NOL-UNIT IV-com... Lightbox

APEX App Builder ▾ SQL Workshop ▾ Team Development ▾ Gallery

Search Schema WKSP\_AMALLESH

Language SQL Rows 10 Clear Command Find Tables Save Run

```
1 alter table employee rename to emp
```

Results Explain Describe Saved SQL History

Table altered.

0.07 seconds

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

Oracle APEX 25.2.4

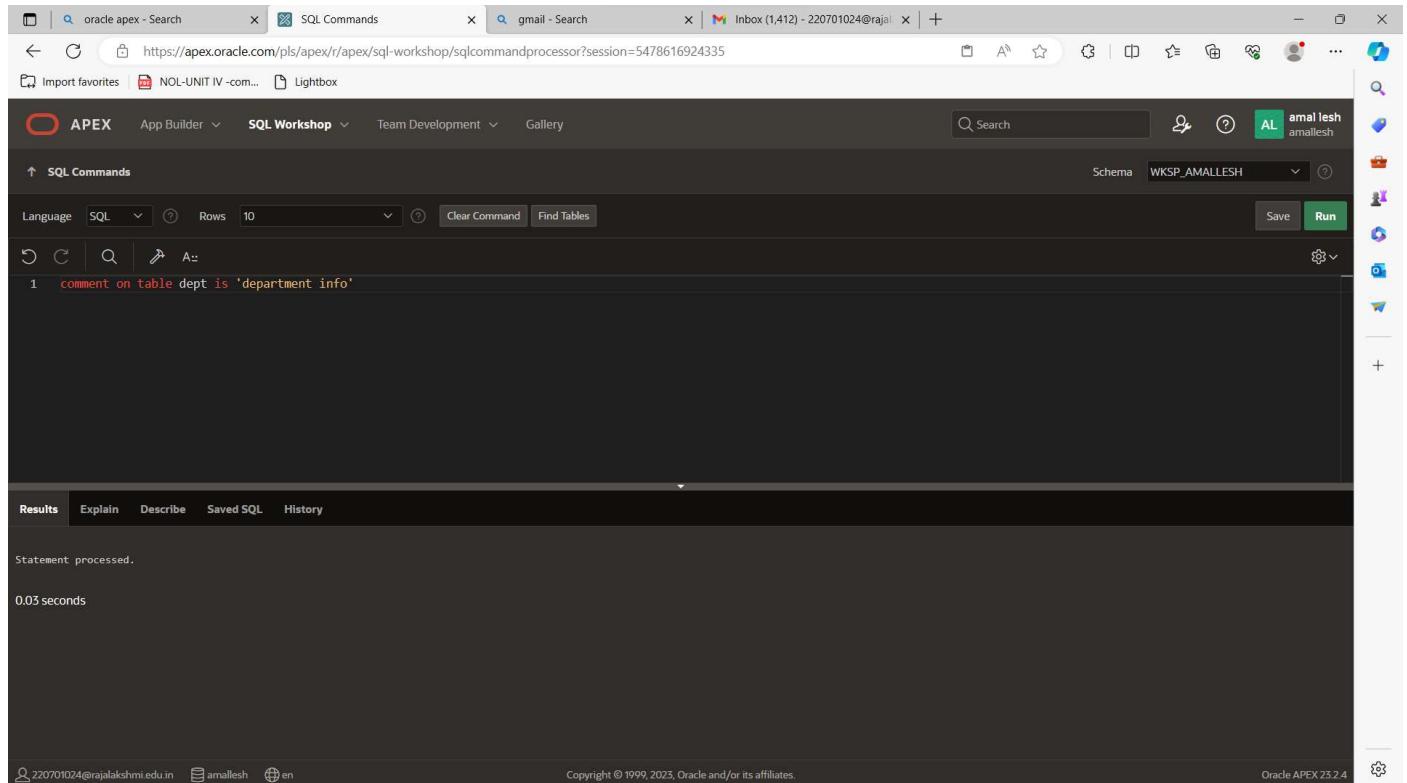
7.Add a comment on DEPT and EMP tables. Confirm the modification by describing the table.

### QUERY:

Comment on table emp as ‘employee info’

Comment on table dept as ‘department info’;

### OUTPUT:



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

```
1 comment on table dept is 'department info'
```

Below the command, the results show:

Statement processed.  
0.03 seconds

The bottom right corner of the interface displays the text "Oracle APEX 25.2.4".

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 browser window with three tabs: "oracle apex - Search", "SQL Commands", and "gmail - Search". The "SQL Commands" tab is active, displaying the URL <https://apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=5478616924335>. The page title is "APEX". The main content area is titled "SQL Commands" and contains a code editor with the following SQL command:

```
1 alter table emp drop column first_name;
```

Below the code editor, the results pane shows the output of the query:

Table altered.  
0.74 seconds

At the bottom of the page, there is footer information: "Copyright © 1999, 2023, Oracle and/or its affiliates.", "Oracle APEX 25.2.4", and a user profile icon.

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

DATE:

# MANIPULATING DATA

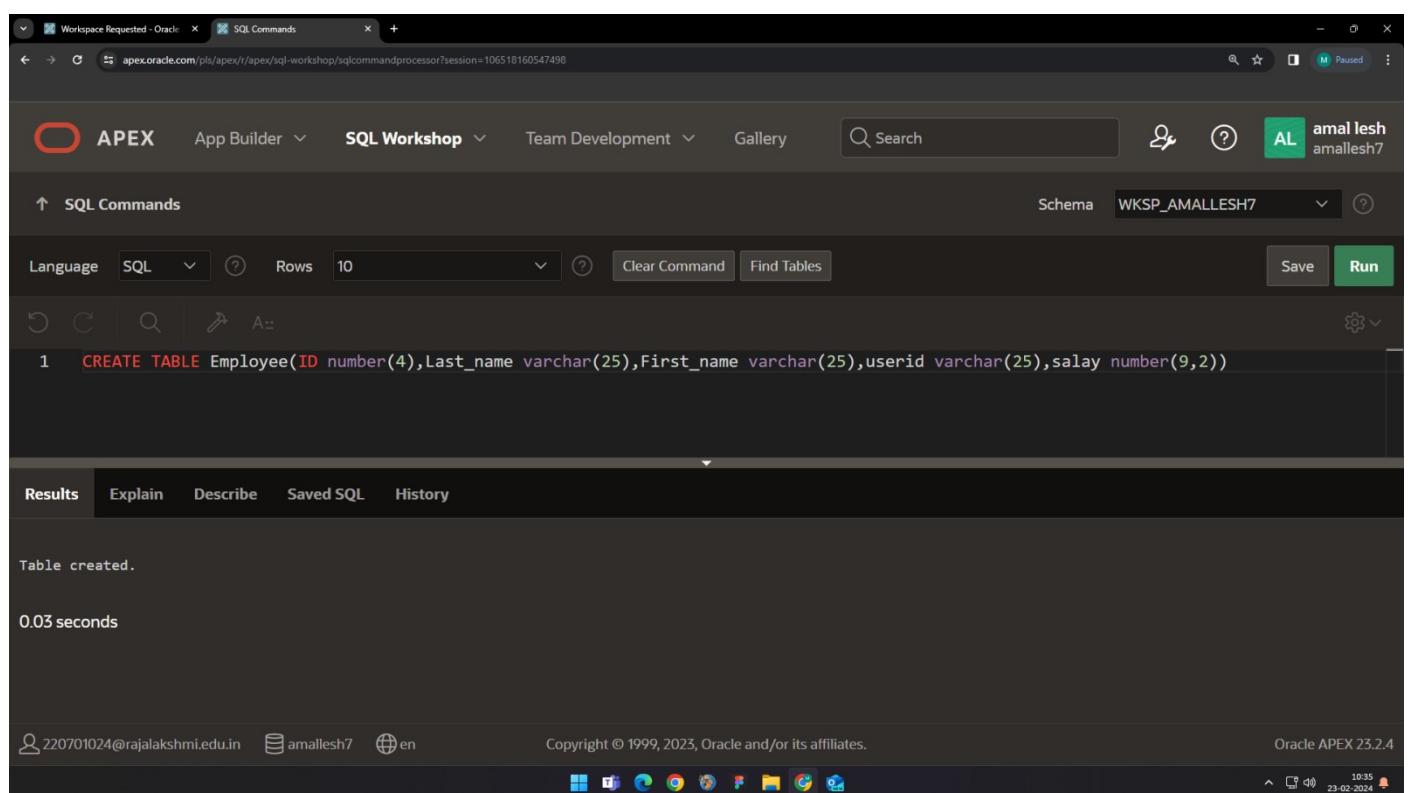
1.Create MY\_EMPLOYEE table with the following structure

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

## QUERY:

```
Create table Employee(id number(4), last_name varchar(25), userid varchar(25), salary number(9,2));
```

## OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (which is selected), 'Team Development', 'Gallery', and a search bar. The user is logged in as 'amal lesh amallesh7'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_AMALLESH7'. Below the title are buttons for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', 'Find Tables', 'Save', and 'Run'. The SQL command entered is:

```
1 CREATE TABLE Employee(ID number(4),Last_name varchar(25),First_name varchar(25),userid varchar(25),salay number(9,2))
```

The results section shows the output of the command:

```
Table created.

0.03 seconds
```

At the bottom, the footer displays the user's email ('220701024@rajalakshmi.edu.in'), session ID ('amallesh7'), language ('en'), copyright information ('Copyright © 1999, 2023, Oracle and/or its affiliates.'), and the version ('Oracle APEX 23.2.4').

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

ID	Last_name	First_name	Userid	salary
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860

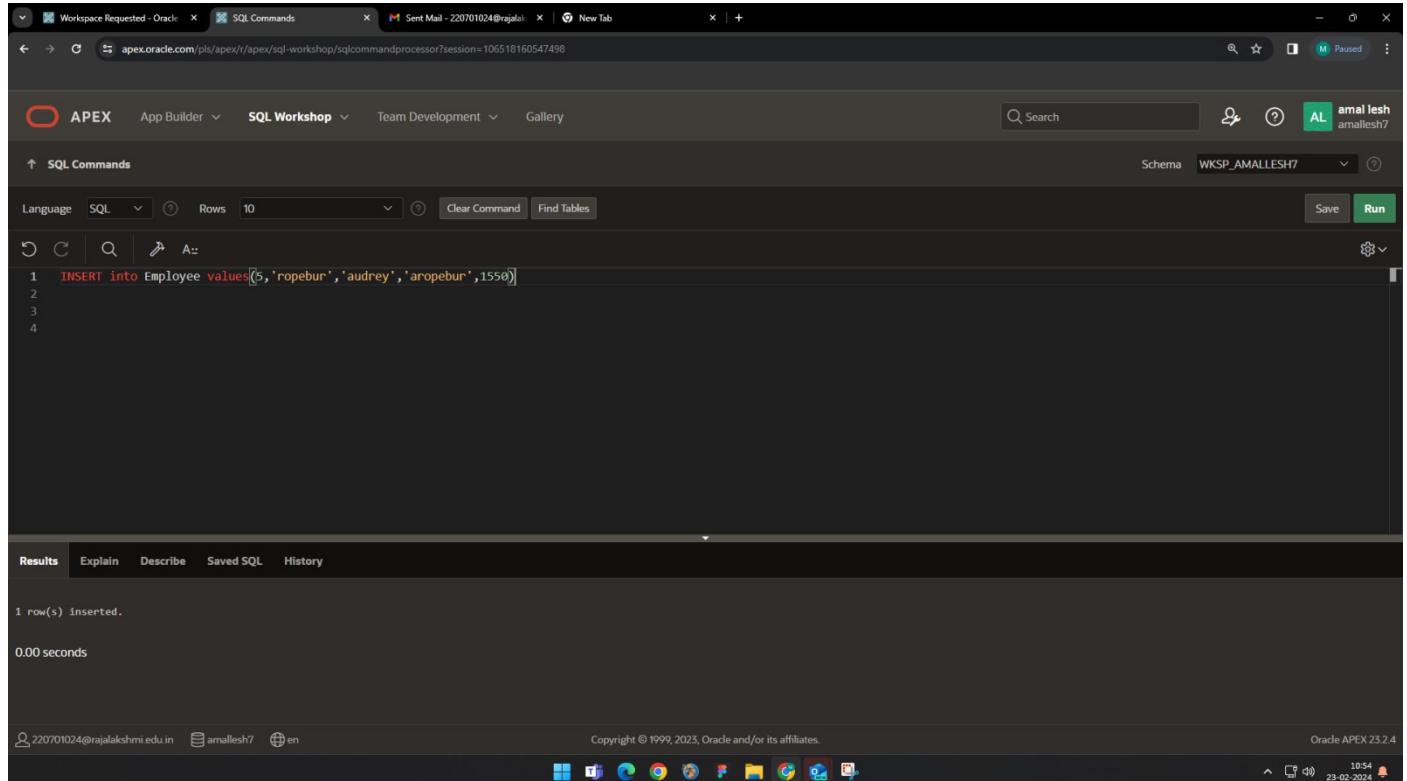
3	Biri	Ben	bbiri	1100
4	Newman	Chad	Cnewman	750
5	Ropebur	Audrey	aropebur	1550

### QUERY:

Insert into Employee values(1,’Patel’,Ralph’, ‘rpatel’, 895);

Insert into Employee values(2,’Dancs’,’Betty’, ‘bdancs’, 860);

### OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, a single-line insert statement is entered:

```
1 INSERT into Employee values(5,'ropebur','audrey','aropebur',1550)
```

The 'Run' button is highlighted in green at the top right of the input area. Below the command, the results section displays:

1 row(s) inserted.  
0.00 seconds

At the bottom, the footer includes the user information 220701024@rajalakshmi.edu.in, amalles7, and the copyright notice Copyright © 1999, 2025, Oracle and/or its affiliates.

3.Display the table with values.

### QUERY:

Select \* from Employee;

### OUTPUT:

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

```
1 SELECT * FROM Employee;
```

The results section displays the following data:

	first_name	last_name	email	hire_date	job_id	salary
1	patel	ralph	rpatel	1986-01-15	10	895
5	ropebur	audrey	aropetur	1986-01-15	20	1550
4	newman	chad	cnewman	1986-01-15	30	750
2	dances	betty	bdances	1986-01-15	40	860

5 rows returned in 0.01 seconds

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

**QUERY:**

Insert into Employee values(6,’amal’, ‘dane’, ‘fab’, 300);

**OUTPUT:**

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

```
1 insert into employee values(6,'amal','dane','fab',300)
```

The results section displays the output:

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

0.01 seconds

5. Make the data additions permanent.

**QUERY:**

Select \* from employee;

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The user has run a query to select all columns from the employee table. The results are displayed in a grid format with columns: ID, LAST\_NAME, FIRST\_NAME, USERID, and SALAY. The data includes rows for employees with IDs 3, 1, 5, 6, 7, 4, and 2, each with their respective last name, first name, user ID, and salary.

ID	LAST_NAME	FIRST_NAME	USERID	SALAY
3	Drexler	ben	bbiri	1100
1	patel	ralph	rpatel	895
5	ropebur	audrey	aropebur	1550
6	amal	dane	fab	300
7	santa	demon	eron	500
4	newman	chad	cnewman	750
2	dancs	betty	bdancs	860

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

Copyright © 1999, 2025, Oracle and/or its affiliates. Oracle APEX 25.2.4

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

**QUERY:**

```
update Employee set last_name='Drexler' where userid='bbiri';
```

## OUTPUT:

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

```
1 UPDATE Employee set Last_Name='Drexler' where userid='bbiri';
2
3
4
```

Below the code, the results pane shows:

1 row(s) updated.  
0.00 seconds

At the bottom, the status bar indicates the user's email (220701024@rajalakshmi.edu.in), session ID (amallesh7), and the system time (Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4).

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

Update Employee set salary=1000 where salary<900;

**OUTPUT:**

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

```
1 update employee set salary=1000 where salary<900;
2 select*from employee;
3
4
5
```

Below the code, the results pane shows a table of employee data:

ID	LAST_NAME	FIRST_NAME	USERID	SALAY
3	Drexler	ben	bbiri	1100
1	patel	ralph	rpatel	1000
5	ropebur	audrey	aropetur	1550
6	amal	dane	fab	1000
7	santa	demon	eron	1000
4	newman	chad	cnewman	1000
2	dancs	betty	bdancs	1000

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

At the bottom, the status bar indicates the user's email (220701024@rajalakshmi.edu.in), session ID (amallesh7), and the system time (Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4).

8.Delete Betty dancs from MY\_EMPLOYEE table.  
**QUERY:**

Delete from Employee where first\_name='betty';

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, a query is entered:

```

1 delete from employee where first_name='betty';
2 select *from employee;
3
4

```

The results section displays the following table:

ID	LAST_NAME	FIRST_NAME	USERID	SALAY
3	Drexler	ben	bbiri	1100
1	patel	ralph	rpatel	1000
5	ropebur	audrey	arophebur	1550
6	amal	dane	fab	1000
7	santa	demon	eron	1000
4	newman	chad	cnewman	1000

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

At the bottom, the status bar shows: 220701024@rajalakshmi.edu.in amallesh7 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4 11:31 23-02-2024

9.Empty the fourth row of the emp table.

### QUERY:

Delete from Employee where id=4;

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, a query is entered:

```

1 delete from employee where id=4;
2 select *from employee;
3
4

```

The results section displays the following table:

ID	LAST_NAME	FIRST_NAME	USERID	SALAY
3	Drexler	ben	bbiri	1100
1	patel	ralph	rpatel	1000
5	ropebur	audrey	arophebur	1550
6	amal	dane	fab	1000
7	santa	demon	eron	1000

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

At the bottom, the status bar shows: 220701024@rajalakshmi.edu.in amallesh7 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4 11:34 23-02-2024

Evaluation Procedure	Marks awarded
----------------------	---------------

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

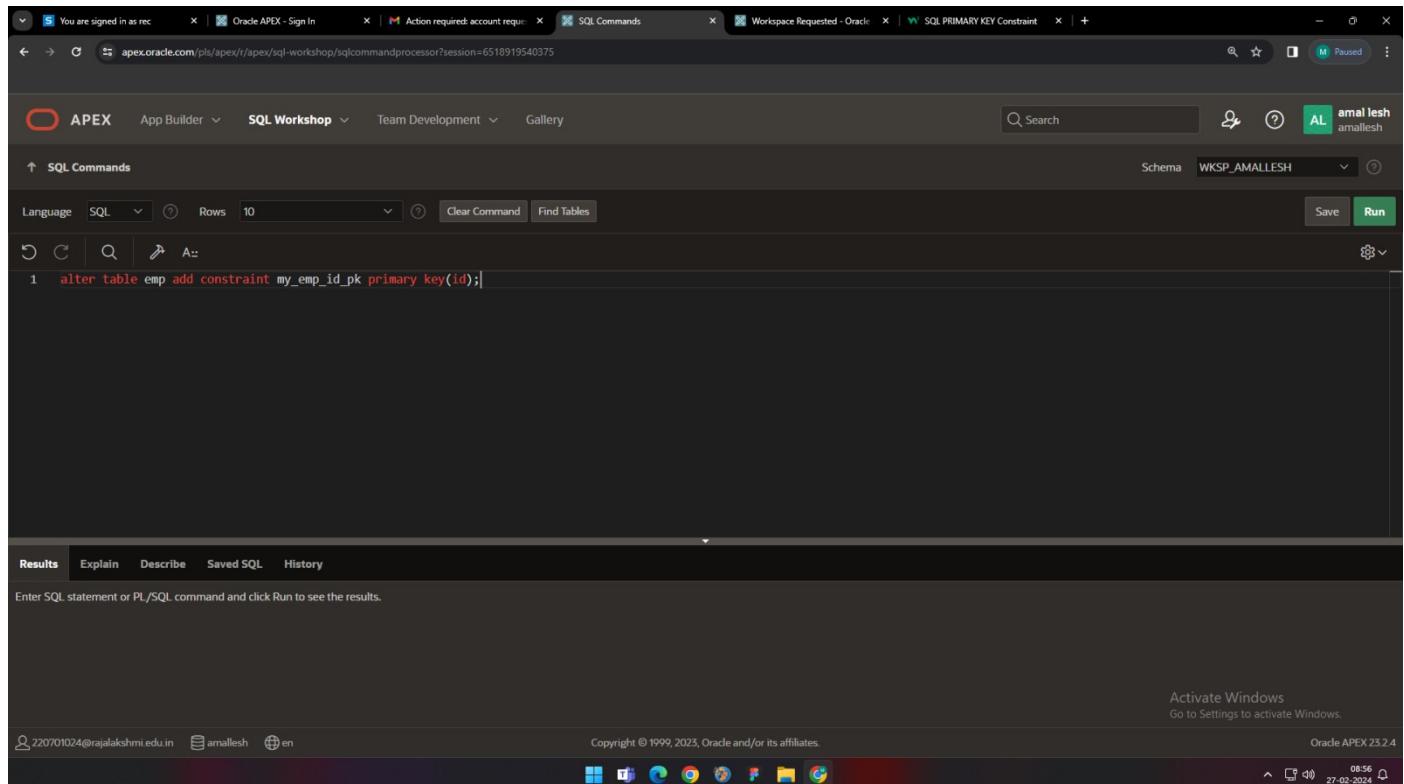
## INCLUDING CONSTRAINTS

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

### QUERY:

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

### OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, there are tabs for 'APEX', 'App Builder', 'SQL Workshop' (which is selected), 'Team Development', and 'Gallery'. On the right side, there's a user profile for 'amal lesh' (AL). The main area is titled 'SQL Commands'. Below it, there are buttons for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. The SQL command entered is: `1 alter table emp add constraint my_emp_id_pk primary key(id);`. At the bottom, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. A note says 'Enter SQL statement or PL/SQL command and click Run to see the results.' The status bar at the bottom shows the user's email (220701024@rajalakshmi.edu.in), the session ID (amallesh), the language (en), copyright information (Copyright © 1999, 2023, Oracle and/or its affiliates.), the Oracle APEX version (Oracle APEX 23.2.4), and the system date and time (27-02-2024 08:56).

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

### QUERY:

```
Alter table dept add constraint my_dept_id_pk primary key(dept_id);
```

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, there are tabs for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right side, it shows the user is signed in as amallesh (session 6518919540375). The main area is titled 'SQL Commands' with a sub-section 'constraint my\_dept\_id\_pk primary key(dept\_id);'. Below the code editor, the 'Results' tab is selected, showing the output: 'Table altered.' and '0.06 seconds'. The bottom status bar includes the user's email (220701024@rajalakshmi.edu.in), the schema (WKSP\_AMALLESH), and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.'

```
1 alter table dept add constraint my_dept_id_pk primary key(dept_id);
2
```

Table altered.  
0.06 seconds

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

#### QUERY:

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

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, there are tabs for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right side, it shows the user is signed in as amallesh (session 104859993733038). The main area is titled 'SQL Commands' with a sub-section 'constraint my\_emp\_dept\_id\_fk foreign key (dept\_id) references emp(id);'. Below the code editor, the 'Results' tab is selected, showing the output: 'Table altered.' and '0.06 seconds'. The bottom status bar includes the user's email (220701024@rajalakshmi.edu.in), the schema (WKSP\_AMALLESH), and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.'

```
1 alter table emp add constraint my_emp_dept_id_fk foreign key (dept_id) references emp(id);
2
```

Table altered.  
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:

```
Alter table emp add (commission number(2,2), constraint emp_commission(commission>0));
```

## OUTPUT:



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

```
1 alter table emp add (commission
```

The Results pane displays the output of the command:

```
Table altered.
```

Execution time is shown as 0.06 seconds.

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

# WRITING BASIC SQL SELECT STATEMENTS

1. The following statement executes successfully.

## Identify the Errors

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

FROM employees;

**QUERY:**

```
Select emp_id, last_name, "salary*"12 "ANNUAL_SALARY" from employee;
```

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, a query is entered:

```
1 select id, last_name, salary*12 from emp;  
2
```

The results tab displays the output:

ID	LAST_NAME	SALARY*12
2	dans	10320
1	patel	10740
3	berri	15200
4	newman	9000

4 rows returned in 0.03 seconds

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

**QUERY:**

```
select * from employee;
```

## OUTPUT:

```

1 select*from dept;
2

```

DEPT_ID	DEPT_NAME	MANAGER_ID	LOCATION_ID
1	frank	29	430
4	newman	34	750
3	hoptoit	96	200

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

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

#### QUERY:

Select emp\_id as emp\_number, last\_name, job\_id, hire\_date from employee;

#### OUTPUT:

```

1 Select emp_id as emp_number, last_name, job_id, hire_date from employee;
2

```

EMP_NUMBER	LAST_NAME	JOB_ID	HIRE_DATE
101	bbiri	24	09/10/2018

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

4. Provide an alias STARTDATE for the hire date.

**QUERY:**

Select hire\_date as Startdate from employee;

**OUTPUT:**

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

```
1 Select hire_date as Startdate from employee;
```

The results section shows the output:

STARTDATE
09/10/2018

1 rows returned in 0.01 seconds

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

**QUERY:**

Select distinct job\_id from employee;

**OUTPUT:**

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

```
1 Select distinct job_id from employee;
```

The results section shows the output:

JOB_ID
24

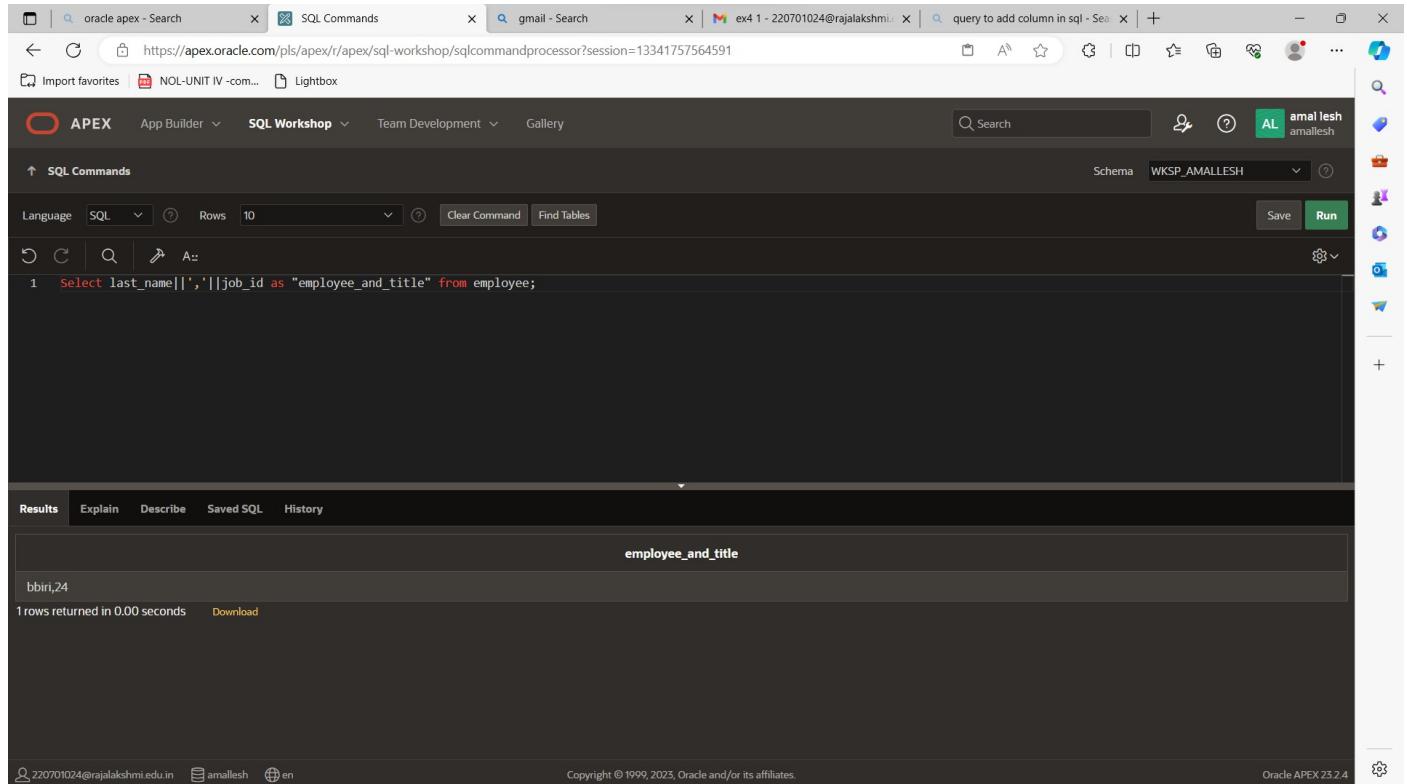
1 rows returned in 0.00 seconds

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

**QUERY:**

```
Select last_name||','||job_id as "employee_and_title" from employee;
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The query `Select last\_name||','||job\_id as "employee\_and\_title" from employee;` is entered in the SQL command editor. The results show one row: 'bbiri,24'. The schema is set to 'WKSP\_AMALLESH'.

employee_and_title
bbiri,24

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

**QUERY:**

```
select emp_id||','||first_name||','||last_name||','||email||','||emp_number||','||hire_date||','||job_id||'  
'||salary||','||manager_id||','||dept_id as "the_output" from employee;
```

**OUTPUT:**

oracle apex - Search X SQL Commands X gmail - Search X ex4 1 - 220701024@rajalakshmi X query to add column in sql - Se X +

Import favorites NOL-UNIT IV -com... Lightbox

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language SQL Rows 10 Clear Command Find Tables

Schema WKSP\_AMALLESH Save Run

```
1 emp_id||','||first_name||','||last_name||','||email||','||emp_number||','||hire_date||','||job_id||','||salary||','||manager_id||','||'|dept_id as "the_output" from employee;
```

Results Explain Describe Saved SQL History

the\_output

101,dane,bbir,ABC@gmail.com,8220489702,09/10/2018,24,30000,35,,45

1 rows returned in 0.01 seconds Download

220701024@rajalakshmi.edu.in amallesh en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

Evaluation Procedure	Marks awarded
Practice Evaluation (5)	
Viva(5)	
Total (10)	
Faculty Signature	

# 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:

OUTPUT:

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

```
1 select last_name,salary from emp where salary<12000;
```

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

LAST_NAME	SALARY
dancs	860
patel	895
berri	1100
newman	750

4 rows returned in 0.01 seconds

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

QUERY:

OUTPUT:

The screenshot shows the Oracle APEX interface with the SQL Workshop module selected. A query is being run against the emp table:

```
1 select last_name,dept_number from emp where id=176;
```

The results show one row:

LAST_NAME	DEPT_NUMBER
berri	2

1 rows returned in 0.01 seconds

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

### QUERY:

### OUTPUT:

The screenshot shows the Oracle APEX interface with the SQL Workshop module selected. A query is being run against the emp table to find employees whose salary is not between 5000 and 12000:

```
1 select last_name,salary from emp where salary not between 5000 and 12000;
```

The results show two rows:

LAST_NAME	SALARY
patel	3500
newman	750

2 rows returned in 0.01 seconds

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

#### QUERY:

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes tabs for 'oracle apex - Search', 'SQL Commands', 'gmail - Search', and 'Sent Mail - 220701024@rajalakshmi.edu.in'. The main workspace is titled 'APEX' and contains a 'SQL Workshop' tab. The SQL command window displays the following query:

```
1 select last_name,salary,commission from emp where commission>0;
```

The results tab shows the output of the query:

LAST_NAME	SALARY	COMMISSION
Jatel	3500	2

Below the table, it says '1 rows returned in 0.01 seconds'. The bottom of the page includes links for 'Download' and 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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:

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, there are tabs for 'azure portal - Search', 'oracle apex - Search', 'SQL Commands', 'gmail - Search', and 'Sent Mail - 220701024@rajalakshmi.edu.in'. The 'SQL Commands' tab is active. The main area shows a SQL command window with the following code:

```
1 select last_name,dept_id from emp where dept_id in (20,50)
2 order by last_name;
```

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

LAST_NAME	DEPT_ID
newman	50
patel	20

Text at the bottom of the results pane indicates "2 rows returned in 0.00 seconds". The footer of the page includes links for "Download", "Copyright © 1999, 2023, Oracle and/or its affiliates.", and "Oracle APEX 23.2.4".

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:

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, there are tabs for 'azure portal - Search', 'oracle apex - Search', 'SQL Commands', 'gmail - Search', and 'Sent Mail - 220701024@rajalakshmi.edu.in'. The 'SQL Commands' tab is active. The main area shows a SQL command window with the following code:

```
1 select last_name as Employee,salary as Monthly_salary from emp where salary between 5000 and 12000 AND dept_id in (20,50)
2 order by last_name;
```

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

EMPLOYEE	MONTHLY_SALARY
berri	11000
dalns	8000

Text at the bottom of the results pane indicates "2 rows returned in 0.00 seconds". The footer of the page includes links for "Download", "Copyright © 1999, 2023, Oracle and/or its affiliates.", and "Oracle APEX 23.2.4".

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

### QUERY:

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, a query is entered:

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

The results show one row:

LAST_NAME	HIRE_DATE
hgkjj	05/07/1994

1 rows returned in 0.01 seconds

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

### QUERY

### OUTPUT:

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

```
1 select last_name,job_title from employee where manager_id is null;
```

The results table shows one row:

LAST_NAME	JOB_TITLE
hgkjj	sales

1 rows returned in 0.01 seconds

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

### QUERY:

### OUTPUT:

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

```
1 select last_name,salary,commission from emp where commission is not null
2 order by salary ,commission desc;
```

The results table shows three rows:

LAST_NAME	SALARY	COMMISSION
newman	750	98
patel	3500	56
dancs	8000	27

3 rows returned in 0.01 seconds

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

### QUERY:

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, a query is entered:

```
1 select last_name from emp where last_name like 'a%' or last_name like 'e%';
```

The results are displayed in a table with one column labeled "LAST\_NAME". The results are:

LAST_NAME
aancse
aerri
ewman

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

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

### QUERY:

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, there are several tabs: oracle apex - Search, SQL Commands, gmail - Search, Sent Mail - 220701024@ra..., query to get the lastname..., mysql - Query to retrieve..., and others. The current tab is 'SQL Commands'. The main area shows a SQL command being run:

```
1 select last_name from emp where last_name like 'a%' or last_name like 'e%';
```

The results pane displays the output:

LAST_NAME
aancse
erri
ewman

3 rows returned in 0.01 seconds. There is a 'Download' link below the results.

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:

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, there are several tabs: oracle apex - Search, SQL Commands, gmail - Search, Sent Mail - 220701024@ra..., query to get the lastname..., mysql - Query to retrieve..., and others. The current tab is 'SQL Commands'. The main area shows a SQL command being run:

```
1 select last_name,job_id,salary from emp where job_id in ('SALES','STOCK') and salary not in (2500,3500,7000);
```

The results pane displays the output:

LAST_NAME	JOB_ID	SALARY
patabil	SALES	40000
pitt	STOCK	30000

2 rows returned in 0.02 seconds. There is a 'Download' link below the results.

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

### QUERY:

## OUTPUT:

The screenshot shows a web browser window with the URL <https://apex.oracle.com/pls/apex/f?p=100:10:55010104@tfipisid>. The page title is "APEX". The main content area displays a query result from the "EMPLOYEES" table. The results are presented in a grid format with columns: LAST\_NAME, SALARY, and COMMISSION\_PCT. One row is visible, showing 'MARTINEZ' with a salary of 3200 and a commission percentage of 0.05. The interface includes various APEX navigation and configuration buttons.

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

## RESULT:

# SINGLE ROW FUNCTIONS

**EX\_NO:6**

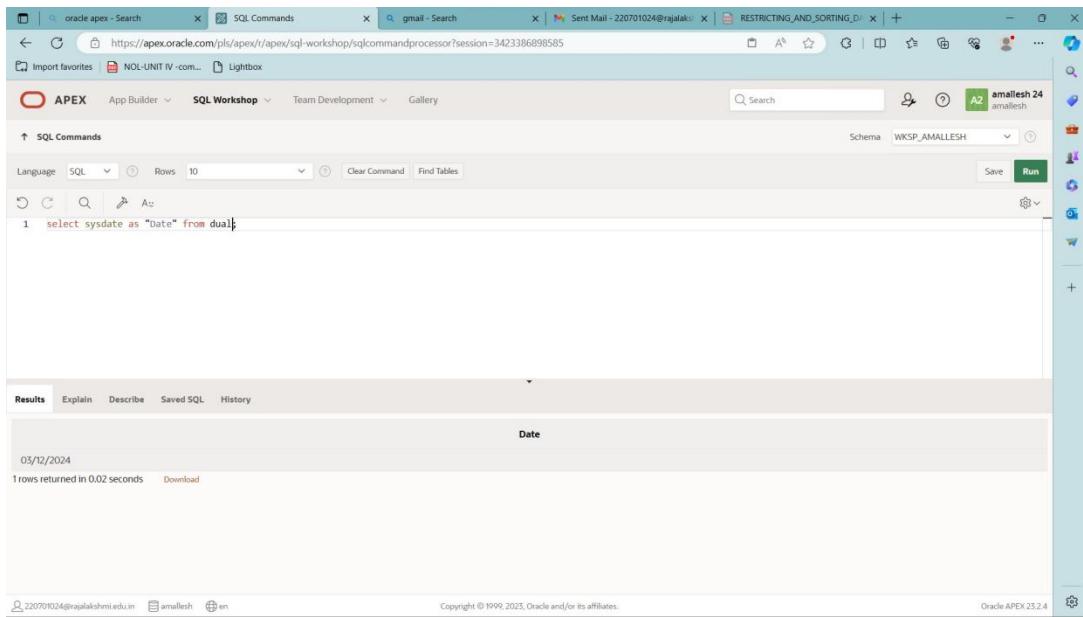
**DATE:**

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

**QUERY:**

```
select sysdate from dual;
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, the URL is https://apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=3423386898585. The main area displays a SQL command: "1 select sysdate as "Date" from dual;". Below the command, the results tab is selected, showing a single row with the date 03/12/2024. The results are returned in 0.02 seconds. The schema is set to WKSP\_AMALLESH.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. In the command editor, a SQL query is written:

```
1 select id,last_name,salary,salary+(15.5/100*salary)"new_salary"from emp;
```

The results pane displays the output of the query:

ID	LAST_NAME	SALARY	new_salary
2	dances	8000	9240
1	patel	3500	4042.5
176	berri	11000	12705
4	newman	750	866.25

4 rows returned in 0.01 seconds

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

#### QUERY:

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

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the command editor, an update query is written:

```
1 update emp set increase=((salary+salary*0.155)-salary);
```

The results pane displays the output of the query:

4 row(s) updated.

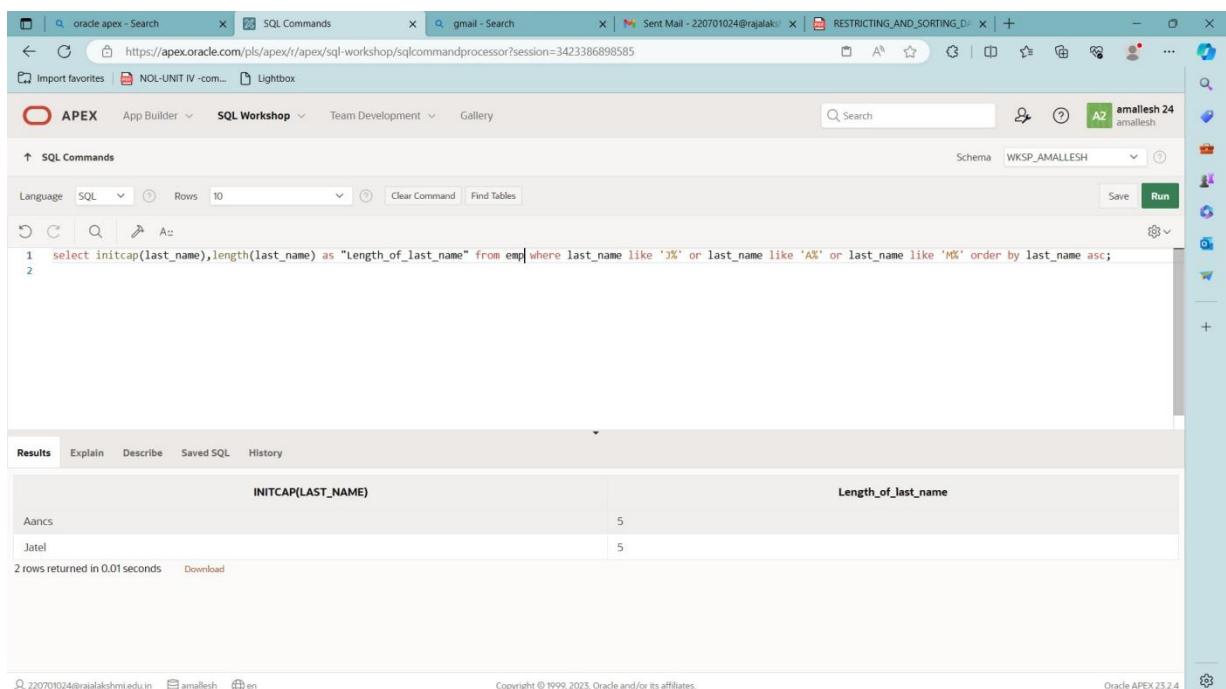
0.04 seconds

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

#### QUERY:

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

#### OUTPUT:



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

```
1 select initcap(last_name),length(last_name) as "Length_of_last_name" from emp| where last_name like 'J%' or last_name like 'A%' or last_name like 'M%' order by last_name asc;
2
```

The Results tab displays the output:

INITCAP(LAST_NAME)	Length_of_last_name
Aancs	5
Jatel	5

2 rows returned in 0.01 seconds

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

#### QUERY:

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, there are several tabs: "oracle apex - Search", "SQL Commands", "gmail - Search", "Sent Mail - 220701024@raj...", "query to get the lastname!", and "mysql - Query to retrieve...". The "SQL Workshop" tab is active.

In the main workspace, under the "SQL Commands" section, a query is entered:

```
1 select last_name from emp where last_name like 'h%';
```

The results pane shows the output:

LAST_NAME
herri

1 rows returned in 0.01 seconds. There is a "Download" link below the results.

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

#### QUERY:

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

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface with the same tab arrangement as the previous screenshot.

In the main workspace, the query is:

```
1 select last_name,round((sysdate-hire_date)/30,0) as "MONTHS_WORKED" from employees order by round((sysdate-hire_date)/30,0) asc;
2
```

The results pane shows the output:

LAST_NAME	MONTHS_WORKED
bbiri	67
hgkij	363
hddin	415

3 rows returned in 0.03 seconds. There is a "Download" link below the results.

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

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX interface with the SQL Workshop tab selected. In the SQL Commands editor, the following query is entered:

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

The Results tab is selected, displaying the output:

DREAM_SALARIES
hddin earns 40000 monthly but wants 120000
hgkjj earns 70000 monthly but wants 210000
bbiri earns 30000 monthly but wants 90000

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

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, a query is being run:

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

The results show three rows of employee data with their last names and formatted salaries:

LAST_NAME	SALARY
huddin	\$\$\$\$\$\$\$\$\$\$40000
hgkji	\$\$\$\$\$\$\$\$\$70000
bbiri	\$\$\$\$\$\$\$\$\$30000

3 rows returned in 0.01 seconds

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

#### QUERY:

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

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, the query is run:

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

The results show three rows of employee data with their last names, hire dates, and formatted review dates:

LAST_NAME	HIRE_DATE	REVIEW
huddin	02/24/1990	Monday, the 27 of August,1990
hgkji	05/07/1994	Monday, the 14 of November,1994
bbiri	09/10/2018	Monday, the 11 of March,2019

3 rows returned in 0.01 seconds

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX interface, specifically the SQL Workshop tab. A user named 'amallesh' is logged in. The query entered is:

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

The results section displays the following data:

LAST_NAME	HIRE_DATE	DAY
bbiri	09/10/2018	Monday
hddin	02/24/1990	Saturday
hgkjj	05/07/1994	Saturday

3 rows returned in 0.00 seconds. The results are shown in a grid format with three columns: LAST\_NAME, HIRE\_DATE, and DAY. The DAY column shows the day of the week corresponding to the hire date.

**RESULT:**

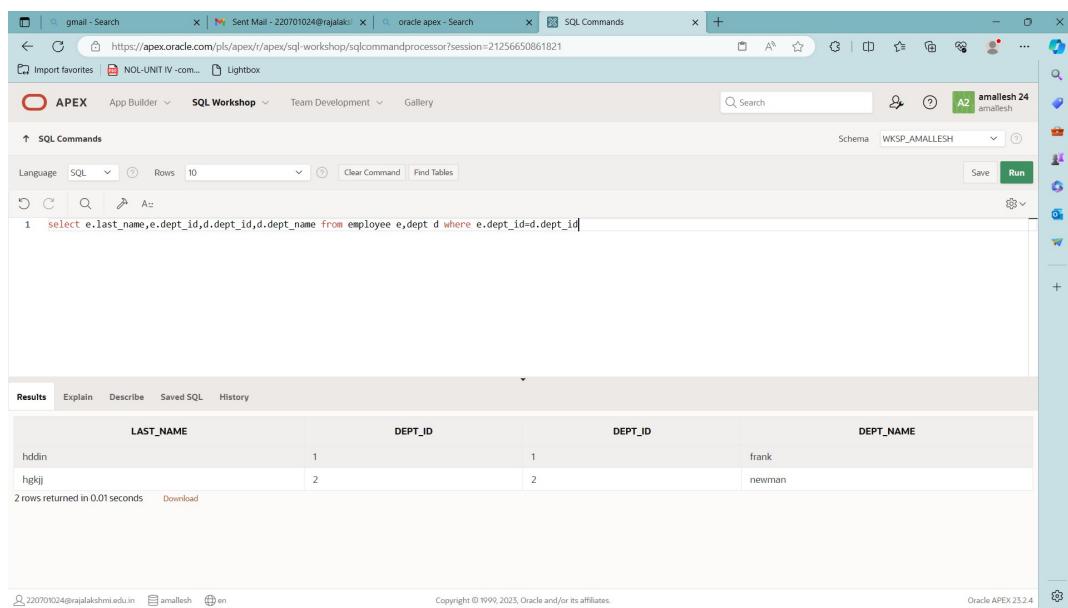
# DISPLAYING DATA FROM MULTIPLE TABLES

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,a.department_name from em e join adr a on e.employee_id
=a.employee_id;
```

**OUTPUT:**



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

```
1 select e.last_name,e.dept_id,d.dept_id,d.dept_name from employee e,dept d where e.dept_id=d.dept_id
```

The results pane displays the following data:

LAST_NAME	DEPT_ID	DEPT_ID	DEPT_NAME
heldin	1	1	frank
hgkij	2	2	newman

2 rows returned in 0.01 seconds

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

**QUERY:**

```
select e.job_title,a.location_id from em e join adr a on e.employee_id=a.employee_id where
e_department_id=80;
```

**OUTPUT:**

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

```
1 select distinct job_id,location_id from employee,dept where employee.dept_id=dept.dept_id and employee.dept_id=80;
```

The results table has two columns: JOB\_ID and LOCATION\_ID. One row is returned:

JOB_ID	LOCATION_ID
stock check	430

1 rows returned in 0.01 seconds

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

### QUERY:

Select e.last\_name,a.department\_name,a.location\_id,a.city from em e join adr a on employee\_id where e.commission is not null

### OUTPUT:

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

```
1 select last_name,dept_name,location_id,city from emp where commission is not null;
```

The results table has four columns: LAST\_NAME, DEPT\_NAME, LOCATION\_ID, and CITY. Three rows are returned:

LAST_NAME	DEPT_NAME	LOCATION_ID	CITY
Aancs	admin	23342	Madurai
Jatel	cse	7689	chennai
newman	marketing	88798	Bangalore

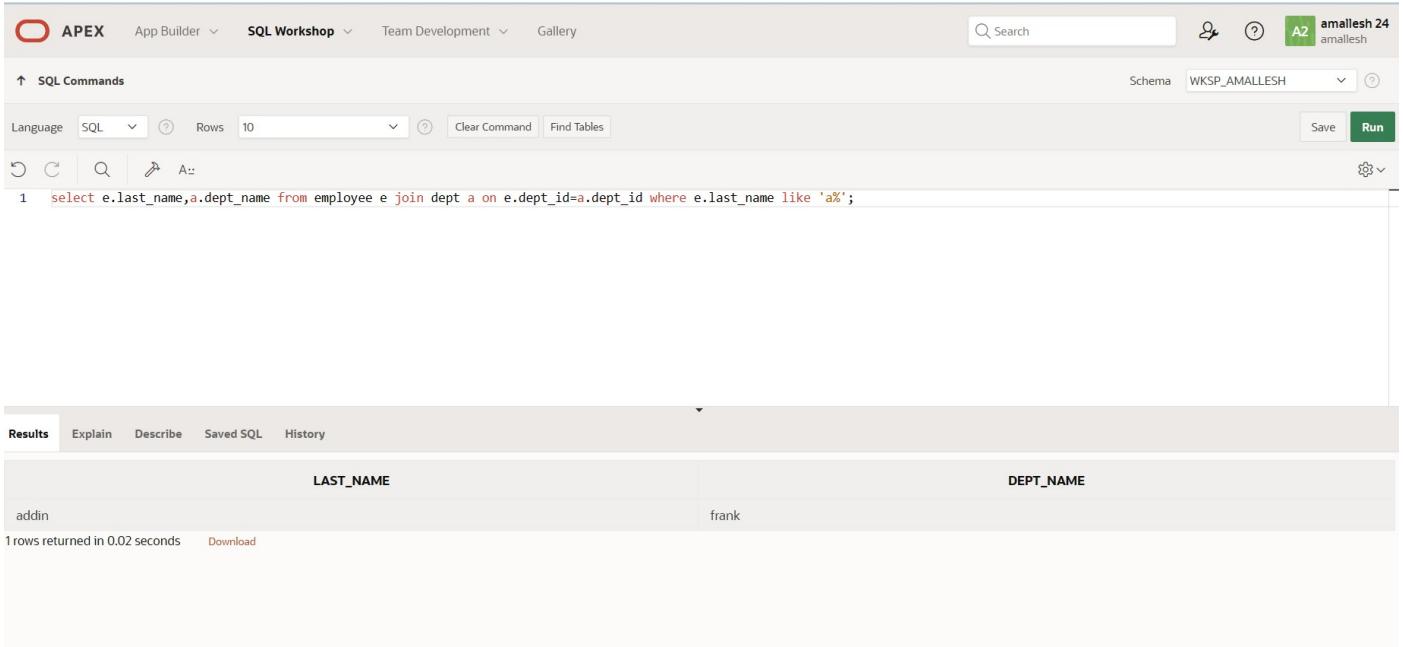
3 rows returned in 0.01 seconds

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

## QUERY:

```
select a.last_name,a.department_name from em e join adr a on employee_id=a.employee_id where e.last_name like 'a%';
```

## OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user information (amallesh 24 amallesh), and a schema dropdown set to WKSP\_AMALLESH. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab shows the query: `select e.last_name,a.dept_name from employee e join dept a on e.dept_id=a.dept_id where e.last_name like 'a%';`. The Results tab displays the output:

LAST_NAME	DEPT_NAME
addin	frank

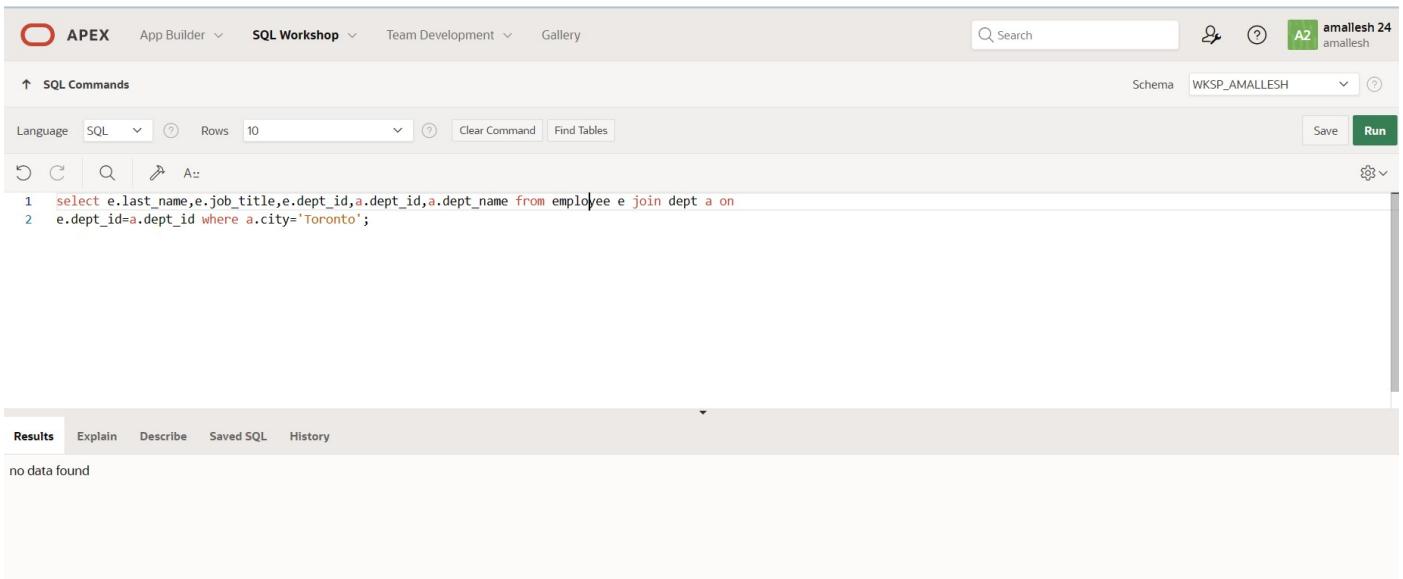
1 rows returned in 0.02 seconds [Download](#)

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

## QUERY:

```
select e.last_name,e.job_title,e.department_id,a.department_id,a.department_name from em e join adr on e.employee_id=a.employee_id where a.city='Toronto';
```

## OUTPUT:



The screenshot shows the Oracle SQL Workshop interface, identical to the previous one but with a different query. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user information (amallesh 24 amallesh), and a schema dropdown set to WKSP\_AMALLESH. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab shows the query: `select e.last_name,e.job_title,e.department_id,a.department_id,a.department_name from employee e join dept a on e.department_id=a.department_id where a.city='Toronto';`. The Results tab displays the output:

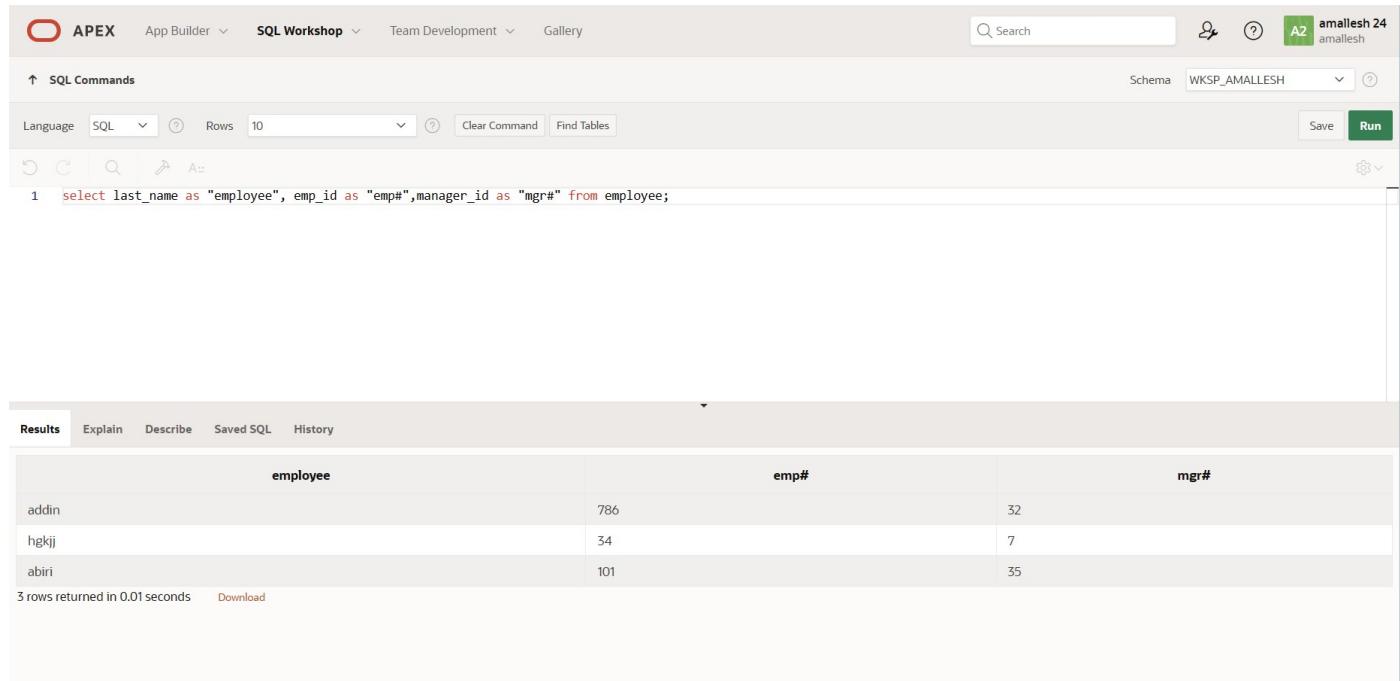
no data found

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

### QUERY:

```
select last_name as "employee", employee_id as "emp#", mg_nm as "manager", manager_id as "mgr#" from em;
```

### 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 is a search bar, a user icon for 'amallesh 24 amallesh', and a schema dropdown set to 'WKSP\_AMALLESH'. Below the toolbar, the SQL Commands tab is selected, showing the following SQL code:

```
1 select last_name as "employee", emp_id as "emp#", manager_id as "mgr#" from employee;
```

The results section displays the output of the query:

employee	emp#	mgr#
addin	786	32
hgkjj	34	7
abiri	101	35

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

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

### QUERY:

```
select e.last_name, e.employee_id, e.mg_nm, a.department_name, a.location_id, a.city, e.salary from em e  
Left outer join adr on e.employee_id=a.employee_id order by e.employee_id;
```

## 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 amallesh 24 with session ID A2. The SQL Commands tab is selected, showing the following query:

```
1 select e.last_name,e.emp_id,a.dept_name,a.location_id,a.city,e.salary from employee e
2 Left outer join dept a on e.dept_id=a.dept_id order by e.emp_id;
```

The Results tab displays the output of the query:

LAST_NAME	EMP_ID	DEPT_NAME	LOCATION_ID	CITY	SALARY
hgkjj	34	newman	750	toronto	7000
abiri	101	-	-	-	30000
addin	786	frank	430	mumbai	40000

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

At the bottom of the page, there are footer links for support, documentation, and Oracle APEX 23.2.4.

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

## QUERY:

Select e.last\_name,e.department\_id,a.department\_name from em e join adr a on e.employee\_id=a.employee\_id order by e.last\_name;

## 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 amallesh 24 with session ID A2. The SQL Commands tab is selected, showing the following query:

```
1 Select e.last_name,e.dept_id,a.dept_name from employee e join dept a on e.dept_id=a.dept_id order by e.last_name;
```

The Results tab displays the output of the query:

LAST_NAME	DEPT_ID	DEPT_NAME
addin	1	frank
hgkjj	2	newman

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

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 em join em davier on (davier\_last\_name='davier') where davier.hire\_date<e.hire\_date;

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The top right shows the user 'amallesh 24 amallesh' and the schema 'WKSP\_AMALLESH'. The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the query is:

```
1 Select e.last_name,e.hire_date from employee e join employee davies on (davies.last_name='davies') where davies.hire_date<e.hire_date;
```

Under 'Results', the output is:

LAST_NAME	HIRE_DATE
abiri	09/10/2018

1 rows returned in 0.01 seconds. The bottom of the screen shows copyright information and the version 'Oracle APEX 23.2.4'.

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" ,hire_date as "emp hire",e.mg_nm as "manager",a.mg_hrdt as "mgr hire" from em e join adr a on e.employee_id=a.employee_id where e.hire_date<a.mg_hrdt;
```

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The top right shows the user 'amallesh 24 amallesh' and the schema 'WKSP\_AMALLESH'. The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the query is:

```
1 Select e.last_name,e.hire_date from employee e join employee davies on (davies.last_name='davies') where davies.hire_date<e.hire_date;
```

Under 'Results', the output is:

LAST_NAME	HIRE_DATE
abiri	09/10/2018

1 rows returned in 0.01 seconds. The bottom of the screen shows copyright information and the version 'Oracle APEX 23.2.4'.

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

EX.NO:8

REG.NO:220701024

DATE:

## AGGREGATING DATA USING GROUP FUNCTIONS

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

## QUERY:

Select max(salary) as “highest”,min(salary) as “lowest”, sum(salary) as “sum”,avg(salary) as “average” from em;

## OUTPUT:

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

### **QUERY:**

Select max(salary) as “highest”,min(salary) as “lowest”, sum(salary) as “sum”,avg(salary) as “average” from em group by job title;

## OUTPUT:

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

```
1 Select max(salary) as "highest",min(salary) as "lowest", sum(salary) as "sum",avg(salary) as "average" from employee group by job_title;
```

The results table has four columns: highest, lowest, sum, and average. The data is as follows:

highest	lowest	sum	average
7000	7000	7000	7000
30000	30000	30000	30000
40000	40000	40000	40000

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

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 count(last_name),job_title from em group by job_title;
```

## OUTPUT:

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

```
1 select count(last_name),job_title from employee group by job_title;
```

The results table has two columns: COUNT(LAST\_NAME) and JOB\_TITLE. The data is as follows:

COUNT(LAST_NAME)	JOB_TITLE
1	sales
1	administration
1	Marketing

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

220701024@rajalakshmi.edu.in amallesh en Copyright © 1999,2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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(manager_id) as "number of managers" from em;
```

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' and 'SQL Workshop' are selected. The schema is set to 'WKSP\_AMALLESH'. The SQL command entered is: `1 Select count(manager_id) as "number of managers" from employee;`. The results tab shows a single row with the value '3' under the column 'number of managers'. The status bar at the bottom indicates '1 rows returned in 0.01 seconds'.

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

**QUERY:**

Select max(salary)-min(salary) as “difference” from em;

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is: `1 Select max(salary)-min(salary) as "difference" from employee;`. The results tab shows a single row with the value '33000' under the column 'difference'. The status bar at the bottom indicates '1 rows returned in 0.01 seconds'.

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

**QUERY:**

select manager\_id,min(salary) from em 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 top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema is set to WKSP\_AMALLESH. The query entered is:

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

The results section displays the following data:

MANAGER_ID	MIN(SALARY)
32	40000
35	30000
7	7000

3 rows returned in 0.01 seconds. The bottom of the page includes copyright information for Oracle and the APEX version.

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(last\_name) from em where hire\_date between '01-01-1975' and '12-31-1998';

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema is set to WKSP\_AMALLESH. The query entered is:

```
1 Select count(last_name) from employee where hire_date between '01-01-1975' and '12-31-1998';
```

The results section displays the following data:

COUNT(LAST_NAME)
3

1 rows returned in 0.01 seconds. The bottom of the page includes copyright information for Oracle and the APEX version.

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 as "job",sum(decode(department_id,20,salary)) as  
"dept_20",sum(decode(department_id,50,salary)) as "dept_50",sum(decode(department_id,90,salary)) as  
"dept_90",sum(salary) as "total" from em group by job_id;
```

## OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'amallesh 24 amallesh'. The main area is titled 'SQL Commands' with tabs for SQL, PL/SQL, Rows (set to 10), Clear Command, and Find Tables. A 'Run' button is at the bottom right. The code entered is: `1 Select count(last_name) from employee where hire_date between '01-01-1975' and '12-31-1998';`. The results tab is selected, showing the output: `COUNT(LAST_NAME)` with a value of 3. Below it, it says '1 rows returned in 0.01 seconds' and has a 'Download' link.

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

## QUERY:

```
select d.department_name as department_name,a.city as city,count(e.employee_id) as number_of_people,  
Round(avg(e.salary),2) as salary from adr d join em e on d.employee_id=e.employee_id group by  
d.department_name,d.city;
```

## OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'amallesh 24 amallesh'. The main area is titled 'SQL Commands' with tabs for SQL, PL/SQL, Rows (set to 10), Clear Command, and Find Tables. A 'Run' button is at the bottom right. The code entered is: `1 select d.dept_name as department_name,d.city as city,count(e.emp_id) as number_of_people,  
2 Round(avg(e.salary),2) as salary from dept d join employee e on e.dept_id=d.dept_id group by d.dept_name,d.city;`. The results tab is selected, showing the output: a table with columns DEPARTMENT\_NAME, CITY, NUMBER\_OF\_PEOPLE, and SALARY. The data is: newman, toronto, 1, 7000; frank, mumbai, 1, 40000. Below the table, it says '2 rows returned in 0.06 seconds' and has a 'Download' link.

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

# SUB QUERIES

**EX\_NO:9**

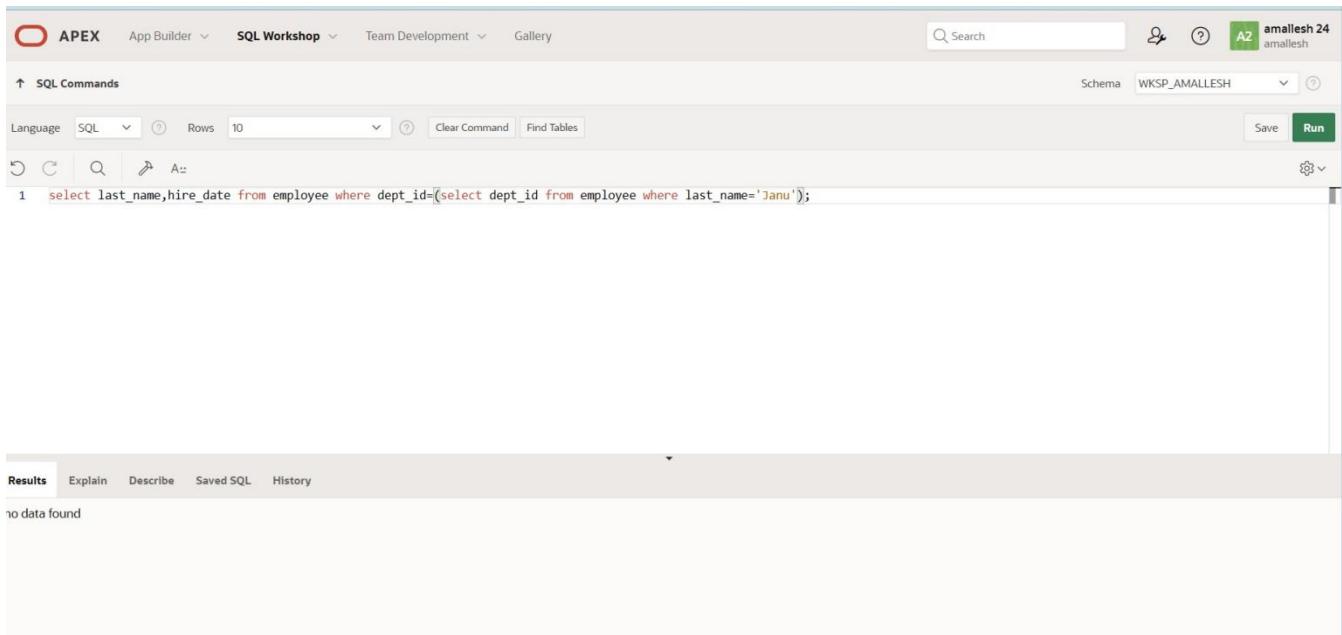
**DATE:**

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A single line of SQL code is entered in the command input field:

```
1 select last_name,hire_date from employees where dept_id=(select dept_id from employee where last_name='Janu'));
```

The 'Run' button is visible at the bottom right of the command input field. Below the command input, there is a results pane with tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, and the message 'no data found' is displayed.

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

**QUERY:**

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

**OUTPUT:**

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language SQL Rows 10 Clear Command Find Tables

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

Save Run

Results Explain Describe Saved SQL History

EMP_ID	LAST_NAME	SALARY
101	abiri	30000
786	Metos	40000

2 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:

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language SQL Rows 10 Clear Command Find Tables

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

Save Run

Results Explain Describe Saved SQL History

EMP_ID	LAST_NAME
786	Metos
34	janu

2 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 location_id=1700);
```

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a user profile for 'amallesh 24 amallesh'. The main area has tabs for SQL Commands, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, showing the query: 'select last\_name,dept\_id,job\_id from employee where dept\_id=(select dept\_id from dept where location\_id=1700);'. Below the query, the results tab is selected, displaying a table with three columns: LAST\_NAME, DEPT\_ID, and JOB\_ID. The data shows two rows: Metos (DEPT\_ID 50, JOB\_ID St\_Clerk) and janu (DEPT\_ID 50, JOB\_ID stock check). The status bar at the bottom indicates '2 rows returned in 0.01 seconds'.

LAST_NAME	DEPT_ID	JOB_ID
Metos	50	St_Clerk
janu	50	stock check

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 'amallesh 24 amallesh' and a schema dropdown set to 'WKSP\_AMALLESH'. The main area is titled 'SQL Commands' with a 'Run' button. A query is entered in the command line:

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

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

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

### QUERY:

```
select 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 'amallesh 24 amallesh' and a schema dropdown set to 'WKSP\_AMALLESH'. The main area is titled 'SQL Commands' with a 'Run' button. A query is entered in the command line:

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

The results section shows a table with three columns: 'EMP\_ID', 'LAST\_NAME', and 'SALARY'. The data is as follows:

EMP_ID	LAST_NAME	SALARY
786	Metos	40000
101	abiri	30000

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

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

### QUERY:

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

### OUTPUT:

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

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

The results window displays the following data:

EMP_ID	LAST_NAME	SALARY
786	Metos	40000
101	abiri	30000

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

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

### RESULT:

# USING THE SET OPERATORS

**EX\_NO:10**

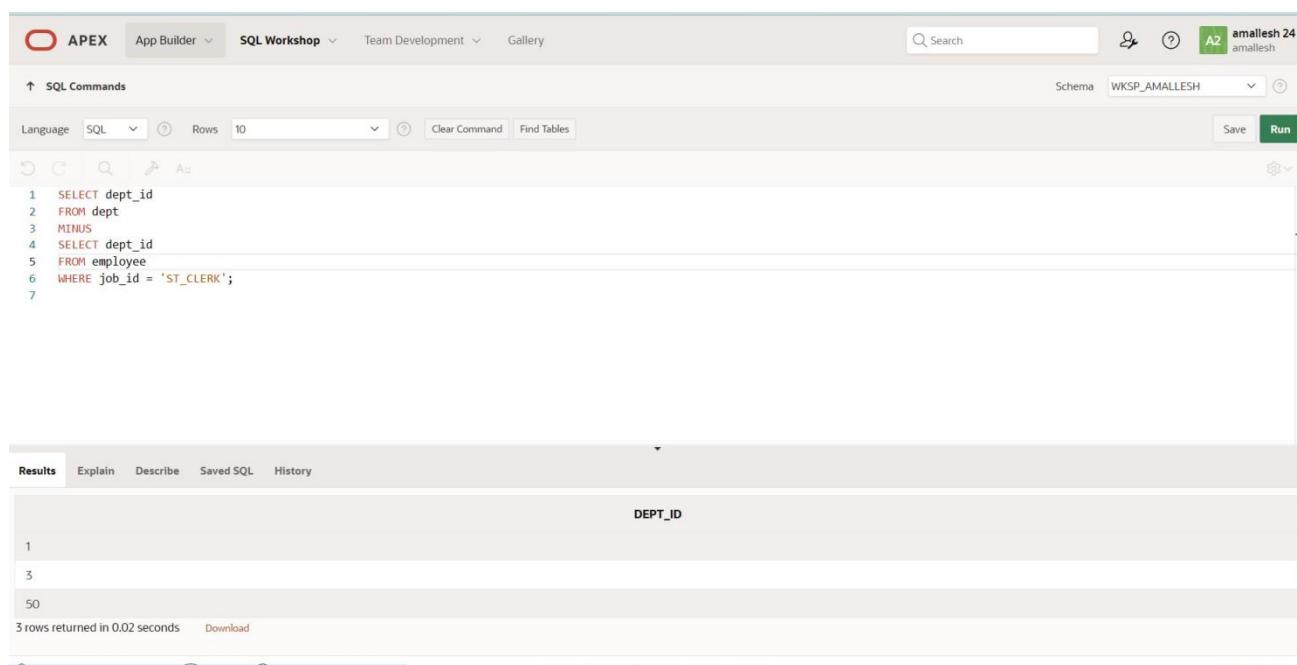
**DATE:**

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 SQL command window contains the following code:

```
1 SELECT dept_id  
2 FROM dept  
3 MINUS  
4 SELECT dept_id  
5 FROM employee  
6 WHERE job_id = 'ST_CLERK';  
7
```

The results window displays the output:

DEPT_ID
1
3
50

3 rows returned in 0.02 seconds

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

**QUERY:**

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

**OUTPUT:**

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

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

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

Results Explain Describe Saved SQL History

Name	DEPT_ID
brop Metos	10
gddh davies	50
dane abiri	20
frank	1

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:

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

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

```
1 select job_id,dept_id from employee where dept_id=10 union
2 select job_id,dept_id from employee where dept_id=50 union
3 select job_id,dept_id from employee where dept_id=20;
```

Results Explain Describe Saved SQL History

JOB_ID	DEPT_ID
St_Clerk	10
stock check	20
stock check	50

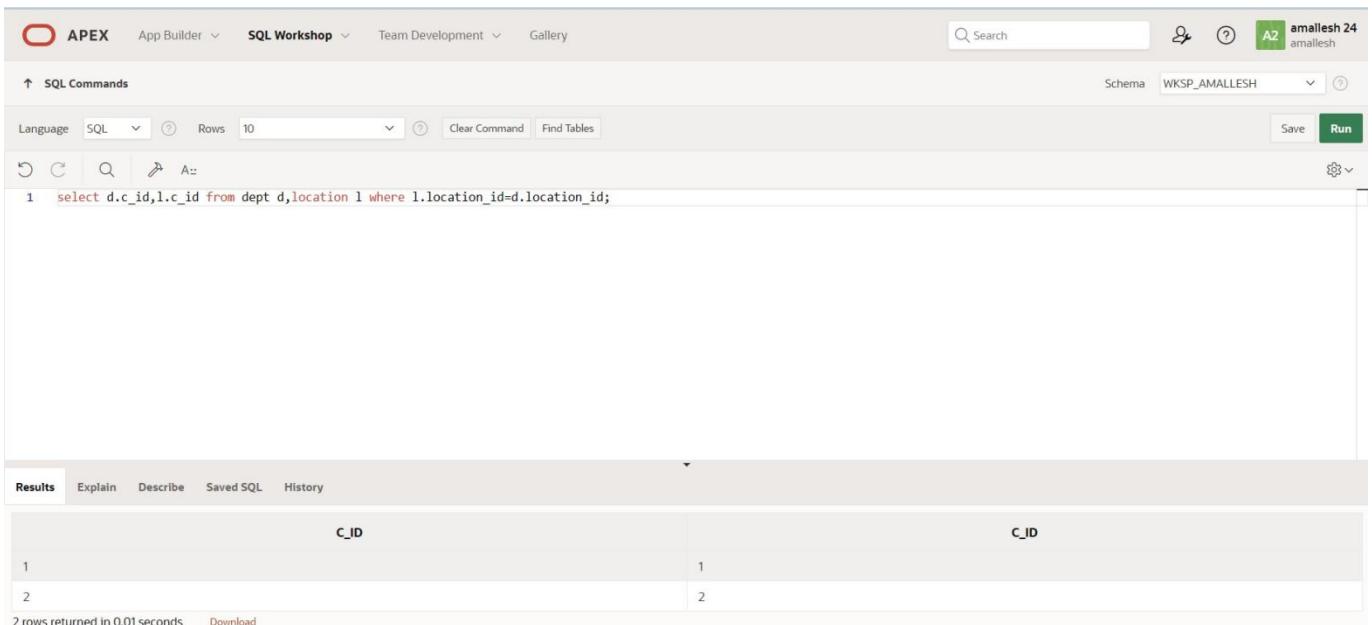
3 rows returned in 0.01 seconds Download

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

#### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a user icon for 'amallesh 24' and a schema dropdown set to 'WKSP\_AMALLESH'. The main area is titled 'SQL Commands' with options for Language (SQL selected), Rows (10), Clear Command, and Find Tables. Below this is a toolbar with icons for Undo, Redo, Search, and Run. The SQL editor contains the following query:

```
1 select d.c_id,l.c_id from dept d,location l where l.location_id=d.location_id;
```

The results section shows a table with two columns, C\_ID, containing the values 1 and 2.

C_ID	C_ID
1	1
2	2

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

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

#### QUERY:

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

#### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. At the top, there are tabs for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, help icons, and a session identifier for user amallesh 24.

In the main area, the SQL Commands tab is selected. The language is set to SQL, rows are set to 10, and there are buttons for Clear Command and Find Tables. The Run button is highlighted in green.

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

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

Name	DEPT_ID
brop Metos	10
gddh davies	50
dane abiri	20
frank	1

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

**RESULT:**



# CREATING VIEWS

**EX\_NO:11**

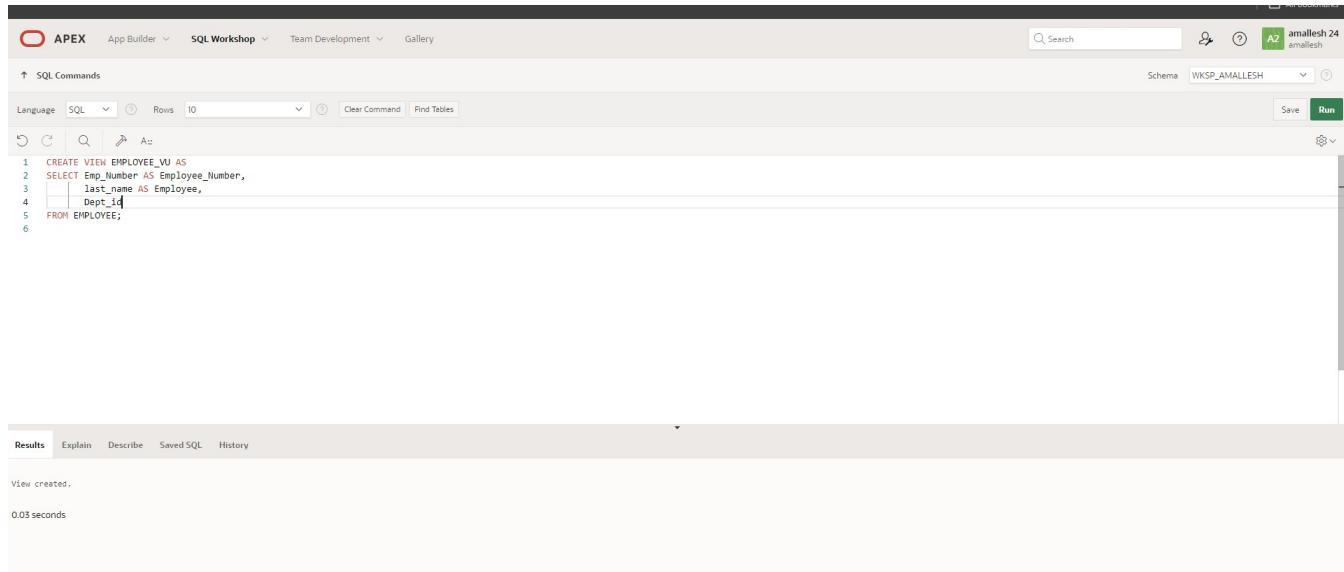
**DATE:**

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is chosen from the dropdown menu. The main area contains a SQL command editor with the following code:

```
1 CREATE VIEW EMPLOYEE_VU AS
2 SELECT Emp_Number AS Employee_Number,
3        last_name AS Employee,
4        Dept_1q AS Department
5   FROM EMPLOYEE;
```

Below the editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, showing the message "View created." and a time of "0.03 seconds".

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

**QUERY:**

```
select * from employees_view;
```

**OUTPUT:**

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

```
1 SELECT * FROM EMPLOYEE_VU;
```

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

EMPLOYEE_NUMBER	EMPLOYEE	DEPT_ID
7359192552	addin	1
8655329084	davies	2
8220489702	abiri	45

At the bottom of the results panel, it says '3 rows returned in 0.05 seconds' and has a 'Download' link.

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. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is active. The main area contains a SQL command editor with the following code:

```
1 SELECT view_name, text
2 FROM user_views;
3
4
5
6
```

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

VIEW_NAME	TEXT
EMPLOYEE_VU	SELECT Emp_Number AS Employee_Number, last_name AS Employee, Dept_Id FROM EMPLOYEE

At the bottom of the results panel, it says '1 rows returned in 0.06 seconds' and has a 'Download' link.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. At the top, there's a navigation bar with links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, it shows the user 'amallesh 24' and the schema 'WKSP\_AMALLESH'. Below the navigation is a toolbar with icons for search, refresh, and run. The main area has tabs for SQL Commands, Language, and SQL. A dropdown menu shows 'Rows: 10'. Buttons for Clear Command and Find Tables are present. The SQL editor contains the following code:

```

1 SELECT Employee, Dept_id
2 FROM EMPLOYEE_VU;
3
4
5
6
7

```

Below the editor, the 'Results' tab is selected. It displays a table with two columns: 'EMPLOYEE' and 'DEPT\_ID'. The data is as follows:

EMPLOYEE	DEPT_ID
addin	1
davies	2
albri	45

Text at the bottom left says '3 rows returned in 0.01 seconds' and 'Download'. On the right, there's a message: 'Activate Windows' and 'Go to Settings to activate Windows.'

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. At the top, there's a navigation bar with links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, it shows the user 'amallesh 24' and the schema 'WKSP\_AMALLESH'. Below the navigation is a toolbar with icons for search, refresh, and run. The main area has tabs for SQL Commands, Language, and SQL. A dropdown menu shows 'Rows: 10'. Buttons for Clear Command and Find Tables are present. The SQL editor contains the following code:

```

1 CREATE VIEW DEPT50 AS
2 SELECT Emp_id AS EMPNO,
3        Last_Name AS EMPLOYEE,
4        Dept_id AS DEPTNO
5   FROM EMPLOYEE
6  WHERE Dept_id = 50
7  WITH CHECK OPTION;
8
9
10
11
12
13

```

Below the editor, the 'Results' tab is selected. It displays the message 'View created.' and '0.03 seconds'.

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

**QUERY:**

Describe dept50;

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'amallesh 24 amallesh'. The main area is titled 'SQL Commands' with tabs for Language (SQL selected), Rows (10), Clear Command, and Find Tables. The command input field contains: 1 SELECT \* FROM DEPT50; 2 . Below the input field, the results tab is selected, showing the message 'no data found'.

7.) Attempt to reassign Matos to department 80

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'amallesh 24 amallesh'. The main area is titled 'SQL Commands' with tabs for Language (SQL selected), Rows (10), Clear Command, and Find Tables. The command input field contains: 1 UPDATE DEPT50 2 SET DEPTNO = 80 3 WHERE EMPLOYEE = 'Matos'; 4 . Below the input field, the results tab is selected, showing the message '0 row(s) updated.' and '0.04 seconds'.

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

#### QUERY:

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

#### OUTPUT:

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

```
1 CREATE VIEW SALARY_VU AS
2 SELECT E.Last_Name AS Employee,
3        D.Dept_Name AS Department,
4        E.Salary,
5        JG.Grade
6   FROM Employee E
7  JOIN Dept D ON E.Dept_Id = D.Dept_Id
8  JOIN Job_Grade JG ON E.Salary BETWEEN JG.Low_Sal AND JG.High_Sal;
```

Below the code editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected. The results pane displays the message 'View created.' and '0.05 seconds'.

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

**RESULT:**

# EXERCISE 12

## PRACTICE QUESTIONS

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Table Created.

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

DESCRIBE f\_global\_locations;

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

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

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

# PRIMARY KEY, FOREIGN KEY, and CHECK Constraints

1. What is the purpose of a

- PRIMARY KEY
- FOREIGN KEY
- CHECK CONSTRAINT

**a. PRIMARY KEY**

Uniquely identify each row in table.

**b. FOREIGN KEY**

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

**c. CHECK CONSTRAINT**

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

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

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

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

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

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

ANIMAL_ID	NAME	LICENSE_TAG_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?

The screenshot shows a database query results interface. At the top, there are tabs for 'Results' (which is selected), 'Explain', 'Describe', 'Saved SQL', and 'History'. Below the tabs is a table with three columns: 'ID', 'Song Title', and 'ARTIST'. Two rows of data are displayed:

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

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

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

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

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

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

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

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

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

# DML Operations and Views

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

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

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

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

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

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

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

```
SELECT * FROM view_copy_d_songs;
```

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

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

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

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

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

```
SELECT * FROM read_copy_d_cds;
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

# Managing Views

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

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

```
SELECT * FROM view_copy_d_songs;
```

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

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

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

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

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

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

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

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

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

# **Indexes and Synonyms**

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

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

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

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

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

3. When will an index be created automatically?

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

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

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

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

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

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

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

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

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

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

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

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

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

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

10. Drop the synonym that you created in question

**DROP SYNONYM dj\_tracks2;**

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

**RESULT:**

# OTHER DATABASE OBJECTS

**EX\_NO:14**

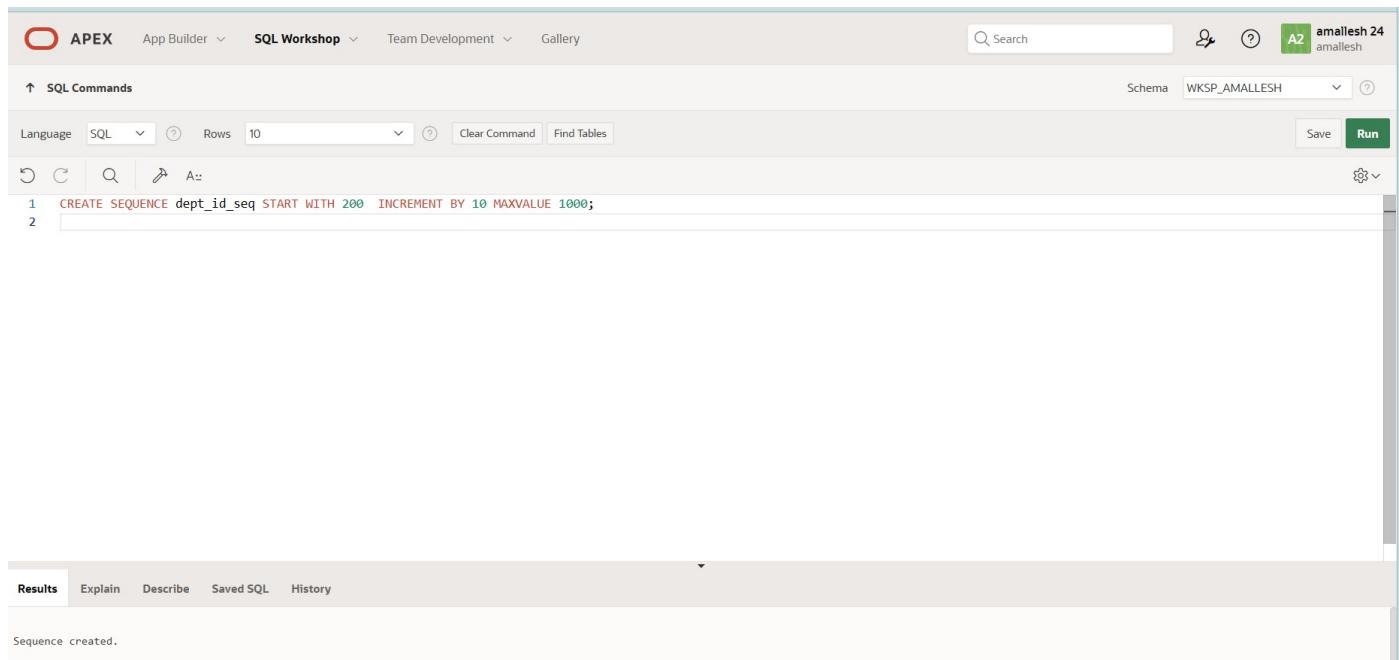
**DATE:**

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

**QUERY:**

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

**OUTPUT:**



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

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

In the bottom results pane, it displays the message: "Sequence created."

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is active. The main area is titled 'SQL Commands'. A search bar at the top right contains the placeholder 'Search'. On the far right, there's a user icon for 'amallesh 24 amallesh' and a gear icon. The command input field contains the following SQL:

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

Below the command, the results section displays the output of the query:

SEQUENCE_NAME	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
DEPT_ID_SEQ	1000	10	200

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

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is active. The main area is titled 'SQL Commands'. A search bar at the top right contains the placeholder 'Search'. On the far right, there's a user icon for 'amallesh 24 amallesh' and a gear icon. The command input field contains the following SQL:

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

Below the command, the results section displays the output of the query:

1 row(s) inserted.

0.01 seconds

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is highlighted. The main area is titled 'SQL Commands'. A single line of SQL code is entered in the command input field:

```
1 CREATE INDEX emp_dept_id_idx ON EMPLOYEE (dept_id);  
2
```

Below the command, the results tab is active, showing the output of the executed query:

Index created.  
0.03 seconds

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

**QUERY:**

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

**OUTPUT:**

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language: SQL Rows: 10 Schema: WKSP\_AMALLESH

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

Save Run

**Results** Explain Describe Saved SQL History

INDEX_NAME	TABLE_NAME	UNIQUENESS
EMP_DEPT_ID_IDX	EMPLOYEE	NONUNIQUE

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

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

**RESULT:**

# CONTROLLING USER ACCESS

**EX\_NO:15**

**DATE:**

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

The CREATE SESSION system privilege

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

The CREATE TABLE privilege

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

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

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

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

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

The ALTER USER statement

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

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

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

7. Query all the rows in your DEPARTMENTS table.

SELECT \* FROM departments;

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

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

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

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

SELECT table\_name FROM user\_tables;

10. Revoke the SELECT 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:**

EXNO:16

# PL/SQL

DATE:

## CONTROL STRUCTURES

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, a PL/SQL block is written and executed. The output pane displays the result of the DBMS\_OUTPUT.PUT\_LINE statement.

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

Incentive = 2400  
Statement processed.  
0.01 seconds

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

**QUERY:**

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

```

/

**OUTPUT:**



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

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

In the Results pane, the output is displayed:

```
Welcome
Statement processed.
0.01 seconds
```

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

#### QUERY:

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

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

#### OUTPUT:

```
1  DECLARE
2      salary_of_emp NUMBER(8,2);
3  PROCEDURE approx_salary (
4      emp      NUMBER,
5      empsal IN OUT NUMBER,
6      addless  NUMBER
7  ) IS
8  BEGIN
9      empsal := empsal + addless;
10 END;
11
12 BEGIN
13     SELECT salary INTO salary_of_emp
14     FROM employees
15     WHERE employee_id = 122;
16     DBMS_OUTPUT.PUT_LINE
17         ('Before invoking procedure, salary_of_emp: ' || salary_of_emp);
18     approx_salary (100, salary_of_emp, 1000);
19     DBMS_OUTPUT.PUT_LINE
20         ('After invoking procedure, salary_of_emp: ' || salary_of_emp);
21 END;
22 /
23
```

Results

```
Before invoking procedure, salary_of_emp: 10000
After invoking procedure, salary_of_emp: 11000
Statement processed.
```

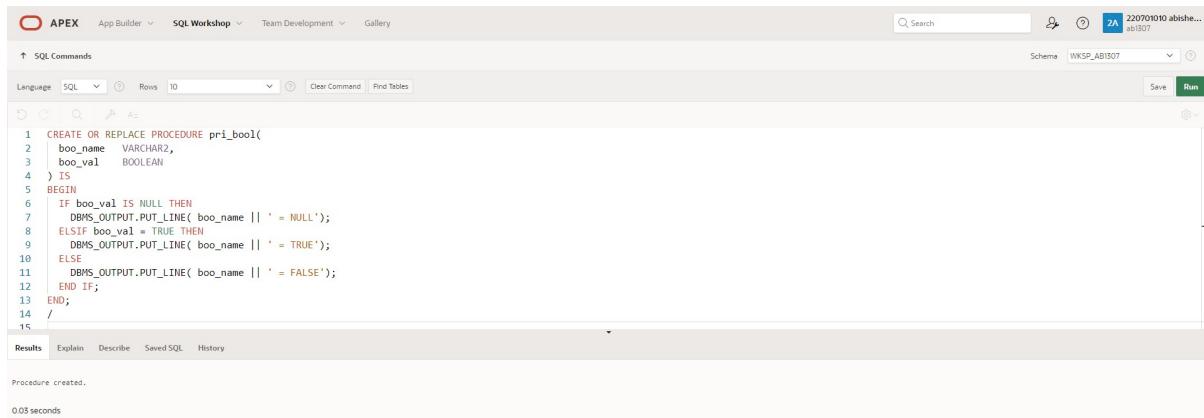
0.01 seconds

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, tabs for 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery' are visible. On the right side, there's a user profile icon and the text '220701010 abishe...'. A search bar and a refresh button are also present. The main workspace is a SQL editor with the following content:

```
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;
/
```

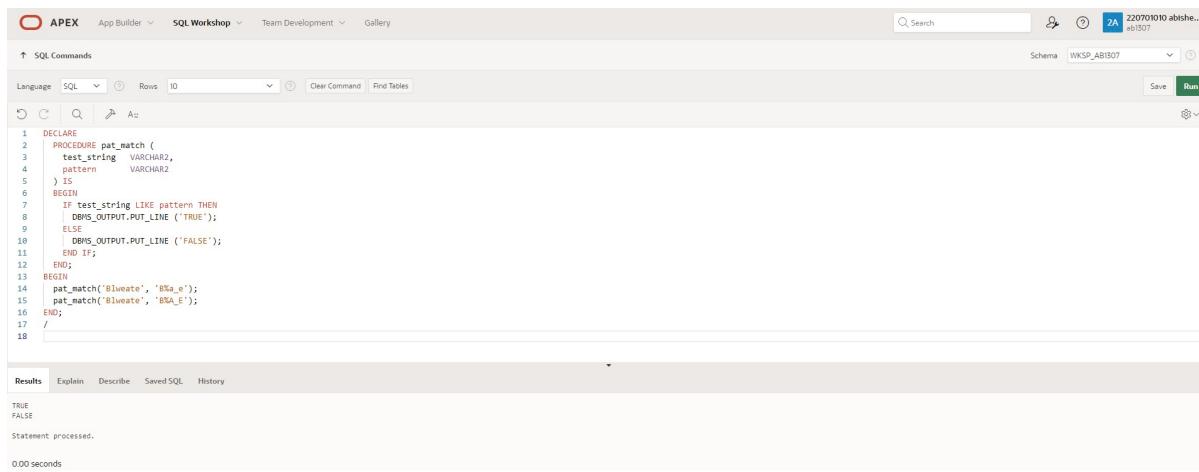
Below the code, the status bar displays 'Procedure created.' and '0.03 seconds'.

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown shows 'WKSP\_AB1507'. The main area displays the PL/SQL code. Below the code, the 'Results' tab is active, showing the output of the execution. The output consists of two rows: 'TRUE' and 'FALSE', indicating the results of the two pattern matches. The status bar at the bottom indicates 'Statement processed.' and '0.00 seconds'.

```
1  DECLARE
2  2    PROCEDURE pat_match (
3    3      test_string  VARCHAR2,
4    4      pattern      VARCHAR2
5    5  ) IS
6  BEGIN
7    IF test_string LIKE pattern THEN
8      DBMS_OUTPUT.PUT_LINE ('TRUE');
9    ELSE
10      DBMS_OUTPUT.PUT_LINE ('FALSE');
11    END IF;
12  END;
13  BEGIN
14    pat_match('Blweate', 'B%a_e');
15    pat_match('Blweate', 'B%A_E');
16  END;
17 /
18
```

Results Explain Describe Saved SQL History

TRUE  
FALSE  
Statement processed.  
0.00 seconds

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

**QUERY:**

DECLARE

num\_small NUMBER := 8;

num\_large NUMBER := 5;

num\_temp NUMBER;

BEGIN

IF num\_small > num\_large THEN

num\_temp := num\_small;

num\_small := num\_large;

num\_large := num\_temp;

END IF;

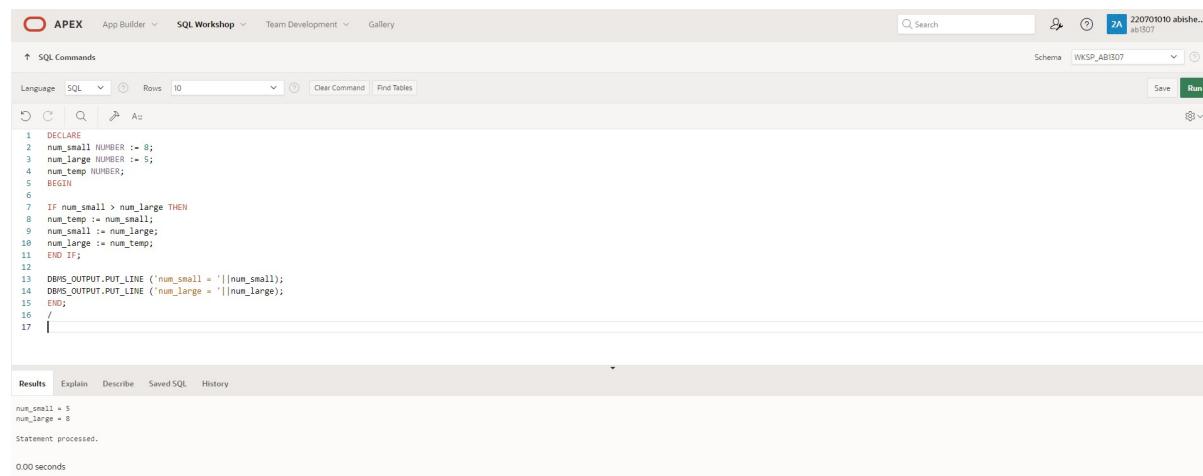
DBMS\_OUTPUT.PUT\_LINE ('num\_small ='||num\_small);

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

END;

/

**OUTPUT:**



```
1 DECLARE
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 |
```

Results Explain Describe Saved SQL History

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

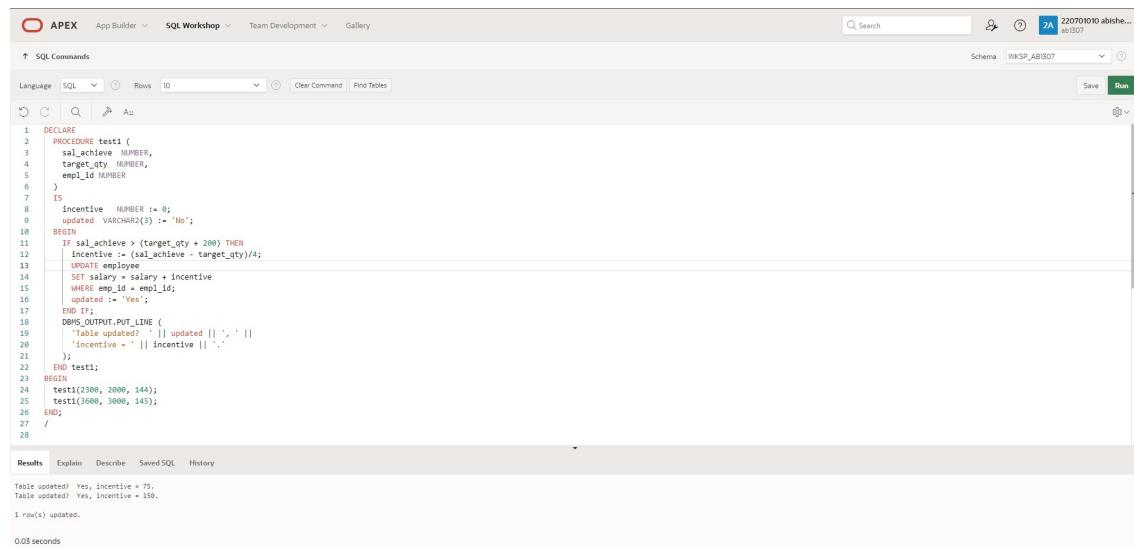
0.00 seconds
```

**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 SQL Workshop interface. The code area contains the PL/SQL procedure 'test1'. When executed, it prints two rows of output. Row 1 shows 'Table updated? Yes' and 'incentive = 75'. Row 2 shows 'Table updated? Yes' and 'incentive = 150'. Both rows have a status of 'Success'.

```
1  DECLARE
2  PROCEDURE test1 (
3    sal_achieve NUMBER,
4    target_qty NUMBER,
5    emp_id NUMBER
6  )
7  IS
8    incentive NUMBER := 0;
9    updated VARCHAR2(3) := 'No';
10   BEGIN
11     IF sal_achieve > (target_qty + 200) THEN
12       incentive := (sal_achieve - target_qty)/4;
13       UPDATE employees
14         SET salary = salary + incentive
15         WHERE employee_id = emp_id;
16       updated := 'Yes';
17     END IF;
18     DBMS_OUTPUT.PUT_LINE (
19       'Table updated? ' || updated || ',' ||
20       'incentive = ' || incentive || '.'
21     );
22   END test1;
23   BEGIN
24     test1(2300, 2000, 144);
25     test1(3600, 3000, 145);
26   END;
27 /
```

Row	Table updated?	Incentive
1	Yes	75
2	Yes	150

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

## QUERY:

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

## OUTPUT:

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

Results

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

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

**QUERY:**

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

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

```
BEGIN
```

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

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

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

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

```
    ELSE
```

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

```
    END IF;
```

```
END;
```

```
/
```

**OUTPUT:**

```
APEX App Builder SQL Workshop Team Development Gallery
SQL Commands
Language SQL Rows 10 Clear Command Find Tables
Save Run
1 tot_emp NUMBER;
2 BEGIN
3     SELECT Count(*)
4         INTO tot_emp
5         FROM employees e
6         join dept d
7             ON e.dept_id = d.dept_id
8         WHERE e.dept_id = 50;
9     dbms_output.Put_line ('The employees are in the department 50: '
10                           ||To_char(tot_emp));
11     IF tot_emp >= 45 THEN
12         dbms_output.Put_line ('There are no vacancies in the department 50.');
13     ELSE
14         dbms_output.Put_line ('There are some vacancies in department 50.');
15     END IF;
16 END;
17 /
18 /
```

The employees are in the department 50:2  
There are some vacancies in department 50.  
Statement processed.

**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

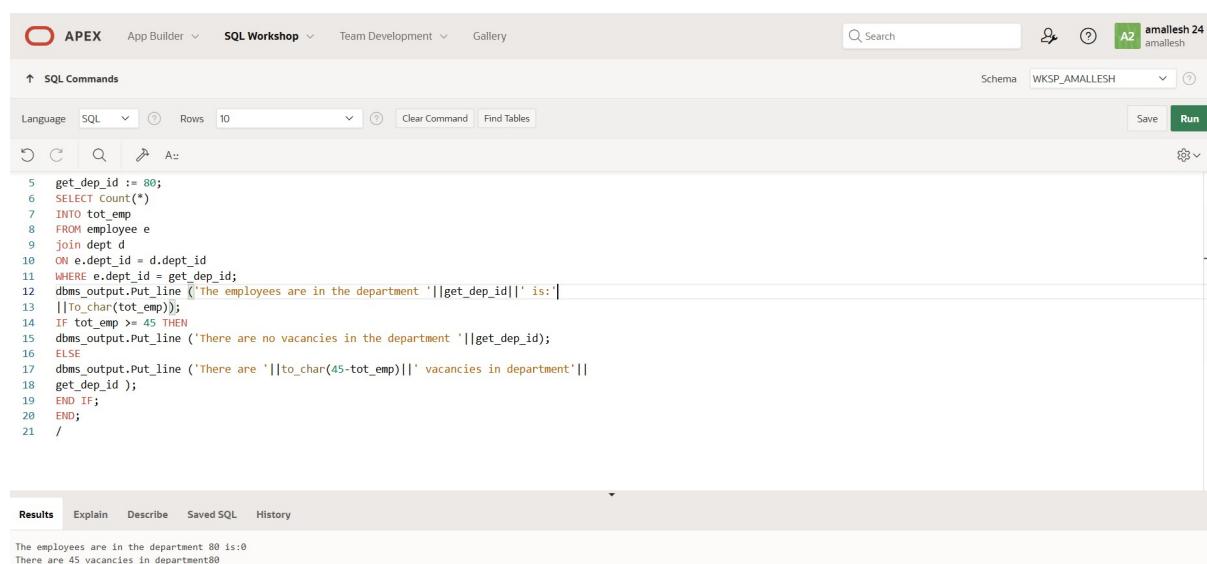
```
    dbms_output.Put_line ('There are'||to_char(45-tot_emp)||' vacancies in department ||
get_dep_id );
```

END IF;

END;

/

**OUTPUT:**



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

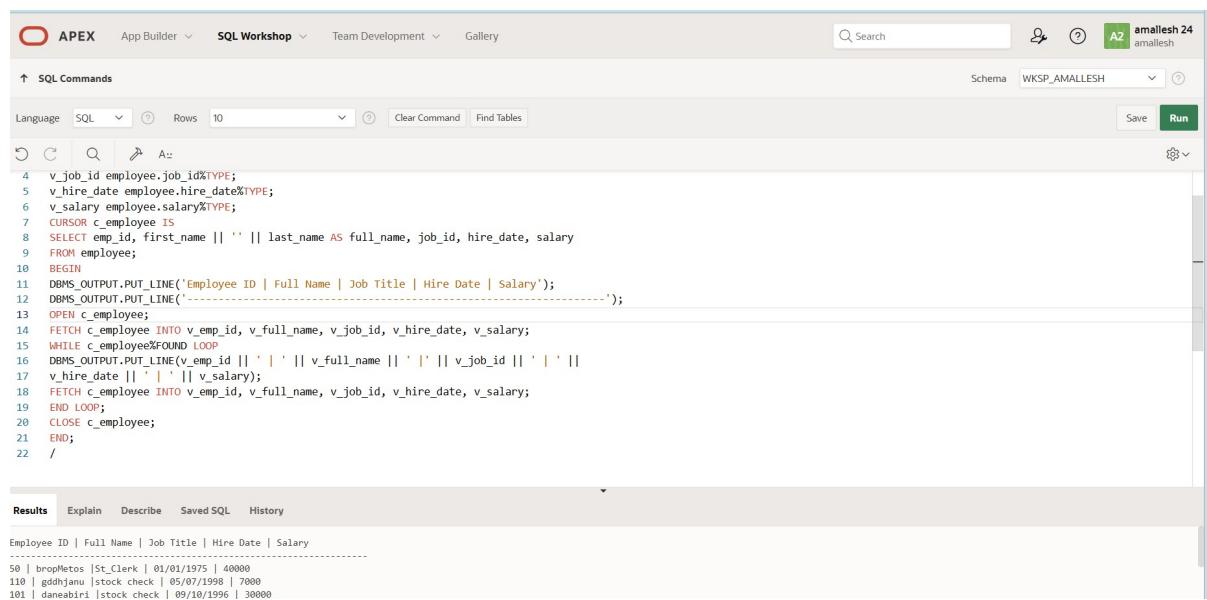
The employees are in the department 80 is:0  
There are 45 vacancies in department 80

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile 'amallesh 24 amallesh'. The main area is titled 'SQL Commands' with a 'Language' dropdown set to 'SQL'. The code area contains the PL/SQL block from above. The results tab at the bottom displays the output:

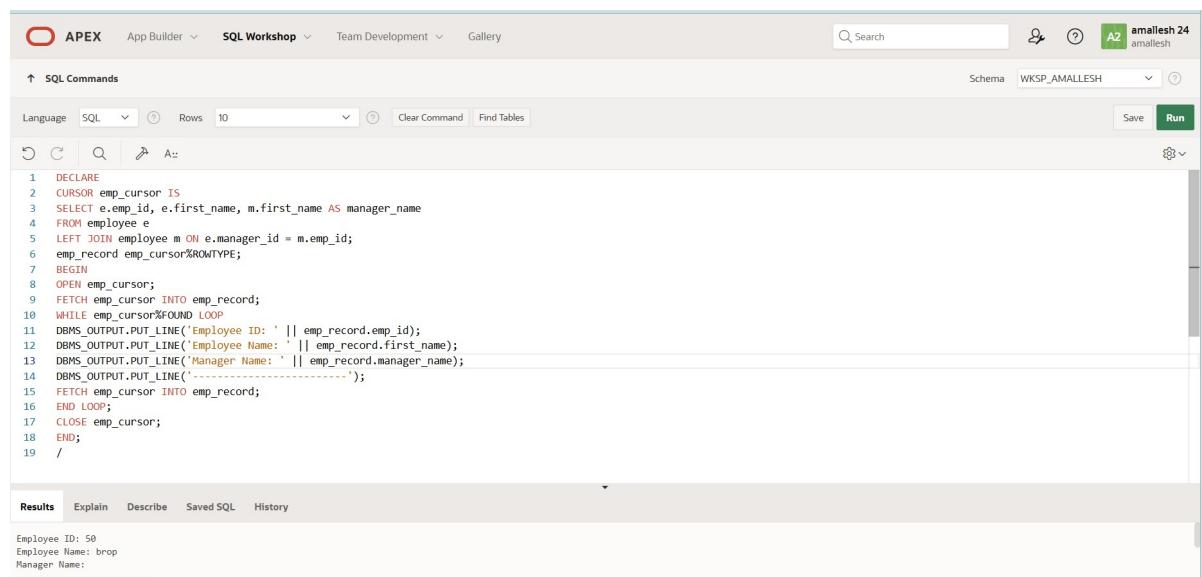
Employee ID	Full Name	Job Title	Hire Date	Salary
50	browMetos	St.Clerk	01/01/1975	40000
110	gddhjanu	stock check	05/07/1998	7000
101	daneabiri	stock check	09/10/1996	30000

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema is set to WKSP\_AMALLESH. The code area contains the PL/SQL block provided above. The results pane at the bottom shows the output of the DBMS\_OUTPUT.PUT\_LINE statements:

```
Employee ID: 50
Employee Name: brop
Manager Name: -----
```

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

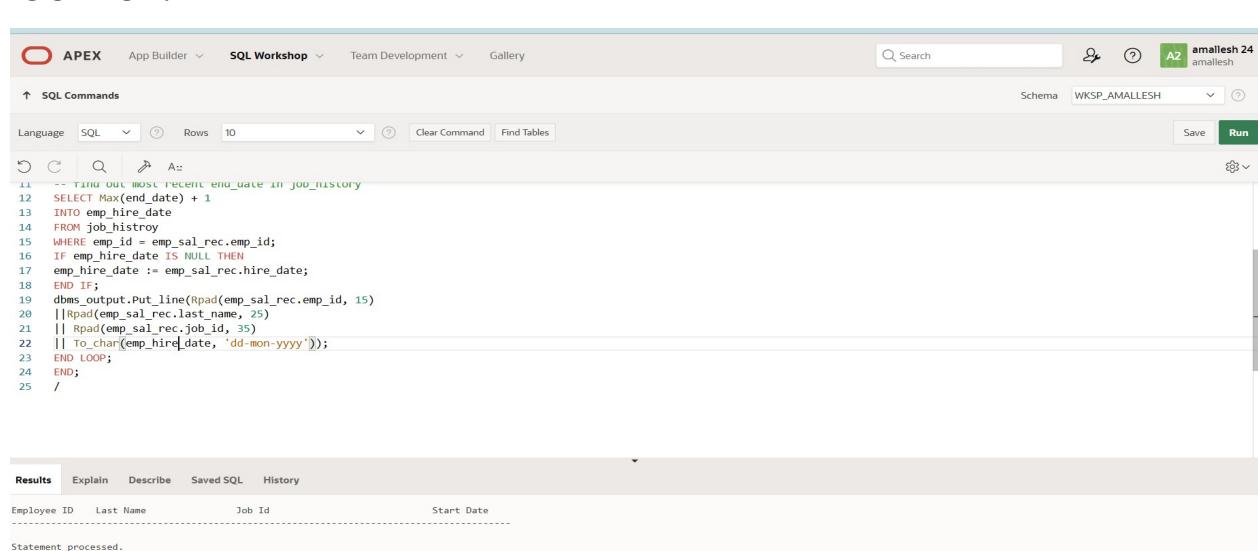
**QUERY:**

DECLARE

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

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

**/OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'amallesh 24 amallesh'. The main area has tabs for SQL Commands, Explain, Describe, Saved SQL, and History. The SQL Commands tab displays the PL/SQL code from the previous step. The Explain, Describe, and History tabs are visible below it. The results section at the bottom shows the output of the executed query.

Employee ID	Last Name	Job Id	Start Date
101	Alfred	SA	1987-06-17
102	Brenda	SA	1989-05-22
103	David	SA	1987-06-26
104	Eduardo	SA	1987-07-03
105	Fiona	SA	1989-05-21
106	Grace	SA	1987-06-20
107	Hany	SA	1989-05-01
108	Juliann	SA	1987-06-27
109	Klaus	SA	1987-07-13
110	Luis	SA	1987-06-26
111	Mart	SA	1987-07-04
112	Nancy	SA	1987-06-20
113	Oscar	SA	1987-06-20
114	Renee	SA	1987-06-20
115	Susan	SA	1987-06-20

Statement processed.

# PROCEDURES AND FUNCTIONS

EX\_NO: 17

DATE:

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'amallesh 24' and a workspace named 'WKSP\_AMALLESH'. The main area is titled 'SQL Commands' and contains a code editor with the following PL/SQL block:

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

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

```
120
Statement processed.
0.00 seconds
```

A notification suggestion in the bottom right corner says: "Turn off notifications from ARMOURY CRATE? We noticed you haven't opened these in a while."

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

**QUERY:**

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

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

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

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

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

## OUTPUT:

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

```
11 WHERE book_id = p_book_id;
12 
13   p_title := p_title || ' - Retrieved';
14 EXCEPTION
15   WHEN NO_DATA_FOUND THEN
16     p_title := NULL;
17     p_author := NULL;
18     p_year_published := NULL;
19 END;
20 
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 
```

The code contains several syntax errors, notably missing semicolons and incorrect declarations. A yellow box highlights the error at line 22: "Error at line 22: PLS-00103: Encountered the symbol "DECLARE"".

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

**RESULT:**

# TRIGGER

EX\_NO: 18

DATE:

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below the toolbar, the 'SQL Commands' tab is active. The main area contains the PL/SQL code for the trigger. At the bottom, the 'Results' tab is selected, displaying the message 'Trigger created.' and a execution time of '0.04 seconds'.

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

Results Explain Describe Saved SQL History

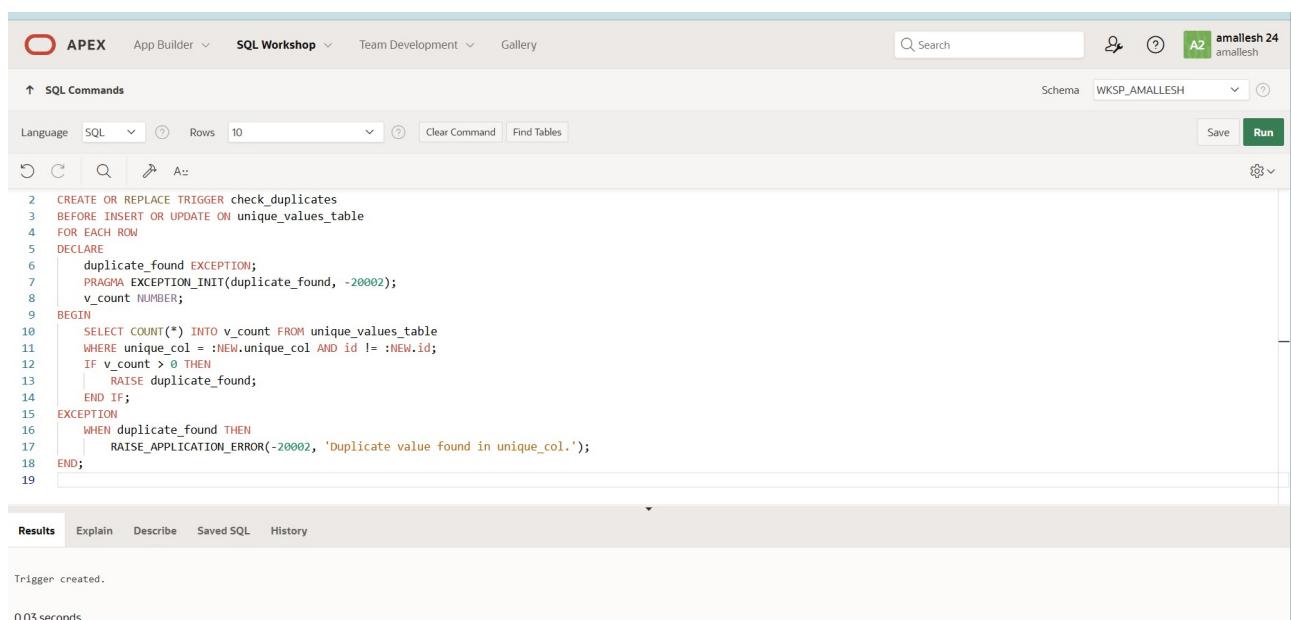
Trigger created.  
0.04 seconds

**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 command is as follows:

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

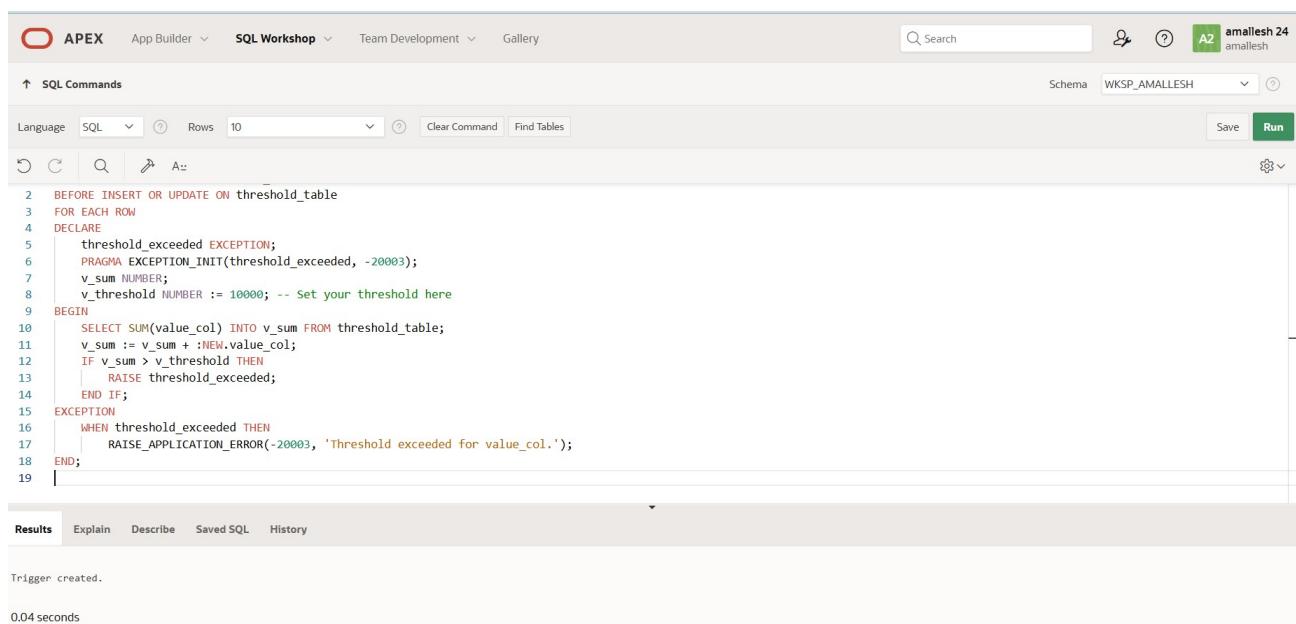
At the bottom of the SQL pane, the status message 'Trigger created.' is visible. Below the pane, the tabs 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History' are shown, with 'Results' being the active tab.

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side of the header shows the user 'amallesh 24 amallesh'. The main workspace is titled 'SQL Commands' and contains the PL/SQL code for the trigger. The code is highlighted with syntax coloring. Below the code, the 'Results' tab is selected, showing the message 'Trigger created.' and '0.04 seconds'.

```
2 BEFORE INSERT OR UPDATE ON threshold_table
3 FOR EACH ROW
4 DECLARE
5     threshold_exceeded EXCEPTION;
6     PRAGMA EXCEPTION_INIT(threshold_exceeded, -20003);
7     v_sum NUMBER;
8     v_threshold NUMBER := 10000; -- Set your threshold here
9 BEGIN
10    SELECT SUM(value_col) INTO v_sum FROM threshold_table;
11    v_sum := v_sum + :NEW.value_col;
12    IF v_sum > v_threshold THEN
13        RAISE threshold_exceeded;
14    END IF;
15 EXCEPTION
16    WHEN threshold_exceeded THEN
17        RAISE_APPLICATION_ERROR(-20003, 'Threshold exceeded for value_col.');
18 END;
19 |
```

Results Explain Describe Saved SQL History

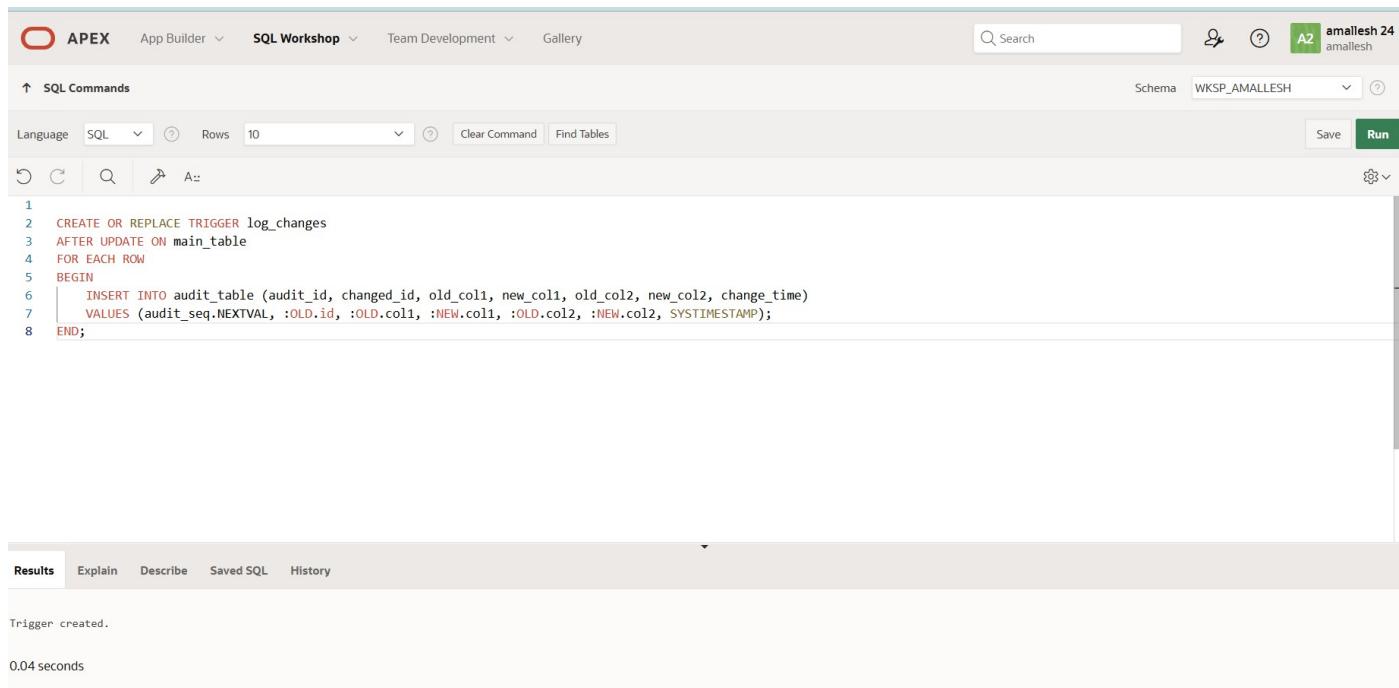
Trigger created.  
0.04 seconds

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

**QUERY:**

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

**OUTPUT:**



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

```
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)
7     VALUES (audit_seq.NEXTVAL, :OLD.id, :OLD.col1, :NEW.col1, :OLD.col2, :NEW.col2,
8 SYSTIMESTAMP);
9 END;
```

In the bottom results pane, the message "Trigger created." is displayed, along with a execution time of "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:**

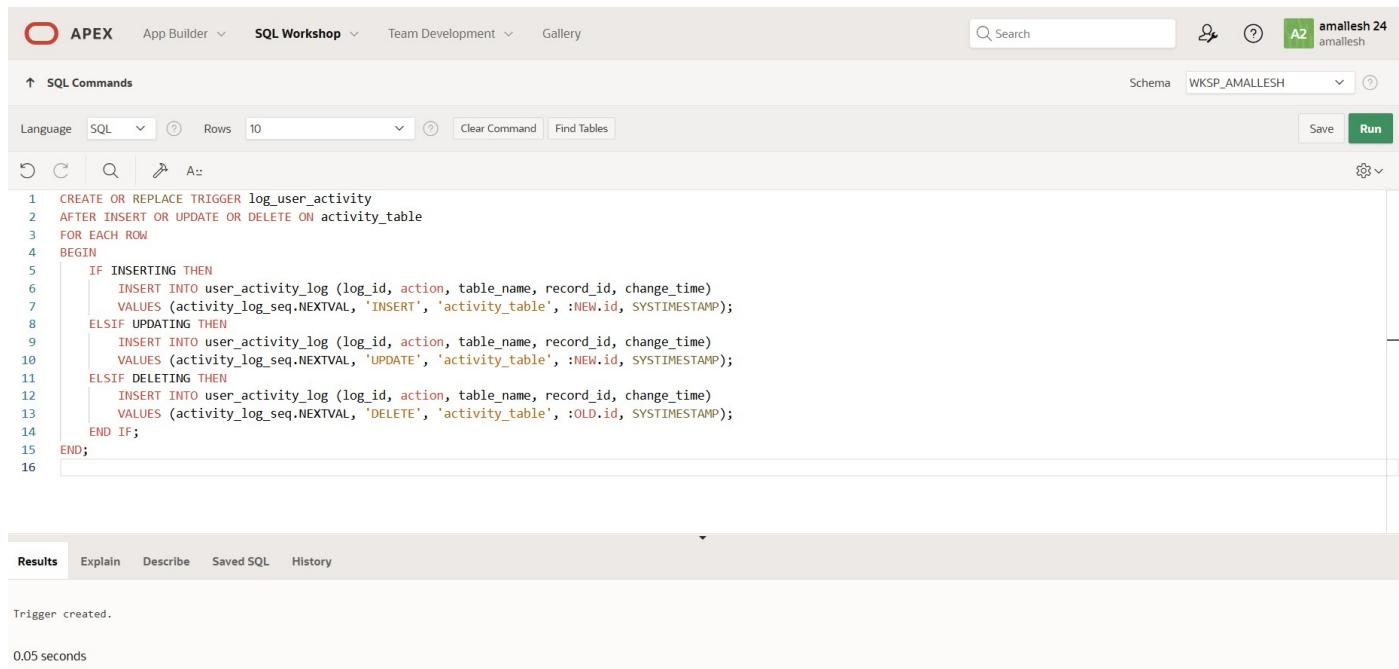
```
CREATE OR REPLACE TRIGGER log_user_activity
AFTER INSERT OR UPDATE OR DELETE ON activity_table
```

```

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 Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. The code entered is a CREATE OR REPLACE TRIGGER statement for 'log\_user\_activity' on the 'activity\_table'. The trigger handles INSERT, UPDATE, and DELETE operations. It inserts a log entry into the 'user\_activity\_log' table with fields: log\_id (auto-generated by activity\_log\_seq), action (either 'INSERT', 'UPDATE', or 'DELETE'), table\_name ('activity\_table'), record\_id (either :NEW.id or :OLD.id), and change\_time (SYSTIMESTAMP). The trigger is named 'log\_user\_activity' and is located in schema 'WKSP\_AMALLESH'. The execution time was 0.05 seconds.

```

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;

```

**Results**

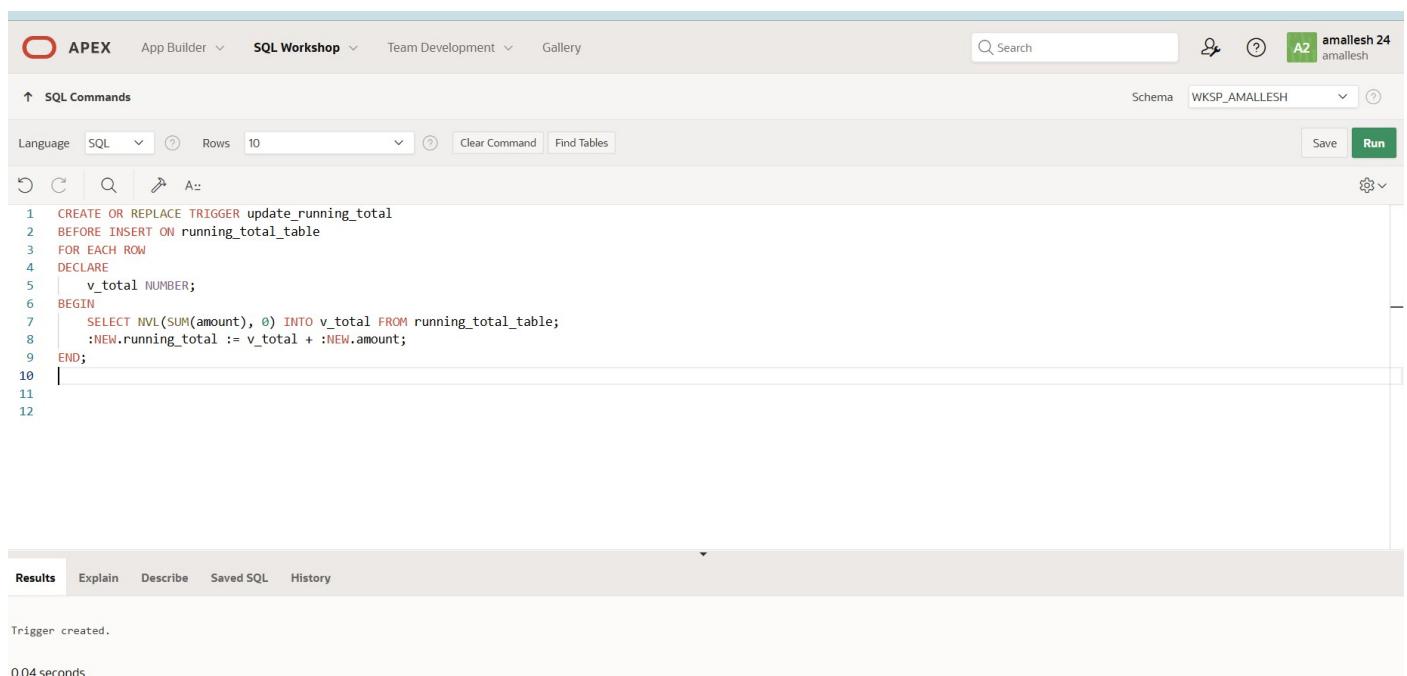
Trigger created.  
0.05 seconds

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

**QUERY:**

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

**OUTPUT:**



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

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

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

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

**QUERY:**

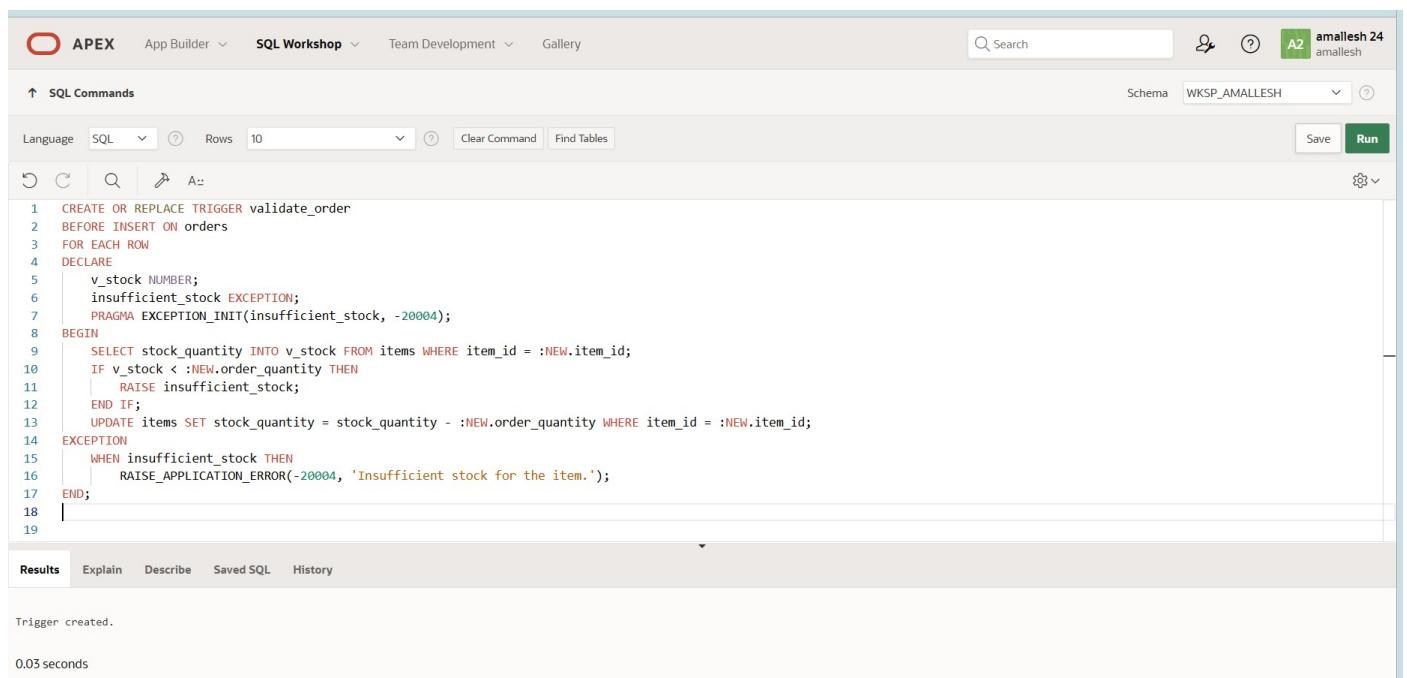
```
CREATE OR REPLACE TRIGGER validate_order
BEFORE INSERT ON orders
FOR EACH ROW
DECLARE
    v_stock NUMBER;
```

```

insufficient_stock EXCEPTION;
PRAGMA EXCEPTION_INIT(insufficient_stock, -20004);
BEGIN
  SELECT stock_quantity INTO v_stock FROM items WHERE item_id = :NEW.item_id;
  IF v_stock < :NEW.order_quantity THEN
    RAISE insufficient_stock;
  END IF;
  UPDATE items SET stock_quantity = stock_quantity - :NEW.order_quantity WHERE item_id
= :NEW.item_id;
EXCEPTION
  WHEN insufficient_stock THEN
    RAISE_APPLICATION_ERROR(-20004, 'Insufficient stock for the item.');
END;

```

## OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a user icon for 'amallesh 24' and a schema dropdown set to 'WKSP\_AMALLESH'. The main area is titled 'SQL Commands' with a language dropdown set to 'SQL'. Below the title are standard toolbar icons for Undo, Redo, Search, Find, and Run. The SQL code area contains the trigger definition shown above. The code is numbered from 1 to 19. The 'Run' button is visible at the bottom right of the code editor. Below the code, the results tab is selected, showing the message 'Trigger created.' and a execution time of '0.03 seconds'.

```

1 CREATE OR REPLACE TRIGGER validate_order
2 BEFORE INSERT ON orders
3 FOR EACH ROW
4 DECLARE
5   v_stock NUMBER;
6   insufficient_stock EXCEPTION;
7   PRAGMA EXCEPTION_INIT(insufficient_stock, -20004);
8 BEGIN
9   SELECT stock_quantity INTO v_stock FROM items WHERE item_id = :NEW.item_id;
10  IF v_stock < :NEW.order_quantity THEN
11    RAISE insufficient_stock;
12  END IF;
13  UPDATE items SET stock_quantity = stock_quantity - :NEW.order_quantity WHERE item_id = :NEW.item_id;
14 EXCEPTION
15  WHEN insufficient_stock THEN
16    RAISE_APPLICATION_ERROR(-20004, 'Insufficient stock for the item.');
17 END;
18

```

Results Explain Describe Saved SQL History

Trigger created.

0.03 seconds

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

**RESULT:**

# MONGO DB

**EX\_NO: 19**

**DATE:**

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

**QUERY:**

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

**OUTPUT:**

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 "Save" button. On the right, there are "Run" and "Save" buttons. The main area contains a command line and its output. The command is:

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

The output shows the results of the query:

```
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:**

The screenshot shows a MongoDB query editor interface. At the top, there's a navigation bar with a logo, "myCompiler", "English", "Recent", "Login", and "Sign up". Below the bar is a search input field with placeholder text "Enter a title...". Underneath the search field are two buttons: "MongoDB" with a dropdown arrow and a help icon. To the right are "Run" and "Save" buttons. A code input area contains the following MongoDB query:

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

To the right of the code input is a "Output" panel. It displays the command prompt "mycompiler\_mongodb>" followed by "mycompiler\_mongodb>". Below that is the message "[Execution complete with exit code 0]".

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

#### QUERY:

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

#### OUTPUT:

The screenshot shows the same MongoDB query editor interface as the first one. The code input area contains the query from step 3. The output panel shows the command prompt "mycompiler\_mongodb>" followed by "mycompiler\_mongodb>". Below that is the message "[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 shell interface. At the top, there is a search bar labeled "Enter a title..." and a toolbar with a MongoDB icon, a dropdown menu, and two buttons: "Run" and "Save". Below the toolbar, the command entered is:  
1`{\$\_and : [{"address.coord.1": {\$gt : 42}}, {"address.coord.1": {\$lte : 52}}]}, {"\_id:0, restaurant\_id:1, name:1, address:1}` Output

The output window shows the results of the query execution:  
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:**

Enter a title...

MongoDB ▾ i

Ctrl+Enter Run Save

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

**Output**

```
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:**

Enter a title...

MongoDB ▾ i

Ctrl+Enter Run Save

```
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]
```

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. At the top, there is a search bar with placeholder text "Enter a title...". Below the search bar are two buttons: "MongoDB" with a dropdown arrow and a help icon. To the right are "Run" and "Save" buttons. The main area contains a code input field with the following command:

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

Below the code input is a "Output" panel. It displays the command and its execution results:

```
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. At the top, there is a search bar with placeholder text "Enter a title...". Below the search bar are two buttons: "MongoDB" with a dropdown arrow and a help icon. To the right are "Run" and "Save" buttons. The main area contains a code input field with the following command:

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

Below the code input is an "Output" panel. It displays the command and its execution results:

```
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 query editor interface. On the left, there is a search bar labeled "Enter a title..." and a toolbar with a MongoDB icon, a dropdown menu, and two buttons for "Run" and "Save". Below the toolbar, a code input field contains the MongoDB query: `db.restaurants.find({ "address.coord": { $elemMatch: { $type: "double" } } })`. To the right of the code input is a panel titled "Output" which displays the command prompt "mycompiler\_mongodb>" followed by "[Execution complete with exit code 0]".

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

**QUERY:**

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

**OUTPUT:**

Enter a title...

MongoDB ▾



Run

Save

```
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

Save

```
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 search bar labeled "Enter a title..." and a dropdown menu set to "MongoDB". Below the search bar is a code editor containing the following query:

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

On the right, there is a "Run" button and a "Save" button. Below the code editor is a "Output" panel with the following content:

```
mycompiler_mongodb>
mycompiler_mongodb>
```

[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 ▾



▶ Run

Save

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

#### 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 ▾



▶ Run

Save

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

#### 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 query editor interface. At the top, there is a search bar labeled "Enter a title..." and a toolbar with a MongoDB icon, a dropdown menu, and two buttons: "Run" and "Save". Below the toolbar, the query code is displayed in three numbered lines:

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

Line 1 starts with "mycompiler\_mongodb>" followed by "mycompiler\_mongodb>". Line 2 starts with "[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 Save
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 Save
1{ $lt: 5 } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $min: ["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. At the top, there is a search bar labeled "Enter a title...". Below it, a dropdown menu says "MongoDB" with a "Save" button next to it. On the right, there are "Run" and "Save" buttons. The main area contains a command line with the following text:  
1 db.restaurants.find({ \$and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }] })  
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]  
The output panel has a vertical scrollbar on its right side.

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 Save  
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 Save  
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 shell interface. On the left, there is a search bar with placeholder text 'Enter a title...'. Below it are two dropdown menus: one labeled 'MongoDB' with a dropdown arrow, and another with a refresh icon. To the right are two buttons: a green 'Run' button and a blue 'Save' button. The main area contains a code input field with the following MongoDB query:

```
1 'A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $ne: "American" } })
```

Next to the input field is a 'Output' button. To the right of the output button is a text area showing the results of the query execution:

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

A vertical scrollbar is visible on the far right of the interface.

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:

Enter a title...

MongoDB ▾ ?

Run Save

```
1:db.core.find({ "core": 6 }), $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $in: ["American", "Chinese"] }}})
```

Output

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

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

#### QUERY:

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

#### OUTPUT:

Enter a title...

MongoDB ▾ ?

Run Save

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

Output

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

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

**RESULT:**

# 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 a search bar labeled "Enter a title...". Below it are two buttons: "MongoDB" with a dropdown arrow and a help icon. On the right, there are two buttons: "Run" (green) and "Save" (blue). The main area contains a command line with the text "1 db.movies.find({ year: 1893 })". To the right, under the heading "Output", the response is shown: "mycompiler\_mongodb>" followed by "[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 a search bar labeled "Enter a title...". Below it are two buttons: "MongoDB" with a dropdown arrow and a help icon. On the right, there are two buttons: "Run" (green) and "Save" (blue). The main area contains a command line with the text "1 db.movies.find({ runtime: { \$gt: 120 } })". To the right, under the heading "Output", the response is shown: "mycompiler\_mongodb>" followed by "[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..." and a dropdown menu set to "MongoDB". Below the search bar is a code input field containing the query: "db.movies.find({ genres: 'Short' })". To the right of the code input are two buttons: "Run" (green) and "Save" (blue). On the far right, there is an "Output" panel. The output panel displays the results of the query execution. It shows the prompt "mycompiler\_mongodb>" followed by the command itself, and the message "[Execution complete with exit code 0]".

```
Enter a title...
MongoDB ▾ ⓘ
1 db.movies.find({ genres: 'Short' })
mycompiler_mongodb>
mycompiler_mongodb>
[Execution complete with exit code 0]
```

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

**QUERY:**

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

**OUTPUT:**

Enter a title...

MongoDB ▾ Ctrl+Enter

Run Save

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

Output

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

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

**QUERY:**

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

**OUTPUT:**

Enter a title...

MongoDB ▾ Ctrl+S

Run Save

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

Output

```
mycompiler_mongodb>
mycompiler_mongodb>
[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. At the top, there is a search bar with placeholder text "Enter a title...". Below the search bar are three buttons: a green MongoDB icon, a dropdown menu labeled "MongoDB", and an information icon. To the right are two buttons: a green "Run" button and a blue "Save" button. The main area contains a code input field with the following command:

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

On the right side, there is a "Output" panel. It displays the results of the query, which are empty. The output text is as follows:

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

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a MongoDB shell interface. At the top, there is a search bar with placeholder text "Enter a title...". Below the search bar are three buttons: a green MongoDB icon, a dropdown menu labeled "MongoDB", and an information icon. To the right are two buttons: a green "Run" button and a blue "Save" button. The main area contains a code input field with the following command:

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

On the right side, there is a "Output" panel. It displays the results of the query, which are empty. The output text is as follows:

```
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:**

The screenshot shows a MongoDB shell interface. On the left, there is a search bar labeled "Enter a title...". Below it, a dropdown menu says "MongoDB" with a dropdown arrow, and a small info icon. To the right are two buttons: "Run" (green) and "Save" (blue). In the main area, a command is typed into the shell:

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

On the right, under the heading "Output", the results of the query are displayed:

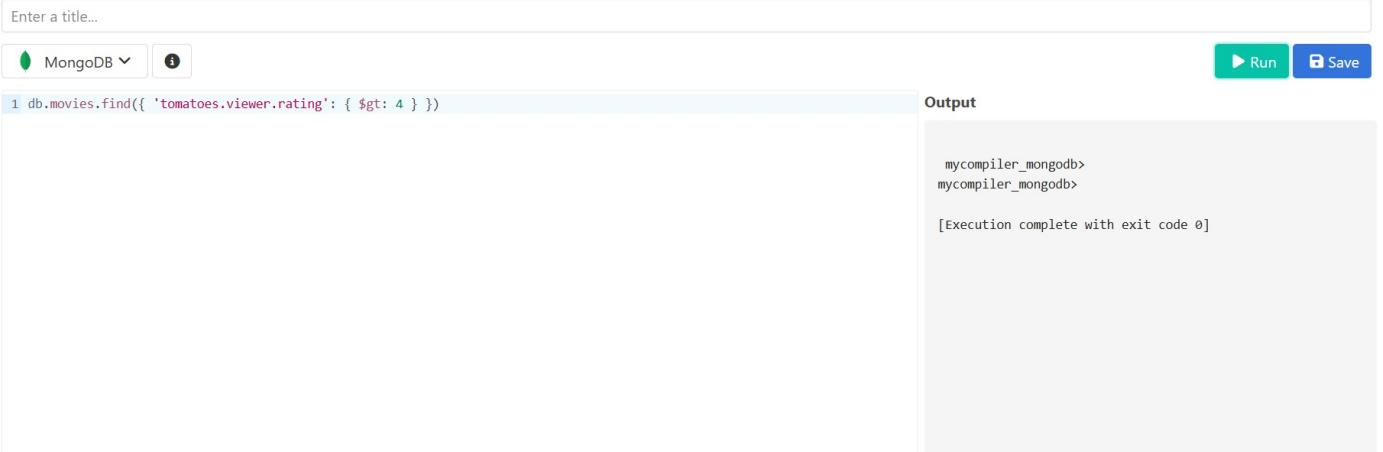
```
mycompiler_mongodb>
mycompiler_mongodb>
[Execution complete with exit code 0]
```

**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:**



Enter a title...

MongoDB ▾

Run Save

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

Output

```
mycompiler_mongodb>
mycompiler_mongodb>

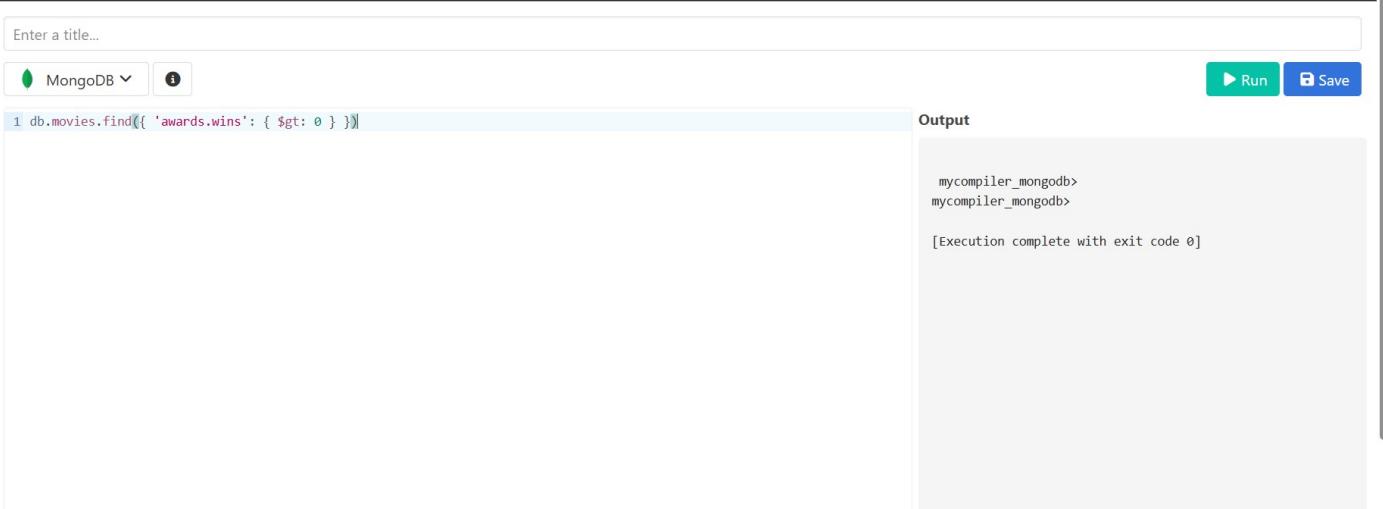
[Execution complete with exit code 0]
```

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

**QUERY:**

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

**OUTPUT:**



Enter a title...

MongoDB ▾

Run Save

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

Output

```
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 query interface. At the top, there is a search bar labeled "Enter a title...". Below it are two buttons: "MongoDB" with a dropdown arrow and a help icon. To the right are "Run" and "Save" buttons. The main area displays the query results:

```
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]

Explore design possibilities with Figma. Build prototypes, and easily translate your work into code.

**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 ▾ Info

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

Run Save

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...

MongoDB ▾ Info

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

Run Save

Output

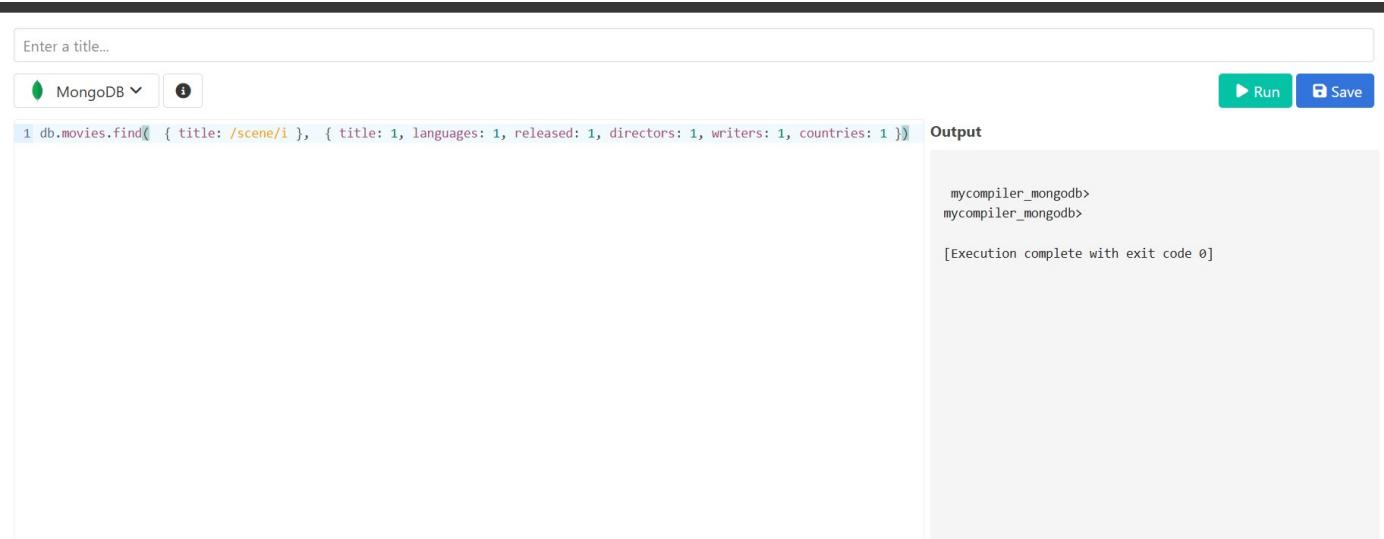
```
mycompiler_mongodb>
mycompiler_mongodb>
[Execution complete with exit code 0]
```

**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 labeled "Enter a title...". Below it, a dropdown menu says "MongoDB" with a "Save" button next to it. On the right, there are "Run" and "Save" buttons. In the center, a code editor window contains the command: `1 db.movies.find( { title: /scene/i }, { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 } )`. To the right of the code editor is an "Output" panel. The output shows the results of the query: `mycompiler_mongodb>`, `mycompiler_mongodb>`, and [Execution complete with exit code 0].

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

Viva(5)	
Total (15)	
Faculty Signature	

**RESULT:**