

EXNO:1**CREATING AND MANAGING TABLE****DATE:**

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

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

QUERY: Create table DEPARTMENT(id number(7),name varchar2(25));

OUTPUT:

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

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

Below the command, the results show:

Table created.
0.03 seconds

At the bottom, the footer includes:

22070101@rajalakshmi.edu.in ab1507 en Copyright © 1999, 2025, Oracle and/or its affiliates. Oracle APEX 23.2.4

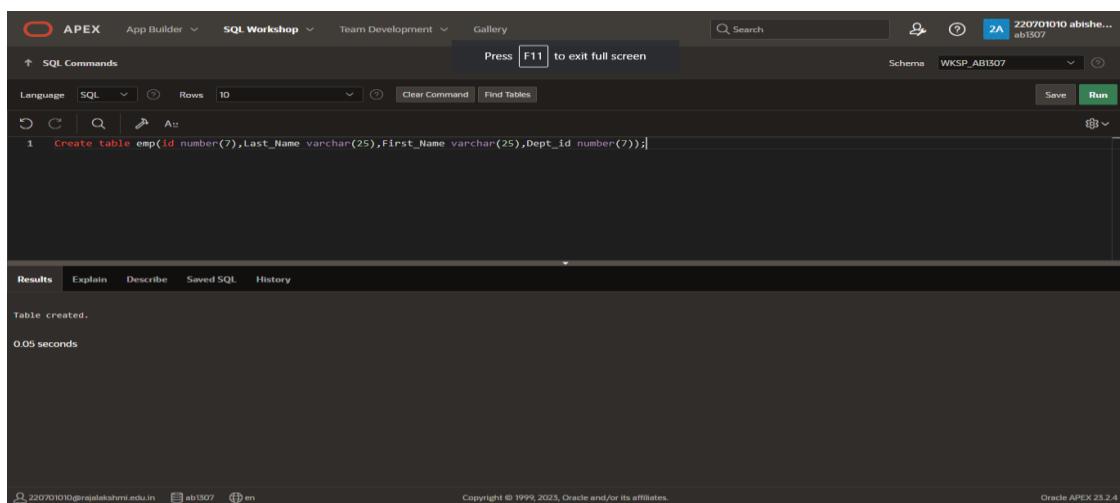
2. Create the EMP table based on the following instance chart. Confirm that the table is created.

Column name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				
Nulls/Unique				

FK table				
FK column				
Data Type	Number	Varchar2	Varchar2	Number
Length	7	25	25	7

QUERY: Create table emp(id number(7),Last_Name varchar(25),First_Name varchar(25),Dept_id number(7));

OUTPUT:



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

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

After running the command, the results show:

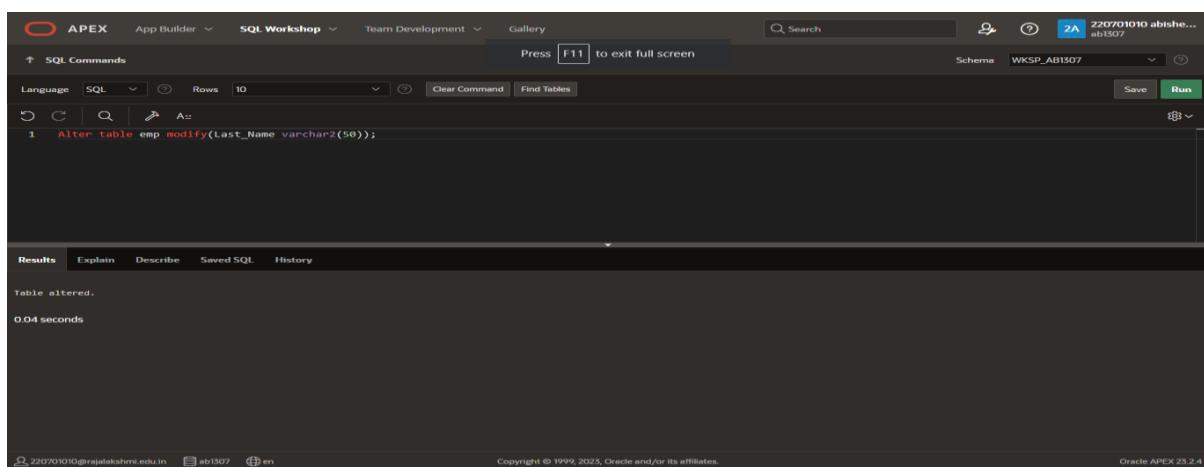
```
Table created.
```

Execution time: 0.05 seconds.

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

QUERY: Alter table emp modify(Last_Name varchar2(50));

OUTPUT:



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

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

After running the command, the results show:

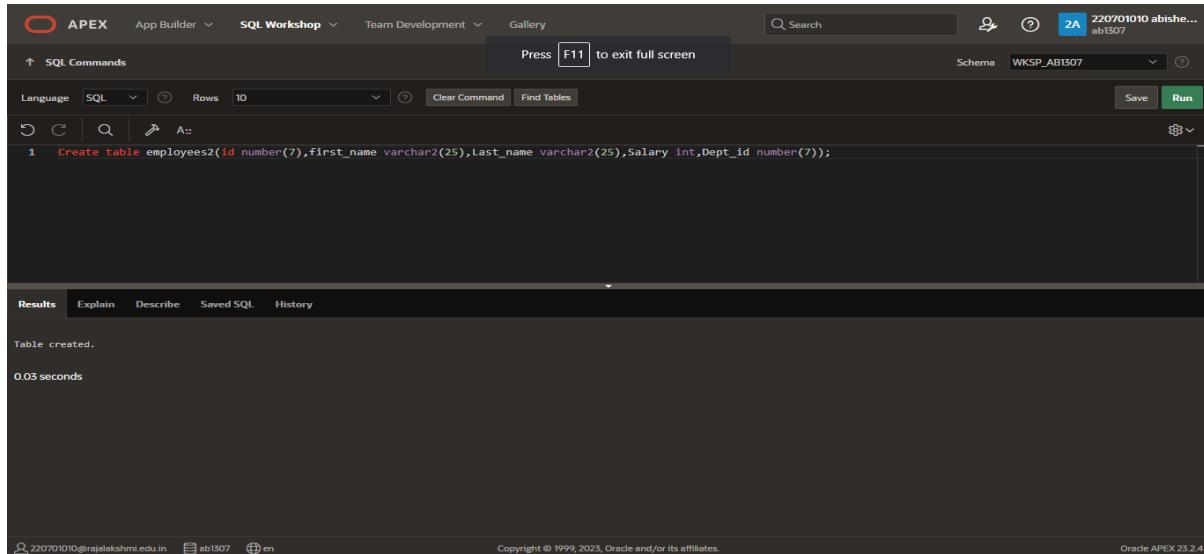
```
Table altered.
```

Execution time: 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 columns.
Name the columns Id, First_name, Last_name, salary and Dept_id respectively.**

QUERY: Create table employees2(id number(7),first_name varchar2(25),Last_name varchar2(25),Salary int,Dept_id number(7));

OUTPUT:



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

```
1 Create table employees2(id number(7),first_name varchar2(25),Last_name varchar2(25),Salary int,Dept_id number(7));
```

Below the command, the results show:

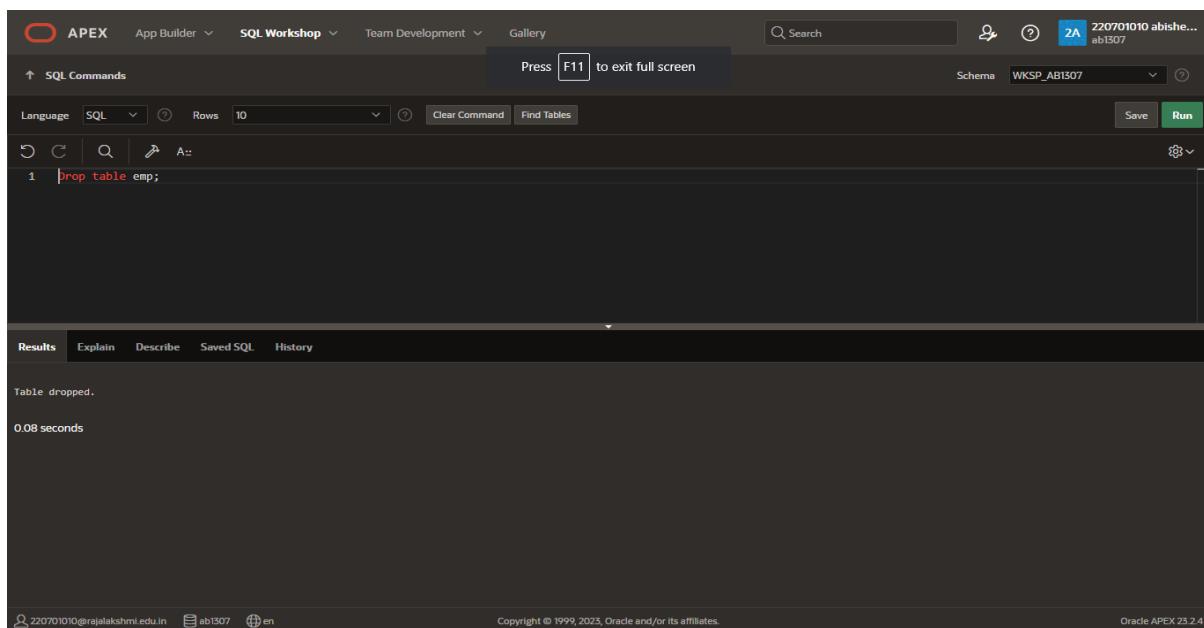
```
Table created.
0.03 seconds
```

The bottom status bar indicates the user is 220701010@rajalakshmi.edu.in and the session ID is ab1507. The copyright notice is from Oracle and the version is Oracle APEX 23.2.4.

5. Drop the EMP table.

QUERY: Drop table emp;

OUTPUT:



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

```
1 drop table emp;
```

Below the command, the results show:

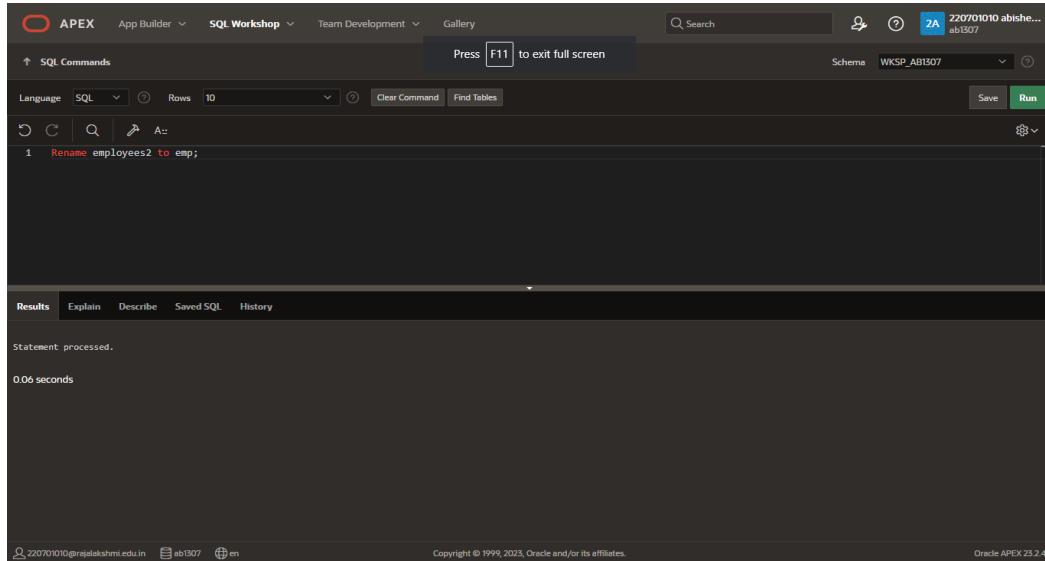
```
Table dropped.
0.08 seconds
```

The bottom status bar indicates the user is 220701010@rajalakshmi.edu.in and the session ID is ab1507. The copyright notice is from Oracle and the version is Oracle APEX 23.2.4.

6. Rename the EMPLOYEES2 table as EMP.

QUERY: Rename employees2 to emp;

OUTPUT:



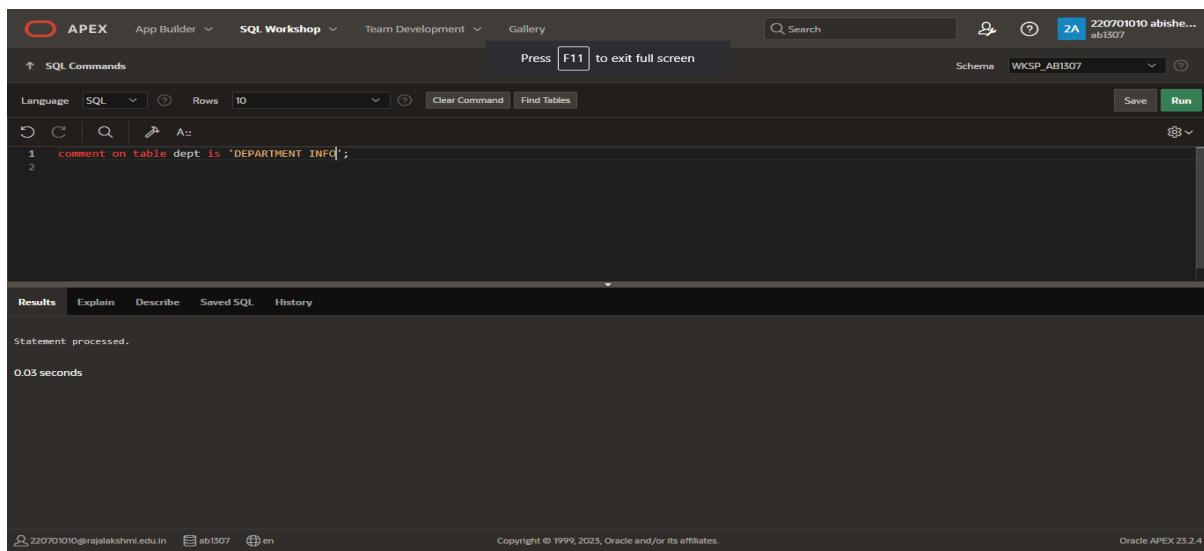
A screenshot of the Oracle APEX SQL Workshop interface. The top navigation bar shows "APEX", "App Builder", "SQL Workshop", "Team Development", and "Gallery". The "SQL Workshop" tab is active. The main area is titled "SQL Commands" with a sub-section "Language: SQL". A command is being typed into the editor: "1 Rename employees2 to emp;". Below the editor, the "Results" tab is selected, showing the output: "Statement processed." and "0.06 seconds". The bottom status bar includes the user ID "220701010@rajalakshmi.edu.in", session ID "ab1507", and locale "en".

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

QUERY:

comment on table dept is 'Department info';

OUTPUT:



A screenshot of the Oracle APEX SQL Workshop interface, similar to the previous one. The "SQL Workshop" tab is active. The main area shows the command: "1 comment on table dept is 'DEPARTMENT INFO';". The "Results" tab is selected, displaying "Statement processed." and "0.03 seconds". The bottom status bar includes the user ID "220701010@rajalakshmi.edu.in", session ID "ab1507", and locale "en".

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

QUERY: Alter table emp drop column first_name;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands pane, the command `Alter table emp drop column first_name;` is entered. The Results pane displays the output: `Table altered.` and `0.06 seconds`. The schema is set to `WKSP_AB1307`.

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

RESULT:

EXNO:2

MANIPULATING DATA

DATE:

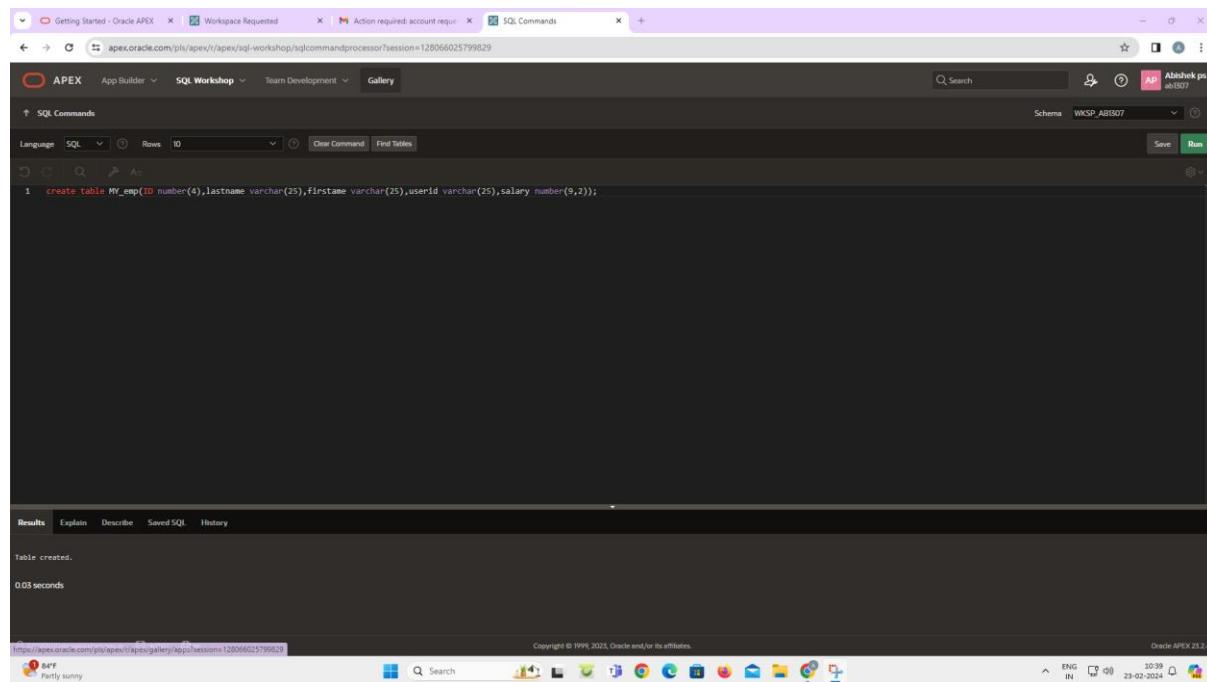
1. Create MY_EMPLOYEE table with the following structure

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

QUERY:

```
Create table My_emp(ID number(4),lastname varchar(25),firstname  
varchar(25),user_id varchar(25),salary number(9,2));
```

OUTPUT:



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

```
create table MY_emp(ID number(4),lastname varchar(25),firstname varchar(25),user_id varchar(25),salary number(9,2));
```

The Results tab displays the output of the command:

```
Table created.  
0.03 seconds
```

The browser address bar shows the URL: <https://apex.oracle.com/pls/apex/r/apex/sql-commandprocessor?session=128066025799629>.

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

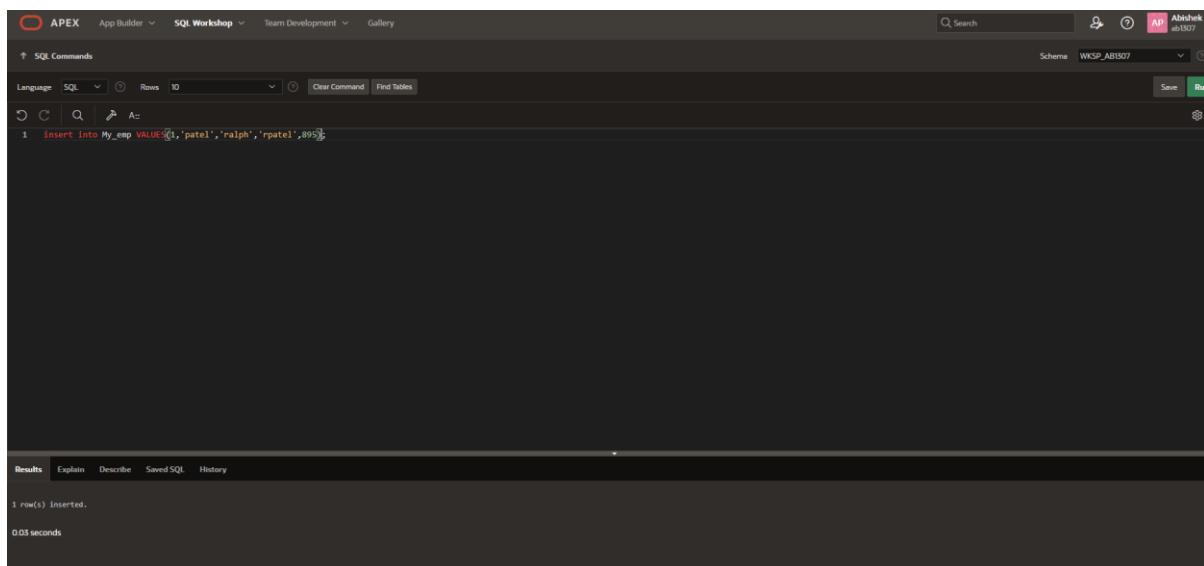
ID	Last_name	First_name	Userid	salary
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	Cnewman	750
5	Ropebur	Audrey	aropebur	1550

QUERY:

```
INSERT INTO My_emp(1,'patel','ralph','rpatel',895);
```

```
INSERT INTO My_emp(2,'dancs','betty','bdancs',860);
```

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 Abhishek. The main area is titled 'SQL Commands' with tabs for Language (SQL selected), Rows (10), and Clear Command/Find Tables. Below this is a code editor with the following content:

```
1 insert into My_emp VALUES(1,'patel','ralph','rpatel',895);
```

The results tab at the bottom displays the output of the executed command:

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

Execution time is listed as 0.03 seconds.

3. Display the table with values.

QUERY: select * from My_emp;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, the command `select * from My_emp;` is entered. The Results tab displays the following data:

ID	LNAME	FNAME	USERID	SALARY
1	patel	ralph	rpatel	895
2	dancs	betty	bdanc	860

2 rows returned in 0.01 seconds

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

QUERY: update My_emp set lname="dexler" where id=3;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, the command `select * from My_emp;` is entered. The Results tab displays the following data:

ID	LASTNAME	FIRSTNAME	USERID	SALARY
2	dancs	betty	bdanc	860
6	smith	blnc	bsmith	8100
4	newman	chad	cnewman	750
3	dexter	ben	bbri	1100
5	ropeur	audrey	aropebur	1550
7	mariana	clark	mcclrk	5020
1	patel	ralph	rpatel	895

7 rows returned in 0.01 seconds

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

QUERY: update My_emp set salary=1000 where salary<900;

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 main area is titled 'SQL Commands'. The command entered is 'select * from My_emp;'. The results section displays a table with four rows of employee data:

ID	LNAME	FNAME	USERID	SALARY
3	dexter	ben	bbiri	1100
1	patel	ralph	rpatel	1000
4	newman	chad	cnewman	1000

Below the table, it says '4 rows returned in 0.00 seconds' and has a 'Download' link. The bottom status bar shows the user's email (220701010@rajalakshmi.edu.in), session ID (ab1307), and language (en). The date and time are also displayed.

6.Delete Betty dancs from MY _EMPLOYEE table.

QUERY: delete from My_emp where lname='dancs';

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 main area is titled 'SQL Commands'. The command entered is 'delete from My_emp where lname='dancs';'. The results section shows the message '1 row(s) deleted.' and '0.04 seconds'.

7.Empty the fourth row of the emp table.

QUERY: delete from My_emp where id=4;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The main area contains a SQL command window with the following content:

```
1 select * from My_emp;
```

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

ID	LNAME	FNAME	USERID	SALARY	COMMISSION	DEPT_ID
3	dexler	ben	bbiri	1100	-	-
1	patel	ralph	rpatel	1000	-	-

Text at the bottom of the results pane indicates "2 rows returned in 0.01 seconds". The footer of the page includes user information (220701010@rajalakshmi.edu.in, ab1307, en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the software version (Oracle APEX 23.2.4).

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

RESULT:

EXNO:3

INCLUDING CONSTRAINTS

DATE:

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

QUERY: ALTER TABLE My_emp ADD CONSTRAINT my_emp_id_pk PRIMARY KEY(ID);

OUTPUT:

Column Name	Data Type	Nullable	Default	Primary Key	Comment	Identity
ID	NUMBER(4,0)	N		1		
LNAME	VARCHAR2(25 BYTE)	Y				
FNAME	VARCHAR2(25 BYTE)	Y				
USERID	VARCHAR2(25 BYTE)	Y				
SALARY	NUMBER(9,2)	Y				

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

QUERY: CREATE TABLE DEPT(DEPT_ID number(4),deptname varchar(25),CONSTRAINT my_dept_id_pk PRIMARY KEY(DEPT_ID));

OUTPUT:

```
1 create table dept(dept_id number(4),dept_name varchar(28),CONSTRAINT my_dept_id_pk PRIMARY KEY(dept_id));
```

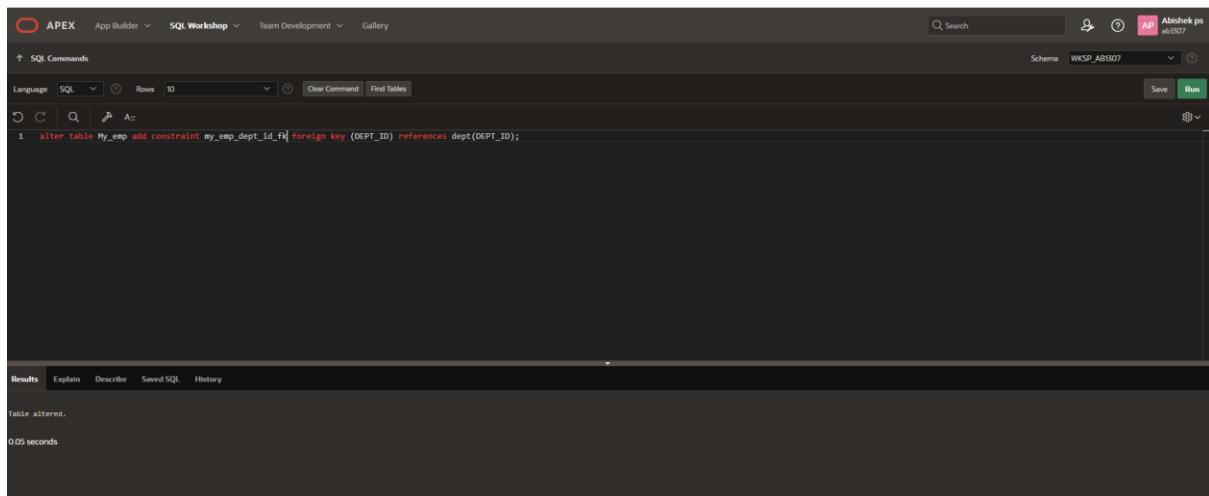
Results Explain Describe Saved SQL History

Table created.
0.06 seconds

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

QUERY: ALTER TABLE My_emp ADD CONSTRAINT my_emp_dept_id_fk FOREIGN KEY(DEPT_ID) REFERENCES DEPT(DEPT_ID);

OUTPUT:



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

```
1 alter table My_emp add constraint my_emp_dept_id_fk foreign key (DEPT_ID) references dept(DEPT_ID);
```

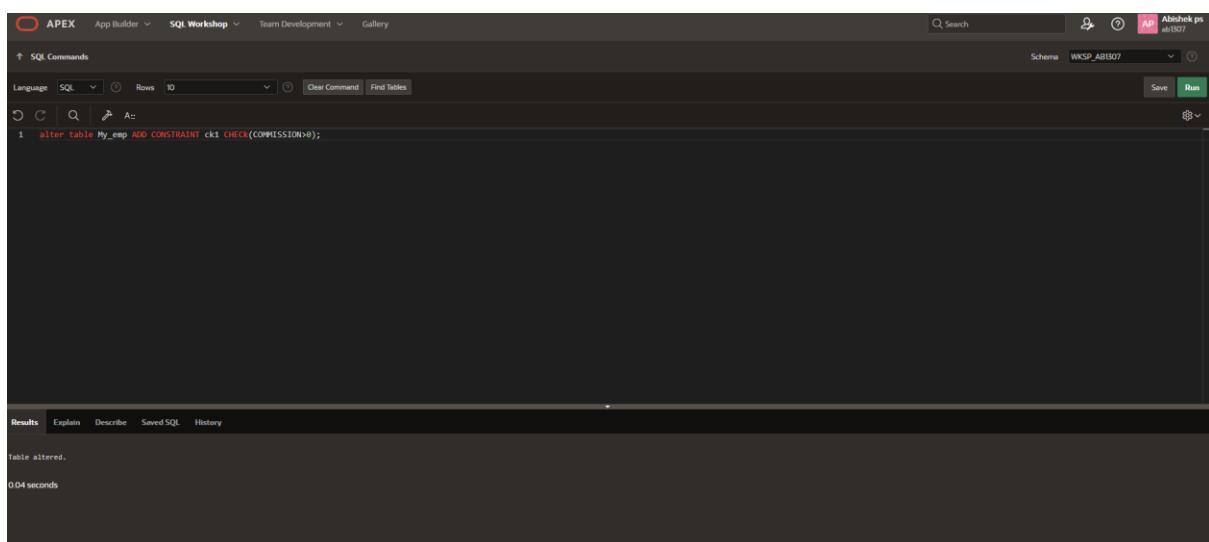
Below the command, the results show:

```
Table altered.  
0.05 seconds
```

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

QUERY: ALTER TABLE My_emp ADD COMMISIO N number(2,2);
ALTER TABLE My_emp ADD CONSTRAINT CK1 CHECK(COMMISSION>0);

OUTPUT:



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

```
1 alter table My_emp ADD CONSTRAINT ck1 CHECK(COMMISSION>0);
```

Below the command, the results show:

```
Table altered.  
0.04 seconds
```

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

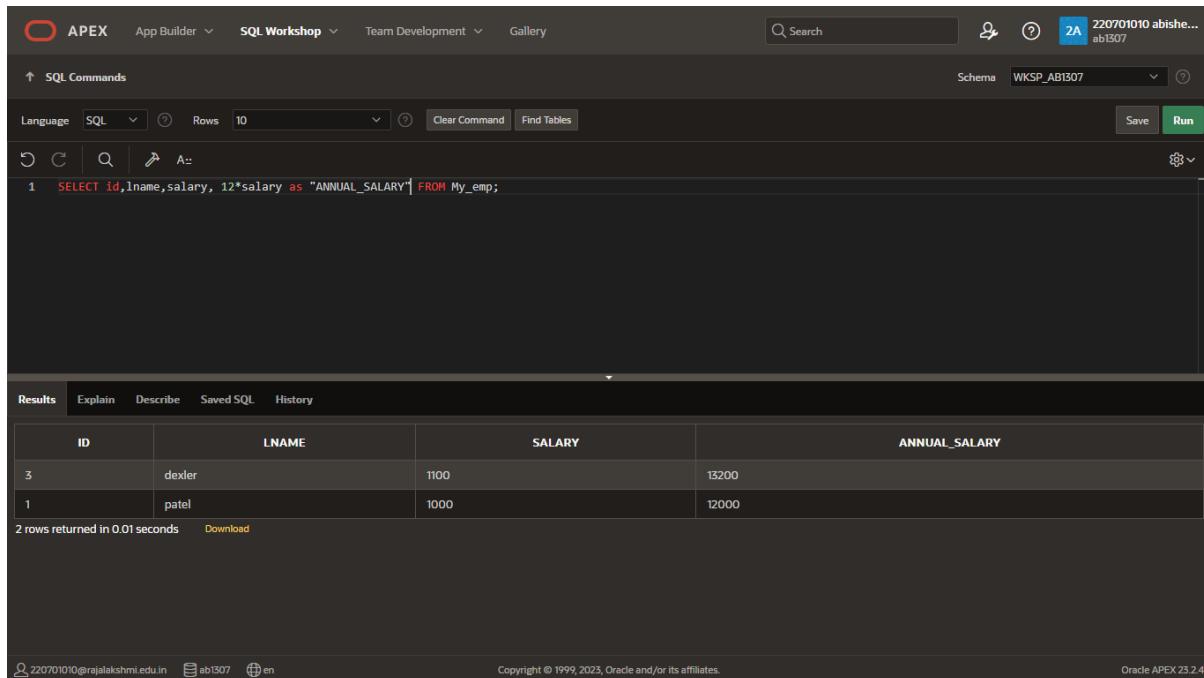
RESULT:

1.Identify the Errors

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

QUERY:

```
SELECT id, lname ,sal,sal*12 as "ANNUAL SALARY" FROM My_emp;
```

OUTPUT:


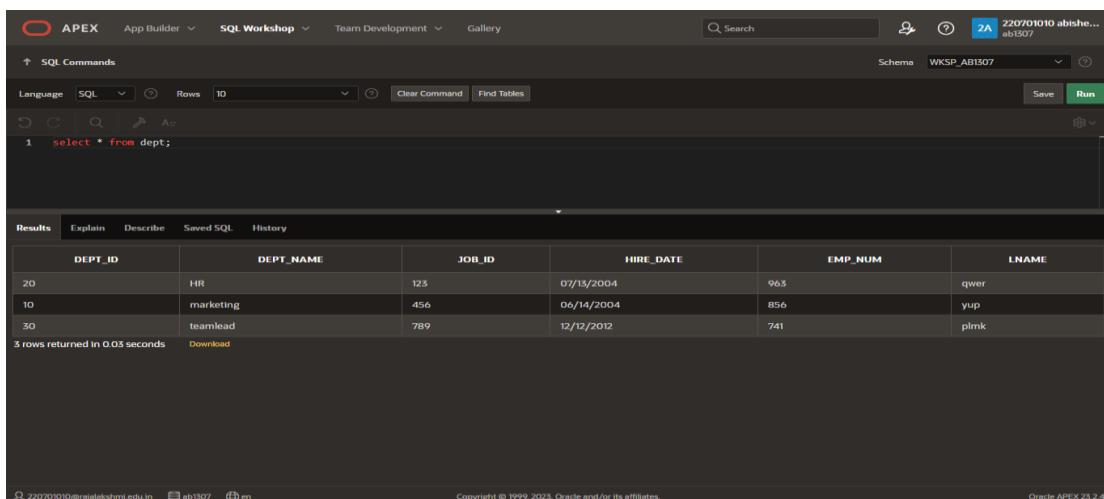
The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side of the header shows the user's name (220701010 abishe...), session ID (ab1307), and schema (WKSP_AB1307). The main workspace is titled 'SQL Commands' and contains the following SQL code:

```
1  SELECT id,lname,salary, 12*salary as "ANNUAL_SALARY" FROM My_emp;
```

The results section displays the output of the query:

ID	LNAME	SALARY	ANNUAL_SALARY
3	dexler	1100	13200
1	patel	1000	12000

Below the table, it says "2 rows returned in 0.01 seconds". The bottom of the page includes copyright information for Oracle and the text "Oracle APEX 23.2.4".

2. Show the structure of departments the table. Select all the data from it.**QUERY:** SELECT * FROM DEPT;**OUTPUT:**


The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side of the header shows the user's name (220701010 abishe...), session ID (ab1307), and schema (WKSP_AB1307). The main workspace is titled 'SQL Commands' and contains the following SQL code:

```
1  select * from dept;
```

The results section displays the output of the query:

DEPT_ID	DEPT_NAME	JOB_ID	HIRE_DATE	EMP_NUM	LNAME
20	HR	123	07/13/2004	963	qwer
10	marketing	456	06/14/2004	856	yup
30	teamlead	789	12/12/2012	741	plmk

Below the table, it says "3 rows returned in 0.03 seconds". The bottom of the page includes copyright information for Oracle and the text "Oracle APEX 23.2.4".

The screenshot shows the Oracle APEX SQL Workshop interface. On the left, the Object Browser displays a tree view of database objects under the DEPARTMENT schema, with the DEPT table selected. The main panel shows the DEPT table structure with the following columns:

Column Name	Data Type	Nullable	Default	Primary Key	Comment	Identity
DEPT_ID	NUMBER(6,0)	N		1		
DEPT_NAME	VARCHAR2(20 BYTE)	Y				
JOB_ID	NUMBER	Y				
HIRE_DATE	DATE	Y				
EMP_NUM	NUMBER	Y				
LNAME	VARCHAR2(25 BYTE)	Y				

At the bottom of the main panel, it says "1 cells selected". The status bar at the bottom indicates the user is connected to "220701010@rajalakshmi.edu.in" and the session ID is "ab1507".

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

QUERY: SELECT emp_num, lname, job_code, hire_date FROM DEPT;

OUTPUT:

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

```
1 select emp_num,lname,job_id,hire_date from dept;
```

The Results tab displays the output of the query:

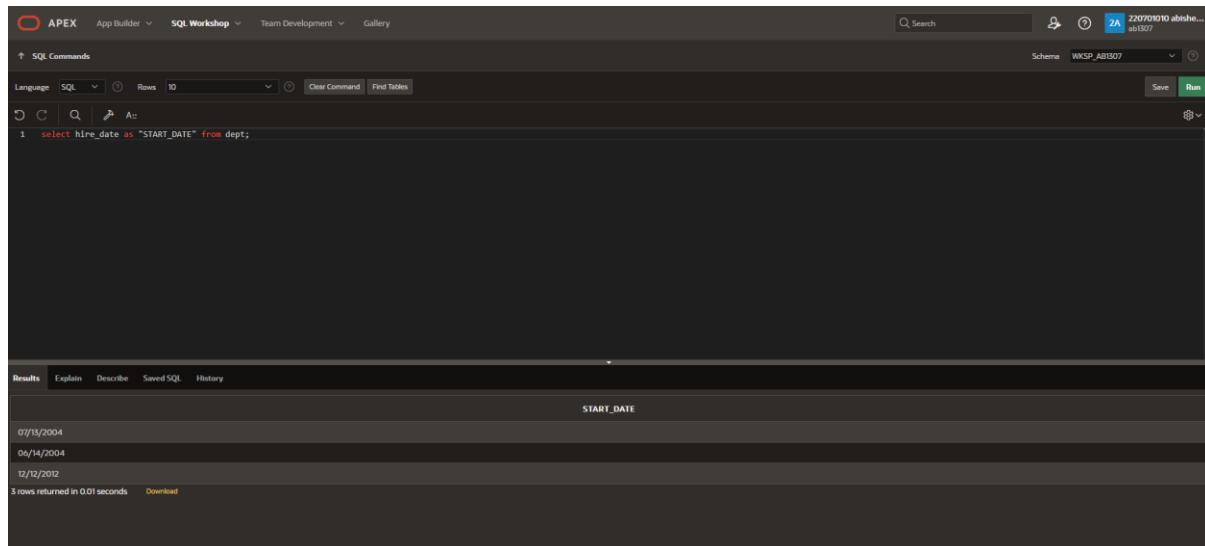
EMP_NUM	LNAME	JOB_ID	HIRE_DATE
963	qwer	123	07/13/2004
856	yup	456	06/14/2004
741	plmkn	789	12/02/2012

At the bottom of the results, it says "3 rows returned in 0.00 seconds".

4. Provide an alias STARTDATE for the hire date.

QUERY: SELECT hire_date as "START DATE" FROM DEPT;

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Commands section has a Language dropdown set to SQL, a Rows dropdown set to 10, and buttons for Clear Command and Find Tables. The main area contains the SQL command: `select hire_date as "START_DATE" from dept;`. The Results tab is selected, showing the output:

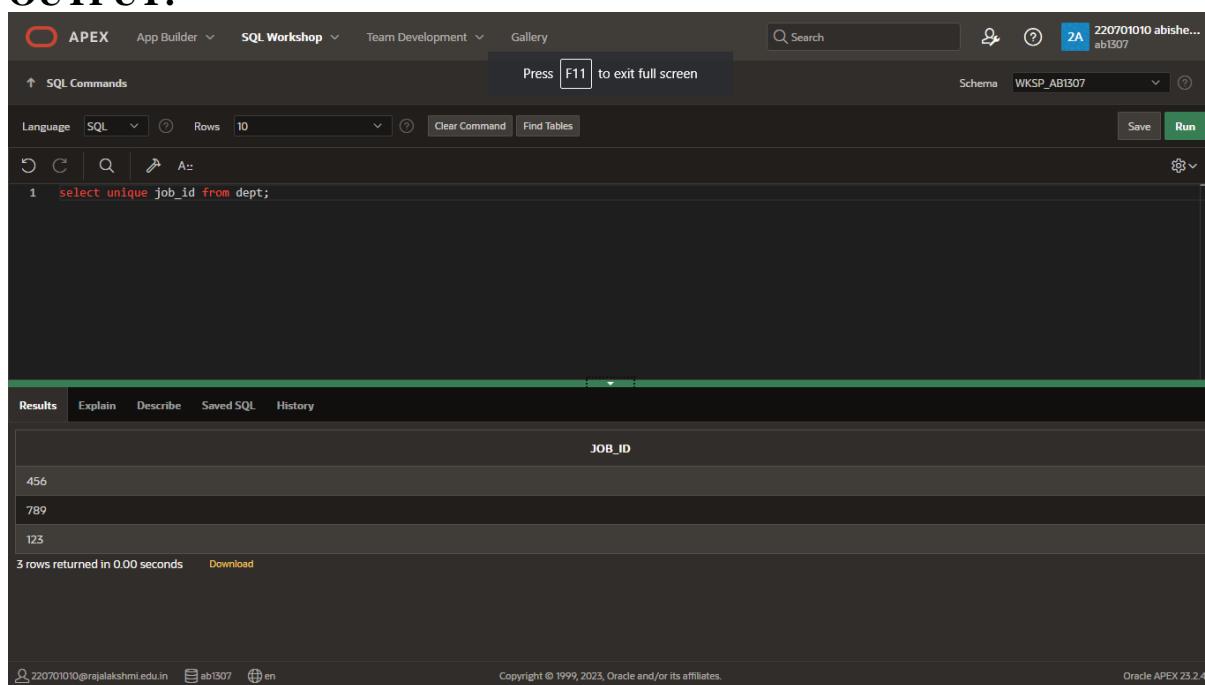
START_DATE
07/13/2004
06/14/2004
12/12/2012

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

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

QUERY: SELECT UNIQUE FROM DEPT;

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Commands section has a Language dropdown set to SQL, a Rows dropdown set to 10, and buttons for Clear Command and Find Tables. The main area contains the SQL command: `select unique job_id from dept;`. The Results tab is selected, showing the output:

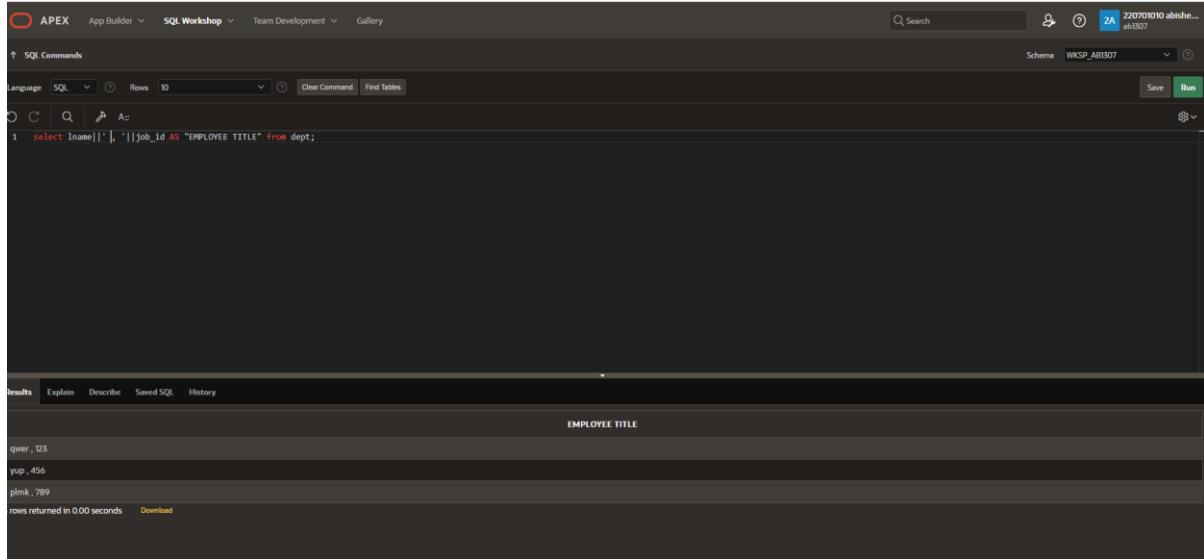
JOB_ID
456
789
123

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

At the bottom of the page, there are footer links: 220701010@rajalekshmi.edu.in, ab1307, en, Copyright © 1999, 2025, Oracle and/or its affiliates, and Oracle APEX 23.2.4.

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

QUERY: SELECT lname||','||job_id as "EMPLOYEE TITLE" FROM DEPT;
OUTPUT:



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

```
1 select lname||','||job_id AS "EMPLOYEE TITLE" from dept;
```

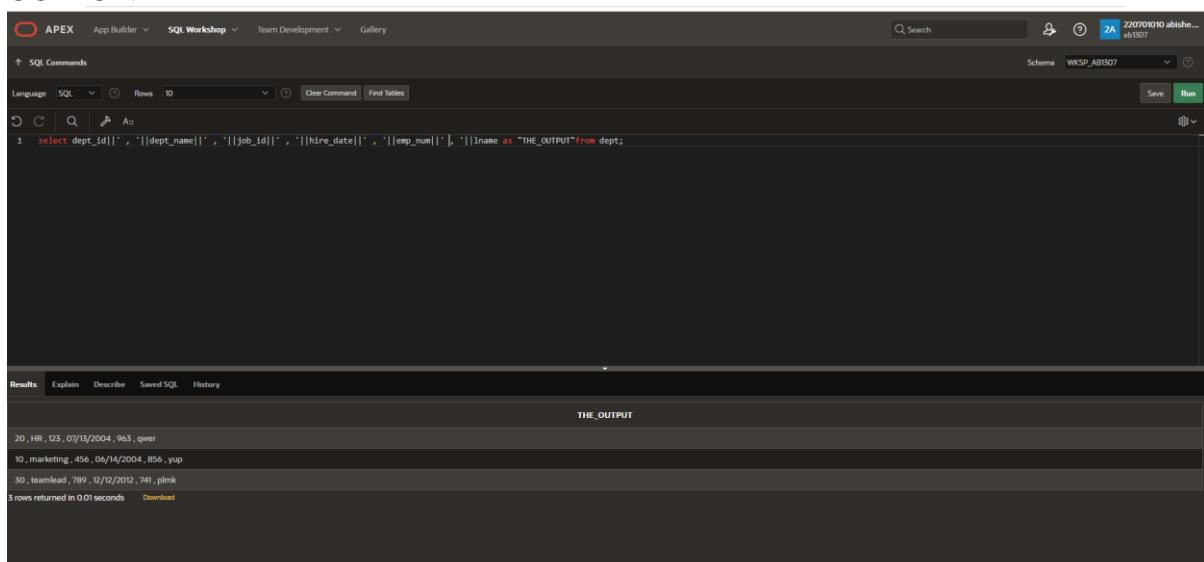
The results are displayed in a table with one column labeled "EMPLOYEE TITLE". The data returned is:

EMPLOYEE TITLE
qwer ,123
yup ,456
plmk ,789

Rows returned in 0.00 seconds.

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

QUERY: SELECT DEPT_ID||','||deptname||','||job_id||','||emp_num||','||hire_date||','||lname as "THE_OUTPUT";
OUTPUT:



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

```
1 select dept_id||','||dept_name||','||job_id||','||hire_date||','||emp_num||','||lname as "THE_OUTPUT"from dept;
```

The results are displayed in a table with one column labeled "THE_OUTPUT". The data returned is:

THE_OUTPUT
20 ,HR ,123 ,07/13/2004,963 ,qwer
10 ,marketing ,456 ,06/14/2004,856 ,yup
30 ,teamlead ,789 ,12/12/2012,741 ,plmk

3 rows returned in 0.01 seconds.

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

RESULT:

EXNO:5

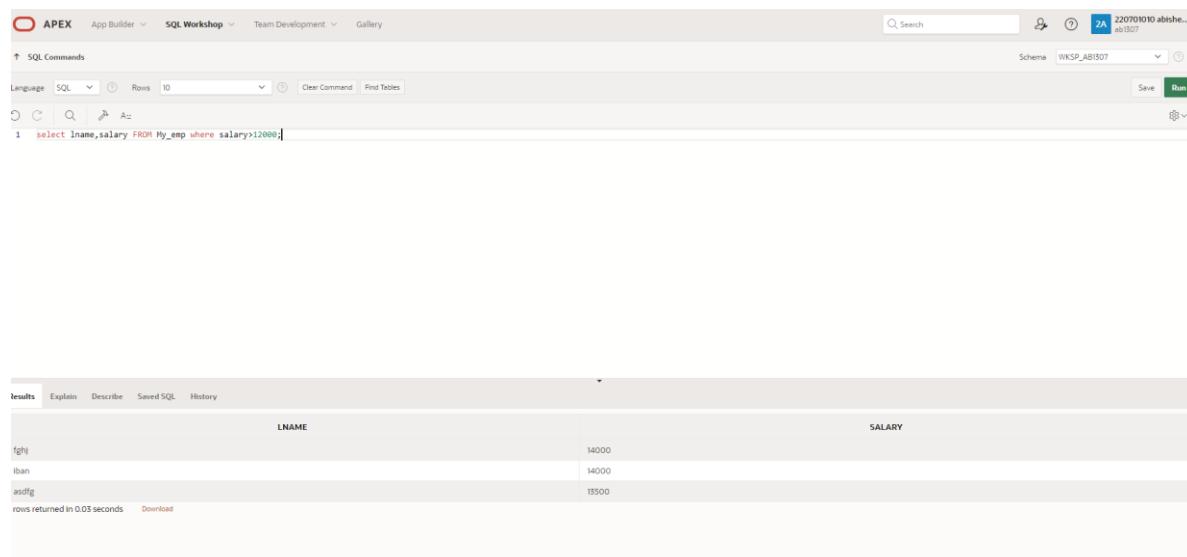
RESTRICTING AND SORTING DATA

DATE:

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

QUERY: select lname, salary from My_emp where salary > 12000;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is: `select lname,salary FROM My_emp where salary>12000;`. The results table displays three rows of data:

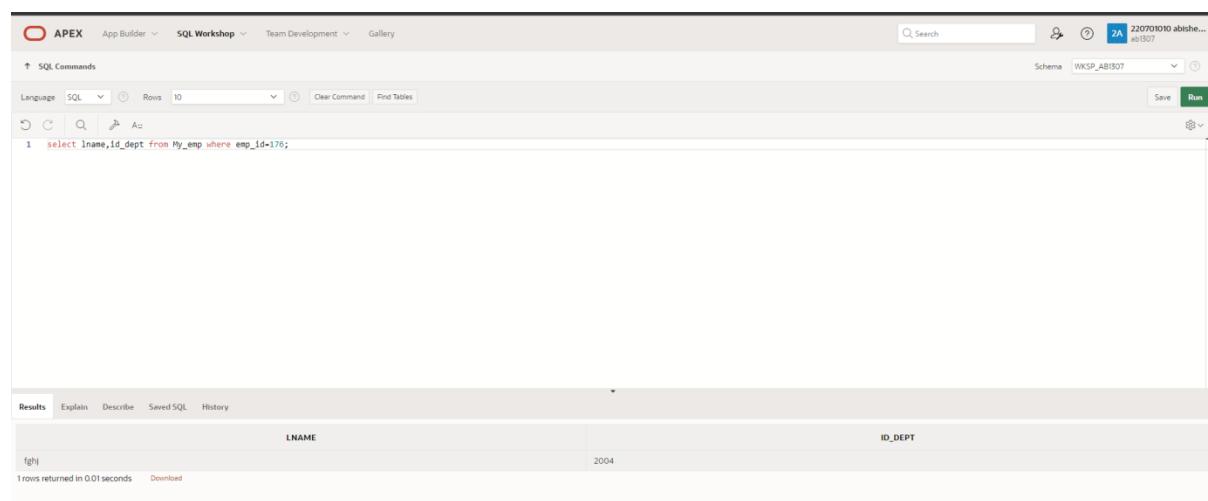
LNAME	SALARY
fghj	14000
iban	14000
asidfg	13500

Rows returned in 0.03 seconds. Download link is available.

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

QUERY: select lname, dept_id from employees where emp_id = 176;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is: `select lname,id_dept from My_emp where emp_id=176;`. The results table displays one row of data:

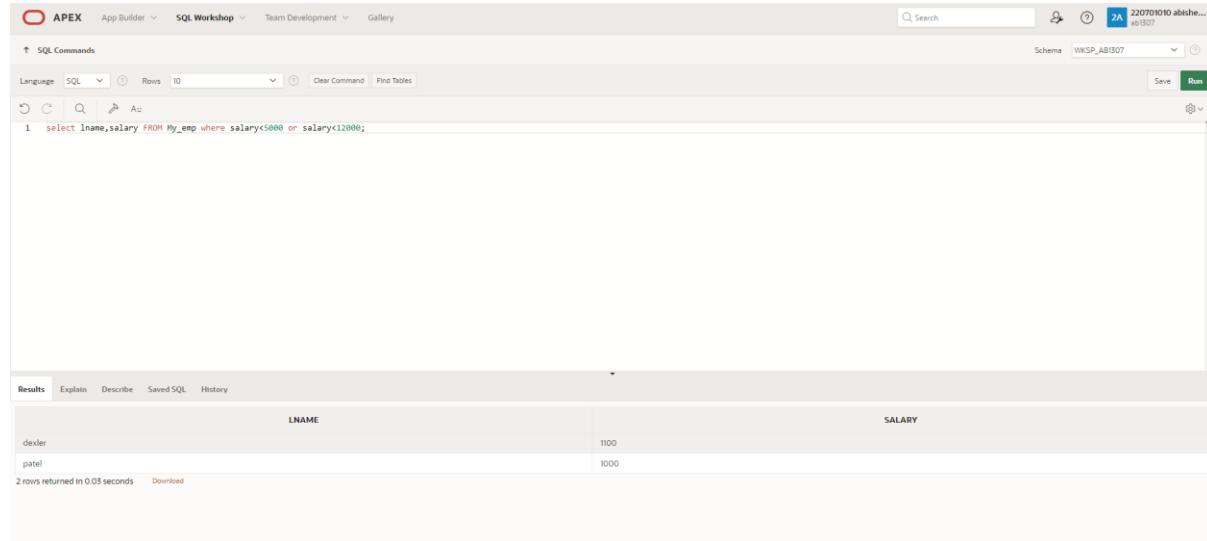
LNAME	ID_DEPT
fghj	2004

1 rows returned in 0.01 seconds. Download link is available.

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

QUERY: select lname, salary from My_emp where salary>12000 or salary<5000;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The SQL Commands pane contains the query: `select lname,salary FROM My_emp where salary<5000 or salary>12000;`. The Results pane displays the output:

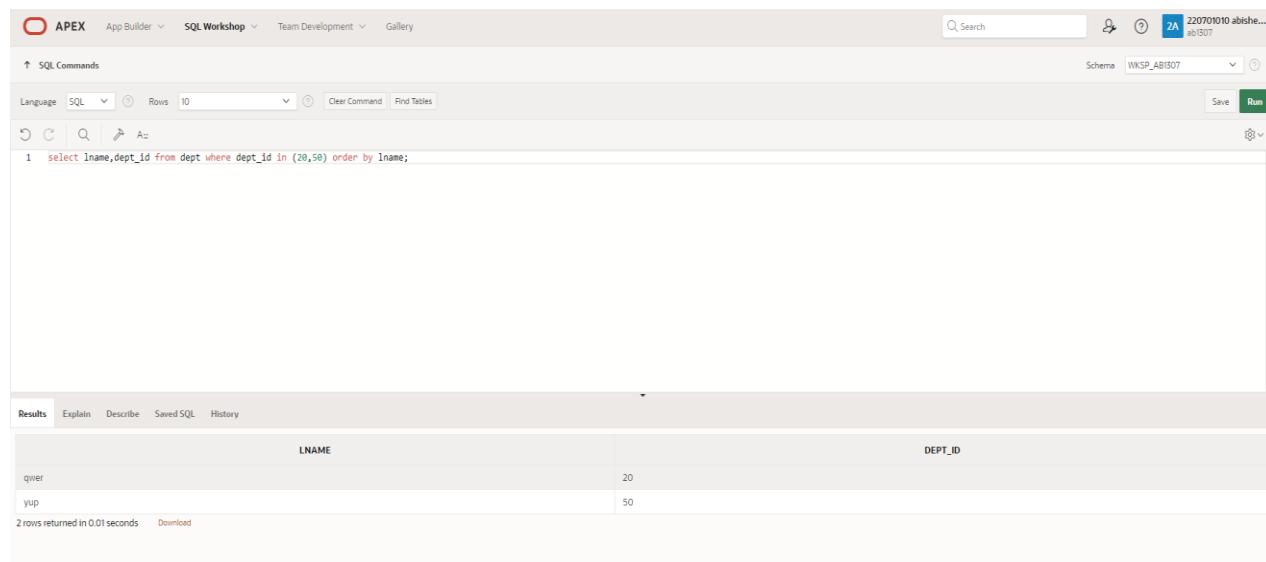
LNAME	SALARY
dexler	1100
patel	1000

2 rows returned in 0.03 seconds

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

QUERY: select lname, dept_id from My_emp where dept_id in (20,50) order by lname;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The SQL Commands pane contains the query: `select lname,dept_id from dept where dept_id in (20,50) order by lname;`. The Results pane displays the output:

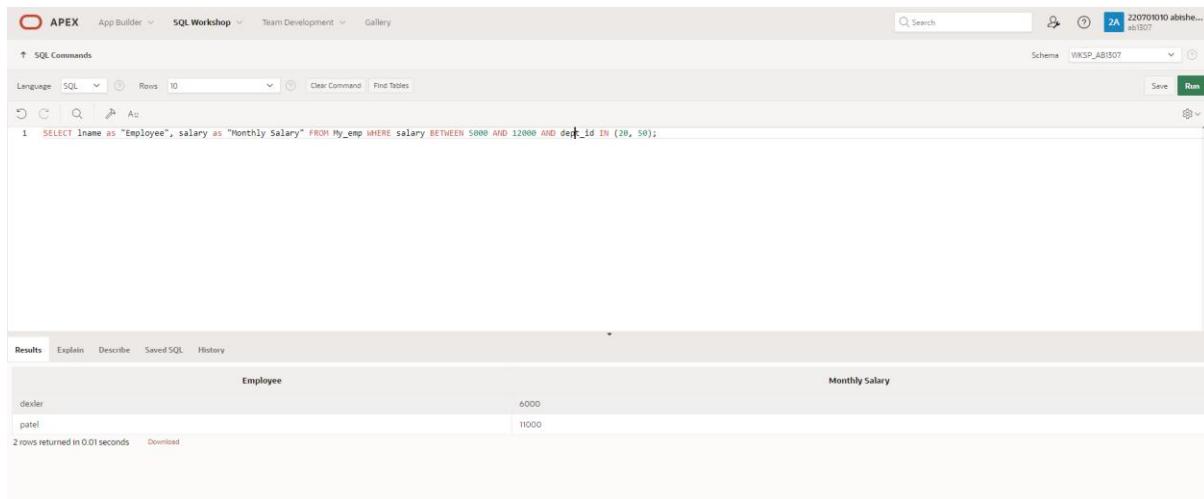
LNAME	DEPT_ID
qwer	20
yup	50

2 rows returned in 0.01 seconds

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

QUERY: select lname “Employee”, salary “Monthly Salary” from My_emp where (salary between 5000 and 12000) and dept_id in (20,50);

OUTPUT:



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

```
1 SELECT lname AS "Employee", salary AS "Monthly Salary" FROM My_emp WHERE salary BETWEEN 5000 AND 12000 AND dept_id IN (20, 50);
```

The Results tab displays the output:

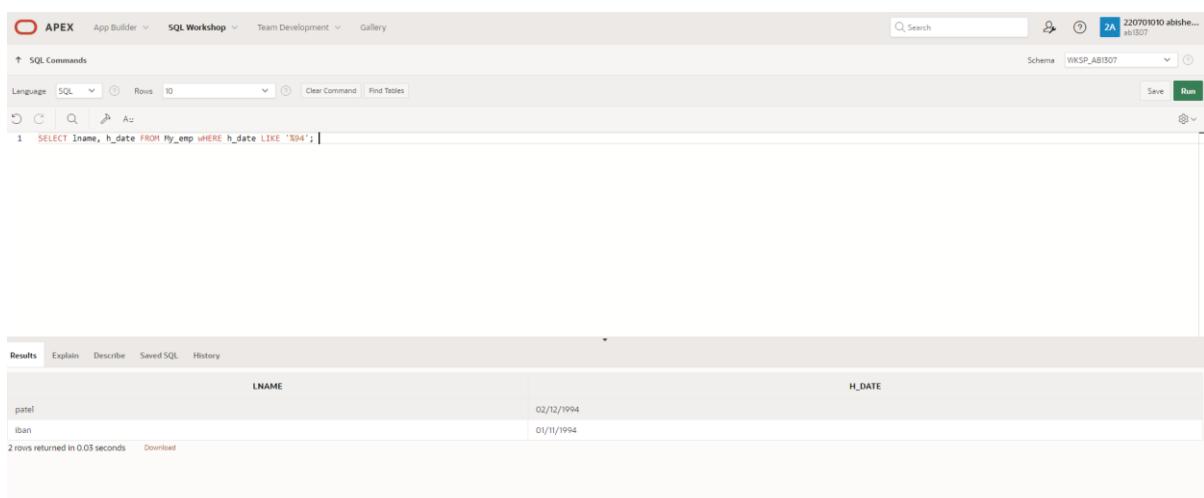
Employee	Monthly Salary
dexter	6000
patel	11000

2 rows returned in 0.01 seconds

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

QUERY: select lname, hire_date from My_emp where hire_date like ‘%94’;

OUTPUT:



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

```
1 SELECT lname, h_date FROM My_emp WHERE h_date LIKE '%94';
```

The Results tab displays the output:

LNAME	H_DATE
patel	02/12/1994
iban	01/11/1994

2 rows returned in 0.03 seconds

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

QUERY: select lname, job_title from employees where mg_id is null;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is: `SELECT lname, job_title FROM My_emp WHERE mg_id =0;`. The results table has two rows: dexter (marketing) and patel (tl).

LNAME	JOB_TITLE
dexter	marketing
patel	tl

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

QUERY: select lname, salary, commission from My_emp where commission is not null order by salary, commission;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is: `SELECT lname, salary,commission FROM My_emp WHERE commission IS NOT NULL ORDER BY salary DESC, commission DESC;`. The results table has three rows: iban (14000, 250), patel (11000, 300), and dexter (6000, 200).

LNAME	SALARY	COMMISSION
iban	14000	250
patel	11000	300
dexter	6000	200

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

QUERY: select lname from My_emp where lname like ‘_a%’;

OUTPUT:

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

```
1 SELECT lname FROM My_emp WHERE lname LIKE '_a%';
```

In the Results pane, the output is displayed as:

LNAME
iban

1 rows returned in 0.01 seconds

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

QUERY: select lname from My_emp where lname like ‘%a%’ and lname like ‘%e%’;

OUTPUT:

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

```
1 SELECT lname FROM My_emp WHERE lname LIKE '%a%' AND lname LIKE '%e%';
```

In the Results pane, the output is displayed as:

LNAME
patel

1 rows returned in 0.01 seconds

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

QUERY: select lname, job_title, salary from employees

where (job_title='sales' or job_title='clerk') and salary not in(2500,3500,7000);

OUTPUT:

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

```
1 SELECT lname, job_title, salary FROM My_emp WHERE job_title IN ('sales', 'stock clerk') AND salary NOT IN (2500,3500,7000);
```

The results table displays two rows:

LNAME	JOB_TITLE	SALARY
patel	sales	11000
iban	stock clerk	14000

2 rows returned in 0.01 seconds

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

QUERY: select lname, salary, commission from My_emp where commission=20;

OUTPUT:

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

```
1 select lname,salary,commission from My_emp where commission=20;
```

The results table displays two rows:

LNAME	SALARY	COMMISSION
dexter	6000	20
patel	11000	20

2 rows returned in 0.01 seconds

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

RESULT:

EX-6

SINGLE ROW FUNCTIONS

DATE:

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

QUERY: select sysdate as "DATE" from dual;

OUTPUT:

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

```
1 select sysdate as "DATE" from dual;
2
```

After running the query, the results window displays a single row with the value '03/12/2024' under the 'DATE' column. The results table has columns labeled 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'.

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

QUERY: select emp_id, lname, salary, salary+(salary*15.5/100) as "New Salary" from My_emp;

OUTPUT:

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

```
1 select emp_id, lname, salary, salary+(salary*15.5/100) as "New Salary"
2 from My_emp;
```

After running the query, the results window displays a table with four columns: 'EMP_ID', 'LNAME', 'SALARY', and 'New Salary'. The data is as follows:

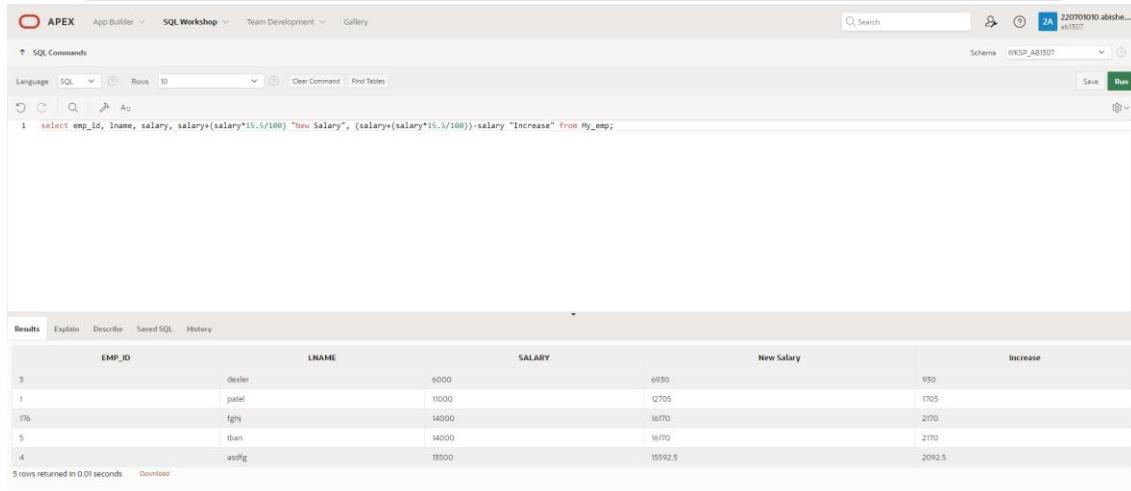
EMP_ID	LNAME	SALARY	New Salary
3	dexler	6000	6930
1	patel	11000	12705
176	fgj	14000	16170
5	iban	14000	16170
4	asdfg	13500	15592.5

The results table has columns labeled 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'.

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

QUERY: select emp_id, lname, salary, salary+(salary*15.5/100) as "New Salary", (salary+(salary*15.5/100))-salary as "Increses" from My_emp;

OUTPUT:



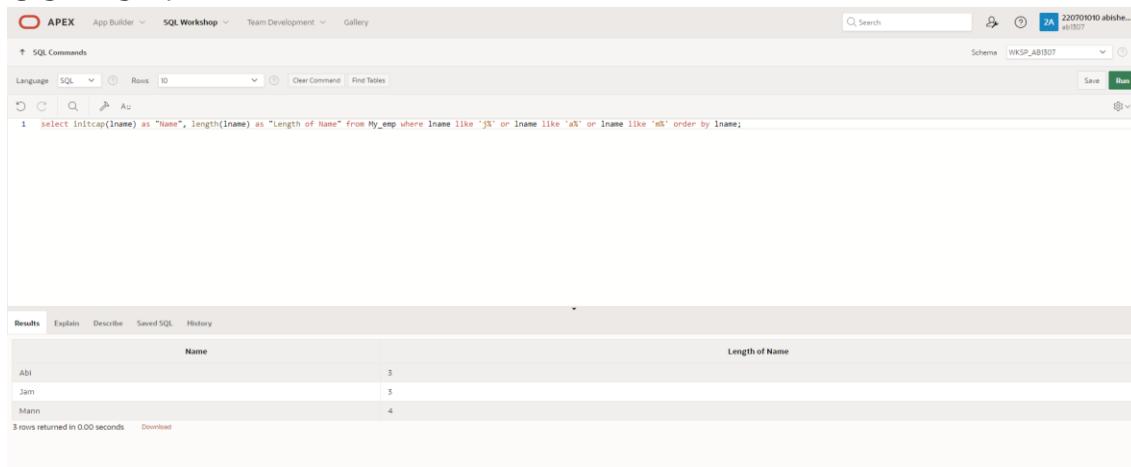
```
1  select emp_id, lname, salary, salary+(salary*15.5/100) "New Salary", (salary+(salary*15.5/100))-salary "Increase" from My_emp;
```

EMP_ID	LNAME	SALARY	New Salary	Increase
3	dexter	6000	6930	930
1	patel	11000	12705	1705
176	lghi	14000	16170	2170
5	iban	14000	16170	2170
4	andfg	15500	15592.5	2092.5

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

QUERY: select initcap(lname) as "Name", length(lname) as "Length of Name" from My_emp where lname like 'J%' or lname like 'A%' or lname like 'M%' order by lname;

OUTPUT:



```
1  select initcap(lname) as "Name", length(lname) as "Length of Name" from My_emp where lname like 'J%' or lname like 'A%' or lname like 'M%' order by lname;
```

Name	Length of Name
Abi	3
Jam	3
Mann	4

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

QUERY: select lname, round(months_between(sysdate,h_date),0) as "Months_worked" from My_emp order by 2;

OUTPUT:

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

```
1 select lname, round(months_between(sysdate,h_date),0) as "Months worked" from My_emp order by 2;
```

The results section displays the following data:

LNAME	Months worked
abt	155
mann	236
dexter	312
patel	361
jam	362

5 rows returned in 0.01 seconds

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

QUERY: select lname||' earns \$'||salary||' monthly but wants \$'||salary*3 as "Dream Salary" from My_emp;

OUTPUT:

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

```
1 select lname||' earns $'||salary||' monthly but wants $'||salary*3 as "Dream Salaries" from My_emp;
```

The results section displays the following data:

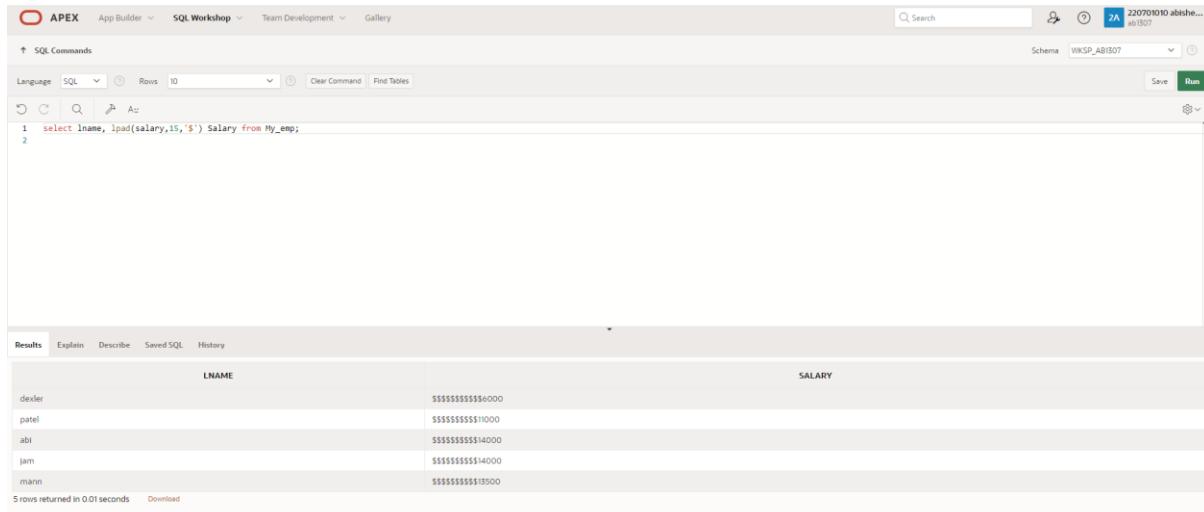
Dream Salaries
dealer earns \$6000 monthly but wants \$18000
patel earns \$10000 monthly but wants \$30000
abi earns \$14000 monthly but wants \$42000
jam earns \$14000 monthly but wants \$42000
mann earns \$18500 monthly but wants \$40500

5 rows returned in 0.01 seconds

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

QUERY: select lname, lpad(salary,15,'\$') Salary from My_emp;

OUTPUT:



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

```
1 select lname, lpad(salary,15,'$') Salary from My_emp;
2
```

The results section displays the following data:

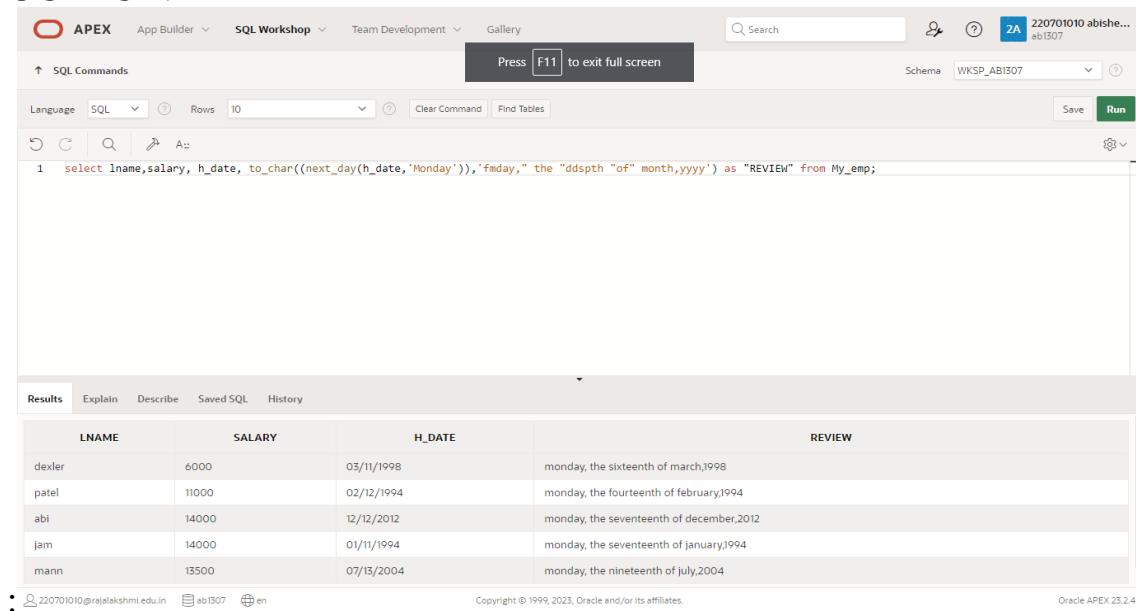
LNAME	SALARY
dexler	\$\$\$\$\$\$\$\$\$\$\$\$\$\$000
patel	\$\$\$\$\$\$\$\$\$\$11000
abi	\$\$\$\$\$\$\$\$\$\$14000
jam	\$\$\$\$\$\$\$\$\$\$14000
mann	\$\$\$\$\$\$\$\$\$\$13500

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

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

QUERY: select lname, h_date, to_char((next_day(h_date,'Monday')),'fmday," the "ddspth "of" month,yyyy') as "REVIEW" from My_emp;

OUTPUT:



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

```
1 select lname,salary, h_date, to_char((next_day(h_date,'Monday')),'fmday," the "ddspth "of" month,yyyy') as "REVIEW" from My_emp;
```

The results section displays the following data:

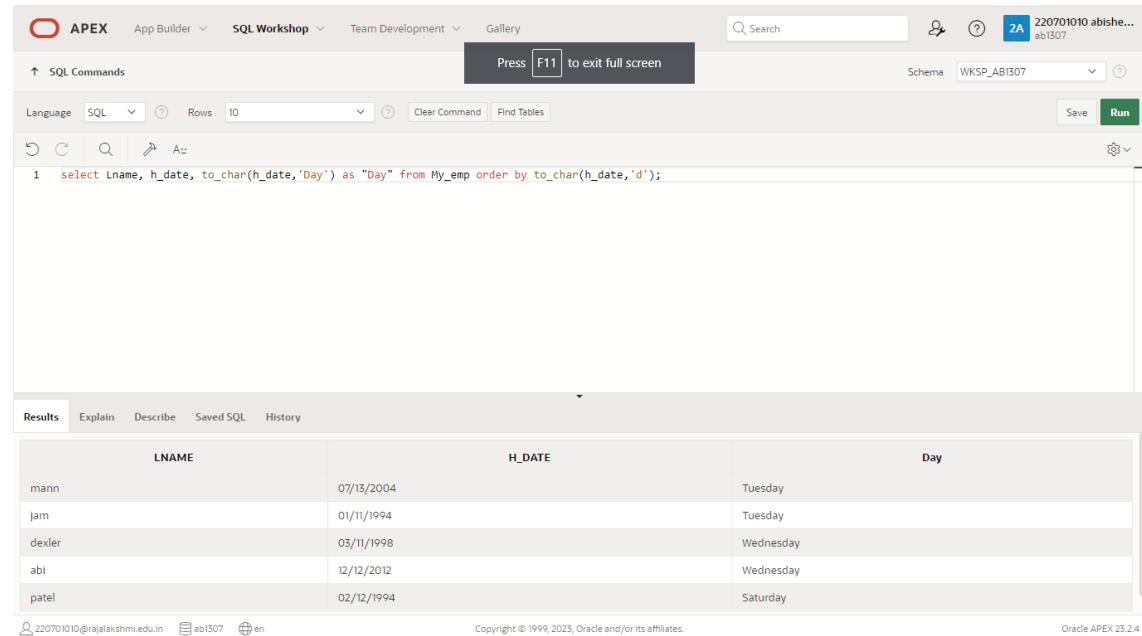
LNAME	SALARY	H_DATE	REVIEW
dexler	6000	03/11/1998	monday, the sixteenth of march,1998
patel	11000	02/12/1994	monday, the fourteenth of february,1994
abi	14000	12/12/2012	monday, the seventeenth of december,2012
jam	14000	01/11/1994	monday, the seventeenth of january,1994
mann	13500	07/13/2004	monday, the nineteenth of july,2004

At the bottom, it shows the URL 220701010@rejalakshmi.edu.in, session ab1307, and Oracle APEX 23.2.4.

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

QUERY: select Lname, h_date, to_char(h_date,'Day')as "Day" from My_emp
order by to_char(h_date-1,'d');

OUTPUT:



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

```
1 select Lname, h_date, to_char(h_date,'Day') as "Day" from My_emp order by to_char(h_date-1,'d');
```

The results section displays the following data:

LNAME	H_DATE	Day
mann	07/13/2004	Tuesday
jam	01/11/1994	Tuesday
debler	03/11/1998	Wednesday
abi	12/12/2012	Wednesday
patel	02/12/1994	Saturday

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

RESULT:

EXNO:7

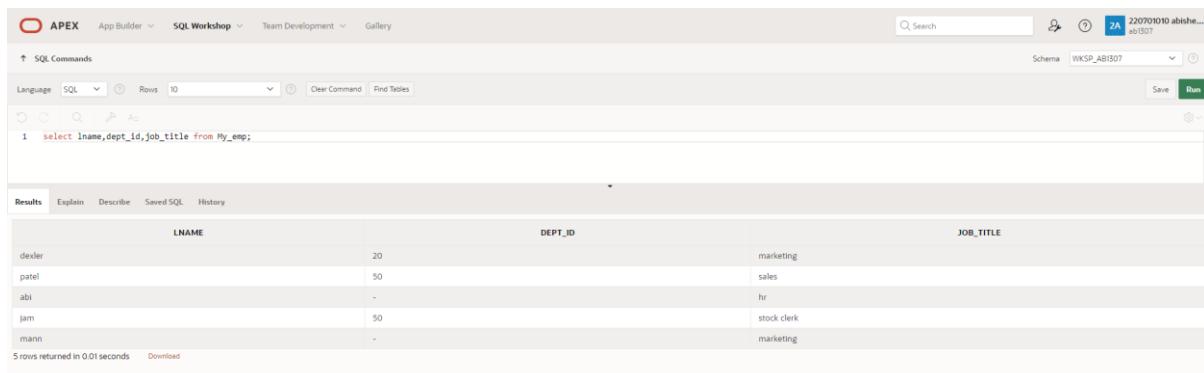
DISPLAYING DATA FROM MULTIPLE TABLES

DATE:

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

QUERY: select lname ,dept_id, dept_name from MY_emp;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is: `select lname,dept_id,job_title from My_emp;`. The results section displays the following data:

LNAME	DEPT_ID	JOB_TITLE
dexter	20	marketing
patel	50	sales
abi	-	hr
jam	50	stock clerk
mann	-	marketing

5 rows returned in 0.01 seconds Download

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 My_emp, dept WHERE My_emp.dept_id = dept.dept_id and My_emp.d_id=80;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is: `SELECT DISTINCT job_id, loc_id FROM My_emp, dept WHERE My_emp.dept_id = dept.dept_id and My_emp.d_id=80;`. The results section displays the following data:

JOB_ID	LOC_ID
123	965
456	6666

2 rows returned in 0.02 seconds Download

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

QUERY: SELECT My_emp.lname,My_emp.job_title, dept.loc_id, My_emp.city FROM My_emp, dept WHERE My_emp.dept_id = dept.dept_id AND My_emp.commission IS NOT NULL;

OUTPUT:



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

```
1 SELECT My_emp.lname,My_emp.job_title, dept.loc_id, My_emp.city FROM My_emp, dept WHERE My_emp.dept_id = dept.dept_id AND My_emp.commission IS NOT NULL;
2
```

The results pane displays the output of the query:

LNAME	JOB_TITLE	LOC_ID	CITY
dexler	marketing	965	chennai
patel	sales	6666	newyork
jam	stock clerk	6666	hyd

3 rows returned in 0.01 seconds Download

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

RESULT:

EXNO:8 AGGREGATING DATA USING GROUP FUNCTIONS DATE:

1. Find the highest, lowest, sum, and average salary of all employees. Label the columns

Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole number

QUERY: select max(salary) as "MAXIMUM", min(salary) as "MINIMUM",sum(salary) as "SSUM",round(avg(salary),2) as "average" from My_emp;

OUTPUT:



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

```
1 select max(salary) as "MAXIMUM", min(salary) as "MINIMUM",sum(salary) as "SSUM",round(avg(salary),2) as "average" from My_emp;
```

The results table displays the following data:

	MAXIMUM	MINIMUM	SSUM	average
14000	6000	58500	11700	

1 rows returned in 0.04 seconds

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

QUERY: select job_title,max(salary) as "MAXIMUM", min(salary) as "MINIMUM",sum(salary) as "SSUM",round(avg(salary),2) as "average" from My_emp group by job_title;

OUTPUT:



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

```
1 select job_title,max(salary) as "MAXIMUM", min(salary) as "MINIMUM",sum(salary) as "SSUM",round(avg(salary),2) as "average" from My_emp group by job_title;
```

The results table displays the following data:

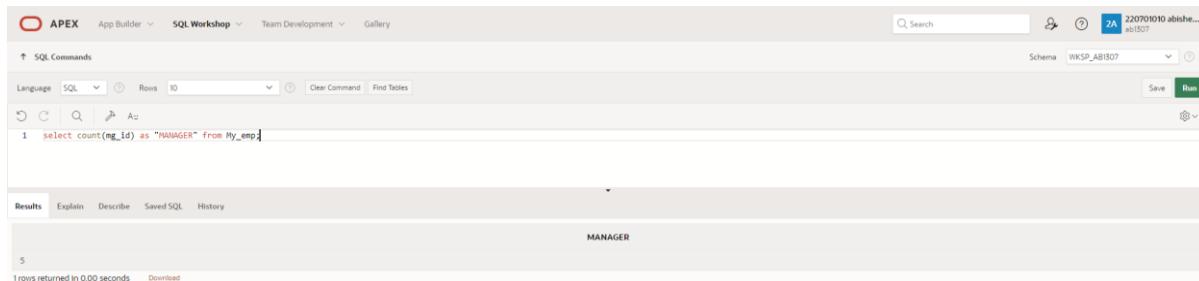
JOB_TITLE	MAXIMUM	MINIMUM	SSUM	average
s	13500	13500	13500	13500
hr	14000	14000	28000	14000
m	11000	6000	17000	8500

3 rows returned in 0.01 seconds

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

QUERY: select count(mg_id) as "MANAGER" from My_emp;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is: `select count(mg_id) as "MANAGER" from My_emp;`. The results pane displays a single row with the value 5 under the column labeled 'MANAGER'. The status bar at the bottom indicates '1 rows returned in 0.00 seconds'.

MANAGER
5

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

QUERY: select max(salary)-min(salary) as "DIFFERENCE"from My_emp;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The SQL command entered is: `select max(salary)-min(salary) as "DIFFERENCE"from My_emp;`. The results pane displays a single row with the value 8000 under the column labeled 'DIFFERENCE'. The status bar at the bottom indicates '1 rows returned in 0.01 seconds'.

DIFFERENCE
8000

EXNO:9

SUB QUERIES

DATE:

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

QUERY: select * from My_emp where lname='zlotkey';

OUTPUT:



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

```
1 select * from My_emp where lname='zlotkey';
```

The results window displays the following data:

EMP_ID	LNAME	FNAME	USERID	SALARY	COMMISSION	DEPT_ID	ID_DEPT	H_DATE	JOB_TITLE	MG_ID	INCREASE	D_ID	CITY
4	zlotkey	Ilkh	345	13500	-	80	80	07/13/1990	s	2	2092	-	-

1 rows returned in 0.05 seconds

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

QUERY: select emp_id, lname, salary from My_emp where salary > (select avg(salary) from My_emp) order by salary asc;

OUTPUT:



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

```
1 select emp_id, lname, salary from My_emp where salary > (select avg(salary) from My_emp) order by salary asc;
```

The results window displays the following data:

EMP_ID	LNAME	SALARY
4	zlotkey	13500
176	abi	14000
5	jam	14000

3 rows returned in 0.01 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.

QUERY: select emp_id,lname from My_emp where id_dept in(select id_dept from My_emp where lname like '%u%');

OUTPUT:

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

```
1 select emp_id,lname from My_emp where id_dept in(select id_dept from My_emp where lname like '%u%');
```

The results table shows two rows:

EMP_ID	LNAME
3	deuler
176	abu

2 rows returned in 0.00 seconds

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

QUERY: select emp_id,lname from my_emp where loc_id in(select loc_id from My_emp where loc_id=1700);

OUTPUT:

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

```
1 select emp_id,lname from my_emp where loc_id in(select loc_id from My_emp where loc_id=1700);
```

The results table shows two rows:

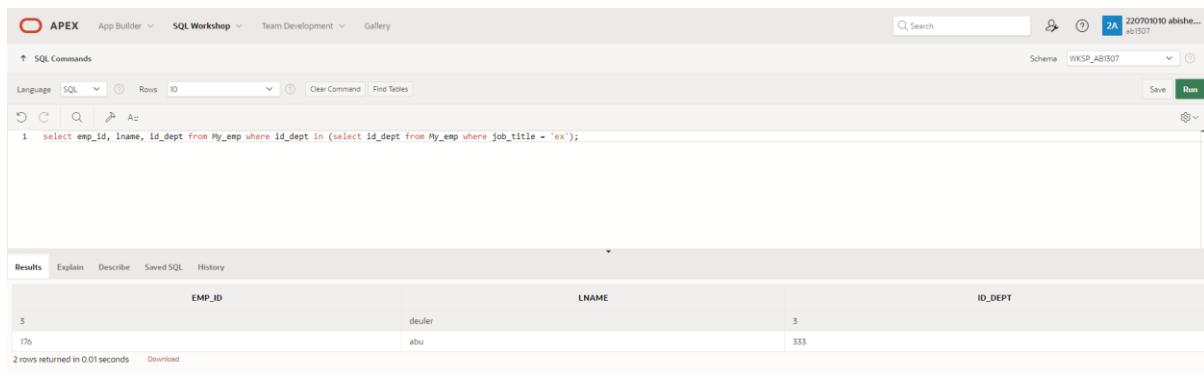
EMP_ID	LNAME
3	deuler
1	patel

2 rows returned in 0.00 seconds

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

QUERY: select emp_id, lname, id_dept from My_emp where id_dept in (select id_dept from My_emp where job_title = 'ex');

OUTPUT:



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

```
1 select emp_id, lname, id_dept from My_emp where id_dept in (select id_dept from My_emp where job_title = 'ex');
```

The results table displays two rows:

EMP_ID	LNAME	ID_DEPT
3	deuler	3
176	abu	333

2 rows returned in 0.01 seconds

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

QUERY: select emp_id, lname, salary from My_emp where salary > (select avg(salary) from my_emp) and id_dept in (select id_dept from My_emp where lname like '%u%');

OUTPUT:



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

```
1 select emp_id, lname, salary from My_emp where salary > (select avg(salary) from my_emp) and id_dept in (select id_dept from My_emp where lname like '%u%');
```

The results table displays one row:

EMP_ID	LNAME	SALARY
176	abu	14000

1 rows returned in 0.01 seconds

EXNO:10

USING THE SET OPERATORS

DATE:

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

QUERY: SELECT dept_id,job_id FROM dept MINUS SELECT dept_id,job_id FROM dept where job_id='clrk';

OUTPUT:

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

```
1 SELECT dept_id,job_id
2  FROM dept
3  MINUS
4  SELECT dept_id,job_id
5  FROM dept where job_id='clrk';
```

The results table displays two rows:

DEPT_ID	JOB_ID
333	mark
987	tl

2 rows returned in 0.00 seconds

- 2. The HR department needs a list of countries that have no departments located in them.**

Display the country ID and the name of the countries. Use set operators to create this report.

QUERY: SELECT coun_id,coun_name FROM dept MINUS SELECT coun_id,coun_name FROM dept WHERE id_dept is not NULL;

OUTPUT:

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

```
1 SELECT coun_id,coun_name FROM dept MINUS SELECT coun_id,coun_name FROM dept WHERE id_dept is not NULL;
2 
```

The results table displays two rows:

COUN_ID	COUN_NAME
1	india
2	sunspear

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

QUERY: select job_id,id_dept from dept where dept_id= 30 union all select job_id,id_dept from dept where dept_id=50 union all select job_id,id_dept from dept where dept_id= 20;

OUTPUT:

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

```
1 select job_id,id_dept from dept where dept_id= 30 union all select job_id,id_dept from dept where dept_id=50 union all select job_id,id_dept from dept where dept_id= 20;
```

The results are displayed in a table:

JOB_ID	ID_DEPT
clrk	22
clrk	-
clrk	-

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

QUERY: select job_id,hire_job from dept intersect select job_id,hire_job from dept where job_id=hire_job;

OUTPUT:

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

```
1 select job_id,hire_job from dept intersect select job_id,hire_job from dept where job_id=hire_job;
```

The results are displayed in a table:

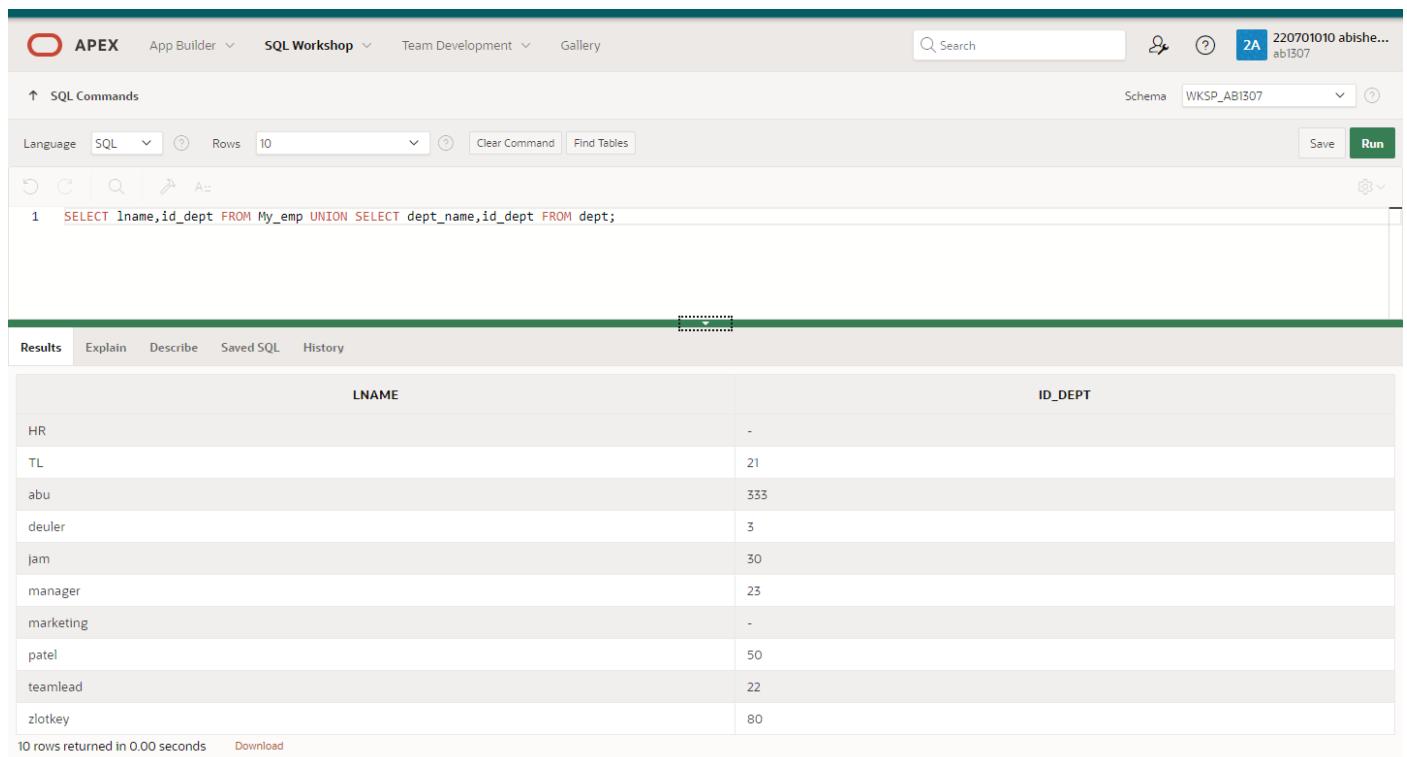
JOB_ID	HIRE_JOB
developer	developer
tl	tl

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

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

QUERY: SELECT lname,id_dept FROM My_emp UNION SELECT dept_name,id_dept FROM dept;

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 (2A 220701010 abishe... ab1307). The main area has tabs for SQL Commands, Explain, Describe, Saved SQL, and History. The SQL Commands tab is active, showing the following command:

```
1  SELECT lname,id_dept FROM My_emp UNION SELECT dept_name,id_dept FROM dept;
```

The results are displayed in a table with two columns: LNAME and ID_DEPT. The data is as follows:

LNAME	ID_DEPT
HR	-
TL	21
abu	333
deuler	3
jam	30
manager	23
marketing	-
patel	50
teamlead	22
zlotkey	80

At the bottom left, it says "10 rows returned in 0.00 seconds".

EXNO:11

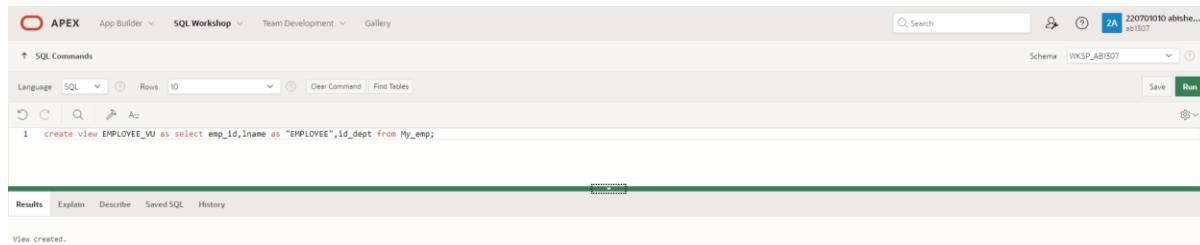
CREATING VIEWS

DATE:

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

Query: create view EMPLOYEE_VU as select emp_id, lname as "EMPLOYEE", id_dept from My_emp;

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The title bar shows 'APEX' and 'SQL Workshop'. The main area contains a SQL command window with the following content:

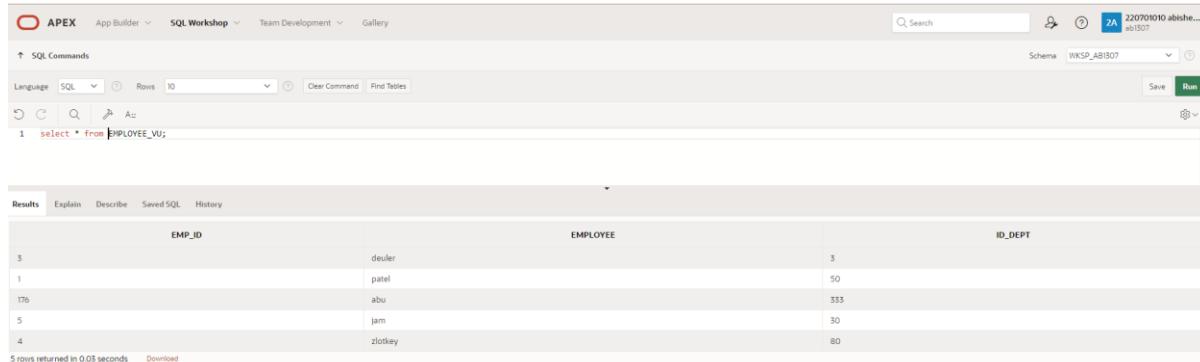
```
1 create view EMPLOYEE_VU as select emp_id, lname as "EMPLOYEE", id_dept from My_emp;
```

The results tab shows the message 'View created.'

2. Display the contents of the EMPLOYEES_VU view.

Query: select * from EMPLOYEE_VU;

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The title bar shows 'APEX' and 'SQL Workshop'. The main area contains a SQL command window with the following content:

```
1 select * from EMPLOYEE_VU;
```

The results tab displays a table with the following data:

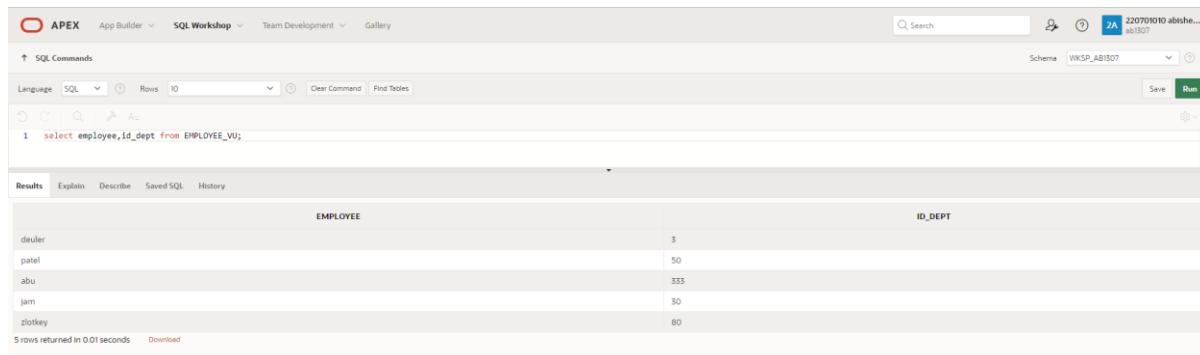
EMP_ID	EMPLOYEE	ID_DEPT
3	deuler	3
1	patel	50
176	abu	333
5	jam	30
4	zlotkey	80

5 rows returned in 0.03 seconds

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

Query: select employee,id_dept from EMPLOYEE_VU;

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The title bar shows 'APEX' and 'SQL Workshop'. The main area contains a SQL command window with the following content:

```
1 select employee,id_dept from EMPLOYEE_VU;
```

The results tab displays a table with the following data:

EMPLOYEE	ID_DEPT
deuler	3
patel	50
abu	333
jam	30
zlotkey	80

5 rows returned in 0.01 seconds

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

Query: create view DEPT50 as select lname as "EMPLOYEE",id_dept as "DEPTNO" from My_emp;

OUTPUT:

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

```
1 create view DEPT50 as select lname as "EMPLOYEE",id_dept as "DEPTNO" from My_emp;
```

The Results pane displays the message "View created." and "0.05 seconds".

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

Query: select * from DEPT50;

OUTPUT:

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

```
1 select * from DEPT50;
```

The Results pane displays the contents of the DEPT50 view:

EMPLOYEE	DEPTNO
deuler	3
patel	50
abu	333
jam	30
zlotkey	80

5 rows returned in 0.01 seconds

6. Attempt to reassign Matos to department 80.

Query: UPDATE DEPT50 SET DEPTNO = 30 WHERE EMPLOYEE = 'Matos';

OUTPUT:

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

```
1 UPDATE DEPT50 SET DEPTNO = 30 WHERE EMPLOYEE = 'Matos';
```

The Results pane displays the message "0 row(s) updated." and "0.01 seconds".

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

Query: create view SALARY_VU as select lname as "employee",job_title as "DEPARTMENT", salary from My_emp;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, the tabs 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery' are visible. On the right side of the header, there is a search bar, a user icon, and session information '2207091010 abishe...'. Underneath the header, the schema 'WKSP_AB1507' is selected. The main area is titled 'SQL Commands'. It contains a toolbar with icons for Undo, Redo, Find, Replace, and Save. A dropdown menu 'Language' is set to 'SQL'. The 'Rows' dropdown is set to '10'. There are buttons for 'Clear Command' and 'Find Tables'. The SQL command 'create view SALARY_VU as select lname as "employee",job_title as "DEPARTMENT", salary from My_emp.' is entered in the text area. Below the command, the results tab is selected, showing the message 'View created.' and a time of '0.05 seconds'.

EXNO:12

INTRO TO CONSTRAINTS

DATE:

NOT NULL AND UNIQUE CONSTRAINTS

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

QUERY: 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));

OUTPUT:



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

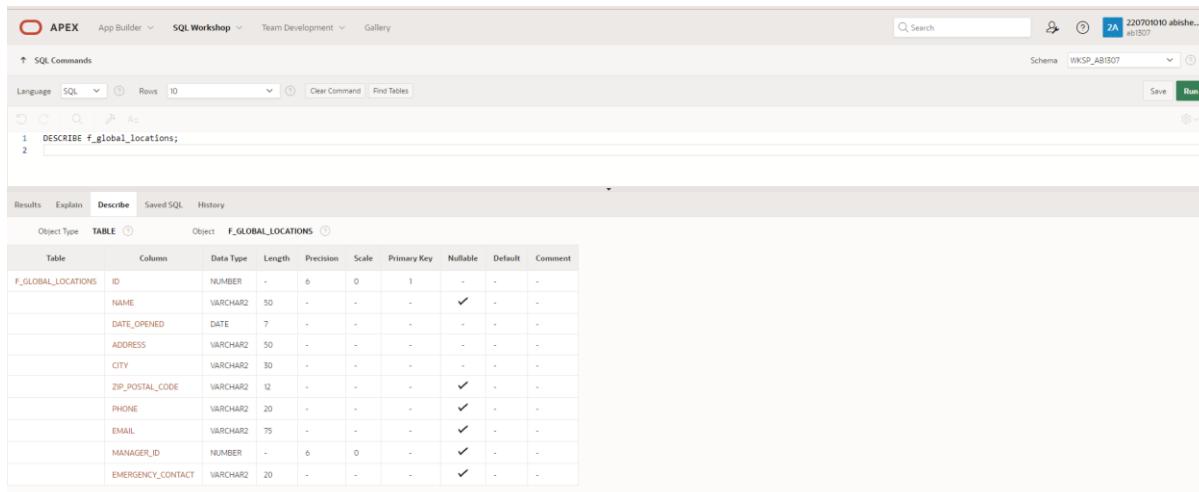
```
1 CREATE TABLE f_global_locations
2 ( `id` NUMBER(6,0) CONSTRAINT f_gln_id_pk PRIMARY KEY ,
3   name VARCHAR2(50),
4   date_opened DATE CONSTRAINT f_gln_dt_opened_nn NOT NULL ENABLE,
5   address VARCHAR2(50) CONSTRAINT f_gln_add_nn NOT NULL ENABLE,
6   city VARCHAR2(30) CONSTRAINT f_gln_city_nn NOT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

The results pane shows the message "Table created." and a execution time of "0.10 seconds".

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

QUERY: DESCRIBE f_global_locations;

OUTPUT:



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

```
1 DESCRIBE f_global_locations;
2
```

The results pane shows the table structure:

Object Type	TABLE	Object	F_GLOBAL_LOCATIONS						
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
F_GLOBAL_LOCATIONS	ID	NUMBER	-	6	0	1	-	-	-
	NAME	VARCHAR2	50	-	-	-	✓	-	-
	DATE_OPENED	DATE	7	-	-	-	-	-	-
	ADDRESS	VARCHAR2	50	-	-	-	-	-	-
	CITY	VARCHAR2	30	-	-	-	-	-	-
	ZIP_POSTAL_CODE	VARCHAR2	12	-	-	-	✓	-	-
	PHONE	VARCHAR2	20	-	-	-	✓	-	-
	EMAIL	VARCHAR2	75	-	-	-	✓	-	-
	MANAGER_ID	NUMBER	-	6	0	-	✓	-	-
	EMERGENCY_CONTACT	VARCHAR2	20	-	-	-	✓	-	-

3. 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.

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

QUERY: CREATE TABLE animals

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area contains the SQL command for creating the 'animals' table. Below the command, the 'Results' tab is active, showing the message 'Table created.' and a timestamp of '0.09 seconds'. The schema 'WKSP_AB1507' is visible in the top right corner.

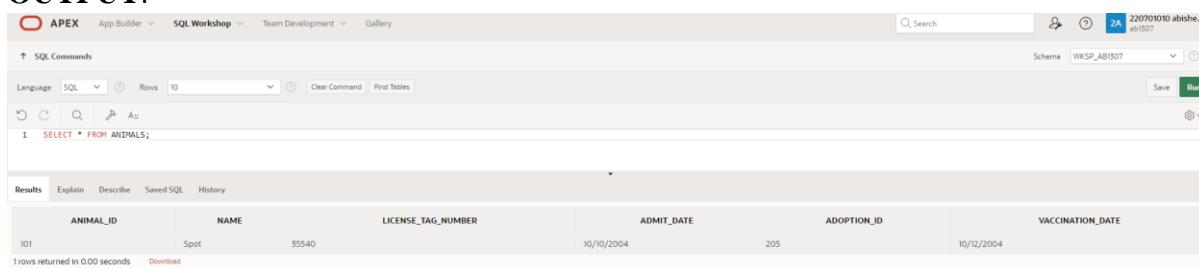
```
1 CREATE TABLE animals  
2 (animal_id NUMBER(6,0) CONSTRAINT anl_id_pk PRIMARY KEY ,  
3 name VARCHAR2(25),  
4 license_tag_number NUMBER(10,0) CONSTRAINT anl_l_tag_num_uk UNIQUE,  
5 admit_date DATE CONSTRAINT anl_adt_dat_nn NOT NULL ENABLE,  
6 adoption_id NUMBER(5,0),  
7 vaccination_date DATE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

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

QUERY: INSERT INTO ANIMALS (ANIMAL_ID, NAME, LICENSE_TAG_NUMBER, ADMIT_DATE, ADOPTION_ID, VACCINATION_DATE)

```
VALUES (101, 'Spot', 35540, '10-10-2004', 205, '10-12-2004');
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area contains the SQL command for selecting all columns from the 'ANIMALS' table. Below the command, the 'Results' tab is active, displaying a single row of data: ANIMAL_ID 101, NAME Spot, LICENSE_TAG_NUMBER 35540, ADMIT_DATE 10/10/2004, ADOPTION_ID 205, and VACCINATION_DATE 10/12/2004. The schema 'WKSP_AB1507' is visible in the top right corner.

```
1 SELECT * FROM ANIMALS;
```

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

EXNO:13

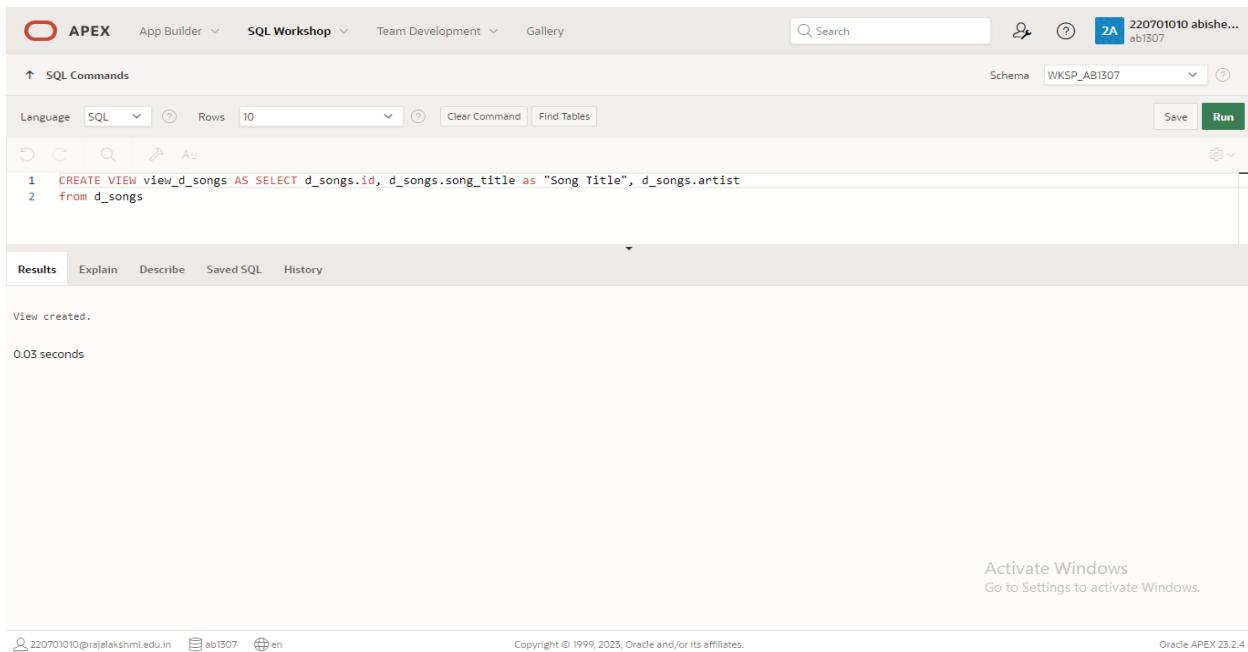
CREATING VIEWS

DATE:

1.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.

QUERY: CREATE VIEW view_d_songs AS SELECT id, title as "Song Title", artist from d_songs;

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 icon, a search bar, and a session identifier '220701010 abishe... ab1307'. Below the toolbar, the schema is set to 'WKSP_AB1307'. The main area is titled 'SQL Commands' with tabs for 'Language' (set to 'SQL'), 'Rows' (set to 10), and buttons for 'Clear Command' and 'Find Tables'. A 'Save' and 'Run' button are also present. The SQL command entered is:

```
1 CREATE VIEW view_d_songs AS SELECT d_songs.id, d_songs.song_title as "Song Title", d_songs.artist
2   from d_songs
```

Below the command, the results tab shows the message 'View created.' and a execution time of '0.03 seconds'. The bottom right corner features an 'Activate Windows' watermark with the text 'Go to Settings to activate Windows.'

2. SELECT * FROM view_d_songs. What was returned?

QUERY: select * from d_songs;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to 'WKSP_ABI307'. The query entered is 'SELECT * FROM view_d_songs;'. The results table has three columns: ID, Song Title, and ARTIST. One row is returned, with ID 1, Song Title 'abc', and ARTIST 'xyz'. The message '1 rows returned in 0.00 seconds' is displayed at the bottom.

ID	Song Title	ARTIST
1	abc	xyz

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

QUERY: 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;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The schema is set to 'WKSP_ABI307'. The query entered is '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;'. The message 'View created.' is displayed at the bottom. The execution time is 0.03 seconds.

4. 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.

QUERY: CREATE OR REPLACE VIEW view_d_events_pkgs AS SELECT d_name as "Name of Event", TO_CHAR(d_date, 'dd-Month-yyyy') as "Event date", theme as "Theme description" FROM d_events;

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. A search bar and user information are also present. The main area is titled 'SQL Commands' and shows the following SQL code:

```
1 CREATE OR REPLACE VIEW view_d_events_pkgs AS
2   SELECT d_name as "Name of Event", TO_CHAR(d_date, 'dd-Month-yyyy') as "Event date", theme as "Theme description"
3   FROM d_events;
```

Below the code, there are tabs for Results, Explain, Describe, Saved SQL, and History. The 'Results' tab is selected. The output section displays the message 'View created.' and '0.03 seconds'. In the bottom right corner, there is an 'Activate Windows' watermark with the text 'Go to Settings to activate Windows.'

EXNO:14

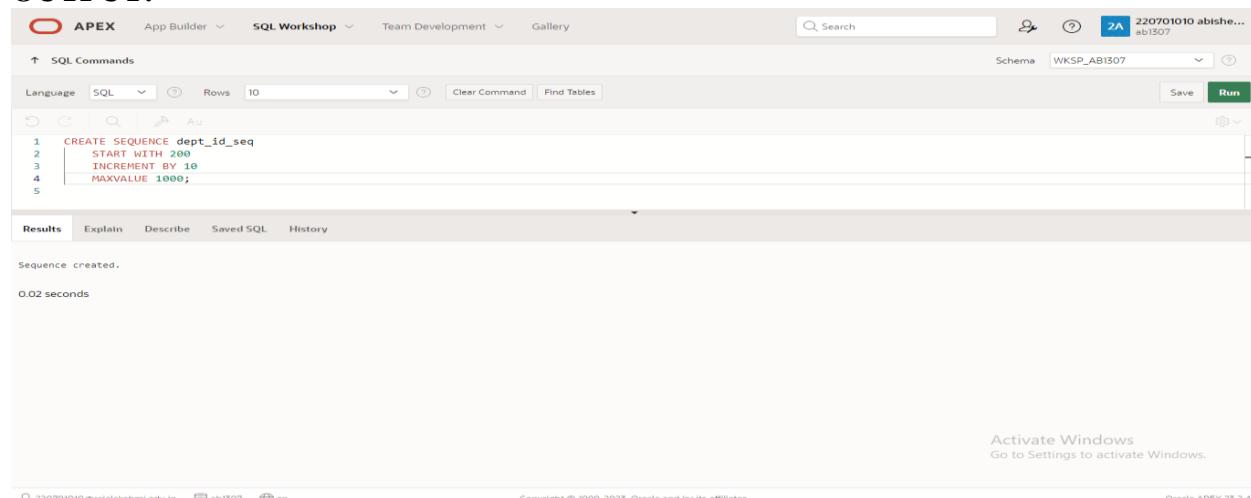
OTHER DATABASE OBJECTS

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

```
1 CREATE SEQUENCE dept_id_seq
2   START WITH 200
3   INCREMENT BY 10
4   MAXVALUE 1000;
5
```

Below the command, the results show:

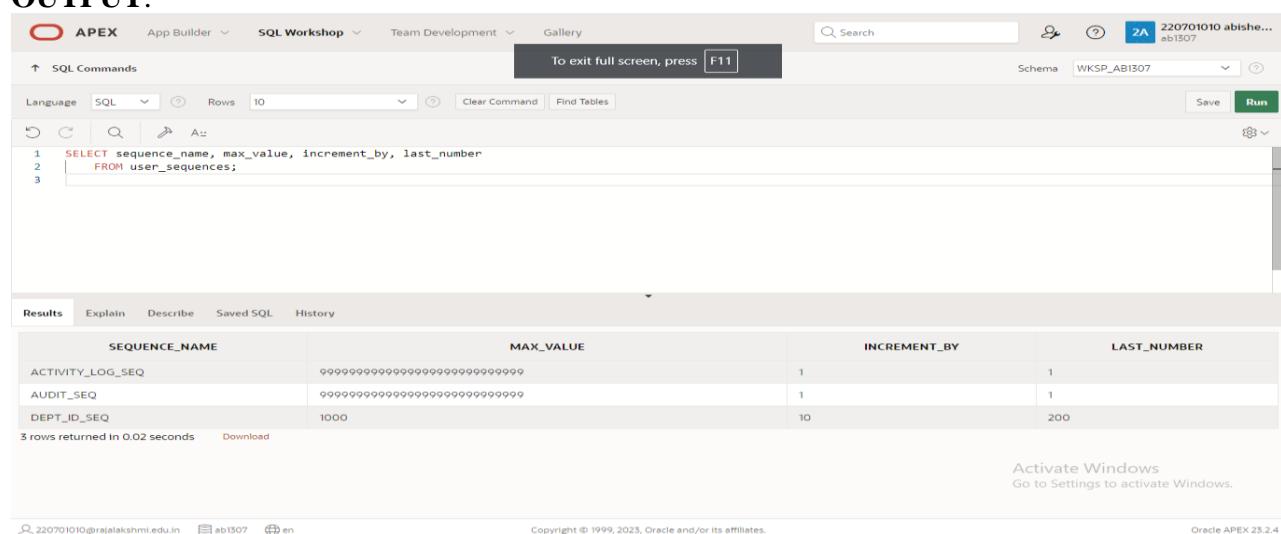
```
Sequence created.
```

At the bottom right, there is a message: "Activate Windows Go to Settings to activate Windows."

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

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

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

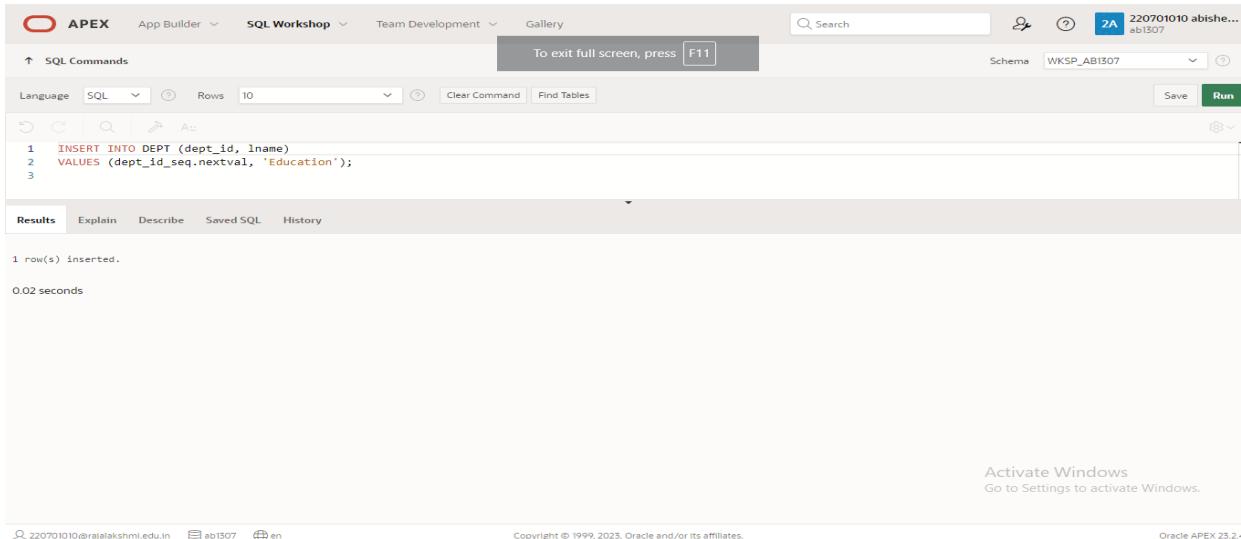
SEQUENCE_NAME	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
ACTIVITY_LOG_SEQ	99999999999999999999999999999999	1	1
AUDIT_SEQ	99999999999999999999999999999999	1	1
DEPT_ID_SEQ	1000	10	200

At the bottom right, there is a message: "Activate Windows Go to Settings to activate Windows."

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

OUTPUT:



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

```
1 INSERT INTO DEPT (dept_id, lname)
2   VALUES (dept_id_seq.nextval, 'Education');
3
```

The results section shows the output:

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

Execution time: 0.02 seconds.

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

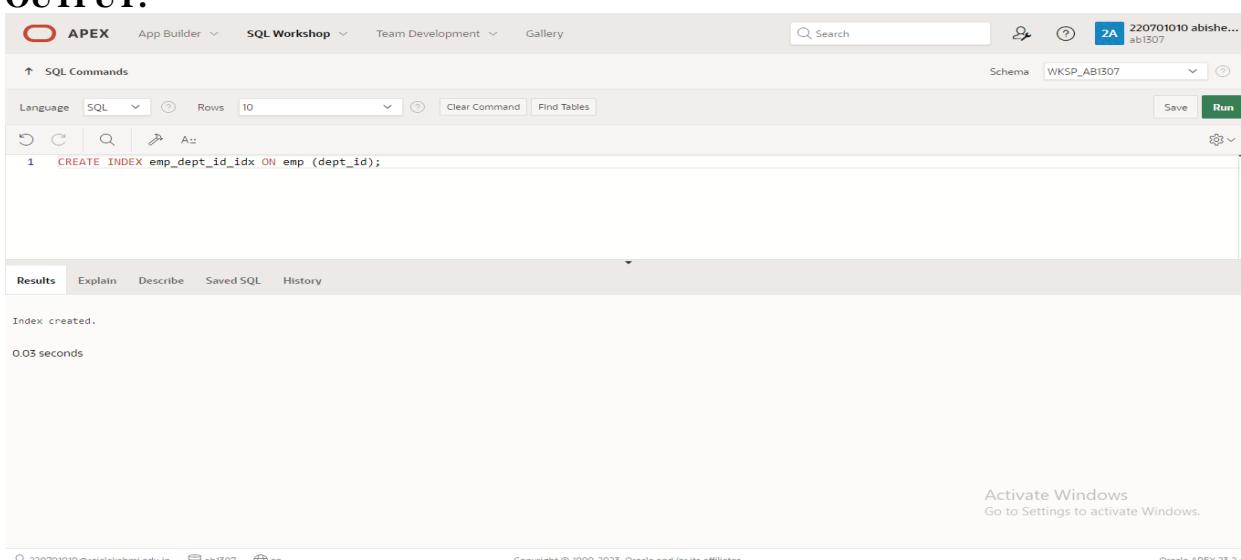
Activate Windows
Go to Settings to activate Windows.

Oracle APEX 23.2.4

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

QUERY: `CREATE INDEX emp_dept_id_idx ON emp (dept_id);`

OUTPUT:



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

```
1 CREATE INDEX emp_dept_id_idx ON emp (dept_id);
```

The results section shows the output:

```
Index created.
```

Execution time: 0.03 seconds.

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

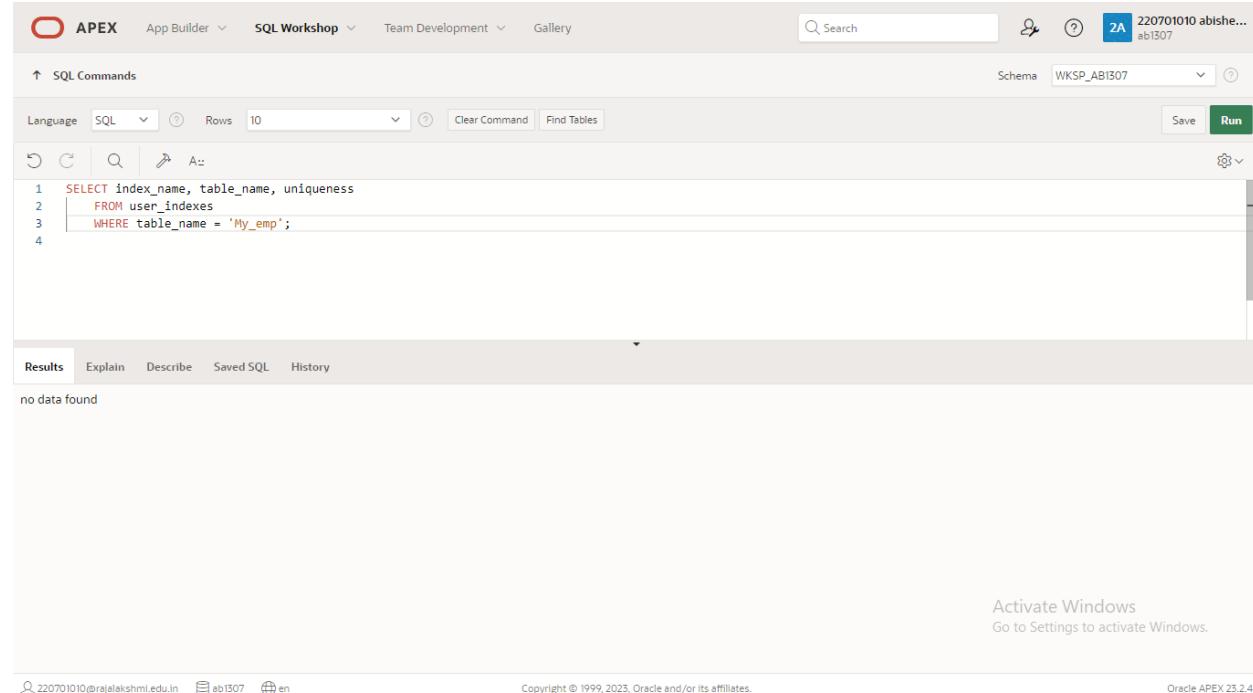
Activate Windows
Go to Settings to activate Windows.

Oracle APEX 23.2.4

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 = 'EMP';

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's name (220701010 abishe... ab1307) and a search bar. Below the header is a toolbar with icons for Undo, Redo, Search, and Run. The main area is titled "SQL Commands" and contains a code editor with the following SQL query:

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

Below the code editor, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, showing the message "no data found". At the bottom of the page, there are footer links for "Activate Windows", "Copyright © 1999, 2023, Oracle and/or its affiliates.", and "Oracle APEX 23.2.4".

Exno:15

CONTROLLING USER ACCESS

date:

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

Ans: The CREATE SESSION system privilege

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

Ans : The CREATE TABLE privilege

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

Ans: 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?

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

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

Ans: 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.

Ans: Team 2 executes the GRANT statement.

```
GRANT select  
ON departments  
TO <user1>;
```

Team 1 executes the GRANT statement.

```
GRANT select  
ON departments  
TO <user2>;
```

WHERE user1 is the name of team 1 and user2 is the name of team 2.

7. Query all the rows in your DEPARTMENTS table.

Ans: 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.

Ans: 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. Create a synonym for the other team's DEPARTMENTS table.

Ans: Team 1 creates a synonym named team2.

```
CREATE SYNONYM team2
FOR <user2>.DEPARTMENTS;
```

Team 2 creates a synonym named team1.

```
CREATE SYNONYM team1
FOR <user1>. DEPARTMENTS;
```

10. Query all the rows in the other team's DEPARTMENTS table by using your synonym.

Ans: Team 1 executes this SELECT statement.

```
SELECT *
FROM team2;
```

Team 2 executes this SELECT statement.

```
SELECT *
FROM team1;
```

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

Ans: SELECT table_name FROM user_tables;

12. Query the ALL_TABLES data dictionary view to see information about all the tables that you can access. Exclude tables that you own.

Ans: SELECT table_name, owner FROM all_tables
 WHERE owner <> <your account>;

13. Revoke the SELECT privilege from the other team.

Ans: Team 1 revokes the privilege.

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

Team 2 revokes the privilege.

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

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

Team 1 executes this INSERT statement.

Ans:

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

Team 2 executes this INSERT statement.

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

EXNO:16

PL/SQL

DATE:

CONTROL STRUCTURES

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, a PL/SQL block is written and executed. The output pane displays the result of the DBMS_OUTPUT.PUT_LINE statement.

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

Incentive = 2400
Statement processed.
0.01 seconds

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

QUERY:

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

/

OUTPUT:



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

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

Below the command window, the 'Results' tab is active, showing the output:

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

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

QUERY:

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

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

OUTPUT:

```
APEX SQL Workshop Team Development Gallery
SQL Commands Schema WKSP_AB1507 Save Run
Language: SQL Rows: 10 Clear Command Find Tables
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 emp_id = 122;
16     DBMS_OUTPUT.PUT_LINE
17     ('Before invoking procedure, salary_of_emp: ' || salary_of_emp);
18     approx_salary (100, salary_of_emp, 1000);
19     DBMS_OUTPUT.PUT_LINE
20     ('After invoking procedure, salary_of_emp: ' || salary_of_emp);
21 END;
22 /
23

Results Explain Describe Saved SQL History
Before invoking procedure, salary_of_emp: 10000
After invoking procedure, salary_of_emp: 11000
Statement processed.

0.01 seconds
```

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side of the header shows the session ID 220709010 abishe... and schema WKSP_AB1507. The main area is titled 'SQL Commands' with tabs for Language (set to SQL), Rows (set to 10), Clear Command, Find Tables, Save, and Run. The code area contains the PL/SQL procedure definition. Below the code, the results tab is selected, showing the message 'Procedure created.' and a execution time of '0.03 seconds'.

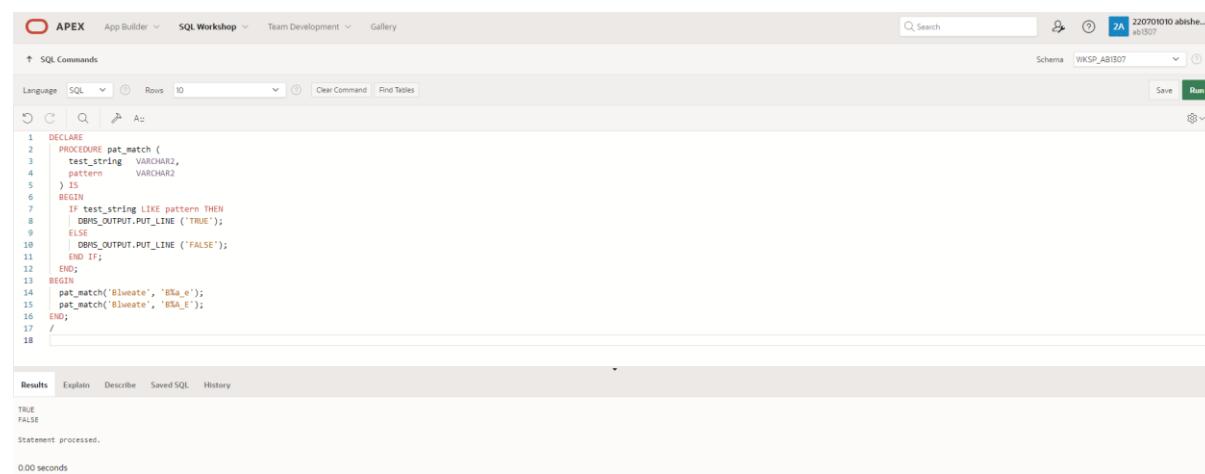
```
1 CREATE OR REPLACE PROCEDURE pri_bool(
2     boo_name  VARCHAR2,
3     boo_val   BOOLEAN
4 ) IS
5 BEGIN
6     IF boo_val IS NULL THEN
7         DBMS_OUTPUT.PUT_LINE( boo_name || ' = NULL');
8     ELSIF boo_val = TRUE THEN
9         DBMS_OUTPUT.PUT_LINE( boo_name || ' = TRUE');
10    ELSE
11    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

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 code editor contains the provided PL/SQL block. The results pane at the bottom shows the output of the `DBMS_OUTPUT.PUT_LINE` statements:

```
TRUE
FALSE
```

Below the results, it says "Statement processed." and "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;

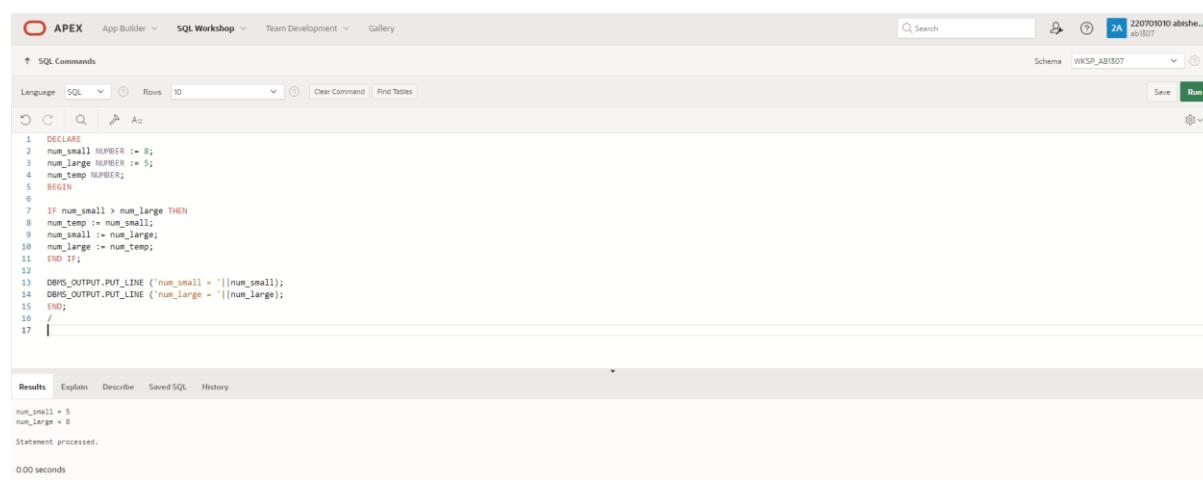
DBMS_OUTPUT.PUT_LINE ('num_small ='||num_small);

DBMS_OUTPUT.PUT_LINE ('num_large ='||num_large);

END;

/

OUTPUT:



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

```
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, the 'Results' tab is active, showing the output:

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

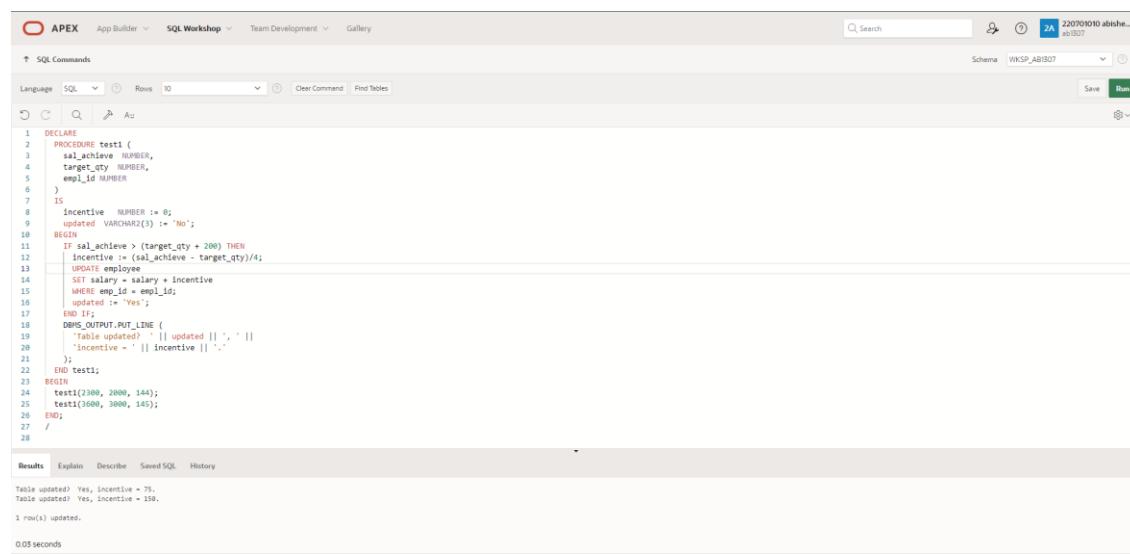
0.00 seconds
```

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:



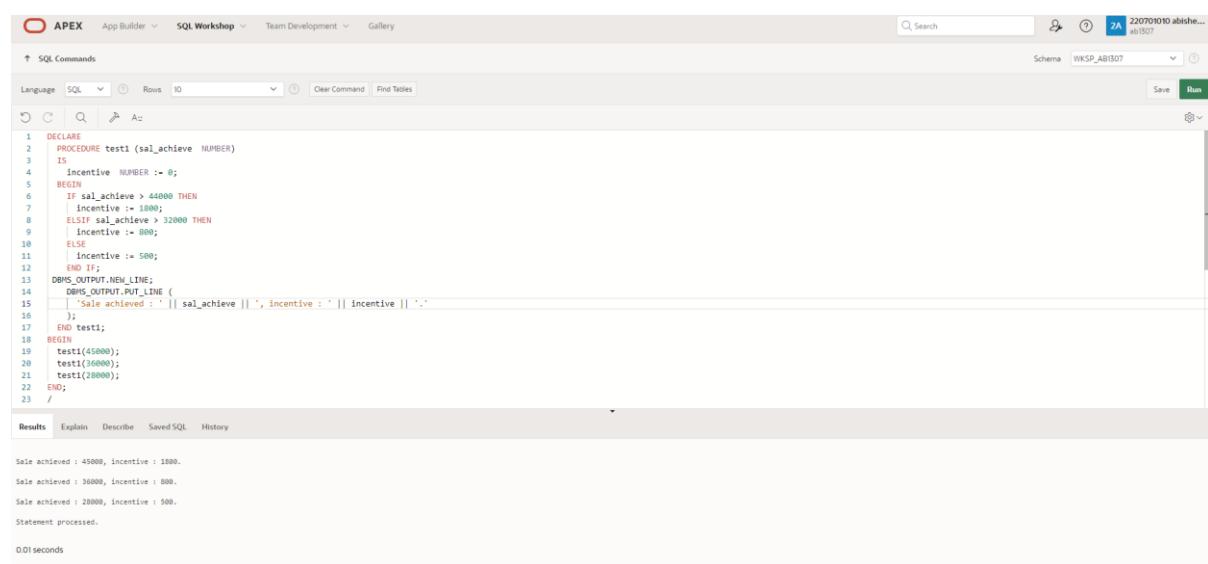
```
APEX App Builder SQL Workshop Team Development Gallery
SQL Commands
Language SQL Rows 10 Clear Command Find Tables
Schema WKSP_AB1B07 Save Run
1  DECLARE
2    PROCEDURE test1 (
3      sal_achieve NUMBER,
4      target_qty NUMBER,
5      emp_id NUMBER
6    )
7    IS
8      incentive NUMBER := 0;
9      updated VARCHAR2(3) := 'No';
10     BEGIN
11       IF sal_achieve > (target_qty + 200) THEN
12         incentive := (sal_achieve - target_qty)/4;
13         UPDATE employees
14           SET salary = salary + incentive
15           WHERE employee_id = emp_id;
16           updated := 'Yes';
17       END IF;
18       DBMS_OUTPUT.PUT_LINE (
19         'Table updated? ' || updated || ',' ||
20         'incentive = ' || incentive || '.';
21     );
22   END test1;
23   BEGIN
24     test1(2300, 2000, 144);
25     test1(3600, 3000, 145);
26   END;
27 /
Results Explain Describe Saved SQL History
Table updated? Yes, Incentive = 75.
Table updated? Yes, Incentive = 150.
1 row(s) updated.
0.05 seconds
```

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The code area contains the PL/SQL procedure definition and its execution. The output pane displays the results of the procedure calls, showing the sales amount and corresponding incentive for each case.

```
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.
0.01 seconds

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

QUERY:

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

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

```
BEGIN
```

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

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

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

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

```
    ELSE
```

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

```
    END IF;
```

```
END;
```

```
/
```

OUTPUT:

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

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

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

QUERY:

DECLARE

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

BEGIN

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

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

IF tot_emp >= 45 THEN

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

ELSE

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

END IF;

END;

/

OUTPUT:

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

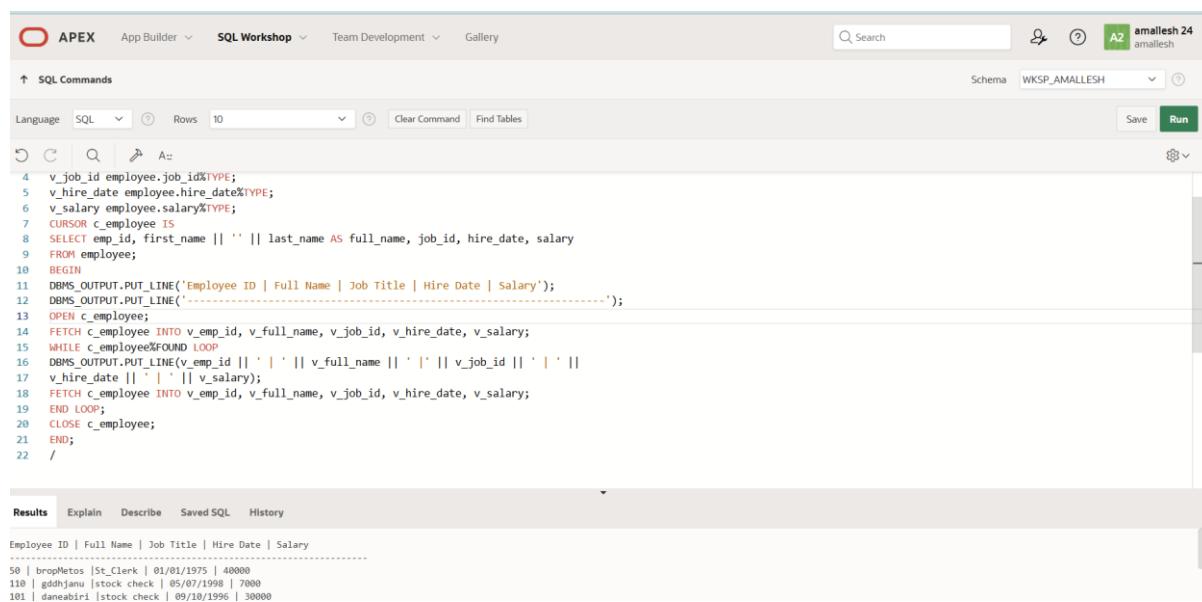
The employees are in the department 80 is:0
There are 45 vacancies in department 80

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The right side shows a user profile for 'amallesh 24 amallesh'. The main area is titled 'SQL Commands' with tabs for Language (set to SQL), Rows (set to 10), and a search bar. Below these are buttons for Clear Command, Find Tables, Save, and Run. The code area contains the PL/SQL block from the previous step. The results tab is selected at the bottom, showing the output of the program. The output is a table with columns: Employee ID | Full Name | Job Title | Hire Date | Salary. It lists three rows of data: 50 | bleepMetos |St_Clerk | 01/01/1975 | 40000, 110 | gddhJanu |stock check | 05/07/1998 | 7000, and 101 | daneabirli |stock check | 09/10/1996 | 30000.

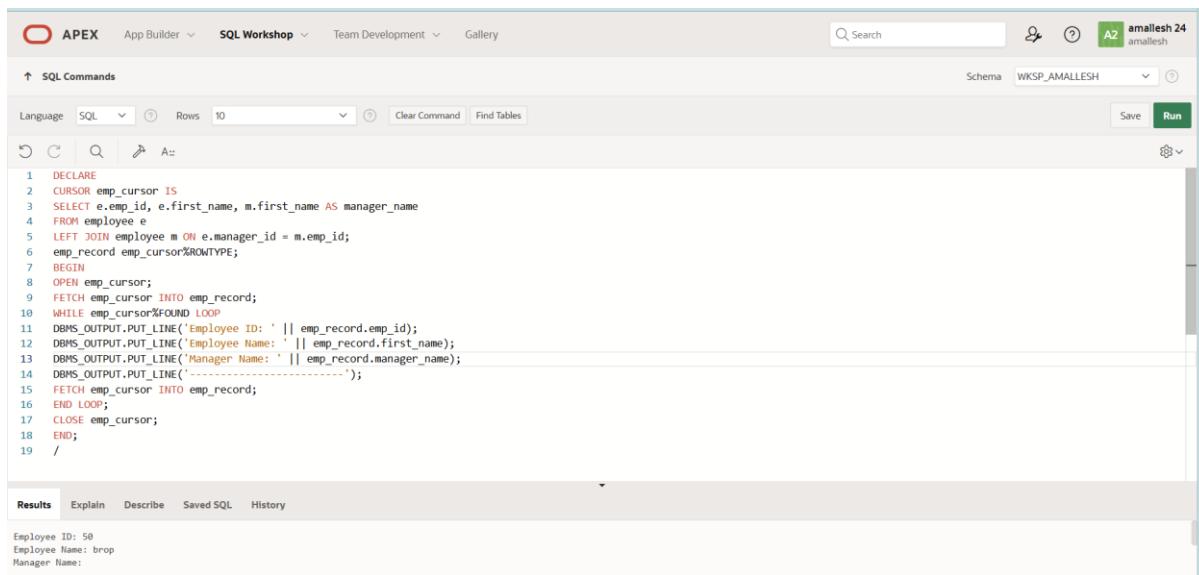
Employee ID	Full Name	Job Title	Hire Date	Salary
50	bleepMetos	St_Clerk	01/01/1975	40000
110	gddhJanu	stock check	05/07/1998	7000
101	daneabirli	stock check	09/10/1996	30000

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema is set to WKSP_AMALLESH. The code area contains the PL/SQL block provided above. The results pane at the bottom shows the output of the DBMS_OUTPUT.PUT_LINE statements, displaying the Employee ID, Employee Name, and Manager Name for each record found in the cursor.

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

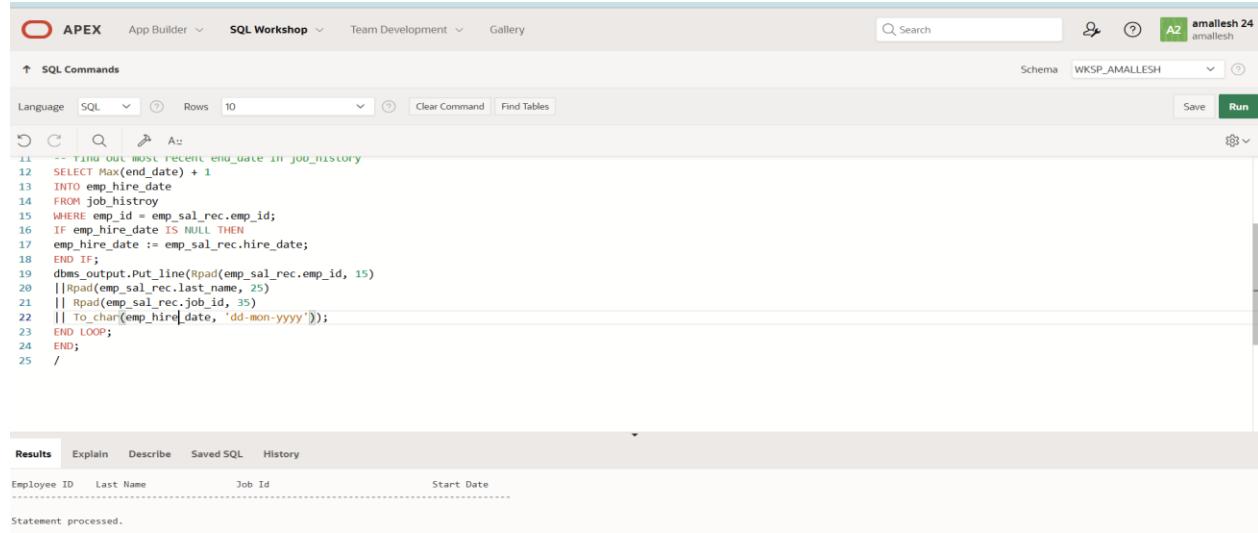
14.) Write a PL/SQL program to display the employee IDs, names, and job history start dates of all employees.

QUERY:

DECLARE

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

/OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'amallesh 24 amallesh'. The main area has tabs for SQL Commands, Explain, Describe, Saved SQL, and History. The SQL Commands tab displays the PL/SQL code with line numbers. The code is as follows:

```
11  -- find out most recent end_date in job_history
12  SELECT Max(end_date) + 1
13  INTO emp_start_date
14  FROM job_history
15  WHERE employee_id = emp_sal_rec.employee_id;
16  IF emp_start_date IS NULL THEN
17    emp_start_date := emp_sal_rec.start_date;
18  END IF;
19  dbms_output.Put_line(Rpad(emp_sal_rec.employee_id, 15)
20  || Rpad(emp_sal_rec.last_name, 25)
21  || Rpad(emp_sal_rec.job_id, 35)
22  || To_char(emp_start_date, 'dd-mon-yyyy'));
23  END LOOP;
24  END;
25  /
```

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

Employee ID	Last Name	Job Id	Start Date

Statement processed.

EXNO:17

PROCEDURES AND FUNCTIONS

DATE:

1.) Factorial of a number using function.

QUERY:

DECLARE

```
1 fac NUMBER := 1;
2 n NUMBER := :1;
3
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;
```

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, a PL/SQL block is entered to calculate the factorial of a number. The code uses a declare section, a begin section with a while loop, and an end section. The loop calculates the factorial by multiplying the current value of 'fac' by 'n' and then decrementing 'n' by 1. Finally, it outputs the result using DBMS_OUTPUT.PUT_LINE. The schema is set to WKSP_AB1307. The results show that the statement was processed in 0.01 seconds. There is also a watermark at the bottom right: "Activate Windows Go to Settings to activate Windows".

```
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;
```

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 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side of the header shows a user profile for '220701010 abishe...' and a schema dropdown set to 'WKSP_AB1307'. The main workspace is titled 'SQL Commands'. The code area contains the following PL/SQL block:

```
1  DECLARE
2      v_book_id NUMBER := 1;
3      v_title VARCHAR2(100);
4      v_author VARCHAR2(100);
5      v_year_published NUMBER;
6  BEGIN
7      v_title := 'Initial Title';
8
9      get_book_info(p_book_id => v_book_id, p_title => v_title, p_author => v_author, p_year_published => v_year_published);
10
11     DBMS_OUTPUT.PUT_LINE('Title: ' || v_title);
12     DBMS_OUTPUT.PUT_LINE('Author: ' || v_author);
13     DBMS_OUTPUT.PUT_LINE('Year Published: ' || v_year_published);
14 END;
15
```

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

```
Title:  
Author:  
Year Published:  
  
Statement processed.  
  
0.03 seconds
```

On the right side of the results area, there is an 'Activate Windows' message: 'Go to Settings to activate Windows.'

At the bottom of the page, there are footer links for '220701010@rajalakshmi.edu.in', 'ab1307', and 'en'. The copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 23.2.4' are also present.

The screenshot shows the Oracle APEX SQL Workshop interface, similar to the first one but with a different query. The top navigation bar and schema selection are identical. The main workspace is titled 'SQL Commands'. The code area contains the following PL/SQL block:

```
1  CREATE OR REPLACE PROCEDURE get_book_info (
2      p_book_id IN NUMBER,
3      p_title IN OUT VARCHAR2,
4      p_author OUT VARCHAR2,
5      p_year_published OUT NUMBER
6  )
7  AS
8  BEGIN
9      SELECT title, author, year_published INTO p_title, p_author, p_year_published
10         FROM books
11        WHERE book_id = p_book_id;
12
13     p_title := p_title || ' - Retrieved';
14 EXCEPTION
15     WHEN NO_DATA_FOUND THEN
16         p_title := NULL;
17         p_author := NULL;
18         p_year_published := NULL;
19 END;
20
```

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

```
Procedure created.  
  
0.04 seconds
```

On the right side of the results area, there is an 'Activate Windows' message: 'Go to Settings to activate Windows.'

At the bottom of the page, there are footer links for '220701010@rajalakshmi.edu.in', 'ab1307', and 'en'. The copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and the version 'Oracle APEX 23.2.4' are also present.

EXNO:18

TRIGGER

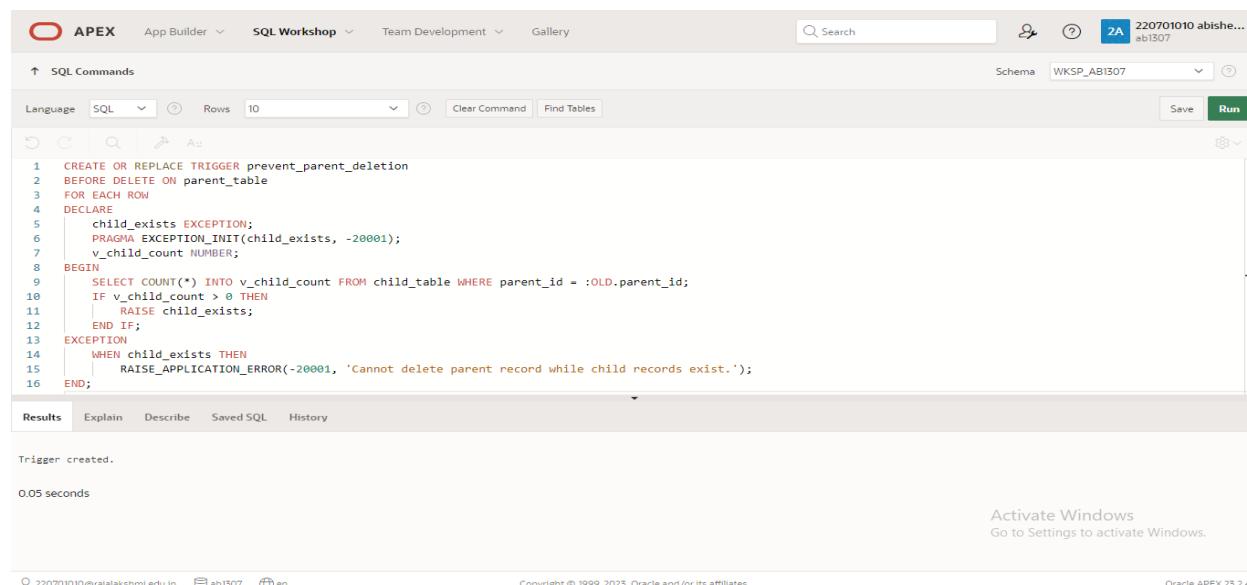
DATE:

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'. The right side shows a user profile '220701010 abishe... ab1307'. The main workspace displays the PL/SQL code for the trigger. The code is as follows:

```
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;
```

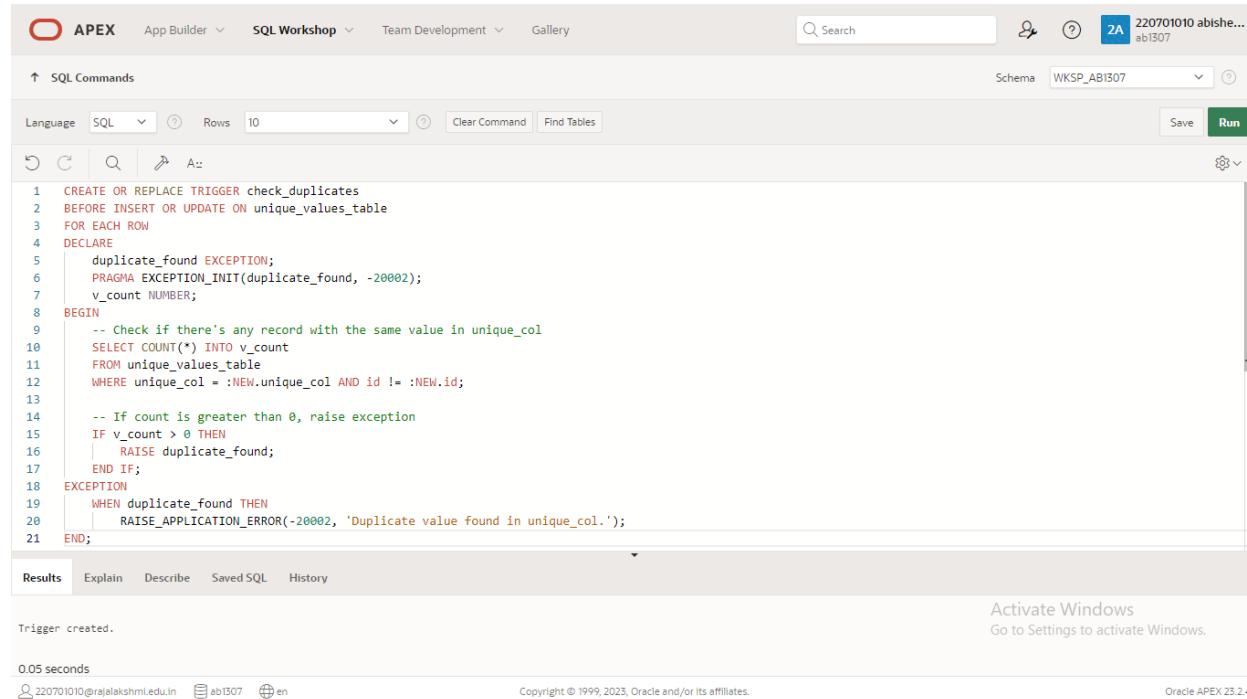
Below the code, the 'Results' tab shows the output: 'Trigger created.' and '0.05 seconds'. The bottom status bar includes the URL '220701010@rajalakshmi.edu.in', session ID 'ab1307', and the message 'Activate Windows Go to Settings to activate Windows.' The footer indicates 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and '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 search bar contains 'Search'. On the right, there are user details '220701010 abishe...', a profile icon, and a 'Run' button. The main workspace is titled 'SQL Commands' and shows the PL/SQL code for the trigger. The code is numbered from 1 to 21. The 'Run' button is highlighted in green. Below the code, the results tab shows the message 'Trigger created.' and a timestamp '0.05 seconds'. The bottom footer includes copyright information 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

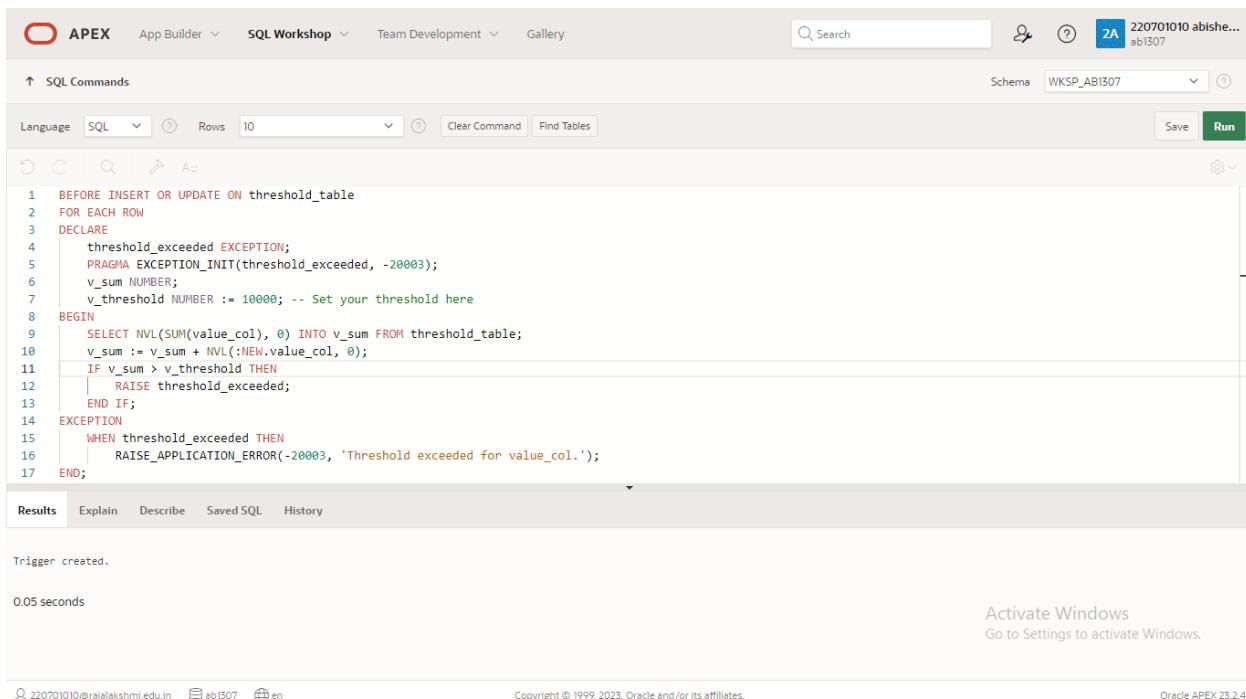
```
1 CREATE OR REPLACE TRIGGER check_duplicates
2 BEFORE INSERT OR UPDATE ON unique_values_table
3 FOR EACH ROW
4 DECLARE
5     duplicate_found EXCEPTION;
6     PRAGMA EXCEPTION_INIT(duplicate_found, -20002);
7     v_count NUMBER;
8 BEGIN
9     -- Check if there's any record with the same value in unique_col
10    SELECT COUNT(*) INTO v_count
11    FROM unique_values_table
12    WHERE unique_col = :NEW.unique_col AND id != :NEW.id;
13
14    -- If count is greater than 0, raise exception
15    IF v_count > 0 THEN
16        RAISE duplicate_found;
17    END IF;
18 EXCEPTION
19    WHEN duplicate_found THEN
20        RAISE_APPLICATION_ERROR(-20002, 'Duplicate value found in unique_col.');
21 END;
```

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' (selected), 'Team Development', and 'Gallery'. The right side of the header shows user information: '220701010 abishe...', '2A', and 'ab1507'. The main workspace is titled 'SQL Commands'. Below the title are buttons for 'Language' (set to 'SQL'), 'Rows' (set to 10), and 'Save/Run'. The SQL editor contains the PL/SQL code for the 'check_threshold' trigger. The code defines a trigger that runs before inserting or updating rows in the 'threshold_table'. It declares variables for the sum of values and the threshold, initializes the sum to zero, and adds the new value to it. If the sum exceeds the threshold, it raises the 'threshold_exceeded' exception. This exception is caught in the exception block, which raises a standard application error with message 'Threshold exceeded for value_col.'. The bottom of the screen shows the results of the command: 'Trigger created.' and '0.05 seconds'. There is also a watermark for 'Activate Windows'.

```
1 BEFORE INSERT OR UPDATE ON threshold_table
2 FOR EACH ROW
3 DECLARE
4     threshold_exceeded EXCEPTION;
5     PRAGMA EXCEPTION_INIT(threshold_exceeded, -20003);
6     v_sum NUMBER;
7     v_threshold NUMBER := 10000; -- Set your threshold here
8 BEGIN
9     SELECT NVL(SUM(value_col), 0) INTO v_sum FROM threshold_table;
10    v_sum := v_sum + NVL(:NEW.value_col, 0);
11    IF v_sum > v_threshold THEN
12        RAISE threshold_exceeded;
13    END IF;
14 EXCEPTION
15     WHEN threshold_exceeded THEN
16         RAISE_APPLICATION_ERROR(-20003, 'Threshold exceeded for value_col.');
17 END;
```

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 APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown is set to 'WKSP_AB1307'. The main workspace displays the PL/SQL code for creating the 'log_changes' trigger. 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,
6     new_col2, change_time)
7     VALUES (audit_seq.NEXTVAL, :OLD.id, :OLD.col1, :NEW.col1, :OLD.col2,
    :NEW.col2, SYSTIMESTAMP);
8 END;
```

Below the code, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab shows the output: "Trigger created." and "0.03 seconds". In the bottom right corner of the workspace, there is a message: "Activate Windows Go to Settings to activate Windows." The footer of the page includes user information (email and session ID), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the version (Oracle APEX 23.2.4).

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 of the header shows a user profile for '220701010 abis...' and a workspace identifier 'ab1307'. The main workspace is titled 'SQL Commands'. The schema dropdown is set to 'WKSP_AB1307'. Below the title bar, there are buttons for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', 'Find Tables', 'Save', and 'Run'. The SQL editor area contains the PL/SQL code for the trigger. The code is numbered from 1 to 15. Lines 1 through 14 are visible, while line 15 is partially cut off at the bottom. The code implements the specified logic for inserting, updating, and deleting rows in the 'activity_table' by logging them into the 'user_activity_log' table. The bottom of the screen shows the results of the command execution, indicating 'Trigger created.' and a execution time of '0.04 seconds'. A watermark for 'Activate Windows' is visible in the bottom right corner.

```
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;
```

Trigger created.
0.04 seconds

Activate Windows
Go to Settings to activate Windows.

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

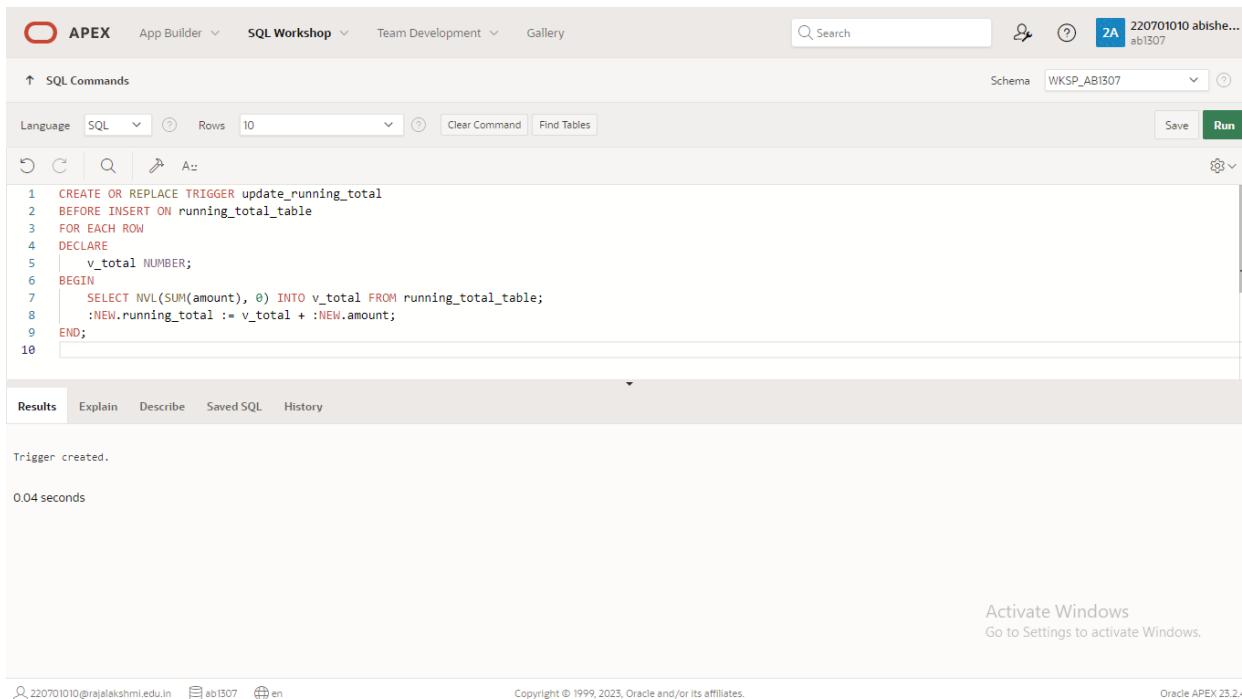
Oracle APEX 23.2.4

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' is selected, along with 'App Builder', 'SQL Workshop', and 'Team Development'. The 'Search' field contains '2A ab1307'. The 'Schema' dropdown is set to 'WKSP_AB1307'. Below the toolbar, there's a 'Language' dropdown set to 'SQL', a 'Rows' input field with '10', and buttons for 'Clear Command' and 'Find Tables'. On the right, there are 'Save' and 'Run' buttons. The main area is titled 'SQL Commands' and contains the PL/SQL code for the trigger. The code is numbered from 1 to 10. The 'Results' tab is selected at the bottom, showing the output: 'Trigger created.' and '0.04 seconds'. In the bottom right corner, there's a message: 'Activate Windows' and 'Go to Settings to activate Windows.'

```
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
```

MONGO DB

EX_NO: 19

DATE:

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

QUERY:

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

OUTPUT:

The screenshot shows a MongoDB shell interface. At the top, there is a search bar labeled "Enter a title...". Below it, there are two buttons: "MongoDB" with a dropdown arrow and a help icon. On the right side, there are two buttons: "Run" with a play icon and "Save" with a disk icon. The main area contains a command line and its output. The command is:

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

The output shows the results of the query execution:

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

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

The output window shows the results of the query execution:

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

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

QUERY:

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

OUTPUT:

The screenshot shows the myCompiler MongoDB interface. It has the same layout as the previous screenshot, with a search bar, "MongoDB" dropdown, and "Run" button. The code editor window contains the same MongoDB query as the previous screenshot.

The output window shows the results of the query execution:

```
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 a MongoDB query editor interface. On the left, there is a text input field labeled "Enter a title...". Below it are two buttons: "MongoDB" with a dropdown arrow and a small info icon. To the right are "Run" and "Save" buttons. The main area contains a code block with the query and its output. The code is:1[{"\$and : [{"address.coord.1": {\$gt : 42}}, {"address.coord.1": {\$lte : 52}}]}, {_id:0, restaurant_id:1, name:1, address:1}] Output

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

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 a MongoDB query editor interface. On the left, there is a text input field labeled "Enter a title..." and a dropdown menu set to "MongoDB". Below these are two buttons: a green "Run" button with "Ctrl+Enter" text above it, and a blue "Save" button with a save icon. A status bar at the bottom indicates "[Execution complete with exit code 0]".

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

Output

```
mycompiler_mongodb>
mycompiler_mongodb>
```

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

QUERY:

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

OUTPUT:

Enter a title...

MongoDB ▾ i Run Save

```
1{{ $and : [ {"address.coord.1": { $gt : 42 }}, {"address.coord.1": { $lte : 52 }}] }, { _id:0, restaurant_id:1, name:1, address:1}}| Output
```

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

7.) Write a MongoDB query to arranged the name of the cuisine in ascending order and for that same cuisine borough should be in descending order.

QUERY:

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

OUTPUT:

Enter a title...

MongoDB ▾ i Run Save

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

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 image shows a screenshot of a MongoDB shell interface. At the top, there is a search bar labeled "Enter a title..." and a toolbar with a MongoDB icon, a dropdown menu, and two buttons: "Run" and "Save". Below the toolbar, a code input field contains the command: `db.restaurants.find({ "address.street": { $exists: true, $ne: "" } })`. To the right of the input field is a "Run" button. Further down, the output window is titled "Output" and displays the results of the query. The output shows the prompt "mycompiler_mongodb>" followed by "[Execution complete with exit code 0]".

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

QUERY:

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

OUTPUT:

The screenshot shows a MongoDB shell interface. On the left, there is a search bar labeled 'Enter a title...' and a dropdown menu set to 'MongoDB'. Below the search bar is a code input field containing the query: `db.restaurants.find({ "address.coord": { $elemMatch: { $type: "double" } } })`. To the right of the code input are two buttons: 'Run' and 'Save'. On the far right, under the heading 'Output', the results of the query execution are displayed. The output shows the prompt 'mycompiler_mongodb>' followed by '[Execution complete with exit code 0]'. This indicates that the query was successfully run but did not return any visible results.

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

QUERY:

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

OUTPUT:

Enter a title...

MongoDB ▾



▶ Run

Save

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

Output

```
mycompiler_mongodb>
mycompiler_mongodb>
```

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

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

QUERY:

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

OUTPUT:

Enter a title...

MongoDB ▾



▶ Run

Save

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

Output

```
mycompiler_mongodb>
mycompiler_mongodb>
```

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

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

QUERY:

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

OUTPUT:

The screenshot shows a MongoDB query editor interface. At the top, there is a search bar labeled "Enter a title...". Below it are two buttons: "MongoDB" with a dropdown arrow and a help icon. To the right are "Run" and "Save" buttons. The main area has a code input field containing the MongoDB query:

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

Below the code input is a vertical scroll bar. To the right, there is an "Output" panel with the following content:

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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

QUERY:

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

OUTPUT:

The screenshot shows a MongoDB shell interface. On the left, there is a search bar labeled 'Enter a title...' and a dropdown menu set to 'MongoDB'. Below the search bar is a code editor containing the following query:

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

To the right of the code editor is a large 'Run' button with a play icon and a 'Save' button with a disk icon. To the far right of the interface is a vertical scroll bar. On the right side of the interface, there is a panel titled 'Output' which displays the results of the executed query:

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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

QUERY:

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

OUTPUT:

Enter a title...

MongoDB ▾ i

Run Save

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

Output

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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

QUERY:

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

OUTPUT:

Enter a title...

MongoDB ▾ ⓘ

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

Run Save

Output

```
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:

Enter a title...

MongoDB ▾ ⓘ

```
1{ "score": { $lt: 5 } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $ne: "American" } })
```

Run Save

Output

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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

QUERY:

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

OUTPUT:

The screenshot shows a MongoDB query editor interface. At the top, there is a search bar labeled "Enter a title...". Below it are two dropdown menus: "MongoDB" and "Run". To the right of these are "Run" and "Save" buttons. The main area contains the MongoDB query:

```
1lt: 5 } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $nin: ["American", "Chinese"] } })
```

Below the query is a "Output" section. It displays the command prompt "mycompiler_mongodb>" followed by the response "mycompiler_mongodb>". At the bottom of the output window, it says "[Execution complete with exit code 0]".

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

QUERY:

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

OUTPUT:

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

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

Next to the input field is a "Output" button. The output pane below shows the command being run and the results:

```
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:

Enter a title...

MongoDB ▾ ⚙ Run Save

```
1 $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], "borough": "Manhattan" }]} Output
```

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

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

QUERY:

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

OUTPUT:

Enter a title...

MongoDB ▾ ⚙ Run Save

```
1 .score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }] }]} Output
```

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

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

QUERY:

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

OUTPUT:

The screenshot shows a MongoDB query editor interface. At the top, there is a search bar labeled "Enter a title...". Below it, a dropdown menu says "MongoDB" with a dropdown arrow, and a small info icon. To the right are two buttons: a green "Run" button and a blue "Save" button. The main area contains the MongoDB query. Below the query, the output window shows the command run and the resulting output from the MongoDB shell.

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

```
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 a MongoDB query editor interface. At the top, there is a search bar labeled "Enter a title...". Below the search bar are two dropdown menus: "MongoDB" and "Save". To the right of these are two buttons: "Run" (green) and "Save" (blue). The main area contains a code input field with the following MongoDB query:

```
1:core": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $nin: ["American", "Chinese"] } })
```

Next to the code is a "Output" button. The output window below shows the command prompt "mycompiler_mongodb>" followed by the command itself. It ends with the message "[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:

My compiler

Enter a title...

MongoDB ▾ i

Run Save

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

Output

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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

RESULT:

MONGO DB

EX_NO: 20

DATE:

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB Compass interface. On the left, there is a search bar labeled "Enter a title...". Below it are two buttons: "MongoDB" with a dropdown arrow and a help icon. On the right, there are two buttons: "Run" (green) and "Save" (blue). The main area has a light blue header with the text "1 db.movies.find({ year: 1893 })". The body of the interface is mostly empty, indicating no results are currently displayed. A small "Output" section on the right shows the command run and the message "[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 Compass interface. On the left, there is a search bar labeled "Enter a title...". Below it are two buttons: "MongoDB" with a dropdown arrow and a help icon. On the right, there are two buttons: "Run" (green) and "Save" (blue). The main area has a light blue header with the text "1 db.movies.find({ runtime: { \$gt: 120 } })". The body of the interface is mostly empty, indicating no results are currently displayed. A small "Output" section on the right shows the command run and the message "[Execution complete with exit code 0]".

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

QUERY:

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

OUTPUT:

The screenshot shows a MongoDB shell interface. At the top, there is a search bar labeled "Enter a title..." and a toolbar with a MongoDB icon, a dropdown menu, and two buttons: "Run" and "Save". Below the toolbar, the command `db.movies.find({ genres: 'Short' })` is entered in the query field. To the right, under the heading "Output", the results are displayed as follows:

```
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 a MongoDB shell interface. At the top, there is a search bar labeled "Enter a title..." and a toolbar with buttons for "MongoDB", "Run", "Save", and keyboard shortcuts "Ctrl+Enter" and "Ctrl+S". Below the toolbar, a code input field contains the command: `1 db.movies.find({ directors: 'William K.L. Dickson' })`. To the right of the input field is a large "Output" panel. The output shows the results of the query: `mycompiler_mongodb>`, `mycompiler_mongodb>`, and [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 a MongoDB shell interface. At the top, there is a search bar labeled "Enter a title..." and a toolbar with buttons for "MongoDB", "Run", "Save", and keyboard shortcuts "Ctrl+Enter" and "Ctrl+S". Below the toolbar, a code input field contains the command: `1 db.movies.find({ countries: 'USA' })`. To the right of the input field is a large "Output" panel. The output shows the results of the query: `mycompiler_mongodb>`, `mycompiler_mongodb>`, and [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:

Enter a title...

MongoDB ▾

Run Save

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

Output

```
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:

Enter a title...

MongoDB ▾

Run Save

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

Output

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

8.) Retrieve all movies from the 'movies' collection that have complete information and have an IMDb rating higher than 7.

QUERY:

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

OUTPUT:

The screenshot shows a MongoDB shell interface. On the left, there is a search bar labeled "Enter a title...". Below it are two buttons: "MongoDB" with a dropdown arrow and a small info icon. To the right are "Run" and "Save" buttons. In the main area, a command is typed into the input field: `1 db.movies.find({ 'imdb.rating': { $gt: 7 } })`. The output window on the right shows the results of the query execution. It starts with the prompt "mycompiler_mongodb>". The output then repeats the command "mycompiler_mongodb>". Finally, it ends with "[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 a MongoDB shell interface. In the top left, there is a search bar with placeholder text "Enter a title...". Below it, a dropdown menu is set to "MongoDB" and a help icon is visible. On the right, there are "Run" and "Save" buttons. The main area contains a command line with the following text:
1 db.movies.find({ 'tomatoes.viewer.rating': { \$gt: 4 } })

In the bottom right corner of the main window, the word "Output" is displayed. The output pane shows the results of the query:
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 a MongoDB shell interface. In the top left, there is a search bar with placeholder text "Enter a title...". Below it, a dropdown menu is set to "MongoDB" and a help icon is visible. On the right, there are "Run" and "Save" buttons. The main area contains a command line with the following text:
1 db.movies.find({ 'awards.wins': { \$gt: 0 } })

In the bottom right corner of the main window, the word "Output" is displayed. The output pane shows the results of the query:
mycompiler_mongodb>
mycompiler_mongodb>
[Execution complete with exit code 0]

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

QUERY:

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

OUTPUT:

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

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

mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]

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

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

QUERY:

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

OUTPUT:

Enter a title...

MongoDB ▾ Info

Run Save

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

Output

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

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

QUERY:

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

OUTPUT:

Enter a title...

MongoDB ▾ Info

Ctrl+Enter Run Save

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

Output

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

14.) Retrieve all movies with title, languages, released, directors, writers, countries from the 'movies' collection in MongoDB that have a word "scene" in the title.

QUERY:

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

OUTPUT:

The screenshot shows a MongoDB shell interface. On the left, there is a search bar labeled "Enter a title...". Below it, a dropdown menu says "MongoDB" with a "Save" button next to it. On the right, there are two buttons: "Run" and "Save". In the center, a code editor window contains the following command:

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

To the right of the code editor is a "Output" panel. It displays the results of the query:

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

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

Viva(5)	
Total (15)	
Faculty Signature	

RESULT: