

CREATING AND MANAGING TABLES

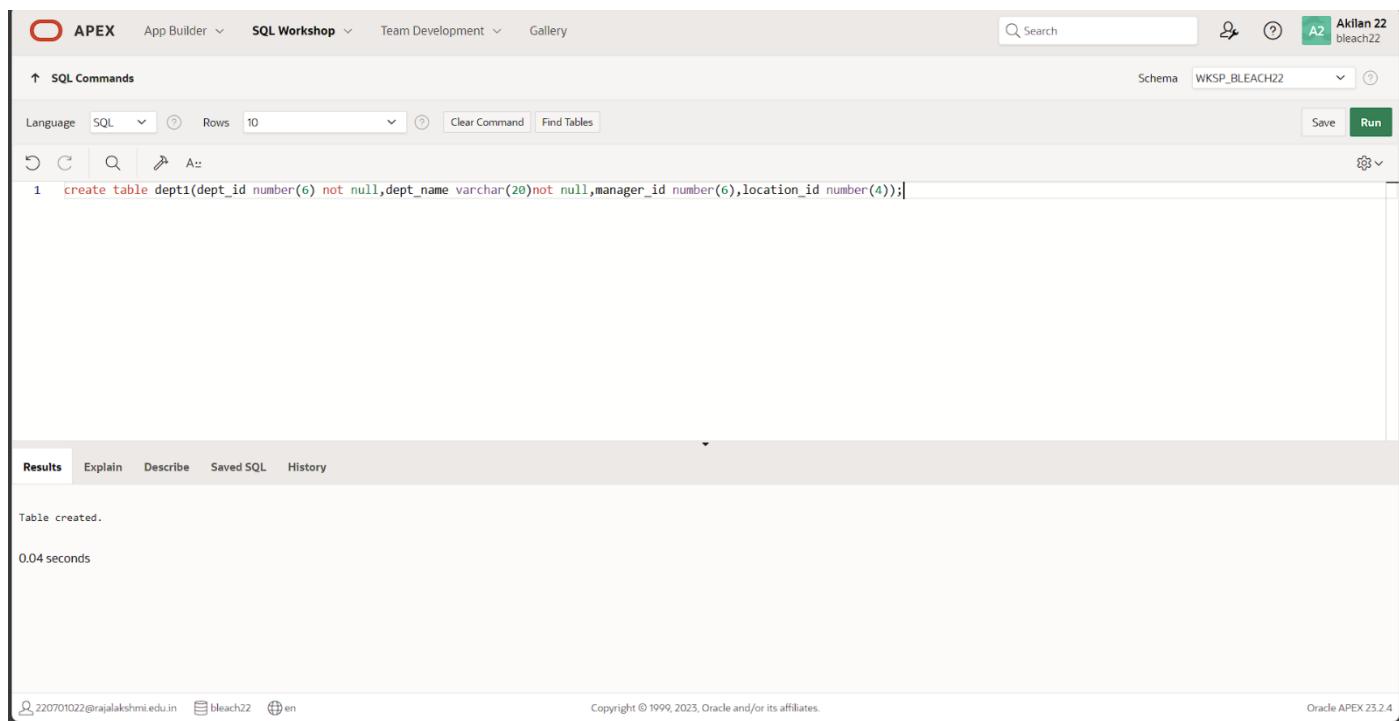
1.Create the DEPT table based on the DEPARTMENT following the table instance chart below. Confirm that the table is created.

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

Query:

Create table DEPT(dept_id number(6)not null, dept_name varchar(20)not null, manager_id number(6), location_id number(4));

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. In the SQL Commands tab, a single line of SQL code is entered to create a table:

```
1 create table dept1(dept_id number(6) not null,dept_name varchar(20)not null,manager_id number(6),location_id number(4));
```

Below the code, the results show:

Table created.
0.04 seconds

At the bottom, the footer includes user information (220701022@rajalakshmi.edu.in, bleach22, en), copyright notice (Copyright © 1999, 2023, Oracle and/or its affiliates.), and the software version (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 top navigation bar, 'APEX' is selected, followed by 'App Builder', 'SQL Workshop' (which is highlighted in blue), 'Team Development', and 'Gallery'. On the right side, there's a user profile 'Akilan 22' and a schema dropdown set to 'WKSP_BLEACH22'. The main area is titled 'SQL Commands'. Below it, there are buttons for 'Language' (set to 'SQL'), 'Rows' (set to 10), 'Clear Command', and 'Find Tables'. The SQL command 'create table emp(id number(7),last_name varchar(25),first_name varchar(25),dept_id number(7));' is entered in the command input field. At the bottom, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The results section displays the message 'Table created.' and '0.04 seconds'. The bottom footer includes the URL '220701022@rajalakshmi.edu.in', the session ID 'bleach22', and the language 'en'. It also mentions 'Copyright © 1999, 2023, Oracle and/or its affiliates.' and 'Oracle APEX 23.2.4'.

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

QUERY:

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

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 EMP modify(last_name varchar(50));
```

Below the command, the results tab is selected, showing the output:

```
Table altered.
```

Execution time: 0.63 seconds

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

QUERY:

```
Create table employees2(id number(6), first_name varchar(20), last_name varchar(25), salary number(8,2), dept_id number(4));
```

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 create table employees2(id number(6),first_name varchar(20),last_name varchar(25),salary number(8,2),dept_id number(4));
```

Below the command, the results tab is selected, showing the output:

```
Table created.
```

Execution time: 0.04 seconds

5.Drop the EMP table.

QUERY:

```
Drop table emp;
```

OUTPUT:

A screenshot of the Oracle 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 shows a SQL command line with the following input:

```
1 drop table emp1;
```

The results pane below shows the output of the command:

Table dropped.
0.09 seconds

6. Rename the EMPLOYEES2 table as EMP.

QUERY:

Renamed employees2 to emp;

OUTPUT:

A screenshot of the Oracle 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 shows a SQL command line with the following input:

```
1 rename employees2 to EMP1;
```

The results pane below shows the output of the command:

Statement processed.
0.05 seconds

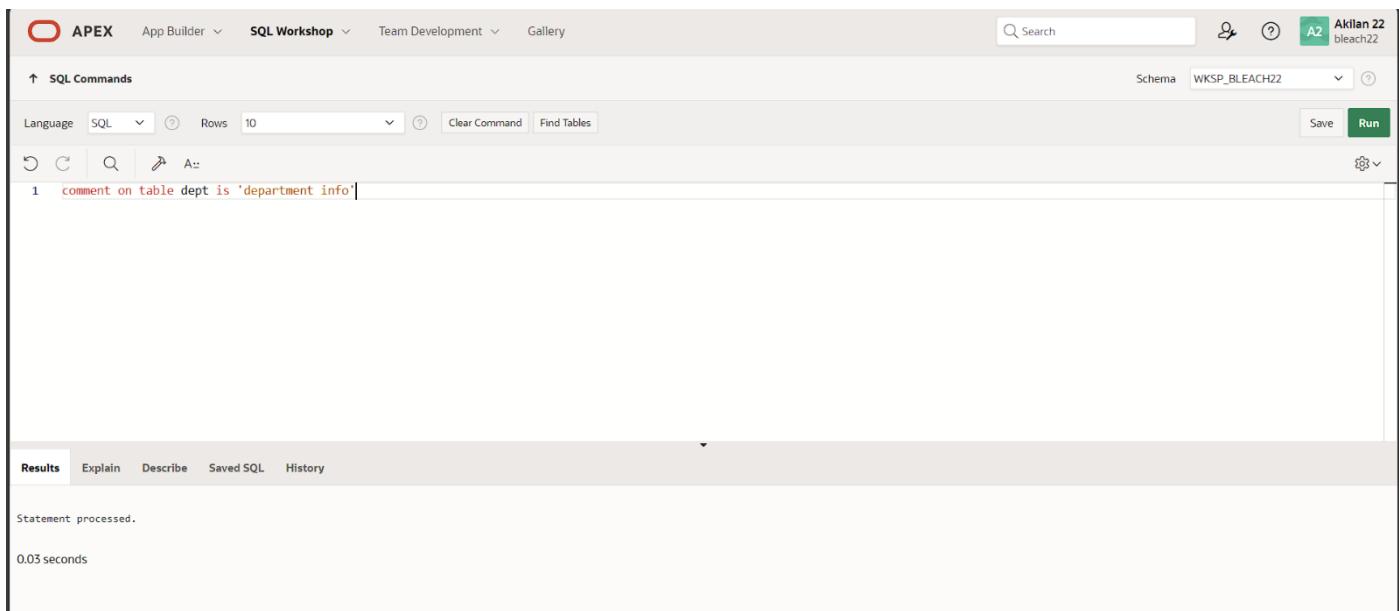
7. Add a comment on DEPT and EMP tables. Confirm the modification by describing the table.

QUERY:

Comment on table emp as 'employee info'

Comment on table dept as 'department info';

OUTPUT:



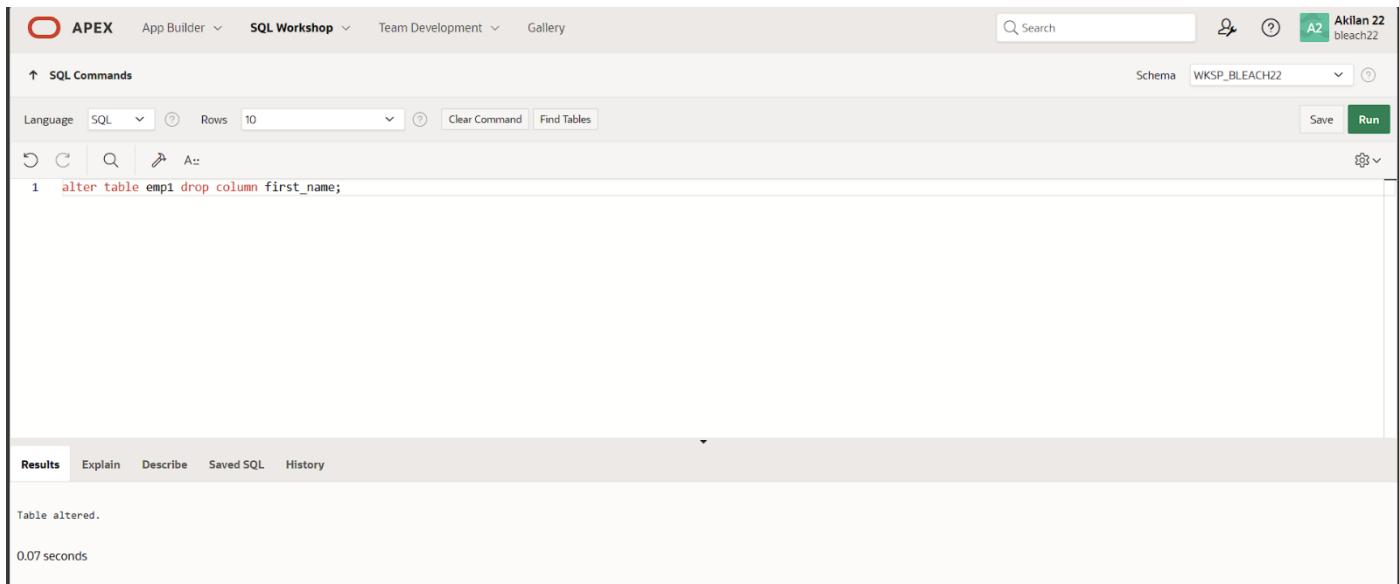
A screenshot of the Oracle SQL Workshop interface. The top navigation bar shows "APEX", "App Builder", "SQL Workshop", "Team Development", and "Gallery". The right side of the header includes a search bar, user information ("A2 Akilan 22 bleach22"), and a "Run" button. The main area is titled "SQL Commands" with tabs for "Language" (set to "SQL"), "Rows" (set to 10), and "Clear Command". Below these are icons for undo, redo, search, and find tables. A command line shows the SQL statement: "comment on table dept is 'department info';". The results section at the bottom shows the output: "Statement processed." and "0.03 seconds".

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

QUERY:

Alter table emp drop column first_name;

OUTPUT:



A screenshot of the Oracle SQL Workshop interface, similar to the previous one. The top navigation bar and right-hand controls are identical. The main area shows the SQL command: "alter table emp1 drop column first_name;". The results section at the bottom shows the output: "Table altered." and "0.07 seconds".

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

MANIPULATING DATA

1.Create MY_EMPLOYEE table with the following structure

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

QUERY:

```
Create table my_employee(id number(4), last_name varchar(25), userid varchar(25), salary number(9,2));
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A single line of SQL code is entered in the command input field: 'create table MY_EMPLOYEE(ID number(4),Last_name varchar(25),First_name varchar(25),Userid varchar(25),Salary number(9,2));'. Below the input field, the status message 'Table created.' is displayed. At the bottom of the interface, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is currently active.

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

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

QUERY:

Insert into my_employee values(1,’Patel’,’Ralph’, ‘rpatel’, 895);

Insert into my_employee values(2,’Dancs’,’Betty’, ‘bdancs’, 860);

OUTPUT:

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

```
1 insert into MY_EMPLOYEE values(2,'Dancs','Betty','bdancs',860);
```

Below the command, the results section shows:

1 row(s) inserted.

0.02 seconds

3.Display the table with values.

QUERY:

Select * from my_employee;

OUTPUT:

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

```
1 select * from MY_EMPLOYEE;
```

Below the command, the results section shows a table with the following data:

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

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

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

QUERY:

Insert into my_employee values(3,’Biri’, ‘Ben’, ‘bbiri’, 1100);

Insert into my_employee values(4,'Newman, 'Chod, 'cnewman',750);

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. In the top navigation bar, 'APEX' and 'SQL Workshop' are selected. The schema is set to 'WKSP_BLEACH22'. The SQL Commands tab is active, showing the command: 'insert into MY_EMPLOYEE values(4,'Newman', 'Chod, 'cnewman',750);'. The Results tab shows the output: '1 row(s) inserted.' and '0.01 seconds'. The Explain, Describe, Saved SQL, and History tabs are also visible.

5. Make the data additions permanent.

QUERY:

Select * from my_employee;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL Commands tab shows the command: 'select * from MY_EMPLOYEE;'. The Results tab displays a table with the following data:

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
4	Newman	Chad	Cnewman	750
3	Biri	Ben	bbiri	1100
2	Dancs	Betty	bdancs	860

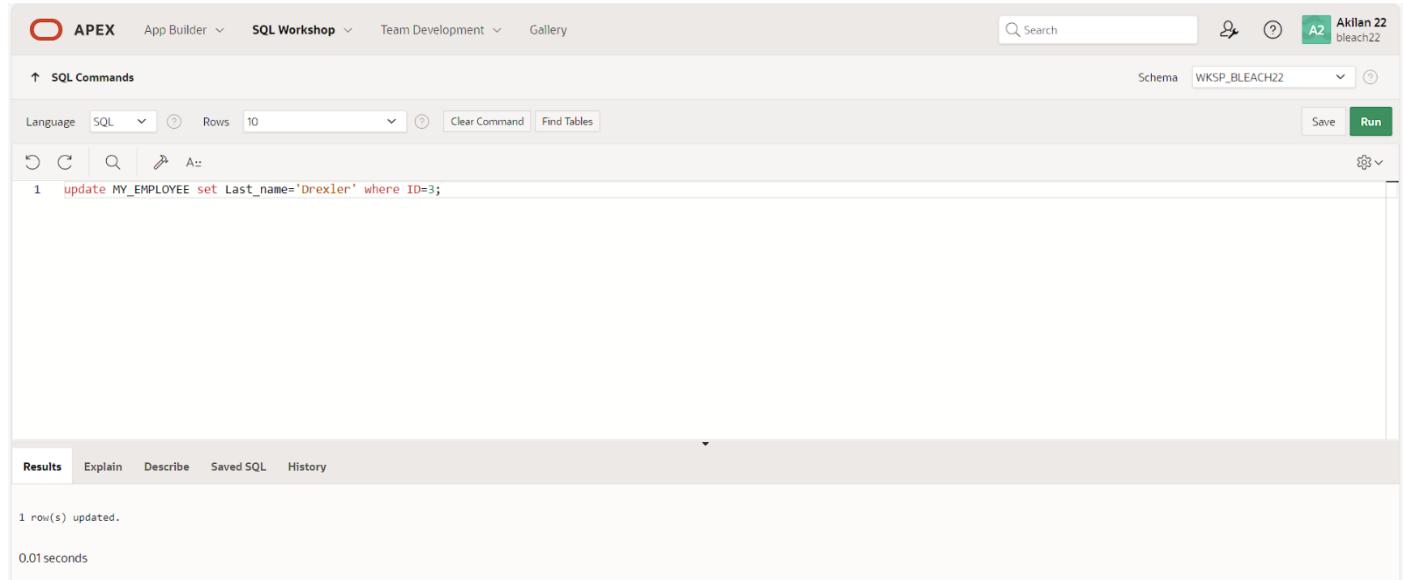
Below the table, it says '4 rows returned in 0.00 seconds'. The bottom of the page includes standard Oracle footer information: 'Copyright © 1999, 2023, Oracle and/or its affiliates.', 'Oracle APEX 23.2.4', and user details '220701022@rajalakshmi.edu.in' and 'bleach22'.

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

QUERY:

update my_employee set last_name='Drexler' where id=3;

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. On the right, there's a search bar, user information ('Akilan 22'), and a session identifier ('A2 WKSP_BLEACH22'). The main area is titled 'SQL Commands' with a sub-section '1 SQL Commands'. It shows the following command:

```
1 update MY_EMPLOYEE set Last_name='Drexler' where ID=3;
```

Below the command, the results tab is selected, showing the output:

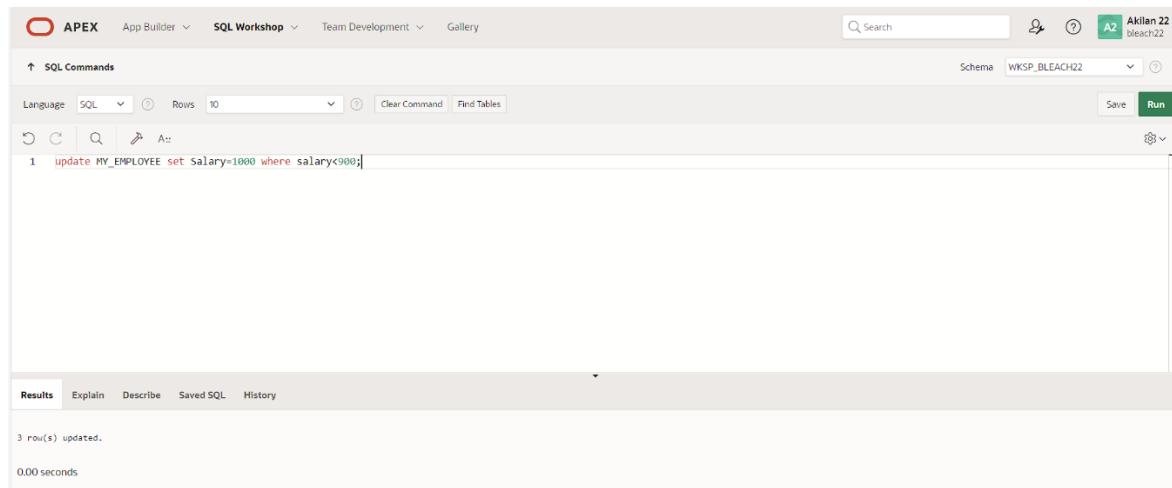
1 row(s) updated.
0.01 seconds

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

QUERY:

Update my_employee set salary=1000 where salary<900;

OUTPUT:



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

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

The results tab shows the output:

3 row(s) updated.
0.00 seconds

8.Delete Betty dancs from MY_EMPLOYEE table.

QUERY:

Delete from my_employee where first_name='Betty';

OUTPUT:

A screenshot of the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'SQL Workshop' tab is active. The main area shows a SQL command: 'delete from MY_EMPLOYEE where First_name='Betty';'. Below the command, the results show '1 row(s) deleted.' and a duration of '0.01 seconds'. The schema is set to 'WKSP_BLEACH22'.

9.Empty the fourth row of the emp table.

QUERY:

Delete from my_employee where id=4;

OUTPUT:

A screenshot of the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'SQL Workshop' tab is active. The main area shows a SQL command: 'delete from MY_EMPLOYEE where ID=4;'. Below the command, the results show '1 row(s) deleted.' and a duration of '0.01seconds'. The schema is set to 'WKSP_BLEACH22'.

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

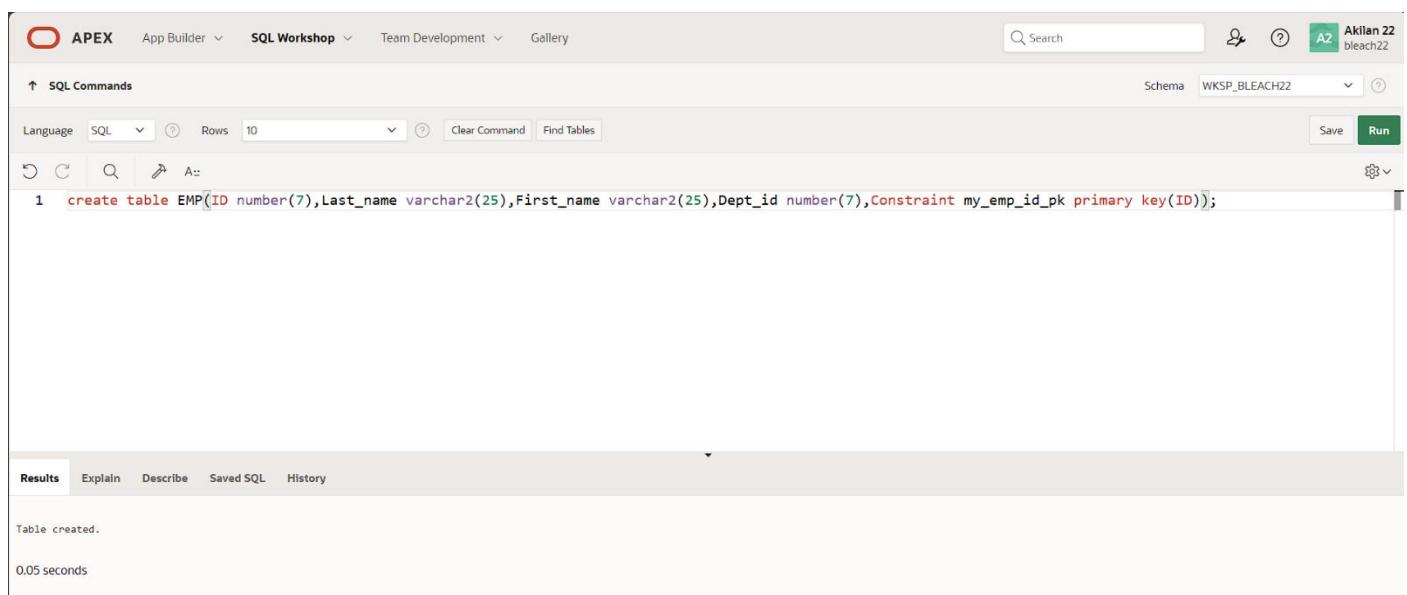
INCLUDING CONSTRAINTS

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

QUERY:

```
Create table emp(id number(7), last_name varchar2(25), first_name varchar(25), dept_id number(7), constraint my_emp_id_pk primary key(id));
```

OUTPUT:



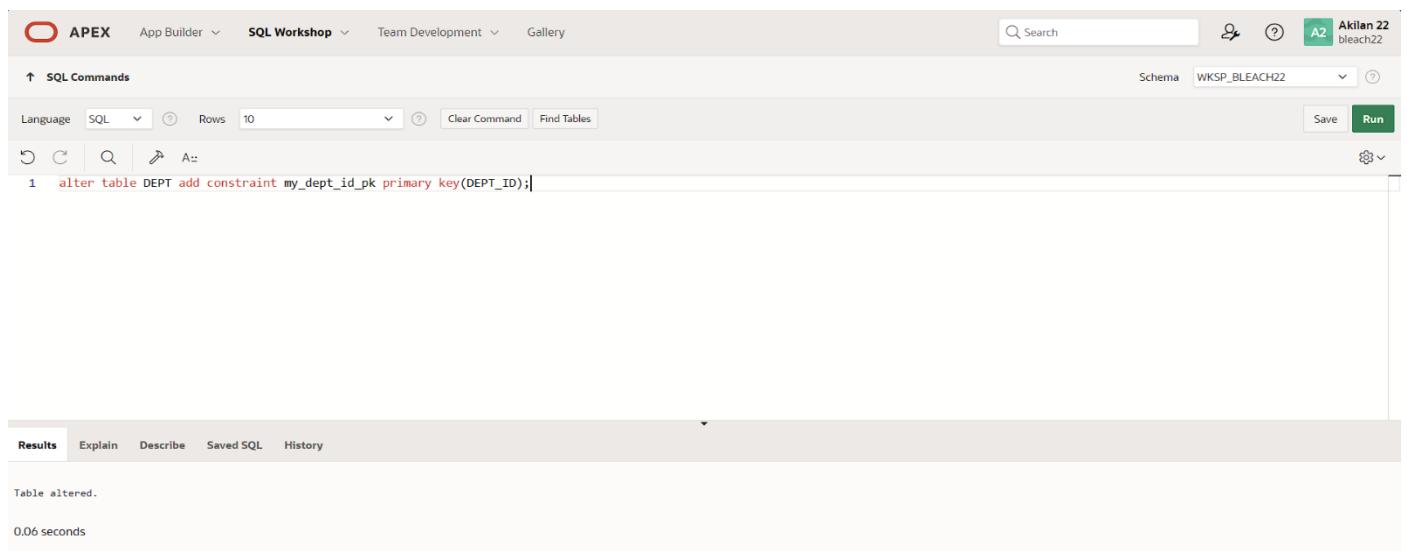
The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A single line of SQL code is entered in the command input field: 'create table EMP(ID number(7),Last_name varchar2(25),First_name varchar2(25),Dept_id number(7),Constraint my_emp_id_pk primary key(ID));'. Below the input field, the status message 'Table created.' is displayed, followed by '0.05 seconds' execution time. The bottom navigation bar includes tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'.

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

QUERY:

```
Alter table dept add constraint my_dept_id_pk primary key(dept_id);
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A single line of SQL code is entered in the command input field: 'alter table DEPT add constraint my_dept_id_pk primary key(DEPT_ID);'. Below the input field, the status message 'Table altered.' is displayed, followed by '0.06 seconds' execution time. The bottom navigation bar includes tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'.

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 dept add constraint my_emp_dept_id_fk foreign key(dept_id) references emp(dept_id);
```

OUTPUT:

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

```
1 alter table DEPT add constraint my_dept_id_fk foreign key(DEPT_ID) references EMP(DEPT_ID);
```

Below the command, the results pane shows the output:

```
Table altered.  
0.07 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 emp add (commission number(2,2), constraint ck check(commission>0));
```

OUTPUT:

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

```
1 alter table EMP add(commission number(2,2),constraint ck check(commission>0));
```

Below the command, the results pane shows the output:

```
Table altered.  
0.07 seconds
```

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

WRITING BASIC SQL SELECT STATEMENTS

- The following statement executes successfully.

Identify the Errors

```
SELECT employee_id, last_name
sal*12 ANNUAL SALARY
```

FROM employees;

QUERY:

```
Select employee_id, last_name, "salary*12" "ANNUAL_SALARY" from employees;
```

OUTPUT:

SQL Commands

```
1 select employee_id, last_name, "SALARY*12" "ANNUAL_SALARY" from employees;
```

Results

EMPLOYEE_ID	LAST_NAME	ANNUAL_SALARY
22	hood	120000

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

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

QUERY:

```
select * from employees;
```

OUTPUT:

SQL Commands

```
1 select * from employees;
```

Results

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	MANAGER_ID	DEPARTMENT_ID
22	robin	hood	rhood@gmail.com	9856231456	09/26/2006	23	10000	56	78

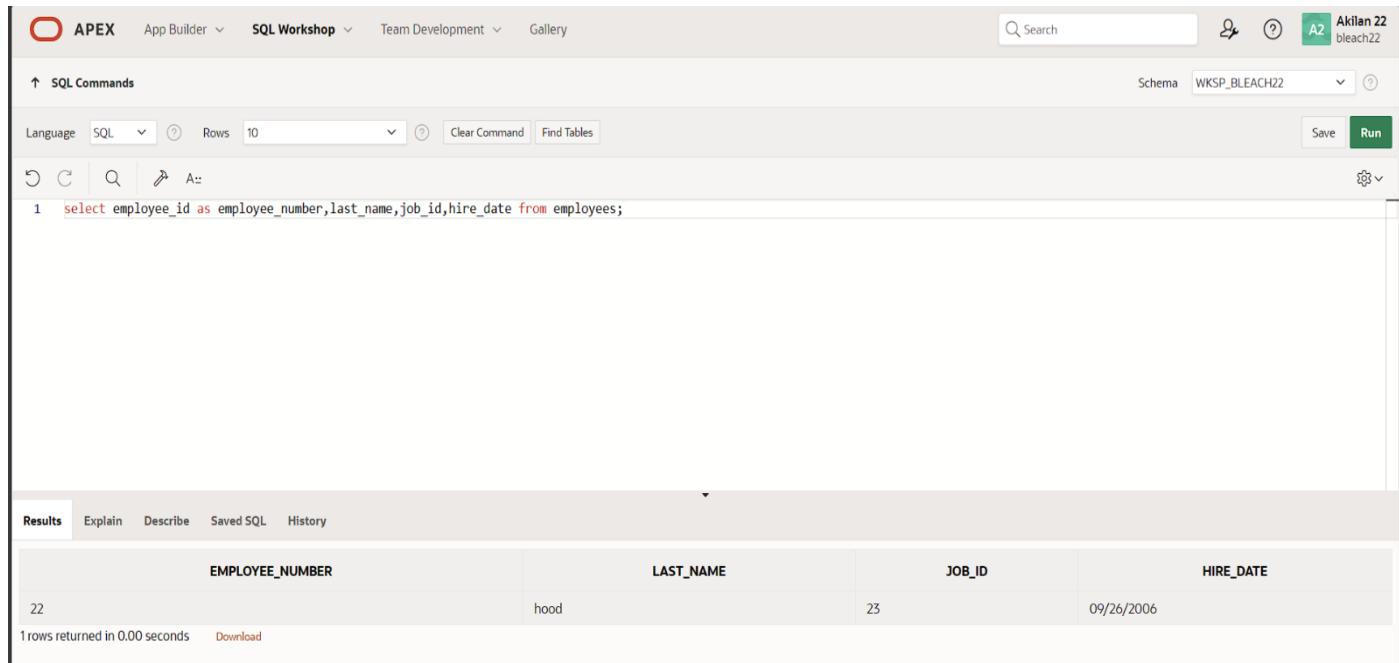
1 rows returned in 0.05 seconds [Download](#)

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

QUERY:

Select employee_id as employee_number, last_name, job_id, hire_date from employees;

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 'Akilan 22' and 'bleach22'. The main area is titled 'SQL Commands' with a search bar and a schema dropdown set to 'WKSP_BLEACH22'. Below the command input field, there are buttons for Language (SQL), Rows (10), Clear Command, Find Tables, Save, and Run. The command entered is:

```
1 select employee_id as employee_number, last_name, job_id, hire_date from employees;
```

The results section shows a single row of data:

EMPLOYEE_NUMBER	LAST_NAME	JOB_ID	HIRE_DATE
22	hood	23	09/26/2006

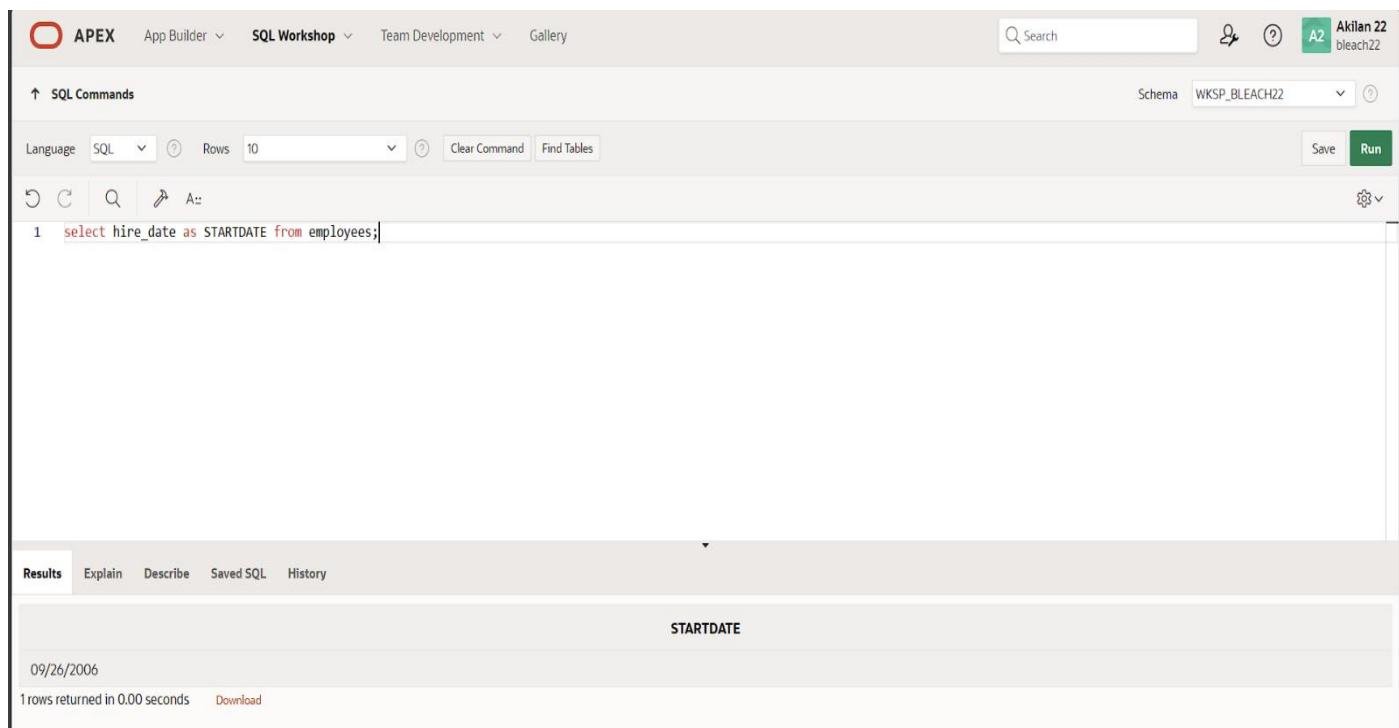
Below the table, it says '1 rows returned in 0.00 seconds' and has a 'Download' link.

4.Provide an alias STARTDATE for the hire date.

QUERY:

Select hire_date as startdate from employees;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface, identical to the previous one but with a different query. The top navigation bar and user profile are the same. The main area is titled 'SQL Commands' with the same button layout. The command entered is:

```
1 select hire_date as STARTDATE from employees;
```

The results section shows a single row of data:

STARTDATE
09/26/2006

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

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

QUERY:

Select distinct job_id from employees;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown is set to WKSP_BLEACH22. The command input area contains the SQL statement: "select distinct job_id from employees;". The results pane shows a single row with the value 23 under the column header JOB_ID. Below the results, it says "1 rows returned in 0.00 seconds".

JOB_ID
23

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

QUERY:

Select last_name||','||job_id as "EMPLOYEE_AND_TITLE" from employees;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown is set to WKSP_BLEACH22. The command input area contains the SQL statement: "select last_name||','||job_id as \"EMPLOYEE_AND_TITLE\" from employees;". The results pane shows a single row with the value hood,23 under the column header EMPLOYEE_AND_TITLE. Below the results, it says "1 rows returned in 0.01 seconds".

EMPLOYEE_AND_TITLE
hood,23

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

QUERY:

```
select employee_id||','||first_name||','||last_name||','||email||','||phone_number||','||hire_date||','||job_id||'  
'||salary||','||manager_id||','||department_id as "the_output" from employees;
```

OUTPUT:

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

```
1 select employee_id||','||first_name||','||last_name||','||email||','||phone_number||','||  
2 hire_date||','||job_id||','||salary||','||manager_id||','||department_id as "THE_OUTPUT"  
3 from employees;
```

Below the code, the 'Results' tab is selected. The output section displays the results of the query:

THE_OUTPUT	
22,robin,hood,hood@gmail.com,9856231456,09/26/2006,23,10000,56,78	

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

Evaluation Procedure	Marks awarded
Practice Evaluation (5)	
Viva(5)	
Total (10)	
Faculty Signature	

RESTRICTING AND SORTING DATA

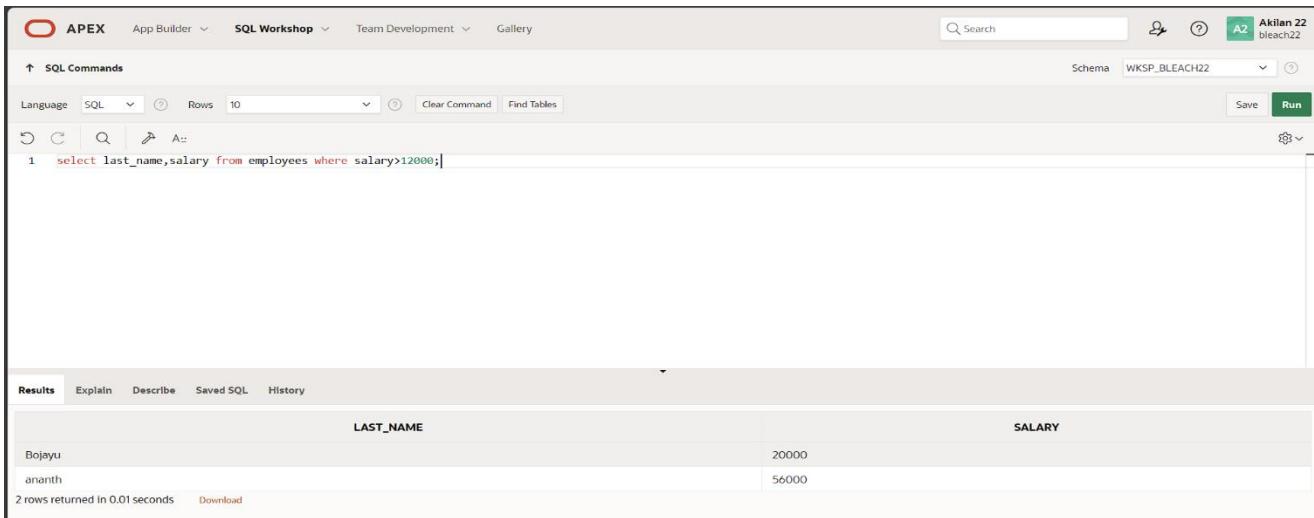
Find the Solution for the following:

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

QUERY:

Select last_name, salary from employees where salary>12000;

OUTPUT:



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

```
1 select last_name,salary from employees where salary>12000;
```

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

LAST_NAME	SALARY
Bojayu	20000
ananth	56000

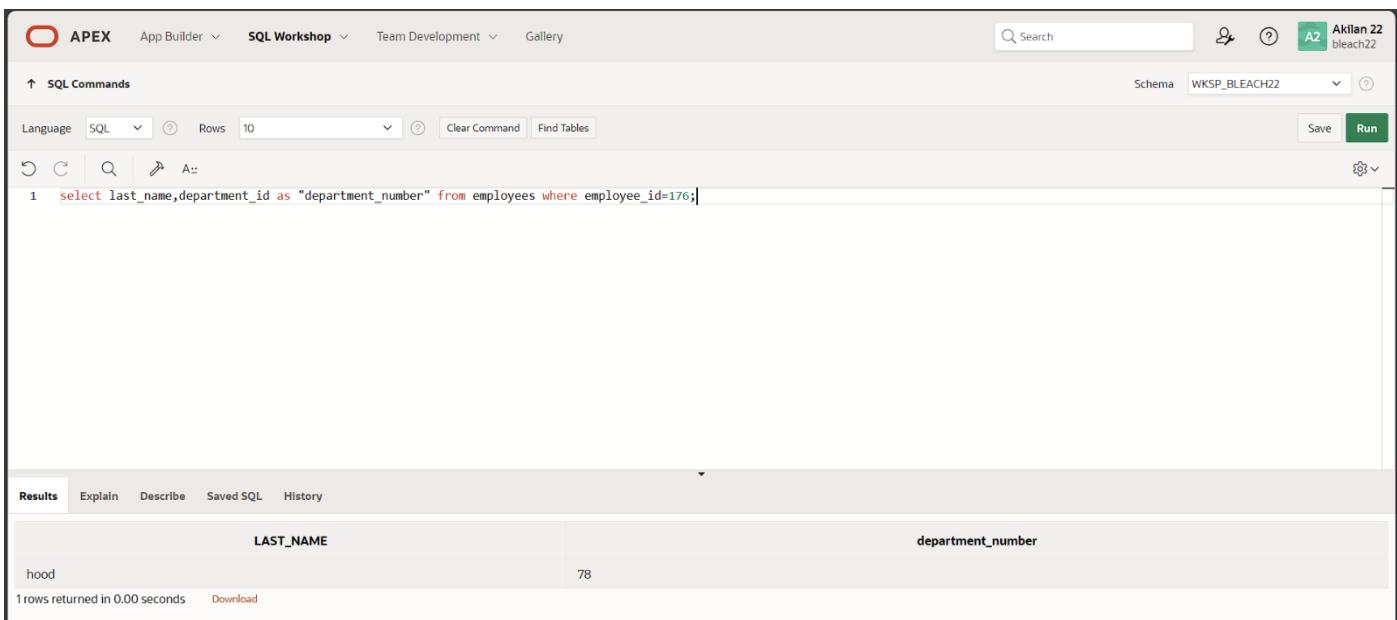
2 rows returned in 0.01 seconds [Download](#)

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

QUERY:

Select last_name, department_id as "department_number" from employees where employee_id=176;

OUTPUT:



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

```
1 select last_name,department_id as "department_number" from employees where employee_id=176;
```

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

LAST_NAME	department_number
hood	78

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

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

QUERY:

Select last_name, salary from employees where salary not between 5000 and 12000;

OUTPUT:

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

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

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

LAST_NAME	SALARY
Bojayu	20000
ananth	56000

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

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

QUERY:

Select last_name, job_id, hire_date from employees where hire_date between '2-10-1998' and '5-1-1998'
Order by hire_date asc;

OUTPUT:

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

```
1 select last_name, job_id, hire_date from employees where hire_date between '2-10-1998' and '5-1-1998' order by hire_date asc;
```

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

LAST_NAME	JOB_ID	HIRE_DATE
Bojayu	89	03/09/1998

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

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

QUERY:

```
select last_name, department_id from employees where department_id in(20,50) order by last_name;
```

OUTPUT:

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

```
1 select last_name,department_id from employees where department_id in(20,50)
2 order by last_name;
```

The results table has two columns: LAST_NAME and DEPARTMENT_ID. The data returned is:

LAST_NAME	DEPARTMENT_ID
hood	50
kumar	20

2 rows returned in 0.00 seconds

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

QUERY:

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

OUTPUT:

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

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

The results table has two columns: EMPLOYEE and MONTHLY SALARY. The data returned is:

EMPLOYEE	MONTHLY SALARY
hood	10000

1 rows returned in 0.01 seconds

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

QUERY:

Select last_name, hire_date from employees where hire_date like '%94';

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL Commands pane contains the query: `select last_name, hire_date from employees where hire_date like '%94';`. The Results pane displays the output:

LAST_NAME	HIRE_DATE
ananth	12/26/1994

1 rows returned in 0.01 seconds

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

QUERY:

Select last_name, job_title from employees where manager_id is null;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL Commands pane contains the query: `select last_name, job_title from employees where manager_id is null;`. The Results pane displays the output:

LAST_NAME	JOB_TITLE
Bojayu	sales representative

1 rows returned in 0.00 seconds

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

QUERY:

select last_name, salary, commission from employees where commission is not null order by salary desc, commission desc;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The SQL Commands pane contains the query: `select last_name, salary, commission from employees where commission is not null order by salary desc, commission desc;`. The Results pane displays the output:

LAST_NAME	SALARY	COMMISION
ananth	56000	.2
Bojayu	20000	.2
hood	10000	.5

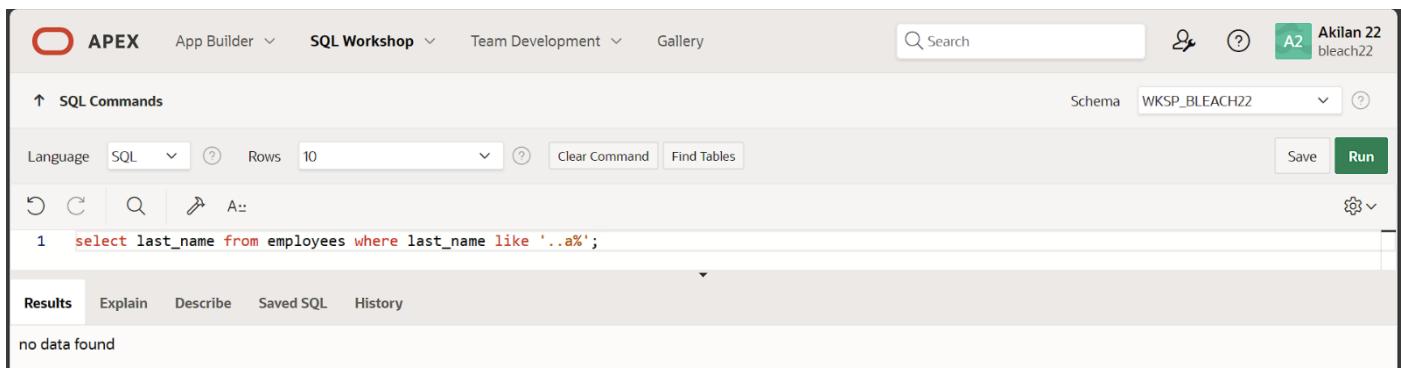
3 rows returned in 0.02 seconds

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

QUERY:

```
select last_name from employees where last_name like '..a%';
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown is set to WKSP_BLEACH22. The SQL command window contains the following code:

```
1 select last_name from employees where last_name like '..a%';
```

The results section shows the message "no data found".

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

QUERY:

```
Select last_name from employees where last_name like '%a%' and last_name like '%e%';
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown is set to WKSP_BLEACH22. The SQL command window contains the following code:

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

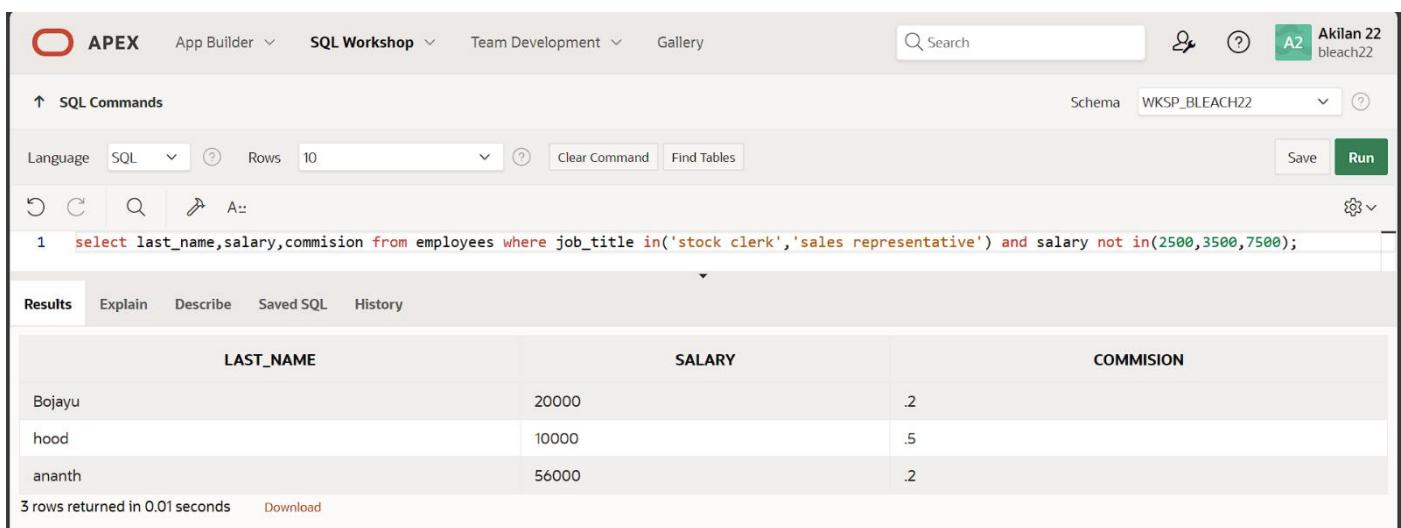
The results section shows the message "no data found".

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

QUERY:

```
Select last_name, salary, commission from employees where job_title in ('stock clerk', 'sales representative') and salary not in(2500,3500,7500);
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown is set to WKSP_BLEACH22. The SQL command window contains the following code:

```
1 select last_name,salary,commission from employees where job_title in('stock clerk','sales representative') and salary not in(2500,3500,7500);
```

The results section displays a table with the following data:

LAST_NAME	SALARY	COMMISSION
Bojayu	20000	.2
hood	10000	.5
ananth	56000	.2

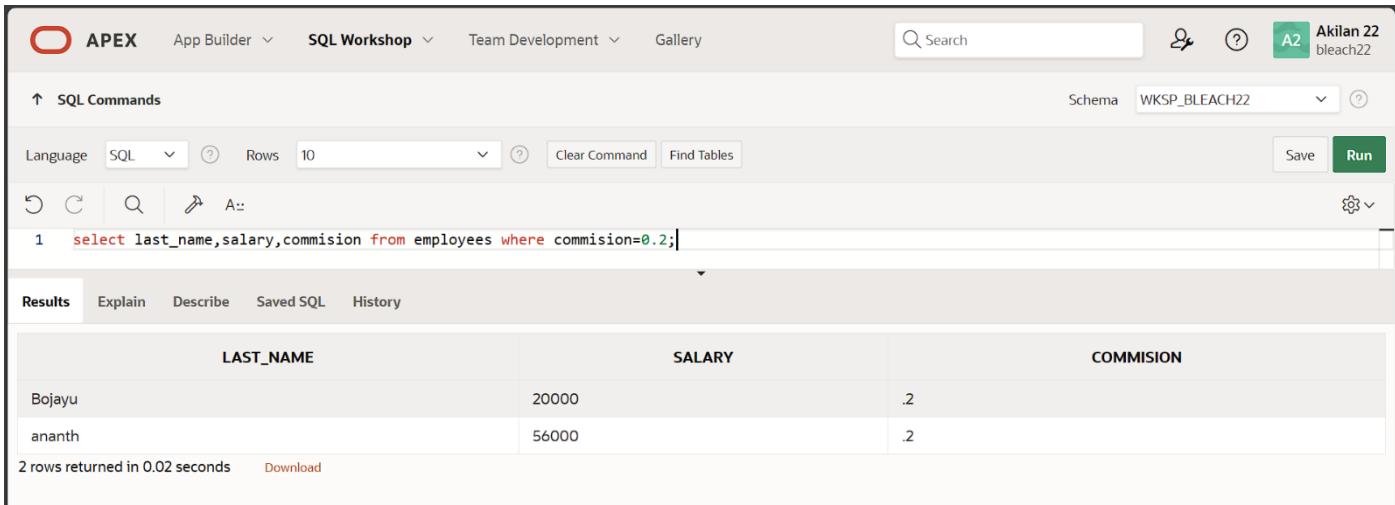
At the bottom of the results, it says "3 rows returned in 0.01 seconds" and has a "Download" link.

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

QUERY:

```
select last_name, salary, commission from employees where commission=0.2;
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and user information ('Akilan 22', 'bleach22') are also present. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP_BLEACH22'. The SQL editor contains the query: 'select last_name, salary, commission from employees where commission=0.2;'. The results tab is selected, displaying a table with three columns: 'LAST_NAME', 'SALARY', and 'COMMISSION'. Two rows are returned: Bojayu with salary 20000 and commission .2, and ananth with salary 56000 and commission .2.

LAST_NAME	SALARY	COMMISSION
Bojayu	20000	.2
ananth	56000	.2

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

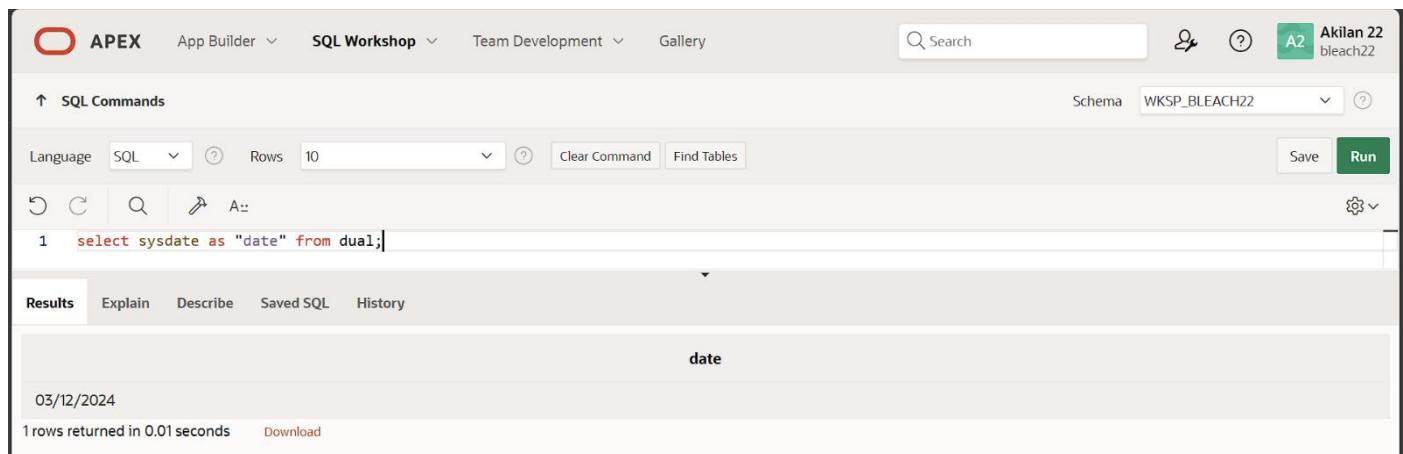
SINGLE ROW FUNCTIONS

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. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown is set to WKSP_BLEACH22. The main area shows the following SQL command:

```
1 select sysdate as "date" from dual;
```

The results pane shows the output:

date
03/12/2024

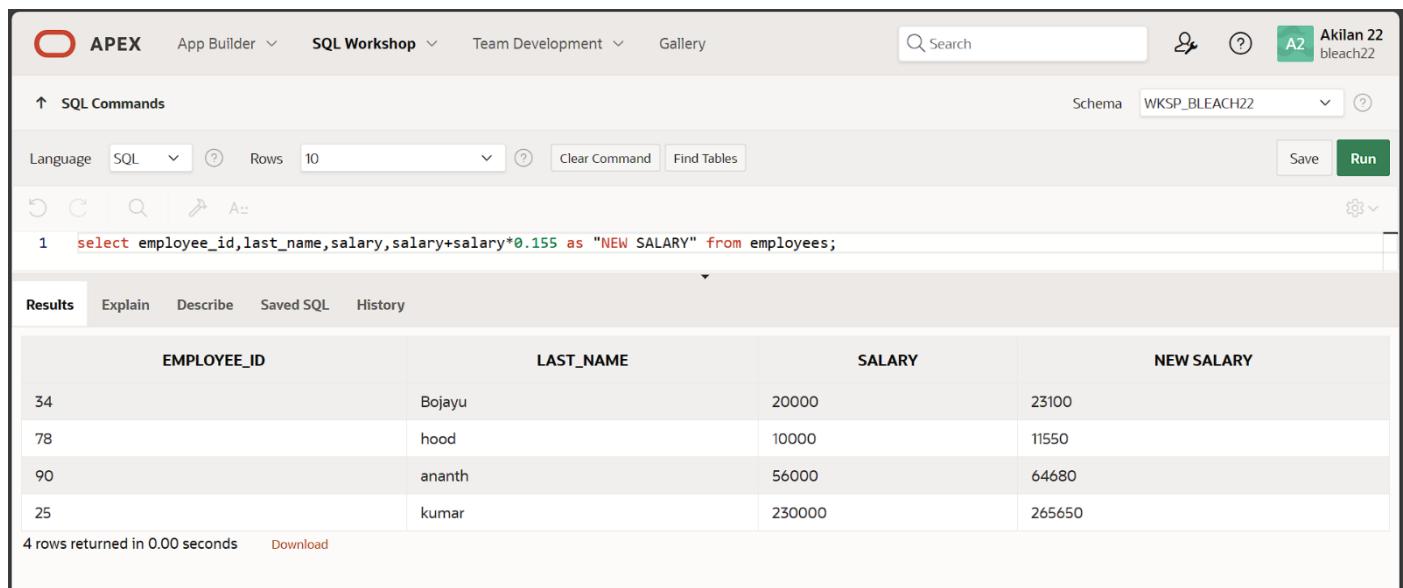
Below the results, it says "1 rows returned in 0.01 seconds".

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

QUERY:

Select employee_id, last_name, salary, salary+salary*0.155 as “new salary” from employees;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown is set to WKSP_BLEACH22. The main area shows the following SQL command:

```
1 select employee_id, last_name, salary, salary+salary*0.155 as "NEW SALARY" from employees;
```

The results pane shows the output:

EMPLOYEE_ID	LAST_NAME	SALARY	NEW SALARY
34	Bojayu	20000	23100
78	hood	10000	11550
90	ananth	56000	64680
25	kumar	230000	265650

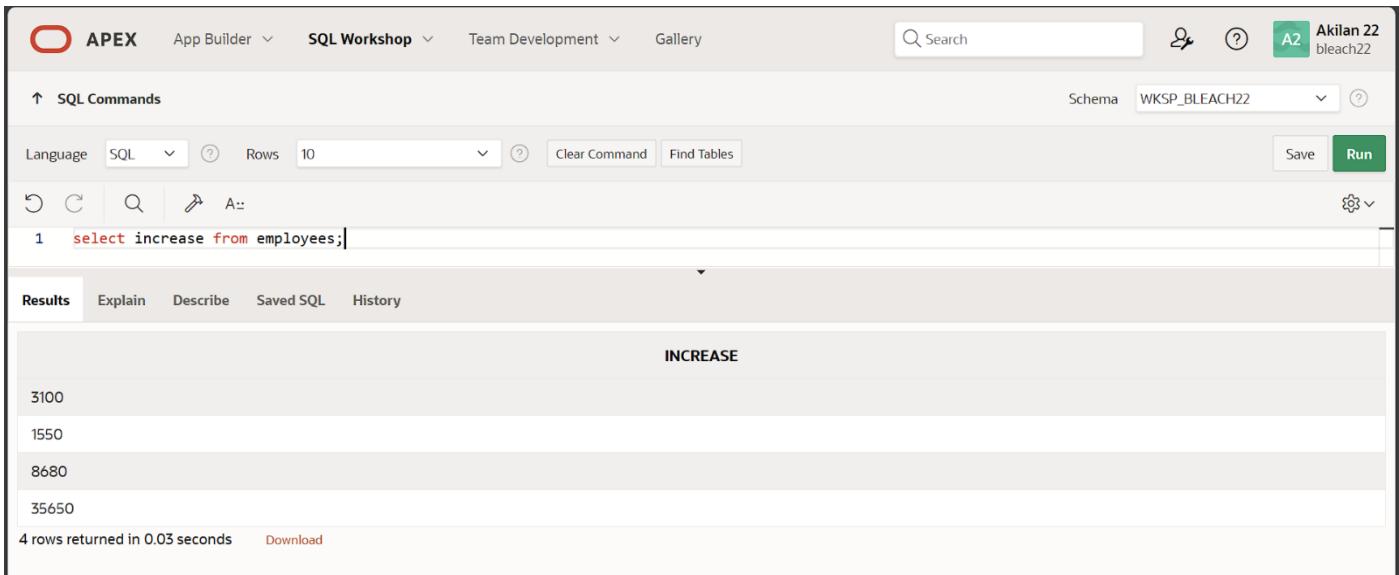
Below the results, it says "4 rows returned in 0.00 seconds".

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

QUERY:

```
Alter table employees add(increase number(10)) update employees set increase=(salary+salary*0.155)-salary select increase from employees;
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the SQL Commands pane, the command `select increase from employees;` is entered. The Results pane displays the output:

INCREASE
3100
1550
8680
35650

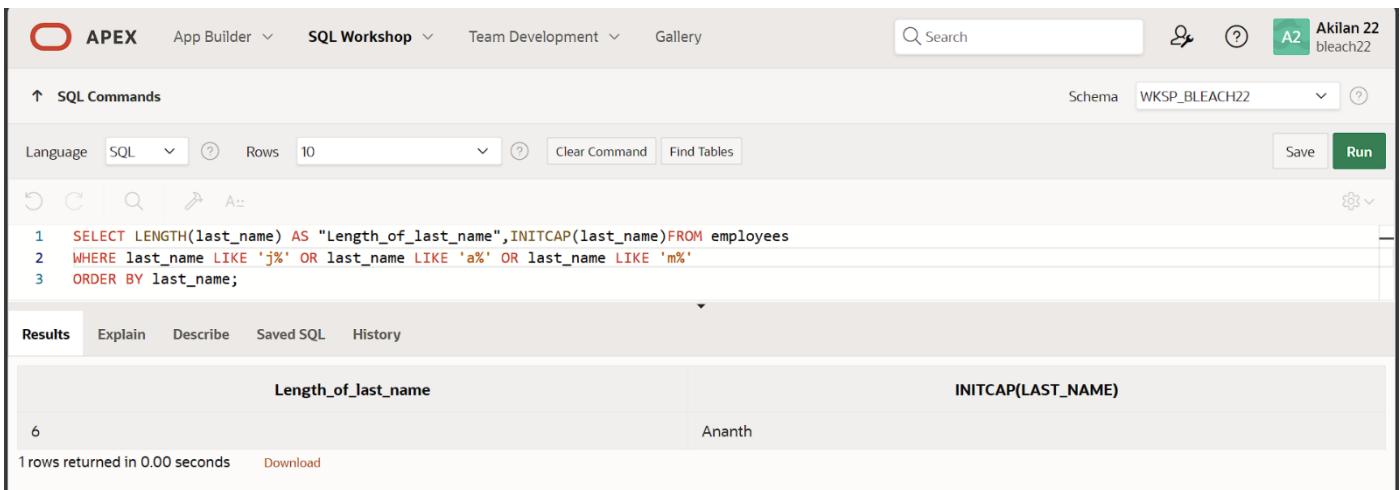
4 rows returned in 0.03 seconds

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

QUERY:

```
Select length(last_name) as "length_of_last_name",initcap(last_name) from employees where last_name like 'j%' or last_name like 'A%' or last_name like 'M%' order by last_name;
```

OUTPUT:



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

```
1 SELECT LENGTH(last_name) AS "Length_of_last_name",INITCAP(last_name)FROM employees
2 WHERE last_name LIKE 'j%' OR last_name LIKE 'a%' OR last_name LIKE 'm%'
3 ORDER BY last_name;
```

The Results pane displays the output:

Length_of_last_name	INITCAP(LAST_NAME)
6	Ananth

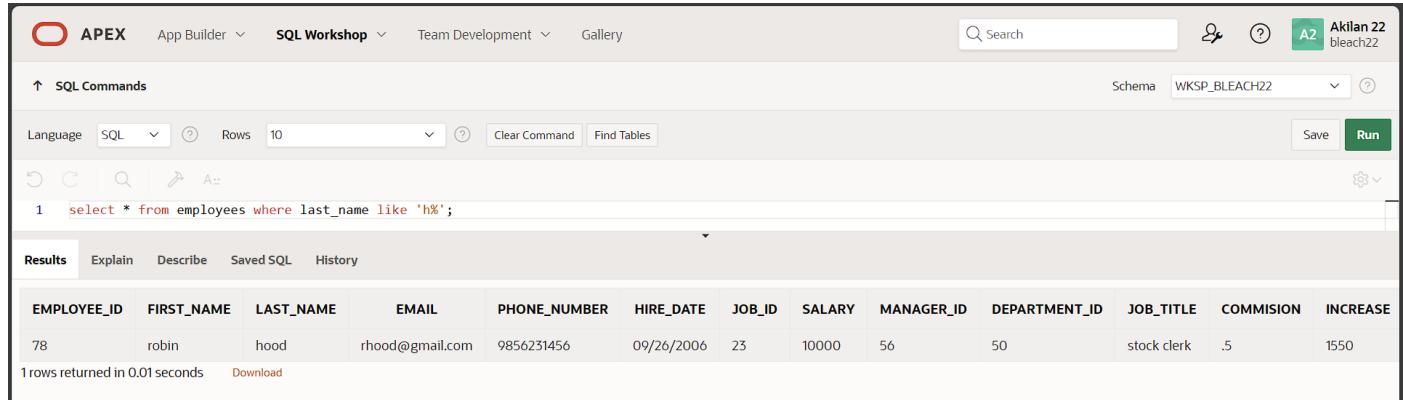
1 rows returned in 0.00 seconds

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

QUERY:

```
Select * from employees where last_name like 'H%';
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon, and a session identifier 'A2 Akilan 22 bleach22'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the query: 'select * from employees where last_name like 'h%''. The Results tab displays the following table:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	MANAGER_ID	DEPARTMENT_ID	JOB_TITLE	COMMISION	INCREASE
78	robin	hood	rhood@gmail.com	9856231456	09/26/2006	23	10000	56	50	stock clerk	.5	1550

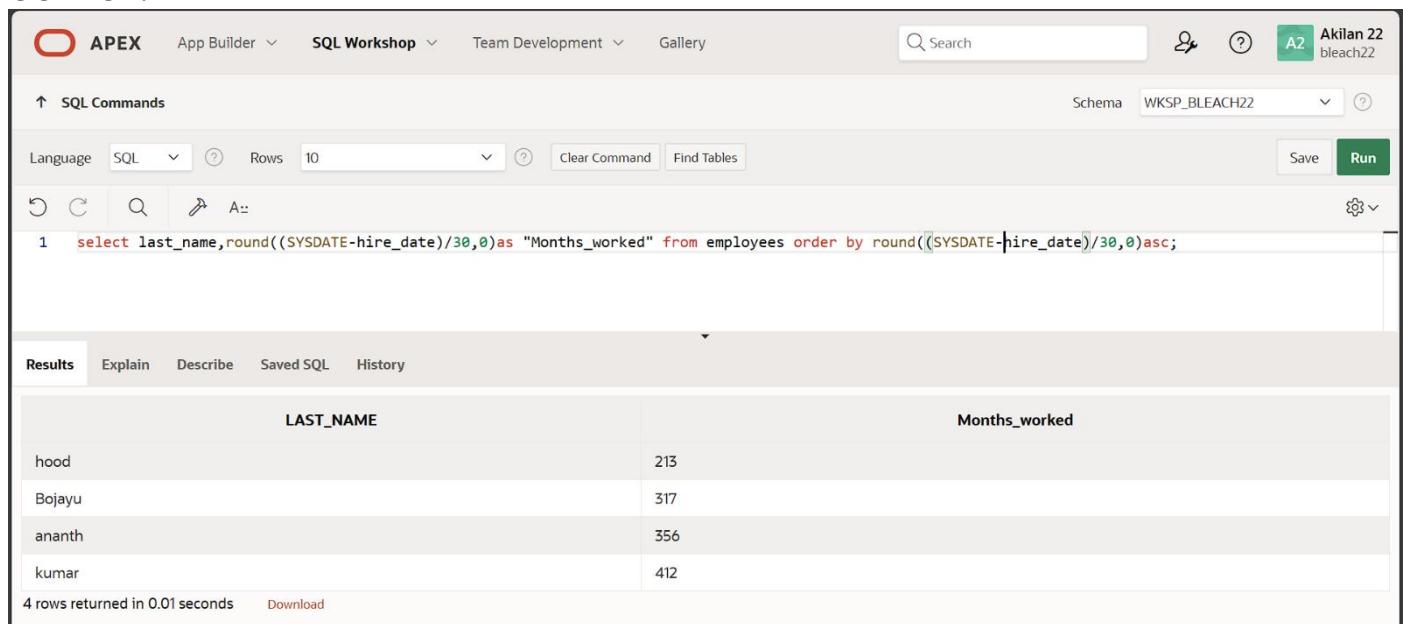
1 rows returned in 0.01 seconds [Download](#)

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

QUERY:

```
select last_name,round((sysdate-hire_date),30,0) as months_worked from employees order by round((sysdate-hire_date)/30,0) asc;
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon, and a session identifier 'A2 Akilan 22 bleach22'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the query: 'select last_name,round((SYSDATE-hire_date)/30,0) as "Months_worked" from employees order by round((SYSDATE-hire_date)/30,0) asc;'. The Results tab displays the following table:

LAST_NAME	Months_worked
hood	213
Bojayu	317
ananth	356
kumar	412

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

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

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

QUERY:

```
select last_name||'earns'||salary||'monthly but wants'||salary *3 as "dream salary" from employees;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. A search bar and user information (Akilan 22, bleach22) are also present. The main area is titled 'SQL Commands' with tabs for Language (set to SQL), Rows (set to 10), Clear Command, Find Tables, Schema (set to WKSP_BLEACH22), Save, and Run. The SQL command entered is:

```
1 select last_name ||'earns'|| salary||' monthly but wants'||salary*3 as "dream_salary" from employees;
```

The results tab is selected, showing the output:

	dream_salary
Bojayu	earns 20000 monthly but wants 60000
hood	earns 10000 monthly but wants 30000
ananth	earns 56000 monthly but wants 168000
kumar	earns 230000 monthly but wants 690000

4 rows returned in 0.01 seconds. There is a 'Download' link at the bottom.

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

QUERY:

```
select last_name, lpad(salary,15,'$1') as "salary" from employees;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. A search bar and user information (Akilan 22, bleach22) are also present. The main area is titled 'SQL Commands' with tabs for Language (set to SQL), Rows (set to 10), Clear Command, Find Tables, Schema (set to WKSP_BLEACH22), Save, and Run. The SQL command entered is:

```
1 select last_name, lpad(salary,15,'$')as "salary" from employees;
```

The results tab is selected, showing the output:

LAST_NAME	salary
Bojayu	\$\$\$\$\$\$\$\$\$\$20000
hood	\$\$\$\$\$\$\$\$\$\$10000
ananth	\$\$\$\$\$\$\$\$\$\$56000
kumar	\$\$\$\$\$\$\$\$\$\$230000

4 rows returned in 0.01 seconds. There is a 'Download' link at the bottom.

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

QUERY:

```
select last_name,hire_date,to_char(next_day(add_months(hire_date,6),'monday'),'fmday,"the"FMDD"of" Fmmmonth,yyyy') as review from employees;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is connected to Schema WKSP_BLEACH22. The SQL Commands tab is active, displaying the following SQL code:

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

The Results tab displays the output of the query:

LAST_NAME	HIRE_DATE	REVIEW
Bojayu	03/09/1998	Monday, The 14 of September, 1998
hood	09/26/2006	Monday, The 02 of April, 2007
ananth	12/26/1994	Monday, The 03 of July, 1995
kumar	05/02/1990	Monday, The 05 of November, 1990

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

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

QUERY:

```
Select last_name,hire_date to_char(hire_date,'day') as day from employees order by to_char(hire_date,'day');
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The user is connected to Schema WKSP_BLEACH22. The SQL Commands tab is active, displaying the following SQL code:

```
1 select last_name,hire_date,to_char(hire_date,'Day')as "Day" from employees order by to_char(hire_date,'Day');
```

The Results tab displays the output of the query:

LAST_NAME	HIRE_DATE	Day
Bojayu	03/09/1998	Monday
ananth	12/26/1994	Monday
hood	09/26/2006	Tuesday
kumar	05/02/1990	Wednesday

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

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

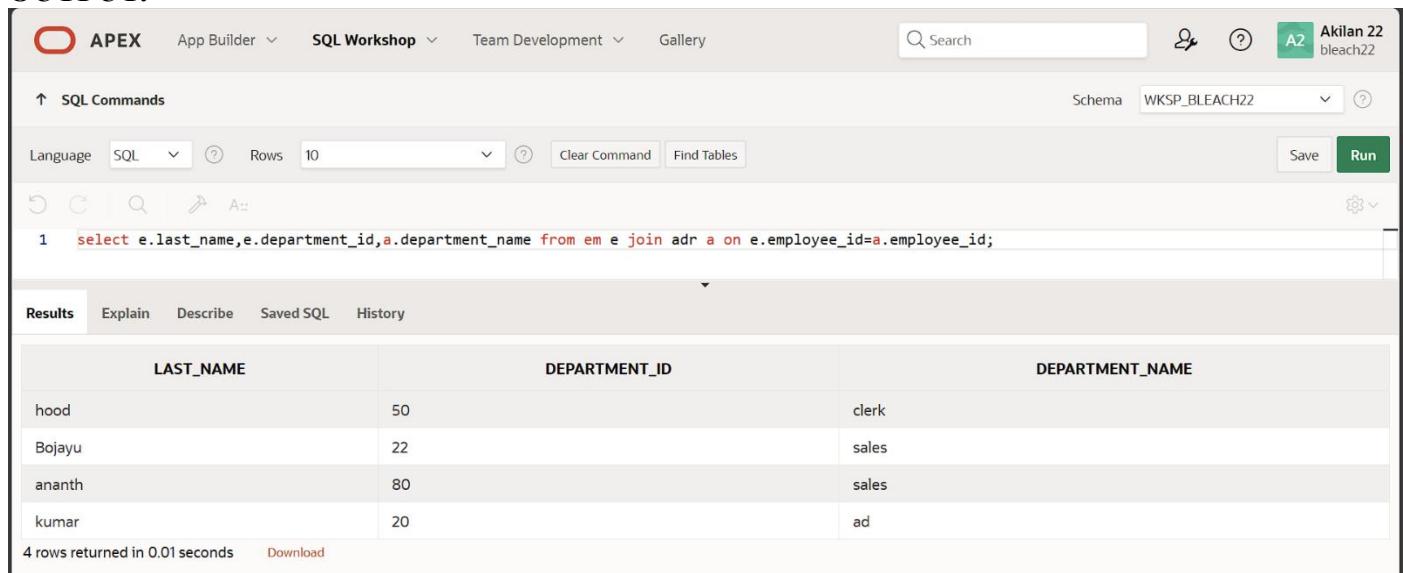
DISPLAYING DATA FROM MULTIPLE TABLES

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

QUERY:

```
select e.last_name,e.department_id,a.department_name from em e join adr a on e.employee_id=a.employee_id;
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown is set to WKSP_BLEACH22. The main area displays the following SQL command:

```
1 select e.last_name,e.department_id,a.department_name from em e join adr a on e.employee_id=a.employee_id;
```

The results section shows the output of the query:

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
hood	50	clerk
Bojayu	22	sales
ananth	80	sales
kumar	20	ad

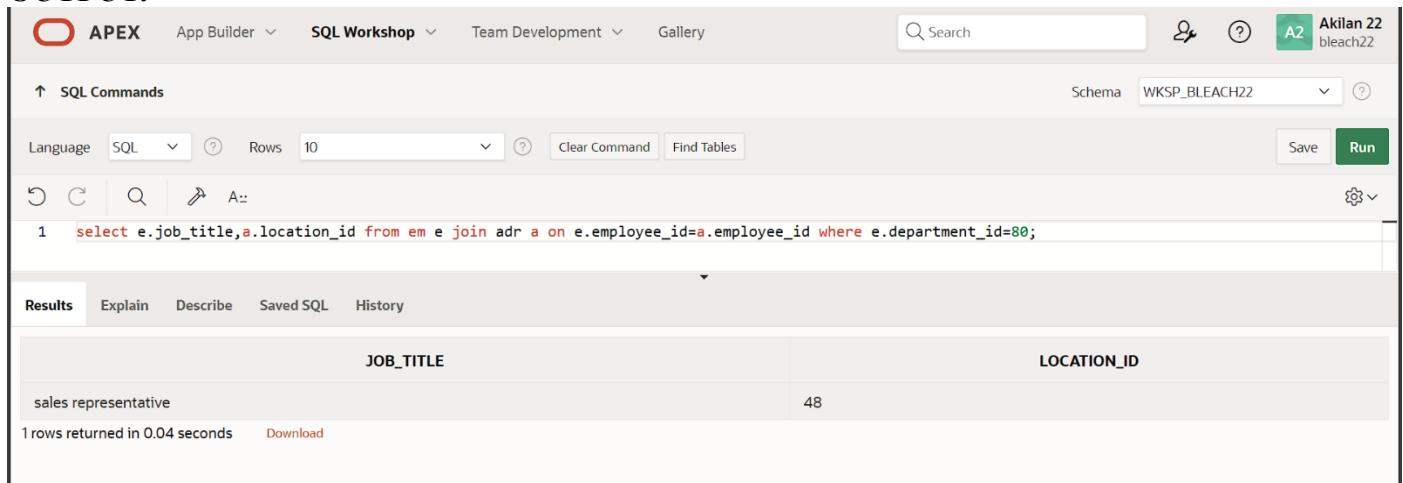
4 rows returned in 0.01 seconds

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

QUERY:

```
select e.job_title,a.location_id from em e join adr a on e.employee_id=a.employee_id where e.department_id=80;
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown is set to WKSP_BLEACH22. The main area displays the following SQL command:

```
1 select e.job_title,a.location_id from em e join adr a on e.employee_id=a.employee_id where e.department_id=80;
```

The results section shows the output of the query:

JOB_TITLE	LOCATION_ID
sales representative	48

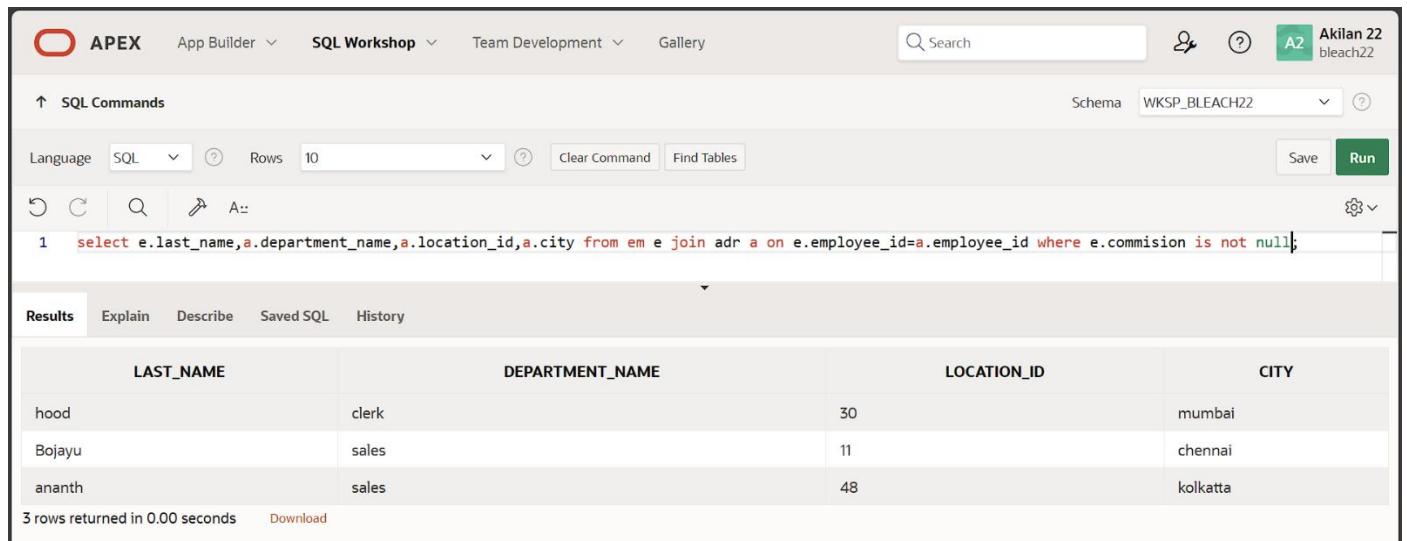
1 rows returned in 0.04 seconds

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

QUERY:

Select e.last_name,a.department_name,a.location_id,a.city from em e join adr a on employee_id where e.commission is not null

OUTPUT:



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

```
1 select e.last_name,a.department_name,a.location_id,a.city from em e join adr a on e.employee_id=a.employee_id where e.commission is not null;
```

The results table displays the following data:

LAST_NAME	DEPARTMENT_NAME	LOCATION_ID	CITY
hood	clerk	30	mumbai
Bojayu	sales	11	chennai
ananth	sales	48	kolkatta

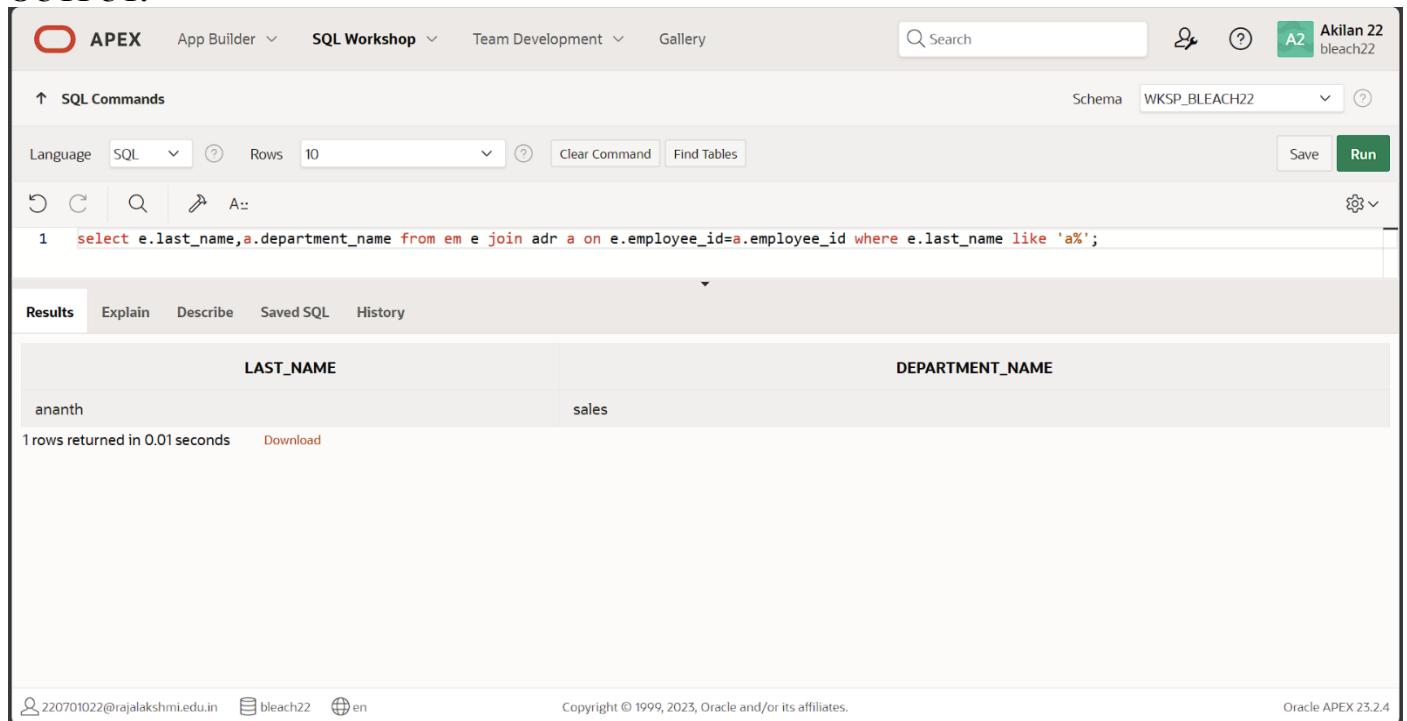
3 rows returned in 0.00 seconds

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

QUERY:

select a.last_name,a.department_name from em e join adr a on employee_id=a.employee_id where e.last_name like 'a%';

OUTPUT:



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

```
1 select e.last_name,a.department_name from em e join adr a on e.employee_id=a.employee_id where e.last_name like 'a%';
```

The results table displays the following data:

LAST_NAME	DEPARTMENT_NAME
ananth	sales

1 rows returned in 0.01 seconds

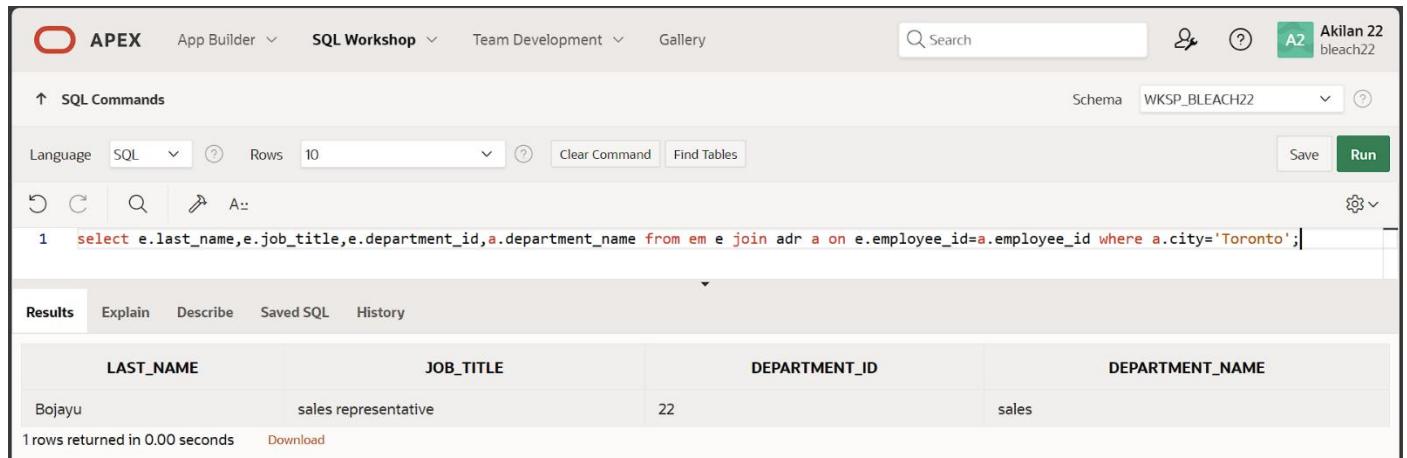
At the bottom of the page, there are footer links: 220701022@rajalakshmi.edu.in, bleach22, en, Copyright © 1999, 2023, Oracle and/or its affiliates., and Oracle APEX 23.2.4.

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

QUERY:

```
select e.last_name,e.job_title,e.department_id,a.department_id,a.department_name from em e join adr on e.employee_id=a.employee_id where a.city='Toronto';
```

OUTPUT:



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

```
1 select e.last_name,e.job_title,e.department_id,a.department_id,a.department_name from em e join adr a on e.employee_id=a.employee_id where a.city='Toronto';
```

The results table has four columns: LAST_NAME, JOB_TITLE, DEPARTMENT_ID, and DEPARTMENT_NAME. One row is returned:

LAST_NAME	JOB_TITLE	DEPARTMENT_ID	DEPARTMENT_NAME
Bojayu	sales representative	22	sales

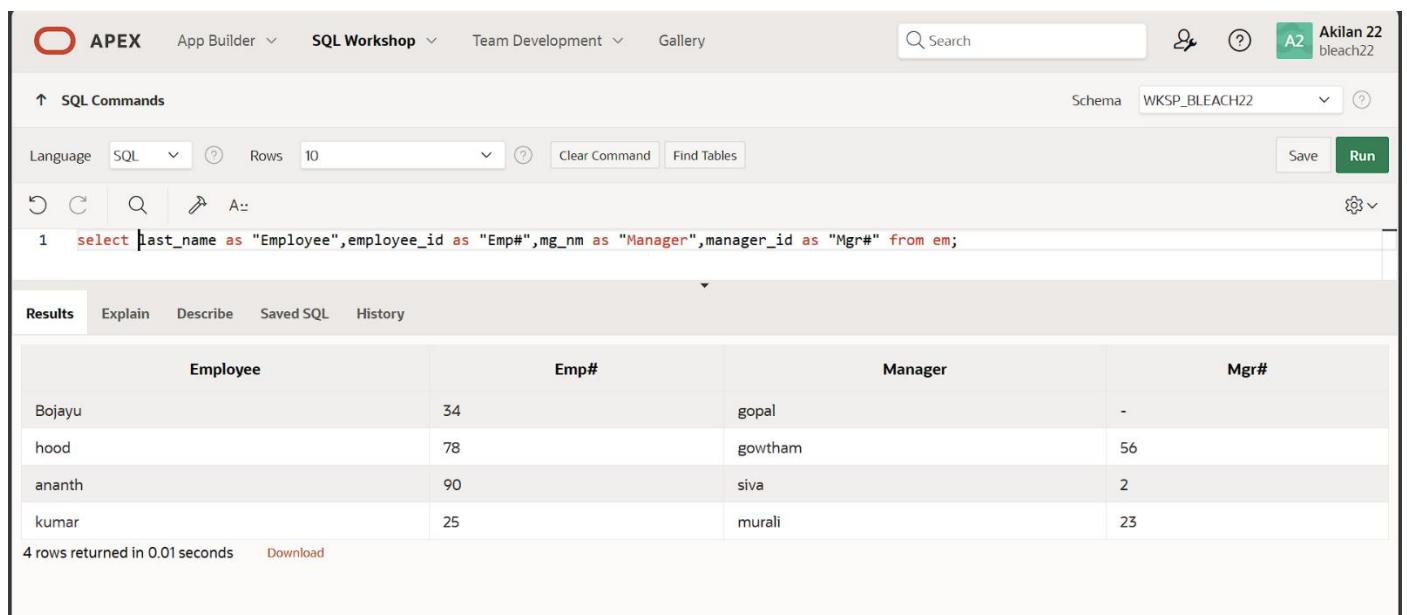
1 rows returned in 0.00 seconds

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

QUERY:

```
select last_name as "employee", employee_id as "emp#",mg_nm as "manager",manager_id as "mgr#" from em;
```

OUTPUT:



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

```
1 select last_name as "Employee",employee_id as "Emp#",mg_nm as "Manager",manager_id as "Mgr#" from em;
```

The results table has four columns: Employee, Emp#, Manager, and Mgr#. Four rows are returned:

Employee	Emp#	Manager	Mgr#
Bojayu	34	gopal	-
hood	78	gowtham	56
ananth	90	siva	2
kumar	25	murali	23

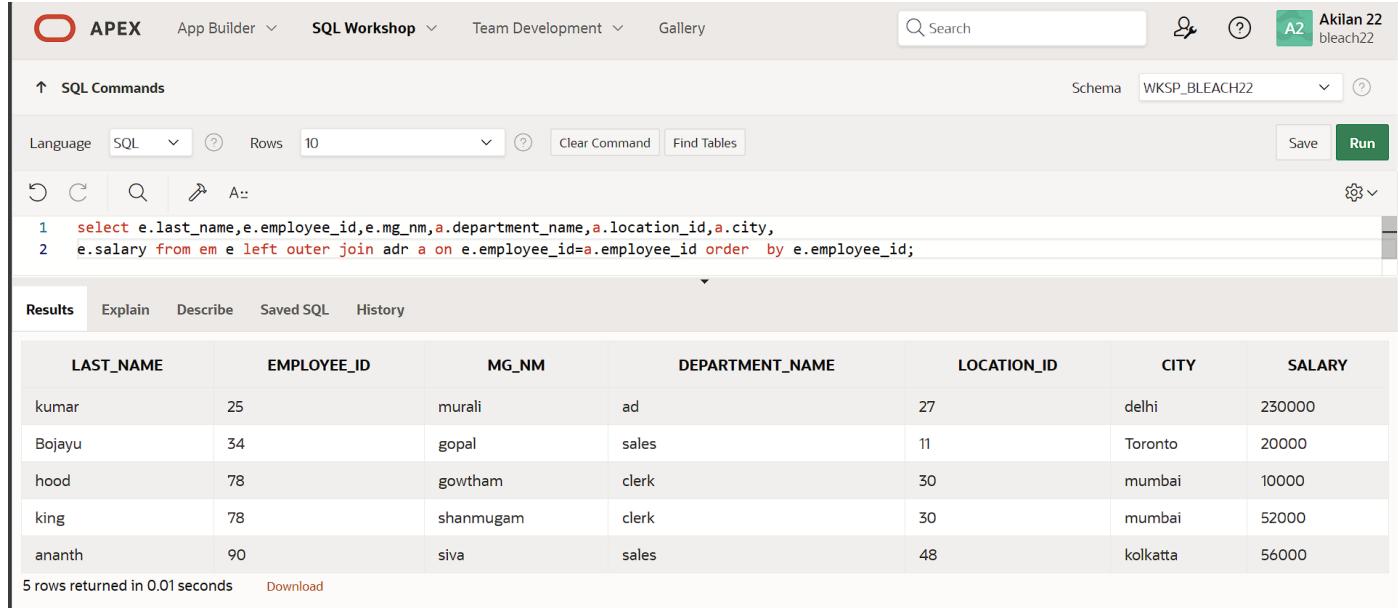
4 rows returned in 0.01 seconds

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

QUERY:

```
select e.last_name,e.employee_id,e.mg_nm,a.department_name,a.location_id,a.city,e.salary from em e  
Left outer join adr on e.employee_id=a.employee_id order by e.employee_id;
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. A search bar and user information (Akilan 22, bleach22) are also present. The SQL Commands tab is selected, showing the following SQL code:

```
1 select e.last_name,e.employee_id,e.mg_nm,a.department_name,a.location_id,a.city,  
2 e.salary from em e left outer join adr a on e.employee_id=a.employee_id order by e.employee_id;
```

The Results tab displays the query results in a grid format:

LAST_NAME	EMPLOYEE_ID	MG_NM	DEPARTMENT_NAME	LOCATION_ID	CITY	SALARY
kumar	25	murali	ad	27	delhi	230000
Bojayu	34	gopal	sales	11	Toronto	20000
hood	78	gowtham	clerk	30	mumbai	10000
king	78	shanmugam	clerk	30	mumbai	52000
ananth	90	siva	sales	48	kolkatta	56000

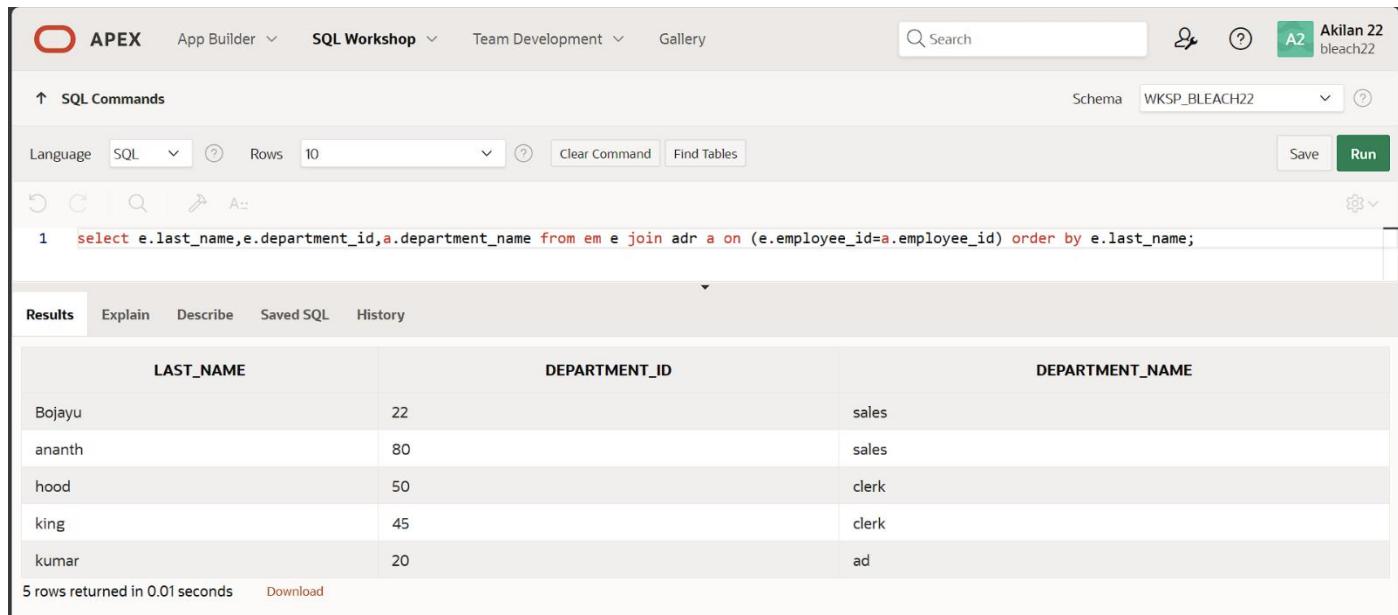
Below the table, it says "5 rows returned in 0.01 seconds" and there is a "Download" link.

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

QUERY:

```
Select e.last_name,e.department_id,a.department_name from em e join adr a on  
e.employee_id=a.employee_id order by e.last_name;
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. A search bar and user information (Akilan 22, bleach22) are also present. The SQL Commands tab is selected, showing the following SQL code:

```
1 select e.last_name,e.department_id,a.department_name from em e join adr a on (e.employee_id=a.employee_id) order by e.last_name;
```

The Results tab displays the query results in a grid format:

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Bojayu	22	sales
ananth	80	sales
hood	50	clerk
king	45	clerk
kumar	20	ad

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

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

QUERY:

```
Select e.last_name,e.hire_date from em join em davier on (davier.last_name='davier') where davier.hire_date<e.hire_date;
```

OUTPUT:

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

```
1 select e.last_name,e.hire_date from em e join em davier on(davier.last_name='davier')where davier.hire_date < e.hire_date;
```

The results table has two columns: LAST_NAME and HIRE_DATE. The data is:

LAST_NAME	HIRE_DATE
Bojayu	03/09/1998
hood	09/26/2006
ananth	12/26/1994
kumar	05/02/1990

4 rows returned in 0.01 seconds

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

QUERY:

```
Select e.last_name as "employee",e.hire_date as "emp hire",e.mg_nm as "manager",a.mg_hrdate as "mgr hire" from em e join adr a on e.employee_id=a.employee_id where e.hire_date<a.mg_hrdate;
```

OUTPUT:

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

```
1 select e.last_name as "employee",e.hire_date as "emp hire",e.mg_nm as "Manager",a.mg_hrdate as "mgr hire" from em e join adr a on
2 (e.employee_id=a.employee_id) where e.hire_date<a.mg_hrdate;
```

The results table has four columns: employee, emp hire, Manager, and mgr hire. The data is:

employee	emp hire	Manager	mgr hire
ananth	12/26/1994	siva	09/15/2001
king	02/03/1856	shanmugam	03/26/2000
davier	03/05/1974	bruce	06/29/1994

3 rows returned in 0.01 seconds

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

EX.NO:8

REG.NO:220701022

DATE:26/03/2024

AGGREGATING DATA USING GROUP FUNCTIONS

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

QUERY:

Select max(salary) as “highest”,min(salary) as “lowest”, sum(salary) as “sum”,avg(salary) as “average” from em;

OUTPUT:

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

QUERY:

Select max(salary) as “highest”,min(salary) as “lowest”, sum(salary) as “sum”,avg(salary) as “average” from em group by job title;

OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for App Builder, SQL Workshop, Team Development, and Gallery, along with a search bar and user profile information (A2 Akilan22 bleach22). The main workspace is titled "SQL Commands". The query entered is:

```
1 select max(salary) as "highest",min(salary) as "lowest",sum(salary) as "sum",avg(salary) as "average" from em group by job_title;
```

The results section displays the following data:

	highest	lowest	sum	average
10000	10000	10000	10000	10000
63000	63000	63000	63000	63000
56000	20000	76000	38000	
230000	52000	282000	141000	

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

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

QUERY:

```
select count(last_name),job_title from em group by job_title;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The query `select count(last_name),job_title from em group by job_title;` has been run. The results table has two columns: `COUNT(LAST_NAME)` and `JOB_TITLE`. The data is as follows:

COUNT(LAST_NAME)	JOB_TITLE
1	stock clerk
1	clerk
2	sales representative
2	advertisement

4 rows returned in 0.01 seconds

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

QUERY:

```
Select count(manager_id) as "number of managers" from em;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The query `select count(manager_id) as "number of managers" from em;` has been run. The results table has one column labeled `number of managers`. The data is as follows:

number of managers
5

1 rows returned in 0.00 seconds

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

QUERY:

```
Select max(salary)-min(salary) as "difference" from em;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The query `select max(salary)-min(salary) as "difference" from em;` has been run. The results table has one column labeled `difference`. The data is as follows:

difference
220000

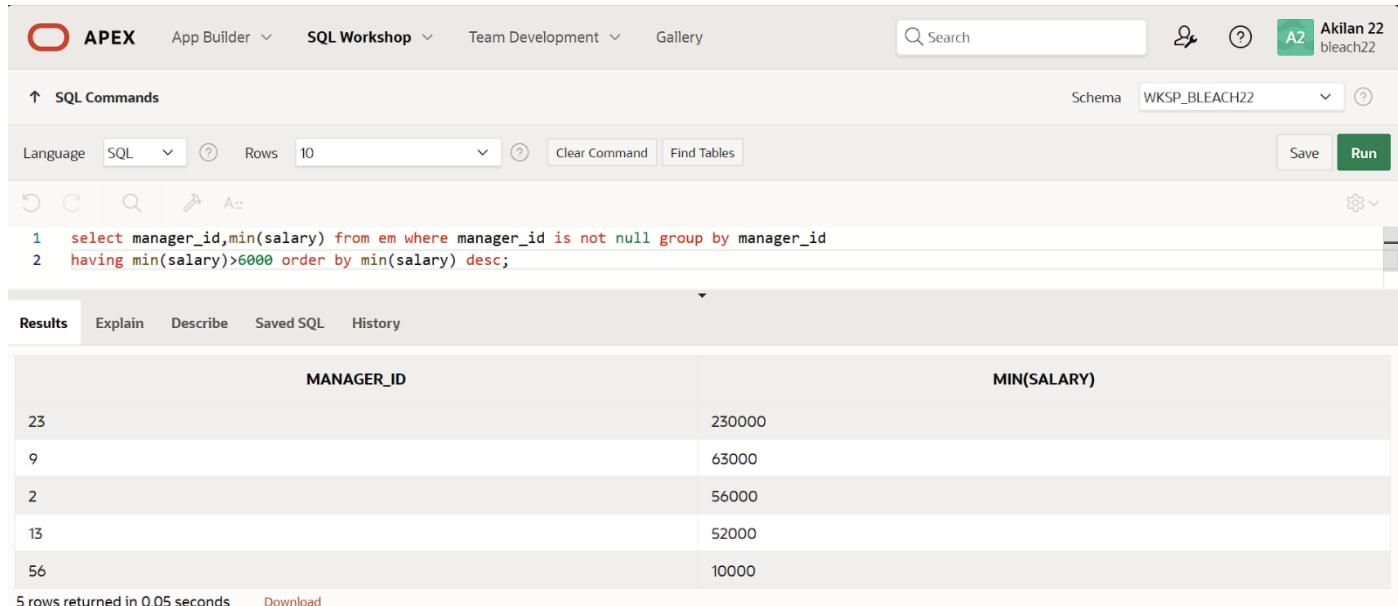
1 rows returned in 0.00 seconds

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

QUERY:

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

OUTPUT:



MANAGER_ID	MIN(SALARY)
23	230000
9	63000
2	56000
13	52000
56	10000

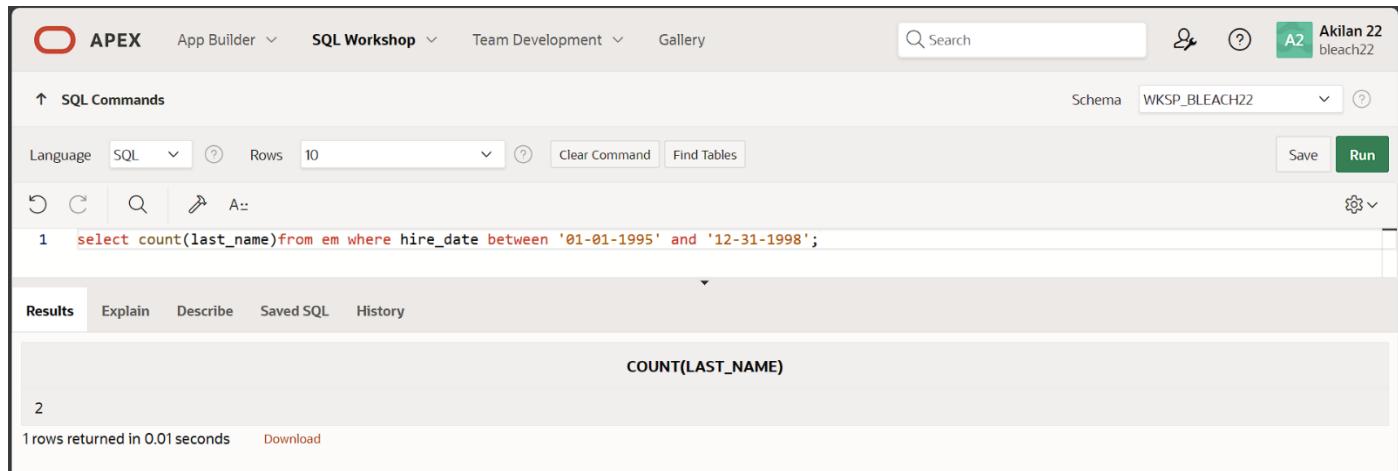
5 rows returned in 0.05 seconds [Download](#)

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

QUERY:

```
Select count(last_name) from em where hire_date between '01-01-1975' and '12-31-1998';
```

OUTPUT:



COUNT(LAST_NAME)
2

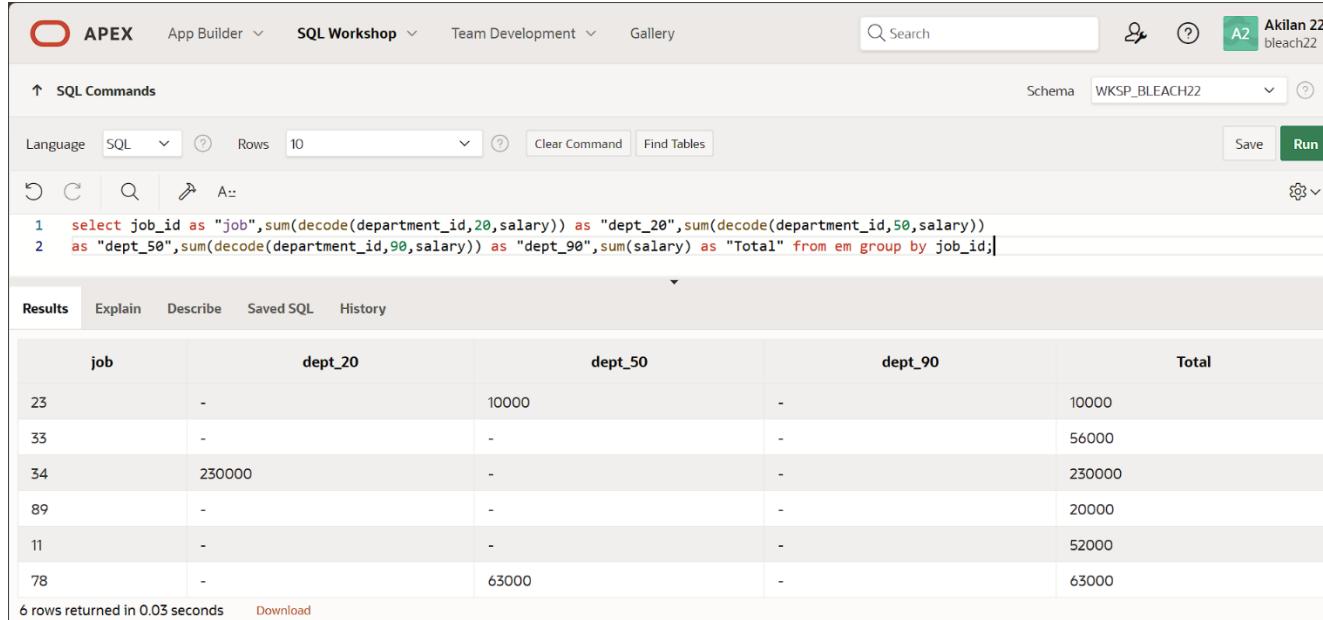
1 rows returned in 0.01 seconds [Download](#)

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

QUERY:

```
select job_id as "job",sum(decode(department_id,20,salary)) as  
"dept_20",sum(decode(department_id,50,salary)) as "dept_50",sum(decode(department_id,90,salary)) as  
"dept_90",sum(salary) as "total" from em group by job_id;
```

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_BLEACH22. The main area displays the SQL command and its results. The SQL command is:

```
1 select job_id as "job",sum(decode(department_id,20,salary)) as "dept_20",sum(decode(department_id,50,salary))  
2 as "dept_50",sum(decode(department_id,90,salary)) as "dept_90",sum(salary) as "Total" from em group by job_id;
```

The results section shows the output of the query:

job	dept_20	dept_50	dept_90	Total
23	-	10000	-	10000
33	-	-	-	56000
34	230000	-	-	230000
89	-	-	-	20000
11	-	-	-	52000
78	-	63000	-	63000

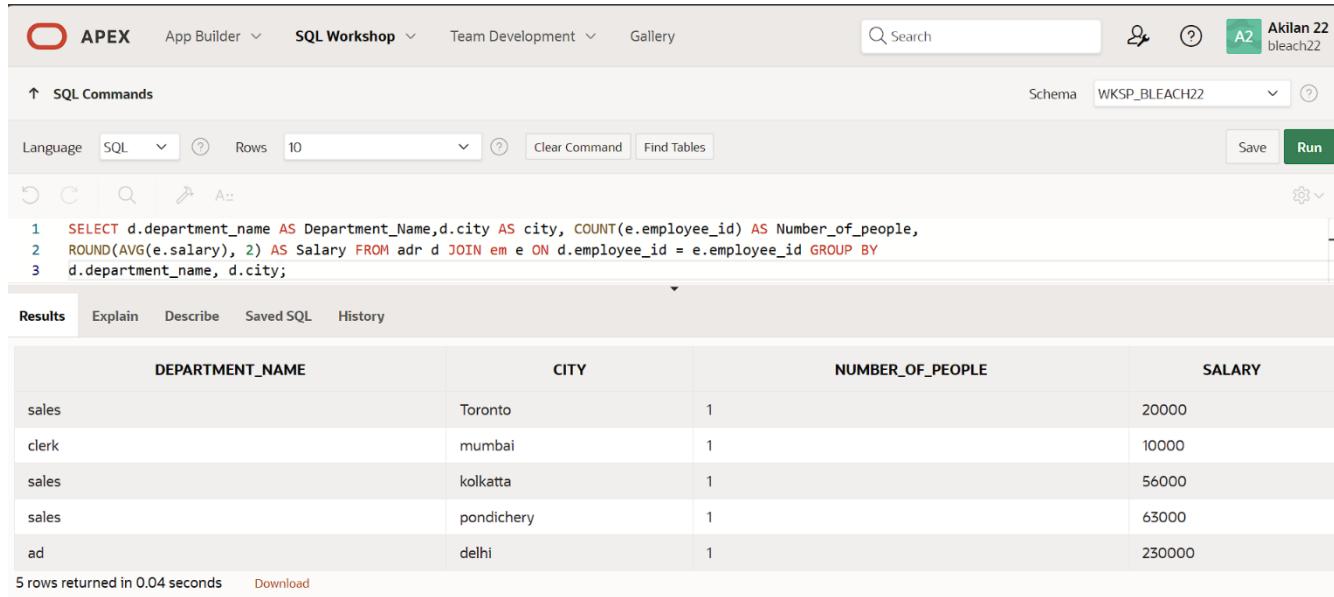
6 rows returned in 0.03 seconds [Download](#)

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

QUERY:

```
select d.department_name as department_name,a.city as city,count(e.employee_id) as number_of_people,  
Round(avg(e.salary),2) as salary from adr d join em e on d.employee_id=e.employee_id group by  
d.department_name,d.city;
```

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_BLEACH22. The main area displays the SQL command and its results. The SQL command is:

```
1 SELECT d.department_name AS Department_Name,d.city AS city, COUNT(e.employee_id) AS Number_of_People,  
2 ROUND(AVG(e.salary), 2) AS Salary FROM adr d JOIN em e ON d.employee_id = e.employee_id GROUP BY  
3 d.department_name, d.city;
```

The results section shows the output of the query:

DEPARTMENT_NAME	CITY	NUMBER_OF_PEOPLE	SALARY
sales	Toronto	1	20000
clerk	mumbai	1	10000
sales	kolkatta	1	56000
sales	pondichery	1	63000
ad	delhi	1	230000

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

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

SUB QUERIES

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

QUERY:

```
Select last_name,hire_date from em where department_id=(select department_id from em where last_name='hood') and last_name not in ('hood');
```

OUTPUT:

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

```
1 select last_name,hire_date from em where department_id=(select department_id from em where last_name='hood') and last_name not in ('hood');
```

The results table has two columns: LAST_NAME and HIRE_DATE. One row is returned:

LAST_NAME	HIRE_DATE
davier	03/05/1974

1 rows returned in 0.02 seconds

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

QUERY:

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

OUTPUT:

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

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

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

EMPLOYEE_ID	LAST_NAME	SALARY
25	kannan	230000

1 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 employee_id, last_name from em where employee_id=(select employee_id from em where last_name like '%u%');
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'Akilan 22' and the schema 'WKSP_BLEACH22'. The main area is titled 'SQL Commands' with a search bar and a 'Run' button. Below it, there are icons for Undo, Redo, Find, and Copy. The SQL command entered is:

```
1 select employee_id, last_name from em where employee_id=(select employee_id from em where last_name like '%u%');
```

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

EMPLOYEE_ID	LAST_NAME
34	Bojayu

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

QUERY:

```
Select last_name, department_id, job_id from em where employee_id=(select employee_id from adr where location_id=1700);
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'Akilan 22' and the schema 'WKSP_BLEACH22'. The main area is titled 'SQL Commands' with a search bar and a 'Run' button. Below it, there are icons for Undo, Redo, Find, and Copy. The SQL command entered is:

```
1 select last_name, department_id, job_id from em where employee_id=(select employee_id from adr where location_id=1700);
```

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

LAST_NAME	DEPARTMENT_ID	JOB_ID
hood	50	23

1 rows returned in 0.01 seconds Download

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

QUERY:

```
Select last_name, salary from em where mg_nm='king';
```

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 'Akilan 22' and the schema 'WKSP_BLEACH22'. The main area is titled 'SQL Commands' with a search bar and a 'Run' button. Below it, there are icons for Undo, Redo, Find, and Copy. The SQL command entered is:

```
1 select last_name, salary from em where mg_nm='king';
```

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

LAST_NAME	SALARY
anand	56000

1 rows returned in 0.01 seconds Download

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

QUERY:

```
select e.department_id,e.last_name,e.job_id from em e join adr a on e.employee_id=a.employee_id where a.department_name='executive';
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'Akilan 22' and 'bleach22'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP_BLEACH22'. The SQL editor contains the following query:

```
1 select e.department_id,e.last_name,e.job_id from em e join adr a on e.employee_id=a.employee_id where a.department_name='executive';
```

The results tab is selected, displaying the following data:

DEPARTMENT_ID	LAST_NAME	JOB_ID
50	hood	23
50	davier	78

Below the table, it says '2 rows returned in 0.02 seconds' and has a 'Download' link.

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

QUERY:

```
select employee_id,last_name,salary from em where salary > (select avg(salary) from em) and employee_id=(select employee_id from em where last_name like '%u%');
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'Akilan 22' and 'bleach22'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP_BLEACH22'. The SQL editor contains the following query:

```
1 select employee_id,last_name,salary from em where salary > (select avg(salary) from em) and
2 | employee_id=(select employee_id from em where last_name like '%u%');|
```

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

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

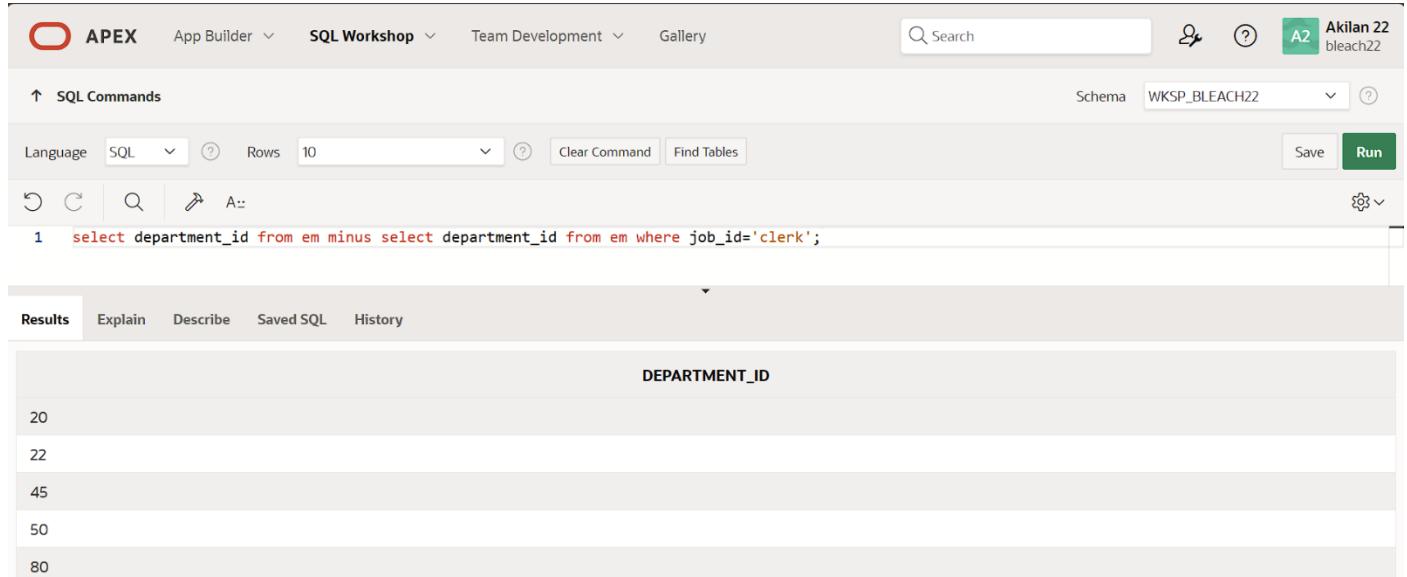
USING THE SET OPERATORS

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

QUERY:

```
Select department_id from em minus select department_id from em where job_id='clerk';
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a user icon, a help icon, and a session identifier 'A2 Akilan 22 bleach22'. Below the navigation is a search bar and a schema dropdown set to 'WKSP_BLEACH22'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the following code:

```
1 select department_id from em minus select department_id from em where job_id='clerk';
```

The Results tab displays the output:

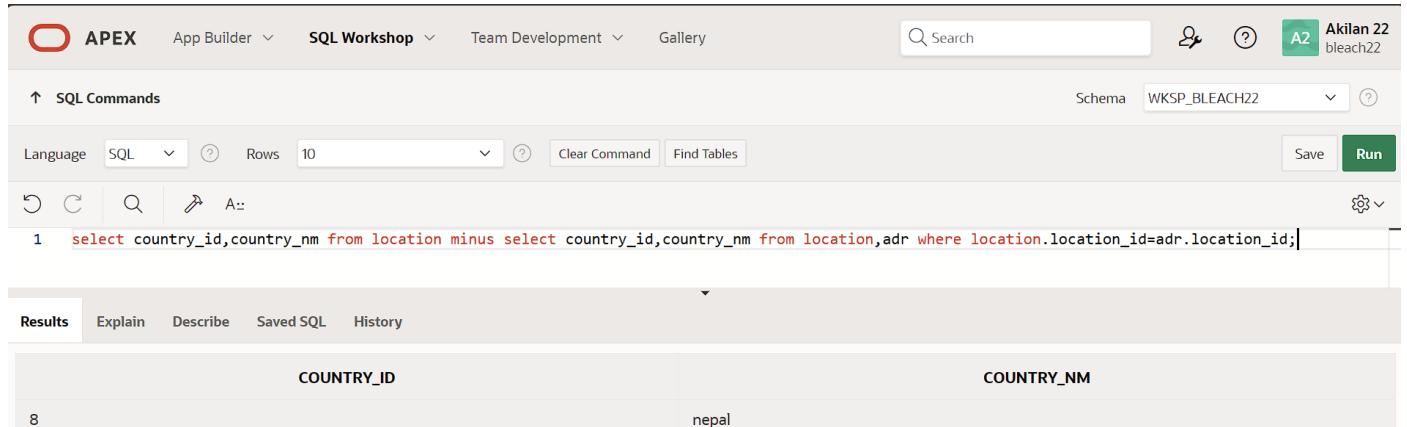
DEPARTMENT_ID
20
22
45
50
80

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

QUERY:

```
Select country_id,country_nm from location minus select country_id,country_nm from location,adr where Location.location_id=adr.location_id;
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface, identical to the previous one but with a different query. The top navigation bar and session information are the same. The SQL Commands tab contains the following code:

```
1 select country_id,country_nm from location minus select country_id,country_nm from location,adr where location.location_id=adr.location_id;
```

The Results tab displays the output:

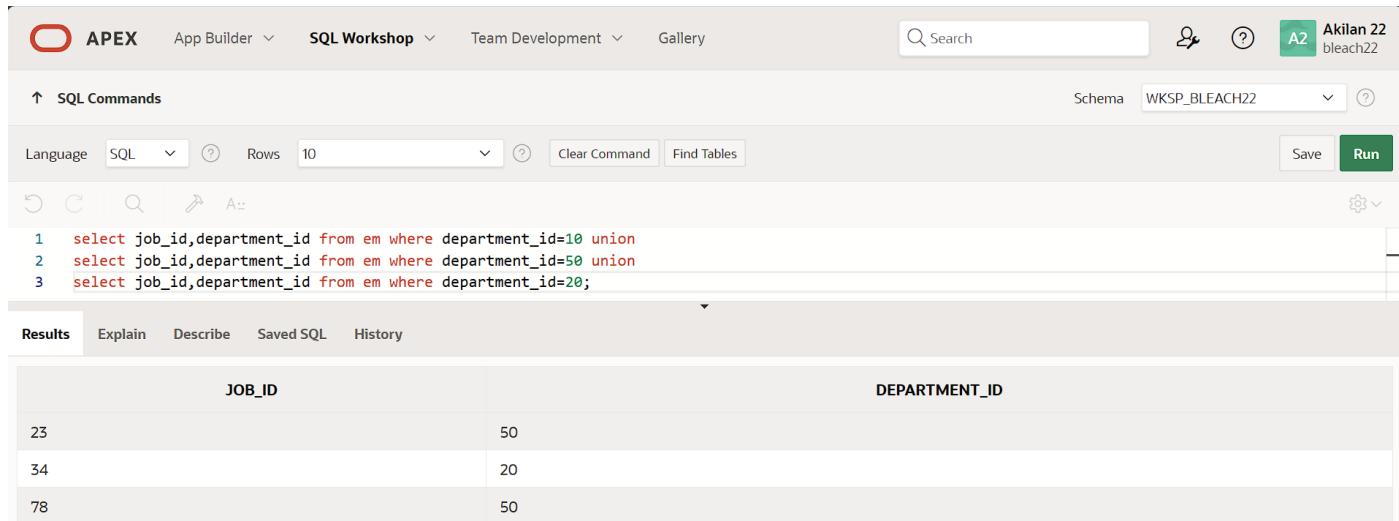
COUNTRY_ID	COUNTRY_NM
8	nepal

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

QUERY:

```
select job_id,department_id from em where department_id=10 union select job_id,department_id from em where department_id=50 union select job_id,department_id=20;
```

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 'Akilan 22' and a workspace named 'bleach22'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP_BLEACH22'. The SQL editor contains the following code:

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

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

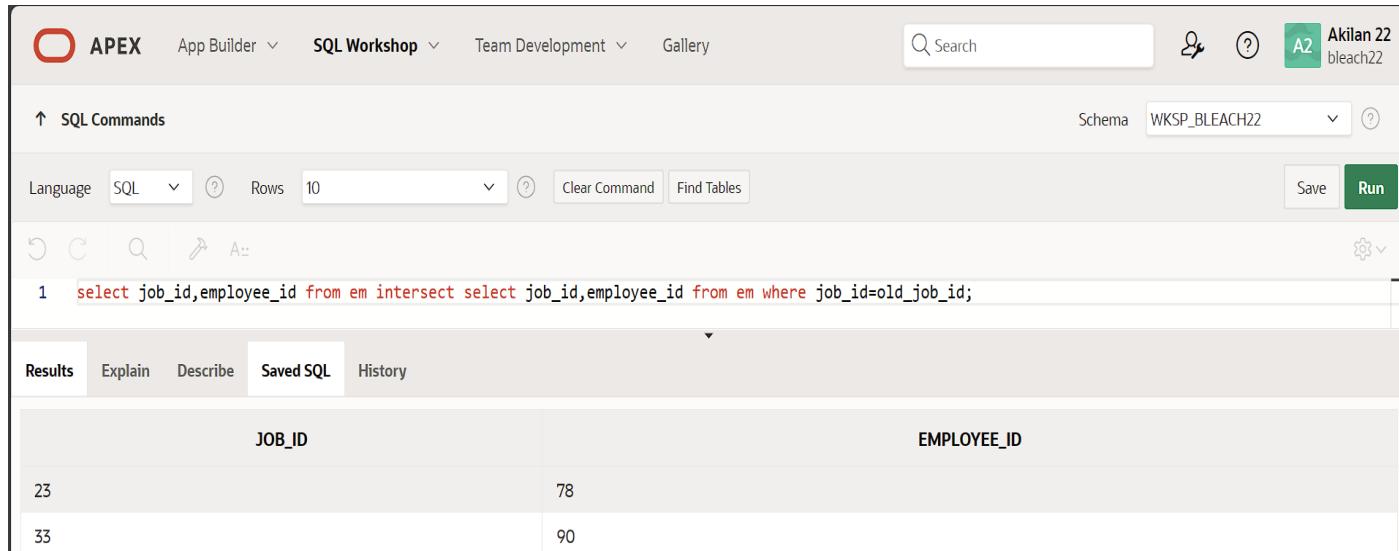
JOB_ID	DEPARTMENT_ID
23	50
34	20
78	50

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

QUERY:

```
Select job_id,employee_id from em intersect select job_id,employee_id from em where job_id=old_job_id;
```

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 'Akilan 22' and a workspace named 'bleach22'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP_BLEACH22'. The SQL editor contains the following code:

```
1 select job_id,employee_id from em intersect select job_id,employee_id from em where job_id=old_job_id;
```

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

JOB_ID	EMPLOYEE_ID
23	78
33	90

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

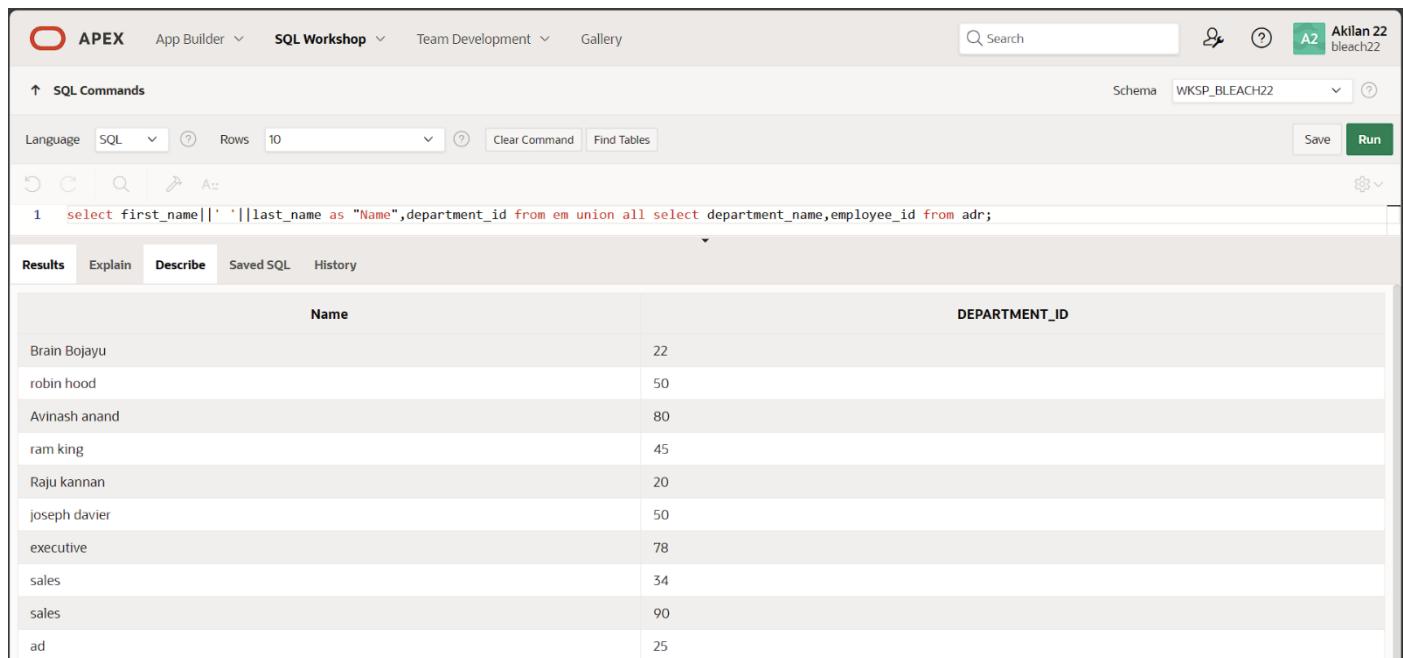
- Last name and department ID of all the employees from the EMPLOYEES table, regardless of whether or not they belong to a department.

- 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 first_name||' '||last_name as "name",department_id from em union all select  
department_name,employee_id from adr;
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. On the right, there's a search bar, a user icon (A2 Akilan 22), and a schema dropdown set to WKSP_BLEACH22. Below the navigation is a toolbar with icons for Undo, Redo, Find, and Run. The main area is titled 'SQL Commands' and contains a code editor with the following SQL query:

```
1 select first_name||' '||last_name as "Name",department_id from em union all select department_name,employee_id from adr;
```

Below the code editor is a results grid. The 'Results' tab is selected, showing the output of the query:

Name	DEPARTMENT_ID
Brain Bojayu	22
robin hood	50
Avinash anand	80
ram king	45
Raju kannan	20
joseph davier	50
executive	78
sales	34
sales	90
ad	25

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

CREATING VIEWS

Find the Solution for the following:

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 employee_id, last_name as "employee", department_id from em;

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. A search bar is at the top right. On the right side, there's a user profile for 'Akilan 22' and a workspace identifier 'A2 WKSP_BLEACH22'. The main area is titled 'SQL Commands'. It has dropdown menus for 'Language' (set to 'SQL') and 'Rows' (set to 10), along with 'Clear Command' and 'Find Tables' buttons. On the far right are 'Save' and 'Run' buttons. The SQL command input field contains the code: '1 create view employee_vu as select employee_id, last_name as "employee", department_id from em;'. Below the input field, the status message 'View created.' is displayed. At the bottom, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History', with 'Results' being the active tab.

2.Display the contents of the EMPLOYEES_VU view.

QUERY:

Select * from employee_vu;

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface again. The top navigation bar and user profile are identical to the previous screenshot. The 'SQL Commands' section now contains the query: '1 select * from employee_vu;'. The 'Results' tab is active, displaying the output of the query. The output is a table with three columns: 'EMPLOYEE_ID', 'employee', and 'DEPARTMENT_ID'. The data rows are:

EMPLOYEE_ID	employee	DEPARTMENT_ID
34	Bojayu	22
78	hood	50
90	anand	80
37	king	45
25	kannan	20
67	davier	50

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

QUERY:

```
select view_name,text from user_views;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The command entered is 'select view_name, text from user_views;'. The results pane displays a single row with 'EMPLOYEE_VU' in the VIEW_NAME column and the corresponding SQL text in the TEXT column.

VIEW_NAME	TEXT
EMPLOYEE_VU	select employee_id, last_name as "employee", department_id from em

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

QUERY:

```
select employee,department_id from employee_vu;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The command entered is 'select employee, department_id from employee_vu;'. The results pane displays a table with two columns: 'EMPLOYEE' and 'DEPARTMENT_ID', listing various employees and their respective department IDs.

EMPLOYEE	DEPARTMENT_ID
Bojayu	22
hood	80
anand	80
king	45
kannan	20
davier	50

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

QUERY:

```
Create view dept50 as select employee_id empno,last_name employee,department_id deptno from em where Department_id=50;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The command entered is 'create view dept50 as select employee_id empno,last_name employee,department_id deptno from em where department_id=50;'. The results pane shows the command was successfully executed.

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

QUERY:

Describe dept50;

Select * from dept50;

OUTPUT:

The screenshot shows two separate sessions in the Oracle SQL Workshop. The top session displays the results of a SELECT query:

empno	employee	deptno
78	hood	50
67	davier	50

The bottom session shows the output of the DESCRIBE command for the DEPT50 view:

Object Type	Object	Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
VIEW	DEPT50	DEPT50	empno	NUMBER	-	6	0	-	✓	-	-
		employee	VARCHAR2	25	-	-	-	✓	-	-	-
		deptno	NUMBER	-	4	0	-	✓	-	-	-

7. Attempt to reassign Matos to department 80.

QUERY:

Update dept50 set deptno=80 where employee='matos';

OUTPUT:

The screenshot shows the result of the UPDATE command:

1 row(s) updated.

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

QUERY:

Create or replace view salary_vu as select e.last_name as "employee",d.department_name as "department",e.salary as "salary" from em e,adr d where e.employee_id=d.employee_id;

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A schema dropdown shows 'WKSP_BLEACH22'. Below it, a language dropdown is set to 'SQL', and a 'Run' button is visible. The code entered is:

```
1 create or replace view salary_vu as
2 select e.last_name as "employee",d.department_name as "department",e.salary as "salary" from em e,adr d where e.employee_id=d.employee_id;
```

Below the code, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The status message 'View created.' is displayed at the bottom.

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

INTRO TO CONSTRAINTS:NOT NULL AND UNIQUE

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

QUERY:

```
Create table global_location(id number(10),name varchar(25) not null,date_opened number(10),address Varchar(30),city number(10),zip_code number(10) not null,phone_number number(10) not null,email varchar(25)not null,manager_id number(10) not null,em_contact number(10) not null);
```

OUTPUT:

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

```
1 create table global_location(id number(10),name varchar(25)not null,date_opened number(10),address Varchar(30),city number(10),zip_code number(10) not null,
2 phone number(10) not null,email varchar(25) not null,manager_id number(10)not null,em_contact number(10)not null);
```

The Results tab displays the output:

```
Table created.
```

Execution time: 0.04 seconds

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

QUERY:

```
Alter table global_location add constraint my_id_pk primary key(id);
```

OUTPUT:

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

```
1 alter table global_location add constraint my_id_pk primary key(id);
```

The Results tab displays the output:

```
Table altered.
```

Execution time: 0.10 seconds

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

QUERY:

```
describe global_location;
```

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 'Akilan 22' and the schema 'WKSP_BLEACH22'. The main area has tabs for SQL Commands, Results, Explain, Describe (selected), Saved SQL, and History. The SQL Commands tab contains the query '1 describe global_location;'. The Results tab displays the structure of the 'GLOBAL_LOCATION' table:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
GLOBAL_LOCATION	ID	NUMBER	-	10	0	1	-	-	-
	NAME	VARCHAR2	25	-	-	-	-	-	-
	DATE_OPENED	NUMBER	-	10	0	-	✓	-	-
	ADDRESS	VARCHAR2	30	-	-	-	✓	-	-
	CITY	VARCHAR2	10	-	-	-	✓	-	-
	ZIP_CODE	NUMBER	-	10	0	-	-	-	-
	PHONE	NUMBER	-	10	0	-	-	-	-
	EMAIL	VARCHAR2	25	-	-	-	-	-	-
	MANAGER_ID	NUMBER	-	10	0	-	-	-	-
	EM_CONTACT	NUMBER	-	10	0	-	-	-	-

4. 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 gff_location(id number(4),loc_name varchar(20) not null,date_opened date,address varchar(30),
City varchar(20),zip_postal varchar(20) not null,phone varchar(25)not null,email varchar(80) not
null,manager_id number(4) not null,contact varchar(40) not null,constraint uc unique(id,email));
```

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 'Akilan 22' and the schema 'WKSP_BLEACH22'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the CREATE TABLE statement:

```
1 create table gff_location(id number(4),loc_name varchar(20) not null,date_opened date,address varchar(30),city varchar(20),zip_postal varchar(20) not null,
2 phone varchar(15) not null,email varchar(80) not null,manager_id number(4) not null,contact varchar(40) not null,constraint uc unique(id,email));
```

The Results tab shows the message 'Table created.'

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

CREATING VIEWS

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

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.

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

2. SELECT * FROM view_d_songs. What was returned?

Results Explain Describe Saved SQL History

ID	Song Title	ARTIST
47	Hurrah for Today	The Jubilant Trio
49	Lets Celebrate	The Celebrants

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

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

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

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.

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

5. It is company policy that only upper-level management be allowed access to individual employee salaries. The department managers, however, need to know the minimum, maximum, and average salaries, grouped by department. Use the Oracle database to prepare a view that displays the needed information for department managers.

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

DML Operations and Views

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

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

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

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

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

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

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

```
SELECT * FROM view_copy_d_songs;
```

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

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

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

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

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

```
SELECT * FROM read_copy_d_cds;
```

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

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

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

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

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

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

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

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

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

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

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

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

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

DELETE, INSERT, MODIFY restricted if it contains:

Group functions
GROUP BY CLAUSE
DISTINCT
pseudocolumn ROWNUM Keyword

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

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

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

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

Managing Views

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

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

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

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

ORA-00942: table or view does not exist

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

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

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

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

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

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

Indexes and Synonyms

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

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

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

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

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

3. When will an index be created automatically?

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

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

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

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

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

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

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

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

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

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

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

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

```
CREATE SYNONYM dj_tracks2 FOR d_track_listings;
```

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

10. Drop the synonym that you created in question

```
DROP SYNONYM dj_tracks2;
```

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

OTHER DATABASE OBJECTS

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

QUERY:

Create sequence dept_id_seq start with 200 increment by 10 maxvalue 1000;

OUTPUT:



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

```
1 create sequence dept_id_seq start with 200 increment by 10 maxvalue 1000;
```

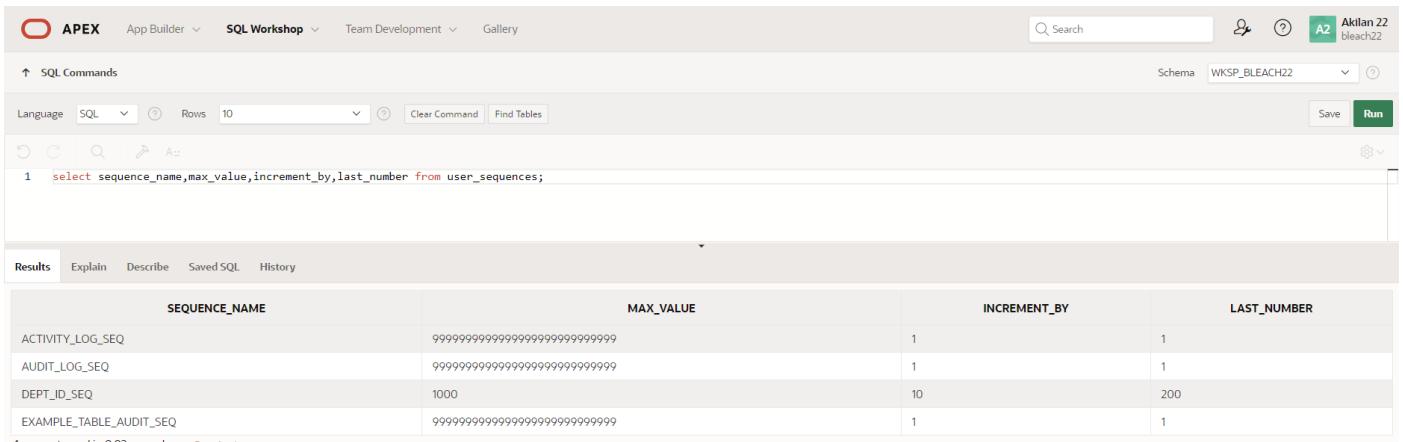
The results pane shows the message "Sequence created."

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

QUERY:

Select sequence_name,max_value,increment_by,last_number from user_sequences;

OUTPUT:



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

```
1 select sequence_name,max_value,increment_by,last_number from user_sequences;
```

The results pane displays a table with the following data:

SEQUENCE_NAME	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
ACTIVITY_LOG_SEQ	99999999999999999999999999999999	1	1
AUDIT_LOG_SEQ	99999999999999999999999999999999	1	1
DEPT_ID_SEQ	1000	10	200
EXAMPLE_TABLE_AUDIT_SEQ	99999999999999999999999999999999	1	1

4 rows returned in 0.02 seconds

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

QUERY:

Insert into dept values(dept_id_seq.nextval,'education'),insert into dept values(dept_id_seq.nextval,'administration');

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The top navigation bar shows "APEX", "App Builder", "SQL Workshop", "Team Development", and "Gallery". The right side of the header includes a search bar, user information ("A2 Akilan 22 bleach22"), and a "Run" button. The main area is titled "SQL Commands" with a "Results" tab selected. The SQL editor contains the following code:

```
1 insert into dept values(dept_id_seq.nextval,'education'),insert into dept values(dept_id_seq.nextval,'administration');
```

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

QUERY:

```
Create index emp_dept_id_idx on employees(dept_id);
```

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The top navigation bar shows "APEX", "App Builder", "SQL Workshop", "Team Development", and "Gallery". The right side of the header includes a search bar, user information ("A2 Akilan 22 bleach22"), and a "Run" button. The main area is titled "SQL Commands" with a "Results" tab selected. The SQL editor contains the following code:

```
1 create index emp_dept_id_idx on employees(dept_id);
```

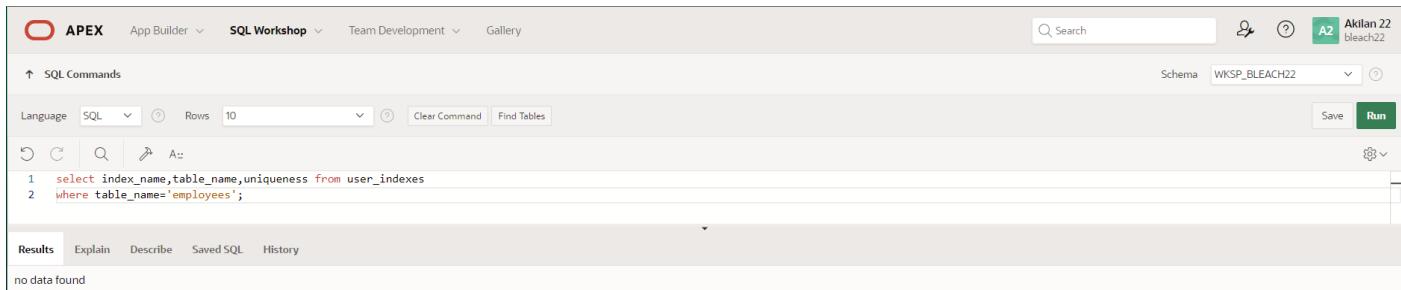
The results pane at the bottom shows the message "Index created."

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

QUERY:

```
Select index_name,table_name,uniqueness from user_indexes where table_name='employees';
```

OUTPUT:



A screenshot of the Oracle SQL Workshop interface. The top navigation bar shows "APEX", "App Builder", "SQL Workshop", "Team Development", and "Gallery". The right side of the header includes a search bar, user information ("A2 Akilan 22 bleach22"), and a "Run" button. The main area is titled "SQL Commands" with a "Results" tab selected. The SQL editor contains the following code:

```
1 select index_name,table_name,uniqueness from user_indexes
2 where table_name='employees';
```

The results pane at the bottom shows the message "no data found".

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

CONTROLLING USER ACCESS

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

The CREATE SESSION system privilege

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

The CREATE TABLE privilege

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

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

4. You are the DBA. You are creating many users who require the same system privileges.

What should you use to make your job easier?

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

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

The ALTER USER statement

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

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

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

7. Query all the rows in your DEPARTMENTS table.

SELECT * FROM departments;

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

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

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

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

```
SELECT table_name FROM user_tables;
```

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

Team 1 revokes the privilege.

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

Team 2 revokes the privilege.

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

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

Team 1 executes this INSERT statement.

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

Team 2 executes this INSERT statement.

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

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

CONTROLLING USER ACCESS

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, the following PL/SQL block is written:

```

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

```

In the Results pane, the output is displayed:

```

Incentive = 27600
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
    "MyVariable" VARCHAR2(50);
    myvariable2 VARCHAR2(50);
BEGIN
    "MyVariable" := 'Hello, World!';
    myvariable2 := 'PL/SQL Programming';
    DBMS_OUTPUT.PUT_LINE('Quoted identifier: ' || "MyVariable");
    DBMS_OUTPUT.PUT_LINE('Non-quoted identifier: ' || myvariable2);
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;

```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. The main area is titled 'SQL Commands'. A code editor window contains the following PL/SQL block:

```
1 DECLARE
2   "MyVariable" VARCHAR2(50);
3   myvariable2 VARCHAR2(50);
4 BEGIN
5   "MyVariable" := 'Hello, World!';
6   myvariable2 := 'PL/SQL Programming';
7   DBMS_OUTPUT.PUT_LINE('Quoted identifier: ' || "MyVariable");
8   DBMS_OUTPUT.PUT_LINE('Non-quoted identifier: ' || myvariable2);
9 EXCEPTION
10 WHEN OTHERS THEN
11   DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
12 END;
13
```

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

```
Quoted identifier: Hello, World!
Non-quoted identifier: PL/SQL Programming
Statement processed.
```

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

QUERY:

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

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

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. The right side shows a user profile for 'Akilan 22' and a schema dropdown for 'WKSP_BLEACH22'. The main area displays a PL/SQL block:

```
1 DECLARE
2     salary_of_emp NUMBER(8,2);
3     PROCEDURE approx_salary (
4         emp      NUMBER,
5         empsal IN OUT NUMBER,
6         address   NUMBER
7     ) IS
8     BEGIN
9         empsal := empsal + address;
10    END;
11
12 BEGIN
13     SELECT salary INTO salary_of_emp
14     FROM em
15     WHERE employee_id = 122;
16     DBMS_OUTPUT.PUT_LINE
17     ('Before invoking procedure, salary_of_emp: ' || salary_of_emp);
18     approx_salary (100, salary_of_emp, 1000);
19     DBMS_OUTPUT.PUT_LINE
20     ('After invoking procedure, salary_of_emp: ' || salary_of_emp);
21 END;
22 /
23
```

The results tab shows the output of the procedure execution:

```
Before invoking procedure, salary_of_emp: 63000
After invoking procedure, salary_of_emp: 64000
Statement processed.
```

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 shows a user profile for 'Akilesh21 B' and a schema dropdown for 'WKSP_AKILESH21'. The main area displays the creation of the 'pri_bool' procedure:

```
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 /
```

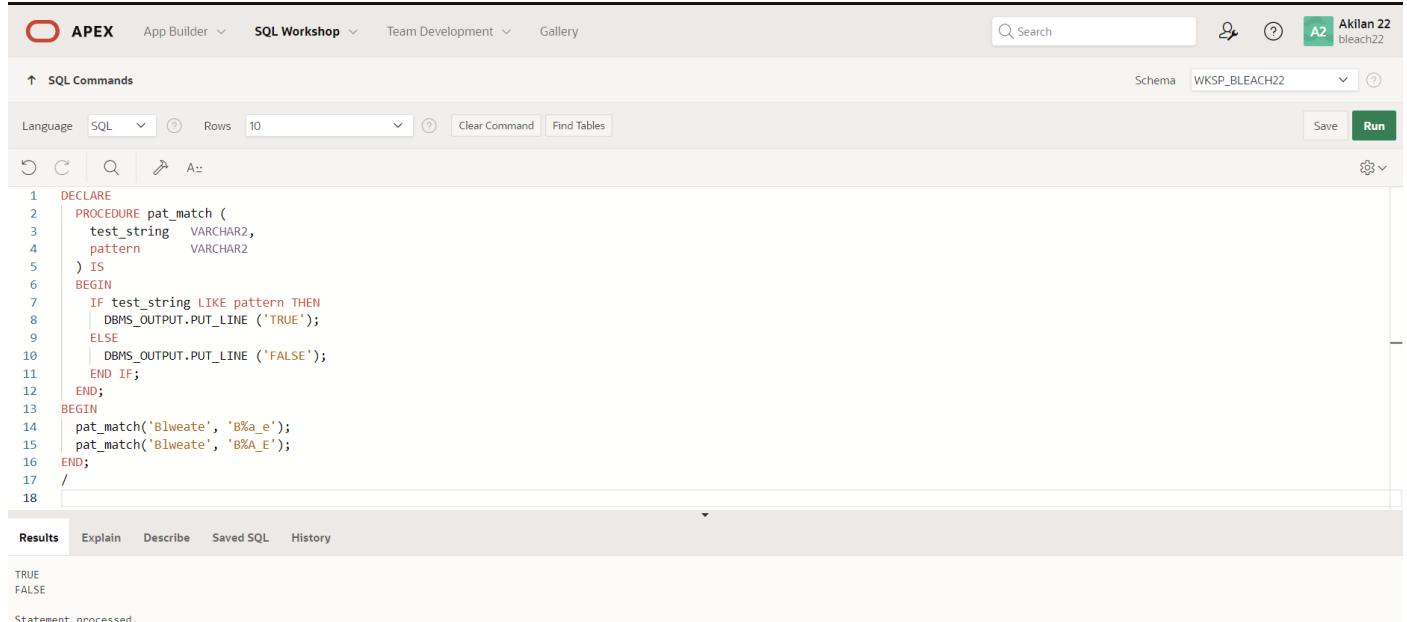
The results tab shows the message 'Procedure created.'

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user information (A2 Akilan 22 bleach22), and a schema dropdown set to WKSP_BLEACH22. The main area is titled 'SQL Commands' and contains the PL/SQL code from the previous section. Below the code, the 'Results' tab is selected, showing the output: 'TRUE' and 'FALSE'. The status bar at the bottom indicates 'Statement processed.'

```
1  DECLARE
2  PROCEDURE pat_match (
3    test_string  VARCHAR2,
4    pattern      VARCHAR2
5  ) IS
6  BEGIN
7    IF test_string LIKE pattern THEN
8      DBMS_OUTPUT.PUT_LINE ('TRUE');
9    ELSE
10      DBMS_OUTPUT.PUT_LINE ('FALSE');
11    END IF;
12  END;
13 BEGIN
14   pat_match('Blweate', 'B%a_e');
15   pat_match('Blweate', 'B%A_E');
16 END;
17 /
18
```

Results Explain Describe Saved SQL History

TRUE
FALSE

Statement processed.

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

QUERY:

DECLARE

num_small NUMBER := 8;

num_large NUMBER := 5;

num_temp NUMBER;

BEGIN

IF num_small > num_large THEN

num_temp := num_small;

num_small := num_large;

num_large := num_temp;

END IF;

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

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

END;

/

OUTPUT:

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

Results

num_small	num_large
5	8

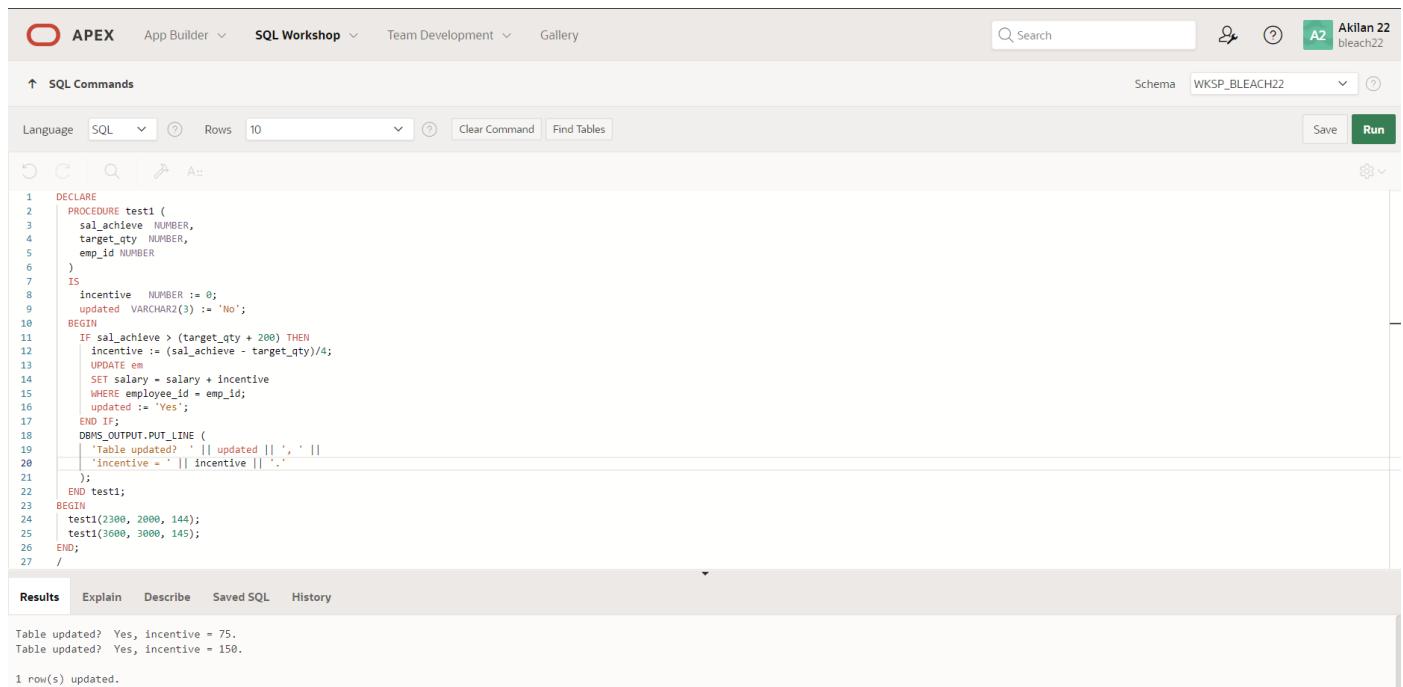
Statement processed.

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there are search, schema dropdown (WKSP_BLEACH22), and user information (Akilan 22, bleach22). The main area is titled 'SQL Commands' with tabs for Language (SQL selected), Rows (10), and Clear Command. Below is the code editor with the PL/SQL procedure. The bottom section shows the results of the execution.

```
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 tab is selected. The output shows two rows of data:

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

1 row(s) updated.
```

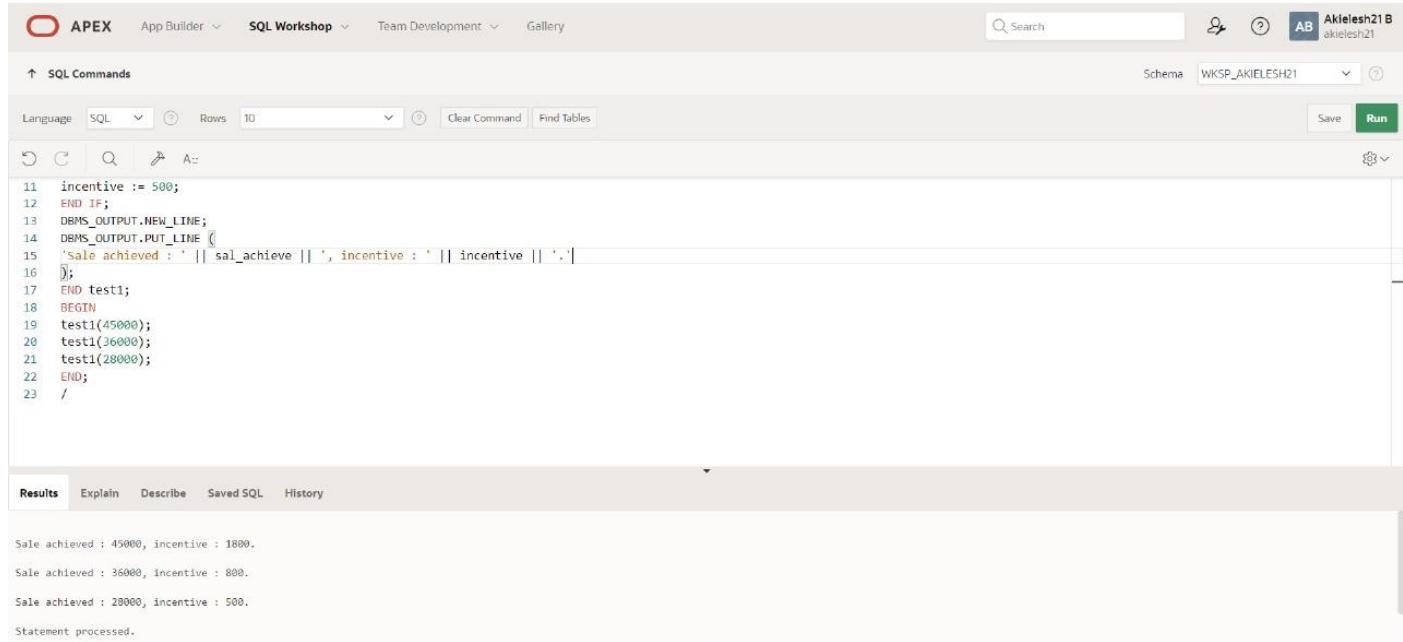
8.) Write a PL/SQL procedure to calculate incentive achieved according to the specific sale limit

QUERY:

DECLARE

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, user profile, and a schema dropdown set to WKSP_AKILESH21. The main area is titled 'SQL Commands' and contains the PL/SQL code from the previous block. Below the code, the 'Results' tab is selected, displaying the output of the procedure execution: 'Sale achieved : 45000, incentive : 1800.', 'Sale achieved : 36000, incentive : 800.', 'Sale achieved : 28000, incentive : 500.', and 'Statement processed.'

```
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 /
```

Results Explain Describe Saved SQL History

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

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

QUERY:

DECLARE

```
tot_emp NUMBER;
```

BEGIN

```
    SELECT Count(*)
```

```
    INTO tot_emp
```

```
    FROM employees e
```

```
        join mydept d
```

```
            ON e.department_id = d.deptid
```

```
    WHERE e.department_id = 50;
```

```
dbms_output.Put_line ('The employees are in the department 50: '
```

```
    ||To_char(tot_emp));
```

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

```
    dbms_output.Put_line ('There are no vacancies in the department 50.');
```

```
ELSE
```

```
    dbms_output.Put_line ('There are some vacancies in department 50.');
```

```
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 shows a user profile (A2 Akilan 22) and a schema dropdown (WKSP_BLEACH22). The main area is titled 'SQL Commands' and contains the following PL/SQL code:

```
1  DECLARE
2  |      tot_emp NUMBER;
3  BEGIN
4  |      SELECT Count(*)
5  |      INTO tot_emp
6  |      FROM employees e
7  |          join mydept d
8  |              ON e.department_id = d.deptid
9  |      WHERE e.department_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 'Results' tab at the bottom shows the output of the program:

```
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

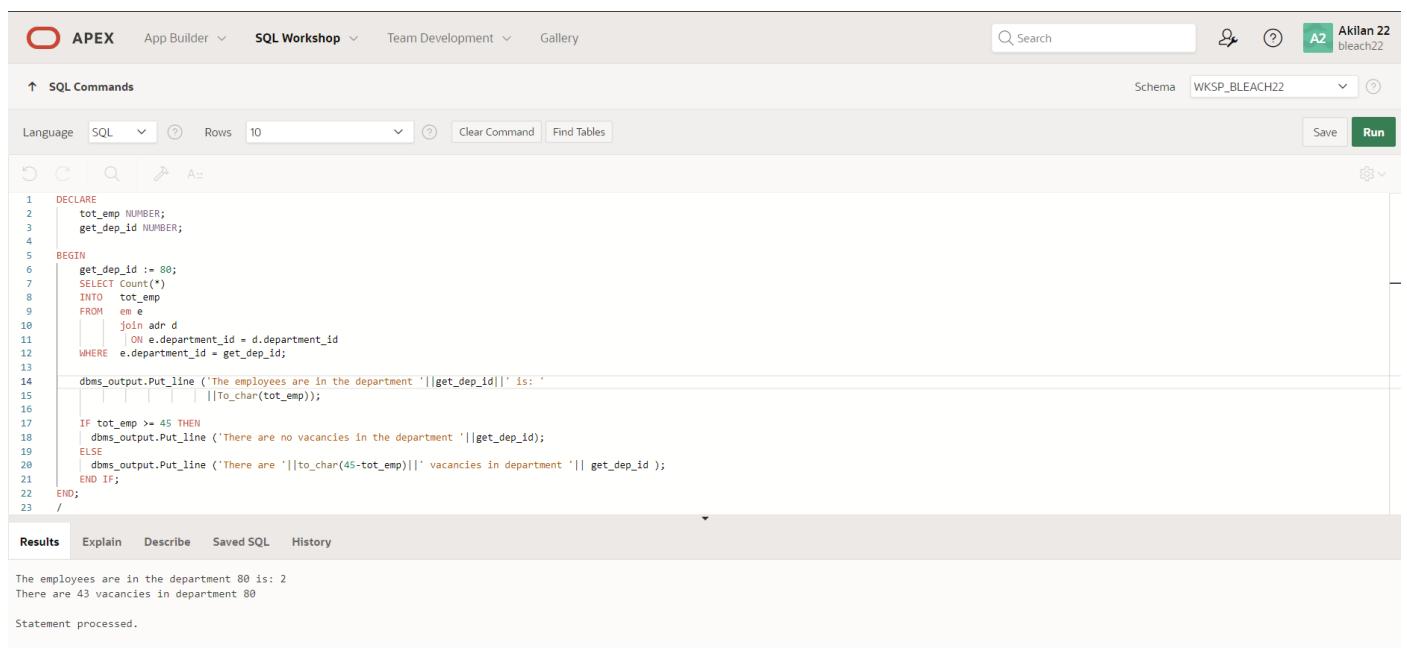
```
dbms_output.Put_line ('There are'||to_char(45-tot_emp)||' vacancies in department'|| get_dep_id );
);
```

END IF;

END;

/

OUTPUT:



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

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

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

QUERY:

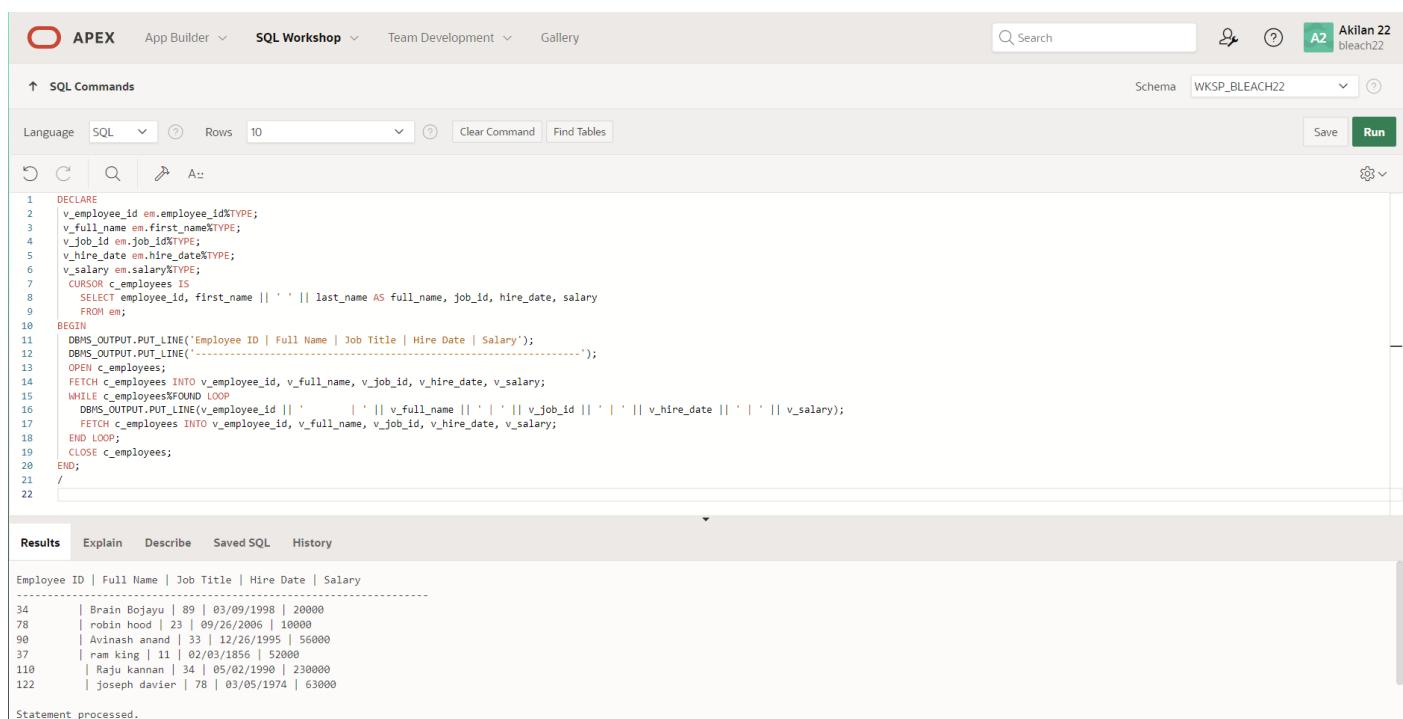
DECLARE

```
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 (selected), Team Development, and Gallery. On the right, there's a user profile for 'Akilan 22' and a schema dropdown set to 'WKSP_BLEACH22'. The main area has tabs for SQL Commands, Results, Explain, Describe, Saved SQL, and History. The SQL Commands tab contains the PL/SQL code from the previous section. The Results tab displays the output:

Employee ID	Full Name	Job Title	Hire Date	Salary
34	Brain BoJoyau	89	03/09/1998	20000
78	robin hood	23	09/26/2006	10000
90	Avinash anand	33	12/26/1995	56000
37	ram king	11	02/03/1856	52000
110	Raju kannan	34	05/02/1990	230000
122	joseph davier	78	03/05/1974	63000

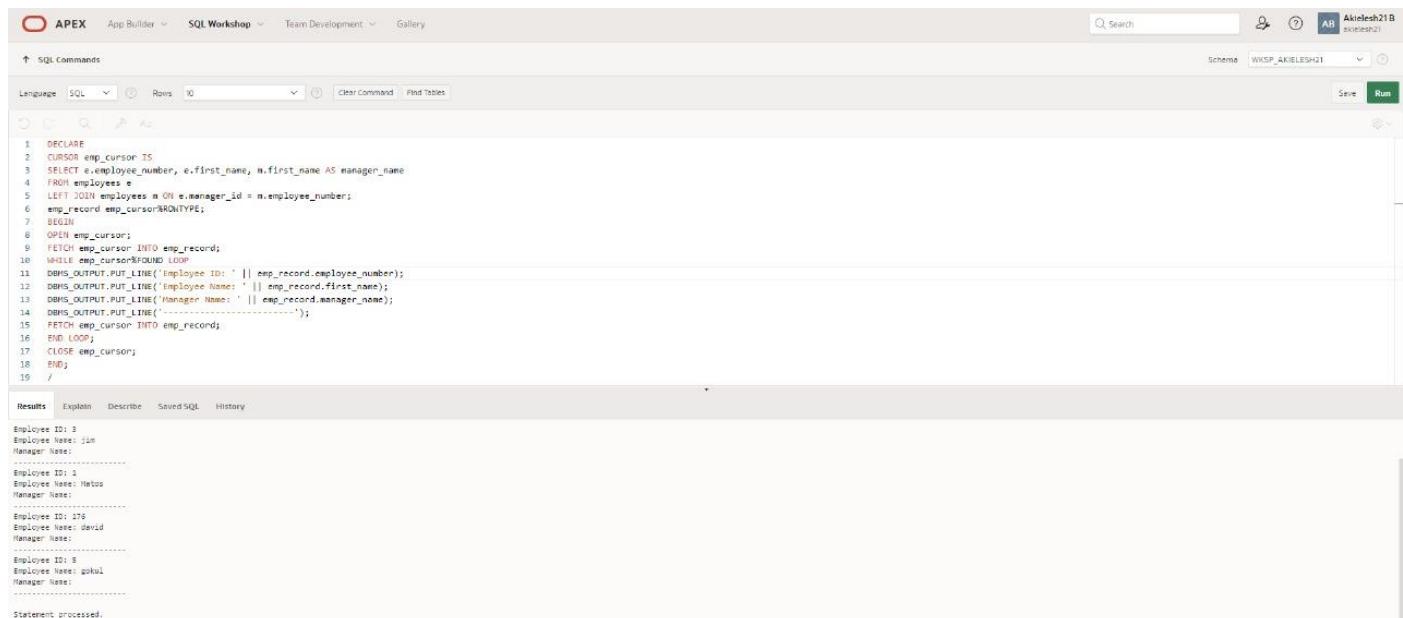
Below the table, a message says 'Statement processed.'

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

QUERY:

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

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface with the following details:

- Toolbar:** APEX, App Builder, SQL Workshop (selected), Team Development, Gallery.
- Search Bar:** Search (placeholder: Search).
- Schema:** WWS_P_AKILESH21
- Query Editor:** Contains the PL/SQL code provided in the question.
- Results Tab:** Displays the output of the executed code, listing employee details for multiple employees (e.g., Jim, Matos, David, Gokul) along with their manager names and a separator line.
- Status:** Statement processed.

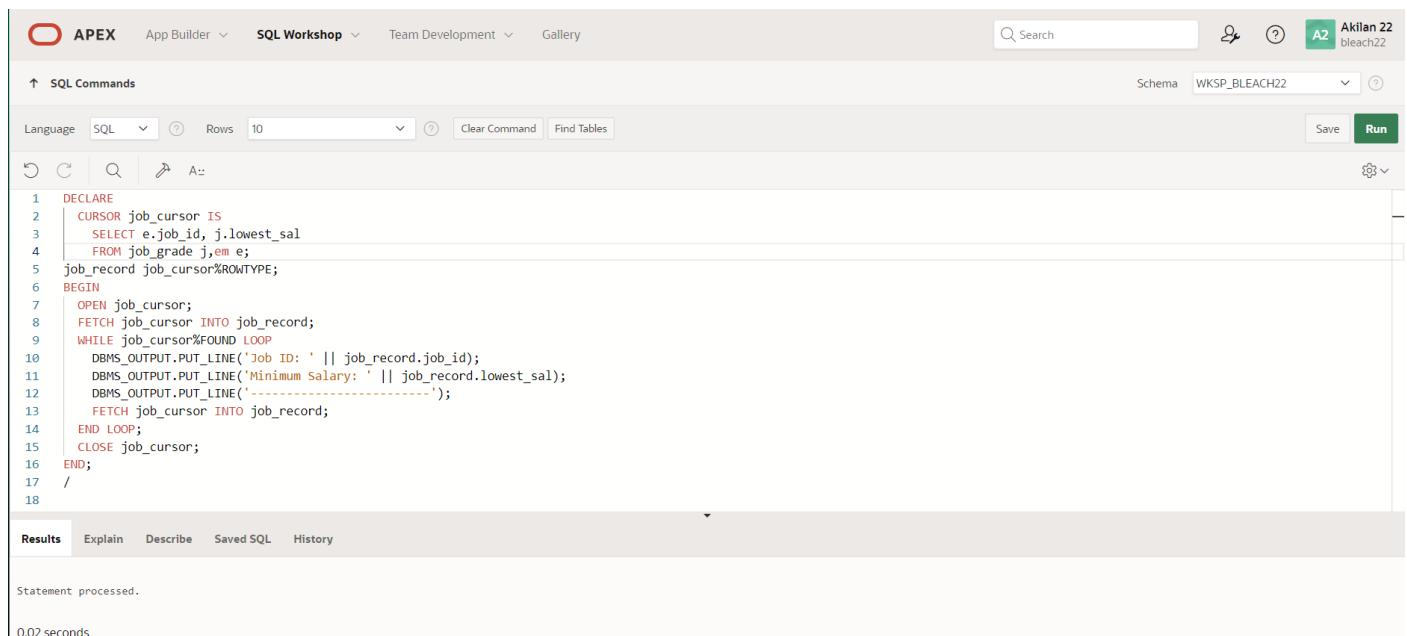
13.) Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all jobs

QUERY:

DECLARE

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

OUTPUT:



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

```
1 DECLARE
2   CURSOR job_cursor IS
3     SELECT e.job_id, j.lowest_sal
4       FROM job_grades j,employees e;
5   job_record job_cursor%ROWTYPE;
6 BEGIN
7   OPEN job_cursor;
8   FETCH job_cursor INTO job_record;
9   WHILE job_cursor%FOUND LOOP
10     DBMS_OUTPUT.PUT_LINE('Job ID: ' || job_record.job_id);
11     DBMS_OUTPUT.PUT_LINE('Minimum Salary: ' || job_record.lowest_sal);
12     DBMS_OUTPUT.PUT_LINE('-----');
13     FETCH job_cursor INTO job_record;
14   END LOOP;
15   CLOSE job_cursor;
16 END;
17 /
```

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

```
Statement processed.
0.02 seconds
```

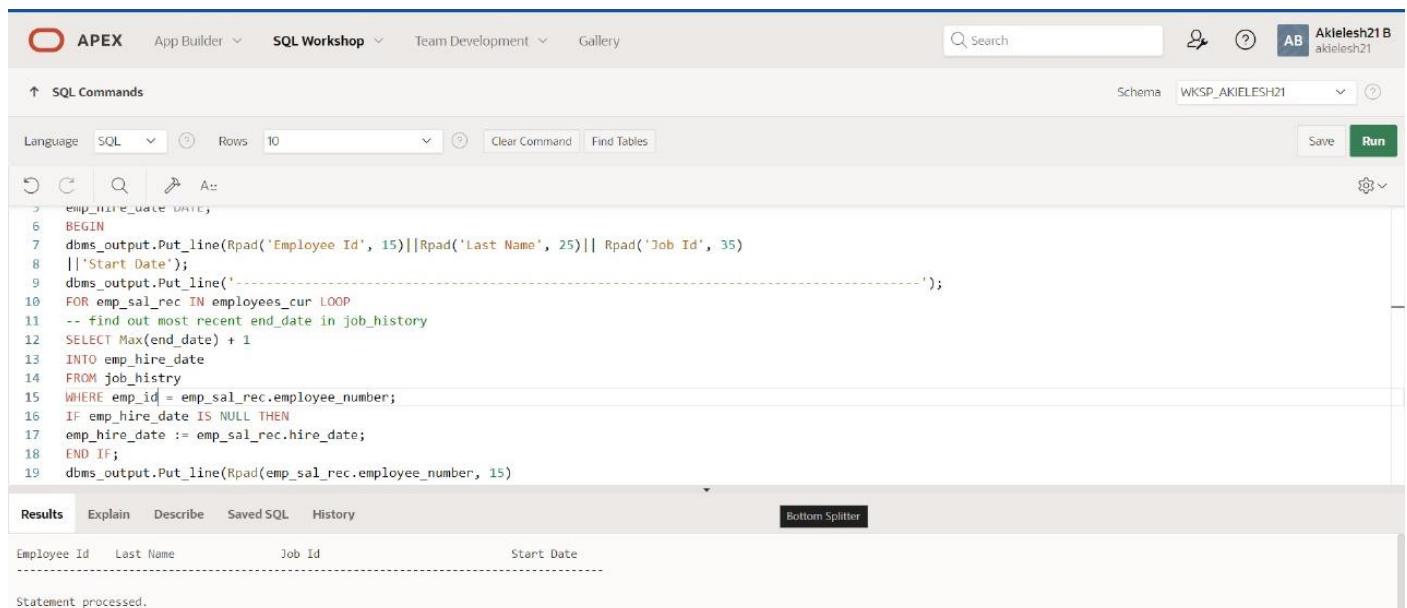
14.) Write a PL/SQL program to display the employee IDs, names, and job history start dates of all employees.

QUERY:

DECLARE

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

OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema is set to WKSP_AKIELESH21. The SQL command window contains the PL/SQL code from the previous section. The results window shows the output of the program, which displays employee information and their start dates. The bottom status bar indicates "Statement processed."

Employee Id	Last Name	Job Id	Start Date
101	ALLEN	10	20-JAN-1981
102	WARD	20	22-FEB-1981
103	MARTIN	30	28-FEB-1981
104	KING	10	17-MAR-1981
105	BLAKE	20	01-MAY-1981
106	JOHNSON	30	03-MAY-1981
107	HUNTER	40	03-MAY-1981
108	MCKEEAN	50	03-MAY-1981
109	TURNER	60	03-MAY-1981
110	ADAMS	70	03-MAY-1981
111	HAZEL	80	03-MAY-1981
112	YOUNG	90	03-MAY-1981
113	SCOTT	10	03-MAY-1981
114	TURNER	20	03-MAY-1981
115	ADAMS	30	03-MAY-1981
116	WARD	40	03-MAY-1981
117	SCOTT	50	03-MAY-1981
118	TURNER	60	03-MAY-1981
119	ADAMS	70	03-MAY-1981
120	WARD	80	03-MAY-1981
121	SCOTT	90	03-MAY-1981
122	TURNER	10	03-MAY-1981
123	ADAMS	20	03-MAY-1981
124	WARD	30	03-MAY-1981
125	SCOTT	40	03-MAY-1981
126	TURNER	50	03-MAY-1981
127	ADAMS	60	03-MAY-1981
128	WARD	70	03-MAY-1981
129	SCOTT	80	03-MAY-1981
130	TURNER	90	03-MAY-1981
131	ADAMS	10	03-MAY-1981
132	WARD	20	03-MAY-1981
133	SCOTT	30	03-MAY-1981
134	TURNER	40	03-MAY-1981
135	ADAMS	50	03-MAY-1981
136	WARD	60	03-MAY-1981
137	SCOTT	70	03-MAY-1981
138	TURNER	80	03-MAY-1981
139	ADAMS	90	03-MAY-1981
140	WARD	10	03-MAY-1981
141	SCOTT	20	03-MAY-1981
142	TURNER	30	03-MAY-1981
143	ADAMS	40	03-MAY-1981
144	WARD	50	03-MAY-1981
145	SCOTT	60	03-MAY-1981
146	TURNER	70	03-MAY-1981
147	ADAMS	80	03-MAY-1981
148	WARD	90	03-MAY-1981
149	SCOTT	10	03-MAY-1981
150	TURNER	20	03-MAY-1981
151	ADAMS	30	03-MAY-1981
152	WARD	40	03-MAY-1981
153	SCOTT	50	03-MAY-1981
154	TURNER	60	03-MAY-1981
155	ADAMS	70	03-MAY-1981
156	WARD	80	03-MAY-1981
157	SCOTT	90	03-MAY-1981
158	TURNER	10	03-MAY-1981
159	ADAMS	20	03-MAY-1981
160	WARD	30	03-MAY-1981
161	SCOTT	40	03-MAY-1981
162	TURNER	50	03-MAY-1981
163	ADAMS	60	03-MAY-1981
164	WARD	70	03-MAY-1981
165	SCOTT	80	03-MAY-1981
166	TURNER	90	03-MAY-1981
167	ADAMS	10	03-MAY-1981
168	WARD	20	03-MAY-1981
169	SCOTT	30	03-MAY-1981
170	TURNER	40	03-MAY-1981
171	ADAMS	50	03-MAY-1981
172	WARD	60	03-MAY-1981
173	SCOTT	70	03-MAY-1981
174	TURNER	80	03-MAY-1981
175	ADAMS	90	03-MAY-1981
176	WARD	10	03-MAY-1981
177	SCOTT	20	03-MAY-1981
178	TURNER	30	03-MAY-1981
179	ADAMS	40	03-MAY-1981
180	WARD	50	03-MAY-1981
181	SCOTT	60	03-MAY-1981
182	TURNER	70	03-MAY-1981
183	ADAMS	80	03-MAY-1981
184	WARD	90	03-MAY-1981
185	SCOTT	10	03-MAY-1981
186	TURNER	20	03-MAY-1981
187	ADAMS	30	03-MAY-1981
188	WARD	40	03-MAY-1981
189	SCOTT	50	03-MAY-1981
190	TURNER	60	03-MAY-1981
191	ADAMS	70	03-MAY-1981
192	WARD	80	03-MAY-1981
193	SCOTT	90	03-MAY-1981
194	TURNER	10	03-MAY-1981
195	ADAMS	20	03-MAY-1981
196	WARD	30	03-MAY-1981
197	SCOTT	40	03-MAY-1981
198	TURNER	50	03-MAY-1981
199	ADAMS	60	03-MAY-1981
200	WARD	70	03-MAY-1981
201	SCOTT	80	03-MAY-1981
202	TURNER	90	03-MAY-1981
203	ADAMS	10	03-MAY-1981
204	WARD	20	03-MAY-1981
205	SCOTT	30	03-MAY-1981
206	TURNER	40	03-MAY-1981
207	ADAMS	50	03-MAY-1981
208	WARD	60	03-MAY-1981
209	SCOTT	70	03-MAY-1981
210	TURNER	80	03-MAY-1981
211	ADAMS	90	03-MAY-1981
212	WARD	10	03-MAY-1981
213	SCOTT	20	03-MAY-1981
214	TURNER	30	03-MAY-1981
215	ADAMS	40	03-MAY-1981
216	WARD	50	03-MAY-1981
217	SCOTT	60	03-MAY-1981
218	TURNER	70	03-MAY-1981
219	ADAMS	80	03-MAY-1981
220	WARD	90	03-MAY-1981
221	SCOTT	10	03-MAY-1981
222	TURNER	20	03-MAY-1981
223	ADAMS	30	03-MAY-1981
224	WARD	40	03-MAY-1981
225	SCOTT	50	03-MAY-1981
226	TURNER	60	03-MAY-1981
227	ADAMS	70	03-MAY-1981
228	WARD	80	03-MAY-1981
229	SCOTT	90	03-MAY-1981
230	TURNER	10	03-MAY-1981
231	ADAMS	20	03-MAY-1981
232	WARD	30	03-MAY-1981
233	SCOTT	40	03-MAY-1981
234	TURNER	50	03-MAY-1981
235	ADAMS	60	03-MAY-1981
236	WARD	70	03-MAY-1981
237	SCOTT	80	03-MAY-1981
238	TURNER	90	03-MAY-1981
239	ADAMS	10	03-MAY-1981
240	WARD	20	03-MAY-1981
241	SCOTT	30	03-MAY-1981
242	TURNER	40	03-MAY-1981
243	ADAMS	50	03-MAY-1981
244	WARD	60	03-MAY-1981
245	SCOTT	70	03-MAY-1981
246	TURNER	80	03-MAY-1981
247	ADAMS	90	03-MAY-1981
248	WARD	10	03-MAY-1981
249	SCOTT	20	03-MAY-1981
250	TURNER	30	03-MAY-1981
251	ADAMS	40	03-MAY-1981
252	WARD	50	03-MAY-1981
253	SCOTT	60	03-MAY-1981
254	TURNER	70	03-MAY-1981
255	ADAMS	80	03-MAY-1981
256	WARD	90	03-MAY-1981
257	SCOTT	10	03-MAY-1981
258	TURNER	20	03-MAY-1981
259	ADAMS	30	03-MAY-1981
260	WARD	40	03-MAY-1981
261	SCOTT	50	03-MAY-1981
262	TURNER	60	03-MAY-1981
263	ADAMS	70	03-MAY-1981
264	WARD	80	03-MAY-1981
265	SCOTT	90	03-MAY-1981
266	TURNER	10	03-MAY-1981
267	ADAMS	20	03-MAY-1981
268	WARD	30	03-MAY-1981
269	SCOTT	40	03-MAY-1981
270	TURNER	50	03-MAY-1981
271	ADAMS	60	03-MAY-1981
272	WARD	70	03-MAY-1981
273	SCOTT	80	03-MAY-1981
274	TURNER	90	03-MAY-1981
275	ADAMS	10	03-MAY-1981
276	WARD	20	03-MAY-1981
277	SCOTT	30	03-MAY-1981
278	TURNER	40	03-MAY-1981
279	ADAMS	50	03-MAY-1981
280	WARD	60	03-MAY-1981
281	SCOTT	70	03-MAY-1981
282	TURNER	80	03-MAY-1981
283	ADAMS	90	03-MAY-1981
284	WARD	10	03-MAY-1981
285	SCOTT	20	03-MAY-1981
286	TURNER	30	03-MAY-1981
287	ADAMS	40	03-MAY-1981
288	WARD	50	03-MAY-1981
289	SCOTT	60	03-MAY-1981
290	TURNER	70	03-MAY-1981
291	ADAMS	80	03-MAY-1981
292	WARD	90	03-MAY-1981
293	SCOTT	10	03-MAY-1981
294	TURNER	20	03-MAY-1981
295	ADAMS	30	03-MAY-1981
296	WARD	40	03-MAY-1981
297	SCOTT	50	03-MAY-1981
298	TURNER	60	03-MAY-1981
299	ADAMS	70	03-MAY-1981
300	WARD	80	03-MAY-1981
301	SCOTT	90	03-MAY-1981
302	TURNER	10	03-MAY-1981
303	ADAMS	20	03-MAY-1981
304	WARD	30	03-MAY-1981
305	SCOTT	40	03-MAY-1981
306	TURNER	50	03-MAY-1981
307	ADAMS	60	03-MAY-1981
308	WARD	70	03-MAY-1981
309	SCOTT	80	03-MAY-1981
310	TURNER	90	03-MAY-1981
311	ADAMS	10	03-MAY-1981
312	WARD	20	03-MAY-1981
313	SCOTT	30	03-MAY-1981
314	TURNER	40	03-MAY-1981
315	ADAMS	50	03-MAY-1981
316	WARD	60	03-MAY-1981
317	SCOTT	70	03-MAY-1981
318	TURNER	80	03-MAY-1981
319	ADAMS	90	03-MAY-1981
320	WARD	10	03-MAY-1981
321	SCOTT	20	03-MAY-1981
322	TURNER	30	03-MAY-1981
323	ADAMS	40	03-MAY-1981
324	WARD	50	03-MAY-1981
325	SCOTT	60	03-MAY-1981
326	TURNER	70	03-MAY-1981
327	ADAMS	80	03-MAY-1981
328	WARD	90	03-MAY-1981
329	SCOTT	10	03-MAY-1981
330	TURNER	20	03-MAY-1981
331	ADAMS	30	03-MAY-1981
332	WARD	40	03-MAY-1981
333	SCOTT	50	03-MAY-1981
334	TURNER	60	03-MAY-1981
335	ADAMS	70	03-MAY-1981
336	WARD	80	03-MAY-1981
337	SCOTT	90	03-MAY-1981
338	TURNER	10	03-MAY-1981
339	ADAMS	20	03-MAY-1981
340	WARD	30	03-MAY-1981
341	SCOTT	40	03-MAY-1981
342	TURNER	50	03-MAY-1981
343	ADAMS	60	03-MAY-1981
344	WARD	70	03-MAY-1981
345	SCOTT	80	03-MAY-1981
346	TURNER	90	03-MAY-1981
347	ADAMS	10	03-MAY-1981
348	WARD	20	03-MAY-1981
349	SCOTT	30	03-MAY-1981
350	TURNER	40	03-MAY-1981
351	ADAMS	50	03-MAY-1981
352	WARD	60	03-MAY-1981
353	SCOTT	70	03-MAY-1981
354	TURNER	80	03-MAY-1981
355	ADAMS	90	03-MAY-1981
356	WARD	10	03-MAY-1981
357	SCOTT	20	03-MAY-1981
358	TURNER	30	03-MAY-1981
359	ADAMS	40	03-MAY-1981
360	WARD	50	03-MAY-1981
361	SCOTT	60	03-MAY-1981
362	TURNER	70	03-MAY-1981
363	ADAMS	80	03-MAY-1981
364	WARD	90	03-MAY-1981
365	SCOTT	10	03-MAY-1981

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

QUERY:

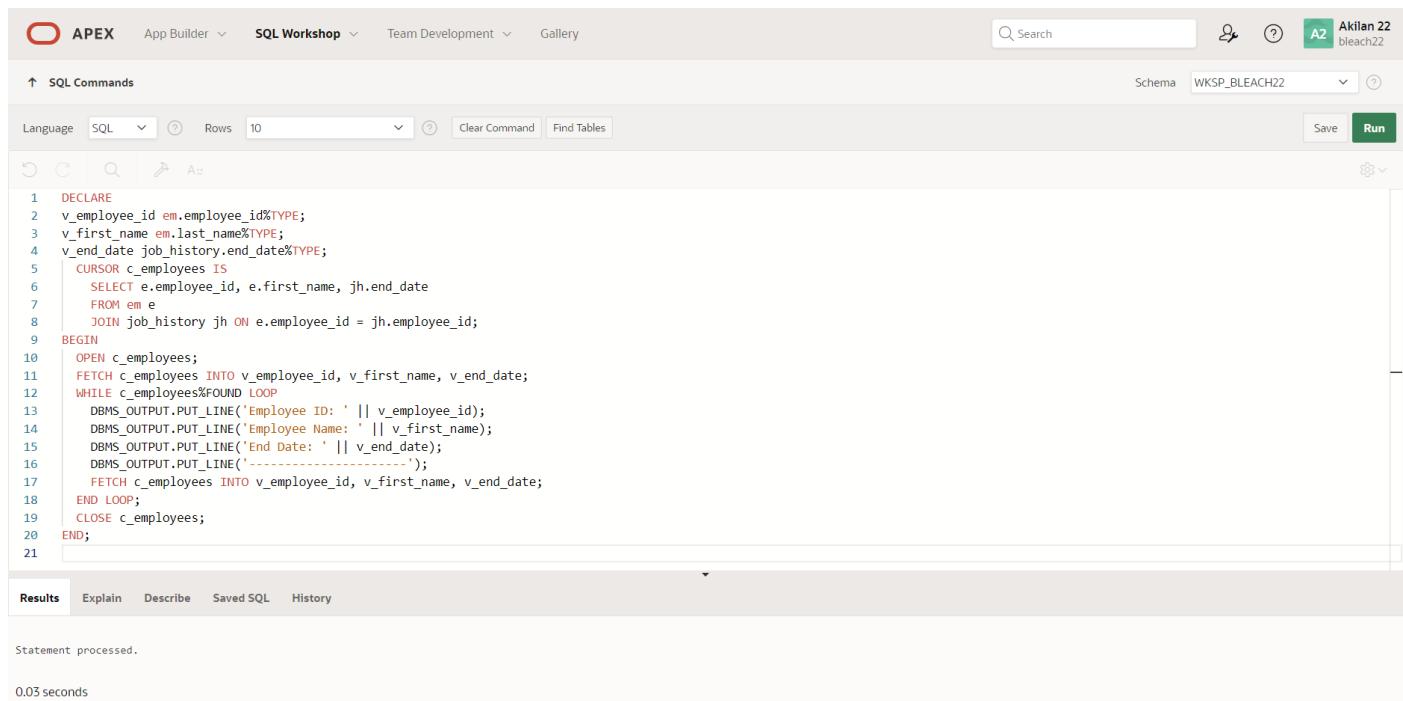
DECLARE

```
v_employee_id employees.employee_id%TYPE;
v_first_name employees.last_name%TYPE;
v_end_date job_history.end_date%TYPE;

CURSOR c_employees IS
  SELECT e.employee_id, e.first_name, jh.end_date
  FROM employees e
  JOIN job_history jh ON e.employee_id = jh.employee_id;

BEGIN
  OPEN c_employees;
  FETCH c_employees INTO v_employee_id, v_first_name, v_end_date;
  WHILE c_employees%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_employee_id);
    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_first_name);
    DBMS_OUTPUT.PUT_LINE('End Date: ' || v_end_date);
    DBMS_OUTPUT.PUT_LINE('-----');
    FETCH c_employees INTO v_employee_id, v_first_name, v_end_date;
  END LOOP;
  CLOSE c_employees;
END;
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. The right side shows a user profile for 'Akilan 22' and 'bleach22'. The main area is titled 'SQL Commands' with a schema dropdown set to 'WKSP_BLEACH22'. The code area contains the PL/SQL block from above, numbered 1 to 21. Below the code, the 'Results' tab is selected, showing the output: 'Statement processed.' and '0.03 seconds'.

```
1  DECLARE
2  v_employee_id em.employee_id%TYPE;
3  v_first_name em.last_name%TYPE;
4  v_end_date job_history.end_date%TYPE;
5  CURSOR c_employees IS
6    SELECT e.employee_id, e.first_name, jh.end_date
7    FROM employees e
8    JOIN job_history jh ON e.employee_id = jh.employee_id;
9  BEGIN
10    OPEN c_employees;
11    FETCH c_employees INTO v_employee_id, v_first_name, v_end_date;
12    WHILE c_employees%FOUND LOOP
13      DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_employee_id);
14      DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_first_name);
15      DBMS_OUTPUT.PUT_LINE('End Date: ' || v_end_date);
16      DBMS_OUTPUT.PUT_LINE('-----');
17      FETCH c_employees INTO v_employee_id, v_first_name, v_end_date;
18    END LOOP;
19    CLOSE c_employees;
20  END;
21
```

Results Explain Describe Saved SQL History

Statement processed.
0.03 seconds

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

CONTROLLING USER ACCESS

Program 1

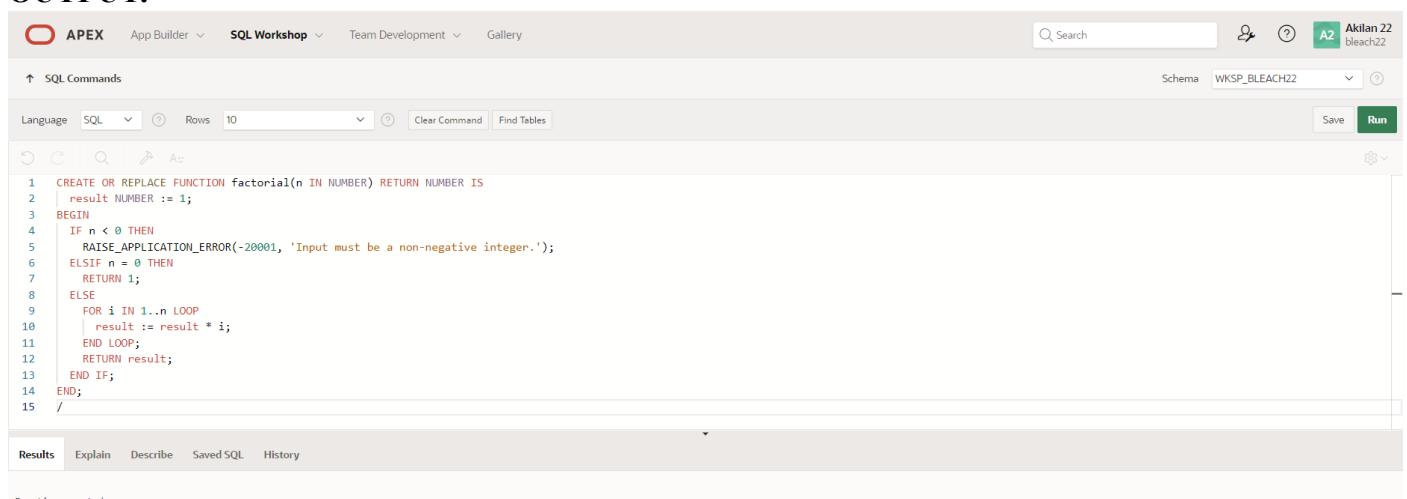
FACTORIAL OF A NUMBER USING FUNCTION

QUERY:

```
CREATE OR REPLACE FUNCTION factorial(n IN NUMBER) RETURN NUMBER IS
    result NUMBER := 1;
BEGIN
    IF n < 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Input must be a non-negative integer.');
    ELSIF n = 0 THEN
        RETURN 1;
    ELSE
        FOR i IN 1..n LOOP
            result := result * i;
        END LOOP;
        RETURN result;
    END IF;
END;
```

/

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop, Team Development, and Gallery. The SQL Workshop tab is selected. The schema dropdown shows WKSP_BLEACH22. The main area displays the SQL code for creating the factorial function, which is identical to the one shown in the previous text block. The code is highlighted in red and orange. Below the code, the Results tab is active, showing the message "Function created.".

```
1 CREATE OR REPLACE FUNCTION factorial(n IN NUMBER) RETURN NUMBER IS
2     result NUMBER := 1;
3 BEGIN
4     IF n < 0 THEN
5         RAISE_APPLICATION_ERROR(-20001, 'Input must be a non-negative integer.');
6     ELSIF n = 0 THEN
7         RETURN 1;
8     ELSE
9         FOR i IN 1..n LOOP
10             result := result * i;
11         END LOOP;
12         RETURN result;
13     END IF;
14 END;
15 /
```

Results Explain Describe Saved SQL History

Function created.

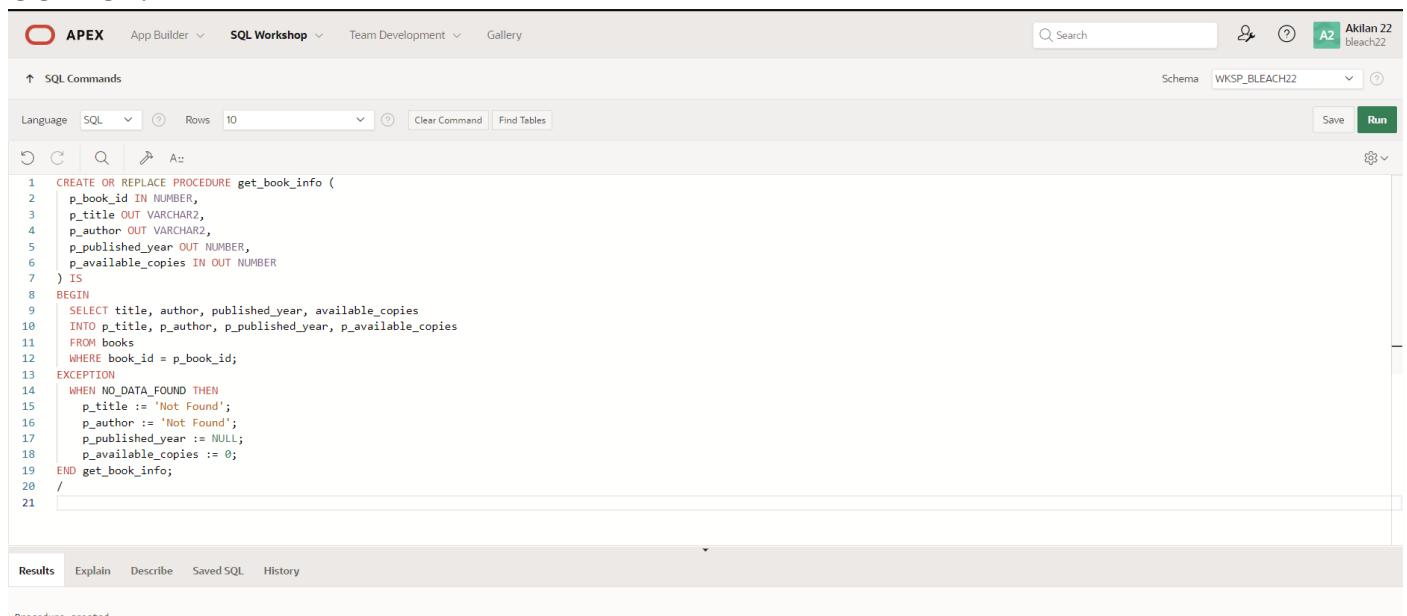
Program 2

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

QUERY:

```
CREATE OR REPLACE PROCEDURE get_book_info (
    p_book_id IN NUMBER,
    p_title OUT VARCHAR2,
    p_author OUT VARCHAR2,
    p_published_year OUT NUMBER,
    p_available_copies IN OUT NUMBER
) IS
BEGIN
    SELECT title, author, published_year, available_copies
    INTO p_title, p_author, p_published_year, p_available_copies
    FROM books
    WHERE book_id = p_book_id;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        p_title := 'Not Found';
        p_author := 'Not Found';
        p_published_year := NULL;
        p_available_copies := 0;
END get_book_info;
```

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 a user profile for 'Akilan 22' and a workspace named 'BLEACH22'. The main area is titled 'SQL Commands' with tabs for Language (set to SQL), Rows (set to 10), and various buttons like Clear Command and Find Tables. Below this is a toolbar with icons for Undo, Redo, Search, and Run. The code editor contains the PL/SQL procedure definition shown in the 'QUERY:' section. At the bottom, there are tabs for Results, Explain, Describe, Saved SQL, and History. The results pane displays the message 'Procedure created.'

```
1 CREATE OR REPLACE PROCEDURE get_book_info (
2     p_book_id IN NUMBER,
3     p_title OUT VARCHAR2,
4     p_author OUT VARCHAR2,
5     p_published_year OUT NUMBER,
6     p_available_copies IN OUT NUMBER
7 ) IS
8 BEGIN
9     SELECT title, author, published_year, available_copies
10    INTO p_title, p_author, p_published_year, p_available_copies
11   FROM books
12  WHERE book_id = p_book_id;
13 EXCEPTION
14    WHEN NO_DATA_FOUND THEN
15        p_title := 'Not Found';
16        p_author := 'Not Found';
17        p_published_year := NULL;
18        p_available_copies := 0;
19 END get_book_info;
20 /
21 
```

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

TRIGGER

Program 1

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

QUERY:

```
CREATE OR REPLACE TRIGGER prevents_parent_deletion
```

```
BEFORE DELETE ON parent_table
```

```
FOR EACH ROW
```

```
DECLARE
```

```
    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
```

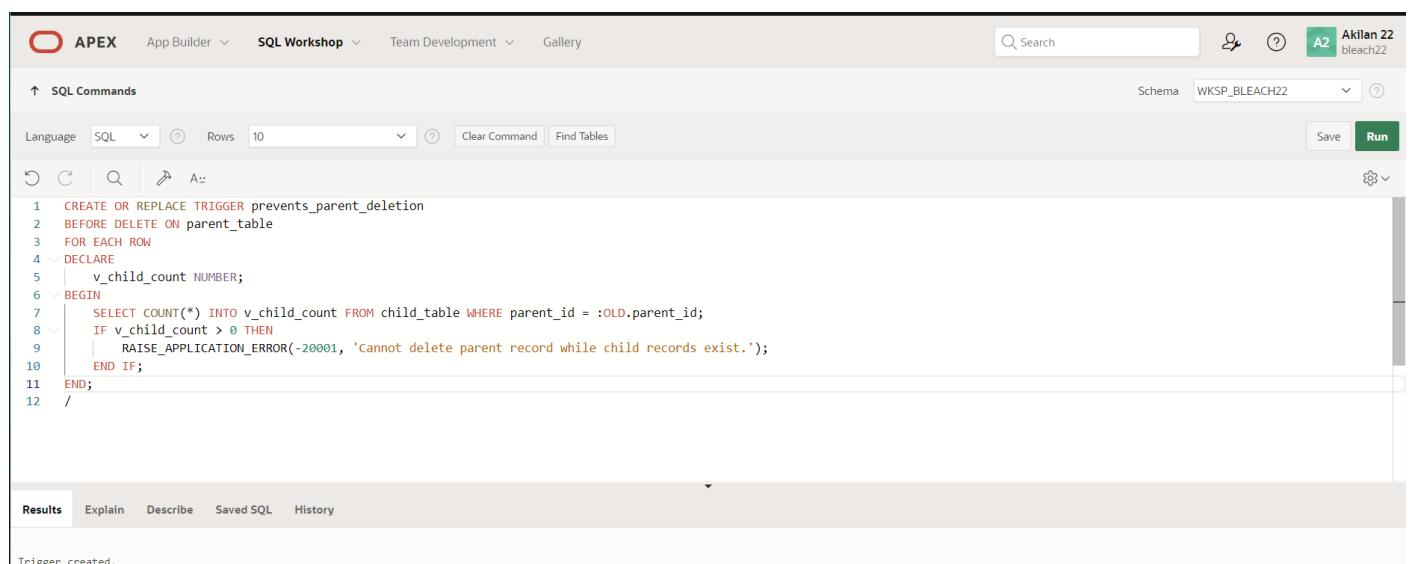
```
        RAISE_APPLICATION_ERROR(-20001, 'Cannot delete parent record while child records exist.');
```

```
    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 shows a user profile 'A2 Akilan 22' and a schema dropdown 'WKSP_BLEACH22'. The main workspace is titled 'SQL Commands' and contains the following PL/SQL code:

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

Below the code, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab displays the message 'Trigger created.'

Program 2

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

QUERY:

```
CREATE OR REPLACE TRIGGER check_duplicate_values
BEFORE INSERT OR UPDATE ON example_table
FOR EACH ROW
DECLARE
    v_count NUMBER;
BEGIN
    IF INSERTING THEN
        SELECT COUNT(*)
        INTO v_count
        FROM example_table
        WHERE unique_column = :NEW.unique_column;
        IF v_count > 0 THEN
            RAISE_APPLICATION_ERROR(-20002, 'Duplicate value found in unique_column during
INSERT.');
        END IF;
    ELSIF UPDATING THEN
        SELECT COUNT(*)
        INTO v_count
        FROM example_table
        WHERE unique_column = :NEW.unique_column
        AND id != :OLD.id;
        IF v_count > 0 THEN
            RAISE_APPLICATION_ERROR(-20003, 'Duplicate value found in unique_column during
UPDATE.');
        END IF;
    END IF;
END;
```

/

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected, followed by 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. On the right, there's a search bar, user information for 'Akilan 22', and a session identifier 'A2 Akilan 22'. Below the navigation, the schema is set to 'WKSP_BLEACH22'. The main area displays the PL/SQL code for a trigger:

```
2 BEFORE INSERT OR UPDATE ON example_table
3 FOR EACH ROW
4 DECLARE
5   v_count NUMBER;
6 BEGIN
7   IF INSERTING THEN
8     SELECT COUNT(*)
9       INTO v_count
10      FROM example_table
11     WHERE unique_column = :NEW.unique_column;
12     IF v_count > 0 THEN
13       RAISE_APPLICATION_ERROR(-20002, 'Duplicate value found in unique_column during INSERT.');
14     END IF;
15   ELSIF UPDATING THEN
16     SELECT COUNT(*)
17       INTO v_count
18      FROM example_table
19     WHERE unique_column = :NEW.unique_column
20     AND id != :OLD.id;
21     IF v_count > 0 THEN
22       RAISE_APPLICATION_ERROR(-20003, 'Duplicate value found in unique_column during UPDATE.');
23     END IF;
24   END IF;
25 END;
26 /
```

Below the code, the 'Results' tab is active, showing the message 'Trigger created.'.

Program 3

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

QUERY:

```
CREATE OR REPLACE TRIGGER restrict_total_value
BEFORE INSERT OR UPDATE ON examples_table
FOR EACH ROW
DECLARE
  v_total NUMBER;
BEGIN
  IF INSERTING THEN
    SELECT SUM(value_column) INTO v_total FROM examples_table;
    v_total := NVL(v_total, 0) + :NEW.value_column;
  ELSIF UPDATING THEN
    SELECT SUM(value_column) INTO v_total FROM examples_table WHERE id != :OLD.id;
    v_total := NVL(v_total, 0) + :NEW.value_column;
  END IF;
  IF v_total > 1000 THEN
    RAISE_APPLICATION_ERROR(-20004, 'Total value of value_column exceeds the threshold of 1000.');
  END IF;
END;
```

OUTPUT:

The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected, followed by 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. On the right, there's a search bar, user information ('A2 Akilan 22'), and a 'Run' button. Below the navigation, the 'Language' dropdown is set to 'SQL'. The main area contains the PL/SQL code for a trigger:

```
1 CREATE OR REPLACE TRIGGER restrict_total_value
2 BEFORE INSERT OR UPDATE ON examples_table
3 FOR EACH ROW
4 DECLARE
5   v_total NUMBER;
6 BEGIN
7   IF INSERTING THEN
8     SELECT SUM(value_column) INTO v_total FROM examples_table;
9     v_total := NVL(v_total, 0) + :NEW.value_column;
10  ELSIF UPDATING THEN
11    SELECT SUM(value_column) INTO v_total FROM examples_table WHERE id != :OLD.id;
12    v_total := NVL(v_total, 0) + :NEW.value_column;
13  END IF;
14  IF v_total > 1000 THEN
15    RAISE_APPLICATION_ERROR(-20004, 'Total value of value_column exceeds the threshold of 1000.');
16  END IF;
17 END;
18 /
```

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

Program 4

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

QUERY:

```
CREATE OR REPLACE TRIGGER example_table_audit_trg
```

```
AFTER INSERT OR UPDATE ON example_tables
```

```
FOR EACH ROW
```

```
DECLARE
```

```
  v_user VARCHAR2(100);
```

```
BEGIN
```

```
  v_user := USER;
```

```
  IF UPDATING THEN
```

```
    INSERT INTO example_table_audit (
```

```
      audit_id,
```

```
      id,column1_old, column1_new,column2_old, column2_new,column3_old,
      column3_new,change_date,changed_by
```

```
    ) VALUES (
```

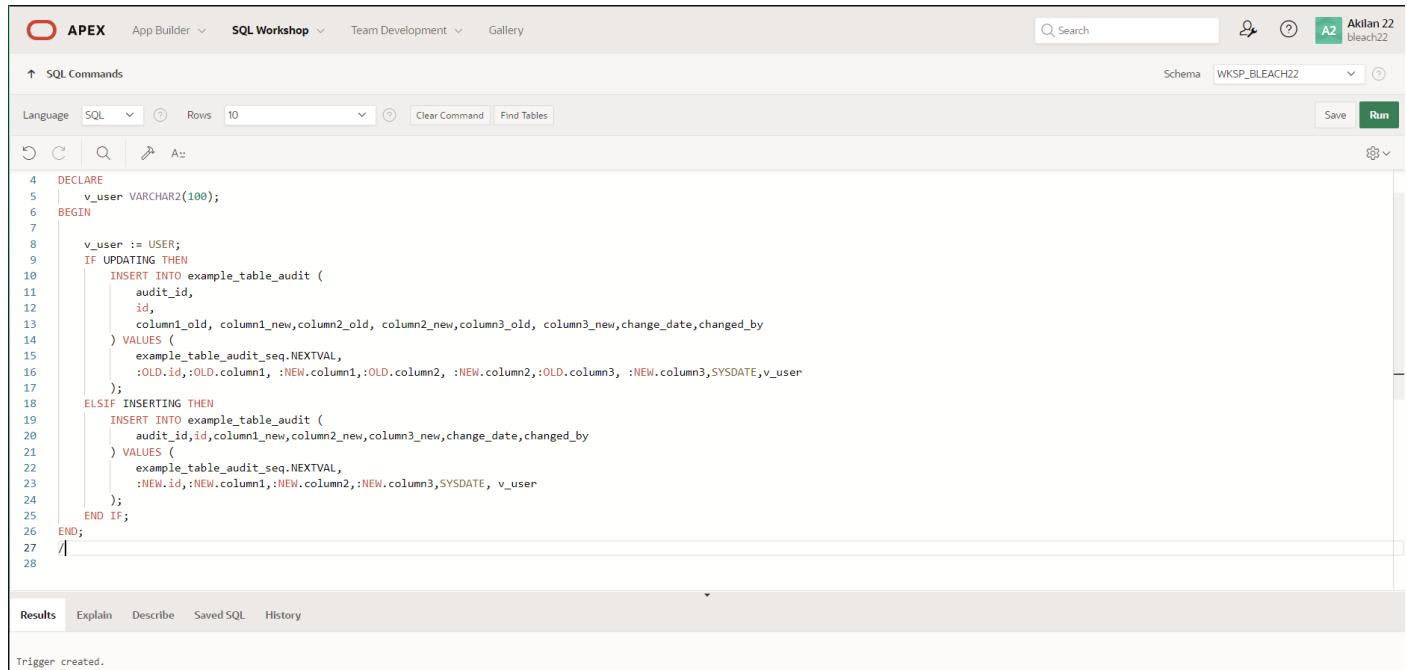
```
      example_table_audit_seq.NEXTVAL,:OLD.id,:OLD.column1, :NEW.column1,:OLD.column2,
      :NEW.column2,:OLD.column3, :NEW.column3,SYSDATE,v_user
```

```

);
ELSIF INSERTING THEN
  INSERT INTO example_table_audit (
    audit_id,id,column1_new,column2_new,column3_new,change_date,changed_by
  ) VALUES (
    example_table_audit_seq.NEXTVAL, :NEW.id,:NEW.column1,:NEW.column2,:NEW.column3,
    SYSDATE,
    v_user
  );
END IF;
END;
/

```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a user icon, and a session identifier 'A2 Akilan 22 bleak22'. The main area is titled 'SQL Commands' and contains the PL/SQL code for the trigger. The code uses dynamic SQL to insert into the audit table based on whether it's an update or an insert. The bottom of the screen shows tabs for Results, Explain, Describe, Saved SQL, and History, with the 'Results' tab currently selected.

```

4  DECLARE
5    v_user VARCHAR2(100);
6  BEGIN
7
8    v_user := USER;
9    IF UPDATING THEN
10      INSERT INTO example_table_audit (
11        audit_id,
12        id,
13        column1_old, column1_new,column2_old, column2_new,column3_old, column3_new,change_date,changed_by
14      ) VALUES (
15        example_table_audit_seq.NEXTVAL,
16        :OLD.id,:OLD.column1, :NEW.column1,:OLD.column2, :NEW.column2,:OLD.column3, :NEW.column3,SYSDATE,v_user
17      );
18    ELSIF INSERTING THEN
19      INSERT INTO example_table_audit (
20        audit_id,id,column1_new,column2_new,column3_new,change_date,changed_by
21      ) VALUES (
22        example_table_audit_seq.NEXTVAL,
23        :NEW.id,:NEW.column1,:NEW.column2,:NEW.column3,SYSDATE, v_user
24      );
25    END IF;
26  END;
27 /

```

Program 5

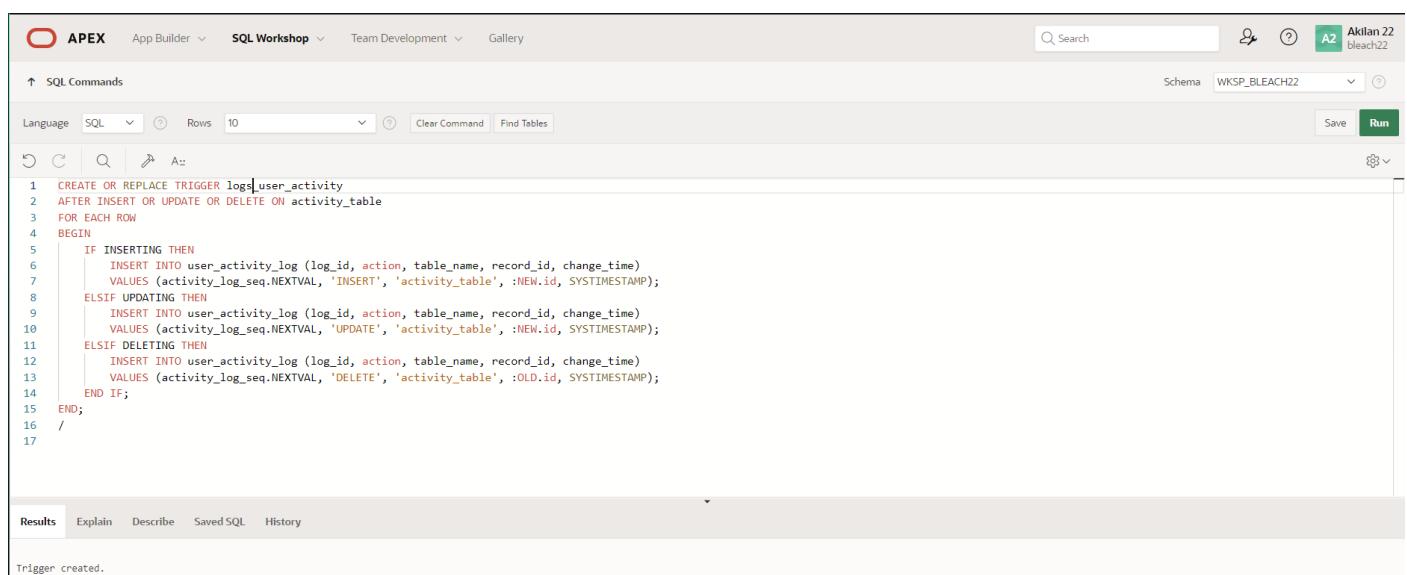
Write a code in PL/SQL to implement a trigger that records user activity (inserts, updates, deletes) in an audit log for a given set of tables.

QUERY:

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

/

OUTPUT:



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

```
1 CREATE OR REPLACE TRIGGER logs_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;
16 /
17
```

At the bottom of the interface, the status message 'Trigger created.' is visible.

Program 6

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

QUERY:

```
CREATE OR REPLACE TRIGGER ex_table_running_total_trg
```

```
BEFORE INSERT ON ex_table
```

```
FOR EACH ROW
```

```
DECLARE
```

```
    v_running_total NUMBER;
```

```
BEGIN
```

```
    SELECT NVL(SUM(amount), 0) + :NEW.amount
```

```
    INTO v_running_total
```

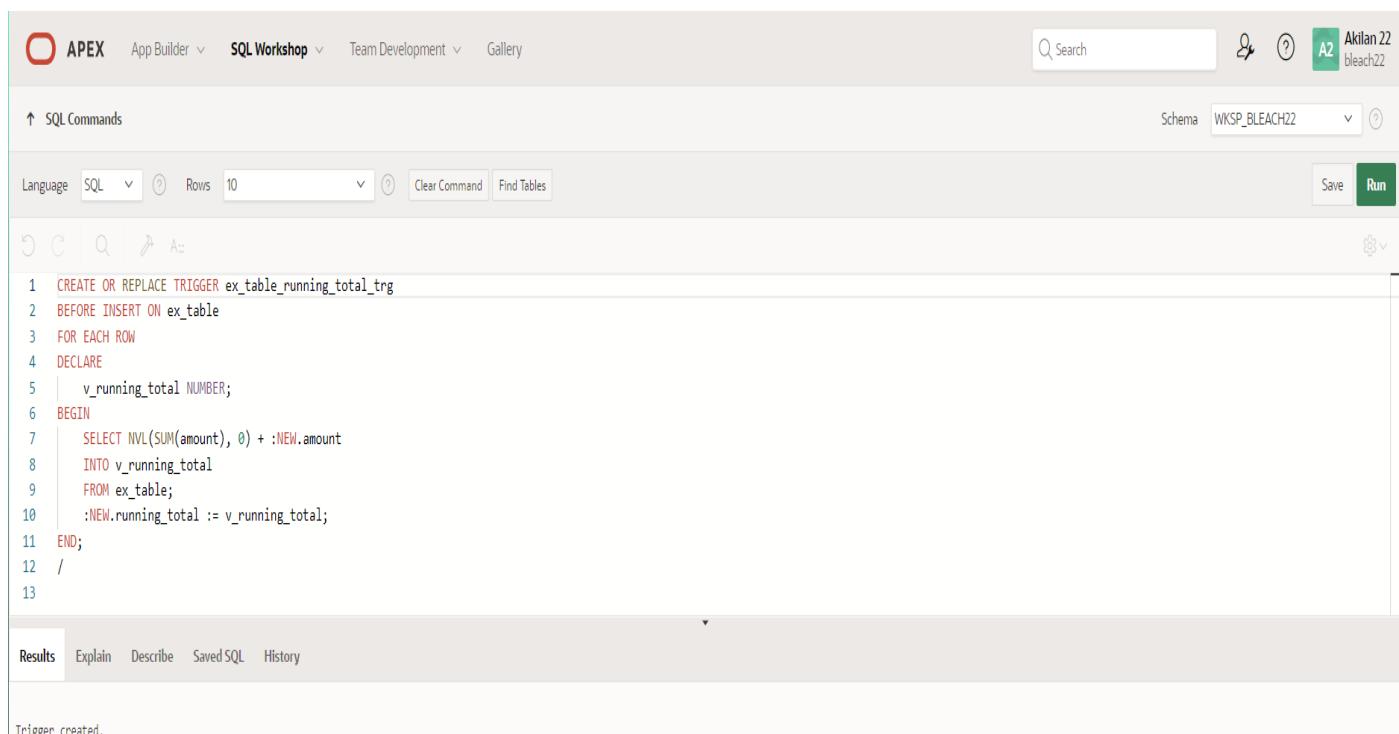
```
    FROM ex_table;
```

```
    :NEW.running_total := v_running_total;
```

```
END;
```

```
/
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes APEX, App Builder, SQL Workshop (selected), Team Development, and Gallery. On the right, there's a search bar, a help icon, and a user profile for 'Akilan22' (bleach22). The main workspace is titled 'SQL Commands'. It has dropdown menus for Language (SQL selected), Rows (10), and a toolbar with Clear Command, Find Tables, Save, and Run buttons. Below the toolbar is a toolbar with icons for Undo, Redo, Search, and others. The SQL editor area contains the trigger creation code. The code is numbered from 1 to 13. Lines 1-12 are visible, ending with a slash (/) on line 12. Line 13 is partially visible at the bottom. The status bar at the bottom shows tabs for Results, Explain, Describe, Saved SQL, and History, with 'Results' currently selected. The results pane below displays the message 'Trigger created.'

```
1 CREATE OR REPLACE TRIGGER ex_table_running_total_trg
2 BEFORE INSERT ON ex_table
3 FOR EACH ROW
4 DECLARE
5     v_running_total NUMBER;
6 BEGIN
7     SELECT NVL(SUM(amount), 0) + :NEW.amount
8     INTO v_running_total
9     FROM ex_table;
10    :NEW.running_total := v_running_total;
11 END;
12 /
13
```

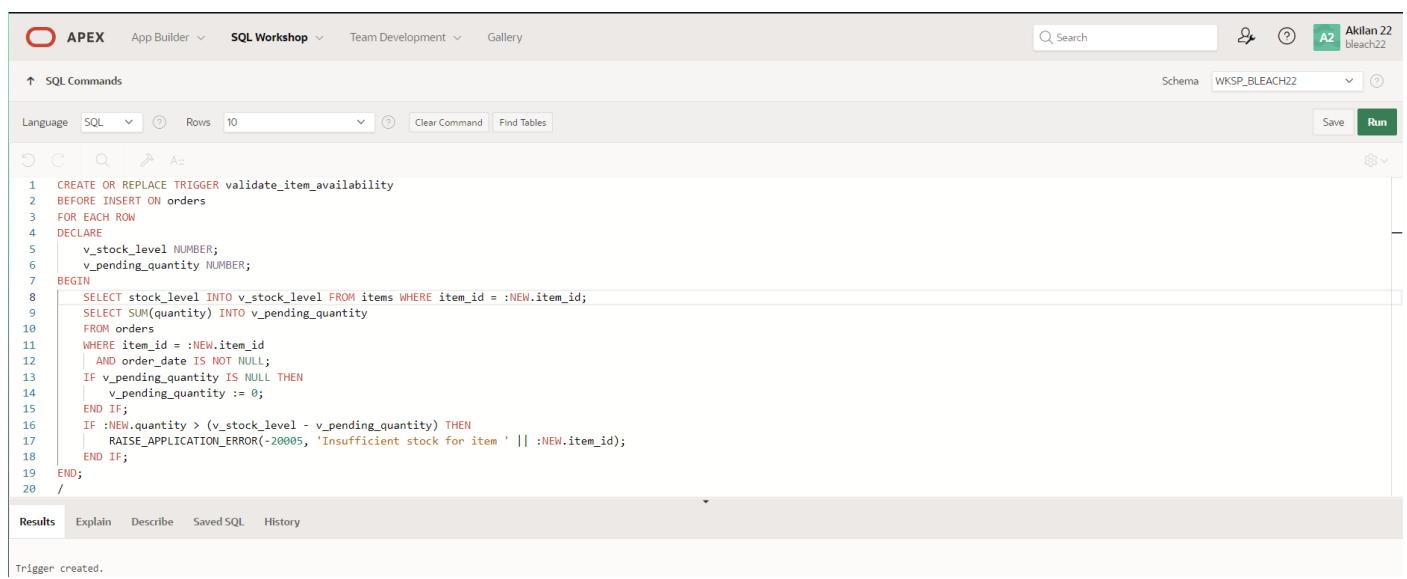
Program 7

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

QUERY:

```
CREATE OR REPLACE TRIGGER validate_item_availability
BEFORE INSERT ON orders
FOR EACH ROW
DECLARE
    v_stock_level NUMBER;
    v_pending_quantity NUMBER;
BEGIN
    SELECT stock_level INTO v_stock_level FROM items WHERE item_id = :NEW.item_id;
    SELECT SUM(quantity) INTO v_pending_quantity
    FROM orders
    WHERE item_id = :NEW.item_id
        AND order_date IS NOT NULL;
    IF v_pending_quantity IS NULL THEN
        v_pending_quantity := 0;
    END IF;
    IF :NEW.quantity > (v_stock_level - v_pending_quantity) THEN
        RAISE_APPLICATION_ERROR(-20005, 'Insufficient stock for item ' || :NEW.item_id);
    END IF;
END;
/
```

OUTPUT:



The screenshot shows the Oracle SQL Workshop interface. In the top navigation bar, 'APEX' is selected. Below it, the tabs 'SQL Workshop' and 'Team Development' are visible. On the right side, there's a user profile 'Akilan 22' and a workspace 'WKSP_BLEACH22'. The main area is titled 'SQL Commands'. At the top of the command window, there are buttons for 'Language' (set to 'SQL'), 'Rows' (set to 10), and 'Clear Command'. Below these are standard SQL navigation icons (refresh, search, etc.). The command itself is displayed in the text area, starting with 'CREATE OR REPLACE TRIGGER validate_item_availability'. The code follows the structure provided in the query section, including variable declarations, a BEGIN block with SELECT statements, IF conditions, and a RAISE_APPLICATION_ERROR statement. The code is numbered from 1 to 20. At the bottom of the command window, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. A status message 'Trigger created.' is visible at the very bottom of the interface.

```
1 CREATE OR REPLACE TRIGGER validate_item_availability
2 BEFORE INSERT ON orders
3 FOR EACH ROW
4 DECLARE
5     v_stock_level NUMBER;
6     v_pending_quantity NUMBER;
7 BEGIN
8     SELECT stock_level INTO v_stock_level FROM items WHERE item_id = :NEW.item_id;
9     SELECT SUM(quantity) INTO v_pending_quantity
10    FROM orders
11    WHERE item_id = :NEW.item_id
12        AND order_date IS NOT NULL;
13    IF v_pending_quantity IS NULL THEN
14        v_pending_quantity := 0;
15    END IF;
16    IF :NEW.quantity > (v_stock_level - v_pending_quantity) THEN
17        RAISE_APPLICATION_ERROR(-20005, 'Insufficient stock for item ' || :NEW.item_id);
18    END IF;
19 END;
20 /
```

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

MONGODB

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: [ { cuisine: { $nin: ['American', 'Chinees'] } }, { name: { $regex: '^Wil', $options: 'i' } } ] },
  { restaurant_id: 1, name: 1, borough: 1, cuisine: 1 } );
```

OUTPUT:

```
MongoDB > db.restaurants.find(
  { $or: [ { cuisine: { $nin: ['American', 'Chinees'] } }, { name: { $regex: '^Wil', $options: 'i' } } ] },
  { restaurant_id: 1, name: 1, borough: 1, cuisine: 1 } );
[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: new ISODate("2014-08-11T00:00:00Z") } },
    { restaurant_id: 1, name: 1, grades: 1 }
  );
);
```

OUTPUT:

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

```
    { $and: [ { "grades.1.grade": "A" }, { "grades.1.score": 9 }, { "grades.1.date": new ISODate("2014-08-11T00:00:00Z") } ] },
```

```
    { restaurant_id: 1, name: 1, grades: 1 } );
```

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code for the query is pasted into the command line. On the right, the 'Output' panel shows the results of the query execution.

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

Output

```
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(
```

```
    { "address.coord.1": { $gt: 42, $lte: 52 } }, { restaurant_id: 1, name: 1, address: 1, "address.coord": 1 } );
```

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code for the query is pasted into the command line. On the right, the 'Output' panel shows an error message indicating a path collision.

```
MongoDB > db.restaurants.find(
  {
    "address.coord.1": { $gt: 42, $lte: 52 }
  },
  { restaurant_id: 1, name: 1, address: 1, "address.coord": 1 }
);
```

Output

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

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

QUERY:

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

OUTPUT:

MongoDB v Run Save

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

Output

```
mycompiler_mongodb>
mycompiler_mongodb>
```

[Execution complete with exit code 0]

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

QUERY:

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

OUTPUT:

MongoDB v Run Save

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

OUTPUT:

MongoDB v Run Save

```
1 db.restaurants.find().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 } });
```

OUTPUT:

MongoDB v Run Save

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

Output

```
mycompiler_mongodb>
mycompiler_mongodb>
```

[Execution complete with exit code 0]

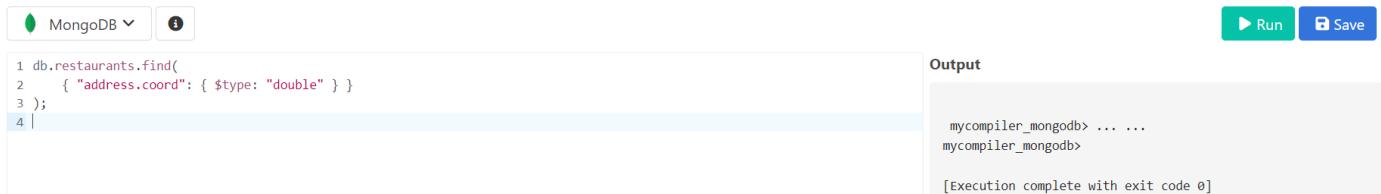
9. Write a MongoDB query which will select all documents in the restaurants collection

where the coord field value is Double.

QUERY:

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

OUTPUT:



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

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

On the right, there is an "Output" panel with the following text:

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

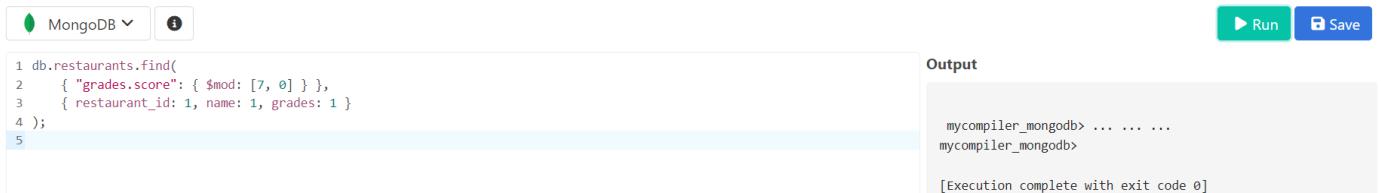
10. Write a MongoDB query which will select the restaurant Id, name and grades for

those restaurants which returns 0 as a remainder after dividing the score by 7.

QUERY:

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

OUTPUT:



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

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

On the right, there is an "Output" panel with the following text:

```
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: { $regex: 'mon', $options: 'i' } },  
  { name: 1, borough: 1, "address.coord": 1, cuisine: 1 }  
);
```

OUTPUT:

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

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

On the right, there is an "Output" panel with the following content:

```
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: { $regex: '^Mad', $options: 'i' } },
  { name: 1, borough: 1, "address.coord": 1, cuisine: 1 }
);
```

OUTPUT:

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

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

On the right, 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.score": { $lt: 5 } }
);
```

OUTPUT:

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

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

On the right, there is an "Output" panel with the following content:

```
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.score": { $lt: 5 }, borough: 'Manhattan' });
```

OUTPUT:

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

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

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

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

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

QUERY:

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

OUTPUT:

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

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

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

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

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

QUERY:

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

OUTPUT:

A screenshot of a MongoDB query interface. On the left, there is a code editor with the following query:

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

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

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

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

QUERY:

```
db.restaurants.find(
  {"grades.score": { $lt: 5 },
   borough: { $in: ['Manhattan', 'Brooklyn'] },
   cuisine: { $nin: ['American', 'Chinees'] }});
```

OUTPUT:

A screenshot of a MongoDB query interface. On the left, there is a code editor with the following query:

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

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

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

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

QUERY:

```
db.restaurants.find(
  {
    grades: { $all: [
      { $elemMatch: { score: 2 } },
      { $elemMatch: { score: 6 } }
    ]}
  });

```

OUTPUT:

MongoDB Run Save

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

Output

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

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

QUERY:

```
db.restaurants.find(
```

```
  {grades: { $all: [{ $elemMatch: { score: 2 } }, { $elemMatch: { score: 6 } }]}, borough: 'Manhattan'});
```

OUTPUT:

MongoDB Run Save

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

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(
```

```
  {grades: { $all: [{ $elemMatch: { score: 2 } }, { $elemMatch: { score: 6 } }]},  
   borough: { $in: ['Manhattan', 'Brooklyn'] }});
```

OUTPUT:

MongoDB Run Save

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

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(
```

```
  {grades: { $all: [{ $elemMatch: { score: 2 } }, { $elemMatch: { score: 6 } }]}, borough: { $in:  
    ['Manhattan', 'Brooklyn'] },  
   cuisine: { $ne: 'American' }});
```

OUTPUT:

MongoDB interface showing a query in the left panel and its execution output in the right panel.

Query (Left Panel):

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

Output (Right Panel):

```
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(
```

```
  {grades: { $all: [ { $elemMatch: { score: 2 } }, { $elemMatch: { score: 6 } } ]}, borough: { $in: ['Manhattan', 'Brooklyn'] }, cuisine: { $nin: ['American', 'Chinese'] }});
```

OUTPUT:

MongoDB interface showing a query in the left panel and its execution output in the right panel.

Query (Left Panel):

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

Output (Right Panel):

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

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

QUERY:

```
db.restaurants.find(
```

```
  { $or: [ { "grades.score": 2 }, { "grades.score": 6 } ]});
```

OUTPUT:

MongoDB interface showing a query in the left panel and its execution output in the right panel.

Query (Left Panel):

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

Output (Right Panel):

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

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

MONGODB

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 code editor with the following query:

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

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

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

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

QUERY:

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

OUTPUT:

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

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

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

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

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

QUERY:

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

OUTPUT:

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

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

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

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

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

QUERY:

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

OUTPUT:

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

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

On the right, there is an "Output" panel with the following text:

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

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

QUERY:

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

OUTPUT:

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

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

On the right, there is an "Output" panel with the following text:

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

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

QUERY:

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

OUTPUT:

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

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

On the right, there is an "Output" panel with the following text:

```
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:

MongoDB Run Save

```
1 db.movies.find(
2   { "imdb.votes": { $gt: 1000 } }
3 );
```

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:

MongoDB Run Save

```
1 db.movies.find(
2   { "imdb.rating": { $gt: 7 } }
3 );
```

Output

```
mycompiler_mongodb> ... ...
mycompiler_mongodb>
```

[Execution complete with exit code 0]

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

QUERY:

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

OUTPUT:

MongoDB Run Save

```
1 db.movies.find(
2   { "tomatoes.viewer.rating": { $gt: 4 } }
3 );
```

Output

```
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:

MongoDB Run Save

```
1 db.movies.find(
2   { "awards.wins": { $gt: 0 } }
3 );
```

Output

```
mycompiler_mongodb> ... ...
mycompiler_mongodb>
```

[Execution complete with exit code 0]

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code block contains the query for finding movies with at least one nomination. On the right, the 'Output' panel shows the command being run and the message '[Execution complete with exit code 0]'.

```
1 db.movies.find(  
2   { "awards.nominations": { $gt: 0 } },  
3   {  
4     title: 1, languages: 1, released: 1, directors: 1, writers: 1,  
5     awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1  
6   }  
7 );  
8
```

Output

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code block contains the query for finding movies where 'Charles Kayser' is in the cast. On the right, the 'Output' panel shows the command being run and the message '[Execution complete with exit code 0]'.

```
1 db.movies.find(  
2   { cast: 'Charles Kayser' },  
3   {  
4     title: 1, languages: 1, released: 1, directors: 1, writers: 1,  
5     awards: 1, year: 1, genres: 1, runtime: 1, cast: 1, countries: 1  
6   }  
7 );  
8
```

Output

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

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

QUERY:

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

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code block contains the query for finding movies released on May 9, 1893. On the right, the 'Output' panel shows the command being run and the message '[Execution complete with exit code 0]'.

```
1 db.movies.find(  
2   { released: new ISODate("1893-05-09T00:00:00.000Z") },  
3   {  
4     title: 1, languages: 1, released: 1, directors: 1, writers: 1,  
5     countries: 1  
6   }  
7 );  
8
```

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: { $regex: 'scene', $options: 'i' } },  
  {title: 1, languages: 1, released: 1, directors: 1, writers: 1,countries: 1});
```

OUTPUT:

The screenshot shows the MongoDB shell interface. On the left, the code block contains the MongoDB query. On the right, the 'Output' panel shows the command being run and the response indicating the operation completed successfully.

```
1 db.movies.find(  
2   { title: { $regex: 'scene', $options: 'i' } },  
3   {  
4     title: 1, languages: 1, released: 1, directors: 1, writers: 1,  
5     countries: 1  
6   }  
7 );  
8
```

Output

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

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