

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 DEPARTMENT (ID number(9),name varchar2(35));

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'Home', 'SQL Workshop', 'SQL Commands', 'Schema ADHISH13', and various icons for saving, running, and navigating. Below the toolbar, there are buttons for 'Autocommit' (checked), 'Rows' (set to 10), and 'Run'. The main area contains the SQL command to create the 'DEPARTMENT' table:

```
create table DEPARTMENT(
  ID number(9),
  NAME varchar2(35)
);
```

Below the SQL input area, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, displaying the message 'Table created.' and '0.02 seconds' execution time.

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 EMPS(ID number(7),Last_Name varchar2(25),First_Name varchar2(25),Dept_ID number(7));

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'Home', 'SQL Workshop', 'SQL Commands', 'Schema ADHISH13', and 'Help'. Below the toolbar, there are buttons for 'Autocommit' (checked), 'Rows' (set to 10), and 'Run'. The main area contains the SQL command to create the table:

```
create table EMPS(
ID number(7),
Last_Name Varchar2(25),
First_Name Varchar2(25),
Dept_ID Number(7)
);
```

The 'Results' tab is selected at the bottom, showing the output: "Table created." and "0.01 seconds".

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

QUERY:

```
Alter table EMPS modify(Last_Name varchar2(50));
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'Home', 'SQL Workshop', 'SQL Commands', 'Schema ADHISH13', and various toolbar icons. Below the toolbar, there are buttons for 'Autocommit' (checked), 'Rows' (set to 10), and 'Run'. The main query editor window contains the SQL command: 'Alter table EMPS modify(Last_name Varchar2(50));'. The status bar at the bottom indicates the results tab is selected, along with other options like Explain, Describe, Saved SQL, and History.

```
Alter table EMPS modify(Last_name Varchar2(50));
```

Table altered.

0.03 seconds

4. Create the EMPLOYEES2 table based on the structure of EMPLOYEES table. Include Only the Employee_id, First_name, Last_name, Salary and Dept_id columns. Name the columns Id, First_name, Last_name, salary and Dept_id respectively.

QUERY:

```
Create table EMPLOYEES2(ID number(7),first_name Varchar2(25),Last_name Varchar2(25),Salary number(7),Dept_id number(7));
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, it says "Home > SQL Workshop > SQL Commands". The schema dropdown is set to "ADHISH13". Below the toolbar, there are buttons for "Autocommit" (checked), "Rows" (set to 10), "Save", and "Run". The main area contains the following SQL code:

```
create table EMPLOYEES2(
Id number(7),
First_Name Varchar2(25),
Last_Name Varchar2(25),
Salary number(7),
Dept_id number(7)
);
```

Below the code, there is a menu bar with tabs: Results (selected), Explain, Describe, Saved SQL, and History.

Table created.

0.00 seconds

5. Drop the EMP table.

QUERY:

Drop table EMPS ;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, it says "Home > SQL Workshop > SQL Commands". The schema dropdown is set to "ADHISH13". Below the toolbar, there are buttons for "Autocommit" (checked), "Rows" (set to 10), "Save", and "Run". The main area contains the following SQL code:

```
Drop table EMPS;
```

Below the code, there is a menu bar with tabs: Results (selected), Explain, Describe, Saved SQL, and History.

Table dropped.

0.04 seconds

6. Rename the EMPLOYEES2 table as EMP.

QUERY:

Rename EMPLOYEES2 to EMPS13 ;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top menu bar includes 'Home', 'SQL Workshop', 'SQL Commands', 'Schema' set to 'ADHISH13', and various toolbar icons. The main area contains the SQL command: 'Rename EMPLOYEES2 to EMPS13'. Below the command, the results section displays the message 'Statement processed.' and a execution time of '0.00 seconds'. The results tab is currently selected.

```
Rename EMPLOYEES2 to EMPS13

Statement processed.

0.00 seconds
```

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

QUERY:

comment on table DEPARTMENT is 'Department info';

comment on table EMPS is 'Employee info';

OUTPUT:

The screenshot shows the SQL Commands tab in Oracle SQL Developer. The command entered is:

```
1 comment on table DEPARTMENT is 'department info'
2
```

The results pane shows:

Statement processed.
0.01 seconds

8. Drop the First_name column from the EMP table and confirm it.

QUERY:

Alter table EMPS Drop column First_name;

OUTPUT:

The screenshot shows the SQL Commands tab in Oracle SQL Developer. The command entered is:

```
Alter table EMPS
Drop column First_name;
```

The results pane shows:

Table altered.
0.07 seconds

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

RESULT:

MANIPULATING DATA

EX_NO:2

DATE:

1. Create MY_EMPLOYEE table with the following structure

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

QUERY:

Create table MY_EMPLOYEE(ID number(4),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_EMPLOYEE(
2 ID number(4),
3 Last_Name Varchar(25),
4 First_Name Varchar(25),
5 Userid Varchar(25),
6 Salary Number(9,2)
7 );
```

The 'Run' button is highlighted in green. In the Results tab, the output is displayed:

```
Table created.
0.05 seconds
```

At the bottom, the footer includes the URL 220701013@rejulakshmi.edu.in, the schema name edhish15, and the copyright notice Copyright © 1999, 2023, Oracle and/or its affiliates.

2. Add the first and second rows data to MY_EMPLOYEE table from the following sample data.

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

QUERY:

```
Insert into 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. The SQL command window contains the following code:

```
1 INSERT INTO MY_EMPLOYEE VALUES(1,'Patel','Ralph','rpatel',895);
2
```

The results tab shows the output of the executed query:

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

At the bottom, the footer includes the URL <https://220701015@rajalakshmi.edu.in>, the schema name `adhisht3`, and the page number `en`. The copyright notice reads "Copyright © 1999-2023, Oracle and/or its affiliates." and the version `Oracle APEX 23.2.4`.

3. Display the table with values.

QUERY:

Select * from MY_EMPLOYEE;

OUTPUT:

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

- SQL Commands** tab is selected.
- Language**: SQL
- Schema**: WKSP_ADHISH13
- Run** button is visible.
- Query History**:
1 select * from MY_EMPLOYEE;
2
3
- Results** tab is selected.
- Table Headers**: ID, LAST_NAME, FIRST_NAME, USERID, SALARY
- Data Rows**:

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
2	dancs	betty	bdancs	860
1	patel	ralph	rpatel	895
- Message**: 2 rows returned in 0.01 seconds
- Download** link is present.

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(5,'ropebar','Audrey','areopbur',1550);
```

OUTPUT:

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

- SQL Commands:** The interface is titled "SQL Commands".
- Language:** Set to "SQL".
- Schema:** Set to "WKSP_ADHISH13".
- Query:** The query entered is: `1 INSERT INTO MY_EMPLOYEE values(5,'ropebar','audrey','aropebur',1550);`.
- Results:** The results section shows the output of the query:
 - 1 row(s) inserted.
 - 0.00 seconds

5. Make the data additions permanent.

QUERY:

Select * from MY_EMPLOYEE;

OUTPUT:

The screenshot shows a SQL command window with the following interface elements:

- Top bar: SQL Commands, Schema (WKSP_ADHISH13), Save, Run.
- Toolbar: Language (SQL), Rows (10), Clear Command, Find Tables.
- Query editor:

```
1 select * from MY_EMPLOYEE;
2
```
- Results tab: Shows a table with 4 rows of data.

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

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

QUERY:

```
Update MY_EMPLOYEE set last_name='drexler' where id=3;
```

OUTPUT:

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

- Toolbar:** Includes "SQL Commands" (dropdown), "Schema" (set to "WKSP_ADHISH13"), "Save", and "Run" buttons.
- Query Editor:** Shows the SQL command:

```
1 Update MY_EMPLOYEE set Last_name='drexler'
2 where id=3;
```
- Results Tab:** Active tab, showing the output of the query.
- Output:**

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

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:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The top right corner shows the user's name 'Adhish K' and schema 'WKSP_ADHISH'. The main workspace is titled 'SQL Commands' and contains the following SQL code:

```
1 update MY_EMPLOYEE
2 set Salary=1000
3 where Salary<900;
```

Below the code, there are buttons for 'Save' and 'Run'. The results tab is selected, showing the output of the query:

3 row(s) updated.
0.00 seconds

At the bottom, the footer includes copyright information: Copyright © 1999, 2025, Oracle and/or its affiliates. and Oracle APEX 23.2.4.

8. Delete Betty dancs from MY_EMPLOYEE table.

QUERY:

Delete from MY_EMPLOYEE where first_name='Betty' and Last_name='dancs';

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema is set to WKSP_ADHISH13. The SQL command input field contains the following code:

```
1 delete from MY_EMPLOYEE  
2 where First_name = 'Betty' and Last_name='Dancs';
```

The results section shows the output of the query:

1 row(s) deleted.
0.02 seconds

At the bottom, there are footer links for 220701013@rajalakshmi.edu.in, adhish13, en, Copyright © 1999, 2023, Oracle and/or its affiliates, and Oracle APEX 23.2.4.

9. Empty the fourth row of the emp table.

QUERY:

Delete from MY_EMPLOYEE where first_name='A';

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side of the header shows the user 'Adhish K' and the schema 'WKSP_ADHISH13'. The main workspace is titled 'SQL Commands' and contains the following SQL code:

```
1 delete from MY_EMPLOYEE
2 where id=4;
```

The 'Run' button is visible at the bottom right of the command input area. Below the command input, the 'Results' tab is selected. The output section displays the results of the executed query:

1 row(s) deleted.
0.00 seconds

At the bottom of the page, there are footer links for user information (22070115@rajalakshmi.edu.in, adhish13, en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the software version (Oracle APEX 23.2.4).

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

RESULT:

INCLUDING CONSTRAINTS

EX_NO:3

DATE:

1. Add a table-level PRIMARY KEY constraint to the EMP table on the ID column. The constraint should be named at creation. Name the constraint my_emp_id_pk.

QUERY:

```
alter table MY_EMPLOYEE add constraint my_emp_id_pk primary key(ID);
```

OUTPUT:

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

```
1 alter table MY_EMPLOYEE add constraint my_emp_id_pk primary key(ID);
2
```

In the bottom results pane, the output is:

```
Table altered.

0.07 seconds
```

At the bottom left, the session information is shown: z20701013@rajalakshmi.edu.in, adhish13, en. At the bottom right, the copyright notice is: Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4.

2. Create a PRIMARY KEY constraint to the DEPT table using the ID column. The constraint should be named at creation. Name the constraint my_dept_id_pk.

QUERY:

```
Create table DEPT(dept_id number(6),dept_name varchar2(10),
Constraints my_dept_id_pk primary key (dept_id);
```

\

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The SQL Commands tab is active. The schema is set to WKSP_ADHISH13. The SQL editor contains the following code:

```
1 create table DEPT(dept_id number(6),dept_name varchar2(10),
2 constraints my_dept_id_pk primary key(dept_id));
```

The results section shows the output of the command:

```
Table created.
```

Execution time: 0.04 seconds.

At the bottom, the footer includes user information (220701015@rajalakshmi.edu.in, adhish13, en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates), and the version (Oracle APEX 23.2.4).

3. Add a column DEPT_ID to the EMP table. Add a foreign key reference on the EMP table that ensures that the employee is not assigned to nonexistent department. Name the constraint my_emp_dept_id_fk.

QUERY:

```
Alter table MY_EMPLOYEE add constraint my_emp_dept_id_fk foreign key(DEPT_ID)references  
MY_EMPLOYEE(ID);
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema is set to WKSP_ADHISH13. The query editor contains the following SQL code:

```
1 alter table MY_EMPLOYEE  
2 add constraint my_emp_dept_id_fk foreign key(DEPT_ID) references MY_EMPLOYEE(ID);
```

The 'Run' button is highlighted in green at the bottom right of the editor. Below the editor, the results section shows the output of the executed command:

Table altered.
0.05 seconds

At the bottom of the page, there are footer links for user information (220701013@rajalakshmi.edu.in, adhish13, en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates), and the Oracle APEX version (Oracle APEX 23.2.4).

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

QUERY:

```
Create table employee(id int primary key);
Alter table MY_EMPLOYEE add constraints ck_1 check(commission>0);
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The top right corner shows the user's name 'Adhish K' and schema 'WKSP_ADHISH13'. The main area is titled 'SQL Commands' with a language dropdown set to 'SQL'. Below the title are buttons for 'Clear Command' and 'Find Tables'. The SQL editor contains the following command:

```
1 alter table MY_EMPLOYEE add constraint ck_1 check(commission>0);
```

The results tab is selected at the bottom, showing the output of the command:

```
Table altered.  
0.04 seconds
```

At the bottom of the page, there are footer links for 'Copyright © 1999, 2023, Oracle and/or its affiliates.', 'Oracle APEX 23.2.4', and session information including the user ID 'z20701013@rajalakshmi.edu.in' and session ID 'adhish13 en'.

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

RESULT

Writing Basic SQL SELECT Statements

EX_NO:4

DATE:

1. The following statement executes successfully.

Identify the Errors

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

QUERY:

```
SELECT MY_EMPLOYEE_ID, LAST_NAME, 'SALARY' FROM MY_EMMPLOYEE;
```

OUTPUT:

The screenshot shows a SQL command window interface. At the top, there's a toolbar with various icons and a dropdown menu labeled "Schema: WKSP_ADHISH13". Below the toolbar, the main area has tabs for "SQL Commands", "Language: SQL", "Rows: 10", and buttons for "Clear Command", "Find Tables", "Save", and "Run". The code input area contains two lines of SQL:

```
1 select EMPLOYEE_ID,Last_name,"SALARY" *12 "Annual_salary" FROM MY_EMMPLOYEE;  
2
```

Below the code, there's a "Results" tab selected. The results table has three columns: "EMPLOYEE_ID", "LAST_NAME", and "Annual_salary". A single row is returned, showing:

EMPLOYEE_ID	LAST_NAME	Annual_salary
1	prasad	168000

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

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

QUERY:

SELECT* FROM MY_EMMLOYEE;

OUTPUT:

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

- Toolbar:** Includes Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run buttons.
- Query Editor:** Shows the query `SELECT * FROM MY_EMMLOYEE;`.
- Results Tab:** Active tab, showing the following table output:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	MANAGER_ID	DEPARTMENT_ID	ANNUAL_SALARY
1	rajesh	prasad	raj@gmail.com	989321875	0	35	14000	13	1	300000

- Message:** 1 rows returned in 0.03 seconds
- Download:** Option to download the results.

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

QUERY:

```
SELECT EMPLOYEE_ID as EMPLOYEE_NUMBER, LAST_NAME, JOB_ID, HIRE_DATE from  
MY_EMMLOYEE;
```

OUTPUT:

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

- Toolbar:** Includes tabs for "SQL Commands", "Language" (set to SQL), "Rows" (set to 10), "Clear Command", "Find Tables", "Save", and "Run".
- Query Editor:** Displays the SQL query:

```
1 SELECT EMPLOYEE_ID as EMPLOYEE_NUMBER, LAST_NAME, JOB_ID, HIRE_DATE from MY_EMMLOYEE;  
2
```
- Results Tab:** Active tab, showing the output of the query.
- Table Output:** A grid showing the results of the query. The columns are labeled "EMPLOYEE_NUMBER", "LAST_NAME", "JOB_ID", and "HIRE_DATE". The single row contains the values: 1, prasad, 35, and 0.
- Message Bar:** Shows "1 rows returned in 0.00 seconds" and a "Download" link.

4. Provide an alias STARTDATE for the hire date.

QUERY:

SELECT HIRE_DATE AS START_DATE FROM MY_EMMPLOYEE;

OUTPUT:

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

- Toolbar:** Includes tabs for "SQL Commands", "Language" (set to SQL), "Rows" (set to 10), "Clear Command", "Find Tables", "Save", and "Run".
- Query Area:** Displays the SQL command:

```
1 SELECT HIRE_DATE AS START_DATE FROM MY_EMMPLOYEE;
2
```
- Results Tab:** Active tab, showing the output of the query.
- Output Table:** A single row with the value "0" under the column header "START_DATE".
- Footer:** Shows "1 rows returned in 0.00 seconds" and a "Download" link.

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

QUERY:

SELECT DISTINCT JOB_ID FROM MY_EMMLOYEE;

OUTPUT:

The screenshot shows the Oracle APEX SQL Command window. The command entered is:

```
1 SELECT DISTINCT JOB_ID FROM MY_EMMLOYEE;
2
3
```

The results section shows a single row with the value 35 under the column header JOB_ID. The status bar at the bottom indicates "1 rows returned in 0.00 seconds".

JOB_ID
35

1 rows returned in 0.00 seconds Download

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

QUERY:

```
SELECT LAST_NAME||','||JOB_ID AS "EMPLOYEE_AND_TITLE" FROM MY_EMMLOYEE;
```

OUTPUT:

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

- Toolbar:** Includes buttons for SQL Commands, Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run.
- Query Editor:** Displays the following SQL code:

```
1 SELECT LAST_NAME||','||JOB_ID AS "EMPLOYEE_AND_TITLE" FROM MY_EMMLOYEE;
2
3
```
- Results Tab:** Active tab, showing the output of the query.
- Output:** A single row with the value "prasad,35" under the column header "EMPLOYEE_AND_TITLE".
- Timing:** 1 rows returned in 0.01 seconds.
- Buttons:** Explain, Describe, Saved SQL, History, and a Download button.

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 EMPLOYEE_ID||','||FIRST_NAME||','||LAST_NAME||','||EMAIL||','||PHONE_NUMBER||','||  
HIRE_DATE||','||SALARY||','||MANAGER_ID||','||DEPARTMENT_ID AS "THE_OUTPUT" FROM  
MY_EMMLOYEE;
```

OUTPUT:

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

- Toolbar:** Includes buttons for Undo, Redo, Find, Rows (set to 10), Clear Command, Find Tables, Save, and Run.
- Schema:** WKSP_ADHISH13
- SQL Editor:** Displays the query:

```
1 :L||','||PHONE_NUMBER||','||HIRE_DATE||','||SALARY||','||MANAGER_ID||','||DEPARTMENT_ID AS "THE_OUTPUT" FROM MY_EMMLOYEE;  
2  
3
```
- Results Tab:** Active tab, showing the output of the query.
- Output:** A single row with the value: 1,rajesh,prasad,raj@gmail.com,989321875,01400013,1
- Timing:** 1 rows returned in 0.01 seconds
- Actions:** Download button available.

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

RESULT:

RESTRICTING AND SORTING DATA

EX_NO:5

DATE:

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

QUERY:

```
select last_name,salary from MY_EMMLOYEE where salary >12000;
```

OUTPUT:

The screenshot shows a SQL command window and its results. The command entered is:

```
1 SELECT LAST_NAME,SALARY FROM MY_EMMLOYEE WHERE SALARY>12000;
```

The results section displays the output:

LAST_NAME	SALARY
prasad	14000

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

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

QUERY:

```
select last_name,department_id from MY_EMMPLOYEE where employee_id=176;
```

OUTPUT:

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

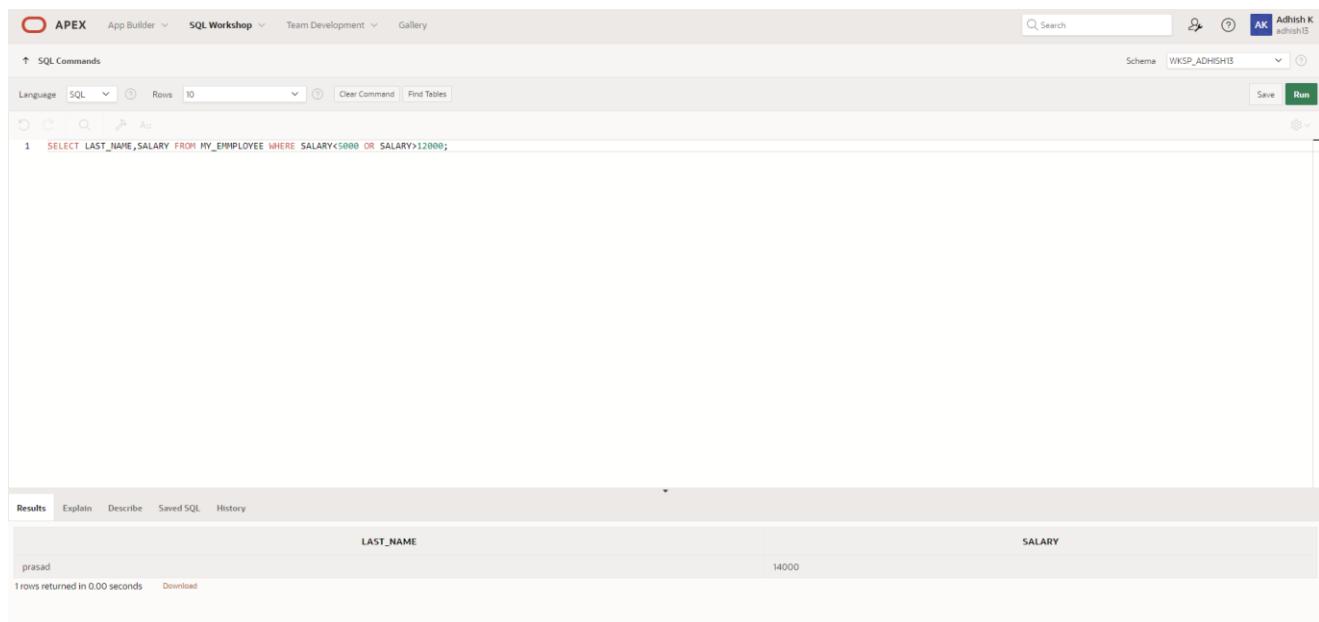
- SQL Commands:** The top bar includes tabs for Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run.
- Query:** The command entered is: `SELECT LAST_NAME, DEPARTMENT_ID FROM MY_EMMPLOYEE WHERE EMPLOYEE_ID=176;`
- Results:** The results tab is selected, displaying the output of the query.
- Output Data:** The result set has two columns: LAST_NAME and DEPARTMENT_ID. One row is returned, showing "prasad" in the LAST_NAME column and "1" in the DEPARTMENT_ID column.
- Timing:** The message "1 rows returned in 0.00 seconds" is displayed at the bottom left.

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

QUERY:

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

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 'Adhish K'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab shows the query: 'SELECT LAST_NAME, SALARY FROM MY_EMMPLOYEE WHERE SALARY<5000 OR SALARY>12000;'. The Results tab displays the output:

LAST_NAME	SALARY
prasad	14000

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

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

QUERY:

```
select last_name,job_id,Hire_Date from MY_EMMLOYEE where Hire_Date between '02/20/1998' and '05/01/1998' order by Hire_date asc;
```

OUTPUT:

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

- SQL Commands Tab:** Contains the SQL query: `select last_name,job_id,hire_date from MY_EMMLOYEE where hire_date between '02/20/1998' and '05/01/1998' order by hire_date asc;`.
- Results Tab:** Displays the query results in a table format.
- Table Headers:** LAST_NAME, JOB_ID, HIRE_DATE
- Table Data:**

LAST_NAME	JOB_ID	HIRE_DATE
prasad	35	04/12/1998
- Message:** 1 rows returned in 0.04 seconds

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

QUERY:

```
select last_name, department_number from MY_EMMLOYEE where department in (20,50) order by last_name asc;
```

OUTPUT:

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

- Header:** SQL Commands, Schema: WKSP_ADHISH13
- Toolbar:** Language (SQL), Rows (10), Clear Command, Find Tables, Save
- Query Editor:** A single line of SQL code: `select last_name, department_id from MY_EMMLOYEE where department_id IN (20,50) order by last_name;`
- Results Tab:** Results (selected), Explain, Describe, Saved SQL, History
- Result Table:** A table with two columns: LAST_NAME and DEPARTMENT_ID. The data row is: prasad, 20.
- Timing:** 1 rows returned in 0.04 seconds
- Download:** A link to download the results.

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

QUERY:

```
select last_name as "EMPLOYEE",salary as "MONTHLY SALARY" from MY_EMMLOYEE where (salary between 5000 and 12000) and (department_id in(20,50)) order by last_name asc;
```

OUTPUT:

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

- Language:** SQL
- Rows:** 10
- Schema:** WKSP_ADHISH15
- SQL Command:**

```
1 select last_name as "EMPLOYEE",salary as "MONTHLY SALARY" from MY_EMMLOYEE where (salary between 5000
2 and 12000) and (department_id in(20,50)) order by last_name asc;
```
- Results:** The results are displayed in a table with two columns: EMPLOYEE and MONTHLY SALARY. The single row returned is VARUN with a salary of 8000.
- Timing:** 1 rows returned in 0.00 seconds.

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

QUERY:

```
select last_name, hire_date from MY_EMMPLOYEE where hire_date like '%94';
```

OUTPUT:

The screenshot shows a SQL command entry field and a results table.

SQL Commands:

```
1 select last_name, hire_date from MY_EMMPLOYEE where hire_date like '%94';
```

Results:

LAST_NAME	HIRE_DATE
aadhithya	03/26/1994

1 rows returned in 0.00 seconds Download

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

QUERY:

```
SELECT last_name, job_id FROM MY_EMMPLOYEE WHERE manager_id IS NULL;
```

OUTPUT:

The screenshot shows a SQL command window and its results. The command window has tabs for 'SQL Commands' and 'Results'. The 'Results' tab is selected, showing the output of the executed SQL query.

SQL Commands:

```
1 SELECT last_name, job_id
2 FROM MY_EMMPLOYEE
3 WHERE manager_id IS NULL;
```

Results:

LAST_NAME	JOB_ID
sabesh	45

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

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

QUERY:

```
SELECT last_name, salary, commisson_amt FROM Y_EMMLOYEE WHERE commisson_amt IS NOT NULL ORDER BY salary DESC, commission_pct DESC;
```

OUTPUT:

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

- SQL Commands:** The schema is set to "WKSP_ADHISH13".
- Language:** SQL.
- Rows:** Set to 10.
- Query:**

```
1 SELECT last_name, salary, commisson_amt
2 FROM MY_EMMLOYEE
3 WHERE commisson_amt IS NOT NULL
4 ORDER BY salary DESC, commisson_amt DESC;
```
- Results:** The results table has columns: LAST_NAME, SALARY, and COMMISSON_AMT. One row is displayed: kumar, 9000, 5000.
- Timing:** 1 rows returned in 0.01 seconds.

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

QUERY:

```
SELECT last_name FROM MY_EMMLOYEE WHERE last_name LIKE '__a%';
```

OUTPUT:

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

- SQL Commands:** The title bar indicates the current schema is "WKSP_ADHISH13".
- Toolbar:** Includes buttons for Language (SQL), Rows (set to 10), Clear Command, Find Tables, Save, and Run.
- Query Editor:** Displays the SQL query:

```
1 SELECT last_name
2 FROM MY_EMMLOYEE
3 WHERE last_name LIKE '__a%';
```
- Results Tab:** Active tab, showing the output of the query.
- Output:** A single row with the value "prasad" under the column "LAST_NAME".
- Message:** "1 rows returned in 0.00 seconds" and a "Download" link.

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

QUERY:

```
SELECT last_name FROM MY_EMMPLOYEE WHERE last_name LIKE '%a%' AND last_name LIKE '%e%';
```

OUTPUT:

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

- SQL Commands:** The query entered is:

```
1 SELECT last_name
2 FROM MY_EMMPLOYEE
3 WHERE last_name LIKE '%a%'
4 AND last_name LIKE '%e%';
```
- Results:** The results table has one column named "LAST_NAME". It contains two rows:

LAST_NAME
naren
sabesh
- Timing:** 2 rows returned in 0.01 seconds.

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

QUERY:

```
select last_name,job_name,salary from MY_EMMLOYEE where job_name in ('hr', 'manager') and salary not in (2500,3500,7000);
```

OUTPUT:

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

- SQL Commands tab is selected.
- Language: SQL.
- Schema: WKSP_ADHISH13.
- Rows: 10.
- Clear Command and Find Tables buttons are available.
- Run button is highlighted.
- SQL Query: `1 select last_name,job_name,salary from MY_EMMLOYEE where job_name in ('hr', 'manager') and salary not in (2500,3500,7000);`
- Results tab is selected.
- Table Headers: LAST_NAME, JOB_NAME, SALARY.
- Table Data:

LAST_NAME	JOB_NAME	SALARY
raj	hr	5000
- Message: 1 rows returned in 0.00 seconds.
- Download link is present.

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

QUERY:

```
select last_name,salary,commisson_amt from MY_EMMLOYEE where commisson_amt=0.2;
```

OUTPUT:

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

- Schema: WKSP_ADHISH13
- Language: SQL
- Rows: 10
- Command: `select last_name,salary,commisson_amt from MY_EMMLOYEE where commisson_amt=0.2;`
- Results tab selected.
- Table output:

LAST_NAME	SALARY	COMMISSON_AMT
VARUN	8000	.2
kumaran	11000	.2

- Message: 2 rows returned in 0.00 seconds
- Download link available.

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

RESULT:

SINGLE ROW FUNCTIONS

EX_NO:6

DATE:

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

QUERY:

Select sysdat as “date” from dual;

OUTPUT:

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

```
1 SELECT SYSDATE AS "DATE" FROM DUAL ;
```

The results window displays the output:

DATE
03/12/2024

Below the results, it says "1 rows returned in 0.00 seconds". The bottom of the page includes copyright information and user details.

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

QUERY:

```
select employee_id, last_name, salary, salary+ (salary*15.5/100) as "NEW SALARY" FROM MY_EMMLOYEE;
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is active. The schema is set to WKSP_ADHISH13. The SQL command entered is:

```
1 select employee_id, last_name, salary, salary+ (salary*15.5/100) as "NEW SALARY" FROM MY_EMMLOYEE;
```

The Results tab is selected, displaying the following output:

EMPLOYEE_ID	LAST_NAME	SALARY	NEW SALARY
256	suban	10000	11550
231	naren	20000	23100
235	raj	5000	5775

At the bottom of the page, there are footer links for 220701013@rajalakshmi.edu.in, adhish13, and en, along with copyright information: Copyright © 1999-2023, Oracle and/or its affiliates. and Oracle APEX 23.2.4.

3. Modify your query lab_03_02.sql to add a column that subtracts the old salary from the new salary. Label the column Increase.

QUERY:

```
select employee_id, last_name, salary, salary+ (salary+ (salary*15.5/100)) as "new salary" , (salary+ (salary*15.5/100))-salary as "Increases" from MY_EMMLOYEE;
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side of the header shows the user 'Adhish K' and the schema 'WKSP_ADHISH13'. The main workspace has tabs for Language (SQL selected), Rows (10), Clear Command, and Find Tables. Below these are icons for Undo, Redo, Search, and Run. The SQL command entered is:

```
1 select employee_id, last_name, salary, salary+ (salary+(salary*15.5/100)) as "new salary", (salary+(salary*15.5/100))-salary as "Increases" from MY_EMMLOYEE;
```

The Results tab is active, displaying the query output:

EMPLOYEE_ID	LAST_NAME	SALARY	new salary	Increases
256	suban	10000	21550	11550
231	naren	20000	43100	23100
235	raj	5000	10775	5775

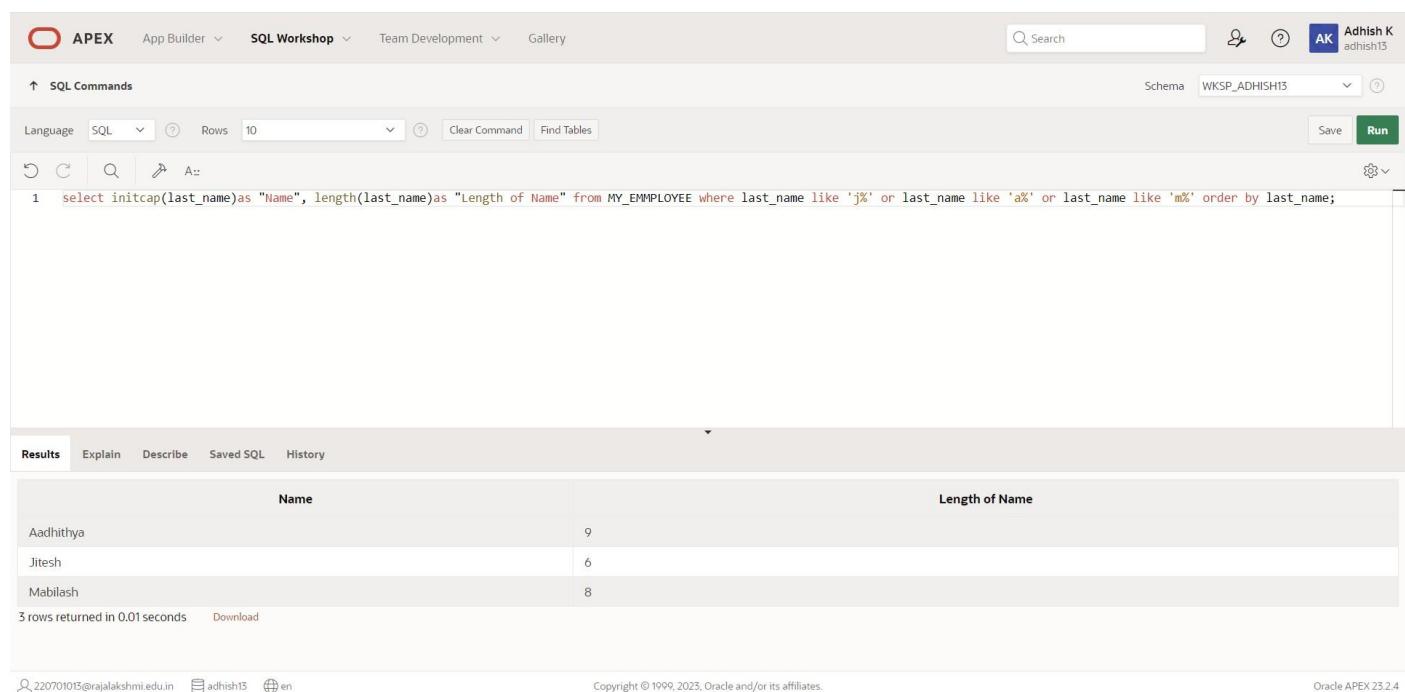
At the bottom of the page, there are footer links for copyright information and Oracle APEX version 23.2.4.

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

QUERY:

```
select initcap(last_name)as "Name", length(last_name)as "Length of Name" from MY_EMPLOYEE where last_name like 'J%' or last_name like 'a%' or last_name like 'm%' order by last_name;
```

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, a user profile for 'Adhish K adhish13', and a 'Run' button. The main workspace is titled 'SQL Commands' and contains the following SQL code:

```
1 select initcap(last_name)as "Name", length(last_name)as "Length of Name" from MY_EMPLOYEE where last_name like 'j%' or last_name like 'a%' or last_name like 'm%' order by last_name;
```

Below the code, the results tab is selected, displaying the output:

Name	Length of Name
Aadhithya	9
Jitesh	6
Mabilash	8

At the bottom, it shows '3 rows returned in 0.01 seconds' and a 'Download' link. The footer includes copyright information for Oracle and the APEX version.

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

QUERY:

```
select last_name from MY_EMMPLOYEE where last_name like 'a%';
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Commands interface. The query `select last_name from MY_EMMPLOYEE where last_name like 'a%';` is entered in the command field. The results pane shows a single row with the value `aadhithya`.

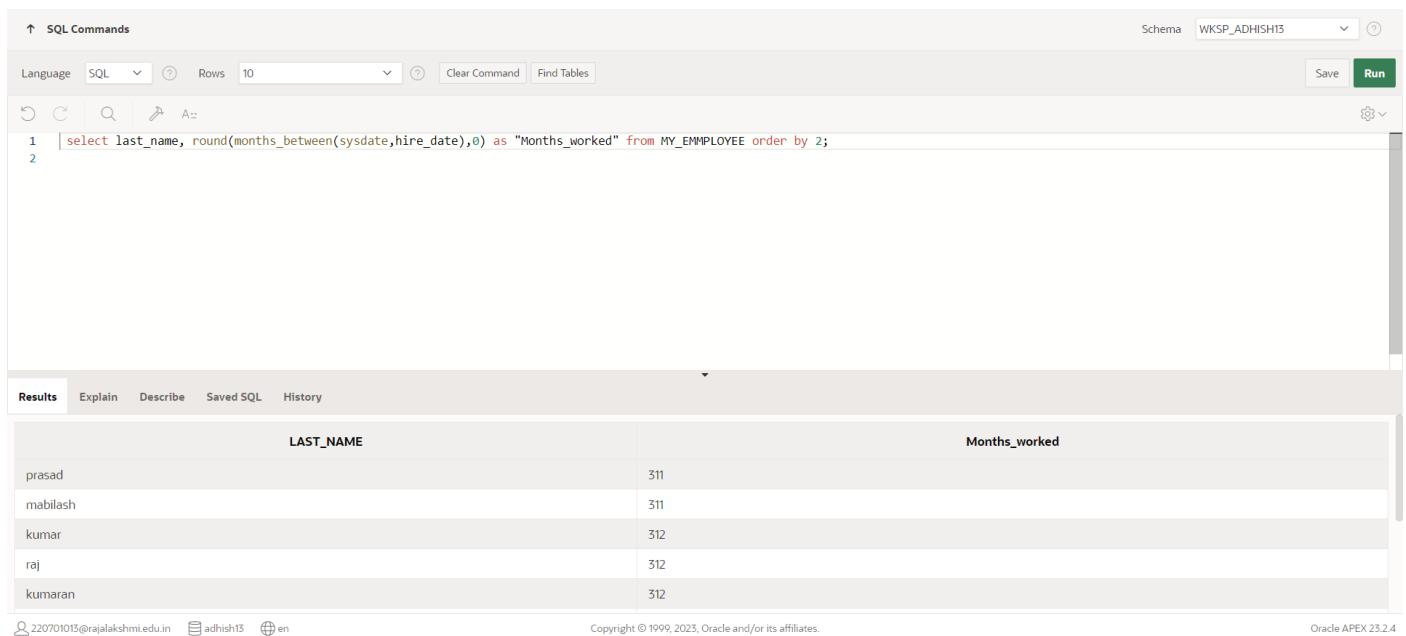
LAST_NAME
aadhithya

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

QUERY:

```
select last_name, round(months_between(sysdate,hire_date),0) as "Months_worked" from MY_EMMPLOYEE  
order by 2;
```

OUTPUT:



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

```
1 | select last_name, round(months_between(sysdate,hire_date),0) as "Months_worked" from MY_EMMPLOYEE order by 2;  
2 |
```

The Results tab displays the output:

LAST_NAME	Months_worked
prasad	311
mabilash	311
kumar	312
raj	312
kumaran	312

At the bottom, the footer includes:

220701015@rajalakshmi.edu.in adhish13 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

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

QUERY:

```
select last_name||' earns $'||salary||' monthly but wants $'||salary*3 as "Dream salary" from MY_EMPLOYEE;
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Worksheet interface. The top section is titled 'SQL Commands' with tabs for Language (set to SQL), Rows (set to 10), and a 'Run' button. The SQL command entered is:

```
1 select last_name||' earns $'||salary||' monthly but wants $'||salary*3 as "Dream Salary" from MY_EMPLOYEE;
2
3
```

The bottom section is titled 'Results' and displays the output of the query:

Dream Salary
suban earns \$10000 monthly but wants \$30000
naren earns \$20000 monthly but wants \$60000
raj earns \$5000 monthly but wants \$15000
aadhiithya earns \$6000 monthly but wants \$18000
VARUN earns \$8000 monthly but wants \$24000

At the bottom of the page, there are footer links for 220701013@rajalakshmi.edu.in, adhish15, en, Copyright © 1999, 2023, Oracle and/or its affiliates., and Oracle APEX 23.2.4.

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

QUERY:

```
select last_name, lpad(salary, 15, '$') Salary from MY_EMMPLOYEE;
```

OUTPUT:

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

```
1 | select last_name, lpad(salary,15,'$') Salary from MY_EMMPLOYEE;
2 |
```

The Results tab displays the output:

LAST_NAME	SALARY
suban	\$\$\$\$\$\$\$\$\$\$10000
naren	\$\$\$\$\$\$\$\$\$\$20000
raj	\$\$\$\$\$\$\$\$\$\$5000
aadhithya	\$\$\$\$\$\$\$\$\$\$6000
VARUN	\$\$\$\$\$\$\$\$\$\$8000

At the bottom, there are footer links: 220701015@rajalakshmi.edu.in, adhish13, en, Copyright © 1999, 2023, Oracle and/or its affiliates, and Oracle APEX 23.2.4.

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

QUERY:

```
select last_name, hire_date, to_char(hire_date, 'day')as day from MY_EMMLOYEE order by to_char(hire_date, 'day');
```

OUTPUT:

The screenshot shows a SQL command entered into a query editor and the resulting table output.

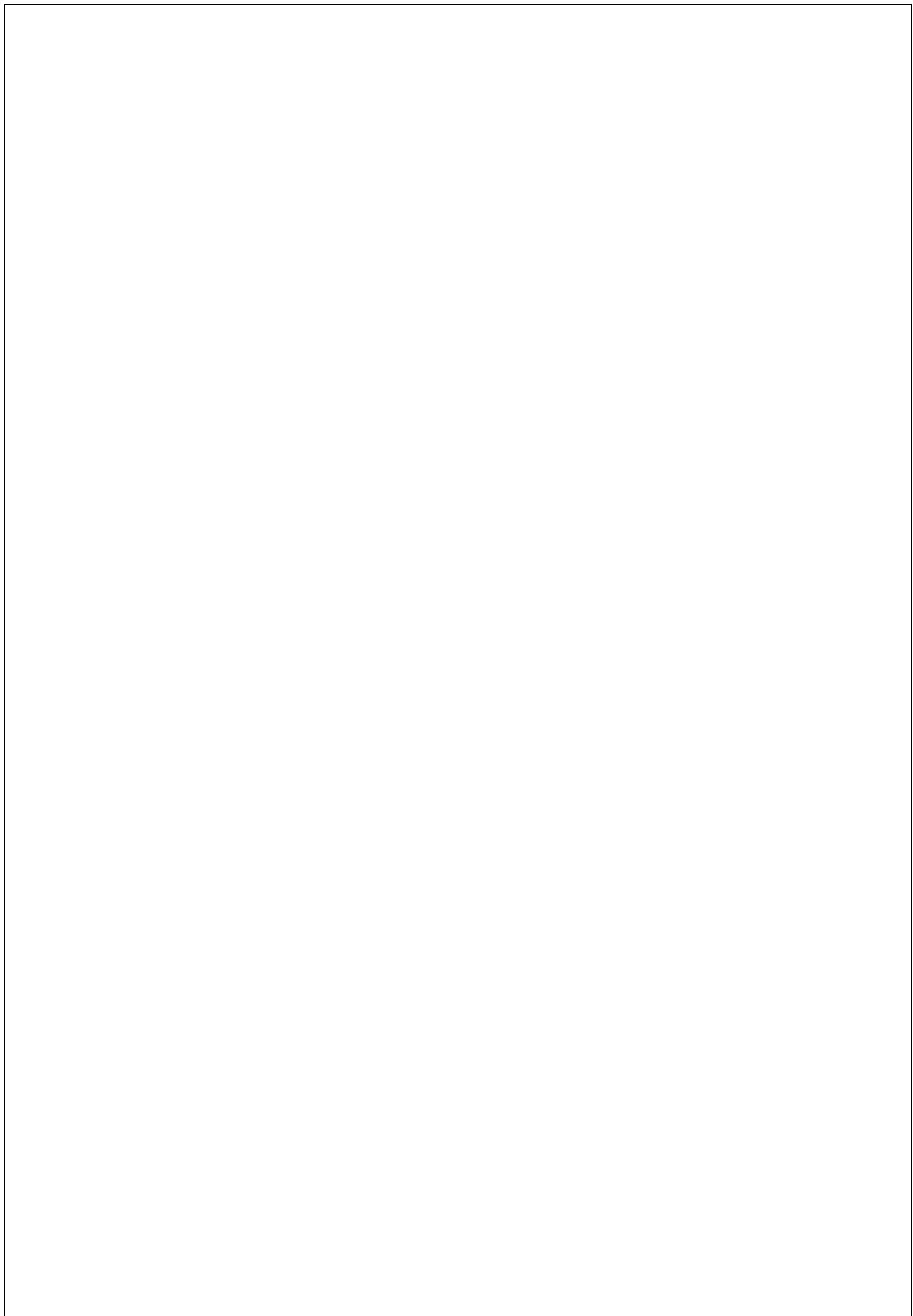
SQL Commands:

```
1 select last_name, hire_date, to_char(hire_date, 'day')as day from MY_EMMLOYEE order by to_char(hire_date, 'day');
```

Results:

LAST_NAME	HIRE_DATE	DAY
naren	02/02/1998	monday
raj	03/21/1998	saturday
suban	02/12/1998	thursday

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



DISPLAYING DATA FROM MULTIPLE DATA

EX_NO:7

DATE:

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

QUERY:

```
select last_name, department_id, DEPARTMENT_NAME from MY_EMMPLOYEE;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows a user profile for 'Adhish K' (adhish13). The main area has tabs for SQL Commands, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, showing the query: 'select last_name,department_id,DEPARTMENT_NAME from MY_EMMPLOYEE;'. The Results tab is selected, displaying the output:

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
suban	27	software
naren	76	development
raj	78	design

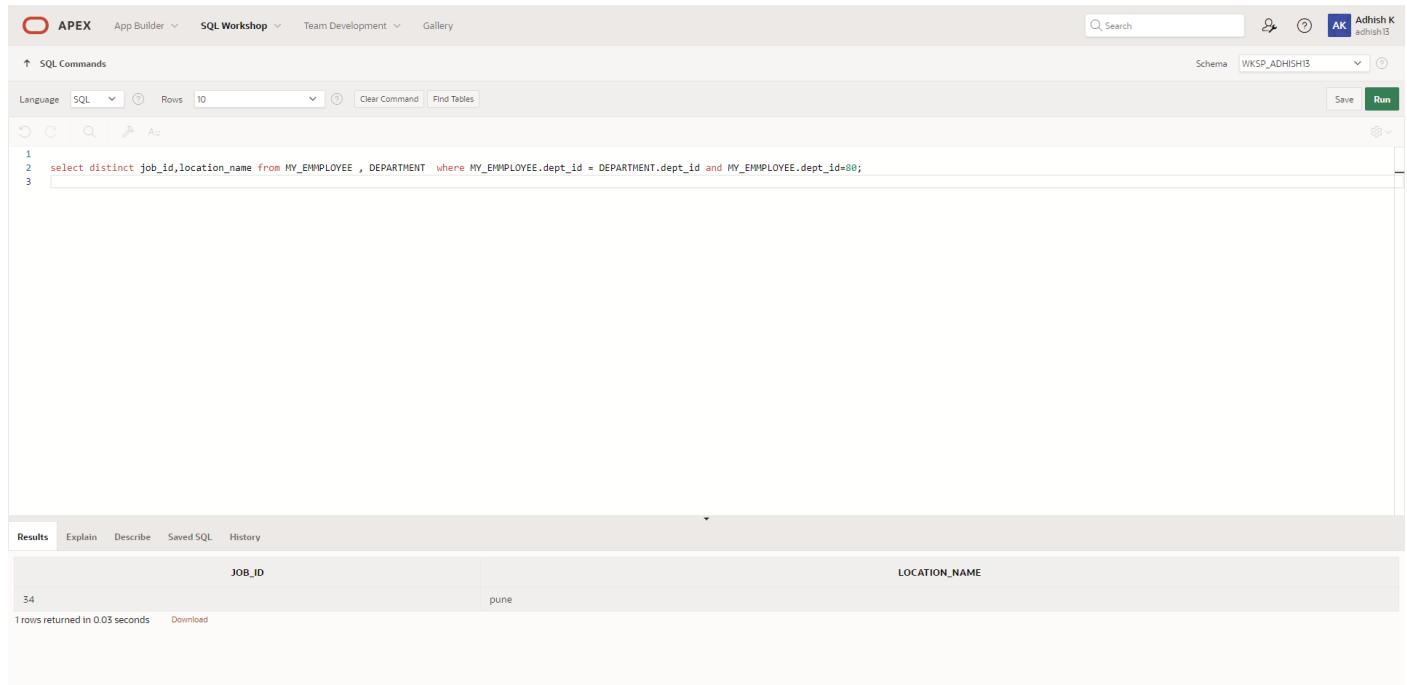
At the bottom, it says '3 rows returned in 0.00 seconds' and has a 'Download' link.

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

QUERY:

```
select distinct job_id, location_name from MY_EMMLOYEE, DEPARTMENT where  
MY_EMMLOYEE.dept_id = DEPARTMENT.dept_id and MY_EMMLOYEE.dept_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 'Adhish K' (adhish15). The main area has tabs for SQL Commands, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, showing the following code:

```
1 select distinct job_id,location_name from MY_EMMLOYEE , DEPARTMENT where MY_EMMLOYEE.dept_id = DEPARTMENT.dept_id and MY_EMMLOYEE.dept_id=80;  
2  
3
```

The results tab displays the output of the query:

JOB_ID	LOCATION_NAME
34	pune

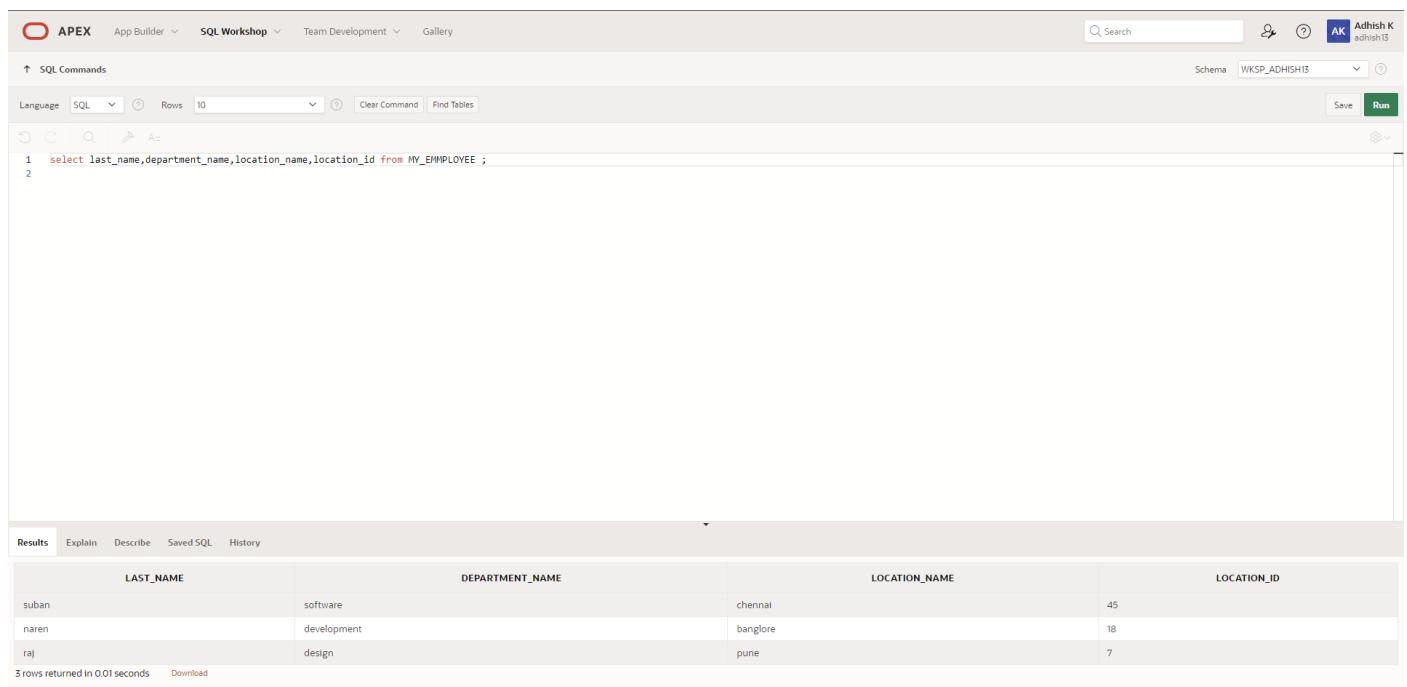
1 rows returned in 0.03 seconds [Download](#)

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

QUERY:

```
select last_name, department_name, location_name, location_id from MY_EMMPLOYEE;
```

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 'Adhish K' (adhish13). The main area has tabs for SQL Commands, SQL (selected), Rows (10), Clear Command, Find Tables, Save, and Run. The SQL command entered is:

```
1 select last_name,department_name,location_name,location_id from MY_EMMPLOYEE ;
2
```

The Results tab is selected, displaying the query results:

LAST_NAME	DEPARTMENT_NAME	LOCATION_NAME	LOCATION_ID
suban	software	chennai	45
naren	development	banglore	18
raj	design	pune	7

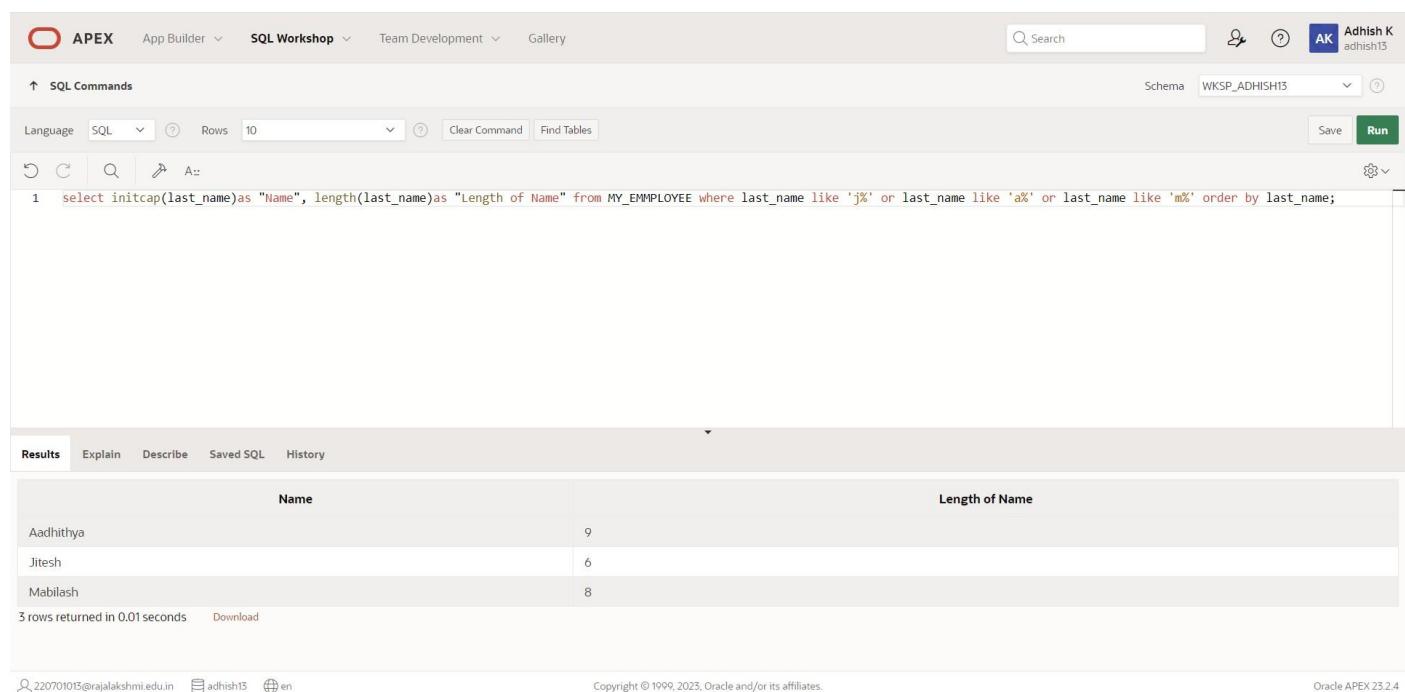
Below the table, it says '3 rows returned in 0.01 seconds' and there is a 'Download' link.

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 initcap(last_name)as "Name", length(last_name)as "Length of Name" from MY_EMPLOYEE where last_name like 'J%' or last_name like 'a%' or last_name like 'm%' order by last_name;
```

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, a user profile for 'Adhish K adhish13', and a 'Run' button. The main workspace is titled 'SQL Commands' and contains the following SQL code:

```
1 select initcap(last_name)as "Name", length(last_name)as "Length of Name" from MY_EMPLOYEE where last_name like 'j%' or last_name like 'a%' or last_name like 'm%' order by last_name;
```

Below the code, the 'Results' tab is selected, displaying the query results:

Name	Length of Name
Aadhithya	9
Jitesh	6
Mabilash	8

At the bottom of the results table, it says '3 rows returned in 0.01 seconds' and has a 'Download' link. The footer of the page includes copyright information for Oracle and links for user profile, schema, and help.

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

QUERY:

```
select last_name from MY_EMMPLOYEE where last_name like 'a%';
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Commands interface. The query `select last_name from MY_EMMPLOYEE where last_name like 'a%';` is entered in the command field. The results pane shows a single row with the value `aadhithya`.

LAST_NAME
aadhithya

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

QUERY:

```
select last_name, round(months_between(sysdate,hire_date),0) as "Months_worked" from MY_EMMPLOYEE  
order by 2;
```

OUTPUT:

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

```
1 | select last_name, round(months_between(sysdate,hire_date),0) as "Months_worked" from MY_EMMPLOYEE order by 2;  
2 |
```

The Results tab displays the output:

LAST_NAME	Months_worked
prasad	311
mabilash	311
kumar	312
raj	312
kumaran	312

At the bottom, the footer includes:

220701015@rajalakshmi.edu.in adhish13 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

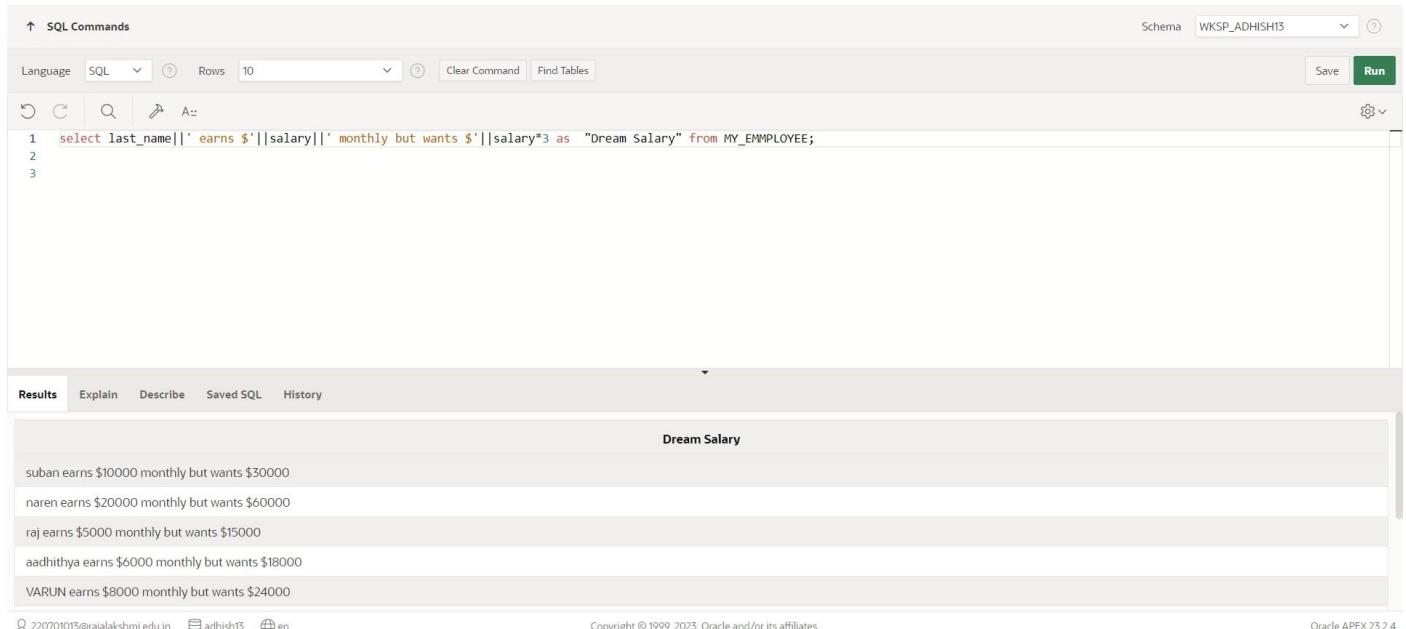
7. Create a report that produces the following for each employee:

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

QUERY:

```
select last_name||' earns $'||salary||' monthly but wants $'||salary*3 as "Dream salary" from MY_EMPLOYEE;
```

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'SQL Commands', 'Language' (set to SQL), 'Rows' (set to 10), 'Clear Command', 'Find Tables', 'Run' button, and 'Schema' (set to WKSP_ADISH13). Below the toolbar, there are icons for Undo, Redo, Find, Replace, and Paste. The main area contains the SQL query:

```
1 select last_name||' earns $'||salary||' monthly but wants $'||salary*3 as "Dream Salary" from MY_EMPLOYEE;
2
3
```

Below the query, the 'Results' tab is selected. The output is displayed in a table with a single column labeled 'Dream Salary'. The data rows are:

Dream Salary
suban earns \$10000 monthly but wants \$30000
naren earns \$20000 monthly but wants \$60000
raj earns \$5000 monthly but wants \$15000
aadhiithya earns \$6000 monthly but wants \$18000
VARUN earns \$8000 monthly but wants \$24000

At the bottom of the page, there are footer links: '220701013@rajalakshmi.edu.in', 'adish13', 'en', 'Copyright © 1999, 2023, Oracle and/or its affiliates.', and 'Oracle APEX 23.2.4'.

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

QUERY:

```
select last_name, lpad(salary, 15, '$') Salary from MY_EMMPLOYEE;
```

OUTPUT:

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

```
1 | select last_name, lpad(salary,15,'$') Salary from MY_EMMPLOYEE;
2 |
```

The Results tab displays the output:

LAST_NAME	SALARY
suban	\$\$\$\$\$\$\$\$\$\$10000
naren	\$\$\$\$\$\$\$\$\$\$20000
raj	\$\$\$\$\$\$\$\$\$\$5000
aadhithya	\$\$\$\$\$\$\$\$\$\$6000
VARUN	\$\$\$\$\$\$\$\$\$\$8000

At the bottom, the footer includes:

220701015@rajalakshmi.edu.in adhish15 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

QUERY:

```
select last_name, hire_date, to_char(hire_date, 'day')as day from MY_EMMLOYEE order by to_char(hire_date, 'day');
```

OUTPUT:

The screenshot shows a SQL command entered into a query editor and its resulting output table.

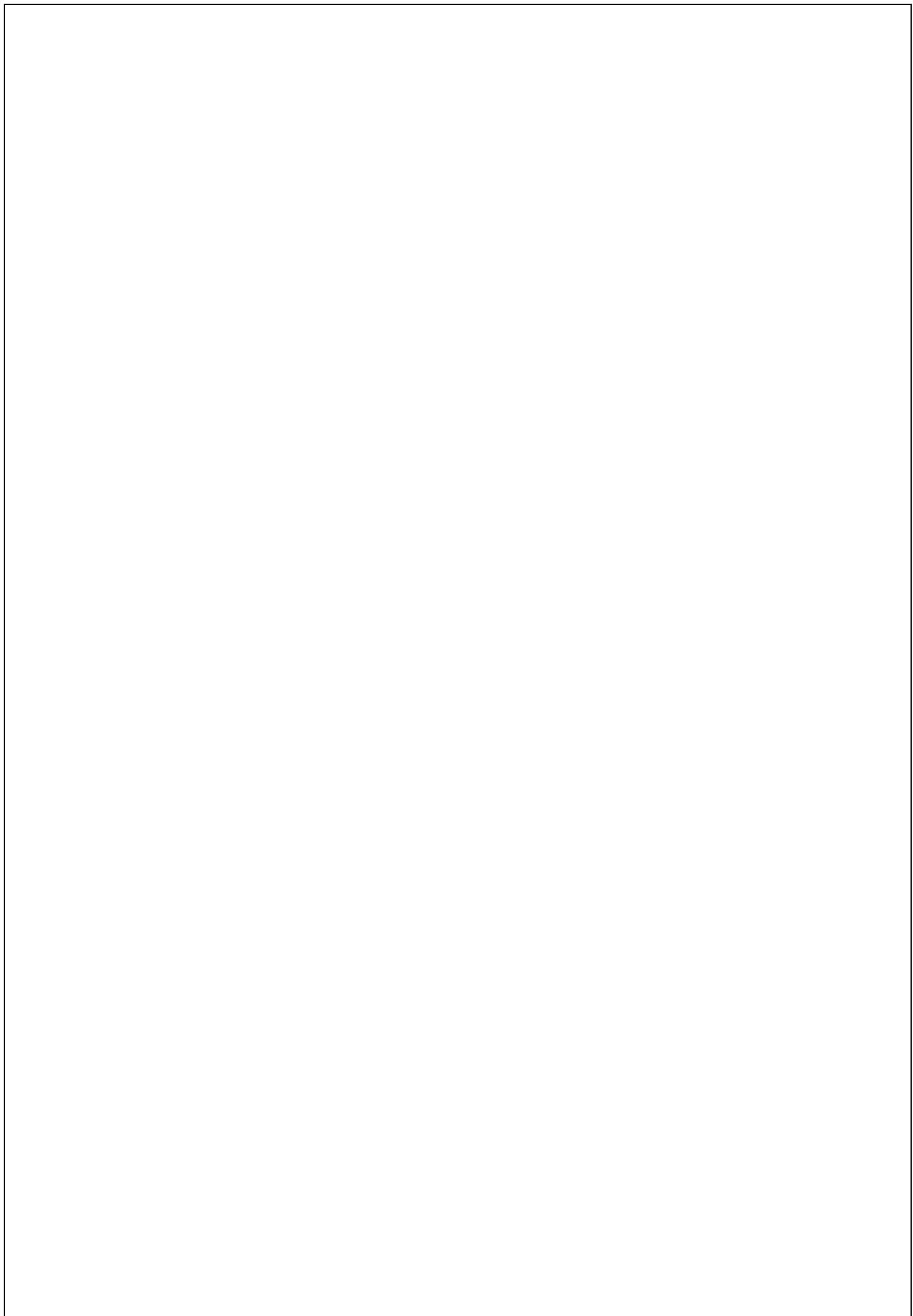
SQL Commands:

```
1 select last_name, hire_date, to_char(hire_date, 'day')as day from MY_EMMLOYEE order by to_char(hire_date, 'day');
```

Results:

LAST_NAME	HIRE_DATE	DAY
naren	02/02/1998	monday
raj	03/21/1998	saturday
suban	02/12/1998	thursday

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



AGGREGATING DATA USING GROUP FUNCTIONS

EX_NO : 8

DATE:

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

TRUE

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

FALSE

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

FALSE

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

QUERY:

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

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 'Adhish K adhish13'. The main area has tabs for 'SQL Commands' and 'Results'. In the 'SQL Commands' tab, the query is entered:

```
1 select max(salary) as "MAXIMUM", min(salary)as "MINI",sum(salary) as "Sum",avg(salary)as "Average" from MY_EMMLOYEE;
```

The 'Results' tab displays the output:

MAXIMUM	MINI	Sum	Average
20000	5000	49000	9800

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

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 'Adhish K' and the schema 'WKSP_ADHISH13'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, displaying the following SQL code:

```
1 select min(salary),max(salary),sum(salary),avg(salary),job_name from MY_EMMLOYEE group by job_name;
```

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

MIN(SALARY)	MAX(SALARY)	SUM(SALARY)	AVG(SALARY)	JOB_NAME
20000	20000	20000	20000	It manager
5000	10000	15000	7500	hr
6000	6000	6000	6000	manager
8000	8000	8000	8000	team lead

Below the table, it says '4 rows returned in 0.01 seconds' and there is a 'Download' link.

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

QUERY:

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

OUTPUT:

The screenshot shows a SQL command window and its results. The command window has tabs for SQL Commands, Explain, Describe, Saved SQL, and History. The SQL tab is active, showing the query: `select job_id, count(*) from MY_EMPLOYEE where job_id='47' group by job_id ;`. The results window shows a single row with columns `JOB_ID` and `COUNT(*)`, containing the values 47 and 1 respectively. A message at the bottom indicates 1 row was returned in 0.01 seconds.

JOB_ID	COUNT(*)
47	1

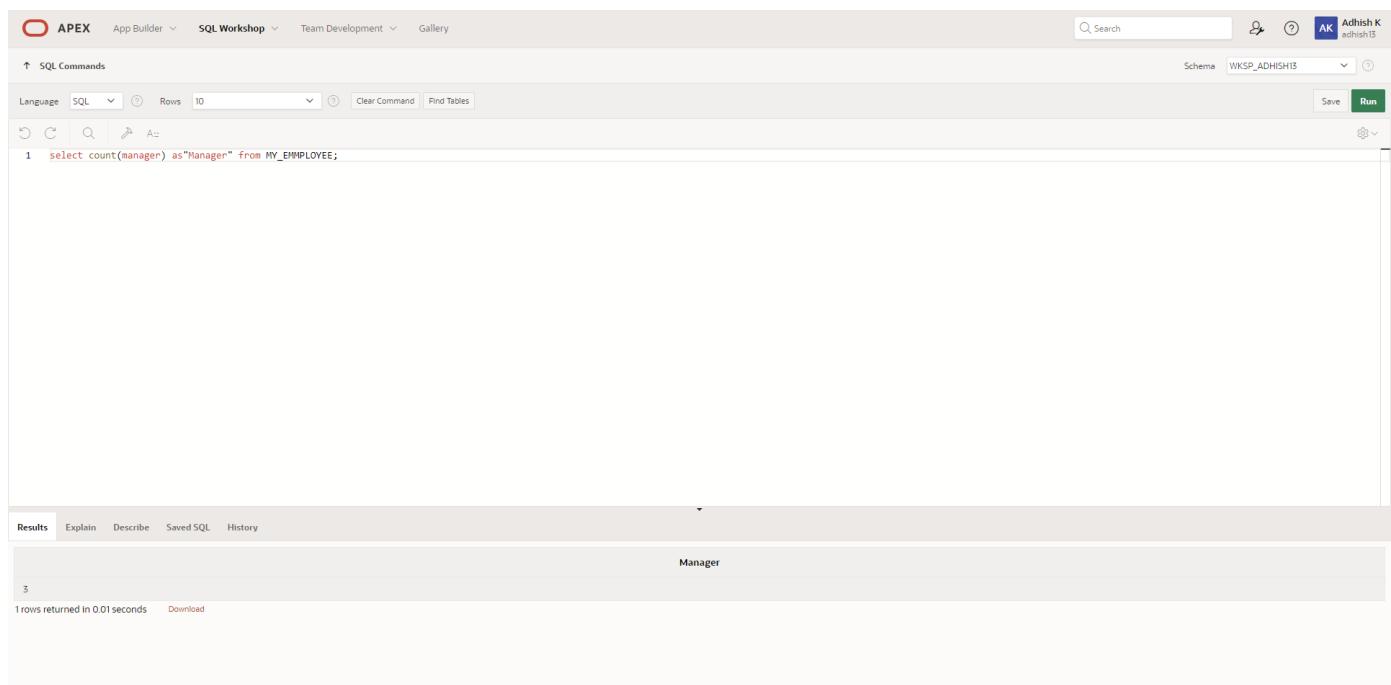
1 rows returned in 0.01 seconds [Download](#)

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected, along with 'SQL Workshop'. The schema is set to 'WKSP_ADHISH13'. The SQL command entered is:

```
1 select count(manager) as"Manager" from MY_EMMPLOYEE;
```

In the results section, there is one row returned, labeled 'Manager' with the value '3'. The status message at the bottom indicates '1 rows returned in 0.01seconds'.

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

QUERY:

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

OUTPUT:

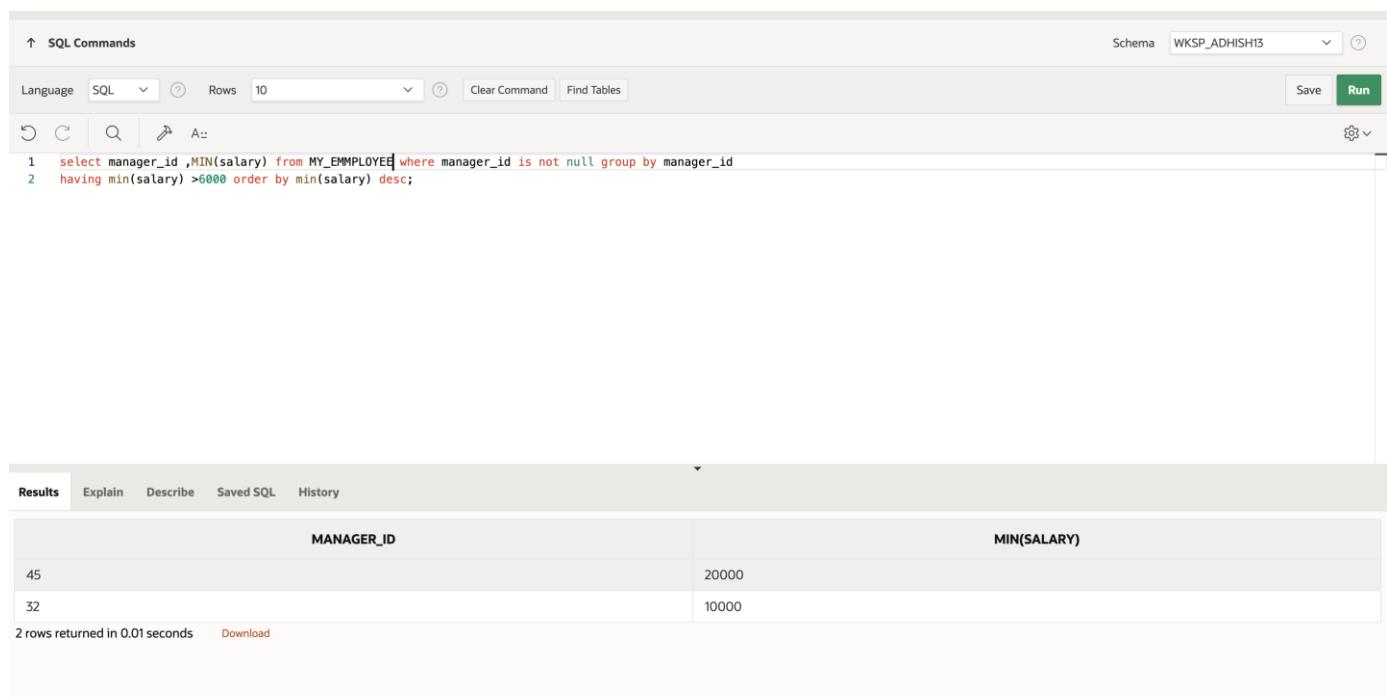
The screenshot shows a SQL command window with the following details:

- SQL Commands:** The query is displayed: `select max(salary)-min(salary) as "DIFFERENCE" from MY_EMMLOYEE;`
- Results:** The results pane shows a single row with the heading "DIFFERENCE" and the value "15000".
- Timing:** The message "1 rows returned in 0.01 seconds" is visible.

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

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

OUTPUT:



The screenshot shows a SQL query editor interface. The top bar includes tabs for 'SQL Commands' (selected), 'Language' (set to 'SQL'), 'Rows' (set to 10), and buttons for 'Clear Command', 'Find Tables', 'Save', and 'Run'. The main area contains the following SQL code:

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

Below the code, the 'Results' tab is selected, showing a table with two rows. The table has columns 'MANAGER_ID' and 'MIN(SALARY)'. The data is as follows:

MANAGER_ID	MIN(SALARY)
45	20000
32	10000

At the bottom left, it says '2 rows returned in 0.01 seconds'. There is also a 'Download' link.

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle SQL developer interface. The top section is the SQL Commands editor with the following details:

- Language: SQL
- Rows: 10
- Schema: WKSP_ADHISH13
- Buttons: Save, Run

The SQL command entered is:

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

The bottom section is the Results grid, which displays the following data:

	TOTAL	1995	1996	1997	1998
4		1	1	1	1

Below the grid, it says "1 rows returned in 0.01 seconds" and has a "Download" link.

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle SQL developer interface. The top section is the SQL Commands editor with the following details:

- Language: SQL
- Rows: 10
- Schema: WKSP_ADHISH13
- Buttons: Save, Run

The SQL command entered is:

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

The bottom section is the Results grid, which displays the following data:

job	Dept20	dept50	dept80	dept90	TOTAL
64	-	-	-	-	6000
34	-	-	5000	-	5000
19	-	-	-	-	10000
47	-	-	-	-	20000

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

OUTPUT:

The screenshot shows a SQL command window and a results table. The command window has tabs for 'SQL Commands' and 'Results'. The 'SQL Commands' tab contains the following SQL code:

```
1 select dept.dept_name as "department_name",dept.location_city as "department location", count(*) "Number of people",round(avg(salary),2) "salary" from dept inner join MY_EMPLOYEE on(dept.department_id =MY_EMPLOYEE.dept_id ) group by dept.dept_name ,dept.location_city;
```

The 'Results' tab displays the output of the query:

department_name	department location	Number of people	salary
sales	chennai	1	10000

Below the table, it says '1 rows returned in 0.05 seconds'.

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

RESULT:

SUB QUERIES

EX_NO:9

DATE:

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

QUERY:

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

OUTPUT:

The screenshot shows a SQL query editor interface. The top bar includes 'SQL Commands' and 'Schema WKSP_ADHISH13'. The main area has tabs for 'Language' (set to 'SQL'), 'Rows' (set to 10), and 'Find Tables'. Below these are standard icons for refresh, search, and run. The code area contains two numbered lines of SQL:

```
1 select last_name,hire_date from MY_EMMLOYEE where dept_id=(select dept_id from  
2 MY_EMMLOYEE where last_name='zlotkey') and last_name not in('Zlotkey');
```

At the bottom, there's a 'Results' tab selected, showing a table with two columns: 'LAST_NAME' and 'HIRE_DATE'. The single row returned is 'zlotkey' in the first column and '01/05/1996' in the second. A note at the bottom says '1 rows returned in 0.01 seconds'.

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

QUERY: select employee_id, last_name, salary from MY_EMMLOYEE where salary > (select avg(salary) from my_emmployee) order by salary;

OUTPUT:

The screenshot shows a SQL query editor interface. At the top, there's a toolbar with various icons and a dropdown for 'Schema' set to 'WKSP_ADHISH13'. Below the toolbar is a search bar and a 'Run' button. The main area contains the SQL query:

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

Below the query, the results tab is selected. The results table has three columns: 'EMPLOYEE_ID', 'LAST_NAME', and 'SALARY'. There is one row returned, which is:

EMPLOYEE_ID	LAST_NAME	SALARY
231	naren	20000

At the bottom left of the results table, it says '1 rows returned in 0.01 seconds' and there is 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 MY_EMMLOYEE where department_id=(select  
department_id from MY_EMMLOYEE where last_name like'%u%');
```

OUTPUT:

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

- SQL Commands:** The schema is set to "WKSP_ADHISH13".
- Language:** SQL
- Rows:** 10
- Buttons:** Clear Command, Find Tables, Save, Run
- Query Editor:** Contains the following SQL code:

```
1 select employee_id,last_name from MY_EMMLOYEE where dept_id=(select dept_id  
2 from MY_EMMLOYEE where last_name like '%u%');
```
- Results Tab:** Selected. Shows the output of the query.
- Output Data:** A single row is displayed:

EMPLOYEE_ID	LAST_NAME
256	suban

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

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

QUERY:

```
select last_name,department_id,job_id from MY_EMMLOYEE where dept_id=(select dept_id  
from dept where location_id=1700);
```

OUTPUT:

The screenshot shows a SQL command window and its results. The SQL command is:

```
1 select last_name,dept_id,job_id from MY_EMMLOYEE where dept_id=(select dept_id from  
2 dept where location_id=1700);
```

The results table has three columns: LAST_NAME, DEPT_ID, and JOB_ID. The data is:

LAST_NAME	DEPT_ID	JOB_ID
suban	27	19
zlotkey	87	64
naren	76	47
raj	80	34

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

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

QUERY:

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

OUTPUT:

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

- SQL Commands:** The top bar includes "SQL Commands", "Schema: WKSP_ADHISH13", and a "Run" button.
- Query:** The query entered is:

```
1 select last_name,salary from MY_EMMLOYEE where manager_id=(select manager_id from  
2 MY_EMMLOYEE where manager_name='King');
```
- Results:** The results tab is selected, showing the output of the query:

LAST_NAME	SALARY
naren	20000

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

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

OUTPUT:

The screenshot shows a SQL command window and its results. The SQL command entered is:

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

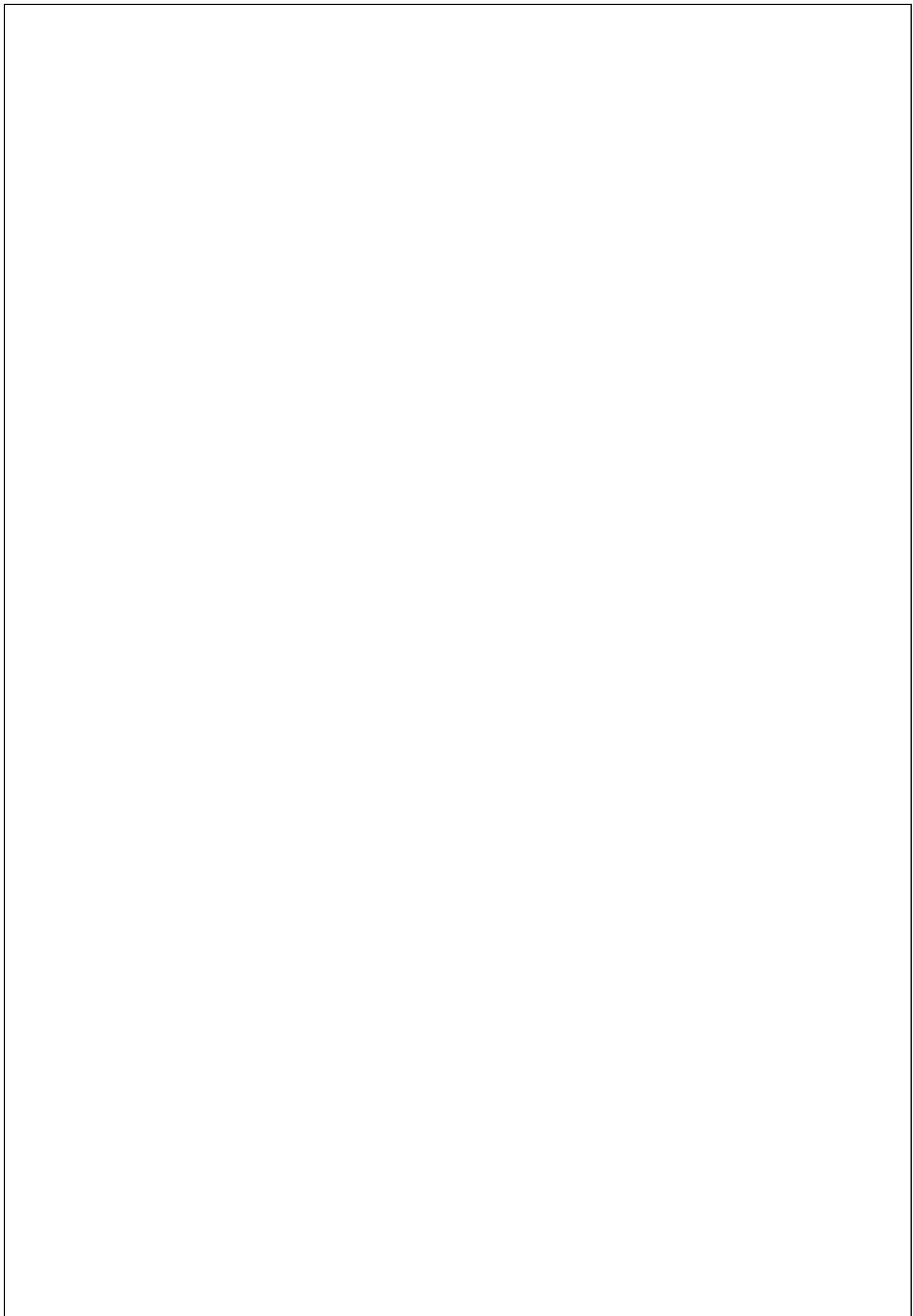
The results section displays a table with three columns: EMPLOYEE_ID, LAST_NAME, and SALARY. The single row returned is:

EMPLOYEE_ID	LAST_NAME	SALARY
231	naren	20000

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

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

RESULT:



USING THE SET OPERATORS

EX.NO:10

DATE:

Find the Solution for the following:

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

QUERY:

```
SELECT dept_id FROM dept MINUS SELECT dept_id FROM MY_EMMPLOYEE  
WHERE job_name = 'st_clerk';
```

OUTPUT:

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

- Language:** SQL
- Schema:** WKSP_ADHISH13
- Commands:** SELECT dept_id FROM dept MINUS SELECT dept_id FROM MY_EMMPLOYEE WHERE job_name = 'st_clerk';
- Results:** DEPT_ID
- Output Data:** 27
78
- Timing:** 2 rows returned in 0.01 seconds

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

QUERY: SELECT DISTINCT job_id, dept_id FROM MY_EMMPLOYEE WHERE dept_id = 10 UNION ALL

SELECT DISTINCT job_id, dept_id FROM MY_EMMPLOYEE WHERE dept_id = 50 UNION ALL

SELECT DISTINCT job_id, dept_id FROM MY_EMMPLOYEE WHERE dept_id = 20

OUTPUT:

The screenshot shows a SQL query editor interface with two tabs: 'SQL Commands' and 'Results'. In the 'SQL Commands' tab, the following SQL code is visible:

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

In the 'Results' tab, the output is displayed in a table format:

JOB_ID	DEPT_ID
19	10
64	50
47	20

Below the table, it says "3 rows returned in 0.00 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,dept_id,TO_CHAR(null) FROM MY_EMMLOYEE
UNION SELECT TO_CHAR(null),dept_id,dept_name FROM dept
```

OUTPUT:



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

```
1 SELECT last_name,dept_id,TO_CHAR(null) FROM MY_EMMLOYEE
2 UNION SELECT TO_CHAR(null),dept_id,dept_name FROM dept
```

The results page displays the output of the executed query:

LAST_NAME	DEPT_ID	TO_CHAR(NULL)
naren	20	-
raj	80	-
suban	10	-
zlotkey	50	-
-	27	sales

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

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

RESULT:

CREATING VIEWS

EX NO:11

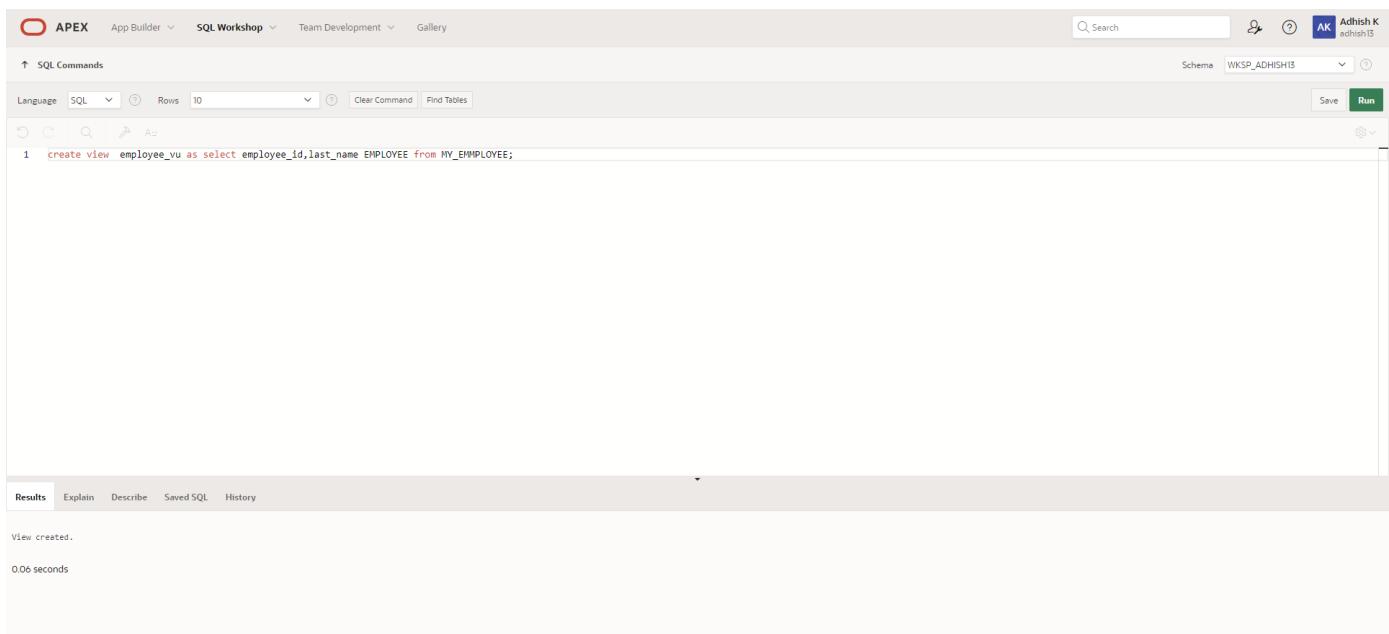
DATE:

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

QUERY:

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

OUTPUT:



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

```
1 create view employee_vu as select employee_id, last_name EMPLOYEE from MY_EMPLOYEE;
```

In the Results pane, the output is displayed as:

```
View created.  
0.06 seconds
```

2.) Display the contents of the EMPLOYEES_VU view.

QUERY:

```
select * from employees_vu;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown shows WKS_P_ADHISH18. The main area contains the SQL command: "select * from employees_vu;". The results section displays the following data:

EMPLOYEE_ID	EMPLOYEE	DEPT_ID
256	suban	10
721	zlotkey	50
231	naren	20
235	raj	80

4 rows returned in 0.04 seconds [Download](#)

3.) Select the view name and text from the USER_VIEWS data dictionary views

QUERY:

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

OUTPUT:

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

QUERY:

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

OUTPUT:

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

- SQL Commands:** The top bar includes "Language: SQL", "Rows: 10", "Clear Command", "Find Tables", "Save", and a "Run" button.
- Query:** The command entered is "select EMMLOYEE,dept_id from employees_vu;".
- Results:** The results tab is selected, displaying the following table:

EMMLOYEE	DEPT_ID
suban	10
zlotkey	50
naren	20
(a)	80

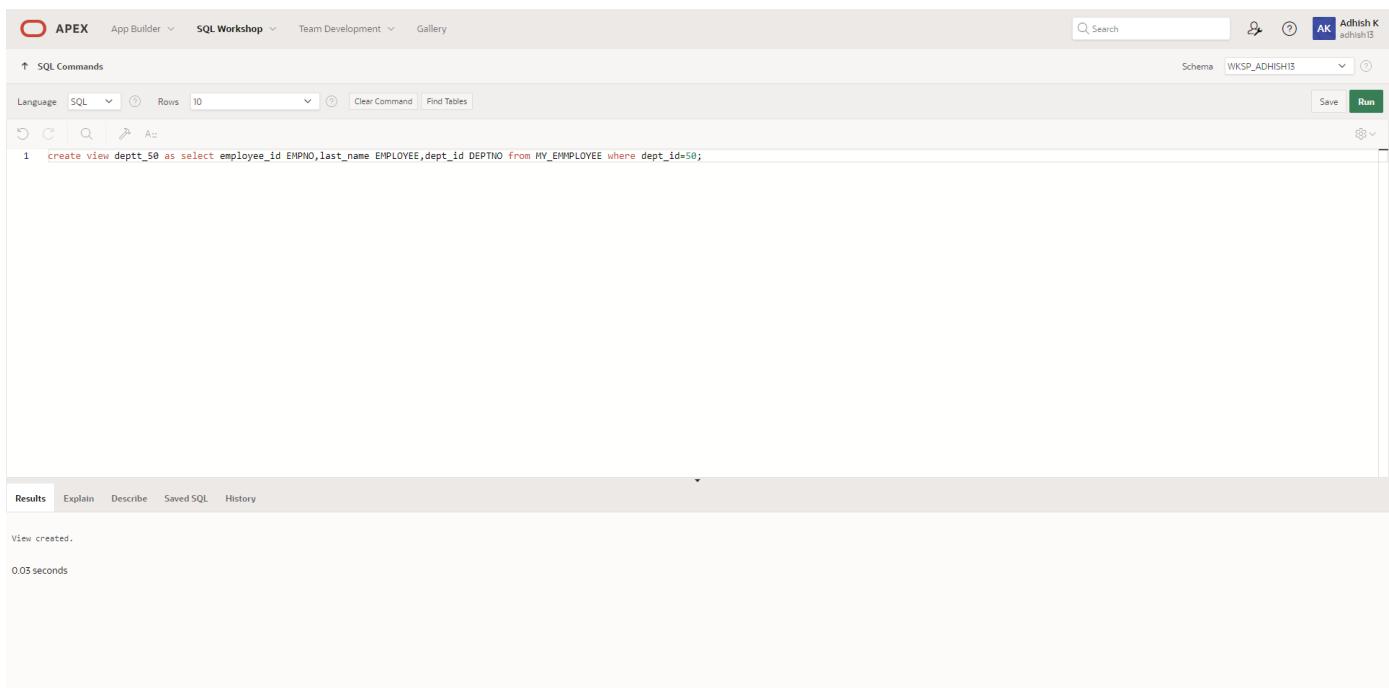
Below the table, it says "4 rows returned in 0.00 seconds" and has a "Download" link.

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

QUERY:

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

OUTPUT:



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

```
1 create view dept_50 as select employee_id EMPNO,last_name EMPLOYEE,dept_id DEPTNO from MY_EMPLOYEE where dept_id=50;
```

In the bottom results pane, the output is shown as:

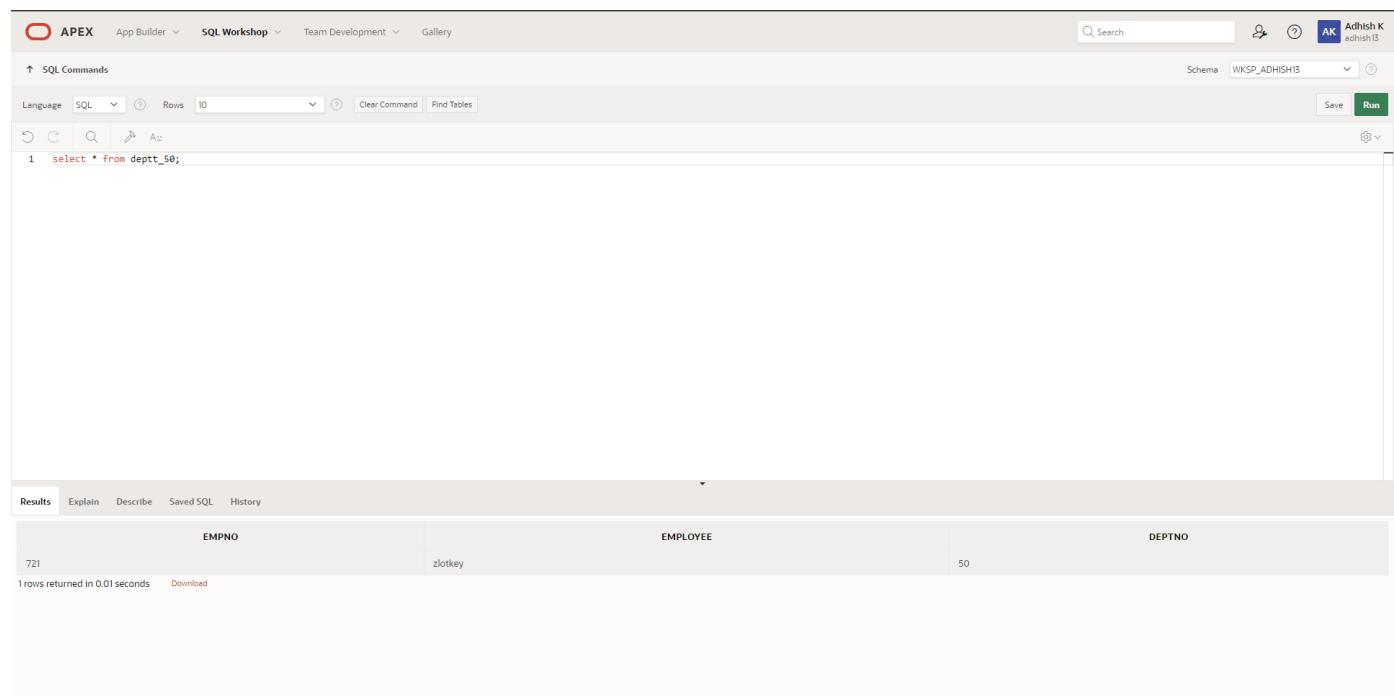
```
View created.  
0.03 seconds
```

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

QUERY:

Describe dept50;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for Adhish K (adhish13). The main area has tabs for SQL Commands, SQL (selected), Clear Command, Find Tables, and Run. The SQL editor contains the command: "1 select * from deptt_50;". Below the editor is a Results tab, which displays the output of the query. The output shows one row with columns EMPNO, EMPLOYEE, and DEPTNO, all containing the value 721. The status bar at the bottom indicates "1 rows returned in 0.01 seconds" and "Download".

EMPNO	EMPLOYEE	DEPTNO
721	zlotkey	50

7.) Attempt to reassign Matos to department 80

QUERY:

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

OUTPUT:

The screenshot shows a SQL command window in Oracle SQL Developer. The command entered is:

```
1 UPDATE deptt_50 set deptno=80 where employee='Matos';
```

The results tab shows the output:

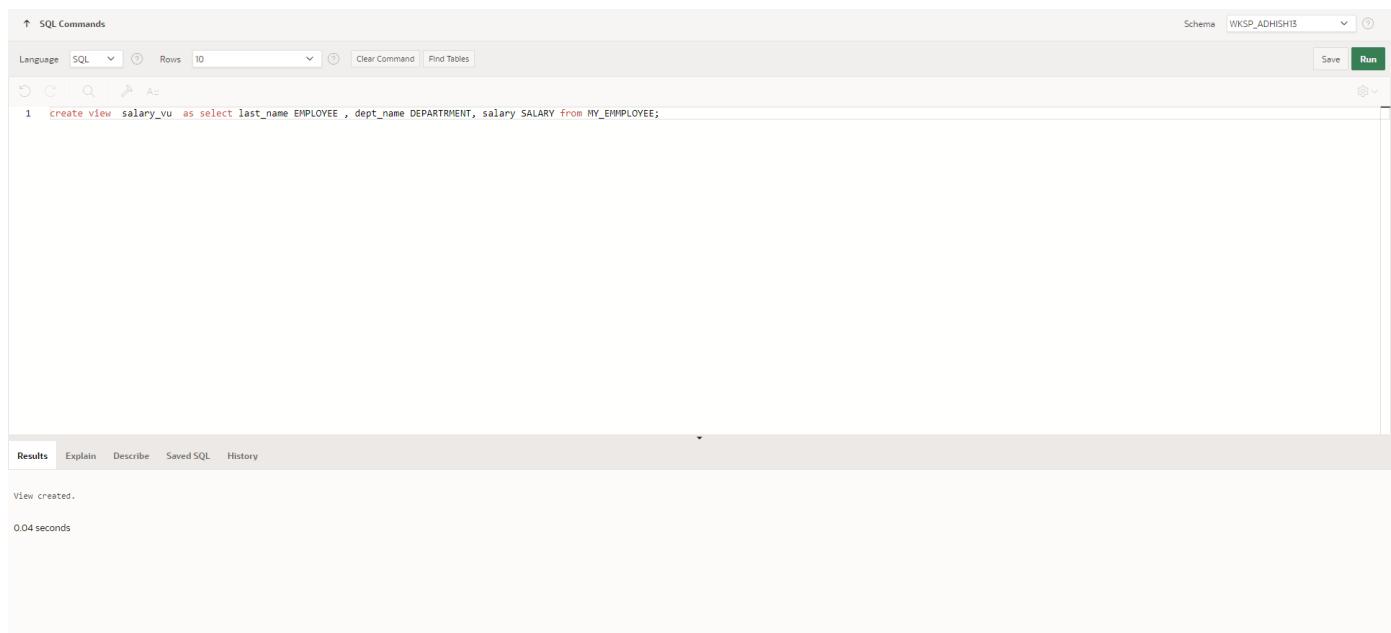
```
0 row(s) updated.  
0.06 seconds
```

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

QUERY:

```
create or replace view salary_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 a SQL command window with the following details:

- Header:** SQL Commands, Schema: WKSP_ADHISH13
- Toolbar:** Language (SQL), Rows (10), Clear Command, Find Tables, Save, Run
- Query Editor:** A single line of SQL code:

```
1 create view salary_vu as select last_name EMPLOYEE , dept_name DEPARTMENT, salary SALARY from MY_EMPLOYEE;
```
- Results Tab:** Shows the output of the query. It includes:
 - Message: View created.
 - Time: 0.04 seconds

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

RESULT:

EXERCISE 12

PRACTICE QUESTIONS

Intro to Constraints; NOT NULL and UNIQUE Constraints

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

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

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

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

2. What are the limitations of constraints that may be applied at the column level and at the table level?
 - Constraints referring to more than one column are defined at Table Level
 - NOT NULL constraint must be defined at column level as per ANSI/ISO SQL standard.
3. Why is it important to give meaningful names to constraints?
 - If a constraint is violated in a SQL statement execution, it is easy to identify the cause with user-named constraints.
 - It is easy to alter names/drop constraint.

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

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

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

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

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

```
CREATE TABLE f_global_locations
(id NUMBER(6,0) CONSTRAINT f_gln_id_pk PRIMARY KEY,
name VARCHAR2(50),
date_opened DATE CONSTRAINT f_gln_dt_opened_nn NOT NULL ENABLE,
address VARCHAR2(50) CONSTRAINT f_gln_add_nn NOT NULL ENABLE,
city VARCHAR2(30) CONSTRAINT f_gln_city_nn NOT NULL ENABLE,
```

```

zip_postal_code VARCHAR2(12),
phone VARCHAR2(20),
email VARCHAR2(75) CONSTRAINT f_gln_email_uk UNIQUE,
manager_id NUMBER(6,0),
emergency_contact VARCHAR2(20)
);

```

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

Table Created.

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

```
DESCRIBE f_global_locations;
```

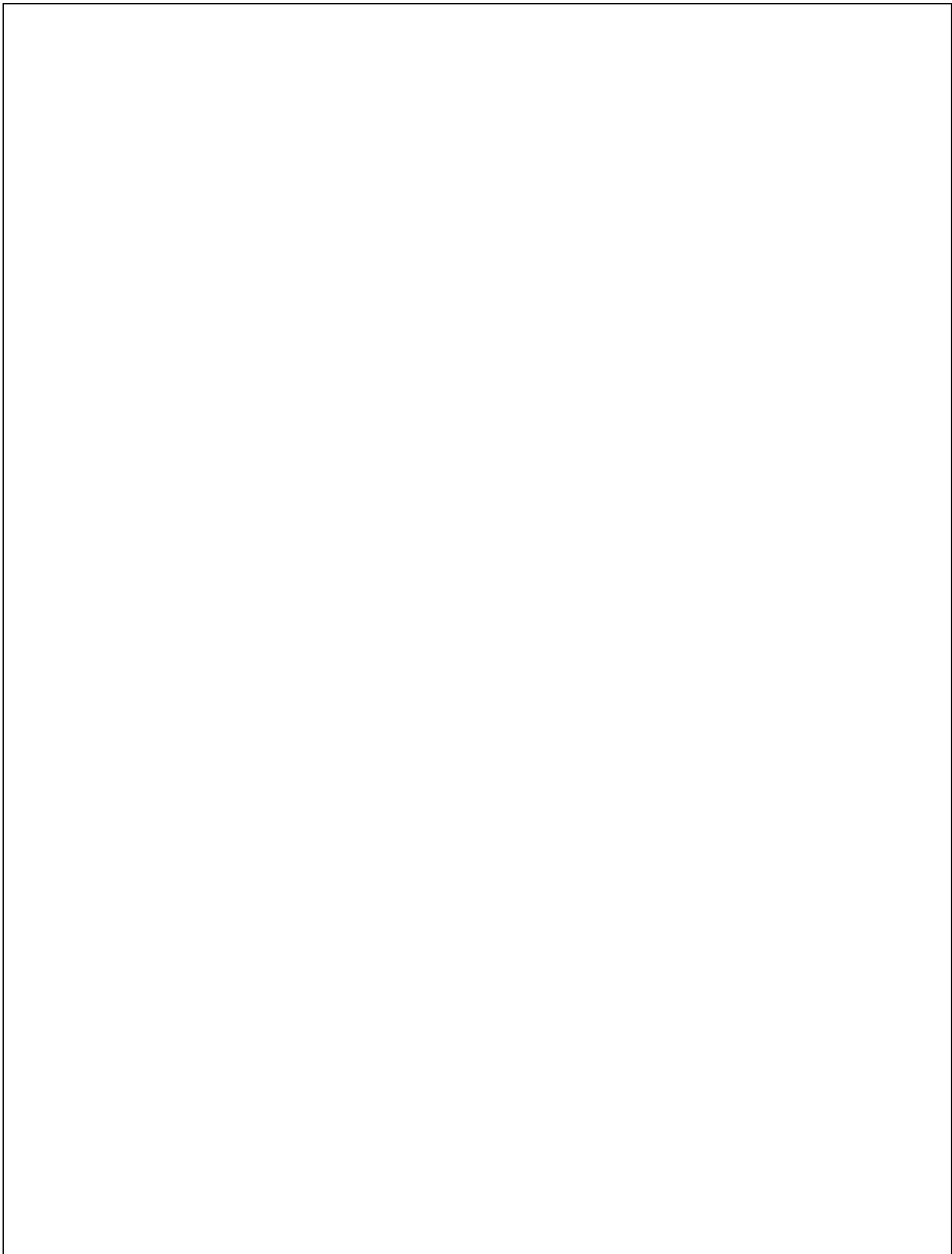
9. Rewrite the CREATE TABLE statement for the Global Fast Foods locations table to define the UNIQUE constraints at the table level. Do not execute this statement.

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

```

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

```



PRIMARY KEY, FOREIGN KEY, and CHECK Constraints

1. What is the purpose of a
 - PRIMARY KEY
 - FOREIGN KEY
 - CHECK CONSTRAINT
 - a. **PRIMARY KEY**
Uniquely identify each row in table.
 - b. **FOREIGN KEY**
Referential integrity constraint links back parent table's primary/unique key to child table's column.
 - c. **CHECK CONSTRAINT**
Explicitly define condition to be met by each row's fields. This condition must be returned as true or unknown.
2. Using the column information for the animals table below, name constraints where applicable at the table level, otherwise name them at the column level. Define the primary key (animal_id). The license_tag_number must be unique. The admit_date and vaccination_date columns cannot contain null values.

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

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

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

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

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

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

YYYY'));

SELECT * FROM animals;

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

COLUMN LEVEL STATEMENT:

```
ALTER TABLE animals
MODIFY ( adoption_id NUMBER(5,0) CONSTRAINT anl_adopt_id_<k REFERENCES adoptions(id)
ENABLE );
```

TABLE LEVEL STATEMENT:

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

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

- a. ON DELETE CASCADE

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

- b. ON DELETE SET NULL

```
ALTER TABLE animals
ADD CONSTRAINT anl_adopt_id_a 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 <ields in self row.
- I cannot use subqueries and scalar subquery expressions.

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

Total (15)	
Faculty Signature	

PRACTICE PROBLEM

Managing Constraints

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

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

1. What are four functions that an ALTER statement can perform on constraints?
 - ADD
 - DROP
 - ENABLE
 - DISABLE

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

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

3. Create a FOREIGN KEY constraint in the copy_d_events table. Name the foreign key copy_d_events_a. This key

references the copy_d_clients table client_number column. What is the syntax you used to create the FOREIGN KEY constraint in the copy_d_events table?

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

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

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

- a. The constraint name for the primary key in the copy_d_clients table is _____.

COPY_D_CLT_CLIENT_NUMBER_PK

5. Drop the PRIMARY KEY constraint on the copy_d_clients table. Explain your results.

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

6. Add the following event to the copy_d_events table. Explain your results.

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

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

RESULT: ORA-02291: integrity constraint (HKUMAR.COPY_D_EVE_CLIENT_NUMBER_FK) violated - parent key not found

7. Create an ALTER TABLE query to disable the primary key in the copy_d_clients table. Then add the values from #6 to the copy_d_events table. Explain your results.

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

8. Repeat question 6: Insert the new values in the copy_d_events table. Explain your results.

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

1 row(s) inserted.

9. Enable the primary-key constraint in the copy_d_clients table. Explain your results.

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

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

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

1 row(s) deleted.

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

Table altered.

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

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

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

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

- C - Check constraint
 - Sub-case - if I see SEARCH_CONDITION something like "FIRST_NAME" IS NOT NULL , its a NOT NULL constraint.
- P - Primary key
- R - Referential integrity (<k>)
- U - Unique key

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

EXERCISE 13

Creating Views

1. What are three uses for a view from a DBA's perspective?
 - Restrict access and display selective columns
 - Reduce complexity of queries from other internal systems. So, providing a way to view same data in a different manner.
 - Let the app code rely on views and allow the internal implementation of tables to be modified later.
2. Create a simple view called view_d_songs that contains the ID, title and artist from the DJs on Demand table for each "New Age" type code. In the subquery, use the alias "Song Title" for the title column.

CREATE VIEW view_d_songs AS

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

3. SELECT * FROM view_d_songs. What was returned?

Results	Explain	Describe	Saved SQL	History									
<table border="1"><thead><tr><th>ID</th><th>Song Title</th><th>ARTIST</th></tr></thead><tbody><tr><td>47</td><td>Hurrah for Today</td><td>The Jubilant Trio</td></tr><tr><td>49</td><td>Lets Celebrate</td><td>The Celebrants</td></tr></tbody></table>					ID	Song Title	ARTIST	47	Hurrah for Today	The Jubilant Trio	49	Lets Celebrate	The Celebrants
ID	Song Title	ARTIST											
47	Hurrah for Today	The Jubilant Trio											
49	Lets Celebrate	The Celebrants											
2 rows returned in 0.00 seconds				Download									

4. REPLACE view_d_songs. Add type_code to the column list. Use aliases for all columns.

Or use alias after the CREATE statement as shown.

CREATE OR REPLACE VIEW view_d_songs AS

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

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

CREATE OR REPLACE VIEW view_d_events_pkgs AS

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

6. It is company policy that only upper-level management be allowed access to individual employee salaries.

The department managers, however, need to know the minimum, maximum, and average salaries, grouped by department. Use the Oracle database to prepare a view that displays the needed information for department managers.

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

DML Operations and Views

Use the DESCRIBE statement to verify that you have tables named copy_d_songs, copy_d_events, copy_d_cds, and copy_d_clients in your schema. If you don't, write a query to create a copy of each.

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

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

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

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

2. Use the CREATE or REPLACE option to create a view of *all* the columns in the copy_d_songs table called view_copy_d_songs.

```
CREATE OR REPLACE VIEW view_copy_d_songs AS
```

```
SELECT *
```

```
FROM copy_d_songs;
```

```
SELECT * FROM view_copy_d_songs;
```

3. Use view_copy_d_songs to INSERT the following data into the underlying copy_d_songs table. Execute a SELECT * from copy_d_songs to verify your DML command. See the graphic.

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

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

4. Create a view based on the DJs on Demand COPY_D_CDS table. Name the view read_copy_d_cds. Select all columns to be included in the view. Add a WHERE clause to restrict the year to 2000. Add the WITH READ ONLY option.

```
CREATE OR REPLACE VIEW read_copy_d_cds AS
```

```
SELECT *
```

```
FROM copy_d_cds  
WHERE year = '2000'  
WITH READ ONLY ;
```

```
SELECT * FROM read_copy_d_cds;
```

5. Using the read_copy_d_cds view, execute a DELETE FROM read_copy_d_cds WHERE cd_number = 90;

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

6. Use REPLACE to modify read_copy_d_cds. Replace the READ ONLY option with WITH CHECK OPTION CONSTRAINT ck_read_copy_d_cds. Execute a SELECT * statement to verify that the view exists.

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

7. Use the read_copy_d_cds view to delete any CD of year 2000 from the underlying copy_d_cds.

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

8. Use the read_copy_d_cds view to delete cd_number 90 from the underlying copy_d_cds table.

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

9. Use the read_copy_d_cds view to delete year 2001 records.

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

10. Execute a SELECT * statement for the base table copy_d_cds. What rows were deleted?

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

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

DELETE,INSERT,MODIFY restricted if it contains:

Group functions
GROUP BY CLAUSE
DISTINCT
pseudocolumn ROWNUM Keyword

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

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

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

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

Managing Views

1. Create a view from the copy_d_songs table called view_copy_d_songs that includes only the title and artist. Execute a SELECT * statement to verify that the view exists.

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

2. Issue a DROP view_copy_d_songs. Execute a SELECT * statement to verify that the view has been deleted.

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

ORA-00942: table or view does not exist

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

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

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

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

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

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

Indexes and Synonyms

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

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

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

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

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

3. When will an index be created automatically?

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

4. Create a nonunique index (foreign key) for the DJs on Demand column (cd_number) in the D_TRACK_LISTINGS table. Use the Oracle Application Express SQL Workshop Data Browser to confirm that the index was created.

CREATE INDEX d_tlg_cd_number_fk_i ON d_track_listings (cd_number);

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

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

6. Use a SELECT statement to display the index_name, table_name, and uniqueness from the data dictionary USER_INDEXES for the DJs on Demand D_EVENTS table.

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

7. Write a query to create a synonym called dj_tracks for the DJs on Demand d_track_listings table.

CREATE SYNONYM dj_tracks FOR d_track_listings;

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

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

9. Create a synonym for the D_TRACK_LISTINGS table. Confirm that it has been created by querying the data dictionary.

CREATE SYNONYM dj_tracks2 FOR d_track_listings;

```
SELECT * FROM user_synonyms WHERE table_NAME = UPPER('d_track_listings');
```

10. Drop the synonym that you created in question

DROP SYNONYM dj_tracks2;

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

RESULT:

CONTROLLING USER ACCESS

EX_NO:15

DATE:

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

The CREATE SESSION system privilege

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

The CREATE TABLE privilege

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

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

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

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

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

The ALTER USER statement

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

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

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

7. Query all the rows in your DEPARTMENTS table.

SELECT * FROM departments;

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

Team 1 executes this INSERT statement. INSERT INTO departments(department_id, department_name) VALUES (500, 'Education'); COMMIT;

Team 2 executes this INSERT statement. INSERT INTO departments(department_id, department_name) VALUES (510, 'Administration'); COMMIT;

9. Query the USER_TABLES data dictionary to see information about the tables that you own.

```
SELECT table_name FROM user_tables;
```

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

Team 1 revokes the privilege.

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

Team 2 revokes the privilege.

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

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

Team 1 executes this INSERT statement.

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

Team 2 executes this INSERT statement.

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

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

RESULT:

PL/SQL

CONTROL STRUCTURES

EX_NO:

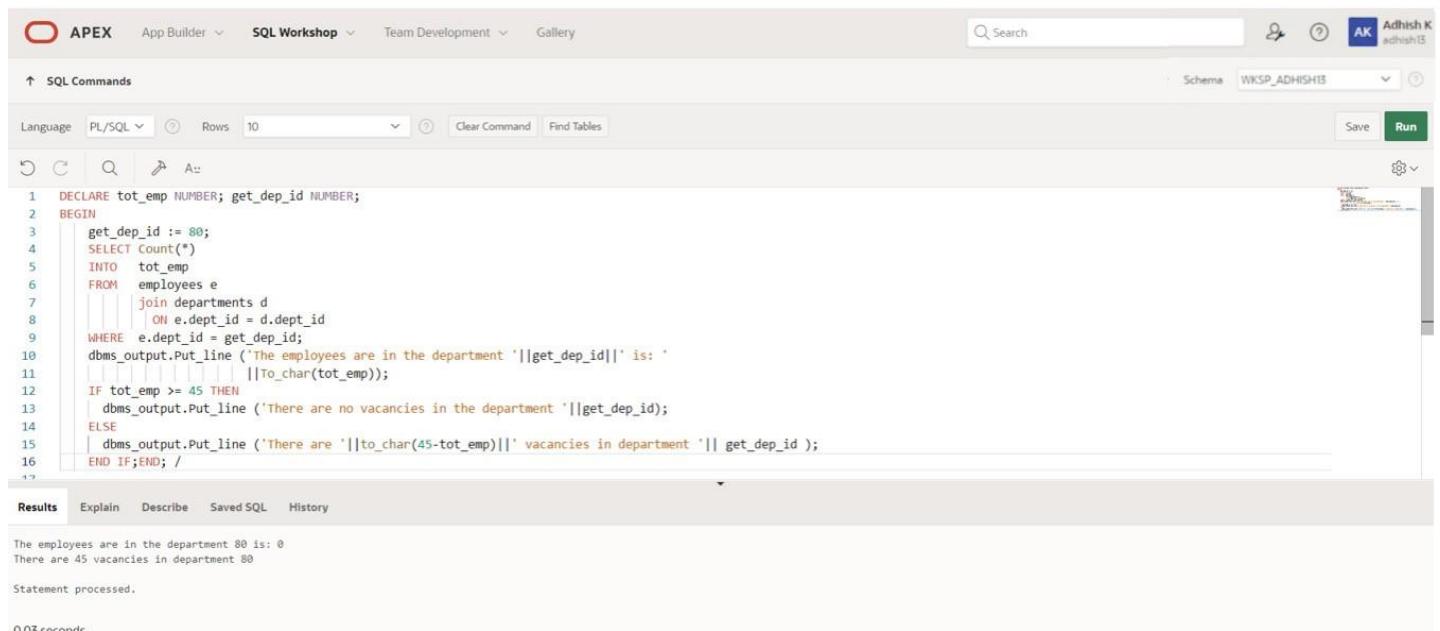
DATE:

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user information ('Adhish K adhish13') are on the right. The main area is titled 'SQL Commands' with tabs for 'Language' (set to 'PL/SQL'), 'Rows' (set to 10), and 'Run'. Below this is a toolbar with icons for undo, redo, search, and run. The code editor contains a PL/SQL block:

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

The results tab shows the output of the executed code:

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

Statement processed.

0.03 seconds
```

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

QUERY:

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

OUTPUT:

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

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

In the results pane below, the output is displayed:

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

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

QUERY:

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

```

END;

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

```

OUTPUT:

```

APEX App Builder SQL Workshop Team Development Gallery Search Schema WKSP_ADHISH13 Save Run
↑ SQL Commands Language PL/SQL Rows 10 Clear Command Find Tables
DECLARE
  CURSOR job_cursor IS
    SELECT e.job_id, j.low_sal
    FROM job_grades j,employees e;
  job_record job_cursor%ROWTYPE;
BEGIN
  OPEN job_cursor;
  FETCH job_cursor INTO job_record;
  WHILE job_cursor%FOUND LOOP
    Job ID: st_clerk
    Minimum Salary: 10000
    -----
    Job ID: 18
    Minimum Salary: 10000
    -----
    Job ID: st_charles
    Minimum Salary: 10000
    -----
    Job ID: 55
    Minimum Salary: 10000
    -----
    Job ID: 88
    Minimum Salary: 10000
    -----
    Job ID: 22

```

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');
  END IF;

```

```

ELSIF boo_val = TRUE THEN
DBMS_OUTPUT.PUT_LINE( boo_name || '= TRUE');
ELSE
DBMS_OUTPUT.PUT_LINE( boo_name || '= FALSE');
END IF;
END;
/

```

OUTPUT:

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

```

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

```

The Results tab displays the output:

```

TRUE
FALSE
Statement processed.
0.00 seconds

```

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

QUERY:

```

DECLARE
PROCEDURE pat_match (
    test_string  VARCHAR2,
    pattern      VARCHAR2
) IS
BEGIN

```

```

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

```

OUTPUT:

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

```

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

```

The results tab shows the output:

Employee ID	Full Name	Job Title	Hire Date	Salary
101	mark roy	st_clerk	02/03/1998	10900
110	steve davies	10	02/05/1999	40900
122	tom janu	st_charles	02/05/2000	50900
106	rocky smith	55	03/25/1996	25900
102	tony stark	88	05/01/1994	28900
1212	winston churchill	22	01/01/2001	50900

Statement processed.

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. In the top navigation bar, 'APEX' is selected. Below it, tabs for 'App Builder', 'SQL Workshop' (which is active), 'Team Development', and 'Gallery' are visible. A search bar and user information 'Adhish K adhish13' are on the right. The main area is titled 'SQL Commands'. It shows a code editor with the following PL/SQL code:

```

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

```

Below the code editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected. The output pane displays the message 'Procedure created.' and '0.03 seconds'.

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

QUERY:

DECLARE

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

IS

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

BEGIN

```
  IF sal_achieve > 44000 THEN
```

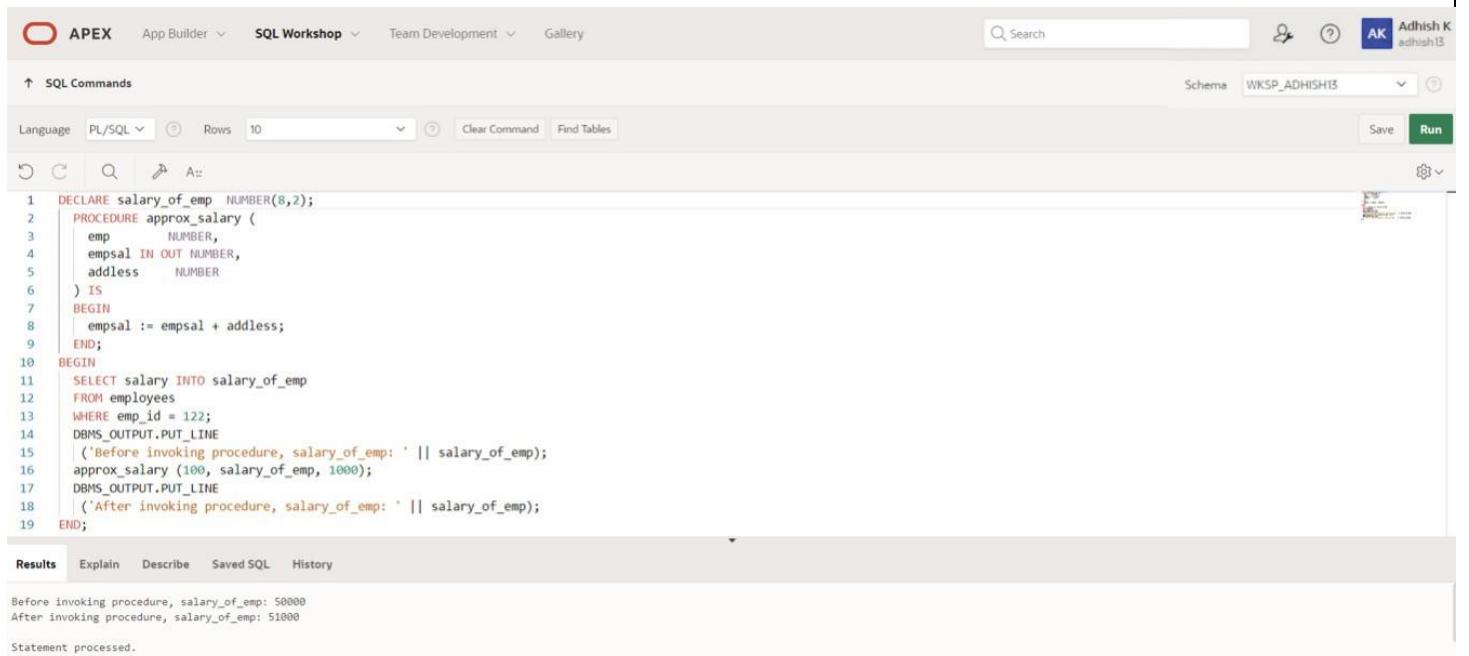
```
    incentive := 1800;
```

```
  ELSIF sal_achieve > 32000 THEN
```

```

incentive := 800;
ELSE
    incentive := 500;
END IF;
DBMS_OUTPUT.NEW_LINE;
DBMS_OUTPUT.PUT_LINE(
    'Sale achieved : ' || sal_achieve || ', incentive : ' || incentive || ')';
END test1;
BEGIN
    test1(45000);
test1(36000);
test1(28000);
END;/
```

OUTPUT:



The screenshot shows the Oracle 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 Adhish K (adhish15). The main area is titled 'SQL Commands' with a 'PL/SQL' dropdown. The code area contains a block of PL/SQL code. The results section shows the output of the code execution.

```

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

Results

```

Before invoking procedure, salary_of_emp: 50000
After invoking procedure, salary_of_emp: 51000
Statement processed.
```

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

QUERY:

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

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

```

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

```

OUTPUT:

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

```

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

```

In the Results tab, the output shows an error message:

```

Error at line 4/23: ORA-06550: line 4, column 23:
PLS-00201: identifier 'Welcome' must be declared
ORA-06512: at "SYS.WWV_DBMS_SQL_APEX_230200", line 801
ORA-06550: line 4, column 1:
PL/SQL: Statement ignored

```

The error message is highlighted with a yellow box.

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

QUERY:

DECLARE

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

BEGIN

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

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

IF tot_emp >= 45 THEN

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

ELSE

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

END IF;

END;

/

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown is set to WKSP_ADHISH15. The main area displays the PL/SQL code. The code declares variables, performs a SELECT COUNT(*) into tot_emp for department 80, and then uses dbms_output.Put_line to print the total number of employees. It then checks if tot_emp is greater than or equal to 45. If true, it prints 'There are no vacancies'. Otherwise, it prints the number of vacancies (45 - tot_emp). The code ends with an IF-THEN-ELSE-END IF block and a final END IF;. The bottom section shows the results of the execution, displaying the output messages: 'The employees are in the department 80 is: 0' and 'There are 45 vacancies in department 80'. The status bar at the bottom indicates the statement was processed in 0.03 seconds.

```
1  DECLARE tot_emp NUMBER; get_dep_id NUMBER;
2  BEGIN
3    get_dep_id := 80;
4    SELECT Count(*)
5    INTO tot_emp
6    FROM employees e
7    join departments d
8    ON e.department_id = d.dept_id
9    WHERE e.department_id = get_dep_id;
10   dbms_output.Put_line ('The employees are in the department'||get_dep_id||' is: '
11   ||To_char(tot_emp));
12  IF tot_emp >= 45 THEN
13    dbms_output.Put_line ('There are no vacancies in the department'||get_dep_id);
14  ELSE
15    dbms_output.Put_line ('There are'||to_char(45-tot_emp)||' vacancies in department'||get_dep_id );
16  END IF;END /
```

Results Explain Describe Saved SQL History

The employees are in the department 80 is: 0
There are 45 vacancies in department 80
Statement processed.
0.03 seconds

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

QUERY:

DECLARE

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

CURSOR c_employees IS

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

BEGIN

```
DBMS_OUTPUT.PUT_LINE('Employee ID | Full Name | Job Title | Hire Date | Salary');
```

```
DBMS_OUTPUT.PUT_LINE('-----');
```

```
OPEN c_employees;
```

```
FETCH c_employees INTO v_employee_id, v_full_name, v_job_id, v_hire_date, v_salary;
```

```
WHILE c_employees%FOUND LOOP
```

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

```
    FETCH c_employees INTO v_employee_id, v_full_name, v_job_id, v_hire_date, v_salary;
```

```
END LOOP;
```

```
CLOSE c_employees;
```

```
END;
```

```
/
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile for 'Adhish K adhish13'. The main area is titled 'SQL Commands' with a 'PL/SQL' language selection. The code area contains a block of PL/SQL code. The results section at the bottom displays the output of the executed code.

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

Results

```
Incentive = 4800
Statement processed.

0.02 seconds
```

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

QUERY:

DECLARE

 CURSOR emp_cursor IS

 SELECT e.employee_id, e.first_name, m.first_name AS manager_name

 FROM employees e

 LEFT JOIN employees m ON e.manager_id = m.employee_id;

 emp_record emp_cursor%ROWTYPE;

BEGIN

 OPEN emp_cursor;

 FETCH emp_cursor INTO emp_record;

 WHILE emp_cursor%FOUND LOOP

 DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp_record.employee_id);

 DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_record.first_name);

 DBMS_OUTPUT.PUT_LINE('Manager Name: ' || emp_record.manager_name);

 DBMS_OUTPUT.PUT_LINE(' -----');

 FETCH emp_cursor INTO emp_record;

 END LOOP;

 CLOSE emp_cursor;

END;

/

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'Adhish K' are also present. The main area is titled 'SQL Commands' with a 'PL/SQL' tab selected. The code area contains a PL/SQL block with numbered lines 1 through 16. Lines 1-9 define variables and a cursor, while lines 10-16 implement a loop to output employee information. The bottom section shows the results of the execution, indicating 'Statement processed.' and a execution time of '0.01 seconds'.

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

Results Explain Describe Saved SQL History

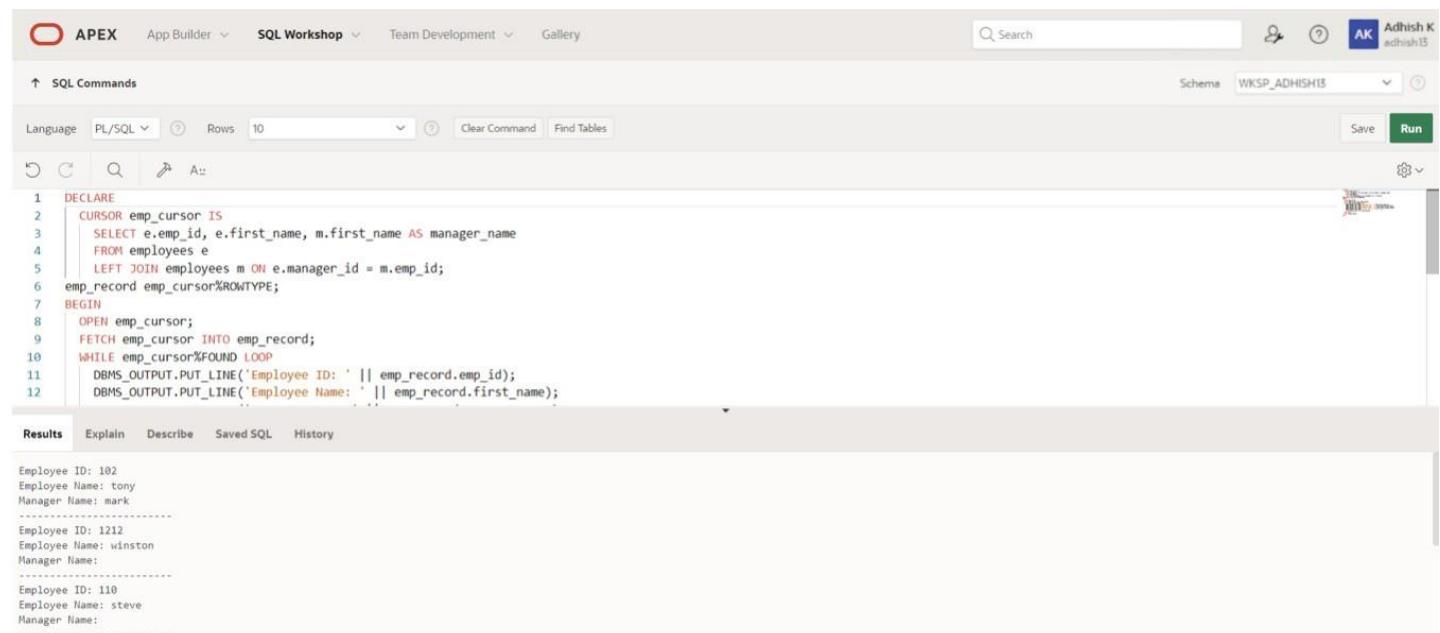
Statement processed.
0.01 seconds

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 search bar contains 'Search' and the schema 'WKSP_ADHISH13'. The main workspace is titled 'SQL Commands' and shows a PL/SQL block being run. The code is as follows:

```
1  DECLARE
2    CURSOR emp_cursor IS
3      SELECT e.emp_id, e.first_name, m.first_name AS manager_name
4        FROM employees e
5        LEFT JOIN employees m ON e.manager_id = m.emp_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.emp_id);
12     DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_record.first_name);
13   END LOOP;
14 END;
15 /
```

The results pane displays the output of the PL/SQL block, listing employee details:

Employee ID	Employee Name	Manager Name
102	tony	mark
1212	winston	
110	steve	

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

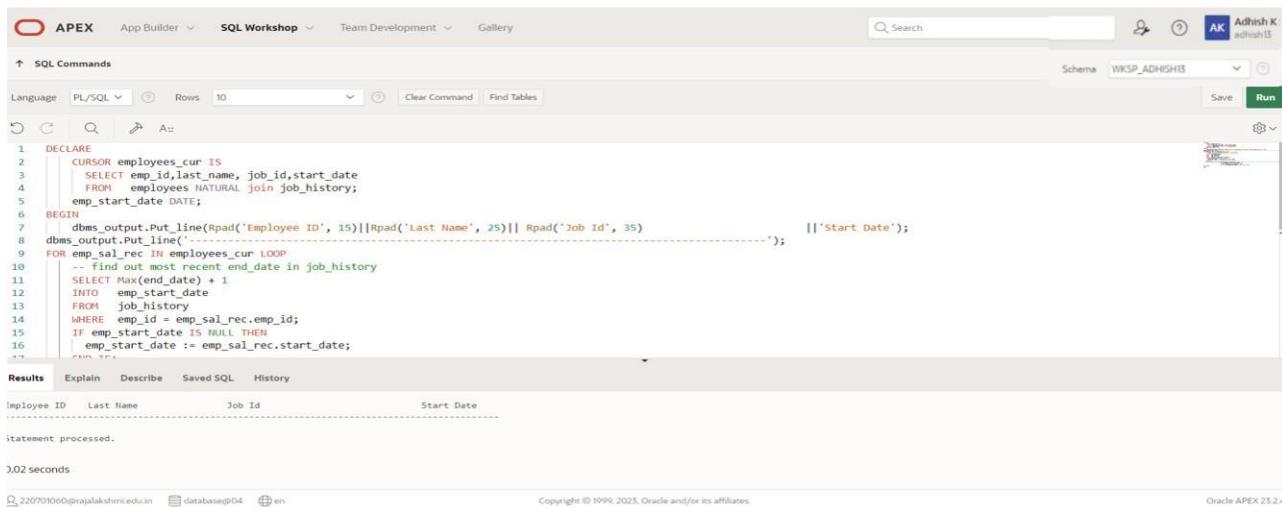
QUERY:

DECLARE

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

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

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile for 'Adhish K' (adhish15). The main workspace has a 'SQL Commands' tab selected. The code area contains the PL/SQL block from above. The 'Results' tab is active, displaying the output of the query. The output shows the employee ID, last name, job ID, and start date for each employee, with the start date formatted as 'dd-mon-yyyy'. The total execution time is listed as 0.02 seconds.

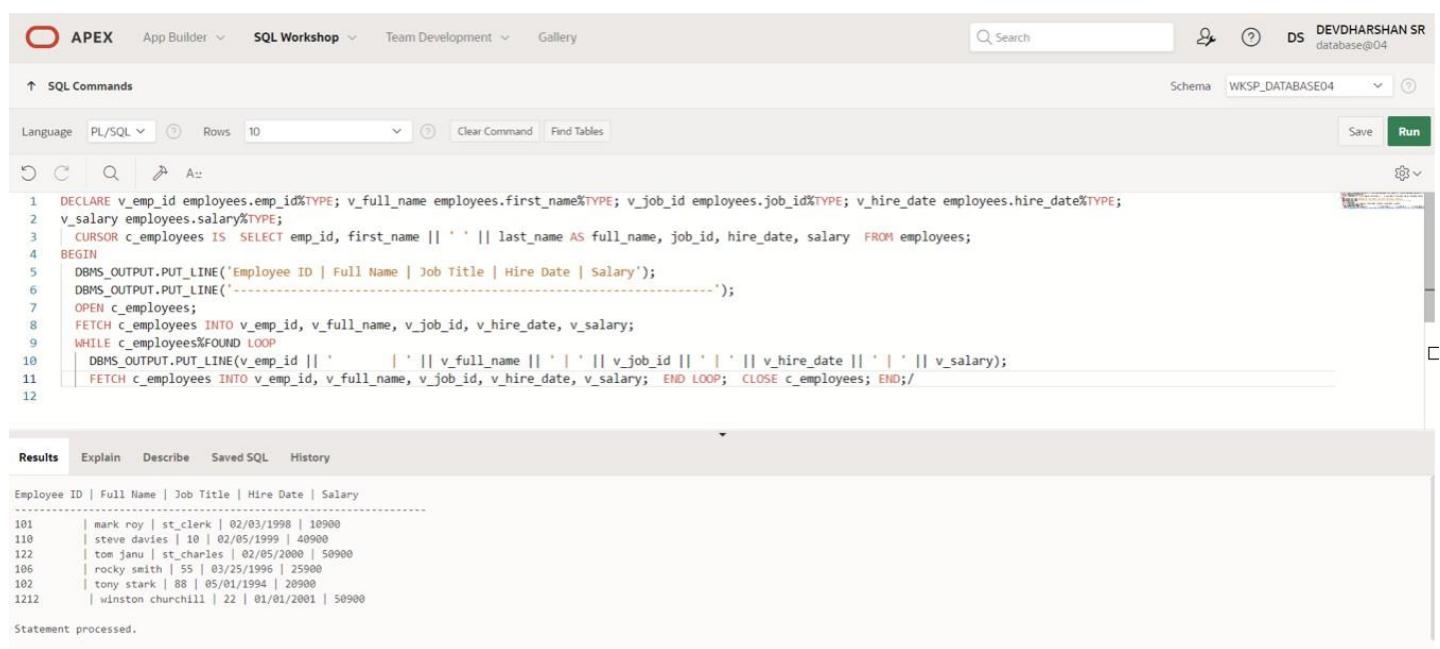
Employee ID	Last Name	Job Id	Start Date
101	Aasper	ADVISOR	01-JAN-1981
102	Buchanan	SALESREP	01-JAN-1981
103	Ford	SALESREP	01-JAN-1981
104	Mavroides	SALESREP	01-JAN-1981
105	Snavely	SALESREP	01-JAN-1981
106	Zlot	SALESREP	01-JAN-1981
107	Alonso	SALESREP	01-JAN-1981
108	Frederick	SALESREP	01-JAN-1981
109	Gundotra	SALESREP	01-JAN-1981
110	Koren	SALESREP	01-JAN-1981
111	Li	SALESREP	01-JAN-1981
112	Petterson	SALESREP	01-JAN-1981
113	Singh	SALESREP	01-JAN-1981
114	Tucker	SALESREP	01-JAN-1981
115	Whalen	SALESREP	01-JAN-1981
116	Yergan	SALESREP	01-JAN-1981
117	Adams	RECEPTIONIST	01-JAN-1981
118	Fuller	RECEPTIONIST	01-JAN-1981
119	King	RECEPTIONIST	01-JAN-1981
120	Markham	RECEPTIONIST	01-JAN-1981
121	Neumann	RECEPTIONIST	01-JAN-1981
122	Platt	RECEPTIONIST	01-JAN-1981
123	Richter	RECEPTIONIST	01-JAN-1981
124	Stiles	RECEPTIONIST	01-JAN-1981
125	Washington	RECEPTIONIST	01-JAN-1981
126	Wells	RECEPTIONIST	01-JAN-1981
127	Winston	RECEPTIONIST	01-JAN-1981
128	Zarathrustra	RECEPTIONIST	01-JAN-1981
129	Abrahams	RECEPTIONIST	01-JAN-1981
130	Andersen	RECEPTIONIST	01-JAN-1981
131	Chen	RECEPTIONIST	01-JAN-1981
132	Frederickson	RECEPTIONIST	01-JAN-1981
133	Hall	RECEPTIONIST	01-JAN-1981
134	King	RECEPTIONIST	01-JAN-1981
135	Landry	RECEPTIONIST	01-JAN-1981
136	Montgomery	RECEPTIONIST	01-JAN-1981
137	Peacock	RECEPTIONIST	01-JAN-1981
138	Quigley	RECEPTIONIST	01-JAN-1981
139	Spangler	RECEPTIONIST	01-JAN-1981
140	Whaley	RECEPTIONIST	01-JAN-1981
141	Wells	RECEPTIONIST	01-JAN-1981
142	Winston	RECEPTIONIST	01-JAN-1981
143	Zarathrustra	RECEPTIONIST	01-JAN-1981
144	Abrahams	RECEPTIONIST	01-JAN-1981
145	Andersen	RECEPTIONIST	01-JAN-1981
146	Chen	RECEPTIONIST	01-JAN-1981
147	Frederickson	RECEPTIONIST	01-JAN-1981
148	Hall	RECEPTIONIST	01-JAN-1981
149	King	RECEPTIONIST	01-JAN-1981
150	Landry	RECEPTIONIST	01-JAN-1981
151	Montgomery	RECEPTIONIST	01-JAN-1981
152	Peacock	RECEPTIONIST	01-JAN-1981
153	Quigley	RECEPTIONIST	01-JAN-1981
154	Spangler	RECEPTIONIST	01-JAN-1981
155	Whaley	RECEPTIONIST	01-JAN-1981
156	Wells	RECEPTIONIST	01-JAN-1981
157	Winston	RECEPTIONIST	01-JAN-1981
158	Zarathrustra	RECEPTIONIST	01-JAN-1981
159	Abrahams	RECEPTIONIST	01-JAN-1981
160	Andersen	RECEPTIONIST	01-JAN-1981
161	Chen	RECEPTIONIST	01-JAN-1981
162	Frederickson	RECEPTIONIST	01-JAN-1981
163	Hall	RECEPTIONIST	01-JAN-1981
164	King	RECEPTIONIST	01-JAN-1981
165	Landry	RECEPTIONIST	01-JAN-1981
166	Montgomery	RECEPTIONIST	01-JAN-1981
167	Peacock	RECEPTIONIST	01-JAN-1981
168	Quigley	RECEPTIONIST	01-JAN-1981
169	Spangler	RECEPTIONIST	01-JAN-1981
170	Whaley	RECEPTIONIST	01-JAN-1981
171	Wells	RECEPTIONIST	01-JAN-1981
172	Winston	RECEPTIONIST	01-JAN-1981
173	Zarathrustra	RECEPTIONIST	01-JAN-1981
174	Abrahams	RECEPTIONIST	01-JAN-1981
175	Andersen	RECEPTIONIST	01-JAN-1981
176	Chen	RECEPTIONIST	01-JAN-1981
177	Frederickson	RECEPTIONIST	01-JAN-1981
178	Hall	RECEPTIONIST	01-JAN-1981
179	King	RECEPTIONIST	01-JAN-1981
180	Landry	RECEPTIONIST	01-JAN-1981
181	Montgomery	RECEPTIONIST	01-JAN-1981
182	Peacock	RECEPTIONIST	01-JAN-1981
183	Quigley	RECEPTIONIST	01-JAN-1981
184	Spangler	RECEPTIONIST	01-JAN-1981
185	Whaley	RECEPTIONIST	01-JAN-1981
186	Wells	RECEPTIONIST	01-JAN-1981
187	Winston	RECEPTIONIST	01-JAN-1981
188	Zarathrustra	RECEPTIONIST	01-JAN-1981
189	Abrahams	RECEPTIONIST	01-JAN-1981
190	Andersen	RECEPTIONIST	01-JAN-1981
191	Chen	RECEPTIONIST	01-JAN-1981
192	Frederickson	RECEPTIONIST	01-JAN-1981
193	Hall	RECEPTIONIST	01-JAN-1981
194	King	RECEPTIONIST	01-JAN-1981
195	Landry	RECEPTIONIST	01-JAN-1981
196	Montgomery	RECEPTIONIST	01-JAN-1981
197	Peacock	RECEPTIONIST	01-JAN-1981
198	Quigley	RECEPTIONIST	01-JAN-1981
199	Spangler	RECEPTIONIST	01-JAN-1981
200	Whaley	RECEPTIONIST	01-JAN-1981
201	Wells	RECEPTIONIST	01-JAN-1981
202	Winston	RECEPTIONIST	01-JAN-1981
203	Zarathrustra	RECEPTIONIST	01-JAN-1981
204	Abrahams	RECEPTIONIST	01-JAN-1981
205	Andersen	RECEPTIONIST	01-JAN-1981
206	Chen	RECEPTIONIST	01-JAN-1981
207	Frederickson	RECEPTIONIST	01-JAN-1981
208	Hall	RECEPTIONIST	01-JAN-1981
209	King	RECEPTIONIST	01-JAN-1981
210	Landry	RECEPTIONIST	01-JAN-1981
211	Montgomery	RECEPTIONIST	01-JAN-1981
212	Peacock	RECEPTIONIST	01-JAN-1981
213	Quigley	RECEPTIONIST	01-JAN-1981
214	Spangler	RECEPTIONIST	01-JAN-1981
215	Whaley	RECEPTIONIST	01-JAN-1981
216	Wells	RECEPTIONIST	01-JAN-1981
217	Winston	RECEPTIONIST	01-JAN-1981
218	Zarathrustra	RECEPTIONIST	01-JAN-1981
219	Abrahams	RECEPTIONIST	01-JAN-1981
220	Andersen	RECEPTIONIST	01-JAN-1981
221	Chen	RECEPTIONIST	01-JAN-1981
222	Frederickson	RECEPTIONIST	01-JAN-1981
223	Hall	RECEPTIONIST	01-JAN-1981
224	King	RECEPTIONIST	01-JAN-1981
225	Landry	RECEPTIONIST	01-JAN-1981
226	Montgomery	RECEPTIONIST	01-JAN-1981
227	Peacock	RECEPTIONIST	01-JAN-1981
228	Quigley	RECEPTIONIST	01-JAN-1981
229	Spangler	RECEPTIONIST	01-JAN-1981
230	Whaley	RECEPTIONIST	01-JAN-1981
231	Wells	RECEPTIONIST	01-JAN-1981
232	Winston	RECEPTIONIST	01-JAN-1981
233	Zarathrustra	RECEPTIONIST	01-JAN-1981
234	Abrahams	RECEPTIONIST	01-JAN-1981
235	Andersen	RECEPTIONIST	01-JAN-1981
236	Chen	RECEPTIONIST	01-JAN-1981
237	Frederickson	RECEPTIONIST	01-JAN-1981
238	Hall	RECEPTIONIST	01-JAN-1981
239	King	RECEPTIONIST	01-JAN-1981
240	Landry	RECEPTIONIST	01-JAN-1981
241	Montgomery	RECEPTIONIST	01-JAN-1981
242	Peacock	RECEPTIONIST	01-JAN-1981
243	Quigley	RECEPTIONIST	01-JAN-1981
244	Spangler	RECEPTIONIST	01-JAN-1981
245	Whaley	RECEPTIONIST	01-JAN-1981
246	Wells	RECEPTIONIST	01-JAN-1981
247	Winston	RECEPTIONIST	01-JAN-1981
248	Zarathrustra	RECEPTIONIST	01-JAN-1981
249	Abrahams	RECEPTIONIST	01-JAN-1981
250	Andersen	RECEPTIONIST	01-JAN-1981
251	Chen	RECEPTIONIST	01-JAN-1981
252	Frederickson	RECEPTIONIST	01-JAN-1981
253	Hall	RECEPTIONIST	01-JAN-1981
254	King	RECEPTIONIST	01-JAN-1981
255	Landry	RECEPTIONIST	01-JAN-1981
256	Montgomery	RECEPTIONIST	01-JAN-1981
257	Peacock	RECEPTIONIST	01-JAN-1981
258	Quigley	RECEPTIONIST	01-JAN-1981
259	Spangler	RECEPTIONIST	01-JAN-1981
260	Whaley	RECEPTIONIST	01-JAN-1981
261	Wells	RECEPTIONIST	01-JAN-1981
262	Winston	RECEPTIONIST	01-JAN-1981
263	Zarathrustra	RECEPTIONIST	01-JAN-1981
264	Abrahams	RECEPTIONIST	01-JAN-1981
265	Andersen	RECEPTIONIST	01-JAN-1981
266	Chen	RECEPTIONIST	01-JAN-1981
267	Frederickson	RECEPTIONIST	01-JAN-1981
268	Hall	RECEPTIONIST	01-JAN-1981
269	King	RECEPTIONIST	01-JAN-1981
270	Landry	RECEPTIONIST	01-JAN-1981
271	Montgomery	RECEPTIONIST	01-JAN-1981
272	Peacock	RECEPTIONIST	01-JAN-1981
273	Quigley	RECEPTIONIST	01-JAN-1981
274	Spangler	RECEPTIONIST	01-JAN-1981
275	Whaley	RECEPTIONIST	01-JAN-1981
276	Wells	RECEPTIONIST	01-JAN-1981
277	Winston	RECEPTIONIST	01-JAN-1981
278	Zarathrustra	RECEPTIONIST	01-JAN-1981
279	Abrahams	RECEPTIONIST	01-JAN-1981
280	Andersen	RECEPTIONIST	01-JAN-1981
281	Chen	RECEPTIONIST	01-JAN-1981
282	Frederickson	RECEPTIONIST	01-JAN-1981
283	Hall	RECEPTIONIST	01-JAN-1981
284	King	RECEPTIONIST	01-JAN-1981
285	Landry	RECEPTIONIST	01-JAN-1981
286	Montgomery	RECEPTIONIST	01-JAN-1981
287	Peacock	RECEPTIONIST	01-JAN-1981
288	Quigley	RECEPTIONIST	01-JAN-1981
289	Spangler	RECEPTIONIST	01-JAN-1981
290	Whaley	RECEPTIONIST	01-JAN-1981
291	Wells	RECEPTIONIST	01-JAN-1981
292	Winston	RECEPTIONIST	01-JAN-1981
293	Zarathrustra	RECEPTIONIST	01-JAN-1981
294	Abrahams	RECEPTIONIST	01-JAN-1981
295	Andersen	RECEPTIONIST	01-JAN-1981
296	Chen	RECEPTIONIST	01-JAN-1981
297	Frederickson	RECEPTIONIST	01-JAN-1981
298	Hall	RECEPTIONIST	01-JAN-1981
299	King	RECEPTIONIST	01-JAN-1981
300	Landry	RECEPTIONIST	01-JAN-1981
301	Montgomery	RECEPTIONIST	01-JAN-1981
302	Peacock	RECEPTIONIST	01-JAN-1981
303	Quigley	RECEPTIONIST	01-JAN-1981
304	Spangler	RECEPTIONIST	01-JAN-1981
305	Whaley	RECEPTIONIST	01-JAN-1981
306	Wells	RECEPTIONIST	01-JAN-1981
307	Winston	RECEPTIONIST	01-JAN-1981
308	Zarathrustra	RECEPTIONIST	01-JAN-1981
309	Abrahams	RECEPTIONIST	01-JAN-1981
310	Andersen	RECEPTIONIST	01-JAN-1981
311	Chen	RECEPTIONIST	01-JAN-1981
312	Frederickson	RECEPTIONIST	01-JAN-1981
313	Hall	RECEPTIONIST	01-JAN-1981
314	King	RECEPTIONIST	01-JAN-1981
315	Landry	RECEPTIONIST	01-JAN-1981
316	Montgomery	RECEPTIONIST	01-JAN-1981
317	Peacock	RECEPTIONIST	01-JAN-1981
318	Quigley	RECEPTIONIST	01-JAN-1981
319	Spangler	RECEPTIONIST	01-JAN-1981
320	Whaley	RECEPTIONIST	01-JAN-1981
321	Wells	RECEPTIONIST	01-JAN-1981
322	Winston	RECEPTIONIST	01-JAN-1981
323	Zarathrustra	RECEPTIONIST	01-JAN-1981
324	Abrahams	RECEPTIONIST	01-JAN-1981
325	Andersen	RECEPTIONIST	01-JAN-1981
326	Chen	RECEPTIONIST	01-JAN-1981
327	Frederickson	RECEPTIONIST	01-JAN-1981
328	Hall	RECEPTIONIST	01-JAN-1981
329	King	RECEPTIONIST	01-JAN-1981
330	Landry	RECEPTIONIST	01-JAN-1981
331	Montgomery	RECEPTIONIST	01-JAN-1981
332	Peacock	RECEPTIONIST	01-JAN-1981
333	Quigley	RECEPTIONIST	01-JAN-1981
334	Spangler	RECEPTIONIST	01-JAN-1981
335	Whaley	RECEPTIONIST	01-JAN-1981
336	Wells	RECEPTIONIST	01-JAN-1981
337	Winston	RECEPTIONIST	01-JAN-1981
338	Zarathrustra	RECEPTIONIST	01-JAN-1981
339	Abrahams	RECEPTIONIST	01-JAN-1981
340	Andersen	RECEPTIONIST	01-JAN-1981
341	Chen	RECEPTIONIST	01-JAN-1981
342	Frederickson	RECEPTIONIST	01-JAN-1981
343	Hall	RECEPTIONIST	01-JAN-1981
344	King	RECEPTIONIST	01-JAN-1981
345	Landry	RECEPTIONIST	01-JAN-1981
346	Montgomery	RECEPTIONIST	01-JAN-1981
347	Peacock	RECEPTIONIST	01-JAN-1981
348	Quigley	RECEPTIONIST	01-JAN-1981
349	Spangler	RECEPTIONIST	01-JAN-1981
350	Whaley	RECEPTIONIST	01-JAN-1981
351	Wells	RECEPTIONIST	01-JAN-1981
352	Winston	RECEPTIONIST	01-JAN-1981
353	Zarathrustra	RECEPTIONIST	01-JAN-1981
354	Abrahams	RECEPTIONIST	01-JAN-1981

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a connection named 'DEVDHARSHAN SR database@04'. The main area is titled 'SQL Commands' with tabs for 'Language' (set to 'PL/SQL'), 'Rows' (set to 10), and 'Run' (button). Below these are icons for 'Clear Command' and 'Find Tables'. The code area contains a numbered PL/SQL block. The results section at the bottom displays the output of the executed query, listing employee details with their hire dates and salaries.

```
1  DECLARE v_emp_id employees.emp_id%TYPE; v_full_name employees.first_name%TYPE; v_job_id employees.job_id%TYPE; v_hire_date employees.hire_date%TYPE;
2  v_salary employees.salary%TYPE;
3  CURSOR c_employees IS SELECT emp_id, first_name || ' ' || last_name AS full_name, job_id, hire_date, salary FROM employees;
4  BEGIN
5    DBMS_OUTPUT.PUT_LINE('Employee ID | Full Name | Job Title | Hire Date | Salary');
6    DBMS_OUTPUT.PUT_LINE('-----');
7    OPEN c_employees;
8    FETCH c_employees INTO v_emp_id, v_full_name, v_job_id, v_hire_date, v_salary;
9    WHILE c_employees%FOUND LOOP
10      DBMS_OUTPUT.PUT_LINE(v_emp_id || ' ' || v_full_name || ' ' || v_job_id || ' ' || v_hire_date || ' ' || v_salary);
11      FETCH c_employees INTO v_emp_id, v_full_name, v_job_id, v_hire_date, v_salary; END LOOP; CLOSE c_employees; END;/
```

Employee ID	Full Name	Job Title	Hire Date	Salary
101	mark roy	st_clerk	02/03/1998	10900
110	steve davies	10	02/05/1999	40900
122	tom janu	st_charles	02/05/2000	50900
106	rocky smith	155	03/25/1996	25900
102	tony stark	88	05/01/1994	28900
1212	winston churchill	22	01/01/2001	50900

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

RESULT:

PROCEDURES AND FUNCTIONS

EX_NO: 17

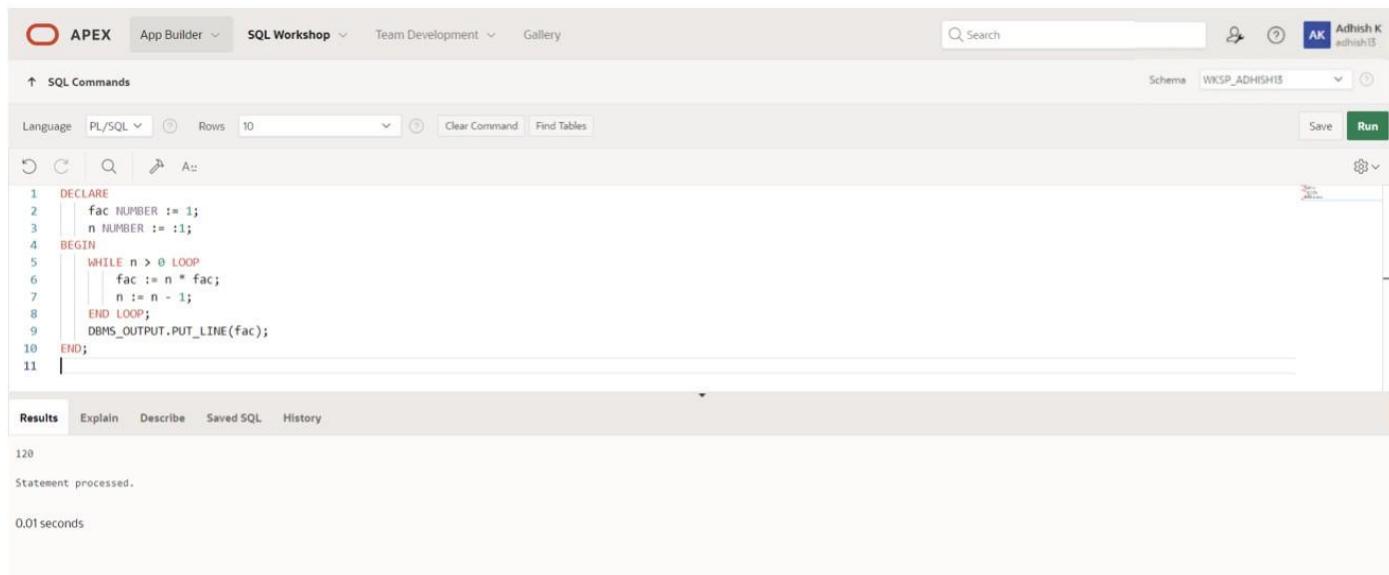
DATE:

1.) Factorial of a number using function.

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. The code entered is a PL/SQL block to calculate factorial. The code is as follows:

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

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

```
120
Statement processed.
0.01 seconds
```

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

QUERY:

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

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

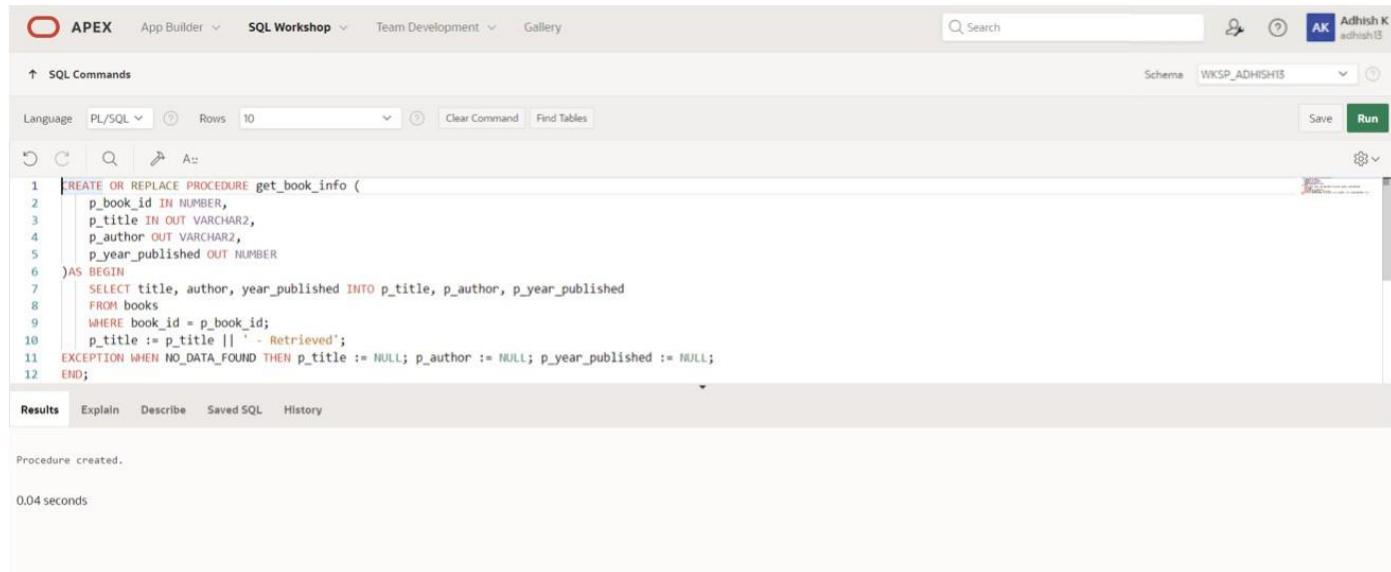
DECLARE
    v_book_id NUMBER := 1;
    v_title VARCHAR2(100);
    v_author VARCHAR2(100);
    v_year_published NUMBER;
BEGIN
    v_title := 'Initial Title';

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

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

END;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, 'SQL Workshop' is active. The main area is titled 'SQL Commands'. The schema dropdown shows 'WKSP_ADHISH13'. A search bar and a user profile for 'Adhish K' are also present. The code editor contains the following PL/SQL procedure:

```
1 CREATE OR REPLACE PROCEDURE get_book_info (
2     p_book_id IN NUMBER,
3     p_title IN OUT VARCHAR2,
4     p_author OUT VARCHAR2,
5     p_year_published OUT NUMBER
6 )AS BEGIN
7     SELECT title, author, year_published INTO p_title, p_author, p_year_published
8     FROM books
9     WHERE book_id = p_book_id;
10    p_title := p_title || ' - Retrieved';
11 EXCEPTION WHEN NO_DATA_FOUND THEN p_title := NULL; p_author := NULL; p_year_published := NULL;
12 END;
```

The results tab shows the message 'Procedure created.' and a execution time of '0.04 seconds'.

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

RESULT:

TRIGGER

EX_NO: 18

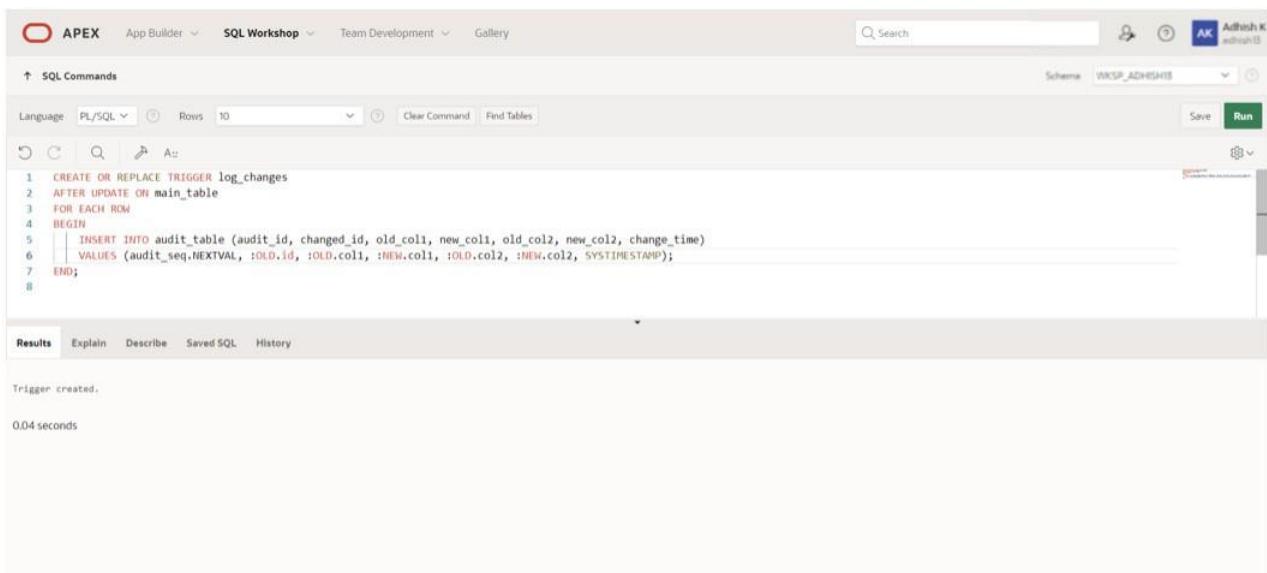
DATE:

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, the tabs 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery' are visible. The 'SQL Workshop' tab is active. On the left, there's a toolbar with icons for Undo, Redo, Find, Replace, and others. The main area is titled 'SQL Commands'. It shows a code editor with the following PL/SQL code:

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

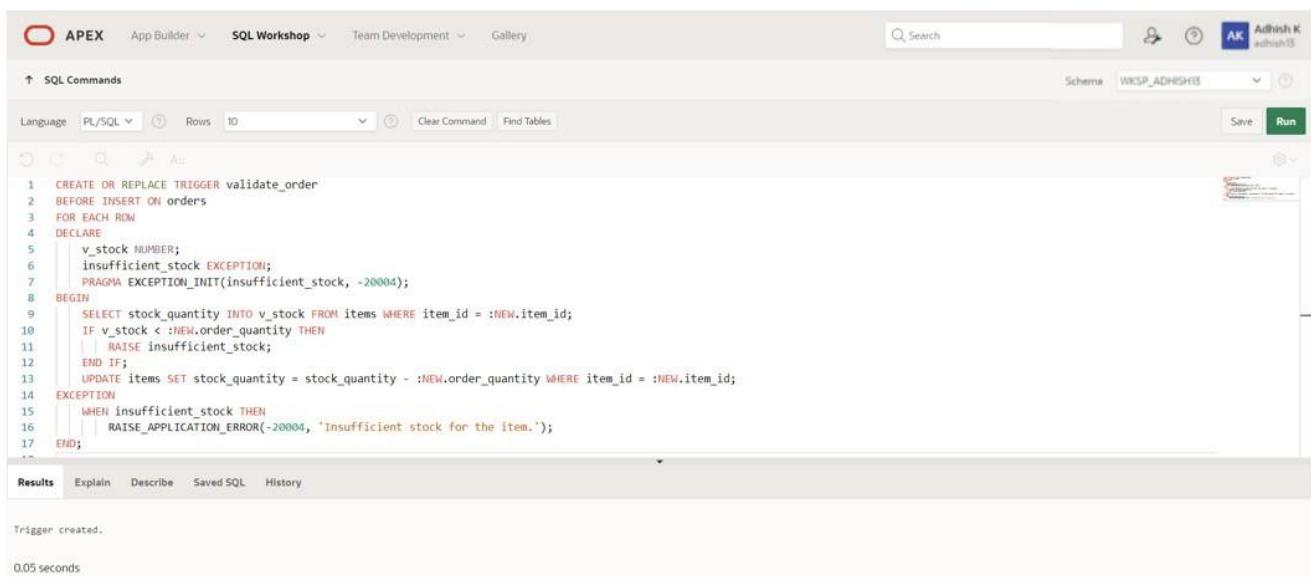
Below the code editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected. The output pane displays the message 'Trigger created.' and '0.04 seconds'.

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, the tabs 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery' are visible. The 'SQL Workshop' tab is active. On the left, there's a toolbar with icons for search, refresh, and run. The main area is titled 'SQL Commands'. A dropdown menu 'Language' is set to 'PL/SQL'. The input field contains the PL/SQL code for the trigger. The code is as follows:

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

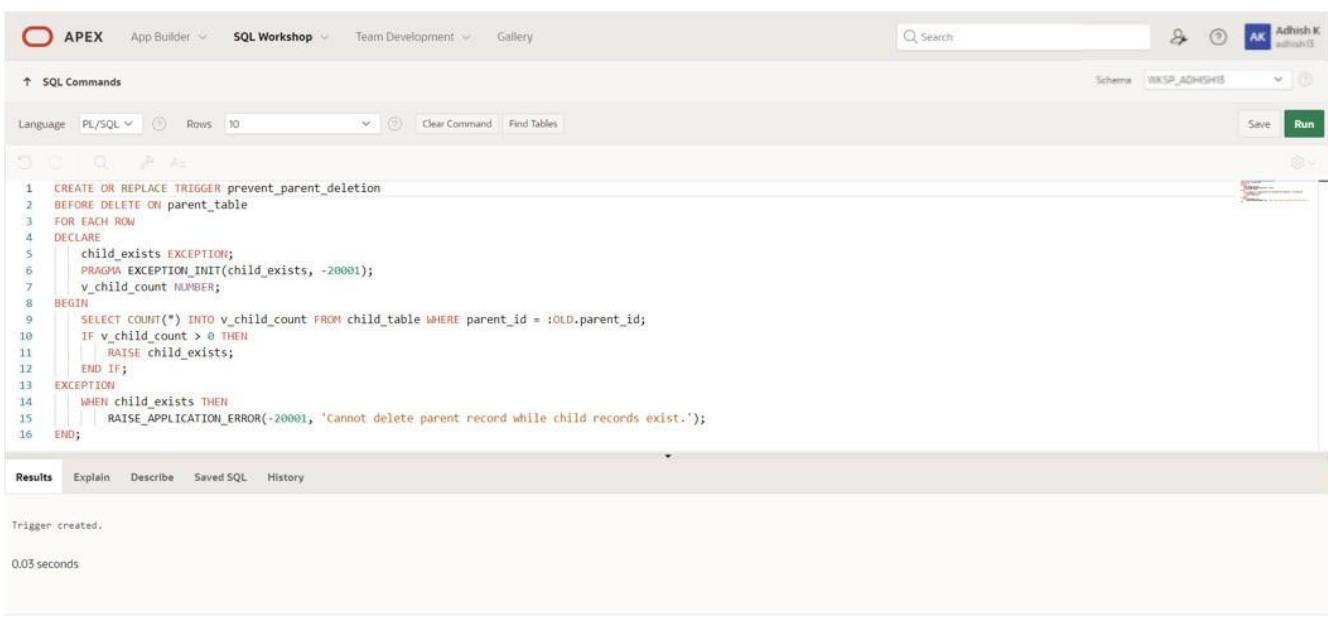
At the bottom of the code editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected. The output pane shows the message 'Trigger created.' and '0.05 seconds'.

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there are user profile icons for 'Adhish K' and 'adhist13'. The main workspace is titled 'SQL Commands' and contains the PL/SQL code for creating a trigger. The code is syntax-highlighted, showing keywords in red and comments in green. The bottom of the screen displays the results of the command execution, indicating that the trigger was successfully created.

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

Results Explain Describe Saved SQL History

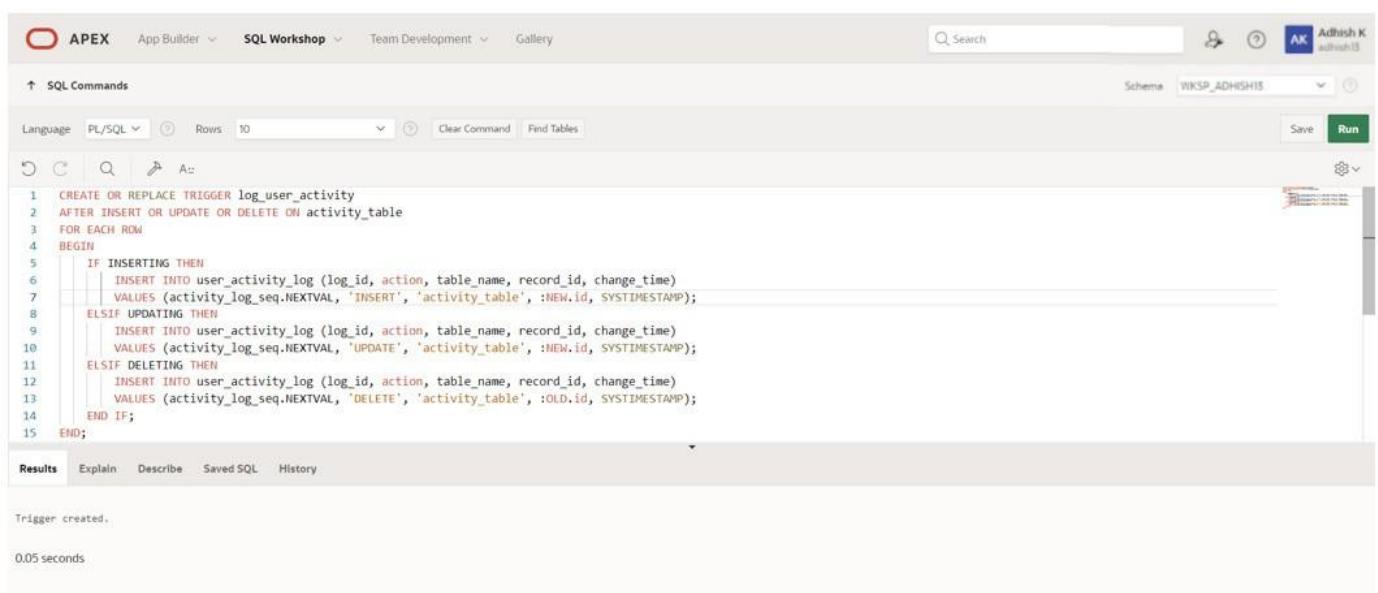
Trigger created.
0.03 seconds

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The search bar contains 'Search'. The schema dropdown is set to 'WKSP_ADHISH13'. The main workspace displays the following PL/SQL code:

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

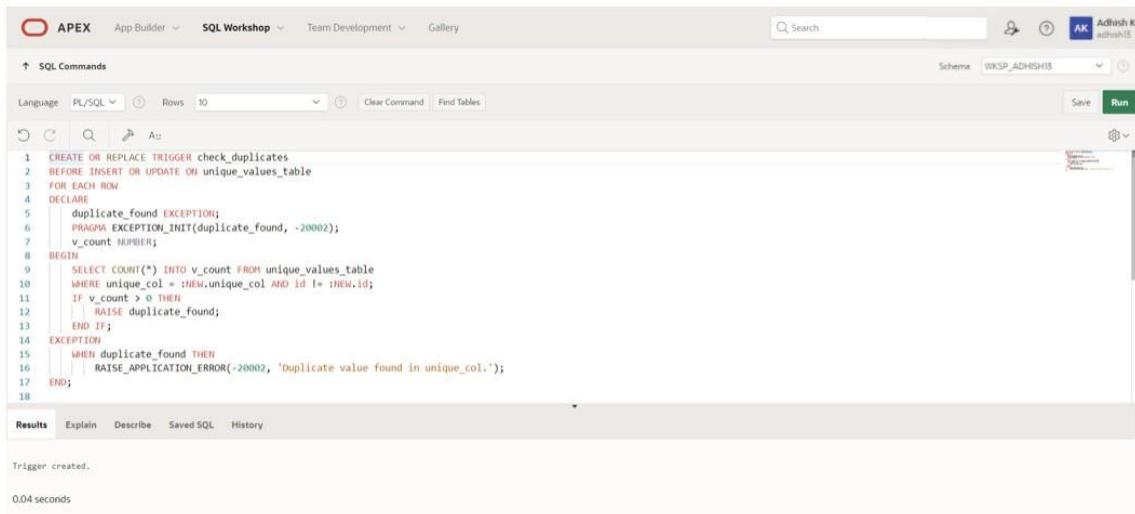
The 'Results' tab at the bottom shows the output: 'Trigger created.' and '0.05 seconds'.

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The main area is titled "SQL Commands". The schema dropdown is set to "WKSP_ADHISH13". The code editor contains the PL/SQL trigger definition provided above. The "Run" button is visible at the top right of the code editor. Below the code editor, there are tabs for Results, Explain, Describe, Saved SQL, and History. The results pane at the bottom displays the message "Trigger created." and "0.04 seconds".

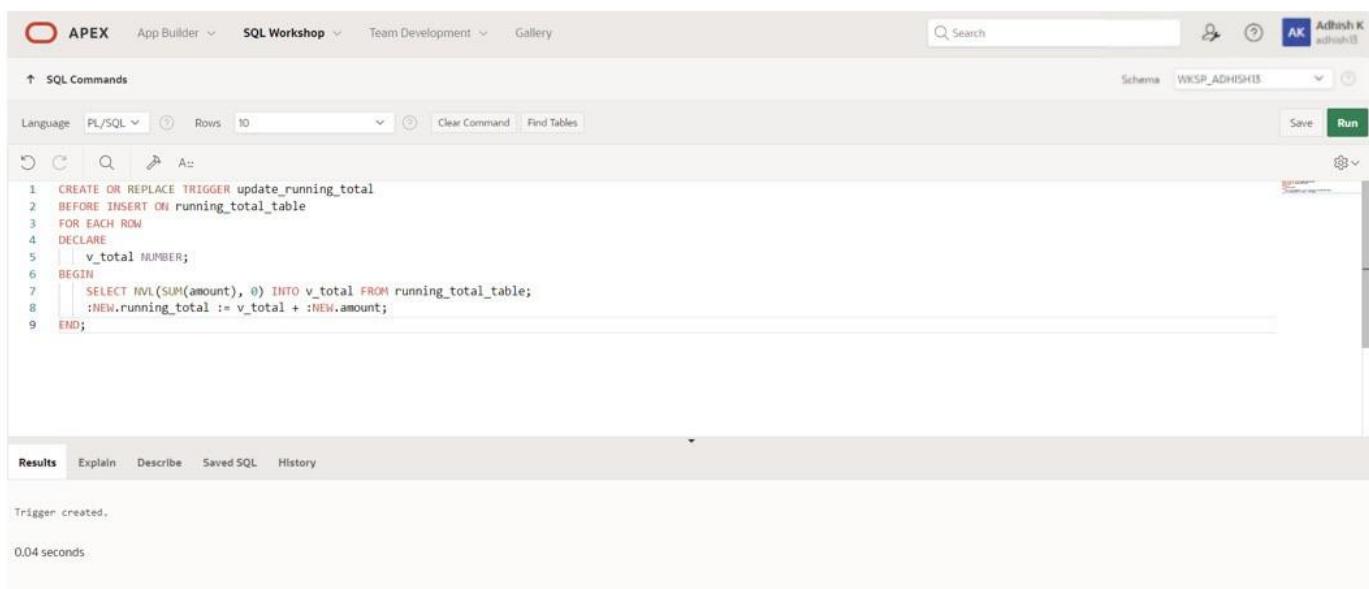
```
1 CREATE OR REPLACE TRIGGER check_duplicates
2 BEFORE INSERT OR UPDATE ON unique_values_table
3 FOR EACH ROW
4 DECLARE
5   duplicate_found EXCEPTION;
6   PRAGMA EXCEPTION_INIT(duplicate_found, -20002);
7   v_count NUMBER;
8 BEGIN
9   SELECT COUNT(*) INTO v_count FROM unique_values_table
10  WHERE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

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. The code for the trigger is entered in the command window. The schema is set to 'WKSP_ADHISH18'. The 'Run' button is visible at the bottom right of the command window. Below the command window, the 'Results' tab is active, showing the output: 'Trigger created.' and '0.04 seconds'.

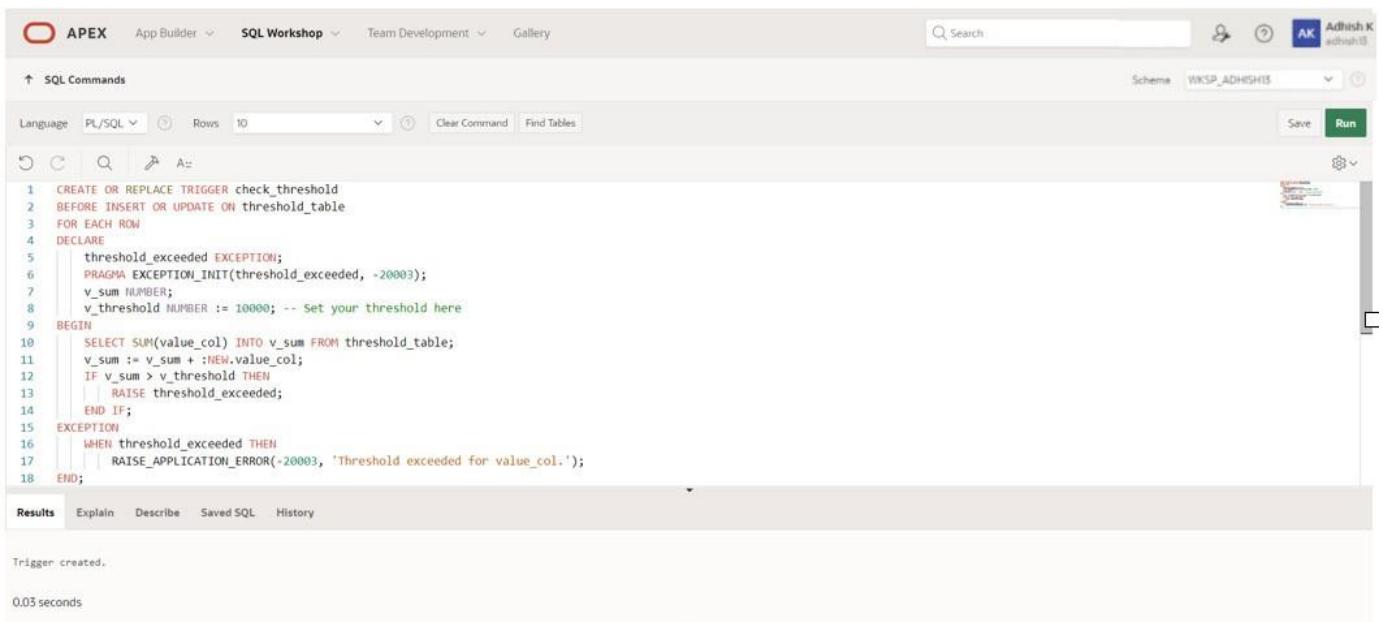
```
1 CREATE OR REPLACE TRIGGER update_running_total
2 BEFORE INSERT ON running_total_table
3 FOR EACH ROW
4 DECLARE
5     v_total NUMBER;
6 BEGIN
7     SELECT NVL(SUM(amount), 0) INTO v_total FROM running_total_table;
8     :NEW.running_total := v_total + :NEW.amount;
9 END;
```

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area displays the PL/SQL code for creating a trigger named 'check_threshold'. The code includes a declaration section with a threshold exceeded exception, a begin section with a sum variable and a threshold check, and an exception section that raises the application error if the threshold is exceeded. The 'Results' tab at the bottom shows the message 'Trigger created.' and a execution time of '0.03 seconds'.

```
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.');
18 END;
```

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

RESULT:

MONGO DB

EX_NO: 19

DATE:

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

QUERY:

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

OUTPUT:

```
adhish_13> db.restaurants.find({$and:[{ $or: [{cuisine: { $nin: ["American", "Chinese"] }}, {name: /^Wil/}]}]}, {restaurant_id: 1, name: 1, borough: 1, cuisine: 1})
[
  {
    _id: ObjectId('66571a0719aee0cbb8cdcdf6'),
    borough: 'Bronx',
    cuisine: 'Bakery',
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
adhish_13> |
```

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:

```
adhish_13> db.restaurants.find({ "grades": { $elemMatch: { "grade": "A", "score": 11, "date": ISODate("2014-08-11T00:00:00Z") } } }, { restaurant_id: 1, name: 1, grades: 1 })
adhish_13> |
```

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("2014-08-1T00:00:00Z") }, { restaurant_id: 1, name: 1, grades: 1 } );
```

OUTPUT:

```
adhish_13> 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 })  
adhish_13> |
```

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:

```
adhish_13> db.restaurants.find({$and : [{"address.coord.1": {$gt : 42}}, {"address.coord.1": {$lte : 52}}]}, {_id:0, restaurant_id:1, name:1, address:1})  
adhish_13> |
```

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:

```

adhish_13> db.restaurants.find({}, { _id: 0 }).sort({ name: 1 })
[
  {
    address: {
      building: '1007',
      coord: [ -73.856077, 40.848447 ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: ISODate('2014-03-03T00:00:00.000Z'),
        grade: 'A',
        score: 2
      },
      {
        date: ISODate('2013-09-11T00:00:00.000Z'),
        grade: 'A',
        score: 6
      },
      {
        date: ISODate('2013-01-24T00:00:00.000Z'),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate('2011-11-23T00:00:00.000Z'),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate('2011-03-10T00:00:00.000Z'),
        grade: 'B',
        score: 14
      }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
adhish_13> |

```

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:

```

adhish_13> db.restaurants.find({}, { _id: 0 }).sort({ name: -1 })
[
  {
    address: {
      building: '1007',
      coord: [ -73.856077, 40.848447 ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: ISODate('2014-03-03T00:00:00.000Z'),
        grade: 'A',
        score: 2
      },
      {
        date: ISODate('2013-09-11T00:00:00.000Z'),
        grade: 'A',
        score: 6
      },
      {
        date: ISODate('2013-01-24T00:00:00.000Z'),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate('2011-11-23T00:00:00.000Z'),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate('2011-03-10T00:00:00.000Z'),
        grade: 'B',
        score: 14
      }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
adhish_13> |

```

7.) Write a MongoDB query to arrange 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:

```
adhish_13> db.restaurants.find({}, { _id: 0 }).sort({ cuisine: 1, borough: -1 })
[ {
  address: {
    building: '1007',
    coord: [ -73.856077, 40.848447 ],
    street: 'Morris Park Ave',
    zipcode: '10462'
  },
  borough: 'Bronx',
  cuisine: 'Bakery',
  grades: [
    {
      date: ISODate('2014-03-03T00:00:00.000Z'),
      grade: 'A',
      score: 2
    },
    {
      date: ISODate('2013-09-11T00:00:00.000Z'),
      grade: 'A',
      score: 6
    },
    {
      date: ISODate('2013-01-24T00:00:00.000Z'),
      grade: 'A',
      score: 10
    },
    {
      date: ISODate('2011-11-23T00:00:00.000Z'),
      grade: 'A',
      score: 9
    },
    {
      date: ISODate('2011-03-10T00:00:00.000Z'),
      grade: 'B',
      score: 14
    }
  ],
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445'
}
]
adhish_13> |
```

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:

```

adhish_13> db.restaurants.find({ "address.street": { $exists: true, $ne: "" } })
[
  {
    _id: ObjectId('66571a0719aee0cbb8cdcdf6'),
    address: {
      building: '1007',
      coord: [ -73.856077, 40.848447 ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: ISODate('2014-03-03T00:00:00.000Z'),
        grade: 'A',
        score: 2
      },
      {
        date: ISODate('2013-09-11T00:00:00.000Z'),
        grade: 'A',
        score: 6
      },
      {
        date: ISODate('2013-01-24T00:00:00.000Z'),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate('2011-11-23T00:00:00.000Z'),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate('2011-03-10T00:00:00.000Z'),
        grade: 'B',
        score: 14
      }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
adhish_13> |

```

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:

```

adhish_13> db.restaurants.find({ "address.coord": { $elemMatch: { $type: "double" } } })
[
  {
    _id: ObjectId('66571a0719aee0cbb8cdcdf6'),
    address: {
      building: '1007',
      coord: [ -73.856077, 40.848447 ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: ISODate('2014-03-03T00:00:00.000Z'),
        grade: 'A',
        score: 2
      },
      {
        date: ISODate('2013-09-11T00:00:00.000Z'),
        grade: 'A',
        score: 6
      },
      {
        date: ISODate('2013-01-24T00:00:00.000Z'),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate('2011-11-23T00:00:00.000Z'),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate('2011-03-10T00:00:00.000Z'),
        grade: 'B',
        score: 14
      }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
adhish_13> |

```

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:

```
adhish_13> db.restaurants.find({ "grades.score": { $mod: [7, 0] } }, { restaurant_id: 1, name: 1, grades: 1 })
[
  {
    _id: ObjectId('66571a0719aee0cbb8cdcdf6'),
    grades: [
      {
        date: ISODate('2014-03-03T00:00:00.000Z'),
        grade: 'A',
        score: 2
      },
      {
        date: ISODate('2013-09-11T00:00:00.000Z'),
        grade: 'A',
        score: 6
      },
      {
        date: ISODate('2013-01-24T00:00:00.000Z'),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate('2011-11-23T00:00:00.000Z'),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate('2011-03-10T00:00:00.000Z'),
        grade: 'B',
        score: 14
      }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
adhish_13> |
```

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:

```
adhish_13> db.restaurants.find({ name: /mon/i }, { name: 1, borough: 1, "address.coord": 1, cuisine: 1 })  
adhish_13> |
```

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:

```
adhish_13> db.restaurants.find({ name: /^Mad/i }, { name: 1, borough: 1, "address.coord": 1, cuisine: 1 })  
adhish_13> |
```

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:

```
adhish_13> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } } })  
[  
  {  
    _id: ObjectId('66571a0719aee0cbb8cdcdf6'),  
    address: {  
      building: '1007',  
      coord: [ -73.856077, 40.848447 ],  
      street: 'Morris Park Ave',  
      zipcode: '10462'  
    },  
    borough: 'Bronx',  
    cuisine: 'Bakery',  
    grades: [  
      {  
        date: ISODate('2014-03-03T00:00:00.000Z'),  
        grade: 'A',  
        score: 2  
      },  
      {  
        date: ISODate('2013-09-11T00:00:00.000Z'),  
        grade: 'A',  
        score: 6  
      },  
      {  
        date: ISODate('2013-01-24T00:00:00.000Z'),  
        grade: 'A',  
        score: 10  
      },  
      {  
        date: ISODate('2011-11-23T00:00:00.000Z'),  
        grade: 'A',  
        score: 9  
      },  
      {  
        date: ISODate('2011-03-10T00:00:00.000Z'),  
        grade: 'B',  
        score: 14  
      }  
    ],  
    name: 'Morris Park Bake Shop',  
    restaurant_id: '30075445'  
  }  
]  
adhish_13> |
```

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:

```
adhish_13> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, "borough": "Manhattan" })
adhish_13> |
```

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:

```
adhish_13> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }] })
```

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

QUERY:

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

OUTPUT:

```
adhish_13> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $ne: "American" } })
adhish_13> |
```

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:

```
adhish_13> db.restaurants.find({ "grades": { $elemMatch: { "score": { $lt: 5 } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $nin: ["American", "Chinese"] } })
adhish_13> |
```

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:

```

adhish_13> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }] })
[
  {
    _id: ObjectId('66571a0719aee0cbb8cdcdf6'),
    address: {
      building: '1007',
      coord: [ -73.856977, 40.848447 ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: ISODate('2014-03-03T00:00:00.000Z'),
        grade: 'A',
        score: 2
      },
      {
        date: ISODate('2013-09-11T00:00:00.000Z'),
        grade: 'A',
        score: 6
      },
      {
        date: ISODate('2013-01-24T00:00:00.000Z'),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate('2011-11-23T00:00:00.000Z'),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate('2011-03-10T00:00:00.000Z'),
        grade: 'B',
        score: 14
      }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
adhish_13> |

```

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:

```

adhish_13> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], "borough": "Manhattan" })
adhish_13> |

```

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:

```
adhish_13> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "boroug
```

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:

```
adhish_13> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "boroug
```

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:

```
adhish_13> db.restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "boroug
```

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:

```
adhish_13> db.restaurants.find({ $or: [{ "grades.score": 2 }, { "grades.score": 6 }] })
[
  {
    _id: ObjectId('66571a0719aee0cbb8cdcdf6'),
    address: {
      building: '1007',
      coord: [ -73.856077, 40.848447 ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: ISODate('2014-03-03T00:00:00.000Z'),
        grade: 'A',
        score: 2
      },
      {
        date: ISODate('2013-09-11T00:00:00.000Z'),
        grade: 'A',
        score: 6
      },
      {
        date: ISODate('2013-01-24T00:00:00.000Z'),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate('2011-11-23T00:00:00.000Z'),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate('2011-03-10T00:00:00.000Z'),
        grade: 'B',
        score: 14
      }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
adhish_13> |
```

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

RESULT:

MONGO DB

EX_NO: 20

DATE:

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

QUERY:

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

OUTPUT:

```
adhish_13> db.movies.find({ year: 1893 })
adhish_13> |
```

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:

```
adhish_13> db.movies.find({ runtime: { $gt: 120 } })
adhish_13> |
```

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

QUERY:

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

OUTPUT:

```
adhish_13> db.movies.find({ genres: 'Short' })
adhish_13> |
```

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:

```
adhish_13> db.movies.find({ directors: 'William K.L. Dickson' })
adhish_13> |
```

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:

```
adhish_13> db.movies.find({ countries: 'USA' })
adhish_13> |
```

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

QUERY:

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

OUTPUT:

```
adhish_13> db.movies.find({ rated: 'UNRATED' })
adhish_13> |
```

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:

```
adhish_13> db.movies.find({ 'imdb.votes': { $gt: 1000 } })
adhish_13> |
```

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:

```
adhish_13> db.movies.find({ 'imdb.rating': { $gt: 7 } })
adhish_13> |
```

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:

```
adhish_13> db.movies.find({ 'tomatoes.viewer.rating': { $gt: 4 } })
adhish_13> |
```

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

QUERY:

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

OUTPUT:

```
adhish_13> db.movies.find({ 'awards.wins': { $gt: 0 } })
adhish_13> |
```

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:

```
adhish_13> 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 } )
adhish_13> |
```

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:

```
adhis...13> 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 })
adhis...13> |
```

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:

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:

```
adhis...13> db.movies.find( { title: /scene/i }, { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 })
adhis...13> |
```

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

RESULT: