

## 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 DEPT21(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. The SQL command entered is:

```
1 create table DEPT21 (dept_id number(6),dept_name varchar(20),manager_id number(6),location_id number(4));
```

The command was executed successfully, and the results pane shows:

```
Table created.
0.03 seconds
```

The interface also displays the schema name 'WKSP\_APEX13451' and the user 'AKASHU218'.

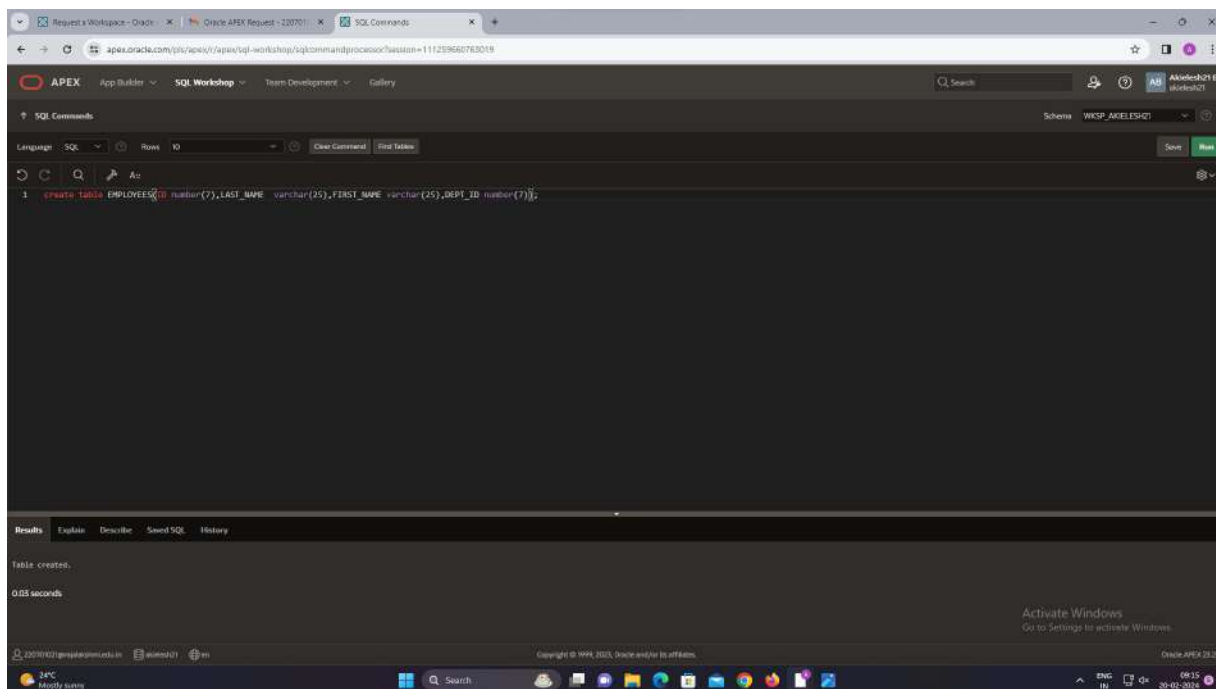
2.Create the EMP table based on the following instance chart. Confirm that the table is created.

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 EMPLOYEES(id number(7), last\_name varchar(25), first\_name varchar(25), dept\_id number(7));

### OUTPUT:

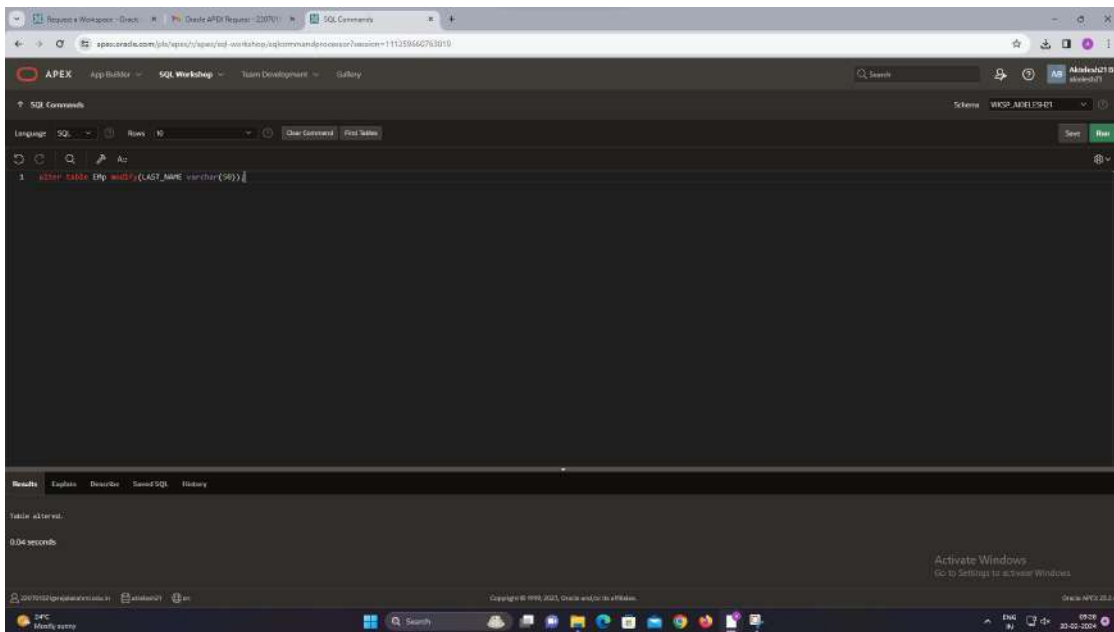


3.Modify the EMP table to allow for longer employee last names. Confirm the modification.(Hint: Increase the size to 50)

### QUERY:

Alter table emp modify(last\_name varchar(50);

### OUTPUT:

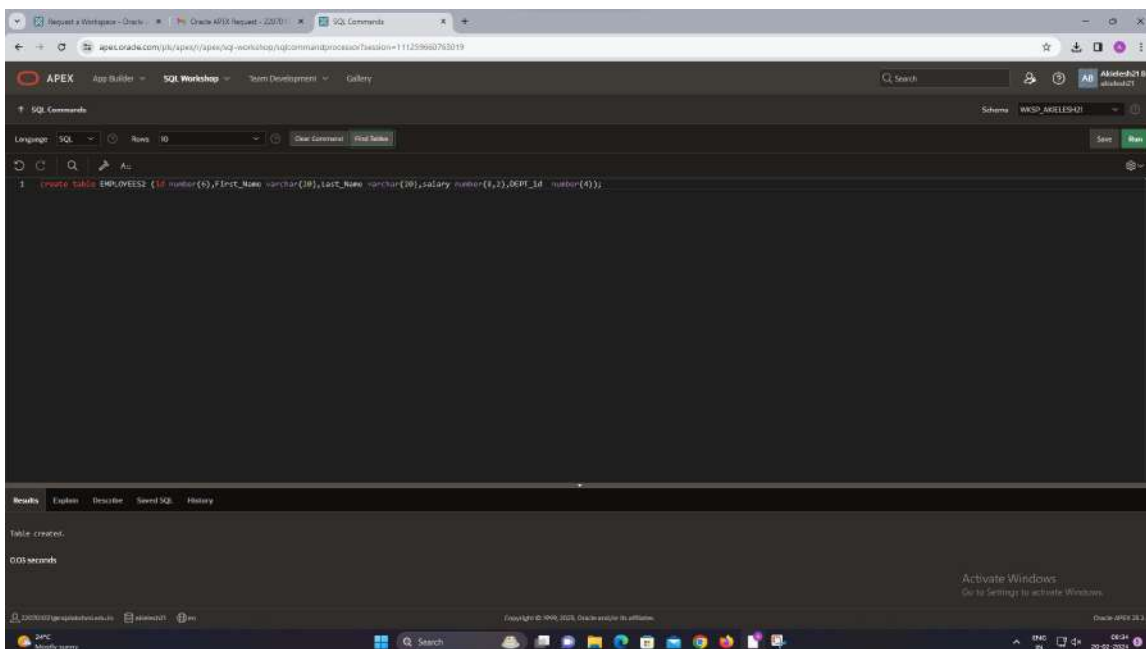


4. Create the EMPLOYEES2 table based on the structure of EMPLOYEES table. Include Only the Employee\_id, First\_name, Last\_name, Salary and Dept\_id columns. Name the columns Id, First\_name, Last\_name, salary and Dept\_id respectively.

### QUERY:

Create table employees2(id number(6), first\_name varchar(20), last\_name varchar(25), salary number(8,2), dept\_id number(4));

### OUTPUT:

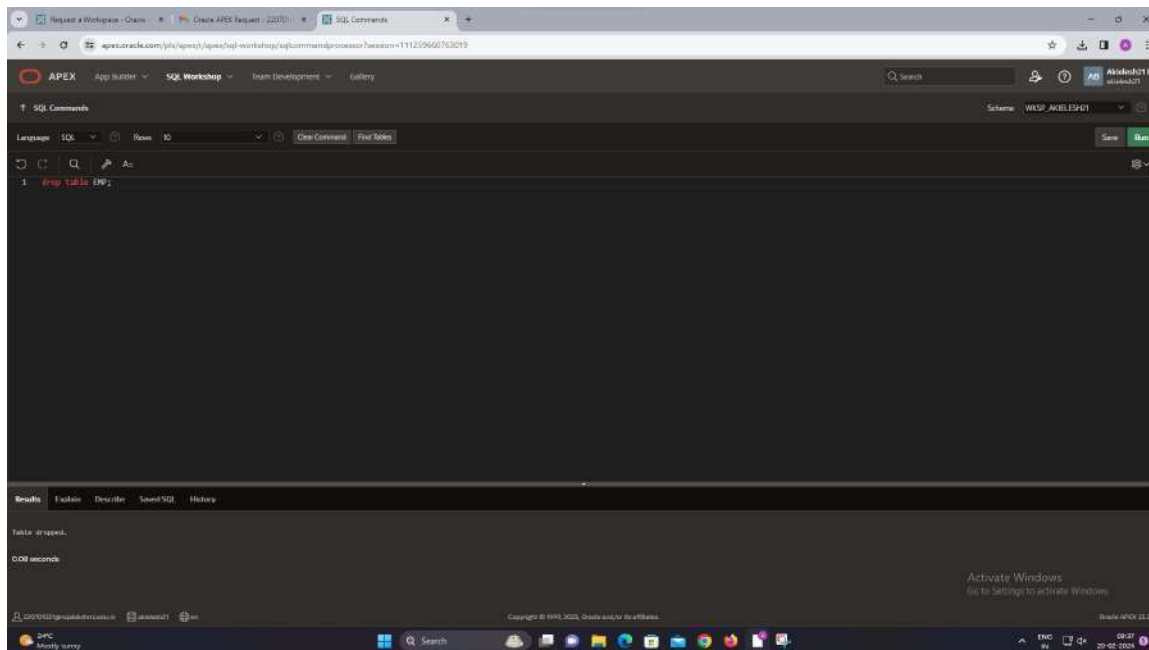


5. Drop the EMP table.

## QUERY:

Drop table emp;

## OUTPUT:

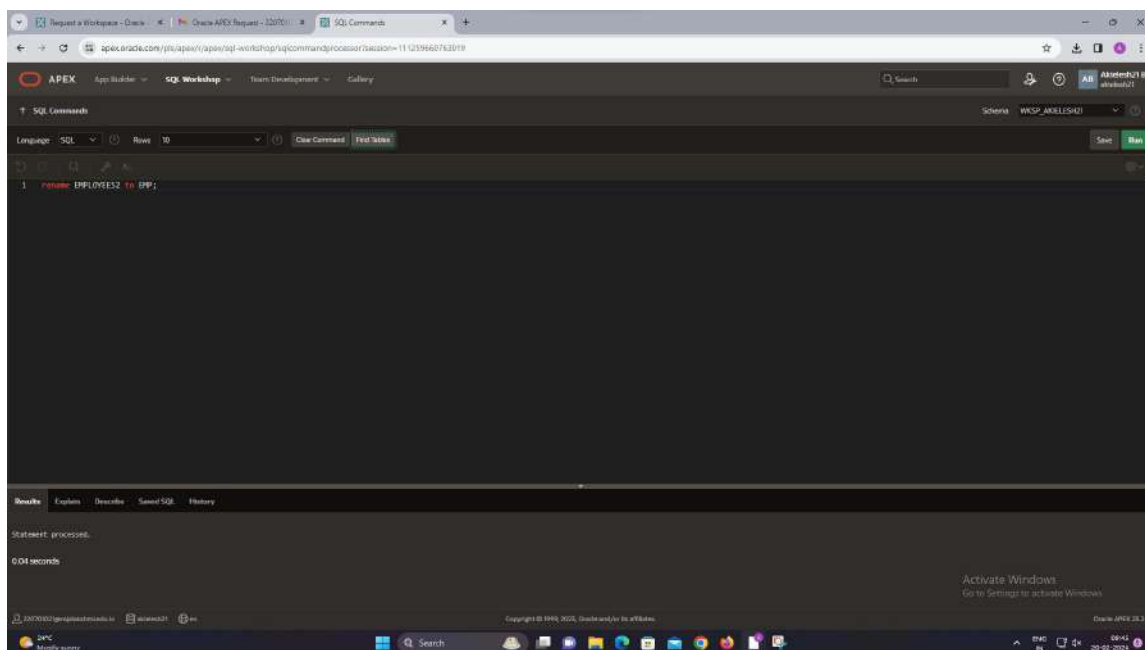


6.Rename the EMPLOYEES2 table as EMP.

## QUERY:

Rename employees2 to emp;

## OUTPUT:



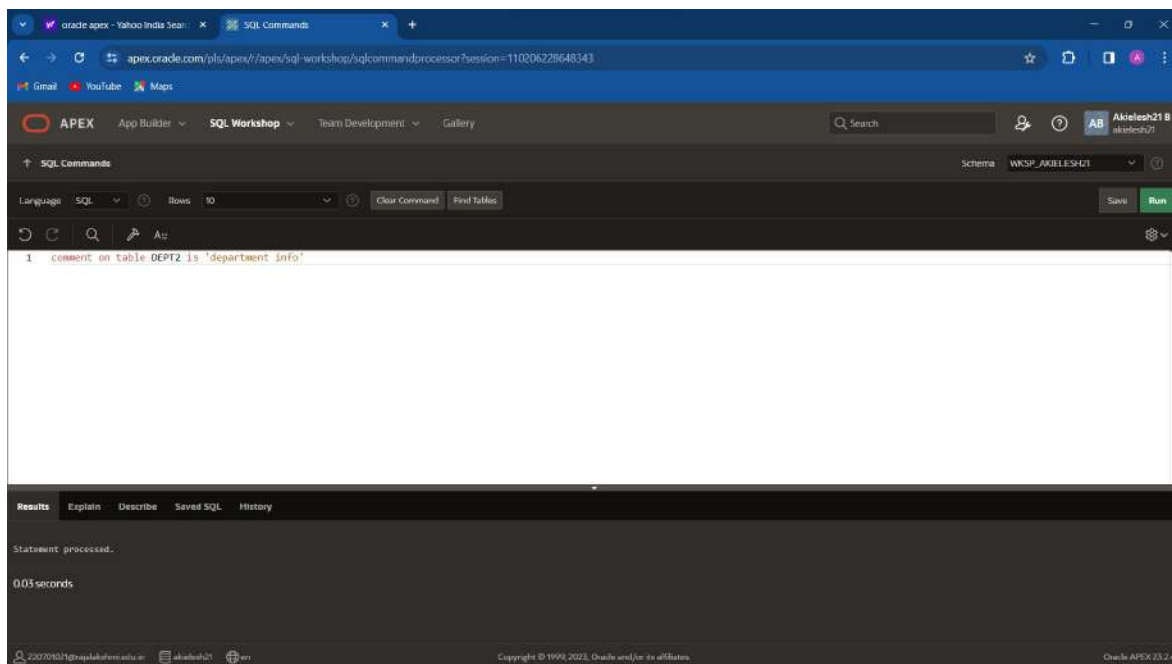
7. Add a comment on DEPT and EMP tables. Confirm the modification by describing the table.

**QUERY:**

Comment on table emp as 'employee info'

Comment on table dept as ‘department info’;

**OUTPUT:**

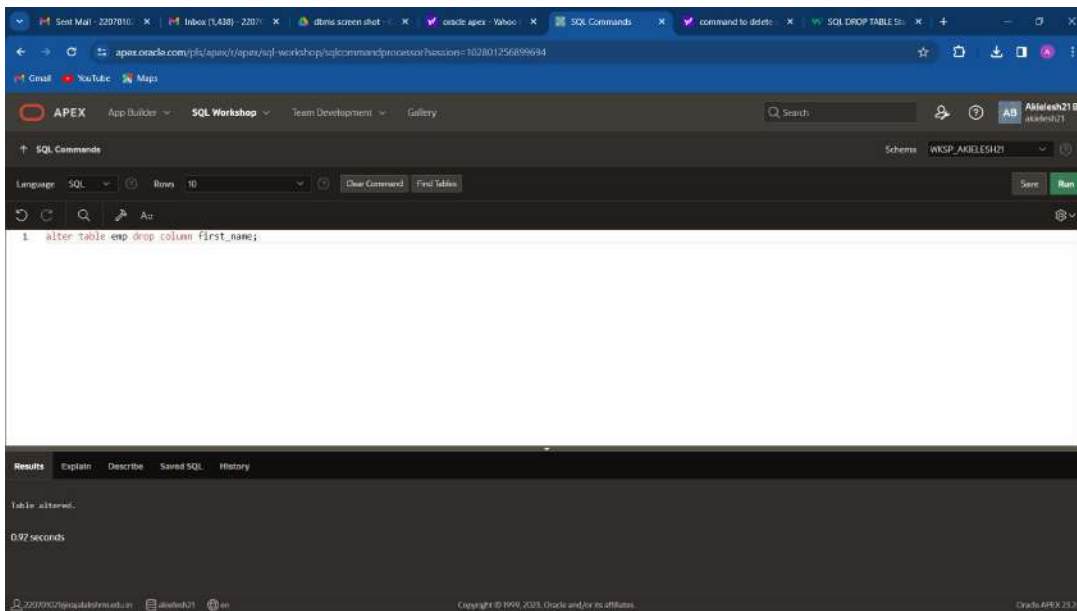


8.Drop the First\_name column from the EMP table and confirm it.

**QUERY:**

```
Alter table emp drop column first_name;
```

**OUTPUT:**



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

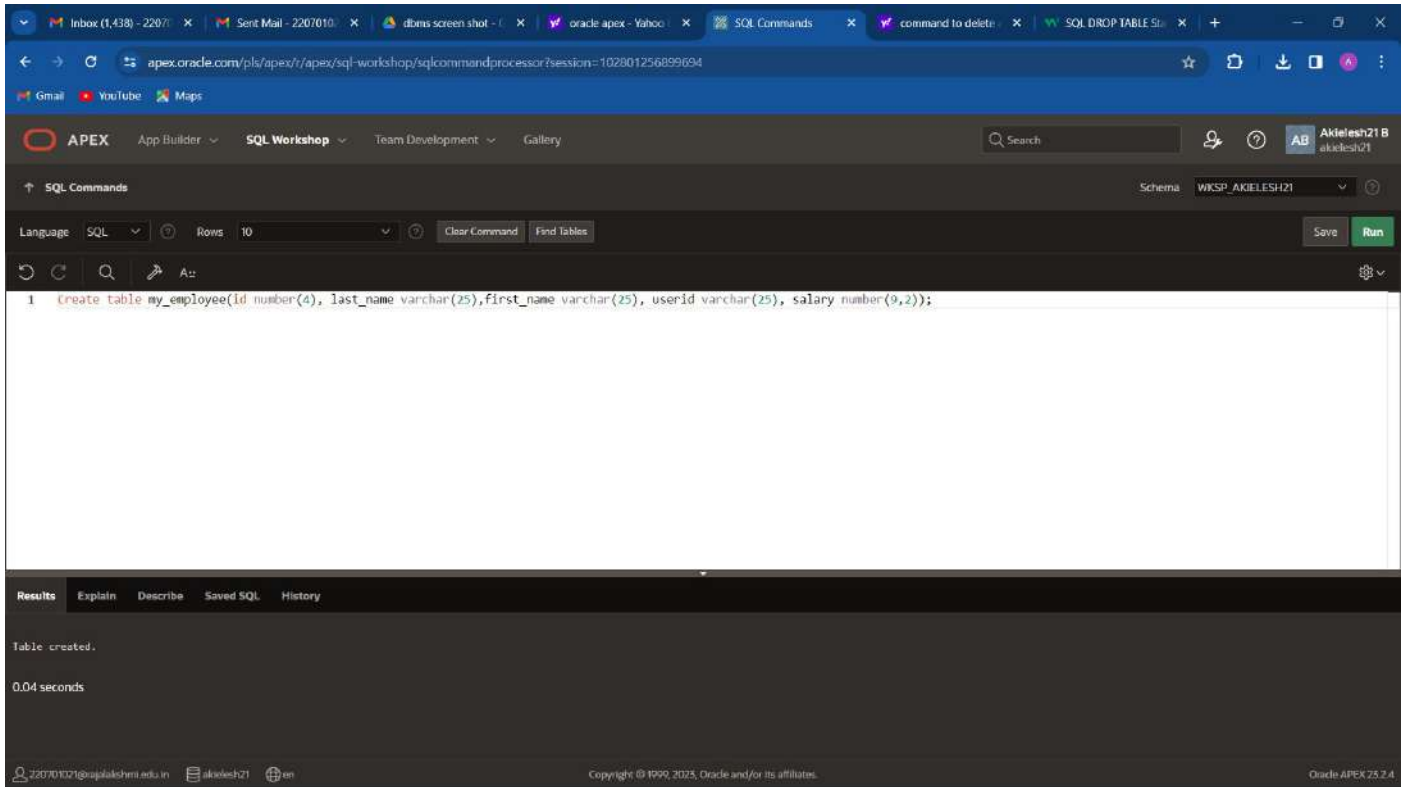
**MANIPULATING DATA**

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),first\_name varchar(25), userid varchar(25), salary number(9,2));

**OUTPUT:**

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

ID	Last_name	First_name	Userid	salary
----	-----------	------------	--------	--------

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

### QUERY:

Insert into my\_employee values(1,'Patel','Ralph', 'rpatel', 895);

Insert into my\_employee values(2,'Dancs','Betty', 'bdancs', 860);

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The main area displays the 'SQL Commands' tab with a schema dropdown set to 'WKSP\_AKIELESH21'. The SQL command entered is: `1 insert into my_employee values(1,'Patel','Ralph', 'rpatel', 895);`. Below the command, the 'Results' tab is active, showing the output: `1 row(s) inserted.` and `0.04 seconds`. The bottom status bar indicates the user is logged in as 'akiesh21' and the version is 'Oracle APEX 25.2.4'.

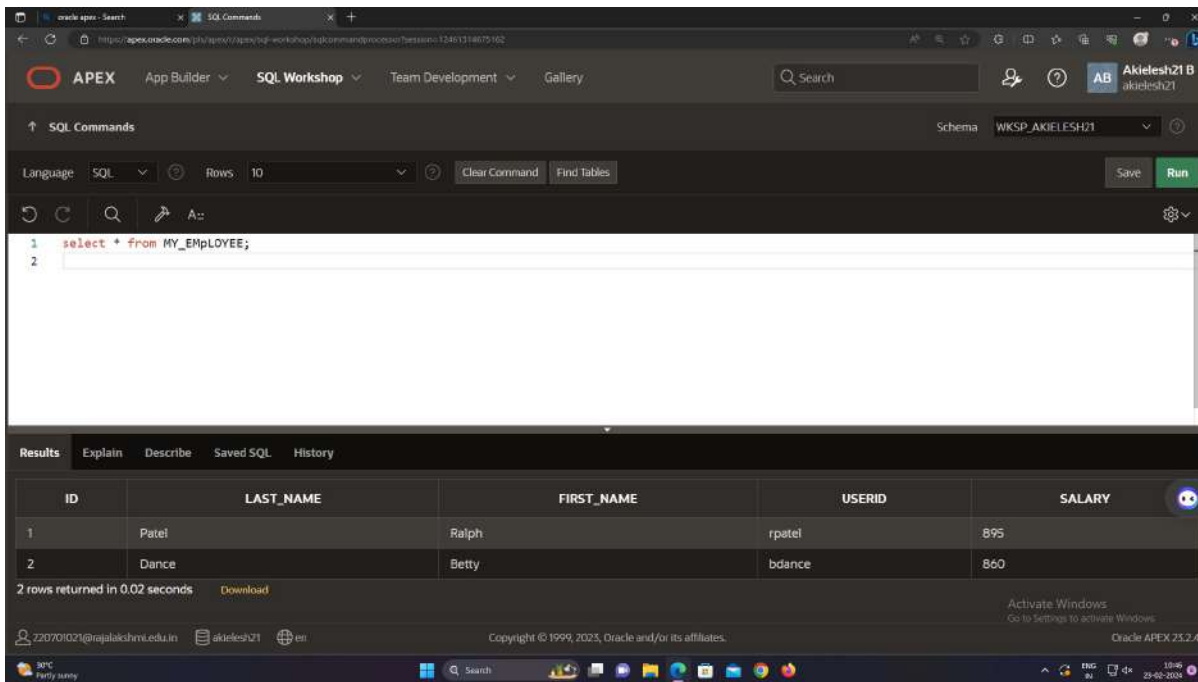
3.Display the table with values.

### QUERY:

Select \* from my\_employee;

### OUTPUT:





Oracle APEX SQL Workshop interface showing the execution of a SQL query. The query is:

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

The results are displayed in a table with 5 columns: ID, LAST\_NAME, FIRST\_NAME, USERID, and SALARY. The results show 2 rows returned in 0.02 seconds.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dance	Betty	bdance	860

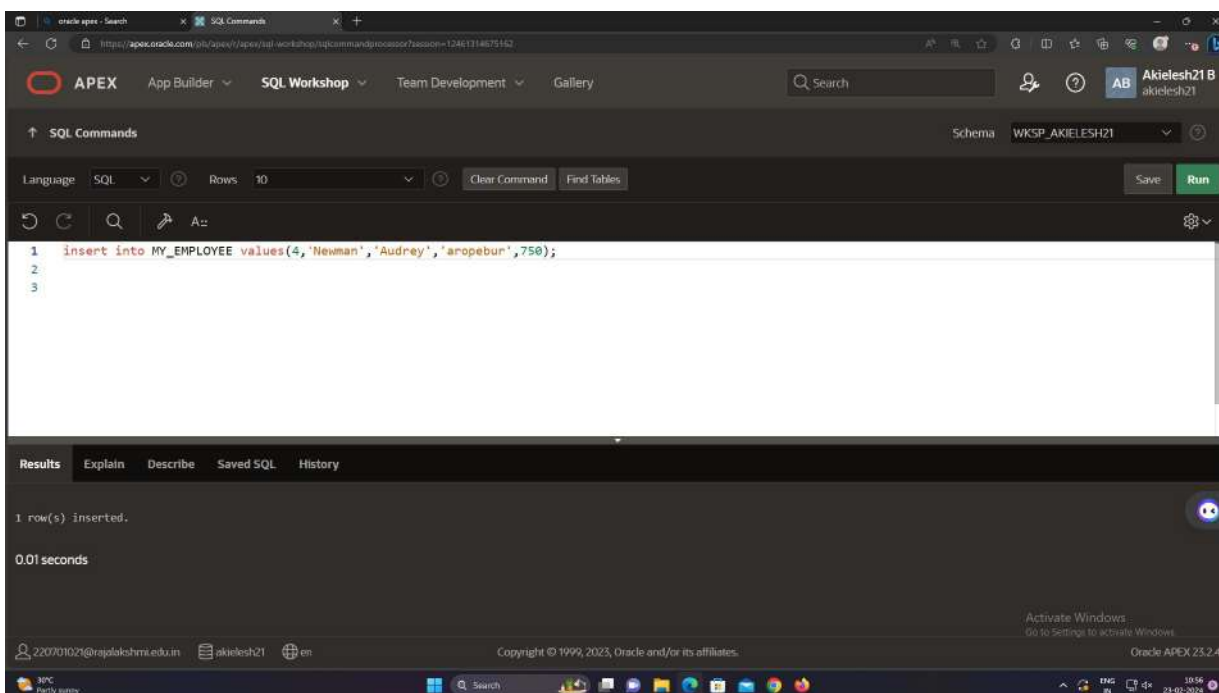
4. Populate the next two rows of data from the sample data. Concatenate the first letter of the first\_name with the first seven characters of the last\_name to produce Userid.

#### QUERY:

Insert into MY\_EMPLOYEE values(3,'Biri', 'Ben', 'bbiri', 1100);

Insert into MY\_EMPLOYEE values(4,'Newman', 'Audrey', 'aropebur',750);

#### OUTPUT:



Oracle APEX SQL Workshop interface showing the execution of two SQL queries. The first query is:

```
1 insert into MY_EMPLOYEE values(3,'Biri', 'Ben', 'bbiri', 1100);
```

The results show 1 row(s) inserted in 0.01 seconds.

The second query is:

```
2 insert into MY_EMPLOYEE values(4,'Newman', 'Audrey', 'aropebur',750);
```

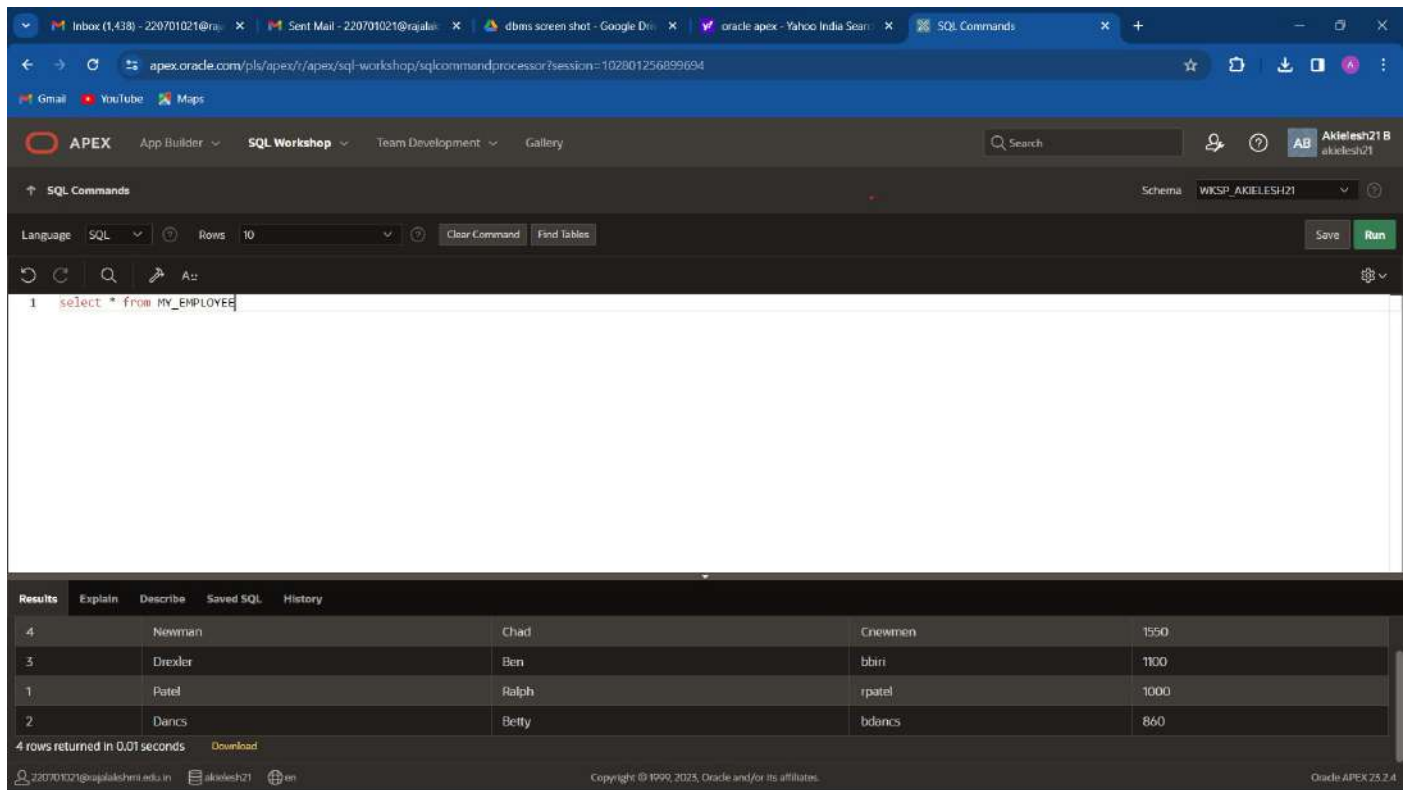
The results show 1 row(s) inserted in 0.01 seconds.

5. Make the data additions permanent.

## QUERY:

Select \* from MY\_EMPLOYEE;

## OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is `select * from MY_EMPLOYEE`. The results are displayed in a table with 4 rows. The table has columns for ID, Last Name, First Name, and Salary. The data is as follows:

ID	Last Name	First Name	Salary
4	Newman	Chad	1550
3	Drexler	Ben	1100
1	Patel	Ralph	1000
2	Dancs	Betty	860

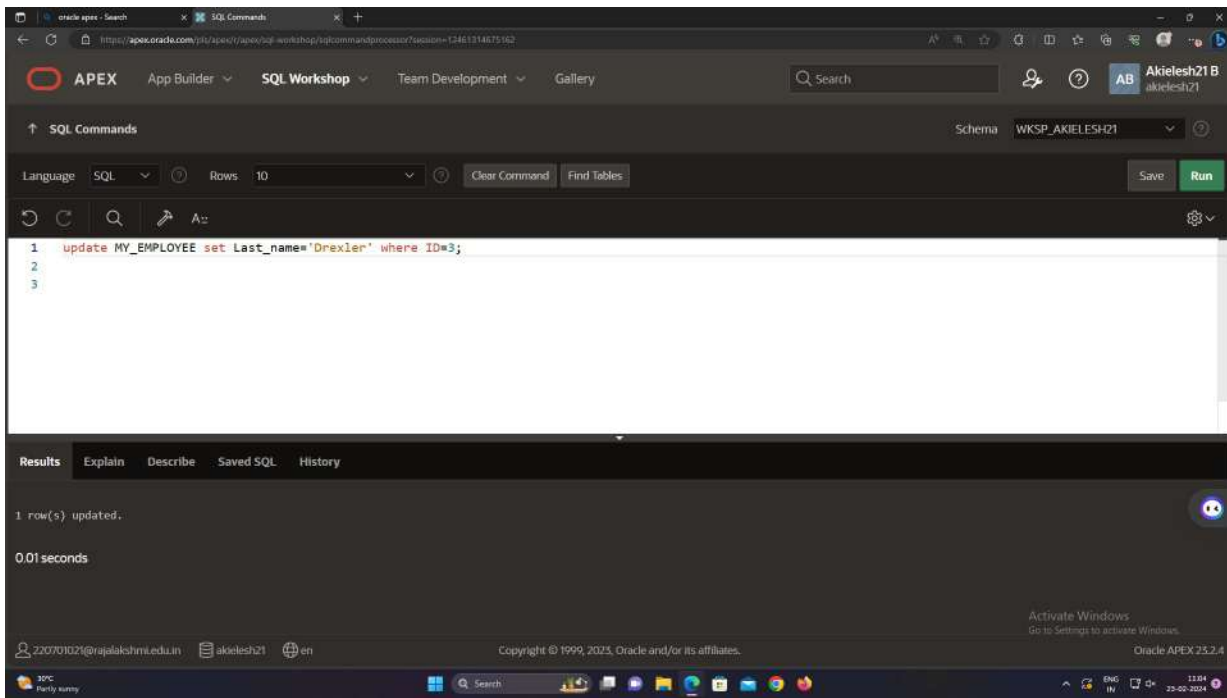
4 rows returned in 0.01 seconds. Download

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

## QUERY:

update my\_employee set last\_name='Drexler' where ID=3;

## OUTPUT:

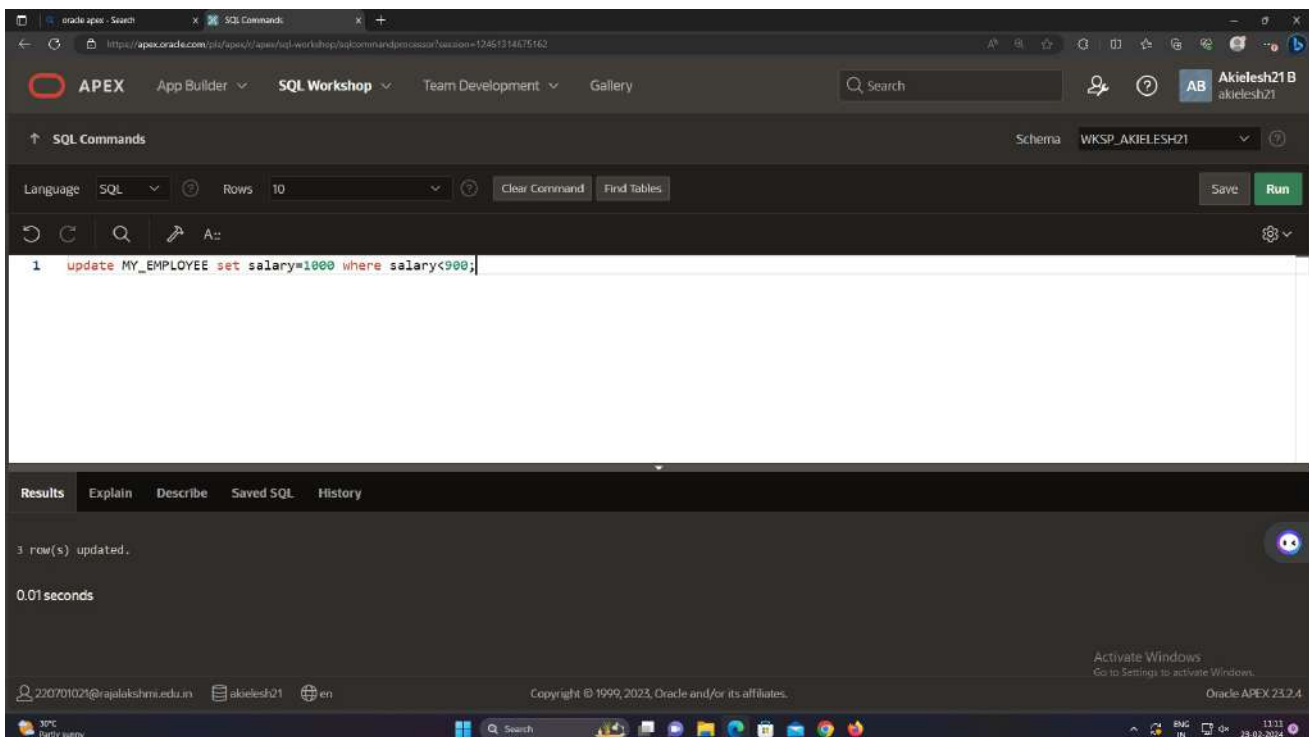


7.Change the salary to 1000 for all the employees with a salary less than 900.

**QUERY:**

Update my\_employee set salary=1000 where salary<900;

**OUTPUT:**

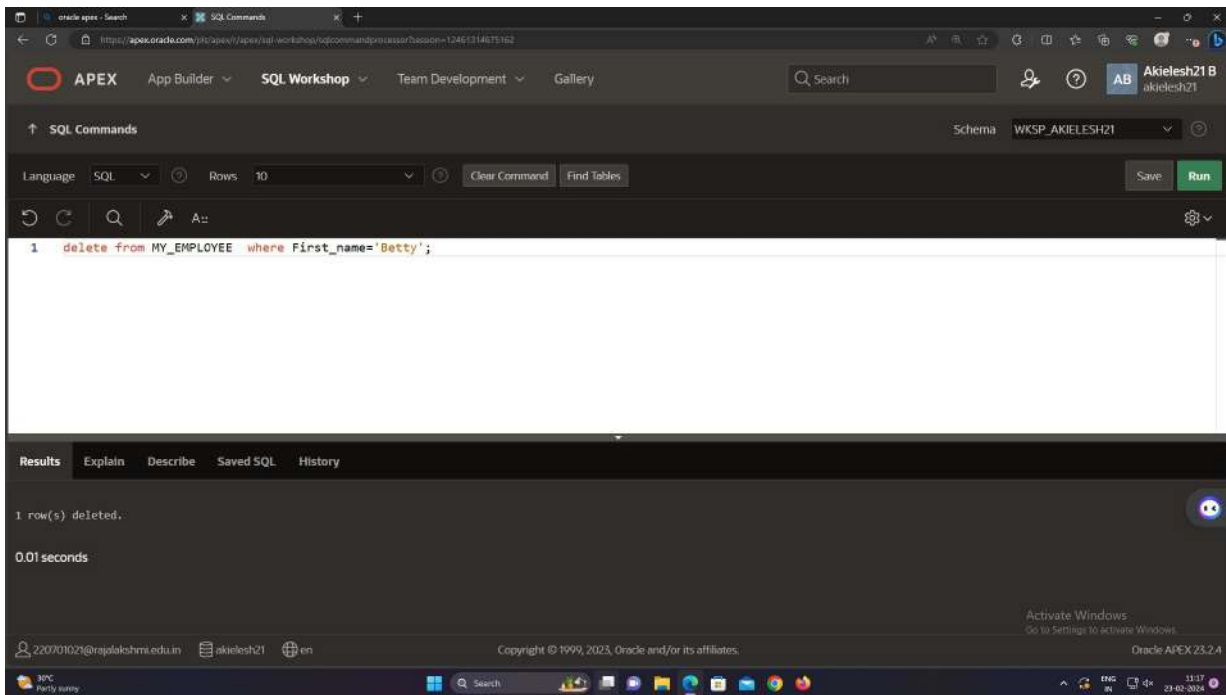


8.Delete Betty dancs from MY \_EMPLOYEE table.

**QUERY:**

Delete from my\_employee where first\_name='Betty';

## OUTPUT:

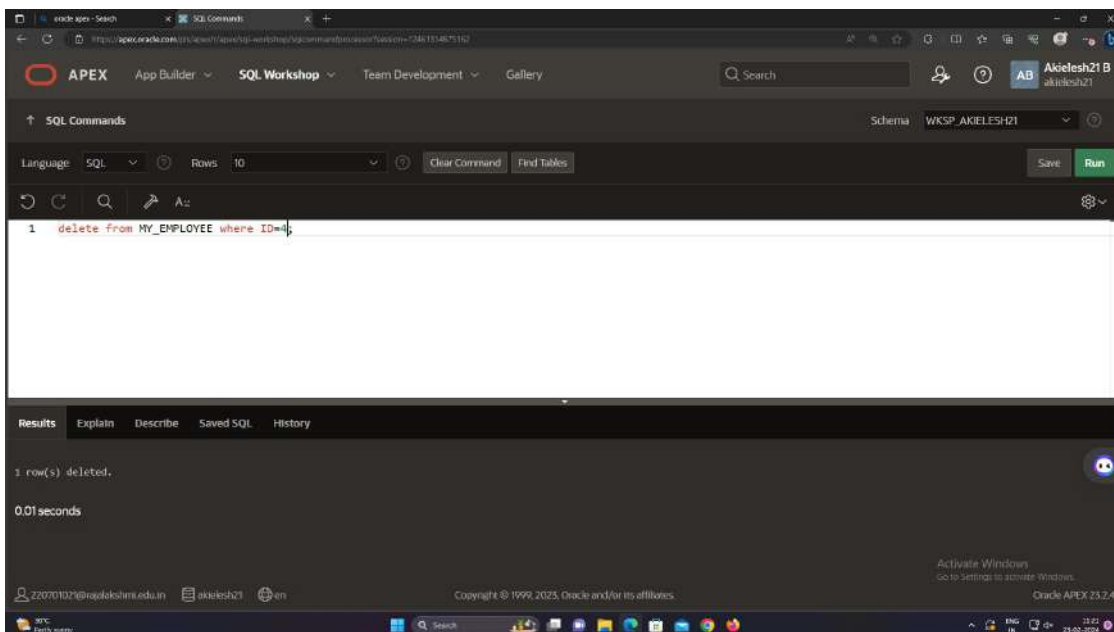


9. Empty the fourth row of the emp table.

## QUERY:

Delete from my\_employee where ID=4;

## OUTPUT:



Evaluation Procedure	Marks awarded
Query(5)	

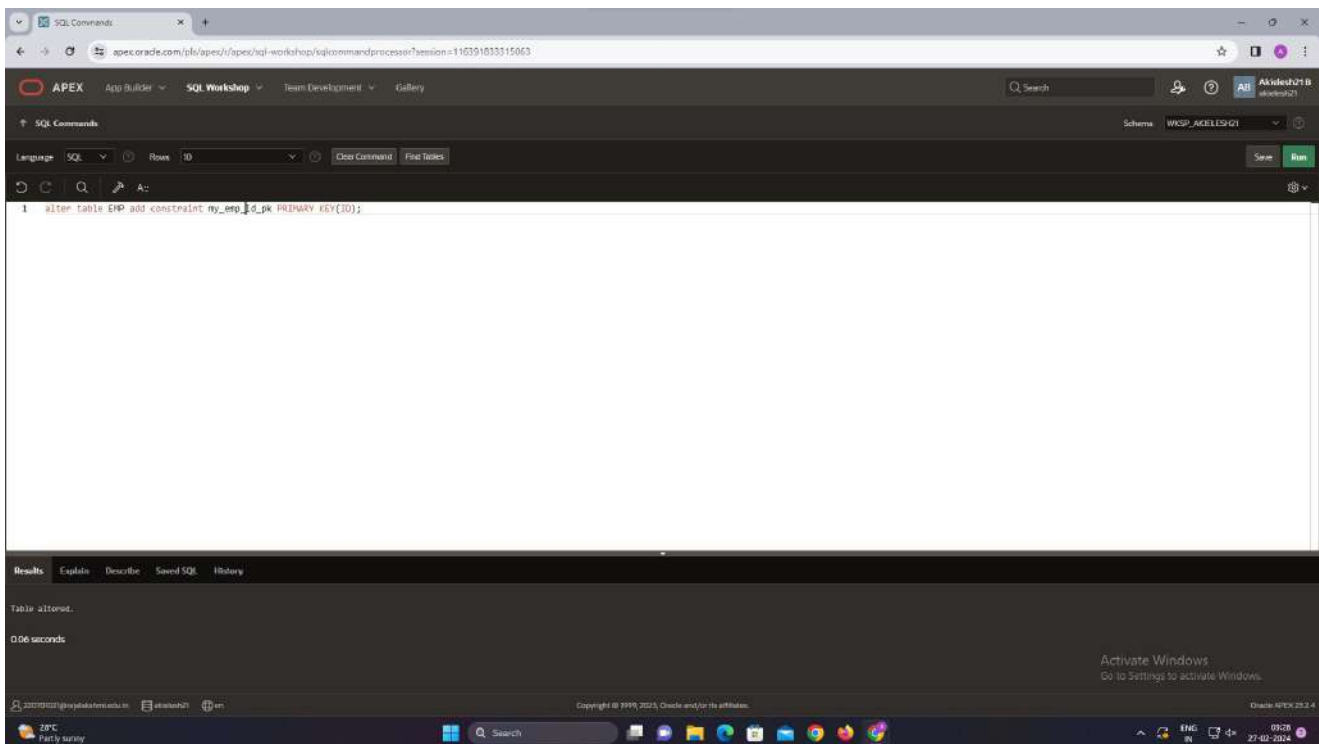
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

**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:**

Alter table EMP add constraint my\_emp\_id\_pk PRIMARY KEY(ID);

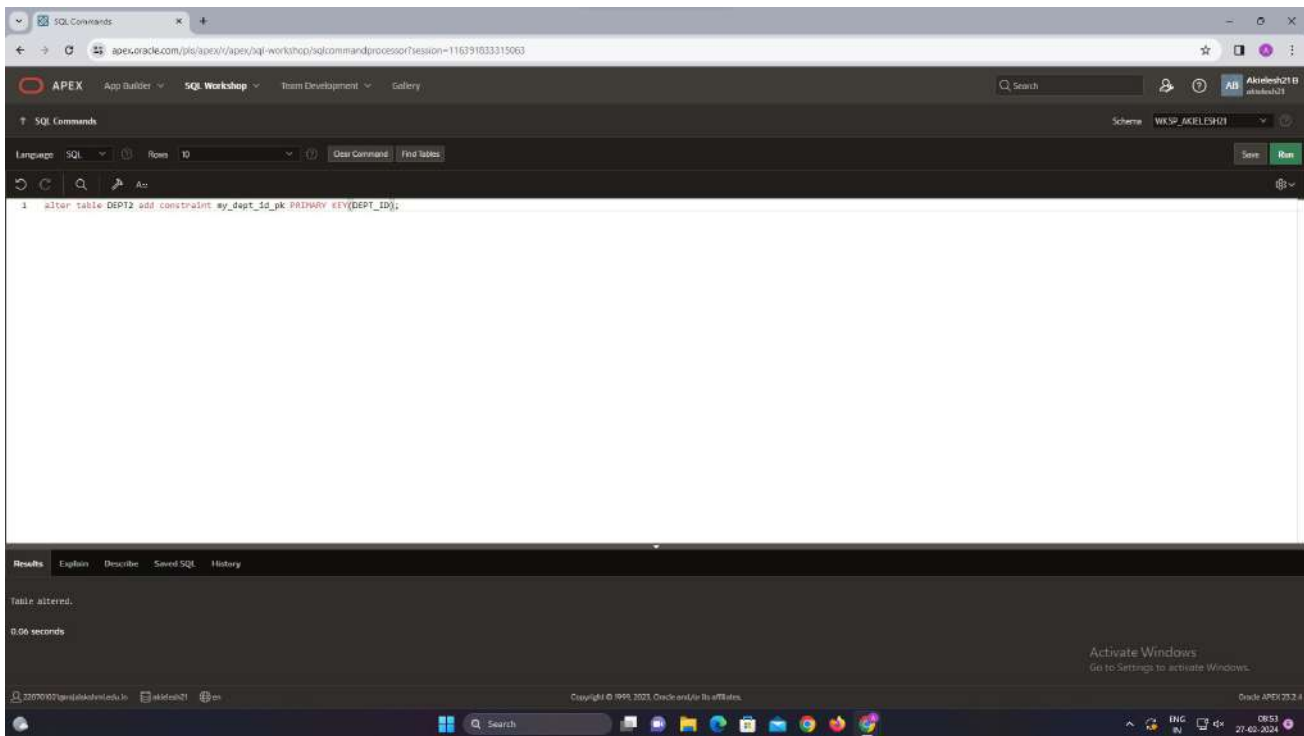
**OUTPUT:**

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

**QUERY:**

Alter table DEPT2 add constraint my\_dept\_id\_pk primary key(DEPT\_id);

**OUTPUT:**

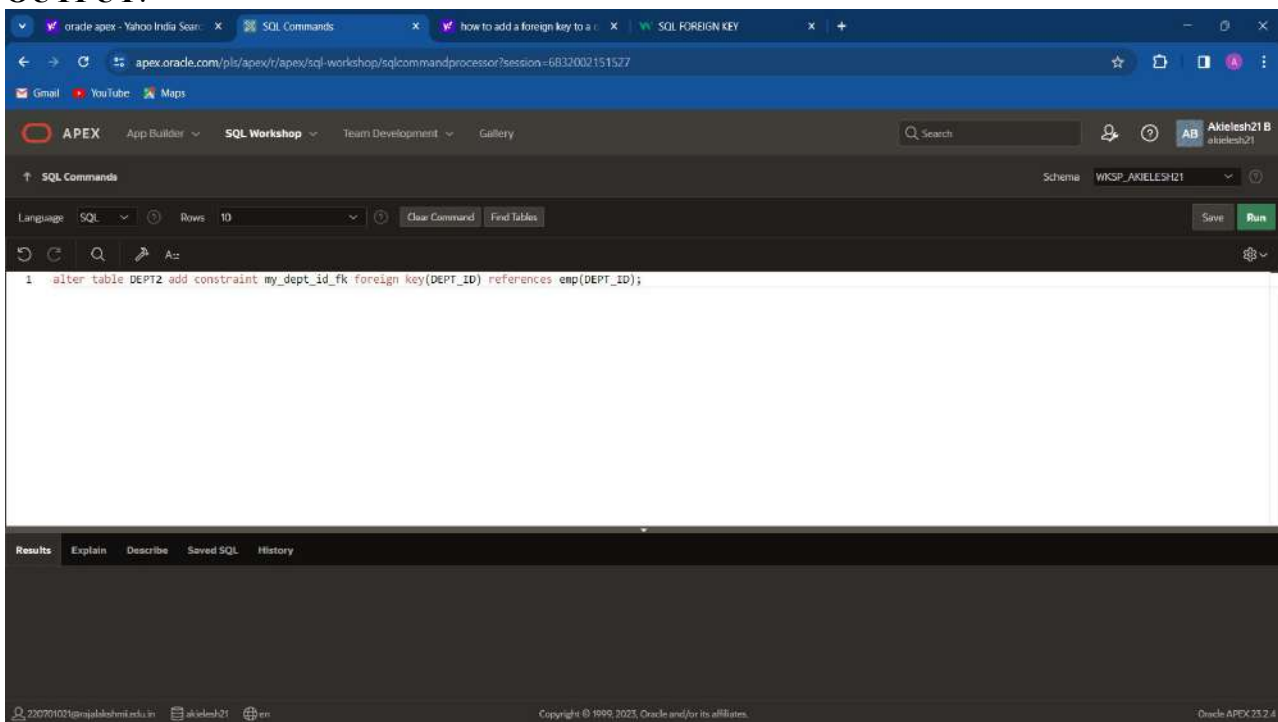


3. Add a column `DEPT_ID` to the `EMP` table. Add a foreign key reference on the `EMP` table that ensures that the employee is not assigned to nonexistent department. Name the constraint `my_emp_dept_id_fk`.

#### QUERY:

Alter table `DEPT2` add constraint `my_emp_dept_id_fk` foreign key(`DEPT_ID`) references `emp(DEPT_ID)`;

#### OUTPUT:

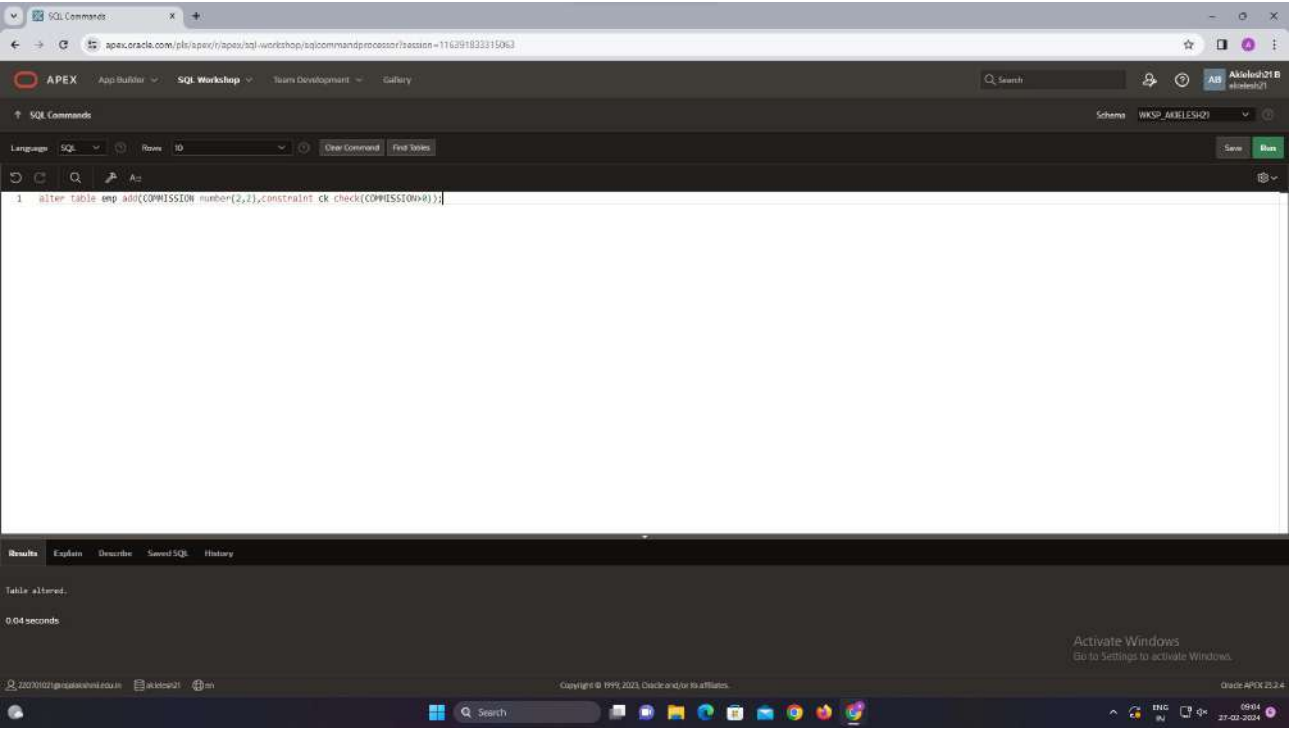


4. Modify the `EMP` table. Add a `COMMISSION` column of `NUMBER` data type, precision 2, scale 2. Add a constraint to the commission column that ensures that a commission value is greater than zero.

#### QUERY:

Alter table `emp` add (commission number(2,2), constraint `ck` check(commission>0);

OUTPUT:



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



# WRITING BASIC SQL SELECT STATEMENTS

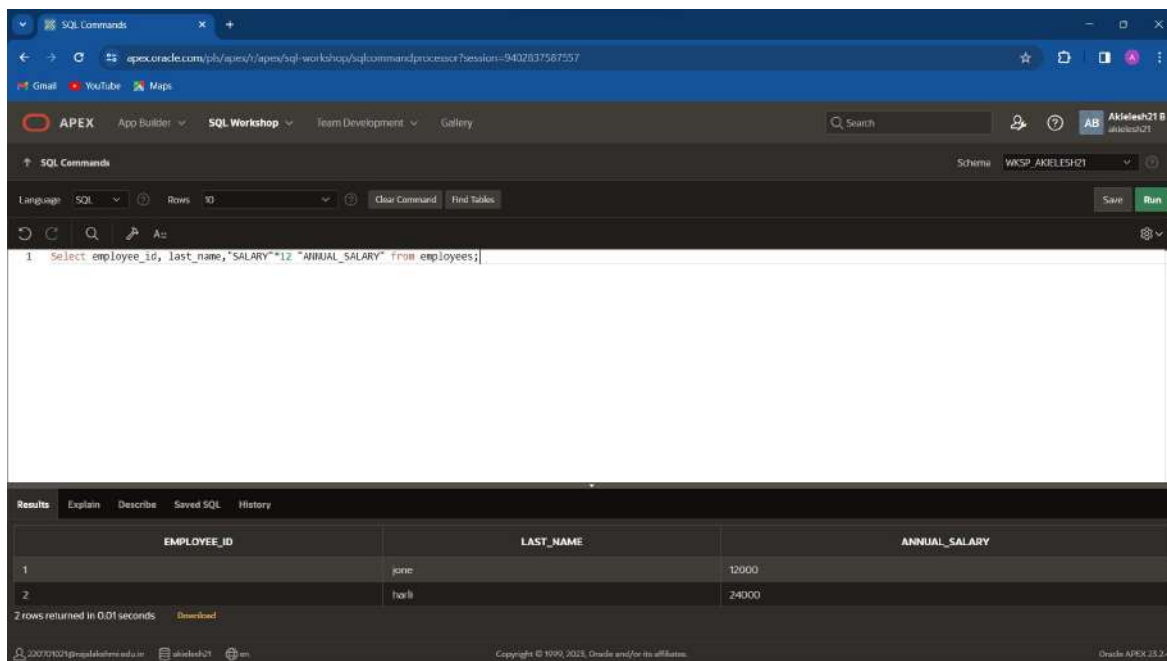
1. The following statement executes successfully.

## 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:



The screenshot displays the Oracle APEX SQL Workshop interface. The SQL command entered is: `SELECT employee_id, last_name, 'sal*'12 'ANNUAL_SALARY' FROM employees;`. The results are shown in a table with three columns: EMPLOYEE\_ID, LAST\_NAME, and ANNUAL\_SALARY. The table contains two rows of data.

EMPLOYEE_ID	LAST_NAME	ANNUAL_SALARY
1	jone	12000
2	harli	24000

2 rows returned in 0.01 seconds

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

## QUERY:

```
select * from employees;
```

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The query entered is `select * from employees;`. The results are displayed in a table with the following columns: EMPLOYEE\_ID, LAST\_NAME, FIRST\_NAME, DEPT\_ID, SALARY, EMAIL, JOB\_CODE, HIRE\_DATE, MANAGER\_ID, and PHONE\_NO. Two rows are returned.

EMPLOYEE_ID	LAST_NAME	FIRST_NAME	DEPT_ID	SALARY	EMAIL	JOB_CODE	HIRE_DATE	MANAGER_ID	PHONE_NO
1	jone	david	101	1000	jone@gmail.com	101	10/09/2004	808	9876543212
2	harli	jim	102	2000	jim@gmail.com	102	04/06/2005	909	8903428945

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\_code, hire\_date from employees;

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The query entered is `select employee_id as employee_number, last_name, job_code, hire_date from employees;`. The results are displayed in a table with the following columns: EMPLOYEE\_NUMBER, LAST\_NAME, JOB\_CODE, and HIRE\_DATE. Two rows are returned.

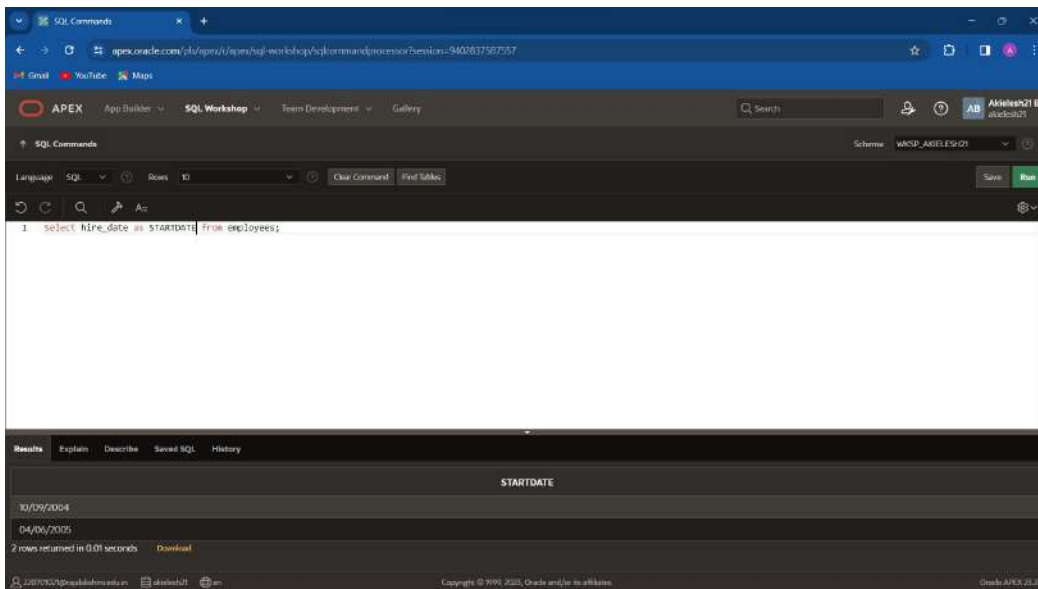
EMPLOYEE_NUMBER	LAST_NAME	JOB_CODE	HIRE_DATE
1	jone	101	10/09/2004
2	harli	102	04/06/2005

4. Provide an alias STARTDATE for the hire date.

#### QUERY:

Select hire\_date as startdate from employees;

#### OUTPUT:

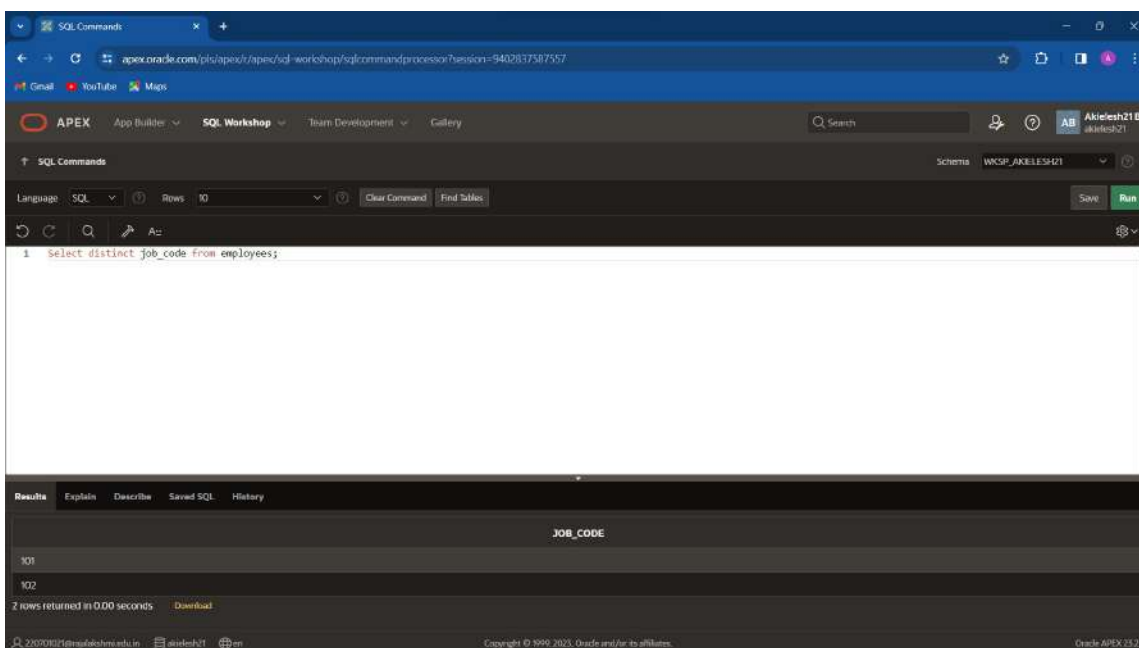


5.Create a query to display unique job codes from the employee table.

**QUERY:**

Select distinct job\_code from employees;

**OUTPUT:**



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:**

SQL Commands

apex.oracle.com/pls/apex/f/apex/sql-workshop/sqlcommandprocessor?session=9402837587557

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language: SQL Rows: 10 Clear Command First Tables Save Run

```
1 select last_name||','||job_code|| as "employee_and_title" from employees;
```

Results Explain Describe Saved SQL History

employee\_and\_title

jone301
har102

2 rows returned in 0.00 seconds Download

Copyright © 1990, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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:

SQL Commands

apex.oracle.com/pls/apex/f/apex/sql-workshop/sqlcommandprocessor?session=110206226648343

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language: SQL Rows: 10 Clear Command First Tables Save Run

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

Results Explain Describe Saved SQL History

the\_output

1, david, jones, jones@gmail.com, 950545732, 30/09/2004, 101, 3000, 808, 101
2, jim, har1, jim@gmail.com, 8905428945, 04/06/2005, 102, 2000, 900, 102

2 rows returned in 0.01 seconds Download

Copyright © 1990, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

# RESTRICTING AND SORTING DATA

EX\_NO:5

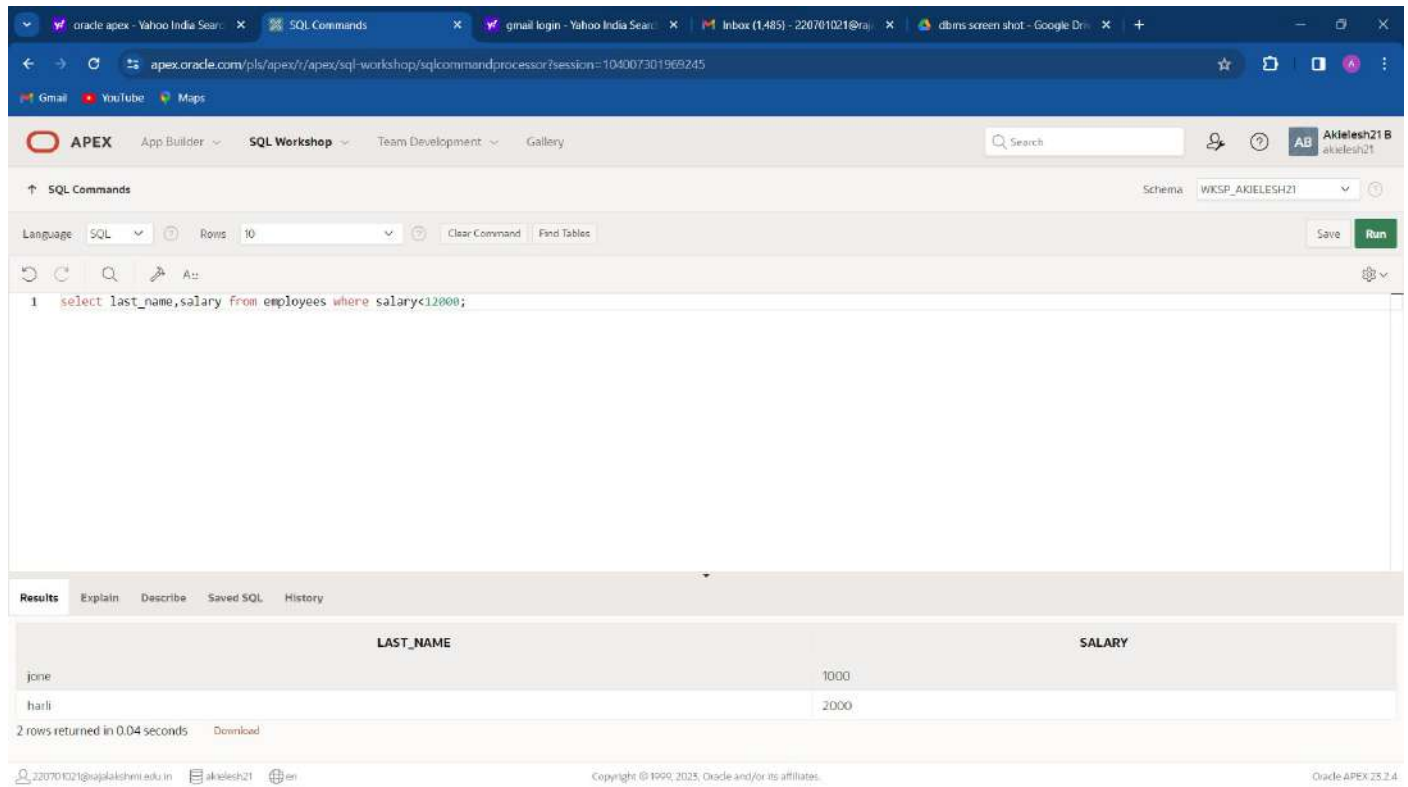
DATE:

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

## QUERY:

Select last\_name from employees where salary>12000;

## OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is: `select last_name,salary from employees where salary<12000;`. The results table displays two rows: one for 'jone' with a salary of 1000, and one for 'harli' with a salary of 2000. The interface includes a search bar, a schema dropdown set to 'WKSP\_AKIELESH21', and a 'Run' button. The bottom of the screen shows the user '220701021@rajalakshmi.edu.in' and the Oracle APEX version '23.2.4'.

LAST_NAME	SALARY
jone	1000
harli	2000

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

## QUERY:

Select last\_name,department\_id from employees where employee\_id=176;

## OUTPUT:

Oracle APEX SQL Workshop interface. The query entered is:

```
1 select last_name,dept_id as department_number from employees where employee_number=176;
```

The results table shows:

LAST_NAME	DEPARTMENT_NUMBER
jone	101

1 rows returned in 0.01 seconds

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:

Oracle APEX SQL Workshop interface. The query entered is:

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

The results table shows:

LAST_NAME	SALARY
tivari	12500
fang	13000
jone	1000

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 'February,20,1998' and 'May,1,1998';

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'SQL Commands' tab is active, displaying a query editor with the following SQL statement:

```
1 Select last_name,job_code,hire_date from employees where hire_date between 'February,20,1998' and 'May,1,1998'
```

Below the query editor, the 'Results' tab is selected, showing a table with the following data:

LAST_NAME	JOB_CODE	HIRE_DATE
Janu	101	03/04/1998

At the bottom of the results section, it states '1 rows returned in 0.01 seconds' and provides a 'Download' link. The footer of the interface includes the user's email address '220701021@rajalakshmi.edu.in', the username 'akiesh21', the language 'en', the copyright notice 'Copyright © 1999, 2023, Oracle and/or its affiliates.', and the version 'Oracle APEX 23.2.4'.

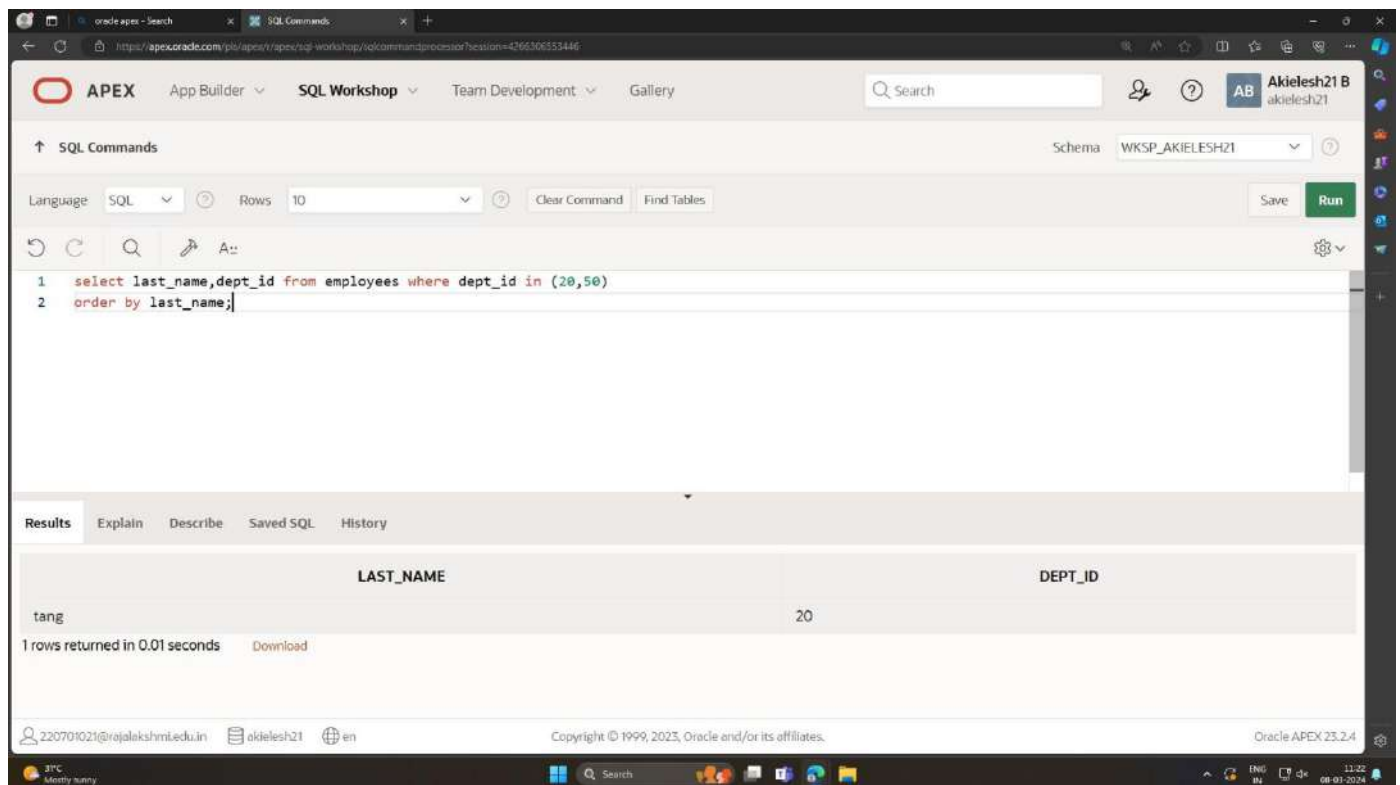


5. Display the last name and department number of all employees in departments 20 and 50 in alphabetical order by name.(hints: in, orderby)

**QUERY:**

select last\_name,department\_id from employees where department\_id in(20,50) order by last\_name;

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The user is logged in as 'Akiesh21 B' with the email 'akiesh21'. The 'SQL Commands' tab is active, showing a query in the editor:

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

The 'Results' tab is selected, displaying a table with two columns: 'LAST\_NAME' and 'DEPT\_ID'. The table contains one row with the values 'tang' and '20'. Below the table, it states '1 rows returned in 0.01 seconds' and provides a 'Download' link. The footer of the interface shows the user's email '220701021@rajalekshmi.edu.in', the username 'akiesh21', and the version 'Oracle APEX 23.2.4'.

LAST_NAME	DEPT_ID
tang	20

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 asc;

**OUTPUT:**

The screenshot shows the Oracle APEX 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 dept_id in (10,50)
2 order by last_name;
```

The results are displayed in a table with two columns: **EMPLOYEE** and **MONTHLY\_SALARY**.

EMPLOYEE	MONTHLY_SALARY
harli	6000
tony	11000

2 rows returned in 0.00 seconds. Download

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 '1994';

### OUTPUT:

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

```
1 select last_name,hire_date from employees where hire_date like '%1994';
```

The results are displayed in a table with two columns: **LAST\_NAME** and **HIRES DATE**.

LAST_NAME	HIRES DATE
harli	04/06/1994

1 rows returned in 0.00 seconds. Download

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

### QUERY

select last\_name,job\_id from employees where manager\_id is null;

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The user is logged in as 'Akielesh21 B'. The 'SQL Commands' tab is active, showing a query: `select last_name, job_title from employees where manager_id is null;`. The 'Run' button is green. Below the query, the 'Results' tab is selected, displaying a table with two columns: 'LAST\_NAME' and 'JOB\_TITLE'. The table contains one row with the values 'harli' and 'staff'. Below the table, it says '1 rows returned in 0.01 seconds' and 'Download'.

LAST_NAME	JOB_TITLE
harli	staff

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

## QUERY:

select last\_name,salary,commission\_pct from employees where commission\_pct is not null order by salary desc;

## OUTPUT:

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

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

The results table shows 3 rows returned in 0.03 seconds:

LAST_NAME	SALARY	COMMISSION
harli	6000	500
tang	11000	2000
tkoark	12500	1000

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 APEX SQL Workshop interface. The query entered is:

```
1 select last_name from employees where last_name like '__a%';
```

The results table shows 1 row returned in 0.02 seconds:

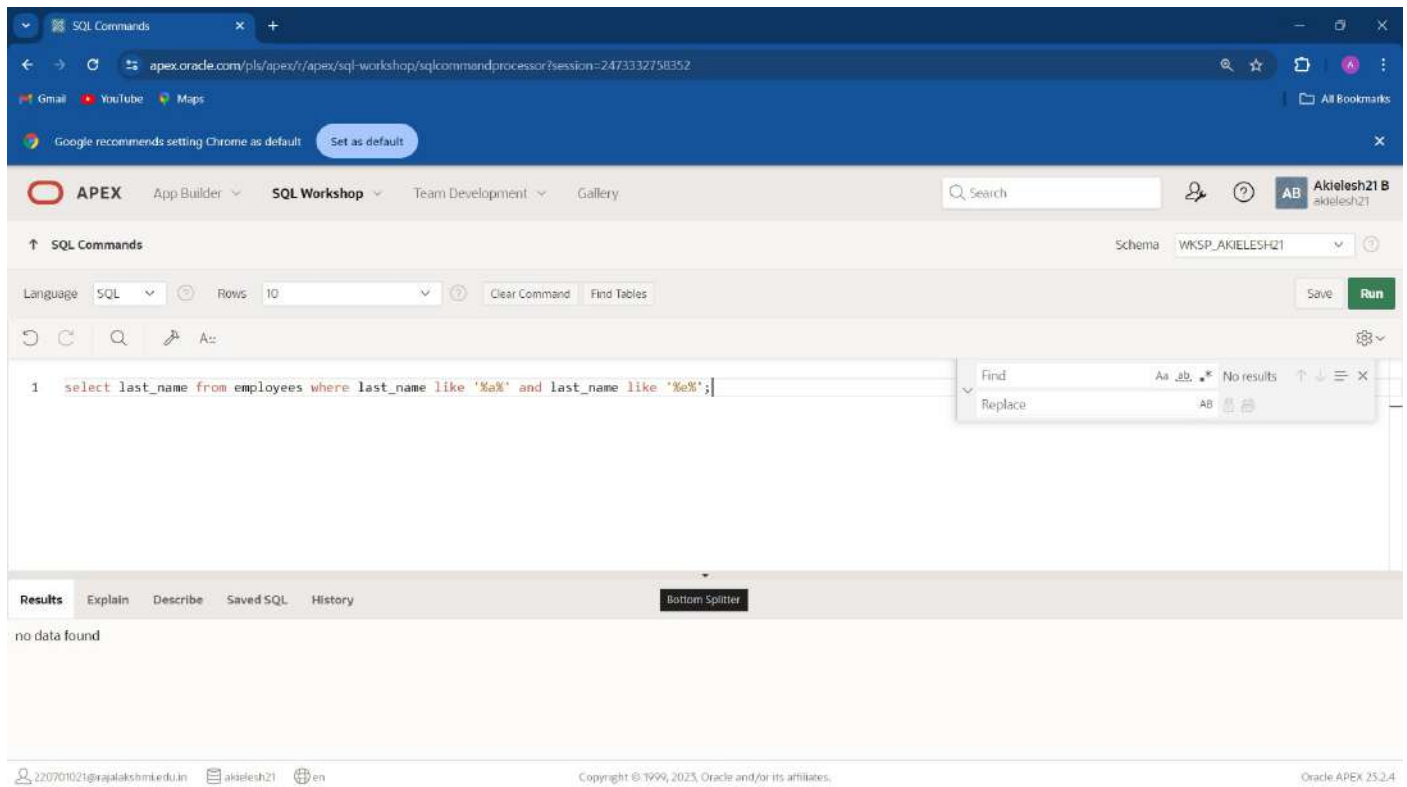
LAST_NAME
arli

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:**



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

**QUERY:**

select last\_name,job\_id,salary from employees where job\_id in ('sales representative','stock clerk') and salary not in(2500,3500,7000);

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The query entered is: `1 select last_name,job_code,salary from employees where job_title in ('sales_man','staff') AND salary not in(2500,3500,7000);`. The results are displayed in a table with columns LAST\_NAME, JOB\_CODE, and SALARY.

LAST_NAME	JOB_CODE	SALARY
elivari	345	1950
Hang	343	1705
jone	101	155

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\_pct from employees where commission\_pct=0.2;

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The query entered is: `1 select last_name,salary,commission from employees where commission= 0.2;`. The results are displayed in a table with columns LAST\_NAME, SALARY, and COMMISSION.

LAST_NAME	SALARY	COMMISSION
Hang	1705	.2

1 rows returned in 0.00 seconds

**RESULT:**

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

# SINGLE ROW FUNCTIONS

EX\_NO:6

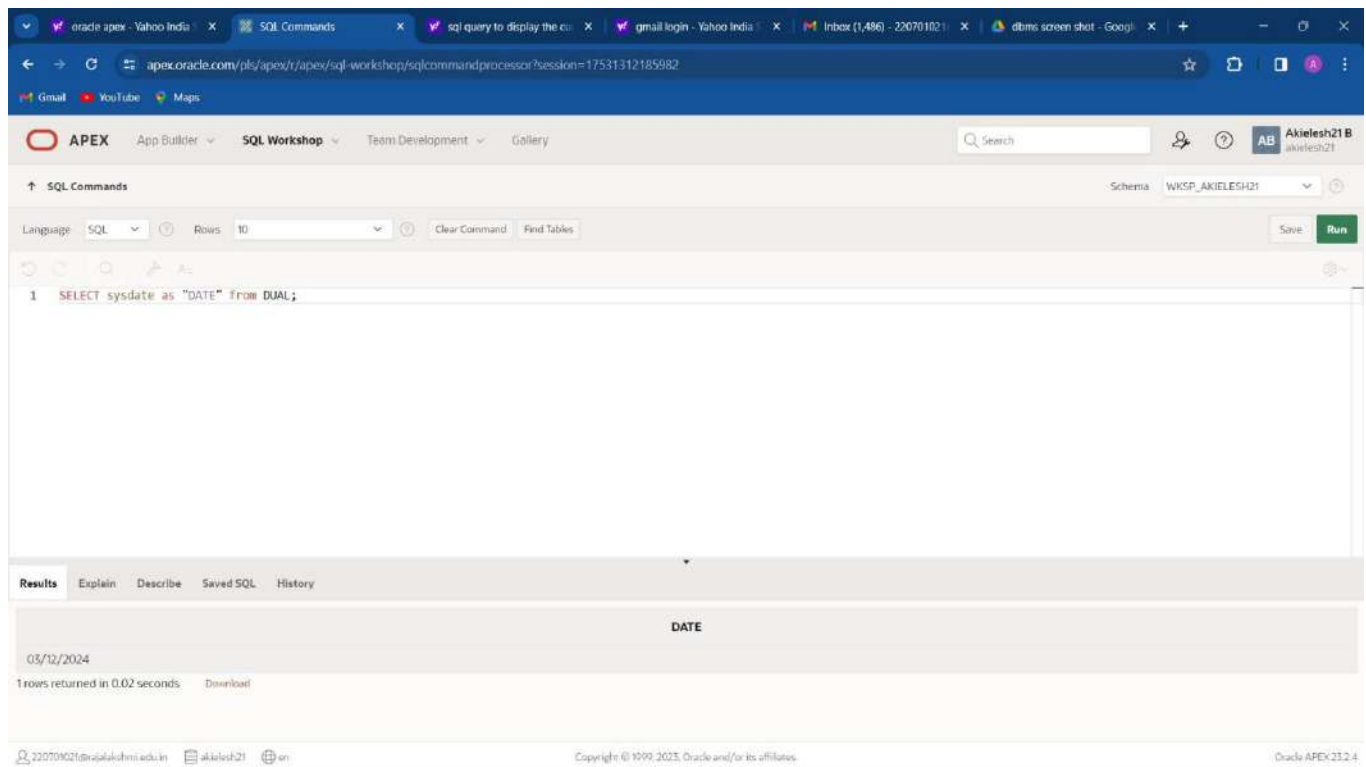
DATE:

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

**QUERY:**

select sysdate from dual;

**OUTPUT:**



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

```
1 SELECT sysdate as "DATE" from DUAL;
```

The results pane shows a single row with the date 03/12/2024 under the column header DATE.

DATE
03/12/2024

1 rows returned in 0.02 seconds

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

**QUERY:**

select employee\_id, last\_name, salary, salary+(15.5/100\*salary) "new\_salary" from employees;



## OUTPUT:

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

```
1 select employee_number,last_name,salary,salary+salary*0.155 as new_salary from employees;
```

The results are displayed in a table with the following columns: EMPLOYEE\_NUMBER, LAST\_NAME, SALARY, and NEW\_SALARY. The table contains four rows of data:

EMPLOYEE_NUMBER	LAST_NAME	SALARY	NEW_SALARY
4	tiwari	12500	14437.5
5	lang	11000	12705
176	jone	9000	1155
2	helli	6000	6930

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

## QUERY:

```
select employee_id,last_name,salary,salary+(15.5/100*salary) "new_salary",new_salary-salary as "Increase" from employees;
```

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is: `1 update employees set salary=(salary+salary*0.155)-salary;`. The results section shows: `3 row(s) updated.` and `0.04 seconds`. The schema is set to `WKSP_AKIELESH21`.

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

### QUERY:

`select initcap(last_name),length(last_name) as "Length_of_last_name" from employees where last_name like 'J%' or last_name like 'A%' or last_name like 'M%' order by last_name asc;`

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is: `1 select initcap(last_name),length(last_name) as "length_of_the_lastname" from emp where last_name like 'J%' or last_name like 'A%' or last_name like 'M%'`  
`2 order by last_name asc;`. The results section shows a table with two columns: `INITCAP(LAST_NAME)` and `length_of_the_lastname`. The data rows are: `Jhonson` with length `7`, and `Miller` with length `6`. The results section also shows: `2 rows returned in 0.04 seconds`. The schema is set to `WKSP_AKIELESH21`.

INITCAP(LAST_NAME)	length_of_the_lastname
Jhonson	7
Miller	6

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

**QUERY:**

Select last\_name from employees where last\_name like '%H';

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The browser address bar displays the URL: `apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=1775674049776`. The page title is "SQL Commands". The user is logged in as "Akielesh21 B". The schema is set to "WKSP\_AKIELESH21". The query entered in the command window is: `1 select last_name from employees where last_name like 'H%';`. The results are displayed in a table with the column header "LAST\_NAME". The first row contains the value "Hang". The status bar at the bottom indicates "1 rows returned in 0.00 seconds".

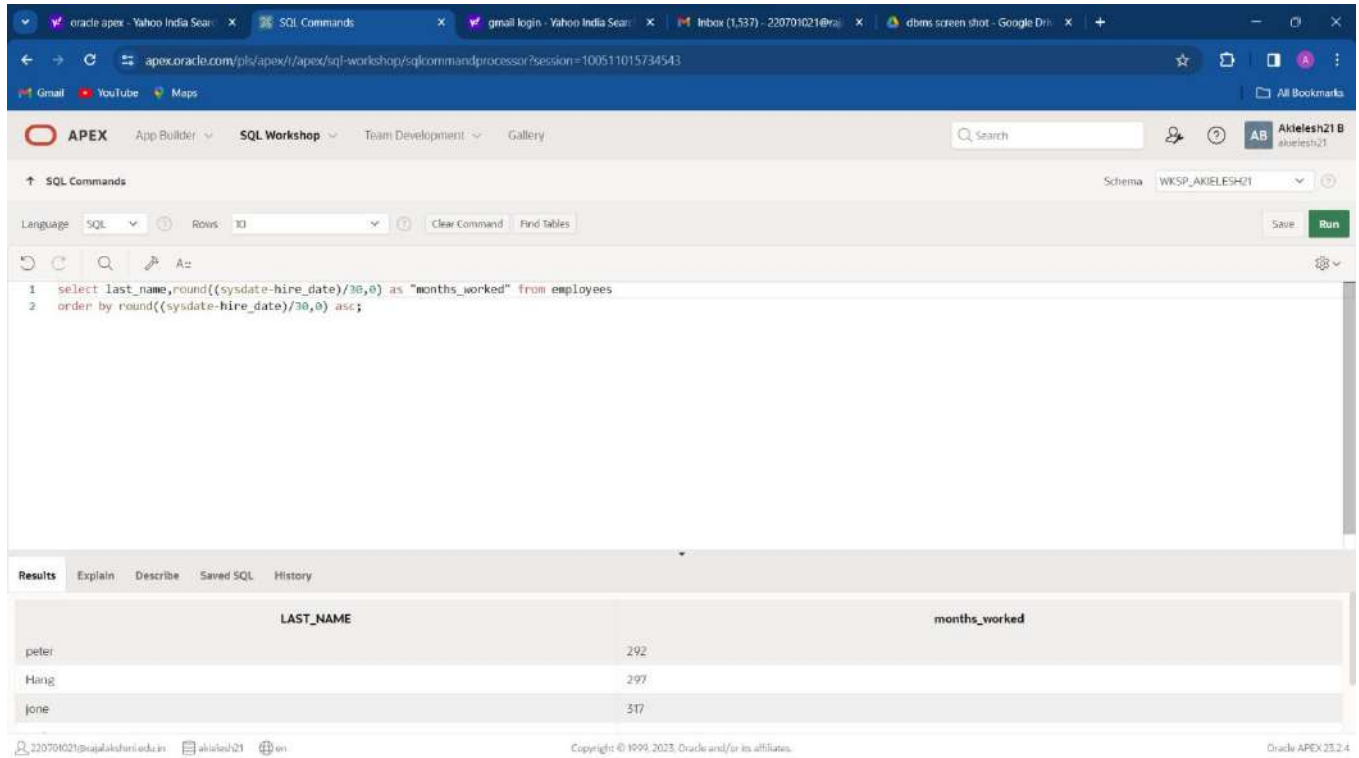
LAST_NAME
Hang

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 APEX SQL Workshop interface. The SQL command entered is:

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

The results are displayed in a table with the following data:

LAST_NAME	months_worked
peter	292
Haring	297
jone	317

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\_SALARIES" from employees;

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'SQL Commands' tab is active, displaying a query editor with the following SQL statement:

```
1 select last_name || 'learn' || salary || 'monthly_but_want' || salary*3 as "dream_salary" from employees;
```

Below the editor, the 'Results' tab is selected, showing the output of the query. The results are displayed in a table with one column, 'dream\_salary', and three rows of data:

dream_salary
elvislearn1938monthly_but_want5814
Hanglearn1705monthly_but_want5195
jonelearn155monthly_but_want465

The bottom of the interface shows the user 'akielesh21 B' and the Oracle APEX version '21.2.4'.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The query entered is: `select last_name, lpad(salary, 15, '$') as "salary" from employees;` The results table shows the last names of three employees and their salaries formatted with leading dollar signs to a total width of 15 characters.

LAST_NAME	salary
elivari	\$\$\$\$\$\$\$\$\$\$\$1938
Hang	\$\$\$\$\$\$\$\$\$\$\$1705
jone	\$\$\$\$\$\$\$\$\$\$\$155

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

### QUERY:

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

### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface with the following query: `select last_name, hire_date, to_char(next_day(add_months(hire_date, 6), 'monday'), 'fmday, "the"fmd"of"fmonth, yyyy') as review from employees;` The results table displays the last names, hire dates, and the calculated review dates for three employees.

LAST_NAME	HIRE_DATE	REVIEW
elivari	05/02/1992	monday, the 05 of november, 1992
Hang	11/03/1999	monday, the 05 of may, 2000
jone	03/04/1998	monday, the 07 of september, 1998

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:

SQL Commands

apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=104333882606023

Gmail YouTube Maps

Google recommends setting Chrome as default Set as default

APEX App Builder SQL Workshop Team Development Gallery

Search

AB Akielesh21 B akielesh21

SQL Commands

Schema WKSP\_AKIELESH21

Language SQL Rows 10 Clear Command Find Tables Save Run

1 SELECT last\_name,hire\_date,TO\_CHAR(hire\_date,'Day') as Day from employees order by TO\_CHAR(hire\_date,'Day');

Results Explain Describe Saved SQL History

LAST_NAME	HIRE_DATE	DAY
peter	04/07/2000	Friday
Zlotke	05/02/1992	Saturday
arli	04/06/1994	Wednesday

220701021@rajalakshmi.edu.in akielesh21 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

Faculty Signature		
----------------------	--	--

**RESULT:**

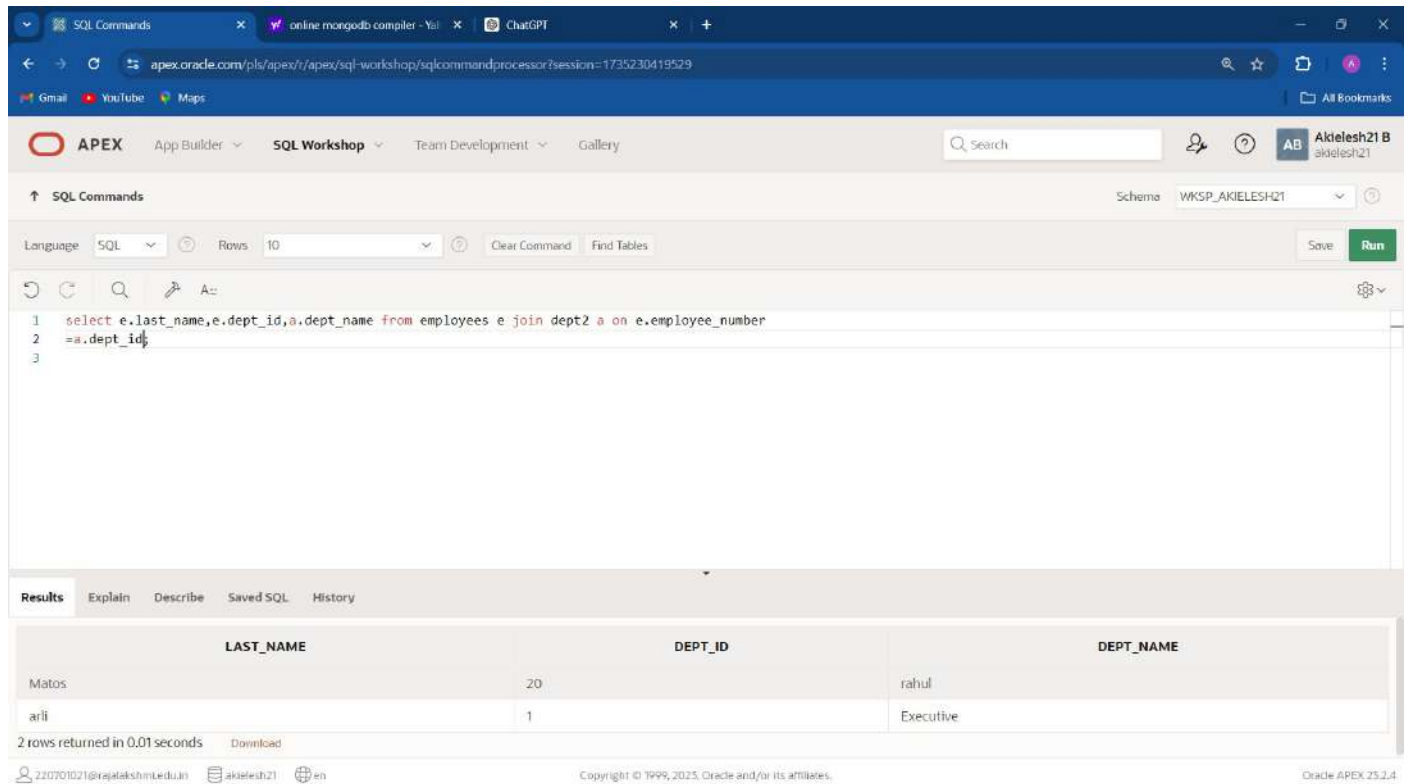


## DISPLAYING DATA FROM MULTIPLE TABLES

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 emp e join dept a on e.employee_id  
=a.employee_id;
```

**OUTPUT:**

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

```
1 select e.last_name,e.dept_id,a.dept_name from employees e join dept2 a on e.employee_number  
2 =a.dept_id  
3
```

The results are displayed in a table with the following columns: LAST\_NAME, DEPT\_ID, and DEPT\_NAME. The results show two rows:

LAST_NAME	DEPT_ID	DEPT_NAME
Matos	20	rahul
arli	1	Executive

2 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 emp e join dept a on e.employee_id=a.employee_id where  
e_department_id=80;
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The user is logged in as 'Akielesh21 B'. The 'SQL Commands' section is active, showing a query: `select e.job_title from employees e join dept2 a on e.employee_number=a.dept_id where e.dept_id=80;`. The 'Results' tab is selected, displaying a table with one row: `ST_CLERK`. The table has a column header `JOB_TITLE`. The footer shows the user's email, session ID, and copyright information.

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 SQL Workshop interface. The query entered is: `select last_name, dept_name from employees e join dept2 on commission is not null;`. The results table shows three rows for the employee 'Zlotke' with department names 'Executive', 'saran', and 'rahul'.

LAST_NAME	DEPT_NAME
Zlotke	Executive
Zlotke	saran
Zlotke	rahul

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 SQL Workshop interface. The query entered is: `select e.last_name,a.dept_name from employees e join dept2 a on employee_number=a.dept_id where e.last_name like 'a%';`. The results table shows one row for the employee 'arli' with department name 'Executive'.

LAST_NAME	DEPT_NAME
arli	Executive

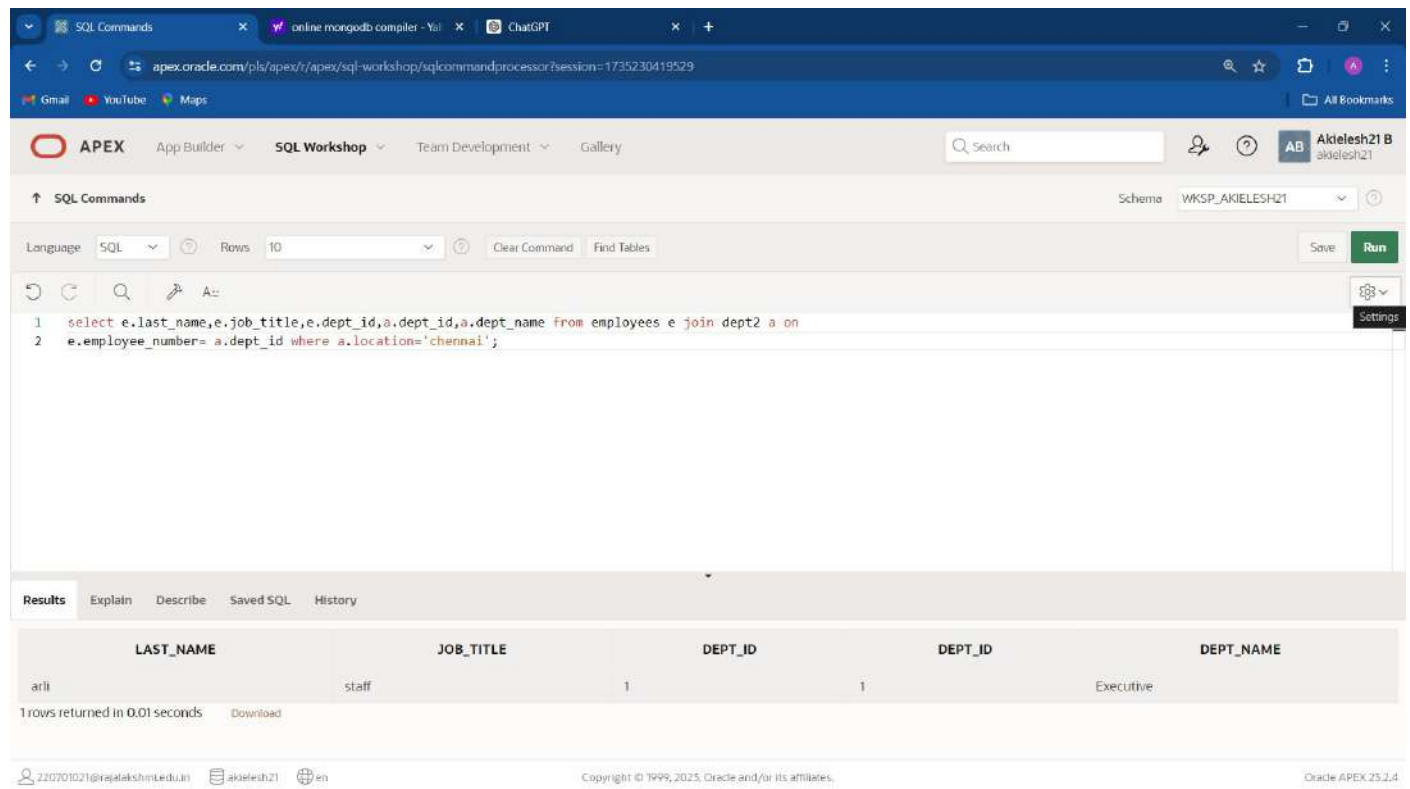
1 rows returned in 0.00 seconds

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

### QUERY:

```
select e.last_name,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 APEX SQL Workshop interface. The query entered is:

```
1 select e.last_name,e.job_title,e.dept_id,a.dept_id,a.dept_name from employees e join dept2 a on
2 e.employee_number= a.dept_id where a.location='chennai';
```

The results are displayed in a table with the following columns: LAST\_NAME, JOB\_TITLE, DEPT\_ID, DEPT\_ID, and DEPT\_NAME. The table contains one row of data:

LAST_NAME	JOB_TITLE	DEPT_ID	DEPT_ID	DEPT_NAME
ariti	staff	1	1	Executive

1 rows returned in 0.01 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 APEX SQL Workshop interface. The query executed is:

```
1 select last_name as "employee", employee_number as "emp#", manager_id as "mgr#" from employees;
```

The results are displayed in a table with the following columns: employee, emp#, and mgr#.

employee	emp#	mgr#
Zlotke	4	567
Matos	3	2017
Janu	176	2018

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 APEX SQL Workshop interface. The query executed is:

```
1 select e.last_name,e.employee_number,a.dept_name,e.salary from employees e
2 left outer join dept2 a on e.employee_number=a.dept_id order by e.employee_number;
```

The results are displayed in a table with the following columns: LAST\_NAME, EMPLOYEE\_NUMBER, DEPT\_NAME, and SALARY.

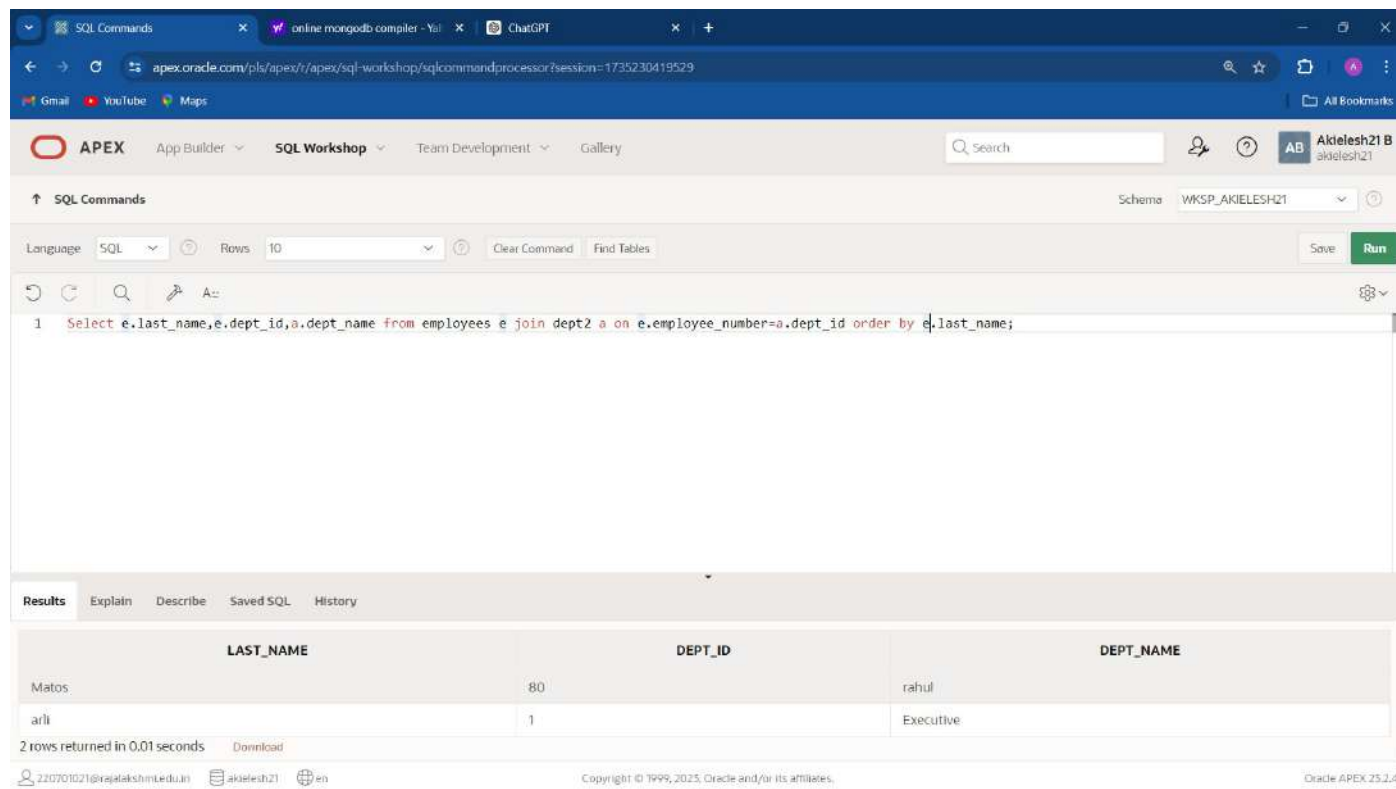
LAST_NAME	EMPLOYEE_NUMBER	DEPT_NAME	SALARY
aril	1	Executive	45000
Matos	3	rahul	78000
Zlotke	4	-	90000

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 APEX SQL Workshop interface. The query entered is: `1 Select e.last_name,e.dept_id,a.dept_name from employees e join dept2 a on e.employee_number=a.dept_id order by e.last_name;` The results are displayed in a table with three columns: LAST\_NAME, DEPT\_ID, and DEPT\_NAME. The results show two rows: Matos (DEPT\_ID: 80, DEPT\_NAME: rahul) and arli (DEPT\_ID: 1, DEPT\_NAME: Executive). The interface also shows the schema WKSP\_AKIELESH21 and the user AKIELESH21.

LAST_NAME	DEPT_ID	DEPT_NAME
Matos	80	rahul
arli	1	Executive

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 APEX SQL Workshop interface. The top navigation bar includes links for App Builder, SQL Workshop, Team Development, and Gallery. The main area is titled "SQL Commands" and shows a query in the "SQL" language. The query is:

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

The results are displayed in a table with two columns: LAST\_NAME and HIRE\_DATE. The results are as follows:

LAST_NAME	HIRE_DATE
Zlotke	05/02/1992
Matos	11/03/1999
Janu	05/04/1998

The bottom of the interface shows the user's name (Akelesh21 B), the schema (WKSP\_AKIELESH21), and the Oracle APEX version (23.3.4).

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",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:

SQL Commands

online mongodb compiler - Yaj

ChatGPT

apex.oracle.com/pls/apex/f/apex/sql-workshop/sqlcommandprocessor?session=1735230419529

GmailYouTubeMaps

APEX

App Builder

SQL Workshop

Team Development

Gallery

Search

AB Akielesh21 B akielesh21

SQL Commands

Schema WKSP\_AKIELESH21

Language SQL

Rows 10

Clear Command

Find Tables

Save

Run

1 e as "employee",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;

Results

Explain

Describe

Saved SQL

History

LAST_NAME	HIRE_DATE
Zlotke	05/02/1992
Matos	11/03/1999
Janu	03/04/1998

220701021@rajalakshmi.edu.in akielesh21 en

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

Oracle APEX 23.2.4

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

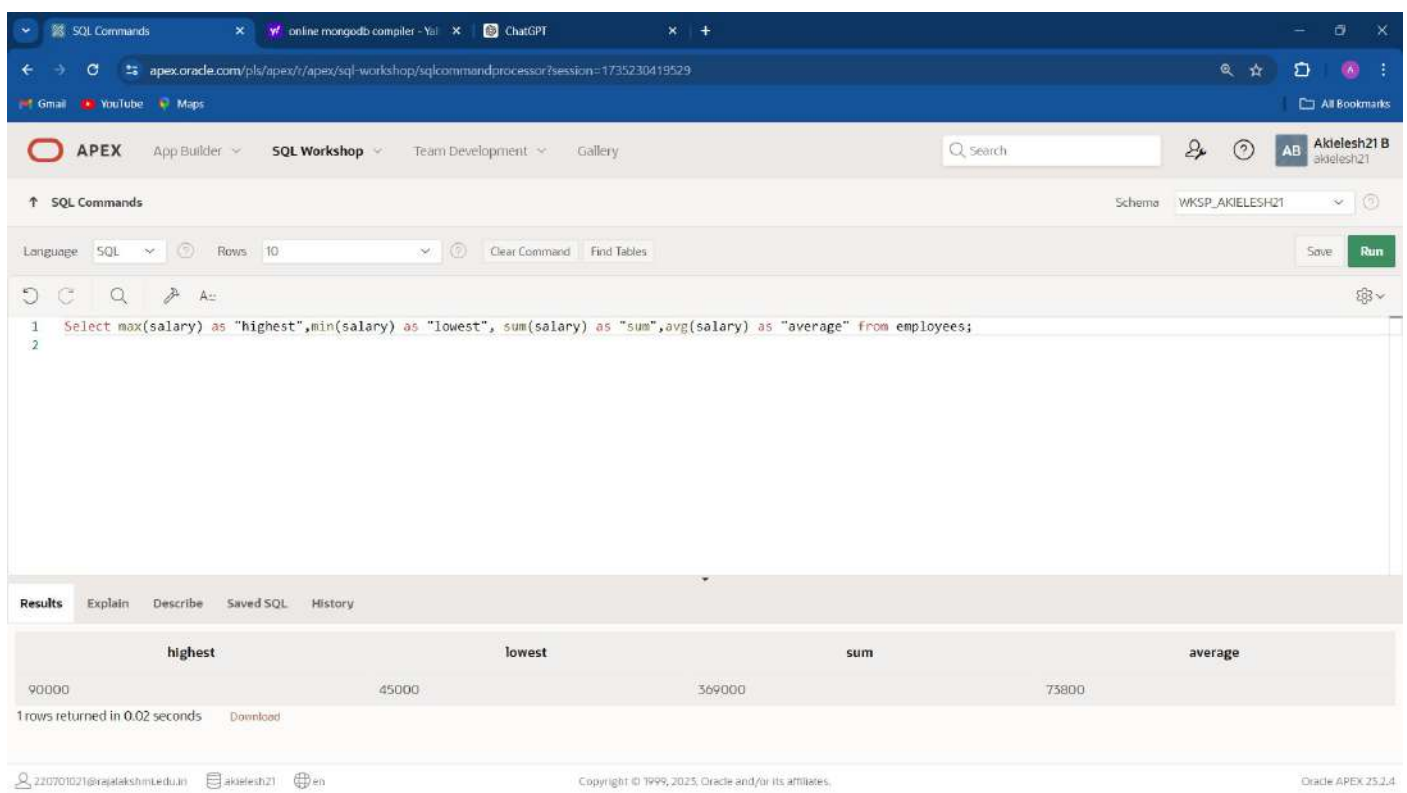


## AGGREGATING DATA USING GROUP FUNCTIONS

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:**

The screenshot shows the Oracle APEX SQL Workshop interface. The query entered is: `Select max(salary) as "highest",min(salary) as "lowest", sum(salary) as "sum",avg(salary) as "average" from employees;`. The results are displayed in a table with the following data:

highest	lowest	sum	average
90000	45000	369000	73800

1 rows returned in 0.02 seconds

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;

SQL Commands | online mongodb compiler - Yal | ChatGPT

apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=1735230419529

Gmail | YouTube | Maps | All Bookmarks

APEX | App Builder | SQL Workshop | Team Development | Gallery

Search

AB Akielesh21 B akielesh21

SQL Commands

Schema: WKSP\_AKIELESH21

Language: SQL | Rows: 10 | Clear Command | Find Tables | Save | Run

```

1 select max(salary) as "highest",min(salary) as "lowest", sum(salary) as "sum",avg(salary) as "average" from employees group by job_title;
2
3

```

Results | Explain | Describe | Saved SQL | History

highest	lowest	sum	average
45000	45000	45000	45000
78000	78000	78000	78000
89000	89000	89000	89000

220701021@rajalakshmi.edu.in | akielesh21 | en | Copyright © 1999, 2025, Oracle and/or its affiliates. | Oracle APEX 23.2.4

## OUTPUT:

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:

SQL Commands | online mongodb compiler - Yal | ChatGPT

apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=1735230419529

Gmail | YouTube | Maps | All Bookmarks

APEX | App Builder | SQL Workshop | Team Development | Gallery

Search

AB Akielesh21 B akielesh21

SQL Commands

Schema: WKSP\_AKIELESH21

Language: SQL | Rows: 10 | Clear Command | Find Tables | Save | Run

```

1 select count(last_name),job_title from employees group by job_title;
2
3

```

Results | Explain | Describe | Saved SQL | History

COUNT(LAST_NAME)	JOB_TITLE
1	staff
1	ST_CLERK
1	marketing

220701021@rajalakshmi.edu.in | akielesh21 | en | Copyright © 1999, 2025, Oracle and/or its affiliates. | Oracle APEX 23.2.4

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 APEX SQL Workshop interface. The SQL command entered is: `1 Select count(manager_id) as "number of managers" from employees;`. The results tab is active, displaying a single row with the value 5 under the column header "number of managers". The status bar indicates "1 rows returned in 0.00 seconds".

number of managers
5

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 APEX SQL Workshop interface. The SQL command entered is: `1 Select max(salary)-min(salary) as "difference" from employees;`. The results tab is active, displaying a single row with the value 45000 under the column header "difference". The status bar indicates "1 rows returned in 0.00 seconds".

difference
45000

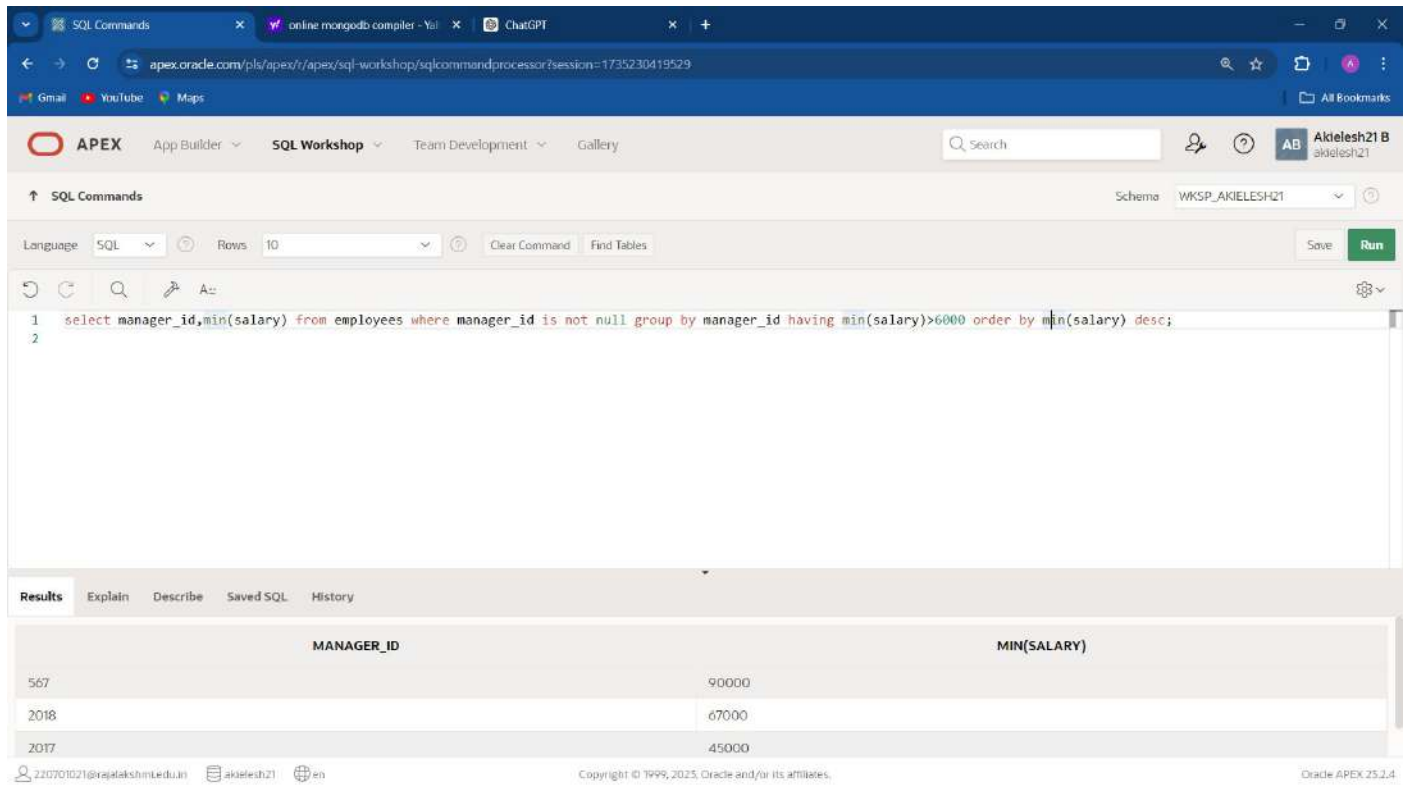
9. Create a report to display the manager number and the salary of the lowest-paid employee for that manager. Exclude anyone whose manager is not known. Exclude any

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:



The screenshot shows the Oracle APEX SQL Workshop interface. The query entered is: `select manager_id,min(salary) from employees where manager_id is not null group by manager_id having min(salary)>6000 order by min(salary) desc;` The results are displayed in a table with two columns: **MANAGER\_ID** and **MIN(SALARY)**. The results are sorted in descending order of minimum salary.

MANAGER_ID	MIN(SALARY)
567	90000
2018	67000
2017	45000

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:

The screenshot shows the APEX SQL Workshop interface. The SQL command entered is: `Select count(last_name) from employees where hire_date between '01-01-1975' and '31-12-1998';`. The results show a single row with the value 3 under the column heading COUNT(LAST\_NAME). The interface includes tabs for Results, Explain, Describe, Saved SQL, and History. The bottom of the screen shows the user's name, Akelesh21, and the Oracle APEX version, 23.2.4.

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 APEX SQL Workshop interface. The SQL command entered is: `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 employees group by job_id;`. The results are displayed in a table with columns: job, dept\_20, dept\_50, dept\_90, and total. The data rows are: (543, 78000, -, -, 168000), (909, -, -, -, 89000), and (101, -, -, -, 67000). The interface includes tabs for Results, Explain, Describe, Saved SQL, and History. The bottom of the screen shows the user's name, Akelesh21, and the Oracle APEX version, 23.2.4.

job	dept_20	dept_50	dept_90	total
543	78000	-	-	168000
909	-	-	-	89000
101	-	-	-	67000

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 APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'SQL Commands' tab is active, displaying a SQL query in the editor. The query is:

```
1 select e.dept_id as department_name,count(e.employee_number) as number_of_people,  
2 Round(avg(e.salary),2) as salary from employees e join dept2 d on e.dept_id=d.dept_id group by e.dept_id;  
3
```

Below the editor, the 'Results' tab is selected, showing a table with the following data:

DEPARTMENT_NAME	NUMBER_OF_PEOPLE	SALARY
1	1	45000

At the bottom of the results section, it states '1 rows returned in 0.03 seconds' and provides a 'Download' link. The footer of the interface includes the user's email '220701021@rajalakshmi.edu.in', the username 'akilesh21', the language 'en', the copyright notice 'Copyright © 1999, 2025 Oracle and/or its affiliates.', and the version 'Oracle APEX 25.1.4'.

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

# SUB QUERIES

**EX\_NO:9**

**DATE:**

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

**QUERY:**

```
select last_name,hire_date from employees where department_id=(select department_id from employees where last_name='Janu') and last_name not in('Janu');
```

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command entered is: `select last_name,hire_date from employees where dept_id=(select dept_id from employees where last_name='Janu');`. The results table shows one row with the last name 'Janu' and hire date '03/04/1998'. The interface includes a search bar, a schema dropdown set to 'WKSP\_AKIELESH21', and a 'Run' button. The bottom of the screen shows the user 'AKIELESH21' and the Oracle APEX version '23.2.4'.

LAST_NAME	HIRE_DATE
Janu	03/04/1998

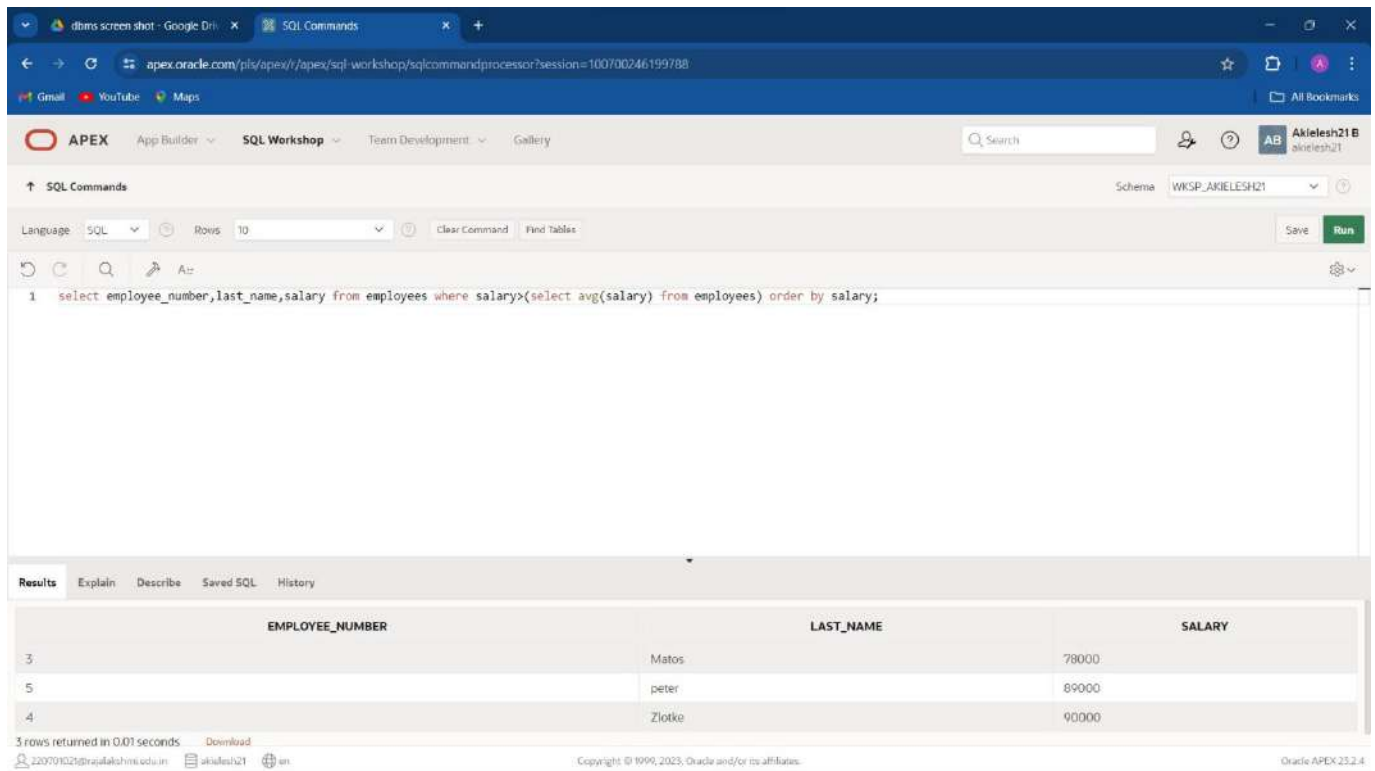
2.) Create a report that displays the employee number, last name, and salary of all employees who earn more than the average salary. Sort the results in order of ascending salary.

**QUERY:**



select employee\_id,last\_name,salary from employees where salary>(select avg(salary) from employees) order by salary;

## OUTPUT:



The screenshot shows the Oracle APEX SQL Workshop interface. The query entered is: `select employee_number,last_name,salary from employees where salary>(select avg(salary) from employees) order by salary;`. The results are displayed in a table with 3 rows.

EMPLOYEE_NUMBER	LAST_NAME	SALARY
3	Matos	78000
5	peter	89000
4	Zlotke	90000

3 rows returned in 0.01 seconds. Download

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

## QUERY:

select employee\_id,last\_name from employees where department\_id=(select department\_id from employees where last\_name like'%u%' );

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The user is logged in as 'AKIELESH21 B'. The 'SQL Commands' tab is active, showing a query editor with the following SQL command:

```
1 select employee_number,last_name from employees where dept_id=(select dept_id from employees where last_name like '%a% ');
```

Below the query editor, the 'Results' tab is selected, displaying the following table:

EMPLOYEE_NUMBER	LAST_NAME
170	Jorru

The results section also indicates '1 rows returned in 0.01 seconds' and provides a 'Download' link. The footer of the interface shows the user's session ID '2207010216', the user 'akiesh21', and the Oracle APEX version '23.2.4'.

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

**QUERY:**

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

**OUTPUT:**

dbms screen shot - Google Drive x SQL Commands x +

apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=100700246199788

Gmail YouTube Maps

APEX App Builder SQL Workshop Team Development Gallery

Search

AB Akielesh21 B akielesh21

SQL Commands Schema WKSP\_AKIELESH21

Language SQL Rows: 10 Clear Command Find Tables Save Run

```
1 select last_name,dept_id,job_code from employees where dept_id=(select dept_id from dept2 where location_id='1700');
```

Results Explain Describe Saved SQL History

LAST_NAME	DEPT_ID	JOB_CODE
arfi	1	102

1 rows returned in 0.00 seconds Download

220701021@vajalakshmi.edu.in akielesh21 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

**QUERY:**

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

**OUTPUT:**

dbms screen shot - Google Drive x SQL Commands x

apex.oracle.com/pls/apex/h/apex/sql-workshop/sqlcommandprocessor?session=100700246199788

Gmail YouTube Maps

APEX App Builder SQL Workshop Team Development Gallery

Search

AB Akielesh21 B akielesh21

SQL Commands Schema WKSP\_AKIELESH21

Language SQL Rows 10 Clear Command Find Tables Save Run

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

Results Explain Describe Saved SQL History

LAST_NAME	SALARY
Matos	78000
arll	45000

2 rows returned in 0.01 seconds Download

2207010216arajabulshini.edu.in akielesh21 Copyright © 1996-2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

**QUERY:**

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

**OUTPUT:**

dbms screen shot - Google Drive x SQL Commands x +

apex.oracle.com/pls/apex/f/apex/sql-workshop/sqlcommandprocessor?session=100700246199788

Gmail YouTube Maps

APEX App Builder SQL Workshop Team Development Gallery

Search

AB Akielesh21 B akielesh21

SQL Commands Schema WKSP\_AKIELESH21

Language SQL Rows 10 Clear Command Find Tables Save Run

```
1 select dept_id,last_name,job_code from employees where dept_id in (select dept_id from dept2 where dept_name='Executive');
```

Results Explain Describe Saved SQL History

DEPT_ID	LAST_NAME	JOB_CODE
1	arli	102

1 rows returned in 0.01 seconds Download

220701021@rajalakshmi.edu.in akielesh21 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.2.4

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

#### QUERY:

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

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The user is logged in as 'Akielesh21 B'. The 'SQL Commands' tab is active, showing a query: `select employee_number, last_name, salary from employees where salary > (select avg(salary) from employees where last_name like '%&');`. The results are displayed in a table with columns 'EMPLOYEE\_NUMBER', 'LAST\_NAME', and 'SALARY'. The results show three rows: (4, Zlotke, 90000), (3, Matos, 78000), and (5, Peter, 89000).

EMPLOYEE_NUMBER	LAST_NAME	SALARY
4	Zlotke	90000
3	Matos	78000
5	Peter	89000

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

## RESULT:

# USING THE SET OPERATORS

EX\_NO:10

DATE:

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

### QUERY:

```
select department_id from employees minus select department_id from employees where job_id='st_clerk';
```

### OUTPUT:

The screenshot displays the Oracle APEX SQL Workshop interface. The SQL command entered is: `select dept_id from employees minus select dept_id from employees where job_title='st_clerk';`. The query has been executed, and the results are shown in a table with the column header **DEPT\_ID**. The results list the department IDs 1, 30, and 80, which are the departments that do not contain the job ID ST\_CLERK.

DEPT_ID
1
30
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,state_province from location minus select country_id,state_province from location,departments where location.location_id=departments.location_id;
```

#### OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The user is logged in as 'Akielesh21 B'. The 'SQL Commands' tab is active, showing a query in the editor. The query is: `1 select first_name||' '||last_name as "Name",dept_id from employees union all select dept_name,dept_id from dept2;`. The 'Results' tab is selected, displaying a table with two columns: 'Name' and 'DEPT\_ID'. The table contains three rows of data.

Name	DEPT_ID
rahul Zlotke	30
jim Matos	80
david Janu	101

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

#### QUERY:



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

**OUTPUT:**

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

```
1 select job_code,dept_id from employees where dept_id=30 union
2 select job_code,dept_id from employees where dept_id=80 union
3 select job_code,dept_id from employees where dept_id=1;
```

The results are displayed in a table with the following data:

JOB_CODE	DEPT_ID
102	1
543	30
343	80

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The user is logged in as 'Akielesh21 B'. The 'SQL Commands' section is active, showing a query in the 'SQL' language with 10 rows. The query is:

```
1 select job_code,employee_number from employees intersect select e.job_code,e.employee_number from employees e where e.job_code=e.job_code;
```

The 'Results' tab is selected, displaying the following data:

JOB_CODE	EMPLOYEE_NUMBER
101	176
102	1
343	3

The footer shows the user's email '220701021@rajalakshmi.edu.in', the username 'akiesh21', and the Oracle APEX version '23.2.4'.

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 employees union all select dept_name,dept_id from departments;
```

#### OUTPUT:

SQL Commands

apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=106347847505918

Gmail

YouTube

Maps

Google recommends setting Chrome as default

Set as default

APEX

App Builder

SQL Workshop

Team Development

Gallery

Search

AB

Akielesh21 B

akielesh21

SQL Commands

Schema WKSP\_AKIELESH21

Language SQL

Rows 10

Clear Command

Find Tables

Save

Run

↶

↷

🔍

📌

A:

1

select first\_name||' '||last\_name as "Name",dept\_id from employees union all select dept\_name,dept\_id from dept2;

Results

Explain

Describe

Saved SQL

History

Name	DEPT_ID
rahul Zlotke	30
jim Matos	80
david Janu	101

220701021@rajalakshmi.edu.in

akielesh21

en

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

Oracle APEX 23.2.4

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

**RESULT:**

# CREATING VIEWS

EX\_NO:11

DATE:

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

## QUERY:

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

## OUTPUT:

The screenshot displays the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'SQL Commands' tab is active, showing a SQL command editor with the following code:

```
1 CREATE VIEW EMPLOYEE_VU AS
2 SELECT EMPLOYEE_NUMBER, last_name AS EMPLOYEE, DEPT_id
3 FROM EMPLOYEES;
4
```

Below the editor, the 'Results' tab is selected, displaying the message 'View created.' and the execution time '0.06 seconds'. The bottom status bar shows the user '2207010321@rajalakshmi.edu.in', the session 'akiesh21', and the Oracle APEX version '23.2.4'.

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

## QUERY:

```
select * from employees_vu;
```

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The query executed is `SELECT * FROM EMPLOYEE_VU;`. The results are displayed in a table with three columns: **EMPLOYEE\_NUMBER**, **EMPLOYEE**, and **DEPT\_ID**. The data returned is as follows:

EMPLOYEE_NUMBER	EMPLOYEE	DEPT_ID
4	etivari	30
3	hang	20
776	jone	101

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 APEX SQL Workshop interface. The query executed is `SELECT VIEW_NAME, TEXT FROM USER_VIEWS;`. The results are displayed in a table with two columns: **VIEW\_NAME** and **TEXT**. The data returned is as follows:

VIEW_NAME	TEXT
EMPLOYEE_VU	SELECT EMPLOYEE_NUMBER, last_name AS EMPLOYEE, DEPT_id FROM EMPLOYEES

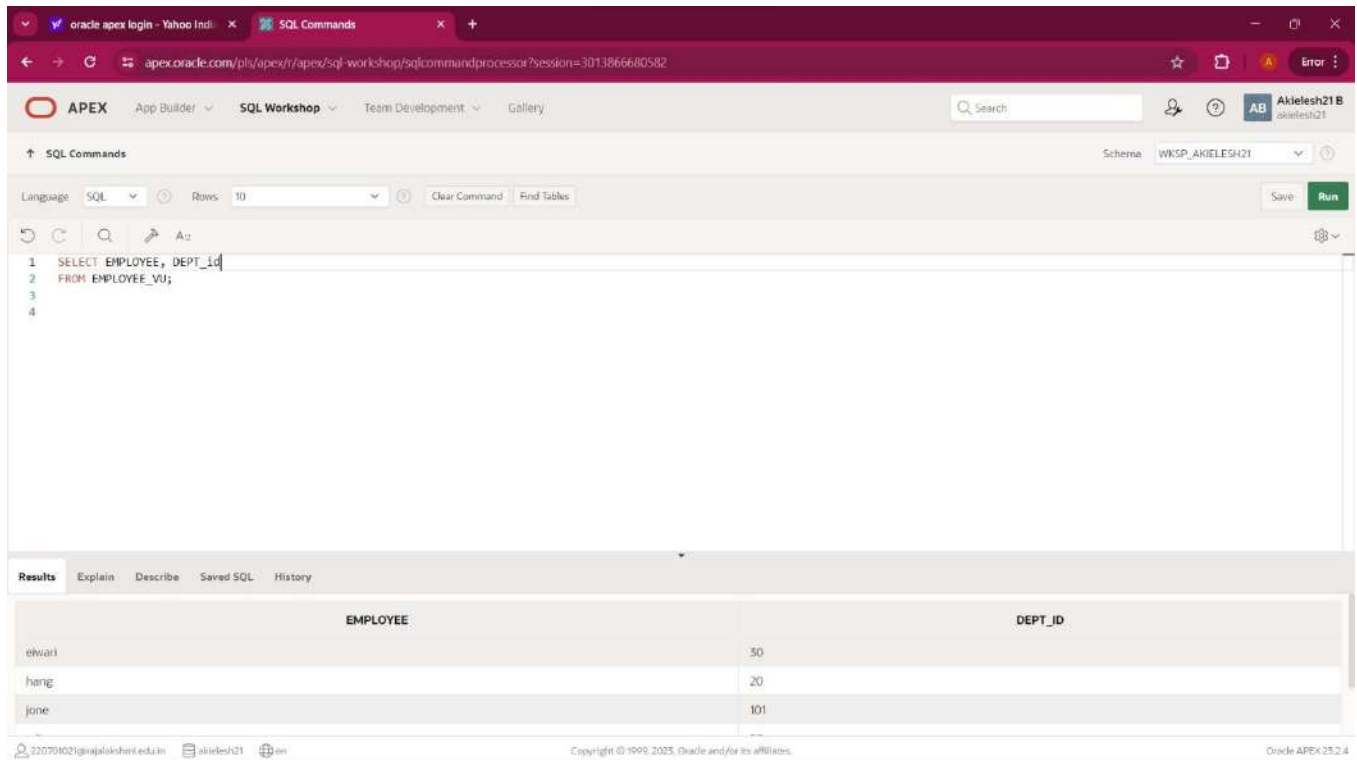
1 rows returned in 0.04 seconds

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'SQL Commands' tab is active, displaying a query editor with the following SQL statement:

```
1 SELECT EMPLOYEE, DEPT_ID
2 FROM EMPLOYEE_VU;
3
4
```

Below the query editor, the 'Results' tab is selected, showing a table with two columns: 'EMPLOYEE' and 'DEPT\_ID'. The table contains three rows of data:

EMPLOYEE	DEPT_ID
ehwari	50
hang	20
jone	101

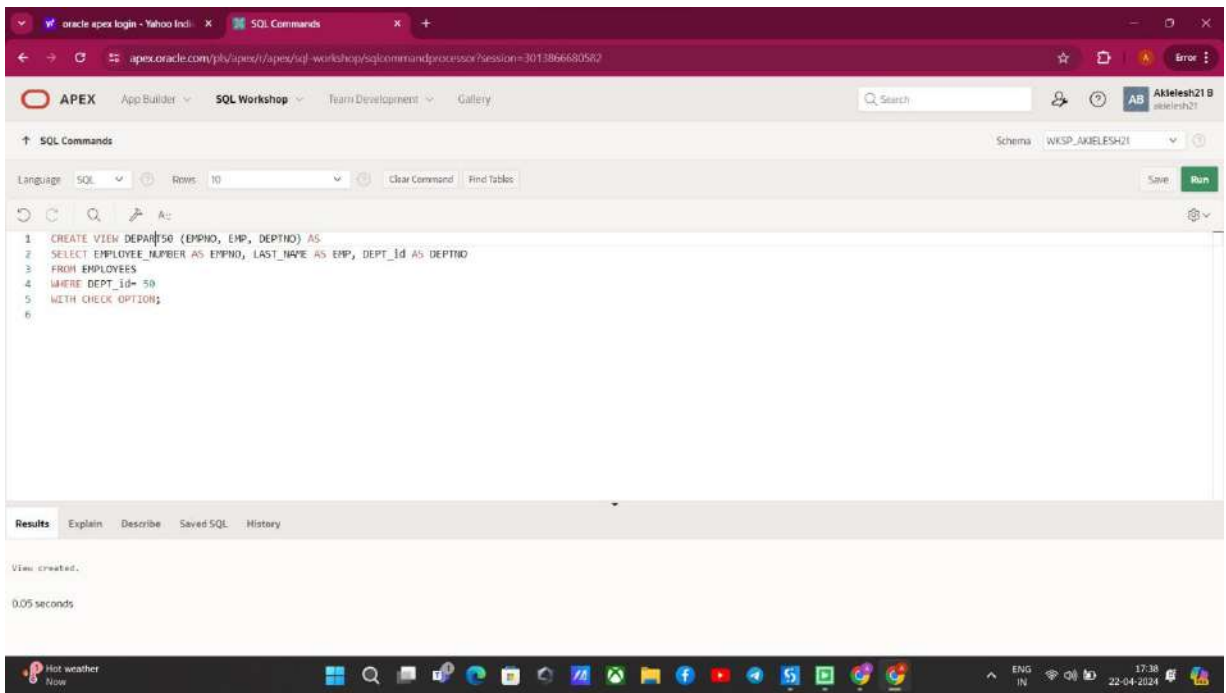
The bottom of the interface shows the footer with copyright information and the Oracle APEX version (25.2.4).

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

**QUERY:**

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

**OUTPUT:**

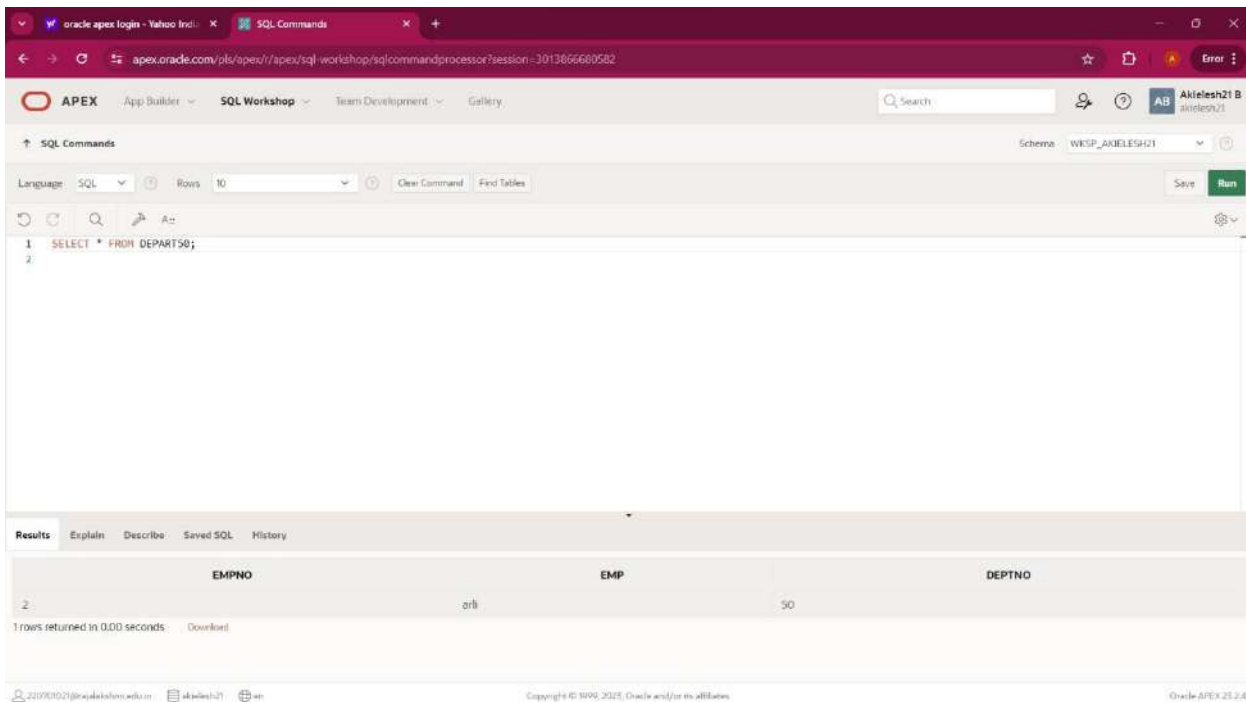


6.) Display the structure and contents of the DEPT50 view.

**QUERY:**

Describe dept50;

**OUTPUT:**



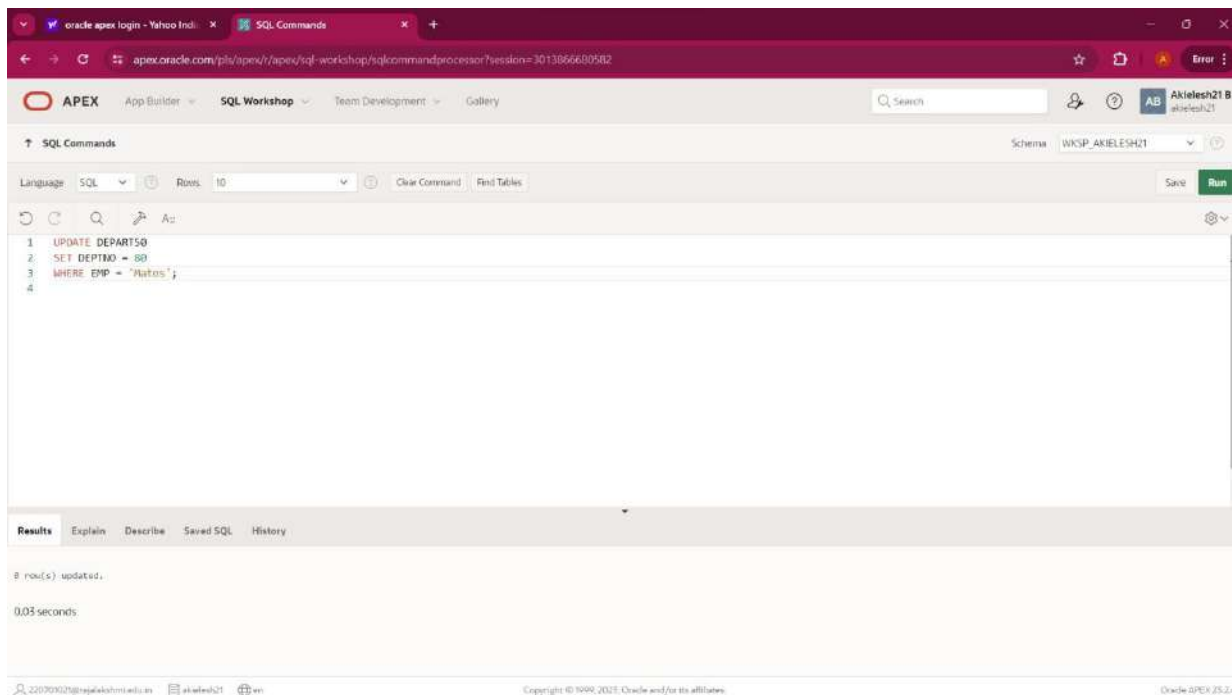
7.) Attempt to reassign Matos to department 80



### QUERY:

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

### OUTPUT:



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

### QUERY:

create or replace view salary\_vu as select e.last\_name "Employee",d.dept\_name Department, e.salary "Salary",j.grade\_level "Grades" from employees e,departments d,job\_grade j where e.department\_id=d.dept\_id and e.salary between j.lowest\_sal and j.highest\_sal;

### OUTPUT:

SQL Commands

apex.oracle.com/pls/apex/f/apex/sql-workshop/sqlcommandprocessor?session=106347847505918

Gmail

YouTube

Maps

Google recommends setting Chrome as default

Set as default

APEX

App Builder

SQL Workshop

Team Development

Gallery

Search

AB Akielesh21 B akielesh21

SQL Commands

Schema WKSP\_AKIELESH21

Language SQL

Rows 10

Clear Command

Find Tables

Save

Run

1

, e.salary "Salary", j.grade\_level "Grades" from employees e, departments d, job\_grade j where e.department\_id=d.dept\_id and e.salary between j.lowest\_sal and j.highest\_sal;

Results

Explain

Describe

Saved SQL

History

LAST_NAME	DEPT_ID	SALARY
Zlotke	30	90000
Matos	80	78000
Janu	101	67000

220701021@rajalakshmi.edu.in

akielesh21

en

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

Oracle APEX 23.2.4

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

**RESULT:**

# EXERCISE 12

## PRACTICE QUESTIONS

### Intro to Constraints; NOT NULL and UNIQUE Constraints

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Table Created.

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

DESCRIBE f\_global\_locations;

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

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

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

# PRIMARY KEY, FOREIGN KEY, and CHECK Constraints

1. What is the purpose of a

- PRIMARY KEY
- FOREIGN KEY
- CHECK CONSTRAINT

**a. PRIMARY KEY**

Uniquely identify each row in table.

**b. FOREIGN KEY**

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

**c. CHECK CONSTRAINT**

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

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

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

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

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

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

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

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

```
SELECT * FROM animals;
```

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

**COLUMN LEVEL STATEMENT:**

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

**TABLE LEVEL STATEMENT:**

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

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

a. ON DELETE CASCADE

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

b. ON DELETE SET NULL

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

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

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

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



# PRACTICE PROBLEM

## Managing Constraints

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

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

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

- ADD
- DROP
- ENABLE
- DISABLE

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

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

3. Create a FOREIGN KEY constraint in the copy\_d\_events table. Name the foreign key copy\_d\_events\_fk. This key references the copy\_d\_clients table client\_number column. What is the syntax you used to create the FOREIGN KEY constraint in the copy\_d\_events table?

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

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

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

a. The constraint name for the primary key in the copy\_d\_clients table is\_\_\_\_\_.

**COPY\_D\_CLT\_CLIENT\_NUMBER\_PK**

5. Drop the PRIMARY KEY constraint on the copy\_d\_clients table. Explain your results.

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

6. Add the following event to the copy\_d\_events table. Explain your results.

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

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

**RESULT:** ORA-02291: integrity constraint (HKUMAR.COPY\_D\_EVE\_CLIENT\_NUMBER\_FK) violated - parent key not found

7. Create an ALTER TABLE query to disable the primary key in the copy\_d\_clients table. Then add the values from #6 to the copy\_d\_events table. Explain your results.

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

8. Repeat question 6: Insert the new values in the copy\_d\_events table. Explain your results.

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

**1 row(s) inserted.**

9. Enable the primary-key constraint in the copy\_d\_clients table. Explain your results.

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

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

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

**1 row(s) deleted.**

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

**Table altered.**

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

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

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

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

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

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

# EXERCISE 13

## Creating Views

- 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.**
- 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';
```

- 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](#)

- 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';
```

- 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);
```

- It is company policy that only upper-level management be allowed access to individual employee salaries. The department managers, however, need to know the minimum, maximum, and average salaries, grouped by

department. Use the Oracle database to prepare a view that displays the needed information for department managers.

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

# DML Operations and Views

Use the DESCRIBE statement to verify that you have tables named copy\_d\_songs, copy\_d\_events, copy\_d\_cds, and copy\_d\_clients in your schema. If you don't, write a query to create a copy of each.

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

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

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

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

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

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

```
SELECT * FROM view_copy_d_songs;
```

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

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

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

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

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

```
SELECT * FROM read_copy_d_cds;
```

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

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

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

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

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

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

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

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

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

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

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

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

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

**DELETE, INSERT, MODIFY restricted if it contains:**

**Group functions  
GROUP BY CLAUSE  
DISTINCT  
pseudocolumn ROWNUM Keyword**

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

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

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

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

# Managing Views

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

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

```
SELECT * FROM view_copy_d_songs;
```

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

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

**ORA-00942: table or view does not exist**

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

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

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

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

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

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



# Indexes and Synonyms

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

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

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

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

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

3. When will an index be created automatically?

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

4. Create a nonunique index (foreign key) for the DJs on Demand column (cd\_number) in the D\_TRACK\_LISTINGS table. Use the Oracle Application Express SQL Workshop Data Browser to confirm that the index was created.

**CREATE INDEX d\_tlg\_cd\_number\_fk\_i  
on d\_track\_listings (cd\_number);**

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

**SELECT ucm.index\_name, ucm.column\_name, ucm.column\_position, uix.uniqueness  
FROM user\_indexes uix INNER JOIN user\_ind\_columns ucm ON uix.index\_name = ucm.index\_name  
WHERE ucm.table\_name = 'D\_SONGS';**

6. Use a SELECT statement to display the index\_name, table\_name, and uniqueness from the data dictionary USER\_INDEXES for the DJs on Demand D\_EVENTS table.

**SELECT index\_name, table\_name, uniqueness FROM user\_indexes where table\_name = 'D\_EVENTS';**

7. Write a query to create a synonym called dj\_tracks for the DJs on Demand d\_track\_listings table.

**CREATE SYNONYM dj\_tracks FOR d\_track\_listings;**

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

**CREATE INDEX d\_ptr\_last\_name\_idx  
ON d\_partners(LOWER(last\_name));**

9. Create a synonym for the D\_TRACK\_LISTINGS table. Confirm that it has been created by querying the data dictionary.

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

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

10. Drop the synonym that you created in question

```
DROP SYNONYM dj_tracks2;
```

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

**RESULT:**

# OTHER DATABASE OBJECTS

EX\_NO:14

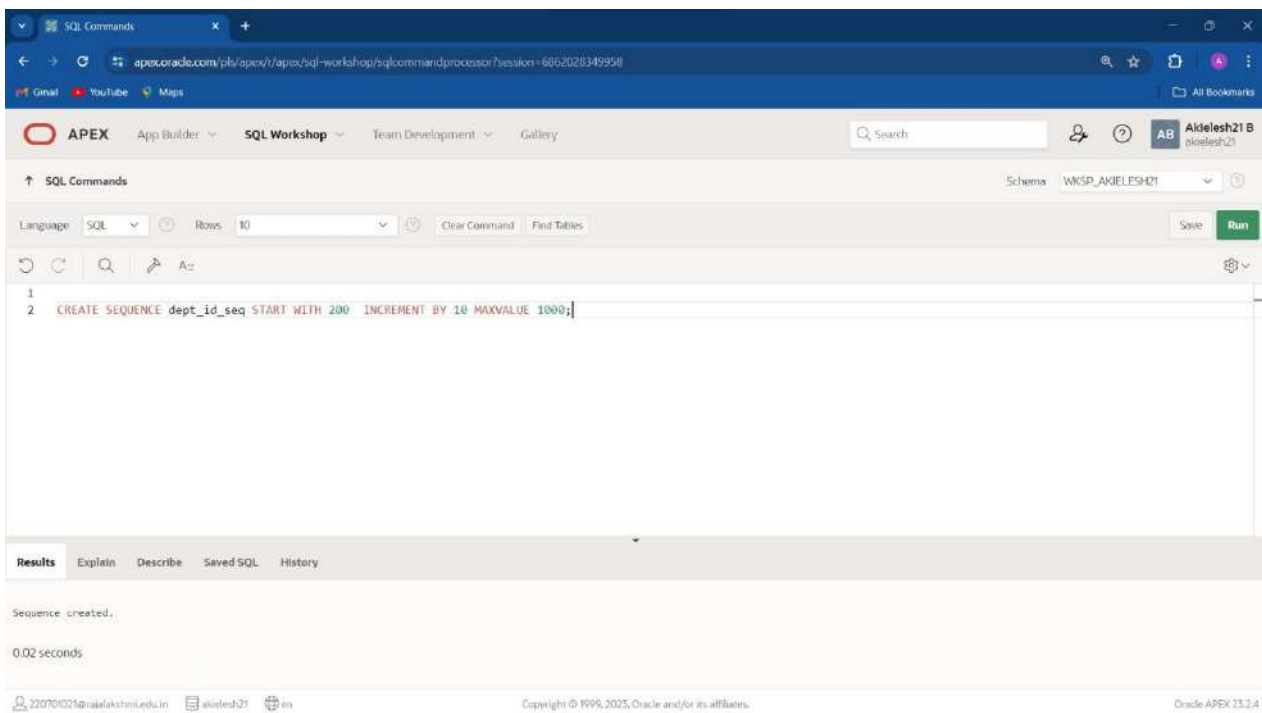
DATE:

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

**QUERY:**

CREATE SEQUENCE dept\_id\_seq START WITH 200 INCREMENT BY 10 MAXVALUE 1000;

**OUTPUT:**

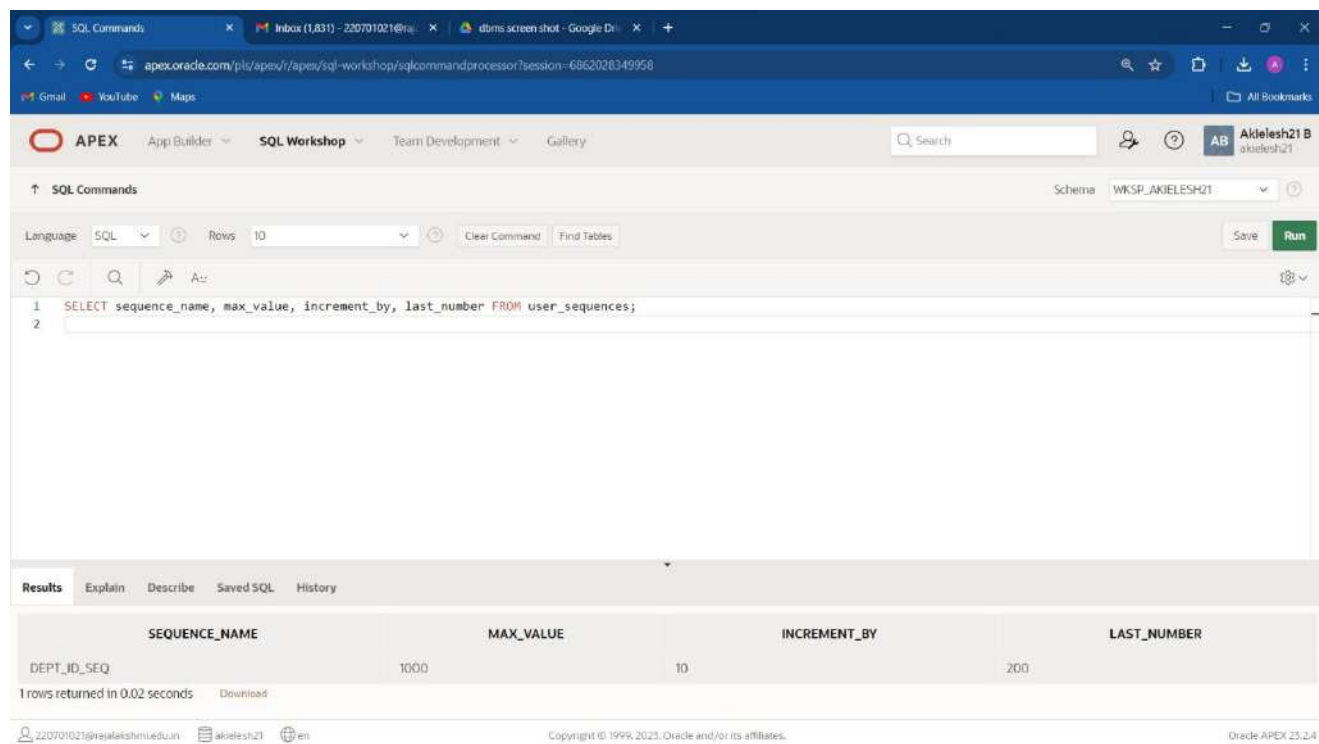


2.) Write a query in a script to display the following information about your sequences: sequence name, maximum value, increment size, and last number

**QUERY:**

SELECT sequence\_name, max\_value, increment\_by, last\_number FROM user\_sequences;

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The user is logged in as 'Akilesh21 B'. The 'SQL Commands' tab is active, showing a query editor with the following SQL statement:

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

The 'Run' button is highlighted in green. Below the query editor, the 'Results' tab is selected, displaying a table with the following data:

SEQUENCE_NAME	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
DEPT_ID_SEQ	1000	10	200

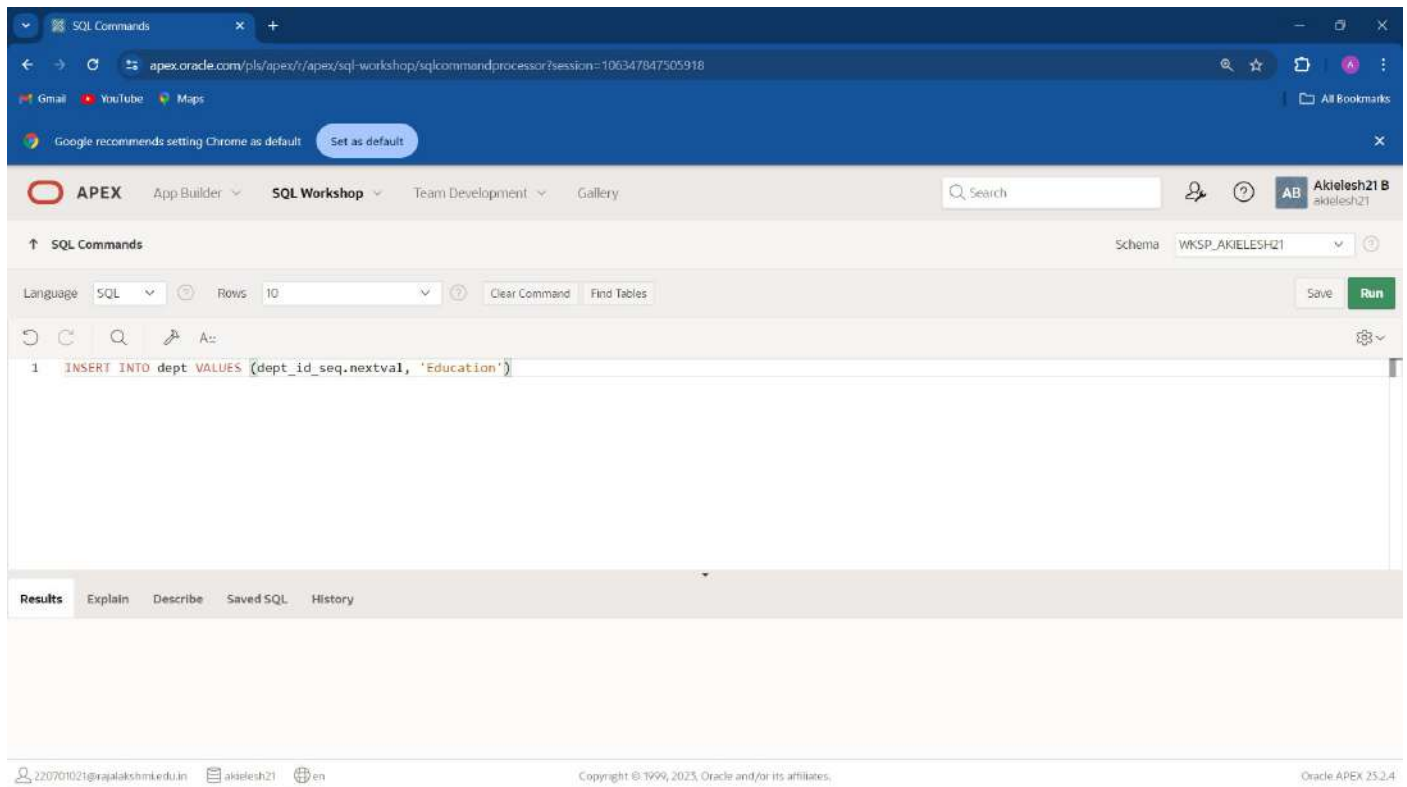
Below the table, it states '1 rows returned in 0.02 seconds' and provides a 'Download' link. The footer of the interface shows the user's email 'z20701021@rajaiaishim.edu.in', the username 'akilesh21', and the Oracle APEX version '23.2.4'.

3.) Write a script to insert two rows into the DEPT table. Name your script lab12\_3.sql. Be sure to use the sequence that you created for the ID column. Add two departments named Education and Administration. Confirm your additions. Run the commands in your script.

**QUERY:**

INSERT INTO dept VALUES (dept\_id\_seq.nextval, 'Education');

**OUTPUT:**

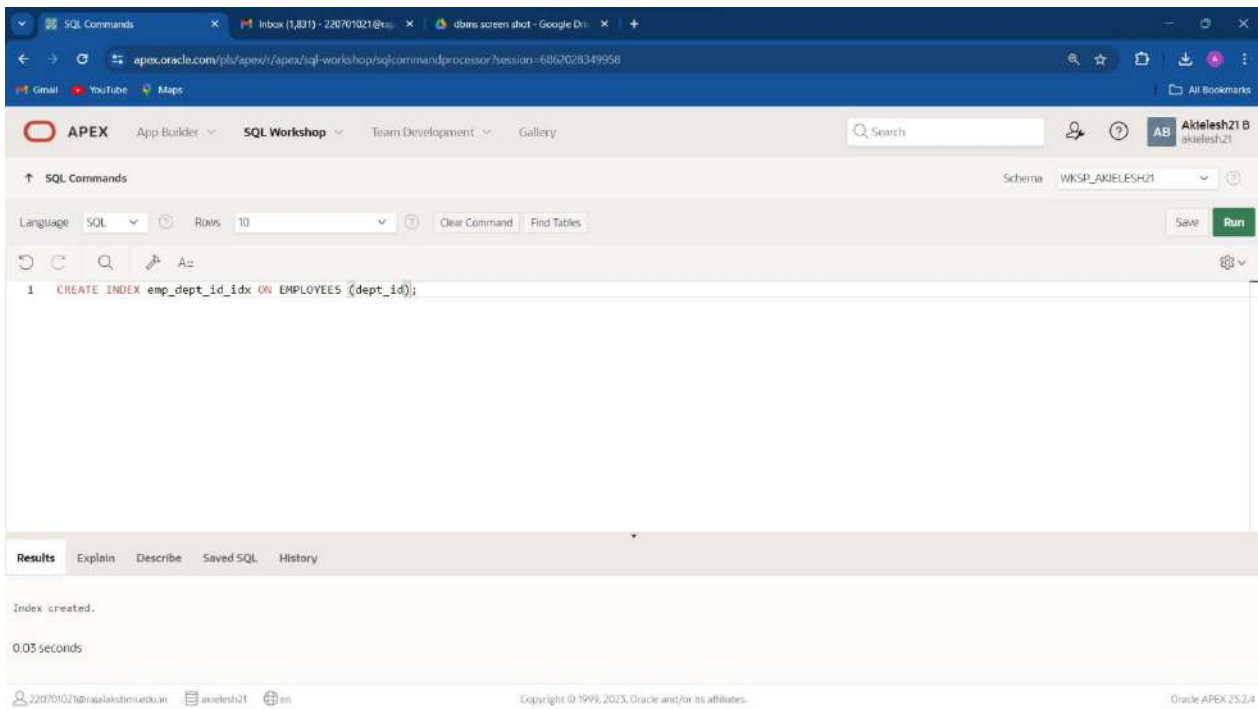


4.) Create a nonunique index on the foreign key column (DEPT\_ID) in the EMP table.

**QUERY:**

CREATE INDEX emp\_dept\_id\_idx ON EMPLOYEES (department\_id);

**OUTPUT:**



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:**

SQL Commands

Inbox (1.831) - 2207010216@ra...

dbms screen shot - Google Dr...

apex.oracle.com/pls/apex/f/apex-sql-workshop/sqlcommandprocessor?session=6862028349958

Gmail

YouTube

Maps

APEX

App Builder

SQL Workshop

Team Development

Gallery

Search

akiesh21 B

akiesh21

SQL Commands

Schema: WKSP\_AKIEESH21

Language: SQL

Rows: 10

Clear Command

Find Tables

Save

Run

1

SELECT index\_name,table\_name,uniqueness FROM user\_indexes WHERE table\_name='EMPLOYEES';

Results

Explain

Describe

Saved SQL

History

INDEX_NAME	TABLE_NAME	UNIQUENESS
EMP_DEPT_ID_IDX	EMPLOYEES	NONUNIQUE

1 rows returned in 0.07 seconds

Download

220701021@rajalakshmi.edu.in

akiesh21

201

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

Oracle APEX 23.2.4

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

**RESULT:**



# CONTROLLING USER ACCESS

**EX\_NO:15**

**DATE:**

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

The CREATE SESSION system privilege

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

The CREATE TABLE privilege

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

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

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

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

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

The ALTER USER statement

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

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

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

7. Query all the rows in your DEPARTMENTS table.

SELECT \* FROM departments;

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

Team 1 executes this INSERT statement. INSERT INTO departments(department\_id, department\_name) VALUES (500, 'Education'); COMMIT;

Team 2 executes this INSERT statement. INSERT INTO departments(department\_id, department\_name) VALUES (510, 'Administration'); COMMIT;

9. Query the USER\_TABLES data dictionary to see information about the tables that you own.

SELECT table\_name FROM user\_tables;

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

Team 1 revokes the privilege.

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

Team 2 revokes the privilege.

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

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

Team 1 executes this INSERT statement.

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

Team 2 executes this INSERT statement.

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

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

**RESULT:**

# PL/SQL

## CONTROL STRUCTURES

**EX\_NO:**

**DATE:**

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

**QUERY:**

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

**OUTPUT:**

The screenshot displays the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'SQL Commands' tab is active, showing a PL/SQL block with the following code:

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

The 'Results' tab is selected, showing the output: 'Incentive = 9360'. Below the output, it states 'Statement processed.' and '0.00 seconds'. The bottom of the interface shows the user 'akielesh21' and the Oracle APEX version '23.2.4'.

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

**QUERY:**

```
DECLARE
WELCOME varchar2(10) := 'welcome';
```

BEGIN

DBMS\_Output.Put\_Line("Welcome");

END;

/

**OUTPUT:**

The screenshot shows the APEX SQL Workshop interface. At the top, there's a navigation bar with 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'Akilan 22' are also visible. Below this, the 'SQL Commands' section is active, showing a PL/SQL block with line numbers 1 through 13. The block declares two variables, 'myvariable' and 'myvariable2', and then prints their values using 'DBMS\_OUTPUT.PUT\_LINE'. It also includes an exception handler. The 'Run' button is highlighted in green. Below the code editor, the 'Results' tab is selected, displaying the output: 'Quoted identifier: Hello, World!' and 'Non-quoted identifier: PL/SQL Programming'. A status message at the bottom indicates 'Statement processed.'

```
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
```

Results Explain Describe Saved SQL History

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 APEX SQL Workshop interface. The SQL Commands tab is active, displaying a PL/SQL block. The block declares a variable salary\_of\_emp, creates a procedure approx\_salary, and then calls it. The output shows the salary before and after the procedure call.

```

1 DECLARE
2   salary_of_emp NUMBER(8,2);
3   PROCEDURE approx_salary (
4     emp NUMBER,
5     emp_sal IN OUT NUMBER,
6     address NUMBER
7   ) IS
8   BEGIN
9     emp_sal := emp_sal + address;
10  END;
11
12 BEGIN
13   SELECT salary INTO salary_of_emp
14   FROM emp
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

```

Results: Explain Describe Saved SQL History

Before invoking procedure, salary\_of\_emp: 5000  
After invoking procedure, salary\_of\_emp: 6000  
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;

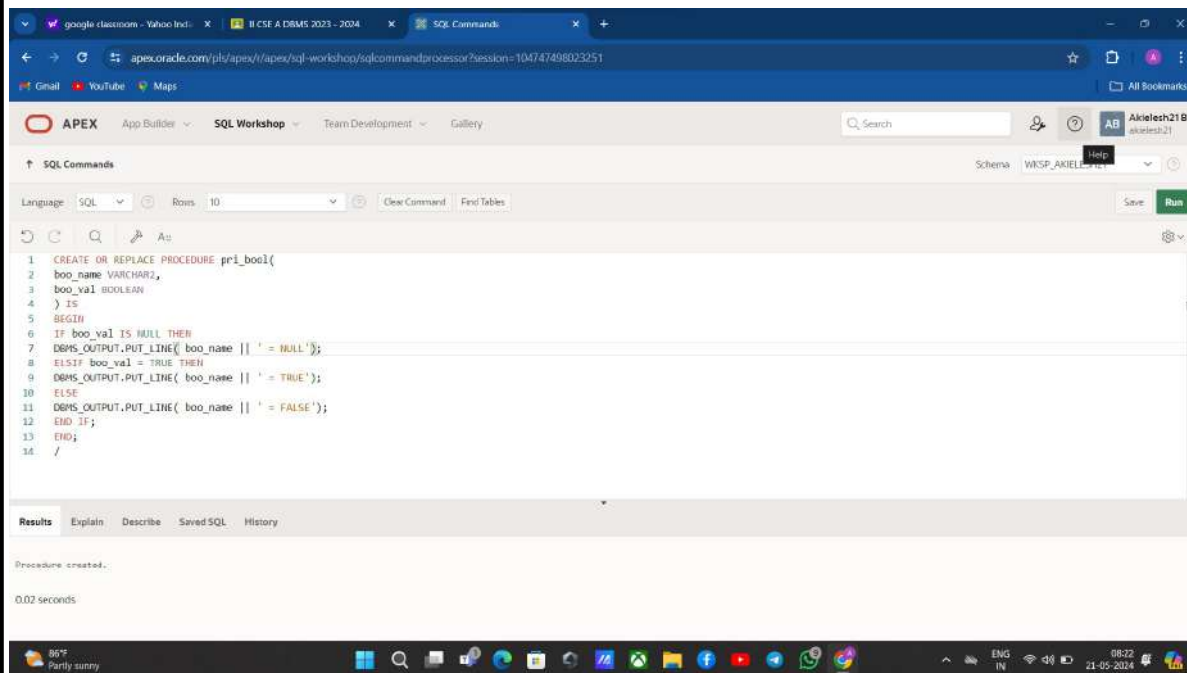
```

END IF;

END;

/

**OUTPUT:**



**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');

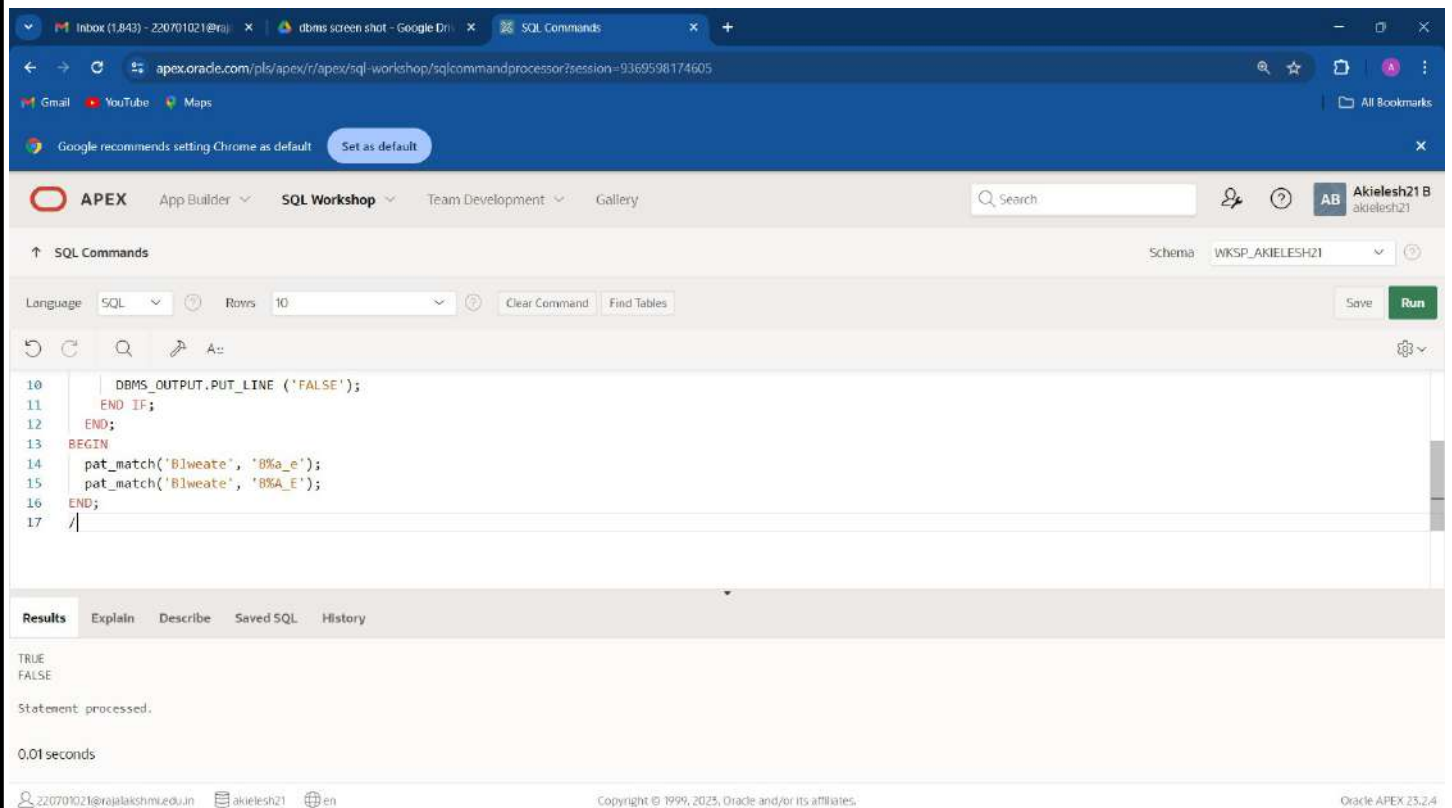
ELSE

```

DBMS_OUTPUT.PUT_LINE ('FALSE');
END IF;
END;
BEGIN
pat_match('Blweate', 'B%a_e');
pat_match('Blweate', 'B%A_E');
END;
/

```

## OUTPUT:



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

## QUERY:

```

DECLARE
num_small NUMBER := 8;
num_large NUMBER := 5;
num_temp NUMBER;
BEGIN
IF num_small > num_large THEN
num_temp := num_small;
num_small := num_large;
num_large := num_temp;
END IF;

```

```

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

```

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'SQL Commands' tab is active, displaying a PL/SQL procedure. The procedure code is as follows:

```

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  /

```

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

```

num_small = 5
num_large = 8
Statement processed.
0.00 seconds

```

The bottom of the interface shows the user 'akiesh21' and the version 'Oracle APEX 23.2.4'.

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

## QUERY:

```

DECLARE
PROCEDURE test1 (
  sal_achieve NUMBER,
  target_qty NUMBER,
  emp_id NUMBER
)
IS
  incentive NUMBER := 0;

```



```

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

```

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'SQL Commands' tab is active, displaying a PL/SQL procedure named 'test1'. The procedure logic is as follows:

```

BEGIN
IF sal_achieve > (target_qty + 200) THEN
    incentive := (sal_achieve - target_qty)/4;
    UPDATE employees
    SET salary = salary + incentive
    WHERE employee_number = emp_id;
    updated := 'Yes';
END IF;
DBMS_OUTPUT.PUT_LINE (
    'Table updated? ' || updated || ', ' ||
    'incentive = ' || incentive || ' '
);
END test1;

```

The 'Results' tab at the bottom shows the output of the procedure execution:

```

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

```

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

**QUERY:**

**DECLARE**

**PROCEDURE test1 (sal\_achieve NUMBER)**

```

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

```

## OUTPUT:

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

```

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  /

```

The Results window shows the following output:

```

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 APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user information 'A2 Aktion 22 bleach22' are on the right. The 'SQL Commands' pane shows the PL/SQL code being executed. The 'Results' pane at the bottom displays 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:

The screenshot displays the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The user is logged in as 'Akielesh21 B'. The 'SQL Commands' tab is active, showing the following SQL code:

```
7 INTO tot_emp  
8 FROM employees e  
9 join dept2 d  
10 ON e.dept_id = d.dept_id  
11 WHERE e.dept_id = get_dep_id;  
12 dbms_output.Put_line ('The employees are in the department '||get_dep_id||' is: '  
13 ||To_char(tot_emp));  
14 IF tot_emp >= 45 THEN  
15 dbms_output.Put_line ('There are no vacancies in the department '||get_dep_id);  
16 ELSE  
17 dbms_output.Put_line ('There are '||to_char(45-tot_emp)||' vacancies in department '||  
18 get_dep_id );  
19 END IF;  
20 END;  
21 /
```

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

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

The bottom of the interface shows the user's email '22070102@apalekhm.edu.in', the username 'akielesh21', and the Oracle APEX version '25.2.4'.

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

**QUERY:**

DECLARE

v\_employee\_id employees.employee\_id%TYPE;

v\_full\_name employees.first\_name%TYPE;

v\_job\_id employees.job\_id%TYPE;

v\_hire\_date employees.hire\_date%TYPE;

v\_salary employees.salary%TYPE;

CURSOR c\_employees IS

SELECT employee\_id, first\_name || ' ' || last\_name AS full\_name, job\_id, hire\_date, salary  
FROM employees;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Employee ID | Full Name | Job Title | Hire Date | Salary');

DBMS\_OUTPUT.PUT\_LINE('-----');

OPEN c\_employees;

FETCH c\_employees INTO v\_employee\_id, v\_full\_name, v\_job\_id, v\_hire\_date, v\_salary;

WHILE c\_employees%FOUND LOOP

DBMS\_OUTPUT.PUT\_LINE(v\_employee\_id || ' ' || v\_full\_name || ' ' || v\_job\_id || ' ' ||  
v\_hire\_date || ' ' || v\_salary);

FETCH c\_employees INTO v\_employee\_id, v\_full\_name, v\_job\_id, v\_hire\_date, v\_salary;

END LOOP;

CLOSE c\_employees;

END;

/

**OUTPUT:**

The screenshot shows the APEX SQL Workshop interface. At the top, there's a navigation bar with 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user information 'A2 Aktien Z2 bleach22' are on the right. Below the navigation bar, the 'SQL Commands' tab is active. The 'Language' is set to 'SQL' and 'Rows' is set to '10'. The 'Schema' is 'WKSP\_BLEACH22'. The 'Save' and 'Run' buttons are visible. The main area displays a PL/SQL program with line numbers 1 through 21. The program declares variables, opens a cursor, fetches data, and prints it using DBMS\_OUTPUT.PUT\_LINE. Below the code, the 'Results' tab is active, showing 'Statement processed.' and '0.03 seconds'.

```
1 DECLARE
2   v_employee_id employees.employee_id%TYPE;
3   v_first_name employees.first_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 emp 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

**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;

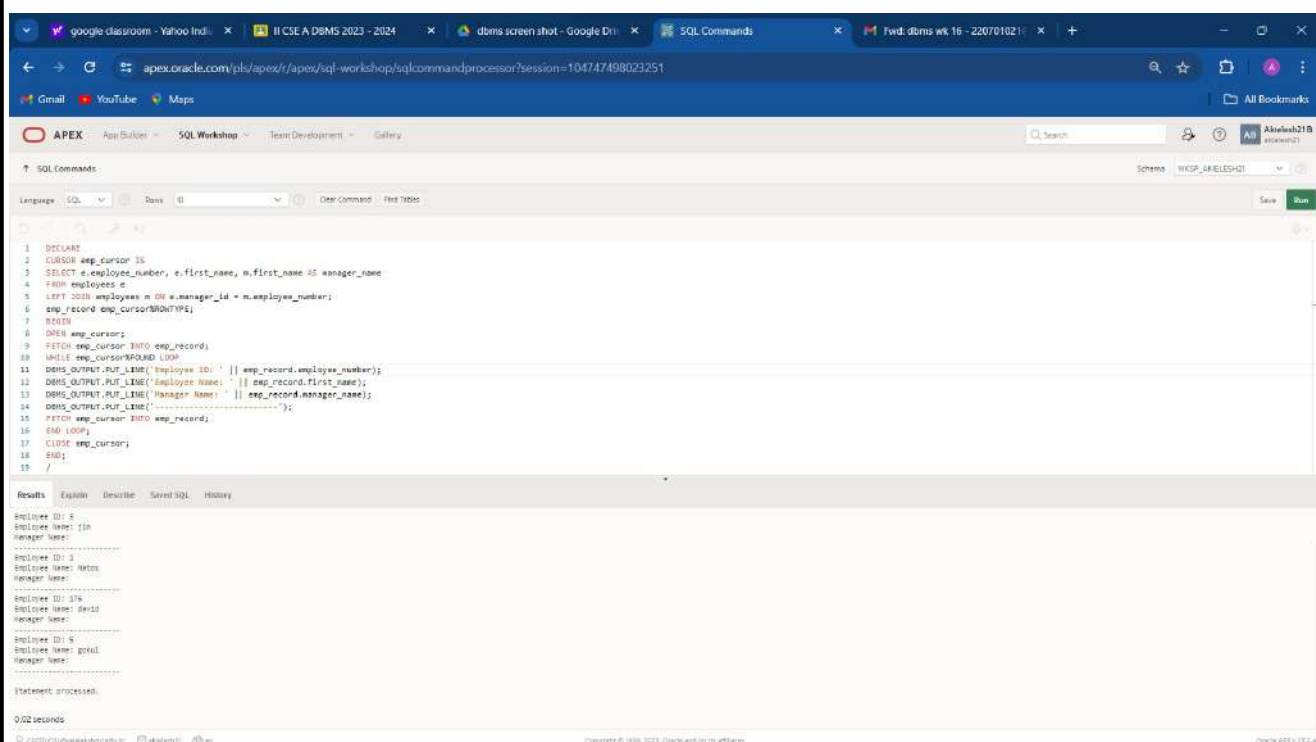
END LOOP;

CLOSE emp\_cursor;

END;

/

### OUTPUT:



The screenshot shows the APEX SQL Commands interface. The SQL command is as follows:

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

The output shows the results of the query for employees 1 through 5:

```
Employee ID: 1
Employee Name: John
Manager Name:
-----
Employee ID: 2
Employee Name: Neena
Manager Name:
-----
Employee ID: 3
Employee Name: Lex
Manager Name:
-----
Employee ID: 4
Employee Name: David
Manager Name:
-----
Employee ID: 5
Employee Name: Scott
Manager Name:
-----
Statement processed.
0.02 seconds
```

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

#### QUERY:

DECLARE

CURSOR job\_cursor IS

SELECT e.job\_id, j.lowest\_sal

FROM job\_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 displays the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user information 'A2 Aktien22 bleach22' are on the right. Below the navigation bar, the 'SQL Commands' section shows the PL/SQL code being executed. The code declares a cursor 'job\_cursor' to select job\_id and lowest\_sal from job\_grades and employees, then loops through the results, printing the job ID, minimum salary, and a separator line. The 'Results' tab at the bottom shows 'Statement processed.' and '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 SQL Commands window contains the following PL/SQL program:

```

1 emp_hire_date DATE;
2 BEGIN
3   dbms_output.put_line(Rpad('Employee ID', 15)||Rpad('Last Name', 25)|| Rpad('Job Id', 35)
4   ||'Start Date');
5   dbms_output.put_line('-----');
6   FOR emp_sal_rec IN employees_cur LOOP
7     -- find out most recent end_date in job_history
8     SELECT Max(end_date) + 1
9     INTO emp_hire_date
10    FROM job_history
11   WHERE emp_id = emp_sal_rec.employee_number;
12   IF emp_hire_date IS NULL THEN
13     emp_hire_date := emp_sal_rec.hire_date;
14   END IF;
15   dbms_output.put_line(Rpad(emp_sal_rec.employee_number, 15)

```

The Results window shows the output of the program:

Employee ID	Last Name	Job Id	Start Date
1	AKIELESH	ADP	22-07-2021

The statement was processed successfully in 0.01 seconds.

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

#### QUERY:

DECLARE

v\_employee\_id employees.employee\_id%TYPE;

v\_first\_name employees.last\_name%TYPE;

v\_end\_date job\_history.end\_date%TYPE;

CURSOR c\_employees IS

SELECT e.employee\_id, e.first\_name, jh.end\_date

FROM employees e

JOIN job\_history jh ON e.employee\_id = jh.employee\_id;

BEGIN

OPEN c\_employees;

FETCH c\_employees INTO v\_employee\_id, v\_first\_name, v\_end\_date;

WHILE c\_employees%FOUND LOOP

DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || v\_employee\_id);

DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || v\_first\_name);

DBMS\_OUTPUT.PUT\_LINE('End Date: ' || v\_end\_date);

DBMS\_OUTPUT.PUT\_LINE('-----');

FETCH c\_employees INTO v\_employee\_id, v\_first\_name, v\_end\_date;

END LOOP;

CLOSE c\_employees;

END;

#### OUTPUT:



RESULT:

# PROCEDURES AND FUNCTIONS

EX\_NO: 17

DATE:

## 1.)Factorial of a number using function.

### QUERY:

DECLARE

fac NUMBER := 1;

n NUMBER := :1;

BEGIN

WHILE n > 0 LOOP

fac := n \* fac;

n := n - 1;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(fac);

END;

### OUTPUT:

The screenshot displays the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'SQL Commands' tab is active, showing a SQL block with the following code:

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

The 'Results' tab is selected, showing the output of the statement: 'Statement processed.' and '0.01 seconds'.

At the bottom, the footer contains the text: 'Copyright © 1999, 2025, Oracle and/or its affiliates. Oracle APEX 23.2.4'.

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

**QUERY:**

```
CREATE OR REPLACE PROCEDURE get_book_info (  
    p_book_id IN NUMBER,  
    p_title IN OUT VARCHAR2,  
    p_author OUT VARCHAR2,  
    p_year_published OUT NUMBER  
)  
AS  
BEGIN  
    SELECT title, author, year_published INTO p_title, p_author, p_year_published  
    FROM books  
    WHERE book_id = p_book_id;  
  
    p_title := p_title || ' - Retrieved';  
EXCEPTION  
    WHEN NO_DATA_FOUND THEN  
        p_title := NULL;  
        p_author := NULL;  
        p_year_published := NULL;  
END;  
  
DECLARE  
    v_book_id NUMBER := 1;
```

```

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

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

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

```

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The browser address bar indicates the URL: `apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=1735230419529`. The user is logged in as 'Akielesh21 B' with the email 'akielesh21'. The schema selected is 'WKSP\_AKIELESH21'.

The SQL Commands section shows the following script being executed:

```

23 v_book_id NUMBER := 1;
24 v_title VARCHAR2(100);
25 v_author VARCHAR2(100);
26 v_year_published NUMBER;
27 BEGIN
28     v_title := 'Initial Title';
29
30     get_book_info(p_book_id => v_book_id, p_title => v_title, p_author => v_author, p_year_published => v_year_published);
31
32     DBMS_OUTPUT.PUT_LINE('Title: ' || v_title);
33     DBMS_OUTPUT.PUT_LINE('Author: ' || v_author);
34     DBMS_OUTPUT.PUT_LINE('Year Published: ' || v_year_published);
35 END;
36

```

The Results section shows the output of the script:

```

Table created.

0.04 seconds

```

The footer of the page includes the user's email '220701021@rajalakshmi.edu.in', the username 'akielesh21', the language 'en', the copyright notice 'Copyright © 1999, 2025, Oracle and/or its affiliates.', and the version 'Oracle APEX 23.2.4'.

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

**RESULT:**

# TRIGGER

**EX\_NO: 18**

**DATE:**

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the Oracle APEX SQL Workshop interface. The browser address bar displays the URL: `apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=1735230419529`. The page header includes the APEX logo, navigation links (App Builder, SQL Workshop, Team Development, Gallery), a search bar, and a user profile for 'Akielesh21 B'. The main content area is titled 'SQL Commands' and shows a schema dropdown set to 'WKSP\_AKIELESH21'. Below this, there are controls for 'Language' (SQL), 'Rows' (10), and buttons for 'Clear Command', 'Find Tables', 'Save', and 'Run'. The SQL editor contains the following PL/SQL code:

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

Below the editor, the 'Results' tab is active, displaying the message 'Trigger created.' and the execution time '0.04 seconds'. The footer shows the user's email '220701021@rajatashrimaliedu.in', the username 'akielesh21', the language 'en', the copyright notice 'Copyright © 1999, 2025, Oracle and/or its affiliates.', and the version 'Oracle APEX 23.3.4'.

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

**QUERY:**

```
CREATE OR REPLACE TRIGGER check_duplicates
BEFORE INSERT OR UPDATE ON unique_values_table
FOR EACH ROW
DECLARE
    duplicate_found EXCEPTION;
    PRAGMA EXCEPTION_INIT(duplicate_found, -20002);
    v_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_count FROM unique_values_table
```

```
WHERE unique_col = :NEW.unique_col AND id != :NEW.id;
IF v_count > 0 THEN
    RAISE duplicate_found;
END IF;
EXCEPTION
    WHEN duplicate_found THEN
        RAISE_APPLICATION_ERROR(-20002, 'Duplicate value found in unique_col.');
```

END;

## OUTPUT:

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes links for App Builder, SQL Workshop, Team Development, and Gallery. The main area displays the SQL Commands tab with a schema dropdown set to WKSP\_AKIELESH21. The SQL editor contains the following code:

```
1  -- Create the unique_values_table
2  CREATE TABLE unique_values_table (
3      id NUMBER PRIMARY KEY,
4      unique_col VARCHAR2(50) UNIQUE,
5      other_col VARCHAR2(50)
6  );
7
8  -- Create the trigger check_duplicates
9  CREATE OR REPLACE TRIGGER check_duplicates
10 BEFORE INSERT OR UPDATE ON unique_values_table
11 FOR EACH ROW
12 DECLARE
13     duplicate_found EXCEPTION;
14     PRAGMA EXCEPTION_INIT(duplicate_found, -20002);
15     v_count NUMBER;
```

Below the editor, the Results tab shows the message "Trigger created." and the execution time "0.05 seconds". The footer includes the user "220701021@rajatashmileduan", the username "akiesh21", and the Oracle APEX version "23.2.4".

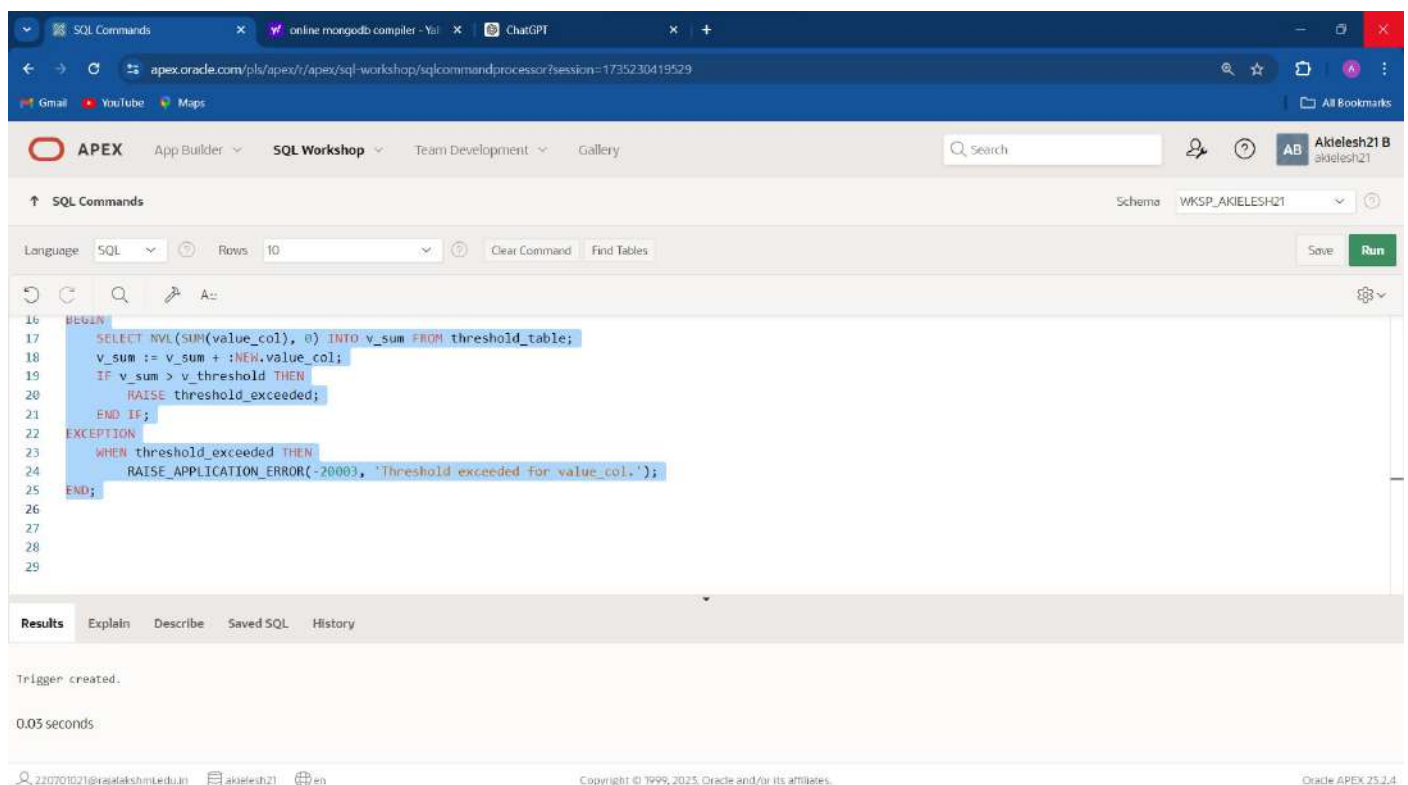
### 3.) Write a code in PL/SQL to create a trigger that restricts the insertion of new rows if the total of a column's values exceeds a certain threshold

#### QUERY:

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

END;

#### OUTPUT:



The screenshot displays the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The 'SQL Commands' tab is active, showing a query editor with the following PL/SQL code:

```
16 BEGIN
17     SELECT NVL(SUM(value_col), 0) INTO v_sum FROM threshold_table;
18     v_sum := v_sum + :NEW.value_col;
19     IF v_sum > v_threshold THEN
20         RAISE threshold_exceeded;
21     END IF;
22 EXCEPTION
23     WHEN threshold_exceeded THEN
24         RAISE_APPLICATION_ERROR(-20003, 'Threshold exceeded for value_col.');
```

The code is highlighted in blue. Below the editor, the 'Results' tab is selected, displaying the message 'Trigger created.' and the execution time '0.05 seconds'. The bottom status bar shows the user '220701021@rajatsharma.edu.in', the workspace 'akiesh21', and the version 'Oracle APEX 23.2.4'.

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

**QUERY:**

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

**OUTPUT:**

SQL Commands

online mongodb compiler - Yal

ChatGPT

apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=1735230419529

GmailYouTubeMaps

APEX

App Builder

SQL Workshop

Team Development

Gallery

Search

AB

Akiesh21 B  
akiesh21

SQL Commands

Schema

WKSP\_AKIELESH21

Language

SQL

Rows

10

Clear Command

Find Tables

Save

Run

1

-- Create the main\_table

2

CREATE TABLE main\_table (

3

id NUMBER PRIMARY KEY,

4

col1 VARCHAR2(50),

5

col2 VARCHAR2(50)

6

);

7

8

-- Create the audit\_table

9

CREATE TABLE audit\_table (

10

audit\_id NUMBER PRIMARY KEY,

11

changed\_id NUMBER,

12

old\_col1 VARCHAR2(50),

13

new\_col1 VARCHAR2(50),

14

old\_col2 VARCHAR2(50),

15

new\_col2 VARCHAR2(50),

16

change\_time TIMESTAMP

17

);

18

Results

Explain

Describe

Saved SQL

History

Error at line 1/54: ORA-00933: SQL command not properly ended

220701021@rajatashrmi.edu.in

akiesh21

en

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

Oracle APEX 23.2.4

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

### QUERY:

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

### OUTPUT:

The screenshot displays the Oracle APEX SQL Workshop interface. The top navigation bar includes links for APEX, App Builder, SQL Workshop, Team Development, and Gallery. The main workspace shows the 'SQL Commands' tab with a schema dropdown set to 'WKSP\_AKIELESH21'. The SQL editor contains the PL/SQL code for creating the trigger, which is identical to the code provided in the 'QUERY' section. The 'Run' button is highlighted in green. Below the editor, the 'Results' tab is active, displaying the message 'Trigger created.' and the execution time '0.03 seconds'. The footer of the interface shows the user '220701021@rajalakshmi.edu.in', the session 'akiesh21', and the Oracle APEX version '23.2.4'.

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

**QUERY:**

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

**OUTPUT:**

SQL Commands

online mongodb compiler - Yal

ChatGPT

apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=1735230419529

GmailYouTubeMaps

APEX

App Builder

SQL Workshop

Team Development

Gallery

Search

AB

Akielesh21 B

akielesh21

SQL Commands

SchemaWKSP\_AKIELESH21

LanguageSQLRows10Clear CommandFind TablesSaveRun

↶↷🔍🔗A-Z⚙️

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

Results

Explain

Describe

Saved SQL

History

Trigger created.

0.03 seconds

220701021@rajatashriniedu.in

akielesh21

en

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

Oracle APEX 23.3.4



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

**QUERY:**

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

END;

**OUTPUT:**

SQL Commands

online mongodb compiler - Yal

ChatGPT

apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=1735230419529

Gmail

YouTube

Maps

APEX

App Builder

SQL Workshop

Team Development

Gallery

Search

AB Akelesh21 B akelesh21

SQL Commands

Schema WKSP\_AKIELESH21

Language SQL

Rows 10

Clear Command

Find Tables

Save

Run

```

5      v_stock NUMBER;
6      insufficient_stock EXCEPTION;
7      PRAGMA EXCEPTION_INIT(insufficient_stock, -20004);
8  BEGIN
9      SELECT stock_quantity INTO v_stock FROM items WHERE item_id = :NEW.item_id;
10     IF v_stock < :NEW.order_quantity THEN
11         RAISE insufficient_stock;
12     END IF;
13     UPDATE items SET stock_quantity = stock_quantity - :NEW.order_quantity WHERE item_id = :NEW.item_id;
14 EXCEPTION
15     WHEN insufficient_stock THEN
16         RAISE_APPLICATION_ERROR(-20004, 'Insufficient stock for the item.');
```

Results

Explain

Describe

Saved SQL

History

Trigger created.

0.03 seconds

220701021@rajalakshmi.edu.in

akelesh21

en

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

Oracle APEX 23.3.4

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

**RESULT:**

# MONGO DB

**EX\_NO: 19**

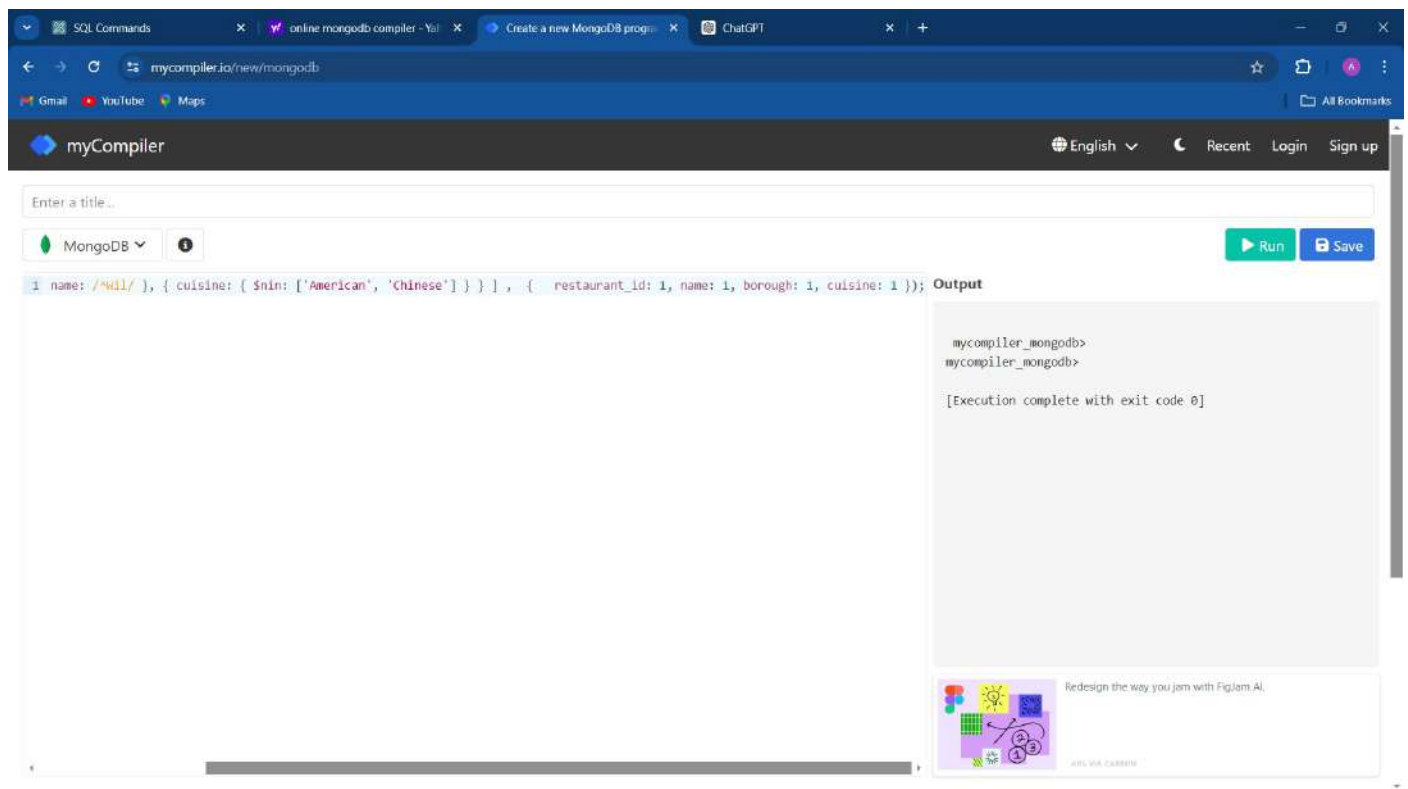
**DATE:**

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

**QUERY:**

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

**OUTPUT:**

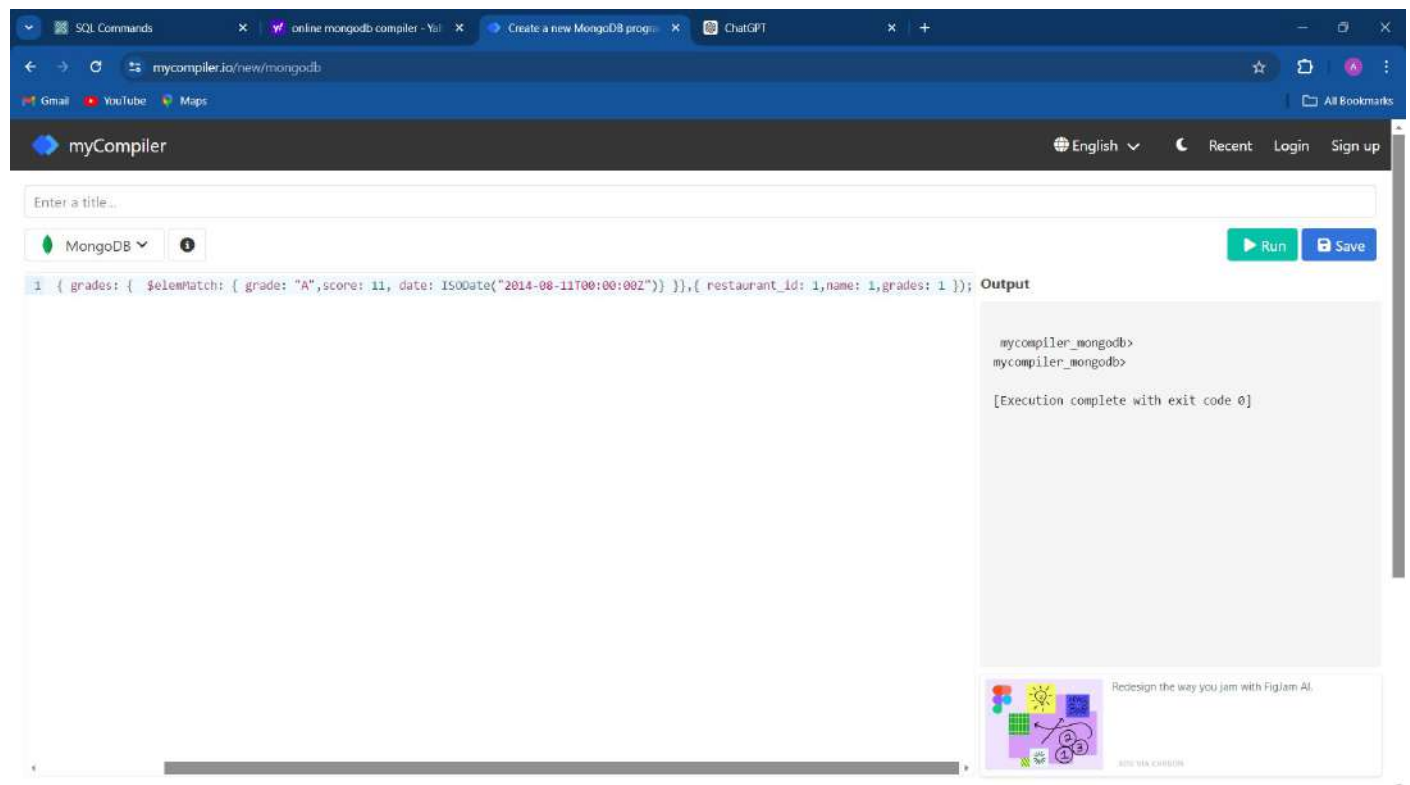


2.)Write a MongoDB query to find the restaurant Id, name, and grades for those restaurants which achieved a grade of "A" and scored 11 on an ISODate "2014-08- 11T00:00:00Z" among many of survey dates.

**QUERY:**

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

### OUTPUT:



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

### QUERY:

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

### OUTPUT:

The screenshot shows the myCompiler.io web interface for MongoDB. The browser tabs include 'SQL Commands', 'online mongodb compiler - Yal', 'Create a new MongoDB program', and 'ChatGPT'. The address bar shows 'mycompiler.io/new/mongodb'. The page has a dark header with 'myCompiler' logo, language selector (English), and links for 'Recent', 'Login', and 'Sign up'. Below the header is a form to 'Enter a title...' and a dropdown menu set to 'MongoDB'. To the right of the dropdown are 'Run' and 'Save' buttons. The main area contains a MongoDB query: `insert.1.grade": "A", "grades.1.score": 9, "grades.1.date": ISODate("2014-08-17T00:00:00Z") }, { "restaurant_id": 1, name: 1, grades: 1 } };`. To the right of the query is an 'Output' section showing the command prompt: `mycompiler_mongodb>` and the message: `[Execution complete with exit code 0]`. At the bottom right, there is a small advertisement for FigJam AI.

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

#### QUERY:

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

#### OUTPUT:

The screenshot shows the myCompiler.io web interface for MongoDB. The browser tabs include 'SQL Commands', 'online mongodb compiler - Yal...', 'Create a new MongoDB program', and 'ChatGPT'. The address bar shows 'mycompiler.io/new/mongodb'. The page has a dark blue header with the 'myCompiler' logo and links for 'English', 'Recent', 'Login', and 'Sign up'. Below the header is a form to 'Enter a title...'. The main area has a 'MongoDB' dropdown and a 'Run' button. The code editor contains the following MongoDB query:

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

The output window shows the following text:

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

At the bottom right, there is an advertisement for FigJam AI with the text 'Redesign the way you jam with FigJam AI.' and 'BUILT VIA CHORDS'.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the myCompiler.io web interface for MongoDB. The browser tabs include 'SQL Commands', 'online mongodb compiler - Yal...', 'Create a new MongoDB program...', and 'ChatGPT'. The address bar shows 'mycompiler.io/new/mongodb'. The page has a dark blue header with the 'myCompiler' logo and links for 'English', 'Recent', 'Login', and 'Sign up'. Below the header is a form to 'Enter a title...'. The main area has a 'MongoDB' dropdown and a 'Run' button. The code editor contains the query: `1 db.restaurants.find({}, { _id: 0 }).sort({ name: 1 });`. The 'Output' panel shows the command prompt: `mycompiler_mongodb>`, `mycompiler_mongodb>`, and the message: `[Execution complete with exit code 0]`. At the bottom, there is an advertisement for FigJam AI with the text 'Redesign the way you jam with FigJam AI.' and 'BUILT VIA CHORDS'.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the myCompiler.io web interface for MongoDB. The browser tabs include 'SQL Commands', 'online mongodb compiler - Yal...', 'Create a new MongoDB program...', and 'ChatGPT'. The address bar shows 'mycompiler.io/new/mongodb'. The page has a dark blue header with the 'myCompiler' logo and links for 'English', 'Recent', 'Login', and 'Sign up'. Below the header is a search bar with the text 'Enter a title...'. A dropdown menu shows 'MongoDB' with a green leaf icon. To the right are 'Run' and 'Save' buttons. The main area contains a code editor with the following query: 

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

. To the right of the code editor is an 'Output' panel showing the execution results: 

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

. At the bottom of the page, there is a promotional banner for FigJam AI with the text 'Redesign the way you jam with FigJam AI.' and 'JOIN VIA CHATBOT'.

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows a web browser with the myCompiler.io interface. The browser tabs include 'SQL Commands', 'online mongodb compiler - Yal', 'Create a new MongoDB program', and 'ChatGPT'. The address bar shows 'mycompiler.io/new/mongodb'. The page has a dark blue header with the myCompiler logo and navigation links for 'English', 'Recent', 'Login', and 'Sign up'. Below the header, there is a text input field 'Enter a title...' and a dropdown menu set to 'MongoDB'. To the right of the dropdown are 'Run' and 'Save' buttons. The main area contains a code editor with the following MongoDB query:

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

To the right of the code editor is an 'Output' panel. It shows the command prompt 'mycompiler\_mongodb>' and the message '[Execution complete with exit code 0]'. At the bottom of the page, there is a small advertisement for FigJam AI with the text 'Redesign the way you jam with FigJam AI.' and 'BUILT VIA CHORDS'.

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

**QUERY:**

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

**OUTPUT:**

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

**QUERY:**

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

**OUTPUT:**

myCompiler

Enter a title...

MongoDB

Run Save

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

Output

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

Redesign the way you jam with FigJam AI.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the myCompiler.io web interface for MongoDB. The browser tabs include 'SQL Commands', 'online mongodb compiler - Yal', 'Create a new MongoDB program', and 'ChatGPT'. The address bar shows 'mycompiler.io/new/mongodb'. The page has a dark blue header with the 'myCompiler' logo and links for 'English', 'Recent', 'Login', and 'Sign up'. Below the header is a search bar with the text 'Enter a title...'. A dropdown menu shows 'MongoDB' with a green leaf icon. To the right are 'Run' and 'Save' buttons. The main area contains a code editor with the following query: 

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

 The 'Output' panel on the right shows the command prompt: 

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

 At the bottom right, there is a small advertisement for FigJam AI with the text 'Redesign the way you jam with FigJam AI.' and 'BUILT VIA CHORDS'.

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

**QUERY:**

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

**OUTPUT:**

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

**QUERY:**

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

**OUTPUT:**

myCompiler

Enter a title...

MongoDB

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

Run Save

Output

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

Redesign the way you jam with FigJam AI.

AI/UX VIA CHRON

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a web browser with the myCompiler.io interface. The browser tabs include 'SQL Commands', 'online mongodb compiler - Yal...', 'Create a new MongoDB program...', and 'ChatGPT'. The address bar shows 'mycompiler.io/new/mongodb'. The page has a dark blue header with the myCompiler logo and links for 'English', 'Recent', 'Login', and 'Sign up'. Below the header is a search bar with the text 'Enter a title...'. A dropdown menu shows 'MongoDB' with a green icon. To the right of the dropdown are 'Run' and 'Save' buttons. The main area contains a code editor with the following query: 

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

. To the right of the code editor is an 'Output' panel. The output panel shows the following text: 

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

. At the bottom of the output panel is a small advertisement for FigJam AI with the text 'Redesign the way you jam with FigJam AI.' and 'BRIAN VIA CHRONIC'.

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

**QUERY:**

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

**OUTPUT:**

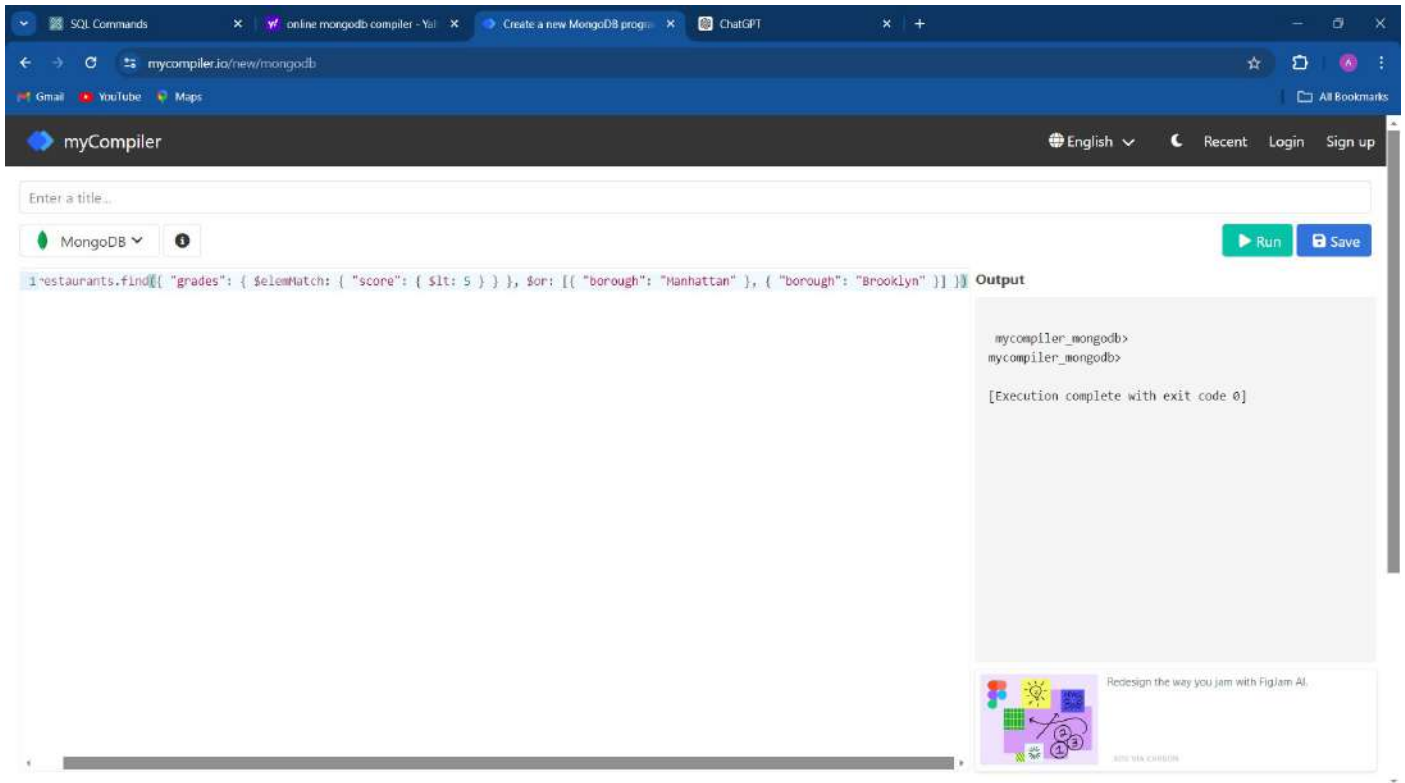
15. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn.

**QUERY:**

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

**OUTPUT:**





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

**QUERY:**

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

**OUTPUT:**

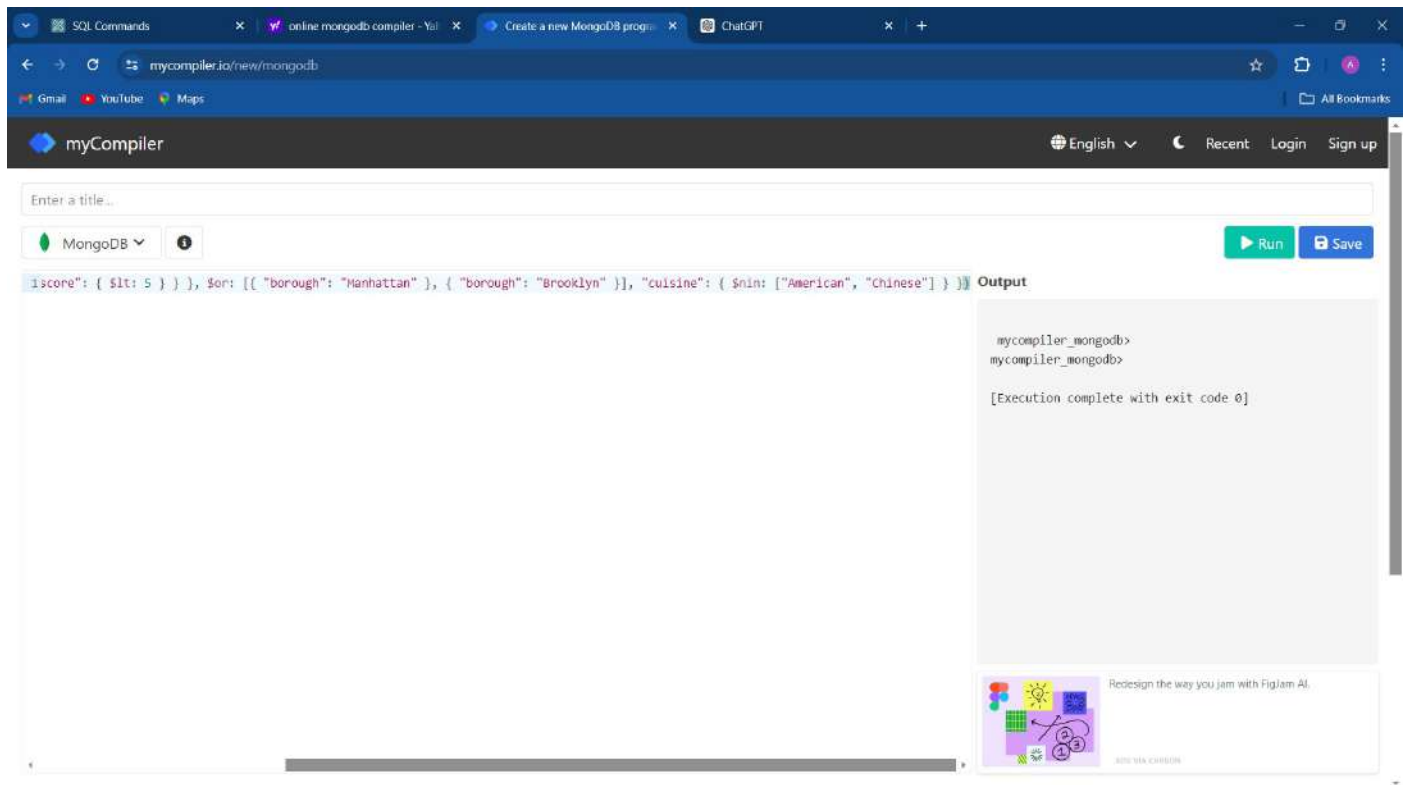
The screenshot shows the myCompiler.io web interface. At the top, there are browser tabs for 'SQL Commands', 'online mongodb compiler - Yal...', 'Create a new MongoDB program', and 'ChatGPT'. The address bar shows 'mycompiler.io/new/mongodb'. The page has a dark blue header with the 'myCompiler' logo and links for 'English', 'Recent', 'Login', and 'Sign up'. Below the header, there's a text input 'Enter a title...' and a dropdown menu set to 'MongoDB'. To the right of the dropdown are 'Run' and 'Save' buttons. The main area contains a MongoDB query in a text editor: `{ $elemMatch: { "score": { $lt: 5 } } }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $ne: "American" } }`. To the right of the editor is an 'Output' panel showing the command prompt: `mycompiler_mongodb>`, `mycompiler_mongodb>`, and the message `[Execution complete with exit code 0]`. At the bottom right, there's a small advertisement for FigJam AI.

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

**QUERY:**

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

**OUTPUT:**



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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the myCompiler.io web interface for MongoDB. The browser tabs include 'SQL Commands', 'online mongodb compiler - Yal...', 'Create a new MongoDB program...', and 'ChatGPT'. The address bar shows 'mycompiler.io/new/mongodb'. The page has a dark blue header with the 'myCompiler' logo and links for 'English', 'Recent', 'Login', and 'Sign up'. Below the header is a search bar with the text 'Enter a title...'. A dropdown menu shows 'MongoDB' with a green leaf icon. To the right are 'Run' and 'Save' buttons. The main area contains a code editor with the following query: 

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

 The 'Output' panel on the right shows the command prompt: 

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

 At the bottom right, there is a small advertisement for FigJam AI with the text 'Redesign the way you jam with FigJam AI.' and 'BUILT VIA CHORDS'.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the myCompiler.io web interface. The browser tabs include 'SQL Commands', 'online mongodb compiler - Yal...', 'Create a new MongoDB program', and 'ChatGPT'. The address bar shows 'mycompiler.io/new/mongodb'. The page has a dark blue header with 'myCompiler' and navigation links for 'English', 'Recent', 'Login', and 'Sign up'. Below the header is a search bar with 'Enter a title...' and a 'MongoDB' dropdown menu. To the right of the search bar are 'Run' and 'Save' buttons. The main area contains a MongoDB query in a text editor: `restaurants.find({ $and: [{ "grades.grade": "A", "grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], "borough": "Manhattan" })`. To the right of the editor is an 'Output' section showing the command prompt: `mycompiler_mongodb>`, `mycompiler_mongodb>`, and the message `[Execution complete with exit code 0]`. At the bottom right, there is a small advertisement for FigJam AI.

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

**QUERY:**

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

**OUTPUT:**

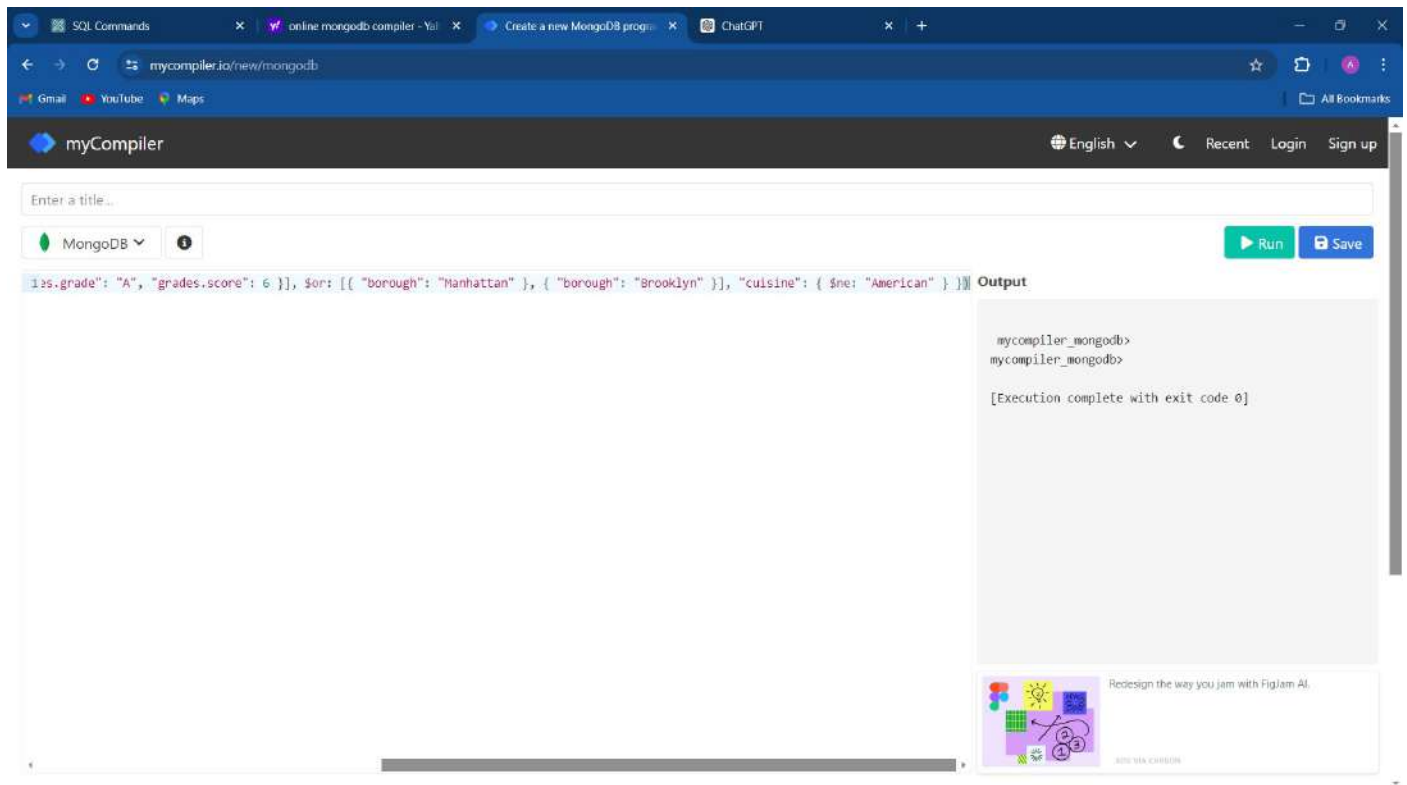
The screenshot shows the myCompiler.io web interface for MongoDB. The browser tabs include 'SQL Commands', 'online mongodb compiler - Yal...', 'Create a new MongoDB program', and 'ChatGPT'. The address bar shows 'mycompiler.io/new/mongodb'. The page has a dark blue header with 'myCompiler' and navigation links for 'English', 'Recent', 'Login', and 'Sign up'. Below the header is a search bar with 'Enter a title...' and a 'MongoDB' dropdown menu. To the right of the search bar are 'Run' and 'Save' buttons. The main area contains a MongoDB query in a text editor: `1\{"grades.score": 2 }, { "grades.grade": "A", "grades.score": 6 }], $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" } ]}`. To the right of the editor is an 'Output' section showing the execution results: `mycompiler_mongodb>  
mycompiler_mongodb>  
[Execution complete with exit code 0]`. At the bottom right, there is a small advertisement for FigJam AI.

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

**QUERY:**

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

**OUTPUT:**



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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the myCompiler.io web interface for MongoDB. The browser tabs include 'SQL Commands', 'online mongodb compiler - Yal...', 'Create a new MongoDB program', and 'ChatGPT'. The address bar shows 'mycompiler.io/new/mongodb'. The page has a dark blue header with the myCompiler logo and links for 'English', 'Recent', 'Login', and 'Sign up'. Below the header is a search bar with the text 'Enter a title...'. A dropdown menu shows 'MongoDB' with a green leaf icon. To the right are 'Run' and 'Save' buttons. The main area contains a MongoDB query: 

```
1, "grades.score": 6 }, $or: [{ "borough": "Manhattan" }, { "borough": "Brooklyn" }], "cuisine": { $in: ["American", "Chinese"] } }
```

. To the right of the query is an 'Output' section with a light gray background. It shows the command prompt 'mycompiler\_mongodb>' and the message '[Execution complete with exit code 0]'. At the bottom right, there is a small advertisement for FigJam AI with the text 'Redesign the way you jam with FigJam AI.' and 'BRIAN VIA CHRON'.

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:**



SQL Commands

online mongodb compiler - Yal

Create a new MongoDB program

ChatGPT

mycompiler.io/new/mongodb

Gmail

YouTube

Maps

myCompiler

English

Recent

Login

Sign up

Enter a title...

MongoDB

1

Run


Save

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

Output

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```



Redesign the way you jam with FigJam AI.

AI/ML VIA CHORDS

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

Faculty Signature	
----------------------	--

**RESULT:**

# MONGO DB

**EX\_NO: 20**

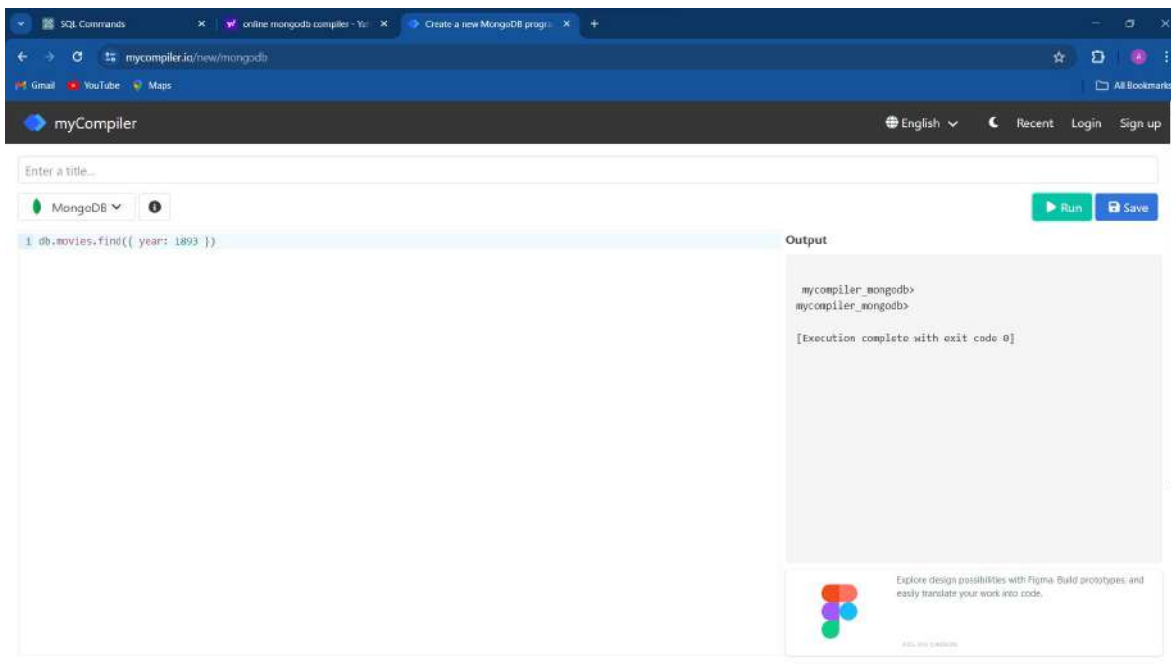
**DATE:**

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

**QUERY:**

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

**OUTPUT:**

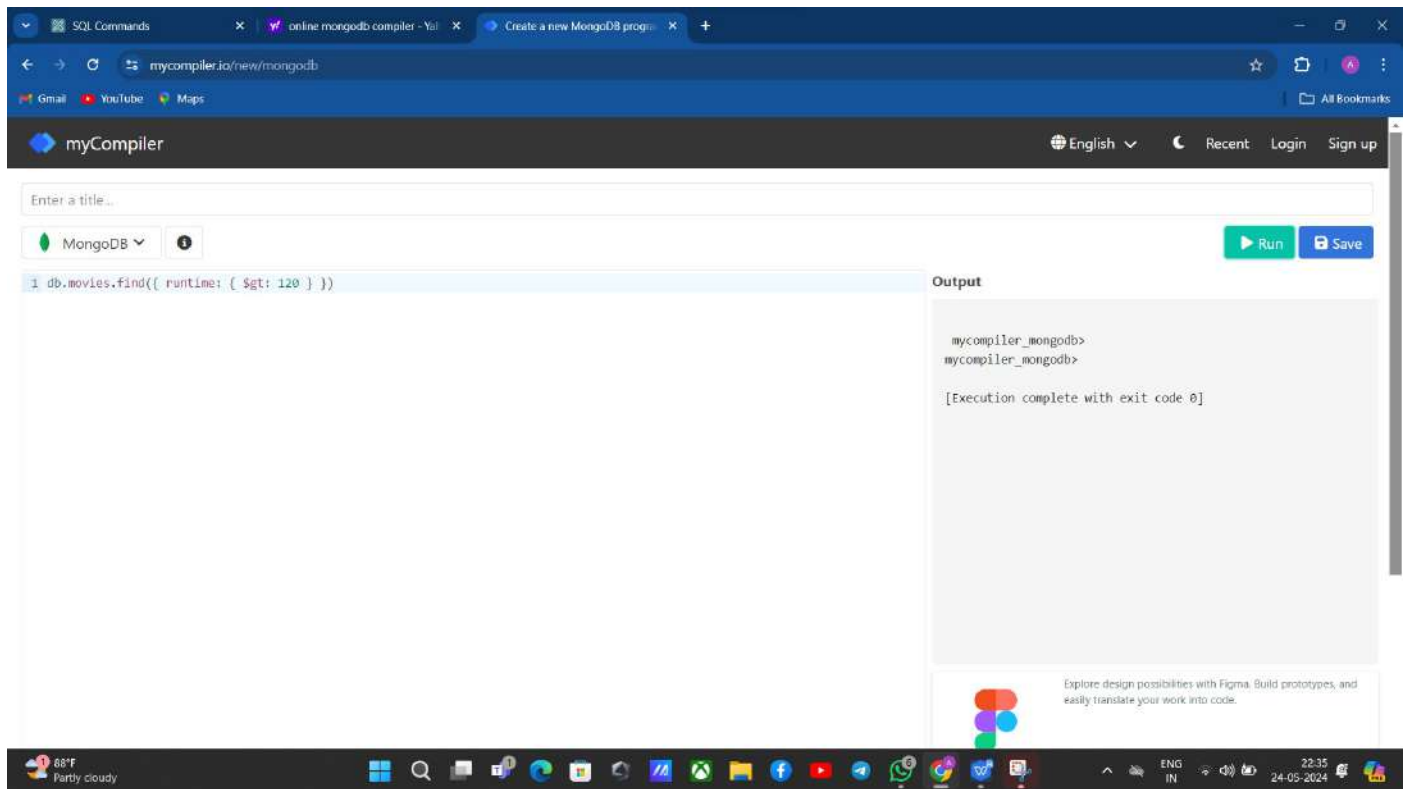


**2.) Find all movies with full information from the 'movies' collection that have a runtime greater than 120 minutes.**

**QUERY:**

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

**OUTPUT:**



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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the myCompiler.io web interface for MongoDB. The browser tabs include 'SQL Commands', 'online mongodb compiler - Yal...', and 'Create a new MongoDB program...'. The address bar shows 'mycompiler.io/new/mongodb'. The page has a dark blue header with the myCompiler logo and navigation links for 'English', 'Recent', 'Login', and 'Sign up'. Below the header, there is a text input field 'Enter a title...' and a dropdown menu set to 'MongoDB'. To the right of the dropdown are 'Run' and 'Save' buttons. The main area contains a code editor with the query: 

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

. To the right of the code editor is an 'Output' panel showing the execution results: 

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

. At the bottom right, there is a Figma advertisement with the text 'Explore design possibilities with Figma. Build prototypes, and easily translate your work into code.' and a 'GET VIA CARRDS' link.

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a web browser window with the URL `mycompiler.io/new/mongodb`. The page has a dark blue header with the myCompiler logo and navigation links for English, Recent, Login, and Sign up. Below the header, there is a text input field for a title, a dropdown menu set to 'MongoDB', and 'Run' and 'Save' buttons. The main area contains a code editor with the following query:

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

To the right of the code editor is an 'Output' panel. It shows the following text:

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

At the bottom right, there is a small advertisement for Figma with the text: 'Explore design possibilities with Figma. Build prototypes, and easily translate your work into code.'

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows a web browser window with the URL `mycompiler.io/new/mongodb`. The page has a dark blue header with the `myCompiler` logo and navigation links for `English`, `Recent`, `Login`, and `Sign up`. Below the header, there is a text input field labeled "Enter a title...". To the left of the code editor is a dropdown menu set to `MongoDB` with a small icon to its right. To the right of the code editor are two buttons: a green `Run` button and a blue `Save` button. The code editor contains the following MongoDB query:

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

To the right of the code editor is an `Output` panel. It displays the following text:

```
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

At the bottom right of the interface, there is a small advertisement for Figma with the text: "Explore design possibilities with Figma. Build prototypes, and easily translate your work into code."

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

**QUERY:**

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

**OUTPUT:**

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

**QUERY:**

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

**OUTPUT:**



The screenshot shows the myCompiler.io web interface for MongoDB. The browser tabs include 'SQL Commands', 'online mongodb compiler - Yal...', and 'Create a new MongoDB program...'. The address bar shows 'mycompiler.io/new/mongodb'. The page has a dark blue header with the myCompiler logo, language settings (English), and user options (Recent, Login, Sign up). Below the header, there's a form to 'Enter a title...' and a dropdown menu set to 'MongoDB'. A green 'Run' button and a blue 'Save' button are visible. The main area contains a MongoDB query: `1 db.movies.find([{"imdb.votes": {"$gt": 1000}}])`. To the right, the 'Output' section shows the command prompt `mycompiler_mongodb>` and the message `[Execution complete with exit code 0]`. At the bottom right, there's a Figma advertisement with the text 'Explore design possibilities with Figma. Build prototypes, and easily translate your work into code.'

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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the myCompiler.io web interface for MongoDB. The browser tabs include 'SQL Commands', 'online mongodb compiler - Yal...', and 'Create a new MongoDB program...'. The address bar shows 'mycompiler.io/new/mongodb'. The page has a dark blue header with the 'myCompiler' logo and navigation links for 'English', 'Recent', 'Login', and 'Sign up'. Below the header, there is a text input field 'Enter a title...' and a dropdown menu set to 'MongoDB'. To the right of the dropdown are 'Run' and 'Save' buttons. The main area contains a code editor with the following query: 

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

. To the right of the code editor is an 'Output' panel showing the command prompt 'mycompiler\_mongodb>' and the message '[Execution complete with exit code 0]'. At the bottom right, there is a Figma advertisement with the text 'Explore design possibilities with Figma. Build prototypes, and easily translate your work into code.' and a 'GET VIA CARRDS' link.

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

**QUERY:**

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

**OUTPUT:**

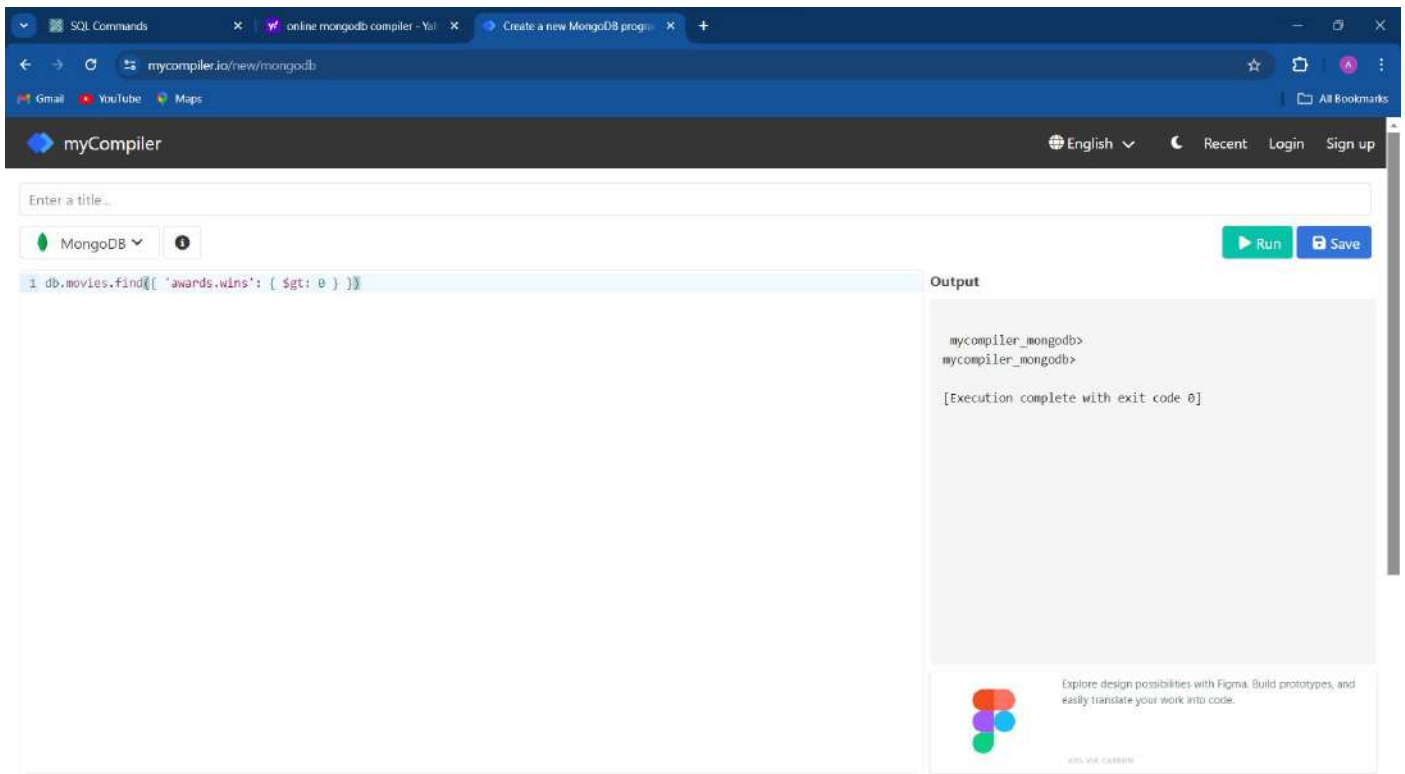
The screenshot shows the myCompiler.io web interface for MongoDB. The browser tabs include 'SQL Commands', 'online mongodb compiler - Yal...', and 'Create a new MongoDB program...'. The address bar shows 'mycompiler.io/new/mongodb'. The page has a dark blue header with the myCompiler logo and links for 'English', 'Recent', 'Login', and 'Sign up'. Below the header, there's a section to 'Enter a title...' and a dropdown menu set to 'MongoDB'. A green 'Run' button and a blue 'Save' button are visible. The main code editor contains the query: `1 db.movies.find([ 'tomatoes.viewer.rating': { $gt: 4 } ])]`. To the right, the 'Output' panel shows the command prompt `mycompiler_mongodb>` and the message `[Execution complete with exit code 0]`. At the bottom right, there is a Figma advertisement with the text 'Explore design possibilities with Figma. Build prototypes, and easily translate your work into code.'

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

**QUERY:**

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

**OUTPUT:**

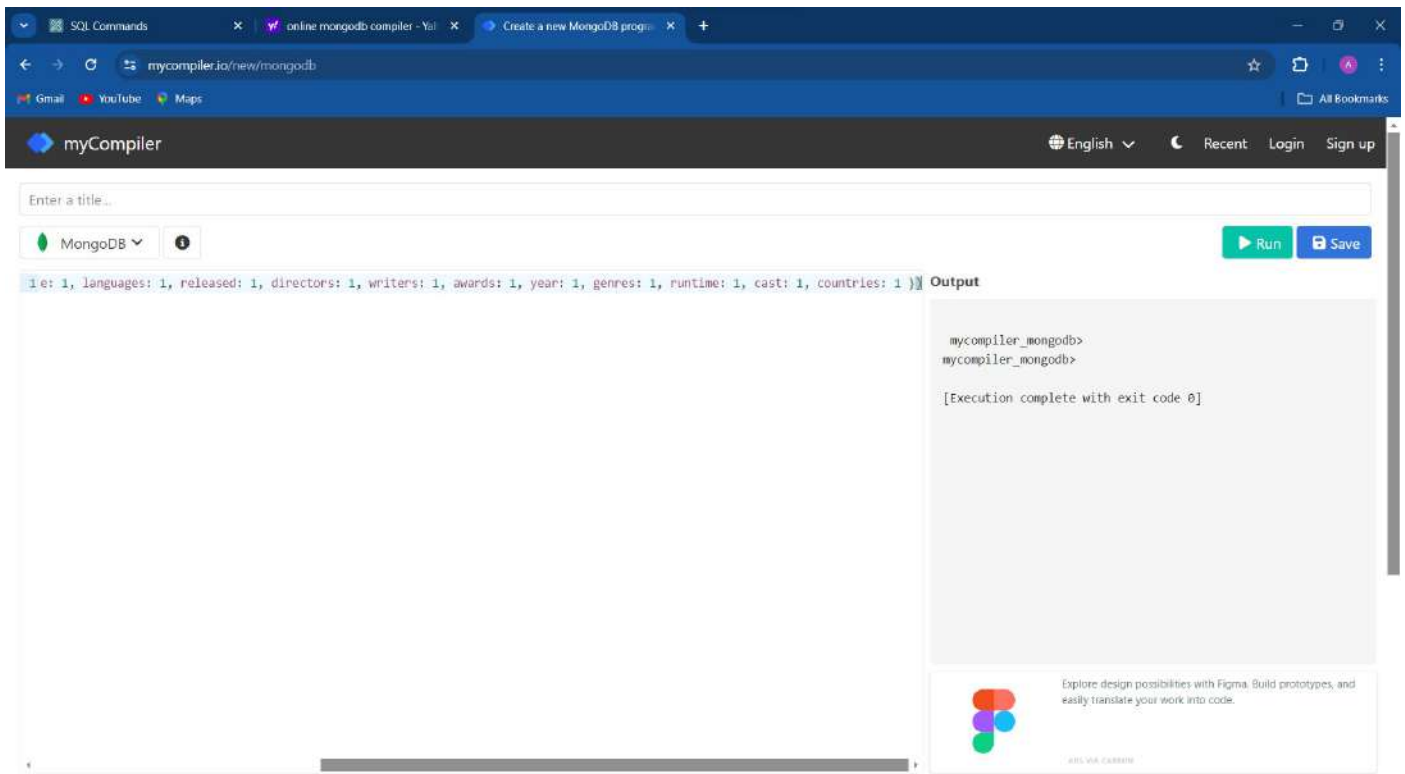


**11.) Find all movies with title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries from the 'movies' collection in MongoDB that have at least one nomination.**

**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:**

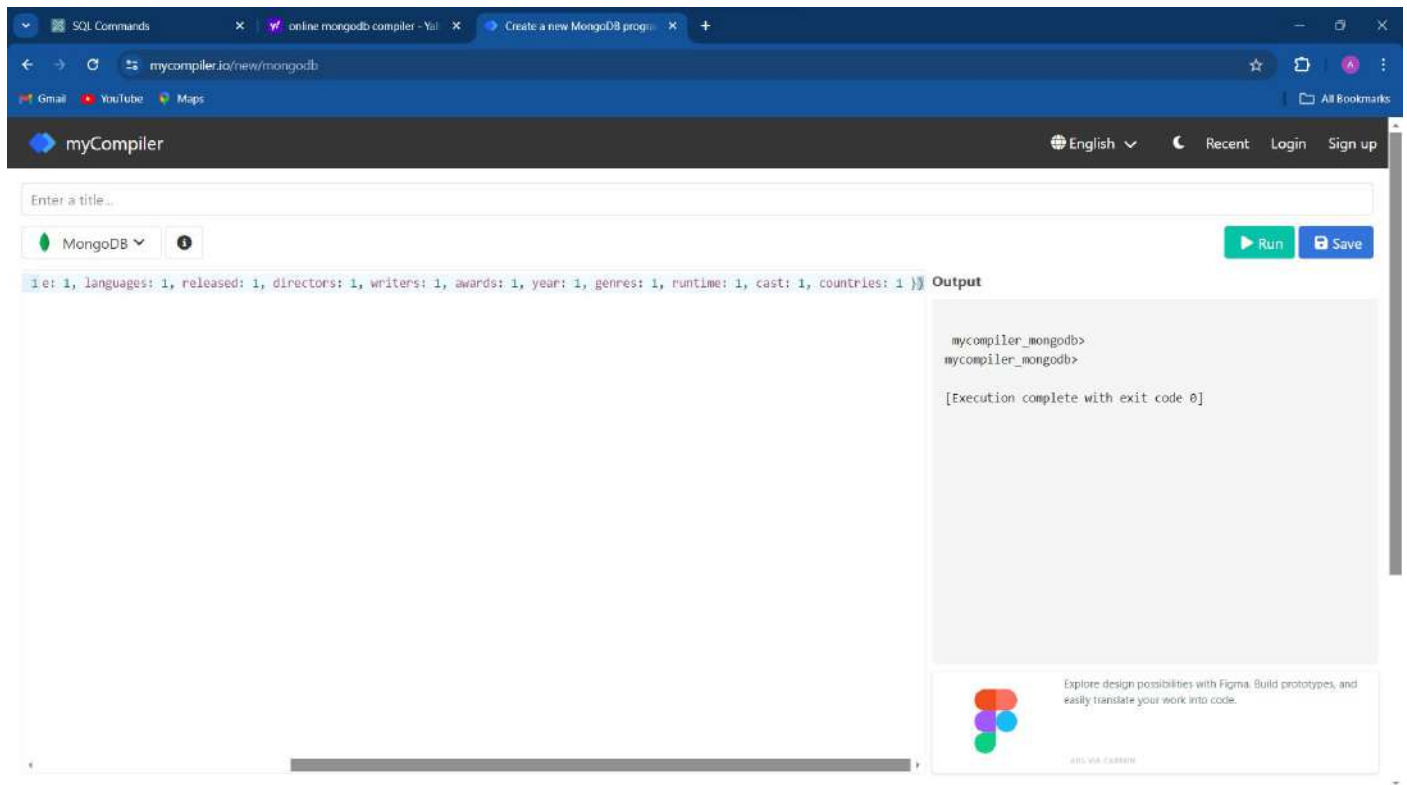


**12.) Find all movies with title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries from the 'movies' collection in MongoDB with cast including "Charles Kayser".**

**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:**



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

**QUERY:**

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

**OUTPUT:**

The screenshot shows the myCompiler.io web interface for MongoDB. The browser tabs include 'SQL Commands', 'online mongodb compiler - Yal', and 'Create a new MongoDB program'. The address bar shows 'mycompiler.io/new/mongodb'. The page has a dark blue header with the 'myCompiler' logo and links for 'English', 'Recent', 'Login', and 'Sign up'. Below the header, there's a form to 'Enter a title...' and a dropdown menu set to 'MongoDB'. To the right of the form are 'Run' and 'Save' buttons. The main area contains a MongoDB query: `find: ISODate("1893-05-09T00:00:00.000Z"), { title: 1, languages: 1, released: 1, directors: 1, writers: 1, countries: 1 }}`. To the right of the query is an 'Output' section showing the command `mycompiler_mongodb>` and the message `[Execution complete with exit code 0]`. At the bottom right, there is a Figma advertisement.

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

**QUERY:**

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

**OUTPUT:**

SQL Commands

online mongodb compiler - Yal

Create a new MongoDB program

mycompiler.io/new/mongodb

Gmail YouTube Maps

English Recent Login Sign up

Enter a title...

MongoDB

Run Save

```

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

```


Output

```

mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]

```



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

GET VIA CARRDS

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



Faculty Signature	
-------------------	--

**RESULT:**