

# Creating and Managing Tables

EX\_NO:1

DATE:

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

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

QUERY:

```
CREATE TABLE DEPT (ID NUMBER(7), NAME VARCHAR(25));
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is active. The main area is titled 'SQL Commands'. A single line of SQL code is entered: 'create table DEPT(ID int,NAME varchar(25));'. The 'Run' button at the bottom right is highlighted in green. At the bottom of the screen, a message 'Table created.' is displayed, along with the user information '220701018@rajalakshmi.edu.in' and 'ak18'. The footer indicates 'Copyright © 1999, 2025, Oracle and/or its affiliates' and 'Oracle APEX 25.2.4'.

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), LNAME VARCHAR(25), FNAME VARCHAR(25), DEPT_ID NUMBER(7));
```

## OUTPUT:

A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, it shows the user's name 'AJAY KRISHNAN' and schema 'WKSP\_AK18'. The main area is titled 'SQL Commands' with a dropdown menu showing 'Language: SQL'. Below this is a text input field containing the SQL command: 'create table EMP(ID int, LAST\_NAME varchar(25), FIRST\_NAME varchar(25), DEPT\_ID int);'. At the bottom of the command input field are buttons for 'Clear Command' and 'Find Tables'. To the right of the command input are buttons for 'Save' and 'Run'. The results section below shows the output: 'Table created.'. The status bar at the bottom indicates the user's email '220701018@rajalakshmi.edu.in', the schema 'ak18', and the session language 'en'. It also shows copyright information 'Copyright © 1999, 2025, Oracle and/or its affiliates.' and the version 'Oracle APEX 25.2.4'.

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 (LNAME VARCHAR(50));**

## OUTPUT:

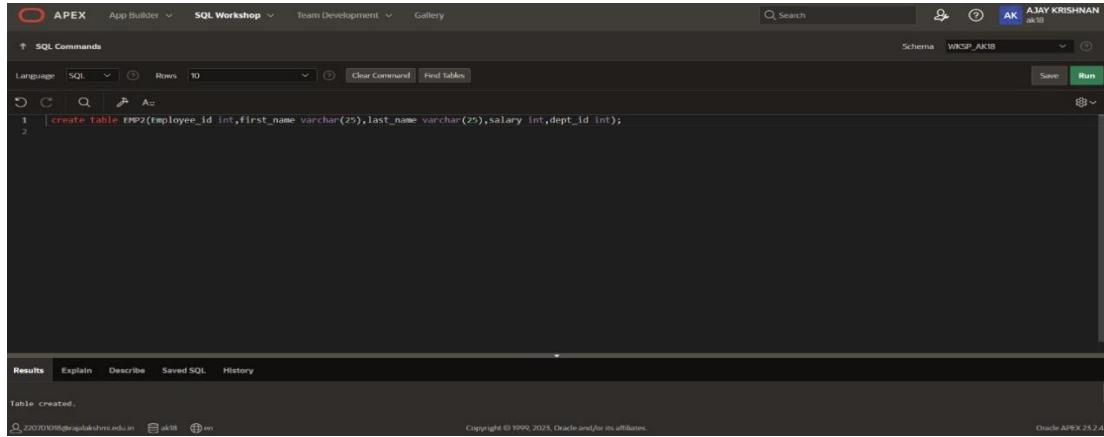
A screenshot of the Oracle APEX SQL Workshop interface, similar to the previous one but with a different command. The top navigation bar and user information are identical. The main area is titled 'SQL Commands' with a dropdown menu showing 'Language: SQL'. Below this is a text input field containing the SQL command: 'alter table EMP modify(LAST\_NAME varchar(50));'. At the bottom of the command input field are buttons for 'Clear Command' and 'Find Tables'. To the right of the command input are buttons for 'Save' and 'Run'. The results section below shows the output: 'Table altered.'. The status bar at the bottom indicates the user's email '220701018@rajalakshmi.edu.in', the schema 'ak18', and the session language 'en'. It also shows copyright information 'Copyright © 1999, 2025, Oracle and/or its affiliates.' and the version 'Oracle APEX 25.2.4'.

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 EMPLOYEES2 (ID NUMBER(7), FIRST\_NAME VARCHAR(25), LAST\_NAME VARCHAR(25), SALARY NUMBER(7), DEPT\_ID NUMBER(7));**

## OUTPUT:



A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, it shows the user 'AJAY KRISHNAN' and the schema 'WKSP\_AK18'. The main area is titled 'SQL Commands' with a sub-section 'Create Table'. The code entered is:

```
1 | create table EMP2(employee_id int,first_name varchar(25),last_name varchar(25),salary int,dept_id int);
```

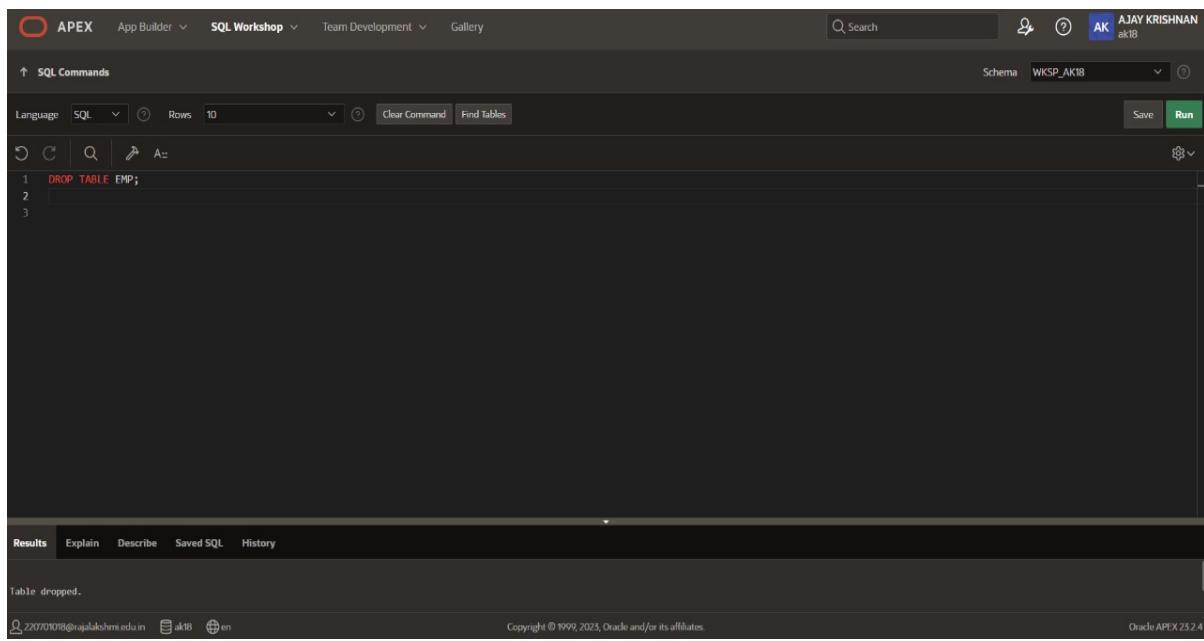
The results section shows the message 'Table created.' and the timestamp '22/07/2023 09:45:21'. The bottom right corner indicates 'Oracle APEX 25.2.4'.

5. Drop the EMP table.

## QUERY:

```
DROP TABLE EMP;
```

## OUTPUT:



A screenshot of the Oracle APEX SQL Workshop interface, identical to the previous one but with a different command. The top navigation bar and schema information are the same. The main area shows the command:

```
1 | DROP TABLE EMP;
```

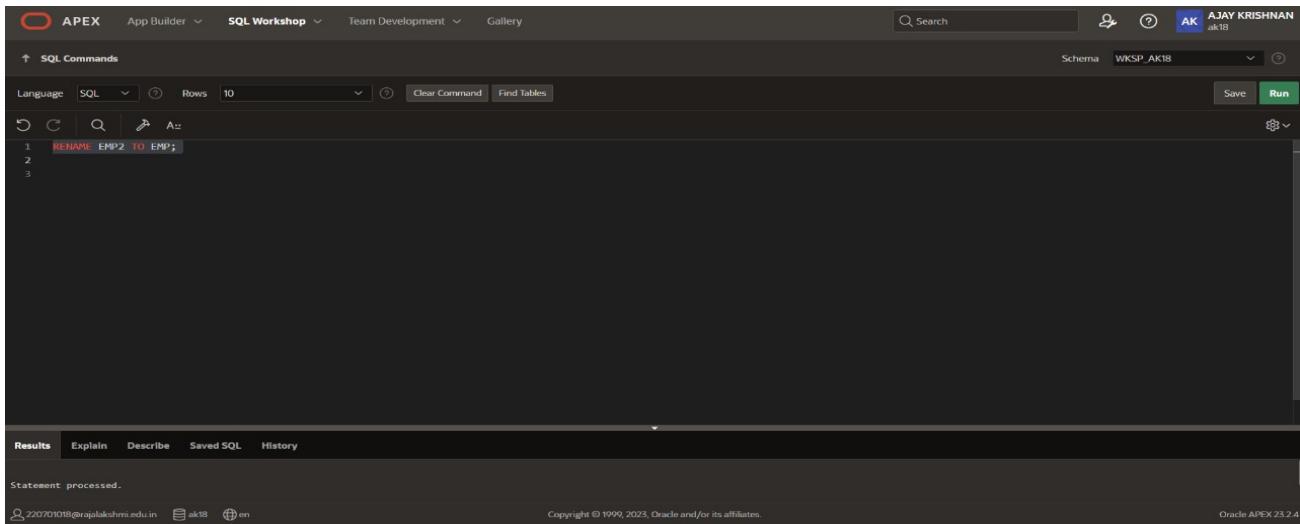
The results section shows the message 'Table dropped.' and the timestamp '22/07/2023 09:45:21'. The bottom right corner indicates 'Oracle APEX 25.2.4'.

6. Rename the EMPLOYEES2 table as EMP.

## QUERY:

```
alter table employee2 rename to emp;
```

## OUTPUT:



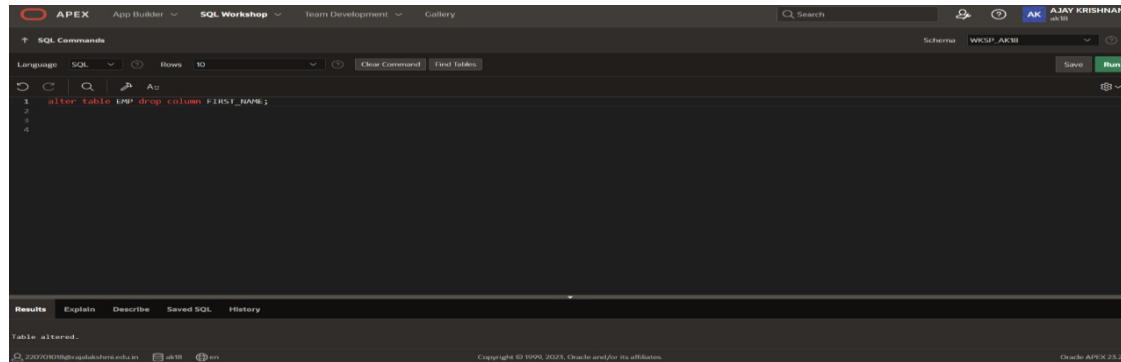
The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. The command entered is 'RENAME EMP2 TO EMP;'. Below the command, the status message 'Statement processed.' is displayed. The bottom right corner of the interface shows 'Oracle APEX 23.2.4'.

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

## QUERY:

**ALTER TABLE EMP DROP COLUMN FIRST\_NAME;**

## OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. The command entered is 'alter table EMP drop column FIRST\_NAME;'. Below the command, the status message 'Table altered.' is displayed. The bottom right corner of the interface shows 'Oracle APEX 23.2.4'.

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

**RESULT:**

## 2. MANIPULATING DATA

EX\_NO:2

DATE:

- 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 MY_EMPLOYEE (ID NUMBER(4) NOT NULL, LAST_NAME VARCHAR(25), FIRST_NAME  
VARCHAR(25), USERID VARCHAR(25), SALARY NUMBER(9,2));
```

OUTPUT:

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

```
1 create table MY_EMP(ID int,FIRST_NAME varchar(25),LAST_NAME varchar(25),SALARY int,USERID varchar(25));
```

The 'Results' tab shows the output: "Table created." Below the interface, the URL is 220701018@rajalakshmi.edu.in, the schema is akb8, and the session ID is 1n.

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

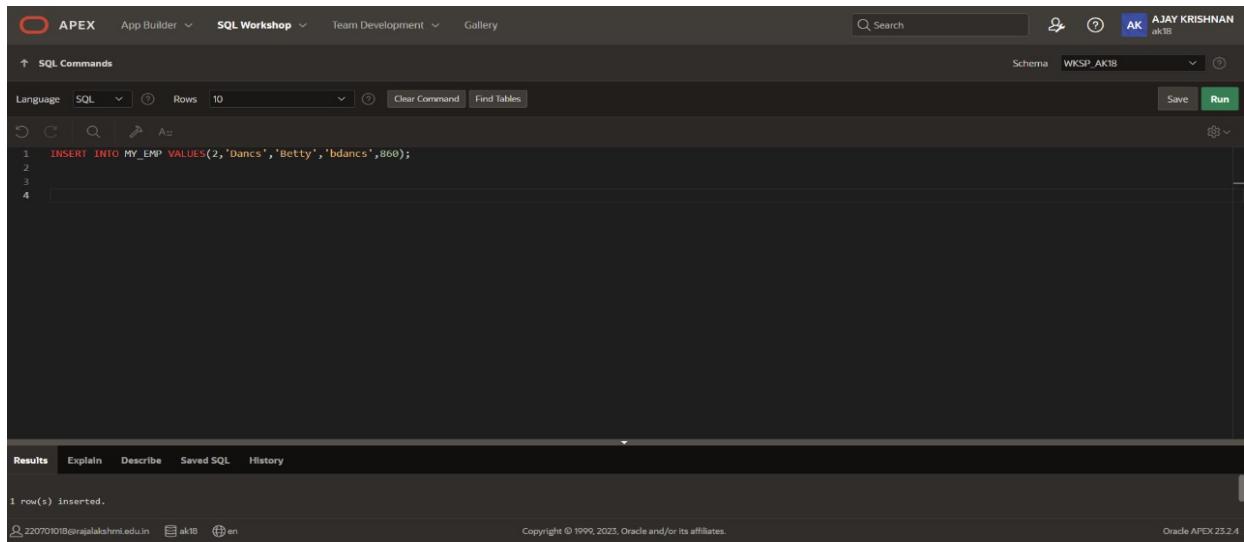
ID	Last_name	First_name	Userid	salary
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	Cnewman	750
5	Ropebur	Audrey	aropebur	1550

QUERY:

```
INSERT INTO MY_EMPLOYEE VALUES(1, 'PATEL', 'RALPH', 'RPATEL', 895);
```

```
INSERT INTO MY_EMPLOYEE VALUES(2, 'DANCS', 'BETTY', 'BDANCS', 860);
```

## OUTPUT:



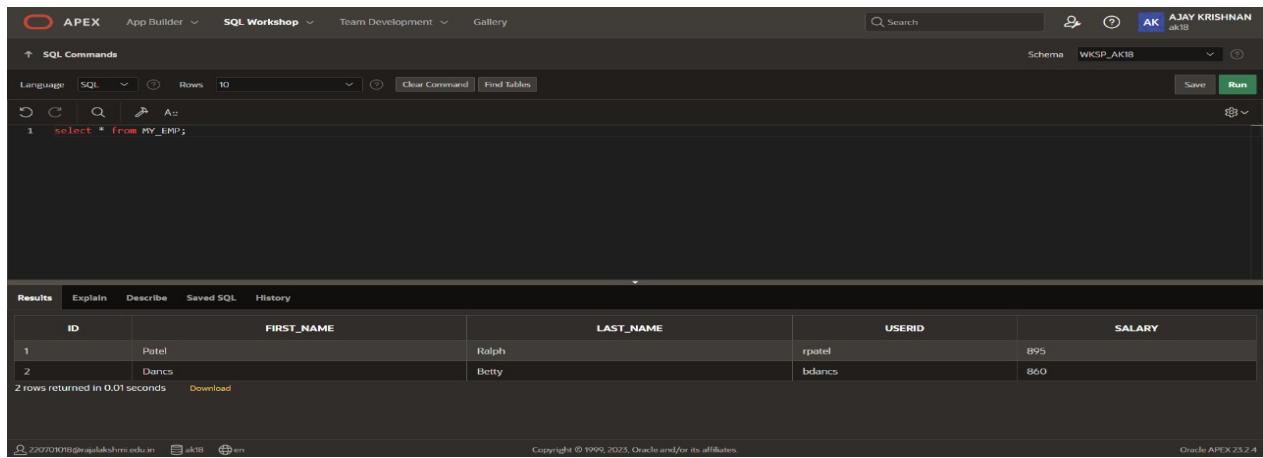
The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is active. The main area is titled 'SQL Commands'. A single line of SQL code is entered: 'INSERT INTO MY\_EMP VALUES(2,'Dancs','Betty','bdancs',860);'. The 'Run' button at the bottom right is highlighted in green.

3. Display the table with values.

## QUERY:

```
SELECT * FROM MY_EMPLOYEE;
```

## OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is active. The main area is titled 'SQL Commands'. A single line of SQL code is entered: 'select \* from MY\_EMP;'. The 'Run' button at the bottom right is highlighted in green.

**Results**

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

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

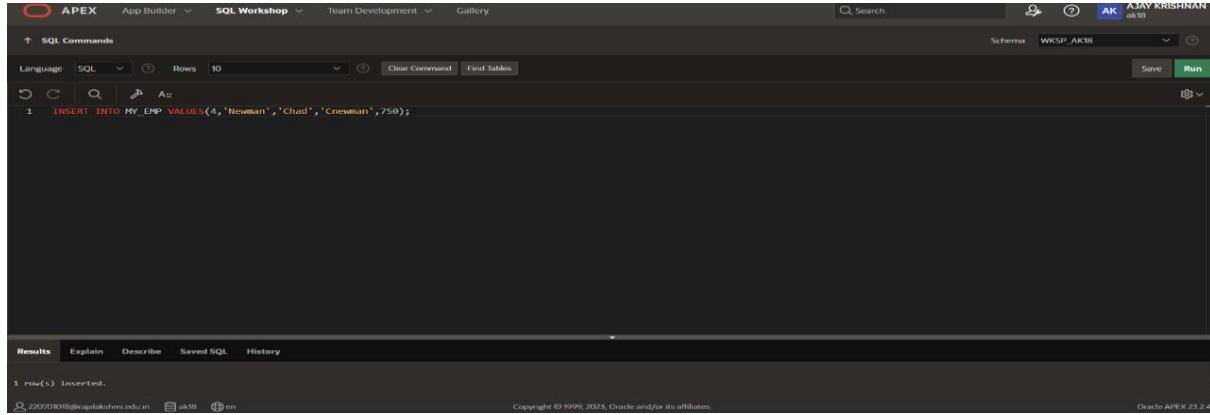
4. Populate the next two rows of data from the sample data. Concatenate the first letter of the first name with the first seven characters of the last name to produce Userid.

## QUERY:

```
INSERT INTO MY_EMPLOYEE VALUES(3, 'BIRI', 'BEN', 'BBIRI', 1100);
```

```
INSERT INTO MY_EMPLOYEE VALUES(4, 'NEWMAN', 'CHAD', 'CNEWMAN', 750);
```

## OUTPUT:



A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a user profile for 'AJAY KRISHNAN' and a schema dropdown set to 'WNSP\_AK18'. The main area shows a SQL command line with the following code:

```
1 INSERT INTO MY_EMP VALUES(4,'Newman','Chad','Crewman',750);
```

The results section below shows the output:

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

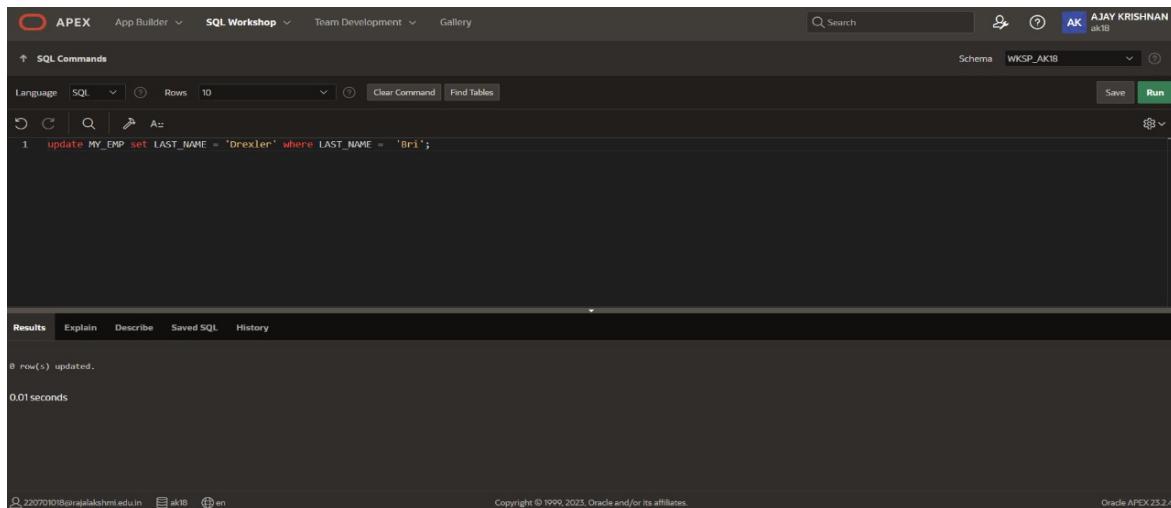
At the bottom, it says 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

## QUERY:

**UPDATE MY\_EMPLOYEE SET LAST\_NAME='DREXLER' WHERE ID=3;**

## OUTPUT:



A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a user profile for 'AJAY KRISHNAN' and a schema dropdown set to 'WNSP\_AK18'. The main area shows a SQL command line with the following code:

```
1 update MY_EMP set LAST_NAME = 'Drexler' where LAST_NAME = 'Bri';
```

The results section below shows the output:

```
0 row(s) updated.
```

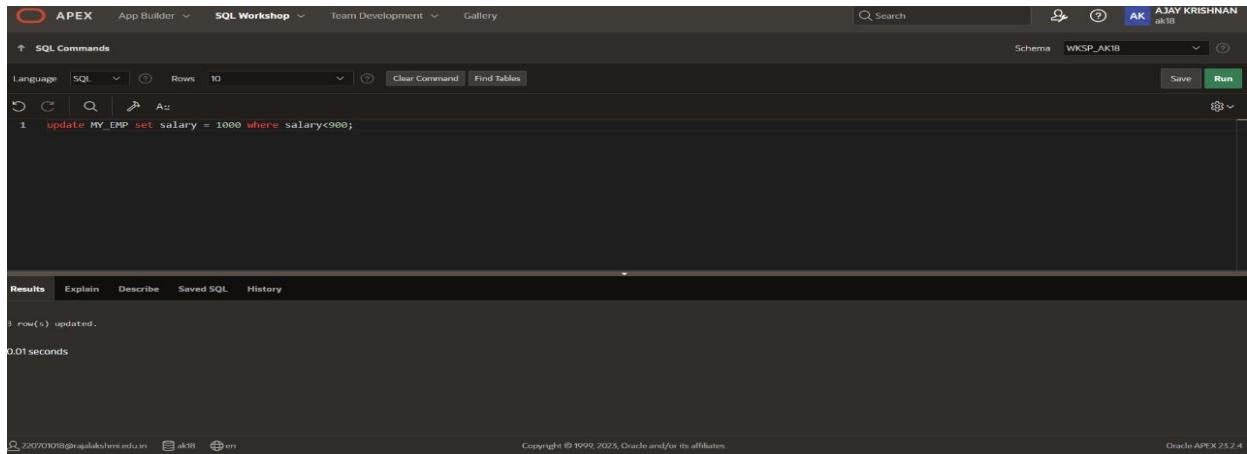
At the bottom, it says 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

## QUERY:

**UPDATE MY\_EMPLOYEE SET SALARY=1000 WHERE SALARY<900;**

## OUTPUT:



A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, it shows 'AJAY KRISHNAN' and 'Schema WKSP\_AK18'. The main area is titled 'SQL Commands' with a sub-section '↑ SQL Commands'. It has dropdowns for 'Language' (SQL) and 'Rows' (10), and buttons for 'Clear Command' and 'Find Tables'. Below this is a code editor with the following SQL command:

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

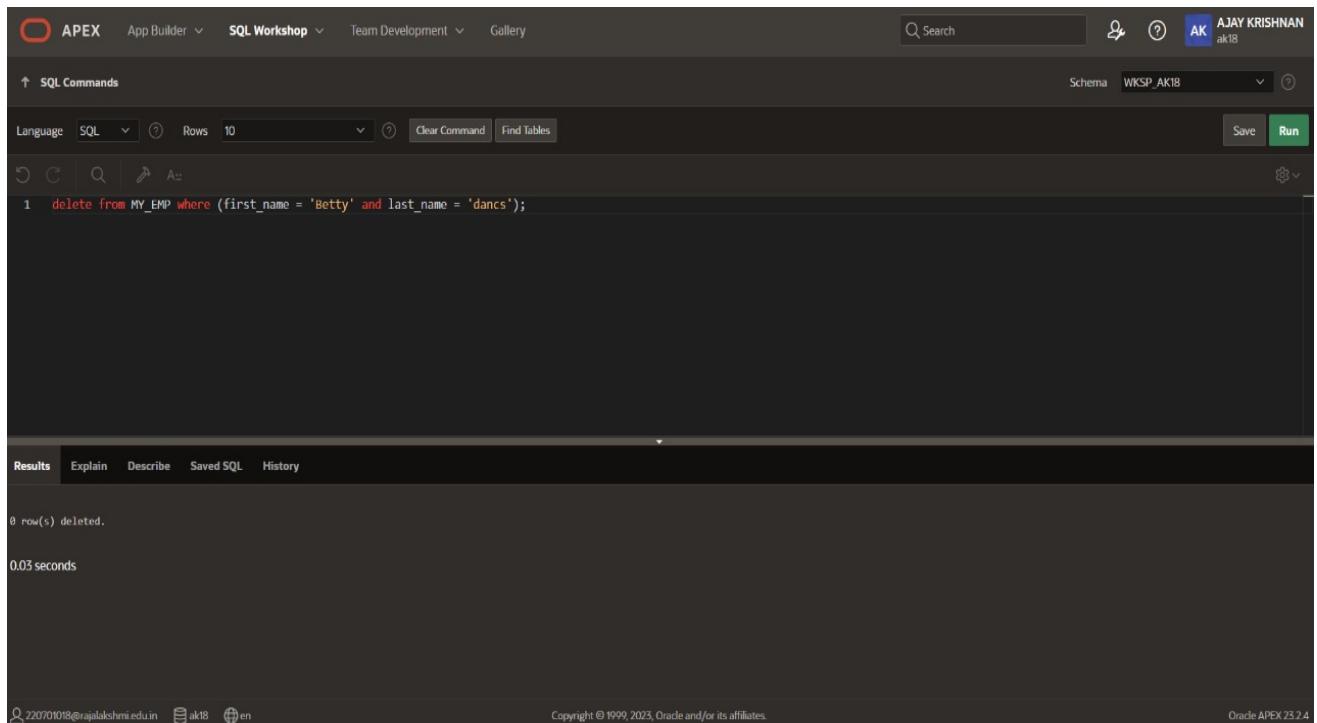
The results section shows '0 row(s) updated.' and '0.01 seconds'. At the bottom, it displays the user's email (220701018@rajalakshmi.edu.in), session ID (ak18), and the Oracle APEX version (23.2.4).

8. Delete Betty dancs from MY\_EMPLOYEE table.

## QUERY:

**DELETE FROM MY\_EMPLOYEE WHERE LAST\_NAME='DANCS';**

## OUTPUT:



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

```
1 delete from MY_EMP where (first_name = 'Betty' and last_name = 'dancs');
```

The results section shows '0 row(s) deleted.' and '0.03 seconds'. The bottom information remains the same, including the user's email, session ID, and Oracle APEX version.

9. Empty the fourth row of the emp table.

## QUERY:

**DELETE FROM MY\_EMPLOYEE WHERE ID=4;**

## OUTPUT:

A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar includes links for App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The main area shows a SQL command being run:

```
1 delete from MY_IMP WHERE (ID=4);
```

The results section shows the output of the query:

```
1 row(s) deleted.  
0.01 seconds
```

At the bottom, it displays the URL <https://220701038@ajaykrishnan.in>, session information for user ak18, and the copyright notice "Copyright © 1999, 2022, Oracle and/or its affiliates".

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

**RESULT:**

# INCLUDING CONSTRAINTS

EX\_NO: 3

DATE:

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

QUERY:

```
ALTER TABLE EMP ADD CONSTRAINT MY_EMP_ID_PK PRIMARY KEY(ID);
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The right side shows a user profile for 'AJAY KRISHNAN' with the schema 'WKSP\_AK18'. The main area is titled 'SQL Commands' with a 'Language' dropdown set to 'SQL'. Below it, there are buttons for 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. On the far right are 'Save' and 'Run' buttons. The code input field contains the command: '1 alter table myemployee add constraint my\_emp1\_id\_pk primary key(emp\_id);'. The results section below shows the output: 'Table altered.' and '0.10 seconds'. At the bottom, it displays the user's email '220701018@rajalakshmi.edu.in', the schema 'ak18', and the language 'en'. The footer notes 'Copyright © 1999, 2025, Oracle and/or its affiliates.' and 'Oracle APEX 25.2.4'.

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

QUERY:

```
CREATE TABLE DEPT1(DEPT_ID NUMBER(7) NOT NULL, NAME VARCHAR(25);
```

## OUTPUT:

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

```
1 CREATE TABLE DEPT1(DEPT_ID NUMBER(?) NOT NULL, NAME VARCHAR(25));
```

Below the command line, the 'Results' tab is active, showing the output of the query:

```
Table created.
```

Execution time is listed as 0.04 seconds. The bottom right corner of the interface displays '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(DEPT_ID) REFERENCES DEPT(ID);
```

## OUTPUT:

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

```
1 ALTER TABLE myemployee ADD CONSTRAINT MY_EMP_DEPT_ID_FK FOREIGN KEY(dept_id) REFERENCES myemployee(emp_id);
```

Below the command line, the 'Results' tab is active, showing the output of the query:

```
Table altered.
```

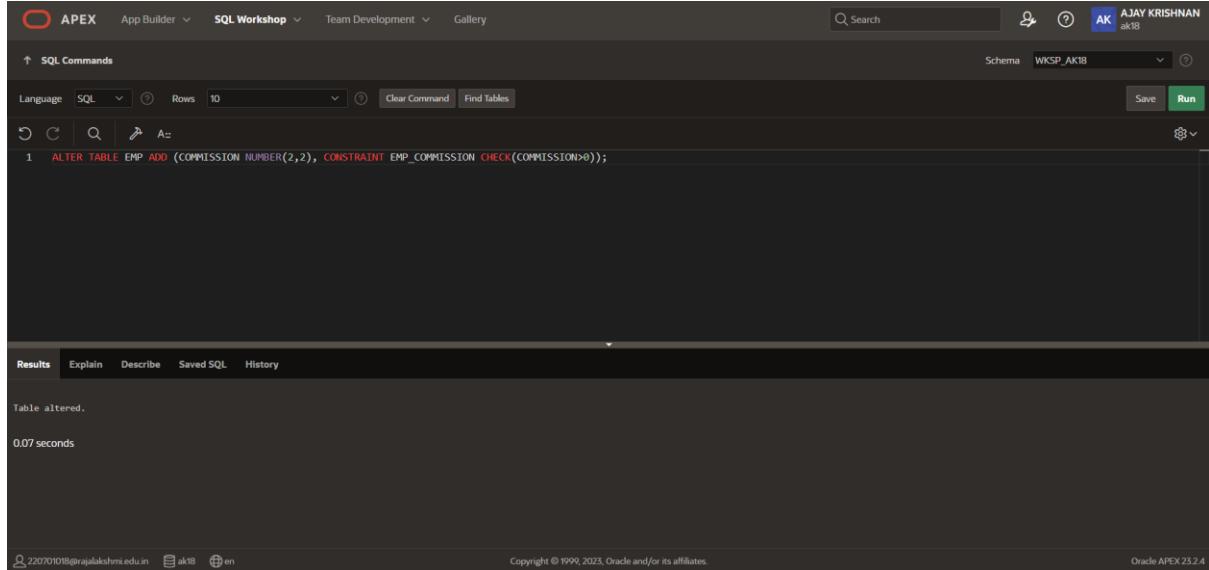
Execution time is listed as 0.08 seconds. The bottom right corner of the interface displays 'Oracle APEX 23.2.4'.

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_COMMISSION  
CHECK(COMMISSION>0));
```

## 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'. A single line of SQL code is entered: 'ALTER TABLE EMP ADD (COMMISSION NUMBER(2,2), CONSTRAINT EMP\_COMMISSION CHECK(COMMISSION>0));'. Below the code, the output shows 'Table altered.' and a execution time of '0.07 seconds'. The bottom right corner of the interface displays '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:

## True OR False

1. The following statement executes successfully.

## Identify the Errors

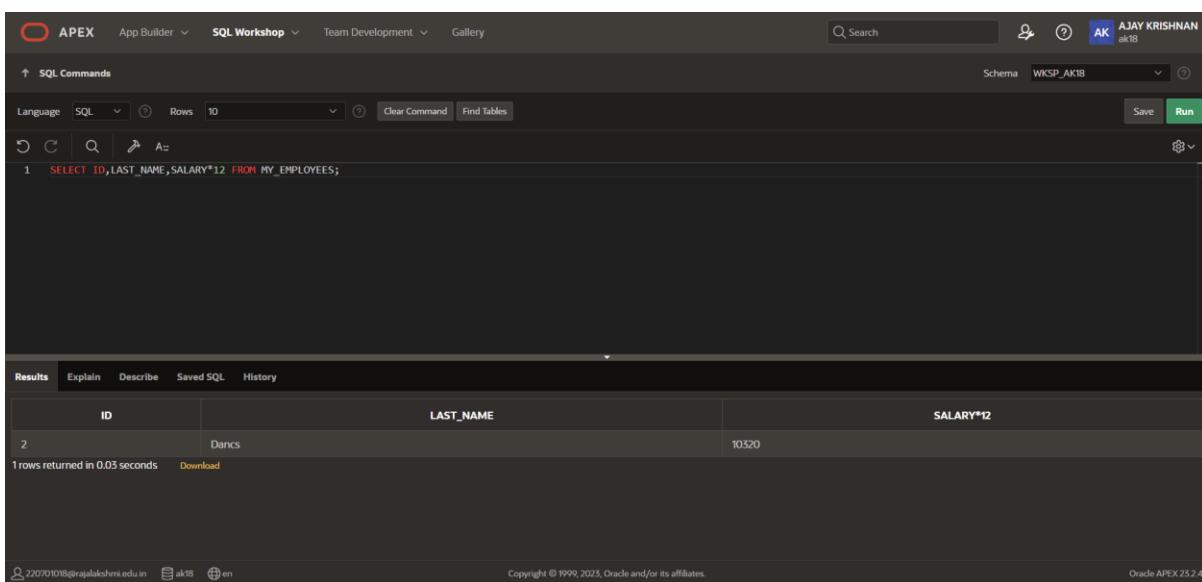
```
SELECT employee_id, last_name  
sal*12 ANNUAL SALARY  
FROM employees;
```

## Queries

QUERY:

```
SELECT ID, LAST_NAME, SALARY*12 FROM MY_EMPLOYEE;
```

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is signed in as 'AJAY KRISHNAN' (ak18). The SQL Commands tab is active, showing the following command:

```
1 SELECT ID, LAST_NAME, SALARY*12 FROM MY_EMPLOYEE;
```

The results section displays the output of the query:

ID	LAST_NAME	SALARY*12
2	Dance	10520

Below the table, it says '1 rows returned in 0.03 seconds' and provides a 'Download' link. The bottom of the page includes copyright information for Oracle and the APEX version.

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

QUERY:

```
SELECT * FROM MY_EMPLOYEE;
```

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. On the right, it shows the user 'AJAY KRISHNAN' and the schema 'WKSP\_AK18'. The main area has tabs for 'SQL Commands' and 'Results'. Under 'Results', there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The results table has columns: NAME, ID, LAST\_NAME, FIRST\_NAME, USERID, and SALARY. One row is displayed: arish, 2, Dancs, betty, bdance, 860. Below the table, it says '1 rows returned in 0.04 seconds' and has a 'Download' link. At the bottom, it shows the URL '220701018@rajalakshmi.edu.in', session 'ak18', and page 'en'. Copyright information and 'Oracle APEX 25.2.4' are also at the bottom.

NAME	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
arish	2	Dancs	betty	bdance	860

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 ID, LAST_NAME, JOB_CODE, HIRE_DATE FROM MY_EMPLOYEE;
```

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. On the right, it shows the user 'AJAY KRISHNAN' and the schema 'WKSP\_AK18'. The main area has tabs for 'SQL Commands' and 'Results'. Under 'Results', there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The results table has columns: ID, LAST\_NAME, JOB\_CODE, and HIRE\_DATE. One row is displayed: 2, Dancs, 28, 2/3/2024. Below the table, it says '1 rows returned in 0.01 seconds' and has a 'Download' link. At the bottom, it shows the URL '220701018@rajalakshmi.edu.in', session 'ak18', and page 'en'. Copyright information and 'Oracle APEX 25.2.4' are also at the bottom.

ID	LAST_NAME	JOB_CODE	HIRE_DATE
2	Dancs	28	2/3/2024

4. Provide an alias STARTDATE for the hire date.

## QUERY:

```
SELECT HIRE_DATE AS "START_DATE" FROM MY_EMPLOYEE;
```

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is logged in as AJAY KRISHNAN (ak18). The schema selected is WKSP\_AK18. The SQL Commands tab is active, showing a single line of SQL code: `SELECT HIRE_DATE AS "START_DATE" FROM MY_EMPLOYEES;`. Below the code, the Results tab is selected, displaying the output: `START_DATE` with a single row value of `2/3/2024`. The status bar at the bottom indicates `1 rows returned in 0.00 seconds`.

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

## QUERY:

```
SELECT DISTINCT JOB_CODE FROM MY_EMPLOYEE;
```

## OUTPUT:

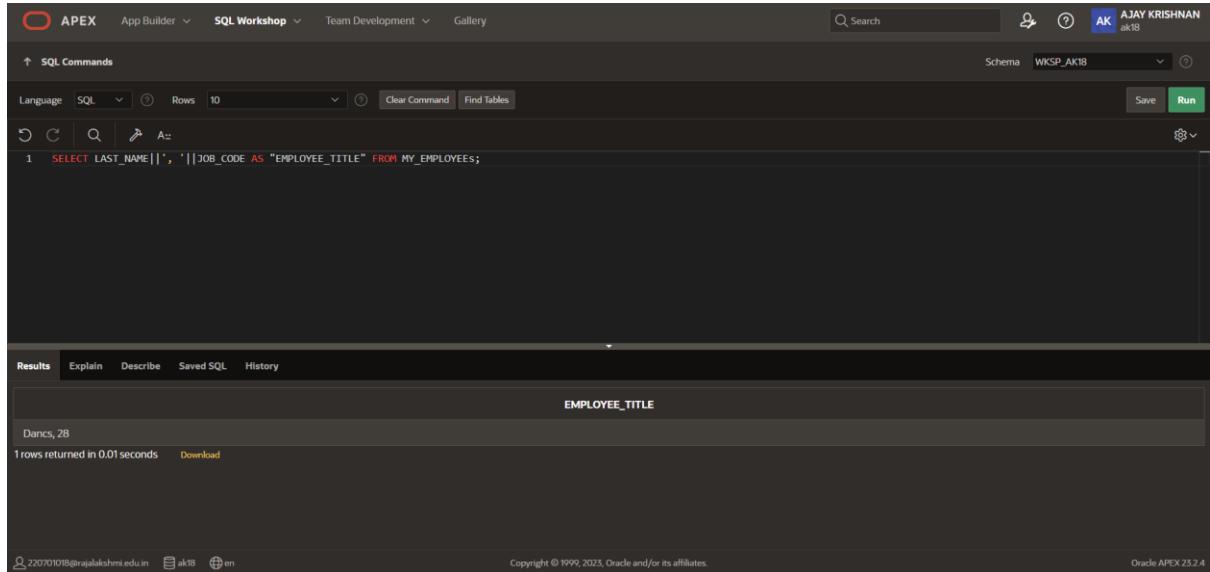
The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is logged in as AJAY KRISHNAN (ak18). The schema selected is WKSP\_AK18. The SQL Commands tab is active, showing the query: `SELECT DISTINCT JOB_CODE FROM MY_EMPLOYEE;`. Below the code, the Results tab is selected, displaying the output: `JOB_CODE` with a single row value of `28`. The status bar at the bottom indicates `1 rows returned in 0.01 seconds`.

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

## QUERY:

```
SELECT LAST_NAME||','||JOB_CODE AS "EMPLOYEE_TITLE" FROM MY_EMPLOYEE;
```

## OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. At the top, there are tabs for 'APEX', 'App Builder', 'SQL Workshop' (which is selected), 'Team Development', and 'Gallery'. On the right, it shows the user 'AJAY KRISHNAN' and the schema 'WKSP\_AK18'. The main area is titled 'SQL Commands' with a dropdown for 'Language' set to 'SQL'. Below that, there are fields for 'Rows' (set to 10) and buttons for 'Clear Command' and 'Find Tables'. The command entered is: 

```
1 SELECT LAST_NAME||', '||JOB_CODE AS "EMPLOYEE_TITLE" FROM MY_EMPLOYEES;
```

 The results section shows a single row: 

EMPLOYEE_TITLE
Dancs, 28

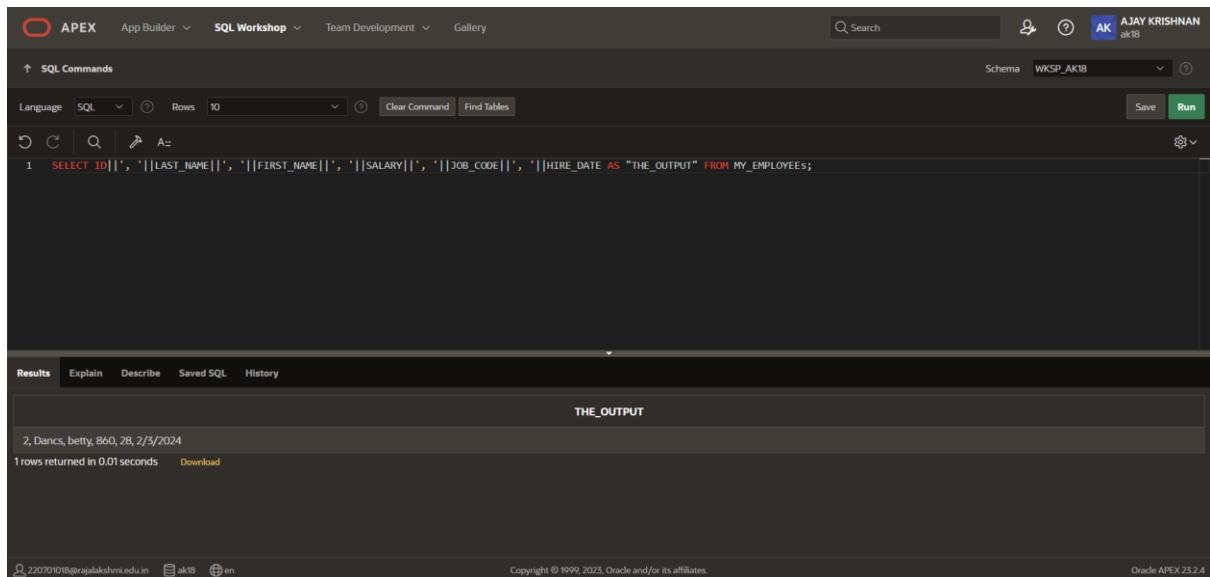
 with the note '1 rows returned in 0.01 seconds'. The bottom of the page includes copyright information and links for 'Download' and language selection.

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 ID||', '||LAST_NAME||', '||FIRST_NAME||', '||SALARY||', '||JOB_CODE||', '||HIRE_DATE  
AS "THE_OUTPUT" FROM MY_EMPLOYEE;
```

## OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The setup is identical to the previous screenshot, with 'SQL Workshop' selected. The command entered is: 

```
1 SELECT ID||', '||LAST_NAME||', '||FIRST_NAME||', '||SALARY||', '||JOB_CODE||', '||HIRE_DATE AS "THE_OUTPUT" FROM MY_EMPLOYEES;
```

 The results section shows a single row: 

THE_OUTPUT
2, Dancs, betty, 860, 28, 2/3/2024

 with the note '1 rows returned in 0.01 seconds'. The bottom of the page includes copyright information and links for 'Download' and language selection.

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

**RESULT:**

# SINGLE ROW FUNCTIONS

EX.NO:6

DATE: 12 – 3 - 24

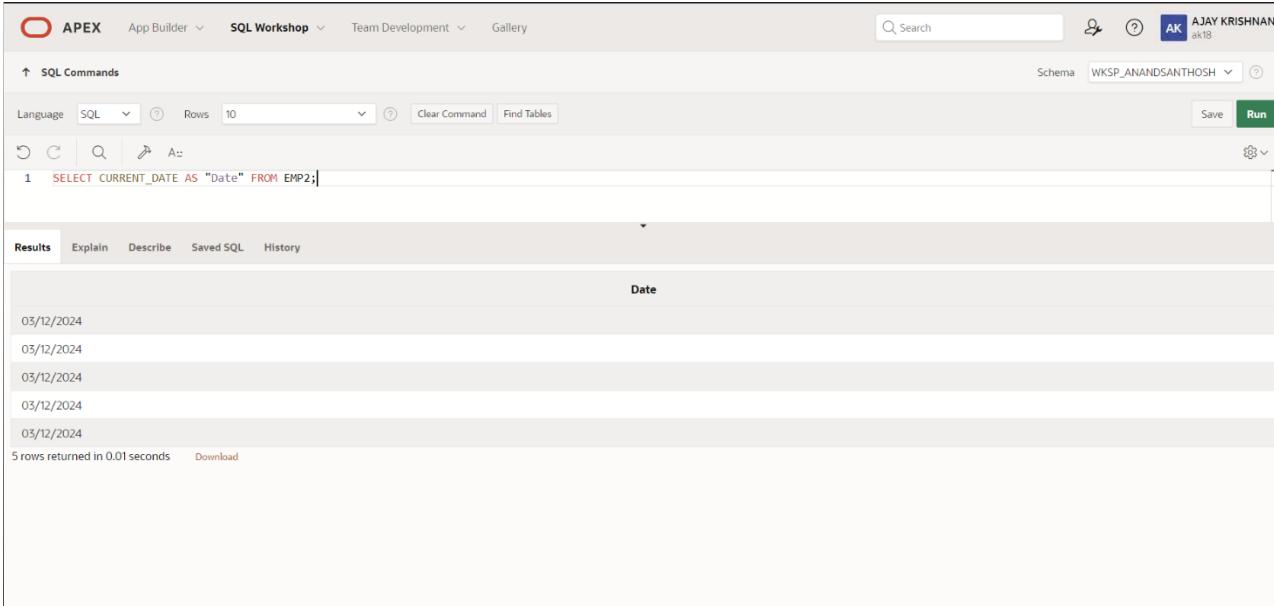
REG.NO: 220701018

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

**QUERY:**

```
SELECT CURRENT_DATE AS "Date" FROM EMP2;
```

**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 AJAY KRISHNAN and a schema dropdown set to WKSP\_ANANDSANTHOSH. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab shows the query: `1 SELECT CURRENT_DATE AS "Date" FROM EMP2;`. The Results tab displays the output: five rows of the current date (03/12/2024). The bottom status bar indicates "5 rows returned in 0.01 seconds".

Date
03/12/2024
03/12/2024
03/12/2024
03/12/2024
03/12/2024

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

### QUERY:

```
SELECT employee_number, last_name, salary, ROUND(salary * 0.155, 0) AS "New Salary"  
FROM EMP2;
```

### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. At the top, there are tabs for APEX, App Builder, SQL Workshop (which is selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'AJAY KRISHNAN', and a schema dropdown set to 'WKSP\_ANANDSANTHOSH'. Below the tabs, there are buttons for Language (SQL), Rows (set to 10), Clear Command, Find Tables, Save, and Run. The main area contains the SQL command:

```
1 SELECT employee_number, last_name, salary, ROUND(salary * 0.155, 0) AS "New Salary" FROM EMP2;
```

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

EMPLOYEE_NUMBER	LAST_NAME	SALARY	New Salary
1004	Jones	90000	13950
1002	Johnson	6000	930
1005	Williams	66000	10230
1005	Brown	9000	1395
1001	Smith	50000	7750

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

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

### QUERY:

```
SELECT employee_number, last_name, salary, ROUND(salary * 0.155, 0) AS "New Salary",  
salary - ROUND(salary * 0.155, 0) AS "Increase" FROM EMP2;
```

### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a user profile for 'AJAY KRISHNAN' and a search bar. The main workspace is titled 'SQL Commands'. It features a toolbar with icons for refresh, undo, redo, search, and run. Below the toolbar, a command line window contains the following SQL query:

```
1 SELECT employee_number, last_name, salary, ROUND(salary * 0.155, 0) AS "New Salary", salary - ROUND(salary * 0.155, 0) AS "Increase" FROM EMP2;
```

Below the command line, the results tab is selected. The results table has columns: EMPLOYEE\_NUMBER, LAST\_NAME, SALARY, New Salary, and Increase. The data is as follows:

EMPLOYEE_NUMBER	LAST_NAME	SALARY	New Salary	Increase
1004	Jones	90000	13950	76050
1002	Johnson	6000	930	5070
1003	Williams	66000	10230	55770
1005	Brown	9000	1395	7605
1001	Smith	50000	7750	42250

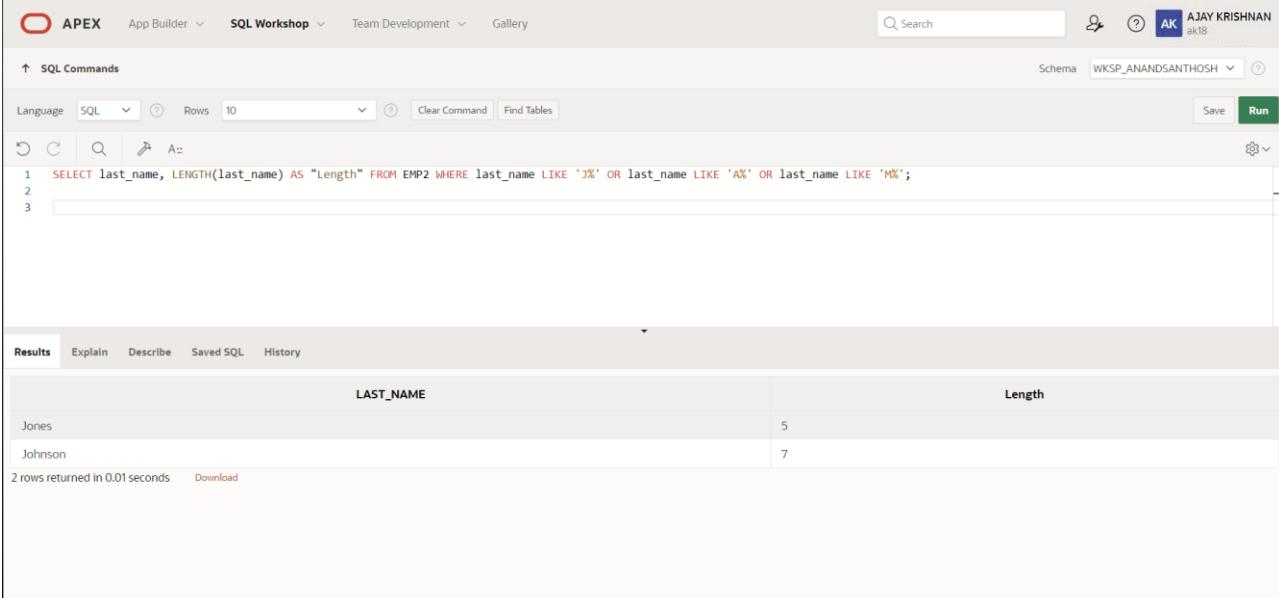
At the bottom left of the results pane, it says '5 rows returned in 0.01 seconds' and there is a 'Download' link.

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 last_name, LENGTH(last_name) AS "Length" FROM EMP2 WHERE last_name LIKE 'J%' OR last_name LIKE 'A%' OR last_name LIKE 'M%';
```

### OUTPUT:



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

```
1 SELECT last_name, LENGTH(last_name) AS "Length" FROM EMP2 WHERE last_name LIKE 'J%' OR last_name LIKE 'A%' OR last_name LIKE 'M%';
```

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

LAST_NAME	Length
Jones	5
Johnson	7

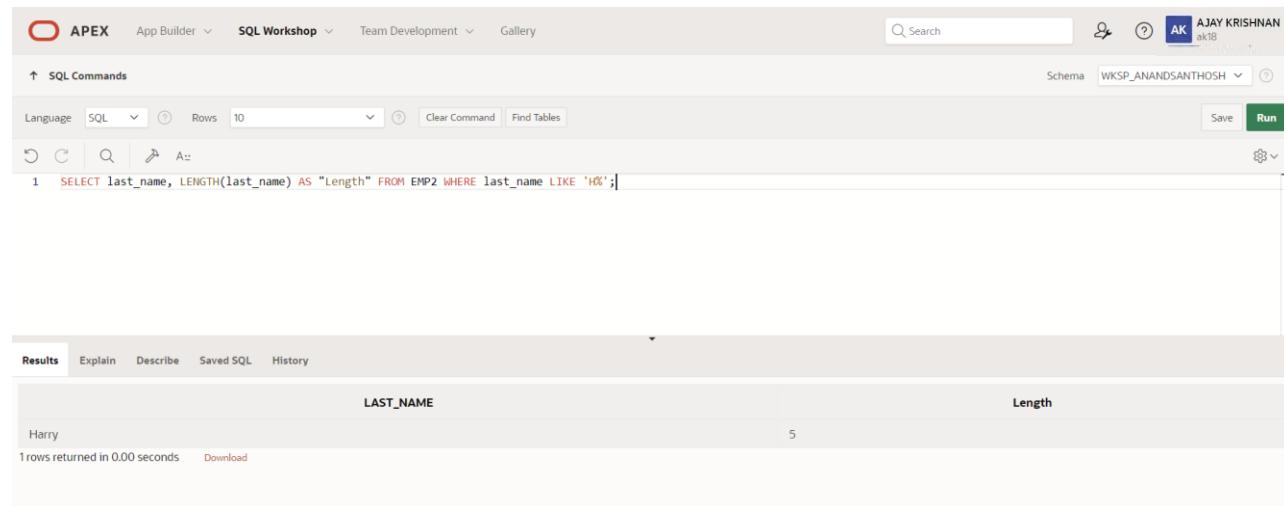
Below the table, it says "2 rows returned in 0.01 seconds".

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

### QUERY:

```
SELECT last_name, LENGTH(last_name) AS "Length" FROM EMP2 WHERE last_name LIKE 'H%';
```

### 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 profile of AJAY KRISHNAN (ak18). The main area is titled 'SQL Commands' with a schema dropdown set to WKSP\_ANANDSANTOSH. The SQL editor contains the following query:

```
1  SELECT last_name, LENGTH(last_name) AS "Length" FROM EMP2 WHERE last_name LIKE 'H%';
```

The results section shows a single row with the following data:

LAST_NAME	Length
Harry	5

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

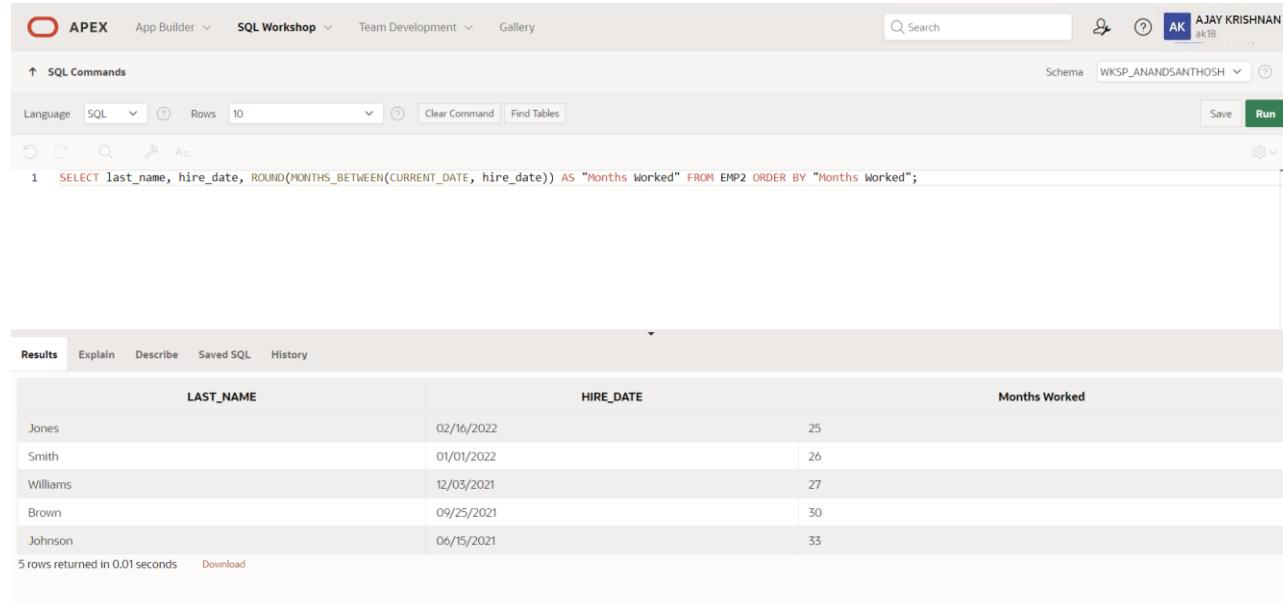
6. The HR department wants to find the length of employment for each employee. For each employee, display the last name and calculate the number of months between today and the date on which the employee was hired. Label the column MONTHS\_WORKED. Order your results by the number of months employed.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

**Note:** Your results will differ.

### QUERY:

```
SELECT last_name, hire_date, ROUND(MONTHS_BETWEEN(CURRENT_DATE, hire_date))  
AS "Months Worked" FROM EMP2 ORDER BY "Months Worked";
```

### 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 AJAY KRISHNAN (ak18) and a schema dropdown for WKSP\_ANANDSANTHOSH. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab shows the query:

```
1  SELECT last_name, hire_date, ROUND(MONTHS_BETWEEN(CURRENT_DATE, hire_date)) AS "Months Worked" FROM EMP2 ORDER BY "Months Worked";
```

The Results tab displays the output:

LAST_NAME	HIRE_DATE	Months Worked
Jones	02/16/2022	25
Smith	01/01/2022	26
Williams	12/03/2021	27
Brown	09/25/2021	30
Johnson	06/15/2021	33

Below the table, it says "5 rows returned in 0.01 seconds" and has a "Download" link.

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

### QUERY:

```
SELECT last_name, salary, ROUND(salary * 3, 0) AS "Dream Salaries" FROM EMP2;
```

### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. At the top, there are tabs for APEX, App Builder, SQL Workshop (which is selected), Team Development, and Gallery. On the right, there's a user profile for 'AJAY KRISHNAN' and a schema dropdown set to 'WKSP\_ANANDSANTHOSH'. The main area shows a SQL command line with the query: 'SELECT last\_name, salary, ROUND(salary \* 3, 0) AS "Dream Salaries" FROM EMP2;'. Below the command line is a results grid. The grid has three columns: 'LAST\_NAME', 'SALARY', and 'Dream Salaries'. The data returned is as follows:

LAST_NAME	SALARY	Dream Salaries
Jones	90000	270000
Johnson	6000	18000
Williams	66000	198000
Brown	9000	27000
Smith	50000	150000

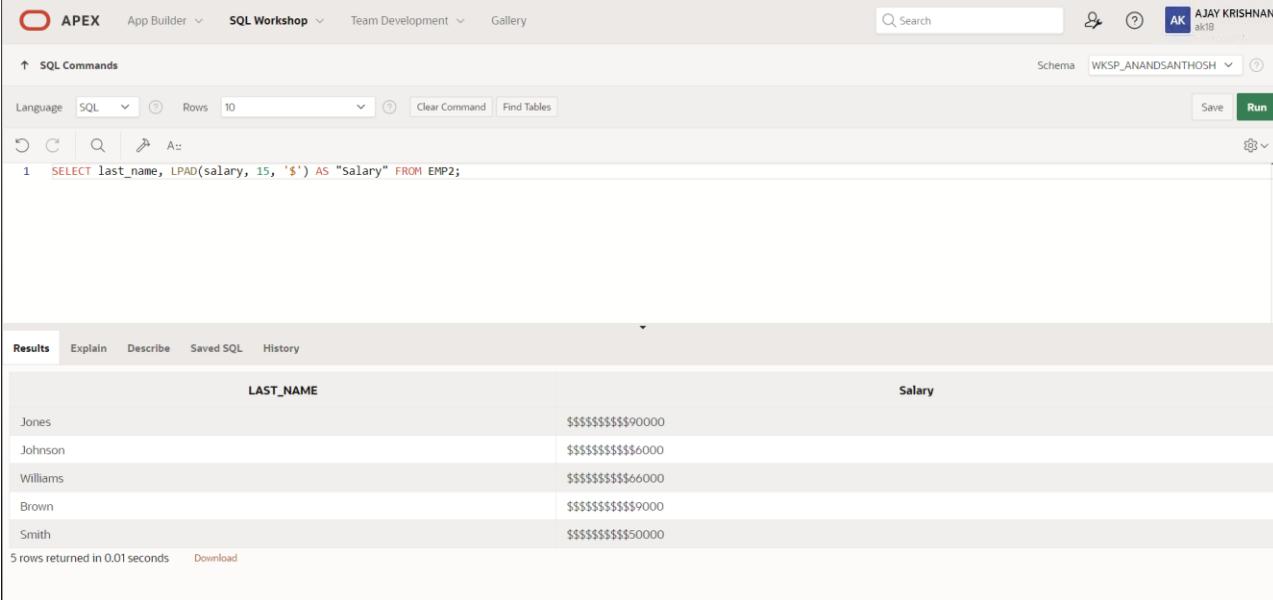
At the bottom left of the results grid, it says '5 rows returned in 0.01 seconds'. There are also 'Download' and 'Print' buttons at the bottom of the results grid.

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

### QUERY:

```
SELECT last_name, LPAD(salary, 15, '$') AS "Salary" FROM EMP2;
```

### 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 'AJAY KRISHNAN' and a schema dropdown set to 'WKSP\_ANANDSANTHOSH'. The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the query is listed:

```
1 SELECT last_name, LPAD(salary, 15, '$') AS "Salary" FROM EMP2;
```

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

LAST_NAME	Salary
Jones	\$\$\$\$\$\$\$\$\$\$90000
Johnson	\$\$\$\$\$\$\$\$\$\$6000
Williams	\$\$\$\$\$\$\$\$\$\$66000
Brown	\$\$\$\$\$\$\$\$\$\$90000
Smith	\$\$\$\$\$\$\$\$\$\$50000

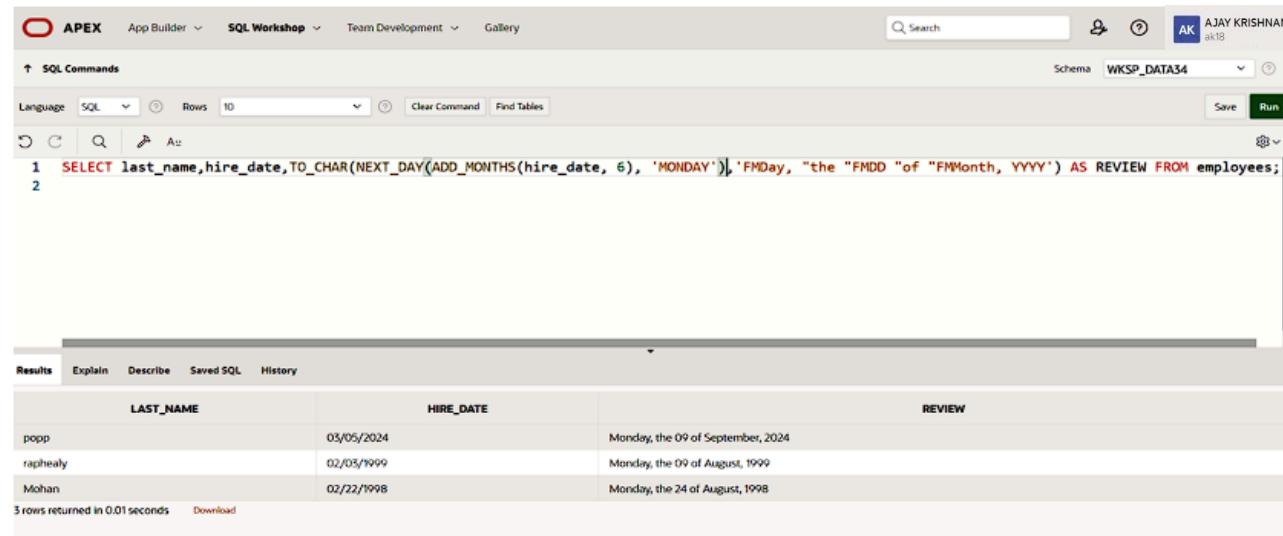
Below the table, it says '5 rows returned in 0.01 seconds' and there's a 'Download' link.

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

### QUERY:

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

### OUTPUT:



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

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

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

LAST_NAME	HIRE_DATE	REVIEW
popp	05/05/2024	Monday, the 09 of September, 2024
raphealy	02/05/1999	Monday, the 09 of August, 1999
Mohan	02/22/1998	Monday, the 24 of August, 1998

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

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

### QUERY:

```
SELECT last_name, TO_CHAR(hire_date, 'Day') AS "Day" FROM EMP2 ORDER BY "Day";
```

### 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 user profile for AJAY KRISHNAN (ak18) and a schema dropdown set to WKSP\_ANANDSANTOSH. The main area has a toolbar with icons for Undo, Redo, Find, Replace, and others. Below the toolbar, the SQL command is entered:

```
1  SELECT last_name, TO_CHAR(hire_date, 'Day') AS "Day" FROM EMP2 ORDER BY "Day";|
```

Under the "Results" tab, the output is displayed in a grid:

LAST_NAME	Day
Williams	Friday
Smith	Saturday
Brown	Saturday
Johnson	Tuesday
Jones	Wednesday

At the bottom left, it says "5 rows returned in 0.01 seconds".

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

## DISPLAYING DATA FROM MULTIPLE TABLES

EX.NO:7

DATE: 15 - 3 - 24

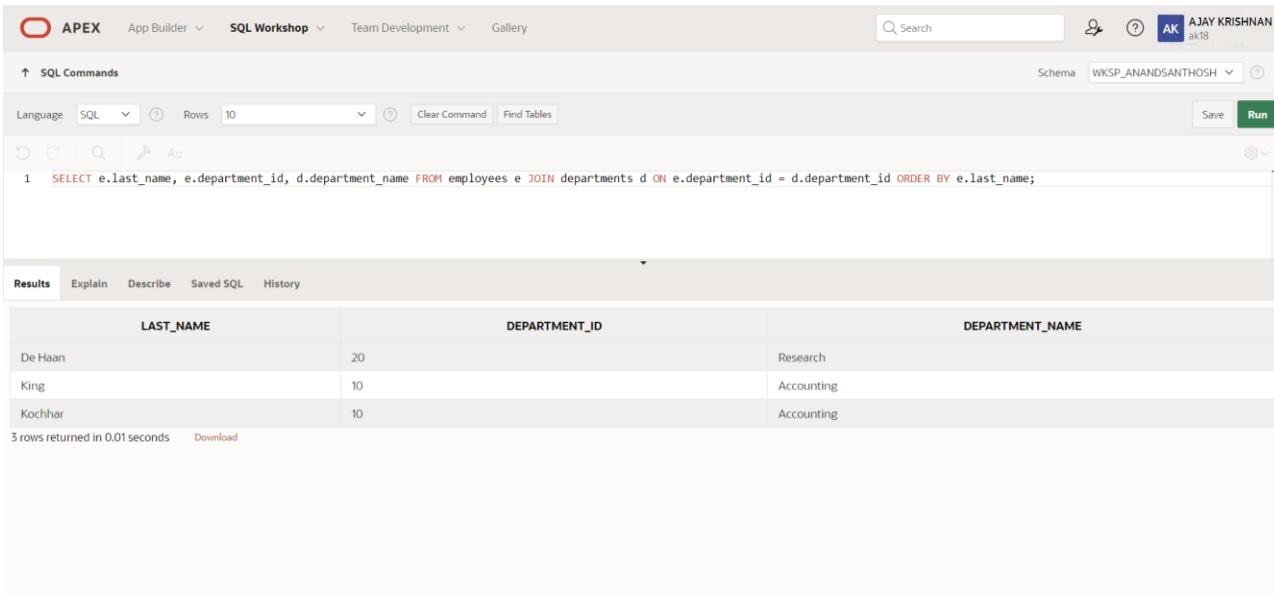
REG.NO: 220701018

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

### **QUERY:**

```
SELECT e.last_name, e.department_id, d.department_name FROM employees e JOIN departments d ON e.department_id = d.department_id ORDER BY e.last_name;
```

### **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 'AJAY KRISHNAN ak18' and a schema dropdown set to 'WKSP\_ANANDSANTHOSH'. The main area has tabs for 'SQL Commands' and 'Results'. In the 'SQL Commands' tab, the query is pasted:

```
1 SELECT e.last_name, e.department_id, d.department_name FROM employees e JOIN departments d ON e.department_id = d.department_id ORDER BY e.last_name;
```

The 'Results' tab displays the output in a table:

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
De Haan	20	Research
King	10	Accounting
Kochhar	10	Accounting

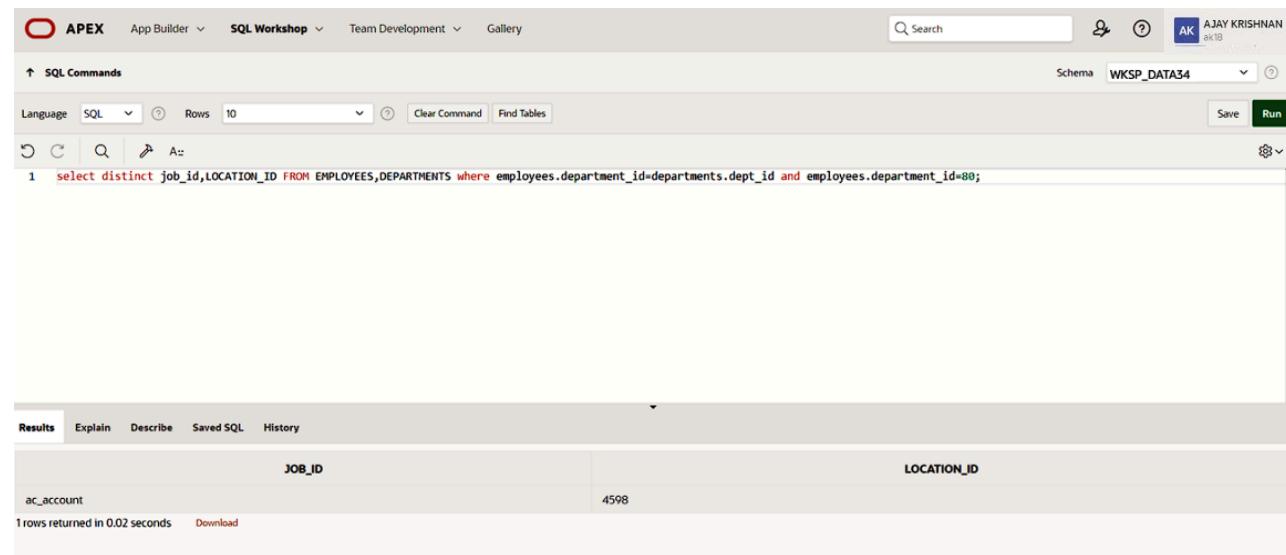
Below the table, it says '3 rows returned in 0.01 seconds'.

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

### QUERY:

```
SELECT DISTINCT j.job_title, l.city FROM jobs j JOIN departments d ON j.min_salary <= d.department_budget AND j.max_salary >= d.department_budget JOIN locations l ON d.location_id = l.location_id WHERE d.department_id = 80;
```

### 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 AJAY KRISHNAN (ak18). The main area has tabs for SQL Commands, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, showing the following command:

```
1 select distinct job_id,LOCATION_ID FROM EMPLOYEES,DEPARTMENTS where employees.department_id=departments.dept_id and employees.department_id=80;
```

The Results tab is selected, displaying the output:

JOB_ID	LOCATION_ID
ac_account	4598

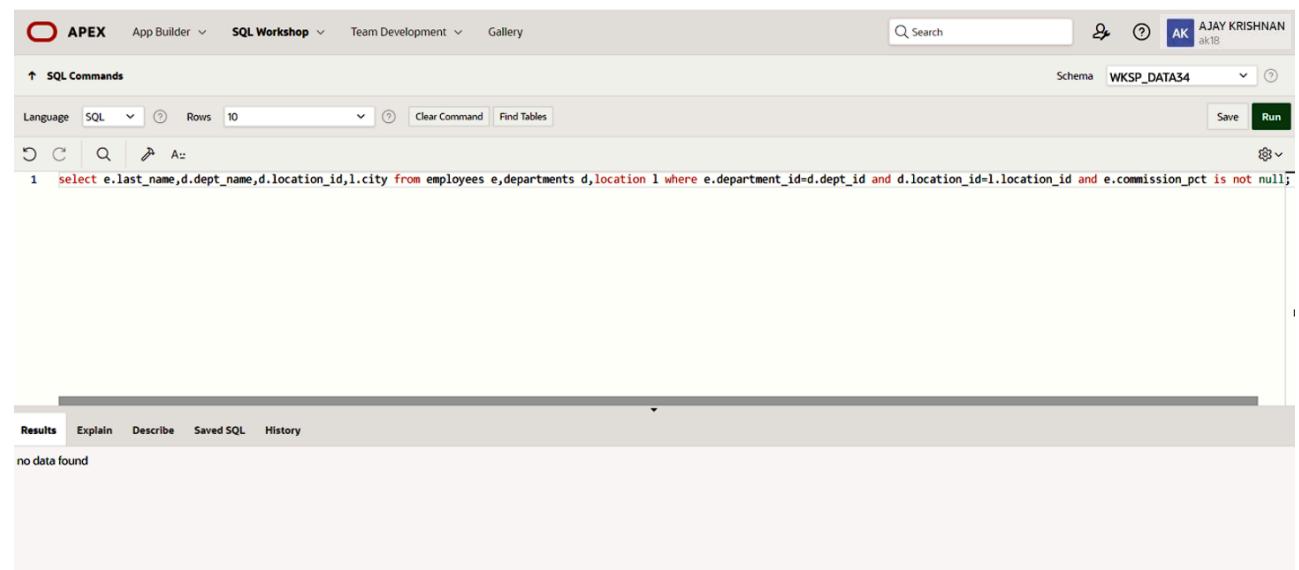
1 rows returned in 0.02 seconds

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

### QUERY:

```
SELECT e.last_name, d.department_name, l.location_id, l.city FROM employees e JOIN departments d ON e.department_id = d.department_id JOIN locations l ON d.location_id = l.location_id WHERE e.commission_pct IS NOT NULL;
```

### OUTPUT



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a user profile for AJAY KRISHNAN (ak18). The main area is titled "SQL Commands". The schema dropdown is set to WKSP\_DATA34. The command input field contains the following SQL query:

```
1 select e.last_name,d.dept_name,d.location_id,l.city from employees e,departments d,location l where e.department_id=d.dept_id and d.location_id=l.location_id and e.commission_pct is not null;
```

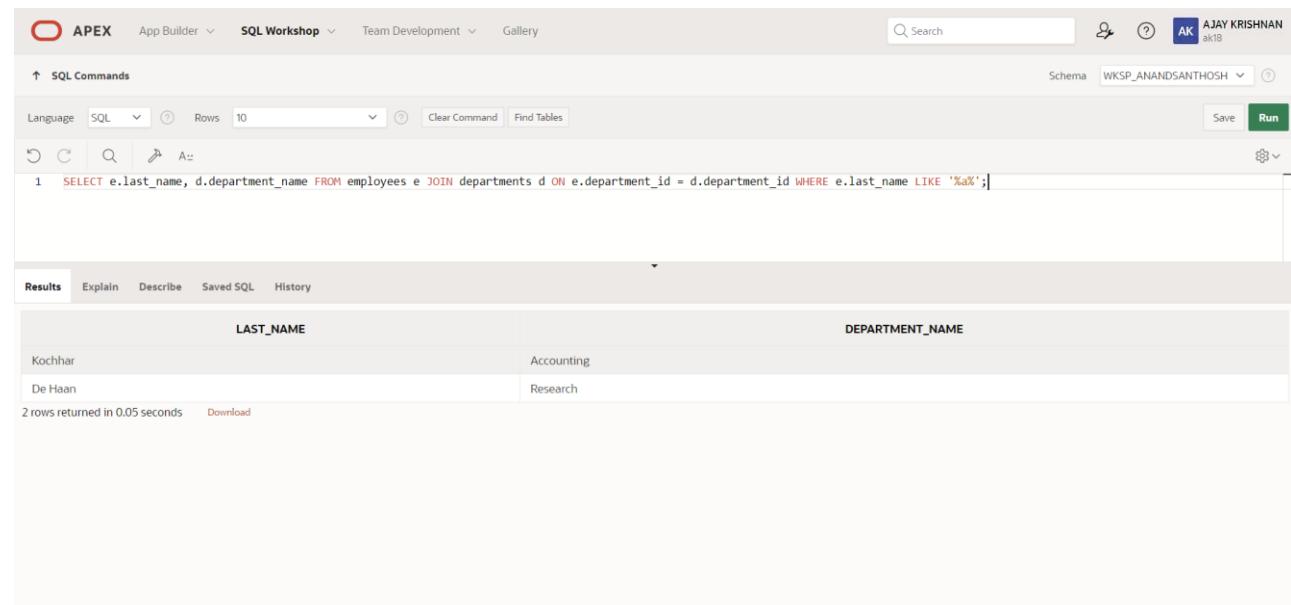
Below the command input, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, showing the message "no data found".

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

### QUERY:

```
SELECT e.last_name, d.department_name FROM employees e JOIN departments d ON e.department_id = d.department_id WHERE e.last_name LIKE '%a%';
```

### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user profile for AJAY KRISHNAN, and a schema dropdown set to WKSP\_ANANDSANTHOSH. The main area is titled "SQL Commands". It has a toolbar with icons for Undo, Redo, Find, Replace, and a Run button. Below the toolbar, a code editor displays the following SQL query:

```
1  SELECT e.last_name, d.department_name FROM employees e JOIN departments d ON e.department_id = d.department_id WHERE e.last_name LIKE '%a%';
```

Below the code editor, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected and shows a table with two rows:

LAST_NAME	DEPARTMENT_NAME
Kochhar	Accounting
De Haan	Research

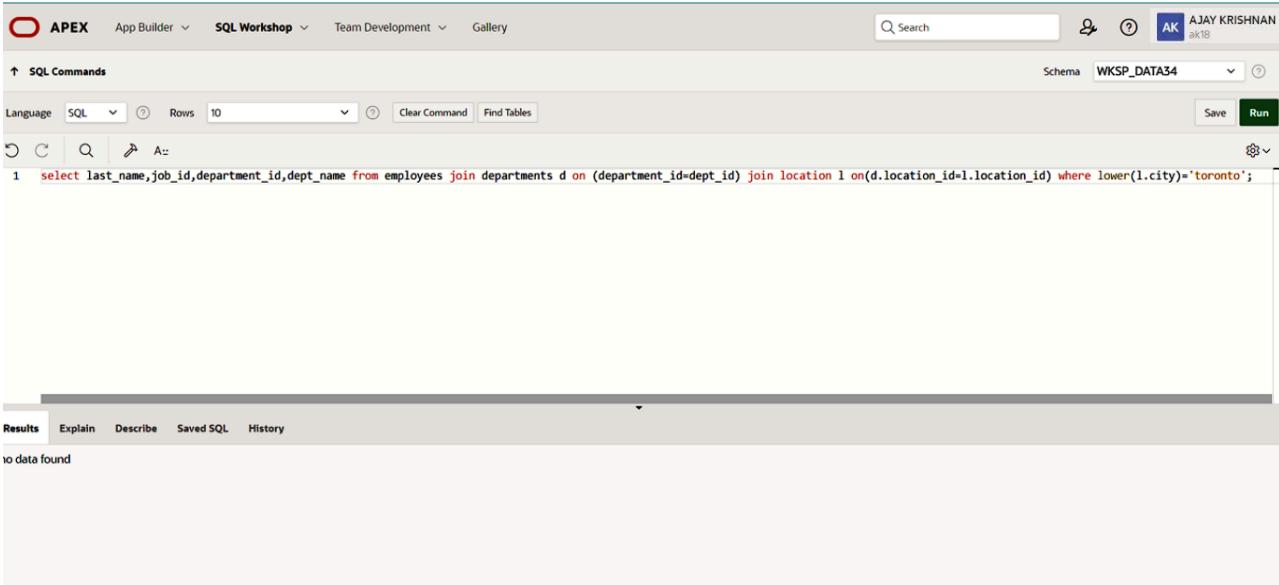
At the bottom left, it says "2 rows returned in 0.05 seconds".

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

### QUERY:

```
SELECT e.last_name, j.job_title, e.department_id, d.department_name FROM employees e JOIN departments d ON e.department_id = d.department_id JOIN jobs j ON e.job_id = j.job_id JOIN locations l ON d.location_id = l.location_id WHERE l.city = 'Toronto';
```

### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for AJAY KRISHNAN (ak18). The main area is titled 'SQL Commands' with tabs for Language (SQL selected) and Rows (set to 10). The schema is set to WKSP\_DATA34. The SQL editor contains the following query:

```
1 select last_name,job_id,department_id,dept_name from employees join departments d on (department_id=dept_id) join location l on(d.location_id=l.location_id) where lower(l.city)='toronto';
```

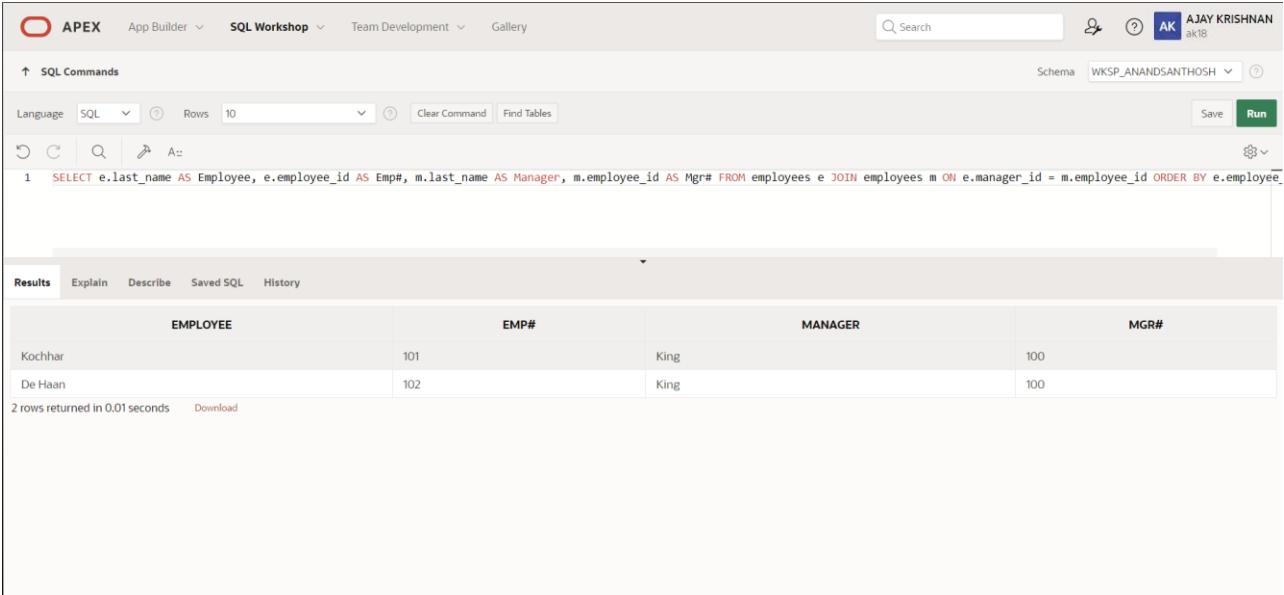
The results tab is selected at the bottom, showing the message "no data found".

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

### QUERY:

```
SELECT e.last_name AS Employee, e.employee_id AS Emp#, m.last_name AS Manager,  
m.employee_id AS Mgr# FROM employees e JOIN employees m ON e.manager_id =  
m.employee_id ORDER BY e.employee_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 is a search bar, a user icon for 'AJAY KRISHNAN', and a schema dropdown set to 'WKSP\_ANANDSANTHOSH'. The main area is titled 'SQL Commands' with a sub-section '1'. The query entered is:

```
1 SELECT e.last_name AS Employee, e.employee_id AS Emp#, m.last_name AS Manager, m.employee_id AS Mgr# FROM employees e JOIN employees m ON e.manager_id = m.employee_id ORDER BY e.employee_id;
```

Below the command, the results tab is selected. The output table has four columns: EMPLOYEE, EMP#, MANAGER, and MGR#. The data returned is:

EMPLOYEE	EMP#	MANAGER	MGR#
Kochhar	101	King	100
De Haan	102	King	100

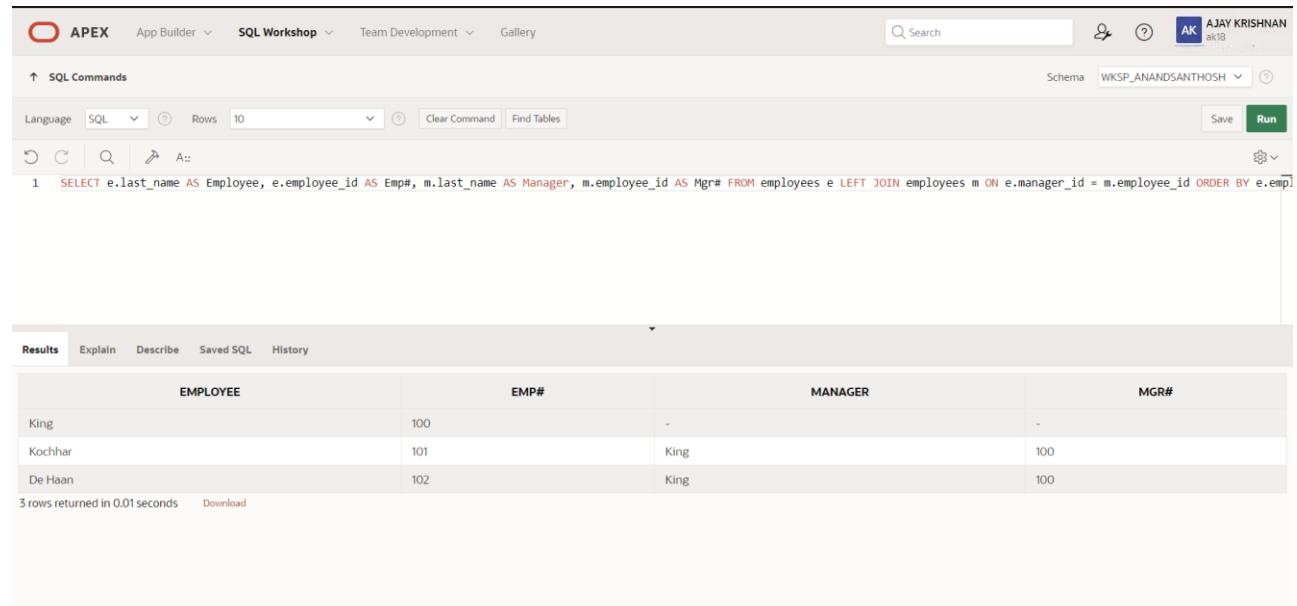
At the bottom left, it says '2 rows returned in 0.01 seconds'.

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

### QUERY:

```
SELECT e.last_name AS Employee, e.employee_id AS Emp#, m.last_name AS Manager,
m.employee_id AS Mgr# FROM employees e LEFT JOIN employees m ON e.manager_id =
m.employee_id ORDER BY e.employee_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 (AJAY KRISHNAN), and schema dropdown (WKSP\_ANANDSANTOSH). The main area is titled "SQL Commands". The query entered is:

```
1 SELECT e.last_name AS Employee, e.employee_id AS Emp#, m.last_name AS Manager, m.employee_id AS Mgr# FROM employees e LEFT JOIN employees m ON e.manager_id = m.employee_id ORDER BY e.employee_id;
```

The results section shows the output of the query:

EMPLOYEE	EMP#	MANAGER	MGR#
King	100	-	-
Kochhar	101	King	100
De Haan	102	King	100

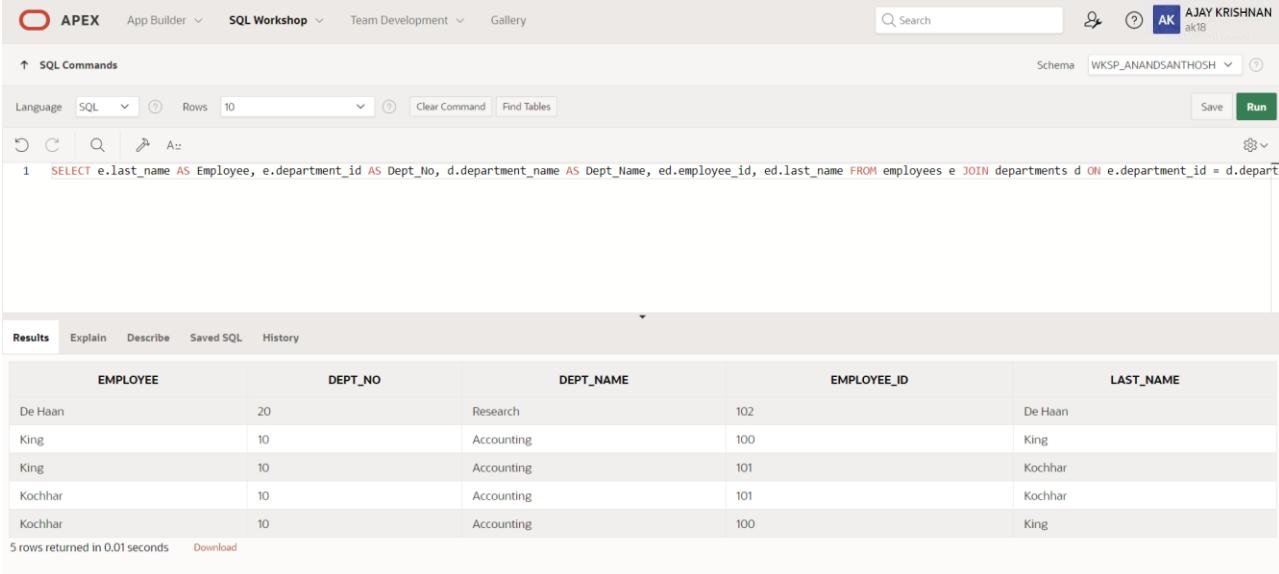
3 rows returned in 0.01 seconds [Download](#)

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

### QUERY:

```
SELECT e.last_name AS Employee, e.department_id AS Dept_No, d.department_name AS Dept_Name, ed.employee_id, ed.last_name FROM employees e JOIN departments d ON e.department_id = d.department_id LEFT JOIN employees ed ON e.department_id = ed.department_id ORDER BY e.last_name;
```

### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user profile for 'AJAY KRISHNAN', and a schema dropdown set to 'WKSP\_ANANDSANTOSH'. The main area is titled 'SQL Commands' with a 'Run' button. Below it, the SQL query is displayed:

```
1  SELECT e.last_name AS Employee, e.department_id AS Dept_No, d.department_name AS Dept_Name, ed.employee_id, ed.last_name FROM employees e JOIN departments d ON e.department_id = d.department_id LEFT JOIN employees ed ON e.department_id = ed.department_id ORDER BY e.last_name;
```

The results section shows the output of the query:

EMPLOYEE	DEPT_NO	DEPT_NAME	EMPLOYEE_ID	LAST_NAME
De Haan	20	Research	102	De Haan
King	10	Accounting	100	King
King	10	Accounting	101	Kochhar
Kochhar	10	Accounting	101	Kochhar
Kochhar	10	Accounting	100	King

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

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:

```
DESCRIBE job_grades;
```

```
SELECT e.first_name || ' ' || e.last_name AS Name, j.job_title, d.department_name, e.salary, jg.grade_level FROM employees e JOIN jobs j ON e.job_id = j.job_id JOIN departments d ON e.department_id = d.department_id JOIN job_grades jg ON e.salary BETWEEN jg.lowest_sal AND jg.highest_sal;
```

### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' and 'SQL Workshop' are selected. The main area contains the SQL command 'DESCRIBE job\_grades;'. Below it, the 'Describe' tab is active, showing the structure of the 'JOB\_GRADES' table. The table has three columns: GRADE\_LEVEL (NUMBER), LOWEST\_SAL (NUMBER), and HIGHEST\_SAL (NUMBER). The GRADE\_LEVEL column has a length of 4, precision of 0, and is defined as the primary key.

Object Type	Table	Object
Table	JOB_GRADES	JOB_GRADES

Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
JOB_GRADES.GRADE_LEVEL	NUMBER	-	4	0	1	-	-	-
JOB_GRADES.LOWEST_SAL	NUMBER	-	8	2	-	-	-	-
JOB_GRADES.HIGHEST_SAL	NUMBER	-	8	2	-	-	-	-

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' and 'SQL Workshop' are selected. The main area contains the SQL command for selecting employee details. Below it, the 'Results' tab is active, displaying the output of the query. The output shows four rows of data: Steven King (Administration President, Accounting, 24000, 6), Neena Kochhar (Administration Vice President, Accounting, 17000, 5), Lex De Haan (Administration Vice President, Research, 17000, 5), and Mary Lane (Sales Support Representative, Sales, 13000, 5).

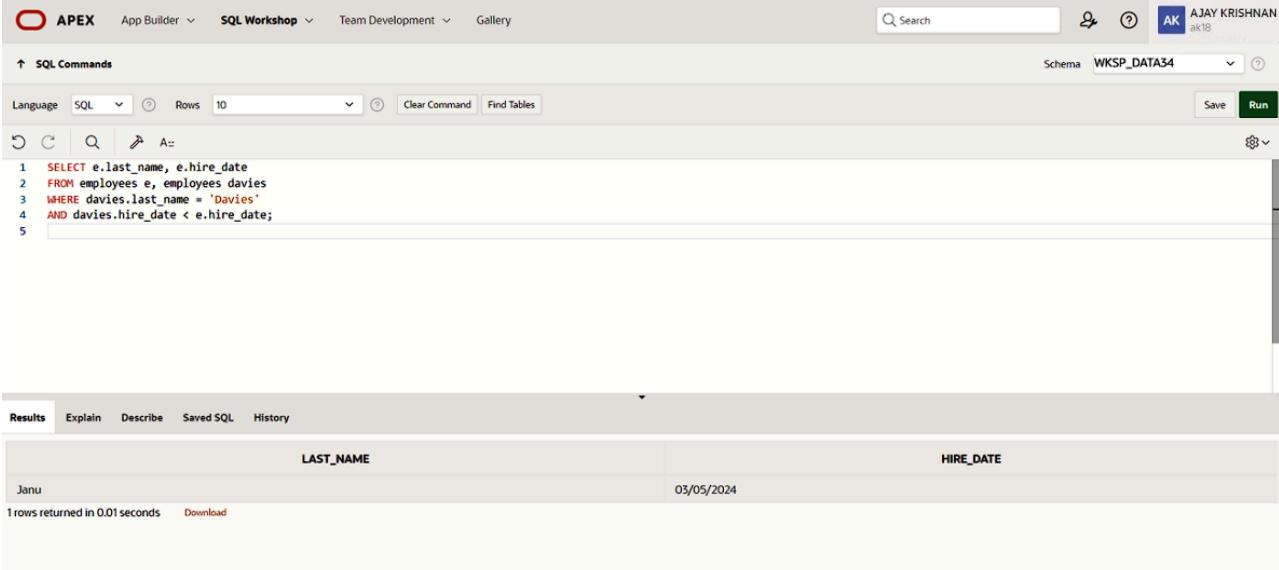
NAME	JOB_TITLE	DEPARTMENT_NAME	SALARY	GRADE_LEVEL
Steven King	Administration President	Accounting	24000	6
Neena Kochhar	Administration Vice President	Accounting	17000	5
Lex De Haan	Administration Vice President	Research	17000	5

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

**QUERY:**

```
SELECT e.first_name || ' ' || e.last_name AS Employee, e.hire_date FROM employees e WHERE e.hire_date > (SELECT e2.hire_date FROM employees e2 WHERE e2.last_name = 'Davies');
```

**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 'AJAY KRISHNAN ak18', and a schema dropdown set to 'WKSP\_DATA34'. The main area is titled 'SQL Commands' with tabs for Language (set to SQL) and Rows (set to 10). Below these are buttons for Clear Command, Find Tables, Save, and Run. The SQL editor contains the following code:

```
1 SELECT e.last_name, e.hire_date
2 FROM employees e, employees davies
3 WHERE davies.last_name = 'Davies'
4 AND davies.hire_date < e.hire_date;
5
```

Below the editor, the results tab is selected. The output table has columns 'LAST\_NAME' and 'HIRE\_DATE'. One row is shown:

LAST_NAME	HIRE_DATE
Janu	05/05/2024

At the bottom left, it says '1 rows returned in 0.01 seconds' and there's a 'Download' link.

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

### QUERY:

```
SELECT e.first_name || ' ' || e.last_name AS Employee, e.hire_date AS Emp_Hired, m.first_name || '  
' || m.last_name AS Manager, m.hire_date AS Mgr_Hired FROM employees e JOIN employees m  
ON e.manager_id = m.employee_id WHERE e.hire_date < m.hire_date ORDER BY e.hire_date;
```

### 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 are user profile icons for 'AJAY KRISHNAN' and 'WKS\_ANANDSANTHOSH'. The main area is titled 'SQL Commands' with a 'Run' button. The query is pasted into the command editor:

```
1 SELECT e.first_name || ' ' || e.last_name AS Employee, e.hire_date AS Emp_Hired, m.first_name || ' ' || m.last_name AS Manager, m.hire_date AS Mgr_Hired FROM employees e JOIN employees m  
ON e.manager_id = m.employee_id WHERE e.hire_date < m.hire_date ORDER BY e.hire_date;
```

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

EMPLOYEE	EMP_HIRED	MANAGER	MGR_HIRED
Lex De Haan	06/15/1987	Steven King	06/17/1987

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

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

# AGGREGATING DATA USING GROUP FUNCTIONS

EX.NO:8

DATE: 19 – 3 - 24

REG.NO: 220701018

Determine the validity of the following three statements. Circle either True or False.

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

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

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

**The HR department needs the following reports:**

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

**QUERY:**

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

**OUTPUT:**

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

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

The results pane displays the following data:

	Maximum	Minimum	Sum	Average
	90000	70000	407000	81400

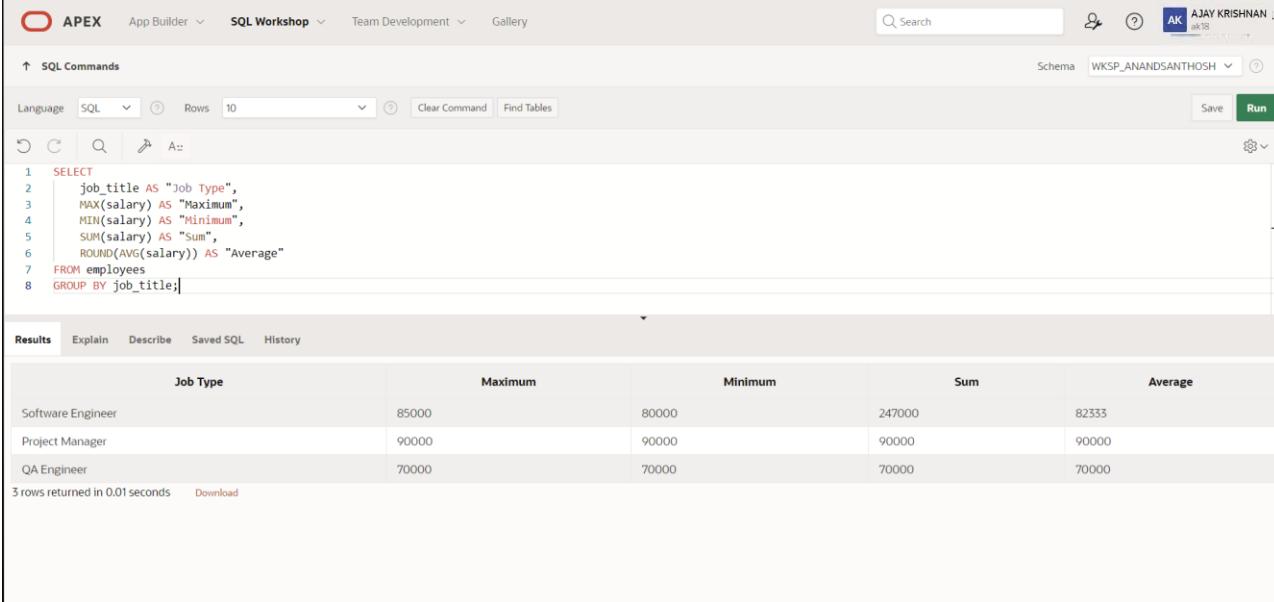
Below the table, it says "1 rows returned in 0.01 seconds".

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

### QUERY:

```
SELECT job_title AS "Job Type", MAX(salary) AS "Maximum", MIN(salary) AS "Minimum",
SUM(salary) AS "Sum", ROUND(AVG(salary)) AS "Average" FROM employees GROUP BY
job_title;
```

### OUTPUT:



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

```
1 SELECT
2   job_title AS "Job Type",
3   MAX(salary) AS "Maximum",
4   MIN(salary) AS "Minimum",
5   SUM(salary) AS "Sum",
6   ROUND(AVG(salary)) AS "Average"
7  FROM employees
8 GROUP BY job_title;
```

The Results pane displays the output of the query:

Job Type	Maximum	Minimum	Sum	Average
Software Engineer	85000	80000	247000	82333
Project Manager	90000	90000	90000	90000
QA Engineer	70000	70000	70000	70000

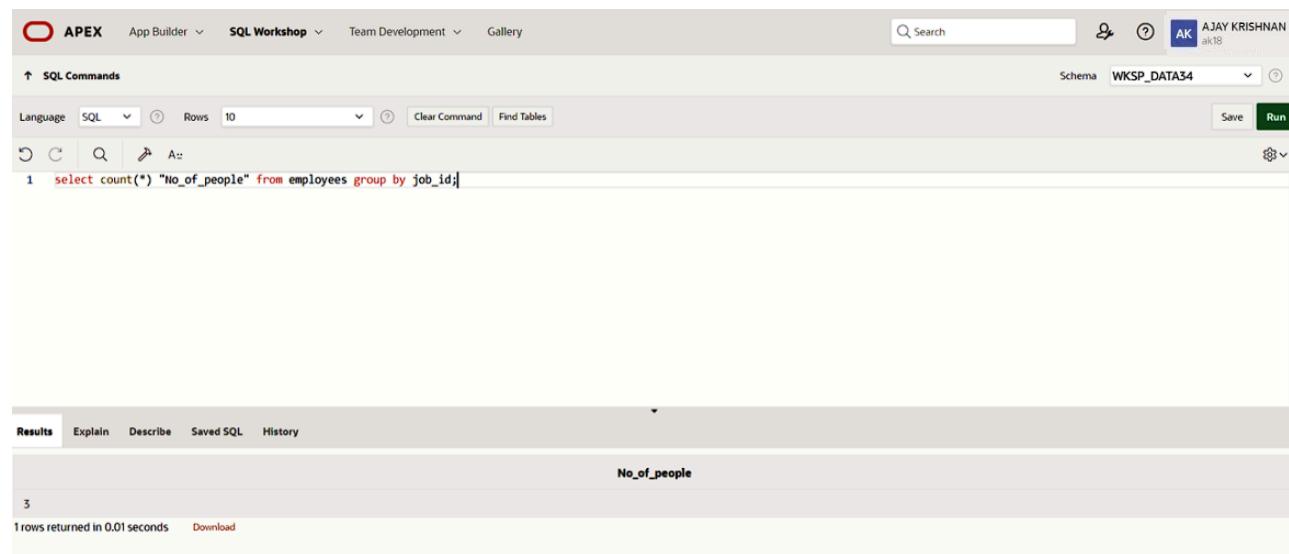
3 rows returned in 0.01 seconds [Download](#)

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

### QUERY:

```
SELECT job_title, COUNT(*) AS "Number of Employees" FROM employees WHERE job_title = &job_title;
```

### 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 AJAY KRISHNAN (ak18). The main area is titled 'SQL Commands' with tabs for Language (set to SQL), Rows (set to 10), Clear Command, Find Tables, Save, and Run buttons. The command entered is:

```
1 select count(*) "No_of_people" from employees group by job_id;
```

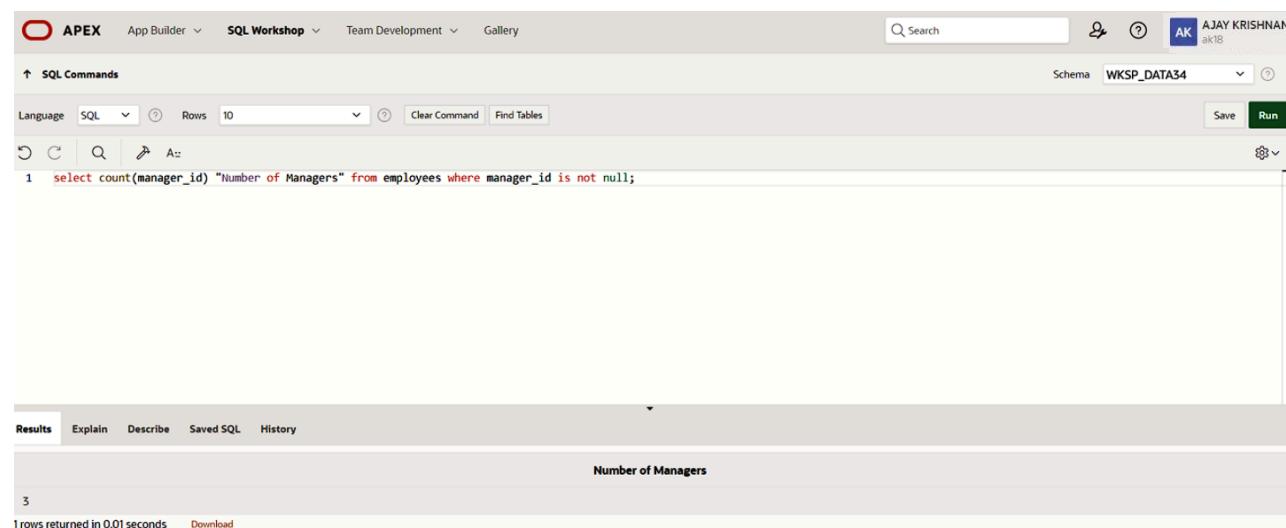
The results section shows a single row with the value '3' under the column 'No\_of\_people'. Below the results, it says '1 rows returned in 0.01 seconds' and has a 'Download' link.

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

### QUERY:

```
SELECT COUNT(*) AS "Number of Managers" FROM employees WHERE manager_id IS NOT NULL;
```

### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user profile for AJAY KRISHNAN (ak18), and a schema dropdown set to WKSP\_DATA34. The main workspace is titled 'SQL Commands' and contains the following SQL code:

```
1 select count(manager_id) "Number of Managers" from employees where manager_id is not null;
```

Below the code, the results tab is selected, showing a single row with the value '3' under the column 'Number of Managers'. The status bar at the bottom indicates '1 rows returned in 0.01 seconds' and provides a 'Download' link.

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

### QUERY:

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

### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. At the top, there are tabs for APEX, App Builder, SQL Workshop (which is selected), Team Development, and Gallery. On the right, there's a user profile for AJAY KRISHNAN (ak18). The main area is titled 'SQL Commands' with a 'Run' button. The code entered is:

```
1 SELECT
2   | MAX(salary) - MIN(salary) AS "DIFFERENCE"
3   | FROM employees;
```

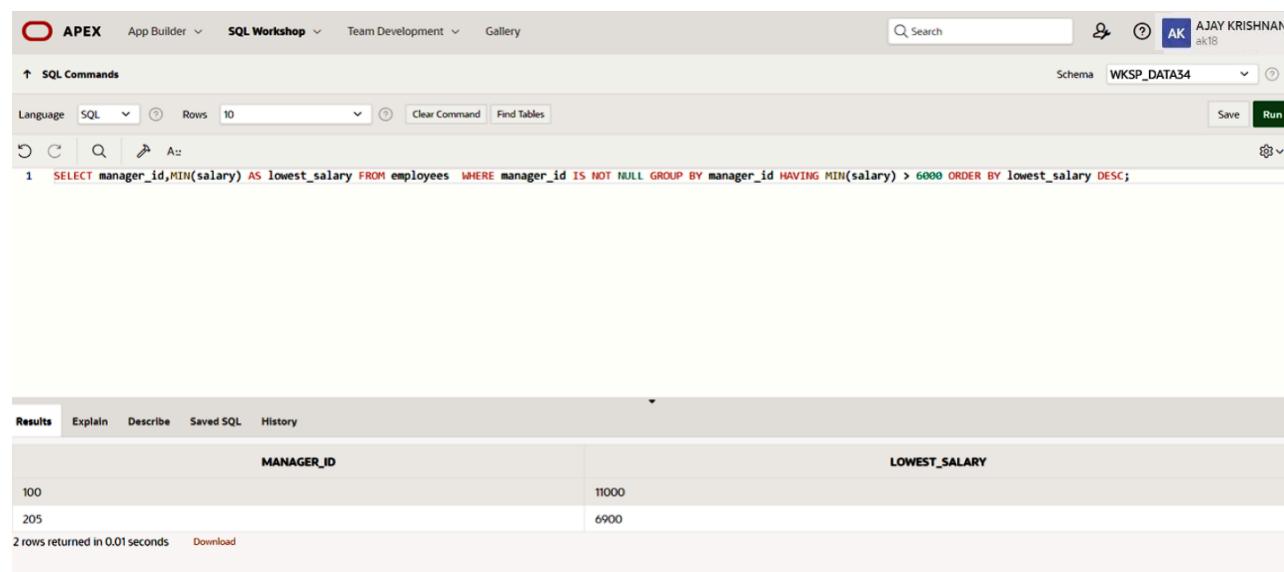
Below the command window, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, showing a single row with the value 20000 under the column labeled 'DIFFERENCE'. Below the table, it says '1 rows returned in 0.00 seconds' and has a 'Download' link.

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

### QUERY:

```
SELECT manager_id, MIN(salary) AS "Lowest Salary" FROM employees WHERE manager_id IS NOT NULL AND salary > 6000 GROUP BY manager_id ORDER BY MIN(salary) DESC;
```

### 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 AJAY KRISHNAN (ak18). The main area has tabs for SQL Commands, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, displaying the following query:

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

Below the query, the Results tab is selected, showing the following table:

MANAGER_ID	LOWEST_SALARY
100	11000
205	6900

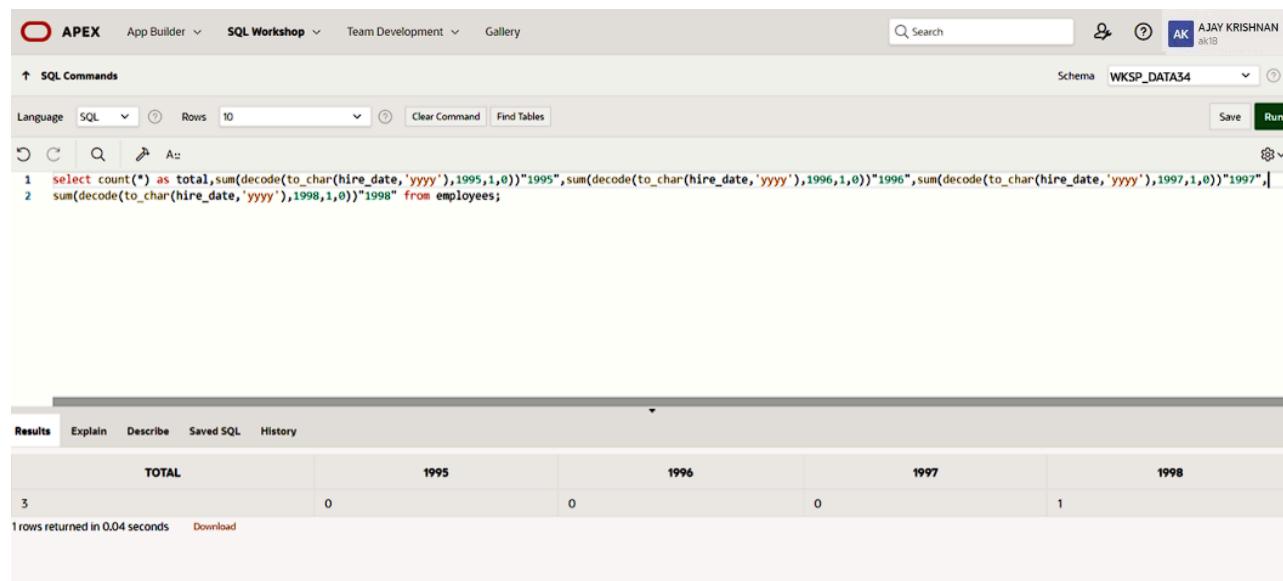
Text at the bottom left indicates "2 rows returned in 0.01 seconds".

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

### QUERY:

```
SELECT COUNT(*) AS "Total Employees",
SUM(CASE WHEN hire_date LIKE '1995%' THEN 1 ELSE 0 END) AS "Employees in 1995",
SUM(CASE WHEN hire_date LIKE '1996%' THEN 1 ELSE 0 END) AS "Employees in 1996",
SUM(CASE WHEN hire_date LIKE '1997%' THEN 1 ELSE 0 END) AS "Employees in 1997",
SUM(CASE WHEN hire_date LIKE '1998%' THEN 1 ELSE 0 END) AS "Employees in 1998"
FROM employees;
```

### 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 AJAY KRISHNAN and a schema dropdown set to WKSP\_DATA34. The main area is titled 'SQL Commands' with tabs for Language (set to SQL), Rows (set to 10), and buttons for Clear Command and Find Tables. Below this is a toolbar with icons for Undo, Redo, Search, and Run. The SQL editor contains the following code:

```
1 select count(*) as total,sum(decode(to_char(hire_date,'yyyy'),1995,1,0))"1995",sum(decode(to_char(hire_date,'yyyy'),1996,1,0))"1996",sum(decode(to_char(hire_date,'yyyy'),1997,1,0))"1997",
2 sum(decode(to_char(hire_date,'yyyy'),1998,1,0))"1998" from employees;
```

Below the editor is a results grid. The 'Results' tab is selected, showing the following data:

TOTAL	1995	1996	1997	1998
3	0	0	0	1

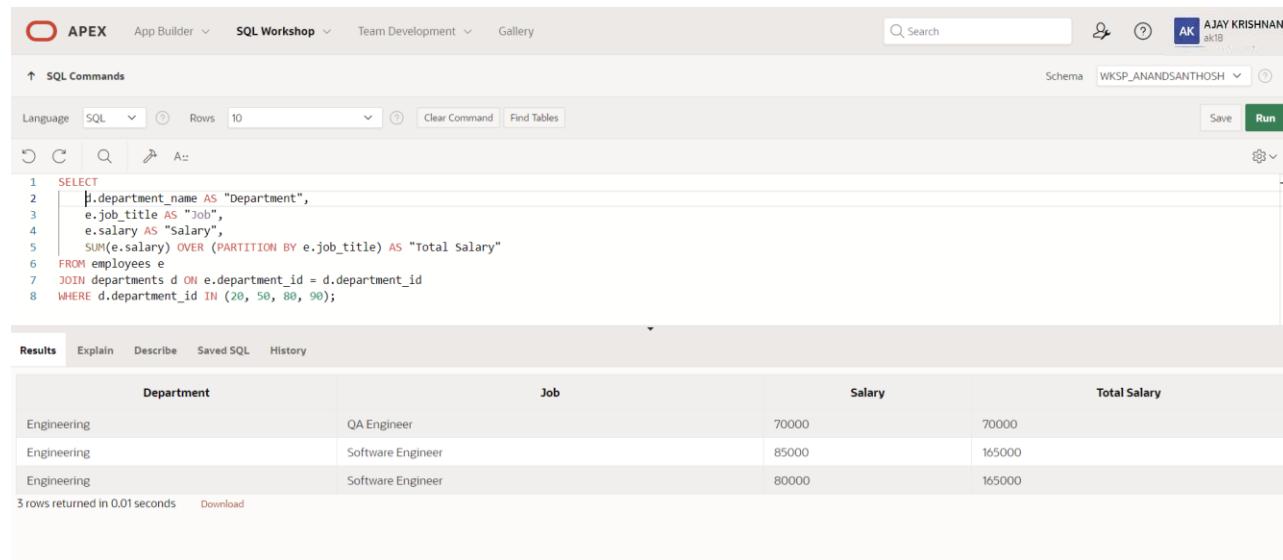
At the bottom left, it says '1 rows returned in 0.04 seconds'. There are also 'Download' and 'Run' buttons.

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 d.department_name AS "Department", e.job_title AS "Job", e.salary AS "Salary",
SUM(e.salary) OVER (PARTITION BY e.job_title) AS "Total Salary"
FROM employees e JOIN departments d ON e.department_id = d.department_id
WHERE d.department_id IN (20, 50, 80, 90);
```

### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The SQL command is pasted into the editor, and the results are displayed in a grid format below. The results show three rows of data corresponding to the departments 20, 50, and 80.

Department	Job	Salary	Total Salary
Engineering	QA Engineer	70000	70000
Engineering	Software Engineer	85000	165000
Engineering	Software Engineer	80000	165000

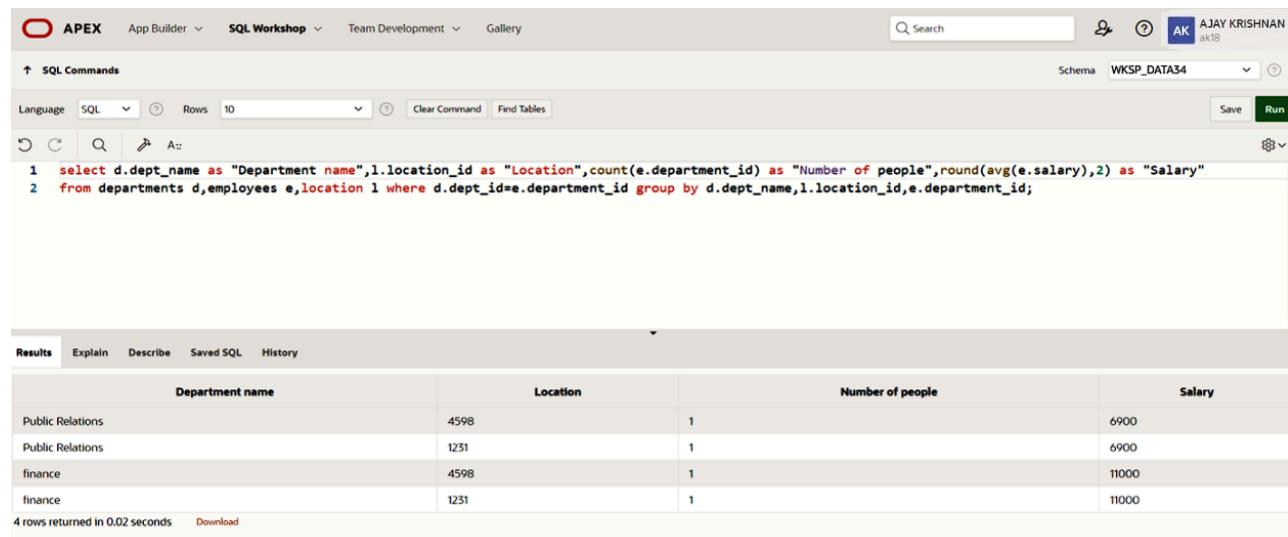
3 rows returned in 0.01 seconds    Download

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

### QUERY:

```
SELECT d.department_name AS "Location", d.location_id AS "Department", COUNT(*) AS  
"Number of people", ROUND(AVG(e.salary), 2) AS "Salary" FROM employees e  
JOIN departments d ON e.department_id = d.department_id GROUP BY d.department_name,  
d.location ORDER BY "Salary" DESC;
```

### 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 AJAY KRISHNAN. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP\_DATA34'. The code entered is:

```
1 select d.dept_name as "Department name",l.location_id as "Location",count(e.department_id) as "Number of people",round(avg(e.salary),2) as "Salary"  
2 from departments d,employees e,location l where d.dept_id=e.department_id group by d.dept_name,l.location_id,e.department_id;
```

The results section displays the following data:

Department name	Location	Number of people	Salary
Public Relations	4598	1	6900
Public Relations	1231	1	6900
finance	4598	1	11000
finance	1231	1	11000

4 rows returned in 0.02 seconds    Download

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

# SUB QUERIES

EX.NO:9

DATE: 26 – 3 - 24

REG.NO: 220701018

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

## **QUERY:**

```
SELECT last_name, hire_date FROM employees  
WHERE department_id IN (SELECT department_id FROM employees WHERE last_name =  
:last_name) AND employee_id != (SELECT employee_id FROM employees WHERE last_name  
:last_name);
```

## **OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. In the top section, there is a 'Bind Variable' dialog with 'LAST\_NAME' set to 'Austin'. Below it, the SQL command is displayed:

```
1 SELECT last_name, hire_date  
2   FROM employees  
3 WHERE department_id IN (SELECT department_id FROM employees WHERE last_name = :last_name)  
4 AND employee_id != (SELECT employee_id FROM employees WHERE last_name = :last_name);
```

In the bottom section, the results are shown in a table:

LAST_NAME	HIRE_DATE
Hunold	01/03/2006
Ernst	05/21/2007
Pataballa	03/05/2006
Lorentz	02/07/2007

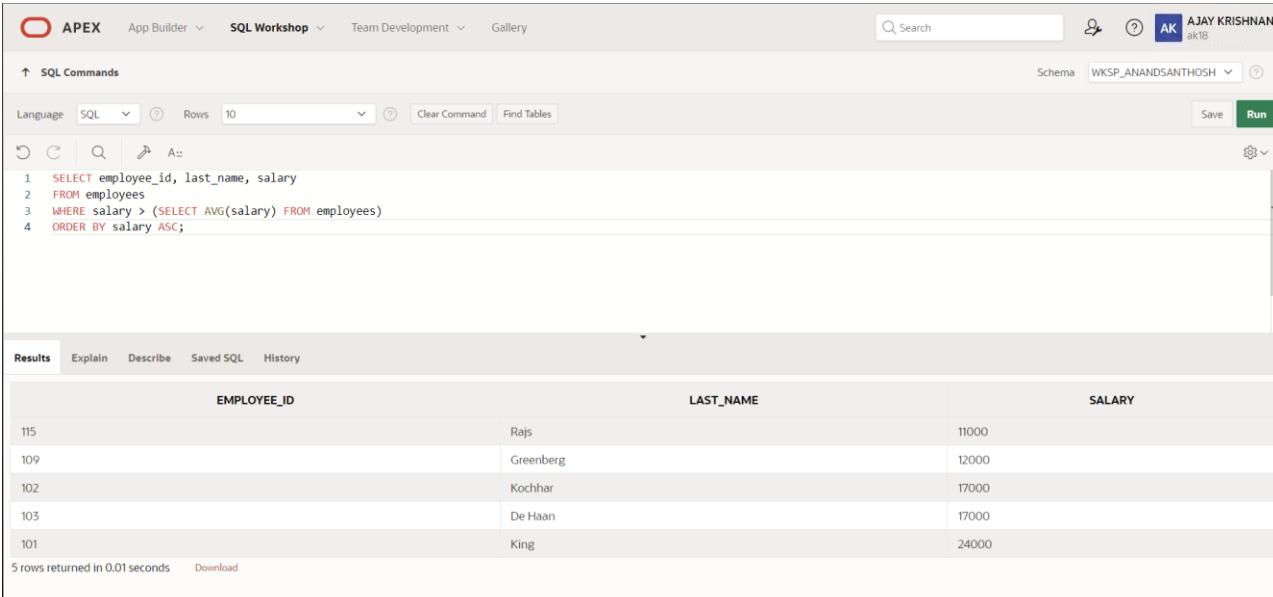
4 rows returned in 0.02 seconds

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

### QUERY:

```
SELECT employee_id, last_name, salary FROM employees WHERE salary > (SELECT AVG(salary) FROM employees) ORDER BY salary ASC;
```

### 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 'AJAY KRISHNAN ak18' and a search bar. Below the toolbar, the schema is set to 'WKSP\_ANANDSANTHOSH'. The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the following query is displayed:

```
1 SELECT employee_id, last_name, salary
2 FROM employees
3 WHERE salary > (SELECT AVG(salary) FROM employees)
4 ORDER BY salary ASC;
```

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

EMPLOYEE_ID	LAST_NAME	SALARY
115	Rajs	11000
109	Greenberg	12000
102	Kochhar	17000
103	De Haan	17000
101	King	24000

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

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

### QUERY:

```
SELECT employee_id, last_name  
FROM employees  
WHERE department_id IN (SELECT department_id FROM employees WHERE last_name LIKE  
'%u%');
```

### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, the tabs 'SQL Workshop' and 'Team Development' are visible. On the right side, the user 'AJAY KRISHNAN' and their schema 'WKSP\_ANANDSANTOSH' are shown.

In the main area, the 'SQL Commands' tab is active. The SQL code entered is:

```
1 SELECT employee_id, last_name  
2 FROM employees  
3 WHERE department_id IN (SELECT department_id FROM employees WHERE last_name LIKE '%u%');
```

Below the code, the 'Results' tab is selected. The output is a table with two columns: 'EMPLOYEE\_ID' and 'LAST\_NAME'. The data returned is:

EMPLOYEE_ID	LAST_NAME
104	Hunold
105	Ernst
106	Austin
107	Pataballa
108	Lorentz

At the bottom left, it says '5 rows returned in 0.01 seconds'. There is also a 'Download' link.

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

### QUERY:

```
SELECT last_name, department_id, job_id FROM employees WHERE department_id IN  
(SELECT department_id FROM departments WHERE location_id = 1700);
```

### OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user's profile: AJAY KRISHNAN (ak18). The main area is titled "SQL Commands". The command entered is:

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

The results tab is selected, displaying the following data:

LAST_NAME	DEPARTMENT_ID	JOB_ID
Janu	100	ac_account
Doe	100	ac_account

Below the table, it says "2 rows returned in 0.04 seconds" and there is a "Download" link.

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

### QUERY:

```
SELECT last_name, salary  
FROM employees  
WHERE manager_id = (SELECT employee_id FROM employees WHERE last_name = 'King');
```

### OUTPUT:

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

```
1 SELECT last_name, salary  
2 FROM employees  
3 WHERE manager_id = (SELECT employee_id FROM employees WHERE last_name = 'King');
```

The results section shows a table with two rows:

LAST_NAME	SALARY
Kochhar	17000
De Haan	17000

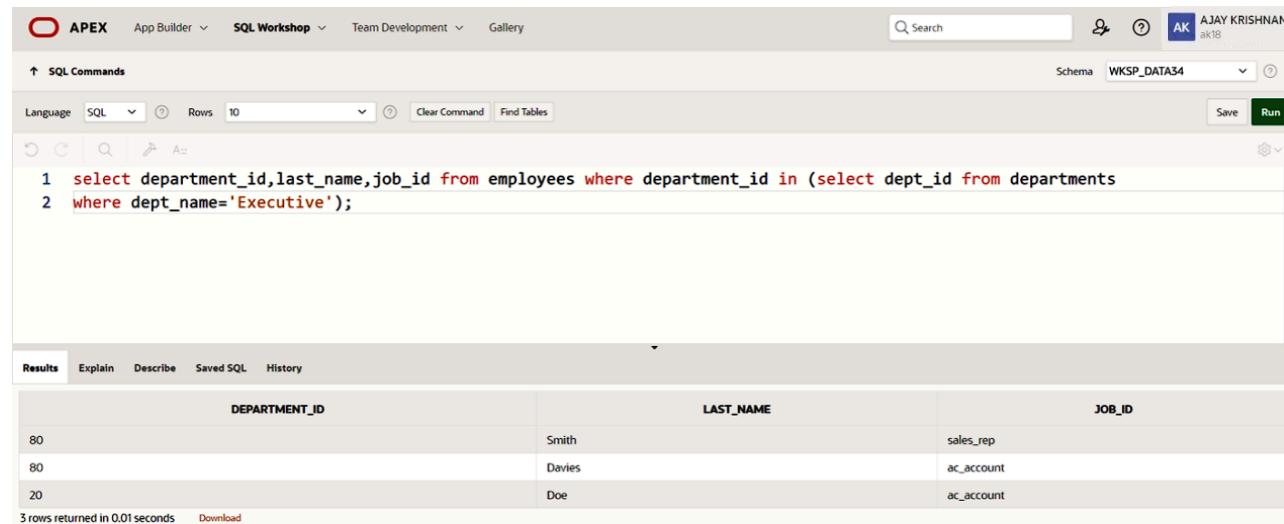
Below the table, it says '2 rows returned in 0.01 seconds' and there is a 'Download' link.

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

**QUERY:**

```
SELECT department_id, last_name, job_id  
FROM employees  
WHERE department_id IN (SELECT department_id FROM departments WHERE  
department_name = 'Executive');
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user's profile: AJAY KRISHNAN (ak18). The main area has a search bar and a schema dropdown set to WKSP\_DATA34. The SQL Commands tab is active, showing the following code:

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

Below the code, the Results tab is selected, displaying the query results in a grid:

DEPARTMENT_ID	LAST_NAME	JOB_ID
80	Smith	SALES REP
80	Davies	AC_ACCOUNT
20	Doe	AC_ACCOUNT

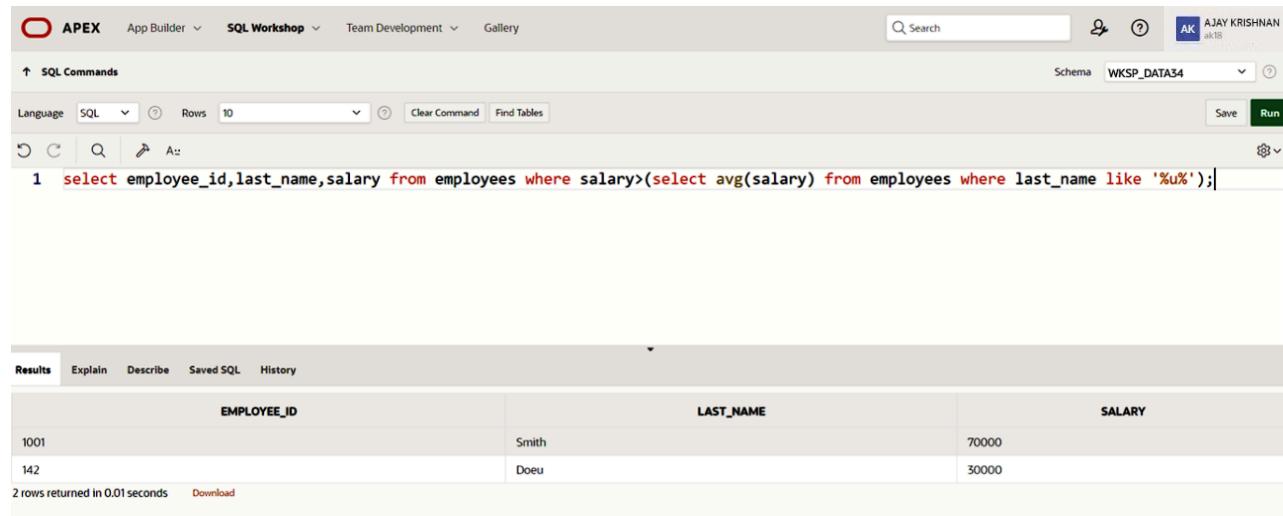
At the bottom left, it says "3 rows returned in 0.01 seconds".

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

**QUERY:**

```
SELECT employee_id, last_name, salary FROM employees WHERE salary > (SELECT AVG(salary) FROM employees) AND department_id IN (SELECT department_id FROM employees WHERE last_name LIKE "%u%");
```

**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 AJAY KRISHNAN. The main area has tabs for SQL Commands, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, showing the following SQL code in the command line:

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

Below the command line, the Results tab is selected, displaying the query results in a table:

EMPLOYEE_ID	LAST_NAME	SALARY
1001	Smith	70000
142	Doeu	30000

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

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

# USING THE SET OPERATORS

EX.NO:10

DATE: 05 - 4 - 24

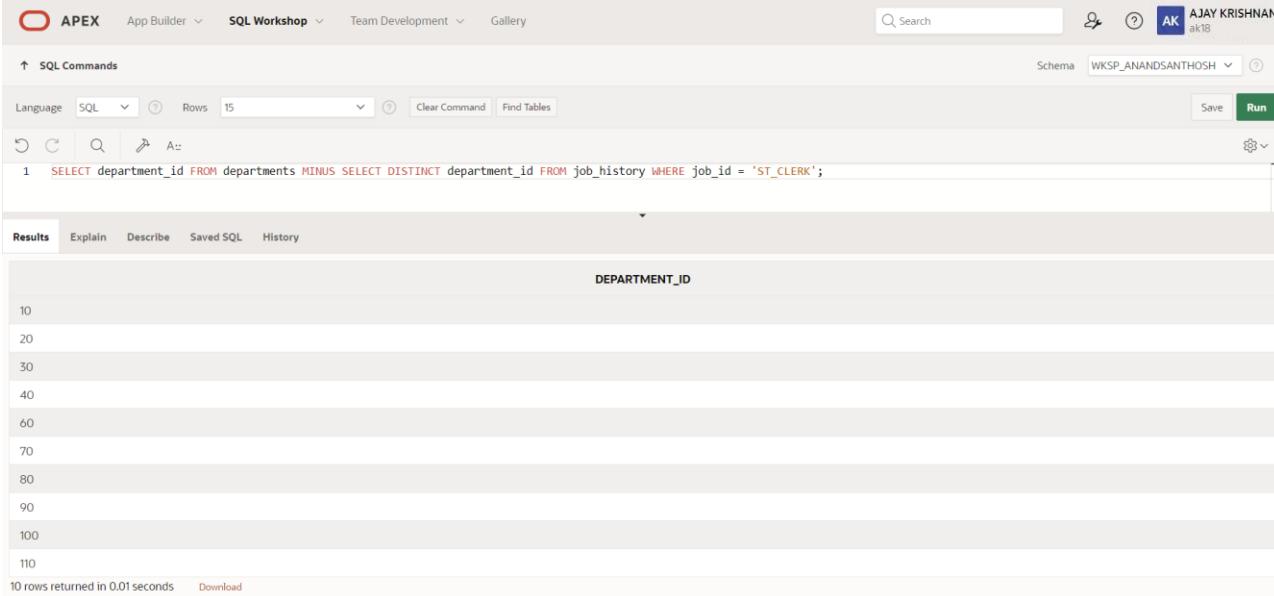
REG.NO: 220701018

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

**QUERY:**

```
SELECT department_id FROM departments MINUS SELECT DISTINCT department_id FROM job_history WHERE job_id = 'ST_CLERK';
```

**OUTPUT:**



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

```
1  SELECT department_id FROM departments MINUS SELECT DISTINCT department_id FROM job_history WHERE job_id = 'ST_CLERK';
```

The results table has one column labeled "DEPARTMENT\_ID" with the following data:

DEPARTMENT_ID
10
20
30
40
60
70
80
90
100
110

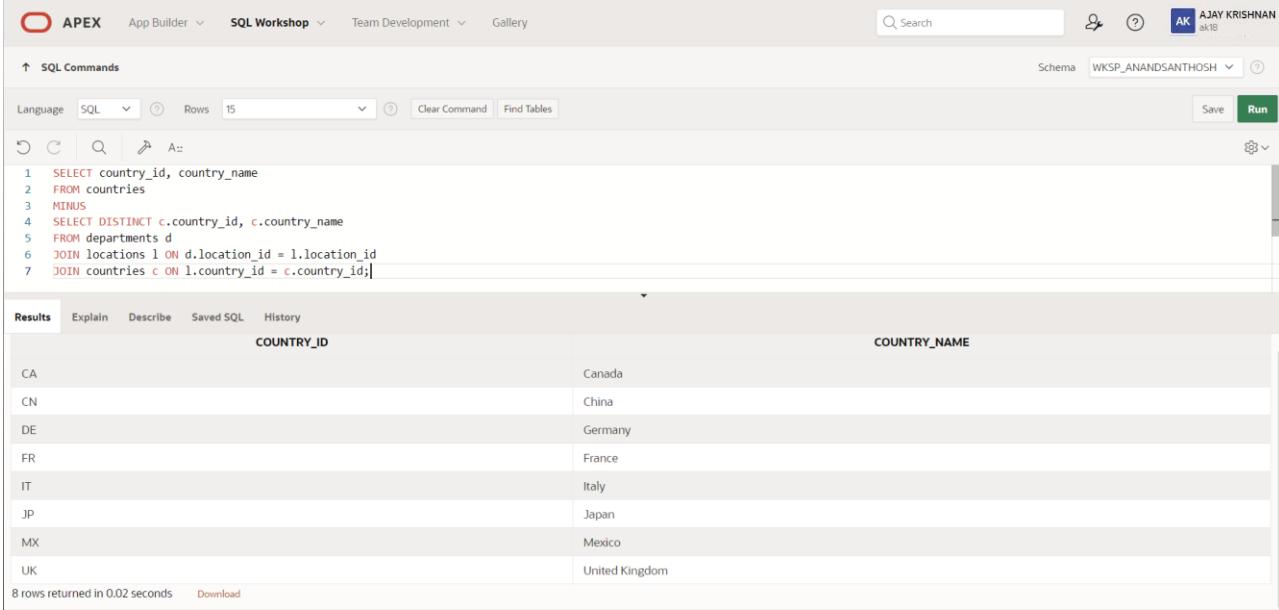
At the bottom, it says "10 rows returned in 0.01 seconds".

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

### QUERY:

```
SELECT country_id, country_name FROM countries
MINUS
SELECT DISTINCT c.country_id, c.country_name FROM departments d
JOIN locations l ON d.location_id = l.location_id
JOIN countries c ON 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'. On the right, there's a user profile for 'AJAY KRISHNAN' and a schema dropdown for 'WKSP\_ANANDSANTOSH'. The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the following query is displayed:

```
1 SELECT country_id, country_name
2 FROM countries
3 MINUS
4 SELECT DISTINCT c.country_id, c.country_name
5 FROM departments d
6 JOIN locations l ON d.location_id = l.location_id
7 JOIN countries c ON l.country_id = c.country_id;
```

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

COUNTRY_ID	COUNTRY_NAME
CA	Canada
CN	China
DE	Germany
FR	France
IT	Italy
JP	Japan
MX	Mexico
UK	United Kingdom

At the bottom left, it says '8 rows returned in 0.02 seconds' and there's 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 job_id,department_id from employees where department_id=10 union select job_id,department_id  
from employees where department_id=50 union select job_id,department_id from employees where  
department_id=20;
```

**OUTPUT:**

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 'AJAY KRISHNAN', and a schema dropdown set to 'WKSP\_DATA34'. The main area has tabs for SQL Commands, Explain, Describe, Saved SQL, and History. The SQL Commands tab shows the following code:

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

Below the code, the Results tab is selected, displaying a table with two rows:

JOB_ID	DEPARTMENT_ID
ac_account	20
hr_rep	20

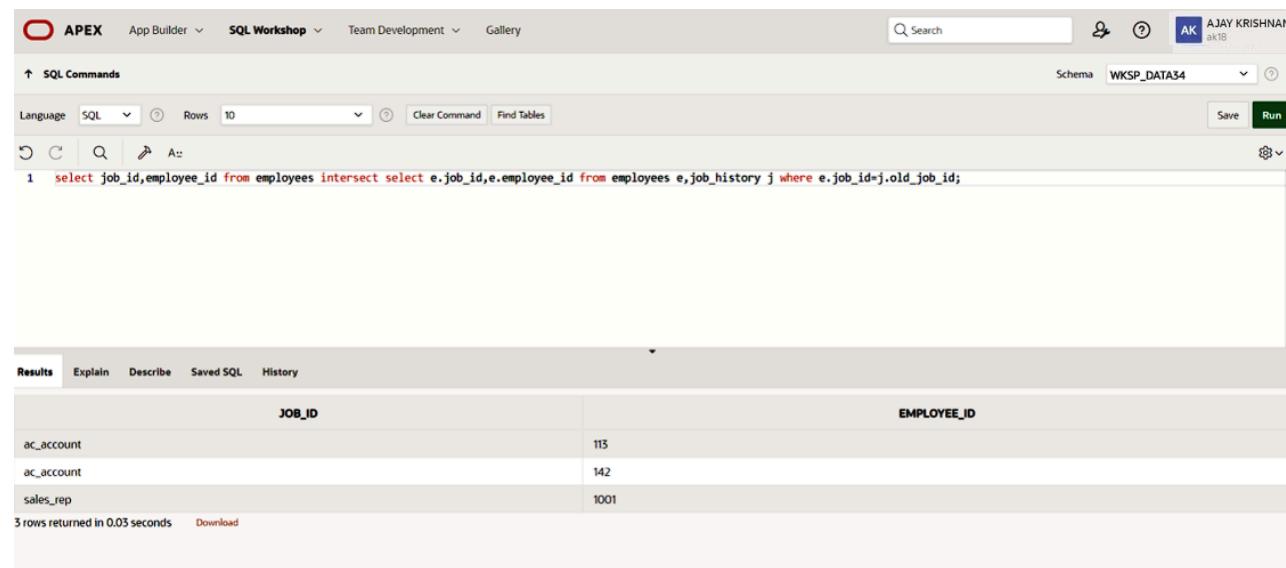
At the bottom left, it says '2 rows returned in 0.01 seconds' and there's a 'Download' link.

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

### QUERY:

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

### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'AJAY KRISHNAN ak18', and a schema dropdown set to 'WKSP\_DATA34'. The main area is titled 'SQL Commands' with a 'Run' button. The query entered is:

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

The results section shows a table with two columns: 'JOB\_ID' and 'EMPLOYEE\_ID'. The data returned is:

JOB_ID	EMPLOYEE_ID
ac_account	113
ac_account	142
sales_rep	1001

Below the table, it says '3 rows returned in 0.03 seconds'.

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

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

### QUERY:

```
SELECT last_name, department_id FROM employees UNION SELECT department_name, department_id FROM departments;
```

### 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 (AJAY KRISHNAN), and schema dropdown (WKSP\_ANANDSANTHOSH). The main area has tabs for SQL Commands, Results (selected), Explain, Describe, Saved SQL, and History. The SQL command entered is:

```
1 SELECT last_name, department_id FROM employees UNION SELECT department_name, department_id FROM departments;
```

The Results tab displays the output in a grid format:

LAST_NAME	DEPARTMENT_ID
Chen	100
Accounting	110
Administration	10
Austin	60
De Haan	90
Ernst	60
Executive	90
Faviet	100
Finance	100
Greenberg	100
Human Resources	40

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

# USING THE SET OPERATORS

EX.NO:11

DATE: 23 – 4 - 24

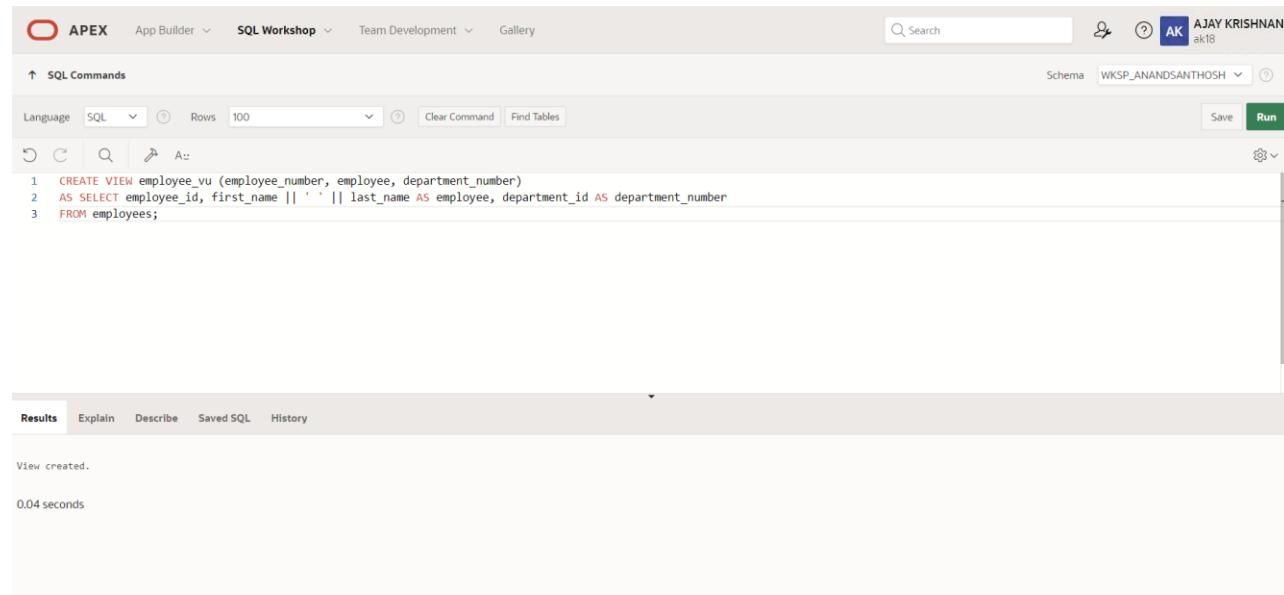
REG.NO: 220701018

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 (employee_number, employee, department_number)
AS SELECT employee_id, first_name || ' ' || last_name AS employee, department_id AS
department_number
FROM employees;
```

## **OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. In the top right corner, there is a user profile for 'AJAY KRISHNAN' with the ID 'ak18'. The main area displays the SQL command for creating the 'employee\_vu' view:

```
1 CREATE VIEW employee_vu (employee_number, employee, department_number)
2 AS SELECT employee_id, first_name || ' ' || last_name AS employee, department_id AS department_number
3 FROM employees;
```

Below the command, the results section shows the message 'View created.' and a execution time of '0.04 seconds'.

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

**QUERY:**

```
SELECT * FROM employee_vu;
```

**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 'AJAY KRISHNAN ak18', and a schema dropdown set to 'WKSP\_ANANDSANTHOSH'. The main area is titled 'SQL Commands' with a 'Run' button. Below it, the query 'SELECT \* FROM employee\_vu;' is entered. The results section displays the following data:

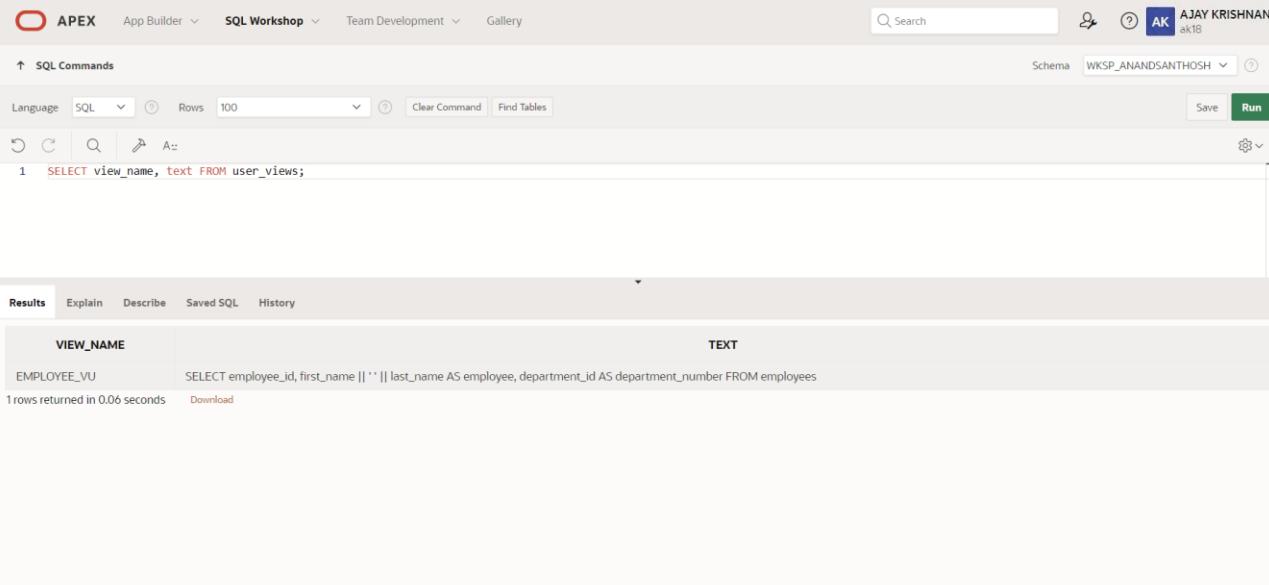
EMPLOYEE_NUMBER	EMPLOYEE	DEPARTMENT_NUMBER
102	Neena Kochhar	90
110	Daniel Faviet	100
107	Valli Pataballa	60
108	Diana Lorentz	60
111	John Chen	100
106	David Austin	60
103	Lex De Haan	90
105	Bruce Ernst	60
109	Nancy Greenberg	100
101	Steven King	90
104	Alexander Hunold	60
112	Ismael Scott	100

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle 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 for AJAY KRISHNAN, and a schema dropdown set to WKSP\_ANANDSANTHOSH. The main area is titled "SQL Commands" and contains the following content:

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

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

VIEW_NAME	TEXT
EMPLOYEE_VU	SELECT employee_id, first_name    ''    last_name AS employee, department_id AS department_number FROM employees

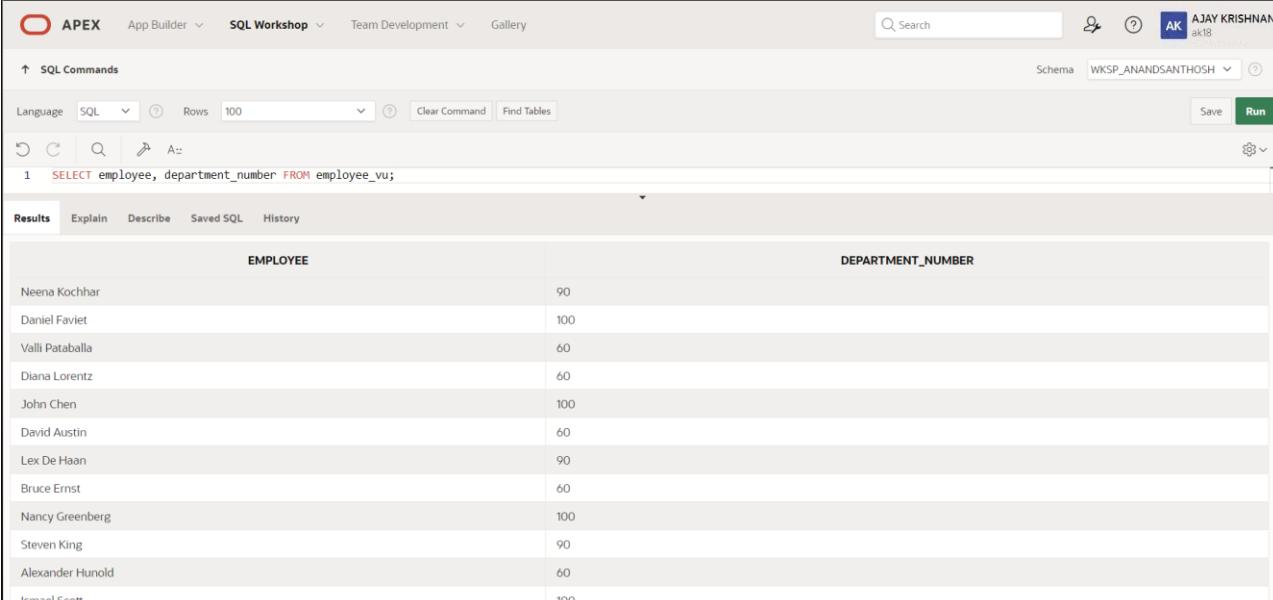
At the bottom left, it says "1 rows returned in 0.06 seconds". There is also a "Download" link.

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

### QUERY:

```
SELECT employee, department_number FROM employee_vu;
```

### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The query `SELECT employee, department_number FROM employee_vu;` has been run, and the results are displayed in a table format. The table has two columns: `EMPLOYEE` and `DEPARTMENT_NUMBER`. The data is as follows:

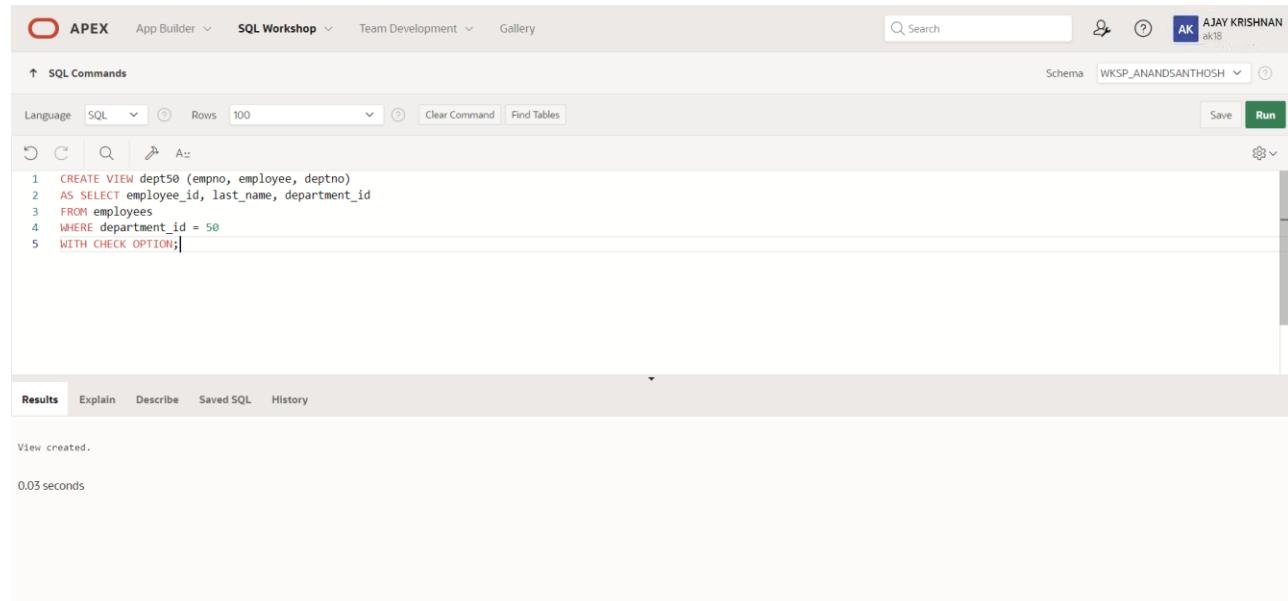
EMPLOYEE	DEPARTMENT_NUMBER
Neena Kochhar	90
Daniel Faviet	100
Valli Pataballa	60
Diana Lorentz	60
John Chen	100
David Austin	60
Lex De Haan	90
Bruce Ernst	60
Nancy Greenberg	100
Steven King	90
Alexander Hunold	60
Ismael Scott	100

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

### QUERY:

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

### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a user profile for 'AJAY KRISHNAN' and a schema dropdown set to 'WKSP\_ANANDSANTOSH'. The main area is titled 'SQL Commands' with a sub-section 'CREATE VIEW'. The code entered is:

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

Below the code, the 'Results' tab is selected, showing the message 'View created.' and a execution time of '0.03 seconds'.

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

**QUERY:**

```
DESCRIBE dept50;
```

**OUTPUT:**

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user profile, and schema dropdown set to WKSP\_ANANDSANHOSH. The main area has tabs for SQL Commands, Results, Explain, Describe (selected), Saved SQL, and History. The SQL Commands tab shows the query: 1 DESCRIBE dept50;. Below it, the Results tab displays the structure of the DEPT50 view:

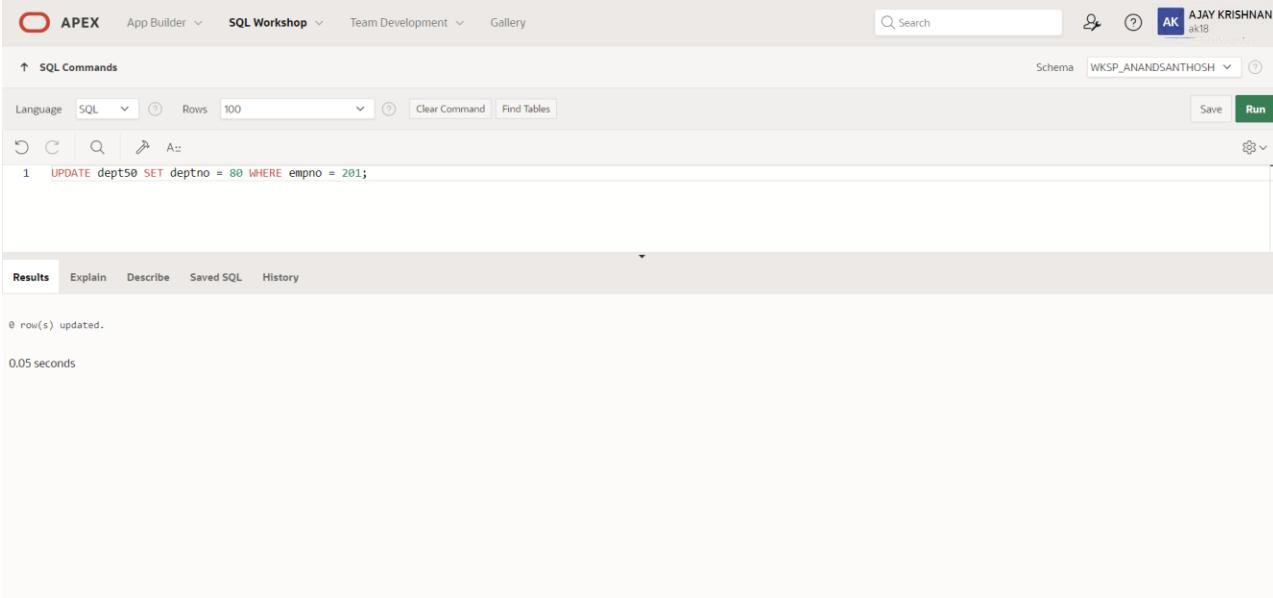
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPT50	EMPNO	NUMBER	-	6	0	-	✓	-	-
	EMPLOYEE	VARCHAR2	25	-	-	-	-	-	-
	DEPTNO	NUMBER	-	4	0	-	✓	-	-

7. Attempt to reassign Matos to department 80.

**QUERY:**

```
UPDATE dept50 SET deptno = 80 WHERE empno = 201;
```

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a user profile for 'AJAY KRISHNAN' and a schema dropdown set to 'WKSP\_ANANDSANTHOSH'. The main workspace is titled 'SQL Commands' and contains a single line of SQL code: '1 UPDATE dept50 SET deptno = 80 WHERE empno = 201;'. Below the code, the 'Results' tab is selected, showing the output: '0 row(s) updated.' and '0.05 seconds'. Other tabs available include Explain, Describe, Saved SQL, and History.

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

### **QUERY:**

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

### **OUTPUT:**

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

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

Below the code, the 'Results' tab is active, showing the message 'View created.' and a execution time of '0.06 seconds'.

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

## **INTRO TO CONSTRAINTS: NOT NULL AND UNIQUE CONSTRAINTS**

**EX.NO:12**

**DATE: 27 – 4 - 24**

**REG.NO: 220701018**

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

```

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

Table Created.

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

DESCRIBE f\_global\_locations;

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

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

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

### **PRIMARY KEY, FOREIGN KEY, and CHECK Constraints**

1. What is the purpose of a

- PRIMARY KEY
- FOREIGN KEY
- CHECK CONSTRAINT

a. PRIMARY KEY Uniquely identify each row in table.

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

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

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

animal\_id NUMBER(6)  
name VARCHAR2(25)  
license\_tag\_number NUMBER(10)  
admit\_date DATE  
adoption\_id NUMBER(5),  
vaccination\_date DATE

animal\_id NUMBER(6) - PRIMARY KEY

name VARCHAR2(25) license\_tag\_number NUMBER(10) - UNIQUE

admit\_date DATE -NOT NULL

adoption\_id NUMBER(5), vaccination\_date DATE -NOT NULL

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

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

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

ANIMAL_ID	NAME	LICENSE_TAG_NUMBER	ADMIT_DATE	ADOPTION_ID	VACCINATION_DATE
101	Spot	35540	10-Oct-2004	205	12-Oct-2004

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

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

COLUMN LEVEL STATEMENT: ALTER TABLE animals MODIFY ( adoption\_id NUMBER(5,0) CONSTRAINT anl\_adopt\_id\_fk REFERENCES adoptions(id) ENABLE );

TABLE LEVEL STATEMENT: ALTER TABLE animals ADD CONSTRAINT anl\_adopt\_id\_fk FOREIGN KEY (adoption\_id) REFERENCES adoptions(id) ENABLE;

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

- a. ON DELETE CASCADE
- b. ON DELETE SET NULL

a. ON DELETE CASCADE ALTER TABLE animals ADD CONSTRAINT anl\_adopt\_id\_fk FOREIGN KEY (adoption\_id) REFERENCES adoptions(id) ON DELETE CASCADE ENABLE ;

b. ON DELETE SET NULL ALTER TABLE animals ADD CONSTRAINT anl\_adopt\_id\_fk FOREIGN KEY (adoption\_id) REFERENCES adoptions(id) ON DELETE SET NULL ENABLE ;

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

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

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

# CREATING VIEWS

EX.NO:13

DATE: 27 – 4 - 24

REG.NO: 220701018

1. What are three uses for a view from a DBA's perspective?

- Restrict access and display selective columns
- Reduce complexity of queries from other internal systems. So, providing a way to view same data in a different manner.
- Let the app code rely on views and allow the internal implementation of tables to be modified later

2. Create a simple view called view\_d\_songs that contains the ID, title and artist from the DJs on Demand table for each "New Age" type code. In the subquery, use the alias "Song Title" for the title column.

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

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

Results	Explain	Describe	Saved SQL	History
ID	Song Title		ARTIST	
47	Hurrah for Today		The Jubilant Trio	
49	Lets Celebrate		The Celebrants	
2 rows returned in 0.00 seconds		<a href="#">Download</a>		

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. Use the same syntax but change table\_name of the other tables.

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

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

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

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

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

```
SELECT * FROM view_copy_d_songs;
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

DELETE, INSERT, MODIFY restricted if it contains:

Group functions

GROUP BY CLAUSE

DISTINCT

pseudocolumn ROWNUM Keyword

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

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

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

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

## Managing Views

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

```
CREATE OR REPLACE VIEW view_copy_d_songs ASSELECT title, artistFROM copy_d_songs;SELECT * FROM view_copy_d_songs;
```

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

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

ORA-00942: table or view does not exist

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

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

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

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

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

```
SELECT ROWNUM, last_name, 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	

# OTHER DATABASE OBJECTS

EX.NO:14

DATE: 07 – 5 - 24

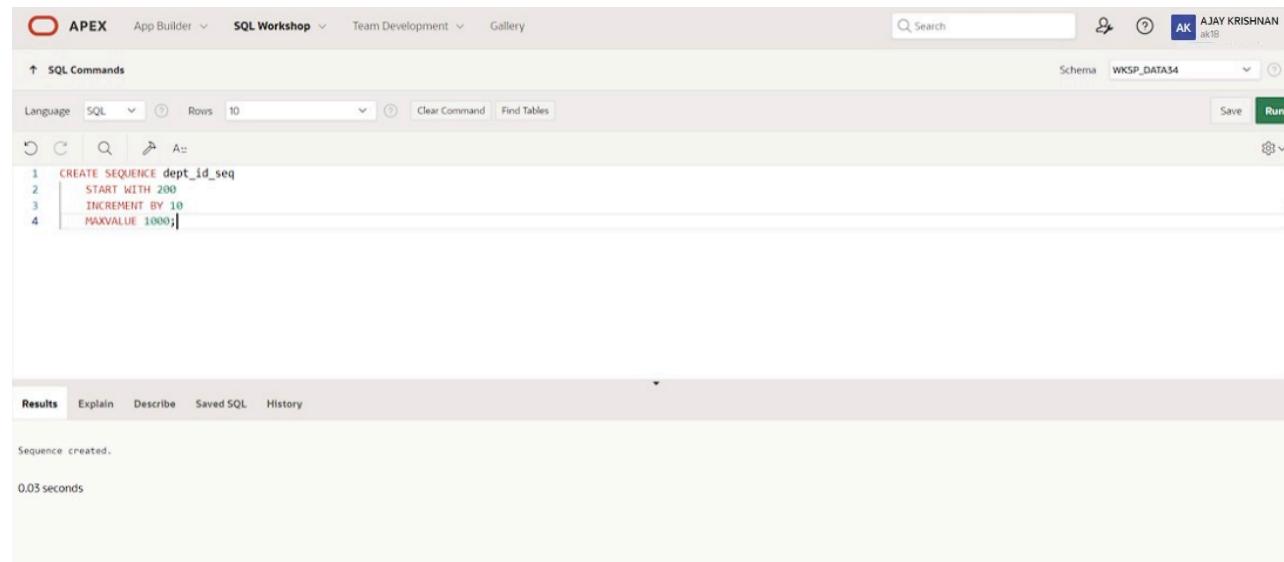
REG.NO: 220701018

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

**QUERY:**

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

**OUTPUT:**



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

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

Below the code, the results show:

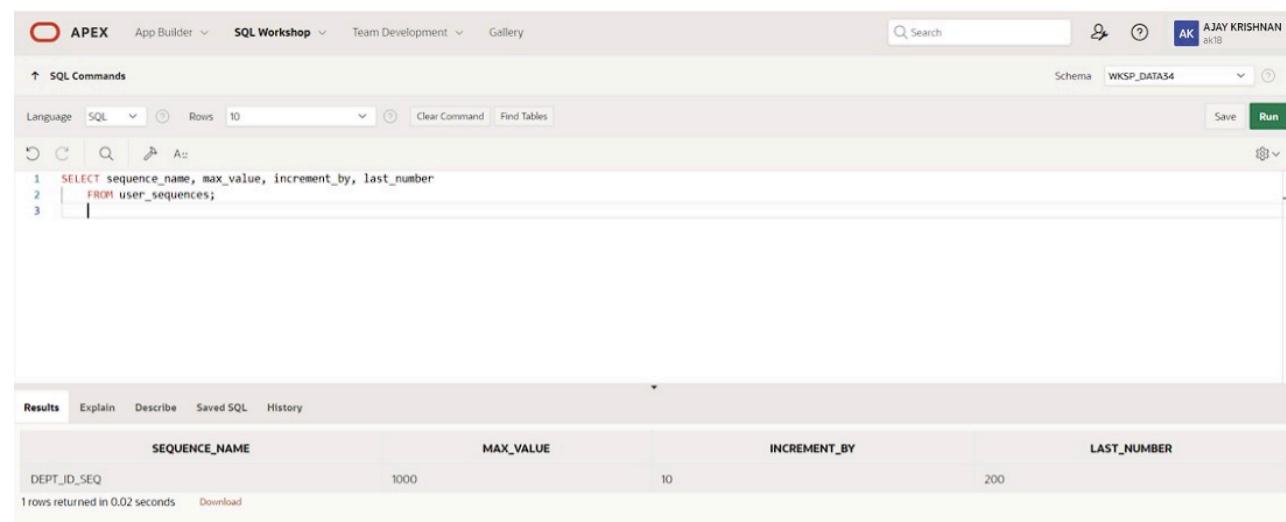
Sequence created.  
0.05 seconds

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

### QUERY:

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

### OUTPUT:



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

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

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

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".

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 APEX SQL Workshop interface. The top navigation bar includes links for App Builder, SQL Workshop, Team Development, and Gallery. The user is signed in as AJAY KRISHNAN (ak18). The main area displays the results of a script execution. The script details are as follows:

Script	exercise14	Status	Complete
View	<input type="radio"/> Detail	<input checked="" type="radio"/> Summary	<input type="radio"/>
Rows	15		
		Go	

The summary table shows the execution results:

Number	Elapsed	Statement	Feedback	Rows
1	0.05	INSERT INTO dept VALUES (dept_id_seq.nextval, 'Education')	1 row(s) inserted.	1

Below the summary, the following metrics are displayed:

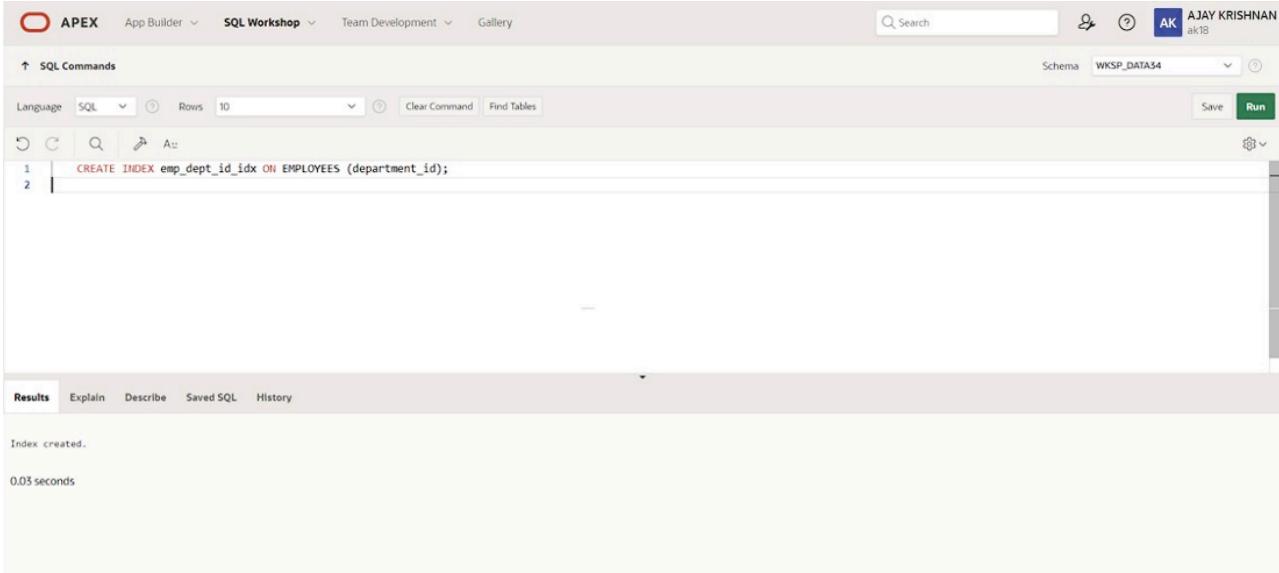
- Statements Processed: 1
- Successful: 1
- With Errors: 0

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected, followed by 'App Builder', 'SQL Workshop' (which is highlighted in blue), 'Team Development', and 'Gallery'. On the right side, there is a user profile for 'AJAY KRISHNAN ak18'. The main area is titled 'SQL Commands'. Under 'Language', 'SQL' is chosen. The 'Rows' dropdown is set to 10. Below the input field, the command is written:

```
1 | CREATE INDEX emp_dept_id_idx ON EMPLOYEES (department_id);
2 |
```

At the bottom of the SQL pane, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is currently active. The output section displays the message 'Index created.' and '0.05 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 SQL Commands pane, the following query is entered:

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

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

INDEX_NAME	TABLE_NAME	UNIQUENESS
EMP_DEPT_ID_IDX	EMPLOYEES	NONUNIQUE

Below the table, it says "1 rows returned in 0.03 seconds".

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

## **CONTROLLING USER ACCESS**

**EX.NO:15**

**DATE: 10 - 5 - 24**

**REG.NO: 220701018**

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;
```

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

**PL/SQL**

# CONTROL STRUCTURES

EX.NO:16

DATE: 21 – 5 - 24

REG.NO: 220701018

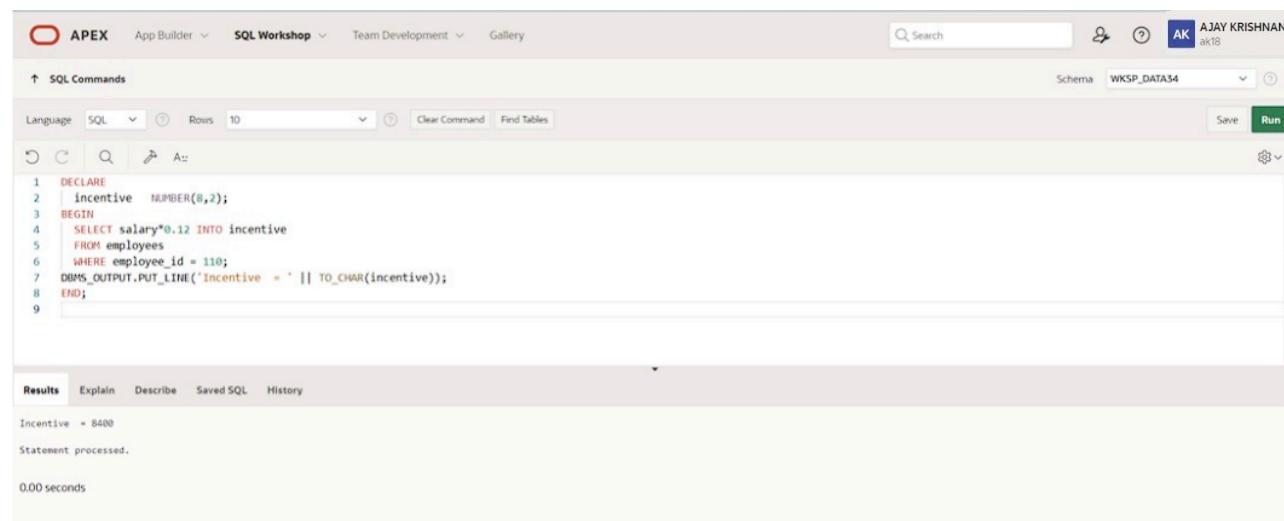
## PROGRAM 1

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

```
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;
```

The code is executed, and the results are shown in the Results tab:

```
Incentive = 8400
Statement processed.

0.00 seconds
```

## PROGRAM 2

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

## QUERY:

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

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

# OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected, followed by 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. On the right side, the user 'AJAY KRISHNAN' is logged in with the schema 'WKSP\_DATA34'. The main area is titled 'SQL Commands' with a sub-section 'Language: SQL'. A code editor window contains the following PL/SQL block:

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

Below the code, the 'Results' tab is active. An error message is displayed in a yellow box:

```
Error at line 4/25: ORA-06550: line 4, column 25:
PLS-00201: identifier 'Welcome' must be declared
ORA-06512: at "SYS.IMPV_DBMS_SQL_APEX_230200", line 881
ORA-06550: line 4, column 3:
PL/SQL: Statement ignored
```

The code block is also partially visible in the results area:

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

This screenshot is identical to the one above, showing the same PL/SQL code, error message, and results in the Oracle SQL Workshop interface.

### PROGRAM 3

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

Sample table: employees

#### 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 links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side of the header shows the user's name, AJAY KRISHNAN, and their schema, WKSP\_DATA34. The main area is titled "SQL Commands". The code editor contains the following PL/SQL procedure:

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

Below the code, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected. The output pane displays the following log messages:

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

At the bottom left, it says "0.00 seconds".

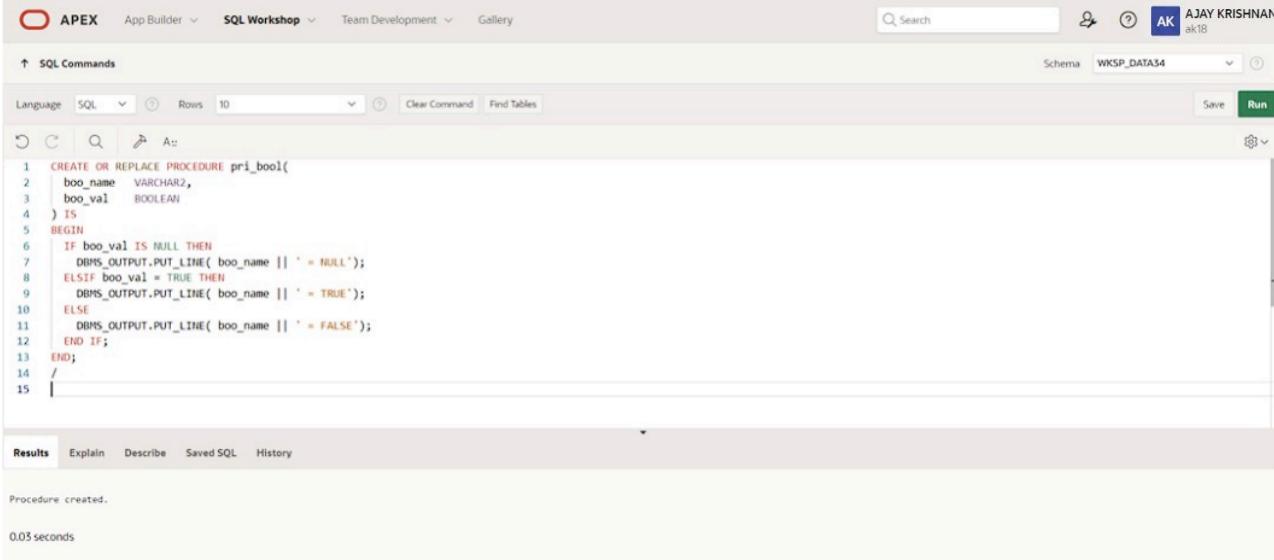
## PROGRAM 4

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

### QUERY:

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

### OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a user profile for AJAY KRISHNAN (ak18). The main workspace is titled 'SQL Commands'. It contains a code editor with the PL/SQL procedure definition. Below the code editor are tabs for Results, Explain, Describe, Saved SQL, and History. The results tab shows the output: 'Procedure created.' and '0.05 seconds'.

```
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;
/
```

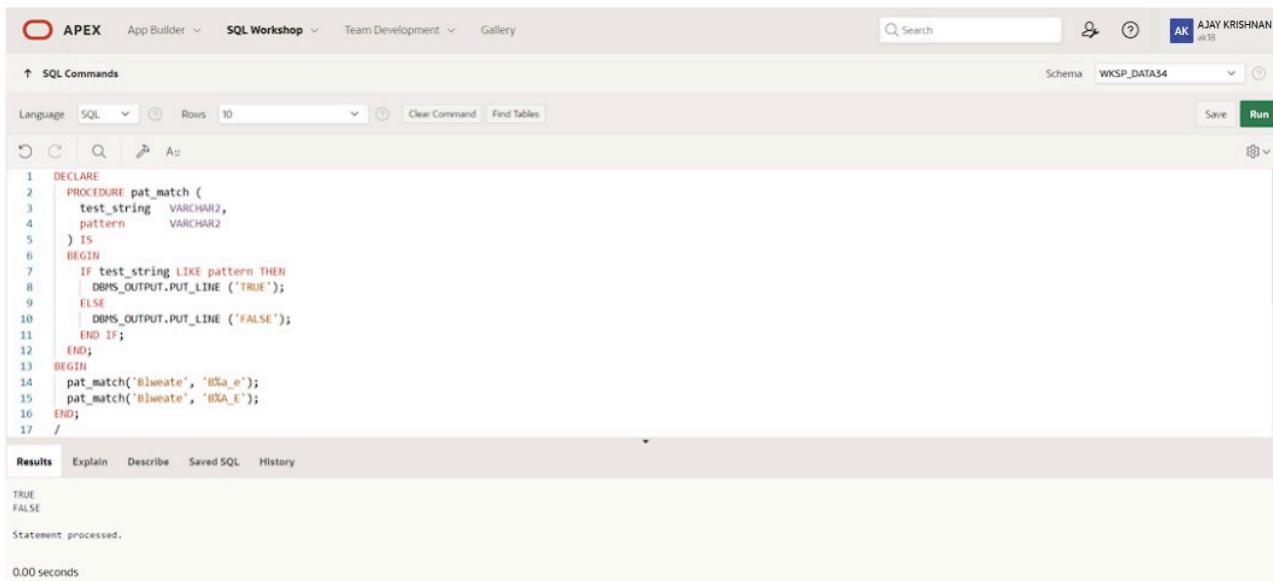
## PROGRAM 5

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 user icon for 'AJAY KRISHNAN' and a 'Run' button. The main area is titled 'SQL Commands' with a 'Schema' dropdown set to 'WKSP\_DATA34'. The code editor contains the provided PL/SQL block. Below the code, the 'Results' tab is selected, showing the output: 'TRUE' and 'FALSE'. The status bar at the bottom indicates '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

## PROGRAM 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.

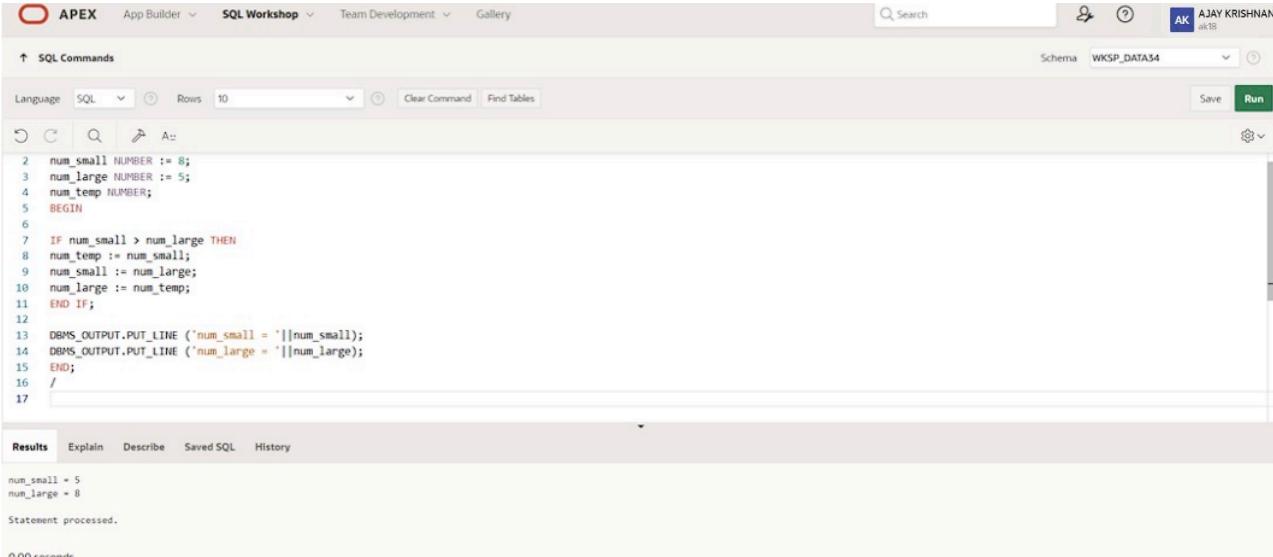
### QUERY:

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. On the right, there's a user profile for AJAY KRISHNAN and a schema dropdown set to WKSP\_DATA54. The main area is titled "SQL Commands". The code editor contains a PL/SQL block with numbered lines from 1 to 17. Lines 13 and 14 show the output of DBMS\_OUTPUT.PUT\_LINE statements. The results pane at the bottom displays the output: "num\_small = 5" and "num\_large = 8", followed by the message "Statement processed." and a timestamp 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

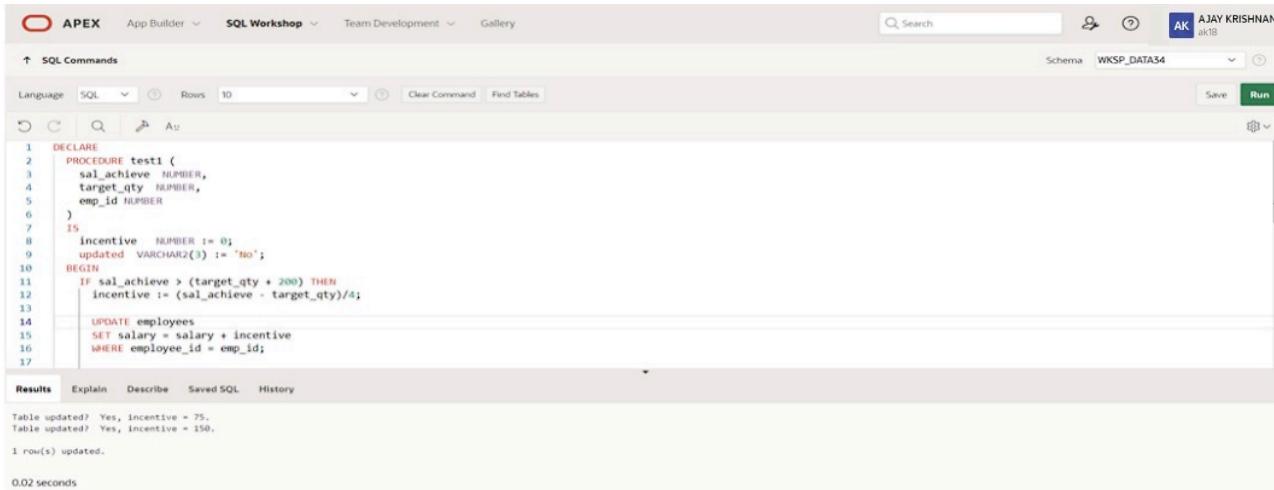
## PROGRAM 7

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 links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is signed in as AJAY KRISHNAN (ak18). The schema selected is WKSP\_DATA34. The main area displays a PL/SQL procedure named test1. The code is as follows:

```
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 VARCHAR(3) := 'No';
10     BEGIN
11         IF sal_achieve > (target_qty * 200) THEN
12             incentive := (sal_achieve - target_qty)/4;
13         END IF;
14         UPDATE employees
15             SET salary = salary + incentive
16           WHERE employee_id = emp_id;
17     END;
```

Below the code, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is active, showing the output of the executed query:

```
Table updated? Yes, incentive = 75.
Table updated? Yes, incentive = 150.
1 row(s) updated.
```

Execution time is listed as 0.02 seconds.

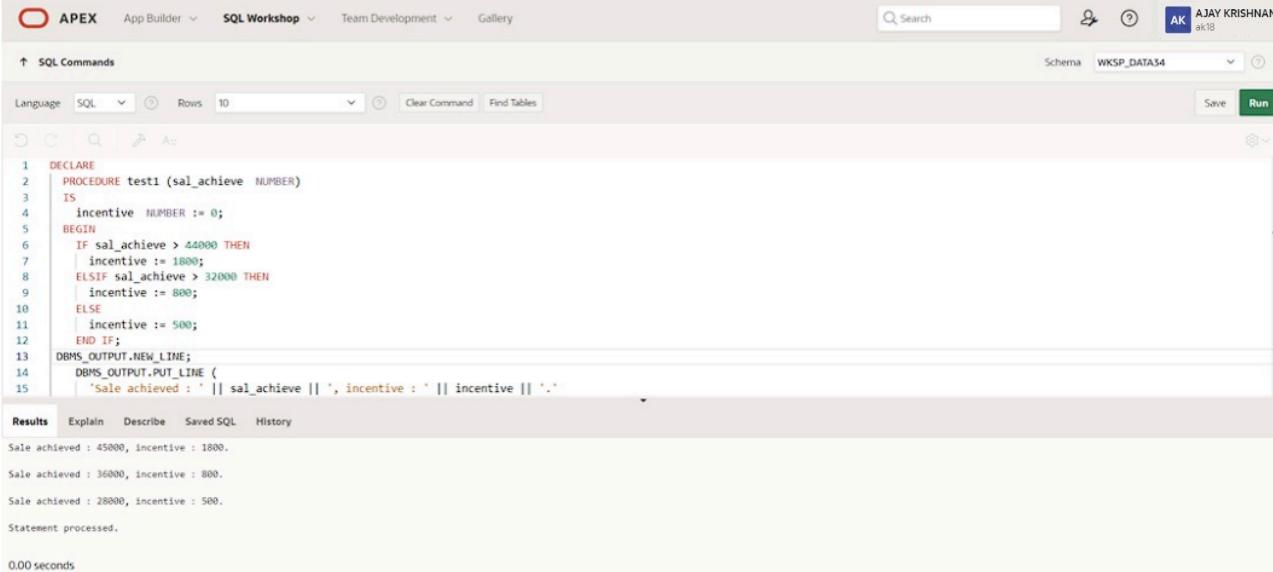
## PROGRAM 8

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

### QUERY:

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

## OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon for 'AJAY KRISHNAN ak18', and a 'Run' button. The main area is titled 'SQL Commands'. The code editor contains the following PL/SQL procedure:

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

The results tab is selected, showing the output of the procedure execution:

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

At the bottom, it says '0.00 seconds'.

## PROGRAM 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. At the top, there are tabs for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a user profile for AJAY KRISHNAN (ak18) with a search bar and various icons. The main area is titled "SQL Commands". Below it, there are buttons for Language (SQL selected), Rows (10), Clear Command, Find Tables, Save, and Run. The code editor contains the following PL/SQL script:

```
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

```

The "Results" tab is selected, showing the output of the script:

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

At the bottom left, it says "0.00 seconds".

## PROGRAM 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:

```
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. At the top, there are tabs for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, it shows the user's profile (AJAY KRISHNAN, ak18) and the schema (WKSP\_DATA34). The main area is titled "SQL Commands". The code entered is a PL/SQL block:

```
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));
```

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

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

At the bottom, it says "0.05 seconds".

## PROGRAM 11

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

### QUERY:

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

## OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. 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 profile for AJAY KRISHNAN, and a schema dropdown set to WKSP\_DATAS4. Below the tabs, there are buttons for Save and Run.

The main area is titled "SQL Commands". It contains the following PL/SQL code:

```
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;
```

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

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	4800

At the bottom left, it says "Statement processed."

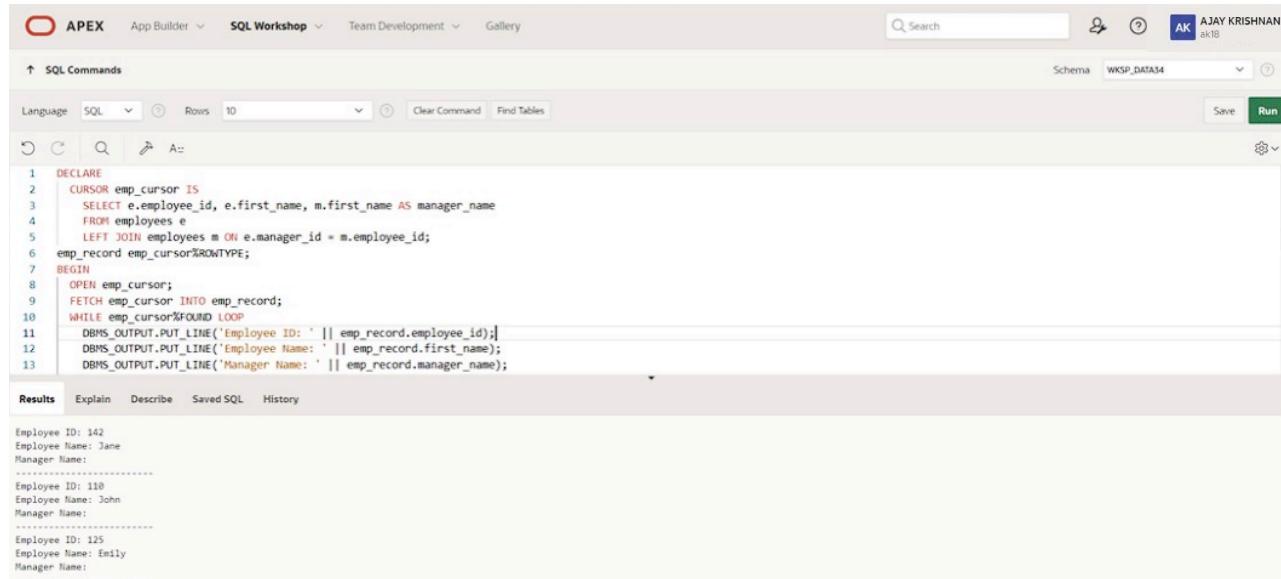
## PROGRAM 12

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'. On the right, there's a user profile for 'AJAY KRISHNAN ak18' and a search bar. The main area is titled 'SQL Commands' with a 'Run' button. Below the title are filters for 'Language: SQL', 'Rows: 10', and buttons for 'Clear Command' and 'Find Tables'. The code editor contains the following PL/SQL block:

```
1  DECLARE
2      CURSOR emp_cursor IS
3          SELECT e.employee_id, e.first_name, m.first_name AS manager_name
4          FROM employees e
5              LEFT JOIN employees m ON e.manager_id = m.employee_id;
6      emp_record emp_cursor%ROWTYPE;
7  BEGIN
8      OPEN emp_cursor;
9      FETCH emp_cursor INTO emp_record;
10     WHILE emp_cursor%FOUND LOOP
11         DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp_record.employee_id);
12         DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_record.first_name);
13         DBMS_OUTPUT.PUT_LINE('Manager Name: ' || emp_record.manager_name);
14     END LOOP;
15  END;
```

The results pane shows the output of the PL/SQL block:

```
Employee ID: 142
Employee Name: Jane
Manager Name:
-----
Employee ID: 118
Employee Name: John
Manager Name:
-----
Employee ID: 125
Employee Name: Emily
Manager Name:
-----
```

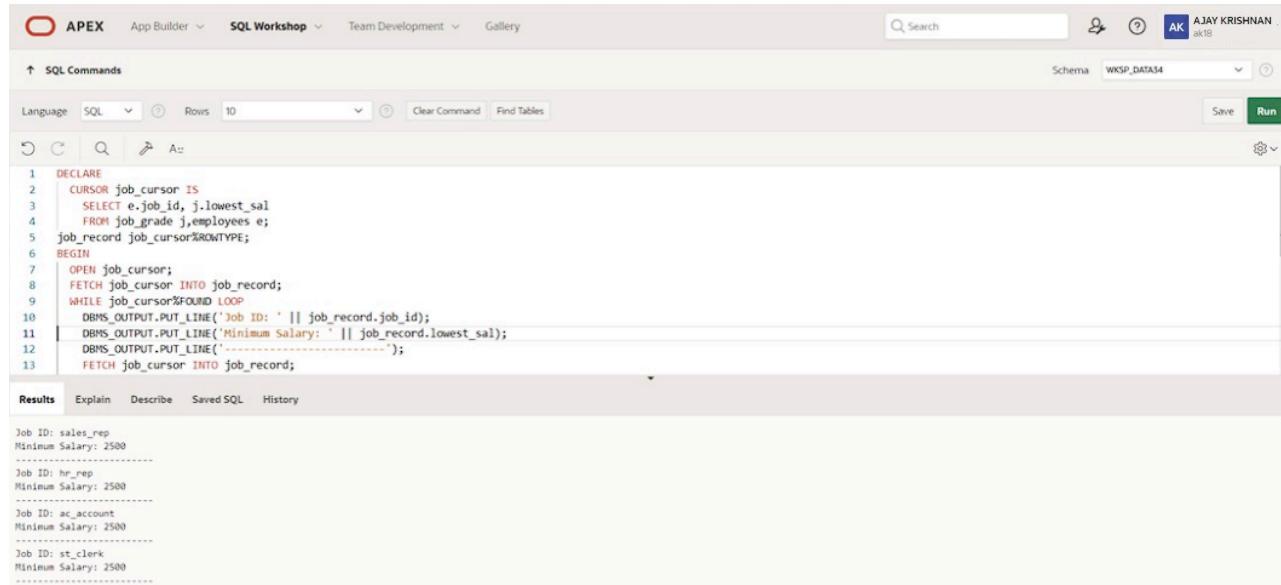
## PROGRAM 13

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. The right side shows the user's profile (AJAY KRISHNAN, ak18) and the schema (WKSP\_DATA34). The main area is titled "SQL Commands" and contains the following PL/SQL code:

```
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;

```

The "Results" tab is selected, displaying the output of the code:

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

## PROGRAM 15

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

### QUERY:

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

## OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user profile AK AJAY KRISHNAN ak18 and the schema WKSP\_DATA34. The main area is titled "SQL Commands" with a language dropdown set to SQL. The code editor contains the following PL/SQL block:

```
1 DECLARE
2     CURSOR employees_cur IS
3         SELECT employee_id,
4                last_name,
5                job_id,
6                start_date
7           FROM employees
8      NATURAL JOIN job_history;
9    emp_start_date DATE;
10 BEGIN
11     dbms_output.Put_line(Rpad('Employee ID', 15)
12                           ||Rpad('Last Name', 25)
13                           ||Rpad('Job Id', 35)
14                           ||'Start Date'));
15
16     dbms_output.Put_line('-----');
17 
```

The results tab displays the output of the PL/SQL block:

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

Statement processed.

Evaluation Procedure	Marks awarded
<b>PL/SQL Procedure(5)</b>	
<b>Program/Execution (5)</b>	
<b>Viva(5)</b>	
<b>Total (15)</b>	
<b>Faculty Signature</b>	

# **PROCEDURES AND FUNCTIONS**

**EX.NO:17**

**DATE: 21 - 5 - 24**

**REG.NO: 220701018**

---

## **Program 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. In the top navigation bar, 'APEX' is selected, followed by 'App Builder', 'SQL Workshop' (which is currently active), 'Team Development', and 'Gallery'. On the right side, the user 'AJAY KRISHNAN' (ak18) is logged in, and the schema 'WKSP\_DATA54' is selected. The main area is titled 'SQL Commands' and contains the following PL/SQL code:

```
1  DECLARE
2      fac NUMBER := 1;
3      n NUMBER := 1; -- Use a bind variable instead of "&1"
4  BEGIN
5      WHILE n > 0 LOOP
6          fac := n * fac;
7          n := n - 1;
8      END LOOP;
9      DBMS_OUTPUT.PUT_LINE(fac);
10 END;
```

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

120  
Statement processed.  
0.00 seconds

## Program 2

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

### **QUERY:**

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

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

```

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

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

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

```

## OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The code area contains a PL/SQL block. The output pane shows the results of the execution, including the title, author, and year published.

```

APEX App Builder SQL Workshop Team Development Gallery
Search Schema WKSP_DATA34 Run
SQL Commands Language SQL Rows 10 Clear Command Find Tables
SQL Commands
1 DECLARE
2     v_book_id NUMBER := 1;
3     v_title VARCHAR2(100);
4     v_author VARCHAR2(100);
5     v_year_published NUMBER;
6 BEGIN
7     v_title := 'Initial Title';
8
9     get_book_info(p_book_id => v_book_id, p_title => v_title, p_author => v_author, p_year_published => v_year_published);
10
11    DBMS_OUTPUT.PUT_LINE('Title: ' || v_title);
12    DBMS_OUTPUT.PUT_LINE('Author: ' || v_author);
13    DBMS_OUTPUT.PUT_LINE('Year Published: ' || v_year_published);
14 END;

```

**Results**

Title: To Kill a Mockingbird - Retrieved  
 Author: Harper Lee  
 Year Published: 1960

Statement processed.  
 0.01 seconds

<b>Evaluation Procedure</b>	<b>Marks awarded</b>
<b>PL/SQL Procedure(5)</b>	
<b>Program/Execution (5)</b>	
<b>Viva(5)</b>	
<b>Total (15)</b>	
<b>Faculty Signature</b>	

# **TRIGGER**

**EX.NO:18**

**DATE: 21 – 5 - 24**

**REG.NO: 220701018**

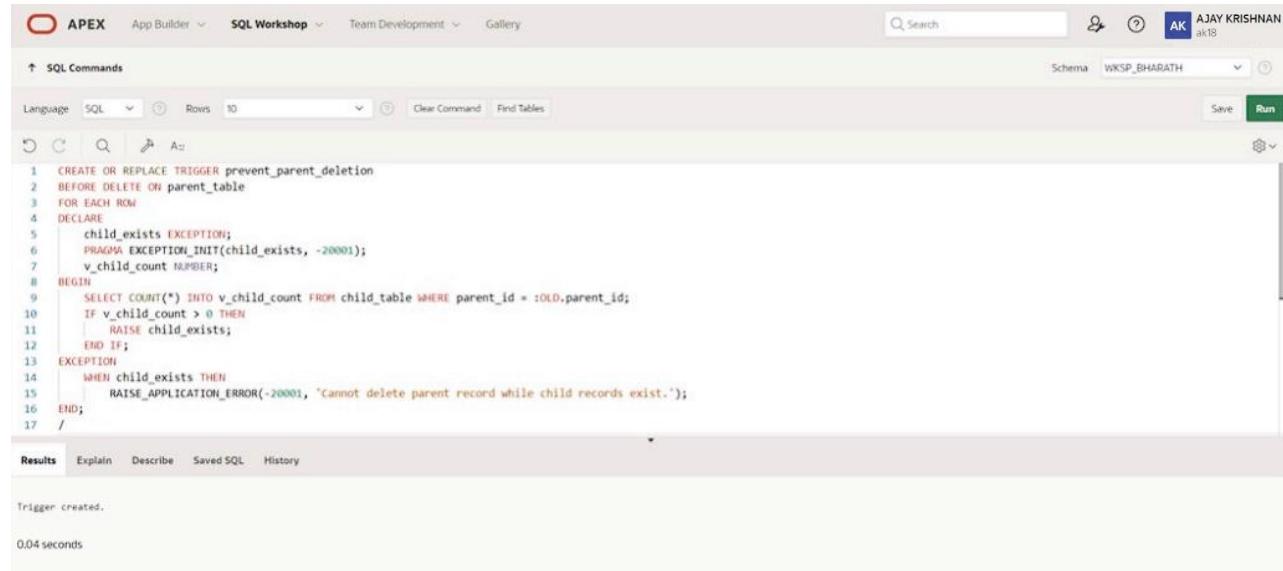
## **Program 1**

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

### **QUERY:**

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

## OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user profile (AJAY KRISHNAN, ak18), and a Run button. The main area is titled "SQL Commands" and contains the following PL/SQL code:

```
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 /
```

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

## Program 2

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

### QUERY:

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

## OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a user profile for AJAY KRISHNAN (ak18). The main area is titled "SQL Commands". The schema is set to WKSP\_DATA34. The code editor contains the following PL/SQL trigger creation script:

```
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
```

Below the code editor, the results tab is selected, showing the message "Trigger created." and a execution time of "0.04 seconds".

### Program 3

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

#### QUERY:

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

## OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side of the header shows the user's profile (AJAY KRISHNAN, ak18) and the schema (WKSP\_DATA34). The main area is titled "SQL Commands". The code editor contains the following PL/SQL trigger creation script:

```
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 EXCEPTION
16    WHEN threshold_exceeded THEN
17        RAISE_APPLICATION_ERROR(-20003, 'Threshold exceeded for value_col.');

```

Below the code editor, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, showing the output: "Trigger created." and "0.04 seconds".

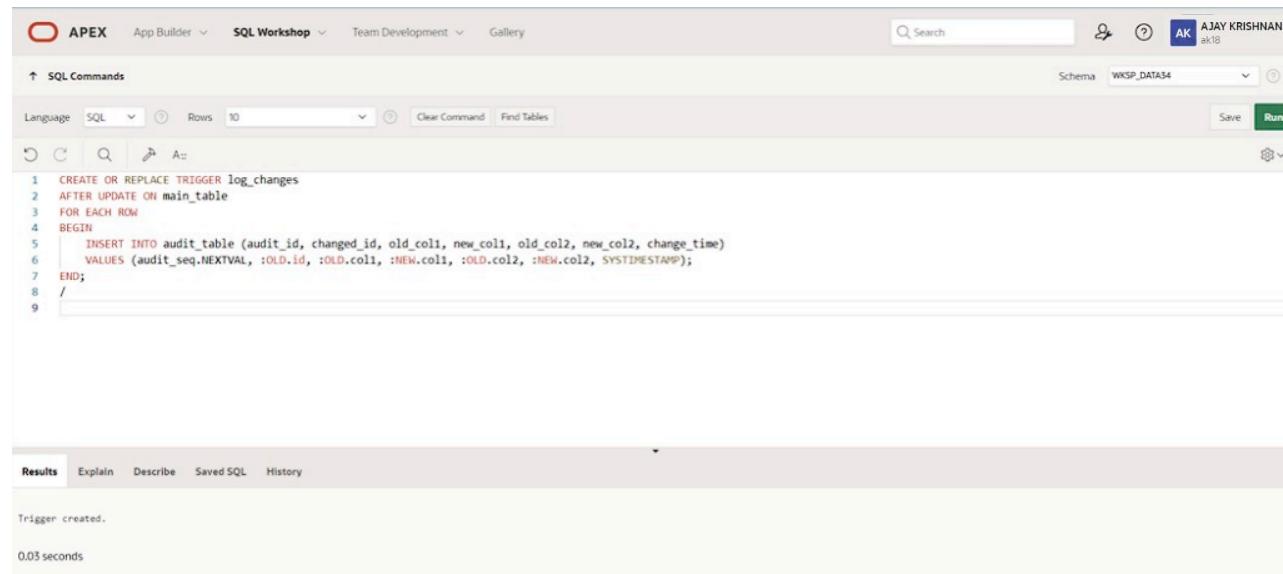
## Program 4

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

### QUERY:

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

### OUTPUT:



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

```
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;
/
```

Below the command, the 'Results' tab is active, showing the output: "Trigger created." and "0.03 seconds".

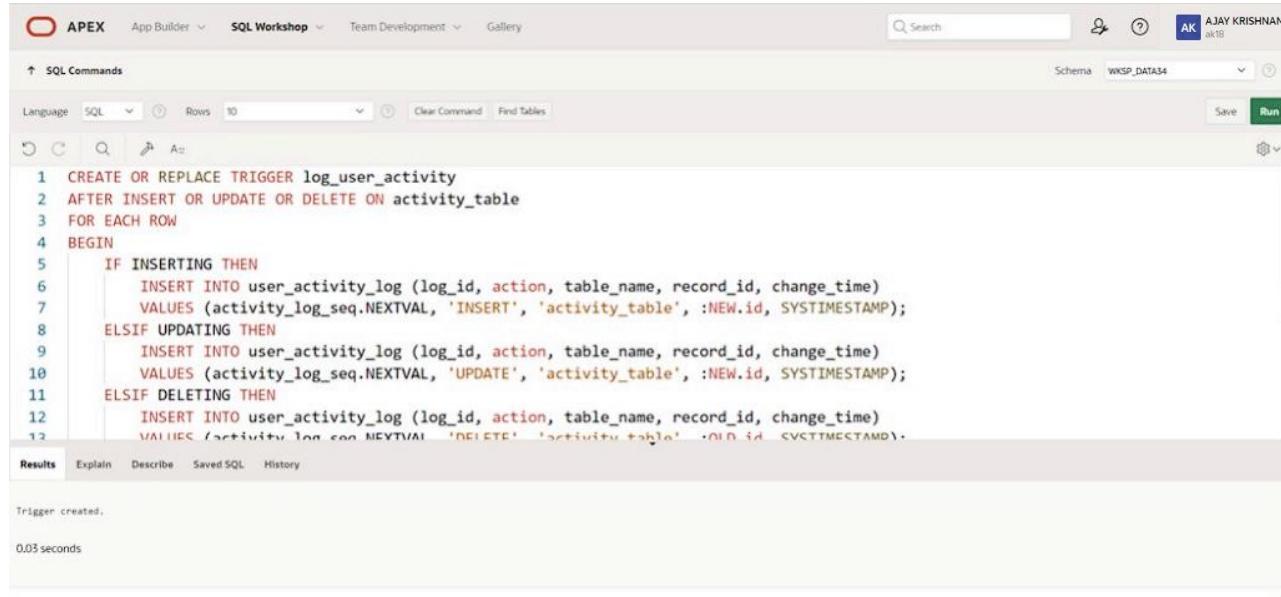
## Program 5

Write a code in PL/SQL to implement a trigger that records user activity (inserts, updates, deletes) in an audit log for a given set of tables.

### QUERY:

```
CREATE OR REPLACE TRIGGER log_user_activity
AFTER INSERT OR UPDATE OR DELETE ON activity_table
FOR EACH ROW
BEGIN
    IF INSERTING THEN
        INSERT INTO user_activity_log (log_id, action, table_name, record_id,
change_time)
        VALUES (activity_log_seq.NEXTVAL, 'INSERT', 'activity_table', :NEW.id,
SYSTIMESTAMP);
    ELSIF UPDATING THEN
        INSERT INTO user_activity_log (log_id, action, table_name, record_id,
change_time)
        VALUES (activity_log_seq.NEXTVAL, 'UPDATE', 'activity_table', :NEW.id,
SYSTIMESTAMP);
    ELSIF DELETING THEN
        INSERT INTO user_activity_log (log_id, action, table_name, record_id,
change_time)
        VALUES (activity_log_seq.NEXTVAL, 'DELETE', 'activity_table', :OLD.id,
SYSTIMESTAMP);
    END IF;
END;
```

## OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user information for AJAY KRISHNAN (ak18), and a schema dropdown set to WKSP\_DATA34. The main area is titled "SQL Commands". The code editor contains the following PL/SQL trigger definition:

```
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);

```

Below the code, the "Results" tab is selected, showing the message "Trigger created." and a execution time of "0.03 seconds".

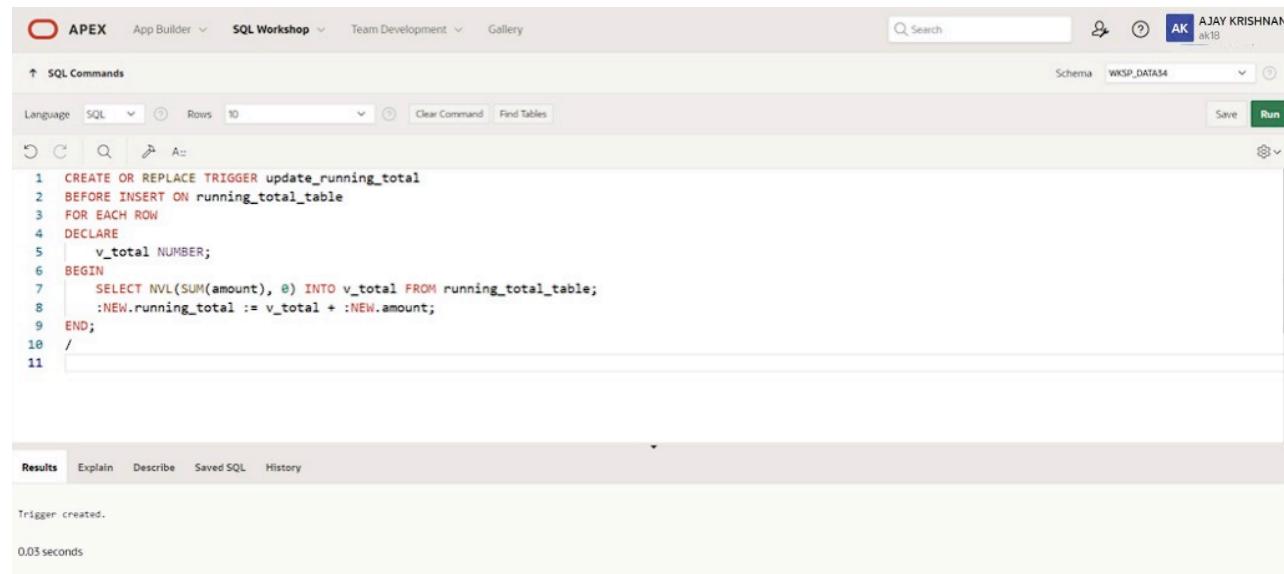
## Program 6

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

### QUERY:

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

### OUTPUT:



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

```
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
```

Below the code, the 'Results' tab is active, showing the output: "Trigger created." and "0.03 seconds".

## Program 7

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

### QUERY:

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

## OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows the user's profile (AJAY KRISHNAN, ak18) and the schema (WKSP\_DATA34). The main area is titled "SQL Commands" and contains the following PL/SQL code:

```
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
```

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

Evaluation Procedure	Marks awarded
PL/SQL Procedure(5)	
Program/Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

# MONGO DB

EX.NO:19

DATE: 24 - 5 - 24

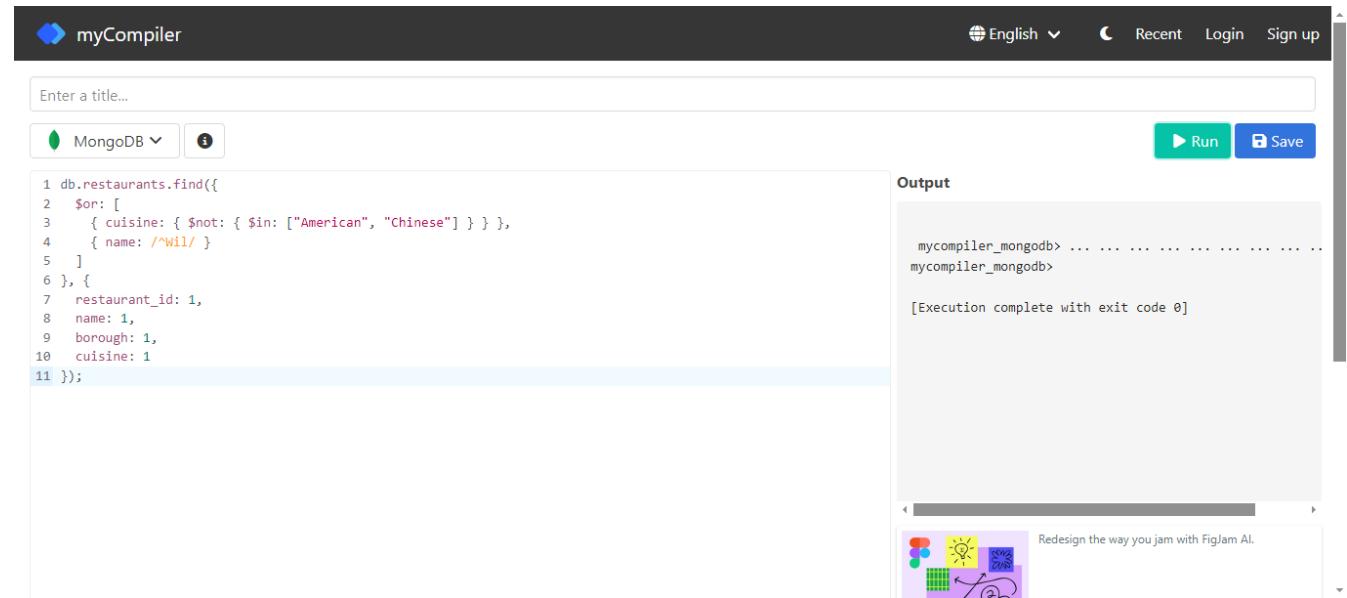
REG.NO: 220701018

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

## **QUERY:**

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

## **OUTPUT:**



The screenshot shows the myCompiler web application interface. At the top, there is a navigation bar with a logo, a search bar labeled "Enter a title...", and buttons for "English", "Recent", "Login", and "Sign up". Below the navigation bar, there is a toolbar with a MongoDB icon, a dropdown menu, and buttons for "Run" and "Save". The main area contains a code editor window with the following MongoDB query:

```
1 db.restaurants.find({  
2   $or: [  
3     { cuisine: { $not: { $in: ["American", "Chinese"] } } },  
4     { name: /^Wil/ }  
5   ],  
6 }, {  
7   restaurant_id: 1,  
8   name: 1,  
9   borough: 1,  
10  cuisine: 1  
11 });
```

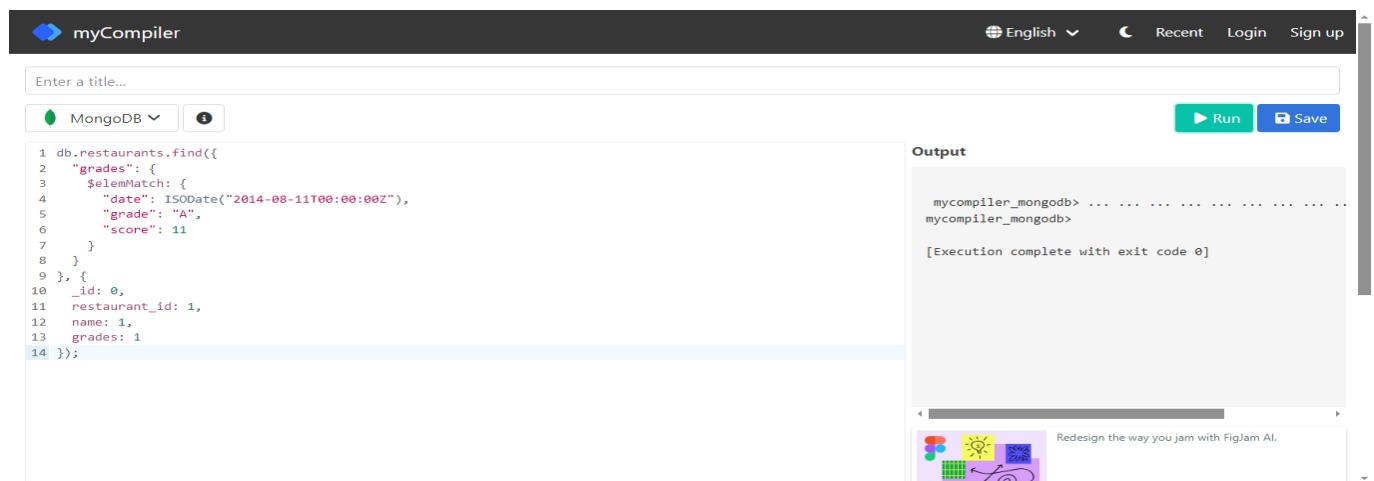
To the right of the code editor is an "Output" window. It displays the command prompt "mycompiler\_mongodb> ... . . ." followed by the message "[Execution complete with exit code 0]". At the bottom of the output window, there is a decorative footer with the text "Redesign the way you jam with FigJam AI." and some colorful icons.

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

### QUERY:

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

### OUTPUT:



The screenshot shows the myCompiler MongoDB interface. On the left, the code editor contains the MongoDB query:

```
1 db.restaurants.find({
2   "grades": {
3     "$elemMatch": {
4       "date": ISODate("2014-08-11T00:00:00Z"),
5       "grade": "A",
6       "score": 11
7     }
8   },
9 }, {
10   "_id": 0,
11   "restaurant_id": 1,
12   "name": 1,
13   "grades": 1
14 });
```

On the right, the 'Output' panel shows the results of the query execution:

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

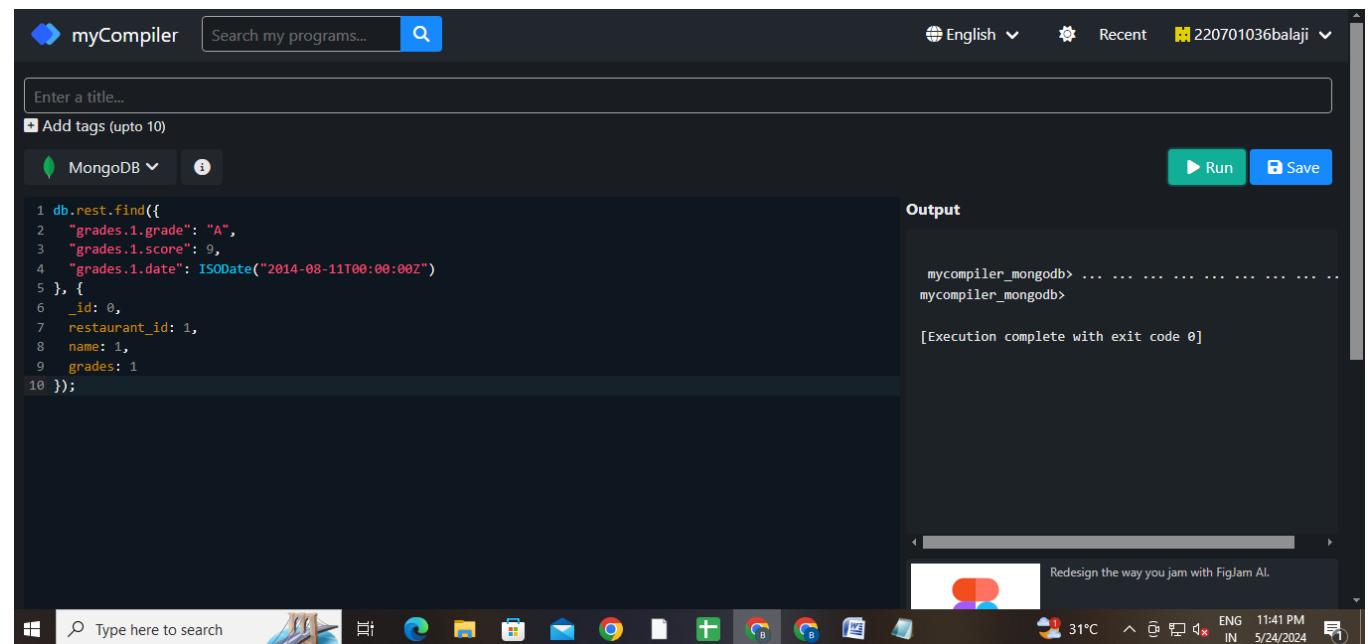
A small watermark for FigJam AI is visible at the bottom right of the interface.

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 MongoDB interface. The code input area contains the following MongoDB query:

```
1 db.restaurants.find({  
2   "grades.1.grade": "A",  
3   "grades.1.score": 9,  
4   "grades.1.date": ISODate("2014-08-11T00:00:00Z")  
5 }, {  
6   _id: 0,  
7   restaurant_id: 1,  
8   name: 1,  
9   grades: 1  
10});
```

The output window shows the results of the query execution:

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

The system tray at the bottom of the screen indicates the date and time as 5/24/2024, 11:41 PM, and the temperature as 31°C.

**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. In the code editor area, a MongoDB query is written:

```
1 db.restaurants.find({  
2   "address.coord.1": { $gt: 42, $lte: 52 }  
3 }, {  
4   _id: 0,  
5   restaurant_id: 1,  
6   name: 1,  
7   address: 1  
8 })$
```

Below the code editor is the 'Output' panel, which displays the command-line interface (CLI) logs:

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

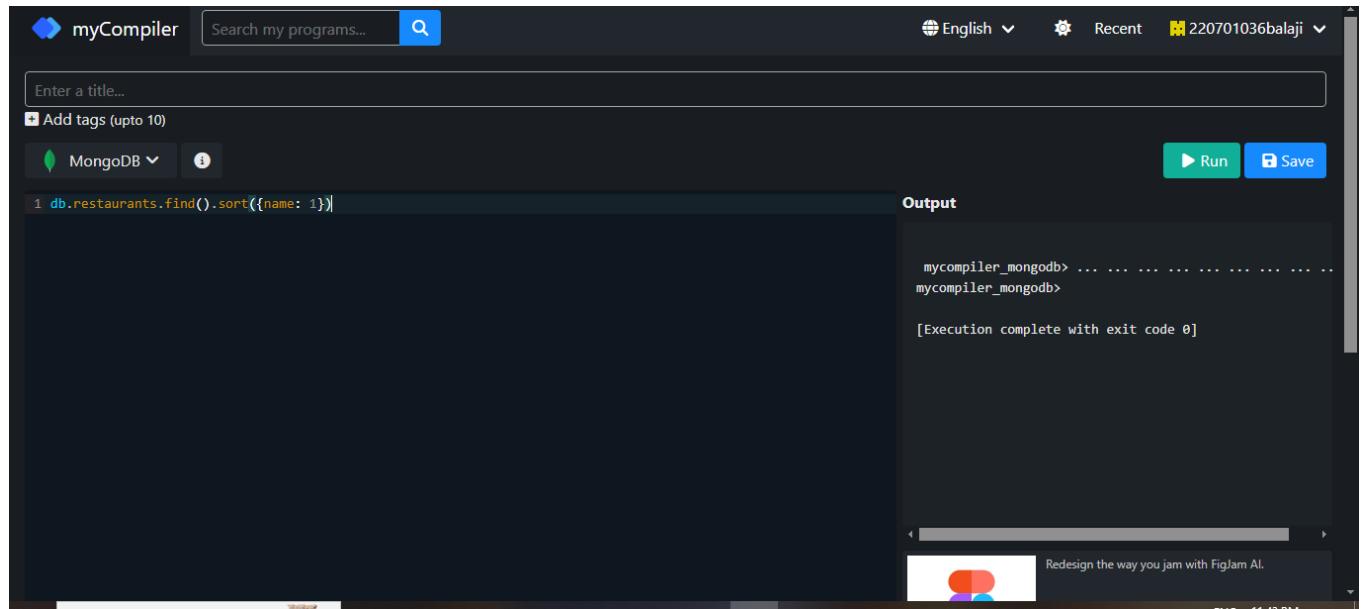
The interface includes a search bar, user profile information (English, Recent, 220701036balaji), and buttons for Run and Save.

**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 application interface. In the top bar, there is a search bar labeled "Search my programs..." and a magnifying glass icon. On the right side of the top bar, there are language settings ("English"), recent projects ("Recent"), and a user profile ("220701036balaji"). Below the top bar, there is a toolbar with a title input field ("Enter a title..."), a "Add tags (upto 10)" button, a dropdown menu set to "MongoDB", and two buttons: "Run" and "Save". The main area contains a code editor with the following MongoDB query:

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

To the right of the code editor is an "Output" panel. The output shows the command being run and its result:

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

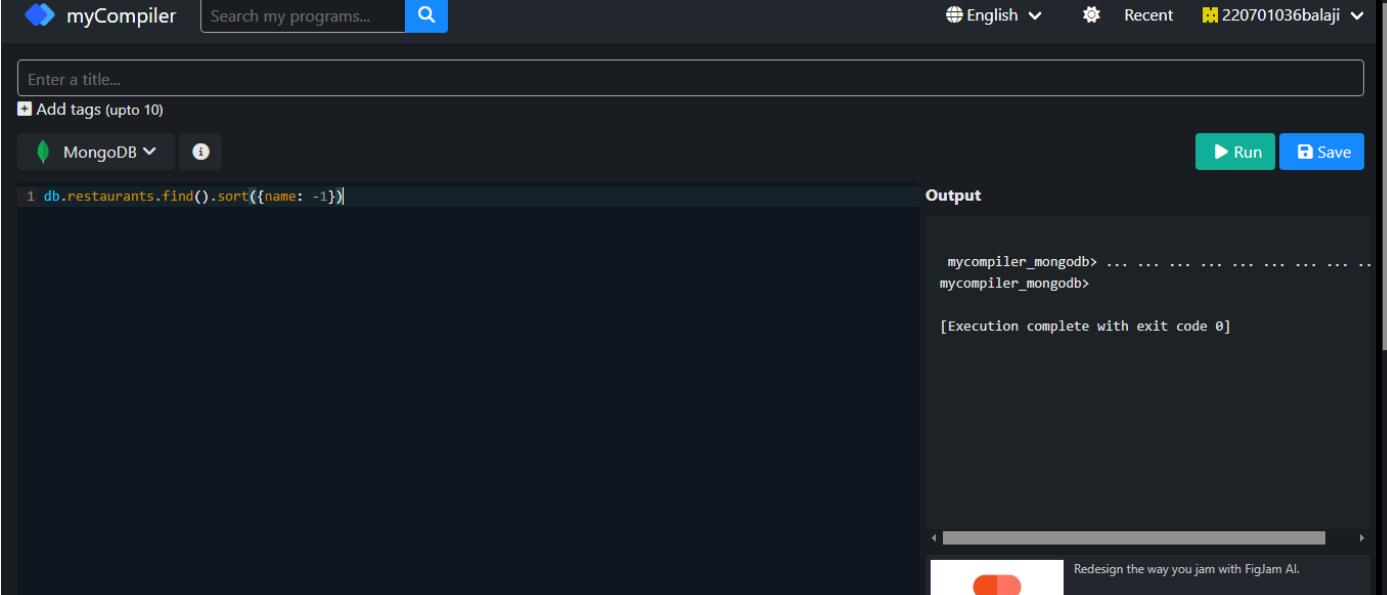
At the bottom of the screen, there is a small advertisement for FigJam AI.

**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. In the top bar, there is a search bar labeled "Search my programs..." and a magnifying glass icon. On the right side, there are language selection ("English"), settings, recent projects ("Recent"), and a user profile ("220701036balaji"). Below the search bar, there is a text input field with placeholder "Enter a title..." and a "Add tags (upto 10)" button. A dropdown menu shows "MongoDB" selected. To the right of the input fields are "Run" and "Save" buttons. The main area contains a code editor with the following content:

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

On the right side of the interface, there is an "Output" panel. It displays the command prompt "mycompiler\_mongodb> ... . . . . . . . . . . . . . . . . ." followed by "[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. At the top, there's a search bar with placeholder text "Search my programs..." and a magnifying glass icon. To the right are language selection ("English"), recent projects, and user information ("220701036balaji"). Below the header, a title input field contains "Enter a title...". A checkbox labeled "Add tags (upto 10)" is checked. The main workspace has a "MongoDB" dropdown and an info icon. On the right are "Run" and "Save" buttons. The code editor displays a single line of MongoDB query: "db.restaurants.find({}, {\_id:0, cuisine:1, borough:1}).sort({cuisine: 1, borough: -1})". To the right, an "Output" panel shows the command prompt "mycompiler\_mongodb> ... . . ." followed by "[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 title input field and a checkbox for adding tags. A MongoDB dropdown is selected. On the left, a code editor window displays the following MongoDB query:

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

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

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

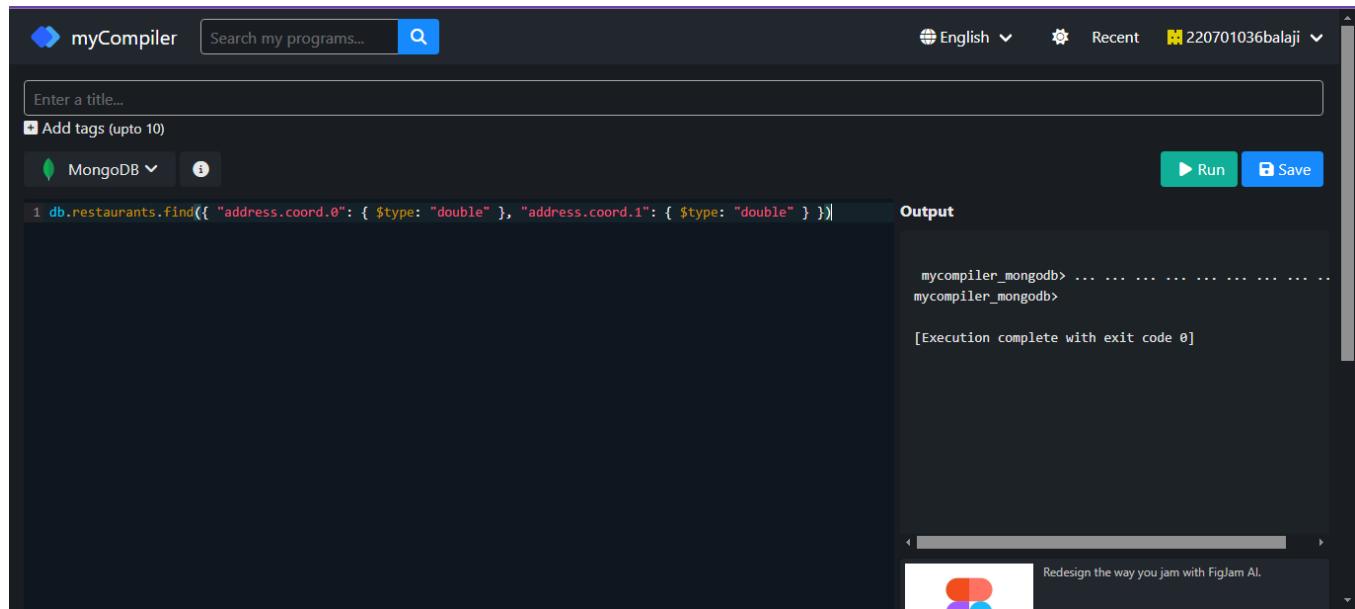
At the bottom right of the interface, there's a watermark for FigJam AI.

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. In the top left, there's a logo and the text 'myCompiler'. A search bar says 'Search my programs...' with a magnifying glass icon. On the right, there are language selection ('English'), recent projects ('Recent'), and user information ('220701036balaji'). Below the search bar is a text input field with placeholder 'Enter a title...' and a checked checkbox 'Add tags (upto 10)'. To the right of the input field are two buttons: 'Run' (green) and 'Save' (blue). Underneath the input field, a dropdown menu is open, showing 'MongoDB' and an 'info' icon. The main area contains a code editor with the following query:

```
1 db.restaurants.find({ "address.coord.0": { $type: "double" }, "address.coord.1": { $type: "double" } })
```

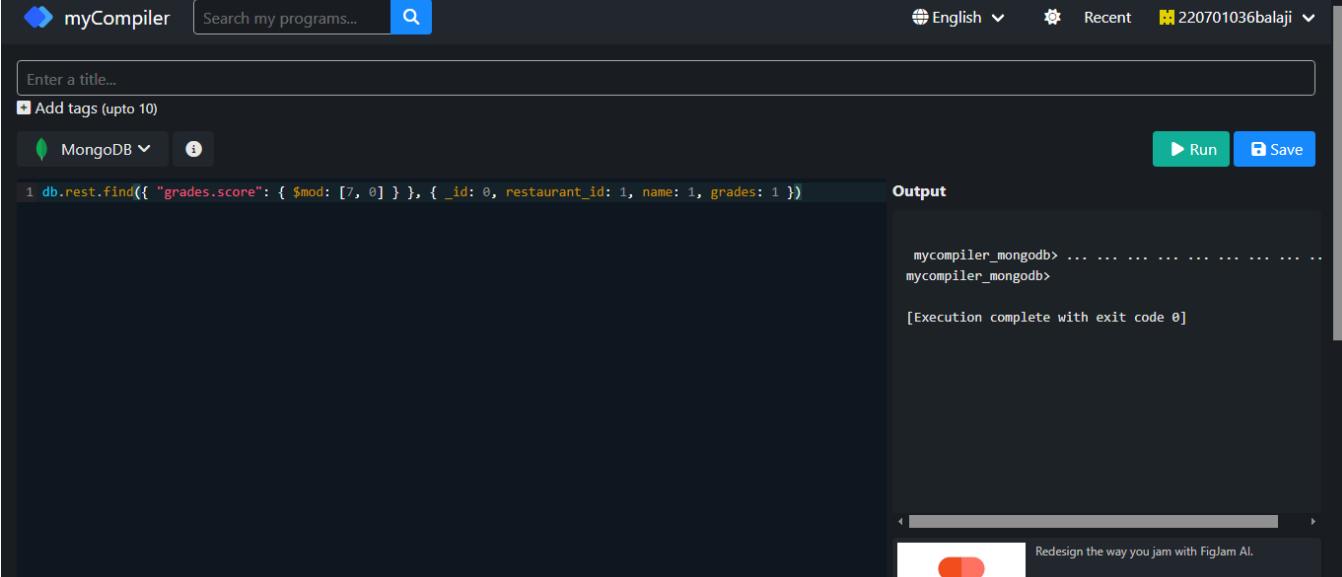
To the right of the code editor is an 'Output' panel. It shows the command prompt 'mycompiler\_mongodb> ...' followed by '[Execution complete with exit code 0]'. At the bottom of the output panel, there's a small advertisement for FigJam AI.

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 interface. In the top bar, there is a logo, a search bar with placeholder text 'Search my programs...', and a magnifying glass icon. On the right side of the top bar, there are language selection ('English'), recent projects ('Recent'), and user account ('220701036balaji') dropdowns. Below the top bar, there is a toolbar with a title input field ('Enter a title...'), a 'Add tags (upto 10)' button, a MongoDB dropdown, and two small icons. To the right of the toolbar are 'Run' and 'Save' buttons. The main area contains a code editor with the following MongoDB query:

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

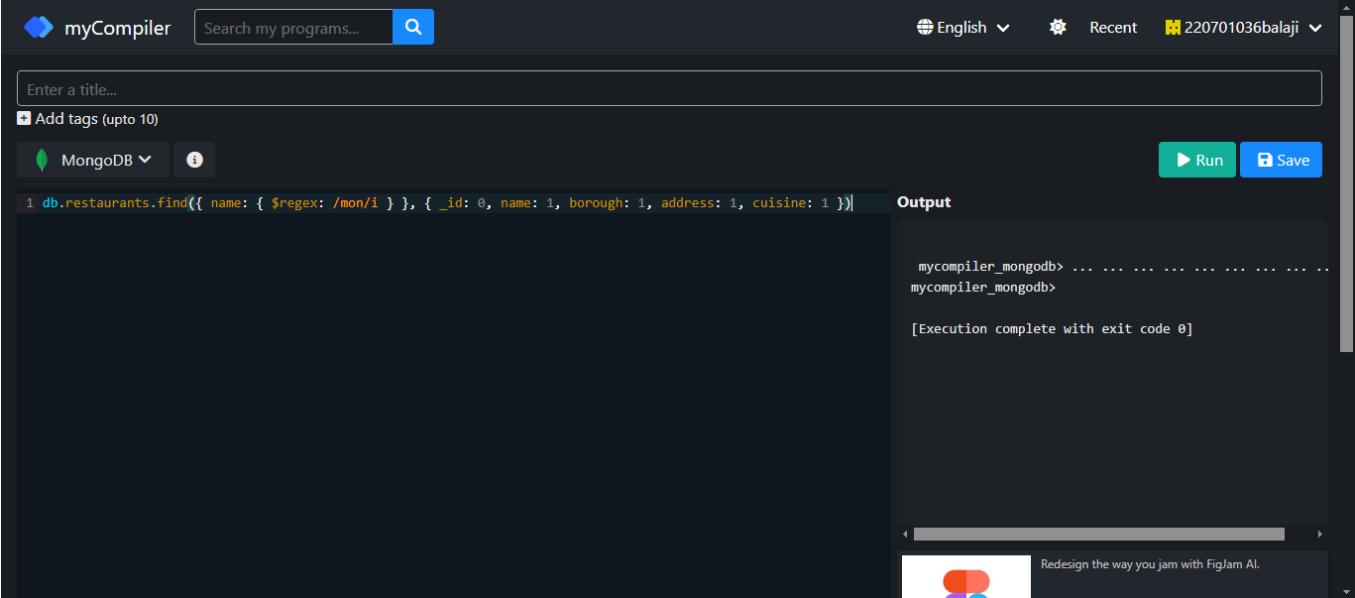
To the right of the code editor is an 'Output' panel. It displays the command prompt 'mycompiler\_mongodb> ...' followed by the output of the query: 'mycompiler\_mongodb>'. Below the output, a message says '[Execution complete with exit code 0]'. At the bottom of the interface, there is a watermark for FigJam AI.

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 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 text input field for titles and a checkbox for adding tags. A MongoDB icon is selected in the dropdown menu. On the right side, there are 'Run' and 'Save' buttons. The main area contains a code editor with the following MongoDB query:

```
1 db.restaurants.find({ name: { $regex: /mon/i } }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 })|
```

Below the code editor is an 'Output' panel. It shows the command being run: `mycompiler_mongodb> db.restaurants.find({ name: { $regex: /mon/i } }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 })`. The output shows the command was successful with an exit code of 0. A watermark for FigJam AI is visible at the bottom right of the interface.

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 interface. At the top, there's a search bar with the placeholder "Search my programs..." and a magnifying glass icon. To the right are language selection ("English"), recent projects, and user information ("220701036balaji"). Below the header, there's a text input field with placeholder "Enter a title..." and a button to "Add tags (upto 10)". On the left, there's a dropdown for "MongoDB" and a help icon. The main area contains a code editor with the following MongoDB query:

```
1 db.restaurants.find({ name: { $regex: /^Mad/i } }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 })
```

To the right of the code editor is a "Run" button and a "Save" button. Further down, the output window displays the results of the query execution:

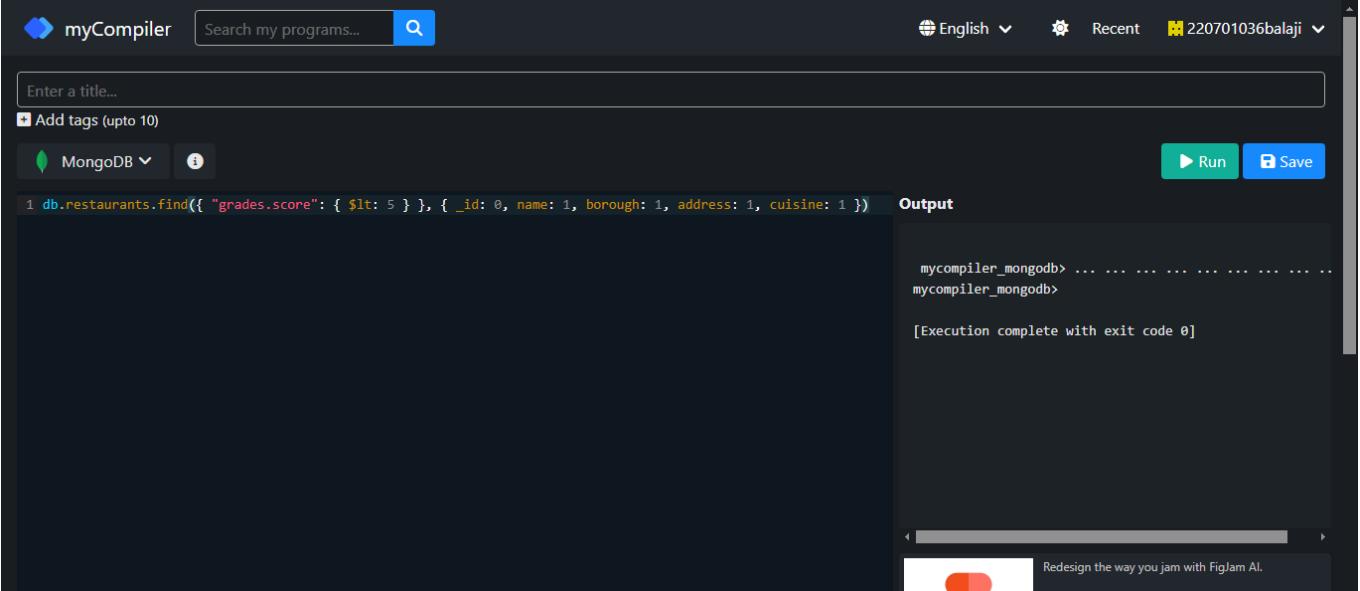
```
mycompiler_mongodb> ... ... ... ... ... ... ...  
[Execution complete with exit code 0]
```

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 interface. In the top left, there's a logo and the text "myCompiler". A search bar says "Search my programs..." with a magnifying glass icon. On the right, there are language selection ("English"), recent projects ("Recent"), and user information ("220701036balaji"). Below the search bar is a text input field with placeholder "Enter a title..." and a "Add tags (upto 10)" button. To the right of the input field are "MongoDB" and "Run" buttons. The main area contains a code editor with the following query:

```
1 db.restaurants.find({ "grades.score": { $lt: 5 } }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 })
```

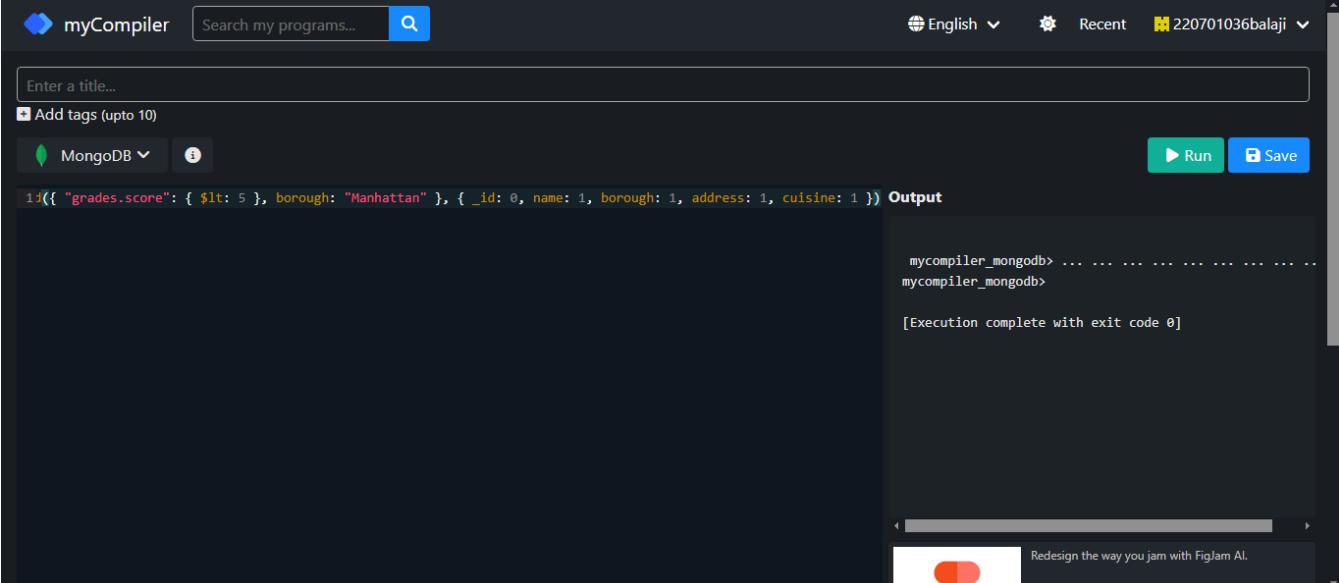
To the right of the code editor is an "Output" panel. It shows the command prompt "mycompiler\_mongodb> ... . . ." followed by "mycompiler\_mongodb>". Below that is the message "[Execution complete with exit code 0]". At the bottom of the interface, there's a watermark for FigJam AI.

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 interface. At the top, there's a search bar for programs and a language dropdown set to MongoDB. Below the search bar is a text input field with placeholder text 'Enter a title...' and a 'Add tags (upto 10)' button. To the right of the input field are 'Run' and 'Save' buttons. The main area contains a code editor with the following MongoDB query:

```
1d({ "grades.score": { $lt: 5 }, borough: "Manhattan" }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 })
```

Below the code editor is an 'Output' panel. The output shows the command being run:

```
mycompiler_mongodb> 1d({ "grades.score": { $lt: 5 }, borough: "Manhattan" }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 })
```

and the response from the MongoDB shell:

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

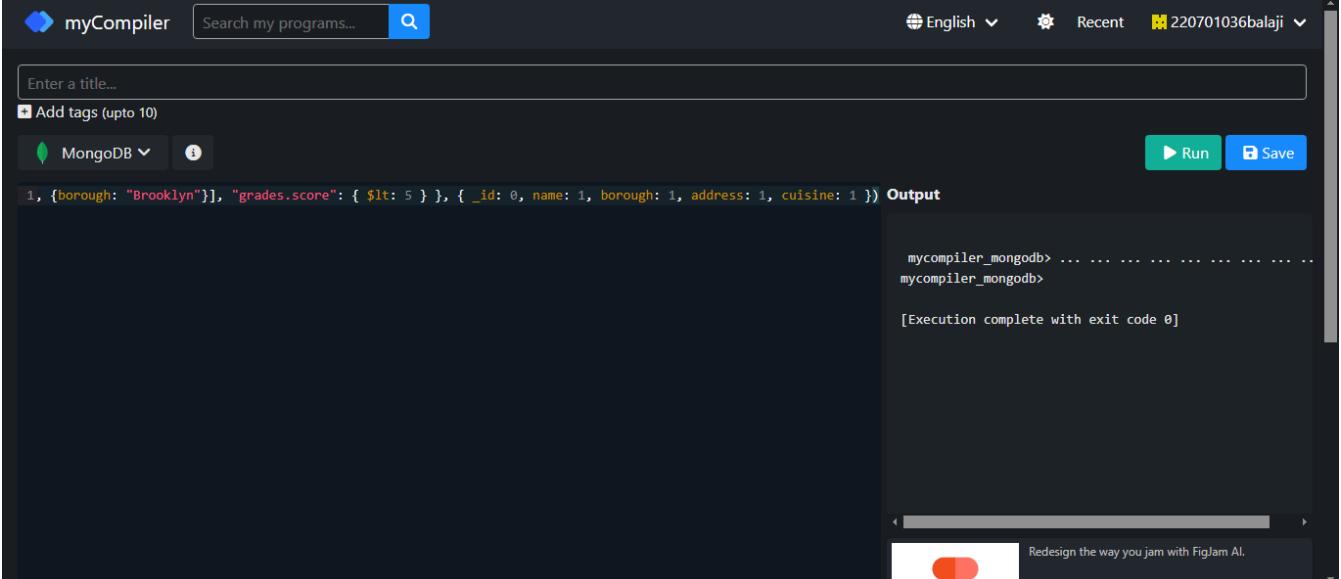
At the bottom of the interface, there's a watermark for FigJam AI.

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. At the top, there's a search bar labeled "Search my programs..." and a "Run" button. Below the search bar, there's a title input field ("Enter a title...") and a "Add tags (upto 10)" button. On the left, there's a dropdown menu set to "MongoDB" and a help icon. On the right, there are "Run" and "Save" buttons. The main area contains a code editor with the following MongoDB query:

```
1, {borough: "Brooklyn"}], "grades.score": { $lt: 5 } }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 }) Output
```

Below the code editor is a terminal window showing the output of the query execution:

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

At the bottom right of the terminal window, there's a watermark for FigJam AI.

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 with a dark theme. At the top left is the logo and name "myCompiler". To its right is a search bar with placeholder text "Search my programs..." and a magnifying glass icon. Further right are language selection ("English"), recent projects ("Recent"), and user account information ("220701036balaji"). Below the header is a large input field with placeholder text "Enter a title...". Underneath it is a button labeled "Add tags (upto 10)" with a plus sign icon. On the left side of the main area, there are two buttons: "MongoDB" with a dropdown arrow and "i" (info) with a question mark icon. On the right side are two buttons: "Run" with a play icon and "Save" with a disk icon. The main content area contains a code snippet and its execution output. The code is:`1 des.score": { $lt: 5 }, cuisine: { $ne: "American" }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 } } Output`  
The output shows the command prompt "mycompiler\_mongodb> ..." followed by a series of dots, indicating a long list of results. Below the prompt is the message "[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:

The screenshot shows the myCompiler interface. At the top, there's a search bar with "Search my programs..." and a magnifying glass icon. On the right, there are language selection ("English"), recent projects ("Recent"), and user information ("220701036balaji"). Below the header, there's a text input field for "Enter a title..." and a "Add tags (upto 10)" button. A "MongoDB" dropdown menu is open, showing "MongoDB" and an info icon. To the right are "Run" and "Save" buttons. The main area contains a code editor with the following text:

```
1$lt: 5 }, cuisine: { $nin: ["American", "Chinese"] } }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 } }) Output
```

Below the code editor, the terminal output is displayed:

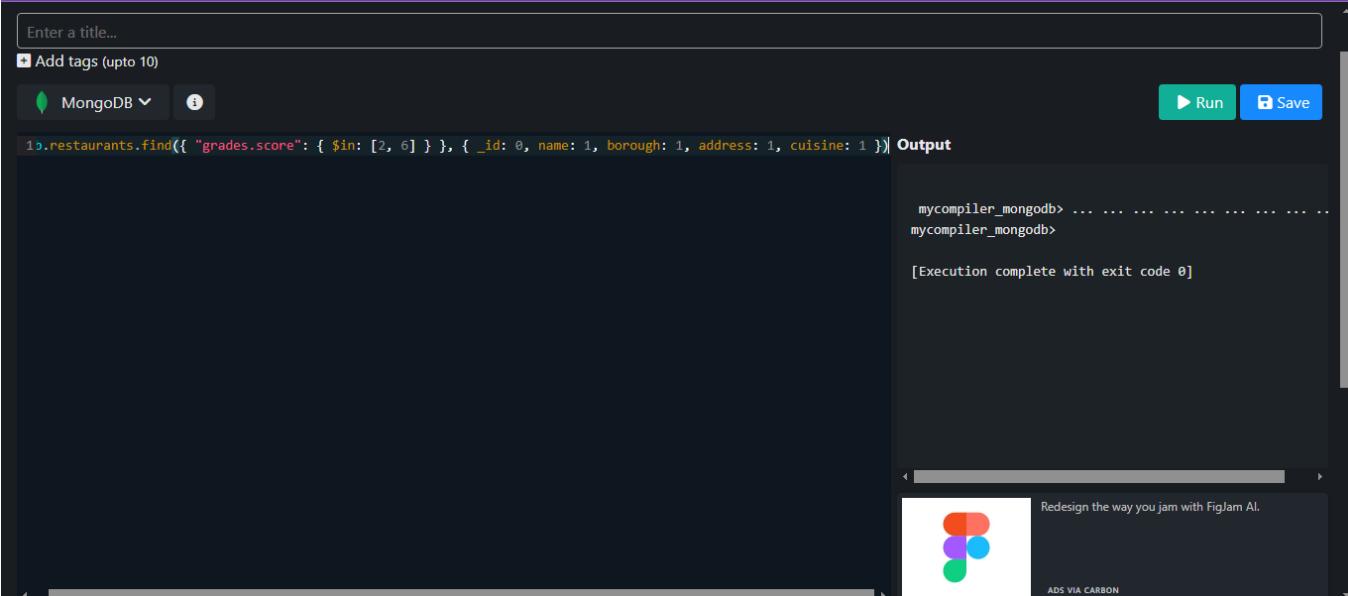
```
mycompiler_mongodb> . . .  
mycompiler_mongodb>  
[Execution complete with exit code 0]
```

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

### QUERY:

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

### OUTPUT:



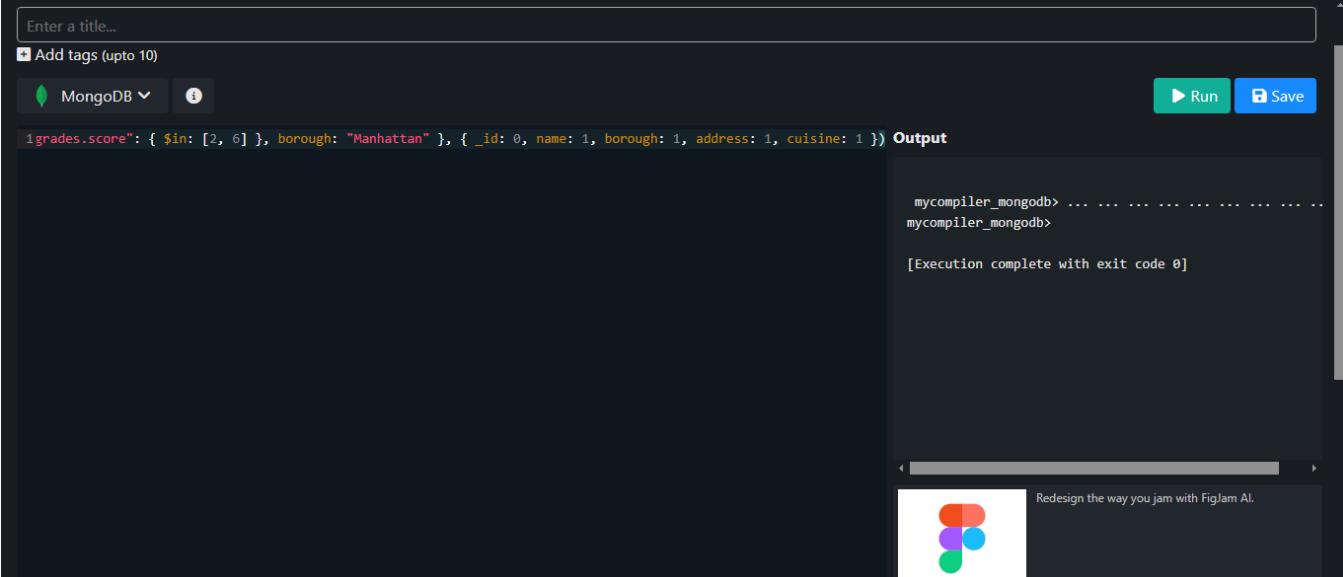
The screenshot shows a MongoDB shell interface. At the top, there is a search bar labeled "Enter a title..." and a checkbox for "Add tags (upto 10)". Below the search bar, there is a dropdown menu set to "MongoDB" and a help icon. On the right side, there are two buttons: "Run" (green) and "Save" (blue). The main area contains the MongoDB command: `db.restaurants.find({ "grades.score": { $in: [2, 6] } }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 })`. To the right of the command, the text "Output" is displayed. The output window shows the command being run and the response from the database: "mycompiler\_mongodb> ...". It also indicates that the execution completed successfully with an exit code of 0. A small advertisement for FigJam AI is visible at the bottom right of the interface.

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:



The screenshot shows a MongoDB query editor interface. At the top, there is a search bar labeled "Enter a title..." and a button "Add tags (upto 10)". Below the search bar, there are two dropdown menus: "MongoDB" and "i". On the right side, there are "Run" and "Save" buttons. The main area displays the MongoDB command and its output. The command is:

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

The output shows the results of the query execution:

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

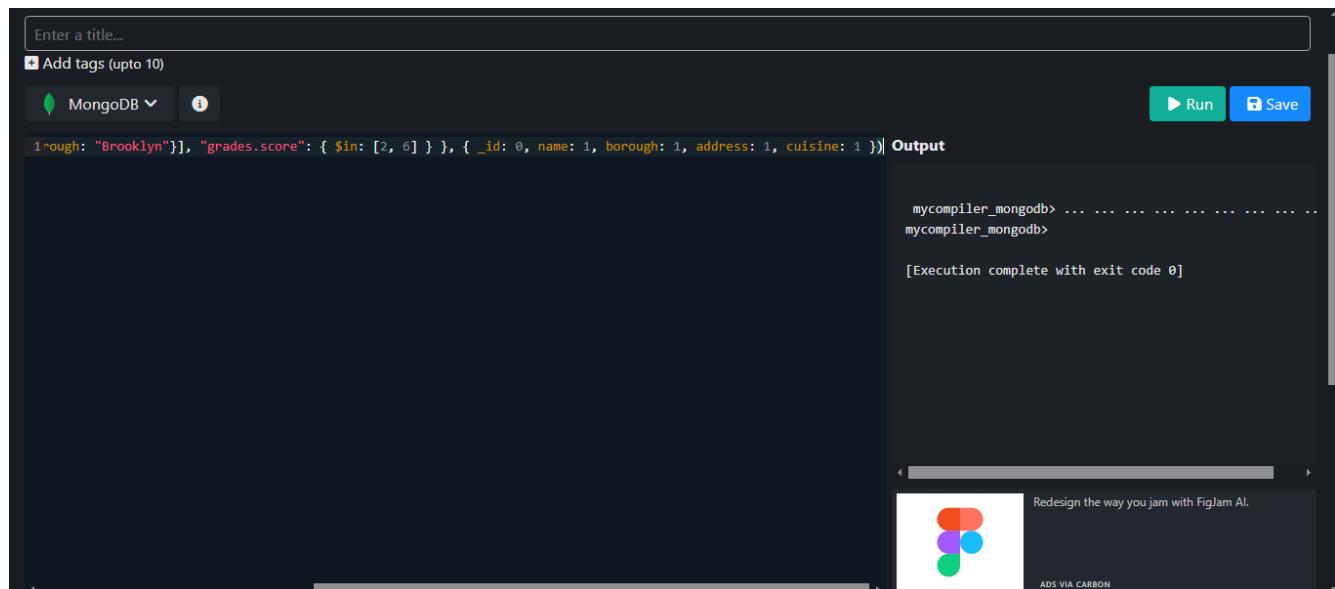
In the bottom right corner of the editor window, there is an advertisement for FigJam AI with the text "Redesign the way you jam with FigJam AI." and "ADS VIA CARBON".

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:



The screenshot shows a MongoDB shell interface. At the top, there is a search bar labeled "Enter a title..." and a button "Add tags (upto 10)". Below the search bar, there are two dropdown menus: one set to "MongoDB" and another with a question mark icon. To the right are "Run" and "Save" buttons. The main area contains a code input field with the following query:

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

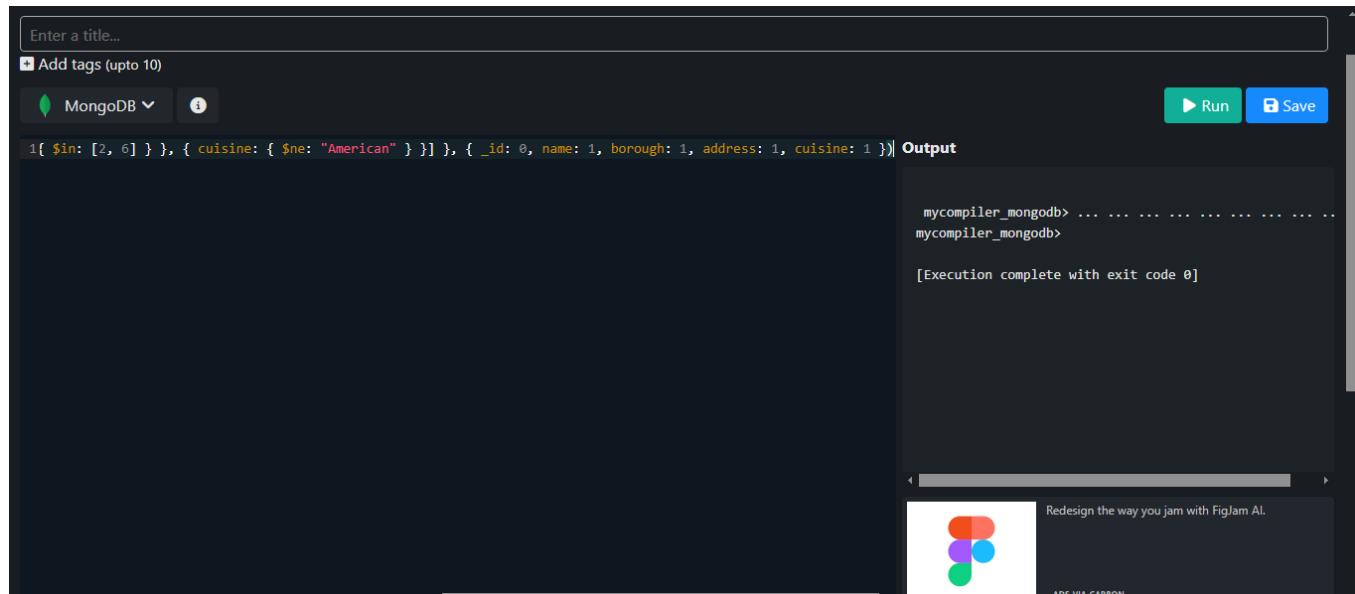
Below the code, the output window shows the command "mycompiler\_mongodb> ..." followed by a series of dots. The command "mycompiler\_mongodb> [Execution complete with exit code 0]" is displayed, indicating the query was successfully run. A small advertisement for FigJam AI is visible at the bottom right of the interface.

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

### QUERY:

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

### OUTPUT:



The screenshot shows a MongoDB query editor interface. At the top, there is a search bar labeled "Enter a title..." and a button "Add tags (upto 10)". Below the search bar, there are two dropdown menus: "MongoDB" and "Run". To the right of the "Run" button is a "Save" button. The main area contains a code editor with the following query:

```
1{ $in: [2, 6] }, { cuisine: { $ne: "American" } }], { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 })| Output
```

Below the code editor, the output window displays the results of the query execution:

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

In the bottom right corner of the output window, there is 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"] } })
```

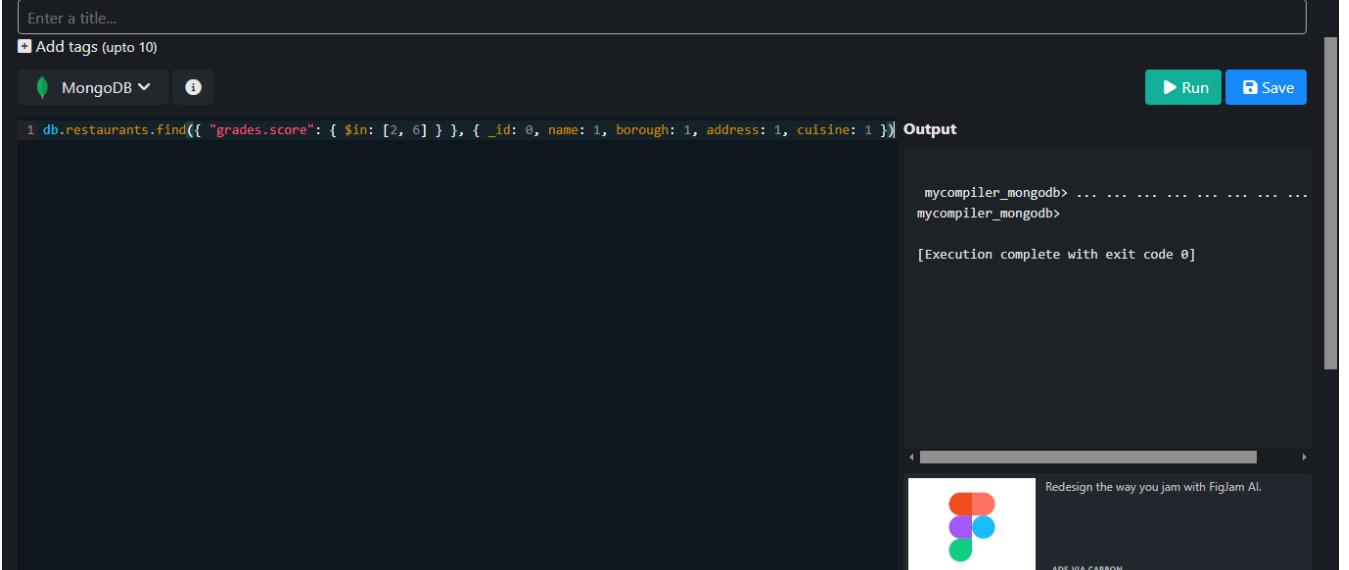
## OUTPUT:

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows a MongoDB shell interface. At the top, there is a search bar labeled "Enter a title..." and a button "+ Add tags (upto 10)". Below the search bar, there are two dropdown menus: "MongoDB" and "Run". To the right of the "Run" button is a "Save" button. The main area contains a command line and its output. The command is: `db.restaurants.find({ "grades.score": { $in: [2, 6] } }, { _id: 0, name: 1, borough: 1, address: 1, cuisine: 1 })`. The output shows the command itself followed by the response from the database: "mycompiler\_mongodb> ...". Below the command line, there is a small advertisement for FigJam AI.

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

# **MONGO DB**

**EX.NO:20**

**DATE: 24 - 5 - 24**

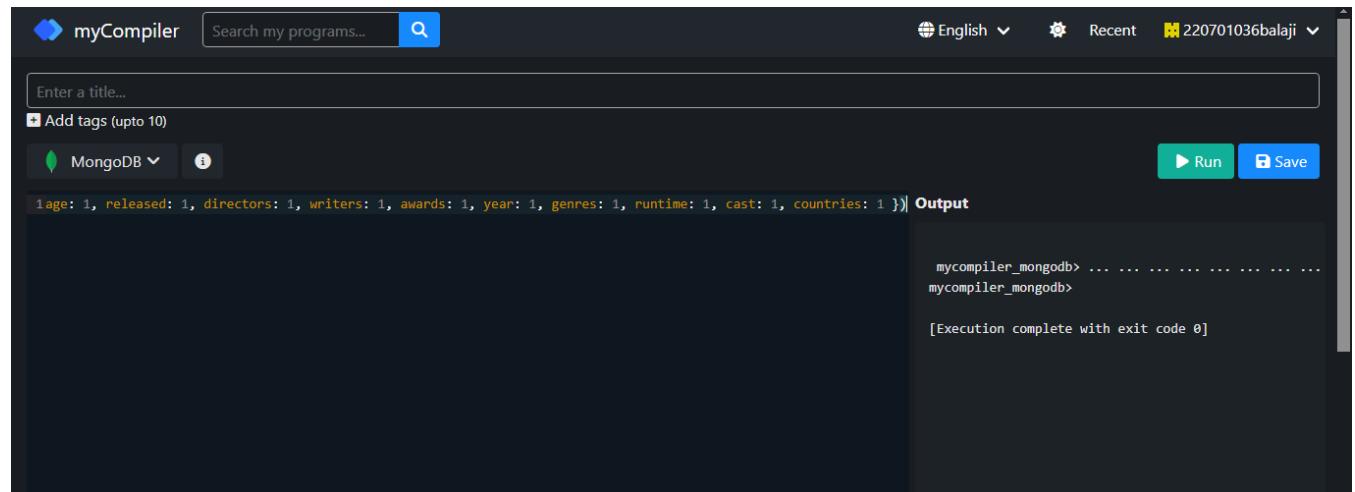
**REG.NO: 220701018**

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

**QUERY:**

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

**OUTPUT:**



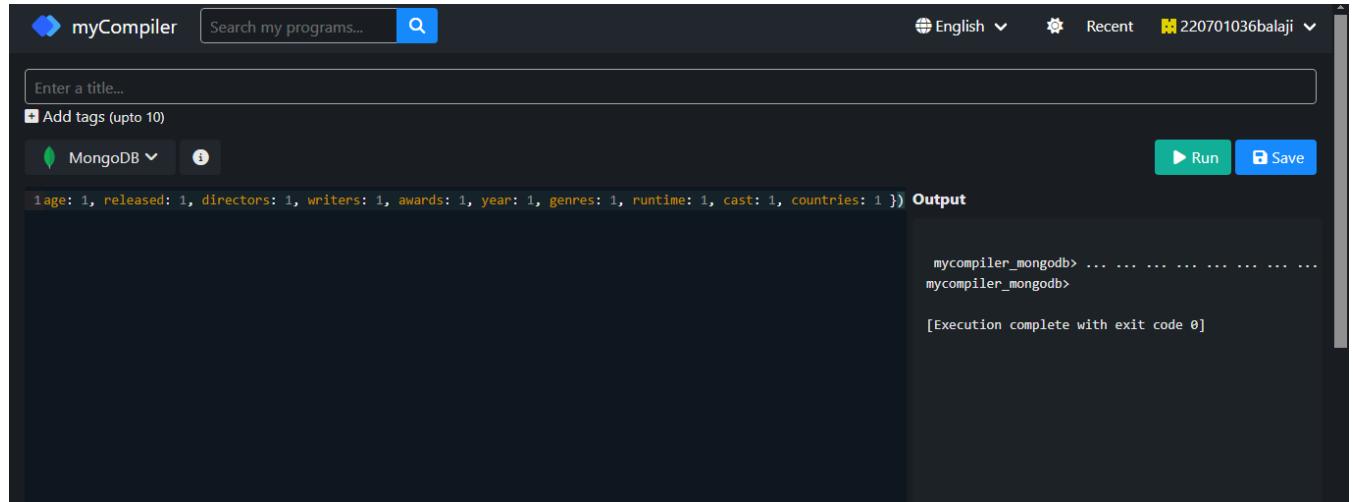
The screenshot shows the myCompiler interface. At the top, there's a search bar labeled "Search my programs..." and a magnifying glass icon. To the right are language selection ("English"), recent projects ("Recent"), and user account ("220701036balaji"). Below the search bar is a text input field with placeholder "Enter a title..." and a "Add tags (upto 10)" button. On the left, there's a dropdown menu set to "MongoDB" and a help icon. On the right, there are "Run" and "Save" buttons. The main area contains a command-line interface window. The command entered is "db.movies.find({ year: 1893 })| Output". The output shows the MongoDB prompt "mycompiler\_mongodb> ... . . . . . . . . . . ." followed by "[Execution complete with exit code 0]".

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the myCompiler interface. In the top left, there's a logo and the text "myCompiler". A search bar says "Search my programs..." with a magnifying glass icon. On the right, there are language selection ("English"), recent projects ("Recent"), and user information ("220701036balaji"). Below the search bar is a text input field with placeholder "Enter a title...". Underneath it is a button "Add tags (upto 10)" with a plus sign icon. To the right of the input fields are two buttons: "Run" (green) and "Save" (blue). The main area contains a code editor with the following MongoDB query:

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

Below the code editor is a "Output" section. It shows the command "mycompiler\_mongodb> ...". The output itself is empty, with only "[Execution complete with exit code 0]" visible at the bottom.

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 with the placeholder "Search my programs..." and a magnifying glass icon. To the right are language selection ("English"), recent projects, and user information ("220701036balaji"). Below the search bar is a text input field with placeholder "Enter a title...". Underneath it is a button labeled "Add tags (upto 10)". On the left, there's a dropdown for "MongoDB" and an info icon. On the right, there are "Run" and "Save" buttons. The main area displays a MongoDB query result:

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

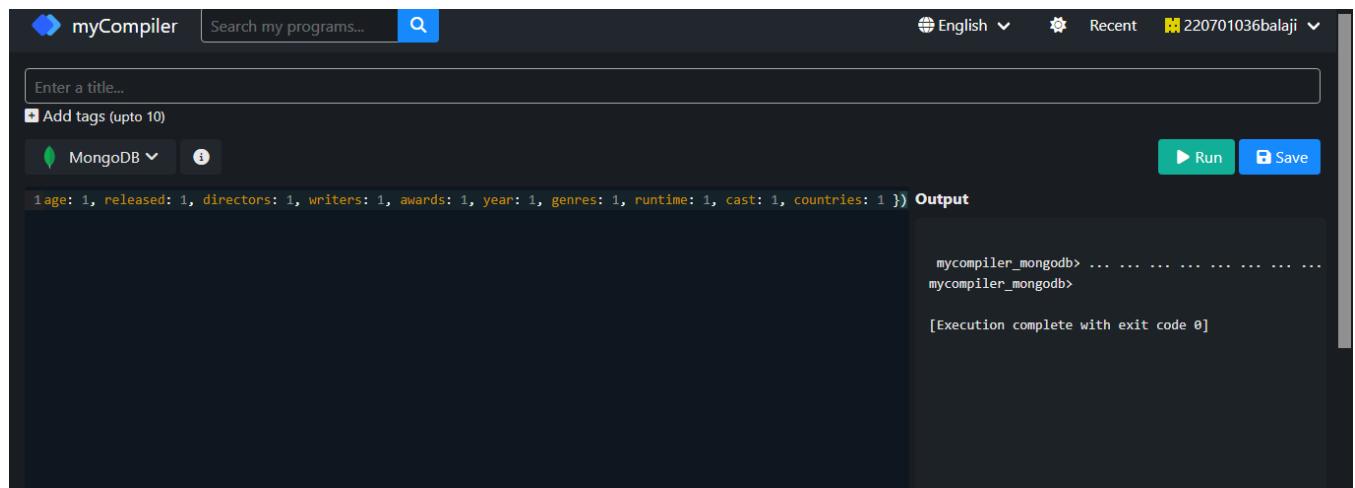
Below the output, the terminal shows the command prompt "mycompiler\_mongodb> ...". The status bar at the bottom indicates "[Execution complete with exit code 0]".

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' application interface. At the top, there's a search bar labeled 'Search my programs...' and a magnifying glass icon. On the right side of the header, there are language settings ('English'), recent projects ('Recent'), and a user profile ('220701036balaji'). Below the header, there's a text input field with placeholder 'Enter a title...', a 'Add tags (upto 10)' button, and a dropdown menu set to 'MongoDB'. To the right of these are 'Run' and 'Save' buttons. The main area contains a code editor with the following MongoDB query:

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

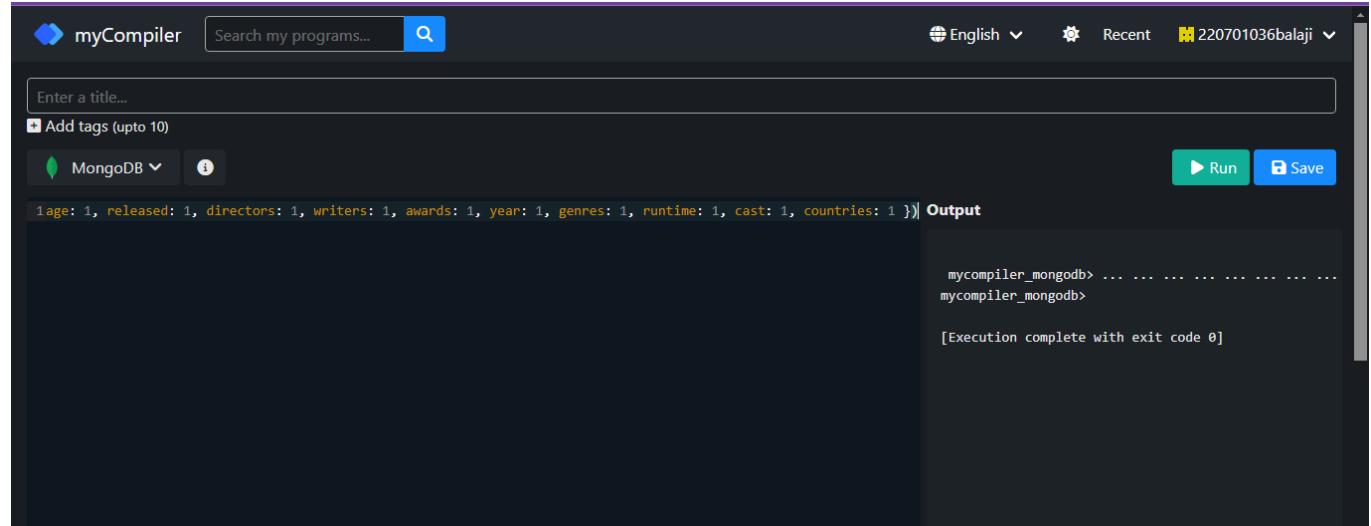
Below the code editor is a terminal window titled 'Output' showing the command prompt 'mycompiler\_mongodb>'. The terminal output is empty, followed by '[Execution complete with exit code 0]'. The overall interface has a dark theme.

**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 web application interface. At the top, there's a navigation bar with the logo 'myCompiler', a search bar 'Search my programs...', and language settings ('English'). Below the header, there's a form with fields for 'Enter a title...' and 'Add tags (upto 10)'. A dropdown menu shows 'MongoDB'. On the right side of the form are 'Run' and 'Save' buttons. The main area contains a code editor with the following MongoDB query:

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

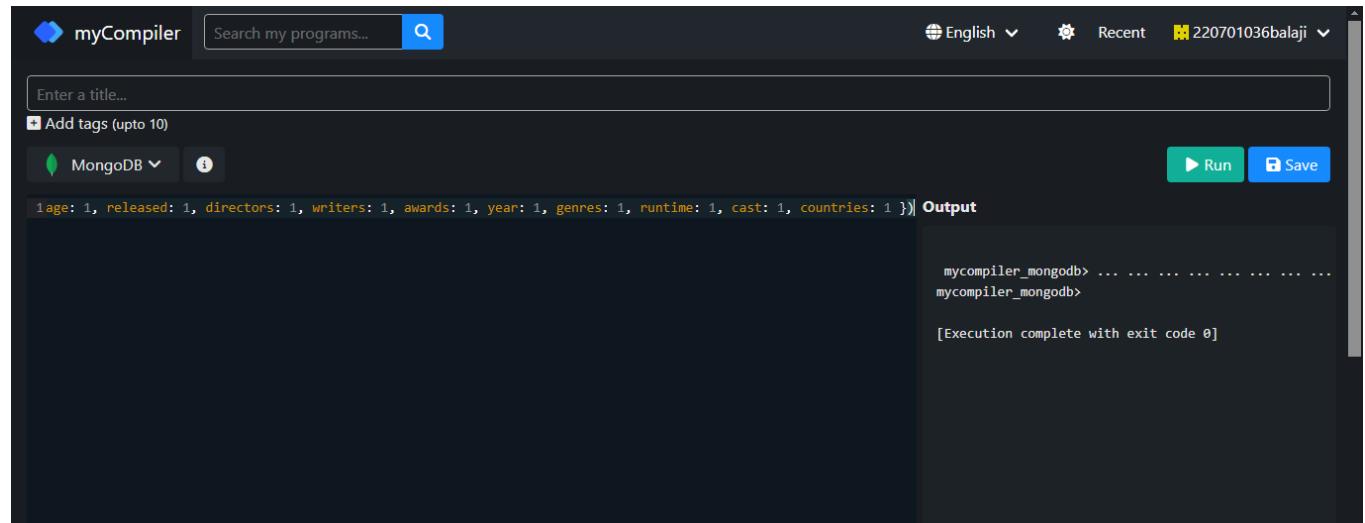
Below the code editor, the output window displays the command prompt 'mycompiler\_mongodb> ...' followed by '[Execution complete with exit code 0]'. The output window is mostly empty, indicating no results were displayed.

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the myCompiler interface. At the top, there's a search bar labeled "Search my programs..." and a magnifying glass icon. To the right are language selection ("English"), recent projects ("Recent"), and user information ("220701036balaji"). Below the search bar is a text input field with placeholder "Enter a title...". Underneath it is a checkbox for "Add tags (upto 10)" and a dropdown menu set to "MongoDB". On the far right of the toolbar are "Run" and "Save" buttons. The main area is titled "Output" and contains the MongoDB command: "db.movies.find({ rated: 'UNRATED' })". The output window shows the command being run in the terminal: "mycompiler\_mongodb> db.movies.find({ rated: 'UNRATED' })". The terminal then displays the results of the query, which are empty. The message "[Execution complete with exit code 0]" is shown at the bottom of the output window.

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

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

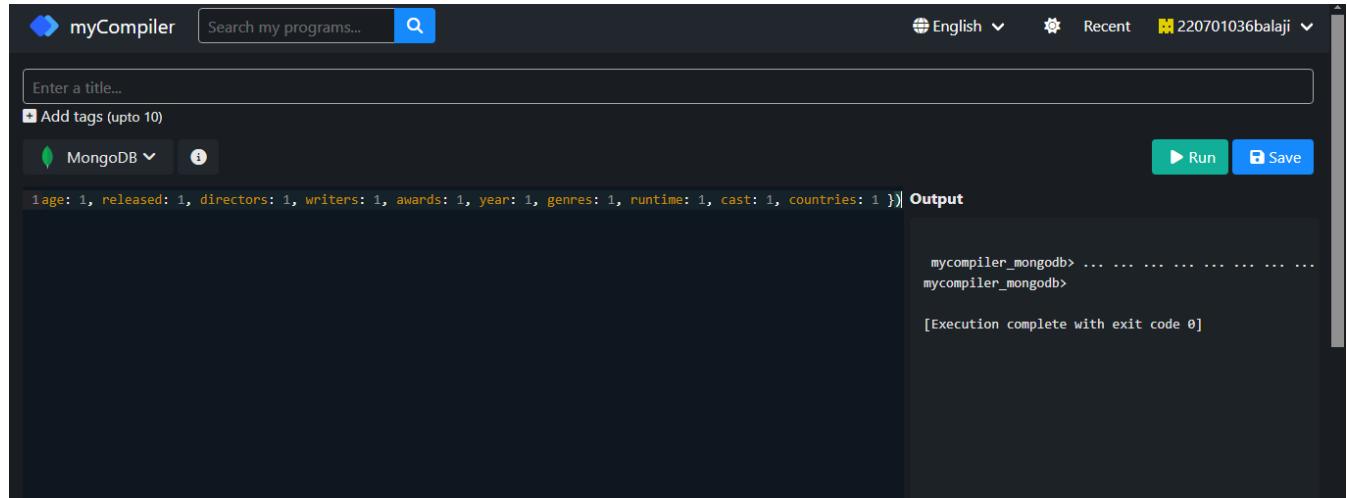
```
[Execution complete with exit code 0]
```

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

**QUERY:**

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

**OUTPUT:**



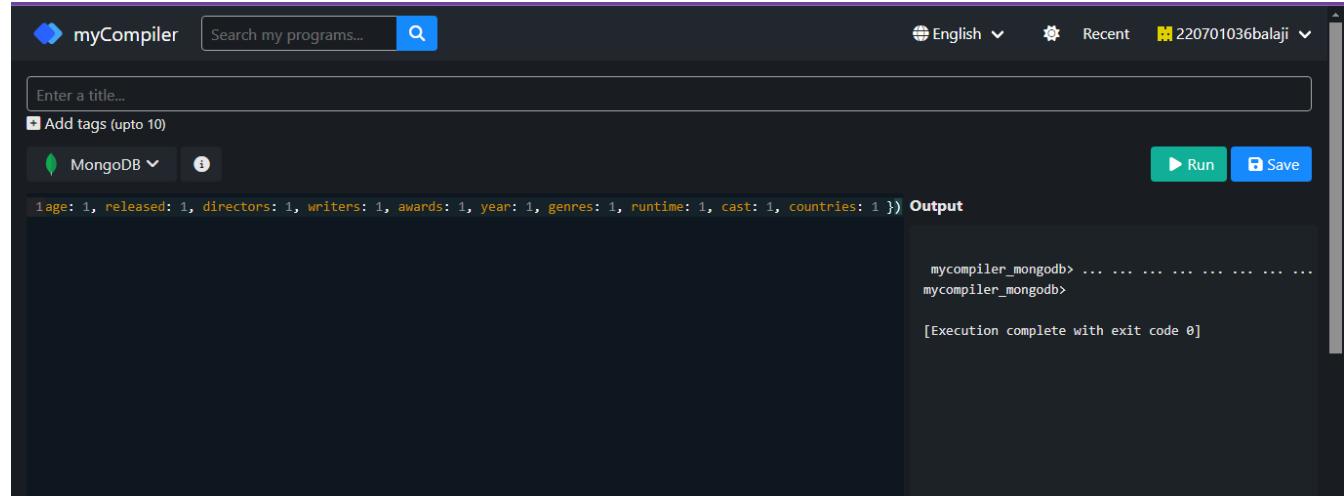
The screenshot shows the myCompiler interface. At the top, there's a search bar labeled "Search my programs..." and a magnifying glass icon. To the right are language selection ("English"), recent projects ("Recent"), and user information ("220701036balaji"). Below the search bar is a text input field with placeholder "Enter a title...". Underneath it is a button labeled "+ Add tags (upto 10)". On the left, there are two buttons: "MongoDB" with a dropdown arrow and "Run" with an info icon. On the right, there are "Run" and "Save" buttons. In the center, the command entered is: `db.movies.find({ 'imdb.votes': { $gt: 1000 } })`. To the right of the command is a "Output" section. The output shows the command being run: `mycompiler_mongodb> db.movies.find({ 'imdb.votes': { $gt: 1000 } })`, followed by a series of dots indicating progress. Finally, the message "[Execution complete with exit code 0]" is displayed.

**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. At the top, there's a search bar for programs and a language selection dropdown set to English. Below the search bar is a text input field labeled "Enter a title...". Underneath it, there's a "Add tags (upto 10)" button and a "MongoDB" dropdown menu. To the right of the input fields are "Run" and "Save" buttons. The main area contains a code editor with the following MongoDB query:

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

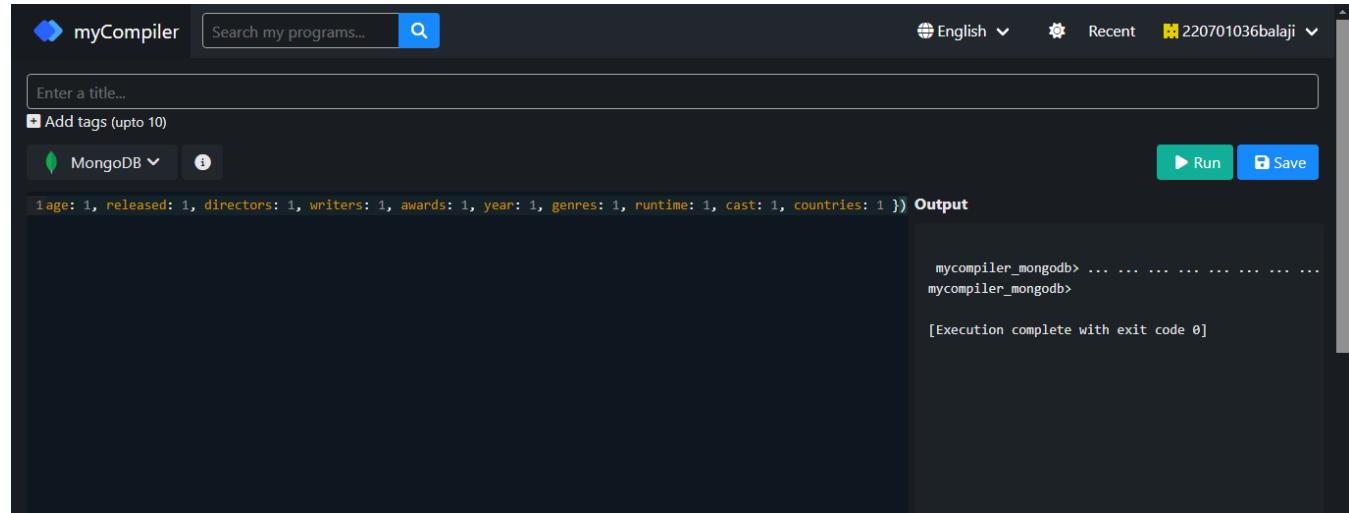
Below the code editor is a "Output" section. It displays the command "mycompiler\_mongodb> ..." followed by a series of dots, indicating the continuation of the output. The status message "[Execution complete with exit code 0]" is shown at the bottom of the output section.

**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:**



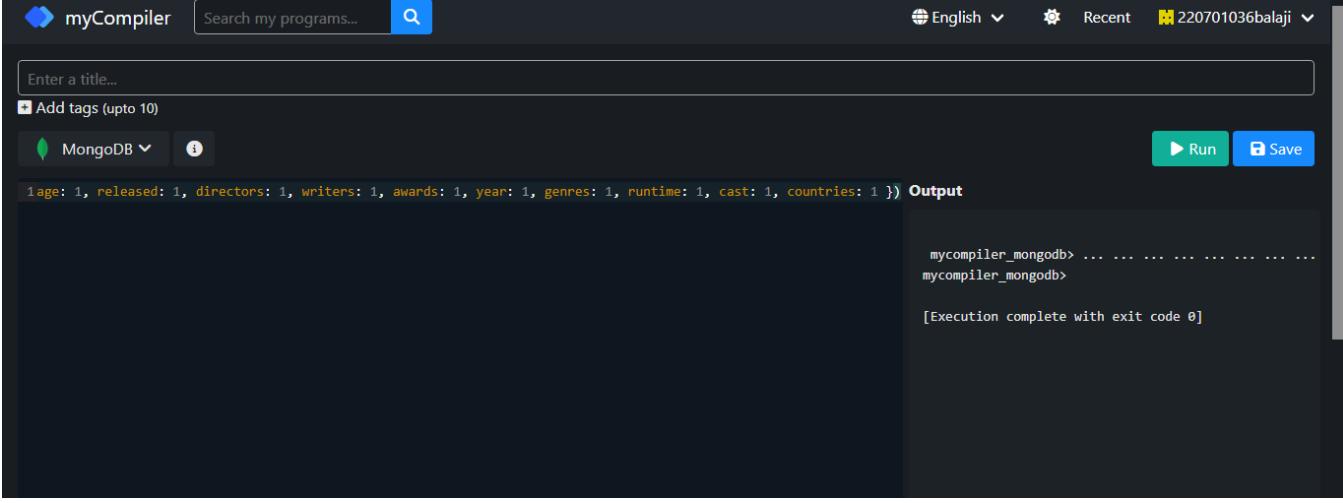
The screenshot shows the myCompiler web application interface. At the top, there's a search bar labeled "Search my programs..." with a magnifying glass icon. To the right are language selection ("English"), recent projects, and user profile ("220701036balaji"). Below the search bar is a text input field with placeholder "Enter a title...". Underneath it is a "Add tags (upto 10)" button. On the left, there's a dropdown menu set to "MongoDB" and a help icon. On the right, there are "Run" and "Save" buttons. The main area contains a command-line interface window titled "Output". The command entered is "db.movies.find({ 'tomatoes.viewer.rating': { \$gt: 4 } })". The output shows the command being run and the response: "mycompiler\_mongodb> ...". The status message at the bottom right of the output window is "[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:**



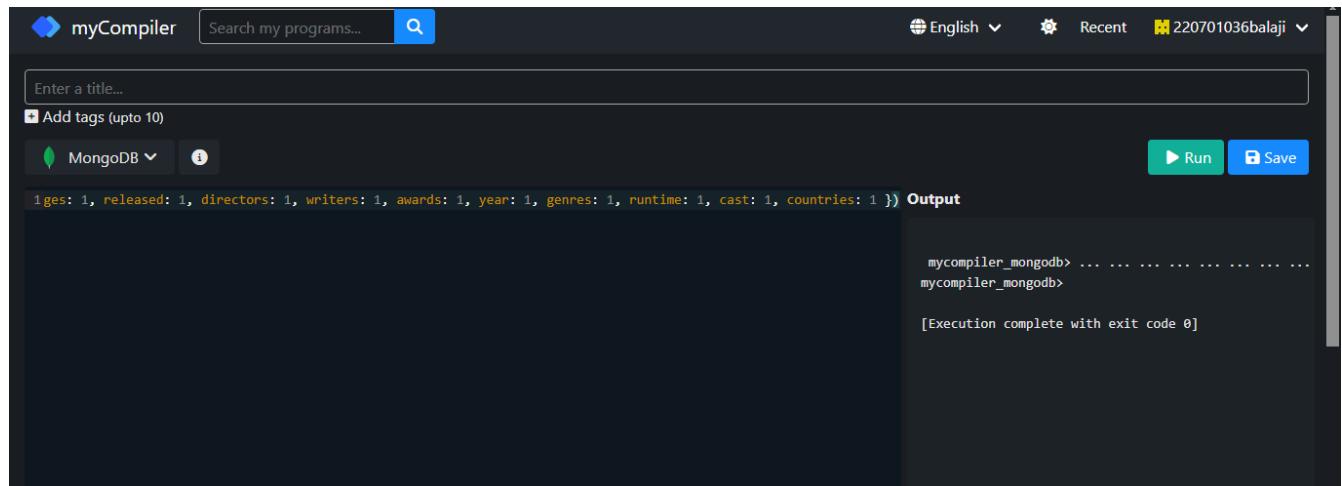
The screenshot shows the myCompiler interface. At the top, there's a search bar for programs and a language selector set to English. On the right, there are buttons for Recent and a user profile (220701036balaji). Below the header, there's a text input field with placeholder "Enter a title..." and a "Add tags (upto 10)" button. To the right of the input field are "Run" and "Save" buttons. A dropdown menu is open, showing "MongoDB" and an "info" icon. The main area contains a command line interface with the following text:  
`1age: 1, released: 1, directors: 1, writers: 1, awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1 })` **Output**  
mycompiler\_mongodb> ...  
mycompiler\_mongodb>  
[Execution complete with exit code 0]

**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:**



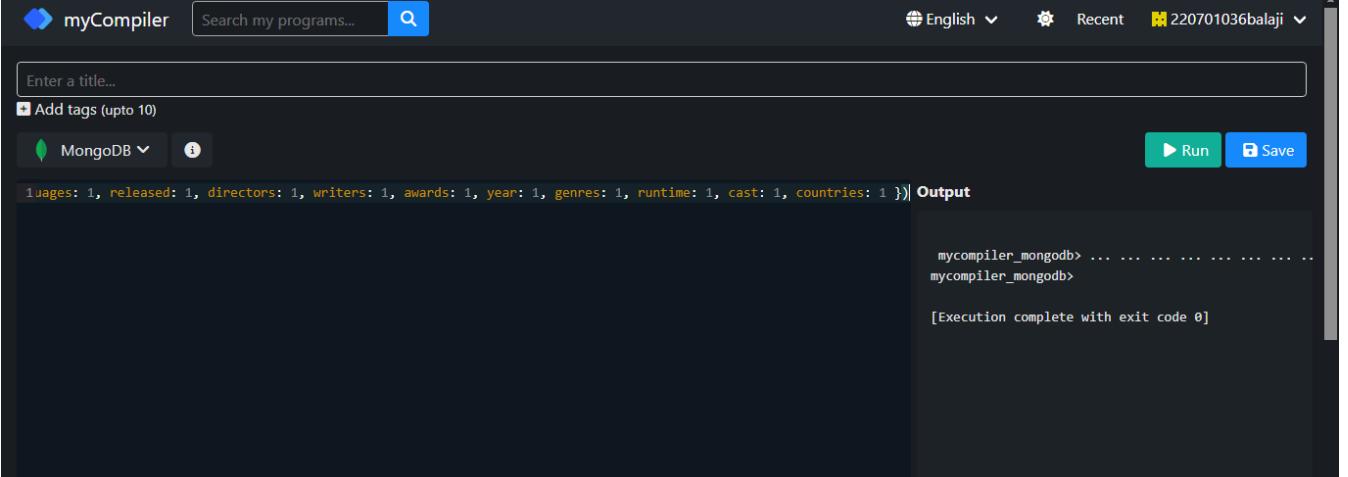
The screenshot shows the myCompiler interface. At the top, there's a search bar labeled "Search my programs..." and a magnifying glass icon. To the right are language selection ("English"), recent projects, and a user profile ("220701036balaji"). Below the search bar is a text input field with placeholder "Enter a title...". Underneath it is a button labeled "+ Add tags (upto 10)". On the left, there are two dropdown menus: "MongoDB" and "i". On the right, there are "Run" and "Save" buttons. The main area is titled "Output" and contains the MongoDB 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 } )". Below the query, the terminal output shows: "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. At the top, there's a search bar labeled "Search my programs..." and a magnifying glass icon. To the right are language selection ("English"), recent projects, and user information ("220701036balaji"). Below the search bar is a text input field with placeholder "Enter a title...". Underneath it is a button for "Add tags (upto 10)". On the left, there are dropdowns for "MongoDB" and "Run/Save" buttons. The main area is titled "Output" and contains the MongoDB query and its execution results.

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

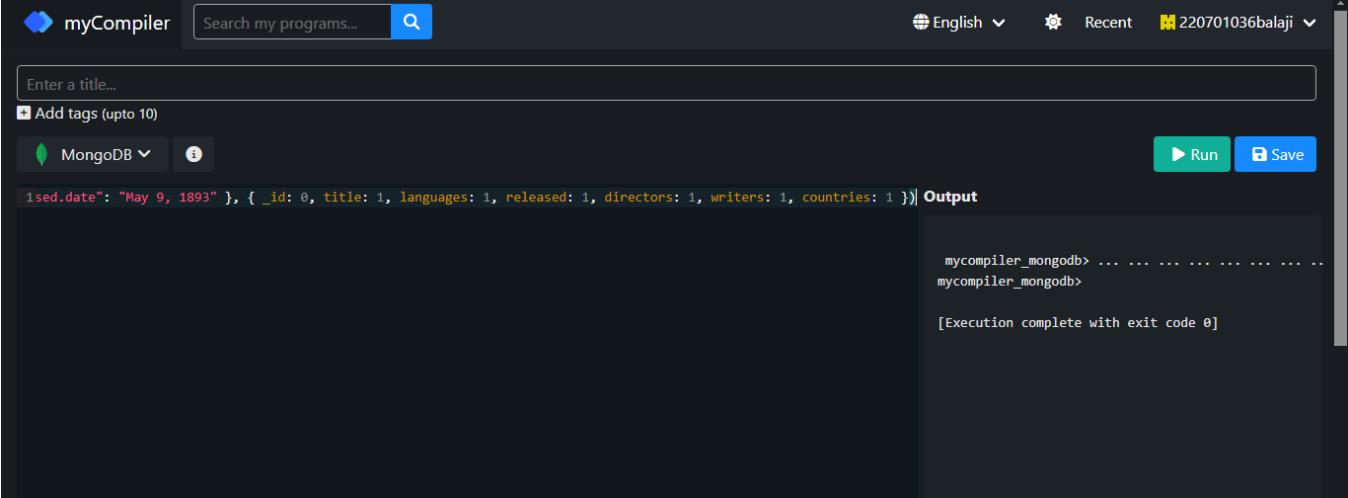
```
mycompiler_mongodb> ... . . . . . . . . .  
mycompiler_mongodb>  
[Execution complete with exit code 0]
```

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the myCompiler interface. At the top, there's a search bar labeled "Search my programs..." and a magnifying glass icon. On the right, there are language selection ("English"), recent projects ("Recent"), and user account ("220701036balaji"). Below the search bar is a text input field with placeholder "Enter a title...". Underneath it is a checkbox for "Add tags (upto 10)" and a dropdown menu set to "MongoDB". To the right of these are "Run" and "Save" buttons. The main area contains a code editor with the following MongoDB query:

```
1sed.date": "May 9, 1893", { _id: 0, title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 }| Output
```

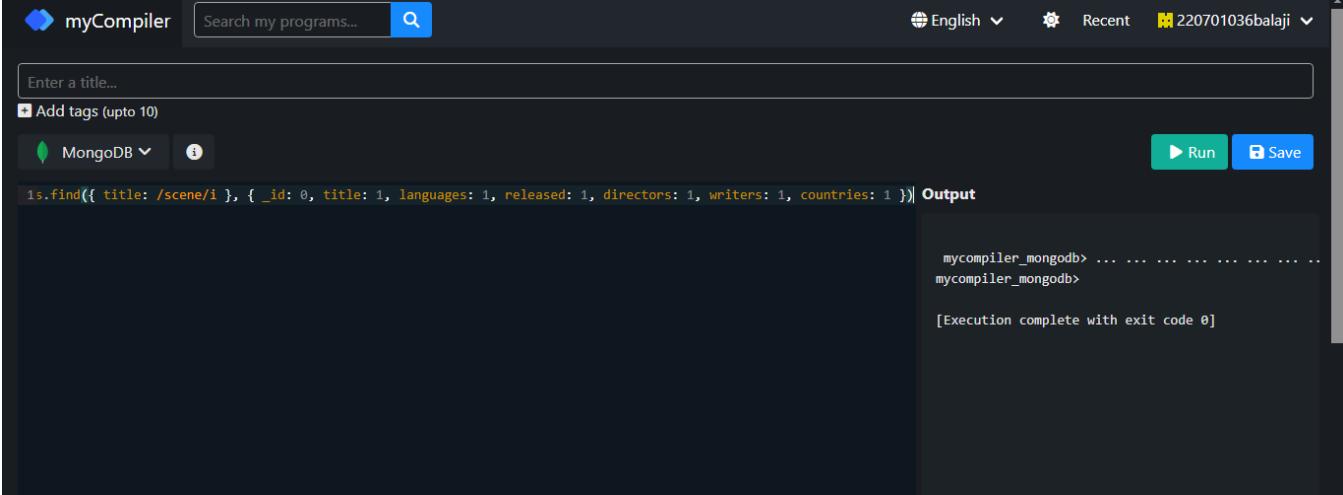
Below the code editor is a terminal window showing the command prompt "mycompiler\_mongodb>". The output of the command is visible in the terminal window, indicating the query was executed successfully with an exit code of 0.

**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 interface. At the top, there's a search bar with 'myCompiler' and a search icon. To the right are language selection ('English'), recent projects, and user information ('220701036balaji'). Below the search bar is a text input field with placeholder 'Enter a title...' and a button to 'Add tags (upto 10)'. On the left, there's a dropdown for 'MongoDB' and a help icon. On the right, there are 'Run' and 'Save' buttons. The main area contains a code editor with the following MongoDB query:

```
db.movies.find({ title: /scene/i }, { _id: 0, title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 })| Output
```

Below the code editor is a terminal window showing the output of the query:

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

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