

# Creating and Managing Tables

EX\_NO:1

DATE:

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

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

## QUERY:

```
Create table dept(id number(7) not null,name varchar2(25));
```

## 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, along with a search bar and user profile information. The main workspace is titled "SQL Commands" and shows the schema "WKSP\_KAVI2005". The SQL editor contains the command to create the "dept" table with columns "id" (number(7) not null) and "name" (varchar2(25)). The "Run" button is visible at the bottom right of the editor. Below the editor, the "Results" tab is selected, displaying the message "Table created." and a execution time of "0.04 seconds". The footer of the page shows copyright information for Oracle and the APEX version "23.2.4".

```
create table dept(deptid number(7) not null, name varchar2(25));
```

Table created.  
0.04 seconds

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

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

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

#### QUERY:

```
Create table emp(id number(7),Last_Name varchar(25),First_Name varchar(25),Deptid number(7));
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. The schema is set to 'WKSP\_KAVI2005'. The code input field contains the SQL command to create the EMP table:

```
1 create table emp(id number(7), lastname varchar(25), firstname varchar(25), deptid number(7));
```

The results section shows the output of the command:

```
Table created.
```

Execution time is listed as 0.03 seconds.

At the bottom, the footer includes user information (kavi@2005) and copyright information (Copyright © 1999, 2023, Oracle and/or its affiliates). The version is Oracle APEX 23.2.4.

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

#### QUERY:

```
Alter table emp modify(Lastname varchar(50));
```

## 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, along with a search bar and user profile. The SQL Workshop tab is selected. The schema dropdown is set to WKSP\_KAVI2005. The main area displays the following SQL command:

```
1 alter table emp modify(lastname varchar(50));|
```

Below the command, the results section shows the output:

```
Table altered.
```

Execution time: 0.05 seconds

At the bottom, the footer includes copyright information and the version Oracle APEX 23.2.4.

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

## QUERY:

Create table employee2(employeeid number(7) not null,Firstname varchar2(25),Lastname varchar2(25),Salary int,Deptid number(7)not null);

## 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, along with a search bar and user profile. The SQL Workshop tab is selected. The schema dropdown is set to WKSP\_KAVI2005. The main area displays the following SQL command:

```
1 create table employees2(employeeid number(6) not null,
2 firstname varchar(25), lastname varchar(25),
3 salary number(10), deptid number(6)not null);|
```

Below the command, the results section shows the output:

```
Table created.
```

Execution time: 0.04 seconds

At the bottom, the footer includes copyright information and the version Oracle APEX 23.2.4.

5. Drop the EMP table.

**QUERY:**

```
Drop table emp;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. The schema is set to 'WKSP\_KAVI2005'. The SQL editor contains the command: 'drop table emp;'. The results tab shows the output: 'Table dropped.' and a execution time of '0.06 seconds'. The bottom footer includes copyright information for Oracle and the APEX version 'Oracle APEX 23.2.4'.

```
drop table emp;
```

Table dropped.  
0.06 seconds

6. Rename the EMPLOYEES2 table as EMP.

**QUERY:**

```
Rename employees2 to emp;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. The schema is set to 'WKSP\_KAVI2005'. The SQL editor contains the command: 'rename employees2 to emp;'. The results tab shows the output: 'Statement processed.' and a execution time of '0.05 seconds'. The bottom footer includes copyright information for Oracle and the APEX version 'Oracle APEX 23.2.4'.

```
rename employees2 to emp;
```

Statement processed.  
0.05 seconds

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

**QUERY:**

comment on table dept is 'Department info';

comment on table emp is Employee info';

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and user information 'KA Kavi arasu kavi@2005'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_KAVI2005'. Below the title are buttons for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. On the right are 'Save' and 'Run' buttons. The SQL editor contains the command: '1 DESCRIBE dept; 2'. The results tab is selected, showing the following table description:

Object Type		TABLE	Object	DEPT					
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPT	DEPTID	NUMBER	-	7	0	-	-	-	-
	NAME	VARCHAR2	25	-	-	-	✓	-	-

At the bottom, the footer displays user information '220701517@rajalakshmi.edu.in kavi@2005 en', copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.', and version 'Oracle APEX 23.2.4'.

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

**QUERY:**

Alter table emp drop column firstname;

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar is identical to the previous screenshot. The SQL editor contains the command: '1 alter table emp drop column firstname;'. The results tab is selected, displaying the output: 'Table altered.' Below the results, the text '0.07 seconds' is shown. The footer information and version are the same as the previous screenshot.

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

**RESULT:**

# MANIPULATING DATA

EX\_NO:2

DATE:

1.Create MY\_EMPLOYEE table with the following structure

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

QUERY:

Create table myemployee(id number(4) not null ,lastname varchar(25),firstname varchar(25),userid varchar(25),salary number(9,2));

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, along with a search bar and user profile information. The main workspace is titled "SQL Commands" and contains the following SQL code:

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

Below the code, the "Results" tab is selected, displaying the output: "Table created." and "0.04 seconds". The bottom footer includes copyright information for Oracle and the APEX version.

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

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

### QUERY:

**Insert into myemployee values(2,'dancs','betty','bdancs',860);**

### 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, along with a search bar and user profile information. The main workspace is titled "SQL Commands" and contains the following SQL code:

```
1 insert into myemployee
2 values(2, 'dancs', 'betty', 'bdancs', 860);
```

Below the code, the "Results" tab is selected, showing the output of the query:

1 row(s) inserted.  
0.00 seconds

At the bottom of the screen, footer information includes the user's email (220701517@rajalakshmi.edu.in), session ID (kavi@2005), language (en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the APEX version (Oracle APEX 23.2.4).

3. Display the table with values.

### QUERY:

**Select \* from myemployee;**

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for App Builder, SQL Workshop, Team Development, and Gallery, along with a search bar and user authentication information (KA Kavi arasu kavi@2005). The SQL Commands tab is active, displaying the following SQL command:

```
1 select * from myemployee;
```

The Results tab shows the output of the query:

ID	LASTNAME	FIRSTNAME	USERID	SALARY
1	patel	ralph	rpatel	895
2	dancs	betty	bdancs	860

Below the table, it says "2 rows returned in 0.02 seconds". The bottom of the screen displays copyright information and the Oracle APEX version (Oracle APEX 23.2.4).

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 myemployee values (4,'newman','chad','cnewman',860);**

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for App Builder, SQL Workshop, Team Development, and Gallery, along with a search bar and user authentication information (KA Kavi arasu kavi@2005). The SQL Commands tab is active, displaying the following SQL command:

```
1 select * from myemployee;
```

The Results tab shows the output of the query, including the newly inserted row:

ID	LASTNAME	FIRSTNAME	USERID	SALARY
1	patel	ralph	rpatel	895
2	dancs	betty	bdancs	860
4	newman	chad	cnewman	750
3	biri	ben	bbiri	1100

Below the table, it says "2 rows returned in 0.02 seconds". The bottom of the screen displays copyright information and the Oracle APEX version (Oracle APEX 23.2.4).

5. Make the data additions permanent.

#### QUERY:

Select \* from myemployee;

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', 'Gallery', a search bar, and a user profile for 'Kavi arasu kavi@2005'. The 'SQL Commands' tab is selected. The SQL editor contains the command: 'select \* from myemployee;'. The results section displays a table with the following data:

ID	LASTNAME	FIRSTNAME	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	arophebur	1550

Below the table, it says '5 rows returned in 0.01 seconds'. The bottom status bar shows the session ID '220701517@rajalakshmi.edu.in', the schema 'kavi@2005', and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' The version 'Oracle APEX 23.2.4' is also visible.

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

#### QUERY:

UPDATE myemployee SET lastname ='drexler' where ID=3;

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', 'Gallery', a search bar, and a user profile for 'KA'. The 'SQL Commands' tab is selected. The SQL editor contains the command: 'UPDATE myemployee SET lastname = 'drexler' WHERE id = 3;'. The results section displays the message '1 row(s) updated.'

Below the message, the bottom status bar shows the session ID '220701517@rajalakshmi.edu.in', the schema 'kavi@2005', and the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' The version 'Oracle APEX 23.2.4' is also visible.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, Gallery, and a search bar. The SQL Workshop tab is active. The schema dropdown is set to 'WKSP\_KAVI2005'. The main area displays the following SQL command:

```
1 UPDATE myemployee
2 SET salary = 1000
3 WHERE salary < 900;
```

Below the command, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, showing the message: "3 row(s) updated." At the bottom, the user information is shown as '220701517@rajalakshmi.edu.in' and 'kavi@2005', along with copyright information and the version 'Oracle APEX 23.2.4'.

8. Delete Betty from MY\_EMPLOYEE table.

**QUERY:**

**Delete from myemployee where firstname='Betty';**

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, Gallery, and a search bar. The SQL Workshop tab is active. The schema dropdown is set to 'WKSP\_KAVI2005'. The main area displays the following SQL command:

```
1 DELETE FROM myemployee
2 WHERE firstname='betty';
3
```

Below the command, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, showing the message: "1 row(s) deleted." At the bottom, the user information is shown as '220701517@rajalakshmi.edu.in' and 'kavi@2005', along with copyright information and the version 'Oracle APEX 23.2.4'.

9. Empty the fourth row of the emp table.

### QUERY:

Delete from emp where employeeid=10004;

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, Gallery, and a search bar. The schema is set to WKSP\_KAVI2005. The main area displays the following SQL command:

```
1 delete from emp
2 where employeeid=10004;
```

Below the command, the Results tab is selected, showing the output: "1 row(s) deleted." The bottom of the screen shows user information (220701517@rajalakshmi.edu.in, kavi@2005), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the version (Oracle APEX 23.2.4).

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

### RESULT:

# INCLUDING CONSTRAINTS

**EX\_NO:3**

**DATE:**

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

**QUERY:**

**Create table emp**

**Add constraint my\_emp\_id\_pk primary key(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, along with a search bar and user profile information. The main workspace is titled 'SQL Commands' and shows the schema 'WKSP\_KAVI2005'. The SQL editor contains the following code:

```
1 alter table emp
2 add constraint my_emp_id_pk primary key (employeeid);
```

The 'Results' tab is selected, displaying the output: 'Table altered.' Below the results, it shows '0.07 seconds' for the execution time. At the bottom of the page, there are footer links for user 220701517@rajalakshmi.edu.in, kavi@2005, and en, along with copyright information for Oracle APEX 23.2.4.

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

**QUERY:**

**Alter table dept**

**Add constraint my\_dept\_id\_pk primary key(deptid);**

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. A search bar and user profile 'Kavi arasu kavi@2005' are also present. The main area is titled 'SQL Commands' and shows the following SQL code:

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

The 'Results' tab is selected, displaying the output:

```
Table altered.
```

Execution time: 0.07 seconds.

At the bottom, the footer includes user information (220701517@rajalakshmi.edu.in, kavi@2005, en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates), and the version (Oracle APEX 23.2.4).

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

### QUERY:

**Alter table emp add constraint my\_emp\_dept\_id\_fk foreign key(deptid) references emp(employeeid);**

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. A search bar and user profile 'Kavi arasu kavi@2005' are also present. The main area is titled 'SQL Commands' and shows the following SQL code:

```
1 Alter table emp add constraint my_emp_dept_id_fk foreign key(deptid) references emp(employeeid);
```

The 'Results' tab is selected, displaying the output:

```
Table altered.
```

Execution time: 0.06 seconds.

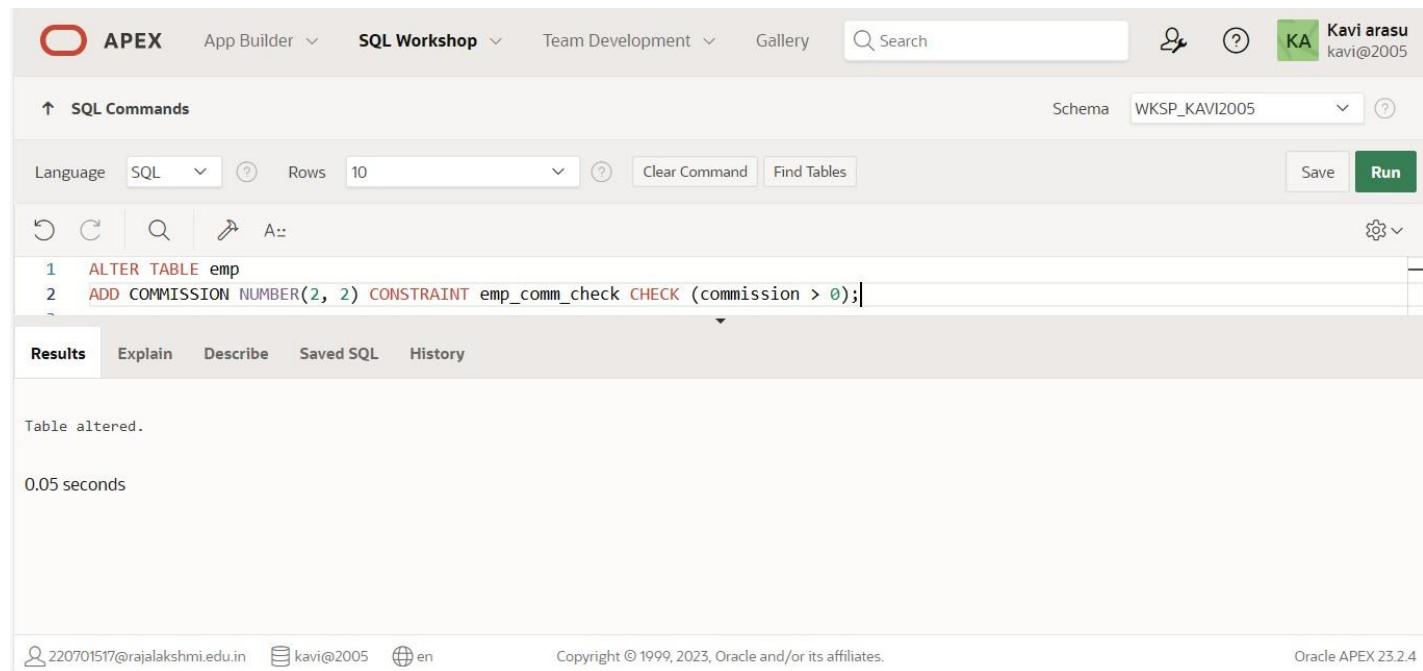
At the bottom, the footer includes user information (220701517@rajalakshmi.edu.in, kavi@2005, en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates), and the version (Oracle APEX 23.2.4).

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

### QUERY:

**ALTER TABLE emp**

**ADD COMMISSION NUMBER(2, 2) CONSTRAINT emp\_comm\_check CHECK (commission > 0);**  
**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, along with a search bar and user profile information for 'Kavi arasu' (kavi@2005). The main workspace is titled 'SQL Commands' and contains the following SQL code:

```
1 ALTER TABLE emp
2 ADD COMMISSION NUMBER(2, 2) CONSTRAINT emp_comm_check CHECK (commission > 0);
```

Below the code, the 'Results' tab is selected, showing the output: 'Table altered.' and '0.05 seconds'. The bottom footer displays copyright information for Oracle and the APEX version: 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

**RESULT:**

# Writing Basic SQL SELECT Statements

EX\_NO:4

DATE:

1. The following statement executes successfully.

## Identify the Errors

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

## QUERY:

```
SELECT employeeid,lastname ,Salary*12 as Annual salary  
From employees;
```

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

## QUERY:

```
Describe Dept;
```

## 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, along with a search bar and user information (Kavi arasu, kavi@2005). The main workspace is titled 'SQL Commands' and contains the SQL command '1 Describe dept;'. Below the command, the 'Describe' tab is selected in the results panel, which displays the structure of the 'DEPT' table. The table has four columns: DEPTID (NUMBER), DEPTNAME (VARCHAR2(25)), MANAGERID (NUMBER), and LOCATIONID (NUMBER). The 'Primary Key' column shows '1' for DEPTID and '✓' for DEPTNAME, MANAGERID, and LOCATIONID. The 'Nullable' column shows '-' for DEPTID and '✓' for the other three columns. The 'Default' and 'Comment' columns show '-' for all columns. At the bottom of the page, there are footer links for copyright information and Oracle APEX version 23.2.4.

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 employeeid,lastname,jobid,hiredate from employees;
```

## 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, along with a search bar and user authentication information (Kavi arasu, kavi@2005). The SQL Workshop tab is selected. The schema is set to WKSP\_KAVI2005. The SQL command window contains the following query:

```
1 Select EmployeeID, lastname, jobid, HireDate from employees;
```

The results section displays the following table:

EMPLOYEEID	LASTNAME	JOBID	HIREDATE
10003	Johnson	MARKETING	2021-12-31
10004	Lee	HRREP	2024-02-15
10005	Brown	FINANCE	2023-07-05
10006	Garcia	ITSUPPORT	2022-05-20
10007	Miller	SALES	2023-11-10

Below the table, it says "5 rows returned in 0.00 seconds". The bottom of the screen shows copyright information and the version Oracle APEX 23.2.4.

4. Provide an alias STARTDATE for the hire date.

## QUERY:

Select hiredate as StartDate from employees;

## OUTPUT:

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

```
1 Select hiredate as StartDate from employees;
```

The results section displays the following table:

STARTDATE
2021-12-31
2022-05-20
2024-02-15
2023-07-05
2023-11-10

Below the table, it says "5 rows returned in 0.00 seconds". The bottom of the screen shows copyright information and the version Oracle APEX 23.2.4.

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

#### QUERY:

Select Unique jobid from employees;

#### OUTPUT:

The screenshot shows the Oracle APEX interface with the SQL Workshop module selected. The query `select unique jobid from employees;` is entered in the command line. The results show four distinct job IDs: FINANCE, HRREP, SALES, and MARKETING.

JOBID
FINANCE
HRREP
SALES
MARKETING

4 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 lastname || ',' || jobid as Title from employees;

#### OUTPUT:

The screenshot shows the Oracle APEX interface with the SQL Workshop module selected. The query `Select lastname || ',' || jobid as Title from employees;` is entered in the command line. The results show the concatenated string for each employee: Johnson,MARKETING; Garcia,HRREP; Lee,HRREP; Brown,FINANCE; and Miller,SALES.

TITLE
Johnson,MARKETING
Garcia,HRREP
Lee,HRREP
Brown,FINANCE
Miller,SALES

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 employeeid ||','|| lastname ||','|| jobid ||','|| email ||','|| salary ||','|| hiredate as "the\_output" from employees;

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. A search bar and user profile are also present. The main area is titled 'SQL Commands' and shows the following command:

```
1  Select employeeid ||','|| lastname ||','|| jobid ||','|| email ||','|| salary ||','|| hiredate as "the_output" from employees;
```

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

the_output
10003,Johnson,MARKETING,alice.johnson@company.com,65000,2021-12-31
10006,Garcia,HRREP,sarah.garcia@company.com,45000,2022-05-20
10004,Lee,HRREP,david.lee@company.com,55000,2024-02-15
10005,Brown,FINANCE,michael.brown@company.com,90000,2023-07-05
10007,Miller,SALES,williammiller@company.com,10000,2023-11-10

Below the results, it says '5 rows returned in 0.00 seconds' and has a 'Download' link. The bottom of the page includes copyright information and the version 'Oracle APEX 23.2.4'.

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

**RESULT:**

# RESTRICTING AND SORTING DATA

EX\_NO:5

DATE:

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

## QUERY:

```
select lastname,salary from employees  
where salary>12000;
```

## OUTPUT:

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

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

The results table displays the following data:

LASTNAME	SALARY
Johnson	65000
Garcia	45000
Lee	55000
Brown	90000

4 rows returned in 0.02 seconds

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

## QUERY:

```
select lastname,departmentid from employees  
where employeeid = 176;
```

## OUTPUT:

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

```
1 select lastname,departmentid from employees  
2 where employeeid = 176;  
3
```

The results table displays the following data:

LASTNAME	DEPARTMENTID
Lee	4

1 rows returned in 0.01 seconds

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

### QUERY:

```
select lastname, salary from employees  
where salary not between 5000 and 12000;
```

### OUTPUT:

LASTNAME	SALARY
Johnson	65000
Garcia	45000
Brown	90000

3 rows returned in 0.01 seconds    Download

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

### QUERY:

```
select lastname,jobid,hiredate from employees where hiredate between '02-20-1998' and '05-01-1998' order by hiredate asc;
```

### OUTPUT:

LASTNAME	JOBID	HIREDATE
Johnson	MARKETING	02/20/1998
Garcia	HRREP	04/01/1998

2 rows returned in 0.01 seconds    Download

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

### QUERY:

select lastname, departmentid from employees where departmentid in (20,50) order by lastname asc;

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile for 'Kavi arasu' (kavi@2005). The main area has a 'SQL Commands' tab selected. The SQL editor contains the following query:

```
1 select lastname, departmentid from employees where departmentid in (20,50) order by lastname asc;
```

The results section displays the output:

LASTNAME	DEPARTMENTID
Johnson	20
Lee	50

Below the table, it says '2 rows returned in 0.00 seconds' and has a 'Download' link. The bottom status bar shows the user's email (220701517@rajalakshmi.edu.in), the schema (WKSP\_KAVI2005), and the Oracle APEX version (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:

select lastname as employee, salary as empsal from employees where departmentid in(20,50) and salary between 5000 and 12000 order by lastname asc;

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile for 'Kavi arasu' (kavi@2005). The main area has a 'SQL Commands' tab selected. The SQL editor contains the following query:

```
1 select lastname as employee, salary as empsal from employees where departmentid in (20,50) and salary between 5000 and 12000 order by lastname asc;
```

The results section displays the output:

EMPLOYEE	EMPSAL
Lea	6000

Below the table, it says '1 rows returned in 0.00 seconds' and has a 'Download' link. The bottom status bar shows the user's email (220701517@rajalakshmi.edu.in), the schema (WKSP\_KAVI2005), and the Oracle APEX version (23.2.4).

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

#### QUERY:

select lastname, hiredate from employees where hiredate like '%1994';

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and user information 'Kavi arasu kavi@2005'. The 'SQL Commands' tab is active, showing the query: 'select lastname, hiredate from employees where hiredate like '%1994';'. The results section displays one row: Lea, 03/12/1994. The bottom status bar shows copyright information and 'Oracle APEX 23.2.4'.

LASTNAME	HIREDATE
Lea	03/12/1994

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

#### QUERY:

select lastname, jobid from employees where managerid is null;

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', a search bar, and user information 'Kavi arasu kavi@2005'. The 'SQL Commands' tab is active, showing the query: 'select lastname, jobid from employees where managerid is null;'. The results section displays one row: Brown, FINANCE. The bottom status bar shows copyright information and 'Oracle APEX 23.2.4'.

LASTNAME	JOBID
Brown	FINANCE

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

**QUERY:**

select lastname, salary, commissionpct from employees where commissionpct is not null order by salary desc, commissionpct desc;

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', and a search bar. The user is logged in as 'Kavi arasu' with email 'kavi@2005'. The schema is set to 'WKSP\_KAVI2005'. The SQL command entered is:

```
1 select lastname, salary, commissionpct from employees where commissionpct is not null order by salary desc, commissionpct desc;
```

The results section displays the following data:

LASTNAME	SALARY	COMMISSIONPCT
Brown	90000	0
Johnson	65000	.2
Garcia	45000	0
Miller	10000	.15
Lee	6000	0

Below the table, it says '5 rows returned in 0.01 seconds' and has a 'Download' link. The bottom status bar shows the user's email '220701517@rajalakshmi.edu.in', the schema 'kavi@2005', and the language 'en'. It also includes copyright information 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and a note to 'Activate Windows'.

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

**QUERY:**

select lastname from employees where lastname like '\_\_a%';

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', and a search bar. The user is logged in as 'Kavi arasu' with email 'kavi@2005'. The schema is set to 'WKSP\_KAVI2005'. The SQL command entered is:

```
1 select lastname from employees where lastname like '__a%';
```

The results section displays the following data:

LASTNAME
Joan
Lea

Below the table, it says '2 rows returned in 0.01 seconds' and has a 'Download' link. The bottom status bar shows the user's email '220701517@rajalakshmi.edu.in', the schema 'kavi@2005', and the language 'en'. It also includes copyright information 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and a note to 'Activate Windows'.

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

#### QUERY:

```
select lastname from employees where lastname like '%a%' and lastname like '%e%';
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', and a search bar. The right side shows a user profile for 'Kavi arasu' (kavi@2005). The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_KAVI2005'. The SQL editor contains the query: 'select lastname from employees where lastname like '%a%' and lastname like '%e%';'. The results tab is selected, displaying a single row: 'Lea'. Below the results, it says '1 rows returned in 0.01 seconds' and provides a 'Download' link. The bottom footer includes user information (220701517@rajalakshmi.edu.in, kavi@2005, en) and copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates). The version is listed as Oracle APEX 23.2.4.

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

#### QUERY:

```
select lastname, salary, COMMISSIONPCT from employees where COMMISSIONPCT= .2;
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', and a search bar. The right side shows a user profile for 'Kavi arasu' (kavi@2005). The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_KAVI2005'. The SQL editor contains the query: 'select lastname, salary, COMMISSIONPCT from employees where COMMISSIONPCT= .2;'. The results tab is selected, displaying a single row: 'Joan'. Below the results, it says '1 rows returned in 0.04 seconds' and provides a 'Download' link. The bottom footer includes user information (220701517@rajalakshmi.edu.in, kavi@2005, en) and copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates). The version is listed as Oracle APEX 23.2.4.

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

### QUERY:

select lastname as employee, salary as empsal from employees where departmentid in (20,50) and salary between 5000 and 12000 order by lastname asc;

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. A search bar and user information (Kavi arasu, kavi@2005) are also present. The main area is titled "SQL Commands" and contains a SQL editor with the following query:

```
1 select lastname as employee, salary as empsal from employees where departmentid in (20,50) and salary between 5000 and 12000 order by lastname asc;
```

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

EMPLOYEE	EMPSAL
Lea	6000

Below the table, it says "1 rows returned in 0.01 seconds" and has a "Download" link. At the bottom of the page, there are footer links for 220701517@rajalakshmi.edu.in, kavi@2005, en, Copyright © 1999, 2023, Oracle and/or its affiliates, and Oracle APEX 23.2.4.

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

### RESULT:

# SINGLE ROW FUNCTIONS

**EX.NO.6**

**DATE:**

**Find the Solution for the following:**

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

**QUERY:**

**SELECT SYSDATE AS "DATE" FROMDUAL;**

**OUTPUT:**

The screenshot shows the Oracle APEX interface with the SQL Workshop tab selected. The query `SELECT SYSDATE AS "DATE" FROMDUAL;` is entered in the command line. The results pane displays a single row with the column `DATE` containing the value `03/13/2024`. The status bar at the bottom indicates `1 rows returned in 0.02 seconds`.

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

**QUERY:**

**SELECT EMPLOYEEID, LASTNAME, Salary, Salary+(15.5/100\*Salary)**

**"NEW\_SALARY"**

**From Employees;**

**OUTPUT:**

The screenshot shows the Oracle APEX interface with the SQL Workshop tab selected. The query `SELECT EMPLOYEEID, LASTNAME, Salary, Salary+(15.5/100*Salary) "NEW_SALARY" FROM Employees;` is entered in the command line. The results pane displays a table with columns `EMPLOYEEID`, `LASTNAME`, `SALARY`, and `NEW_SALARY`. The data is as follows:

EMPLOYEEID	LASTNAME	SALARY	NEW_SALARY
174	Joan	65000	75075
175	Garcia	45000	51975
180	Hamilton	100000	115500
176	Lea	6000	6930
178	Brown	90000	103950
179	Miller	10000	11550

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 EMPLOYEEID, lastname, Salary, (Salary+(Salary*15.5/100))-Salary  
"Increase"  
From Employees;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. A search bar and user profile 'Kavi arasu kavi@2005' are also present. The main area displays the following SQL command:

```
1 SELECT EMPLOYEEID, lastname, Salary, (Salary+(Salary*15.5/100))-Salary "Increase"  
2 From Employees;
```

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

EMPLOYEEID	LASTNAME	SALARY	Increase
174	Joan	65000	10075
175	Garcia	45000	6975
180	Hamilton	100000	15500
176	Lea	6000	930
178	Brown	90000	13950
179	Miller	10000	1550

At the bottom of the page, there are footer links for copyright information and the Oracle APEX version.

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

**QUERY:**

```
Select initcap(lastname) "Name", length(lastname) "Length of Name" from Employees  
where lastname like 'J%' or lastname like 'A%' or lastname like 'M%' order by lastname;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. A search bar and user profile 'Kavi arasu kavi@2005' are also present. The main area displays the following SQL command:

```
1 Select initcap(lastname) "Name", length(lastname) "Length of Name"  
2 from Employees  
3 where lastname like 'J%' or lastname like 'A%' or lastname like 'M%'  
4 order by lastname;
```

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

Name	Length of Name
Joan	4
Miller	6

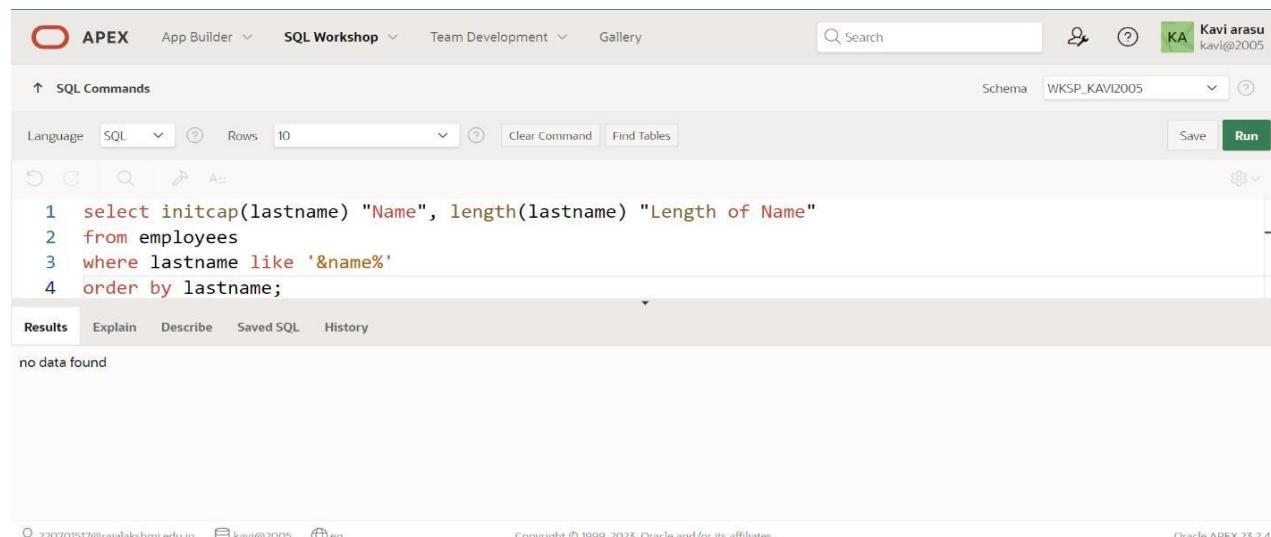
At the bottom of the page, there are footer links for copyright information and the Oracle APEX version.

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

#### QUERY:

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

#### OUTPUT:

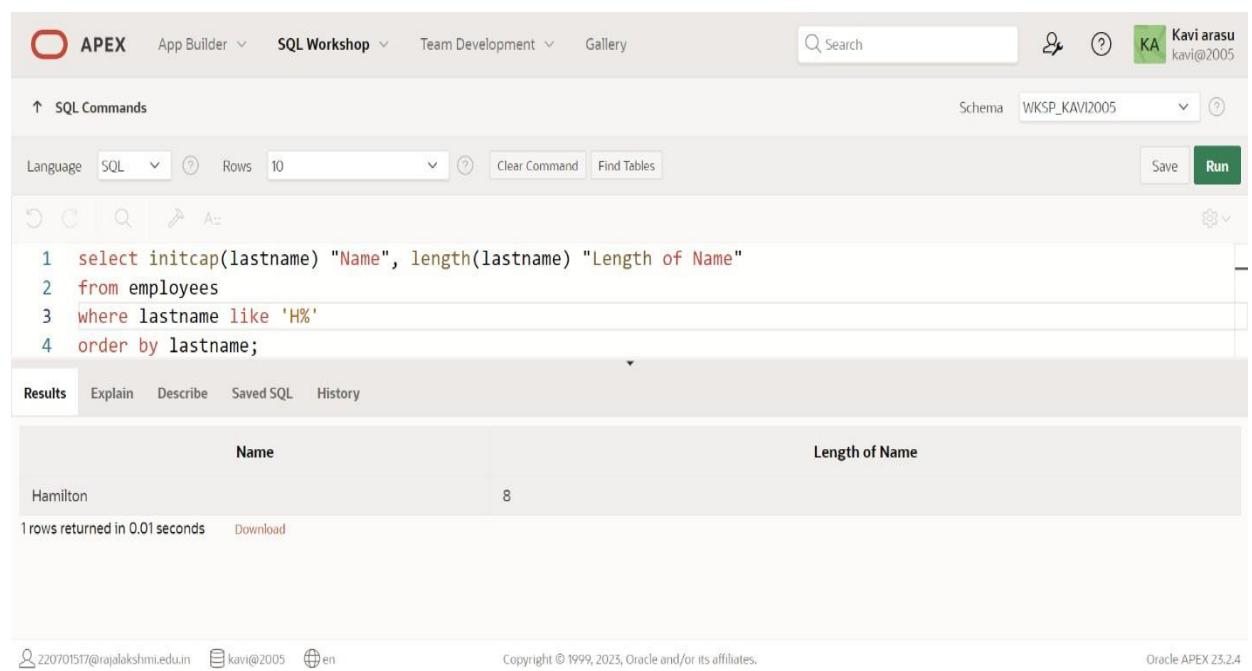


The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile for 'Kavi arasu' at 'kavi@2005'. The main area has tabs for 'SQL Commands' and 'Results'. The SQL command entered is:

```
1 select initcap(lastname) "Name", length(lastname) "Length of Name"  
2 from employees  
3 where lastname like '&name%'  
4 order by lastname;
```

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

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



The screenshot shows the Oracle APEX SQL Workshop interface again. The top navigation bar and user profile are identical. The SQL command now includes a parameter:

```
1 select initcap(lastname) "Name", length(lastname) "Length of Name"  
2 from employees  
3 where lastname like 'H%'  
4 order by lastname;
```

The 'Results' tab is selected, displaying the output:

Name	Length of Name
Hamilton	8

Below the table, it says '1 rows returned in 0.01 seconds' and has a 'Download' link. The bottom of the screen shows copyright information and the version 'Oracle APEX 23.2.4'.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile for 'Kavi arasu kavi@2005'. The main area is titled 'SQL Commands' with a sub-section '1 select lastname, round(months\_between(sysdate,hiredate),0) Months\_worked 2 from employees order by 2;'. Below this, the 'Results' tab is selected, displaying a table with two columns: 'LASTNAME' and 'MONTHS\_WORKED'. The data rows are: Brown (51), Miller (65), Hamilton (113), Garcia (311), Joan (313), and Lea (360). At the bottom, there are footer links for 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

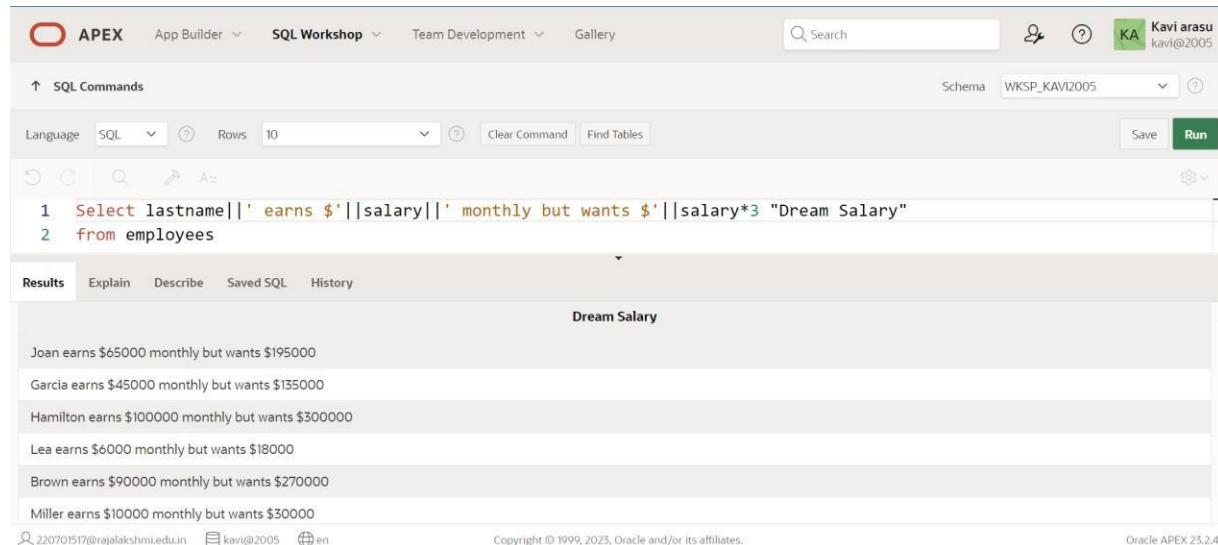
LASTNAME	MONTHS_WORKED
Brown	51
Miller	65
Hamilton	113
Garcia	311
Joan	313
Lea	360

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 lastname||' earns \$'||salary||' monthly but wants \$'||salary\*3 "Dream Salary"  
from employees;

#### OUTPUT:



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

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

The results are displayed in a table with one column labeled "Dream Salary". The output is:

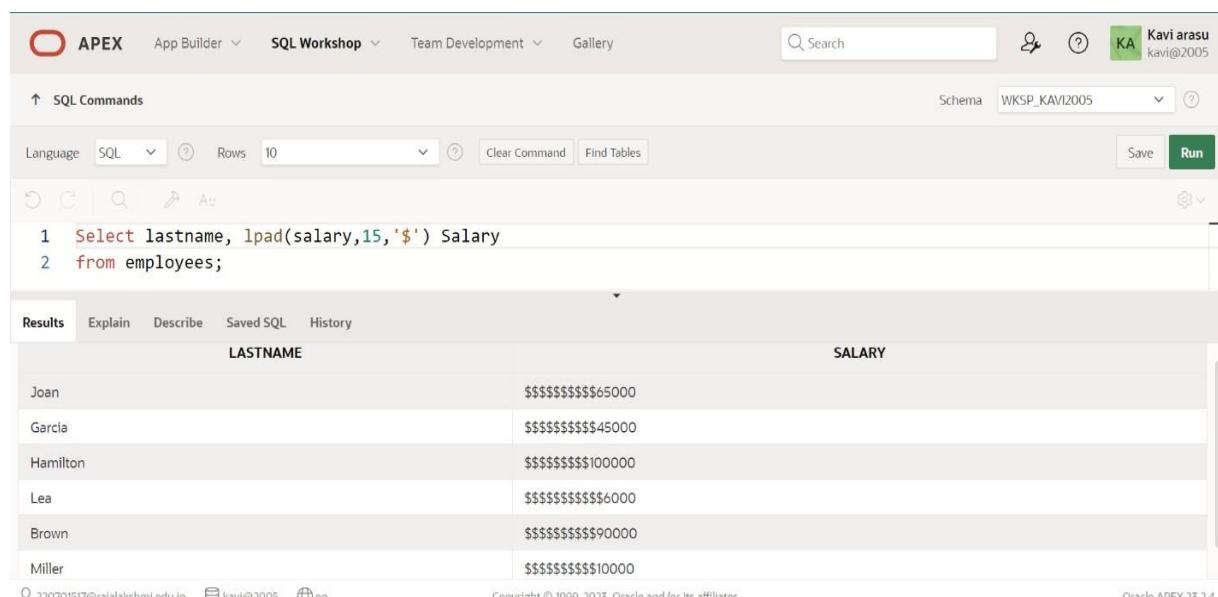
Dream Salary
Joan earns \$65000 monthly but wants \$195000
Garcia earns \$45000 monthly but wants \$135000
Hamilton earns \$100000 monthly but wants \$300000
Lea earns \$6000 monthly but wants \$18000
Brown earns \$90000 monthly but wants \$270000
Miller earns \$10000 monthly but wants \$30000

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

#### QUERY:

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

#### OUTPUT:



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

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

The results are displayed in a table with two columns: "LASTNAME" and "SALARY". The output is:

LASTNAME	SALARY
Joan	\$\$\$\$\$\$\$\$\$\$65000
Garcia	\$\$\$\$\$\$\$\$\$\$45000
Hamilton	\$\$\$\$\$\$\$\$\$\$100000
Lea	\$\$\$\$\$\$\$\$\$\$6000
Brown	\$\$\$\$\$\$\$\$\$\$90000
Miller	\$\$\$\$\$\$\$\$\$\$10000

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX interface with the SQL Workshop tab selected. The query is displayed in the command pane:

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

The results pane displays the following data:

LASTNAME	HIREDATE	REVIEW
Joan	02/20/1998	monday, the twenty-third of february,1998
Garcia	04/01/1998	monday, the sixth of april,1998
Hamilton	10/12/2014	monday, the thirteenth of october,2014
Lea	03/12/1994	monday, the fourteenth of march,1994
Brown	12/05/2019	monday, the ninth of december,2019

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX interface with the SQL Workshop tab selected. The query is displayed in the command pane:

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

The results pane displays the following data:

LASTNAME	HIREDATE	Day
Miller	10/22/2018	Monday
Garcia	04/01/1998	Wednesday
Brown	12/05/2019	Thursday
Joan	02/20/1998	Friday
Lea	03/12/1994	Saturday
Hamilton	10/12/2014	Sunday

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

**RESULT:**

# DISPLAYING DATA FROM MULTIPLE TABLES

EX\_NO:7

DATE:

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

**QUERY:**

```
SELECT e.lastname, e.deptid, d.deptname FROM employees e, dept d  
WHERE e.deptid = d.deptid;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', and a search bar. The right side shows a user profile for 'Kavi arasu' (kavi@2005). The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_KAVI2005'. The SQL editor contains the following query:

```
1 SELECT e.lastname, e.deptid, d.deptname  
2   FROM employees e, dept d  
3  WHERE e.deptid = d.deptid;  
4
```

The results tab is selected, displaying the output:

LASTNAME	DEPTID	DEPTNAME
Joan	100	Marketing
Garcia	100	Marketing
Miller	200	Sales

Below the results, it says '3 rows returned in 0.03 seconds' and provides a 'Download' link. The bottom footer includes copyright information and the version 'Oracle APEX 23.2.4'.

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

**QUERY:**

```
SELECT DISTINCT jobid, locationid FROM employees, dept WHERE employees.deptid = dept.deptid  
AND employees.deptid = 80;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface, similar to the previous one. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', and a search bar. The right side shows a user profile for 'Kavi arasu' (kavi@2005). The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_KAVI2005'. The SQL editor contains the following query:

```
1 SELECT DISTINCT jobid, locationid  
2   FROM employees, dept  
3  WHERE employees.deptid = dept.deptid  
4    AND employees.deptid = 80;  
5
```

The results tab is selected, displaying the output:

JOBID	LOCATIONID
HRREP	5000

Below the results, it says '1 rows returned in 0.01 seconds' and provides a 'Download' link. The bottom footer includes copyright information and the version 'Oracle APEX 23.2.4'.

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

**QUERY:**

```
SELECT e.lastname, d.deptname, d.locationid, l.city FROM employees e, dept d, locations l
```

```
WHERE e.deptid = d.dept AND d.locationid = l.locationid AND e.commissionpct IS NOT NULL;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The top right shows the user 'KA Kavi arasu kavi@2005'. The main area has tabs for 'SQL Commands' and 'Results'. The SQL Commands tab contains the following query:

```
1 SELECT e.lastname, d.deptname, d.locationid, l.city
2   FROM employees e, dept d, locations l
3 WHERE e.deptid = d.deptid
4   AND
5   d.locationid = l.locationid
6   AND e.commissionpct IS NOT NULL;
```

The Results tab displays the output in a grid format:

LASTNAME	DEPTNAME	LOCATIONID	CITY
Joan	Marketing	1000	chennai
Garcia	Marketing	1000	chennai
Lea	IT	5000	Cuddalore
Milper	Sales	2000	TBM

Below the results, it says '4 rows returned in 0.04 seconds'. The bottom of the screen shows copyright information and the version 'Oracle APEX 23.2.4'.

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

**QUERY:**

```
SELECT lastname, deptname FROM employees, dept WHERE employees.deptid = dept.deptid
```

```
AND lastname LIKE '%a%';
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The top right shows the user 'KA Kavi arasu kavi@2005'. The main area has tabs for 'SQL Commands' and 'Results'. The SQL Commands tab contains the following query:

```
1 SELECT lastname, deptname
2   FROM employees, dept
3 WHERE employees.deptid = dept.deptid
4   AND lastname LIKE '%a%';
```

The Results tab displays the output in a grid format:

LASTNAME	DEPTNAME
Milper	Sales

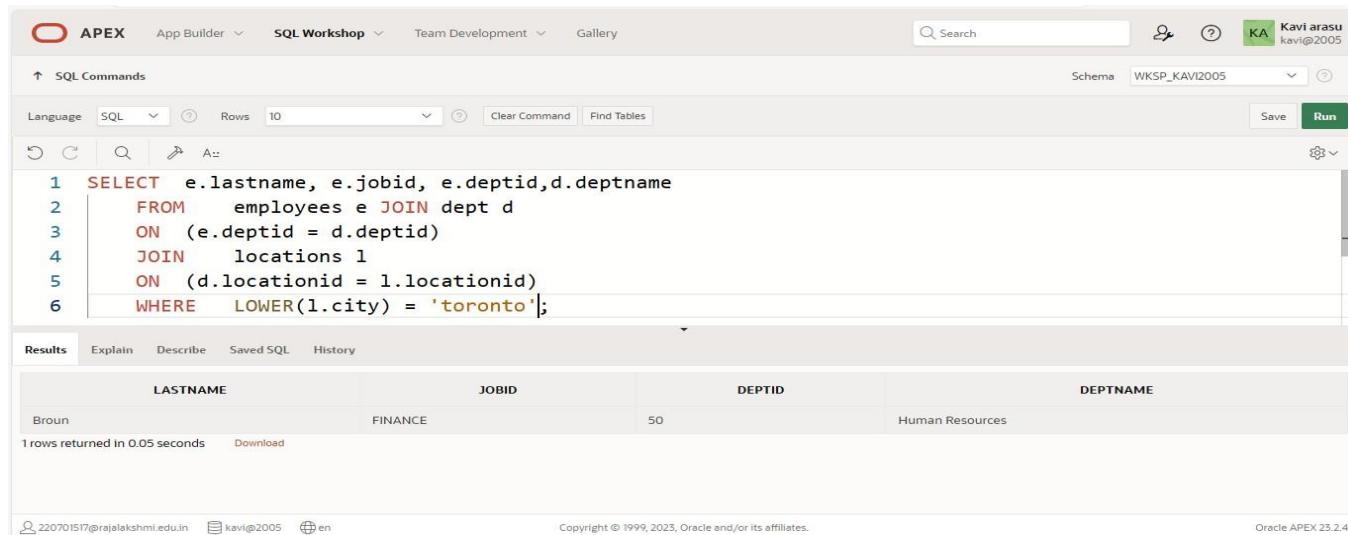
Below the results, it says '1 rows returned in 0.00 seconds'. The bottom of the screen shows copyright information and the version 'Oracle APEX 23.2.4'.

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

**QUERY:**

```
SELECT e.lastname, e.jobid, e.deptid, d.deptname FROM employees e JOIN dept d  
ON (e.deptid = d.deptid) JOIN locations l ON (d.locationid = l.locationid)  
WHERE LOWER(l.city) = 'toronto';
```

**OUTPUT:**



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

```
1 SELECT e.lastname, e.jobid, e.deptid, d.deptname  
2   FROM employees e JOIN dept d  
3     ON (e.deptid = d.deptid)  
4   JOIN locations l  
5     ON (d.locationid = l.locationid)  
6 WHERE LOWER(l.city) = 'toronto';
```

The results section displays a single row:

LASTNAME	JOBID	DEPTID	DEPTNAME
Brown	FINANCE	50	Human Resources

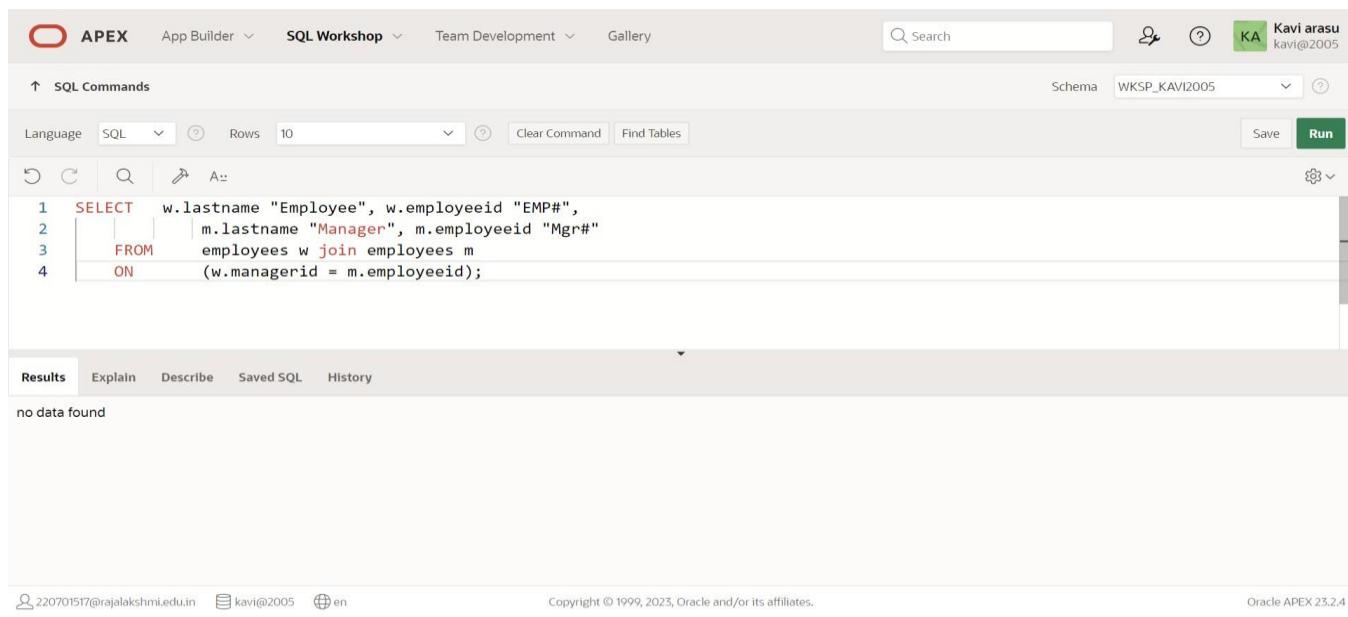
1 rows returned in 0.05 seconds

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

**QUERY:**

```
SELECT w.lastname "Employee", w.employeeid "EMP#", m.lastname "Manager", m.employeeid "Mgr#"  
FROM employees w join employees m ON (w.managerid = m.employeeid);
```

**OUTPUT:**



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

```
1 SELECT w.lastname "Employee", w.employeeid "EMP#",  
2       m.lastname "Manager", m.employeeid "Mgr#"  
3   FROM employees w join employees m  
4     ON (w.managerid = m.employeeid);
```

The results section displays the message:

no data found

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

**QUERY:**

```
SELECT w.lastname "Employee", w.employeeid "EMP#", m.lastname "Manager", m.employeeid "Mgr#"  
FROM employees w LEFT OUTER JOIN employees m  
ON (w.managerid = m.employeeid);
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL Commands tab contains the query:1 SELECT w.lastname "Employee",  
2 m.lastname "Manager", m.employeeid "Mgr#"  
3 FROM employees w  
4 LEFT OUTER JOIN employees m  
5 ON (w.managerid = m.employeeid);

```
Results tab displays the output:
```

Employee	EMP#	Manager	Mgr#
Garcia	175	-	-
Lea	176	-	-
Hamilton	180	-	-
Joan	174	-	-

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.deptid dept, e.lastname employees, c.lastname colleague FROM employees e JOIN employees c  
ON (e.deptid = c.deptid) WHERE e.employeeid <> c.employeeid ORDER BY e.deptid, e.lastname, c.lastname;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL Commands tab contains the query:1 SELECT e.deptid dept, e.lastname employees,  
2 c.lastname colleague  
3 FROM employees e JOIN employees c  
4 ON (e.deptid = c.deptid)  
5 WHERE e.employeeid <> c.employeeid  
6 ORDER BY e.deptid, e.lastname, c.lastname;

```
Results tab displays the output:
```

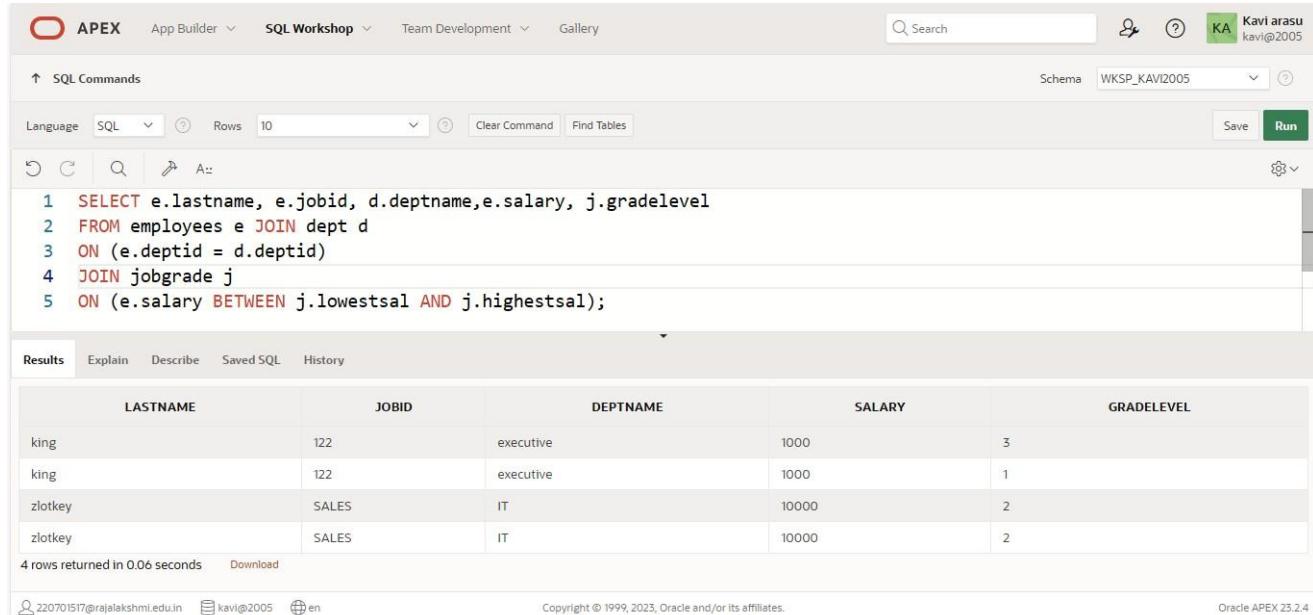
DEPT	EMPLOYEES	COLLEAGUE
100	Garcia	Joan
100	Joan	Garcia

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

**QUERY:**

```
SELECT e.lastname, e.jobid, d.deptname, e.salary, j.gradelevel FROM employees e JOIN dept d  
ON (e.deptid = d.deptid) JOIN jobgrades j  
ON (e.salary BETWEEN j.lowestsal AND j.highestsal);
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to 'WKSP\_KAVI2005'. The query entered is:

```
1 SELECT e.lastname, e.jobid, d.deptname, e.salary, j.gradelevel  
2 FROM employees e JOIN dept d  
3 ON (e.deptid = d.deptid)  
4 JOIN jobgrade j  
5 ON (e.salary BETWEEN j.lowestsal AND j.highestsal);
```

The results table has columns: LASTNAME, JOBID, DEPTNAME, SALARY, and GRADELEVEL. The data returned is:

LASTNAME	JOBID	DEPTNAME	SALARY	GRADELEVEL
king	122	executive	1000	3
king	122	executive	1000	1
zlotkey	SALES	IT	10000	2
zlotkey	SALES	IT	10000	2

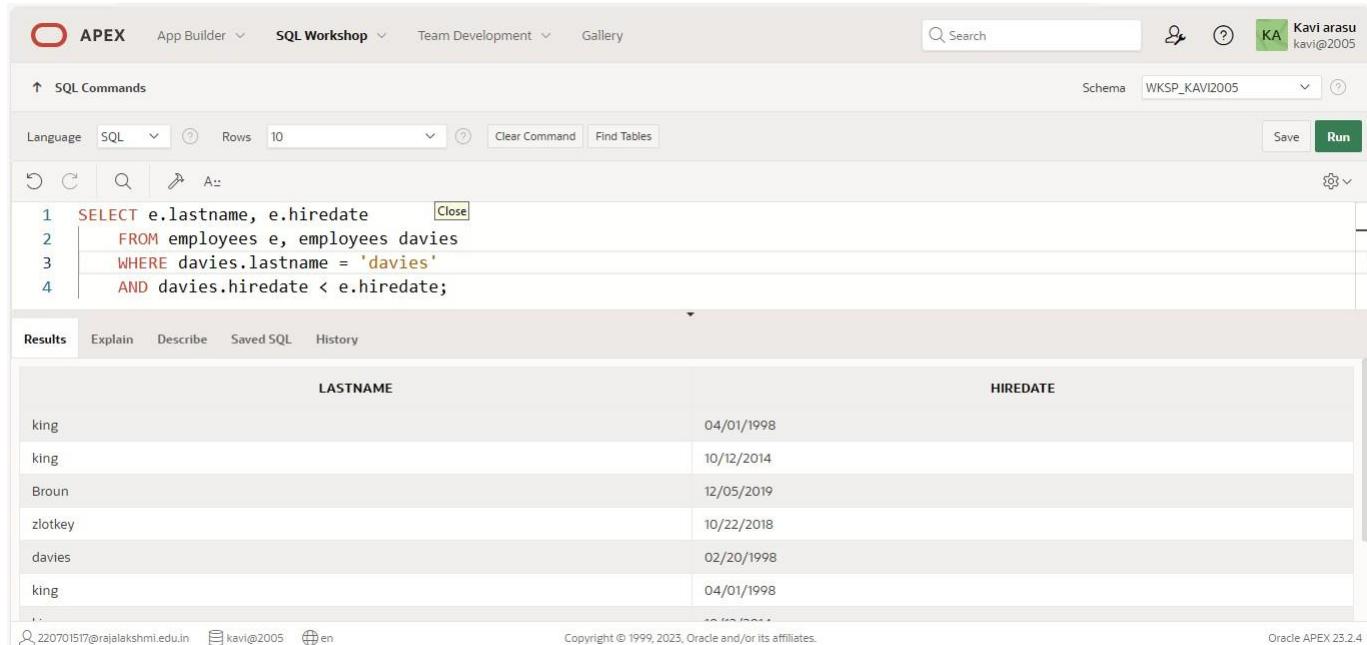
4 rows returned in 0.06 seconds

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

**QUERY:**

```
SELECT e.lastname, e.hiredate FROM employees e, employees davies  
WHERE davies.last_name = 'Davies'  
AND davies.hiredate < e.hiredate
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to 'WKSP\_KAVI2005'. The query entered is:

```
1 SELECT e.lastname, e.hiredate  
2 FROM employees e, employees davies  
3 WHERE davies.lastname = 'davies'  
4 AND davies.hiredate < e.hiredate;
```

The results table has columns: LASTNAME and HIREDATE. The data returned is:

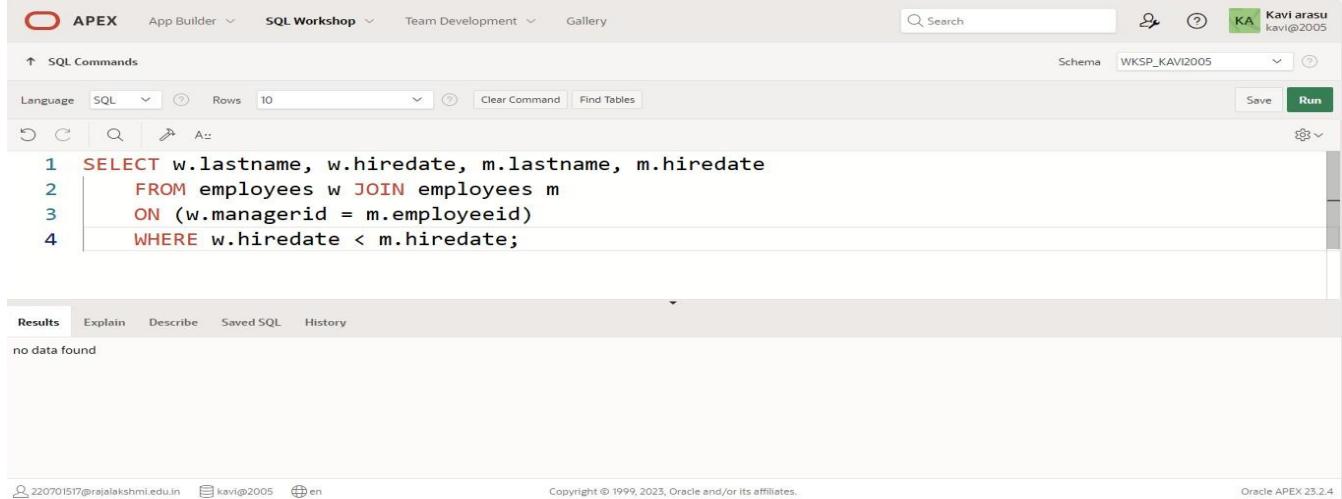
LASTNAME	HIREDATE
king	04/01/1998
king	10/12/2014
Broun	12/05/2019
zlotkey	10/22/2018
davies	02/20/1998
king	04/01/1998

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

### QUERY:

```
SELECT w.lastname, w.hiredate, m.lastname, m.hiredate FROM employees w JOIN employees m  
ON (w.managerid = m.employeeid) WHERE w.hiredate < m.hiredate;
```

### OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'Kavi arasu' (kavi@2005). The main area is titled 'SQL Commands' with a search bar and a schema dropdown set to 'WKSP\_KAVI2005'. Below the command input field, there are buttons for 'Save' and 'Run'. The code entered is:

```
1 SELECT w.lastname, w.hiredate, m.lastname, m.hiredate  
2   FROM employees w JOIN employees m  
3     ON (w.managerid = m.employeeid)  
4 WHERE w.hiredate < m.hiredate;
```

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

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

### RESULT:

# AGGREGATING DATA USING GROUP FUNCTIONS

EX\_NO:8

DATE:

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

**TRUE**

2. Group functions include nulls in calculations.  
True/False

**FALSE**

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

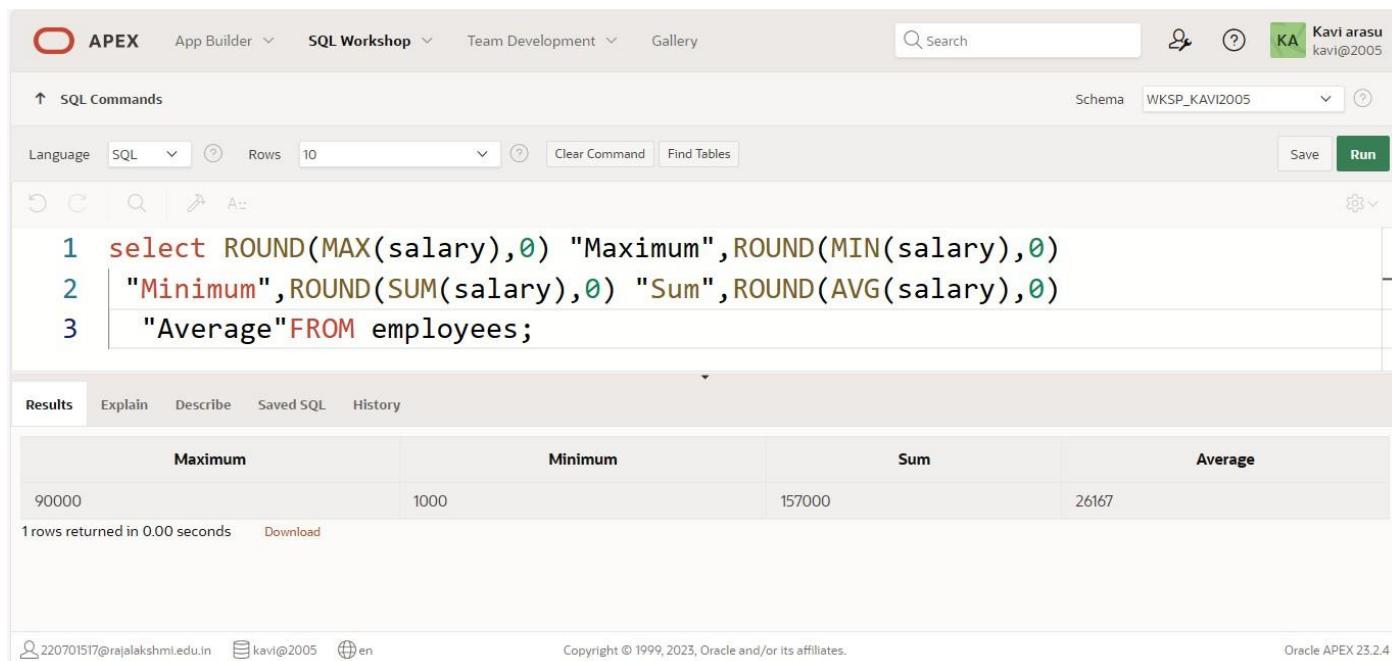
**FALSE**

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

**QUERY:**

```
select ROUND(MAX(salary),0) "Maximum",ROUND(MIN(salary),0)
"Minimum",ROUND(SUM(salary),0) "Sum",ROUND(AVG(salary),0)
"Average"FROM employees;
```

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is identified as Kavi arasu (kavi@2005). The SQL Workshop tab is selected. The SQL command window contains the following query:

```
1 select ROUND(MAX(salary),0) "Maximum",ROUND(MIN(salary),0)
2 "Minimum",ROUND(SUM(salary),0) "Sum",ROUND(AVG(salary),0)
3 "Average"FROM employees;
```

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

	Maximum	Minimum	Sum	Average
	90000	1000	157000	26167

Below the table, it says "1 rows returned in 0.00 seconds". The bottom footer includes copyright information for Oracle and the APEX version.

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

### QUERY:

```
SELECT jobid, ROUND(MAX(salary),0) "Maximum",ROUND(MIN(salary),0)  
"Minimum",ROUND(SUM(salary),0)"Sum",ROUND(AVG(salary),0)  
"Average"FROM employees group by jobid;
```

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The query is displayed in the SQL Commands pane:

```
1 SELECT jobid, ROUND(MAX(salary),0)  
2 "Maximum",ROUND(MIN(salary),0)  
3 "Minimum",ROUND(SUM(salary),0)  
4 "Sum",ROUND(AVG(salary),0)  
5 "Average"FROM employees  
6 group by jobid;
```

The Results pane displays the following table:

JOBID	Maximum	Minimum	Sum	Average
FINANCE	90000	90000	90000	90000
HRREP	45000	1000	52000	17333
SALES	10000	10000	10000	10000
MARKETING	5000	5000	5000	5000

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

### QUERY:

```
SELECT jobid, COUNT(*)FROM employees GROUP BY jobid;
```

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The query is displayed in the SQL Commands pane:

```
1 SELECT jobid, COUNT(*)FROM employees GROUP BY jobid;
```

The Results pane displays the following table:

JOBID	COUNT(*)
FINANCE	1
HRREP	1
SALES	1
MARKETING	1
122	2

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

#### QUERY:

```
SELECT COUNT(DISTINCT managerid) "Number of Managers"FROM employees;
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, and a search bar. The schema is set to WKSP\_KAVI2005. The main area displays the SQL command: `SELECT COUNT(DISTINCT managerid) "Number of Managers"FROM employees;`. The results section shows a single row with the value 4, labeled "Number of Managers". Below the results, it states "1 rows returned in 0.01 seconds" and provides a download link. The bottom footer includes user information (220701517@rajalakshmi.edu.in, kavi@2005, en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the version (Oracle APEX 23.2.4).

Number of Managers
4

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

#### QUERY:

```
SELECT MAX(salary) - MIN(salary) DIFFERENCE FROM employees;
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, Gallery, and a search bar. The schema is set to WKSP\_KAVI2005. The current user is Kavi arasu (kavi@2005). The main area displays the SQL command: `SELECT MAX(salary) - MIN(salary) DIFFERENCE FROM employees;`. The results section shows a single row with the value 89000, labeled "DIFFERENCE". Below the results, it states "1 rows returned in 0.00 seconds" and provides a download link. The bottom footer includes user information (220701517@rajalakshmi.edu.in, kavi@2005, en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the version (Oracle APEX 23.2.4).

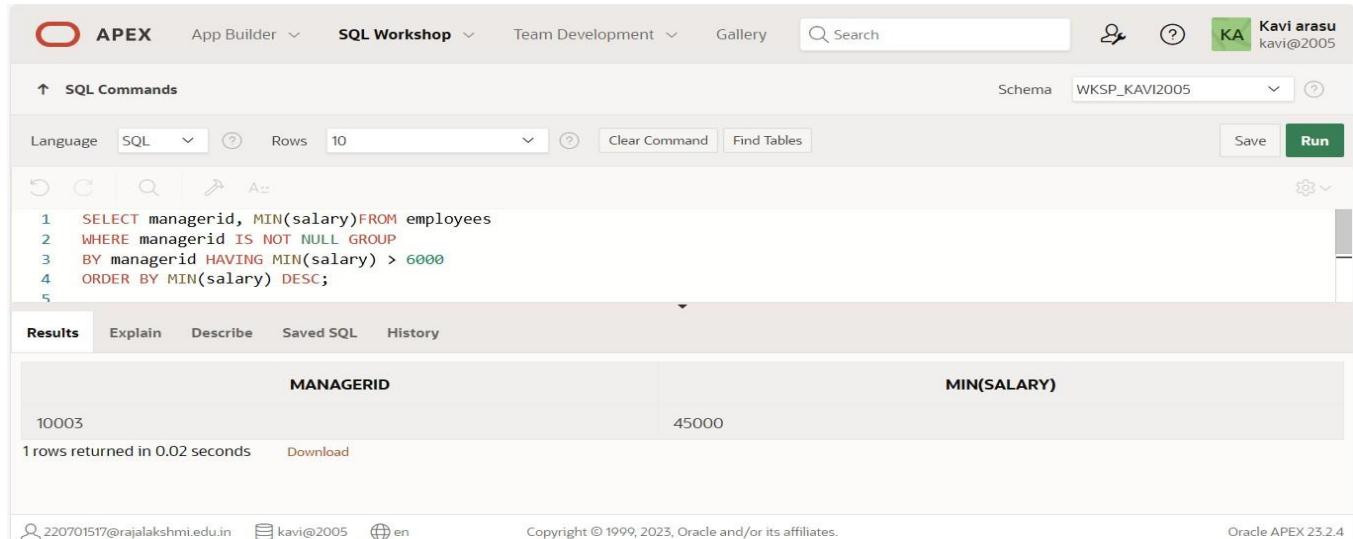
DIFFERENCE
89000

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

#### QUERY:

```
SELECT managerid, MIN(salary)FROM employees WHERE managerid IS NOT NULL GROUP BY managerid HAVING MIN(salary) > 6000 ORDER BY MIN(salary) DESC;
```

#### OUTPUT:



The screenshot shows the Oracle APEX interface with the SQL Workshop tab selected. The query is displayed in the SQL Commands pane:

```
1 SELECT managerid, MIN(salary)FROM employees
2 WHERE managerid IS NOT NULL GROUP
3 BY managerid HAVING MIN(salary) > 6000
4 ORDER BY MIN(salary) DESC;
5
```

The Results pane shows the output:

MANAGERID	MIN(SALARY)
10003	45000

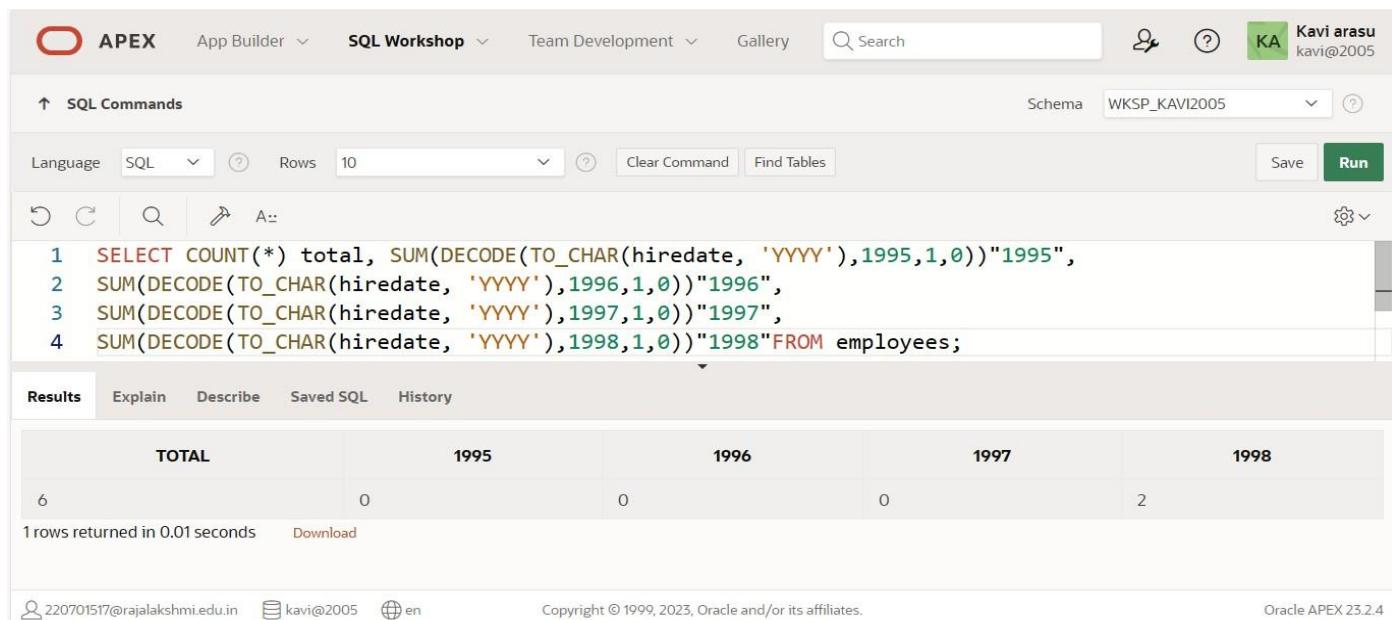
1 rows returned in 0.02 seconds [Download](#)

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

#### QUERY:

```
SELECT COUNT(*) total, SUM(DECODE(TO_CHAR(hiredate, 'YYYY'),1995,1,0))"1995",
SUM(DECODE(TO_CHAR(hiredate, 'YYYY'),1996,1,0))"1996", SUM(DECODE(TO_CHAR(hiredate,
'YYYY'),1997,1,0))"1997", SUM(DECODE(TO_CHAR(hiredate, 'YYYY'),1998,1,0))"1998"FROM employees;
```

#### OUTPUT:



The screenshot shows the Oracle APEX interface with the SQL Workshop tab selected. The query is displayed in the SQL Commands pane:

```
1 SELECT COUNT(*) total, SUM(DECODE(TO_CHAR(hiredate, 'YYYY'),1995,1,0))"1995",
2 SUM(DECODE(TO_CHAR(hiredate, 'YYYY'),1996,1,0))"1996",
3 SUM(DECODE(TO_CHAR(hiredate, 'YYYY'),1997,1,0))"1997",
4 SUM(DECODE(TO_CHAR(hiredate, 'YYYY'),1998,1,0))"1998"FROM employees;
```

The Results pane shows the output:

TOTAL	1995	1996	1997	1998
6	0	0	0	2

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

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

#### QUERY:

```
SELECT jobid "Job",SUM(DECODE(deptid , 20, salary)) "Dept 20",SUM(DECODE(deptid , 50, salary)) "Dept 50",
SUM(DECODE(deptid , 80, salary)) "Dept 80",SUM(DECODE(deptid , 90, salary))
"Dept 90",SUM(salary) "Total" FROM employees GROUP BY jobid;
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The query has been run and the results are displayed in a grid format. The columns are labeled 'Job', 'Dept 20', 'Dept 50', 'Dept 80', 'Dept 90', and 'Total'. The data shows salary distribution across different departments.

Job	Dept 20	Dept 50	Dept 80	Dept 90	Total
FINANCE	-	-	-	-	90000
HRREP	-	-	6000	-	52000
SALES	-	-	-	-	10000
MARKETING	-	-	-	-	5000

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

#### QUERY:

```
Select d.deptname as "deptname",d.loc as "department location", count(*) as "Number of people",round(avg(e.salary),2) as "salary" from dept d inner join employees e on (d.deptid=e.deptid)
Group by d.deptname,d.loc;
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The query has been run and the results are displayed in a grid format. The columns are labeled 'deptname', 'department location', 'Number of people', and 'salary'. The data shows department details with their respective counts and average salaries.

deptname	department location	Number of people	salary
IT	arch	1	10000
executive	admin	2	23000
Human Resources	workshop	1	90000

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

**RESULT:**

## SUB-QUERIES

EX.NO:9

DATE:

Find the Solution for the following:

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

**QUERY:**

```
Select lastname, hiredate  
from employees  
where deptid =  
(select deptid from employees where lastname like 'zlotkey')and lastname <> 'zlotkey';
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', 'Gallery', a search bar, and a user profile 'Kavi arasu kavi@2005'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_KAVI2005'. The SQL editor contains the following code:

```
1 Select lastname, hiredate  
2 from employees  
3 where deptid =  
4 (select deptid from employees where lastname like 'zlotkey')and lastname <> 'zlotkey';
```

The 'Results' tab is selected, displaying the output:

LASTNAME	HIREDATE
Hamilton	10/12/2014

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

At the bottom, the footer includes user information '220701517@rajalakshmi.edu.in' and 'kavi@2005', a language link 'en', copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.', and 'Oracle APEX 23.2.4'.

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

### QUERY:

```
select employeeid, lastname, salary from employees  
where salary > (select avg(salary) from employees) order by salary;
```

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', and a search bar. The right side shows a user profile for 'Kavi arasu' with the email 'kavi@2005'. The main area is titled 'SQL Commands' with tabs for 'Language' (set to 'SQL'), 'Rows' (set to 10), and buttons for 'Clear Command' and 'Find Tables'. The code editor contains the following SQL query:

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

Below the code editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, displaying a table with three columns: 'EMPLOYEEID', 'LASTNAME', and 'SALARY'. The data returned is:

EMPLOYEEID	LASTNAME	SALARY
175	Garcia	45000
178	Brown	90000

Below the table, it says '2 rows returned in 0.01 seconds' and has a 'Download' link. At the bottom, it shows the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 23.2.4'.

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 employeeid, lastname  
from employees  
where deptid in (select deptid from employees where lastname like '%u%');
```

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface, identical to the previous one but with a different query. The top navigation bar, code editor, and table output are the same. The 'Results' table now contains only one row:

EMPLOYEEID	LASTNAME
178	Brown

Below the table, it says '1 rows returned in 0.01 seconds' and has a 'Download' link. At the bottom, it shows the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 23.2.4'.

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

### QUERY:

```
select lastname, deptid, jobid  
from employees  
where deptid in (select deptid from dept where locationid =1700);
```

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', and a search bar. The user is 'Kavi arasu' (kavi@2005). The schema is set to 'WKSP\_KAVI2005'. The SQL Commands tab is active, showing the executed query:

```
1 select lastname, deptid, jobid  
2 from employees  
3 where deptid in (select deptid from dept where locationid =1700);
```

The Results tab displays the output:

LASTNAME	DEPTID	JOBID
davies	100	MARKETING
Garcia	100	HRREP

2 rows returned in 0.00 seconds. The bottom of the page shows copyright information and the version 'Oracle APEX 23.2.4'.

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

### QUERY:

```
select lastname, salary  
from employees  
where managerid in (select employeeid from employees where lastname='King');
```

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', 'Gallery', and a search bar. The user is 'Kavi arasu' (kavi@2005). The schema is set to 'WKSP\_KAVI2005'. The SQL Commands tab is active, showing the executed query:

```
1 select lastname,salary from employees  
2 where managerid in (select employeeid from employees where lastname = 'king');
```

The Results tab displays the output:

LASTNAME	SALARY
King	50000

no data found. The bottom of the page shows copyright information and the version 'Oracle APEX 23.2.4'.

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 deptid, lastname, jobid  
from employees  
where deptid in (select deptid from dept where deptname = 'executive');
```

### 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, along with a search bar and user information (Kavi arasu, kavi@2005). The main area is titled 'SQL Commands' with tabs for Language (set to SQL), Rows (set to 10), Clear Command, and Find Tables. The SQL editor contains the following query:

```
1 select deptid, lastname, jobid  
2 from employees  
3 where deptid in (select deptid from dept where deptname = 'executive');
```

Below the editor, the 'Results' tab is selected, displaying the output in a grid format:

DEPTID	LASTNAME	JOBID
100	davies	MARKETING
100	king	HRREP

At the bottom of the results section, it says '2 rows returned in 0.01 seconds' and has a 'Download' link. The footer of the page includes copyright information (Copyright © 1999, 2023, Oracle and/or its affiliates) and the text 'Oracle APEX 23.2.4'.

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

### QUERY:

```
select employeeid, lastname, salary  
from employees  
where salary > (select avg(salary) from employees) and  
deptid in (select deptid from employees where lastname like '%u%');
```

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface, identical to the previous one but with a different query in the editor. The query is:

```
1 select employeeid, lastname, salary  
2 from employees  
3 where salary > (select avg(salary) from employees) and  
4 deptid in (select deptid from employees where lastname like '%u%');
```

The results show a single row:

EMPLOYEEID	LASTNAME	SALARY
178	Broun	90000

At the bottom, it says '1 rows returned in 0.01 seconds' and has a 'Download' link. The footer includes the same copyright and version information as the first screenshot.

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

**RESULT:**

# USING THE SET OPERATORS

EX.NO:10

DATE:

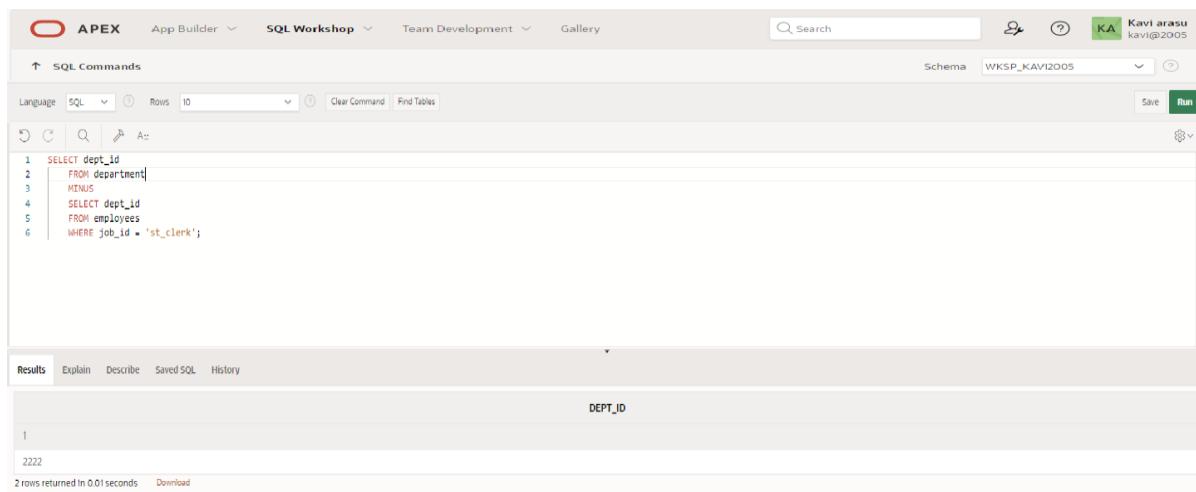
Find the Solution for the following:

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

**QUERY:**

```
SELECT dept_id
FROM department
MINUS
SELECT dept_id
FROM employees
WHERE job_id = 'st_clerk';
```

**OUTPUT:**



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

```
1 SELECT dept_id
2   FROM department
3   MINUS
4   SELECT dept_id
5     FROM employees
6    WHERE job_id = 'st_clerk';
```

In the Results pane, the output is displayed in a table with one column labeled "DEPT\_ID". The value "2222" is shown in the table.

DEPT_ID
2222

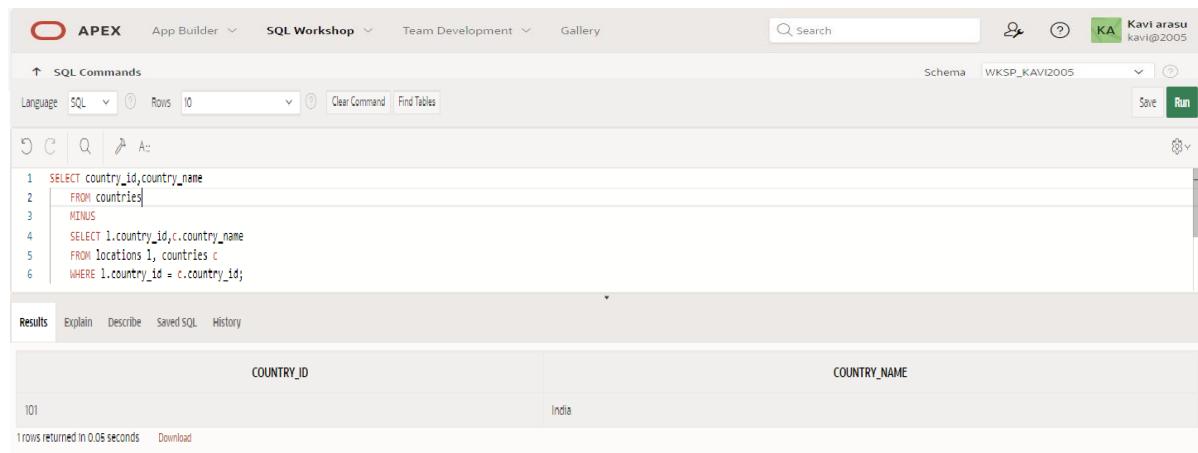
Below the table, it says "2 rows returned in 0.01 seconds" and there is a "Download" link.

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

### QUERY:

```
SELECT country_id,country_name
FROM countries
MINUS
SELECT l.country_id,c.country_name
FROM locations l, countries c
WHERE l.country_id = c.country_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 'Kavi arasu' with the email 'kavi@2005'. The main area has tabs for 'SQL Commands' and 'Results' (selected). The SQL command entered is:

```
1 SELECT country_id,country_name
2   FROM countries
3
4 MINUS
5
6 SELECT l.country_id,c.country_name
7   FROM locations l, countries c
8 WHERE l.country_id = c.country_id;
```

The results section displays a single row of data:

COUNTRY_ID	COUNTRY_NAME
101	India

Below the table, it says '1 rows returned in 0.05 seconds' and has a 'Download' link.

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

#### QUERY:

```
SELECT DISTINCT job_no, dept_id  
FROM employees  
WHERE dept_id = 10
```

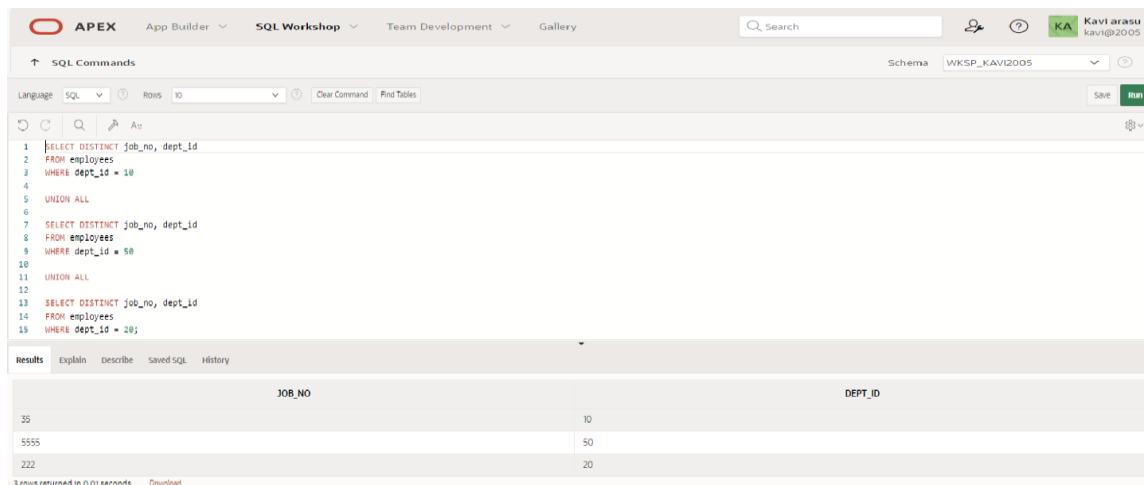
UNION ALL

```
SELECT DISTINCT job_no, dept_id  
FROM employees  
WHERE dept_id = 50
```

UNION ALL

```
SELECT DISTINCT job_no, dept_id  
FROM employees  
WHERE dept_id = 20;
```

#### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for Kaviarasu (kavi@2005). The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the following code:

```
1 SELECT DISTINCT job_no, dept_id  
2 FROM employees  
3 WHERE dept_id = 10  
4  
5 UNION ALL  
6  
7 SELECT DISTINCT job_no, dept_id  
8 FROM employees  
9 WHERE dept_id = 50  
10  
11 UNION ALL  
12  
13 SELECT DISTINCT job_no, dept_id  
14 FROM employees  
15 WHERE dept_id = 20;
```

The Results tab displays the output in a grid:

JOB_NO	DEPT_ID
35	10
5555	50
222	20

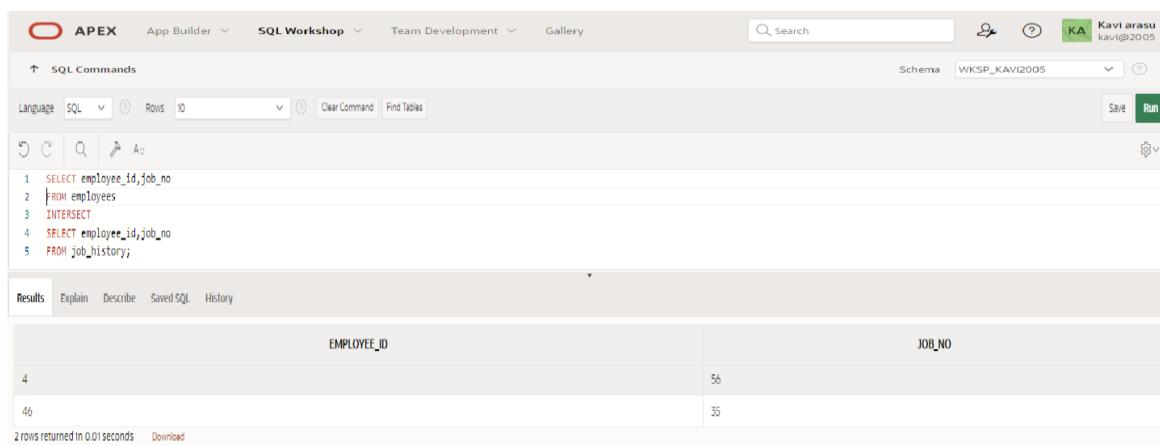
At the bottom, it says "3 rows returned in 0.01 seconds".

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

#### QUERY:

```
SELECT employee_id,job_no  
FROM employees  
INTERSECT  
SELECT employee_id,job_no  
FROM job_history;
```

#### 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 'Kavi arasu' (kavi@2005). The main area has a toolbar with 'SQL Commands', 'Language' set to 'SQL', 'Rows' set to 10, and 'Run' button. Below the toolbar is a code editor with the following SQL query:

```
1 SELECT employee_id,job_no  
2 FROM employees  
3 INTERSECT  
4 SELECT employee_id,job_no  
5 FROM job_history;
```

Below the code editor is a results grid with two columns: 'EMPLOYEE\_ID' and 'JOB\_NO'. The data returned is:

EMPLOYEE_ID	JOB_NO
4	56
46	35

At the bottom of the results grid, 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 last_name,dept_id,TO_CHAR(null)
FROM employees
UNION
SELECT TO_CHAR(null),dept_id,dept_name
FROM department;
```

### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. At the top, there are tabs for App Builder, SQL Workshop (selected), Team Development, and Gallery. Below the tabs is a search bar and a user profile for 'Kavi arasu'. The main area has a 'SQL Commands' tab selected. The SQL code entered is:

```
1 SELECT last_name,dept_id,TO_CHAR(null)
2 FROM employees
3 UNION
4 SELECT TO_CHAR(null),dept_id,dept_name
5 FROM department;
```

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

LAST_NAME	DEPT_ID	TO_CHAR(NULL)
Davies	20	-
King	10	-
Zlot列	2222	-
bhuvana	1	-
brindha	59	-
deepa	50	-
-	1	Executive
-	59	BIO
-	2222	CSE

At the bottom left, it says "9 rows returned in 0.01 seconds". To the right of the results is a marking grid:

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

### RESULT:

**EXNO:11**

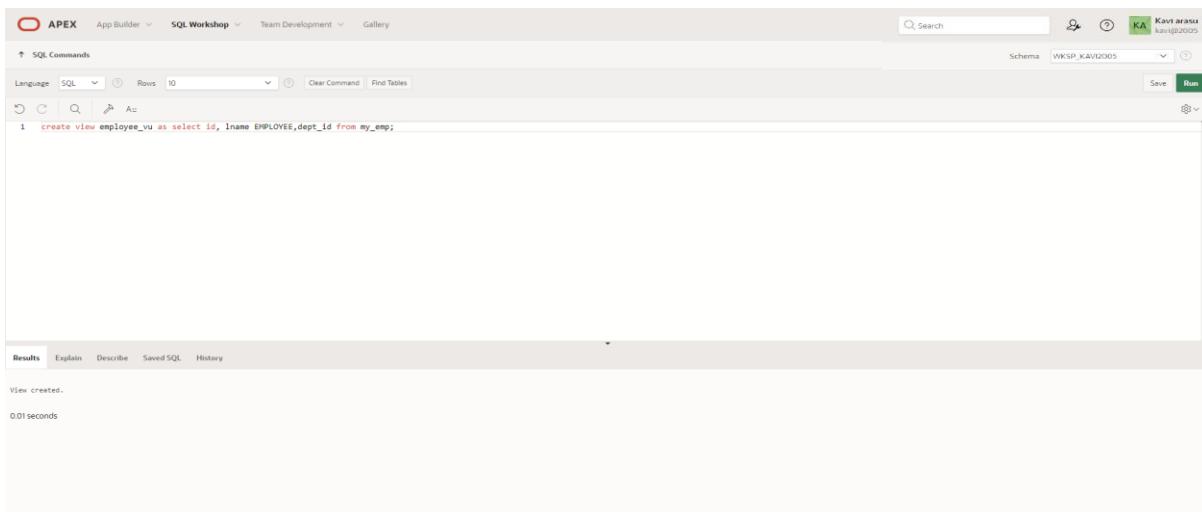
## **CREATING VIEWS**

**DATE:**

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

**QUERY:** create view employee\_vu as select id, lname EMPLOYEE,dept\_id from my\_emp;

**OUTPUT:**



A screenshot of the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The main area shows a SQL command line with the following code:

```
1 create view employee_vu as select id, lname EMPLOYEE,dept_id from my_emp;
```

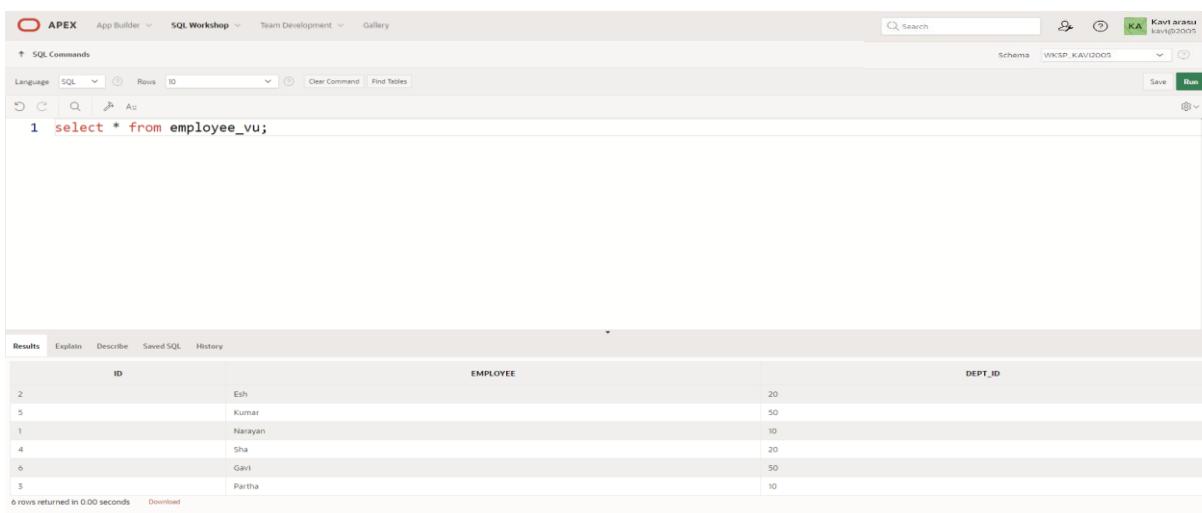
The results pane below shows the output of the command:

```
View created.  
0.01 seconds
```

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

**QUERY:** select \* from employee\_vu;

**OUTPUT:**



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

```
1 select * from employee_vu;
```

The results pane displays the data from the EMPLOYEE\_VU view:

ID	EMPLOYEE	DEPT_ID
2	Esh	20
5	Kumar	50
1	Narayan	10
4	Sha	20
6	Gavl	50
3	Partha	10

6 rows returned in 0.00 seconds

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

**QUERY:** select employee, dept\_id from employee\_vu;

**OUTPUT:**

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

```
1 select employee, dept_id from employee_vu;
```

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

EMPLOYEE	DEPT_ID
Esh	20
Kumar	50
Narayan	10
Sha	20
Gavil	50
Partha	10

Below the table, it says "6 rows returned in 0.01 seconds" and "Downloaded".

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

**QUERY:** create view dept50 as select id EMPNO, lname EMPLOYEE, dept\_id DEPTNO from my\_emp where dept\_id=50 with read only;

**OUTPUT:**

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

```
1 create view dept50 as select id EMPNO, lname EMPLOYEE, dept_id DEPTNO from my_emp where dept_id=50 with read only;
```

In the Results pane, the output is displayed as "View created." and "0.03 seconds".

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

QUERY: select \* from dept50;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The command entered is "select \* from dept50;". The results section displays a table with four rows, each containing EMPNO, EMPLOYEE, and DEPTNO columns. The data is as follows:

EMPNO	EMPLOYEE	DEPTNO
5	Kumar	50
1	Narayan	50
6	Gari	50
3	Partha	50

4 rows returned in 0.01 seconds

## 6. Attempt to reassign Matos to department 80.

QUERY: update dept50 set deptno=80 where employee='Matos';

OUTPUT:

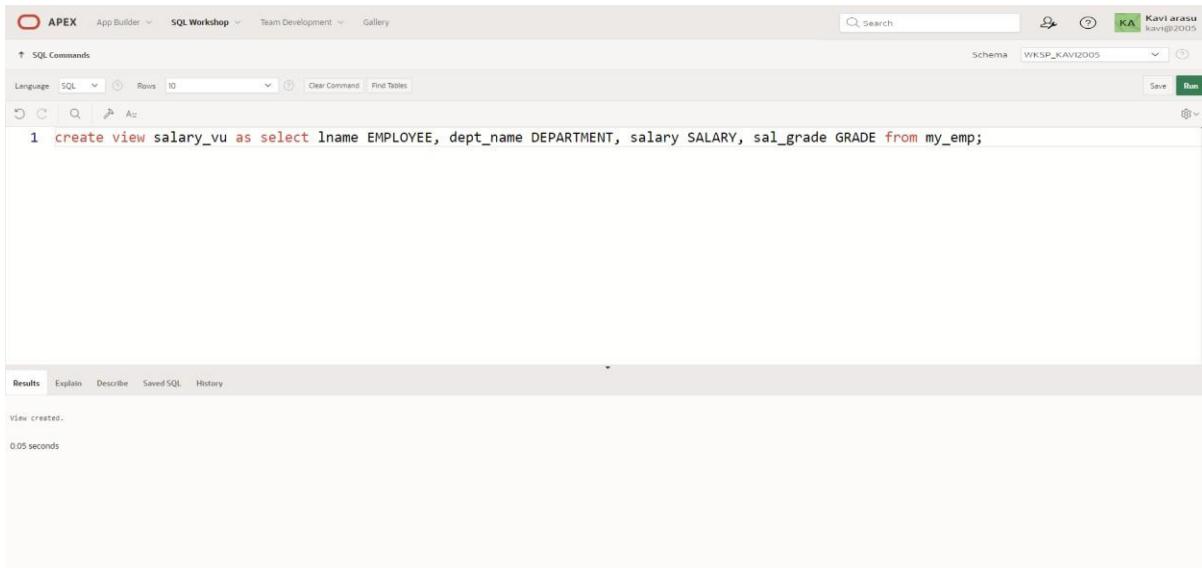
The screenshot shows the Oracle SQL Workshop interface. The command entered is "update dept50 set deptno=80 where employee='Matos';". A yellow box highlights the error message: "Error at line 1/19: ORA-43999: cannot perform a DML operation on a read-only view".

0.01 seconds

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

**QUERY:** create view salary\_vu as select lname EMPLOYEE, dept\_name DEPARTMENT, salary SALARY, sal\_grade GRADE from my\_emp;

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is chosen from a dropdown menu. The main area is titled 'SQL Commands'. A single line of SQL code is entered in the command input field: 'create view salary\_vu as select lname EMPLOYEE, dept\_name DEPARTMENT, salary SALARY, sal\_grade GRADE from my\_emp;'. The 'Run' button is visible at the bottom right of the input field. Below the input field, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is currently selected. The output section displays the message 'View created.' and '0.05 seconds'.

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

**RESULT:**

## **EXNO:12 PRIMARY KEY, FOREIGN KEY AND CHECK CONSTRAINTS**

### **1. What is the purpose of a**

- **PRIMARY KEY** – They provide a unique value that can identify a specific row in a table
- **FOREIGN KEY** - to link data between tables
- **CHECK CONSTRAINT** - to limit the value range that can be placed in a column

### **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**

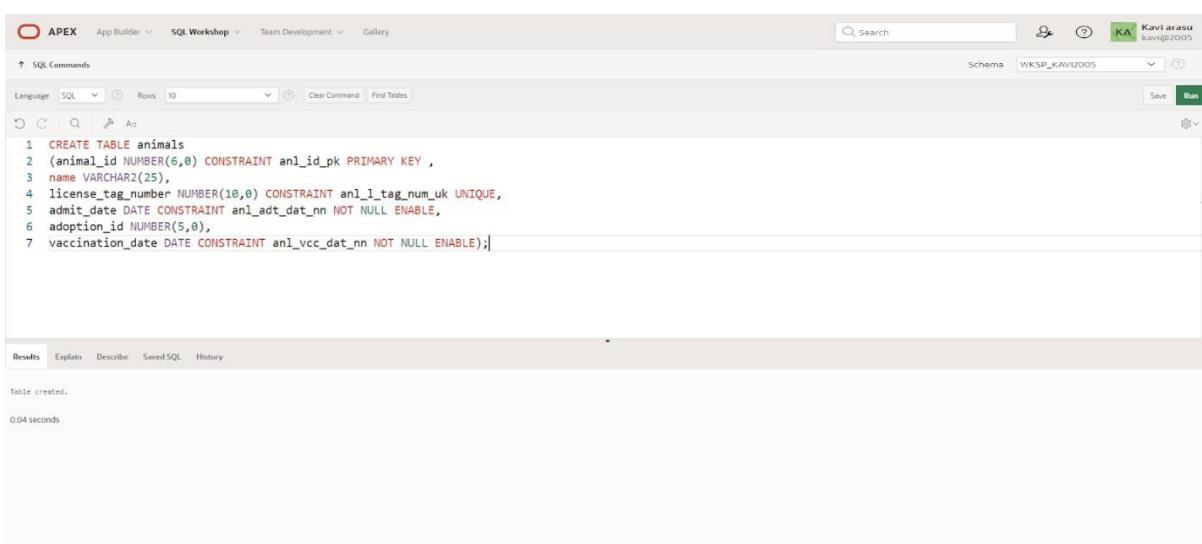
**adoption\_id NUMBER(5),**

**vaccination\_date DATE- NOT NULL**

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

**QUERY: CREATE TABLE animals (animal\_id NUMBER(6,0) CONSTRAINT 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);**

### **OUTPUT:**



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

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

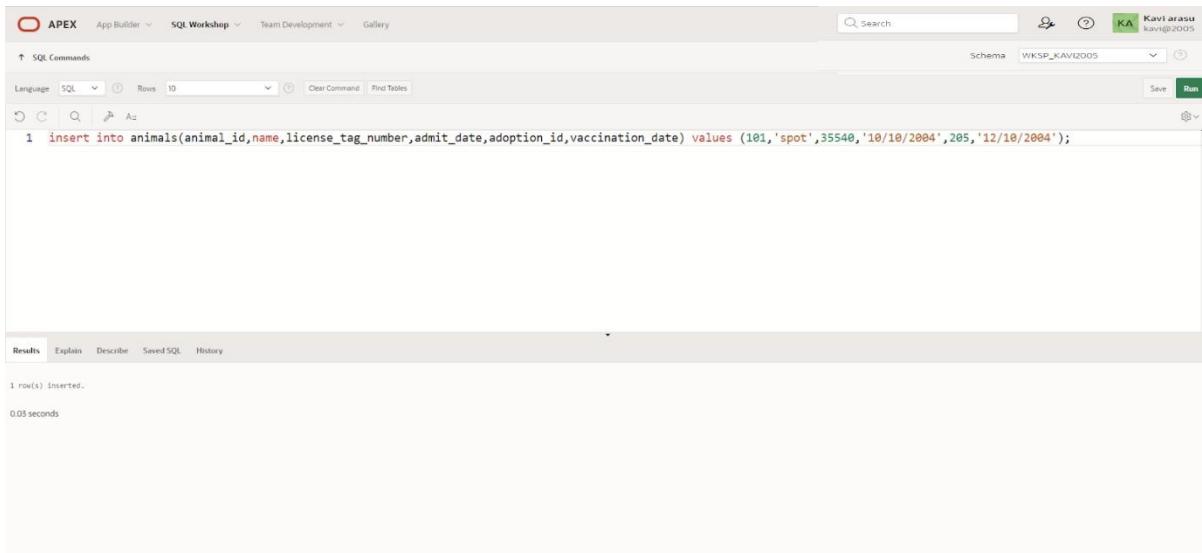
In the Results tab, the output shows:

```
Table created.
0.04 seconds
```

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

**QUERY:**insert into  
animals(animal\_id,name,license\_tag\_number,admit\_date,adoption\_id,  
vaccination\_date) values (101,'spot',35540,'10/10/2004',205,'12/10/2004');

**OUTPUT:**



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

```
1 insert into animals(animal_id,name,license_tag_number,admit_date,adoption_id,vaccination_date) values (101,'spot',35540,'10/10/2004',205,'12/10/2004');
```

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

1 row(s) inserted.  
0.03 seconds

**5. What are the restrictions on defining a CHECK constraint?**

**Ans:** If you define a CHECK constraint on a column it will allow only certain values for this column. If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

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

**RESULT:**

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

Results	Explain	Describe	Saved SQL	History
ID	Song Title	ARTIST		
47	Hurrah for Today			
49	Lets Celebrate			

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

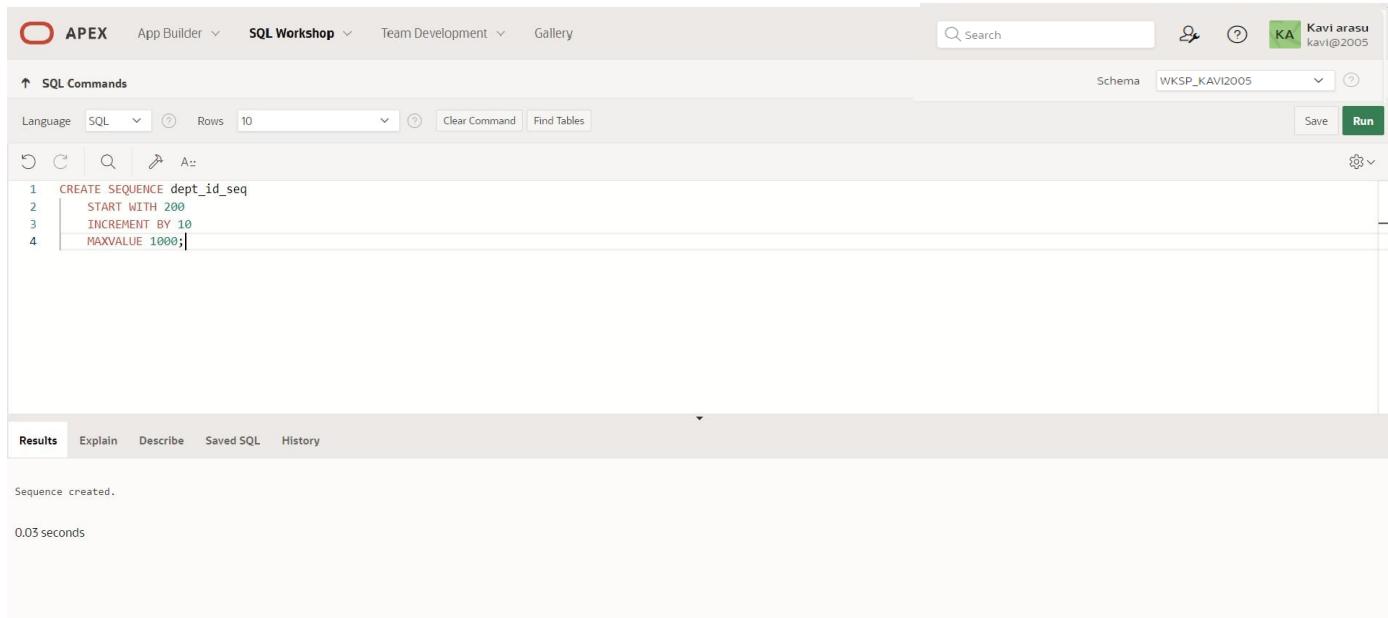
**EX\_NO:14**

**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;
```



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

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

Below the code, the results pane displays the message "Sequence created." and a execution time of "0.03 seconds".

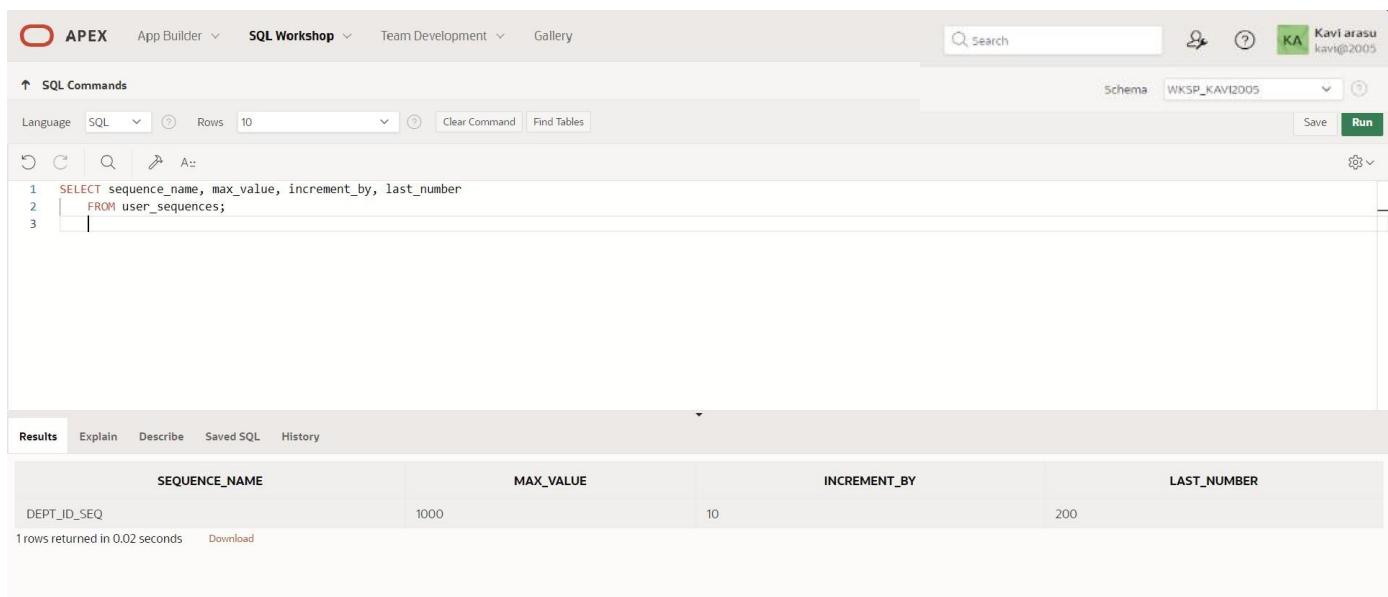
**OUTPUT:**

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, 'SQL Workshop' is selected. The main area contains the following SQL code:

```
1 SELECT sequence_name, max_value, increment_by, last_number
2   FROM user_sequences;
3 |
```

The results pane displays a table with the following data:

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 there is a "Download" link.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user profile (KA Kavi arasu kavi@2005), and a schema dropdown set to WKSP\_KAVI2005. Below the navigation is a toolbar with various icons. The main area is titled "SQL Commands" and shows a summary of the executed script "exercise14". The status is "Complete" with a duration of 0.03 seconds and 15 rows processed. The "Summary" view is selected. The results table has columns: Number, Elapsed, Statement, Feedback, and Rows. One row is listed: "1" with "0.03" elapsed time, the statement "INSERT INTO dept VALUES (dept\_id\_seq.nextval, 'Education')", feedback "1row(s) inserted.", and 1 row. Below the table, three metrics are displayed: "Statements Processed" (1), "Successful" (1), and "With Errors" (0). The bottom right corner shows "row(s) 1 - 1 of 1".

Number	Elapsed	Statement	Feedback	Rows
1	0.03	INSERT INTO dept VALUES (dept_id_seq.nextval, 'Education')	1row(s) inserted.	1

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user profile (KA Kavi arasu kavi@2005), and a schema dropdown set to WKSP\_KAVI2005. Below the navigation is a toolbar with various icons. The main area is titled "SQL Commands" and shows a summary of the executed command. The language is set to SQL, and 10 rows were processed. The command entered was "CREATE INDEX emp\_dept\_id\_idx ON EMPLOYEES (department\_id);". The results tab at the bottom shows the message "Index created." and a timestamp of "0.03 seconds".

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands' with a sub-section 'SQL'. The schema dropdown shows 'WKSP\_KAVI2005'. A query is entered in the command window:

```
1 SELECT index_name, table_name, uniqueness
2   FROM user_indexes
3  WHERE table_name = 'EMPLOYEES';
4
```

The results tab is selected, displaying the output:

INDEX_NAME	TABLE_NAME	UNIQUENESS
EMP_DEPT_ID_IDX	EMPLOYEES	NONUNIQUE

1 rows returned in 0.03 seconds. There is a 'Download' link below the results table.

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

**RESULT:**

# CONTROLLING USER ACCESS

**EX\_NO:15**

**DATE:**

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

The CREATE SESSION system privilege

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

The CREATE TABLE privilege

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

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

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

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

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

The ALTER USER statement

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

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

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

7. Query all the rows in your DEPARTMENTS table.

SELECT \* FROM departments;

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

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

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

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

SELECT table\_name FROM user\_tables;

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

Team 1 revokes the privilege.

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

Team 2 revokes the privilege.

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

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

Team 1 executes this INSERT statement.

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

Team 2 executes this INSERT statement.

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

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

**RESULT:**

# PL/SQL

## CONTROL STRUCTURES

**EX\_NO:**

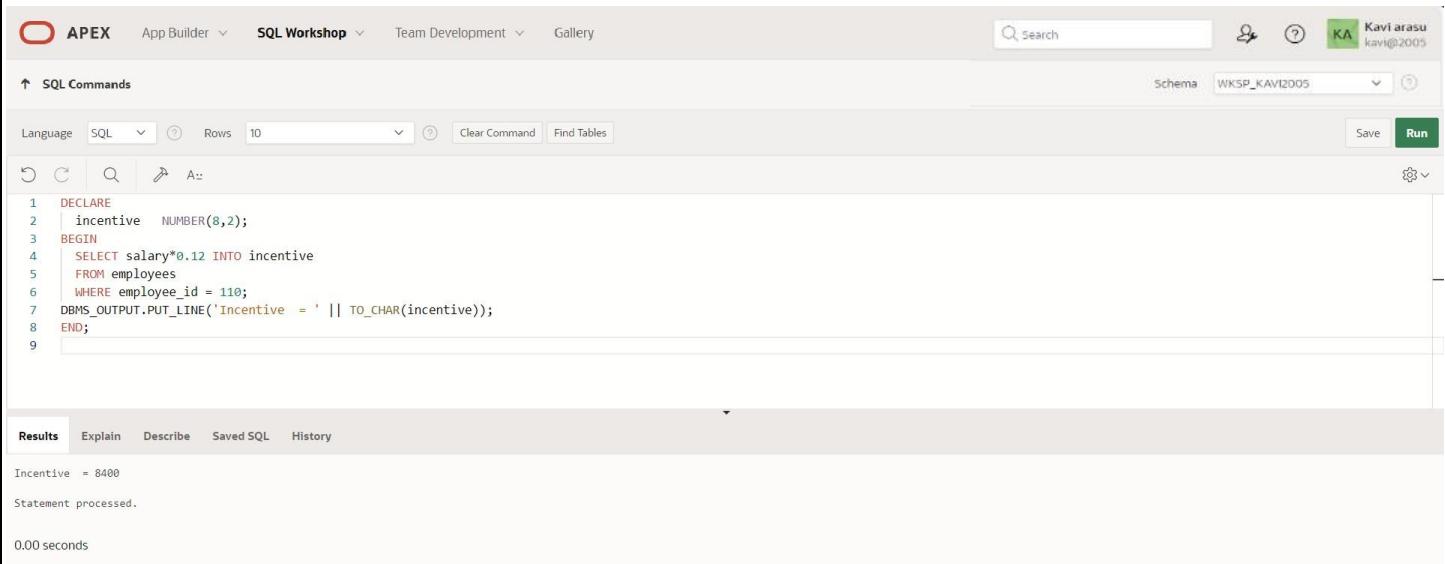
**DATE:**

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'Kavi arasu kavi@2005', and a 'Run' button. The main area is titled 'SQL Commands'. The code entered is a PL/SQL block to calculate an incentive for employee ID 110 and output it. The code is as follows:

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

Below the code, the 'Results' tab is selected. The output shows the result of the query: 'Incentive = 8400'. The status message 'Statement processed.' and execution time '0.00 seconds' are also displayed.

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, the following PL/SQL block is entered:

```
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 with a yellow box highlighting the error message:

```
Error at line 4/25: ORA-06550: line 4, column 25:
PLS-00201: identifier 'Welcome' must be declared
ORA-06512: at "SYS.WN_V_DBMS_SQL_APEX_230200", line 801
ORA-06550: line 4, column 3:
PL/SQL: Statement ignored
```

Below the error message, the PL/SQL block is repeated:

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

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

```
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 with a yellow box highlighting the error message:

```
Error at line 4/25: ORA-06550: line 4, column 25:
PLS-00201: identifier 'Welcome' must be declared
ORA-06512: at "SYS.WN_V_DBMS_SQL_APEX_230200", line 801
ORA-06550: line 4, column 3:
PL/SQL: Statement ignored
```

Below the error message, the PL/SQL block is repeated:

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

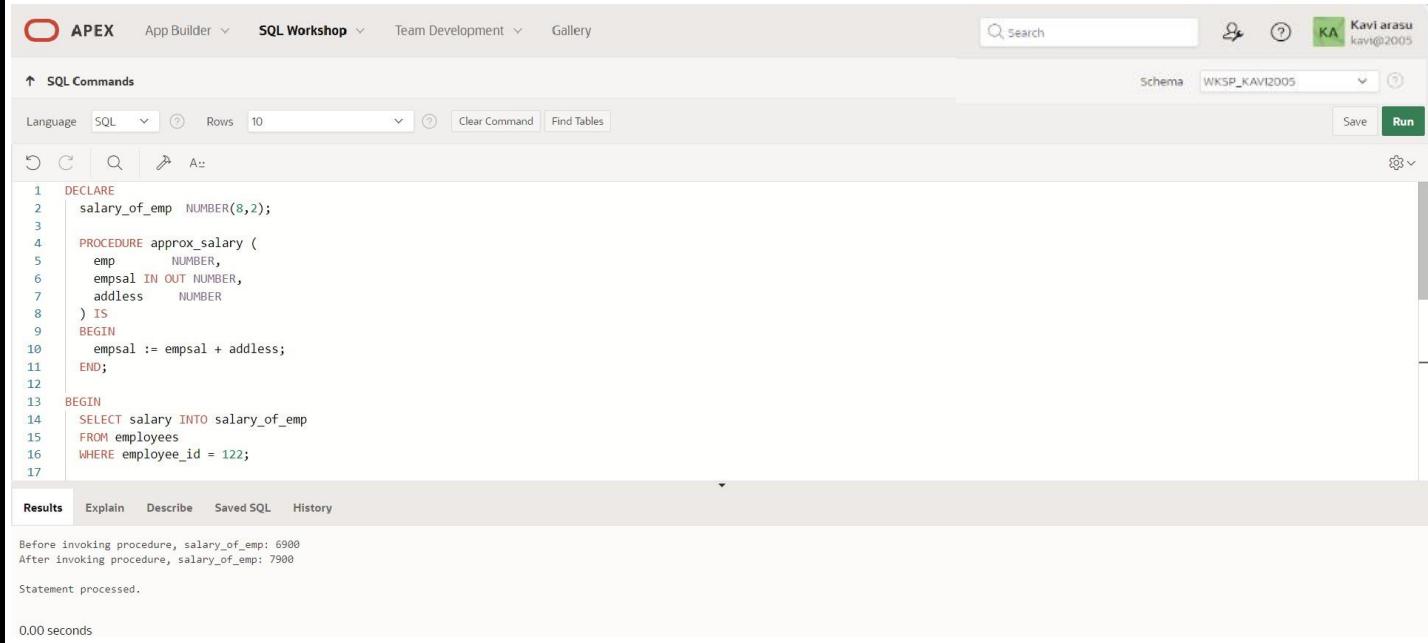
3.) Write a PL/SQL block to adjust the salary of the employee whose ID 122.

#### QUERY:

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

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

#### OUTPUT:



The screenshot shows the Oracle 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 WKS\_P\_KAVI2005. The main area displays the PL/SQL code with line numbers. The code declares a variable salary\_of\_emp, defines a procedure approx\_salary with parameters emp, empsal (IN OUT), and addless, and then executes a SELECT statement to find the salary for employee\_id 122. It also calls the procedure approx\_salary with arguments 100, salary\_of\_emp, and 1000. The bottom section shows the results of the execution, including the output of the DBMS\_OUTPUT.PUT\_LINE statements and the execution statistics.

```
1  DECLARE
2      salary_of_emp NUMBER(8,2);
3
4  PROCEDURE approx_salary (
5      emp      NUMBER,
6      empsal IN OUT NUMBER,
7      addless  NUMBER
8  ) IS
9
10 BEGIN
11     empsal := empsal + addless;
12
13 BEGIN
14     SELECT salary INTO salary_of_emp
15     FROM employees
16     WHERE employee_id = 122;
17 
```

Results Explain Describe Saved SQL History

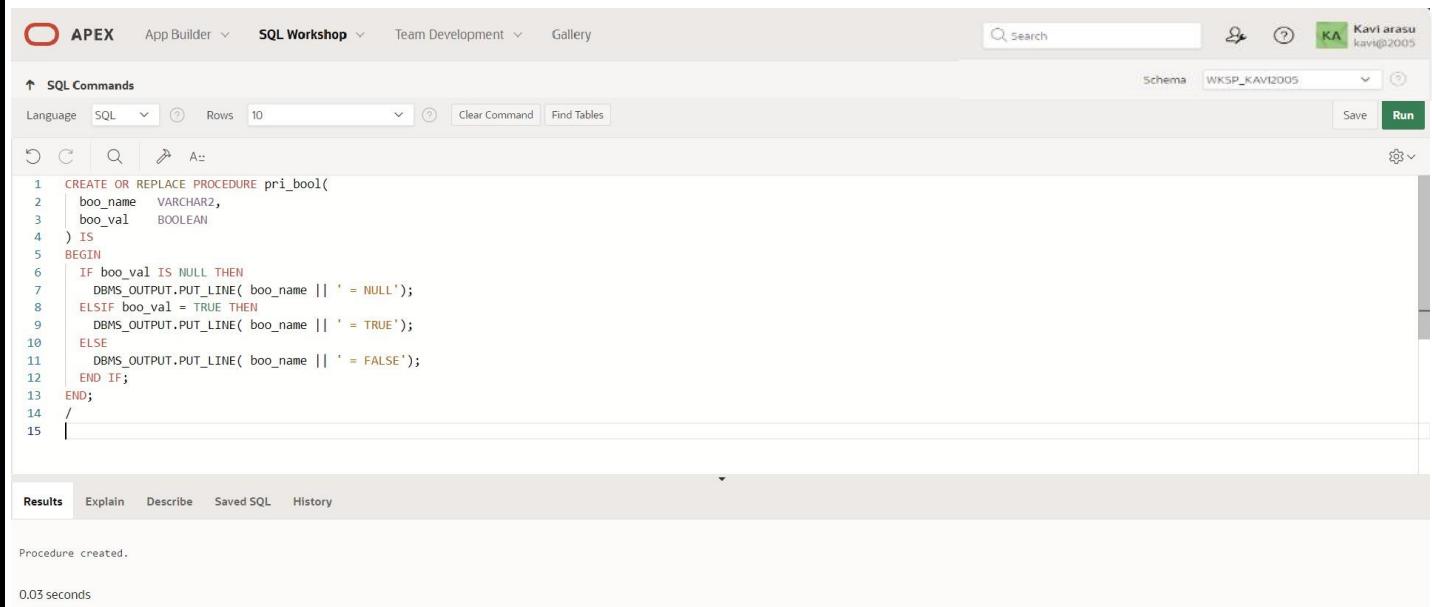
Before invoking procedure, salary\_of\_emp: 6900  
After invoking procedure, salary\_of\_emp: 7900  
Statement processed.  
0.00 seconds

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

#### QUERY:

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

#### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. The code for the procedure 'pri\_bool' is pasted into the command window. The code is as follows:

```
1 CREATE OR REPLACE PROCEDURE pri_bool(
2   boo_name  VARCHAR2,
3   boo_val   BOOLEAN
4 ) IS
5 BEGIN
6   IF boo_val IS NULL THEN
7     DBMS_OUTPUT.PUT_LINE(boo_name || ' = NULL');
8   ELSIF boo_val = TRUE THEN
9     DBMS_OUTPUT.PUT_LINE(boo_name || ' = TRUE');
10  ELSE
11    DBMS_OUTPUT.PUT_LINE(boo_name || ' = FALSE');
12  END IF;
13 END;
14 /
15 |
```

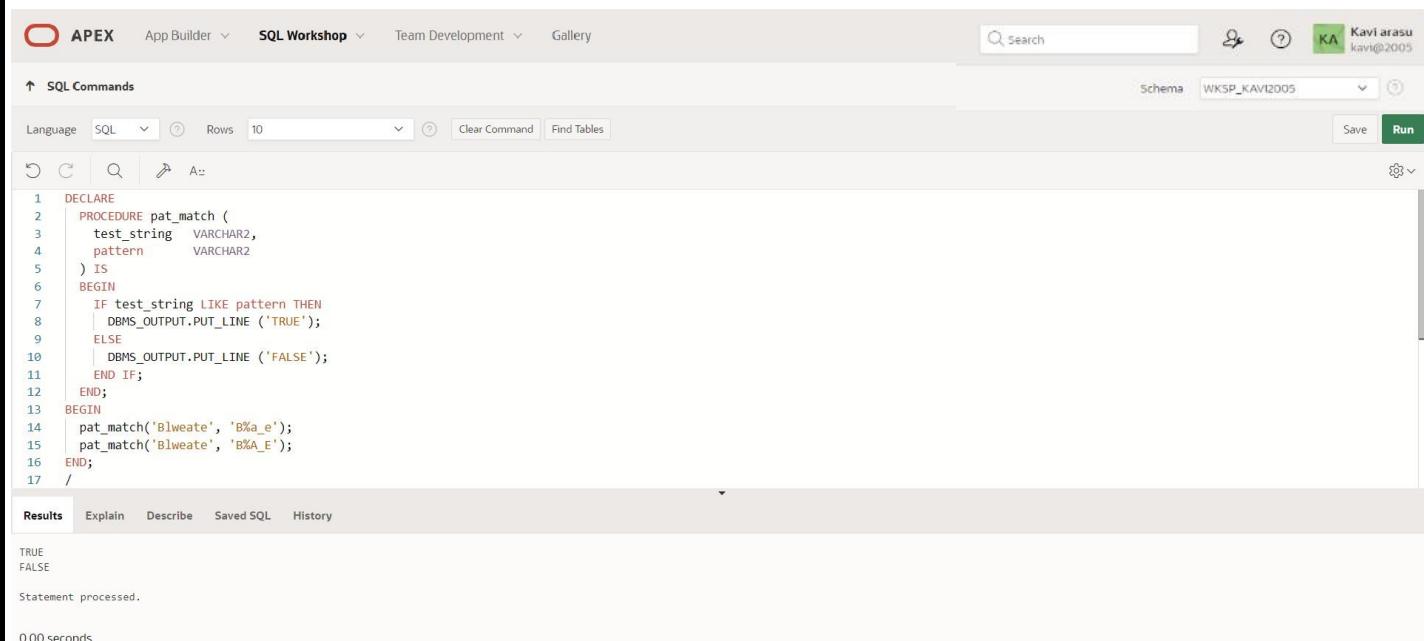
Below the command window, there is a 'Results' tab which displays the message 'Procedure created.' and a time of '0.03 seconds'.

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 (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'Kavi arasu kavi@2005', and a 'Run' button. The main area is titled 'SQL Commands'. The code editor displays the PL/SQL block provided in the question. The code is numbered from 1 to 17. The 'Results' tab is selected at the bottom, showing the output: 'TRUE' and 'FALSE'. Below the results, it says 'Statement processed.' and '0.00 seconds'.

```
1  DECLARE
2    PROCEDURE pat_match (
3      test_string  VARCHAR2,
4      pattern      VARCHAR2
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  /
```

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:

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 'Kavi arasu' with the schema 'WKSP\_KAVI2005'. The main area is titled 'SQL Commands' with a 'Run' button. The code area contains the PL/SQL block from the question, with line numbers 1 through 17. The results tab at the bottom shows the output of the DBMS\_OUTPUT.PUT\_LINE statements: 'num\_small = 5' and 'num\_large = 8'. The status bar indicates 'Statement processed.' and a execution time of '0.00 seconds'.

```
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;
```

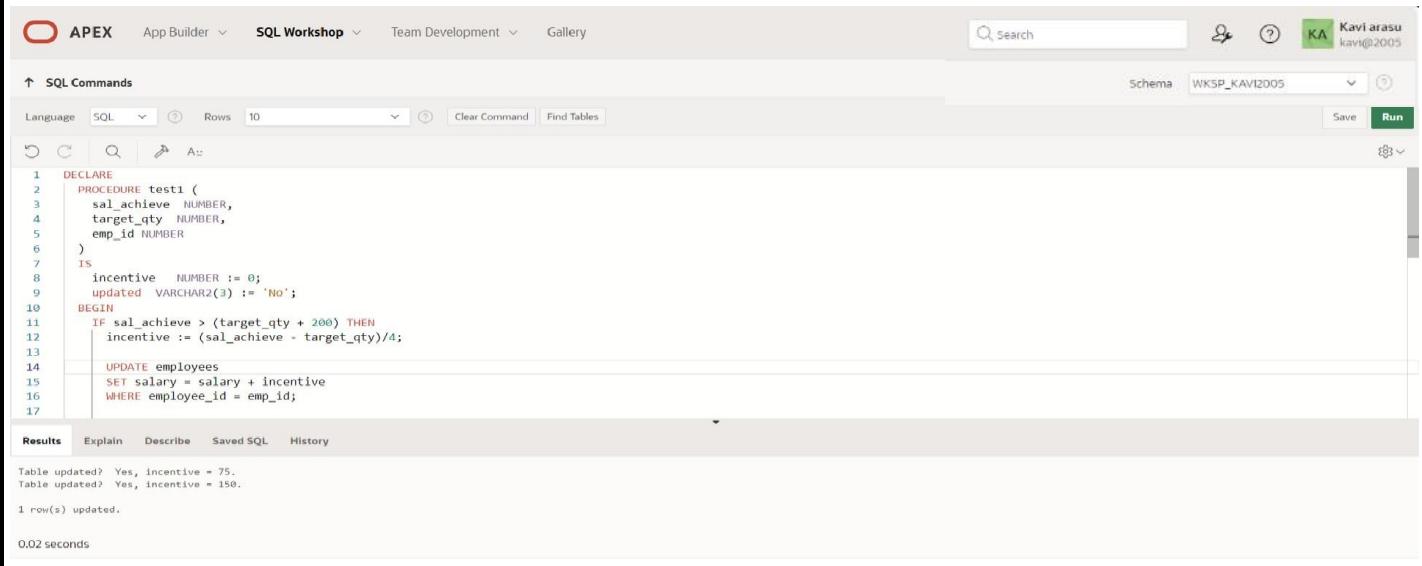
BEGIN

```
test1(2300, 2000, 144);
test1(3600, 3000, 145);
```

END;

/

#### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a user profile 'KA Kaviarasu kavi@2005' and a 'Run' button. The main area is titled 'SQL Commands' with tabs for 'Language' (set to 'SQL'), 'Rows' (set to 10), and 'Clear Command'. Below this is a code editor window containing the provided PL/SQL code. The code is numbered from 1 to 17. The output pane at the bottom shows the results of the execution: two rows were updated, with incentives of 75 and 150 respectively. The total execution time was 0.02 seconds.

```
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     END IF;
17     DBMS_OUTPUT.PUT_LINE (
18         'Table updated? ' || updated || ',' ||
19         'incentive = ' || incentive || '.'
20     );
21 END test1;
```

Results Explain Describe Saved SQL History

Table updated? Yes, incentive = 75.  
Table updated? Yes, incentive = 150.

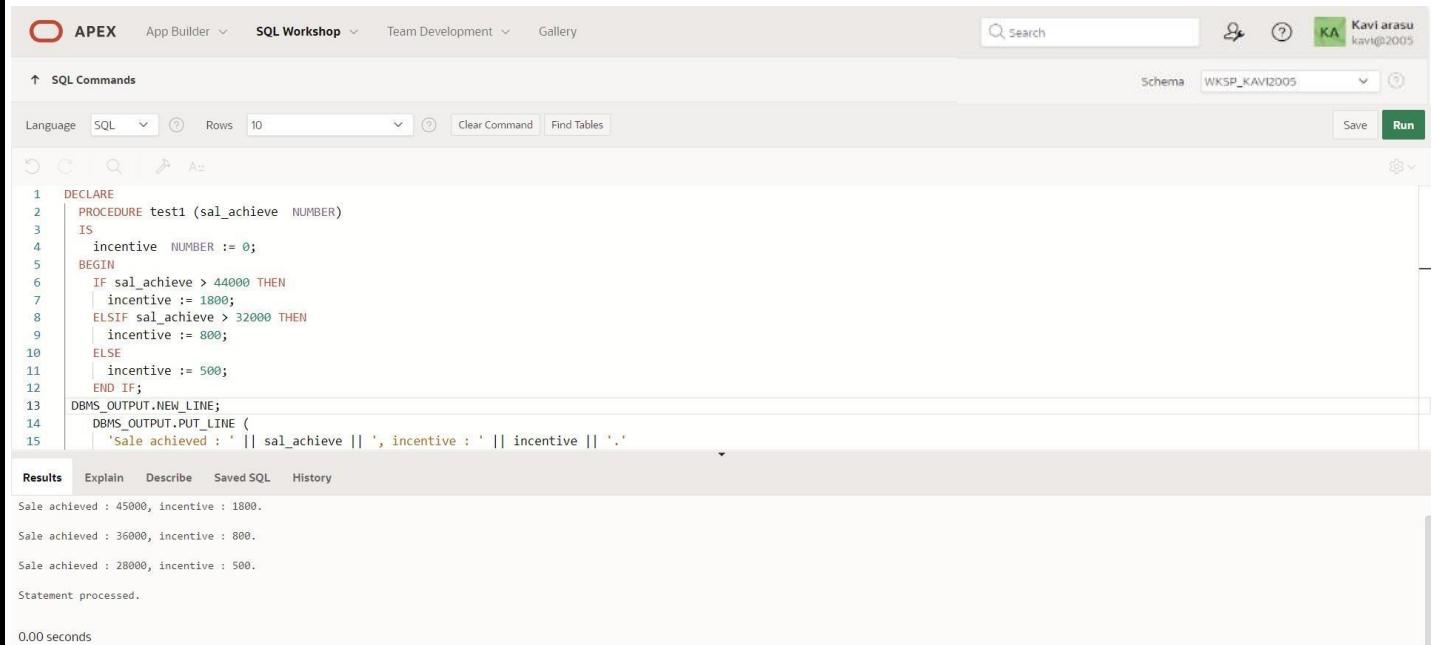
1 row(s) updated.

0.02 seconds

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

**QUERY:**

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



The screenshot shows the Oracle 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\_KAVI2005. The main area displays the PL/SQL code for the test1 procedure. Below the code, the Results tab is active, showing the output of three executions of the procedure with input values 45000, 36000, and 28000, resulting in incentives of 1800, 800, and 500 respectively. The status bar at the bottom indicates 0.00 seconds.

```
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 || '.'
```

Results Explain Describe Saved SQL History

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

0.00 seconds

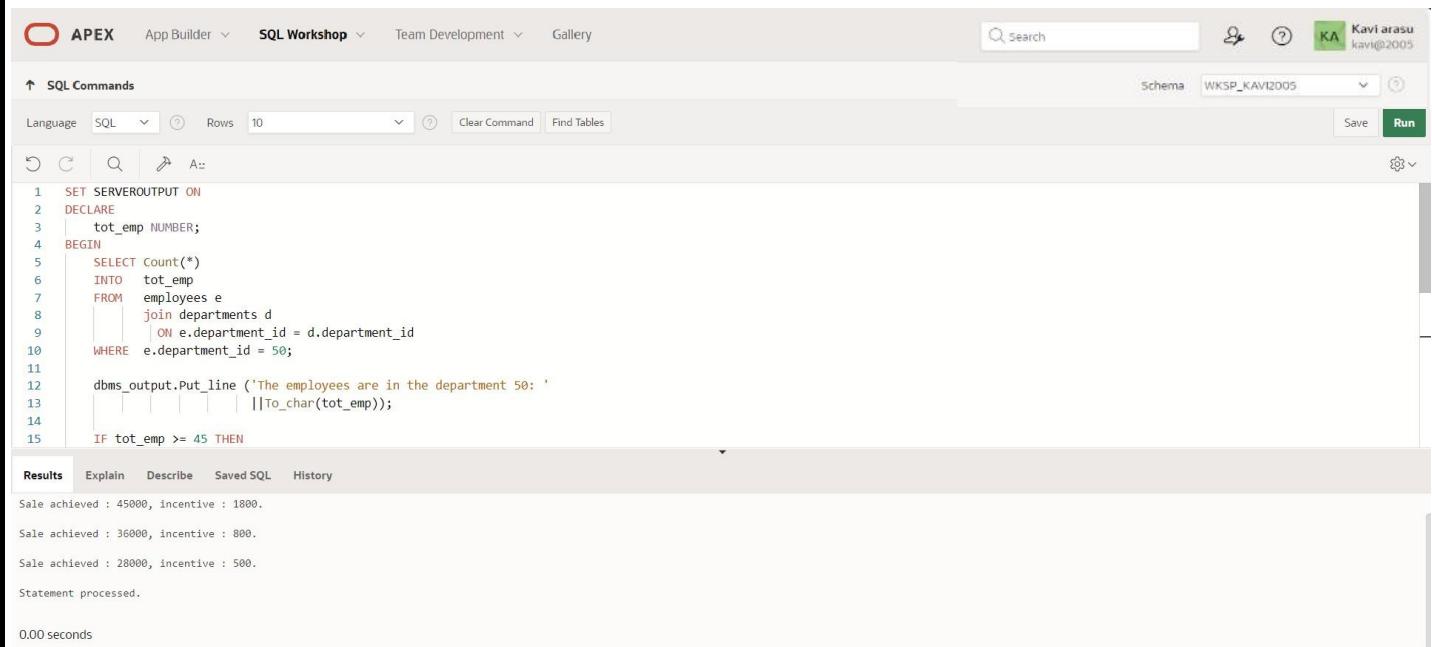
9.) Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

**QUERY:**

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

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands tab, a PL/SQL block is being run. The code counts employees in department 50 and checks for vacancies. The output window shows the results of the query and the execution message.

```
1 SET SERVEROUTPUT ON
2 DECLARE
3     tot_emp NUMBER;
4 BEGIN
5     SELECT Count(*)
6         INTO tot_emp
7         FROM employees e
8         join departments d
9             ON e.department_id = d.department_id
10    WHERE e.department_id = 50;
11
12    dbms_output.Put_line ('The employees are in the department 50: '
13        ||To_char(tot_emp));
14
15    IF tot_emp >= 45 THEN

```

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

0.00 seconds

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

**QUERY:**

DECLARE

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

BEGIN

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

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

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

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

```
ELSE
```

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

```
END IF;
```

```
END;
```

```
/
```

**OUTPUT:**

```
1  DECLARE
2      tot_emp NUMBER;
3      get_dep_id NUMBER;
4
5  BEGIN
6      get_dep_id := 80;
7      SELECT Count(*)
8      INTO tot_emp
9      FROM employees e
10     join departments d
11    ON e.department_id = d.dept_id
12   WHERE e.department_id = get_dep_id;
13
14  dbms_output.Put_line ('The employees are in the department'||get_dep_id||' is: '
15                      ||To_char(tot_emp));
```

The employees are in the department 80 is: 4  
There are 41 vacancies in department 80  
Statement processed.  
0.03 seconds

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

**QUERY:**

DECLARE

v\_employee\_id employees.employee\_id%TYPE;

v\_full\_name employees.first\_name%TYPE;

v\_job\_id employees.job\_id%TYPE;

v\_hire\_date employees.hire\_date%TYPE;

v\_salary employees.salary%TYPE;

CURSOR c\_employees IS

```
SELECT employee_id, first_name || ' ' || last_name AS full_name, job_id, hire_date, salary
FROM employees;
```

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Employee ID | Full Name | Job Title | Hire Date | Salary');

DBMS\_OUTPUT.PUT\_LINE('-----');

OPEN c\_employees;

FETCH c\_employees INTO v\_employee\_id, v\_full\_name, v\_job\_id, v\_hire\_date, v\_salary;

WHILE c\_employees%FOUND LOOP

```
DBMS_OUTPUT.PUT_LINE(v_employee_id || ' ' || v_full_name || ' ' || v_job_id || ' ' ||
v_hire_date || ' ' || v_salary);
```

FETCH c\_employees INTO v\_employee\_id, v\_full\_name, v\_job\_id, v\_hire\_date, v\_salary;

END LOOP;

CLOSE c\_employees;

END;

/

**OUTPUT:**

```
1  DECLARE
2  v_employee_id employees.employee_id%TYPE;
3  v_full_name employees.first_name%TYPE;
4  v_job_id employees.job_id%TYPE;
5  v_hire_date employees.hire_date%TYPE;
6  v_salary employees.salary%TYPE;
7  CURSOR c_employees IS
8    SELECT employee_id, first_name || ' ' || last_name AS full_name, job_id, hire_date, salary
9    FROM employees;
10 BEGIN
11   DBMS_OUTPUT.PUT_LINE('Employee ID | Full Name | Job Title | Hire Date | Salary');
12   DBMS_OUTPUT.PUT_LINE('-----');
13   OPEN c_employees;
```

Employee ID | Full Name | Job Title | Hire Date | Salary

-----

110	John Smith	sales_rep	02/28/1995	70000
125	Emily Johnson	hr_rep	04/26/1998	5000
122	Jaunty Janu	ac_account	03/05/2024	6900
114	den Davies	st_clerk	02/03/1999	11000
142	Jane Doe	ac_account	03/05/1997	30000
115	Vijaya Mohan	st_clerk	02/22/1998	4000

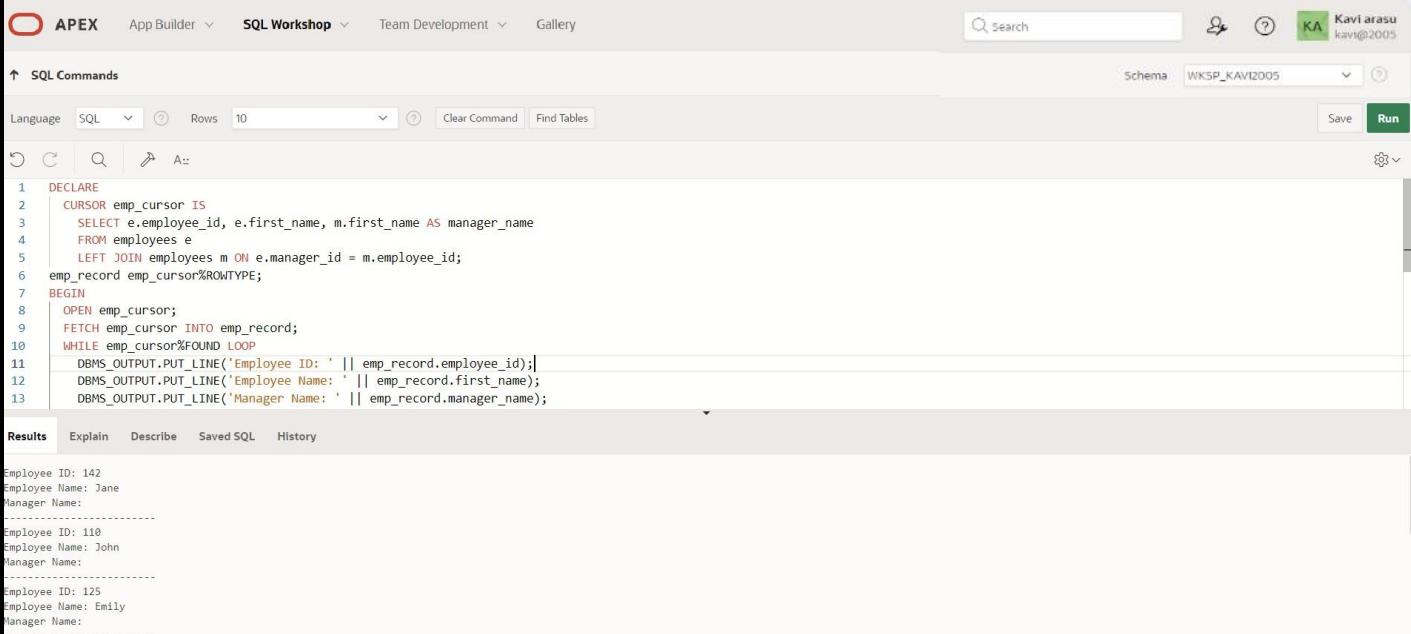
Statement processed.

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user Kavi arasu (kavi@2005) and a schema dropdown set to WKSP\_KAVI2005. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab displays the PL/SQL code from the previous step. The Results tab shows the output of the DBMS\_OUTPUT.PUT\_LINE statements, listing three employees with their respective IDs, names, and manager names, separated by horizontal dashes.

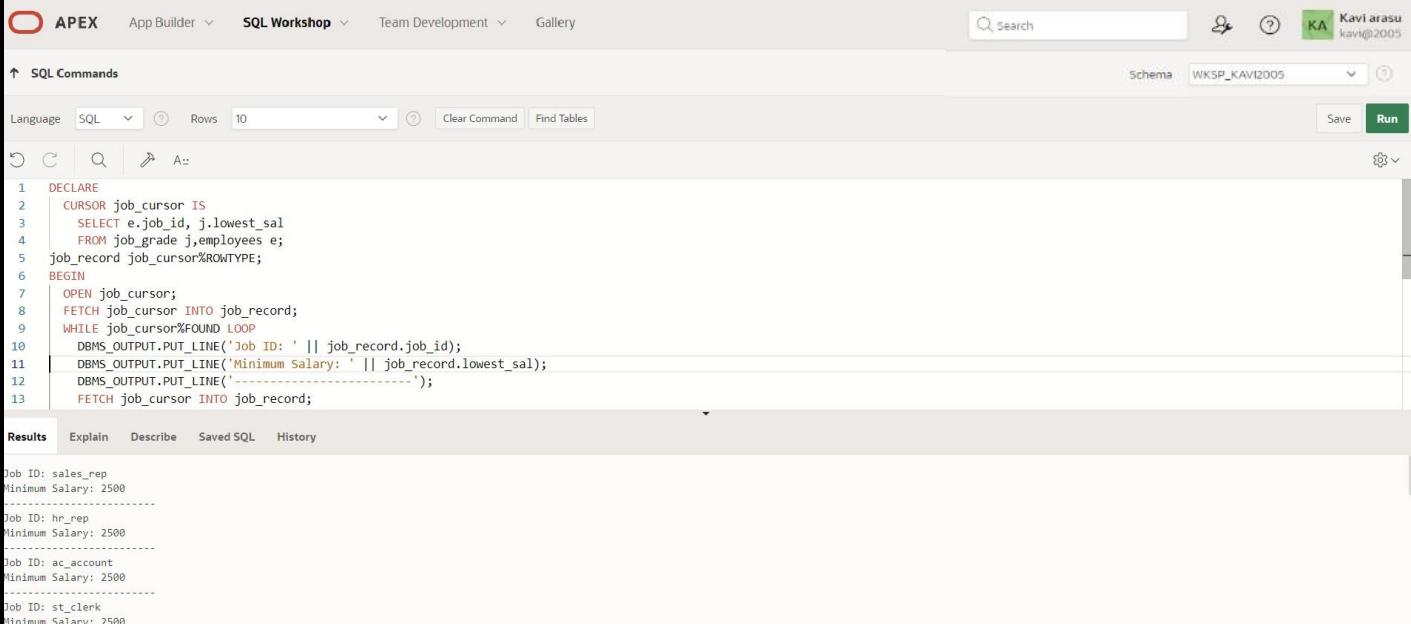
Employee ID	Employee Name	Manager Name
142	Jane	
110	John	
125	Emily	

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

#### QUERY:

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

#### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a user profile for 'Kavi arasu' and a workspace named 'WKSP\_KAVI2005'. The main area has tabs for SQL Commands, Explain, Describe, Saved SQL, and History. The SQL Commands tab displays the PL/SQL code with line numbers. The Results tab shows the output of the executed code, which lists job IDs, titles, and minimum salaries for various jobs.

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

```

Job ID	Title	Minimum Salary
sales_rep	Sales Representative	2500
hr_rep	Human Resources Representative	2500
ac_account	Account Executive	2500
st_clerk	Stock Clerk	2500

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

**QUERY:**

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

## OUTPUT:

The screenshot shows the Oracle 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, a user icon for 'Kavilarasu kavi@2005', and buttons for Save and Run. Below the tabs, it says 'SQL Commands' and shows 'Language: SQL', 'Rows: 10', and buttons for Clear Command and Find Tables. The main area contains a PL/SQL script with numbered lines 1 through 17. Lines 1-9 define a cursor 'employees\_cur' that selects employee\_id, last\_name, job\_id, and start\_date from the employees table, naturally joining it with the job\_history table. Lines 10-14 begin a block and use dbms\_output.Put\_line to print the employee ID, last name, job ID, and start date, each padded to 25 characters. Line 15 prints a separator line, and line 16 prints another separator line. Line 17 ends the block. Below the code, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected and shows a table with four columns: Employee ID, Last Name, Job Id, and Start Date. The data is as follows:

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

Statement processed.

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

## QUERY:

DECLARE

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

## OUTPUT:

The screenshot shows the Oracle 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, a user icon for 'Kavi arasu' (kavi@2005), and a 'Run' button.

In the main area, there's a toolbar with icons for copy, paste, search, and refresh. Below it is a dropdown menu for 'Language' (SQL) and buttons for 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. The 'Schema' dropdown is set to 'WKSP\_KAVI2005'.

The code editor contains a PL/SQL block:

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

```

The results tab is selected, showing the output of the executed code:

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

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

Viva(5)	
Total (15)	
Faculty Signature	

**RESULT:**

## PROCEDURES AND FUNCTIONS

EX.NO: 17

DATE:

1.) Factorial of a number using function.

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'Application', 'SQL Workshop', 'Team Development', 'Galleries', 'Search', 'XA', and 'Kaviarase 6-Jun-2025'. The main area is titled 'SQL Commands' with tabs for 'Language', 'SQL', 'Raw', 'Explained', 'Describe', 'Saved SQL', and 'History'. The code area contains 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;
```

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

```
100
Statement processed.
```

Execution time: 0.00 seconds.

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

**QUERY:**

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

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

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

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

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

## OUTPUT:

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

```
1 CREATE TABLE books {
2   book_id NUMBER PRIMARY KEY,
3   title VARCHAR(100),
4   author VARCHAR(100),
5   year_published NUMBER
6 };
7 INSERT INTO books (book_id, title, author, year_published) VALUES (1, '1984', 'George Orwell', 1949);
8 INSERT INTO books (book_id, title, author, year_published) VALUES (2, 'To Kill a Mockingbird', 'Harper Lee', 1960);
9 INSERT INTO books (book_id, title, author, year_published) VALUES (3, 'The Great Gatsby', 'F. Scott Fitzgerald', 1925);
10
11 CREATE OR REPLACE PROCEDURE get_book_info (
12   p_book_id IN NUMBER,
13   p_title OUT VARCHAR2,
14   p_author OUT VARCHAR2
15 );
16
17 /
```

In the 'Results' tab, the output of the last executed statement is shown:

```
Title: 1984
Author: George Orwell
Year Published: 1949
Statement processed.
```

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

## RESULT:

## TRIGGER

EX\_NO: 18

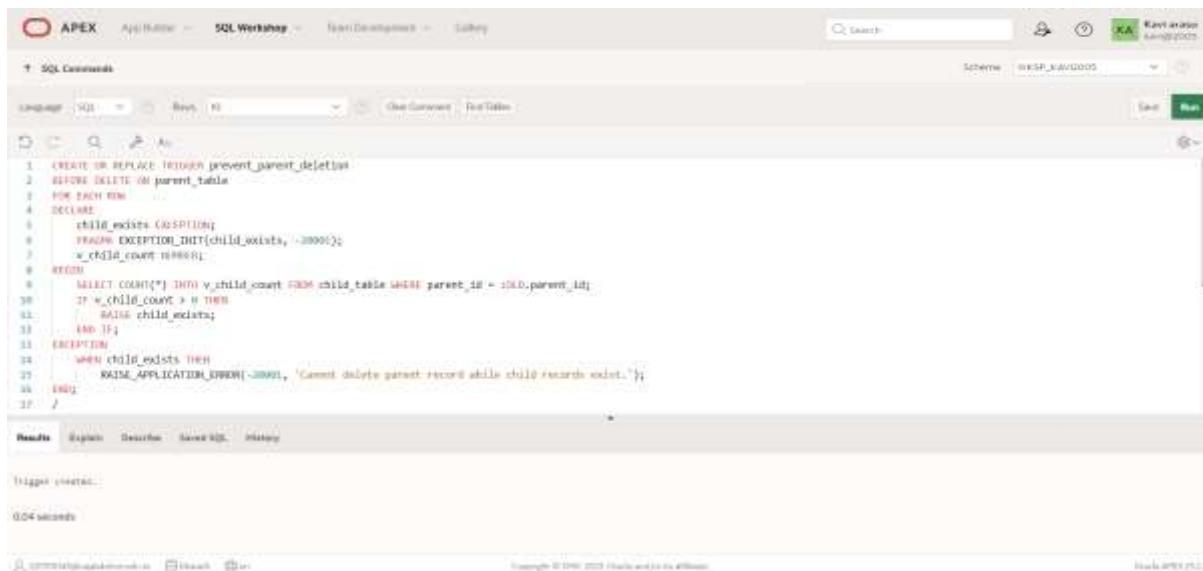
DATE:

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' and 'SQL Workshop' are selected. The main area displays the PL/SQL code for the trigger. At the bottom of the code editor, there are tabs for 'Results', 'Explain', 'Describe', 'Save As...', and 'History'. Below the tabs, the message 'Trigger created.' is displayed, along with a timestamp '0.04 seconds'. The status bar at the bottom right shows 'Oracle Database 11g'.

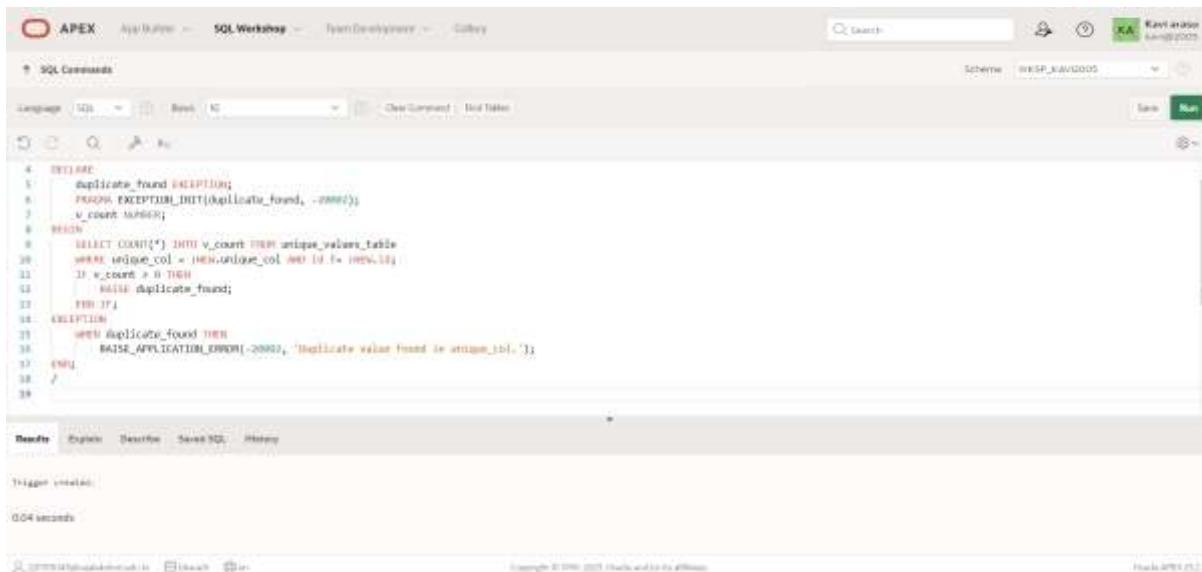
```
1 CREATE OR REPLACE TRIGGER prevent_parent_deletion
2 BEFORE DELETE ON parent_table
3 FOR EACH ROW
4 DECLARE
5     child_exists EXCEPTION;
6     PRAGMA EXCEPTION_INIT(child_exists, -20001);
7     v_child_count NUMBER;
8 BEGIN
9     SELECT COUNT(*) INTO v_child_count FROM child_table WHERE parent_id = :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;
17 /
```

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. Below it, 'SQL Workshop' is active. The main area contains the SQL code for the trigger. At the bottom of the code editor, there are tabs for 'Results', 'Explain', 'Describe', 'Save SQL', and 'History'. A message 'Trigger created.' is displayed above the results table. The results table is empty, showing only the header row.

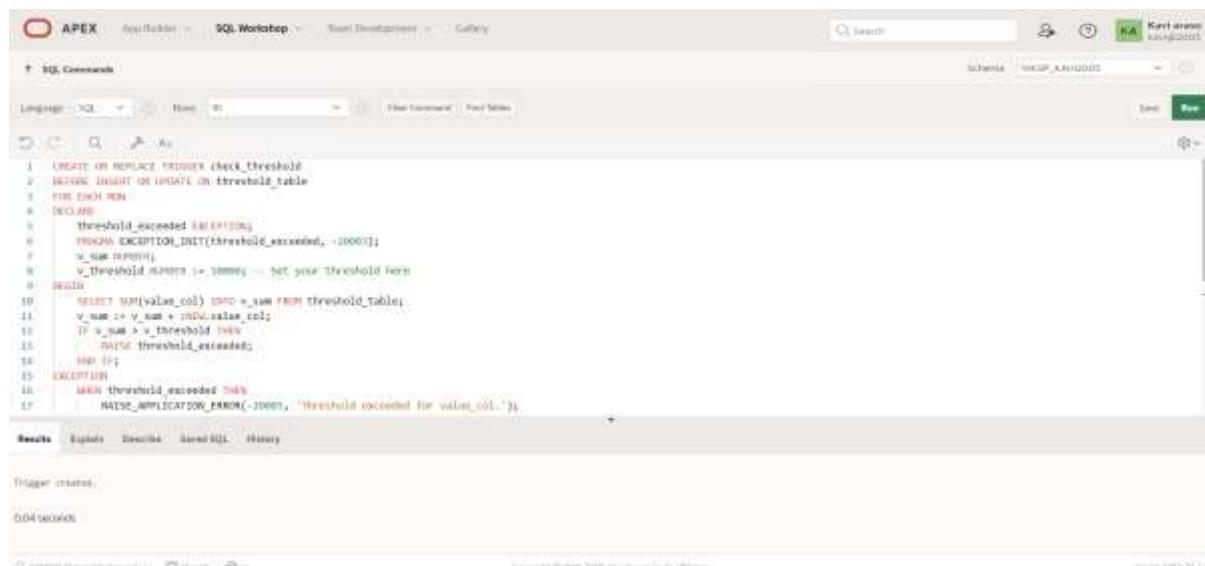
```
4  DECLARE
5      duplicate_found EXCEPTION;
6      PRAGMA EXCEPTION_INIT(duplicate_found, -20002);
7      v_count NUMBER;
8  BEGIN
9      SELECT COUNT(*) INTO v_count FROM unique_values_table
10     WHERE unique_col = :NEW.unique_col AND id != :NEW.id;
11     IF v_count > 0 THEN
12         RAISE duplicate_found;
13     END IF;
14  EXCEPTION
15     WHEN duplicate_found THEN
16         RAISE_APPLICATION_ERROR(-20002, 'Duplicate value found in unique_col.');
17 END;
18 /
19
```

**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 Developer interface with the SQL Worksheet tab selected. The schema is set to 'KARAN\_KAHLADEV'. The code area contains the PL/SQL trigger definition provided above. The results pane at the bottom shows the output: 'Trigger created.' followed by the timestamp '00:45:00.000'.

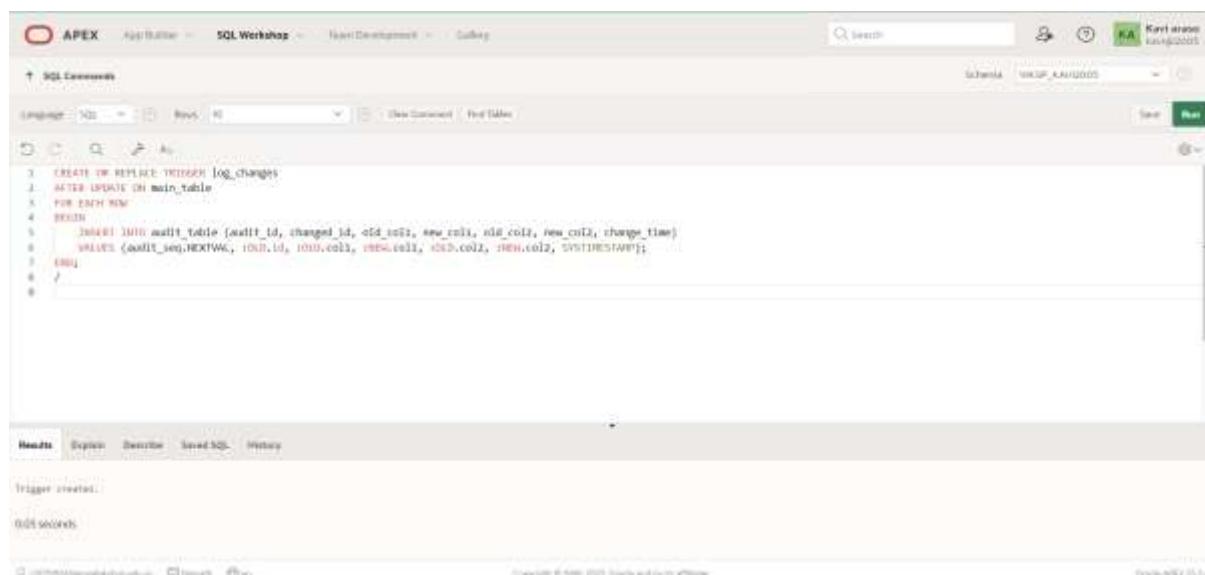
```
1 CREATE OR REPLACE TRIGGER check_threshold
2 BEFORE INSERT OR UPDATE ON threshold_table
3 FOR EACH ROW
4 DECLARE
5     threshold_exceeded EXCEPTION;
6     PRAGMA EXCEPTION_INIT(threshold_exceeded, -20003);
7     v_sum NUMBER;
8     v_threshold NUMBER := 10000; -- Set your threshold here
9 BEGIN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

**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' and 'SQL Workshop' are selected. The main area displays the SQL code 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, change_time)
6     VALUES (audit_seq.NEXTVAL, :OLD.id, :OLD.col1, :NEW.col1, :OLD.col2, :NEW.col2,
7 SYSTIMESTAMP);
8 END;
9 /
```

Below the code, the 'Results' tab is active, showing the output: "Trigger created." and "0.05 seconds". The bottom status bar indicates the session is connected to 'localhost:1521' and shows the version 'Oracle Database 19c'.

**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 APEX SQL Workshop interface. In the top navigation bar, 'APEX' and 'SQL Workshop' are selected. The main area is titled 'SQL Commands'. The code for the trigger is pasted into the command editor. The 'Results' tab at the bottom shows the output: 'Trigger created.' and '0.00 seconds.' The status bar at the bottom right indicates 'Oracle APEX 21.2.4'.

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

**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**

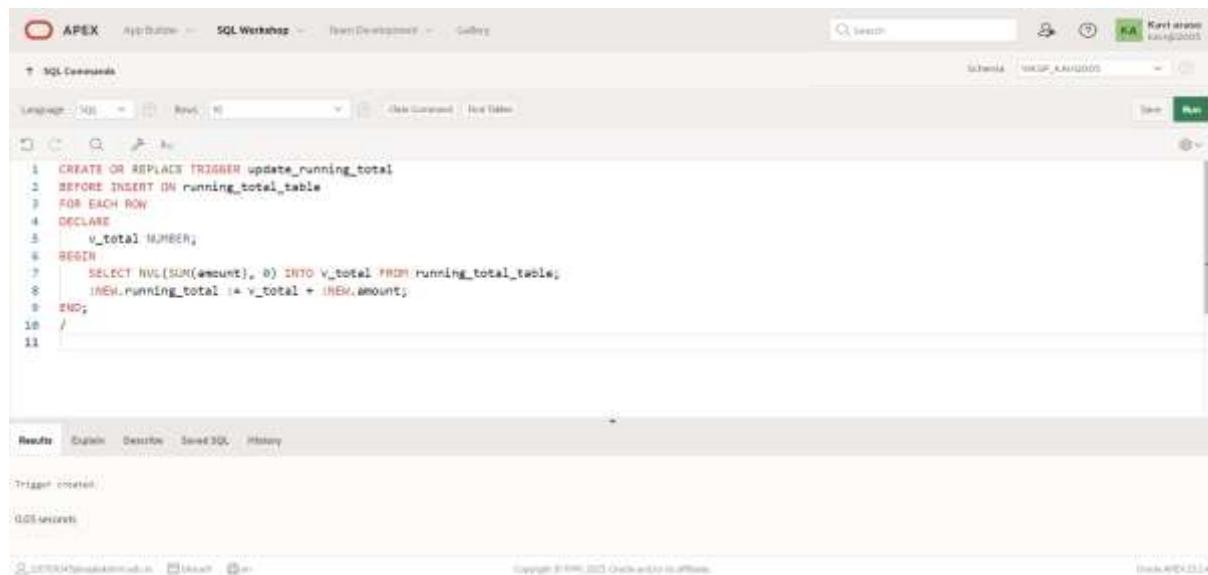
**QUERY:**

```

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 APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, the 'SQL Workshop' tab is active, followed by 'Item Development' and 'Gallery'. On the right side, there's a user profile for 'Kavirajane' and a search bar. The main area is titled 'SQL Commands' and shows the trigger creation code. At the bottom, the 'Results' tab is selected, displaying the message 'Trigger created.' and a time of '0.05 seconds'. The footer includes the URL 'https://apex.kavirajane.com', the copyright notice 'Copyright © 2023 Oracle and/or its affiliates.', and the version 'Oracle APEX 23.1.4'.

```

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 /
11

```

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

## QUERY:

```

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

```

## OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the central workspace, the trigger definition is pasted. The code is as follows:

```

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

```

Below the code, the message "Trigger created." is displayed. At the bottom of the window, the status bar shows "0.05 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:**



KAVIARASU M

MongoDB

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

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:**



KAVIARASU M

MongoDB

```
1 db.restaurants.find( { grades: { $elemMatch: { grade: "A", score: 11, date: ISODate("2014-08-11T00:00:00Z") } } } , 2 { restaurant_id: 1, name: 1, grades: 1 } );
```

Run Save

Output

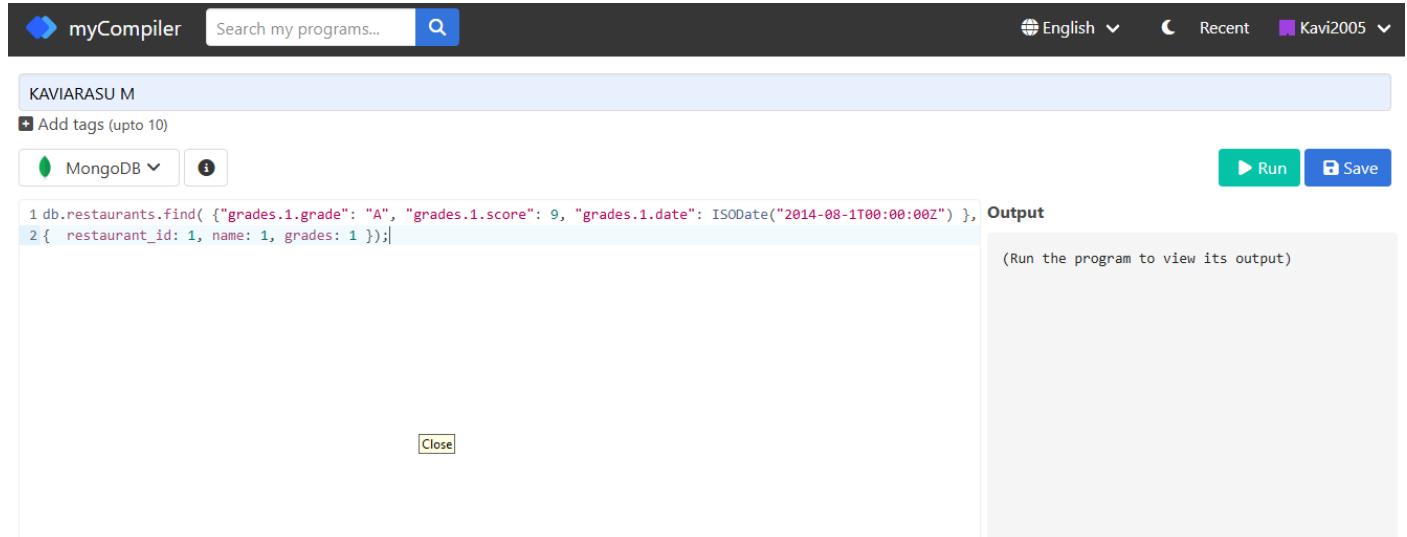
```
mycompiler_mongodb> ...
mycompiler_mongodb>
[Execution complete with exit code 0]
```

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

#### QUERY:

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

#### OUTPUT:



The screenshot shows the myCompiler interface. At the top, there's a navigation bar with the logo, search bar, language (English), recent projects, and user profile (Kavi2005). Below the bar, the title 'KAVIARASU M' and a 'Add tags (upto 10)' button are visible. A toolbar includes a MongoDB dropdown, a help icon, and 'Run' and 'Save' buttons. The main area has a code editor with the following MongoDB query:

```
1 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 } );
```

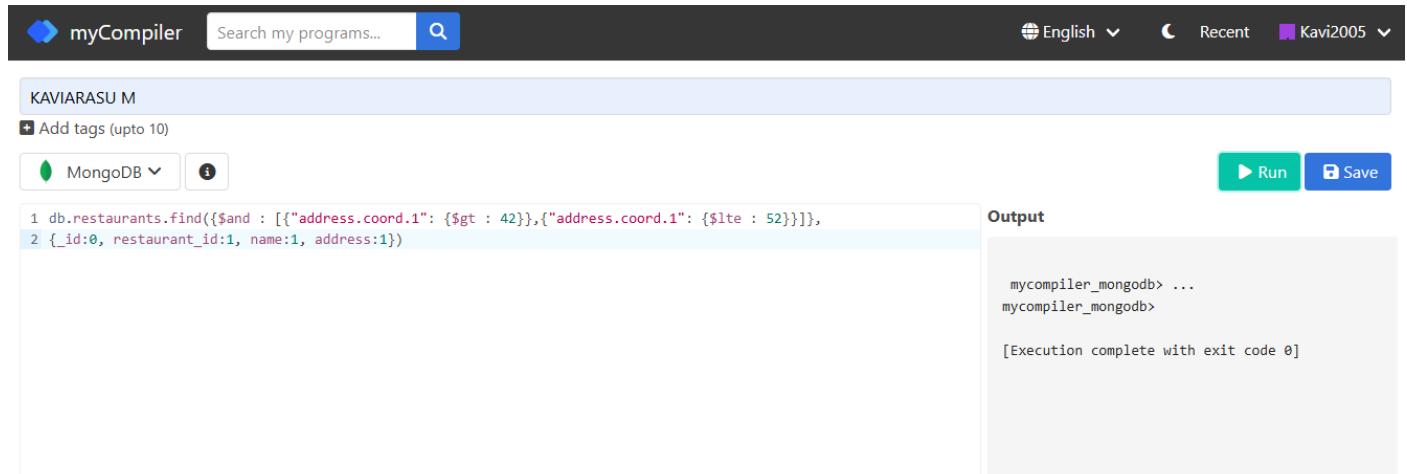
To the right is an 'Output' panel with a placeholder message: '(Run the program to view its output)'. A 'Close' button is at the bottom left of the code editor.

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

#### QUERY:

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

#### OUTPUT:



The screenshot shows the myCompiler interface again. The title 'KAVIARASU M' and code editor are identical to the previous screenshot. The 'Output' panel now displays the results of the executed query:

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

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

#### QUERY:

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

## OUTPUT:

The screenshot shows the myCompiler interface. In the top bar, there is a logo, a search bar with placeholder text "Search my programs...", and language and user settings. Below the header, the user profile "KAVIARASU M" is displayed along with a "Add tags (upto 10)" button. A "MongoDB" dropdown menu is open, and a small info icon is visible. On the right, there are "Run" and "Save" buttons. The main area contains a code editor with the following MongoDB query:

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

To the right of the code editor is the "Output" panel, which displays the results of the executed query:

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

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

## QUERY:

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

## OUTPUT:

The screenshot shows the myCompiler interface. The layout is identical to the previous one, with the "MongoDB" dropdown open and the "Run" button highlighted. The code editor contains the same query as above:

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

The "Output" panel shows the results:

```
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 the myCompiler interface. The "MongoDB" dropdown is open, and the "Run" button is highlighted. The code editor contains the query:

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

The "Output" panel shows the 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 the myCompiler interface. At the top, there's a search bar for programs and a language dropdown set to English. Below the search bar, there's a section for tags and a MongoDB dropdown set to MongoDB. On the right, there are 'Run' and 'Save' buttons. The main area contains a code editor with the following MongoDB query:

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

To the right of the code editor is an 'Output' window. It shows the command being run and the response from the MongoDB shell:

```
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 the myCompiler interface. At the top, there's a search bar for programs and a language dropdown set to English. Below the search bar, there's a section for tags and a MongoDB dropdown set to MongoDB. On the right, there are 'Run' and 'Save' buttons. The main area contains a code editor with the following MongoDB query:

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

To the right of the code editor is an 'Output' window. It shows the command being run and the response from the MongoDB shell:

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

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the myCompiler web application interface. At the top, there's a navigation bar with the logo, search bar, language selection (English), and user profile (Kavi2005). Below the header, the user has selected the MongoDB database. The main area contains a code editor with the following MongoDB query:

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

On the right, there's an "Output" panel showing the results of the query execution:

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

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

### QUERY:

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

### OUTPUT:

The screenshot shows the myCompiler web application interface. At the top, there's a navigation bar with the logo, search bar, language selection (English), and user profile (Kavi2005). Below the header, the user has selected the MongoDB database. The main area contains a code editor with the following MongoDB query:

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

On the right, there's an "Output" panel showing the results of the query execution:

```
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 the myCompiler web application interface. At the top, there's a navigation bar with the logo, search bar, language selection (English), recent projects (Recent), and user profile (Kavi2005). Below the header, the workspace title is "KAVIARASU M". A toolbar includes "Add tags (upto 10)", "MongoDB" dropdown, and "Run/Save" buttons. The code editor contains the following MongoDB query:

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

The output window shows the results of the query execution:

```
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:

The screenshot shows the myCompiler web application interface. At the top, there's a navigation bar with the logo, search bar, language selection (English), recent projects (Recent), and user profile (Kavi2005). Below the header, the workspace title is "KAVIARASU M". A toolbar includes "Add tags (upto 10)", "MongoDB" dropdown, and "Run/Save" buttons. The code editor contains the following MongoDB query:

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

The output window shows the results of the query execution:

```
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:

The screenshot shows the myCompiler web application interface. At the top, there's a navigation bar with the logo, search bar, language selection (English), recent projects (Recent), and user profile (Kavi2005). Below the header, the workspace title is "KAVIARASU M". A toolbar includes "Add tags (upto 10)", "MongoDB" dropdown, and "Run/Save" buttons. The code editor contains the following MongoDB query:

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

The output window shows the results of the query execution:

```
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 the myCompiler interface. In the code editor, the following MongoDB query is written:

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

The output window shows the command being run and the response:

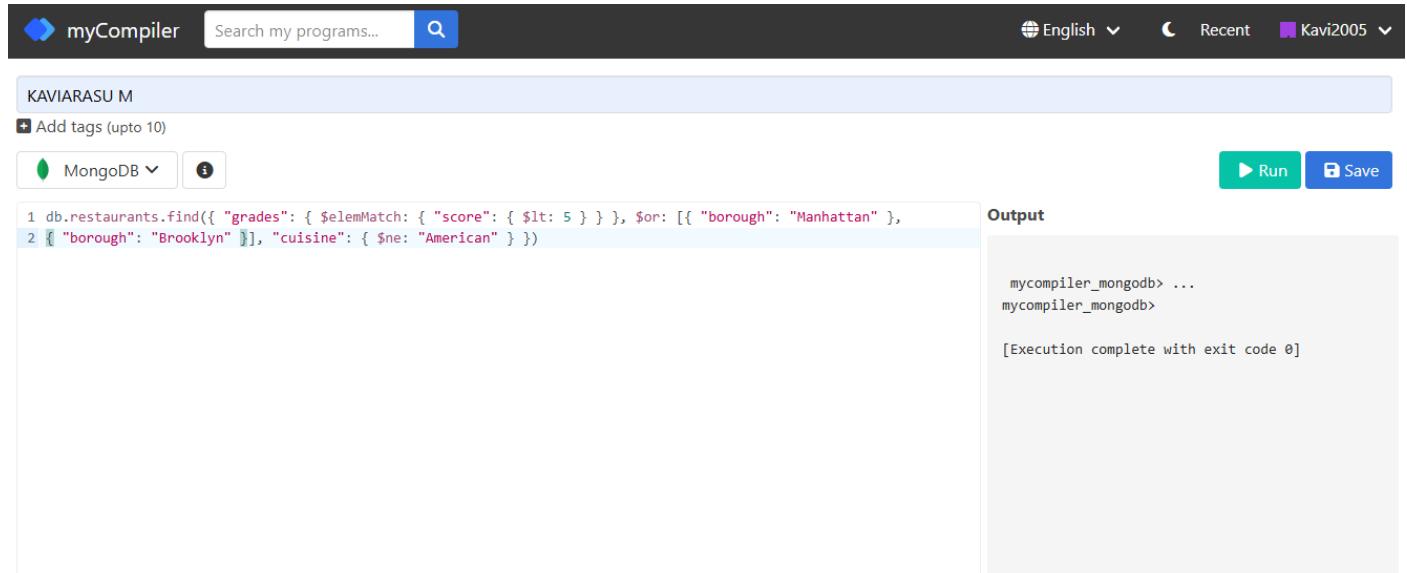
```
mycompiler_mongodb> ...
mycompiler_mongodb>
[Execution complete with exit code 0]
```

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

### QUERY:

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

### OUTPUT:



The screenshot shows the myCompiler interface. In the code editor, the following MongoDB query is written:

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

The output window shows the command being run and the response:

```
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:

KAVIARASU M

+ Add tags (upto 10)

MongoDB Run Save

```
1 db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } } }, { $or: 2 [ { "borough": "Manhattan" }, { "borough": "Brooklyn" } ], "cuisine": 3 { $nin: ["American", "Chinese"] } })
```

Output

```
mycompiler_mongodb> ...
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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

### QUERY:

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

### OUTPUT:

KAVIARASU M

+ Add tags (upto 10)

MongoDB Run Save

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

Output

```
mycompiler_mongodb> ...
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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

### QUERY:

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

### OUTPUT:

myCompiler Search my programs... 

KAVIARASU M  
Add tags (upto 10)

MongoDB  Run Save

```
1 db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }],  
2   "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...  
Add tags (upto 10)

MongoDB  Run Save

```
1 "borough": "Brooklyn" }], "grades.score": { $in: [2, 6] } }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 })| Output
```

mycompiler\_mongodb> ...  
mycompiler\_mongodb>  
[Execution complete with exit code 0]

Redesign the way you jam with FigJam AI.  
 ADS VIA CARBON

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

### QUERY:

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

### OUTPUT:

The screenshot shows a MongoDB query editor interface. At the top, there's a search bar labeled "Enter a title..." and a button "Add tags (upto 10)". Below that is a dropdown "MongoDB" and a help icon. On the right are "Run" and "Save" buttons. The main area is titled "Output" and contains the following text:  
mycompiler\_mongodb> ...  
mycompiler\_mongodb>  
[Execution complete with exit code 0]  
At the bottom right, there's an advertisement for FigJam AI.

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

#### QUERY:

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

#### OUTPUT:

The screenshot shows a MongoDB query editor interface, identical to the one above. It contains the same search bar, tags button, MongoDB dropdown, and Run/Save buttons. The "Output" section shows the same command and its execution results:  
mycompiler\_mongodb> ...  
mycompiler\_mongodb>  
[Execution complete with exit code 0]  
An advertisement for FigJam AI is visible at the bottom right.

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:

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 the myCompiler interface. At the top, there's a search bar for programs and a language dropdown set to MongoDB. Below the search bar, there's a section for adding tags. The main area contains a code editor with the following MongoDB query:

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

On the right side, there's an "Output" window showing the results of the query execution:

```
mycompiler_mongodb> ...
mycompiler_mongodb>
[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 the myCompiler interface. At the top, there's a search bar for programs and a language dropdown set to MongoDB. Below the search bar, there's a section for adding tags. The main area contains a code editor with the following MongoDB query:

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

On the right side, there's an "Output" window showing the results of the query execution:

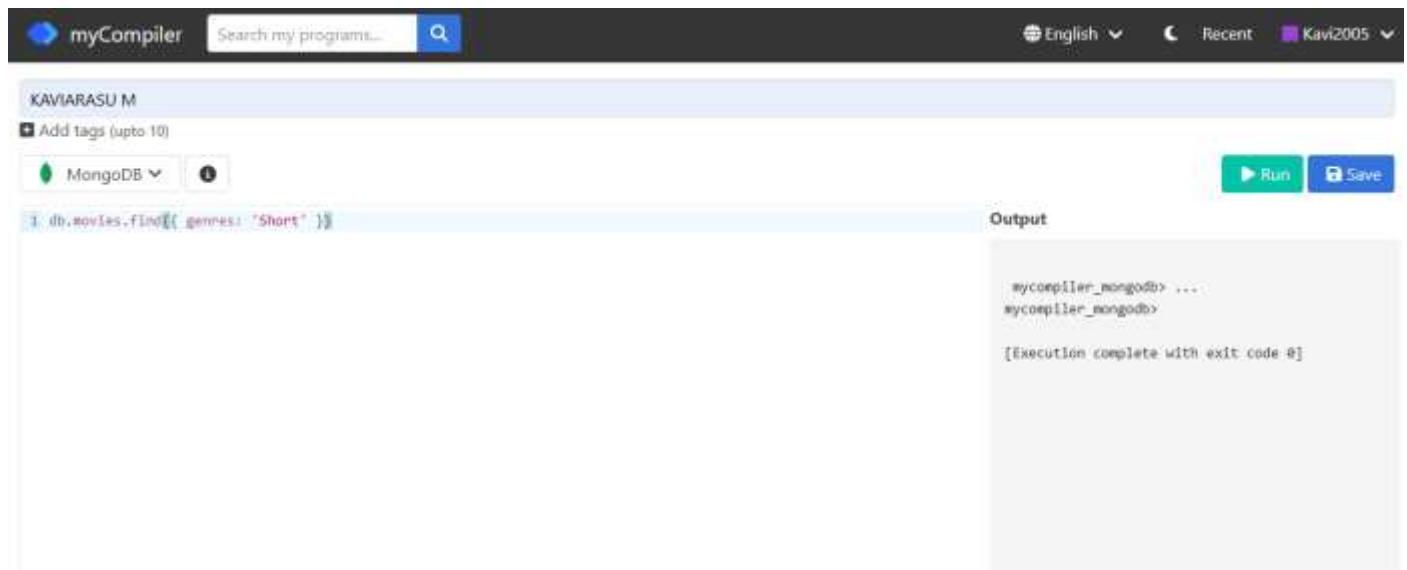
```
mycompiler_mongodb> ...
mycompiler_mongodb>
[Execution complete with exit code 0]
```

**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 the myCompiler interface. At the top, there's a search bar for programs and a language dropdown set to English. Below the search bar, the user is working on a MongoDB session. The code entered is: db.movies.find({ genres: 'Short' }). The output window shows the command being run and the response: "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:**



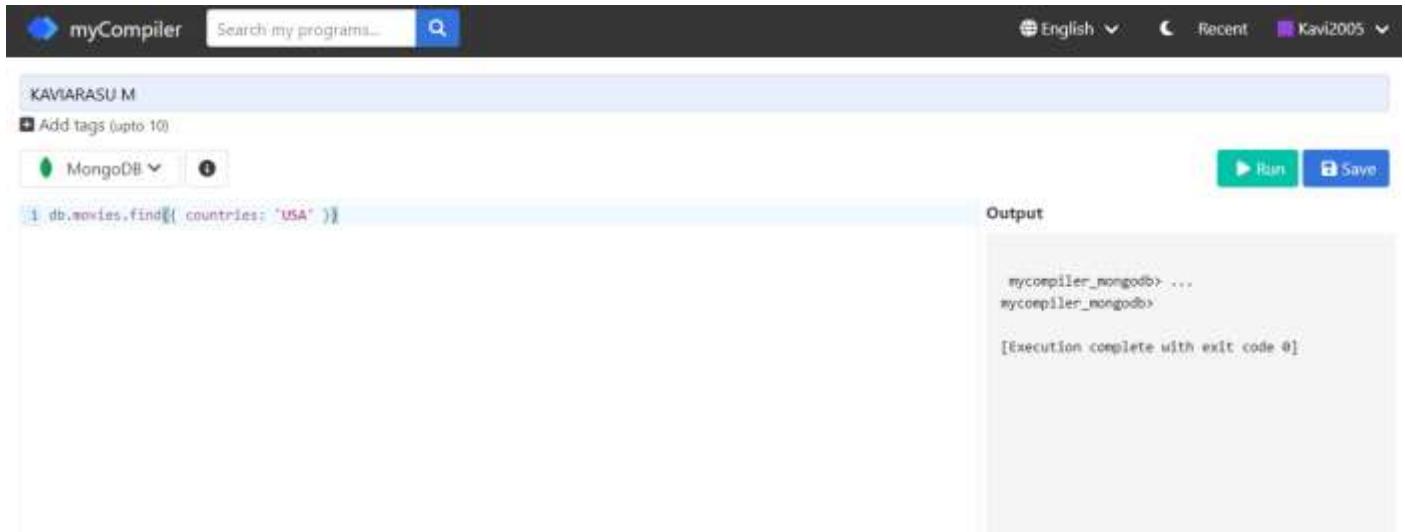
The screenshot shows the myCompiler interface. At the top, there's a search bar for programs and a language dropdown set to English. Below the search bar, the user is working on a MongoDB session. The code entered is: db.movies.find({ directors: 'William K.L. Dickson' }). The output window shows the command being run and the response: "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:**



The screenshot shows the myCompiler interface. At the top, there's a toolbar with icons for file operations, search, and language selection (English). Below the toolbar, the user profile 'KAVIARASU M' is displayed along with a 'Add tags (upto 10)' button. A dropdown menu for 'MongoDB' is open. On the left, a code editor window contains the MongoDB query: `i db.movies.find({ countries: 'USA' })`. On the right, an 'Output' window shows the results of the query execution. The output starts with the prompt 'mycompiler\_mongodb> ...' followed by 'mycompiler\_mongodb>'. Below the prompt, it says '[Execution complete with exit code 0]'. The entire interface has a clean, modern design with a light blue header and white background.

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

**QUERY:**

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

**OUTPUT:**



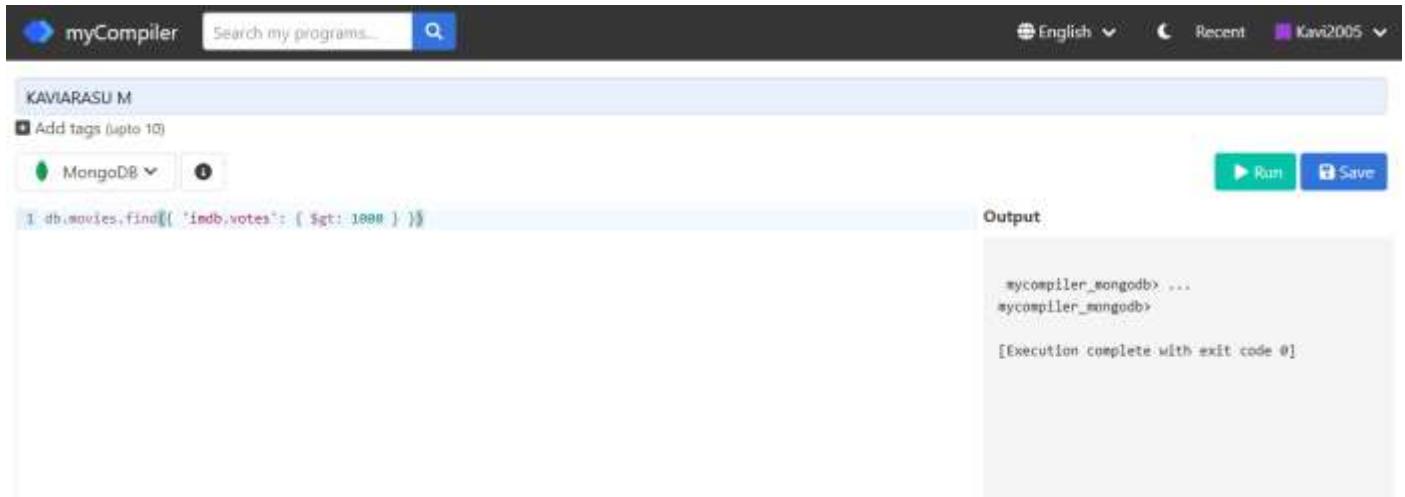
This screenshot is identical to the one above, showing the myCompiler interface. It features the same toolbar, user profile, and MongoDB dropdown. The code editor on the left contains the query `i db.movies.find({ rated: 'UNRATED' })`. The output window on the right shows the results of the query execution, starting with 'mycompiler\_mongodb> ...' and 'mycompiler\_mongodb>'. Below the prompt, it says '[Execution complete with exit code 0]'. The interface maintains its clean, modern aesthetic with a light blue header and white background.

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the myCompiler interface. In the top bar, there are icons for English, Recent, and Kavi2005. Below the bar, it says "KAVIARASU M" and "Add tags (upto 10)". On the left, there's a "MongoDB" dropdown and a "Run" button. The main area contains a code editor with the following command:

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

To the right of the code editor is an "Output" window. It displays the command and its execution results:

```
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 the myCompiler interface. In the top bar, there are icons for English, Recent, and Kavi2005. Below the bar, it says "KAVIARASU M" and "Add tags (upto 10)". On the left, there's a "MongoDB" dropdown and a "Run" button. The main area contains a code editor with the following command:

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

To the right of the code editor is an "Output" window. It displays the command and its execution results:

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

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the myCompiler interface. In the top bar, there are icons for English, Recent, and Kavi2005. Below the bar, it says "KAVIARASU M" and "Add tags (upto 10)". On the left, there's a "MongoDB" dropdown and a "Run" button. The main area contains a code editor with the following MongoDB query:

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

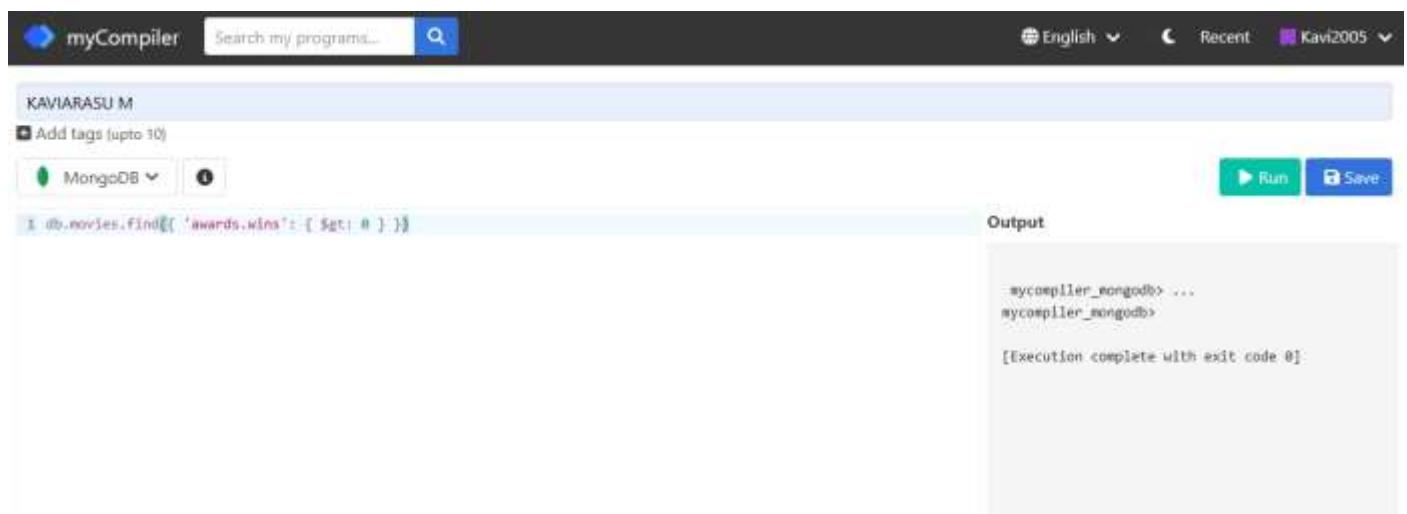
To the right of the code editor is an "Output" window. It displays the command prompt "mycompiler\_mongodb> ...", the response "mycompiler\_mongodb>", and the message "[Execution complete with exit code 0]".

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the myCompiler interface. In the top bar, there are icons for English, Recent, and Kavi2005. Below the bar, it says "KAVIARASU M" and "Add tags (upto 10)". On the left, there's a "MongoDB" dropdown and a "Run" button. The main area contains a code editor with the following MongoDB query:

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

To the right of the code editor is an "Output" window. It displays the command prompt "mycompiler\_mongodb> ...", the response "mycompiler\_mongodb>", and the message "[Execution complete with exit code 0]".

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the myCompiler interface with the MongoDB tab selected. In the code editor, the following command is entered:

```
1 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 } )
```

The output window shows the command being run and the response:

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

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the myCompiler interface with the MongoDB tab selected. In the code editor, the following command is entered:

```
1 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 } )
```

The output window shows the command being run and the response:

```
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:**



The screenshot shows the myCompiler MongoDB interface. The top bar includes the myCompiler logo, a search bar for programs, language selection (English), recent projects (Recent), and user profile (Kavi2005). The main area has tabs for KAVIARASU M and Add tags (upto 10). A MongoDB tab is selected, showing the query: db.movies.find( { released: ISODate("1893-05-09T00:00:00.000Z") }, { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 } ). The output panel shows the command being run and the response: 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 the myCompiler MongoDB interface. The top bar includes the myCompiler logo, a search bar for programs, language selection (English), recent projects (Recent), and user profile (Kavi2005). The main area has tabs for KAVIARASU M and Add tags (upto 10). A MongoDB tab is selected, showing the query: db.movies.find( { title: /scene/i }, { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 } ). The output panel shows the command being run and the response: mycompiler\_mongodb> ... mycompiler\_mongodb>. [Execution complete with exit code 0].

**RESULT:**