

Creating and Managing Tables

DATE:

EX_NO:1

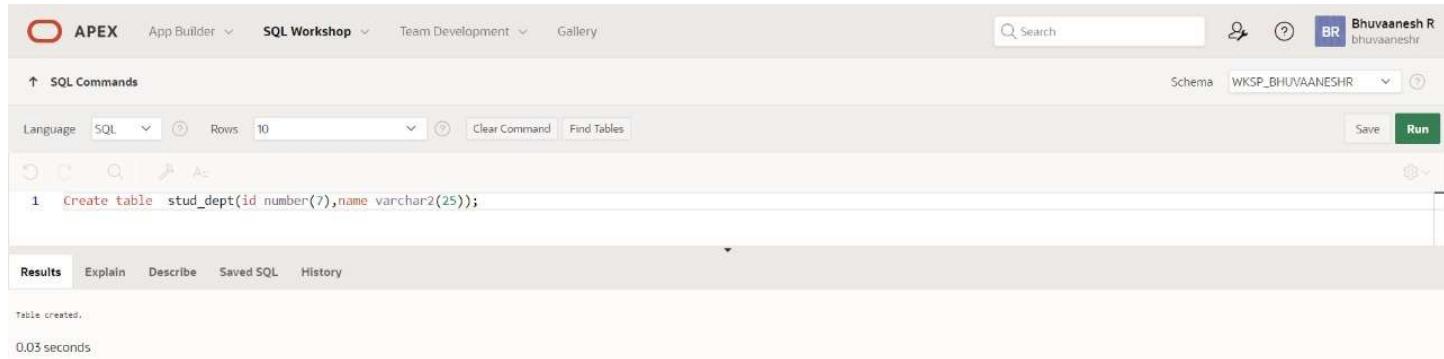
1. Create the DEPT table based on the DEPARTMENT following the table instance chart below. Confirm that the table is created.

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

QUERY:

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

OUTPUT:



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

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

The Results tab displays the output of the command:

```
Table created.  
0.05 seconds
```

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

OUTPUT:



A screenshot of 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 'Bhavaanesh R' (WKSP_BHUVAAINESHR). The main area is titled 'SQL Commands' with tabs for Language (SQL selected), Rows (10), Clear Command, and Find Tables. The command entered is:

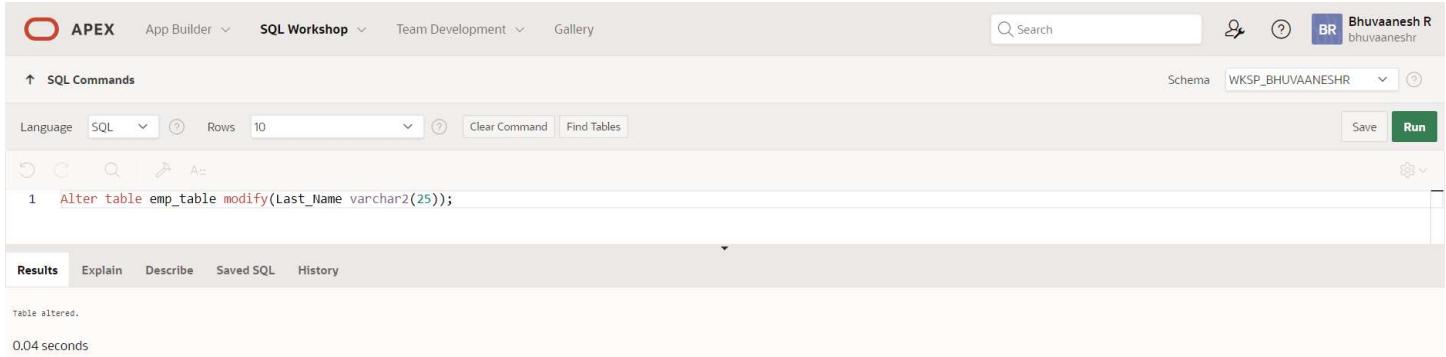
```
1 Create table emp_table(id number(7),Last_Name varchar2(25),First_Name varchar2(25),Dept_id number(7));
```

The results tab shows the output:

```
Table created.  
0.03 seconds
```

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

OUTPUT:



A screenshot of the Oracle SQL Workshop interface, similar to the previous one. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'Bhavaanesh R' (WKSP_BHUVAAINESHR). The main area is titled 'SQL Commands' with tabs for Language (SQL selected), Rows (10), Clear Command, and Find Tables. The command entered is:

```
1 Alter table emp_table modify(Last_Name varchar2(50));
```

The results tab shows the output:

```
Table altered.  
0.04 seconds
```

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, a user icon for 'Bhavaanesh R bhavaaneshr', and a schema dropdown set to 'WKSP_BHUVANESHR'. The main area is titled 'SQL Commands' with a 'Run' button. The code entered is:

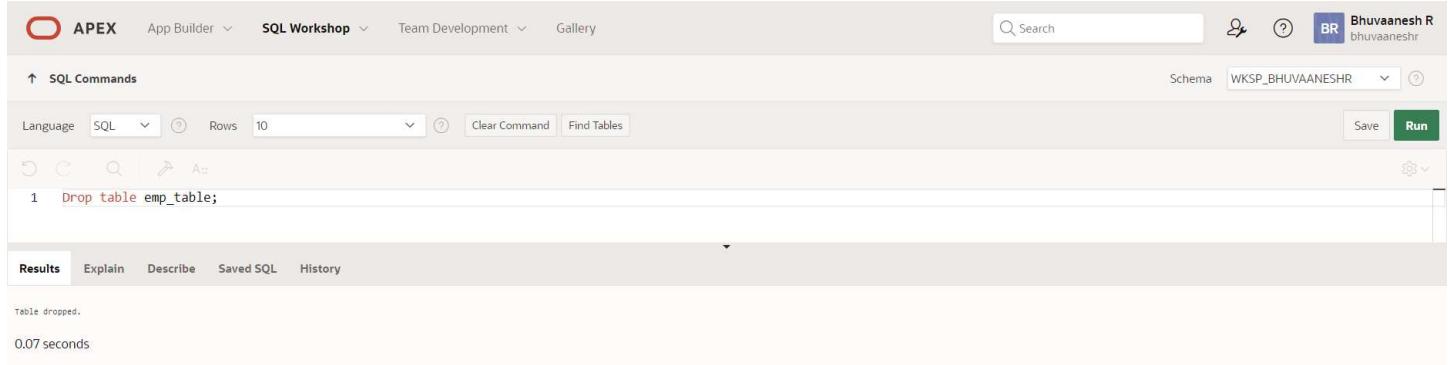
```
1 Create table employee_table(id number(7),first_name varchar2(25),Last_name varchar2(25),Salary int,Dept_id number(7));
```

The results section shows the output of the command:

```
Table created.  
0.03 seconds
```

5.Drop the EMP table.

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, a user icon for 'Bhavaanesh R bhavaaneshr', and a schema dropdown set to 'WKSP_BHUVANESHR'. The main area is titled 'SQL Commands' with a 'Run' button. The code entered is:

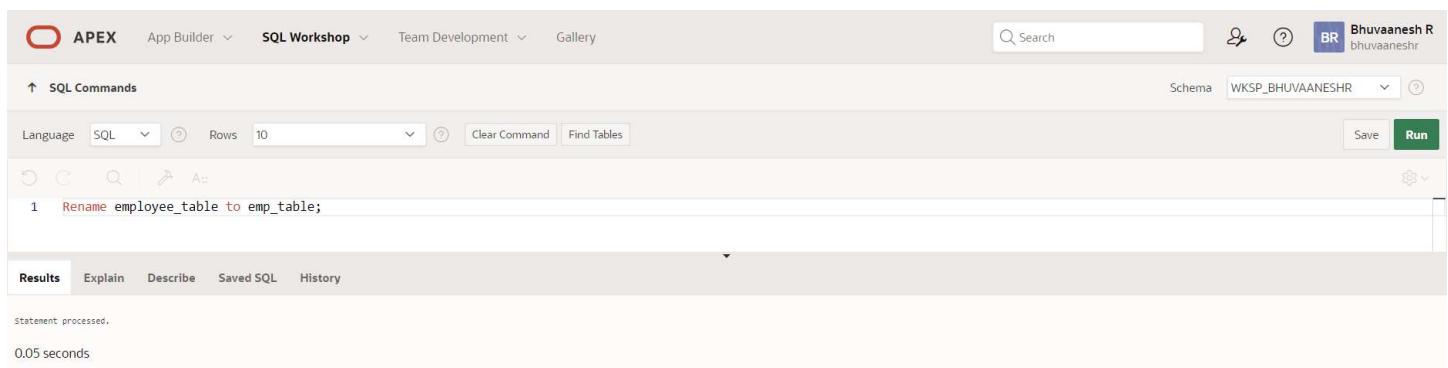
```
1 Drop table emp_table;
```

The results section shows the output of the command:

```
Table dropped.  
0.07 seconds
```

6.Rename the EMPLOYEES2 table as EMP.

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, a user icon for 'Bhavaanesh R bhavaaneshr', and a schema dropdown set to 'WKSP_BHUVANESHR'. The main area is titled 'SQL Commands' with a 'Run' button. The code entered is:

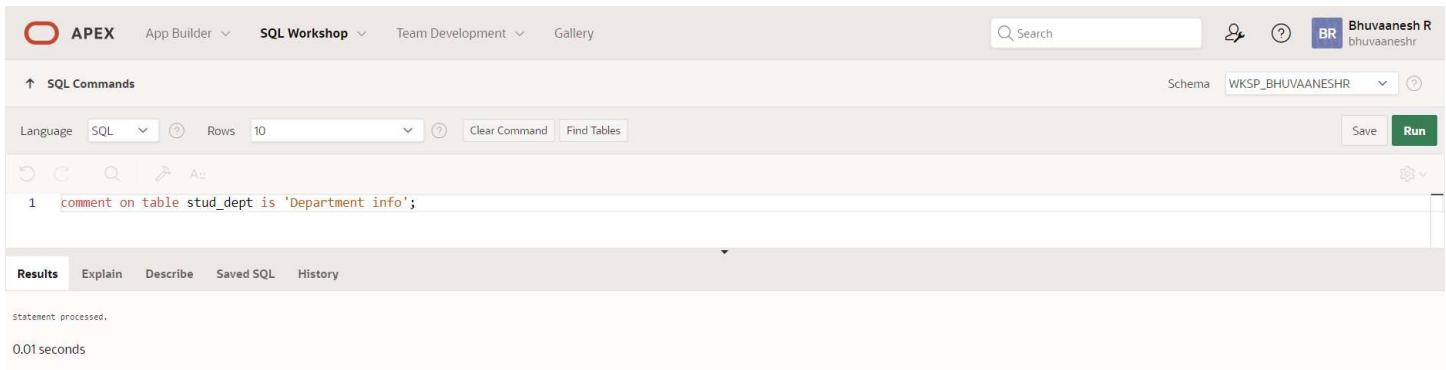
```
1 Rename employee_table to emp_table;
```

The results section shows the output of the command:

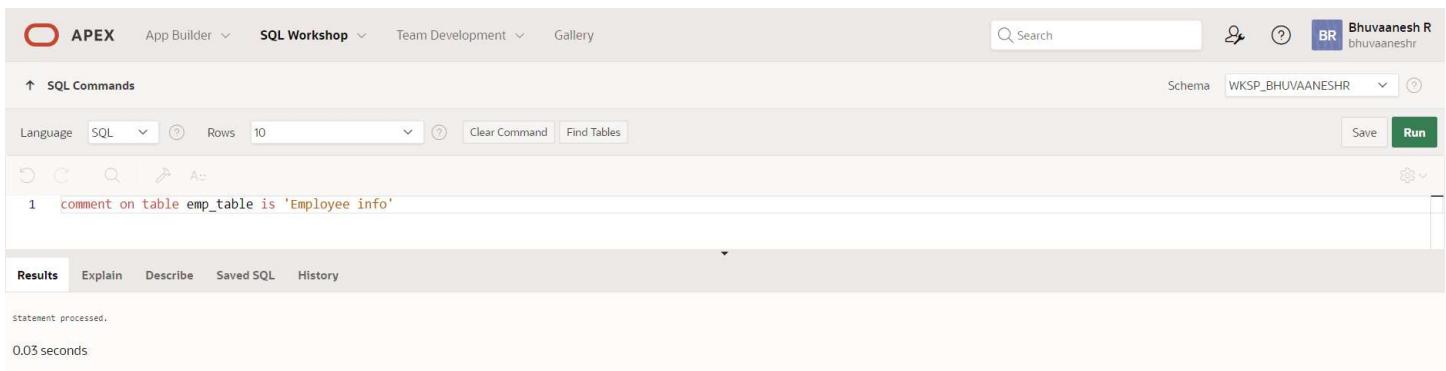
```
Statement processed.  
0.05 seconds
```

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

OUTPUT:



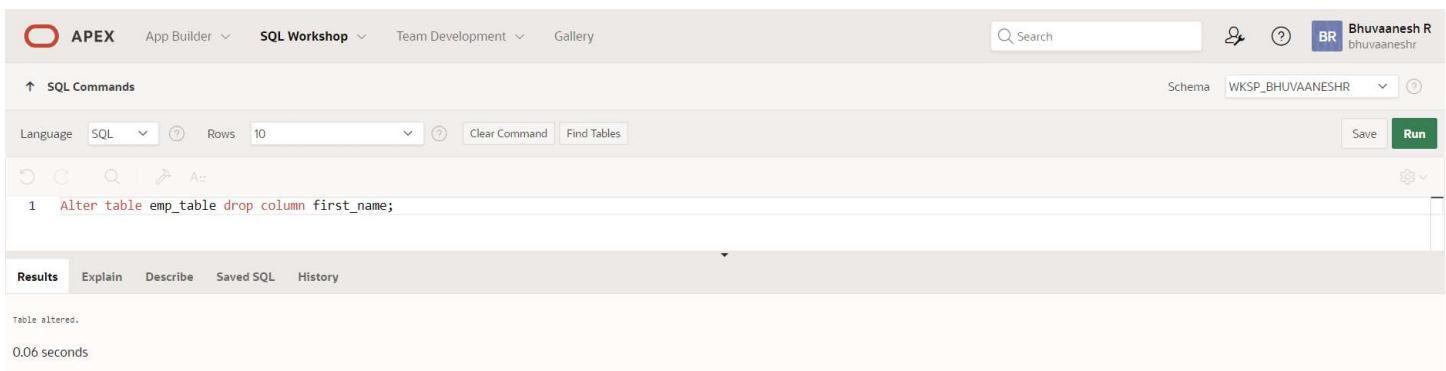
A screenshot of the Oracle SQL Workshop interface. The top navigation bar shows "APEX", "App Builder", "SQL Workshop", "Team Development", and "Gallery". The "SQL Workshop" tab is selected. The main area is titled "SQL Commands". The command entered is: `comment on table stud_dept is 'Department info';`. Below the command, the "Results" tab is selected, showing the output: "Statement processed." and "0.01 seconds". The schema is set to "WKSP_BHUVANESHR".



A screenshot of the Oracle SQL Workshop interface. The top navigation bar shows "APEX", "App Builder", "SQL Workshop", "Team Development", and "Gallery". The "SQL Workshop" tab is selected. The main area is titled "SQL Commands". The command entered is: `comment on table emp_table is 'Employee info'.`. Below the command, the "Results" tab is selected, showing the output: "Statement processed." and "0.03 seconds". The schema is set to "WKSP_BHUVANESHR".

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

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The top navigation bar shows "APEX", "App Builder", "SQL Workshop", "Team Development", and "Gallery". The "SQL Workshop" tab is selected. The main area is titled "SQL Commands". The command entered is: `Alter table emp_table drop column first_name;`. Below the command, the "Results" tab is selected, showing the output: "Table altered." and "0.06 seconds". The schema is set to "WKSP_BHUVANESHR".

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

RESULT:

MANIPULATING DATA

DATE:
EX_NO:2

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)

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a user profile for 'Bhuvanesh R' (bhuvaneshr). The main area is titled 'SQL Commands'. It has tabs for Language (set to SQL), Rows (set to 10), and buttons for Clear Command and Find Tables. Below these is a command input field containing the SQL code to create the 'MYEMPLOYEE' table. The code is: 'CREATE TABLE MYEMPLOYEE(ID NUMBER(7) NOT NULL, LASTNAME VARCHAR2(25), FIRSTNAME VARCHAR2(25), USERID VARCHAR2(25), SALARY NUMBER(9,2))'. The 'Results' tab is selected, showing the output: 'Table created.' and '0.05 seconds'. There are also tabs for Explain, Describe, Saved SQL, and History.

```
1 CREATE TABLE MYEMPLOYEE(ID NUMBER(7) NOT NULL, LASTNAME VARCHAR2(25), FIRSTNAME VARCHAR2(25), USERID VARCHAR2(25), SALARY NUMBER(9,2))
```

Table created.
0.05 seconds

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, there is a single line of SQL code:

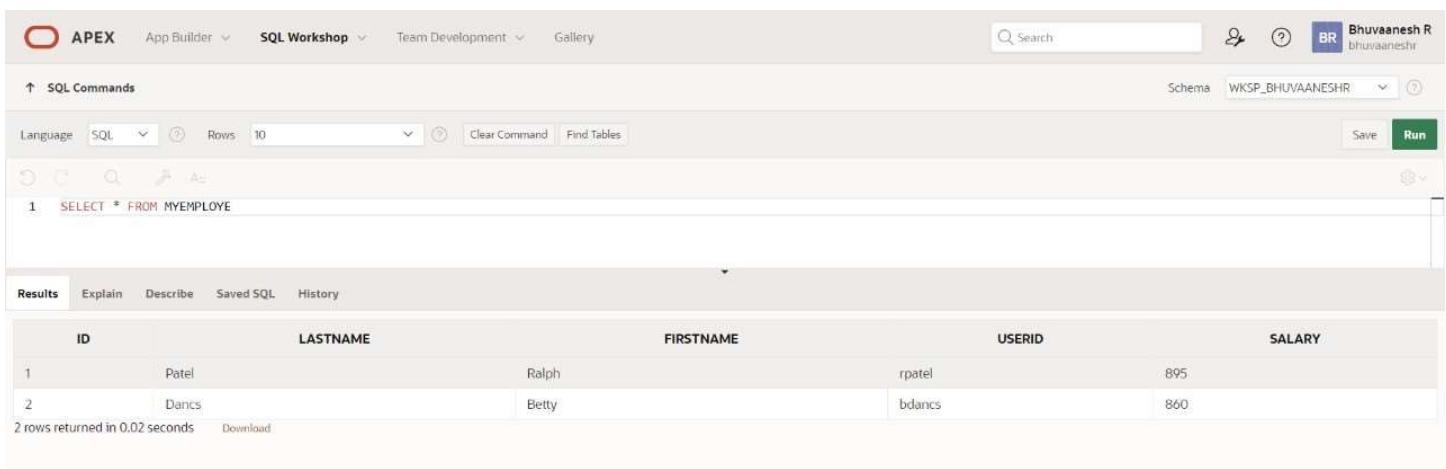
```
1 INSERT INTO MYEMPLOYEE(ID,LASTNAME,FIRSTNAME,USERID,SALARY) VALUES(1,'Patel','Ralph','rpatel',895)
```

In the Results pane, the output is:

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

3.Display the table with values.

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, there is a single line of SQL code:

```
1 SELECT * FROM MYEMPLOYEE
```

In the Results pane, the output is a table displaying the data from the MYEMPLOYEE table:

ID	LASTNAME	FIRSTNAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860

Below the table, it says "2 rows returned in 0.02 seconds".

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.

OUTPUT:



A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar shows "APEX", "App Builder", "SQL Workshop", "Team Development", and "Gallery". The "Schema" dropdown is set to "WKSP_BHUVAAINESHR". The main area is titled "SQL Commands" with a "Language" dropdown set to "SQL". The query editor contains the following SQL statement:

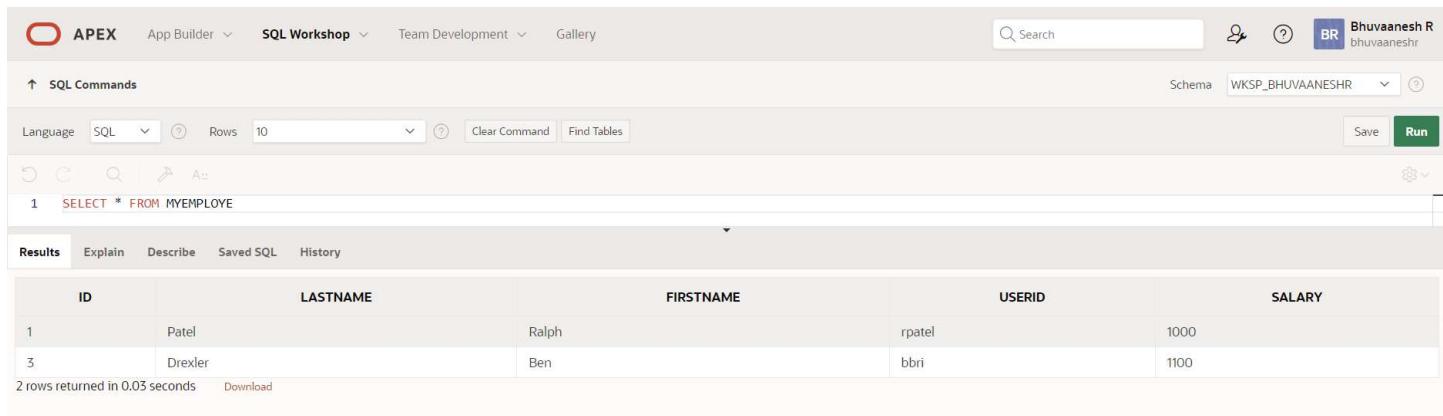
```
1 INSERT INTO MYEMPLOYEE(ID, LASTNAME, FIRSTNAME, USERID, SALARY) VALUES(3, 'Biri', 'Ben', 'bbri', 1100)
```

The results tab shows the output of the query:

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

5.Make the data additions permanent.

OUTPUT:



A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar shows "APEX", "App Builder", "SQL Workshop", "Team Development", and "Gallery". The "Schema" dropdown is set to "WKSP_BHUVAAINESHR". The main area is titled "SQL Commands" with a "Language" dropdown set to "SQL". The query editor contains the following SQL statement:

```
1 SELECT * FROM MYEMPLOYEE
```

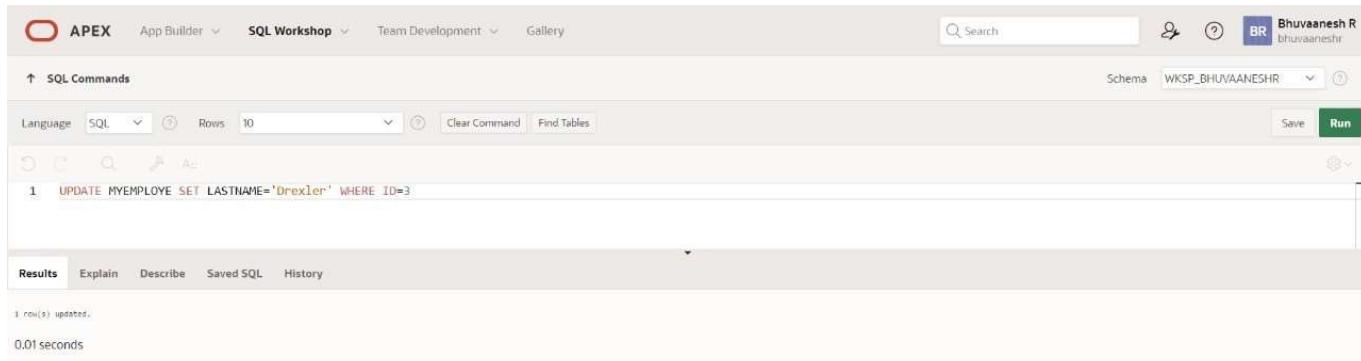
The results tab displays the data from the "MYEMPLOYEE" table:

ID	LASTNAME	FIRSTNAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
3	Drexler	Ben	bbri	1100

2 rows returned in 0.03 seconds

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

OUTPUT:



A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar shows "APEX", "App Builder", "SQL Workshop", "Team Development", and "Gallery". The "Schema" dropdown is set to "WKSP_BHUVAAINESHR". The main area is titled "SQL Commands" with a "Language" dropdown set to "SQL". The query editor contains the following SQL statement:

```
1 UPDATE MYEMPLOYEE SET LASTNAME='Drexler' WHERE ID=3
```

The results tab shows the output of the query:

```
3 row(s) updated.  
0.01 seconds
```

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

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile 'Bhavaanesh R bhavaaneshr' are on the right. The main area shows a SQL command line with the following content:

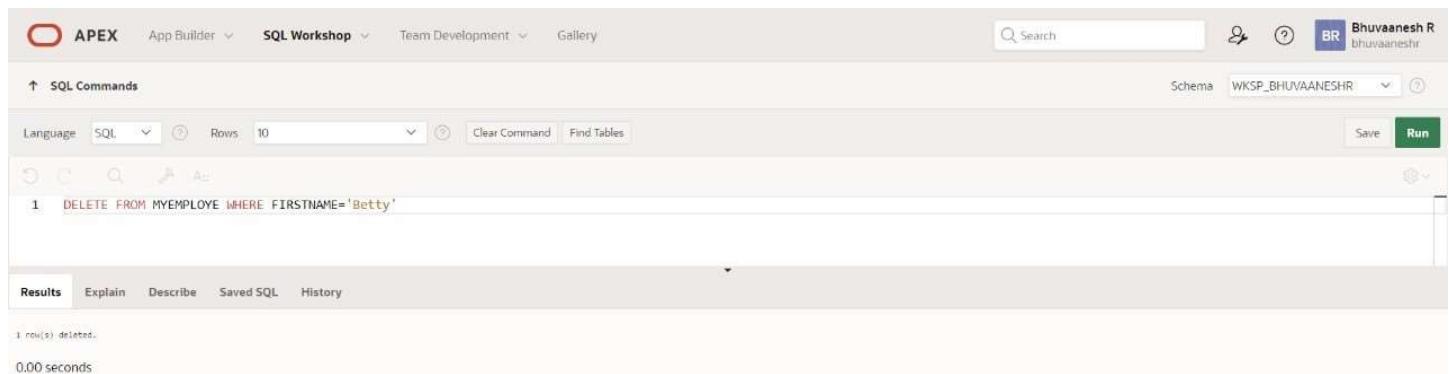
```
1 UPDATE MYEMPLOYEE SET SALARY=1000 WHERE SALARY<900
```

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

```
3 row(s) updated.  
0.01 seconds
```

8.Delete Betty dancs from MY_EMPLOYEE table.

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile 'Bhavaanesh R bhavaaneshr' are on the right. The main area shows a SQL command line with the following content:

```
1 DELETE FROM MYEMPLOYEE WHERE FIRSTNAME='Betty'
```

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

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

9.Empty the fourth row of the emp table.

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile 'Bhavaanesh R bhavaaneshr' are on the right. The main area shows a SQL command line with the following content:

```
1 DELETE FROM MYEMPLOYEE WHERE ID=4
```

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

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

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

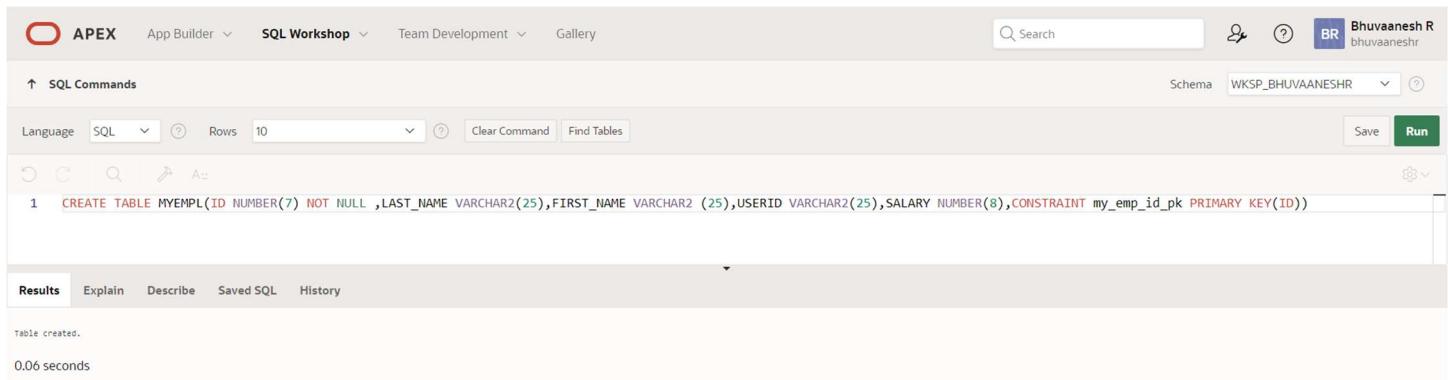
RESULT:

INCLUDING CONSTRAINTS

DATE:
EX_NO:3

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.

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, a user icon for 'Bhavaanesh R bhavaaneshr', and a 'Run' button. The main workspace shows a SQL command line with the following code:

```
1 CREATE TABLE MYEMPL(ID NUMBER(7) NOT NULL ,LAST_NAME VARCHAR2(25),FIRST_NAME VARCHAR2 (25),USERID VARCHAR2(25),SALARY NUMBER(8),CONSTRAINT my_emp_id_pk PRIMARY KEY(ID))
```

The results tab shows the output: "Table created." and "0.06 seconds".

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

OUTPUT:



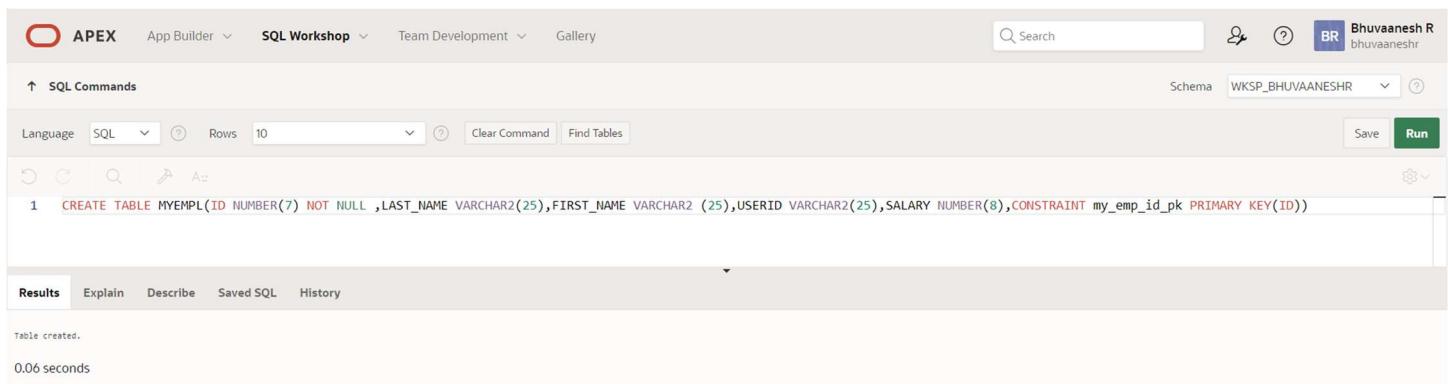
A screenshot of the Oracle SQL Workshop interface, similar to the previous one. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, a user icon for 'Bhavaanesh R bhavaaneshr', and a 'Run' button. The main workspace shows a SQL command line with the following code:

```
1 CREATE TABLE DEPT_TABLE(ID NUMBER(6),DEPT_NAME VARCHAR2(25),MANAGER_ID NUMBER(6),LOCATION_ID NUMBER(4),CONSTRAINT my_dept_id_pk PRIMARY KEY(ID))
```

The results tab shows the output: "Table created." and "0.06 seconds".

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

OUTPUT:



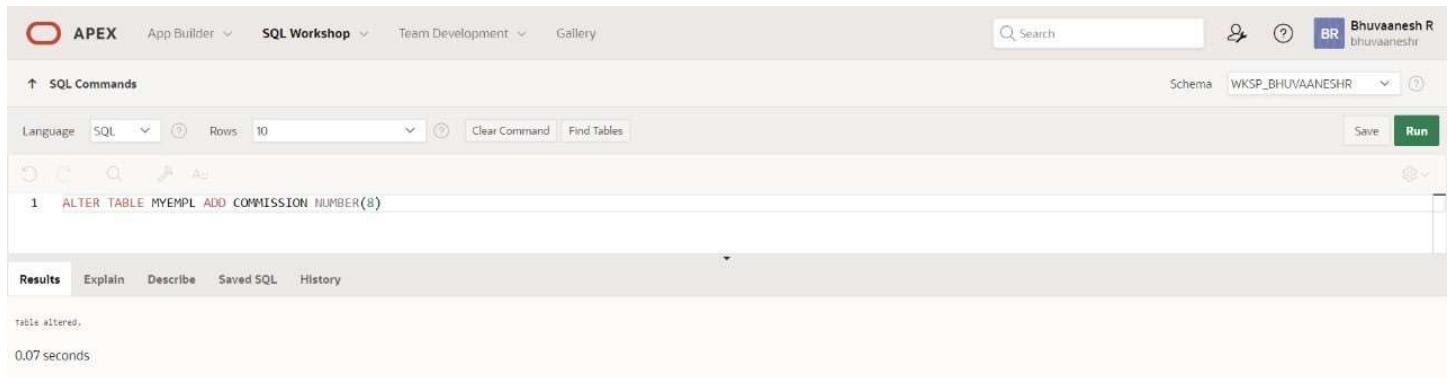
The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown is set to WKSP_BHUVANESHR. The main area displays the following SQL command:

```
1 CREATE TABLE MYEMPL(ID NUMBER(7) NOT NULL ,LAST_NAME VARCHAR2(25),FIRST_NAME VARCHAR2 (25),USERID VARCHAR2(25),SALARY NUMBER(8),CONSTRAINT my_emp_id_pk PRIMARY KEY(ID))
```

The results section shows the message "Table created." and a execution time of "0.06 seconds".

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

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_BHUVANESHR. The main area displays the following SQL command:

```
1 ALTER TABLE MYEMPL ADD COMMISSION NUMBER(8)
```

The results section shows the message "Table altered." and a execution time of "0.07 seconds".

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

RESULT:

Writing Basic SQL SELECT Statements

DATE:
EX_NO:4

1.The following statement executes successfully.

Identify the Errors

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

OUTPUT:

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

```
1 SELECT EMPLOYEE_ID, LAST_NAME, FIRST_NAME, "SALARY" * 12 "ANNUAL_SALARY" FROM EMPLOYEE
```

The Results tab displays the output:

EMPLOYEE_ID	LAST_NAME	FIRST_NAME	ANNUAL_SALARY
157	BHARATH	BEST	180000
541	PATEL	SUNIL	144000
143	KUMAR	BHARATH	171600

3 rows returned in 0.01 seconds

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

OUTPUT:

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

```
1 DESC DEPARTMENT
```

The Results tab displays the description of the DEPARTMENT table:

Object Type	TABLE	Object	DEPARTMENT						
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPARTMENT	DEPT_ID	NUMBER	-	6	0	-	✓	-	-
	DEPT_NAME	VARCHAR2	25	-	-	-	✓	-	-
	MANAGER_ID	VARCHAR2	25	-	-	-	✓	-	-
	LOCATION_ID	NUMBER	-	25	0	-	✓	-	-

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

OUTPUT:

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

```
1 SELECT EMPL_NUMBER, LAST_NAME, JOB_CODE FROM EMPLOYEE
```

The results are displayed in a table:

EMPL_NUMBER	LAST_NAME	JOB_CODE
42	BHARATH	542
225	PATEL	582
12	KUMAR	457

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

4.Provide an alias STARTDATE for the hire date.

OUTPUT:

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

```
1 SELECT HIRE_DATE AS START_DATE FROM EMPLOYEE
```

The results are displayed in a table:

START_DATE
02/20/2016
03/01/2017
06/28/2014

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

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

OUTPUT:

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

```
1 SELECT DISTINCT JOB_CODE FROM EMPLOYEE
```

The results are displayed in a table:

JOB_CODE
457
542
582

3 rows returned in 0.01 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.

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, a user icon for 'Bhuvaanesh R bhuyaaneshr', and a 'Run' button. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL tab is selected, showing the following query:

```
1. SELECT LAST_NAME || ',' || JOB_CODE AS "EMPLOYEE AND TITLE" FROM EMPLOYEE
```

The Results tab shows the output:

EMPLOYEE AND TITLE

BHARATH,542
PATEL,582
KUMAR,457

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

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:

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface, similar to the previous one. The top navigation bar and user information are the same. The SQL tab is selected, showing the following query:

```
1. SELECT LAST_NAME || ',' || JOB_CODE || ',' || EMPL_NUMBER || ',' || HIRE_DATE AS "THE_OUTPUT" FROM EMPLOYEE
```

The Results tab shows the output:

THE_OUTPUT

BHARATH,542,02/20/2016
PATEL,582,225,03/01/2017
KUMAR,457,12,06/28/2014

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

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.

OUTPUT:

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

```
1 SELECT LAST_NAME, SALARY FROM MYEMPL WHERE SALARY > 12000
```

The results table displays the following data:

LAST_NAME	SALARY
Bharathkumar	14300
Bhuvaanesh	55000
Coder	13000

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

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

OUTPUT:

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

```
1 SELECT LAST_NAME, DEPARTMENT_NUMBER FROM MYEMPL WHERE DEPARTMENT_NUMBER = 176
```

The results table displays the following data:

DEPARTMENT_NUMBER
Bhuvaanesh
Ragul

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

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)

OUTPUT:

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

```
1 SELECT LAST_NAME, SALARY FROM MYEMPL WHERE SALARY NOT BETWEEN 5000 AND 12000
```

The results table displays the following data:

LAST_NAME	SALARY
Bharathkumar	14300
Bhuvaanesh	55000
Coder	13000

3 rows returned in 0.03 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)

OUTPUT:

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

```
1 SELECT LAST_NAME, JOB_ID, START_DATE FROM MYEMPL WHERE START_DATE BETWEEN '2-20-1998' AND '05-01-1998' ORDER BY START_DATE ASC
```

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

LAST_NAME	JOB_ID	START_DATE
Bharathkumar	145	03/14/1998
Coder	42	04/01/1998
Ragul	284	04/07/1998

3 rows returned in 0.00 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)

OUTPUT:

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

```
1 SELECT LAST_NAME, DEPARTMENT_NUMBER FROM MYEMPL WHERE DEPARTMENT_NUMBER IN (20,50) ORDER BY LAST_NAME ASC
```

The results table has columns LAST_NAME and DEPARTMENT_NUMBER. The data is:

LAST_NAME	DEPARTMENT_NUMBER
Bharathkumar	20
Bhavaanesh	50
Coder	20
Ragul	50

4 rows returned in 0.01 seconds

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

OUTPUT:

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

```
1 SELECT LAST_NAME AS EMPLOYEE, SALARY AS EMP_SAL FROM MYEMPL WHERE DEPARTMENT_NUMBER IN (20,50) AND SALARY BETWEEN 5000 AND 12000 ORDER BY LAST_NAME ASC
```

The results table has columns LAST_NAME and DEPARTMENT_NUMBER. The data is:

LAST_NAME	DEPARTMENT_NUMBER
Bharathkumar	20
Bhavaanesh	50
Coder	20
Ragul	50

4 rows returned in 0.01 seconds

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

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The schema is set to WKSP_BHUVANESHR. A query is run: `SELECT LAST_NAME, HIRE_DATE FROM MYEMPL WHERE HIRE_DATE LIKE '%94'`. The results show one row: Ani, hired on 02/20/1994.

LAST_NAME	HIRE_DATE
Ani	02/20/1994

1 rows returned in 0.01 seconds

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

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The schema is set to WKSP_BHUVANESHR. A query is run: `SELECT LAST_NAME, JOB_TITLE FROM MYEMPL WHERE MANAGER_NAME IS NULL`. The results show two rows: Ragul (Full Stack Developer) and Coder (Coder).

LAST_NAME	JOB_TITLE
Ragul	Full Stack Developer
Coder	Coder

2 rows returned in 0.00 seconds

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

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The schema is set to WKSP_BHUVANESHR. A query is run: `SELECT LAST_NAME, SALARY, COMMISSION FROM MYEMPL WHERE COMMISSION IS NOT NULL ORDER BY SALARY DESC, COMMISSION DESC`. The results show two rows: Bhuvanesh (55000, 100) and BharathKumar (14300, 500), sorted by salary in descending order and commission in descending order.

LAST_NAME	SALARY	COMMISSION
Bhuvanesh	55000	100
BharathKumar	14300	500

2 rows returned in 0.00 seconds

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

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is: `SELECT LAST_NAME FROM MYEMPL WHERE LAST_NAME LIKE '_a%'`. The results show one row: Bharathkumar.

LAST_NAME
Bharathkumar

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

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is: `SELECT LAST_NAME FROM MY_EMPL WHERE LAST_NAME LIKE '%a%' AND LAST_NAME LIKE '%e%'`. The results show one row: Patel.

LAST_NAME
Patel

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)

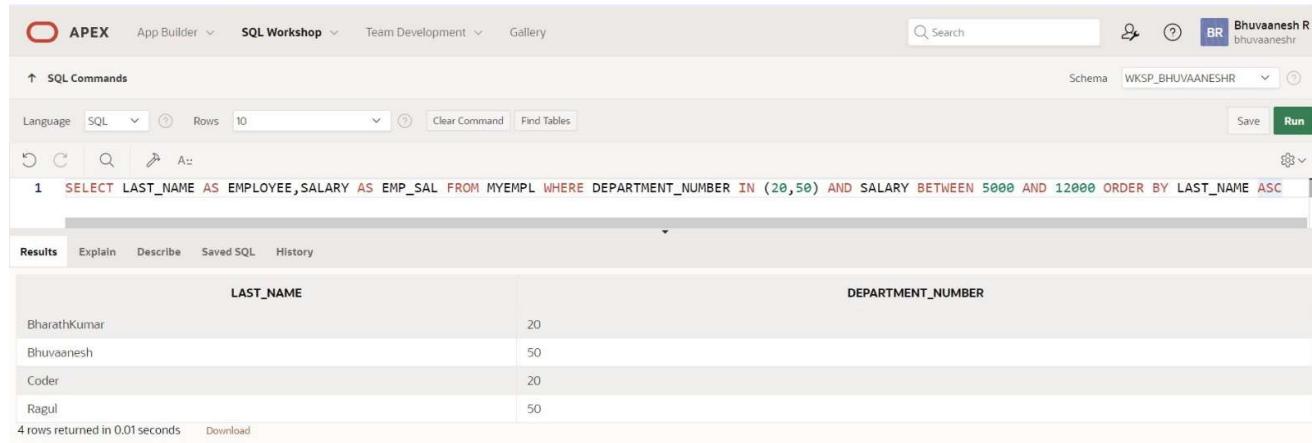
OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is: `SELECT LAST_NAME, SALARY, COMMISSION FROM MYEMPL WHERE COMMISSION=0.2`. The results show three rows: Bharathkumar, Bhuyaanesh, and Coder.

LAST_NAME	SALARY	COMMISSION
Bharathkumar	14300	.2
Bhuyaanesh	55000	.2
Coder	13000	.2

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, user information for 'Bhuvanesh R bhuvaneshr', and a schema dropdown set to 'WKSP_BHUVANESHR'. Below the navigation is a toolbar with 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', 'Find Tables', and 'Run' buttons. The main area displays a SQL command and its results.

```
1 SELECT LAST_NAME AS EMPLOYEE, SALARY AS EMP_SAL FROM MYEMPL WHERE DEPARTMENT_NUMBER IN (20,50) AND SALARY BETWEEN 5000 AND 12000 ORDER BY LAST_NAME ASC
```

The results table has two columns: 'LAST_NAME' and 'DEPARTMENT_NUMBER'. The data is as follows:

LAST_NAME	DEPARTMENT_NUMBER
BharathKumar	20
Bhuvanesh	50
Coder	20
Ragul	50

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

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

RESULT:

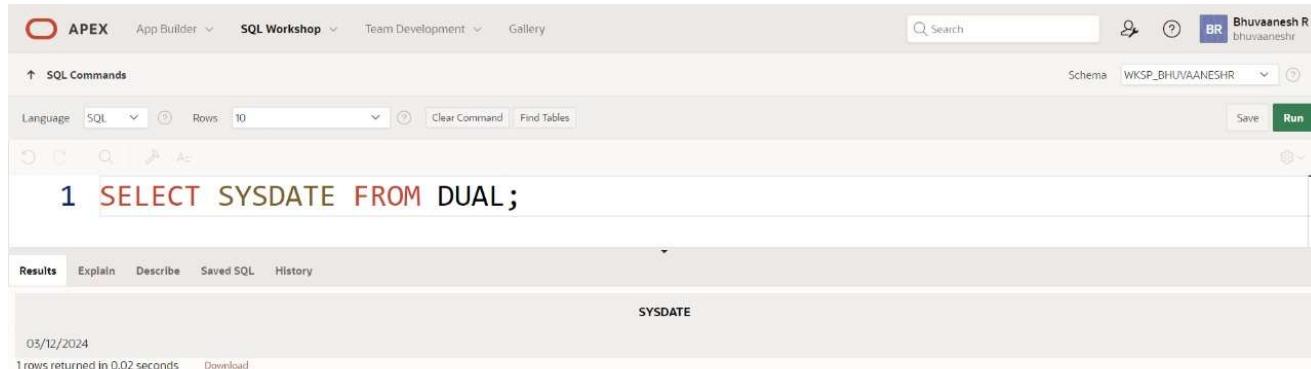
SINGLE ROW FUNCTIONS

EX_NO:6

DATE:

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

OUTPUT:



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

```
1 SELECT SYSDATE FROM DUAL;
```

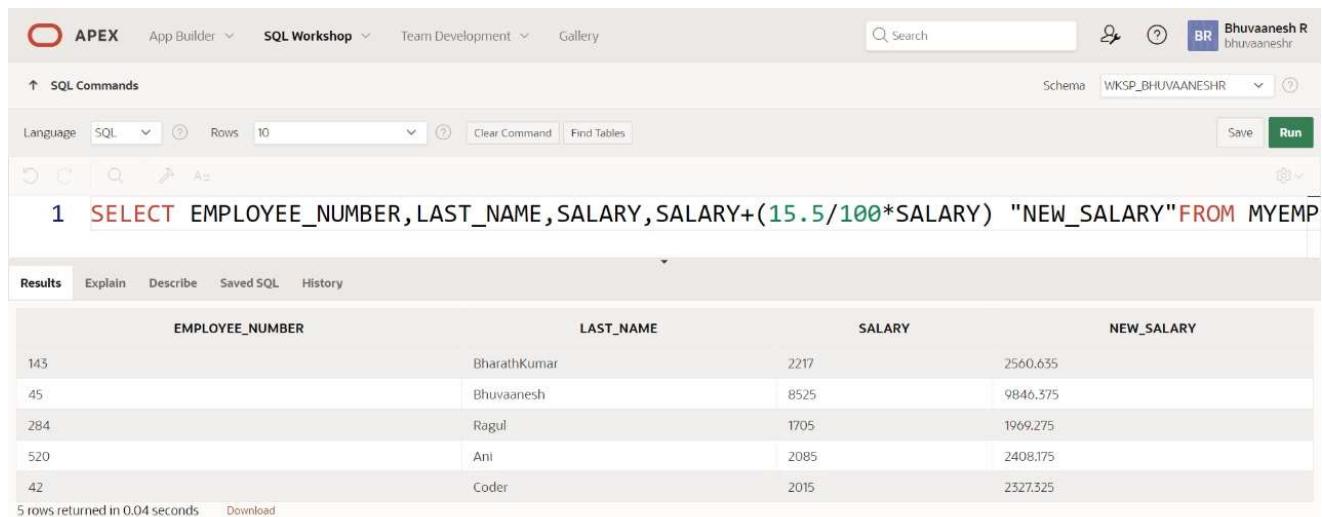
The results section shows the output:

SYSDATE
03/12/2024

1 rows returned in 0.02 seconds

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

OUTPUT:



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

```
1 SELECT EMPLOYEE_NUMBER, LAST_NAME, SALARY, SALARY+(15.5/100*SALARY) "NEW_SALARY" FROM MYEMP
```

The results section shows the output:

EMPLOYEE_NUMBER	LAST_NAME	SALARY	NEW_SALARY
145	BharathKumar	2217	2560.635
45	Bhavaanesh	8525	9846.375
284	Ragul	1705	1969.275
520	Ani	2085	2408.175
42	Coder	2015	2327.325

5 rows returned in 0.04 seconds

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

OUTPUT:

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

```
SELECT EMPLOYEE_NUMBER, LAST_NAME, SALARY, SALARY+(15.5/100*SALARY) "NEW_SALARY", NEW_SALARY-SALARY as "INCREASE" FROM MYEMPL;
```

The results section displays the following data:

EMPLOYEE_NUMBER	LAST_NAME	SALARY	NEW_SALARY	INCREASE
143	BharathKumar	2217	2560.635	783
45	Bhuvaanesh	8525	9846.375	475
284	Ragul	1705	1969.275	795
520	Ani	2085	2408.175	915
42	Coder	2015	2327.325	1485

5 rows returned in 0.01 seconds

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

OUTPUT:

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

```
SELECT INITCAP(LAST_NAME), LENGTH(LAST_NAME) AS "LENGTH_OF_LAST_NAME" FROM MYEMPL WHERE LAST_NAME LIKE 'J%' OR LAST_NAME LIKE 'A%' OR LAST_NAME LIKE 'M%' ORDER BY LAST_NAME;
```

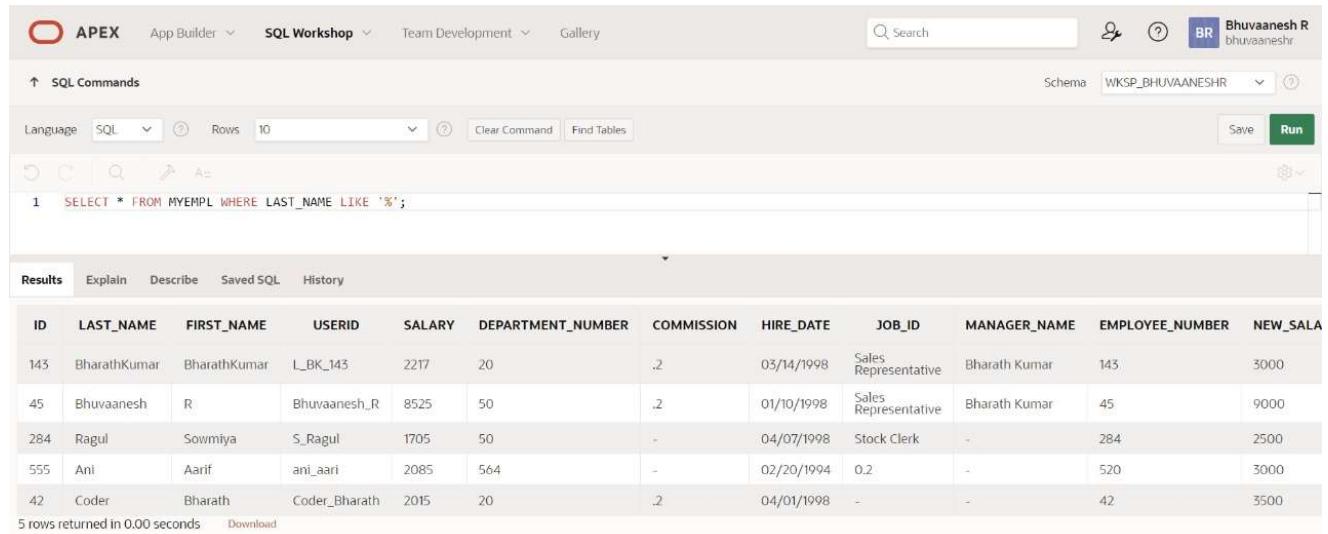
The results section displays the following data:

INITCAP(LAST_NAME)	LENGTH_OF_LAST_NAME
Ani	3

1 rows returned in 0.02 seconds

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

OUTPUT:



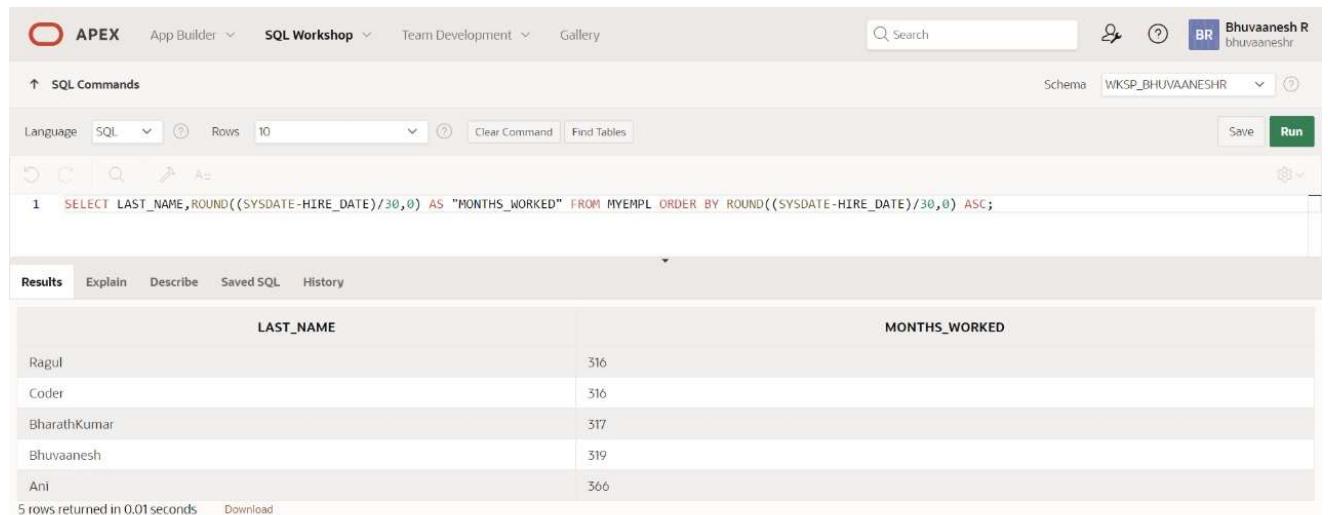
The screenshot shows the Oracle SQL Workshop interface. The schema is set to WKSP_BHUVAAINESHR. A SQL command is entered: `SELECT * FROM MYEMPL WHERE LAST_NAME LIKE '%'`. The results table displays all employees from the MYEMPL table.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY	DEPARTMENT_NUMBER	COMMISSION	HIRE_DATE	JOB_ID	MANAGER_NAME	EMPLOYEE_NUMBER	NEW_SALA
143	BharathKumar	BharathKumar	L_BK_143	2217	20	.2	03/14/1998	Sales Representative	Bharath Kumar	143	3000
45	Bhavaanesh	R	Bhavaanesh_R	8525	50	.2	01/10/1998	Sales Representative	Bharath Kumar	45	9000
284	Ragul	Sowmiya	S_Ragul	1705	50	-	04/07/1998	Stock Clerk	-	284	2500
555	Anil	Aarif	ani_aari	2085	564	-	02/20/1994	0.2	-	520	3000
42	Coder	Bharath	Coder_Bharath	2015	20	.2	04/01/1998	-	-	42	3500

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

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.

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The schema is set to WKSP_BHUVAAINESHR. A SQL command is entered: `SELECT LAST_NAME,ROUND((SYSDATE-HIRE_DATE)/30,0) AS "MONTHS_WORKED" FROM MYEMPL ORDER BY ROUND((SYSDATE-HIRE_DATE)/30,0) ASC;`. The results table displays the last name and the calculated number of months worked for each employee.

LAST_NAME	MONTHS_WORKED
Ragul	316
Coder	316
BharathKumar	317
Bhavaanesh	319
Anil	366

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

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.

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile 'Bhuvaanesh R' are also present. The main area is titled 'SQL Commands' with a sub-section 'Results'. The SQL command entered is:

```
1 SELECT LAST_NAME||' EARNS '||SALARY||' MONTHLY BUT WANTS '||SALARY*3 AS "DREAM_SALARIES" FROM MYEMPL;
```

The results section displays the output:

DREAM_SALARIES	
BharathKumar	EARNS 2217 MONTHLY BUT WANTS 6651
Bhuvaanesh	EARNS 8525 MONTHLY BUT WANTS 25575
Ragul	EARNS 1705 MONTHLY BUT WANTS 5115
Ani	EARNS 2085 MONTHLY BUT WANTS 6255
Coder	EARNS 2015 MONTHLY BUT WANTS 6045

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

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

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile 'Bhuvaanesh R' are also present. The main area is titled 'SQL Commands' with a sub-section 'Results'. The SQL command entered is:

```
1 SELECT LAST_NAME,LPAD(SALARY,15,'$') as "SALARY" FROM MYEMPL;
```

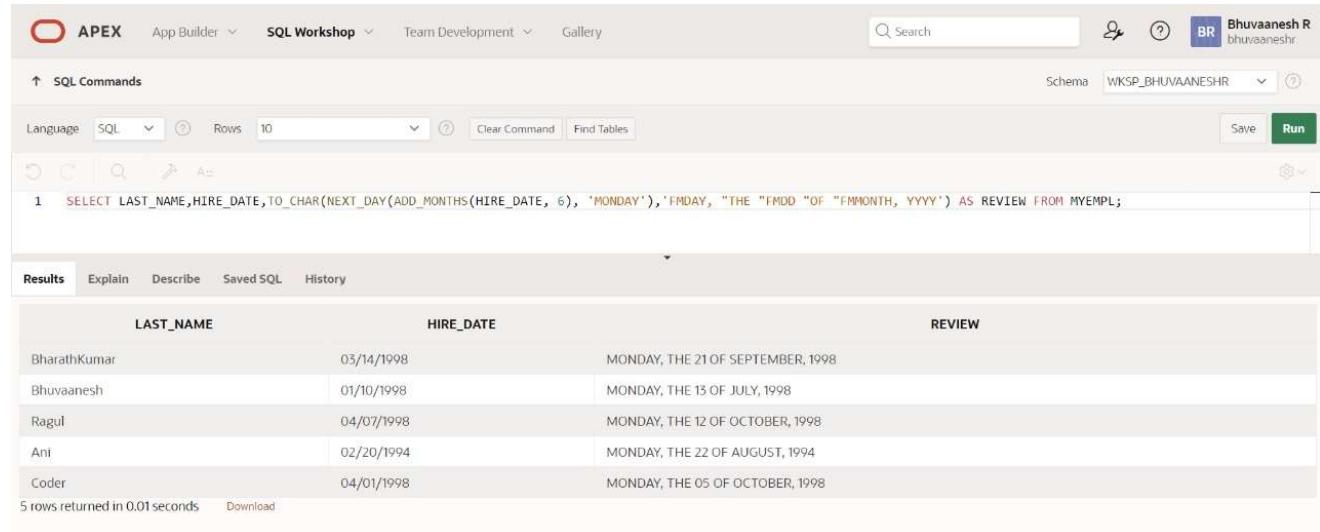
The results section displays the output:

LAST_NAME	SALARY
BharathKumar	\$\$\$\$\$\$\$\$\$\$\$2217
Bhuvaanesh	\$\$\$\$\$\$\$\$\$\$\$8525
Ragul	\$\$\$\$\$\$\$\$\$\$\$1705
Ani	\$\$\$\$\$\$\$\$\$\$\$2085
Coder	\$\$\$\$\$\$\$\$\$\$\$2015

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

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there is a search bar, user profile, and schema dropdown set to 'WKSP_BHUVAAINESHR'. The main area is titled 'SQL Commands' with a 'Run' button. The SQL command entered is:

```
1 SELECT LAST_NAME,HIRE_DATE,TO_CHAR(NEXT_DAY(ADD_MONTHS(HIRE_DATE, 6), 'MONDAY'),'FMDD "THE "FMDD "OF "FMMONTH, YYYY') AS REVIEW FROM MYEMPL;
```

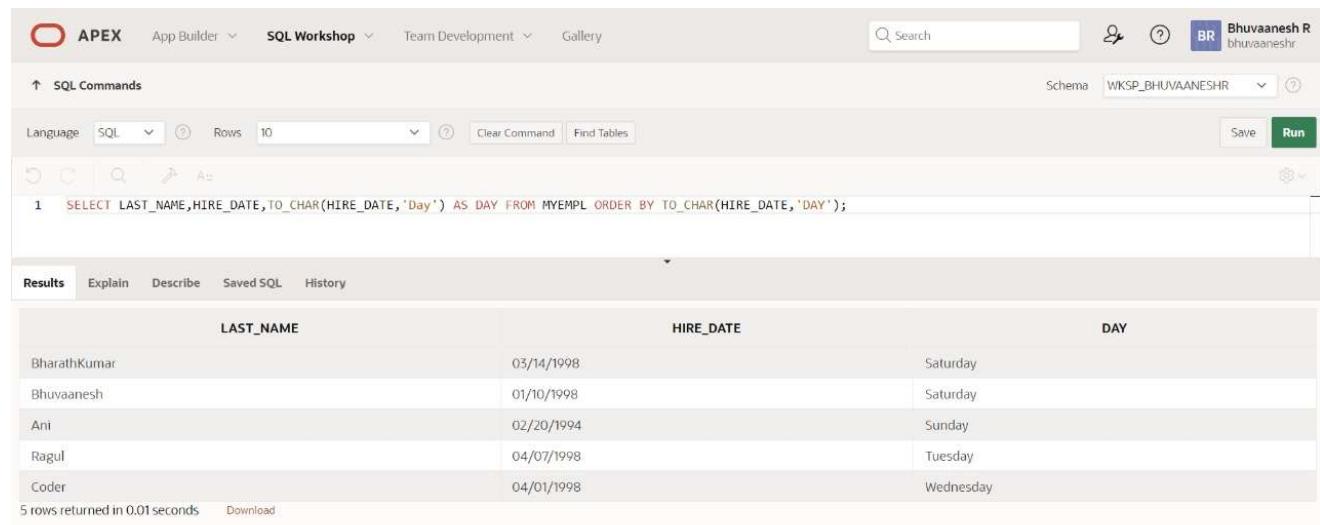
The results section displays the output of the query:

LAST_NAME	HIRE_DATE	REVIEW
BharathKumar	03/14/1998	MONDAY, THE 21 OF SEPTEMBER, 1998
Bhavaanesh	01/10/1998	MONDAY, THE 13 OF JULY, 1998
Ragul	04/07/1998	MONDAY, THE 12 OF OCTOBER, 1998
Ani	02/20/1994	MONDAY, THE 22 OF AUGUST, 1994
Coder	04/01/1998	MONDAY, THE 05 OF OCTOBER, 1998

5 rows returned in 0.01 seconds

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there is a search bar, user profile, and schema dropdown set to 'WKSP_BHUVAAINESHR'. The main area is titled 'SQL Commands' with a 'Run' button. The SQL command entered is:

```
1 SELECT LAST_NAME,HIRE_DATE,TO_CHAR(HIRE_DATE,'Day') AS DAY FROM MYEMPL ORDER BY TO_CHAR(HIRE_DATE,'Day');
```

The results section displays the output of the query:

LAST_NAME	HIRE_DATE	DAY
BharathKumar	03/14/1998	Saturday
Bhavaanesh	01/10/1998	Saturday
Ani	02/20/1994	Sunday
Ragul	04/07/1998	Tuesday
Coder	04/01/1998	Wednesday

5 rows returned in 0.01 seconds

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

RESULT:

DISPLAYING DATA FROM MULTIPLE TABLES

EX_NO:7

DATE:

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

QUERY:

```
Select e.last_name,e.department_id,d.department_id from employees e,departments d where e.department_id=d.department_id;
```

OUTPUT:

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

```
Select e.last_name,e.department_id,d.department_id from employees e,departments d where e.department_id=d.department_id;
```

The results table displays the following data:

LAST_NAME	DEPARTMENT_ID	DEPT_ID
BharathKumar	80	80
Bhavaanesh	3	3

2 rows returned in 0.02 seconds

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

QUERY:

```
select distinct job_id,loc_id from employees e,departments d where e.department_id=d.department_id and e.department_id=80;
```

OUTPUT:

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

```
select distinct e.job_id,d.location_id from employees e,departments d where e.department_id=d.department_id and e.department_id=80;
```

The results table displays the following data:

JOB_ID	LOCATION_ID
sales_rep	143

1 rows returned in 0.03 seconds

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

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile for 'Bhavaanesh R' (bhavaaneshr). The main area has tabs for 'SQL Commands' and 'Results'. The SQL Commands tab contains the following code:

```
1 Select e.last_name,e.department_id,d.dept_name,d.location_id,l.city from employees e,departments d,location l
2 Where e.department_id=d.department_id and d.location_id=l.location_id and e.commission_pct is not null;
```

The Results tab displays the query results in a table:

LAST_NAME	DEPARTMENT_ID	DEPT_NAME	LOCATION_ID	CITY
BharathKumar	80	Executive	143	MAYFAIR

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

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

QUERY:

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

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile for 'Bhavaanesh R' (bhavaaneshr). The main area has tabs for 'SQL Commands' and 'Results'. The SQL Commands tab contains the following code:

```
1 Select e.last_name,d.dept_name from employees e,departments d where e.department_id=d.department_id and last_name like '%a%';
2
```

The Results tab displays the query results in a table:

LAST_NAME	DEPT_NAME
BharathKumar	Executive
Bhavaanesh	ECE

Below the table, it says '2 rows returned in 0.01 seconds' and has a 'Download' link. The bottom of the page includes standard footer links and 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

QUERY:

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

OUTPUT:

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

```
1 Select e.last_name,e.department_id,e.job_id,d.dept_name
2 from employees e
3 join departments d on(e.department_id=d.department_id)
4 join location on (d.location_id=location.location_id) where lower(location.city)='toronto';
```

The results table has columns: LAST_NAME, DEPARTMENT_ID, JOB_ID, and DEPT_NAME. One row is returned:

LAST_NAME	DEPARTMENT_ID	JOB_ID	DEPT_NAME
BharathKumar	80	sales_rep	ECE

1 rows returned in 0.02 seconds. The URL is 220701045@rajalakshmi.edu.in/bhavaaneshr/en

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

QUERY:

```
Select w.last_name "Employee",w.employee_id "emp#",m.last_name "manager",m.employee_id "Mgr#" from employees m on (w.manager_id=m.employee_id);
```

OUTPUT:

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

```
1 Select w.last_name "Employee",w.employee_id "emp#",m.last_name "manager",m.employee_id "Mgr#"
2 From employees m join employees w on (w.manager_id=m.employee_id);
```

The results table has columns: Employee, emp#, manager, and Mgr#. One row is returned:

Employee	emp#	manager	Mgr#
BharathKumar	143	BharathKumar	143

1 rows returned in 0.04 seconds. The URL is 220701045@rajalakshmi.edu.in/bhavaaneshr/en

7. Modify lab4_6.sql to display all employees including King, who has no manager. Order the results by the employee number.

QUERY:

```
Select w.last_name "Employee",w.employee_id "emp#",m.last_name 'manager',m.employee_id "Mgr#"  
from employees w left outer join employees m on (w.manager_id=m.employee_id);
```

OUTPUT:

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

```
1 Select w.last_name "Employee",w.employee_id "emp#",m.last_name "manager",m.employee_id "Mgr#"  
2 from employees w left outer join employees m on (w.manager_id=m.employee_id);
```

The results section displays the following data:

Employee	emp#	manager	Mgr#
BharathKumar	143	BharathKumar	143
Bhuvaanesh	45	-	-
Ragul	284	-	-
Ani	555	-	-
Coder	42	-	-

5 rows returned in 0.01 seconds

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

QUERY:

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

OUTPUT:

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

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

The results section displays the following data:

DEPARTMENTS	COLLEAGUE
50	Ani
50	Coder

2 rows returned in 0.01 seconds

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

QUERY:

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

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 'Bhavaanesh R' (bhavaaneshr). The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the query is pasted. Under 'Results', the output is displayed in a table:

LAST_NAME	JOB_ID	DEPT_NAME	SALARY	JOB_GRADE
BharathKumar	sales_rep	Executive	6217	A
BharathKumar	sales_rep	ECE	6217	A

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

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

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

OUTPUT:

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 'Bhavaanesh R' (bhavaaneshr). The main area has tabs for 'SQL Commands' and 'Results'. Under 'SQL Commands', the query is pasted. Under 'Results', the output is displayed in a table:

LAST_NAME	HIRE_DATE
BharathKumar	03/14/1998
Bhavaanesh	01/10/1998
Ragul	04/07/1998
Coder	04/01/1998

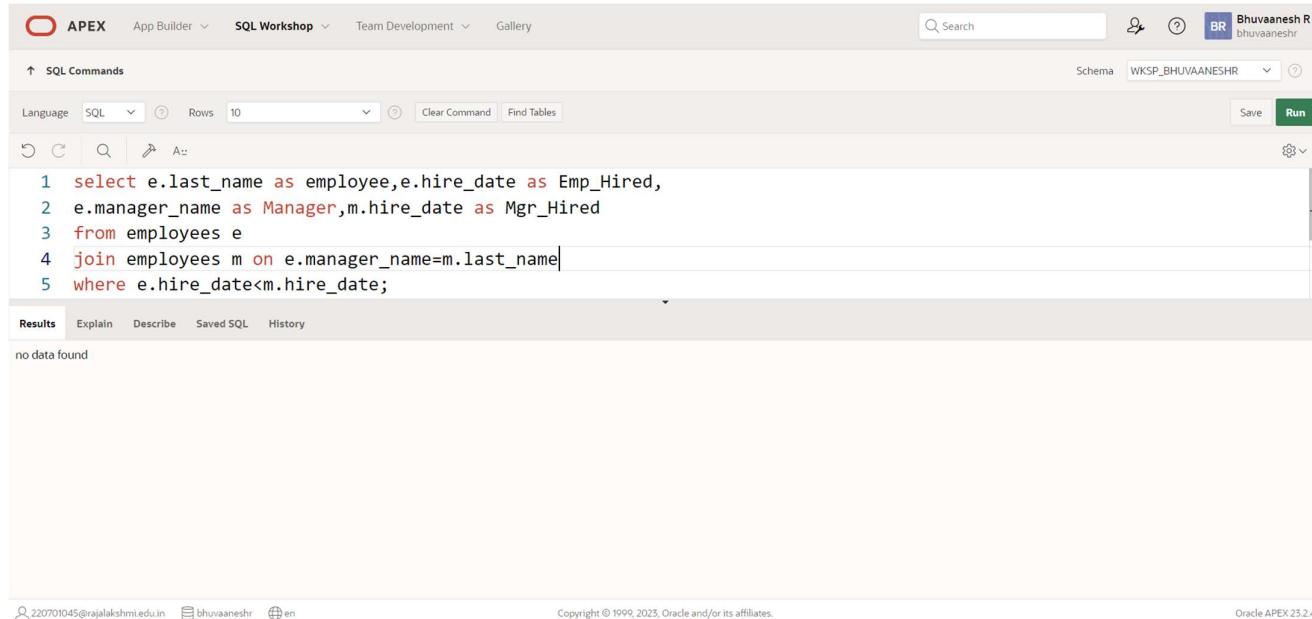
Below the table, it says '4 rows returned in 0.00 seconds' and there's a 'Download' link.

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

QUERY:

```
SELECT e.last_name AS Employee, e.hire_date AS Emp_Hired,  
e.manager_name AS Manager, m.hire_date AS Mgr_Hired  
FROM employees e  
JOIN employees m ON e.manager_name = m.last_name  
WHERE e.hire_date < m.hire_date;
```

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The query has been run successfully, and the results section displays the following output:

```
1 select e.last_name as employee,e.hire_date as Emp_Hired,  
2 e.manager_name as Manager,m.hire_date as Mgr_Hired  
3 from employees e  
4 join employees m on e.manager_name=m.last_name  
5 where e.hire_date<m.hire_date;
```

The results table shows "no data found".

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

RESULT:

AGGREGATING DATA USING GROUP FUNCTIONS

EX_NO : 8

DATE:

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

TRUE

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

FALSE

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

FALSE

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

OUTPUT:



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

```
1 SELECT ROUND(MAX(SALARY),0) "Maximum", ROUND(MIN(SALARY),0) "Minimum", ROUND(SUM(SALARY),0) "sum", ROUND(AVG(SALARY),0) "Average" FROM MYEMPL;
```

The results are:

Maximum	Minimum	sum	Average
8525	1705	16547	3309

1 rows returned in 0.01 seconds Download

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

OUTPUT:



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

```
1 SELECT JOB_ID, ROUND(MAX(SALARY),0) "MAXIMUM", ROUND(MIN(SALARY),0) "Minimum", ROUND(SUM(SALARY),0) "sum", ROUND(AVG(SALARY),0) "average" FROM MYEMPL GROUP BY JOB_ID;
```

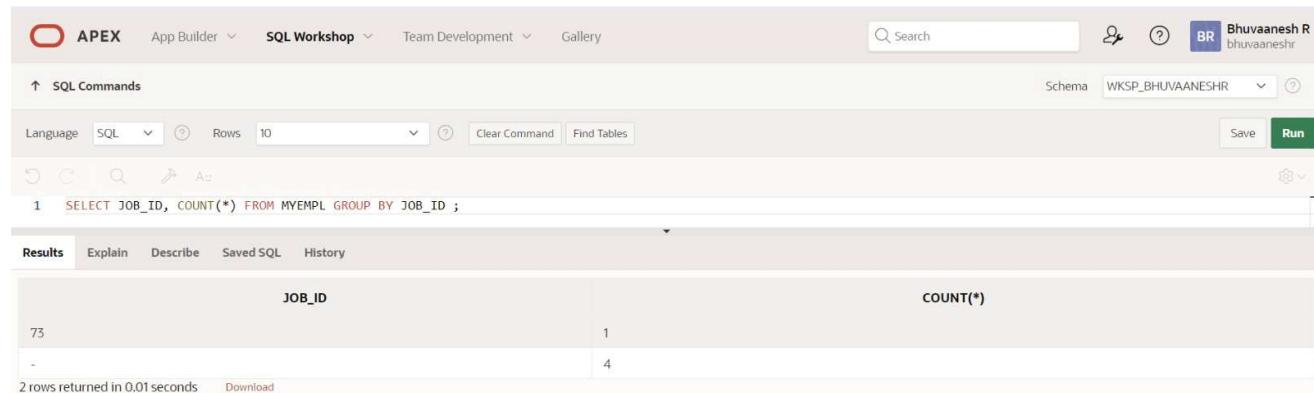
The results are:

JOB_ID	MAXIMUM	Minimum	sum	average
73	2217	2217	2217	2217
-	8525	1705	14330	3583

2 rows returned in 0.01 seconds Download

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile are also present. The main area is titled 'SQL Commands' with a 'Run' button. The SQL editor contains the following query:

```
1 SELECT JOB_ID, COUNT(*) FROM MYEMPL GROUP BY JOB_ID;
```

The results table has columns 'JOB_ID' and 'COUNT(*)'. The data shows two rows: one for JOB_ID 73 with a count of 1, and another row with a blank value for JOB_ID and a count of 4. The total count is 5.

JOB_ID	COUNT(*)
73	1
-	4

2 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.

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile are also present. The main area is titled 'SQL Commands' with a 'Run' button. The SQL editor contains the following query:

```
1 SELECT COUNT(DISTINCT MANAGER_ID ) "Number of managers" FROM MYEMPL;
```

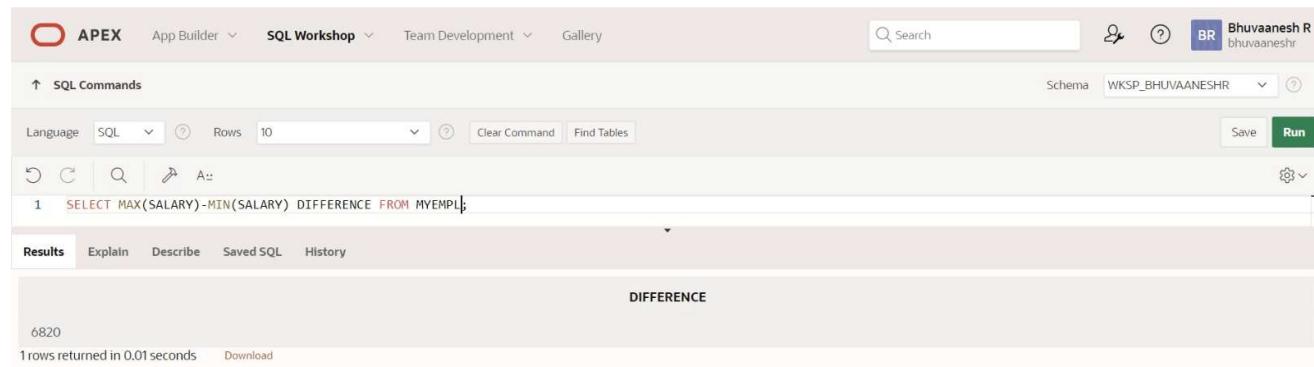
The results table has a single column 'Number of managers'. The data shows a value of 1.

Number of managers
1

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

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user profile are also present. The main area is titled 'SQL Commands' with a 'Run' button. The SQL editor contains the following query:

```
1 SELECT MAX(SALARY)-MIN(SALARY) DIFFERENCE FROM MYEMPL;
```

The results table has a single column 'DIFFERENCE'. The data shows a value of 6820.

DIFFERENCE
6820

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

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.

OUTPUT:



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

```
1 SELECT MANAGER_ID ,MIN(SALARY) FROM MYEMPL WHERE MANAGER_ID IS NOT NULL GROUP BY MANAGER_ID HAVING MIN(SALARY) >6000 ORDER BY MIN(SALARY) DESC;
```

The results table has two columns: MANAGER_ID and MIN(SALARY). The data is:

MANAGER_ID	MIN(SALARY)
143	6217

1 rows returned in 0.00 seconds

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

OUTPUT:



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

```
1 SELECT COUNT(*) AS TOTAL,SUM(DECODE(TO_CHAR(HIRE_DATE,'YYYY'),1995,1,0))"1995",SUM(DECODE(TO_CHAR(HIRE_DATE,'YYYY'),1996,1,0))"1996",SUM(DECODE(TO_CHAR(HIRE_DATE,'YYYY'),1997,1,0))"1997",SUM(DECODE(TO_CHAR(HIRE_DATE,'YYYY'),1998,1,0))"1998" FROM MYEMPL;
```

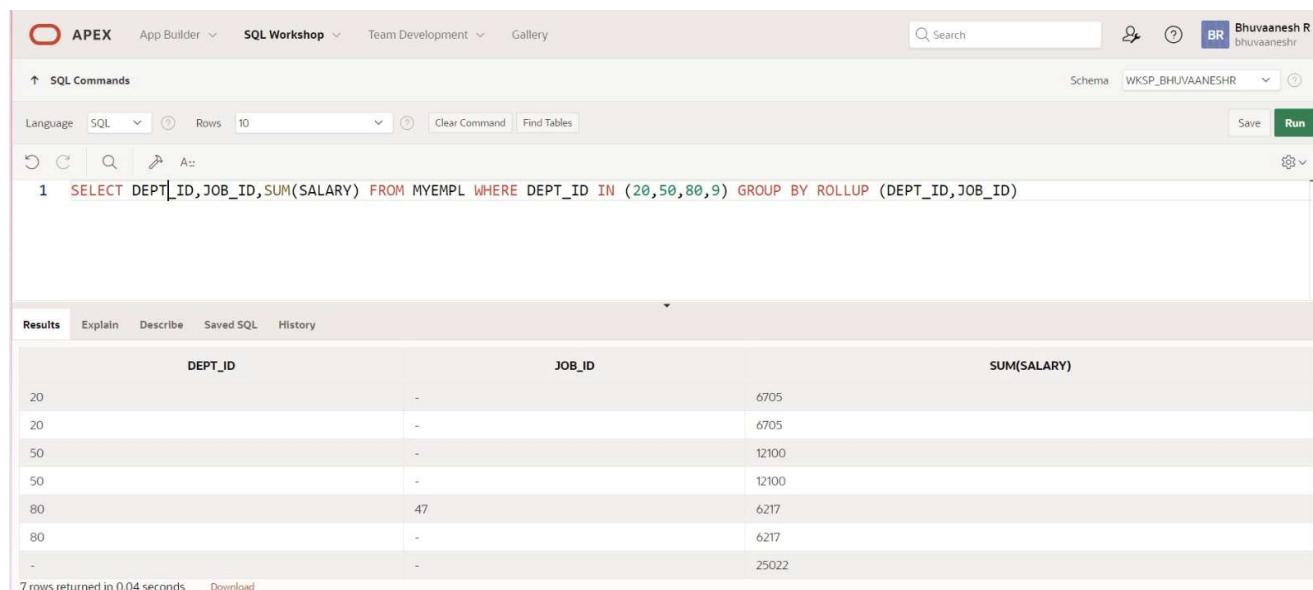
The results table has six columns: TOTAL, 1995, 1996, 1997, 1998. The data is:

TOTAL	1995	1996	1997	1998
5	0	0	0	4

1 rows returned in 0.02 seconds

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

OUTPUT:



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

```
1 SELECT DEPT_ID,JOB_ID,SUM(SALARY) FROM MYEMPL WHERE DEPT_ID IN (20,50,80,9) GROUP BY ROLLUP (DEPT_ID,JOB_ID)
```

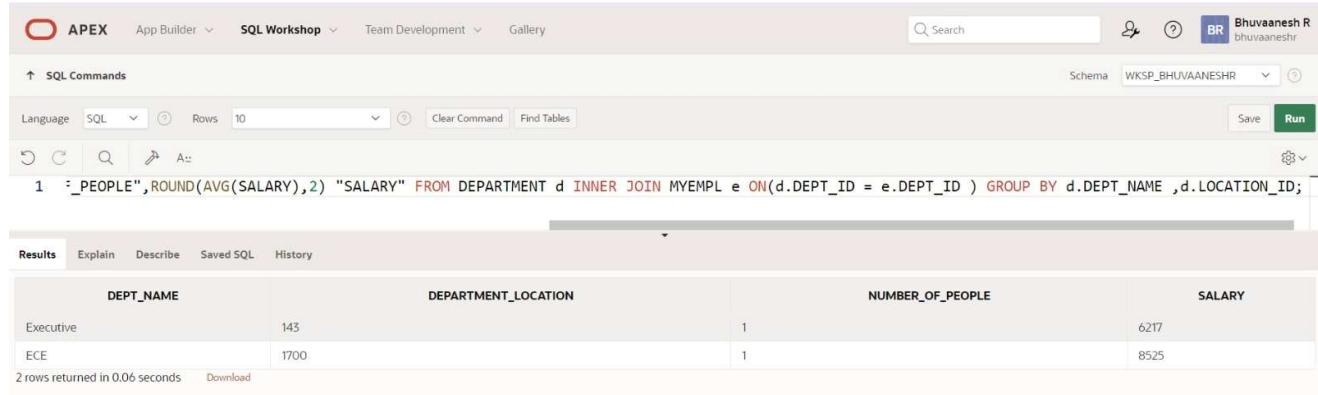
The results table has three columns: DEPT_ID, JOB_ID, and SUM(SALARY). The data is:

DEPT_ID	JOB_ID	SUM(SALARY)
20	-	6705
20	-	6705
50	-	12100
50	-	12100
80	47	6217
80	-	6217
-	-	25022

7 rows returned in 0.04 seconds

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.

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows the user profile "Bhuvaanesh R bhavaaneshr". The main area displays a SQL command window with the following query:

```
1  :=_PEOPLE",ROUND(AVG(SALARY),2) "SALARY" FROM DEPARTMENT d INNER JOIN MYEMPL e ON(d.DEPT_ID = e.DEPT_ID ) GROUP BY d.DEPT_NAME ,d.LOCATION_ID;
```

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

DEPT_NAME	DEPARTMENT_LOCATION	NUMBER_OF_PEOPLE	SALARY
Executive	143	1	6217
ECE	1700	1	8525

Below the table, it says "2 rows returned in 0.06 seconds" and there is a "Download" link.

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

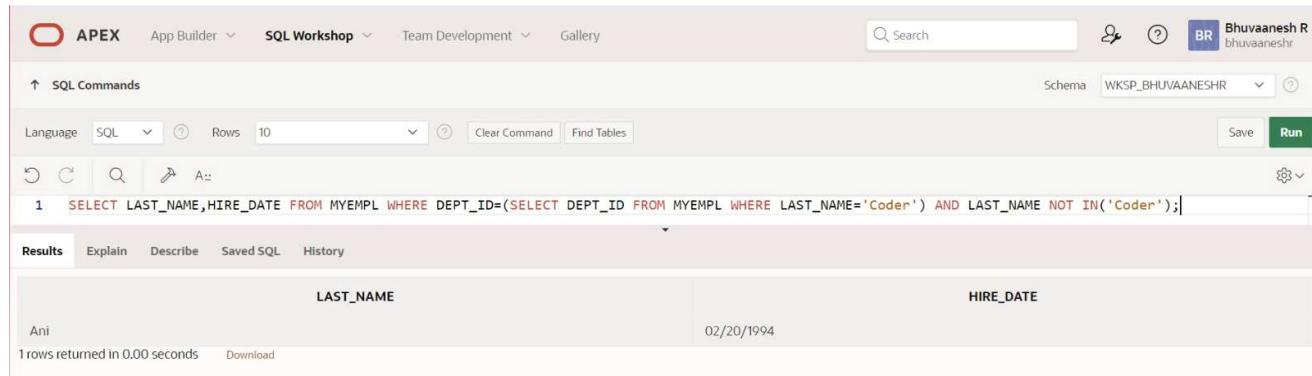
SUB QUERIES

EX_NO:9

DATE:

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

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The top navigation bar shows "APEX", "App Builder", "SQL Workshop", "Team Development", and "Gallery". The right side shows a user profile for "Bhuvaanesh R" (bhuvaareshr) with a "WKSP_BHUVAAINESHR" schema selected. The main area is titled "SQL Commands" with a "Language" dropdown set to "SQL". The SQL editor contains the following query:

```
1 SELECT LAST_NAME,HIRE_DATE FROM MYEMPL WHERE DEPT_ID=(SELECT DEPT_ID FROM MYEMPL WHERE LAST_NAME='Coder') AND LAST_NAME NOT IN('Coder');
```

The results table has columns "LAST_NAME" and "HIRE_DATE". One row is returned, showing "Ani" in the LAST_NAME column and "02/20/1994" in the HIRE_DATE column. The status bar at the bottom indicates "1 rows returned in 0.00 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.

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The top navigation bar shows "APEX", "App Builder", "SQL Workshop", "Team Development", and "Gallery". The right side shows a user profile for "Bhuvaanesh R" (bhuvaareshr) with a "WKSP_BHUVAAINESHR" schema selected. The main area is titled "SQL Commands" with a "Language" dropdown set to "SQL". The SQL editor contains the following query:

```
1 SELECT EMPLOYEE_NUMBER,LAST_NAME,SALARY FROM MYEMPL WHERE SALARY>(SELECT AVG(SALARY) FROM MYEMPL) ORDER BY SALARY;
```

The results table has columns "EMPLOYEE_NUMBER", "LAST_NAME", and "SALARY". One row is returned, showing "45" in the EMPLOYEE_NUMBER column, "Bhuvaanesh" in the LAST_NAME column, and "8525" in the SALARY column. The status bar at the bottom indicates "1 rows returned in 0.00 seconds".

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.

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The schema is set to 'WKSP_BHUVAAINESHR'. A SQL command is entered: 'SELECT * FROM MYEMPL_VU;'. The results show five rows of employee information:

EMPLOYEE_NUMBER	EMPLOYEE	DEPT_ID
143	BharathKumar	80
45	Bhuvaaresh	3
284	Ragul	20
520	Ani	50
42	Coder	

5 rows returned in 0.02 seconds. There is a 'Download' button at the bottom.

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.

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The schema is set to 'WKSP_BHUVAAINESHR'. A SQL command is entered: 'SELECT LAST_NAME,DEPT_ID,JOB_ID FROM MYEMPL WHERE DEPT_ID=(SELECT DEPT_ID FROM DEPARTMENT WHERE LOCATION_ID=1700);'. The results show one row of employee information:

LAST_NAME	DEPT_ID	JOB_ID
Bhuvaaresh	3	

1 rows returned in 0.00 seconds. There is a 'Download' button at the bottom.

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

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The schema is set to 'WKSP_BHUVAAINESHR'. A SQL command is entered: 'SELECT LAST_NAME,SALARY FROM MYEMPL WHERE MANAGER_ID=(SELECT MANAGER_ID FROM MYEMPL WHERE MANAGER_NAME='King'))';'. The results show one row of employee information:

LAST_NAME	SALARY
BharathKumar	2217

1 rows returned in 0.01 seconds. There is a 'Download' button at the bottom.

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

OUTPUT:



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

```
1 SELECT DEPT_ID, LAST_NAME, JOB_ID FROM MYEMPL WHERE DEPT_ID IN (SELECT DEPT_ID FROM DEPARTMENT WHERE DEPT_NAME='Executive');
```

The results table has columns DEPT_ID, LAST_NAME, and JOB_ID. One row is returned:

DEPT_ID	LAST_NAME	JOB_ID
80	BharathKumar	73

1 rows returned in 0.00 seconds

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

OUTPUT:



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

```
1 SELECT EMPLOYEE_NUMBER, LAST_NAME, SALARY FROM MYEMPL WHERE SALARY > (SELECT AVG(SALARY) FROM MYEMPL WHERE LAST_NAME LIKE '%u%');
```

The results table has columns EMPLOYEE_NUMBER, LAST_NAME, and SALARY. One row is returned:

EMPLOYEE_NUMBER	LAST_NAME	SALARY
45	Bhuvaanesh	8525

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:

- 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.

OUTPUT:

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

```
1 SELECT DEPT_ID FROM HYEMPL MINUS SELECT DEPT_ID FROM HYEMPL WHERE JOB_NAME='ST_CLERK';
```

The results table displays the DEPT_ID column with values 80, 113, 115, and 124.

DEPT_ID
80
113
115
124

4 rows returned in 0.04 seconds

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

OUTPUT:

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

```
1 SELECT COUNTRY_ID,STATE_PROVINCE FROM LOCATION MINUS SELECT COUNTRY_ID,STATE_PROVINCE FROM LOCATION,DEPARTMENT WHERE LOCATION.LOCATION_ID=DEPARTMENT.LOCATION_ID;
```

The results table displays the COUNTRY_ID column with value 12 and the STATE_PROVINCE column with value TN.

COUNTRY_ID	STATE_PROVINCE
12	TN

1 rows returned in 0.02 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.

OUTPUT:

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

```
1 SELECT JOB_NAME,DEPT_ID FROM MYEMPL WHERE DEPT_ID=10 UNION  
2 SELECT JOB_NAME,DEPT_ID FROM MYEMPL WHERE DEPT_ID=50 UNION  
3 SELECT JOB_NAME,DEPT_ID FROM MYEMPL WHERE DEPT_ID=20;
```

The results table displays the JOB_NAME column with values HR Manager, Sales Representative, and Stock Clerk, and the DEPT_ID column with values 50, 10, and 20 respectively.

JOB_NAME	DEPT_ID
HR Manager	50
Sales Representative	10
Stock Clerk	20

3 rows returned in 0.00 seconds

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

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, user profile, and schema dropdown set to 'WKSP_BHUVANESHR'. The main area shows a SQL command window with the following query:

```
1 SELECT JOB_ID,EMPLOYEE_NUMBER FROM MYEMPL INTERSECT SELECT e.JOB_ID,e.EMPLOYEE_NUMBER FROM MYEMPL e,JOB_HISTORY j WHERE e.JOB_ID=j.OLD_JOB_ID;
```

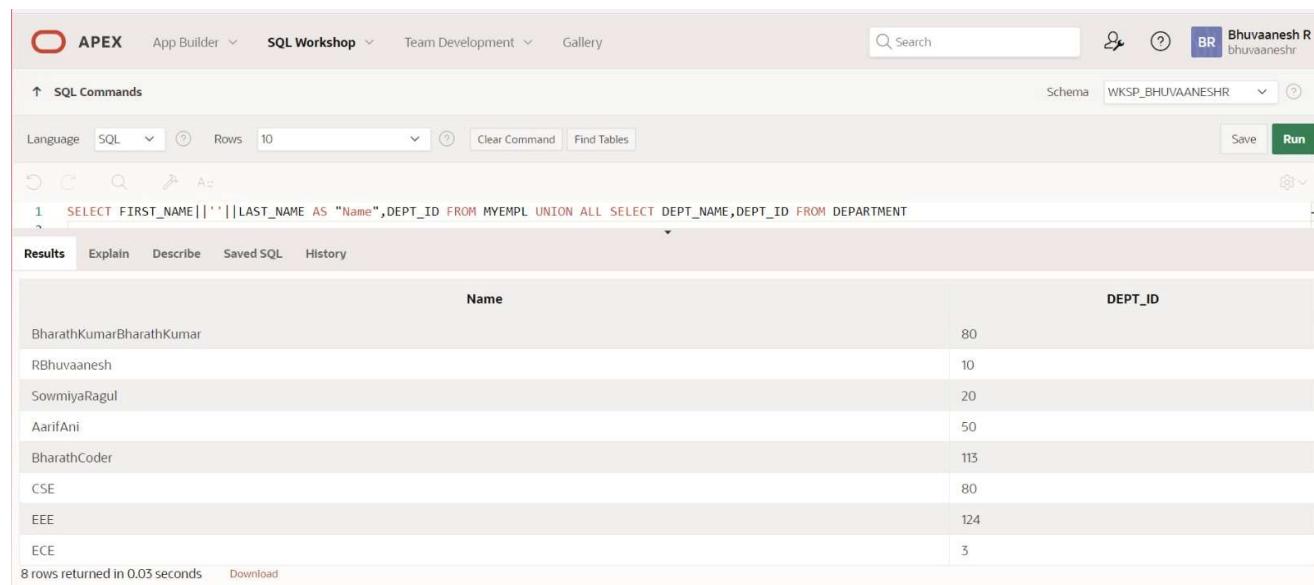
The results tab is selected, displaying the output:

JOB_ID	EMPLOYEE_NUMBER
73	143

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

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

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, user profile, and schema dropdown set to 'WKSP_BHUVANESHR'. The main area shows a SQL command window with the following query:

```
1 SELECT FIRST_NAME || ' ' || LAST_NAME AS "Name",DEPT_ID FROM MYEMPL UNION ALL SELECT DEPT_NAME,DEPT_ID FROM DEPARTMENT
```

The results tab is selected, displaying the output:

Name	DEPT_ID
BharathKumarBharathKumar	80
RBhuvaanesh	10
SowmiyaRagul	20
AarifAni	50
BharathCoder	113
CSE	80
EEE	124
ECE	3

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

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

RESULT:

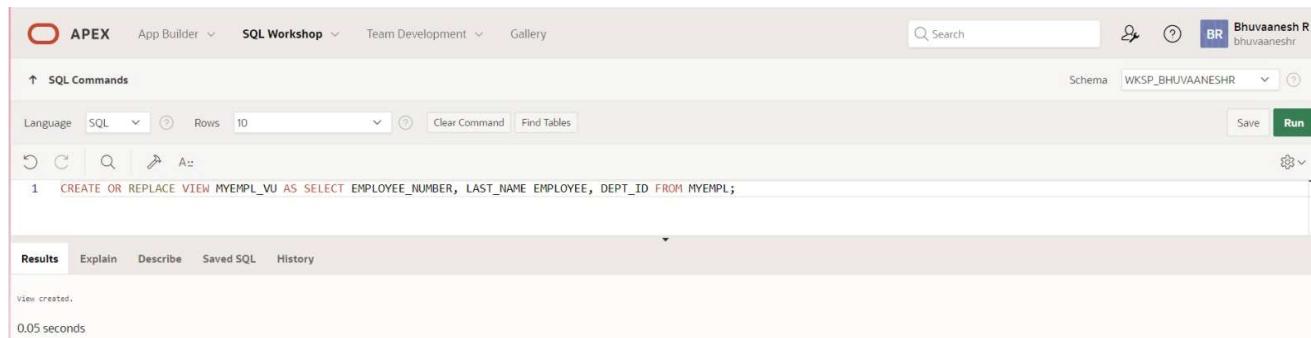
CREATING VIEWS

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.

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:

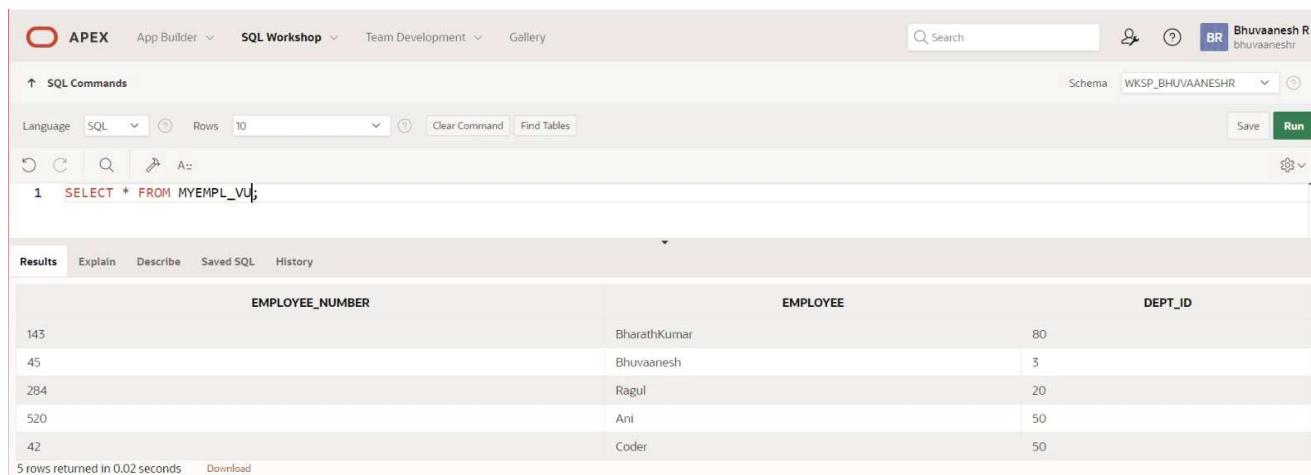
```
1 CREATE OR REPLACE VIEW MYEMPL_VU AS SELECT EMPLOYEE_NUMBER, LAST_NAME EMPLOYEE, DEPT_ID FROM MYEMPL;
```

Below the command, the results pane shows:

View created.
0.05 seconds

2.) Display the contents of the EMPLOYEES_VU view.

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 SELECT * FROM MYEMPL_VU;
```

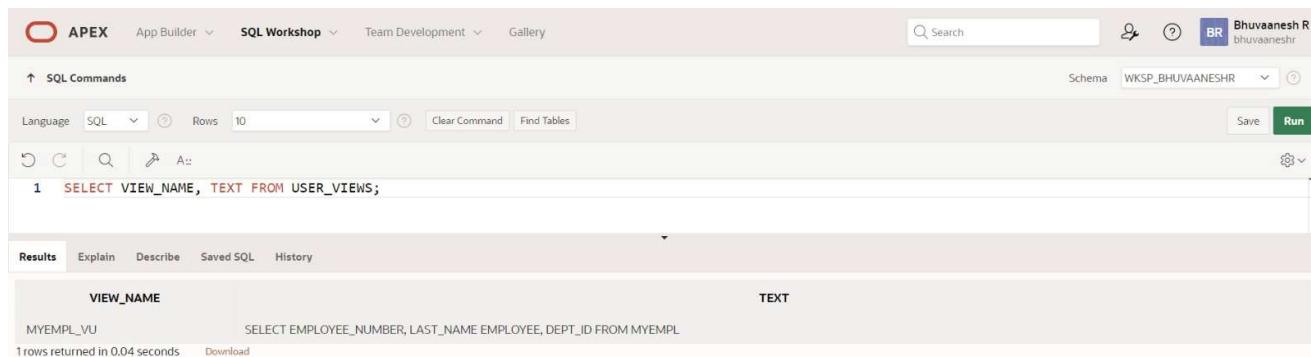
Below the command, the results pane shows the following data:

EMPLOYEE_NUMBER	EMPLOYEE	DEPT_ID
143	BharathKumar	80
45	Bhuvaanesh	3
284	Ragul	20
520	Ani	50
42	Coder	50

5 rows returned in 0.02 seconds Download

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

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 SELECT VIEW_NAME, TEXT FROM USER_VIEWS;
```

Below the command, the results pane shows the following data:

VIEW_NAME	TEXT
MYEMPL_VU	SELECT EMPLOYEE_NUMBER, LAST_NAME EMPLOYEE, DEPT_ID FROM MYEMPL

1 rows returned in 0.04 seconds Download

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

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, it shows the user 'Bhuvaanesh R' and the schema 'WKSP_BHUVAAINESHR'. The main area has a 'Language' dropdown set to 'SQL', a 'Rows' dropdown set to '10', and buttons for 'Clear Command' and 'Find Tables'. A green 'Run' button is at the bottom right. The SQL command entered is: `1 SELECT EMPLOYEE,DEPT_ID FROM MYEMPL_VU;`. Below the command, the results are displayed in a table with two columns: 'EMPLOYEE' and 'DEPT_ID'. The data rows are: BharathKumar (80), Bhuvanesh (3), Ragul (20), Ani (50), and Coder (50). A note at the bottom says '5 rows returned in 0.00 seconds'.

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.

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, it shows the user 'Bhuvaanesh R' and the schema 'WKSP_BHUVAAINESHR'. The main area has a 'Language' dropdown set to 'SQL', a 'Rows' dropdown set to '10', and buttons for 'Clear Command' and 'Find Tables'. A green 'Run' button is at the bottom right. The SQL command entered is: `1 CREATE VIEW dept50 AS SELECT EMPLOYEE_NUMBER, LAST_NAME EMPLOYEE, DEPT_ID FROM MYEMPL WHERE DEPT_ID = 50 WITH CHECK OPTION CONSTRAINT EMP_DEPT_50;`. Below the command, the results show a message 'View created.' and a time of '0.04 seconds'.

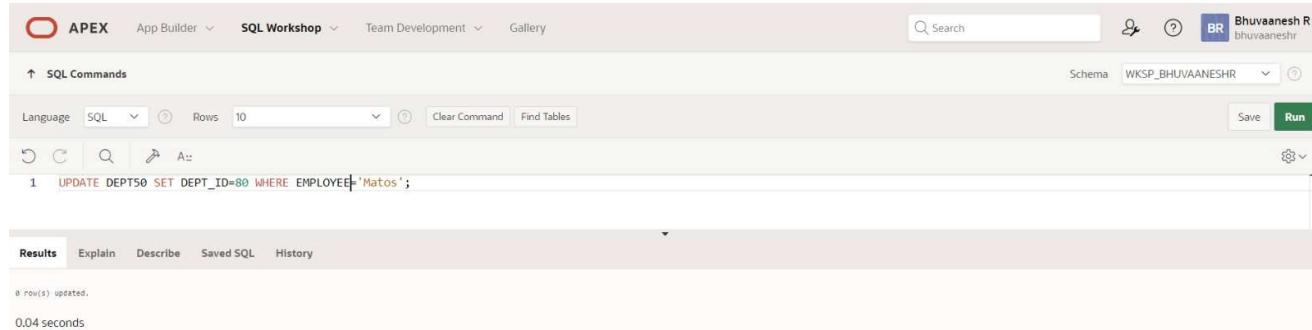
6.) Display the structure and contents of the DEPT50 view.

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, it shows the user 'Bhuvaanesh R' and the schema 'WKSP_BHUVAAINESHR'. The main area has a 'Language' dropdown set to 'SQL', a 'Rows' dropdown set to '10', and buttons for 'Clear Command' and 'Find Tables'. A green 'Run' button is at the bottom right. The SQL command entered is: `1 DESCRIBE DEPT50`. Below the command, the results are displayed in a table titled 'Describe DEPT50'. It shows the object type as 'VIEW' and lists three columns: 'EMPLOYEE_NUMBER', 'EMPLOYEE', and 'DEPT_ID'. The table includes headers for 'Table', 'Column', 'Data Type', 'Length', 'Precision', 'Scale', 'Primary Key', 'Nullable', 'Default', and 'Comment'. The 'Primary Key' column for all three columns has a checkmark.

7.) Attempt to reassign Matos to department 80

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, user information for 'Bhuvanesh R bhuvanesh', and a schema dropdown set to 'WKSP_BHUVANESHR'. The main area shows a SQL command line with the following code:

```
1 UPDATE DEPT50 SET DEPT_ID=80 WHERE EMPLOYEE='Matos';
```

The results tab is selected, showing the output:

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

Execution time: 0.04 seconds.

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

OUTPUT:



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

```
1 CREATE OR REPLACE VIEW SALARY_VU AS SELECT e.LAST_NAME "Employee", d.DEPARTMENT "Department", e.SALARY "Salary", j.JOB_GRADE "Grades" FROM MYEMPL e,DEPARTMENT d,JOB_GRADE j where e.DEPT_ID=d.DEPT_ID AND e.SALARY BETWEEN j.LONGEST_SALARY AND j.HIGHEST_SALARY;
```

The results tab is selected, showing the output:

```
View created.
```

Execution time: 0.03 seconds.

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

RESULT:

EXERCISE 12

PRACTICE QUESTIONS

Intro to Constraints; NOT NULL and UNIQUE Constraints

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Table Created.

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

DESCRIBE f_global_locations;

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

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

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

PRIMARY KEY, FOREIGN KEY, and CHECK Constraints

1. What is the purpose of a

- PRIMARY KEY
- FOREIGN KEY
- CHECK CONSTRAINT

a. **PRIMARY KEY**

Uniquely identify each row in table.

b. **FOREIGN KEY**

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

c. **CHECK CONSTRAINT**

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

2. Using the column information for the animals table below, name constraints where applicable at the table level, otherwise name them at the column level. Define the primary key (animal_id). The license_tag_number must be unique. The admit_date and vaccination_date columns cannot contain null values.

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

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

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

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

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

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

```
SELECT * FROM animals;
```

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

COLUMN LEVEL STATEMENT:

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

TABLE LEVEL STATEMENT:

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

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

a. ON DELETE CASCADE

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

b. ON DELETE SET NULL

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

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

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

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

PRACTICE PROBLEM

Managing Constraints

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

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

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

- ADD
- DROP
- ENABLE
- DISABLE

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

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

3. Create a FOREIGN KEY constraint in the copy_d_events table. Name the foreign key copy_d_events_fk. This key references the copy_d_clients table client_number column. What is the syntax you used to create the FOREIGN KEY constraint in the copy_d_events table?

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

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

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

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

COPY_D_CLT_CLIENT_NUMBER_PK

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

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

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

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

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

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

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

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

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

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

1 row(s) inserted.

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

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

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

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

1 row(s) deleted.

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

Table altered.

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

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

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

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

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

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

EXERCISE 13

Creating Views

1. What are three uses for a view from a DBA's perspective?
 - **Restrict access and display selective columns**
 - **Reduce complexity of queries from other internal systems. So, providing a way to view same data in a different manner.**
 - **Let the app code rely on views and allow the internal implementation of tables to be modified later.**

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

```
CREATE VIEW view_d_songs AS
```

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

3. SELECT * FROM view_d_songs. What was returned?

Results	Explain	Describe	Saved SQL	History
ID	Song Title	ARTIST		
47	Hurrah for Today	The Jubilant Trio		
49	Lets Celebrate	The Celebrants		
2 rows returned in 0.00 seconds			Download	

4. REPLACE view_d_songs. Add type_code to the column list. Use aliases for all columns. Or use alias after the CREATE statement as shown.

```
CREATE OR REPLACE VIEW view_d_songs AS
```

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

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

```
CREATE OR REPLACE VIEW view_d_events_pkgs AS
```

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

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

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

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

DML Operations and Views

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

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

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

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

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

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

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

```
SELECT * FROM view_copy_d_songs;
```

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

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

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

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

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

```
SELECT * FROM read_copy_d_cds;
```

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

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

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

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

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

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

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

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

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

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

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

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

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

DELETE,INSERT,MODIFY restricted if it contains:

Group functions
GROUP BY CLAUSE
DISTINCT
pseudocolumn ROWNUM Keyword

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

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

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

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

Managing Views

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

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

```
SELECT * FROM view_copy_d_songs;
```

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

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

ORA-00942: table or view does not exist

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

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

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

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

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

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

Indexes and Synonyms

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

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

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

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

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

3. When will an index be created automatically?

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

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

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

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

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

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

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

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

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

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

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

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

CREATE SYNONYM dj_tracks2 FOR d_track_listings;

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

10. Drop the synonym that you created in question

DROP SYNONYM dj_tracks2;

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

RESULT:

OTHER DATABASE OBJECTS

EX_NO:14

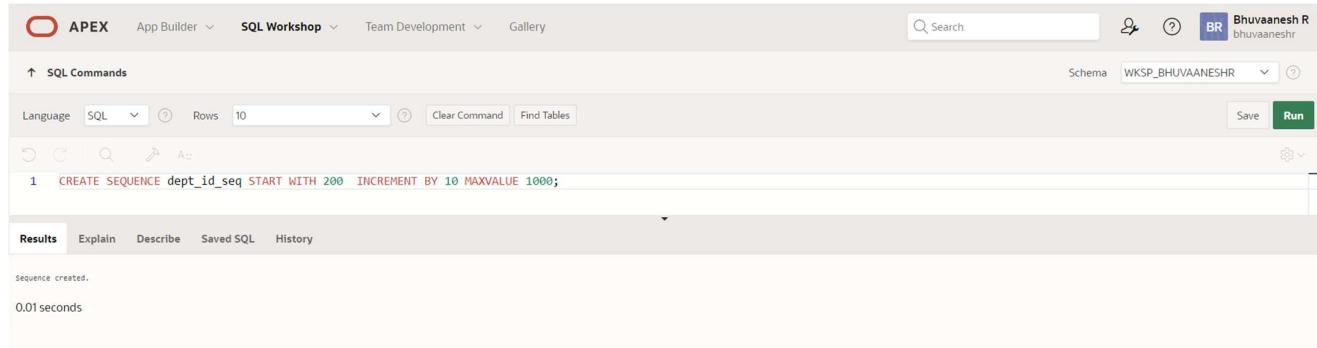
DATE:

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

QUERY:

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

OUTPUT:



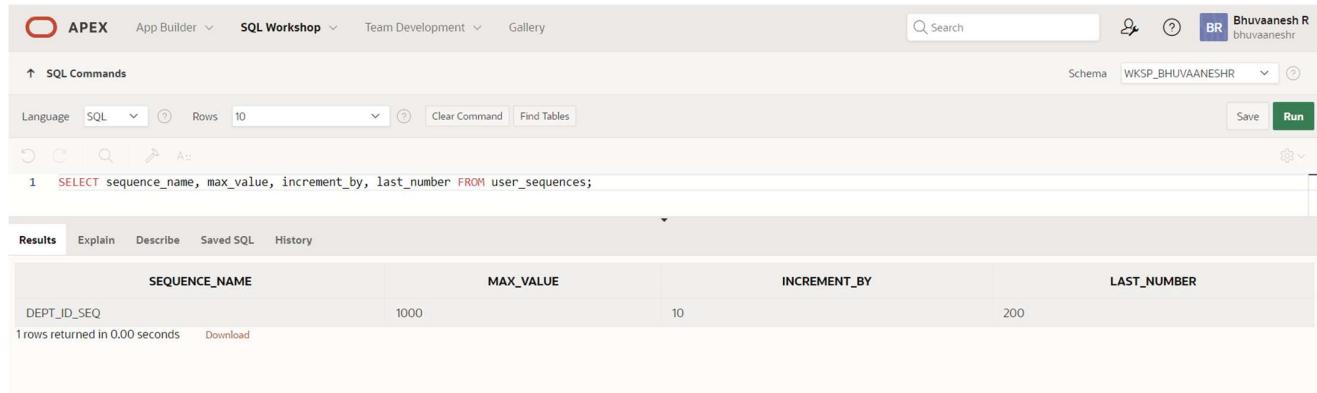
The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area displays the SQL command: 'CREATE SEQUENCE dept_id_seq START WITH 200 INCREMENT BY 10 MAXVALUE 1000;'. Below the command, the results show 'sequence created.' and a execution time of '0.01 seconds'.

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area displays the SQL command: 'SELECT sequence_name, max_value, increment_by, last_number FROM user_sequences;'. Below the command, the results table shows the following data:

SEQUENCE_NAME	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
DEPT_ID_SEQ	1000	10	200

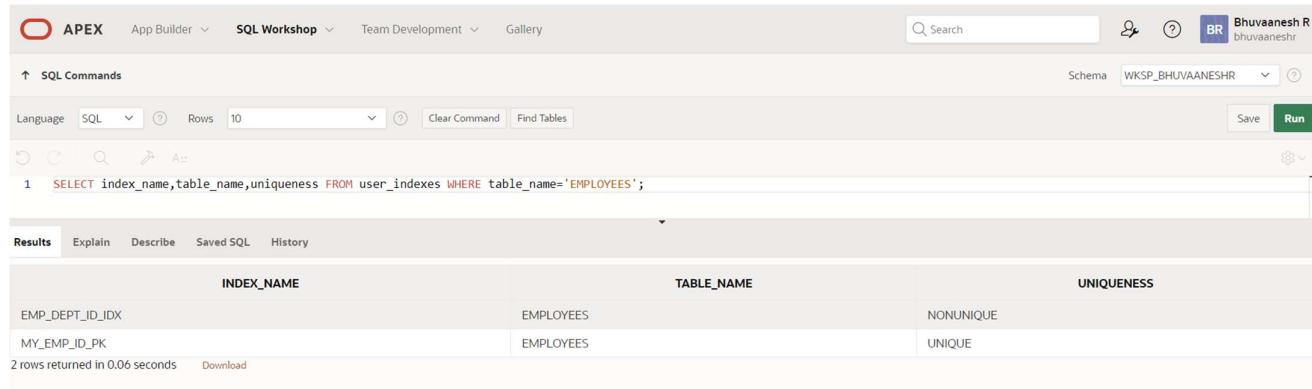
1 rows returned in 0.00 seconds

3.) Write a script to insert two rows into the DEPT table. Name your script lab12_3.sql. Be sure to use the sequence that you created for the ID column. Add two departments named Education and Administration. Confirm your additions. Run the commands in your script.

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The query window contains the command: `SELECT index_name,table_name,uniqueness FROM user_indexes WHERE table_name='EMPLOYEES';`. The results pane displays the following table:

INDEX_NAME	TABLE_NAME	UNIQUENESS
EMP_DEPT_ID_IDX	EMPLOYEES	NONUNIQUE
MY_EMP_ID_PK	EMPLOYEES	UNIQUE

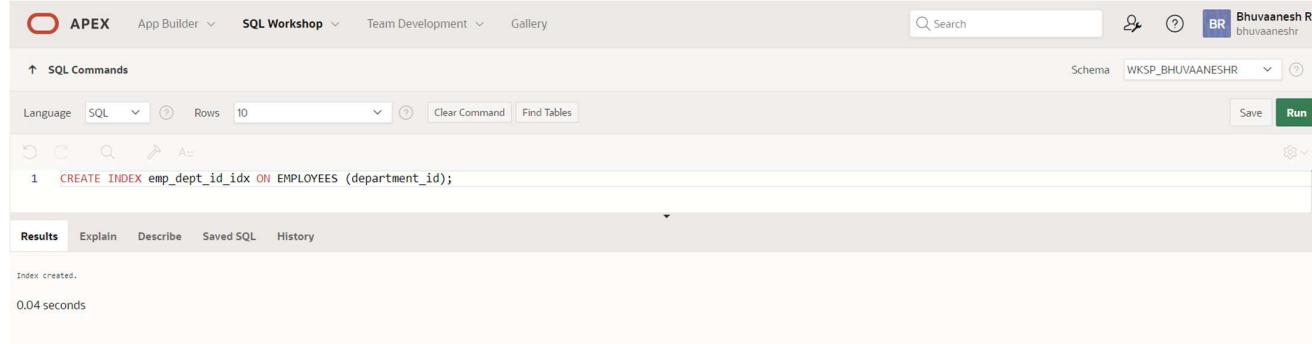
2 rows returned in 0.06 seconds Download

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The query window contains the command: `CREATE INDEX emp_dept_id_idx ON EMPLOYEES (department_id);`. The results pane shows the message: "Index created." and "0.04 seconds".

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The query window contains the command: `SELECT index_name,table_name,uniqueness FROM user_indexes WHERE table_name='EMPLOYEES';`. The results pane displays the following table:

INDEX_NAME	TABLE_NAME	UNIQUENESS
EMP_DEPT_ID_IDX	EMPLOYEES	NONUNIQUE
MY_EMP_ID_PK	EMPLOYEES	UNIQUE

2 rows returned in 0.06 seconds Download

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

RESULT:

CONTROLLING USER ACCESS

EX_NO:15

DATE:

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

The CREATE SESSION system privilege

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

The CREATE TABLE privilege

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

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

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

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

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

The ALTER USER statement

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

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

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

7. Query all the rows in your DEPARTMENTS table.

SELECT * FROM departments;

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

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

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

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

SELECT table_name FROM user_tables;

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

Team 1 revokes the privilege.

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

Team 2 revokes the privilege.

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

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

Team 1 executes this INSERT statement.

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

Team 2 executes this INSERT statement.

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

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

RESULT:

PL/SQL

CONTROL STRUCTURES

EX_NO:

DATE:

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user profile (Bhuvaanesh R), and schema dropdown (WKSP_BHUVAANESHR). The main workspace is titled 'SQL Commands' and contains the following PL/SQL code:

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

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

```
Incentive = 730.2
Statement processed.
0.01 seconds
```

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

QUERY:

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

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

OUTPUT:

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

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

In the results pane, an error message is displayed:

```
Error at line 3/12: ORA-06550: line 3, column 12:
PLS-00103: Encountered the symbol ".," when expecting one of the following:
constant exception can identifier
<a double-quoted delimited-identifier> table columns long
double ref char time timestamp interval date binary national
character nchar
The symbol ".," was substituted for "," to continue.
ORA-06550: line 3, column 12
ORA-06550: line 4, column 12
ORA-06550: line 4, column 12
PLS-00103: Encountered the symbol "END" when expecting one of the following:
begin function pragma procedure subtype type can identifier
<a double-quoted delimited-identifier> current cursor delete
exists prior
1. DECLARE
2. WELCOME varchar2(10) := 'welcome';
3. DBMS_Output.Put_Line("Welcome");
4. END;
5. /
```

The screenshot shows the Oracle SQL Workshop interface. In the command window, the same PL/SQL block is entered:

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

In the results pane, an error message is displayed:

```
Error at line 3/12: ORA-06550: line 3, column 12:
PLS-00103: Encountered the symbol ".," when expecting one of the following:
constant exception can identifier
<a double-quoted delimited-identifier> table columns long
double ref char time timestamp interval date binary national
character nchar
The symbol ".," was substituted for "," to continue.
ORA-06550: line 3, column 12
ORA-06550: line 4, column 12
ORA-06550: line 4, column 12
PLS-00103: Encountered the symbol "END" when expecting one of the following:
begin function pragma procedure subtype type can identifier
<a double-quoted delimited-identifier> current cursor delete
exists prior
1. DECLARE
2. WELCOME varchar2(10) := 'welcome';
3. DBMS_Output.Put_Line("Welcome");
4. END;
5. /
```

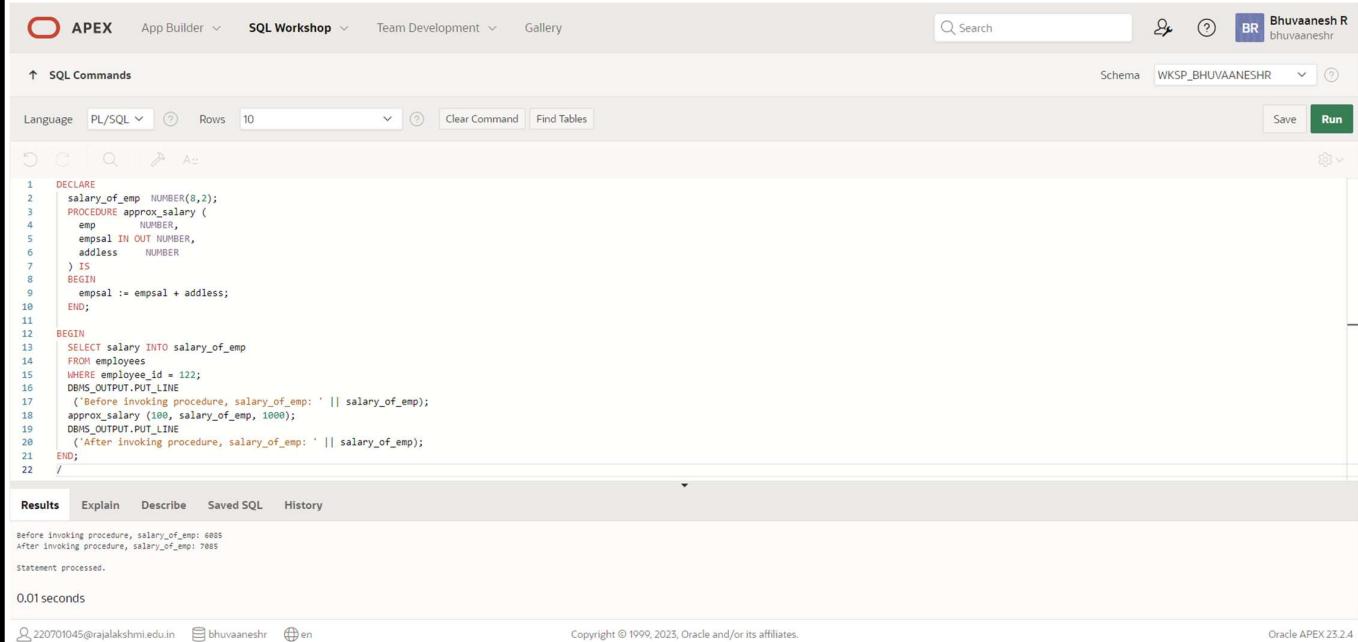
3.) Write a PL/SQL block to adjust the salary of the employee whose ID 122.

QUERY:

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

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

OUTPUT:



The screenshot shows the Oracle 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 'Bhuvaanesh R' (bhuvaareshr). The main workspace displays the PL/SQL code from the previous step. Below the code, the 'Results' tab is selected, showing the output of the executed procedure. The output indicates that the salary was increased by 1000, from 6885 to 7885. The bottom status bar shows the statement was processed in 0.01 seconds and includes copyright information for Oracle APEX 23.2.4.

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

Results Explain Describe Saved SQL History

Before invoking procedure, salary_of_emp: 6885
After invoking procedure, salary_of_emp: 7885
Statement processed.
0.01 seconds

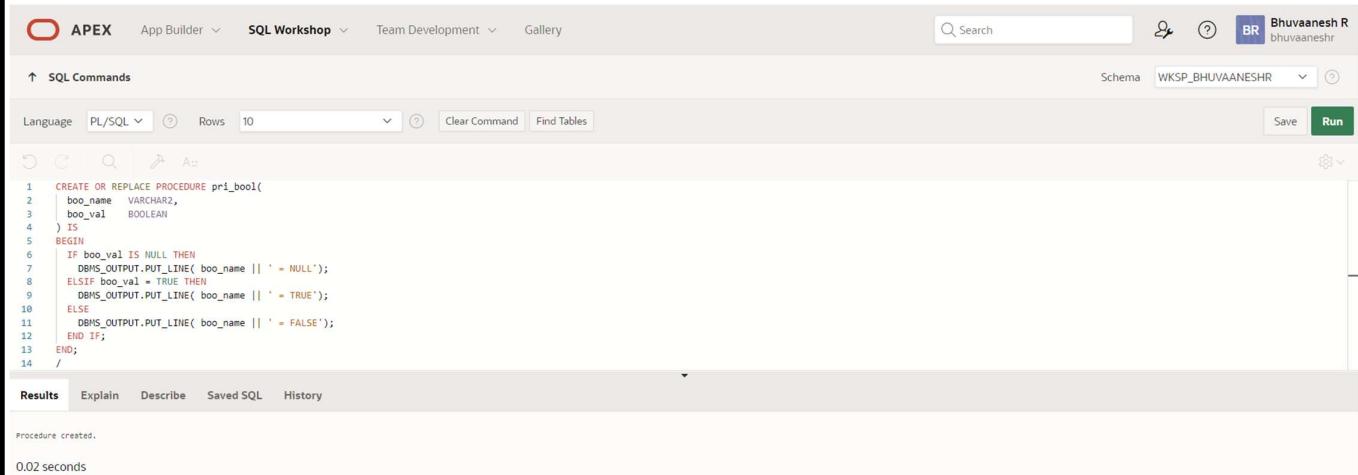
Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected, followed by 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. On the right, there's a search bar, a user profile for 'Bhavaanesh R bhavaaneshr', and a 'Run' button. Below the navigation, the schema 'WKSP_BHUVAAINESHR' is selected. The main area is titled 'SQL Commands' and contains the PL/SQL code for the 'pri_bool' procedure. The code is highlighted with syntax coloring. At the bottom of the code editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, showing the message 'Procedure created.' and a execution time of '0.02 seconds'.

```
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 /
```

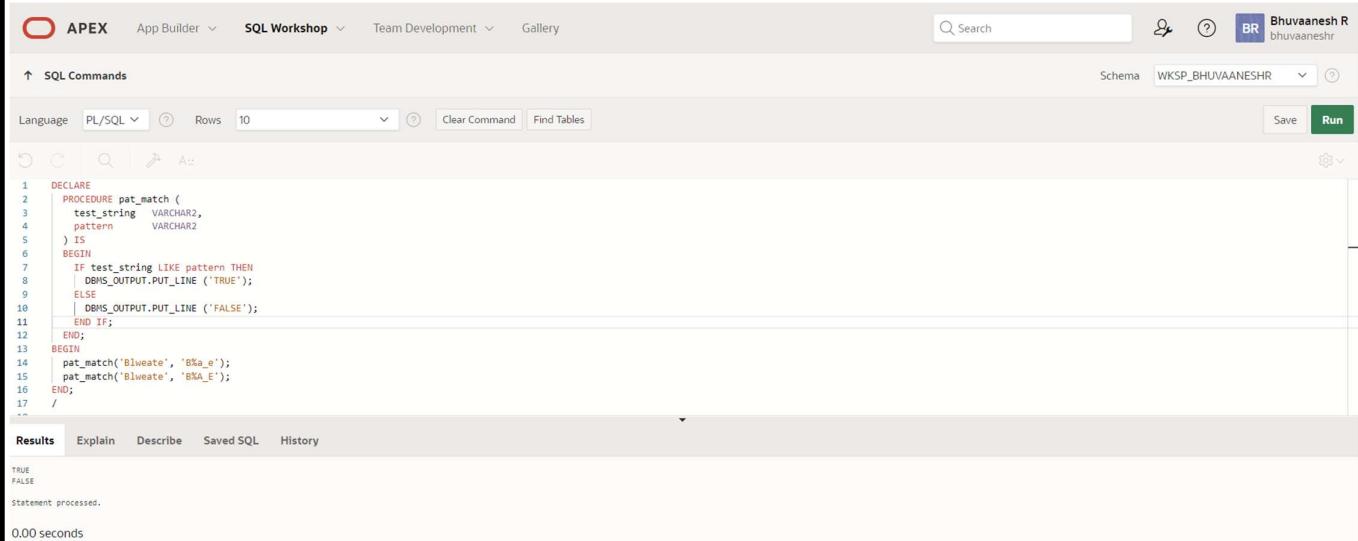
Procedure created.
0.02 seconds

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, user information (Bhuvanesh R), and a Run button. The main area is titled 'SQL Commands' and shows the following code:

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

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

```
TRUE
FALSE
```

Additional text at the bottom indicates the statement was processed in 0.00 seconds.

6.) Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num_small variable and large number will store in num_large variable

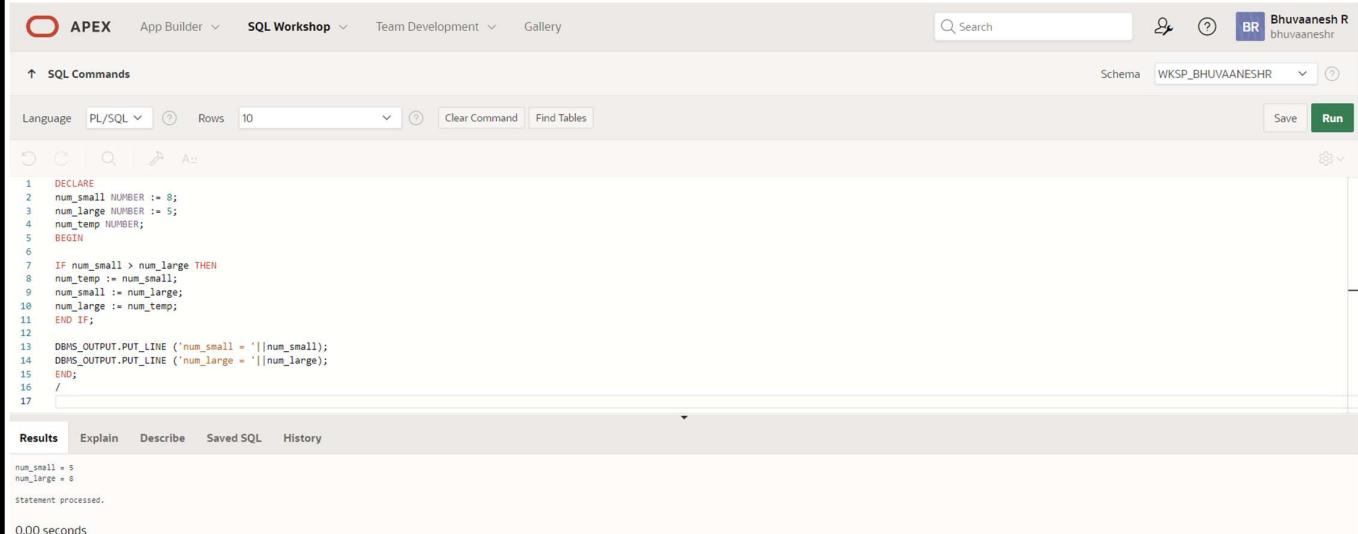
QUERY:

```
DECLARE
num_small NUMBER := 8;
num_large NUMBER := 5;
num_temp NUMBER;
BEGIN

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

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user profile, and session information (Bhavaanesh R, bhavaaneshr). The main workspace displays a PL/SQL block with numbered lines. Lines 1 through 16 show the declaration of variables and the logic for swapping them if the small variable is greater than the large one. Lines 17 and 18 show the output statements using DBMS_OUTPUT.PUT_LINE. The bottom section shows the results tab, which displays the output: num_small = 5 and num_large = 8, followed by a statement processed message and a timestamp of 0.00 seconds.

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

```

Results Explain Describe Saved SQL History

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

0.00 seconds
```

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

QUERY:

```
DECLARE
PROCEDURE test1 (
    sal_achieve NUMBER,
    target_qty NUMBER,
    emp_id NUMBER
)
IS
    incentive NUMBER := 0;
    updated VARCHAR2(3) := 'No';
BEGIN
    IF sal_achieve > (target_qty + 200) THEN
        incentive := (sal_achieve - target_qty)/4;
        UPDATE employees
        SET salary = salary + incentive
        WHERE employee_id = emp_id;
        updated := 'Yes';
    END IF;
    DBMS_OUTPUT.PUT_LINE (
        'Table updated? ' || updated || ',' ||
        'incentive = ' || incentive || '.'
    );
END test1;
BEGIN
    test1(2300, 2000, 144);
    test1(3600, 3000, 145);
END;
/
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is logged in as Bhuvanesh R (bhuvaneshr). The schema selected is WKSP_BHUVANESHR.

The main area is titled "SQL Commands" and contains a code editor with the following PL/SQL script:

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

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

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

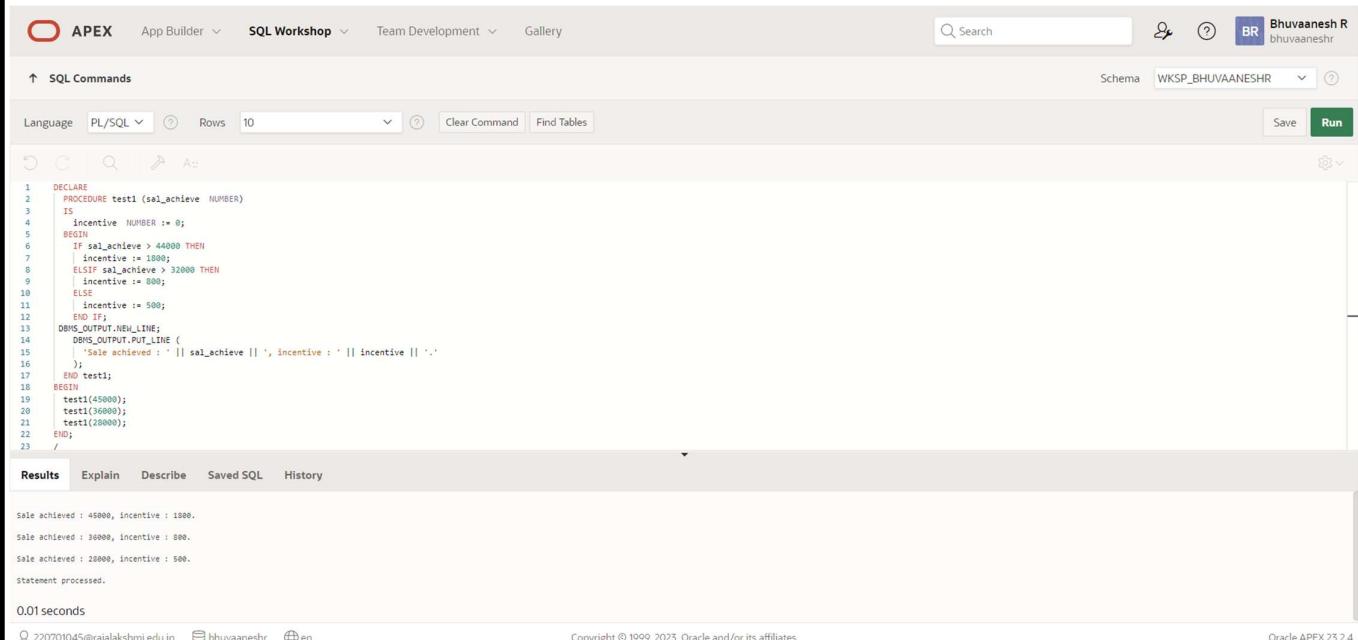
Execution time is listed as 0.01 seconds. The bottom of the page includes copyright information and links for Oracle APEX 23.2.4, user 220701045@rajalakshmi.edu.in, and profile bhuvaneshr.

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 APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows the user 'Bhuvaanesh R' and schema 'WKSP_BHUVANESHR'. The main workspace is titled 'SQL Commands' and contains the PL/SQL code from the previous block. Below the code, the 'Results' tab is selected, displaying the output of the executed procedure calls. The output shows three rows of results:

Sale achieved	Incentive
45000	1800
36000	800
28000	500

At the bottom, it says '0.01 seconds' and includes copyright information for Oracle APEX 23.2.

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

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

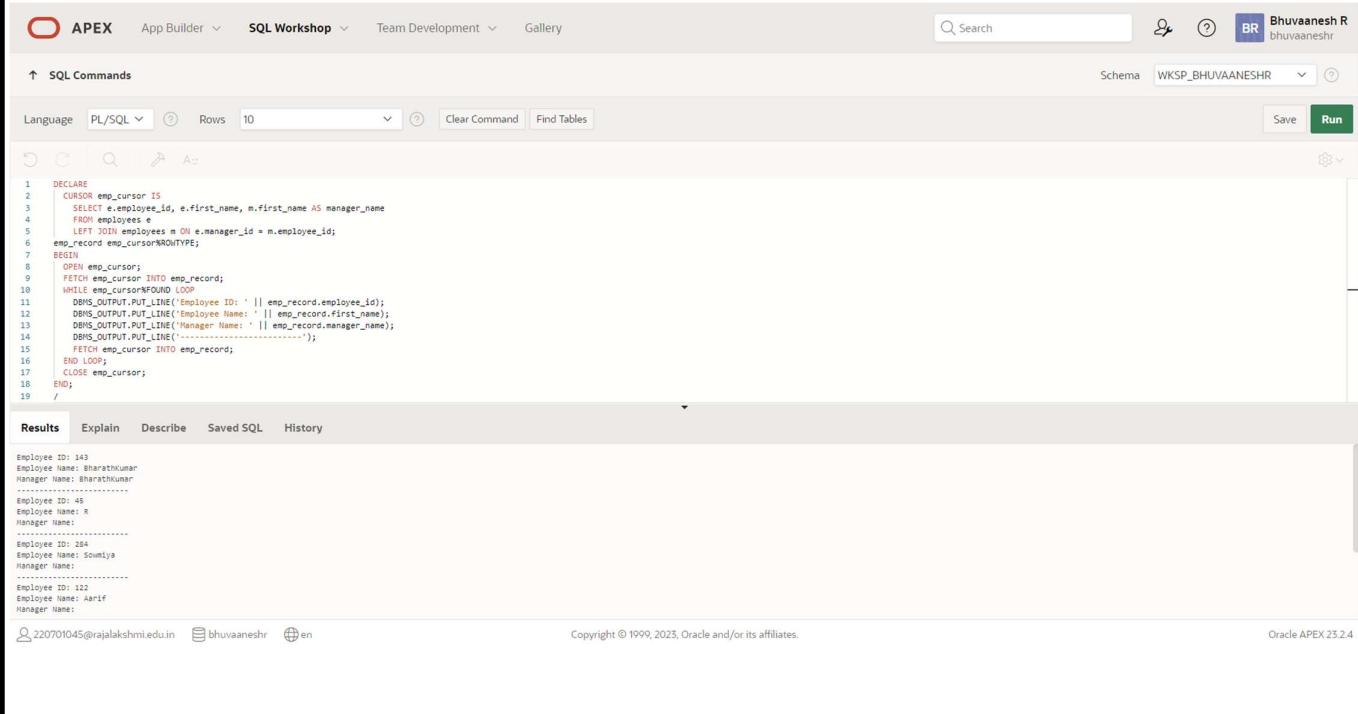
9.) Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

QUERY:

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

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

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is signed in as 'Bhavaanesh R' (bhavaaneshr). The main workspace is titled 'SQL Commands'. The code area contains the provided PL/SQL block. The results pane at the bottom displays the output of the query, listing employee details and their manager names.

```
1  DECLARE
2      CURSOR emp_cursor IS
3          SELECT e.employee_id, e.first_name, m.first_name AS manager_name
4          FROM employees e
5          LEFT JOIN employees m ON e.manager_id = m.employee_id;
6      emp_record.emp_cursor%ROWTYPE;
7  BEGIN
8      OPEN emp_cursor;
9      FETCH emp_cursor INTO emp_record;
10     WHILE emp_cursor%FOUND LOOP
11         DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp_record.employee_id);
12         DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_record.first_name);
13         DBMS_OUTPUT.PUT_LINE('Manager Name: ' || emp_record.manager_name);
14         DBMS_OUTPUT.PUT_LINE('-----');
15         FETCH emp_cursor INTO emp_record;
16     END LOOP;
17     CLOSE emp_cursor;
18  END;
19 /
```

Results

Employee ID	Employee Name	Manager Name
143	Bharathkumar	Bharathkumar
45	R	
28	Somanya	
32	Aarif	

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

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 APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user profile (BR bhuvanesh), and a schema dropdown set to WKSP_BHUVANESHR. The main workspace is titled "SQL Commands" and contains the following PL/SQL code:

```
1  DECLARE
2      tot_emp NUMBER;
3      get_dep_id NUMBER;
4
5  BEGIN
6      get_dep_id := 80;
7      SELECT Count(*)
8          INTO tot_emp
9      FROM employees e
10     JOIN departments d
11        ON e.department_id = d.department_id
12     WHERE e.department_id = get_dep_id;
13
14     dbms_output.Put_line ('The employees are in the department'||get_dep_id||' is: '
15                           ||to_char(tot_emp));
16
17    IF tot_emp >= 45 THEN
18        dbms_output.Put_line ('There are no vacancies in the department'||get_dep_id);
19    ELSE
20        dbms_output.Put_line ('There are'||to_char(45-tot_emp)||' vacancies in department'|| get_dep_id );
21    END IF;
22
23  END;
24
```

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

```
The employees are in the department 80 is:
There are 43 vacancies in department 80
Statement processed.
```

At the bottom, it shows the user information (220701045@irajalakshmi.edu.in, bhuvanesh, en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the version (Oracle APEX 23.2.4).

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

QUERY:

DECLARE

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

CURSOR c_employees IS

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

BEGIN

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

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

OPEN c_employees;

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

WHILE c_employees%FOUND LOOP

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

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

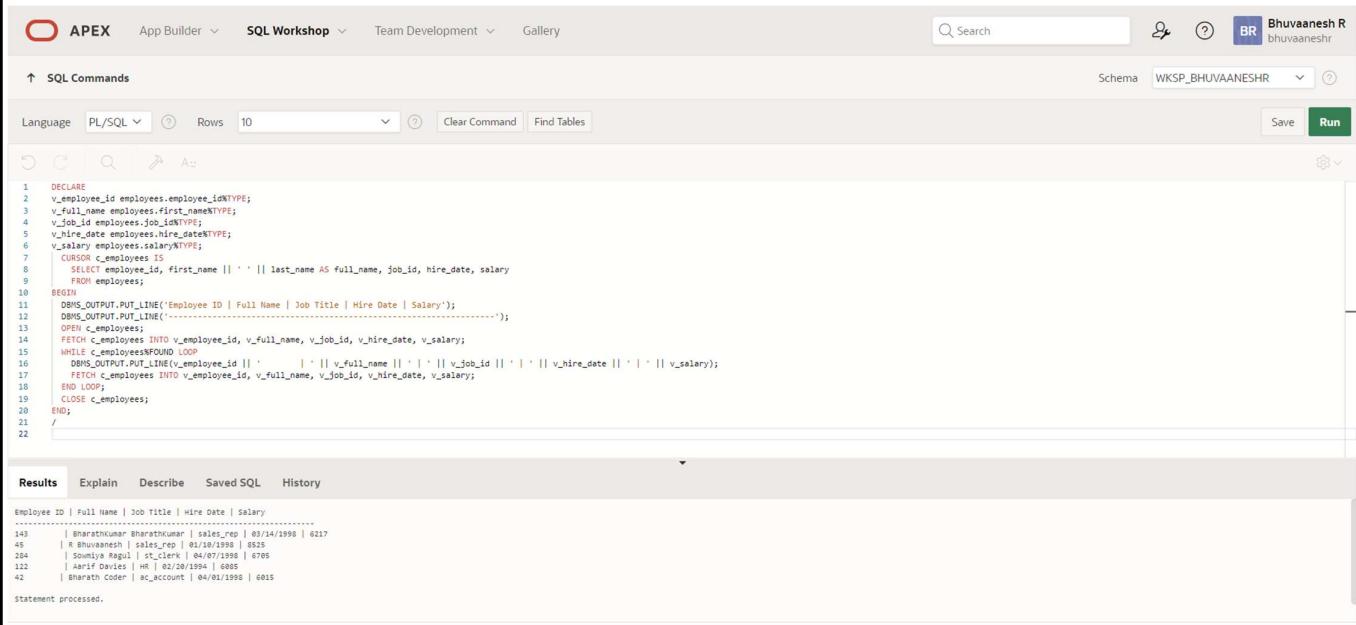
END LOOP;

CLOSE c_employees;

END;

/

OUTPUT:



```
Employee ID | Full Name | Job Title | hire date | salary
143 | Bharathkumar Bharathkumar | sales_rep | 02/14/1998 | 6217
45 | R Bhuvanesh | sales_rep | 01/10/1998 | 8525
284 | Somuya Rayal | st_clerk | 04/07/1998 | 6705
122 | Amit Davies | HR | 02/20/1994 | 4005
42 | Bharath Coder | ac_account | 04/01/1998 | 6015

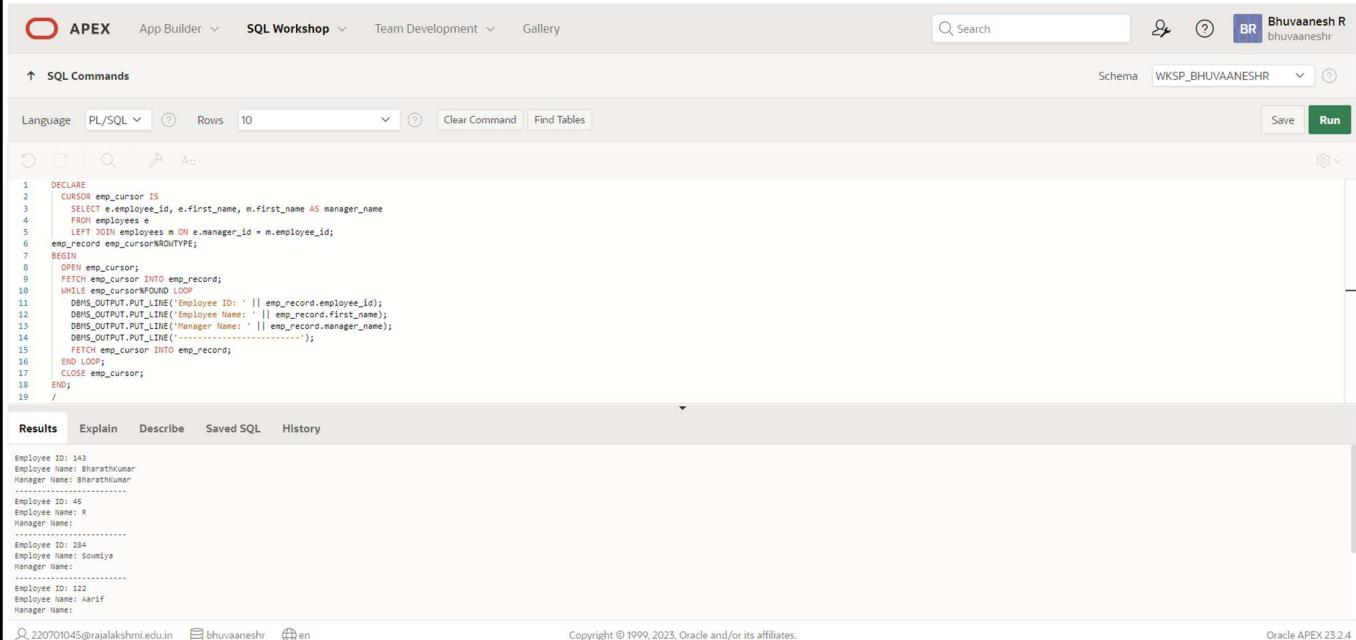
statement processed.
```

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. On the right, there's a user profile for 'Bhuvanesh R' and a schema dropdown set to 'WKSP_BHUVANESH_R'. The main workspace is titled 'SQL Commands' and contains the PL/SQL code from the previous section. Below the code, the 'Results' tab is selected, displaying the output of the program. The output shows four rows of data, each consisting of an Employee ID, Employee Name, and Manager Name, separated by a horizontal line.

Employee ID	Employee Name	Manager Name
103	Bharathkumar	Bharathkumar
48	R	
208	Somviya	
122	Aarif	

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 APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, a user icon, and the schema name 'WKSP_BHUVANESHR'. The main workspace is titled 'SQL Commands' and contains the following PL/SQL code:

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

Below the code, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab displays the output of the executed procedure:

```
Before invoking procedure, salary_of_emp: 6085
After invoking procedure, salary_of_emp: 7085
Statement processed.
```

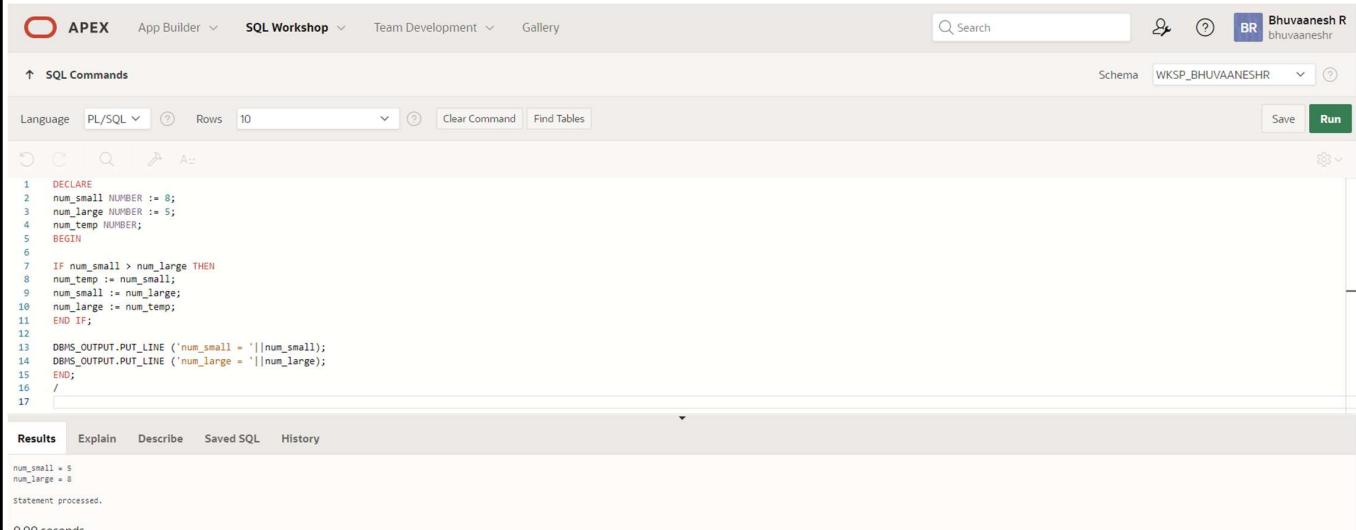
Execution time is shown as '0.01 seconds'. At the bottom, there are footer links for user information and copyright notice.

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'Bhuvanesh R' (bhuvaneshr). The main area is a code editor with the following content:

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

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

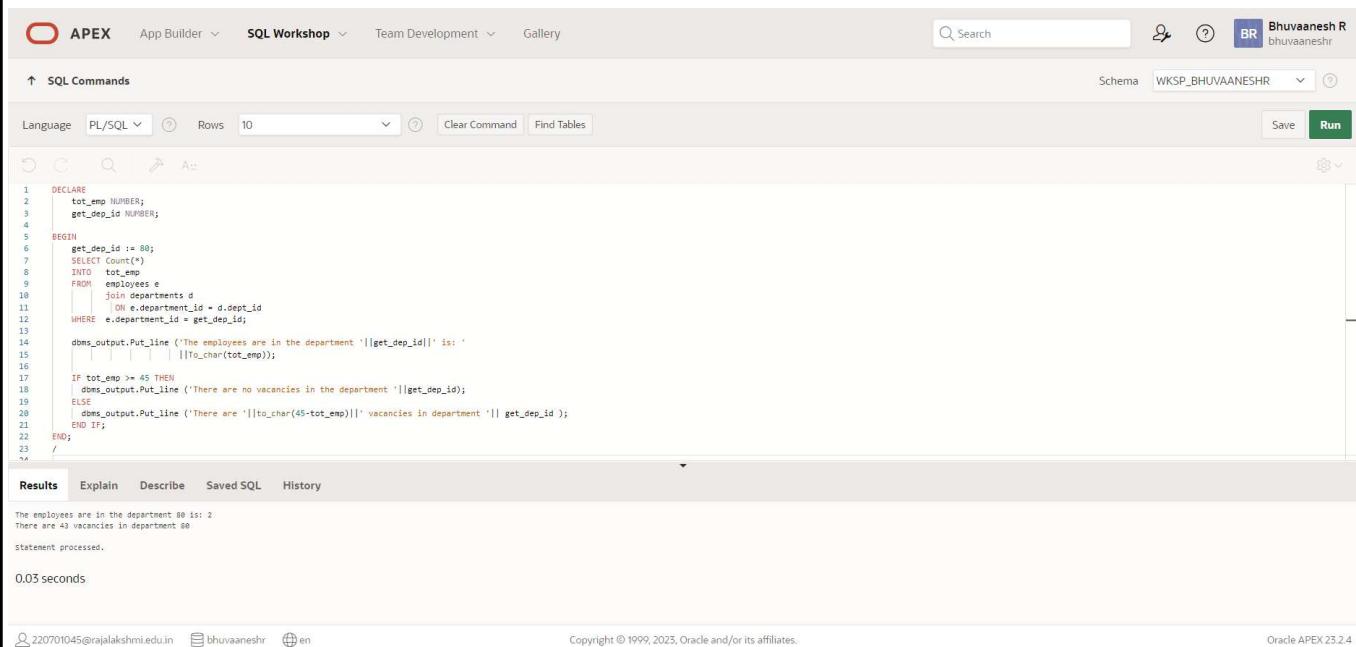
```
num_small = 5
num_large = 8
Statement processed.
0.00 seconds
```

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. 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 'Bhuvaanesh R' and the schema 'WKSP_BHUVAAINESHR'. The main workspace is titled 'SQL Commands' and contains the PL/SQL code from the previous section. Below the code, the 'Results' tab is selected, displaying the output of the program. The output shows the employee ID, name, and end date for each employee, followed by a separator line. At the bottom of the results, it indicates '0.03 seconds' for the execution time.

```
1  DECLARE
2      tot_emp NUMBER;
3      get_dep_id NUMBER;
4
5  BEGIN
6      get_dep_id := 80;
7      SELECT Count(*)
8          INTO tot_emp
9      FROM employees e
10     JOIN departments d
11       ON e.department_id = d.department_id
12      WHERE e.department_id = get_dep_id;
13
14      dbms_output.Put_line ('The employees are in the department '||get_dep_id||' is: '
15                            ||to_char(tot_emp));
16
17      IF tot_emp >= 45 THEN
18          dbms_output.Put_line ('There are no vacancies in the department '||get_dep_id);
19      ELSE
20          dbms_output.Put_line ('There are '||to_char(45-tot_emp)||' vacancies in department '|| get_dep_id );
21      END IF;
22
23  /
24
```

The employees are in the department 80 is: 2
There are 43 vacancies in department 80
Statement processed.
0.03 seconds

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

RESULT:

PROCEDURES AND FUNCTIONS

EX_NO: 17

DATE:

1.) Factorial of a number using function.

QUERY:

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

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user profile, and a connection to 'Bhuvanesh R bhuvaneshr'. The main workspace has tabs for SQL Commands, PL/SQL (selected), and Find Tables. The PL/SQL tab contains the provided PL/SQL code. Below the code, the Results tab is active, showing the output: 'Statement processed.' and '0.00 seconds'.

```
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;
11 
```

Results Explain Describe Saved SQL History

Statement processed.
0.00 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 APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows the user's profile: Bhavaanesh R (bhavaaneshr). The main area displays a PL/SQL script for a procedure named approx_salary. The script declares variables salary_of_emp, emp, empsal, and address, and performs a SELECT operation to retrieve the salary of employee_id 122. It then uses DBMS_OUTPUT.PUT_LINE to print the salary before and after invoking the procedure. The results tab shows the output of these DBMS_OUTPUT statements.

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

Results Explain Describe Saved SQL History

Before invoking procedure, salary_of_emp: 6885
After invoking procedure, salary_of_emp: 7885
Statement processed.

0.01 seconds

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

RESULT:

TRIGGER

EX_NO: 18

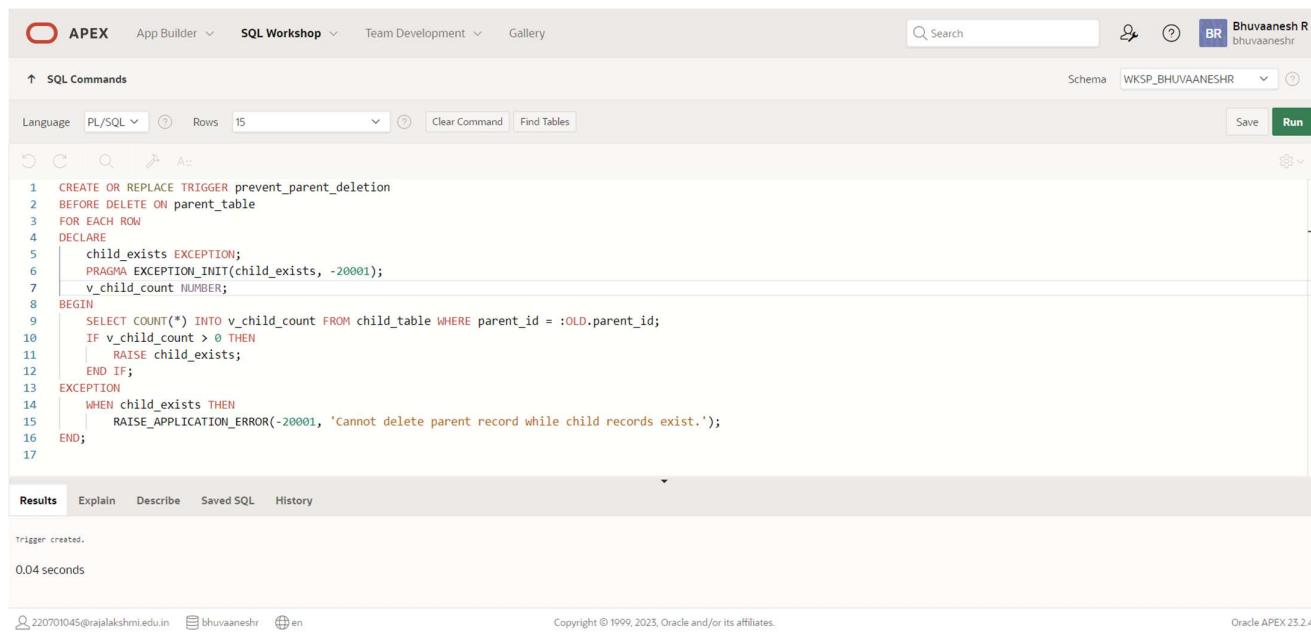
DATE:

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 APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a user profile for 'Bhavaanesh R bhavaaneshr'. The main workspace is titled 'SQL Commands' and contains the PL/SQL code for the trigger. The code is syntax-highlighted, with numbers 1 through 17 preceding each line. The bottom of the screen shows the results of the trigger creation, indicating 'Trigger created.' and a execution time of '0.04 seconds'.

```
1 CREATE OR REPLACE TRIGGER prevent_parent_deletion
2 BEFORE DELETE ON parent_table
3 FOR EACH ROW
4 DECLARE
5     child_exists EXCEPTION;
6     PRAGMA EXCEPTION_INIT(child_exists, -20001);
7     v_child_count NUMBER;
8 BEGIN
9     SELECT COUNT(*) INTO v_child_count FROM child_table WHERE parent_id = :OLD.parent_id;
10    IF v_child_count > 0 THEN
11        RAISE child_exists;
12    END IF;
13 EXCEPTION
14    WHEN child_exists THEN
15        RAISE_APPLICATION_ERROR(-20001, 'Cannot delete parent record while child records exist.');
16 END;
17
```

Results Explain Describe Saved SQL History

Trigger created.
0.04 seconds

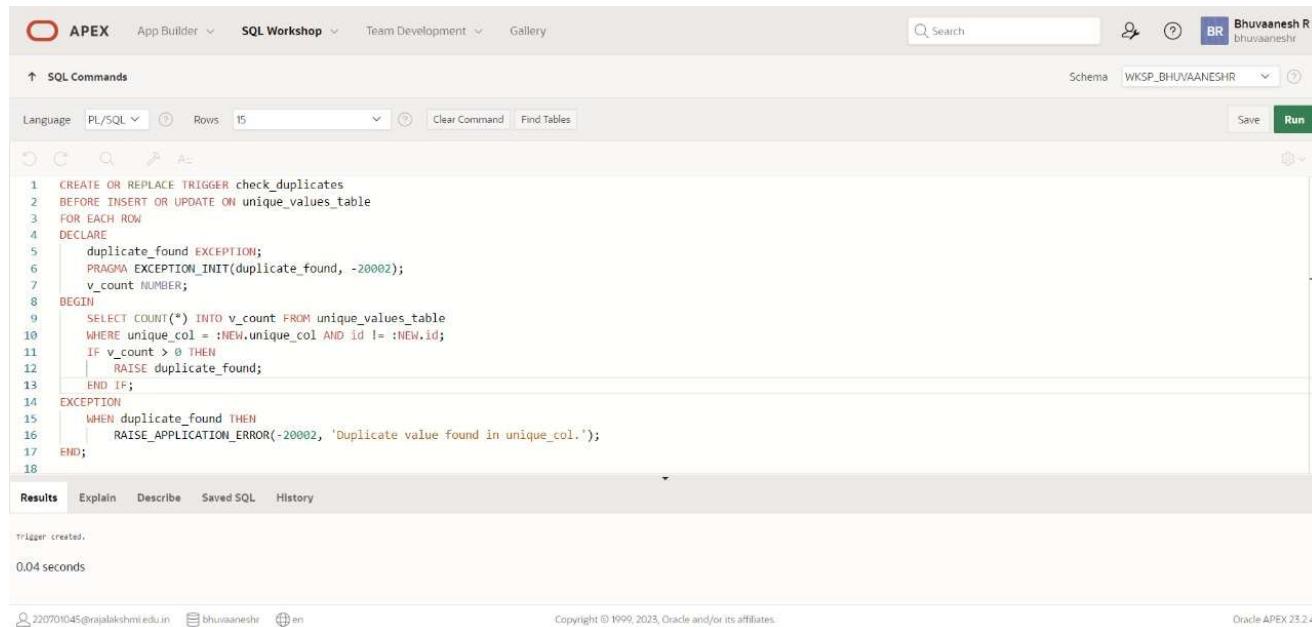
220701045@rajalakshmi.edu.in bhavaaneshr en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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 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 'Bhuvaanesh R' (bhuvaareshr). The main workspace is titled 'SQL Commands'. The schema dropdown is set to 'WKSP_BHUVAAESH R'. The code area contains the PL/SQL trigger definition provided above. The 'Results' tab at the bottom is selected, showing the output: 'Trigger created.' and '0.04 seconds'. The bottom footer includes copyright information for Oracle and the APEX version 'Oracle APEX 23.2.4'.

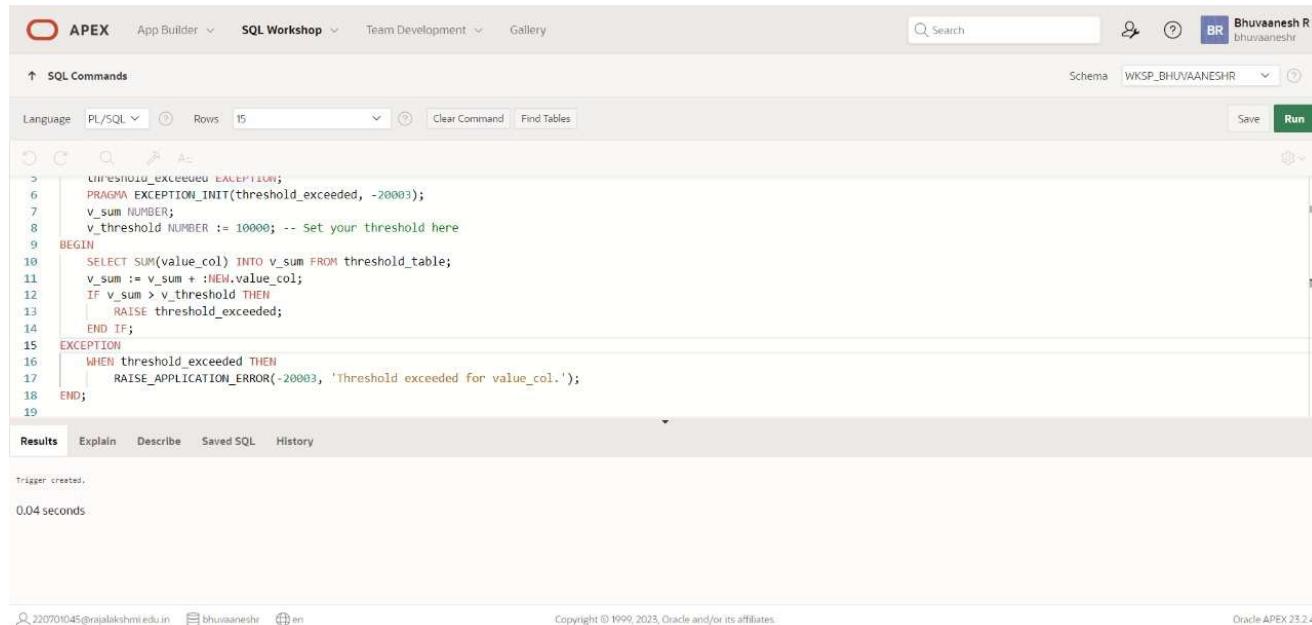
```
1 CREATE OR REPLACE TRIGGER check_duplicates
2 BEFORE INSERT OR UPDATE ON unique_values_table
3 FOR EACH ROW
4 DECLARE
5     duplicate_found EXCEPTION;
6     PRAGMA EXCEPTION_INIT(duplicate_found, -20002);
7     v_count NUMBER;
8 BEGIN
9     SELECT COUNT(*) INTO v_count FROM unique_values_table
10    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;
18
```

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', 'Gallery', 'Search', and a user icon for 'Bhuvanesh R bhuvaneshr'. The main workspace is titled 'SQL Commands' and contains the PL/SQL code for the trigger. The code is as follows:

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

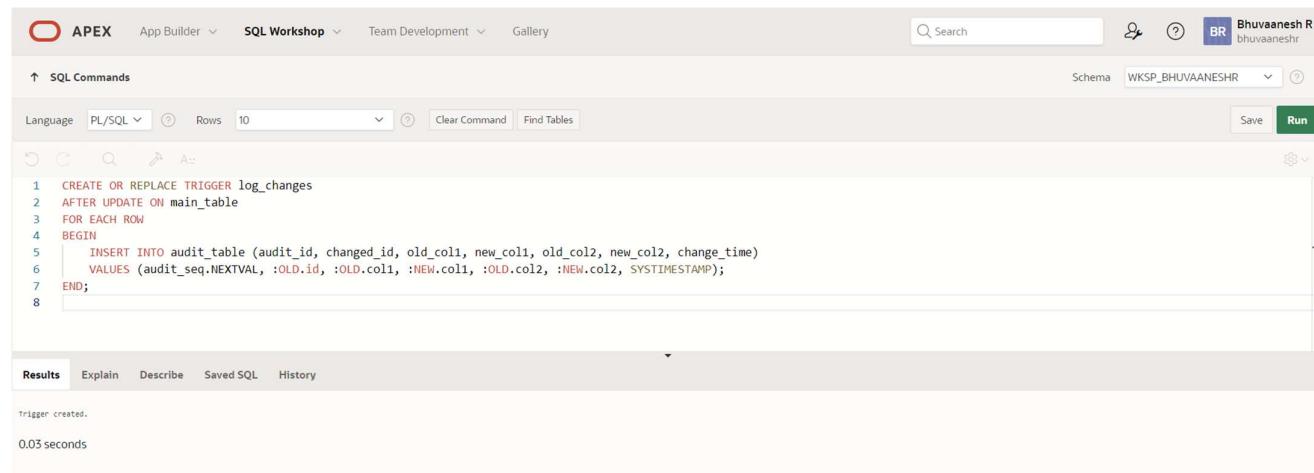
Below the code, the 'Results' tab shows the output: 'Trigger created.' and '0.04 seconds'. The bottom footer includes copyright information for Oracle and the APEX version.

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

QUERY:

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

OUTPUT:



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

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

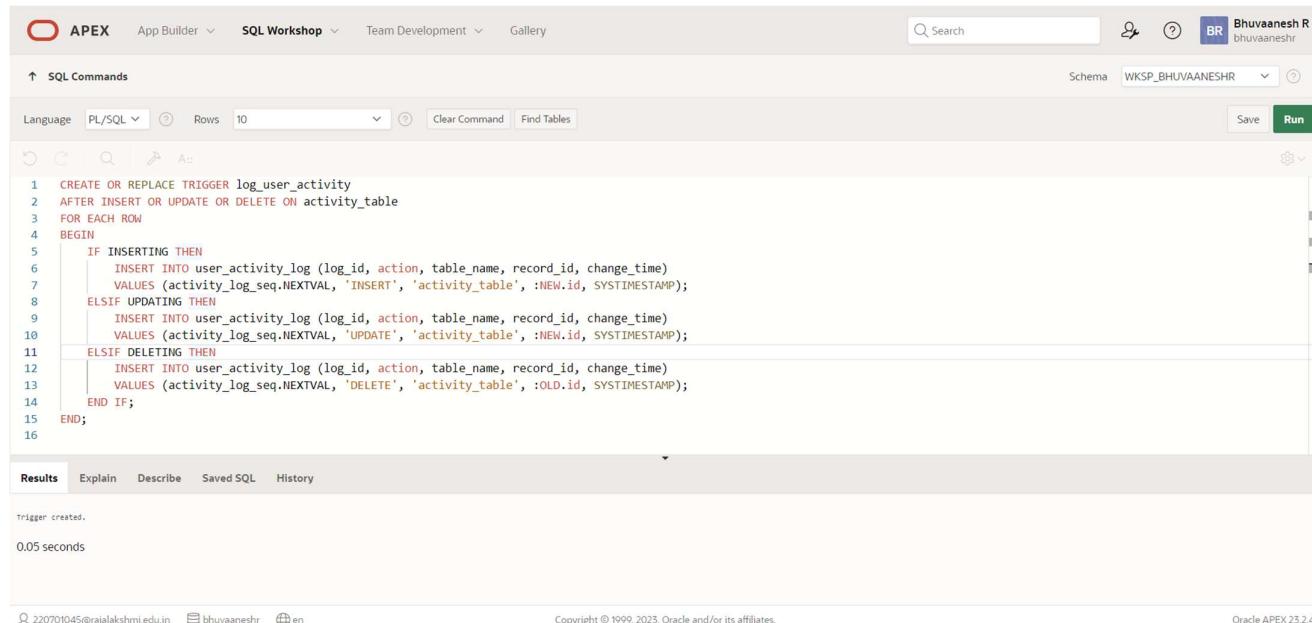
Below the code, the 'Results' tab is selected, showing the message "Trigger created." and a execution time of "0.03 seconds".

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 APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows the user 'Bhuvaanesh R' and their schema 'WKSP_BHUVAAINESHR'. The main workspace is titled 'SQL Commands' and contains the PL/SQL code for the trigger. The code is syntax-highlighted, showing keywords in red and identifiers in blue. The 'Run' button is visible at the top right of the code area. Below the code, the 'Results' tab is selected, displaying the message 'Trigger created.' and '0.05 seconds'. The bottom footer includes user information and copyright details.

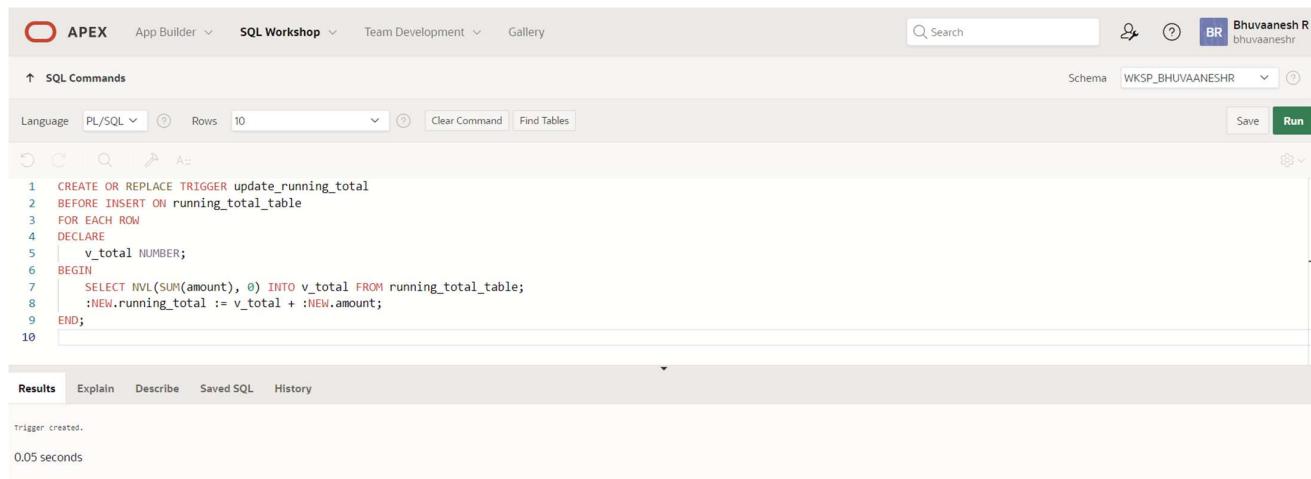
```
1 CREATE OR REPLACE TRIGGER log_user_activity
2 AFTER INSERT OR UPDATE OR DELETE ON activity_table
3 FOR EACH ROW
4 BEGIN
5   IF INSERTING THEN
6     INSERT INTO user_activity_log (log_id, action, table_name, record_id, change_time)
7       VALUES (activity_log_seq.NEXTVAL, 'INSERT', 'activity_table',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

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows the user 'Bhuvaanesh R' with the identifier 'WKSP_BHUVANESHR'. The main workspace is titled 'SQL Commands' and contains the PL/SQL code for the trigger. The code is as follows:

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

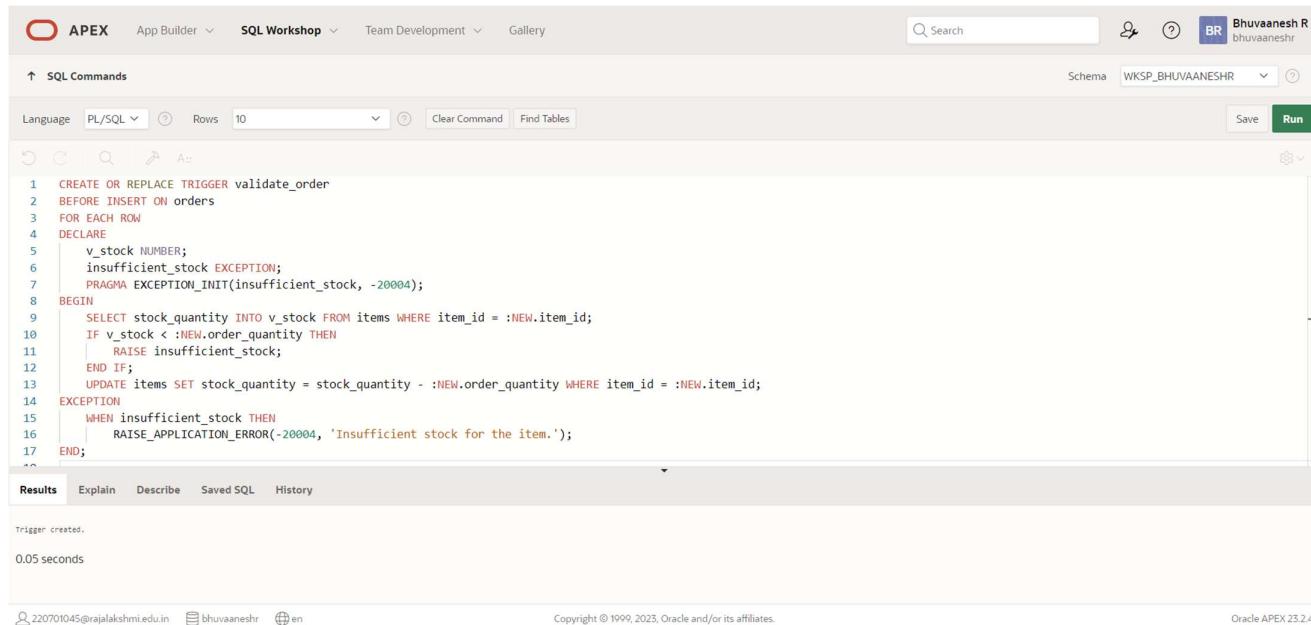
Below the code, the 'Results' tab is selected, showing the message 'Trigger created.' and '0.05 seconds'.

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', 'Gallery', 'Search', and user information 'Bhavaanesh R bhavaaneshr'. The main workspace is titled 'SQL Commands' and contains the PL/SQL code for the trigger. The code is highlighted in blue and black. The bottom section shows the 'Results' tab with the message 'Trigger created.' and a timestamp '0.05 seconds'. The footer includes copyright information 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

```
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;
```

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

RESULT:

MONGO DB

EX_NO: 19

DATE:

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code is:1 db.restaurants.find(
2 {
3 \$or: [
4 { cuisine: { \$nin: ['American', 'Chinese'] } },
5 { name: { \$regex: '^Wil', \$options: 'i' } }
6]
7 },
8 { restaurant_id: 1, name: 1, borough: 1, cuisine: 1 }
9);
10

On the right, the output window shows the command being run and the result:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code is:

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

On the right, the output window shows the command being run and the result:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code is:

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

On the right, the output window shows the command being run and the result:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code block contains the following command:

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

On the right, the output panel shows the results of the execution. It includes a header 'Output' and the following text:

```
mycompiler_mongodb> ... ... ... ... ... Uncaught
MongoServerError[Location31249]: Path collision at address.co
mycompiler_mongodb>
[Execution complete with exit code 0]
```

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code block contains the following command:

```
1 db.restaurants.find().sort({ name: 1 });
2
```

On the right, the output panel shows the results of the execution. It includes a header 'Output' and the following text:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code block contains the following command:

```
1 db.restaurants.find().sort({ name: -1 });
2
```

On the right, the output panel shows the results of the execution. It includes a header 'Output' and the following text:

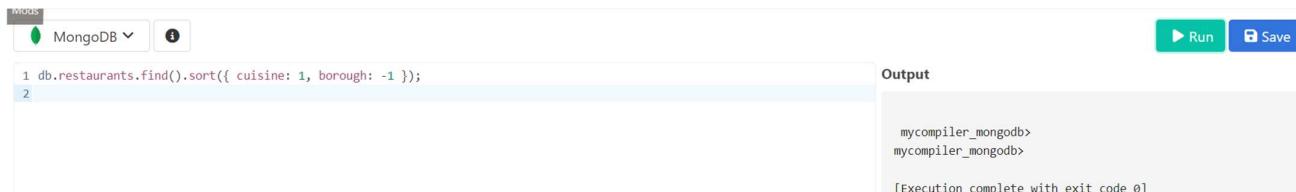
```
mycompiler_mongodb>
mycompiler_mongodb>
[Execution complete with exit code 0]
```

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:



The screenshot shows the MongoDB shell interface. On the left, there is a code editor with the following command:

```
1 db.restaurants.find().sort({ cuisine: 1, borough: -1 });  
2
```

On the right, there is an "Output" panel displaying the results of the command:

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

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

QUERY:

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

OUTPUT:



The screenshot shows the MongoDB shell interface. On the left, there is a code editor with the following command:

```
1 db.restaurants.find({ "address.street": { $exists: true } });  
2
```

On the right, there is an "Output" panel displaying the results of the command:

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

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

QUERY:

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

OUTPUT:



The screenshot shows the MongoDB shell interface. On the left, there is a code editor with the following command:

```
1 db.restaurants.find(  
2   { "address.coord": { $type: "double" } }  
3 );  
4 |
```

On the right, there is an "Output" panel displaying the results of the command:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code block contains the query:

```
1 db.restaurants.find()
2   { "grades.score": { $mod: [7, 0] } },
3   { restaurant_id: 1, name: 1, grades: 1 }
4 };
5
```

. On the right, the 'Output' panel shows the results of the execution, which are empty, followed by the message '[Execution complete with exit code 0]'.

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code block contains the query:

```
1 db.restaurants.find()
2   { name: { $regex: 'mon', $options: 'i' } },
3   { name: 1, borough: 1, "address.coord": 1, cuisine: 1 }
4 };
5
```

. On the right, the 'Output' panel shows the results of the execution, which are empty, followed by the message '[Execution complete with exit code 0]'.

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code block contains the query:

```
1 db.restaurants.find()
2   { name: { $regex: '^Mad', $options: 'i' } },
3   { name: 1, borough: 1, "address.coord": 1, cuisine: 1 }
4 };
5
```

. On the right, the 'Output' panel shows the results of the execution, which are empty, followed by the message '[Execution complete with exit code 0]'.

13. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5.

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code block contains the query:

```
1 db.restaurants.find(
2   { "grades.score": { $lt: 5 } }
3 );
4
```

. On the right, the 'Output' panel shows the results of the execution, which are empty, followed by the message '[Execution complete with exit code 0]'.

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code block contains the following command:

```
1 db.restaurants.find
2 { "grades.score": { $lt: 5 }, borough: 'Manhattan' }
3 ;
4
```

On the right, the output window shows the results of the query execution:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code block contains the following command:

```
1 db.restaurants.find
2 {
3     "grades.score": { $lt: 5 },
4     borough: { $in: ['Manhattan', 'Brooklyn'] }
5 }
6 ;
7
```

On the right, the output window shows the results of the query execution:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code block contains the following command:

```
1 db.restaurants.find
2 {
3     "grades.score": { $lt: 5 },
4     borough: { $in: ['Manhattan', 'Brooklyn'] },
5     cuisine: { $ne: 'American' }
6 }
7 ;
8
```

On the right, the output window shows the results of the query execution:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code is pasted:

```
1 db.restaurants.find(
2   {
3     "grades.score": { $lt: 5 },
4     borough: { $in: ['Manhattan', 'Brooklyn'] },
5     cuisine: { $nin: ['American', 'Chinese'] }
6   }
7 );
8
```

On the right, the output window shows the command being run and the response:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code is pasted:

```
1 db.restaurants.find(
2   {
3     grades: { $all: [
4       { $elemMatch: { score: 2 } },
5       { $elemMatch: { score: 6 } }
6     ]}
7   }
8 );
9
```

On the right, the output window shows the command being run and the response:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code is pasted:

```
1 db.restaurants.find(
2   {
3     grades: { $all: [
4       { $elemMatch: { score: 2 } },
5       { $elemMatch: { score: 6 } }
6     ]},
7     borough: 'Manhattan'
8   }
9 );
10
```

On the right, the output window shows the command being run and the response:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code is displayed:

```
1 db.restaurants.find(
2   {
3     grades: { $all: [
4       { $elemMatch: { score: 2 } },
5       { $elemMatch: { score: 6 } }
6     ]},
7     borough: { $in: ['Manhattan', 'Brooklyn'] }
8   }
9 );
10
11
```

On the right, the 'Output' panel shows the command being run and the response:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code is displayed:

```
1 db.restaurants.find(
2   {
3     grades: { $all: [
4       { $elemMatch: { score: 2 } },
5       { $elemMatch: { score: 6 } }
6     ]},
7     borough: { $in: ['Manhattan', 'Brooklyn'] },
8     cuisine: { $ne: 'American' }
9   }
10 );
11
```

On the right, the 'Output' panel shows the command being run and the response:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code is displayed:

```
1 db.restaurants.find(
2   {
3     grades: { $all: [
4       { $elemMatch: { score: 2 } },
5       { $elemMatch: { score: 6 } }
6     ]},
7     borough: { $in: ['Manhattan', 'Brooklyn'] },
8     cuisine: { $nin: ['American', 'Chinese'] }
9   }
10 );
11
```

On the right, the 'Output' panel shows the command being run and the response:

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

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

QUERY:

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

OUTPUT:

The screenshot shows a MongoDB shell interface. On the left, the code is displayed:

```
1 db.restaurants.find(
2   {
3     $or: [
4       { "grades.score": 2 },
5       { "grades.score": 6 }
6     ]
7   }
8 );
9
```

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

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

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

RESULT:

MONGO DB

EX_NO: 20

DATE:

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, there is a code editor window containing the following command:

```
1 db.movies.find()
2   { year: 1893 }
3 );
4
```

On the right, there is an "Output" window showing the results of the command:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, there is a code editor window containing the following command:

```
1 db.movies.find()
2   { runtime: { $gt: 120 } }
3 );
4
```

On the right, there is an "Output" window showing the results of the command:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, there is a code editor window containing the following command:

```
1 db.movies.find(
2   { genres: 'Short' }
3 );
4
```

On the right, there is an "Output" window showing the results of the command:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, there is a code editor with the following command:

```
1 db.movies.find(  
2   { directors: 'William K.L. Dickson' }  
3 );  
4
```

On the right, there is an "Output" panel with the results of the execution:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, there is a code editor with the following command:

```
1 db.movies.find(  
2   { countries: 'USA' }  
3 );  
4
```

On the right, there is an "Output" panel with the results of the execution:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, there is a code editor with the following command:

```
1 db.movies.find(  
2   { rated: 'UNRATED' }  
3 );  
4
```

On the right, there is an "Output" panel with the results of the execution:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, there is a code editor with the following command:

```
1 db.movies.find(
2   { "imdb.votes": { $gt: 1000 } }
3 );
4
```

On the right, there is an "Output" panel with the following results:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, there is a code editor with the following command:

```
1 db.movies.find(
2   { "imdb.rating": { $gt: 7 } }
3 );
4
```

On the right, there is an "Output" panel with the following results:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, there is a code editor with the following command:

```
1 db.movies.find(
2   { "tomatoes.viewer.rating": { $gt: 4 } }
3 );
4
```

On the right, there is an "Output" panel with the following results:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code block contains the command: db.movies.find({ "awards.wins": { \$gt: 0 } }). On the right, the output panel shows the results of the query execution, which is empty in this case. A green 'Run' button and a blue 'Save' button are visible at the top right.

```
1 db.movies.find(  
2   { "awards.wins": { $gt: 0 } }  
3 );  
4
```

Output

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code block contains the command: 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 }). On the right, the output panel shows the results of the query execution, which is empty in this case. A green 'Run' button and a blue 'Save' button are visible at the top right.

```
1 db.movies.find(  
2   { "awards.nominations": { $gt: 0 } },  
3   {  
4     title: 1, languages: 1, released: 1, directors: 1, writers: 1,  
5     awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1  
6   }  
7 );  
8
```

Output

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code block contains the command: 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 }). On the right, the output panel shows the results of the query execution, which is empty in this case. A green 'Run' button and a blue 'Save' button are visible at the top right.

```
1 db.movies.find(  
2   { cast: 'Charles Kayser' },  
3   {  
4     title: 1, languages: 1, released: 1, directors: 1, writers: 1,  
5     awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1  
6   }  
7 );  
8
```

Output

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code block contains the following command:

```
1 db.movies.find(
2   { released: new ISODate("1893-05-09T00:00:00.000Z") },
3   {
4     title: 1, languages: 1, released: 1, directors: 1, writers: 1,
5     countries: 1
6   }
7 );
8
```

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

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code block contains the following command:

```
1 db.movies.find(
2   { title: { $regex: 'scene', $options: 'i' } },
3   {
4     title: 1, languages: 1, released: 1, directors: 1, writers: 1,
5     countries: 1
6   }
7 );
8
```

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

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

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

RESULT: