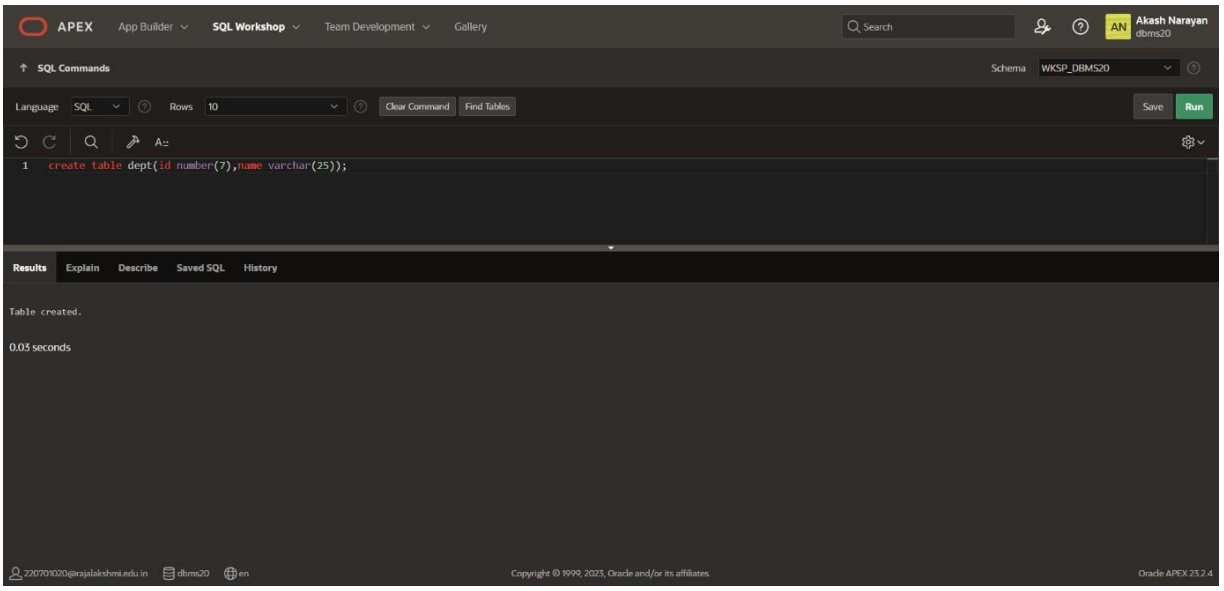
EXNO:1 **CREATING AND MANAGING TABLE** **DATE**:

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

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

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

OUTPUT:



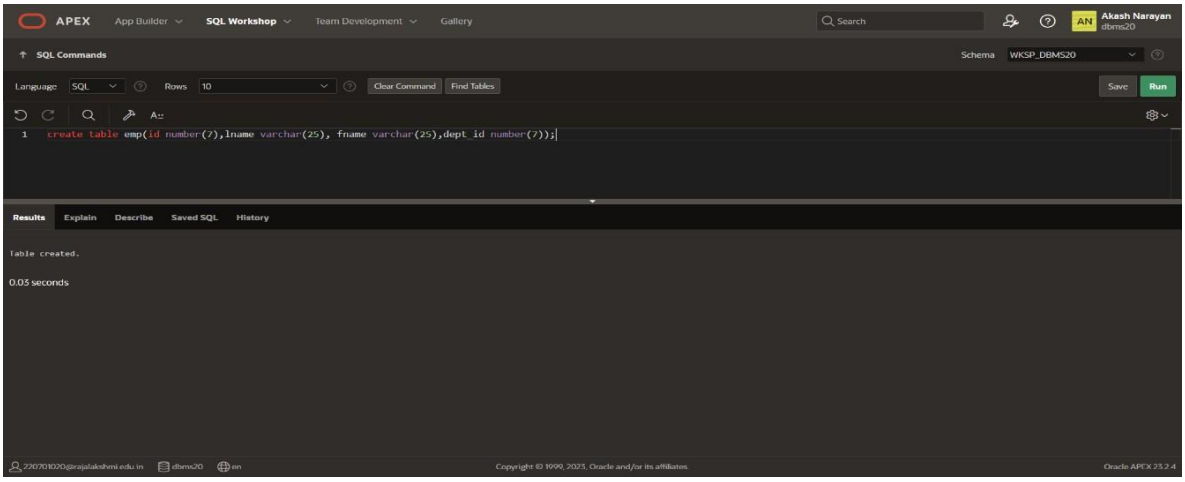
2.Create the EMP table based on the following instance chart. Confirm that the table is created.

Column name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				

Nulls/Unique				
FK table				
FK column				
Data Type	Number	Varchar2	Varchar2	Number
Length	7	25	25	7

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

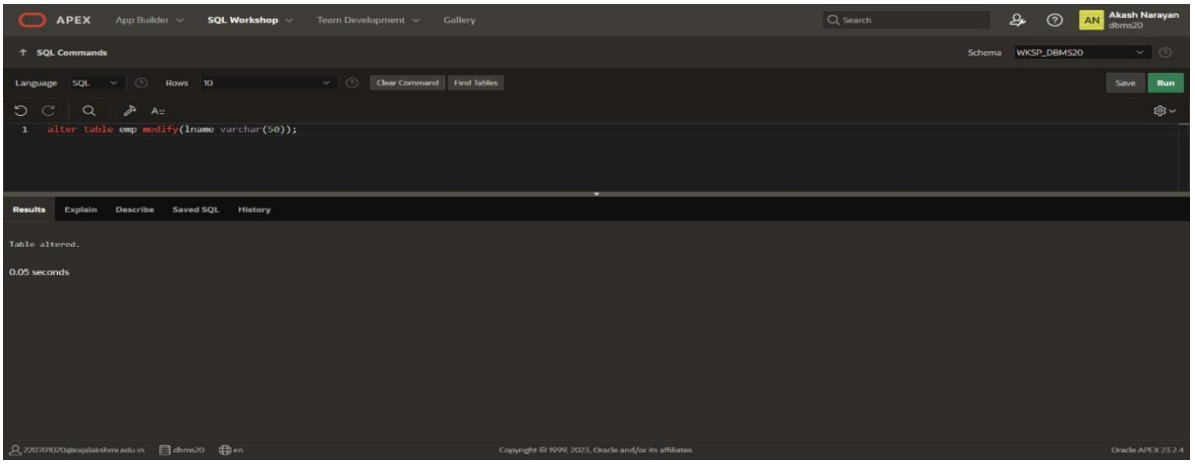
OUTPUT:



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

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

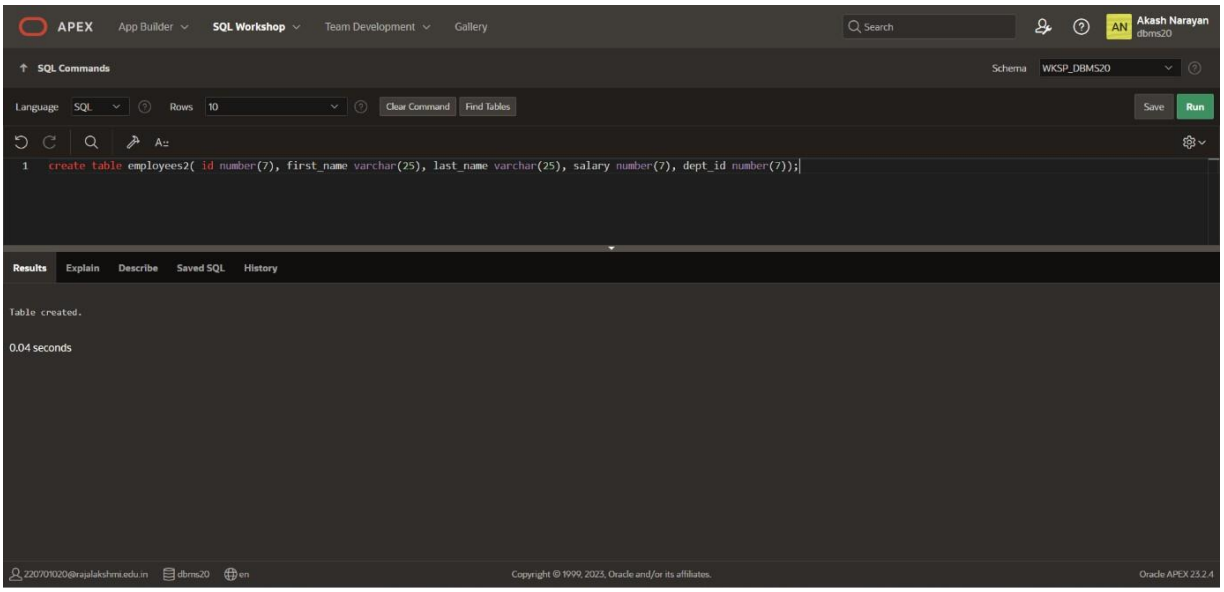
OUTPUT:



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

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

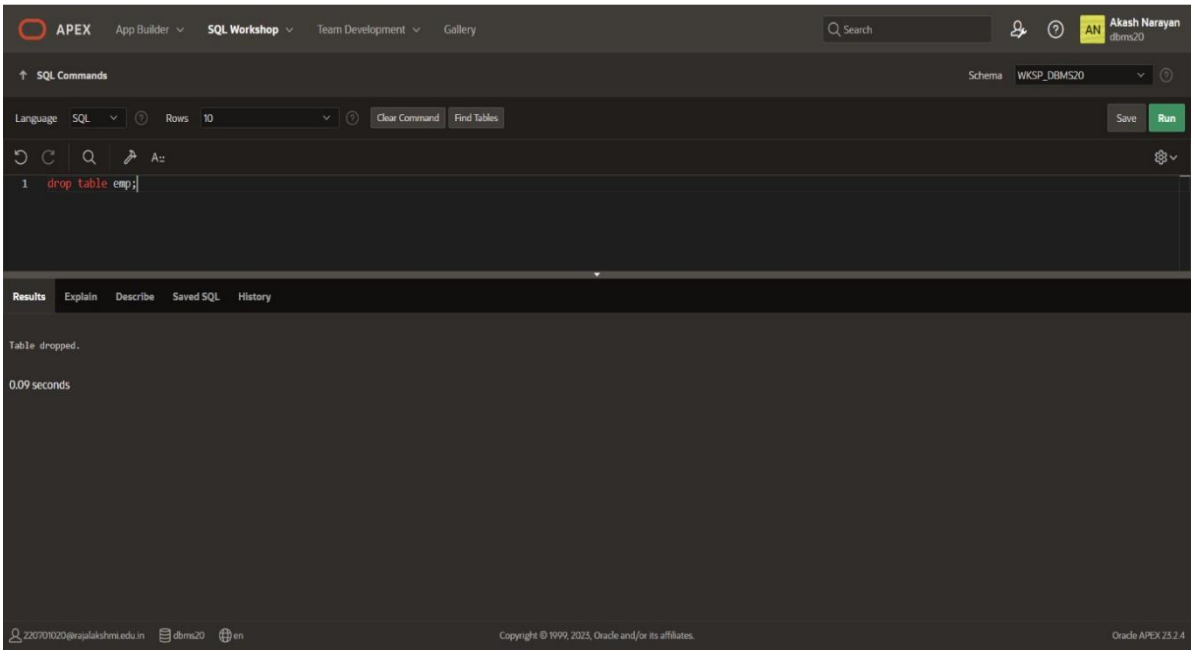
OUTPUT:



5. Drop table emp.

QUERY: Drop table emp;

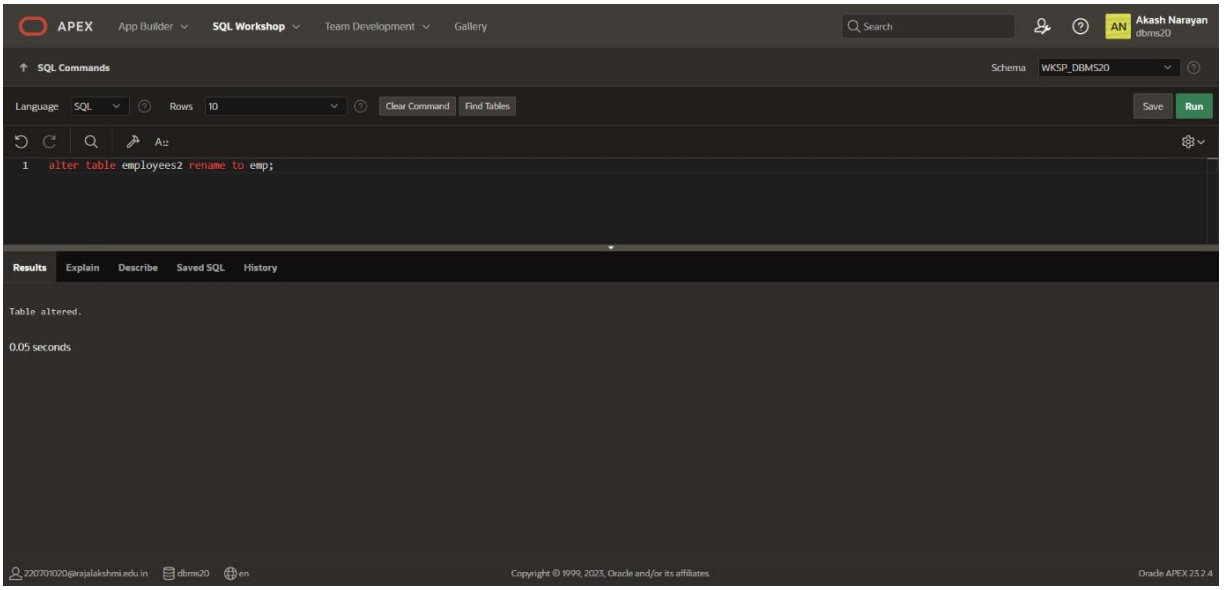
OUTPUT:



6.Rename the EMPLOYEES2 table as EMP.

QUERY: alter table employees2 rename to emp;

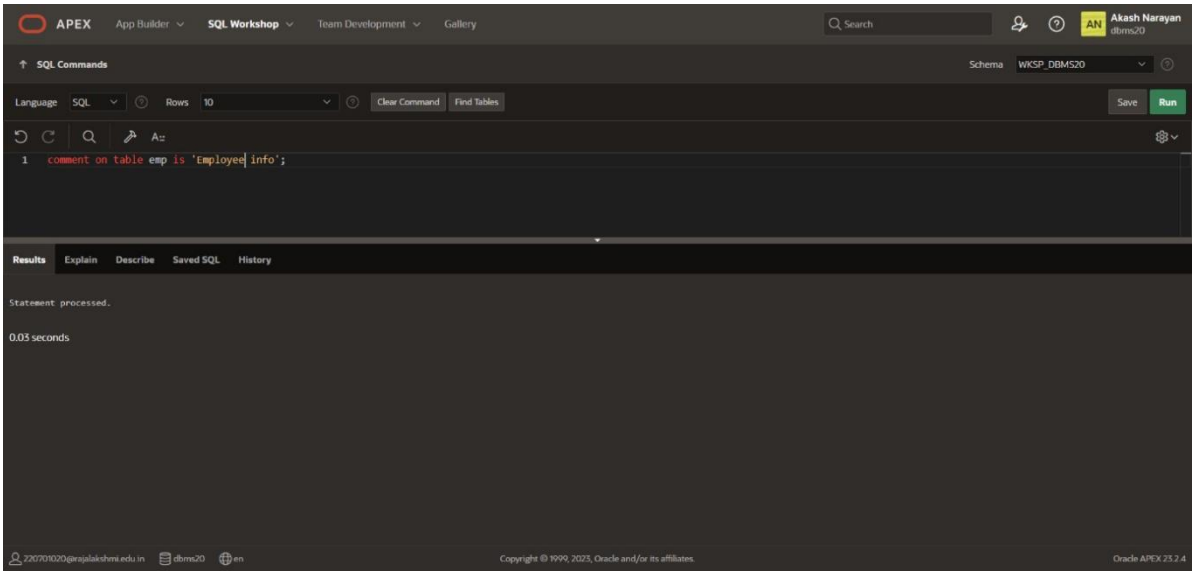
OUTPUT:



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

QUERY: comment on table dept is 'Employee info';

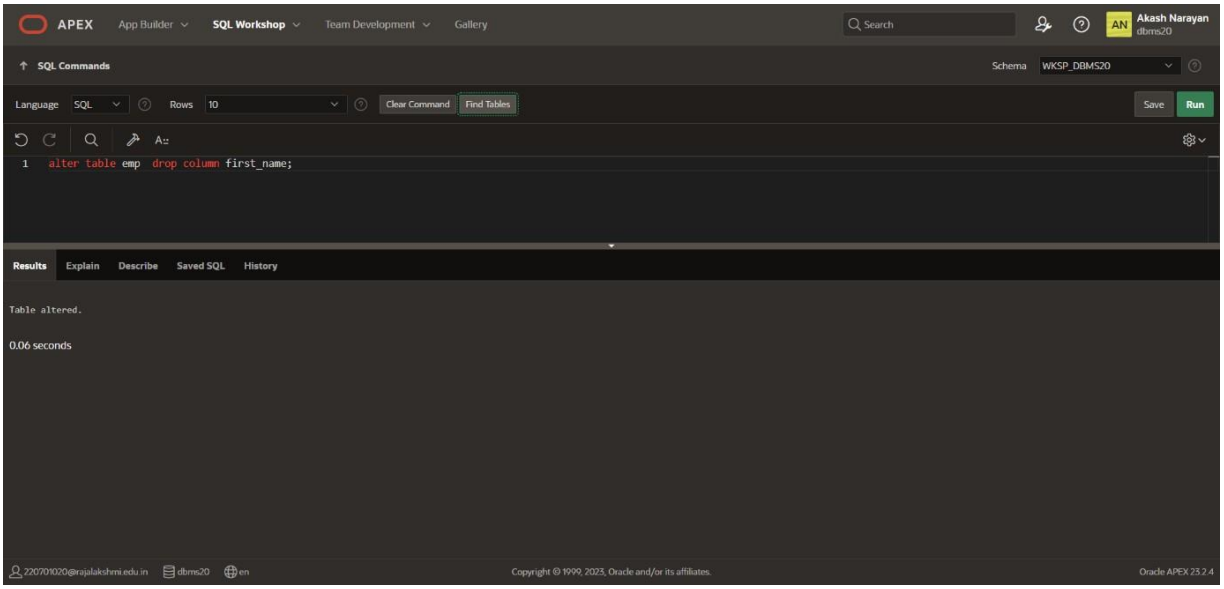
OUTPUT:



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	

RESULT:

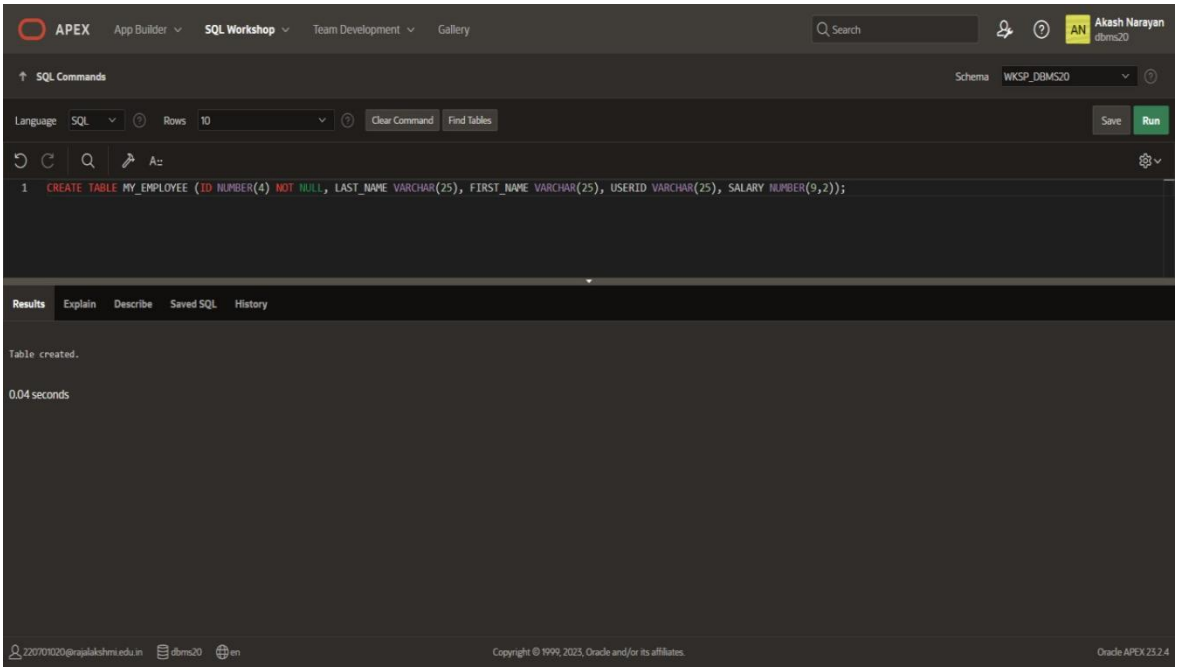
EXNO:2 MANIPULATING DATA DATE:

1.Create MY_EMPLOYEE table with the following structure

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

QUERY: Create table My_employee(ID number(4) not null, last_name varchar(25),first_name varchar(25),userid varchar(25),salary number(9,2));

OUTPUT:

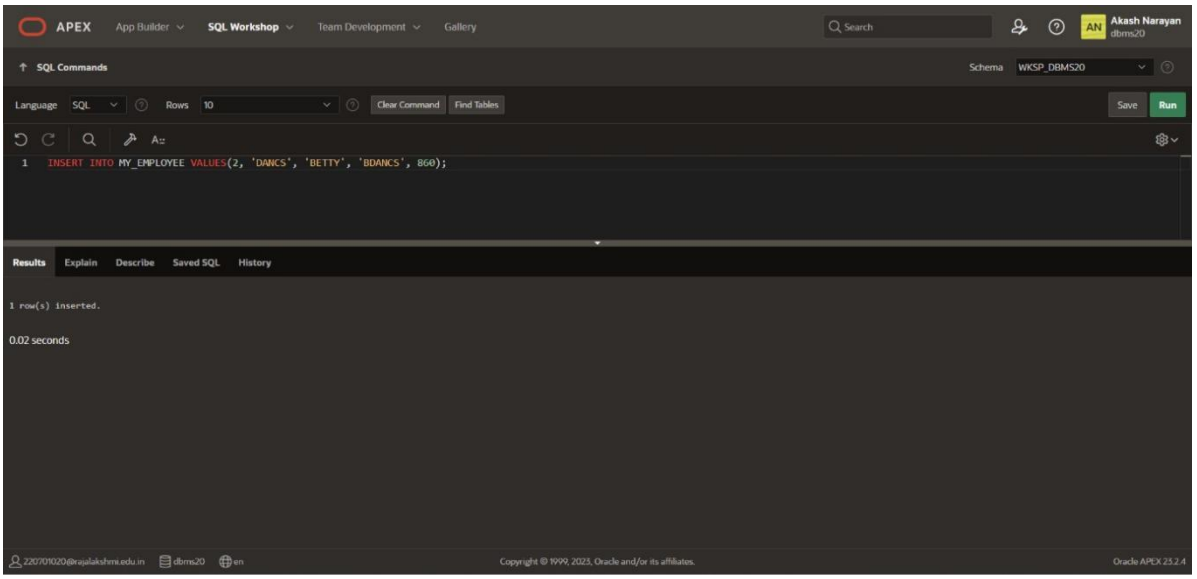


2.Add the first and second rows data to MY_EMPLOYEE table from thefollowing sample data.

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(1,'patel','ralph','rpatel',895); insert into my_employee(2,'dancs','betty','bdancs',860);

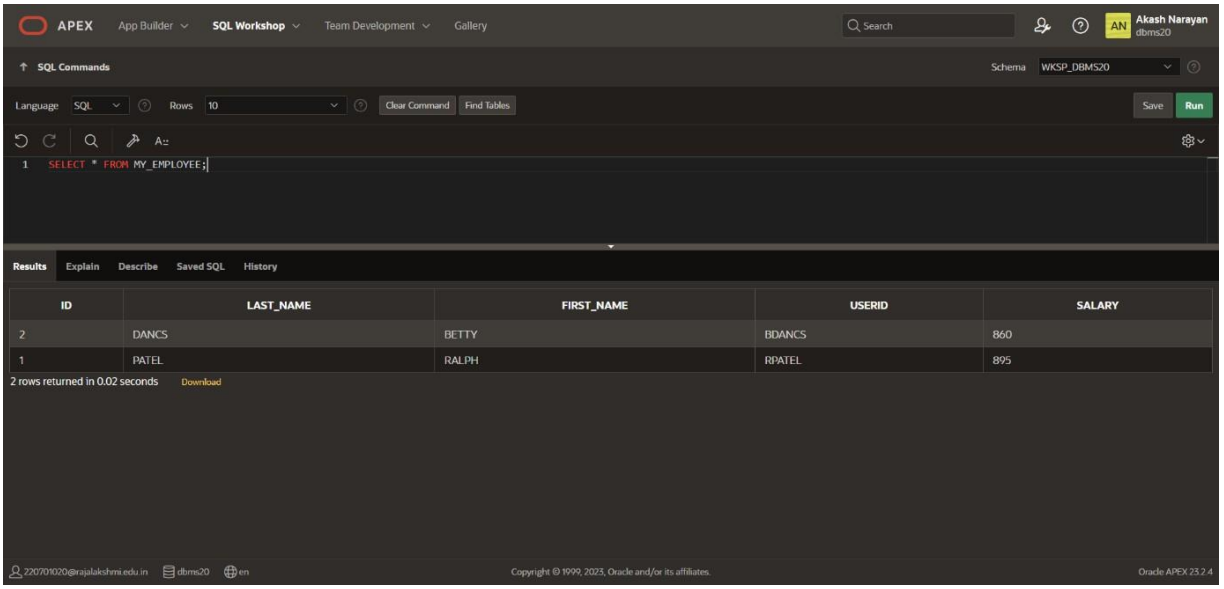
OUTPUT:



3.Display the table with values.

QUERY: select * from My_Employee;

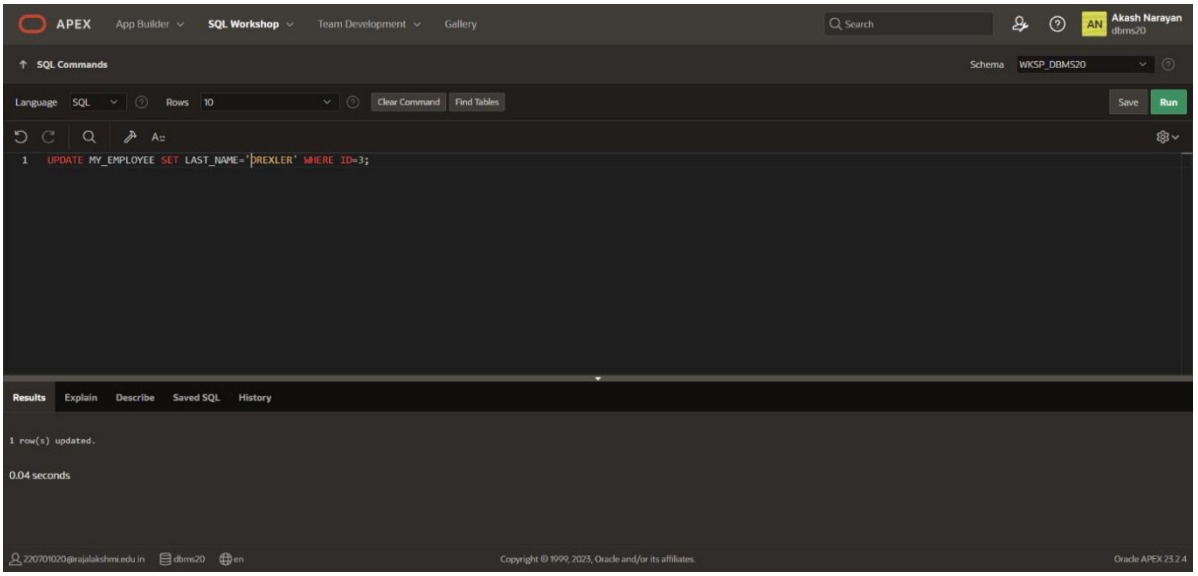
OUTPUT:



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

QUERY: update My_employee set lname=”drexler” where id=3;

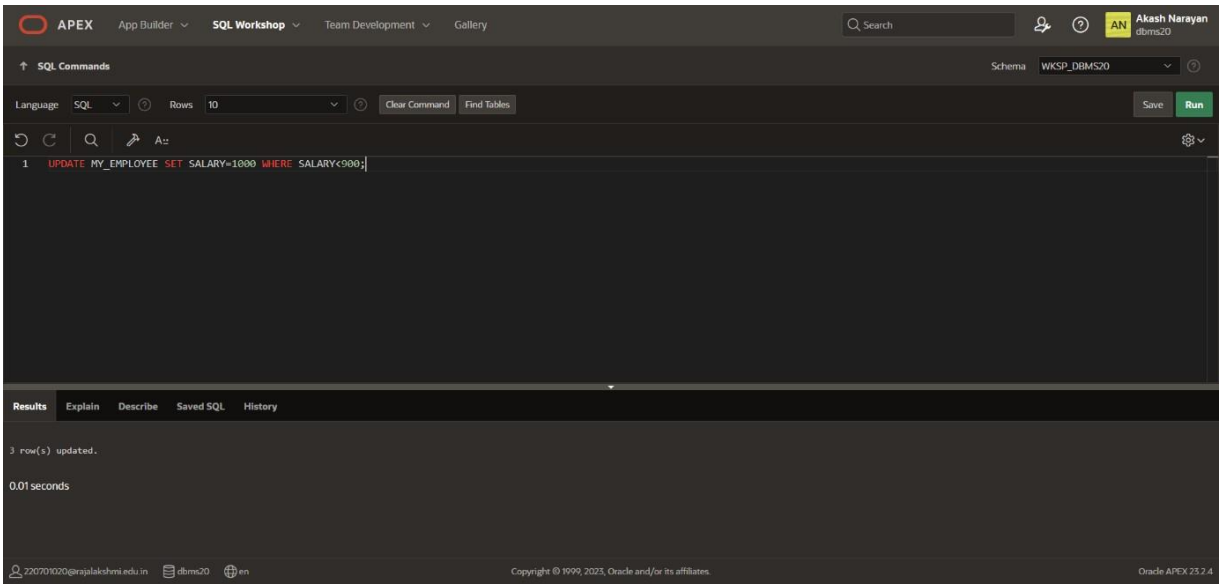
OUTPUT:



5.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;

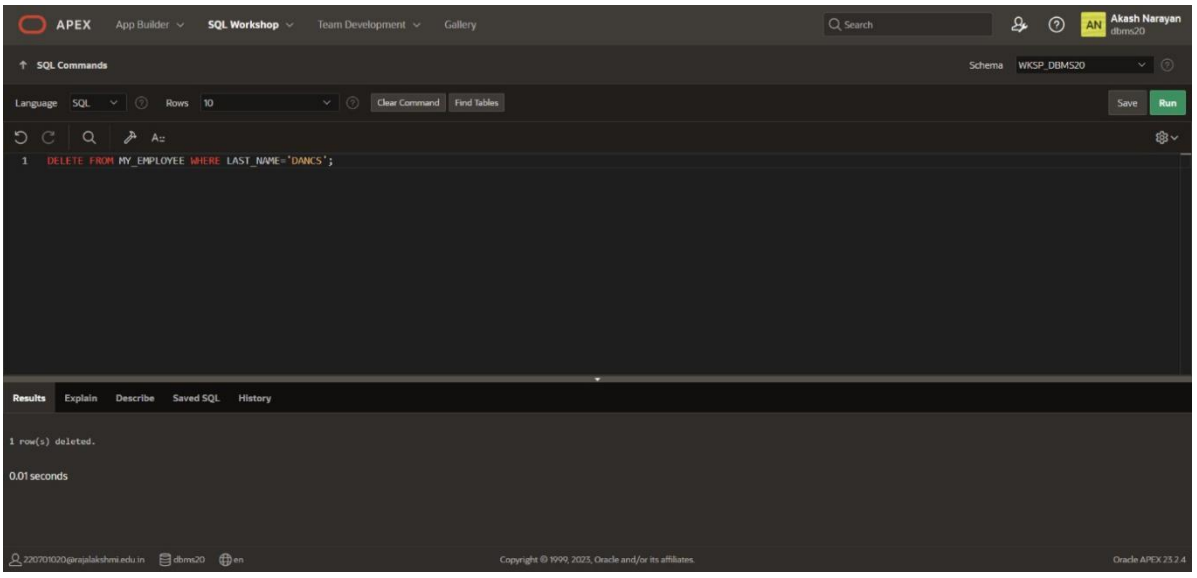
OUTPUT:



6.Delete Betty dancs from MY _EMPLOYEE table.

QUERY: delete from My_Employee where last_name='dancs';

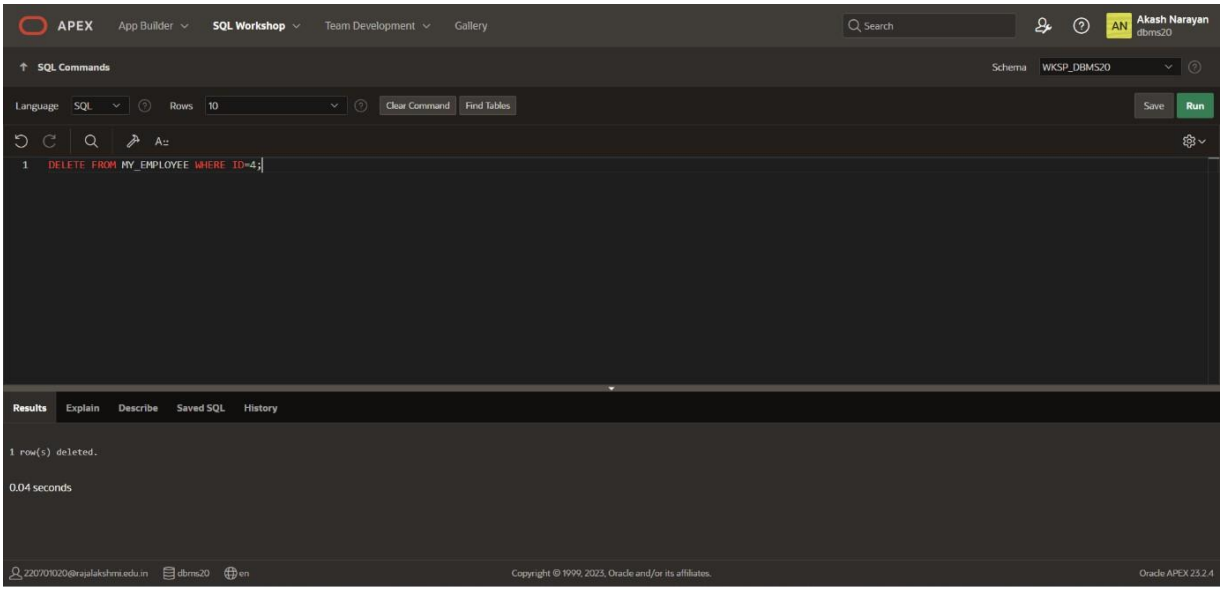
OUTPUT:



7.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	

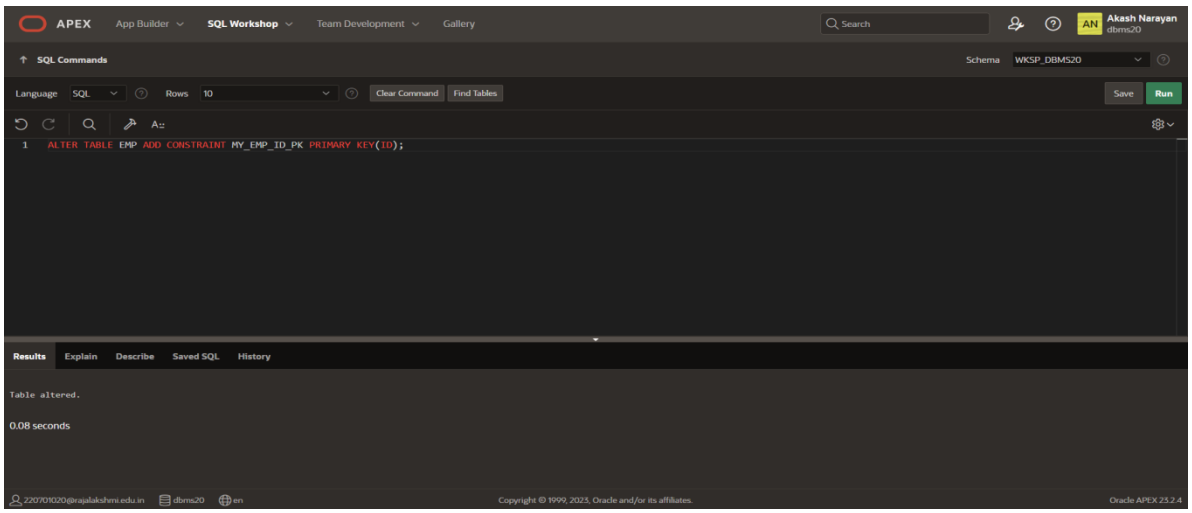
RESULT:

EXNO:3 **INCLUDING CONSTRAINTS** **DATE:**

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

QUERY: alter table emp add constraint my_emp_id_pk primary key(id);

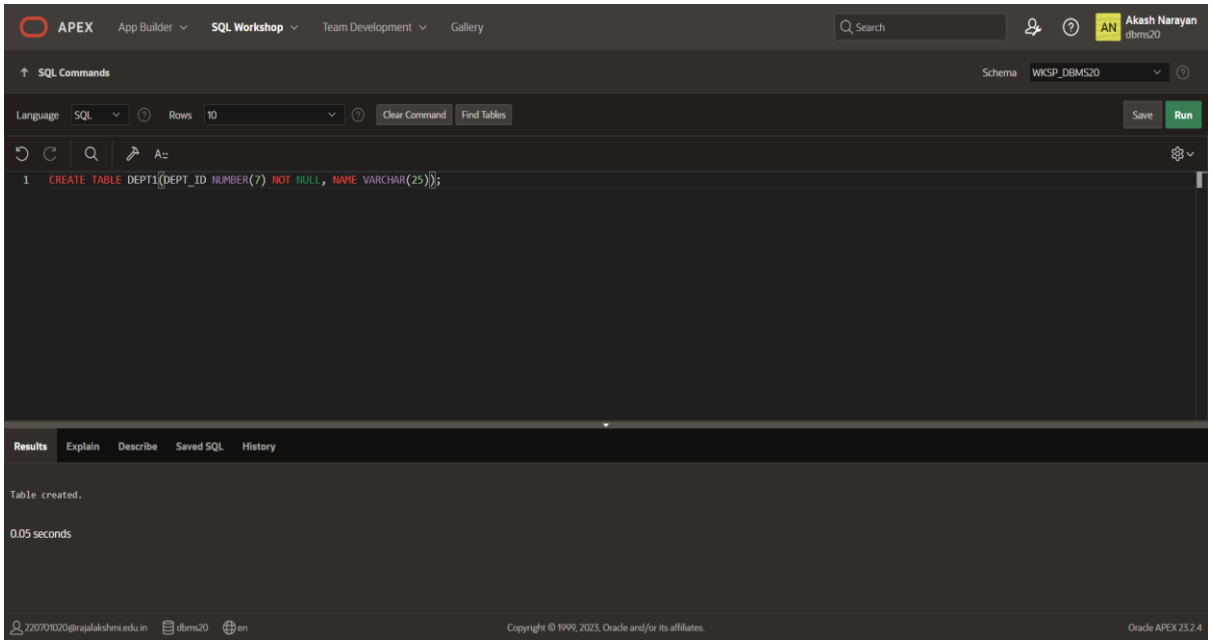
OUTPUT:



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

QUERY: create table dept1(dept_id number(7) not null, name varchar(25);

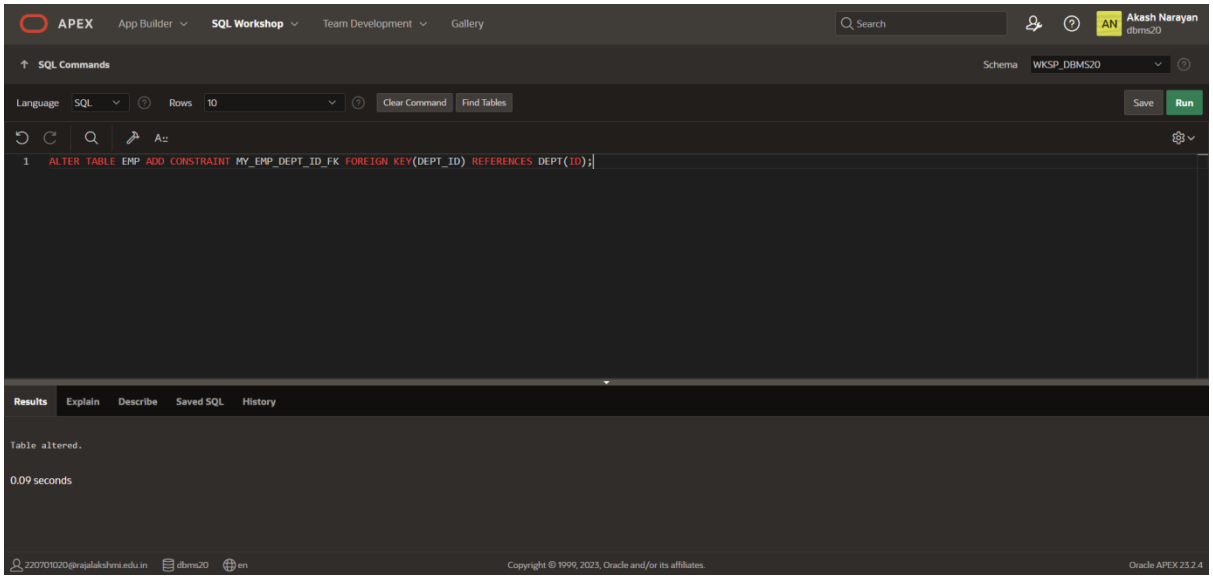
OUTPUT:



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 deparment. Name the constraint my_emp_dept_id_fk.

QUERY: alter table emp add constraint my_emp_dept_id_fk foreign key(dept_id) references dept(id);

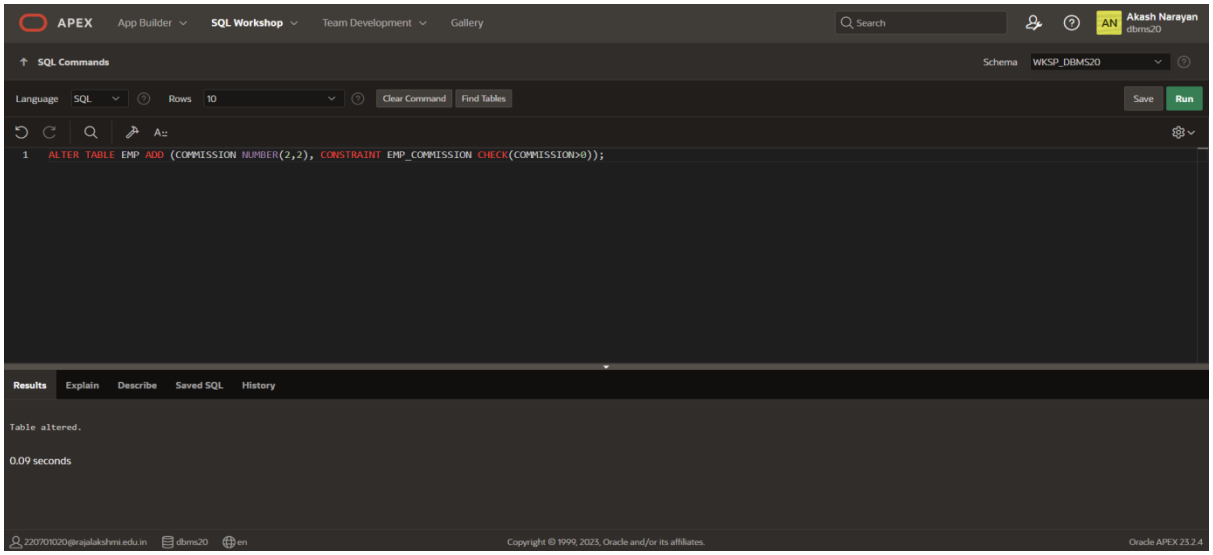
OUTPUT:



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 emp_commission check(commission>0));

OUTPUT:



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

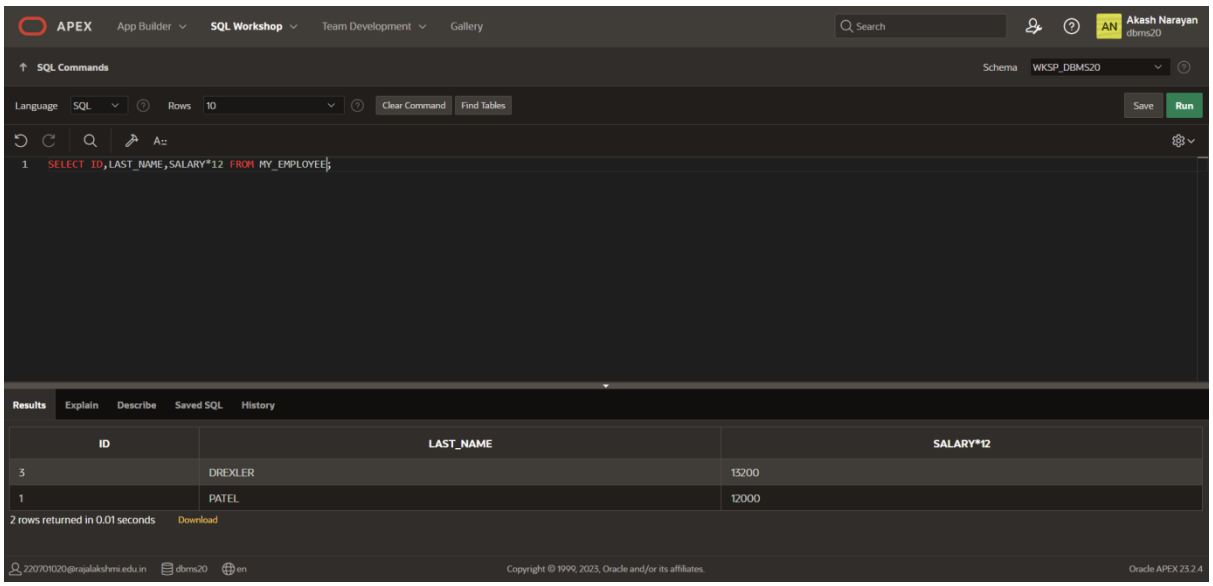
RESULT:

EXNO:4 WRITING BASIC SQL SELECT STATEMENTS DATE:

1.Identify the Errors
SELECT employee_id, last_name ,sal*12 ANNUAL SALARY FROM employees;

QUERY: select id,last_name,salary*12 from my_employee;

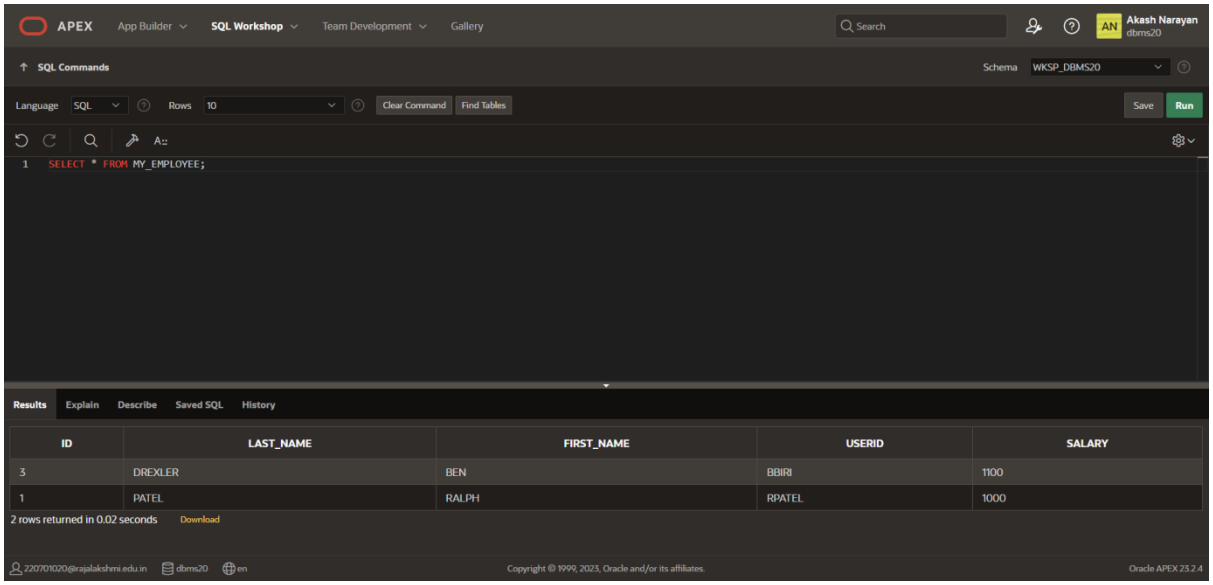
OUTPUT:



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

QUERY: select * from my_employee;

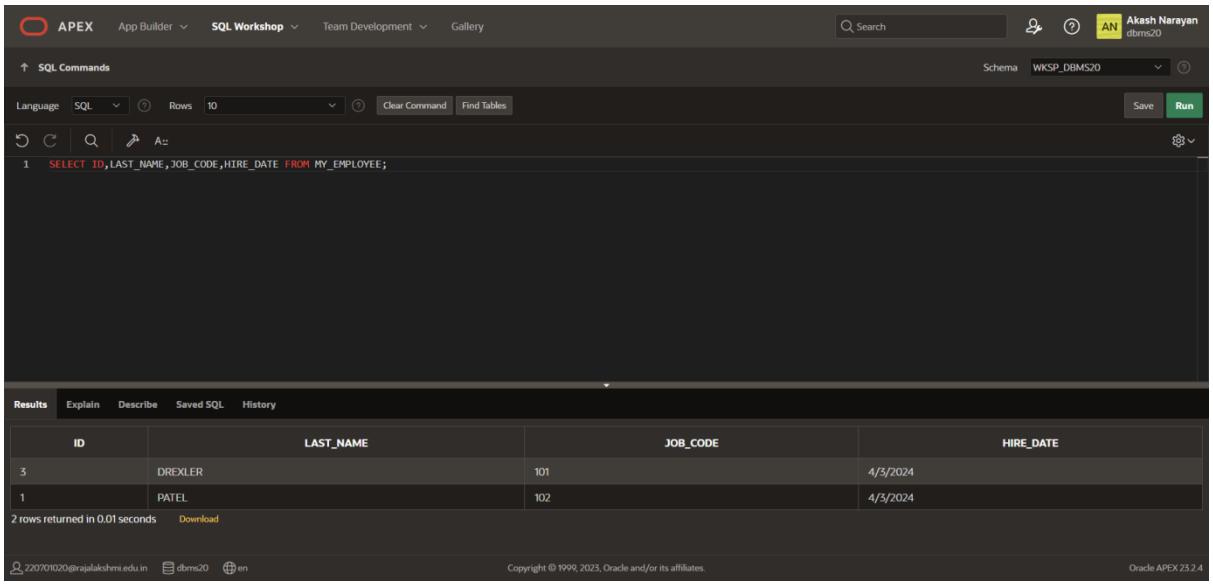
OUTPUT:



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 id,last_name,job_code,hire_date from my_employee;

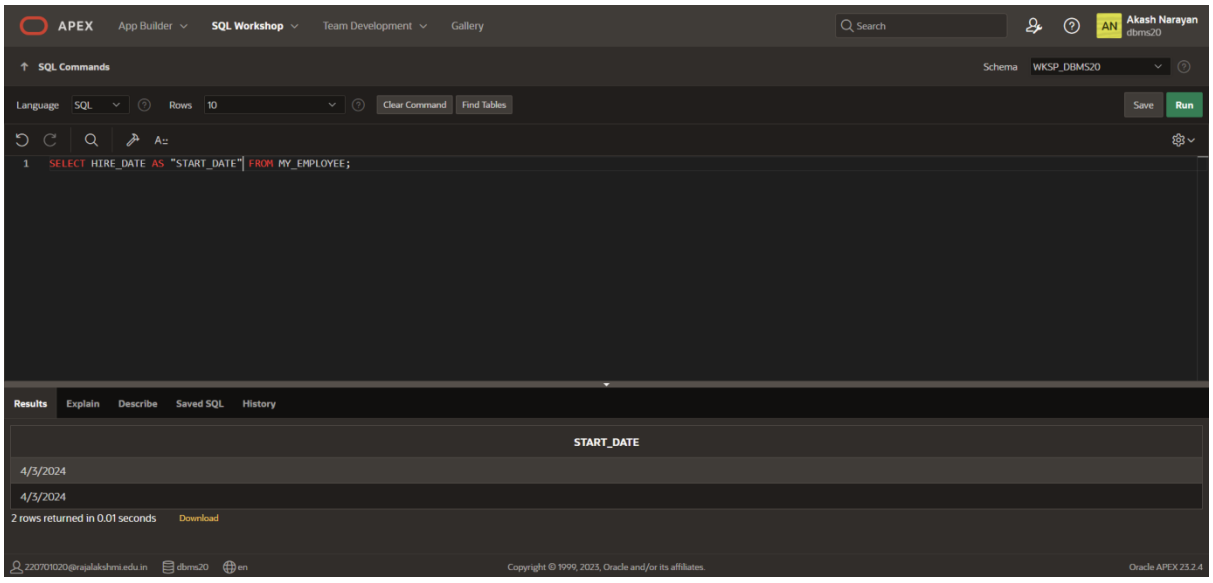
OUTPUT:



4. Provide an alias STARTDATE for the hire date.

QUERY: Select hire_date as “start_date” from my_employee;

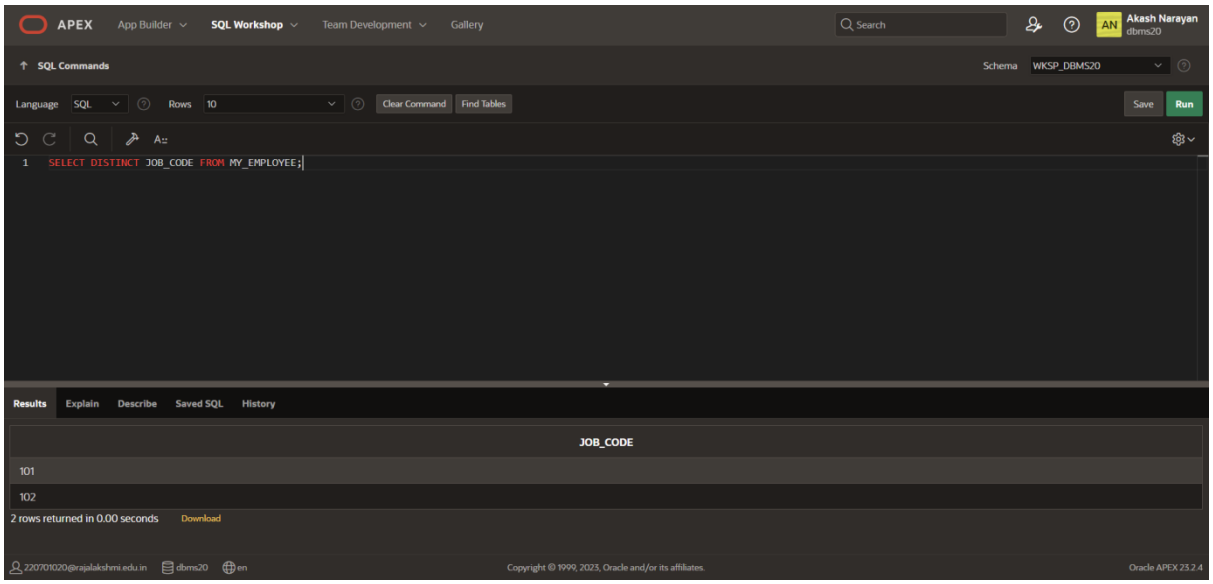
OUTPUT:



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

QUERY: select distinct job_code from my_employee;

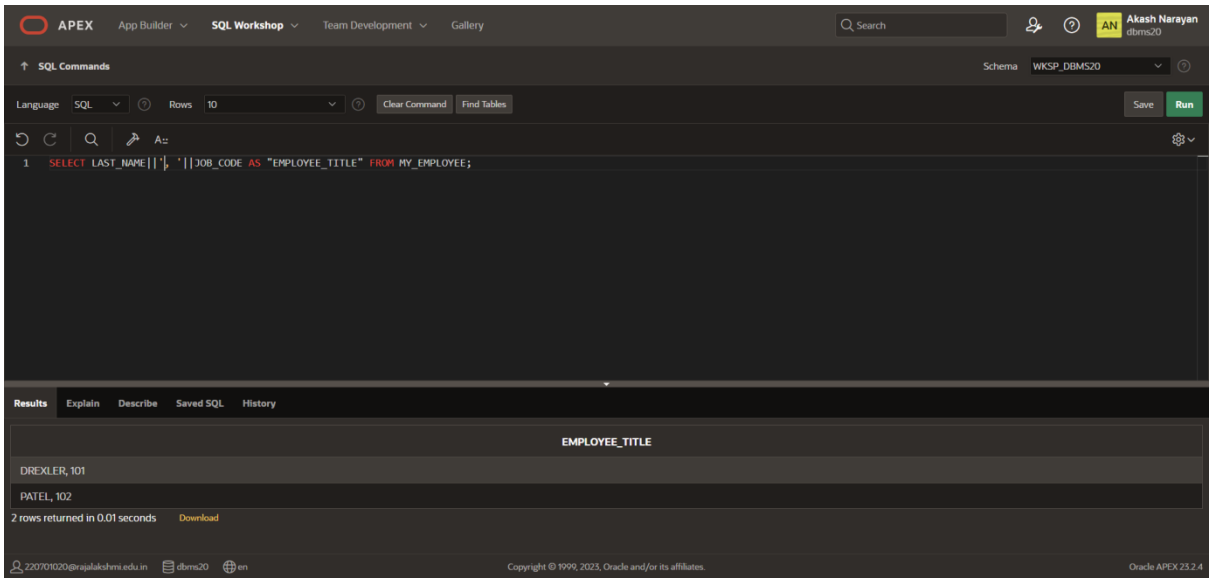
OUTPUT:



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_code as "employee_title" from my_employee;

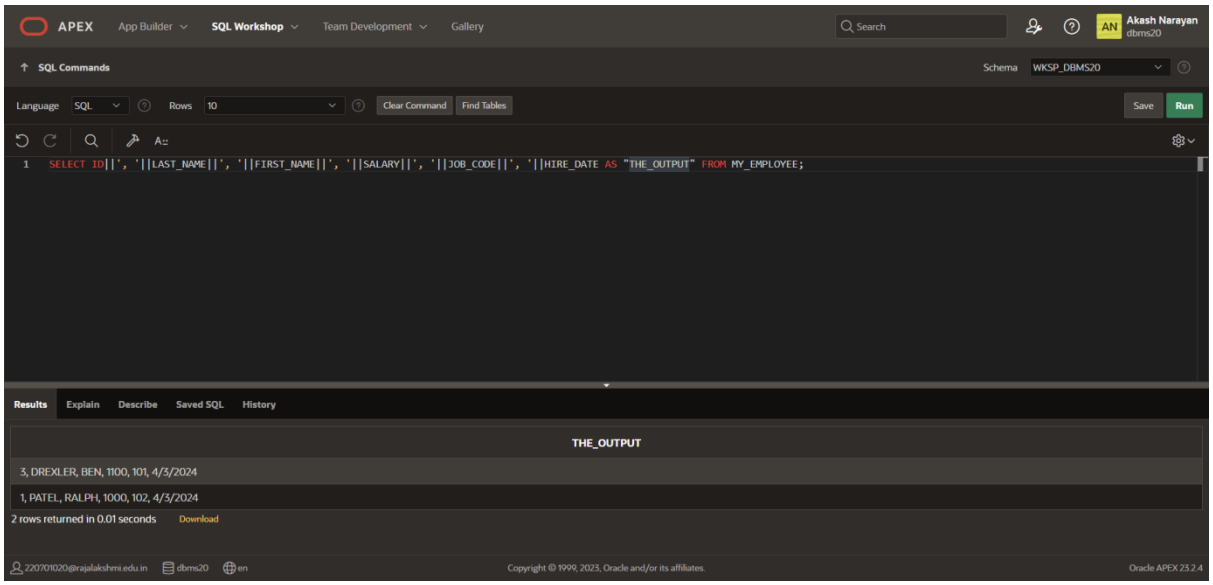
OUTPUT:



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 id||', '||last_name||', '||first_name||', '||salary||', '||job_code||', '||hire_date as "the_output" from my_employee;

OUTPUT:



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

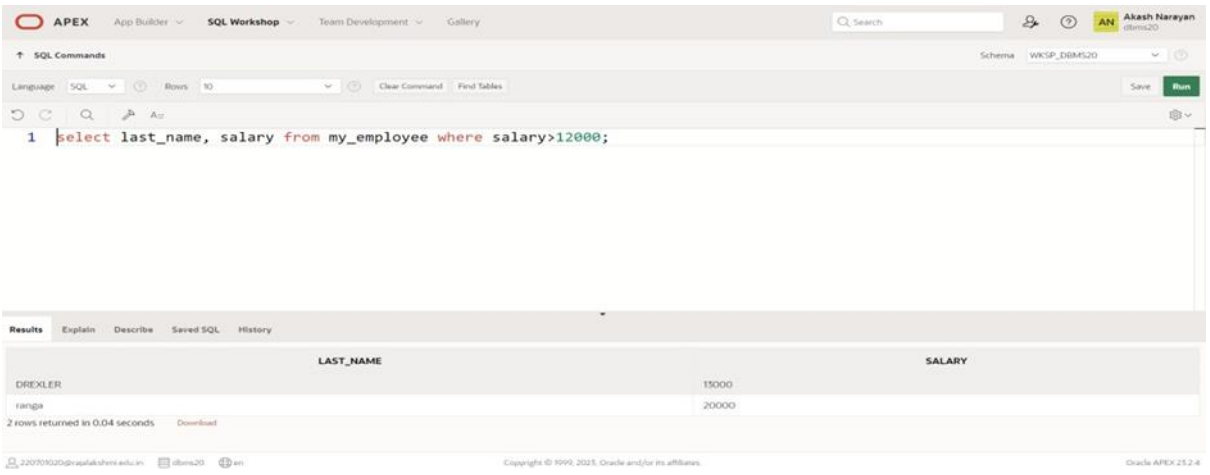
RESULT:

EXNO:5 **RESTRICTING AND SORTING DATA** **DATE:**

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

QUERY: select last_name, salary from My_employee where salary > 12000;

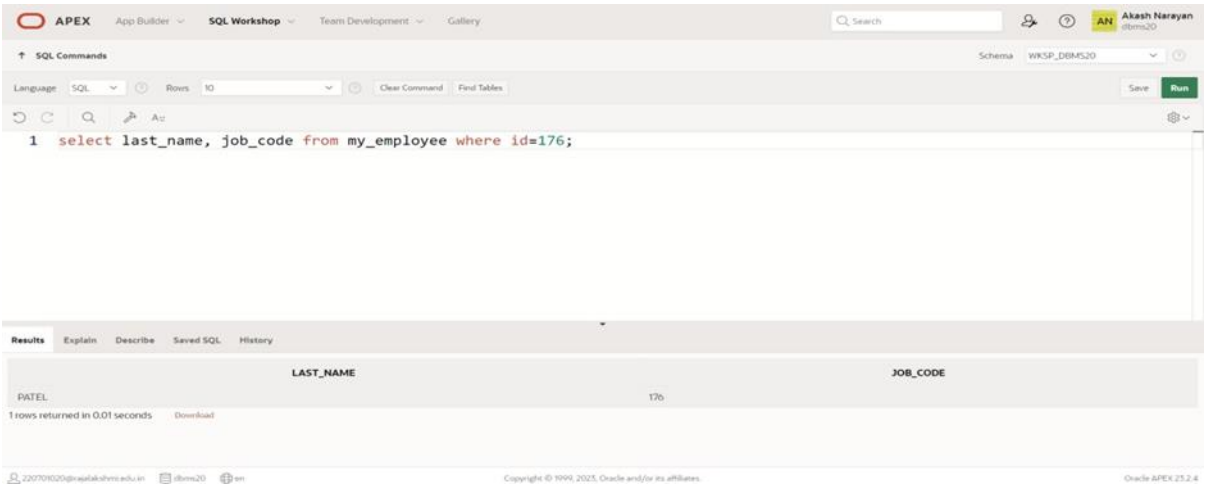
OUTPUT:



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

QUERY: select last_name, dept_id from My_employee where emp_id = 176;

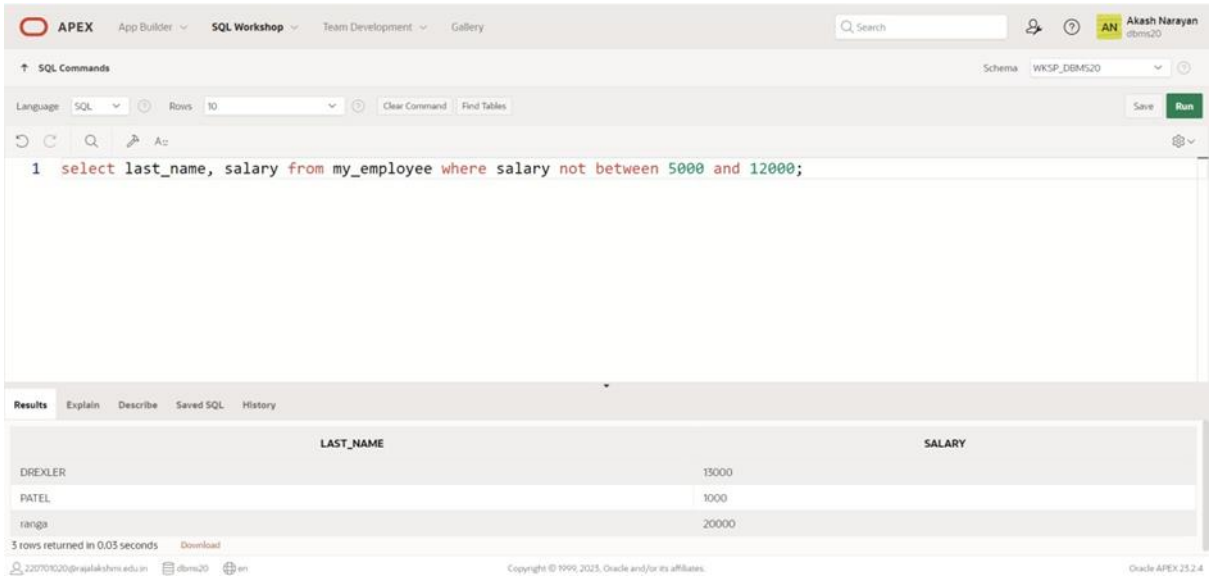
OUTPUT:



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

QUERY: select last_name, salary from My_employee where salary>12000 or salary<5000;

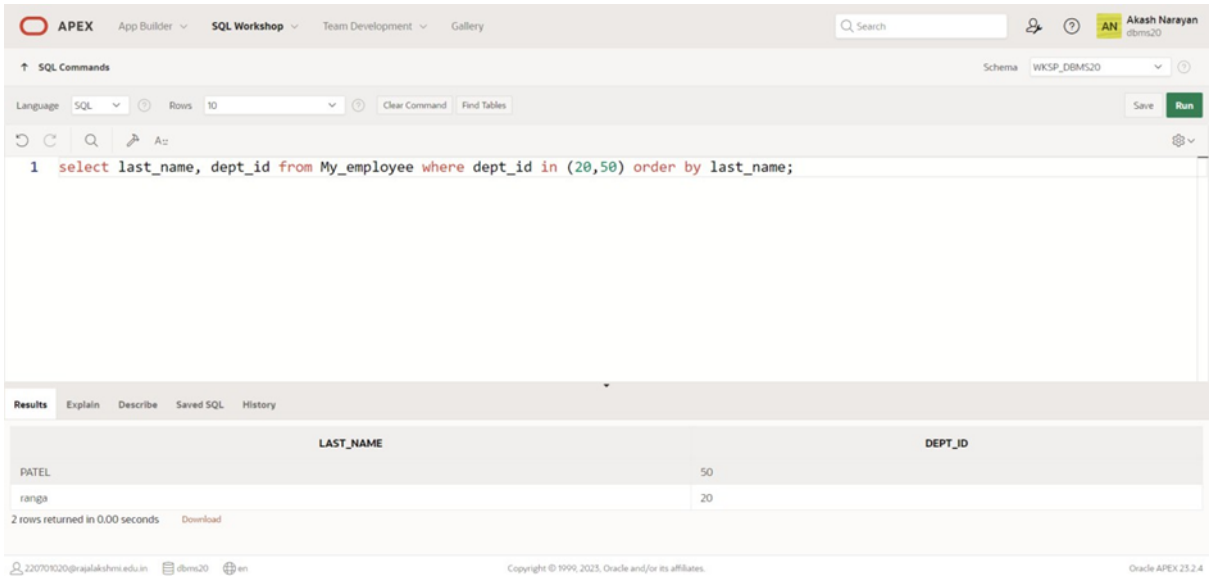
OUTPUT:



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

QUERY: select last_name, dept_id from My_employee where dept_id in (20,50) order by last_name;

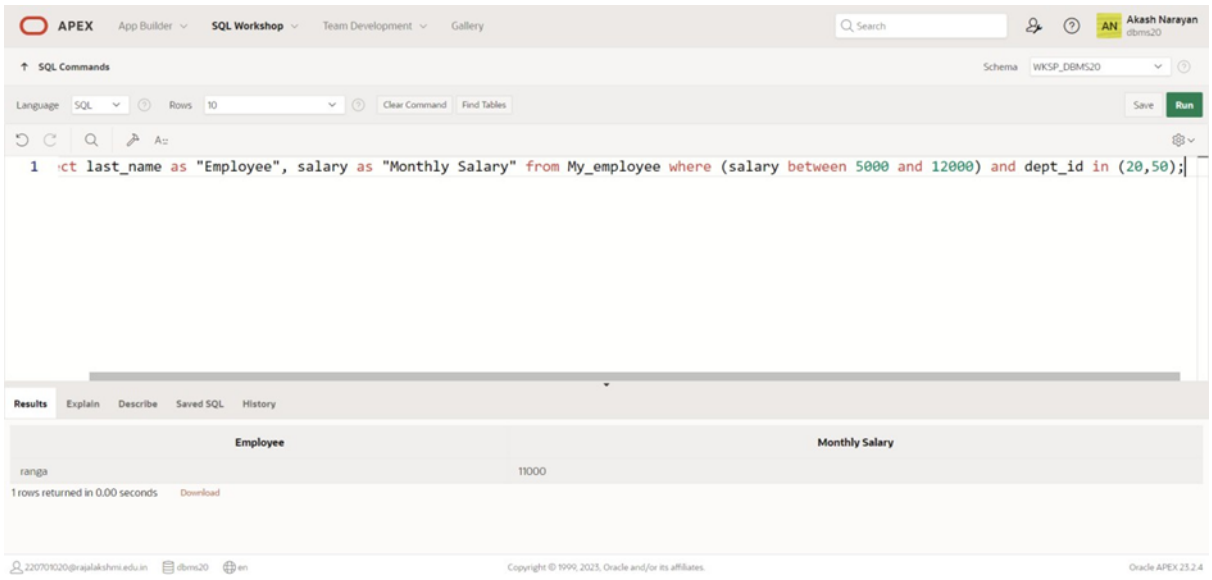
OUTPUT:



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

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

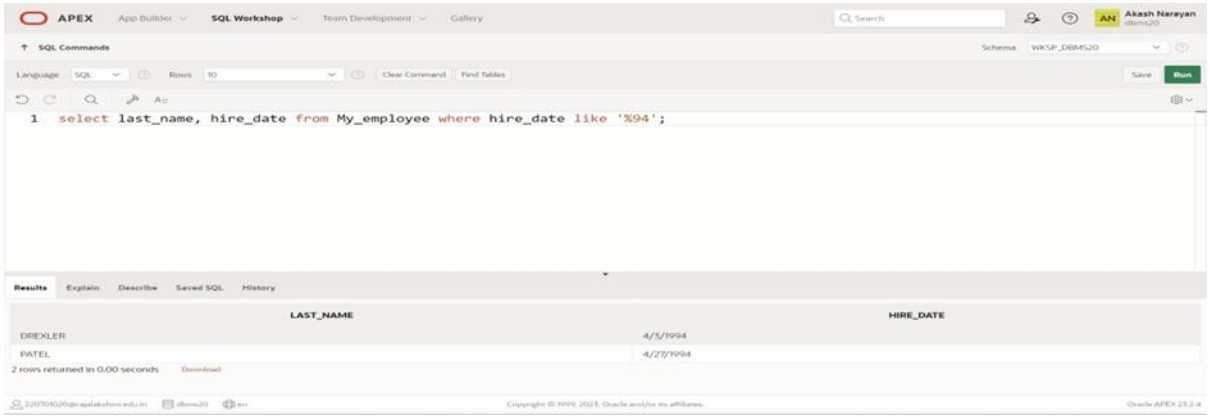
OUTPUT:



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

QUERY: select last_name, hire_date from My_employee where hire_date like ‘%94’;

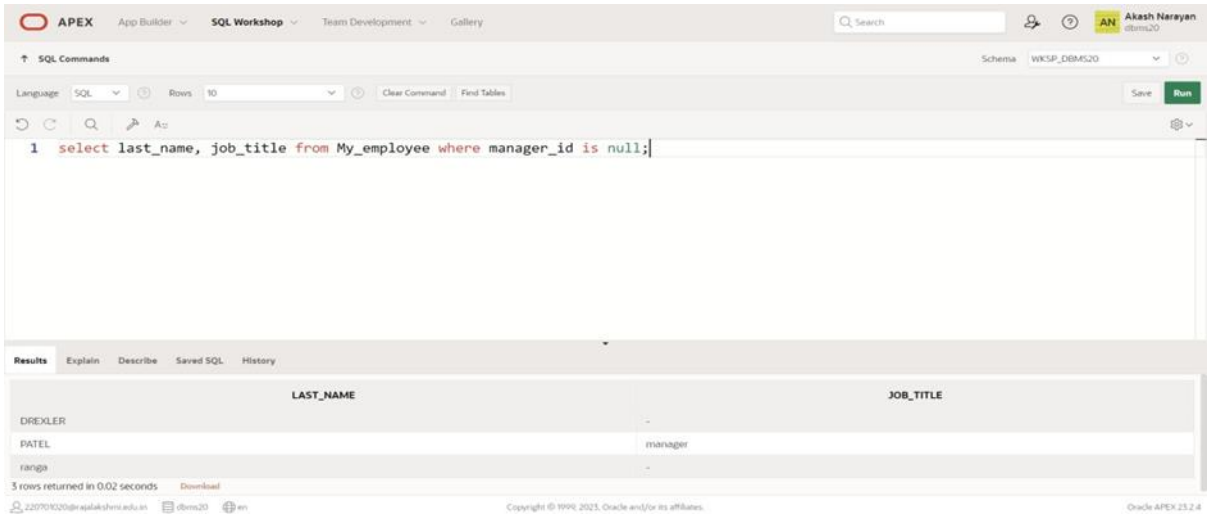
OUTPUT:



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

QUERY: select last_name, job_title from My_employee where manager_id is null;

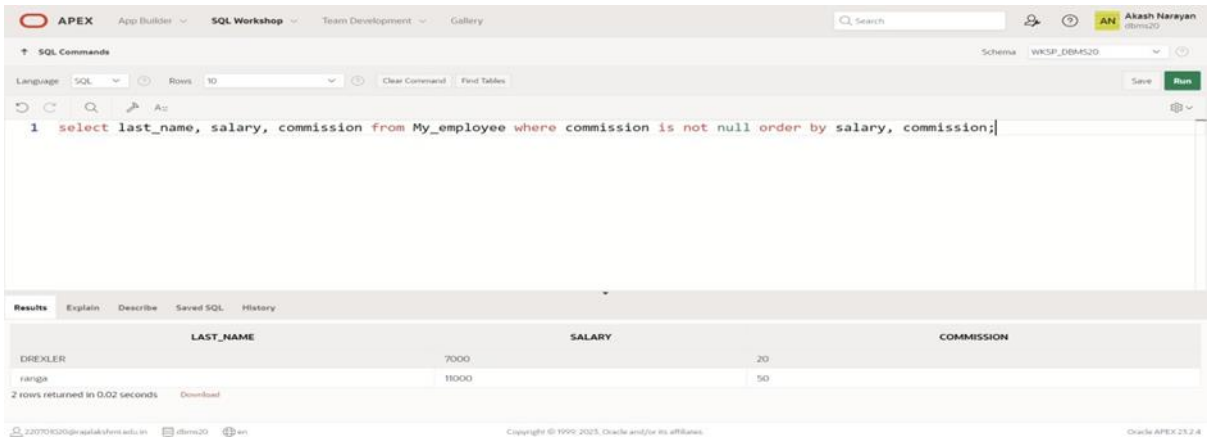
OUTPUT:



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

QUERY: select last_name, salary, commission from My_employee where commission is not null order by salary, commission;

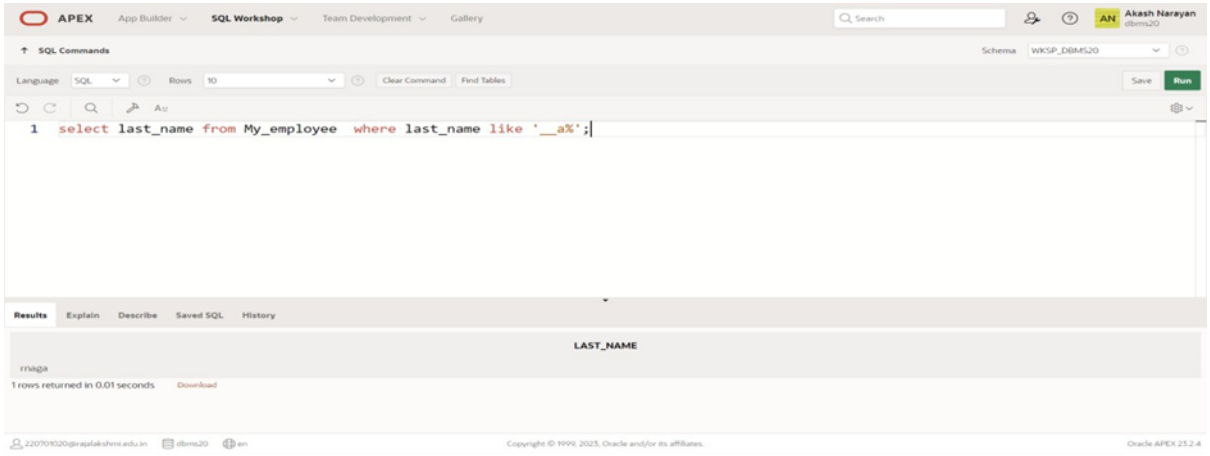
OUTPUT:



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

QUERY: select last_name from My_employee where last_name like ‘ a%’;

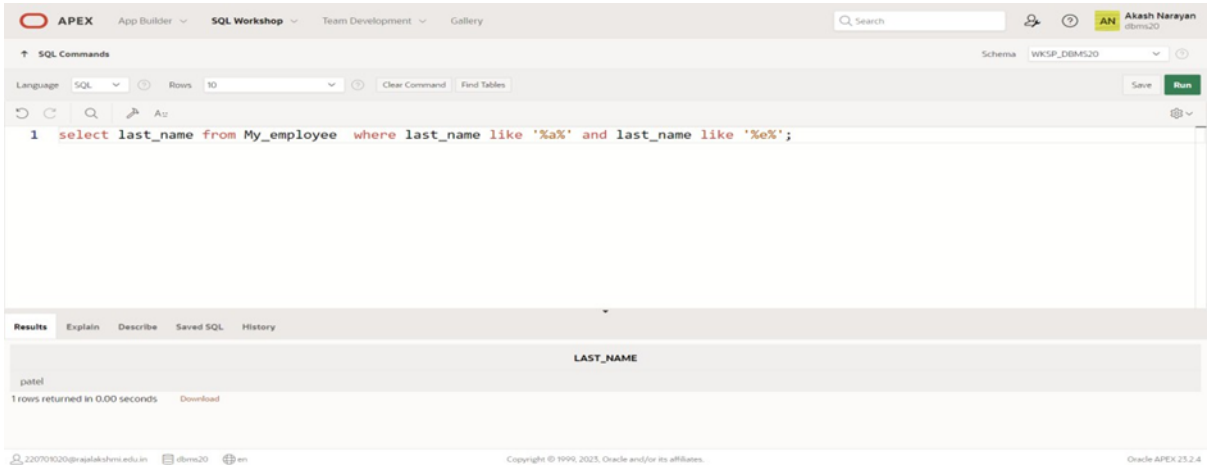
OUTPUT:



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

QUERY: select last_name from My_employee where last_name like ‘%a%’ and last_name like ‘%e%’;

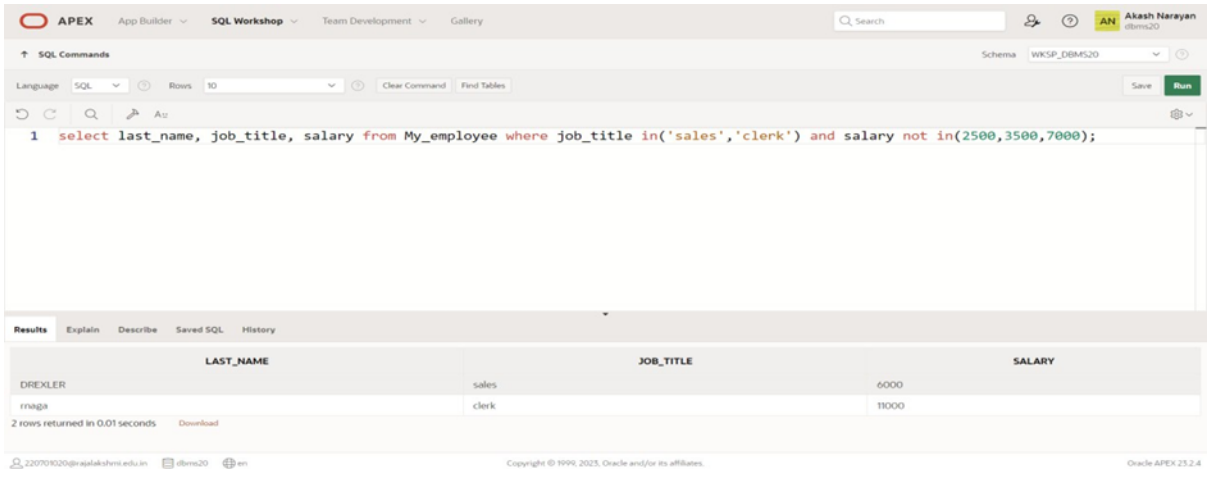
OUTPUT:



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

QUERY: select last_name, job_title, salary from My_employee where job_title in('sales','clerk') and salary not in(2500,3500,7000);

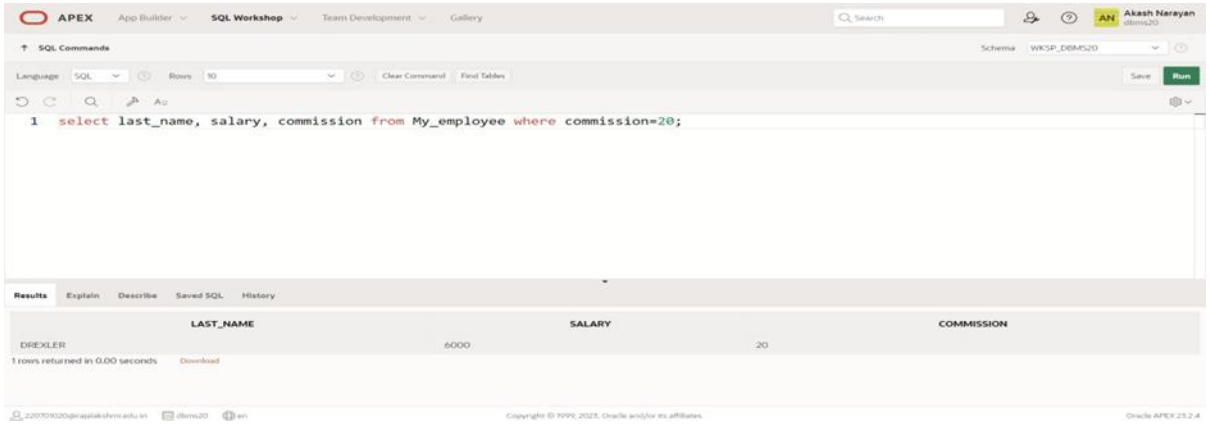
OUTPUT:



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

QUERY: select last_name, salary, commission from My_employee where commission=20;

OUTPUT:



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

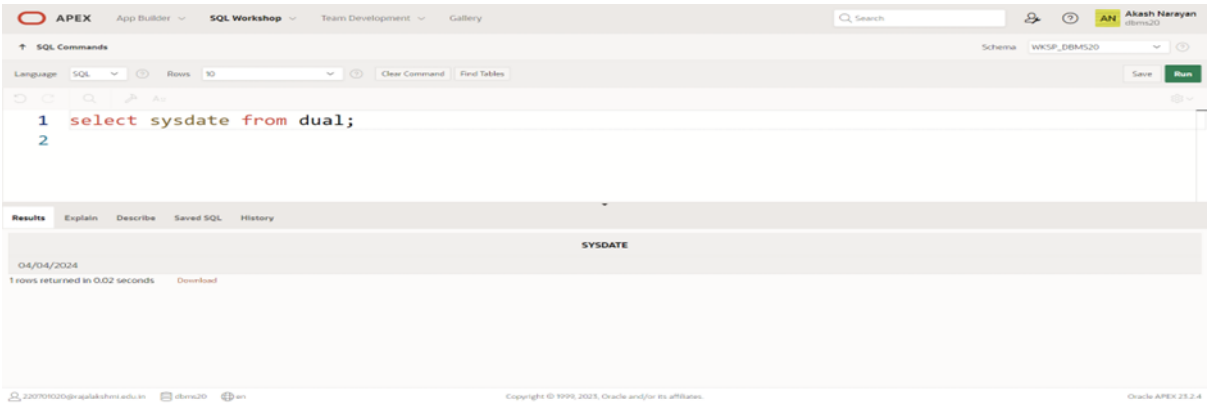
RESULT:

EXNO:6 SINGLE ROW FUNCTIONS DATE:

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

QUERY: select sysdate as “DATE” from dual;

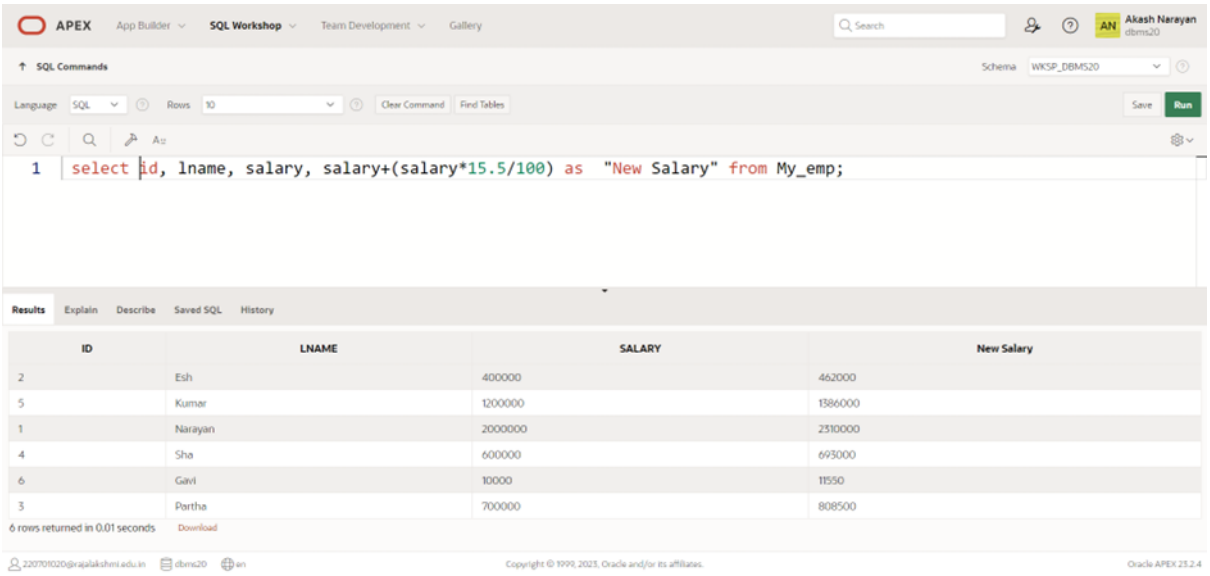
OUTPUT:



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

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

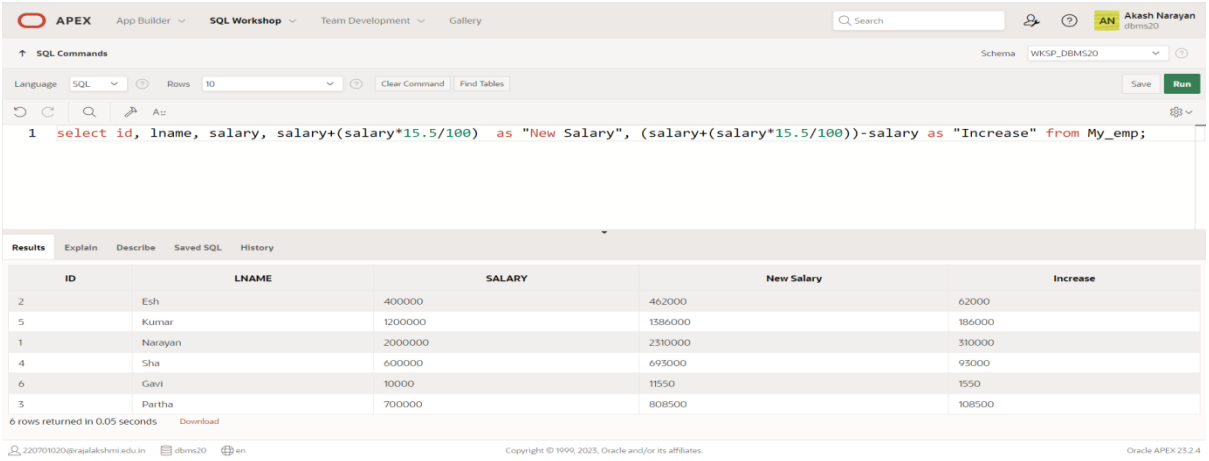
OUTPUT:



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

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

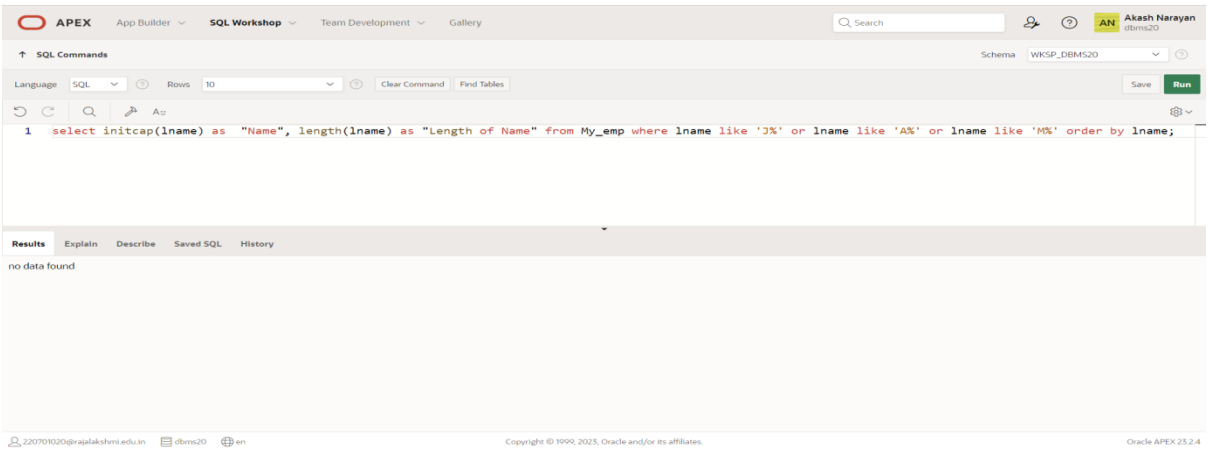
OUTPUT:



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

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

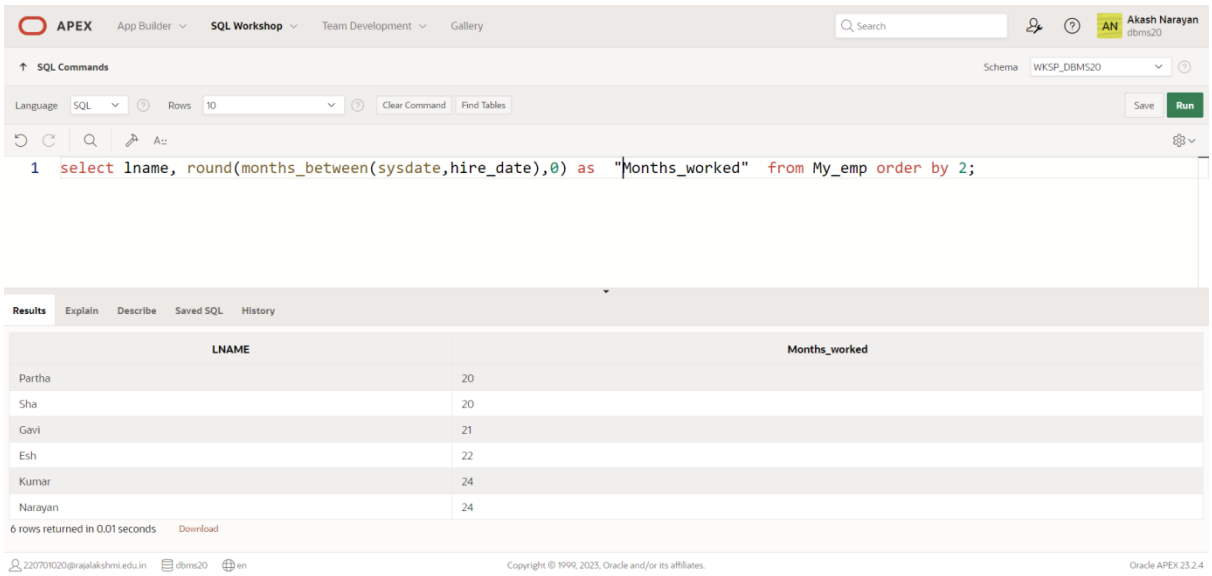
OUTPUT:



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

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

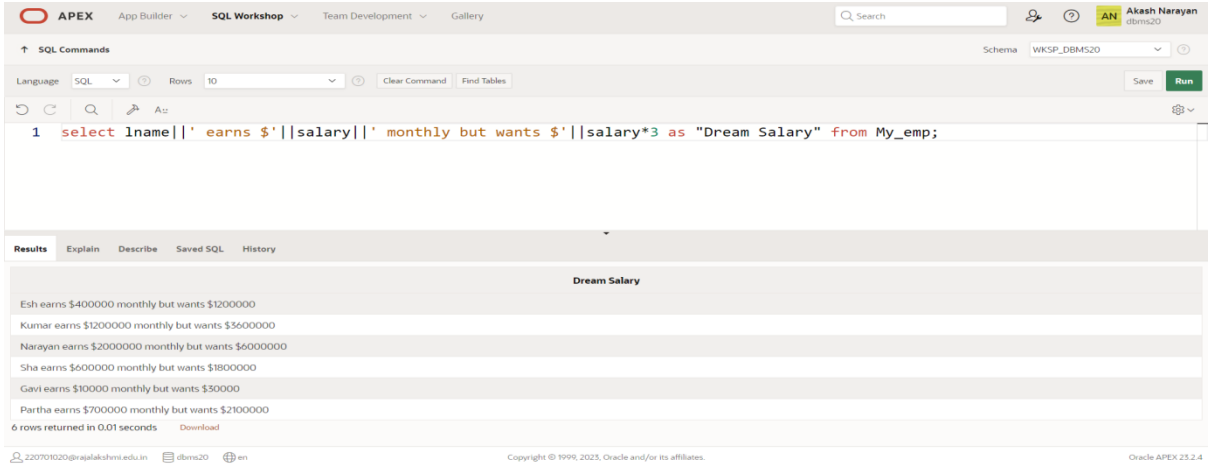
OUTPUT:



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

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

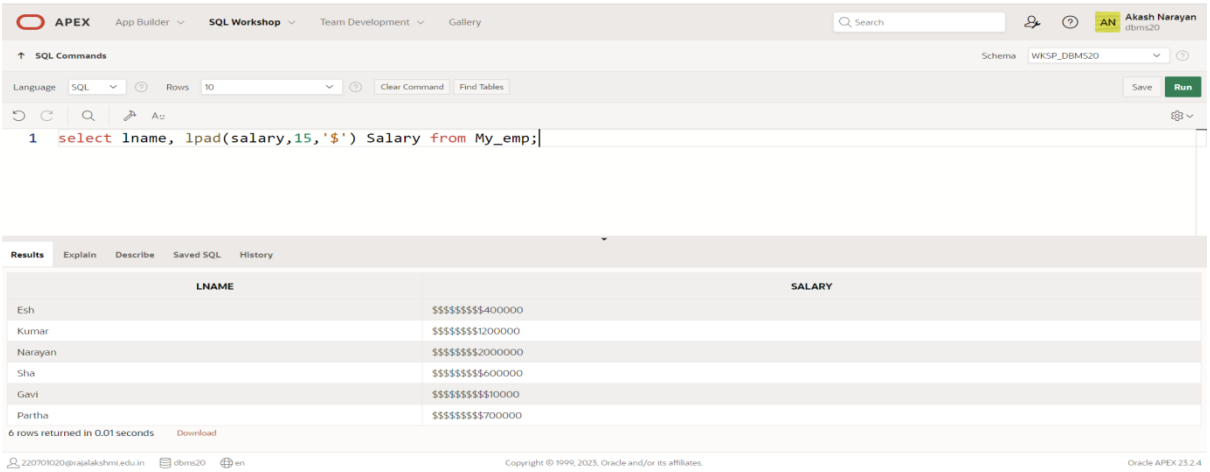
OUTPUT:



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

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

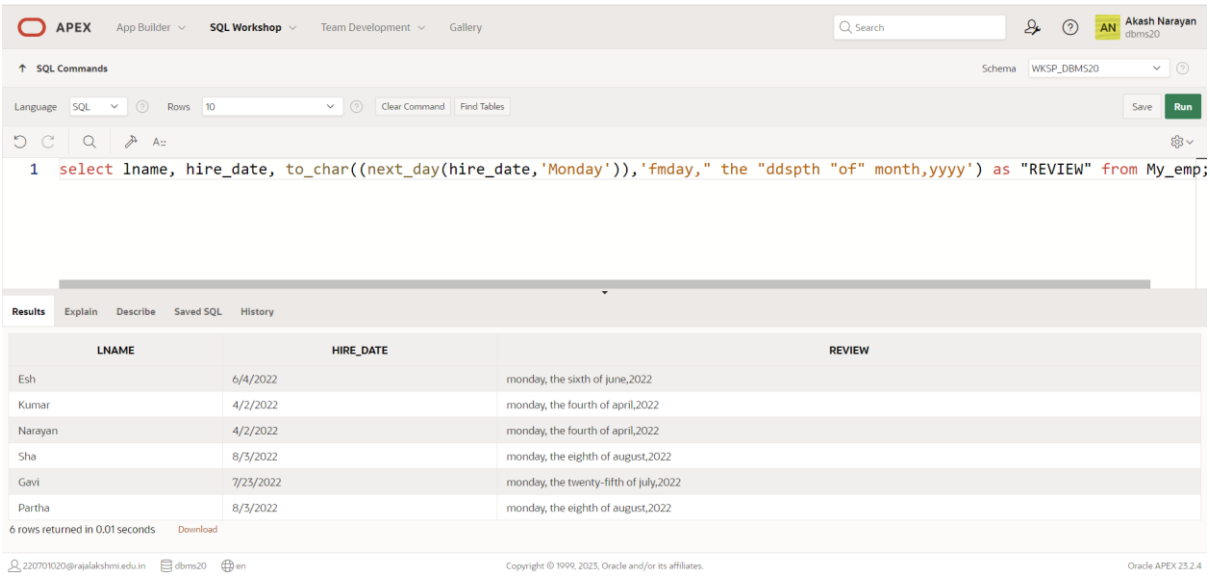
OUTPUT:



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

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

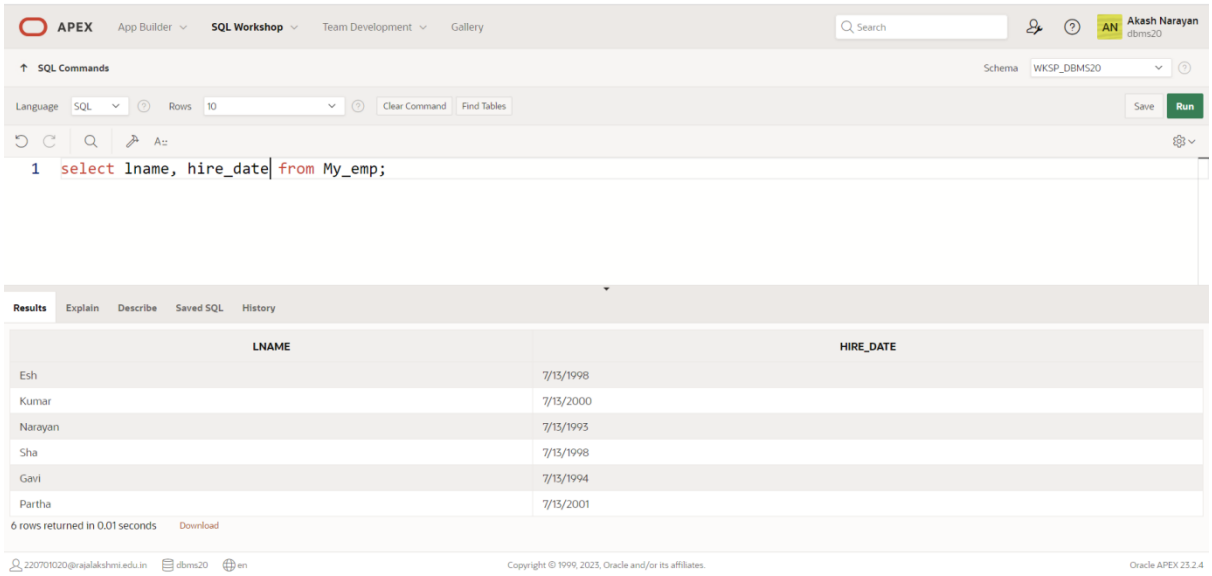
OUTPUT:



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

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

OUTPUT:



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

RESULT:

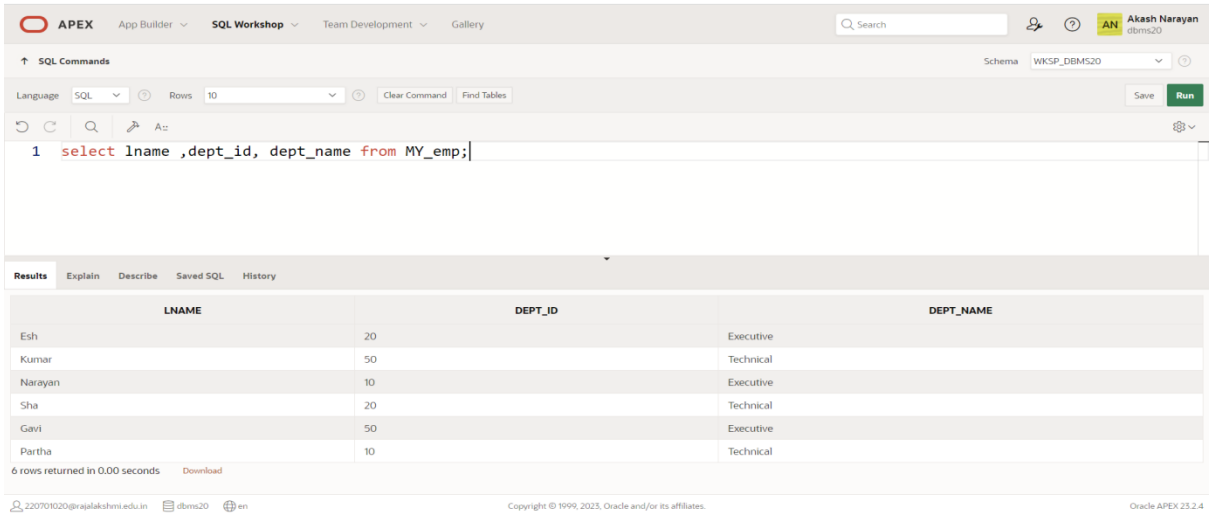
DISPLAYING DATA FROM MULTIPLE TABLES

EXNO:7 **DATE:**

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

QUERY: select lname ,dept_id, dept_name from MY_emp;

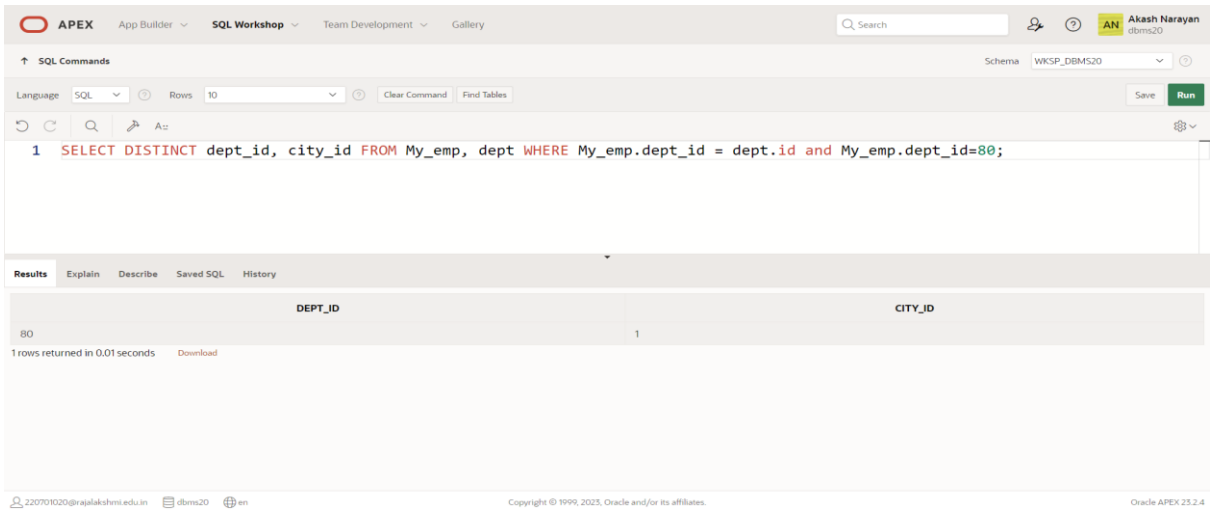
OUTPUT:



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

QUERY: SELECT DISTINCT job_id, loc_id FROM My_emp, dept WHERE My_emp.dept_id = dept.dept_id and My_emp.d_id=80;

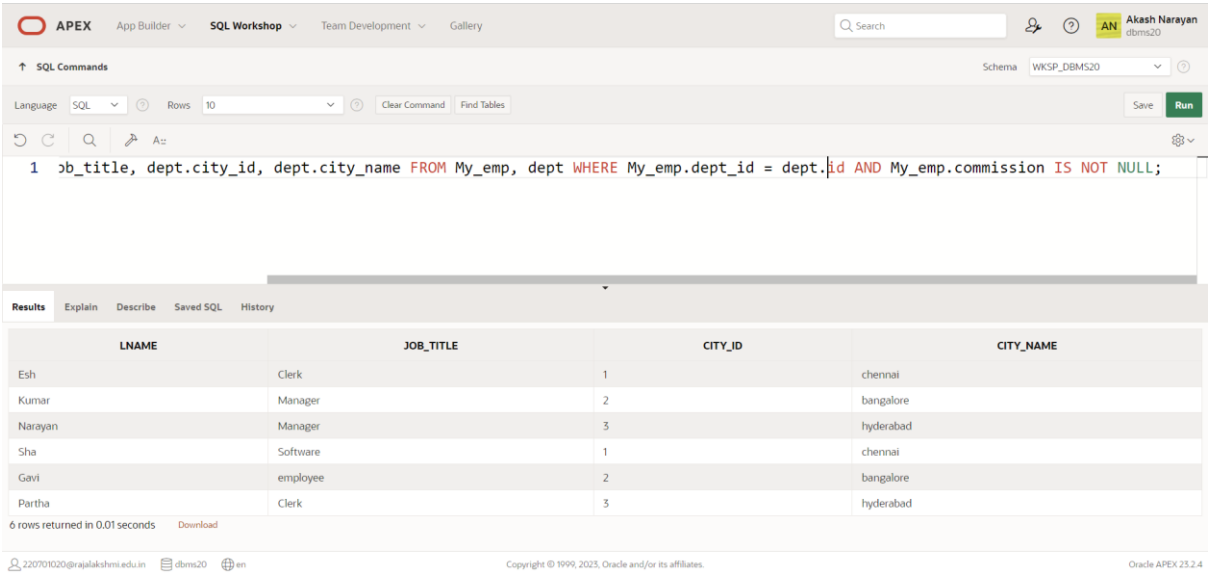
OUTPUT:



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

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

OUTPUT:



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

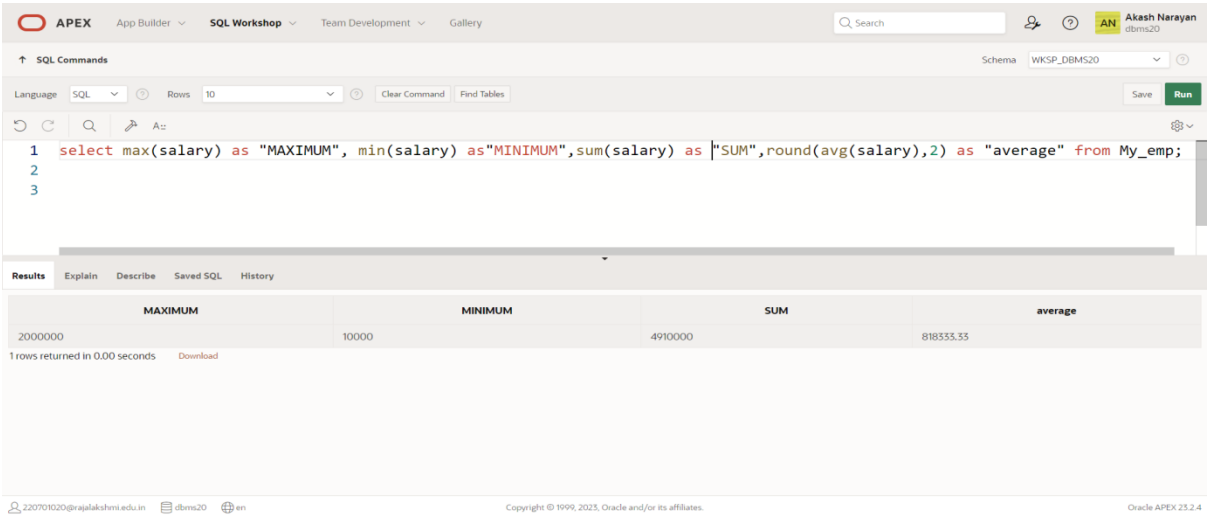
RESULT:

EXNO:8 AGGREGATING DATA USING GROUP FUNCTIONS DATE:

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

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

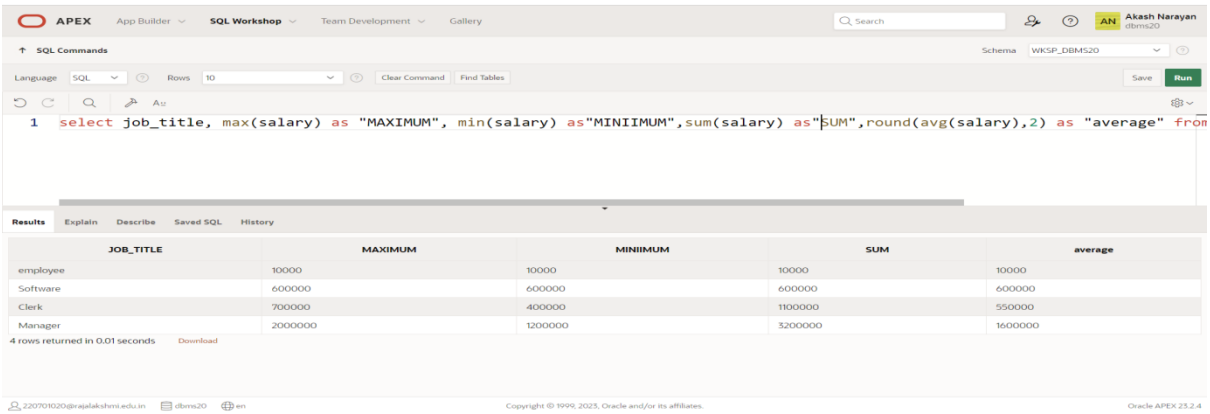
OUTPUT:



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

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

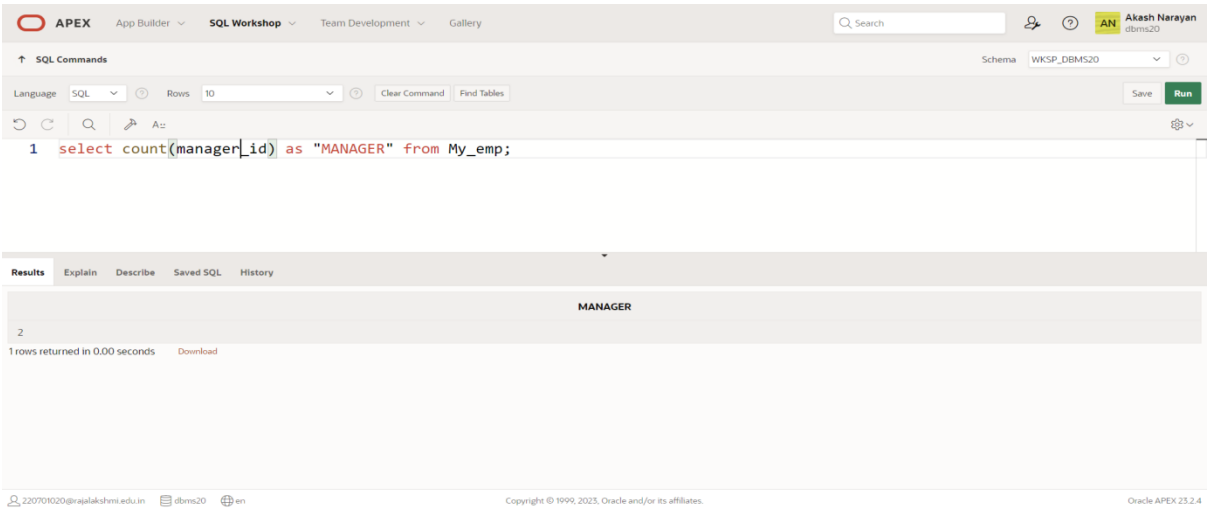
OUTPUT:



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

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

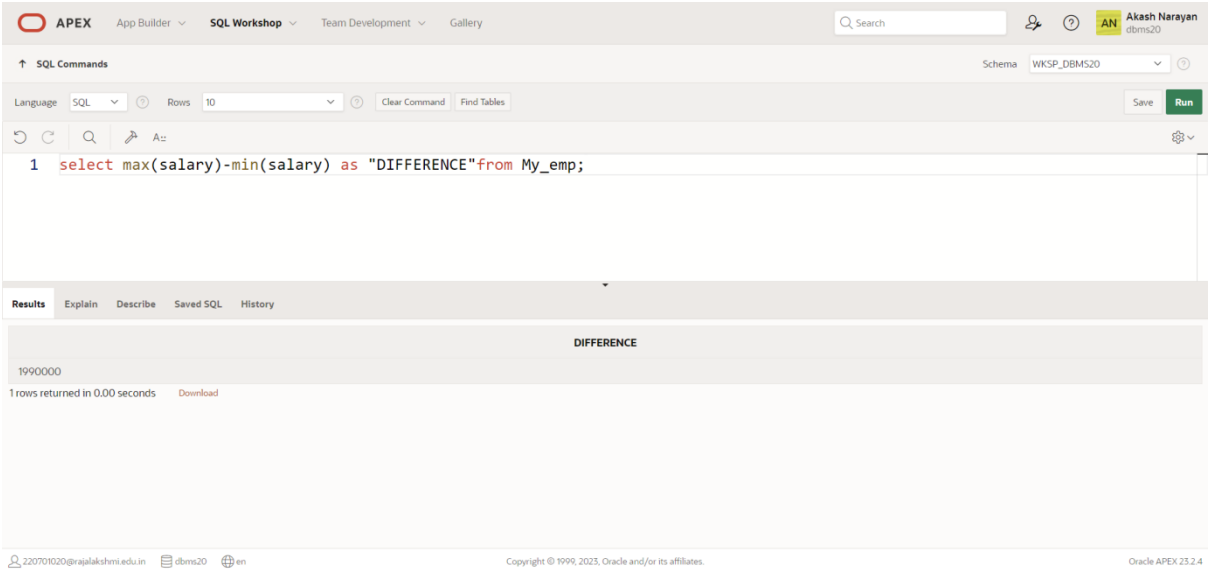
OUTPUT:



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

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

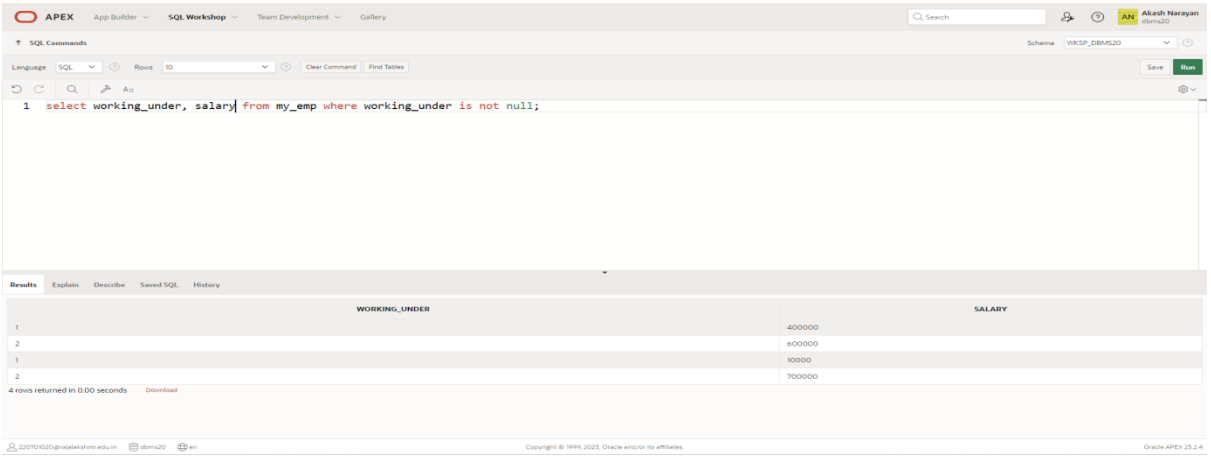
OUTPUT:



5.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 $6000 or less. Sort the output in descending order of salary

QUERY: select working_under, min(salary) from my_emp where working_under is not null;

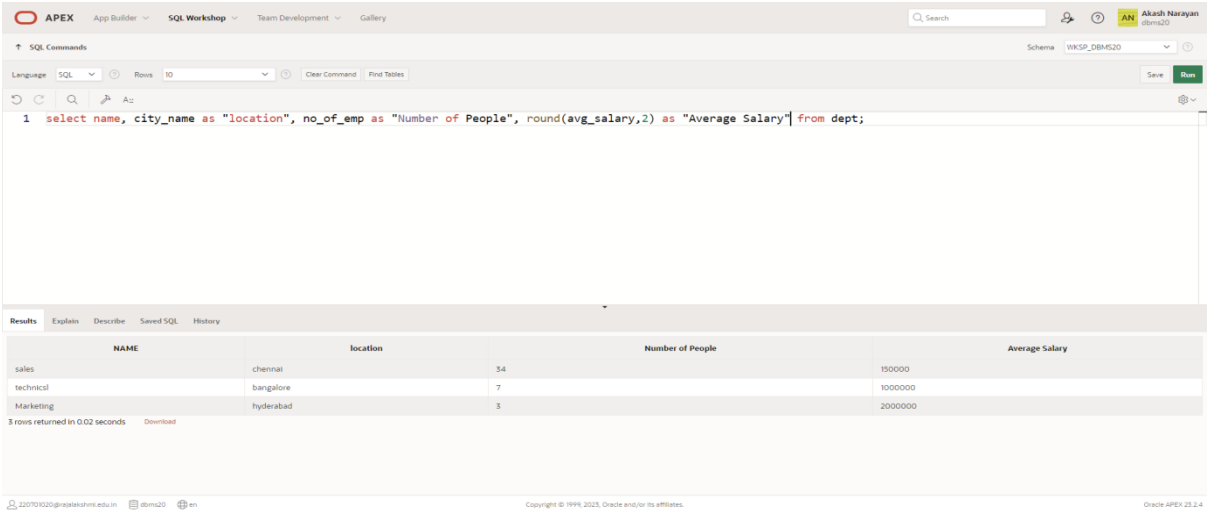
OUTPUT:



6. Write a query to display each department's name, location, number of employees and the average salary for all the employees in that department. Name the column name- Location, Number of people and salary respectively. Round the average salary to 2 decimal places.

QUERY: select name, city_name as "location", no_of_emp as "Number of People", round(avg_salary,2) as "Average Salary" from dept;

OUTPUT:



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

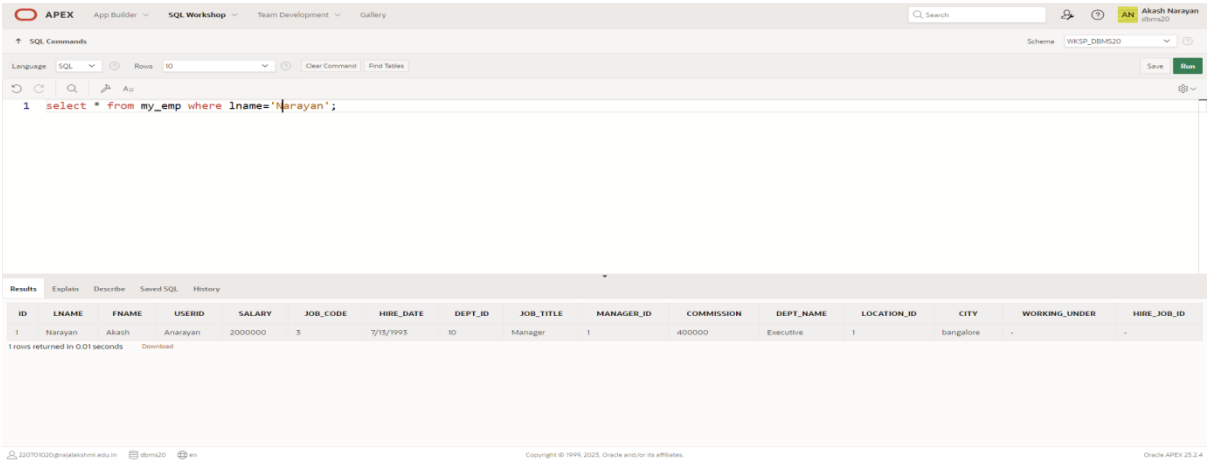
RESULT:

EXNO:9 **SUB QUERIES** **DATE:**

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

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

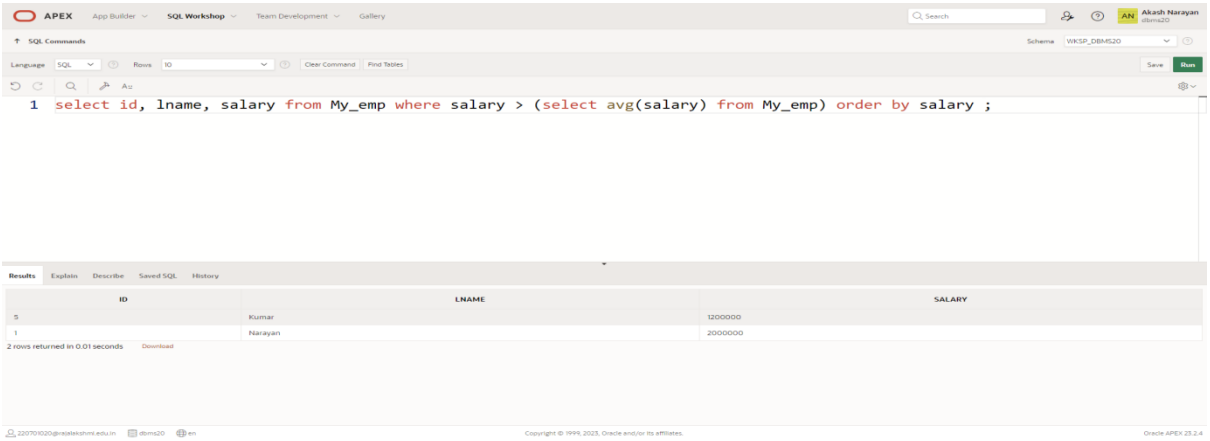
OUTPUT:



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

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

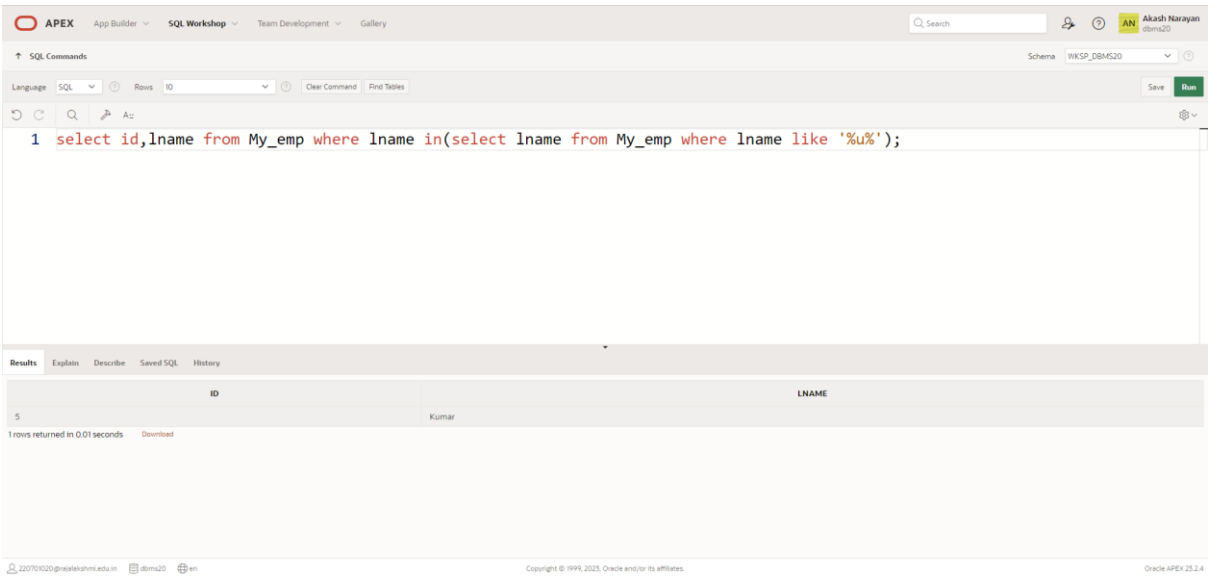
OUTPUT:



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

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

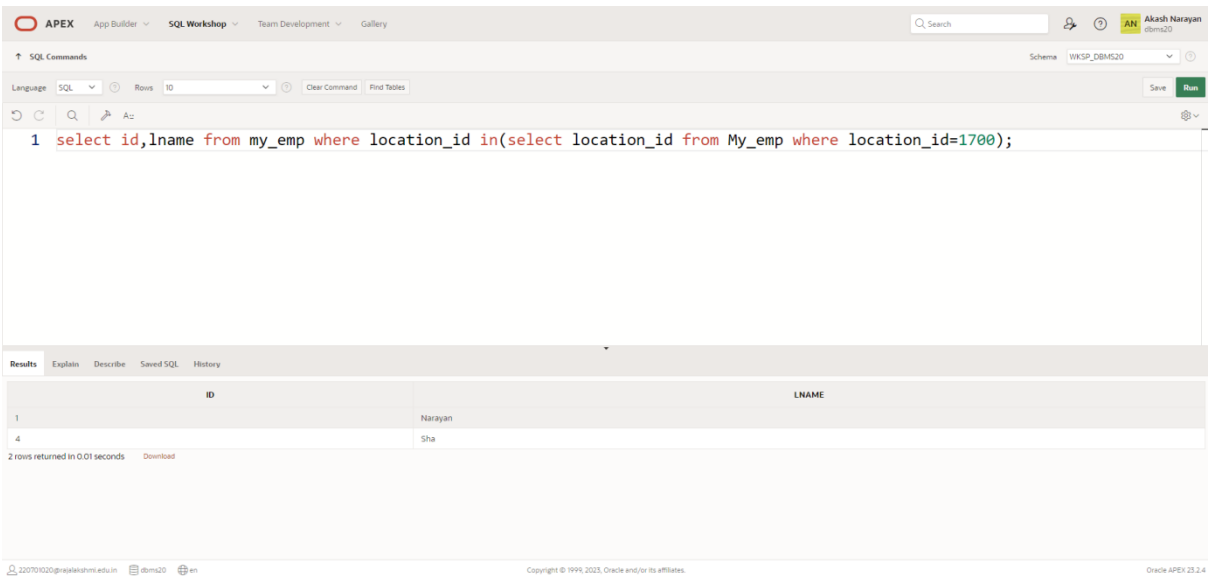
OUTPUT:



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

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

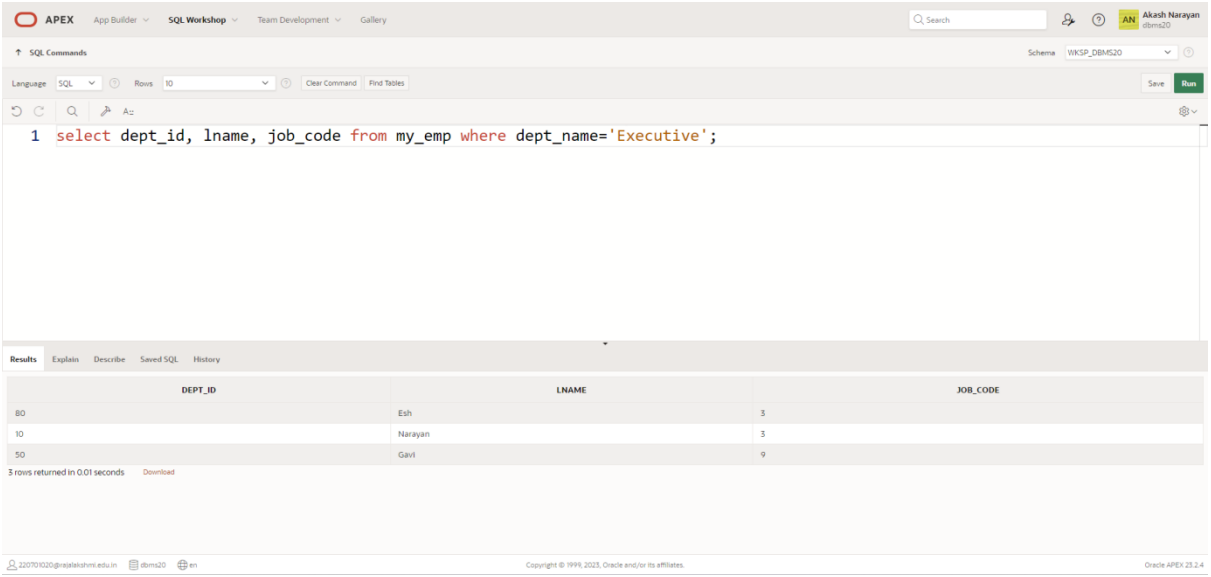
OUTPUT:



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

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

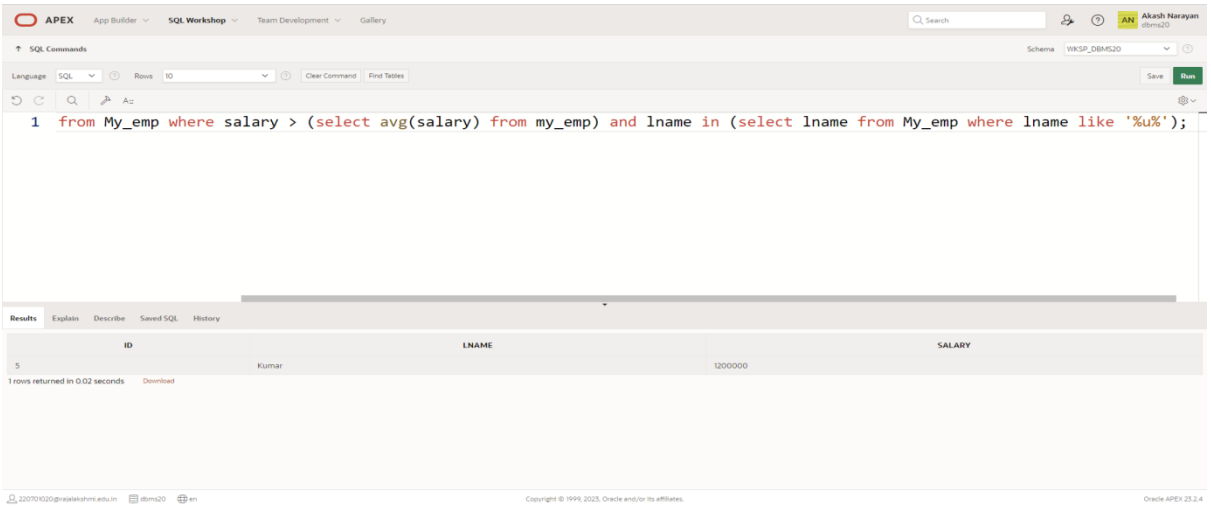
OUTPUT:



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

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

OUTPUT:



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

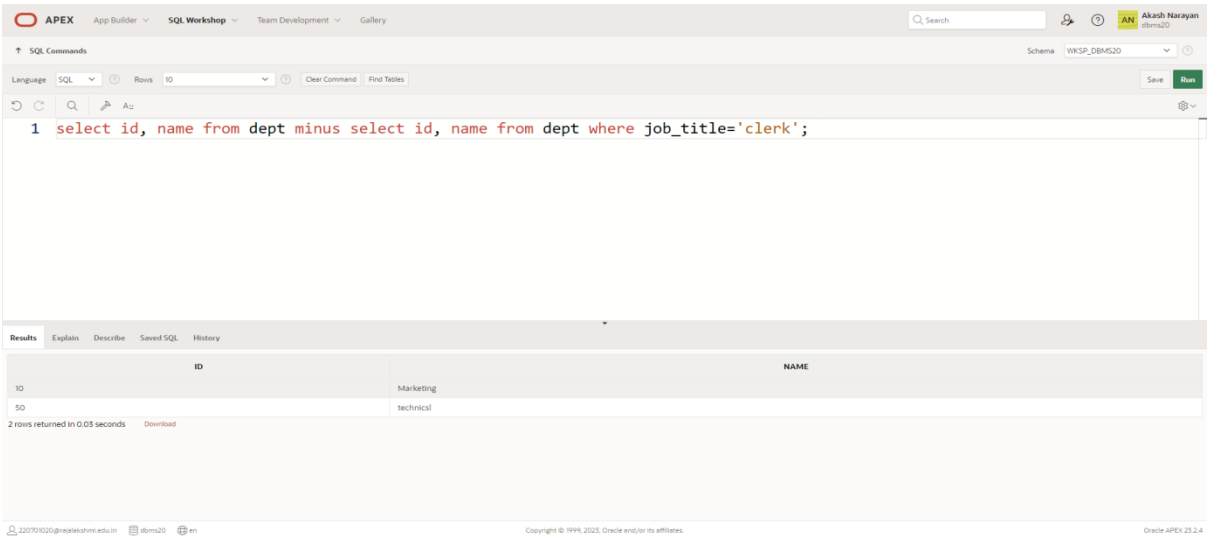
RESULT:

EXNO:10 USING THE SET OPERATORS DATE:

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

QUERY: select id, name from dept minus select id, name from dept where job_title='clerk';

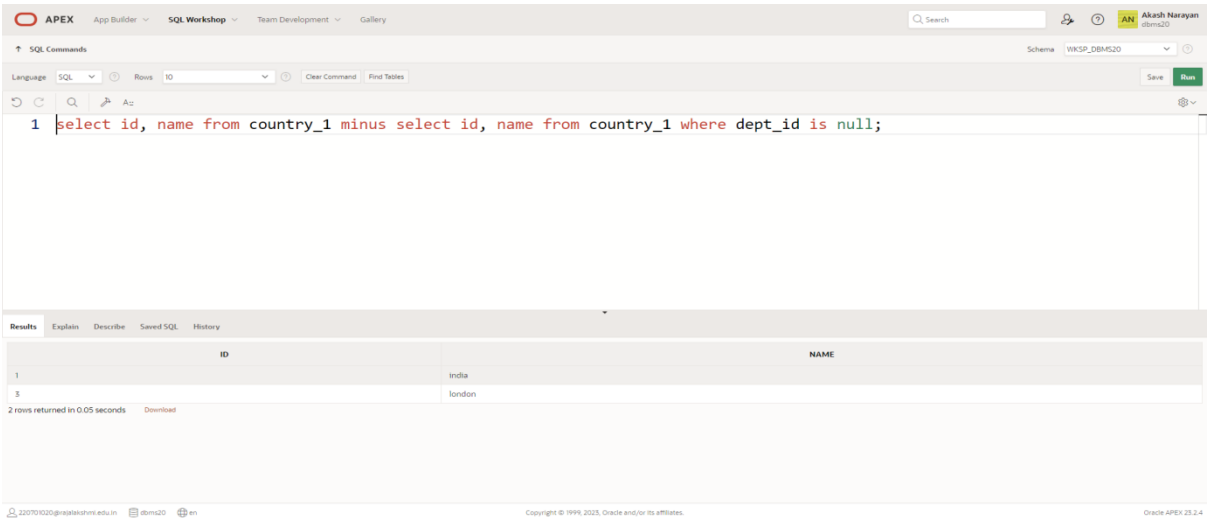
OUTPUT:



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 id, name from country_1 minus select id, name from country_1 where dept_id is null;

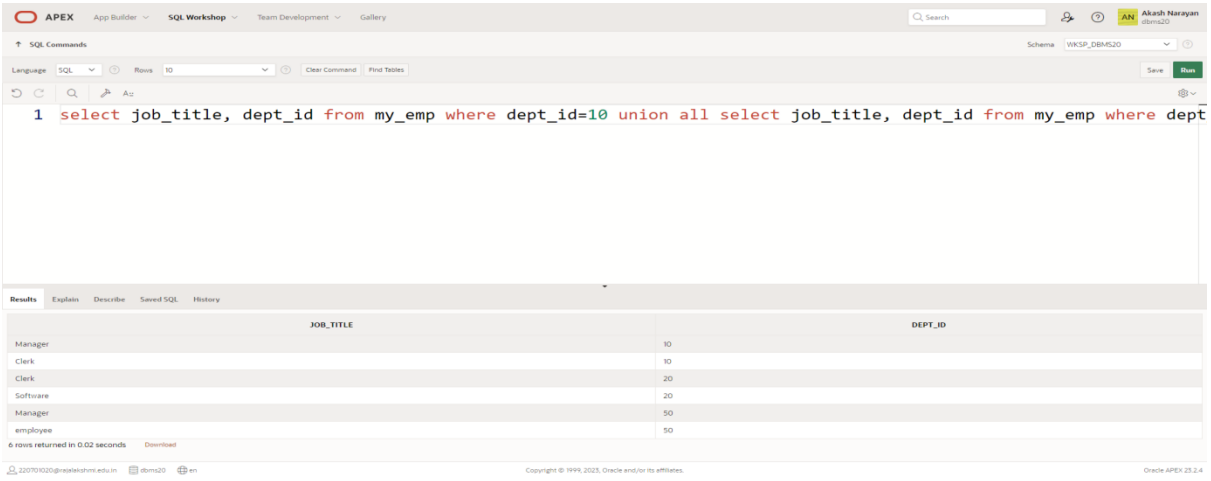
OUTPUT:



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_title, dept_id from my_emp where dept_id=10 union all select job_title, dept_id from my_emp where dept_id=20 union all select job_title, dept_id from my_emp where dept_id=50;;

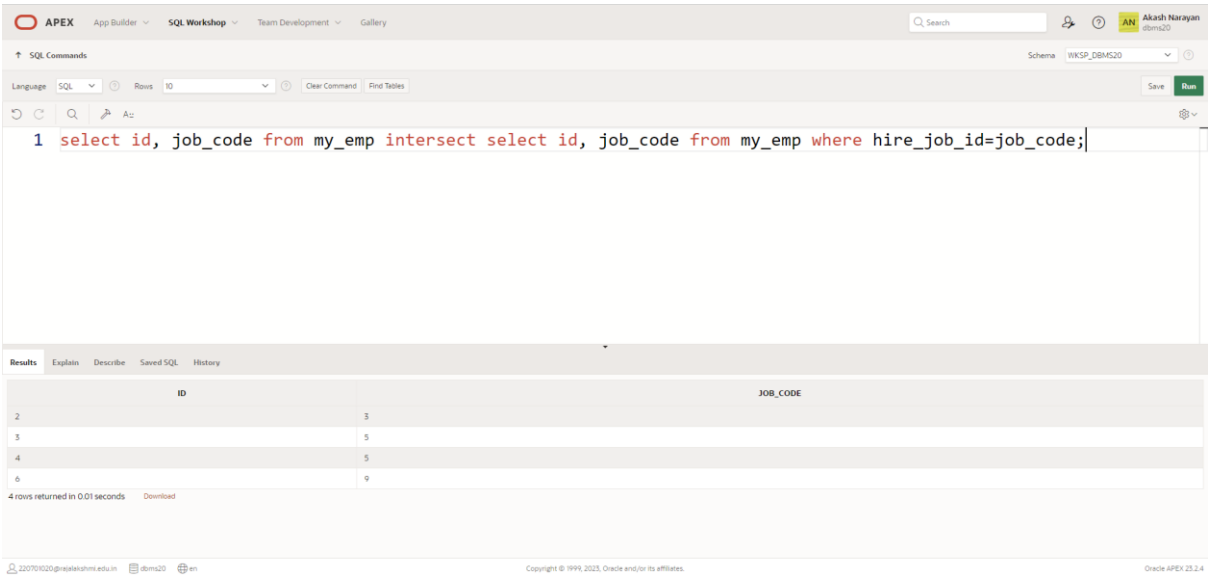
OUTPUT:



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:

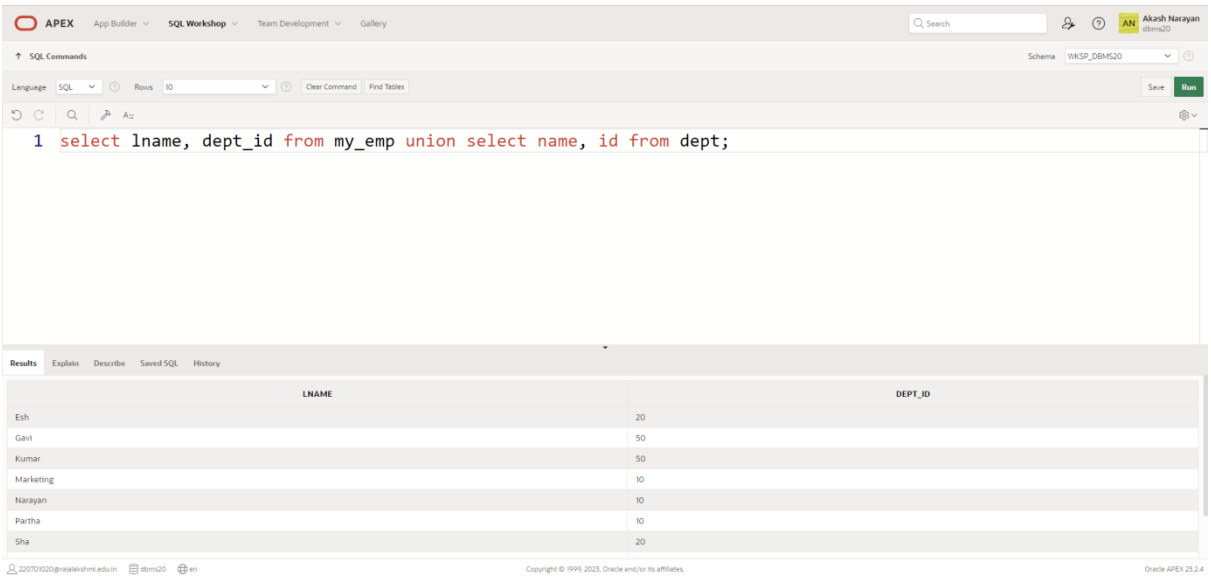
OUTPUT:



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

QUERY: select lname, dept_id from my_emp union select name, id from dept;

OUTPUT:



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

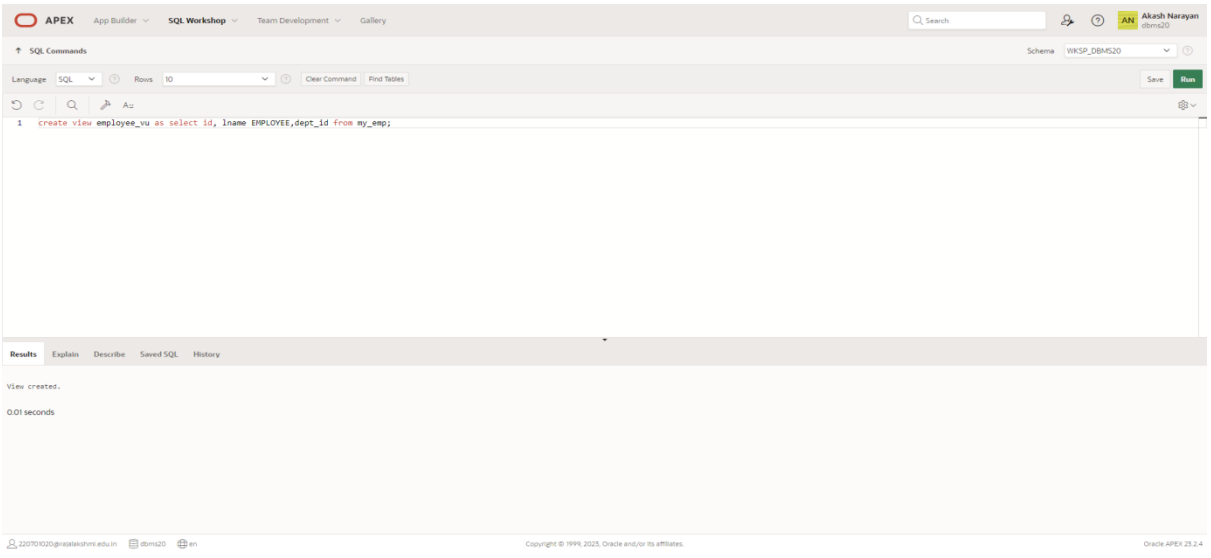
RESULT:

EXNO:11 **CREATING VIEWS** **DATE:**

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

QUERY: create view employee_vu as select id, lname EMPLOYEE,dept_id from my_emp;

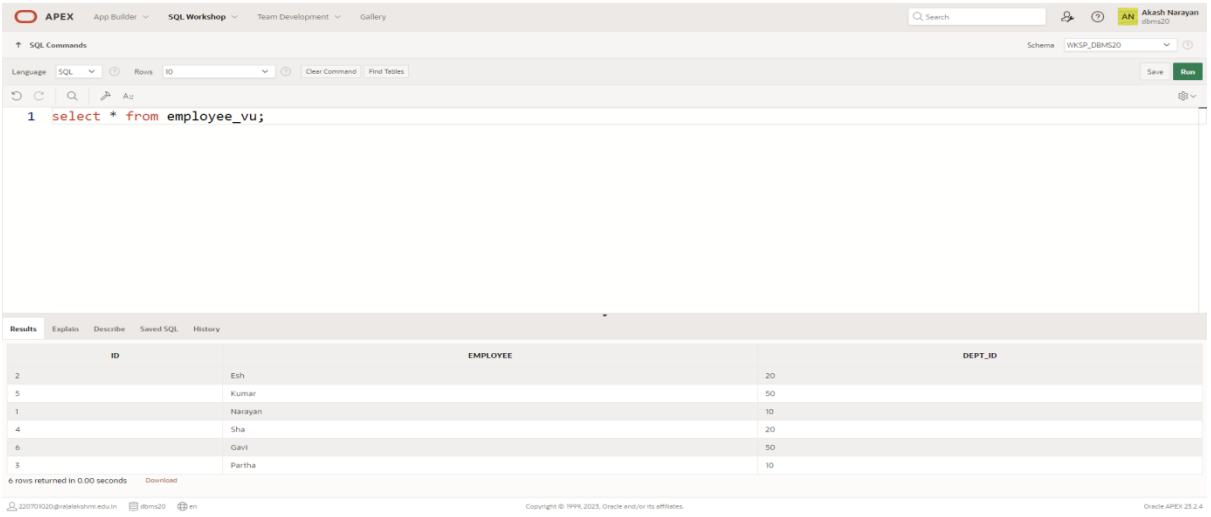
OUTPUT:



2.Display the contents of the EMPLOYEES_VU view.

QUERY: select * from employee_vu;

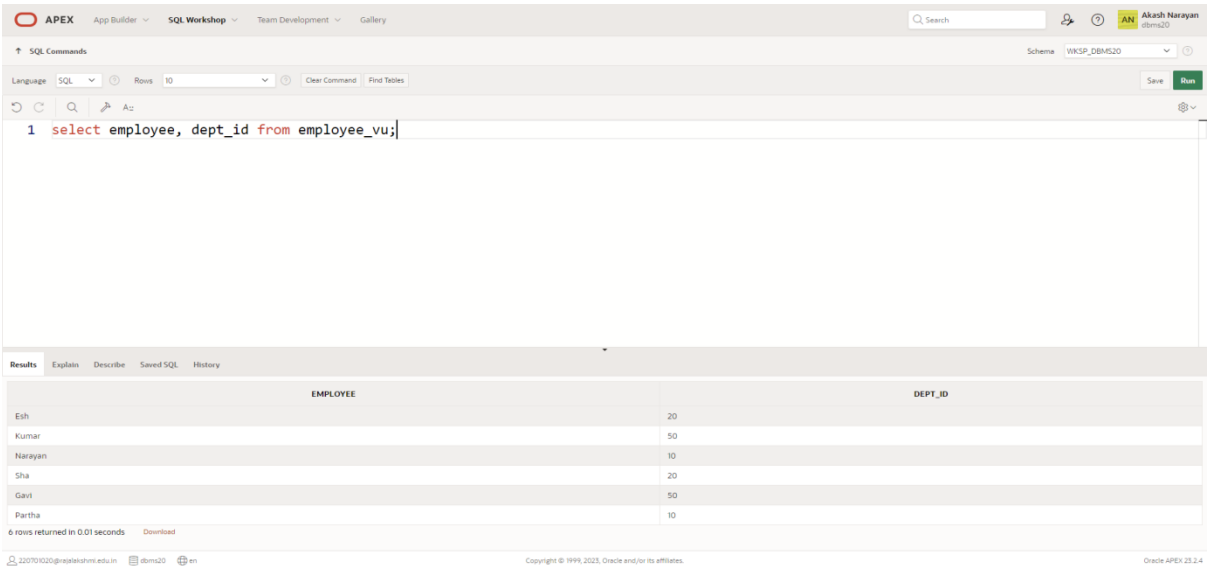
OUTPUT:



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

QUERY: select employee, dept_id from employee_vu;

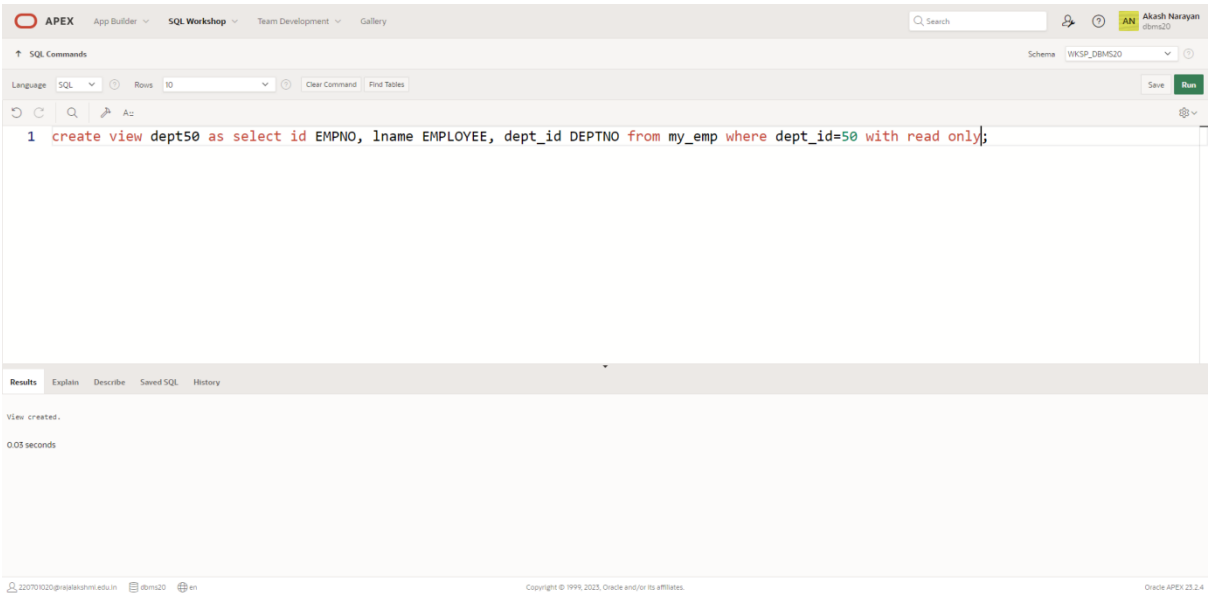
OUTPUT:



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

QUERY: create view dept50 as select id EMPNO, lname EMPLOYEE, dept_id DEPTNO from my_emp where dept_id=50 with read only;

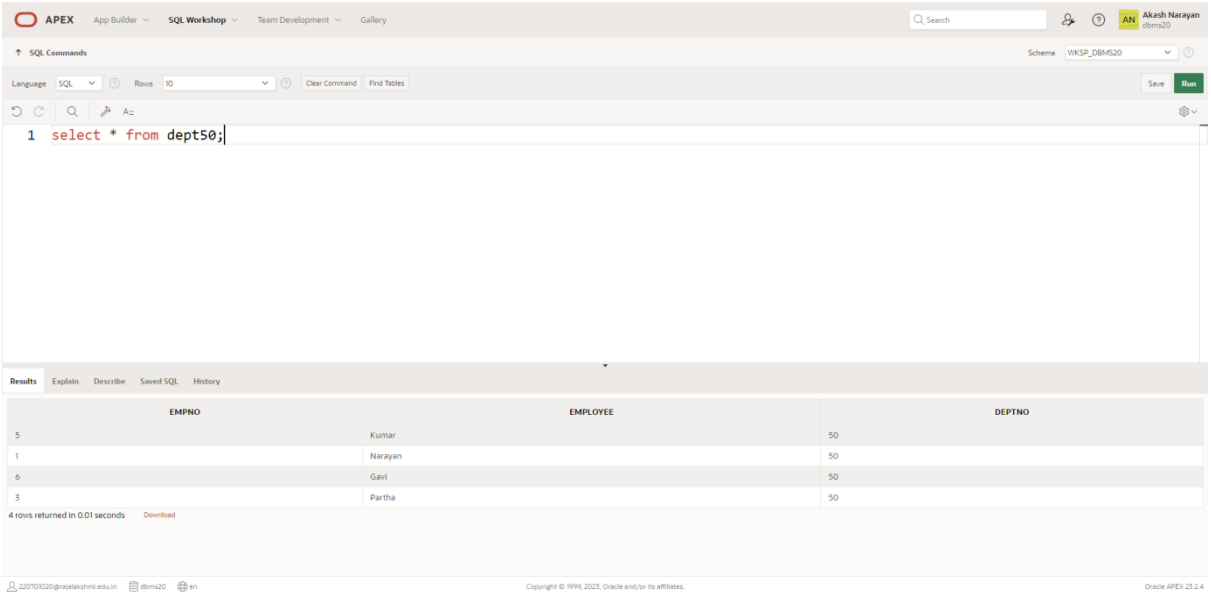
OUTPUT:



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

QUERY: select * from dept50;

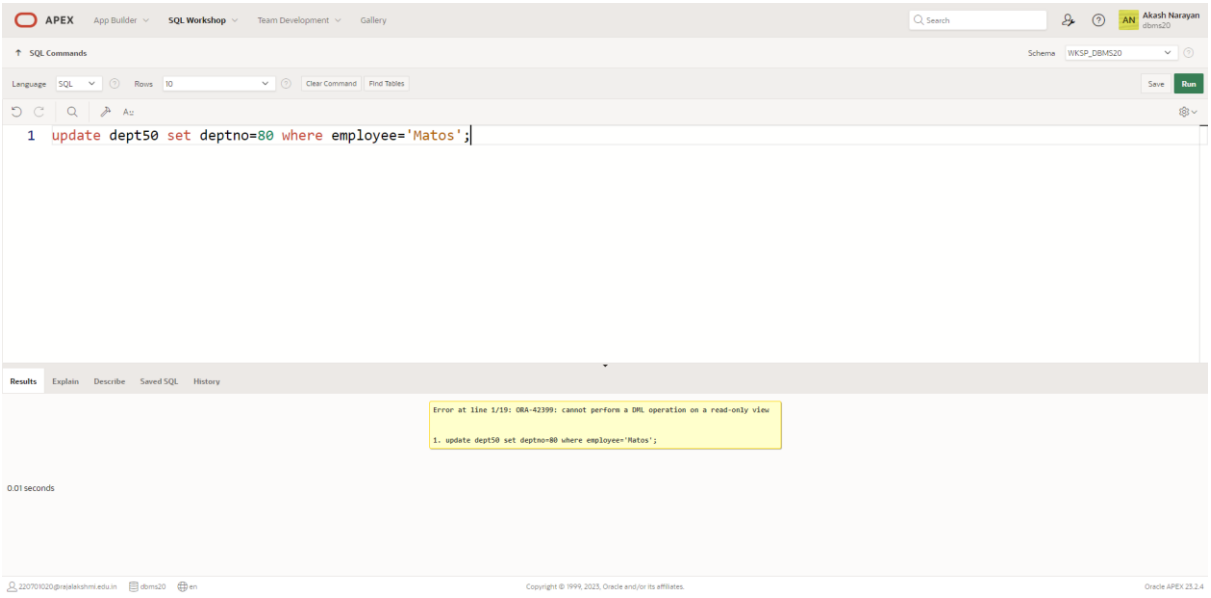
OUTPUT:



6.Attempt to reassign Matos to department 80.

QUERY: update dept50 set deptno=80 where employee='Matos';

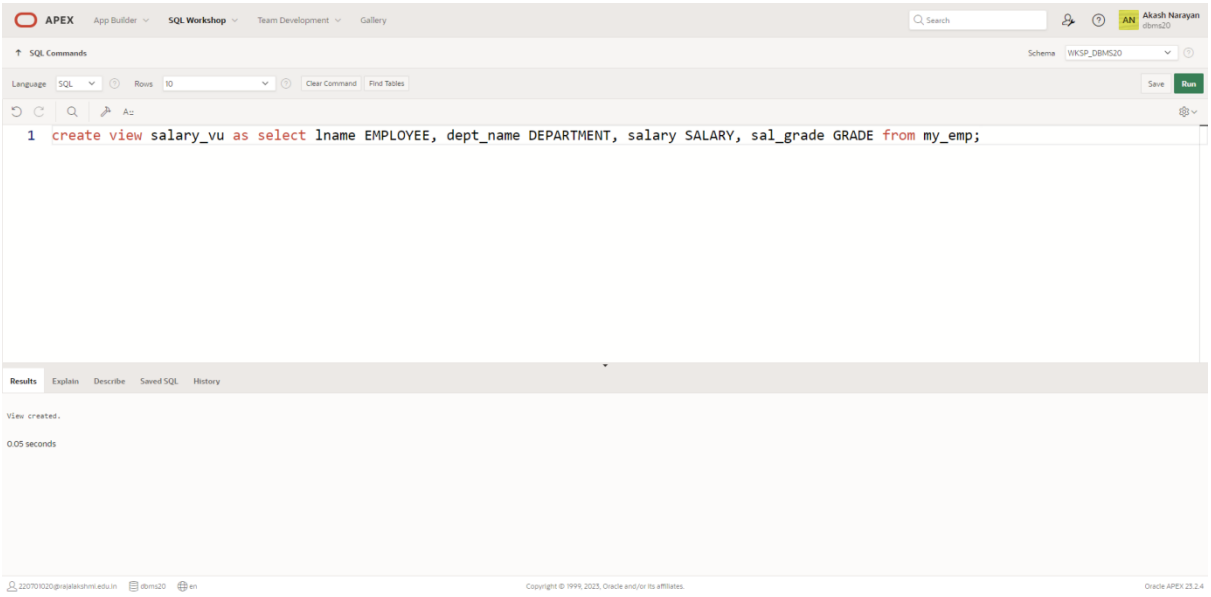
OUTPUT:



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

QUERY: create view salary_vu as select lname EMPLOYEE, dept_name DEPARTMENT, salary SALARY, sal_grade GRADE from my_emp;

OUTPUT:



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

RESULT:

INTRODUCTION TO CONSTRAINTS: NOT NULL AND UNIQUE CONSTRAINTS

EXNO:12 **DATE:**

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

Ans: A constraint is a criteria or a condition based on which a certain action is done.

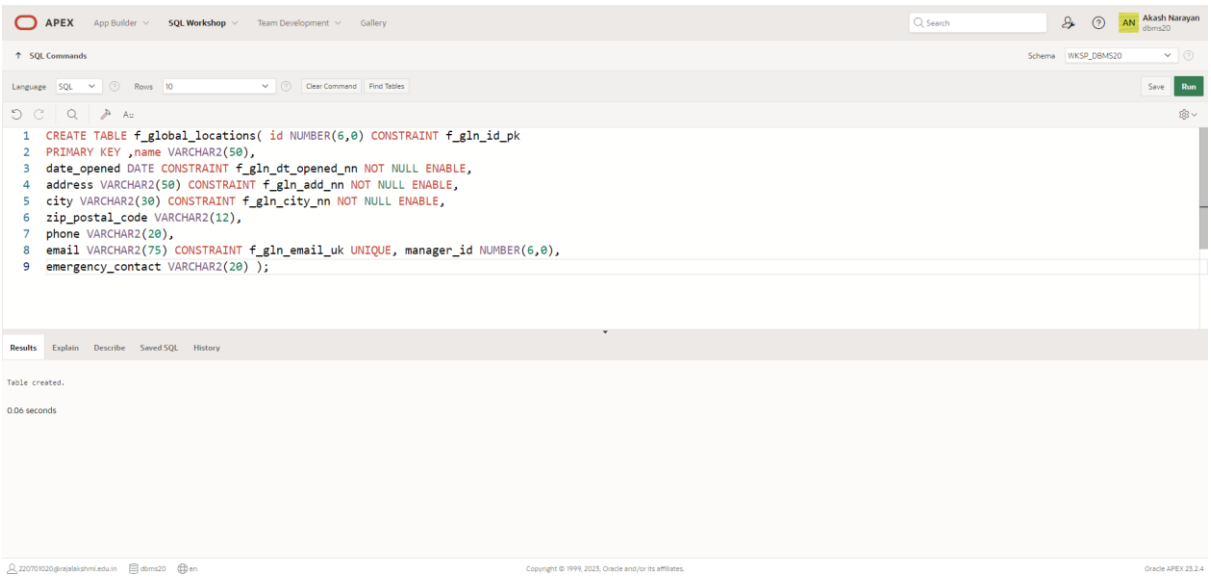
2. Why is it important to give meaningful names to constraints?

Ans: To understand the data easily and access it faster.

3.Write the create table statement for the Global fast foods locations table to define the constraints at the column level.

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

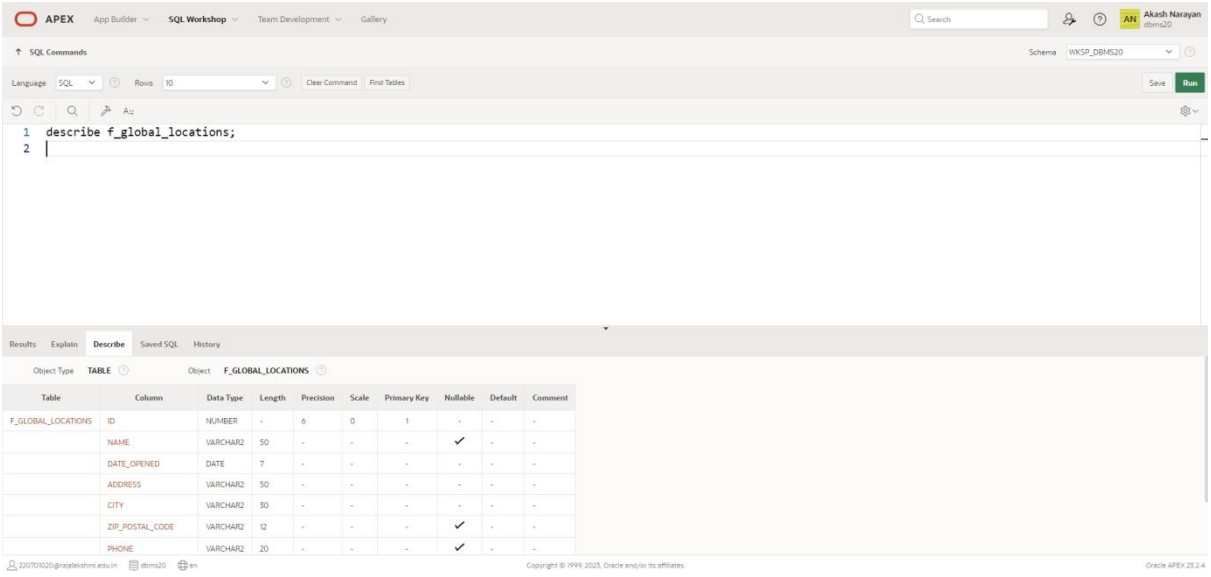
OUTPUT:



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

QUERY: describe f_global_locations;

OUTPUT:



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

QUERY:

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

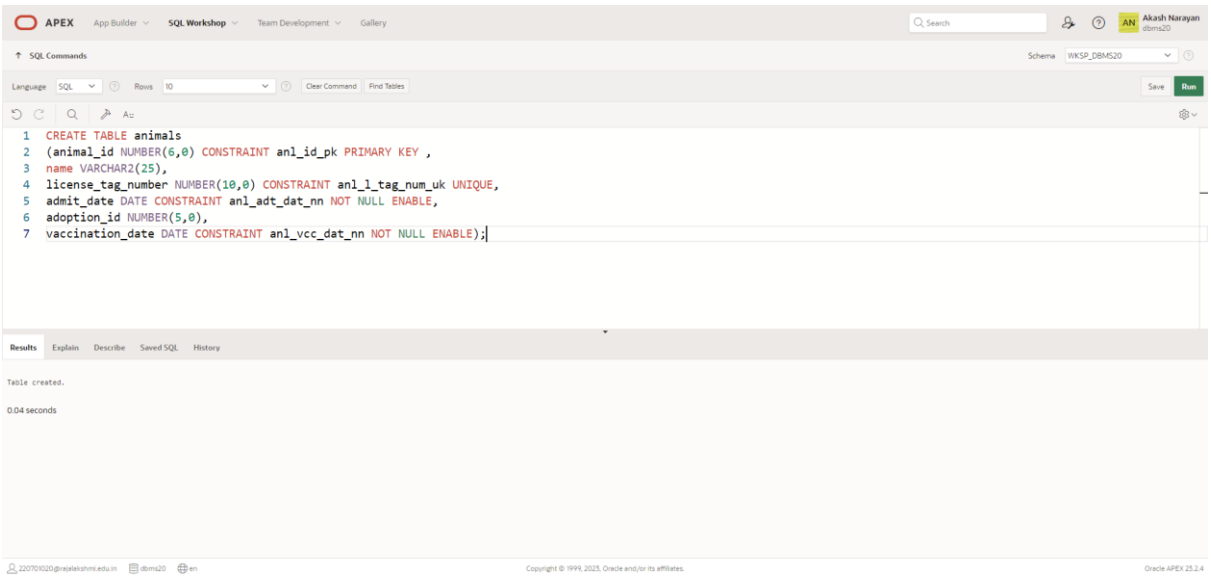
PRIMARY KEY, FOREIGN KEY AND CHECK CONSTRAINTS

1. What is the purpose of a
• PRIMARY KEY – They provide a unique value that can identify a specific row in a table
• FOREIGN KEY - to link data between tables
• CHECK CONSTRAINT - to limit the value range that can be placed in a column

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

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

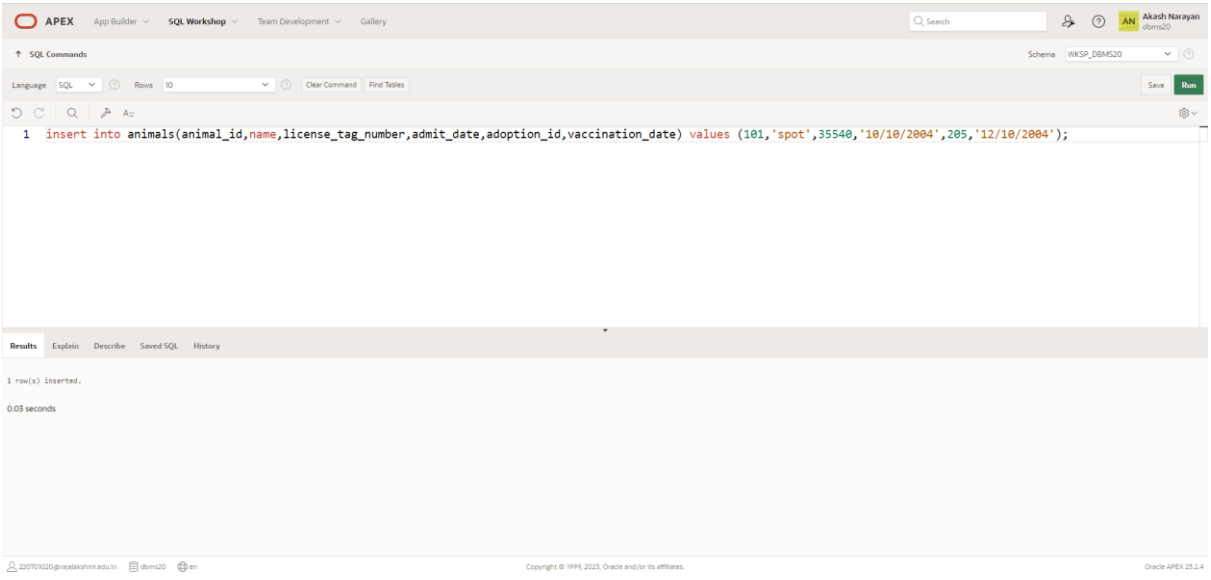
OUTPUT:



3. Enter one row into the table. Execute a SELECT * statement to verify your input.

QUERY:insert into animals(animal_id,name,license_tag_number,admit_date,adoption_id, vaccination_date) values (101,'spot',35540,'10/10/2004',205,'12/10/2004');

OUTPUT:



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

Ans: If you define a CHECK constraint on a column it will allow only certain values for this column. If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

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

RESULT:

CREATING VIEWS

EXNO:13 **DATE:**

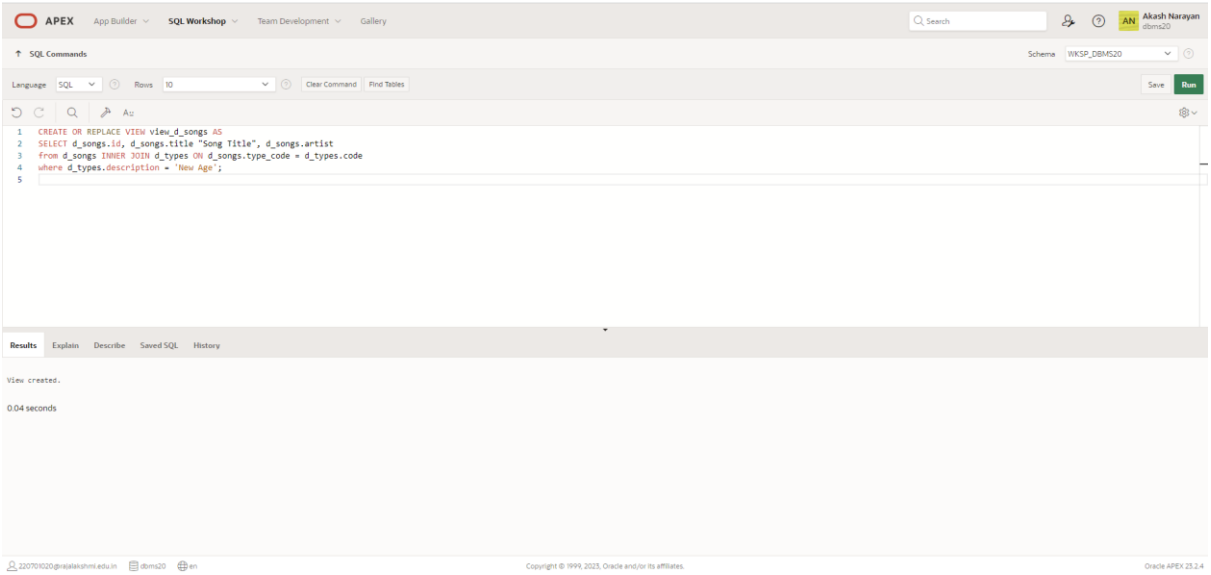
1. **What are three uses for a view from a DBA's perspective?**

Ans: Restrict access and display selective columns
Reduce complexity of queries from other internal systems. So, providing a way to view same data in a different manner.
Let the app code rely on views and allow the internal implementation of tables to be modified later.

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

QUERY: CREATE OR REPLACE 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';

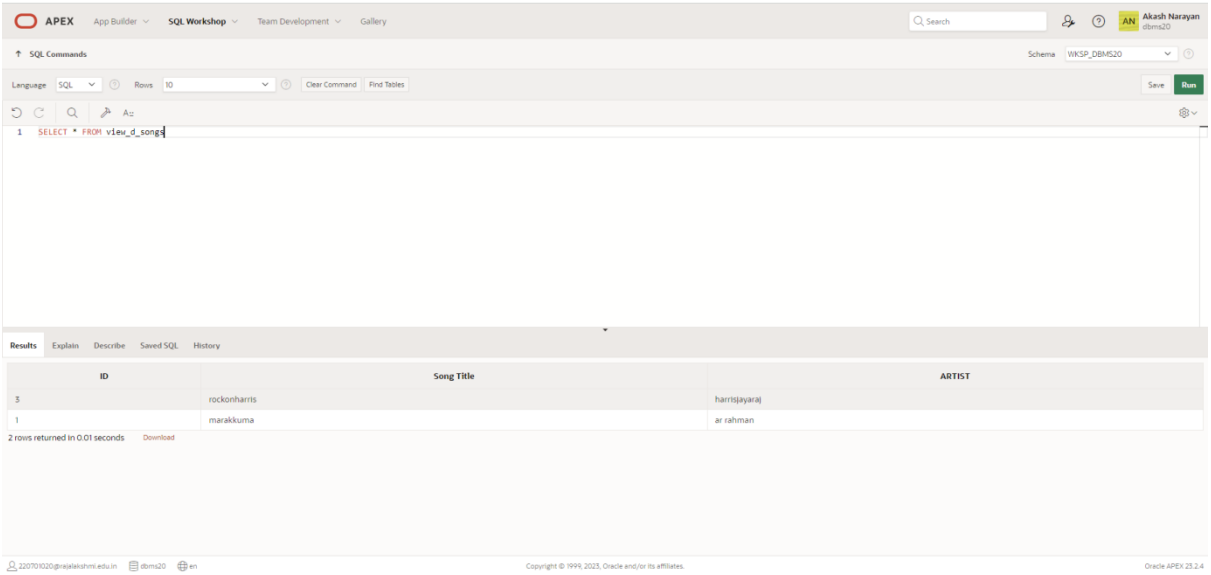
OUTPUT:



3. SELECT * FROM view_d_songs. What was returned?

QUERY: SELECT * FROM view_d_songs

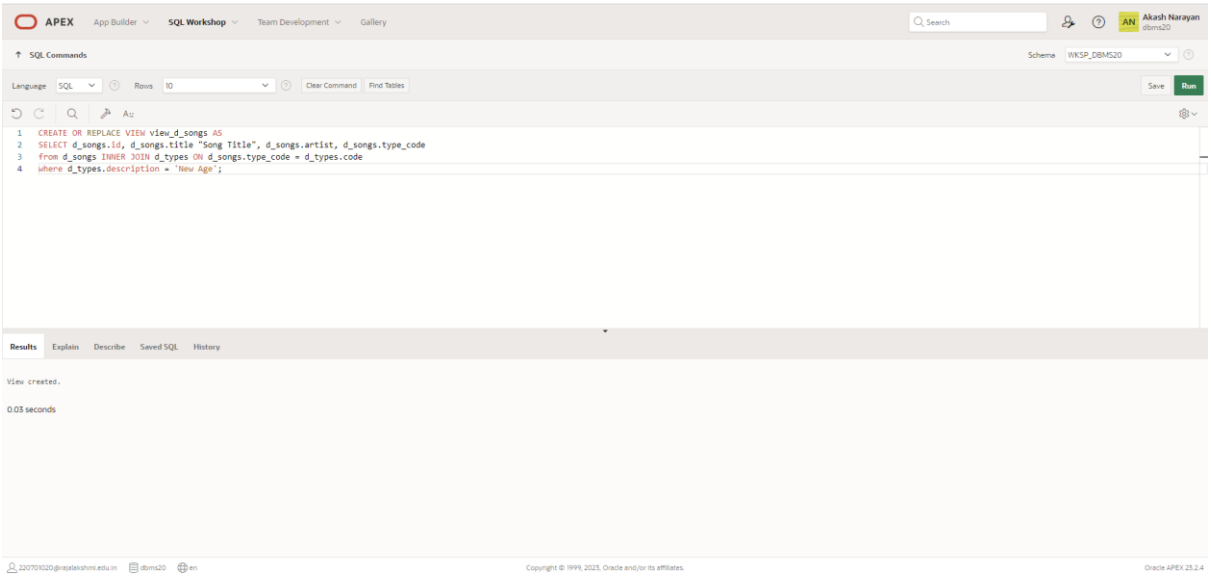
OUTPUT:



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

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

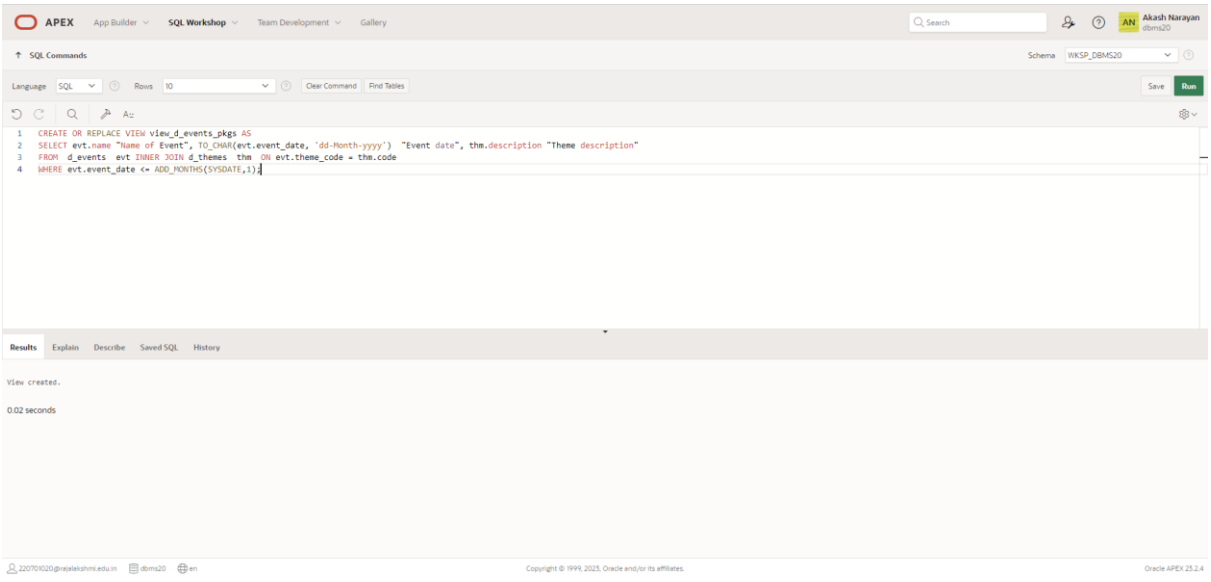
OUTPUT:



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

QUERY: 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);

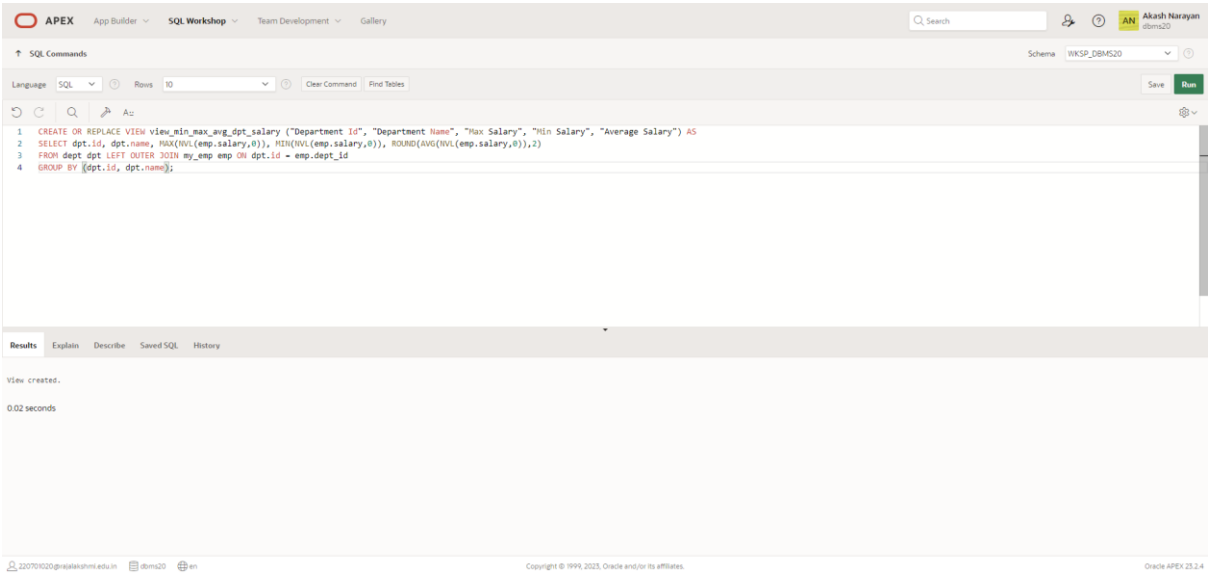
OUTPUT:



6. It is company policy that only upper-level management be allowed access 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.

QUERY: CREATE OR REPLACE VIEW view_min_max_avg_dpt_salary ("Department Id", "Department Name", "Max Salary", "Min Salary", "Average Salary") AS
SELECT dpt.id, dpt.name, MAX(NVL(emp.salary,0)), MIN(NVL(emp.salary,0)), ROUND(AVG(NVL(emp.salary,0)),2)
FROM dept dpt LEFT OUTER JOIN my_emp emp ON dpt.id = emp.dept_id
GROUP BY (dpt.id, dpt.name);

OUTPUT:

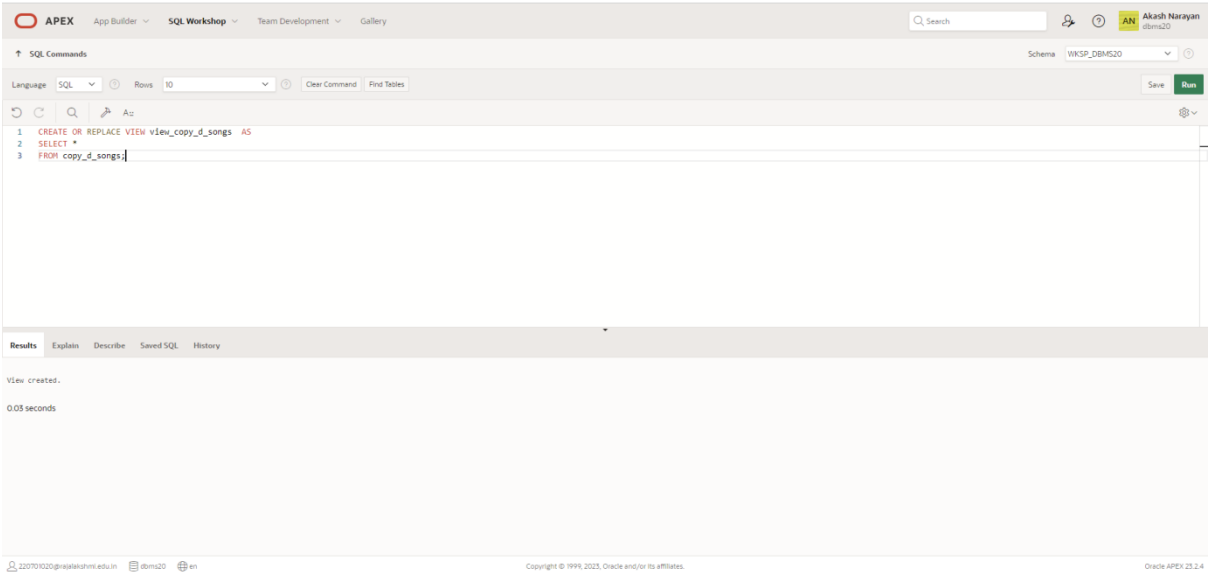


DML OPERATIONS AND VIEWS

1.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.

QUERY: CREATE OR REPLACE VIEW view_copy_d_songs AS
SELECT *
FROM copy_d_songs;

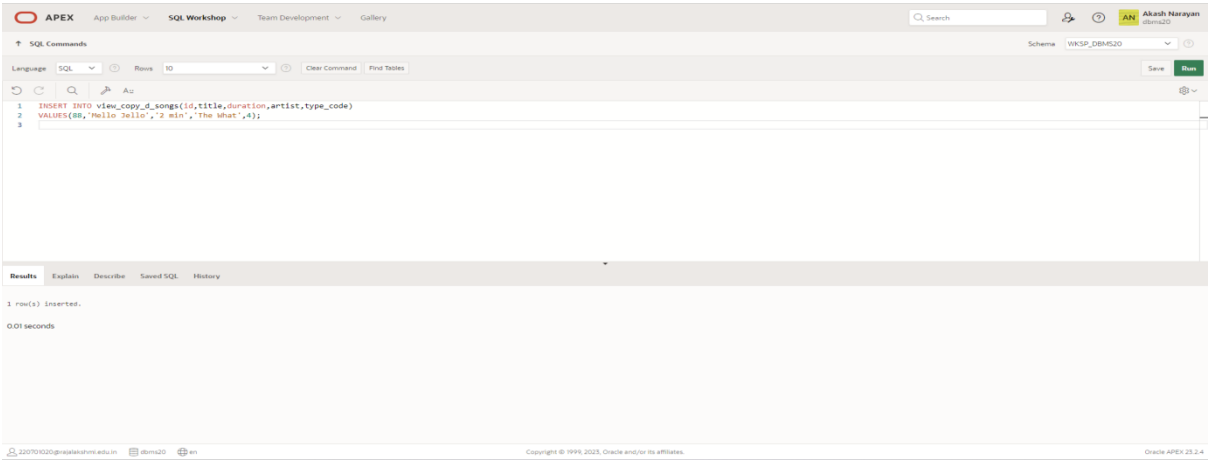
OUTPUT:



2. 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.

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

OUTPUT:

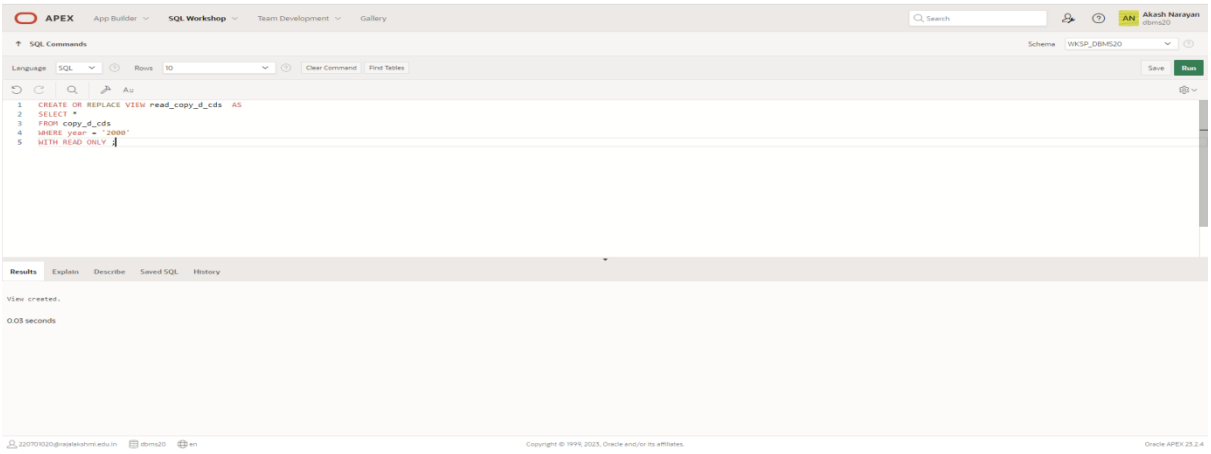


3.Create a view based on the DJs on Demand COPY_D_CDS table. Name the view read_copy_d_cds. Select all columns to be included in the view. Add a WHERE clause to restrict the year to 2000. Add the WITH READ ONLY option.

QUERY: 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;

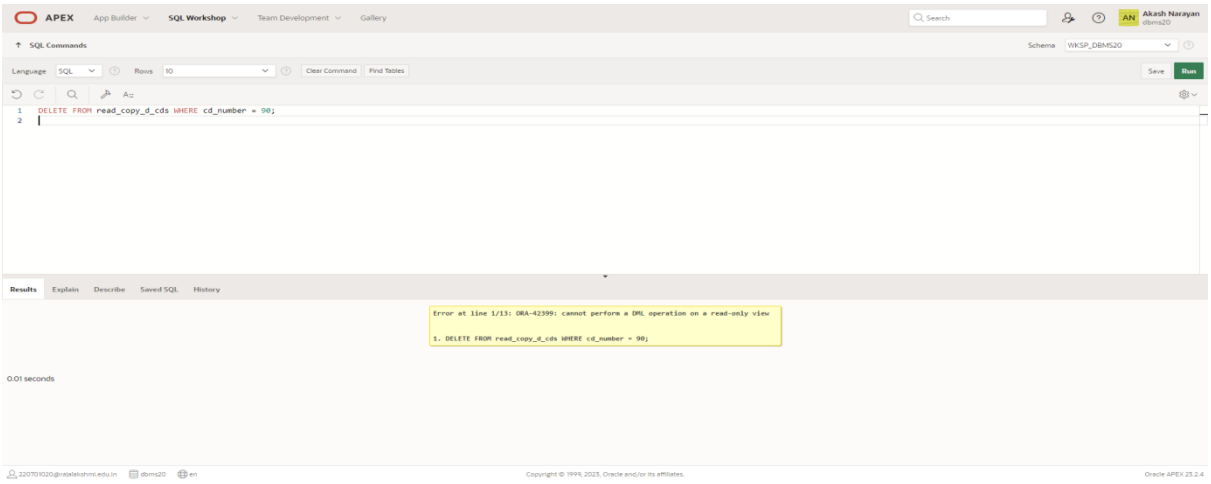
OUTPUT:



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

QUERY: DELETE FROM read_copy_d_cds WHERE cd_number = 90;

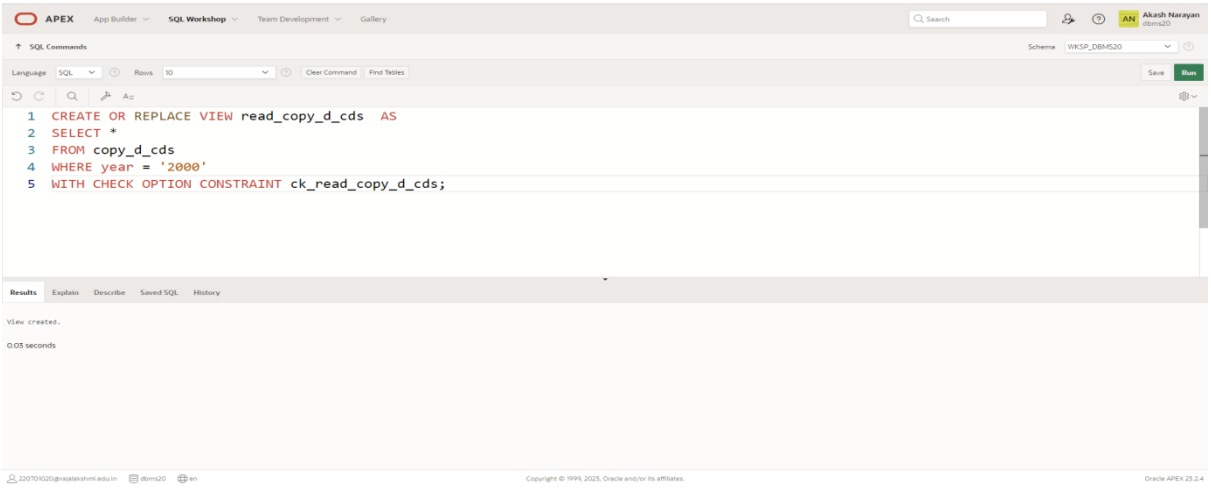
OUTPUT:



5. 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.

QUERY: 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;

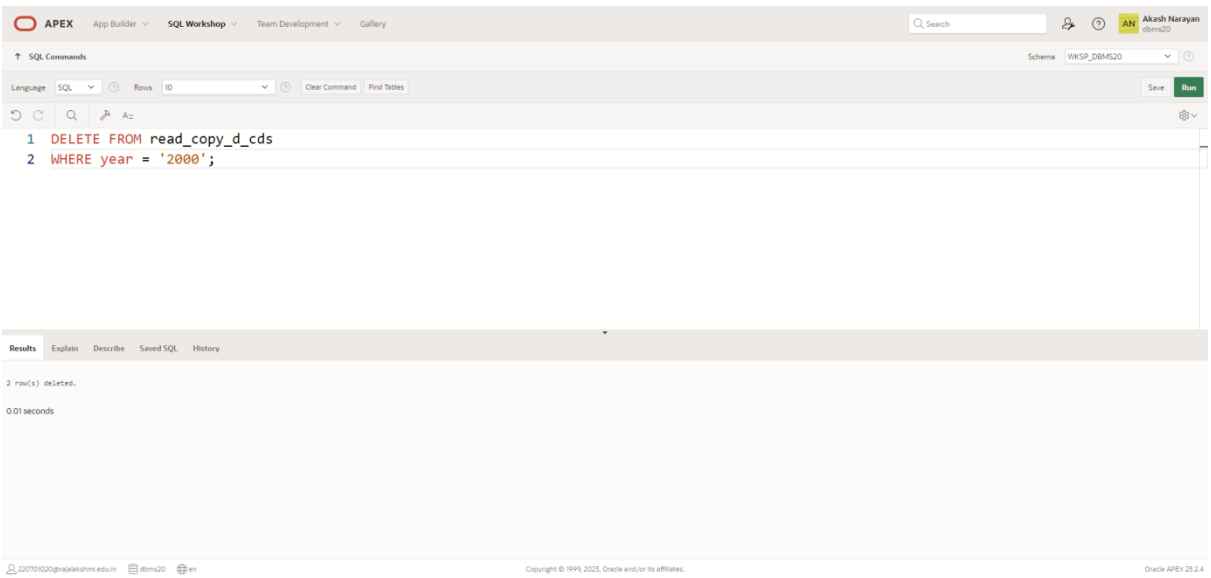
OUTPUT:



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

QUERY: DELETE FROM read_copy_d_cds
WHERE year = '2000';

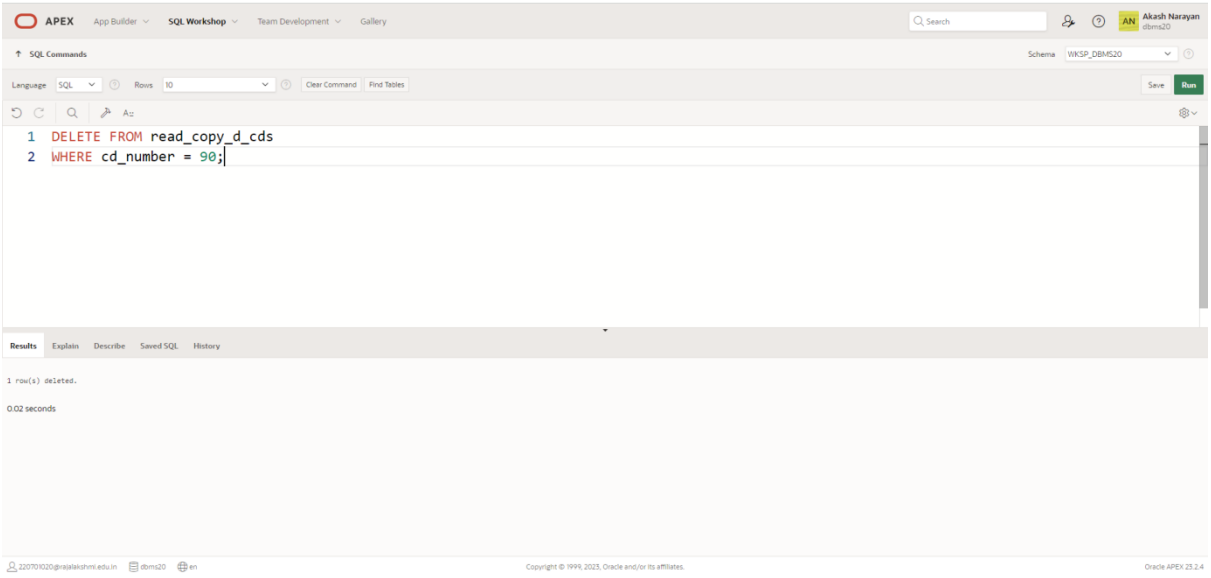
OUTPUT:



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

QUERY: DELETE FROM read_copy_d_cds
WHERE cd_number = 90;

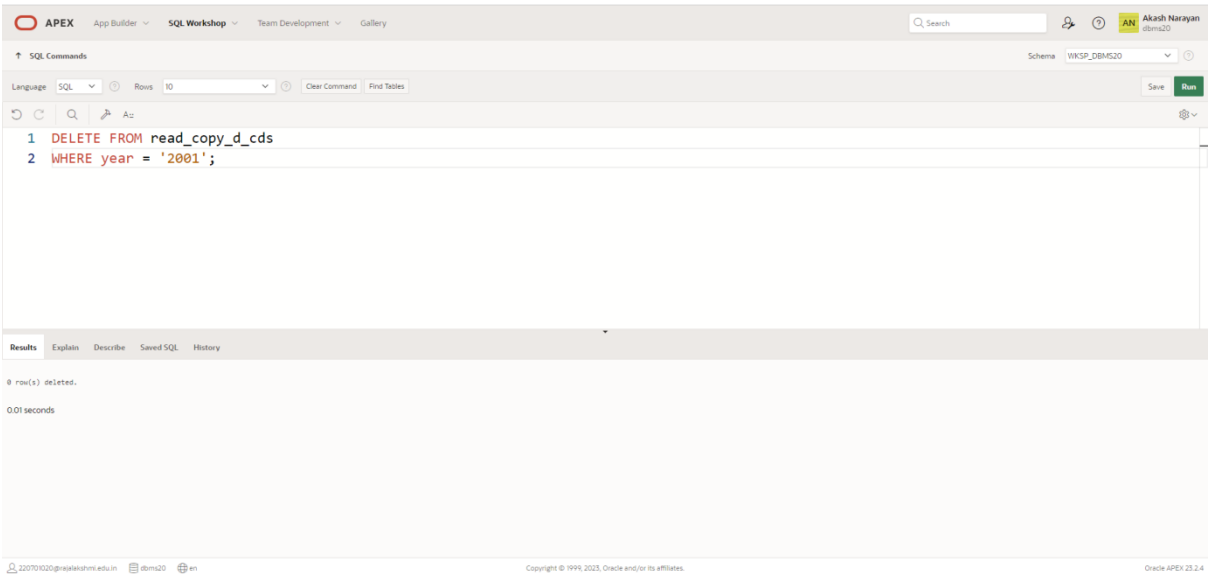
OUTPUT:



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

QUERY: DELETE FROM read_copy_d_cds
WHERE year = '2001';

OUTPUT:

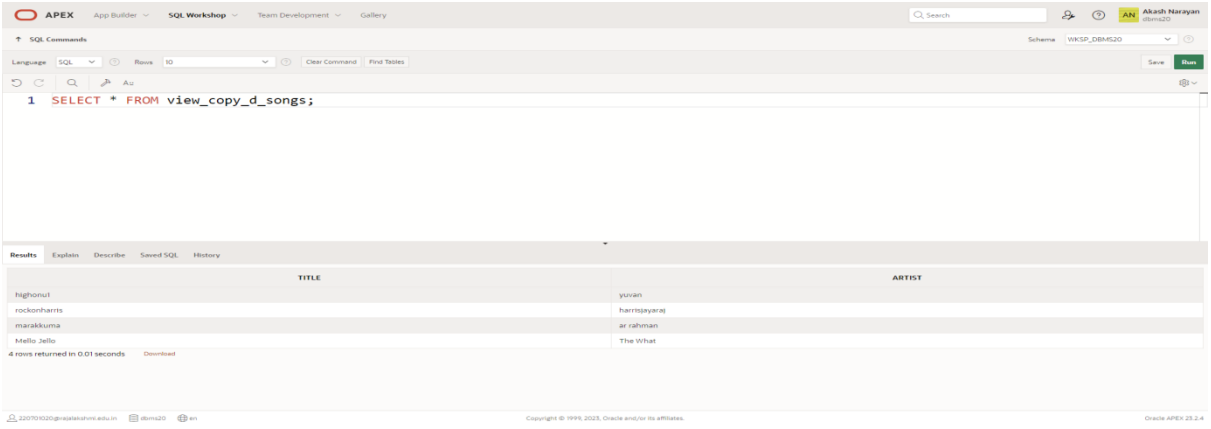


MANAGING VIEWS

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.

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

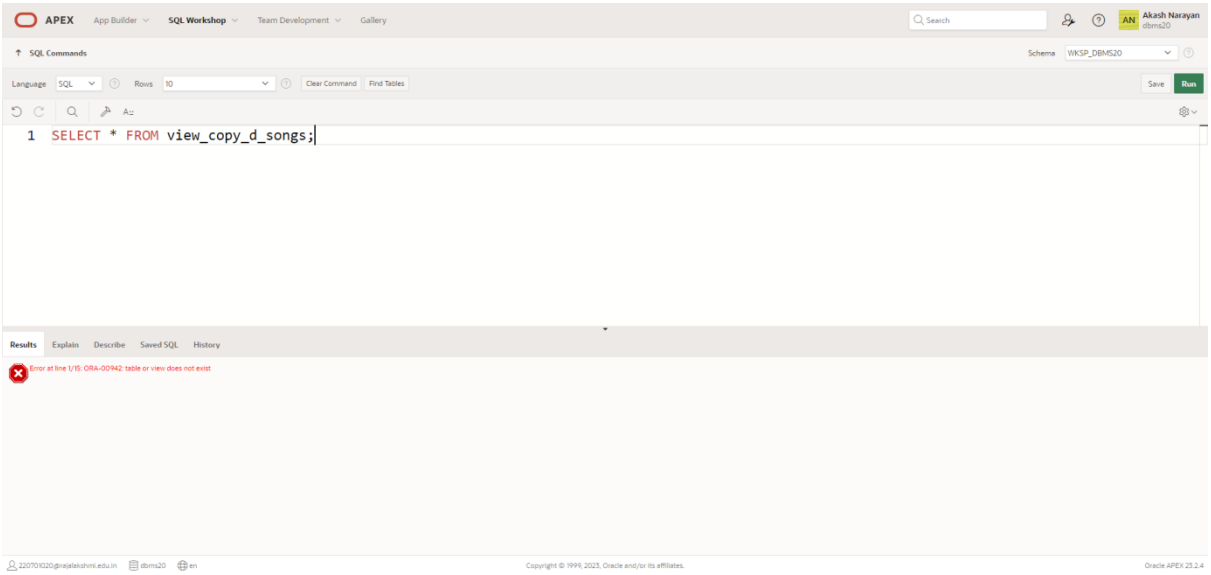
OUTPUT:



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

QUERY: DROP VIEW view_copy_d_songs;

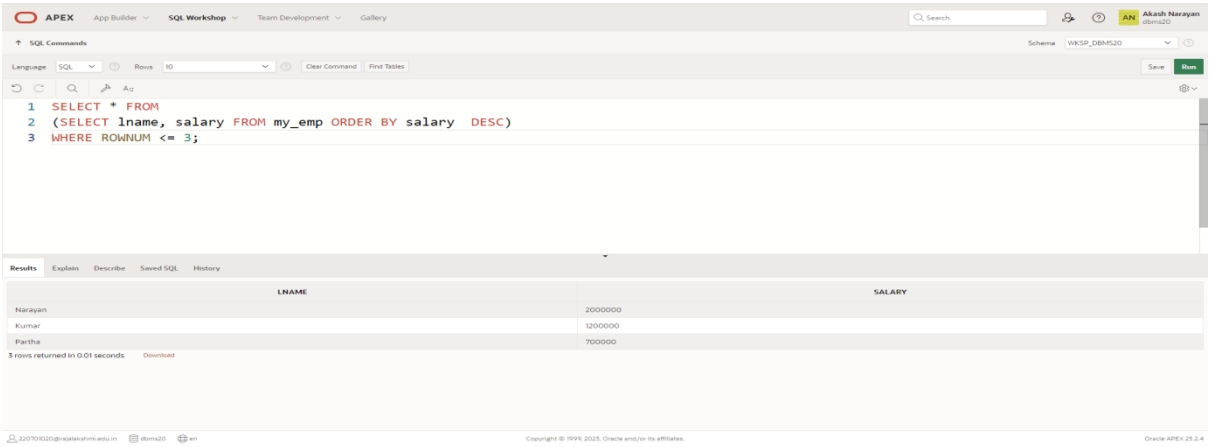
OUTPUT:



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.

QUERY: SELECT * FROM
(SELECT lname, salary FROM my_emp ORDER BY salary DESC)
WHERE ROWNUM <= 3;

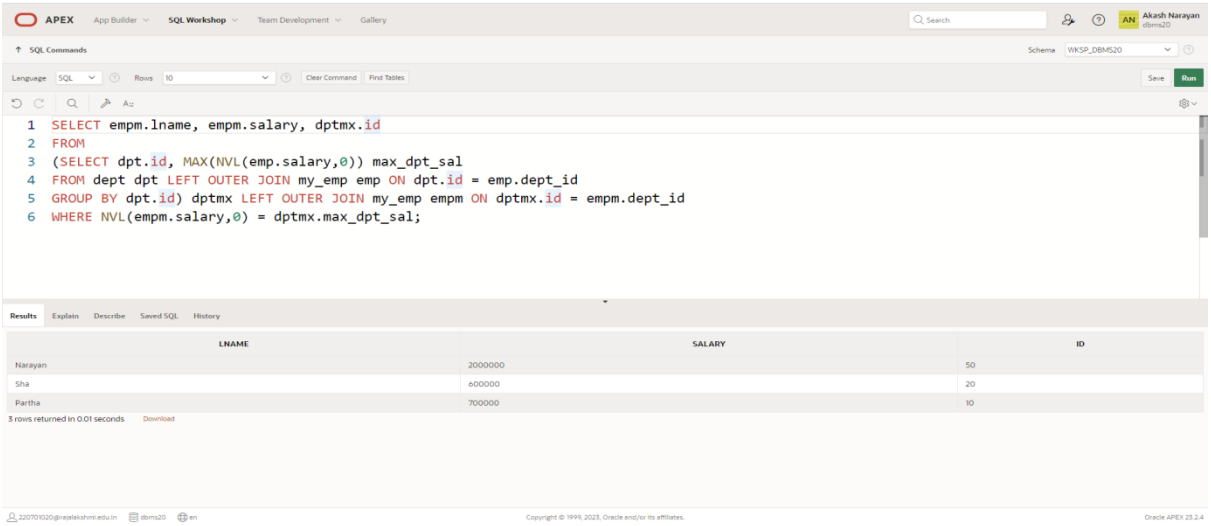
OUTPUT:



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.

QUERY: SELECT empm.lname, empm.salary, dptmx.id
FROM
(SELECT dpt.id, MAX(NVL(emp.salary,0)) max_dpt_sal
FROM dept dpt LEFT OUTER JOIN my_emp emp ON dpt.id = emp.dept_id
GROUP BY dpt.id) dptmx LEFT OUTER JOIN my_emp empm ON dptmx.id = empm.dept_id
WHERE NVL(empm.salary,0) = dptmx.max_dpt_sal;

OUTPUT:

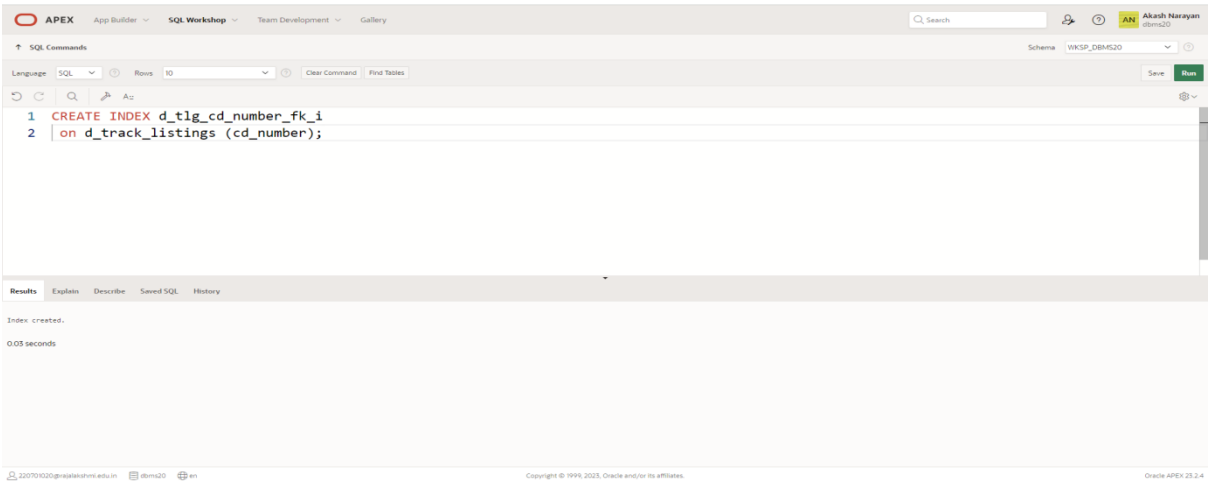


INDEXES AND SYNONYMS

1. 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.

QUERY: CREATE INDEX d_tlg_cd_number_fk_i
on d_track_listings (cd_number);

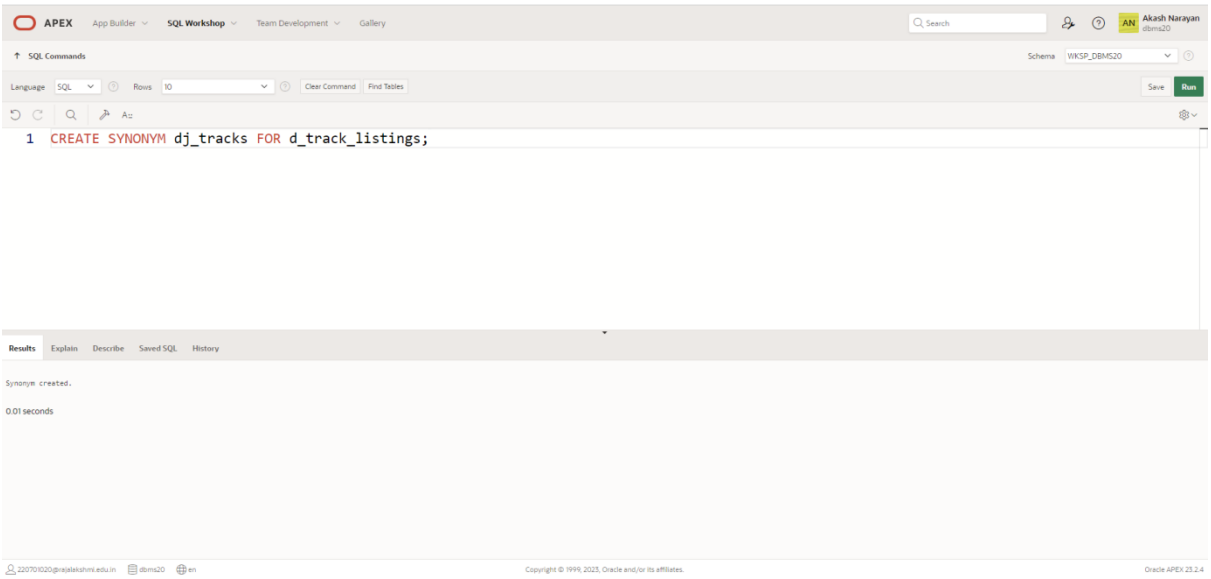
OUTPUT:



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

QUERY: CREATE SYNONYM dj_tracks FOR d_track_listings;

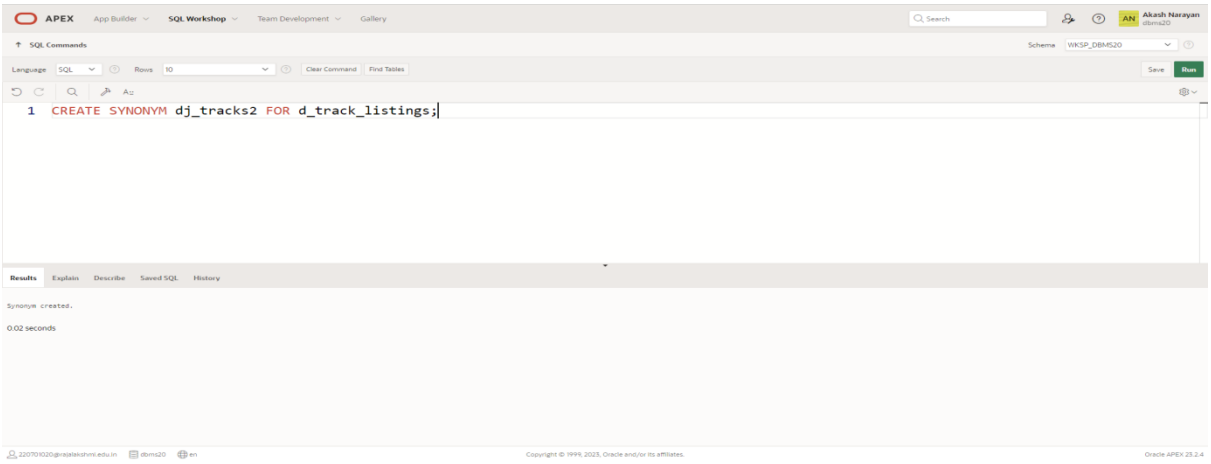
OUTPUT:



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

QUERY: CREATE SYNONYM dj_tracks2 FOR d_track_listings;

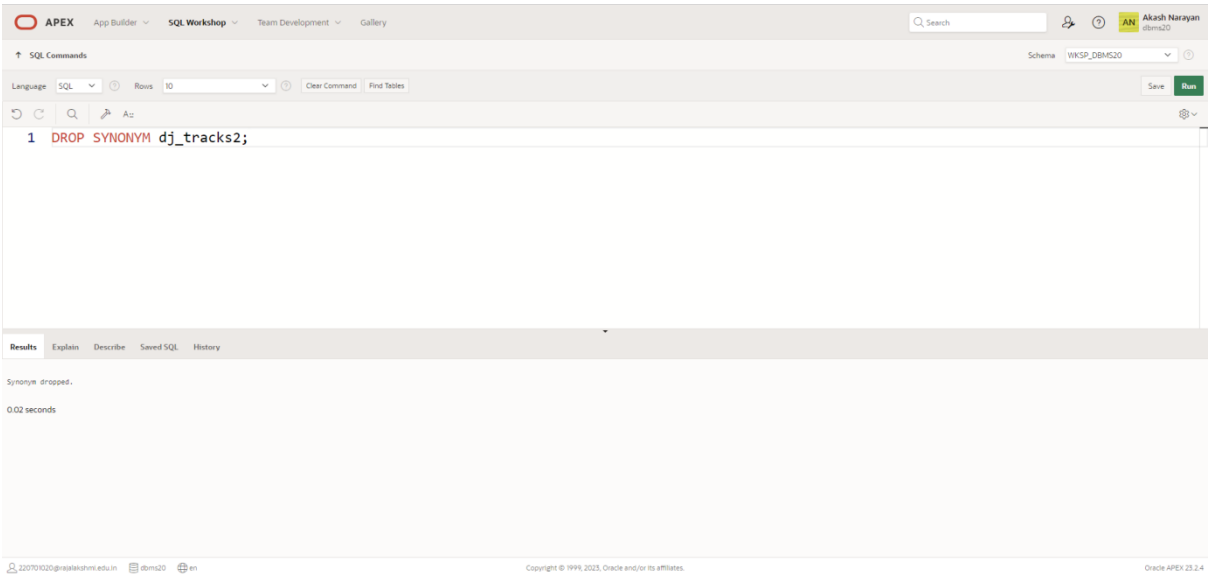
OUTPUT:



4. Drop the synonym that you created in question

QUERY: DROP SYNONYM dj_tracks2;

OUTPUT:



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

RESULT:

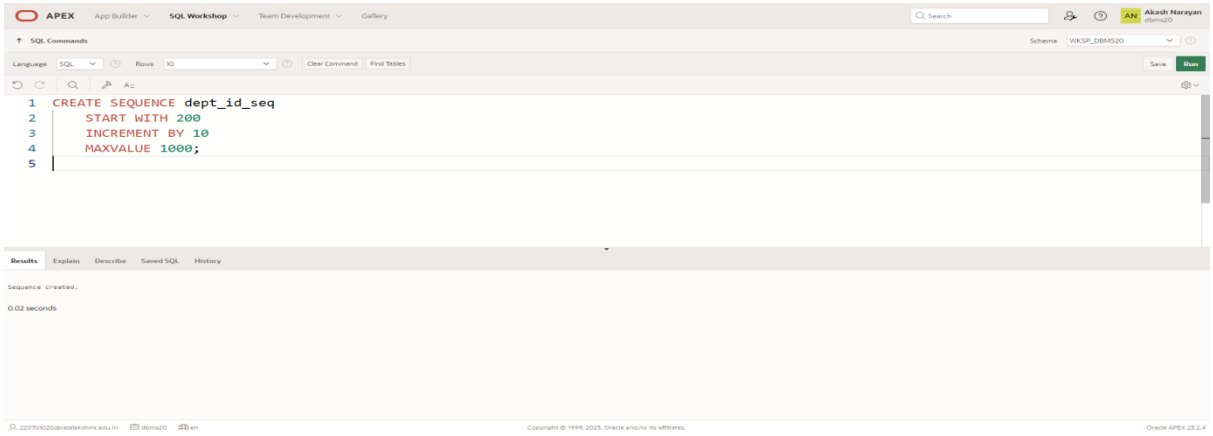
OTHER DATABASE OBJECTS

EXNO: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;

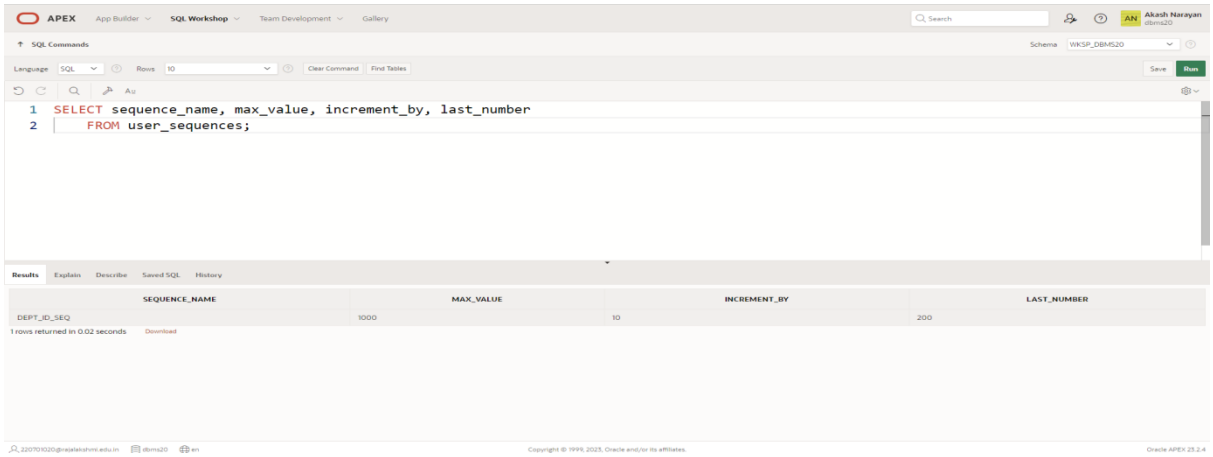
OUTPUT:



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;

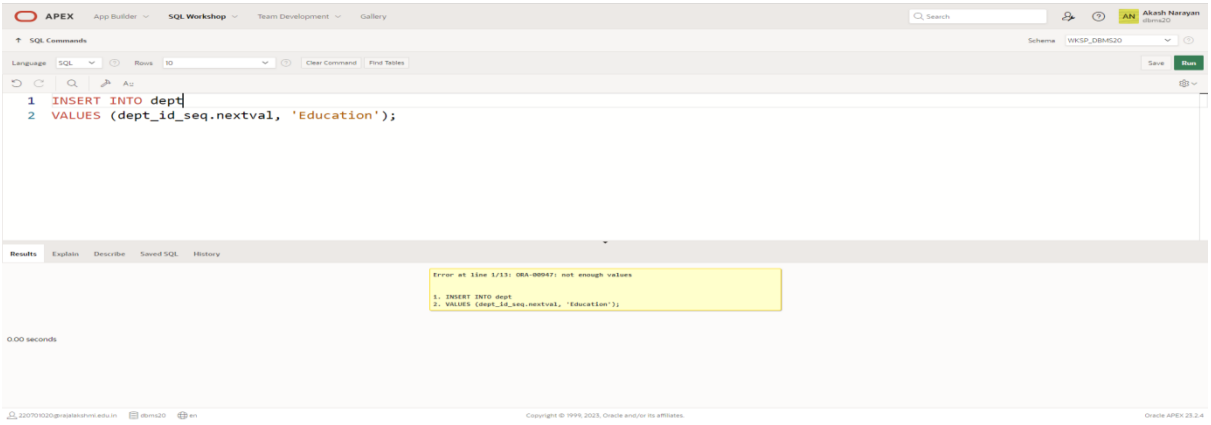
OUTPUT:



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

QUERY: INSERT INTO dept
VALUES (dept_id_seq.nextval, 'Education');
INSERT INTO dept
VALUES (dept_id_seq.nextval, 'Administration');

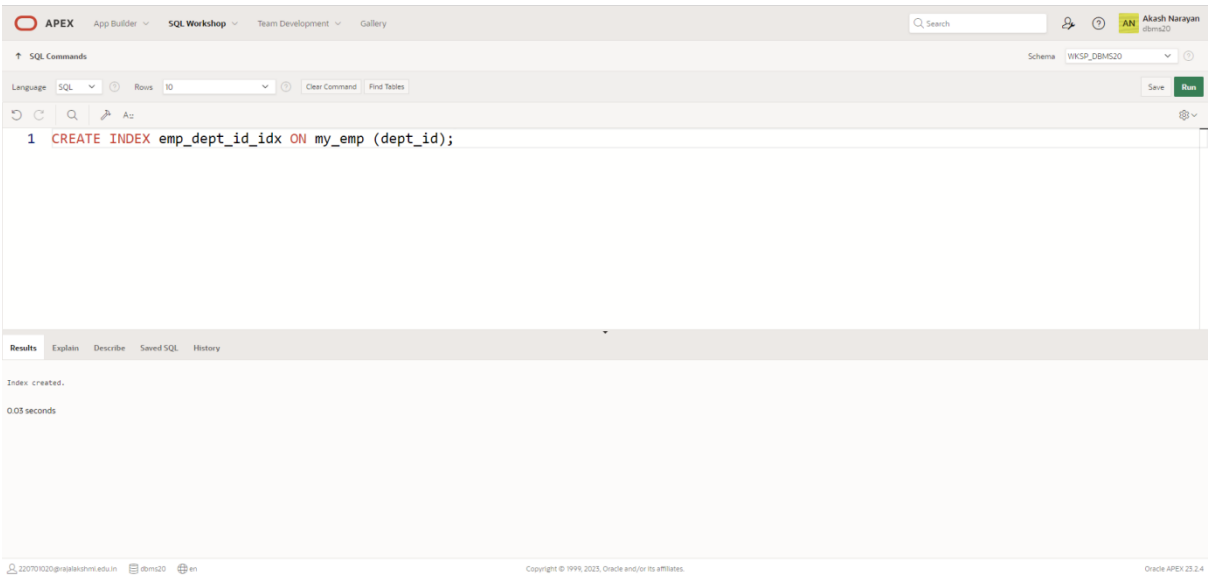
OUTPUT:



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

QUERY: CREATE INDEX emp_dept_id_idx ON my_emp (dept_id);

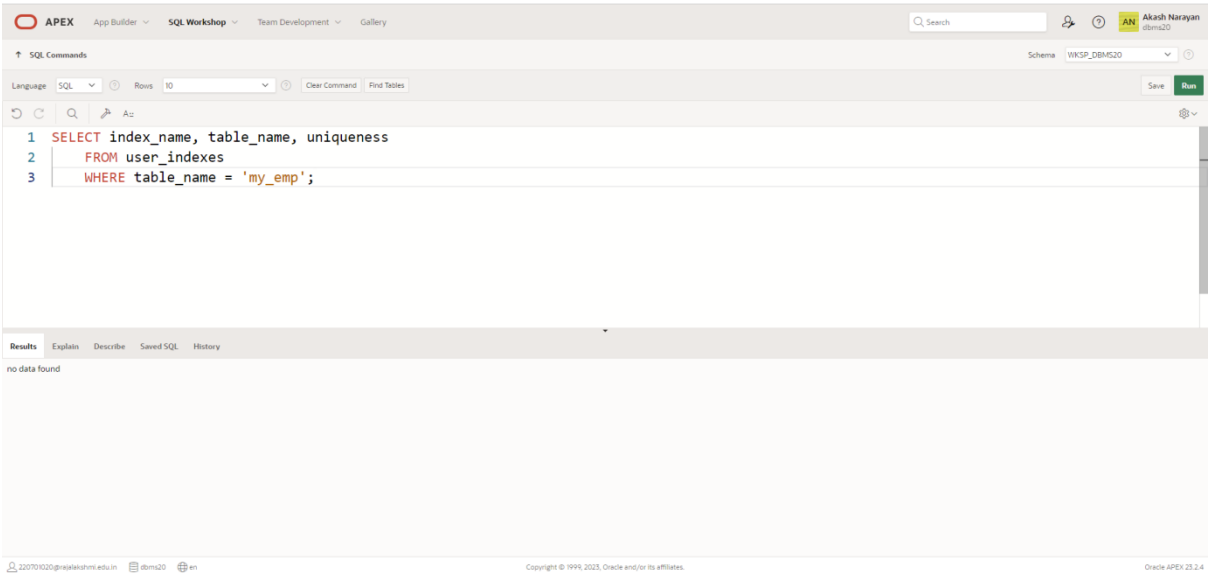
OUTPUT:



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 = 'my_emp';

OUTPUT:



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

RESULT:

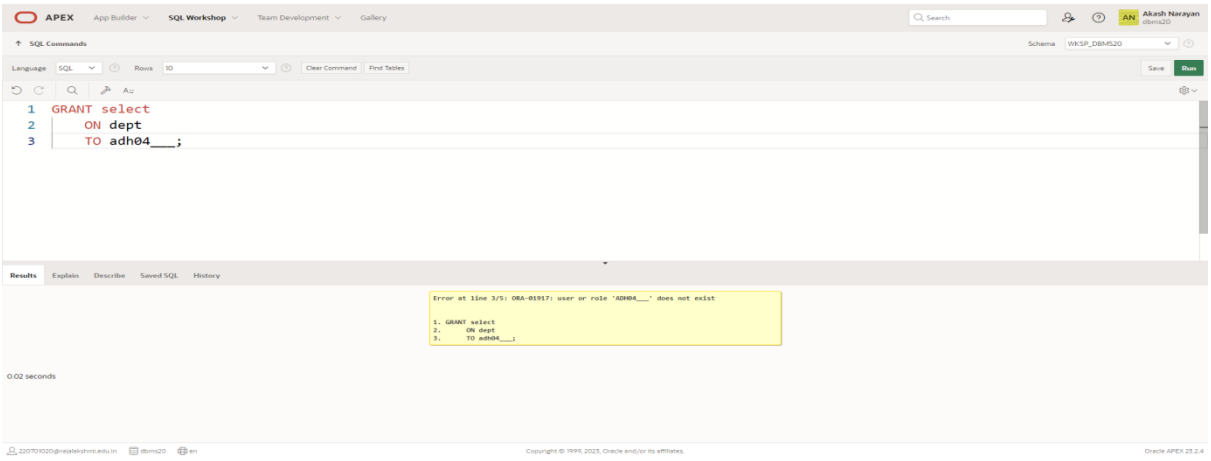
CONTROLLING USER ACCESS

EXNO:15 **DATE:**

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

QUERY: GRANT select
ON departments
TO <adh04___>;

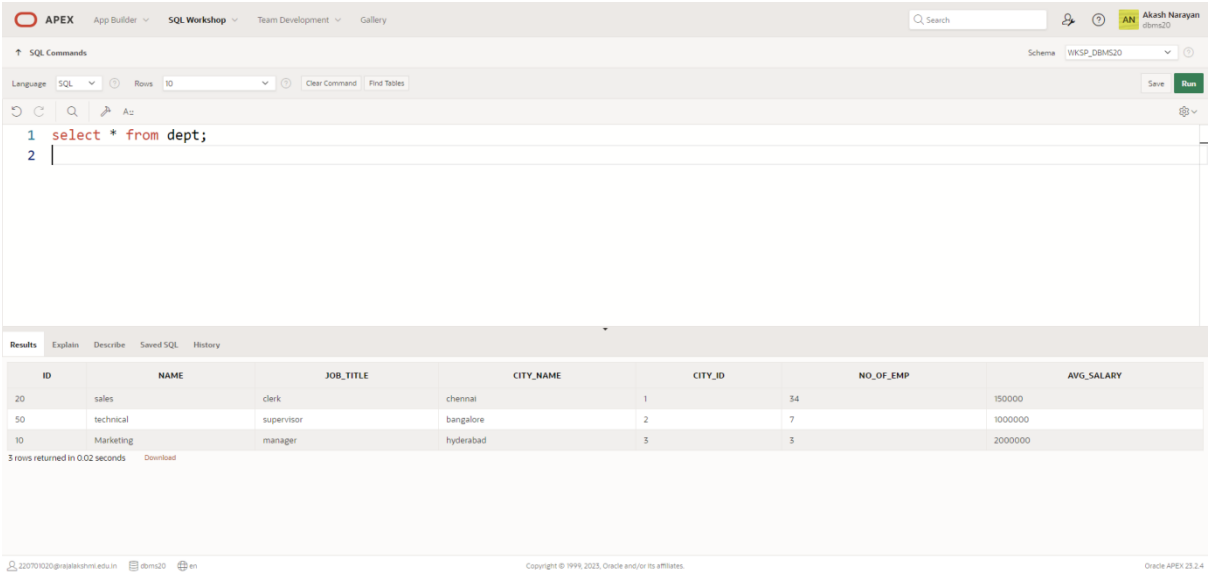
OUTPUT:



2. Query all the rows in your DEPARTMENTS table.

QUERY: select * from dept;

OUTPUT:



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

RESULT:

PL/SQL
CONTROL STRUCTURES

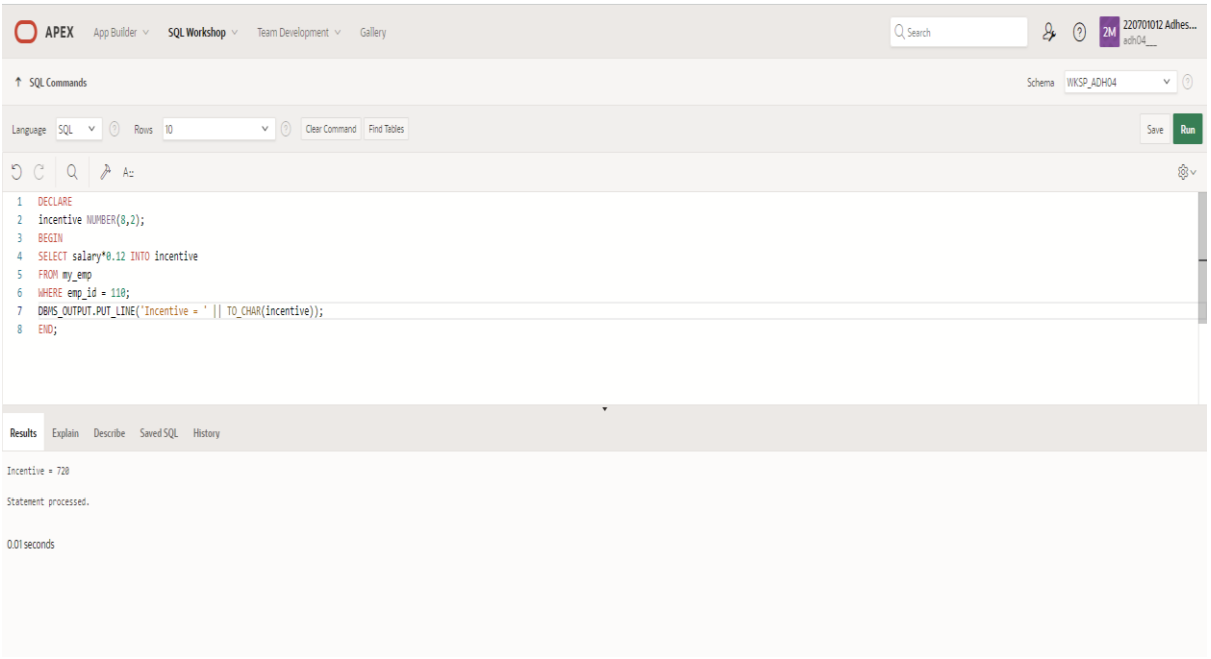
EXNO:16 **DATE:**

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;
```

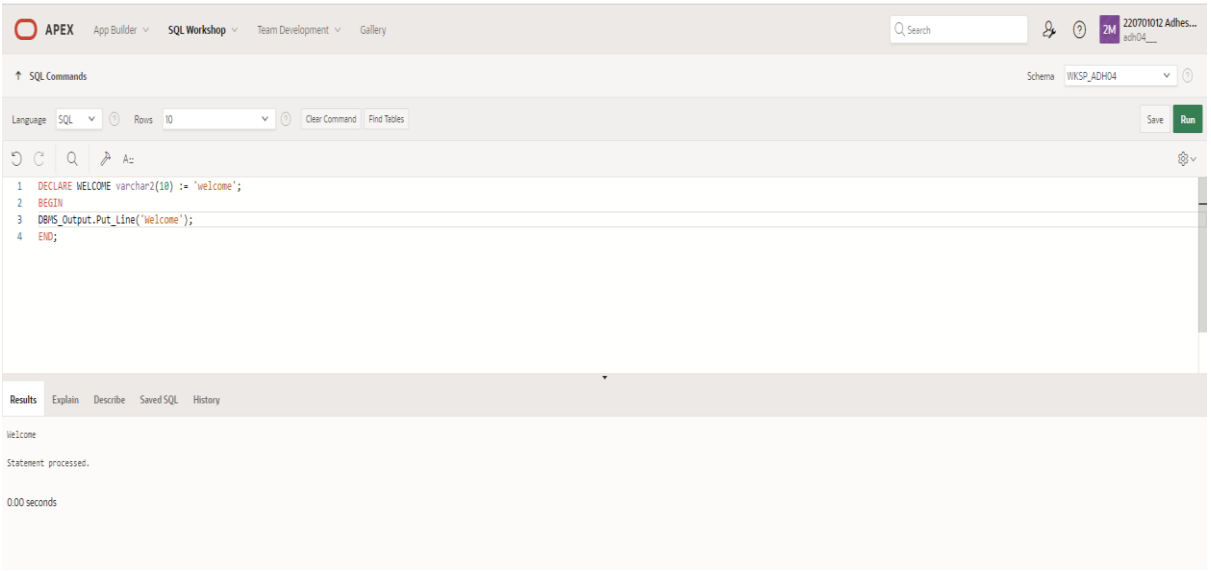
OUTPUT:



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:

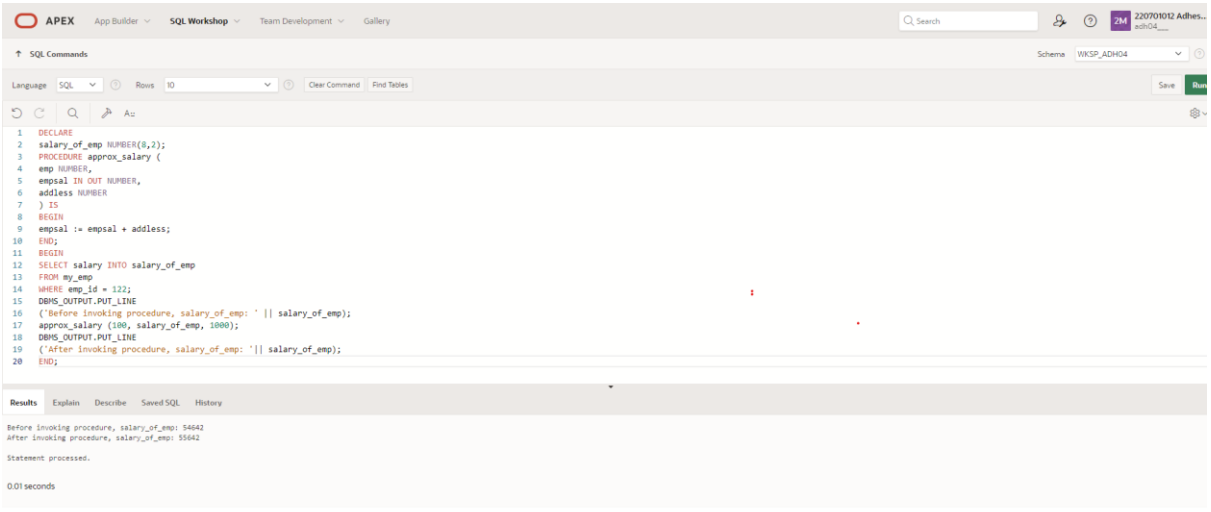


3. Write a PL/SQL block to adjust the salary of the employee whose ID 122.

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:

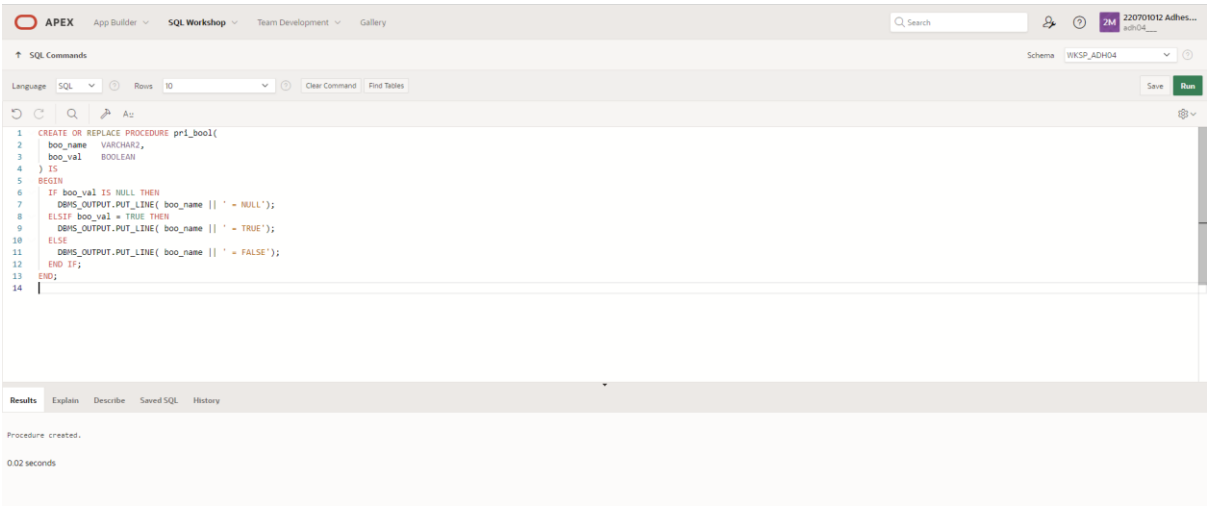


4. Write a PL/SQL block to create a procedure using the "IS [NOT] NULL Operator" and show AND operator returns TRUE if and only if both operands are TRUE.

QUERY:

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

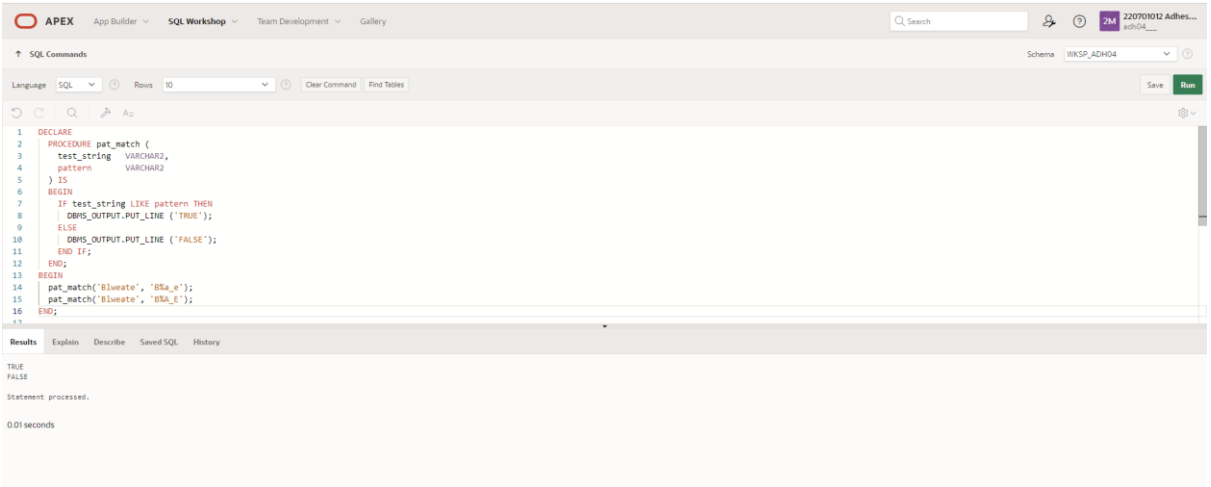
OUTPUT:



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:



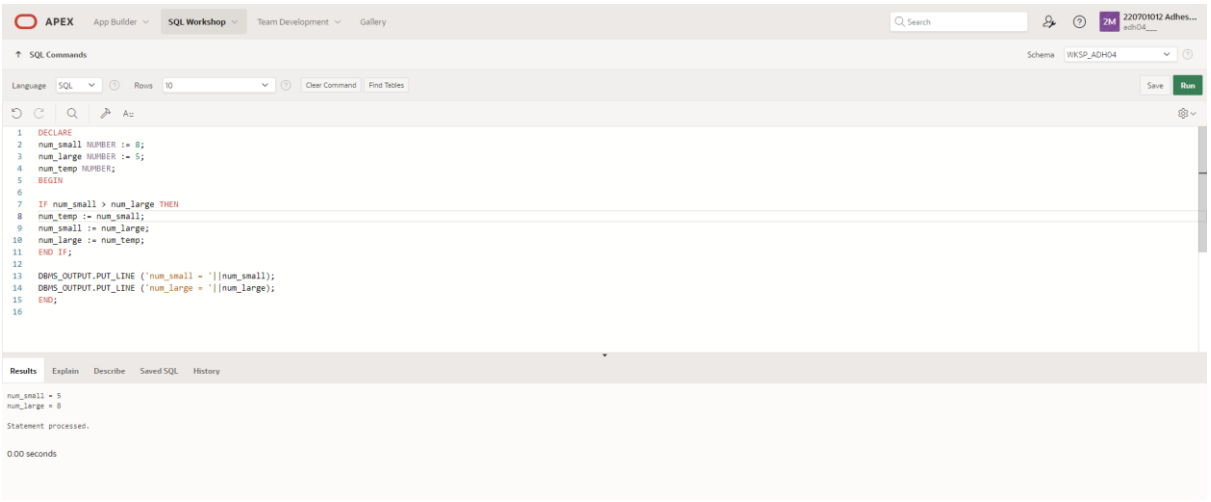
6. Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num_small variable and large number will store in num_large variable

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:

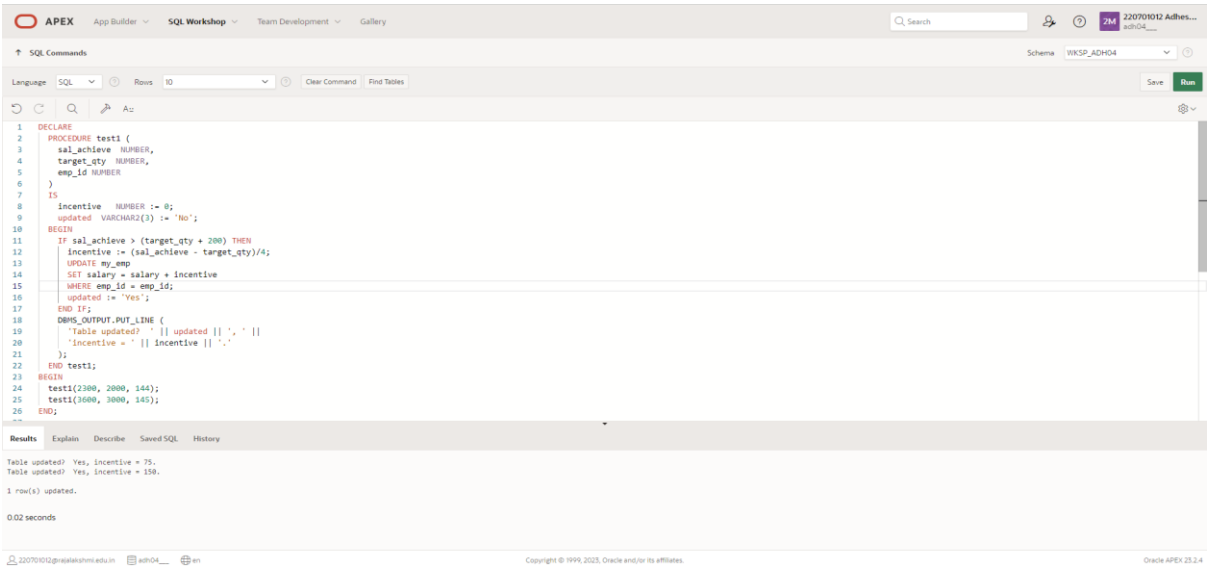


7. Write a PL/SQL procedure to calculate the incentive on a target achieved and display the message either the record updated or not.

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:

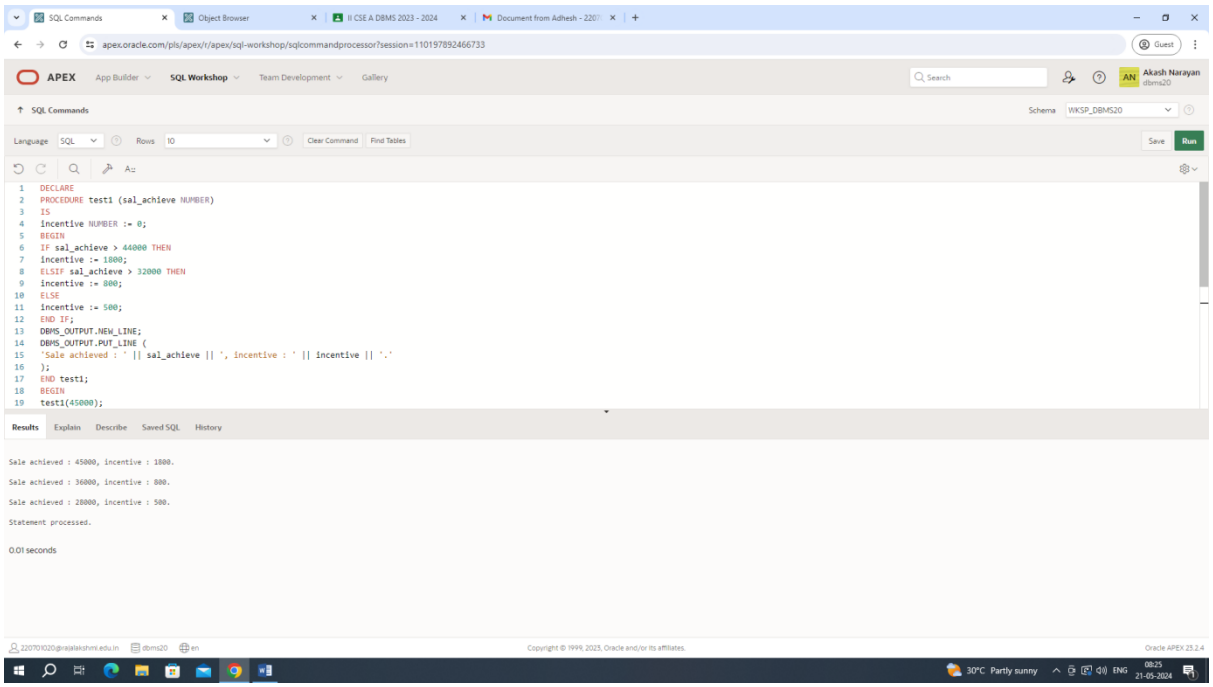


8. Write a PL/SQL procedure to calculate incentive achieved according to the specific sale limit

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:

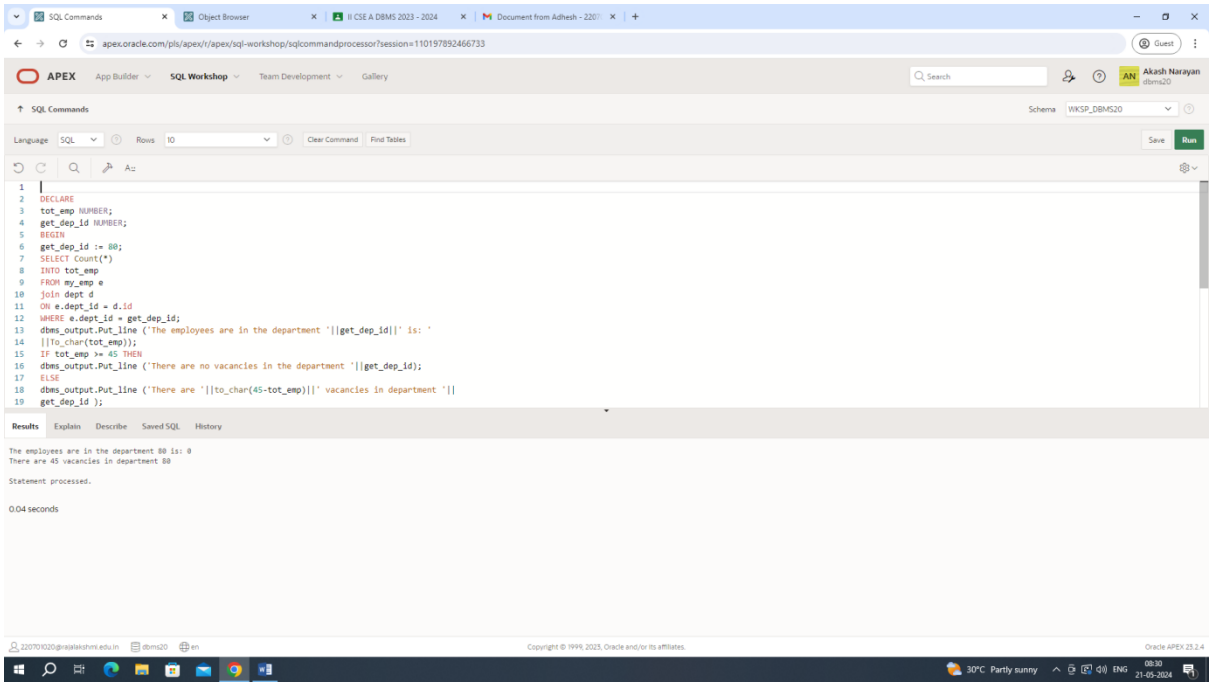


9. Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

QUERY:

```
DECLARE
   tot_emp NUMBER;
      get_dep_id NUMBER;

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

OUTPUT:

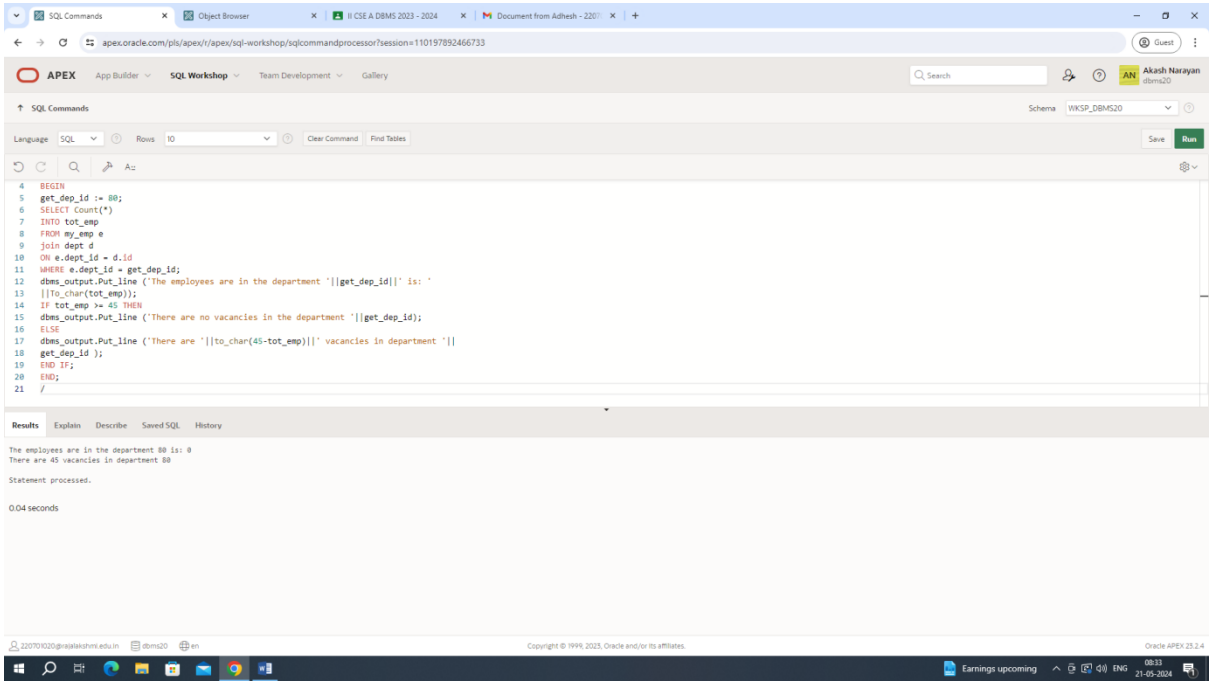


10. Write a PL/SQL program to count number of employees in a specific department and check whether this department have any vacancies or not. If any vacancies, how many vacancies are in that department.

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:

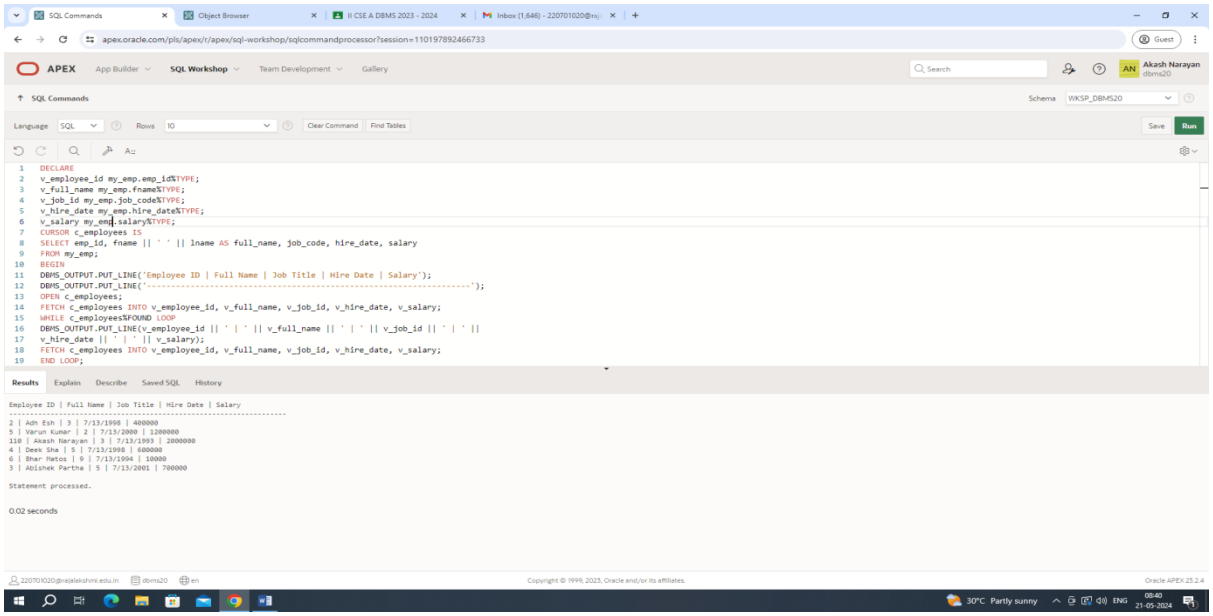


11. Write a PL/SQL program to display the employee IDs, names, job titles, hire dates, and salaries of all employees

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:

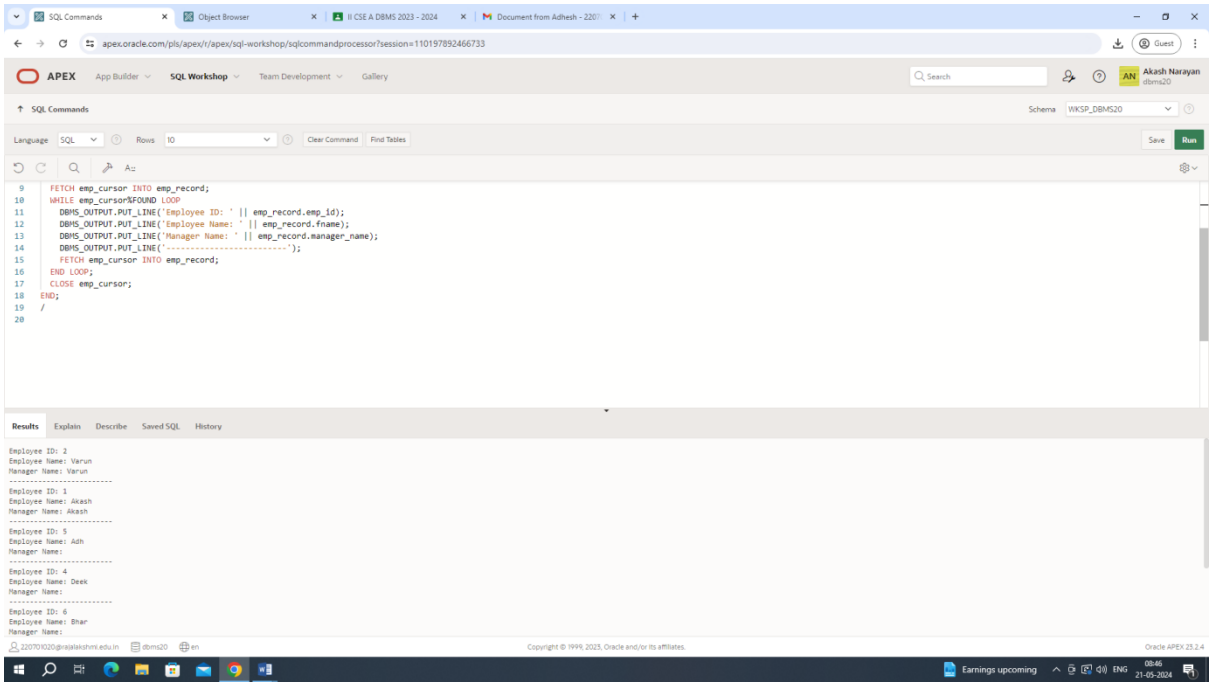


12. Write a PL/SQL program to display the employee IDs, names, and department names of all employees.

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:

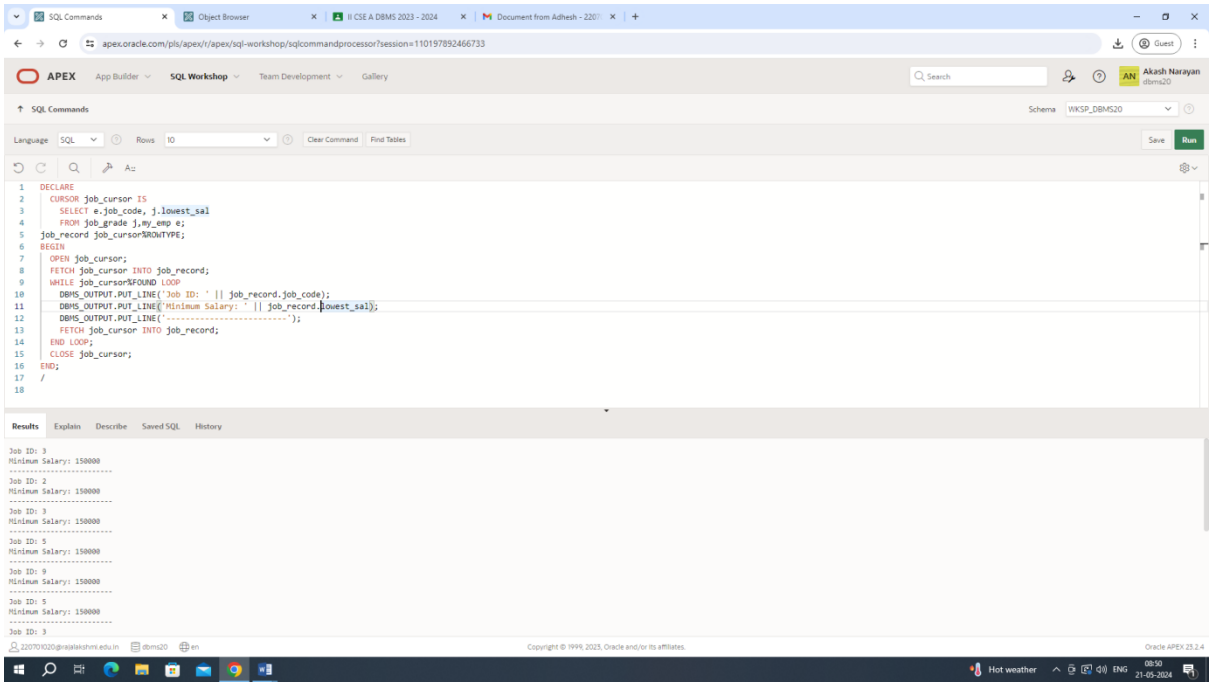


13. Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all jobs

QUERY:

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

OUTPUT:

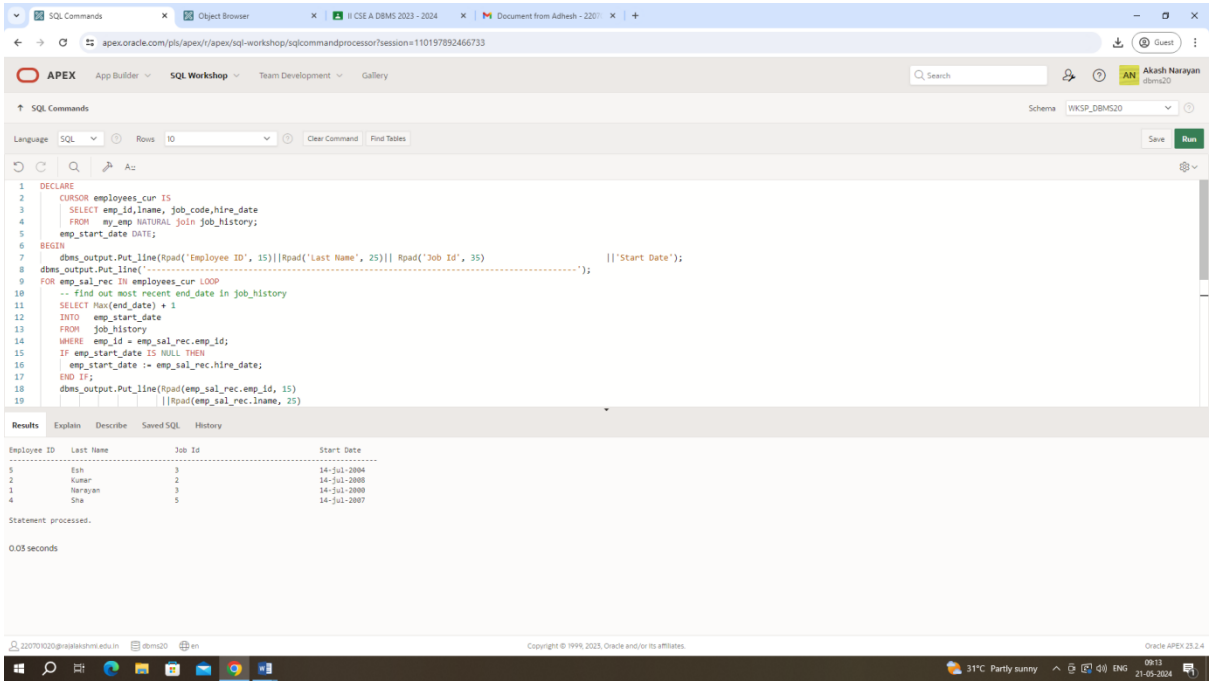


14. Write a PL/SQL program to display the employee IDs, names, and job history start dates of all employees.

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:

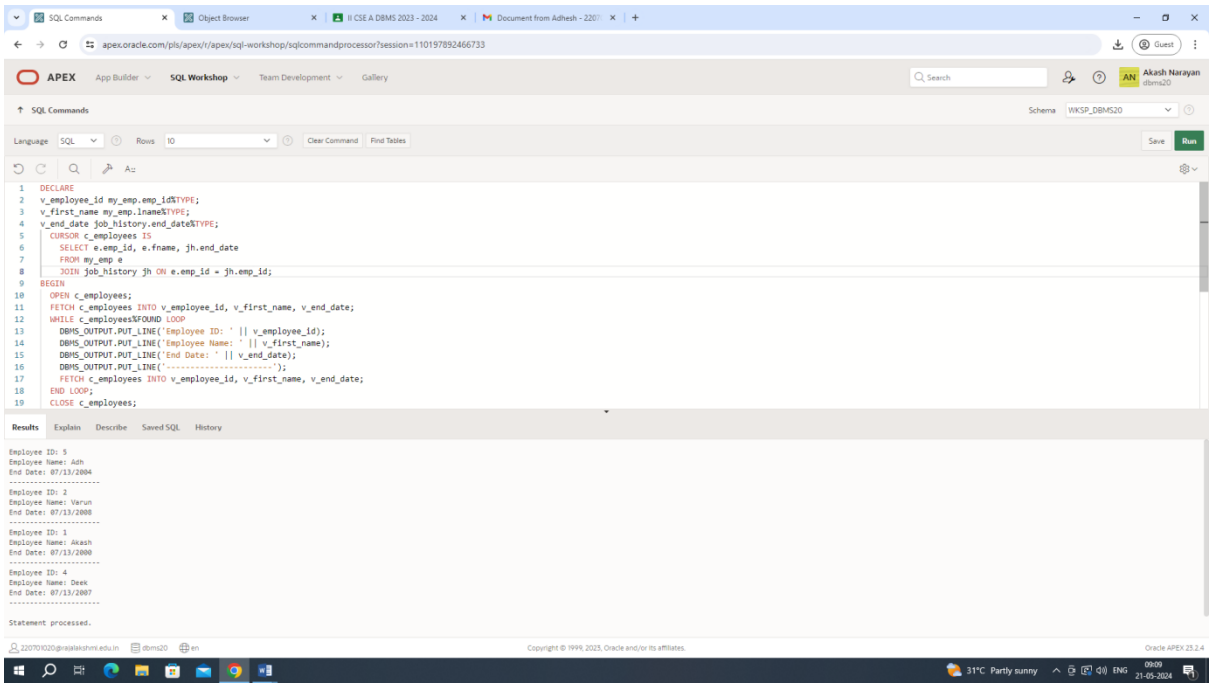


15. Write a PL/SQL program to display the employee IDs, names, and job history end dates of all employees.

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:



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

RESULT:

PROCEDURES AND FUNCTIONS

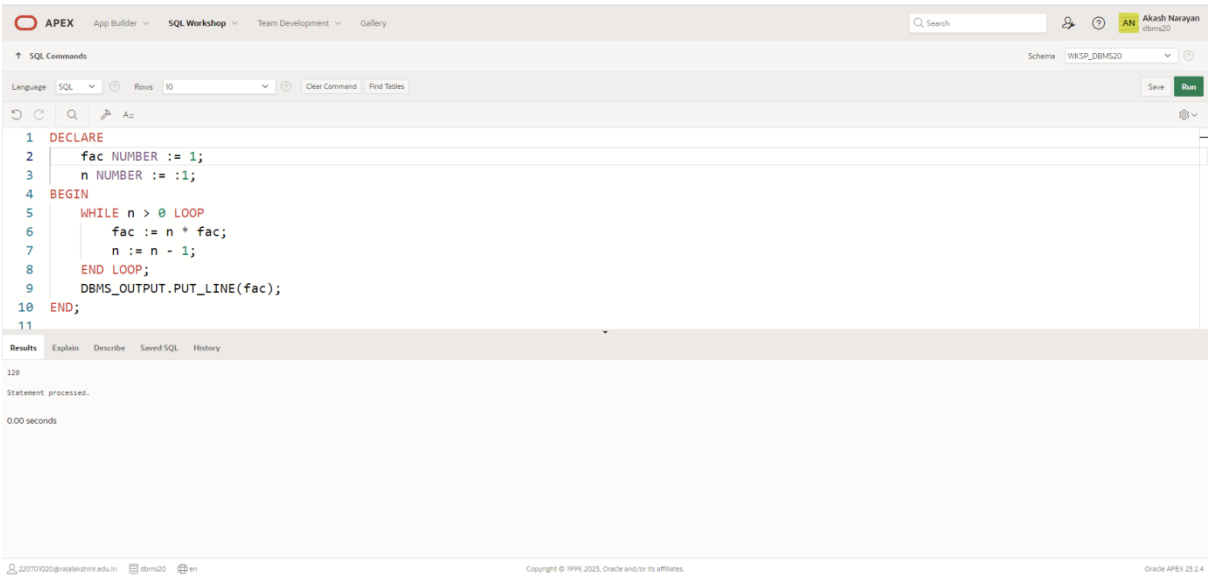
EXNO:17 **DATE:**

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;
```

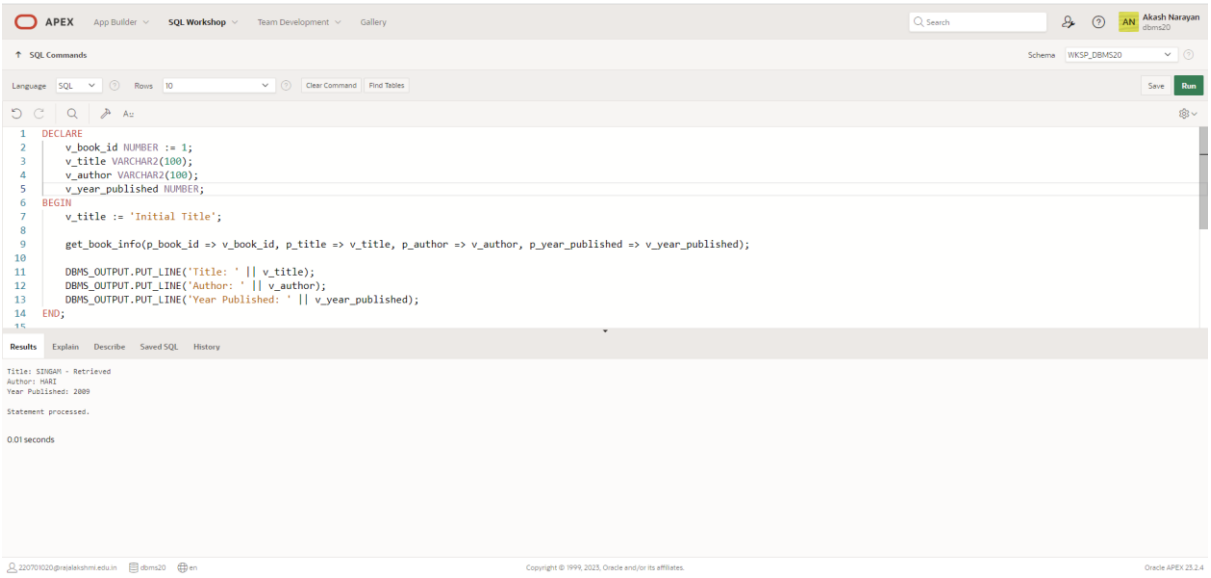
OUTPUT:



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:



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

RESULT:

TRIGGER

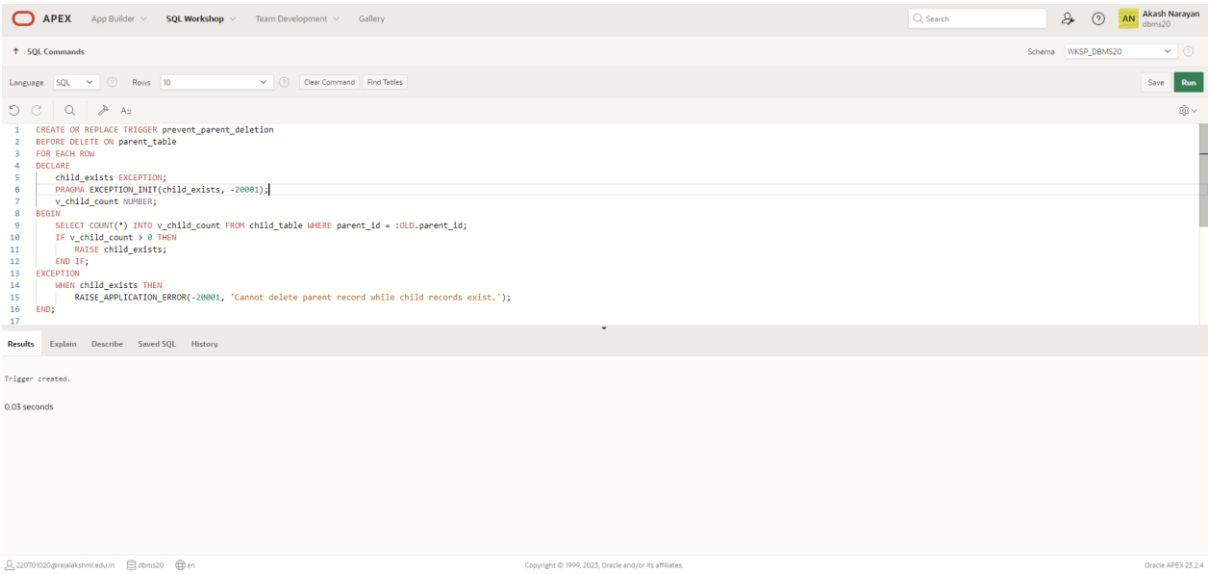
EXNO:18 **DATE:**

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

QUERY:

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

OUTPUT:

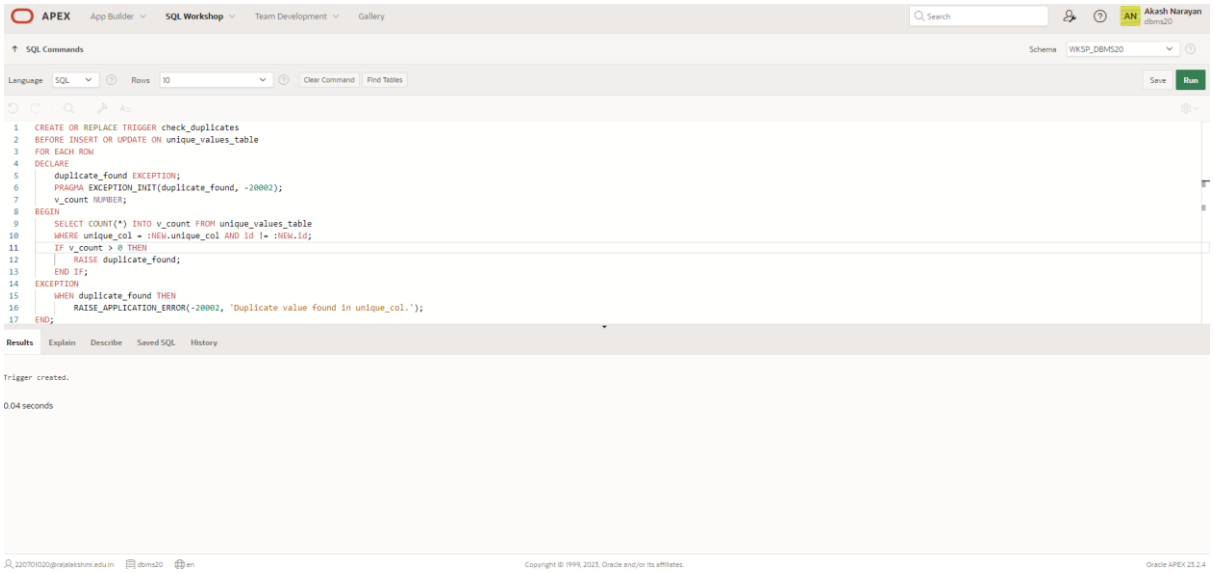


2. Write a code in PL/SQL to create a trigger that checks for duplicate values in a specific column and raises an exception if found

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:

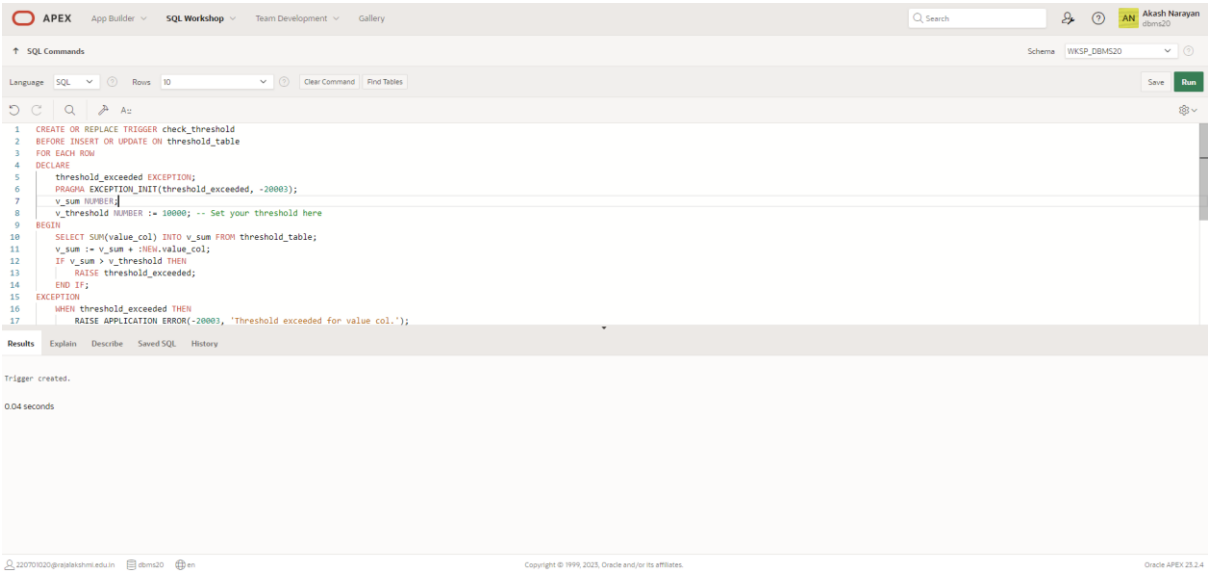


3. Write a code in PL/SQL to create a trigger that restricts the insertion of new rows if the total of a column's values exceeds a certain threshold

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;
```

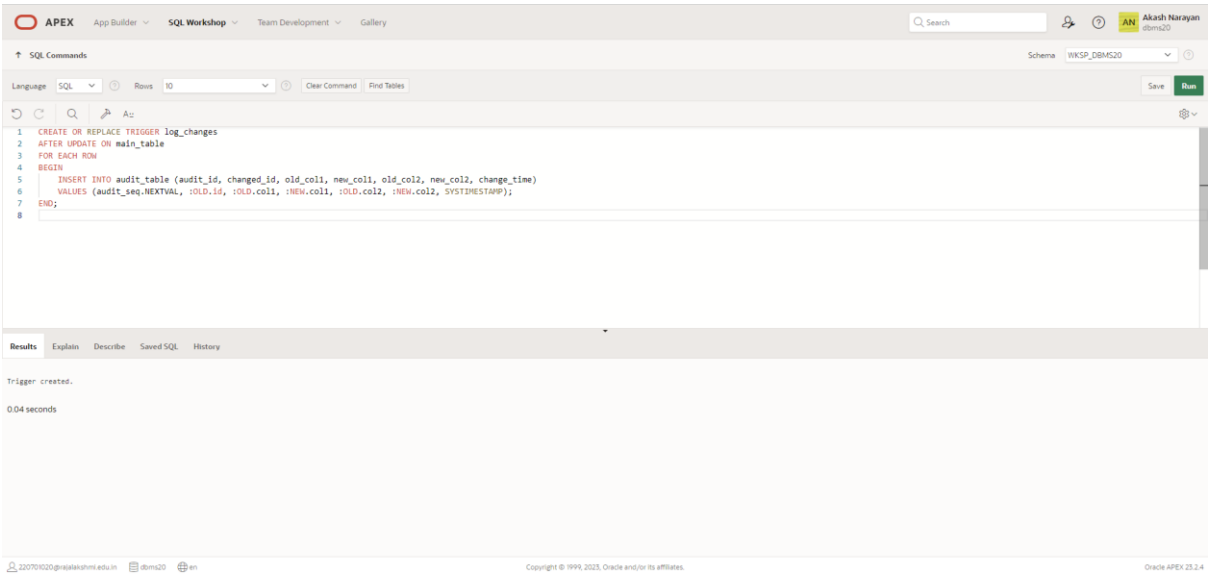
OUTPUT:



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:

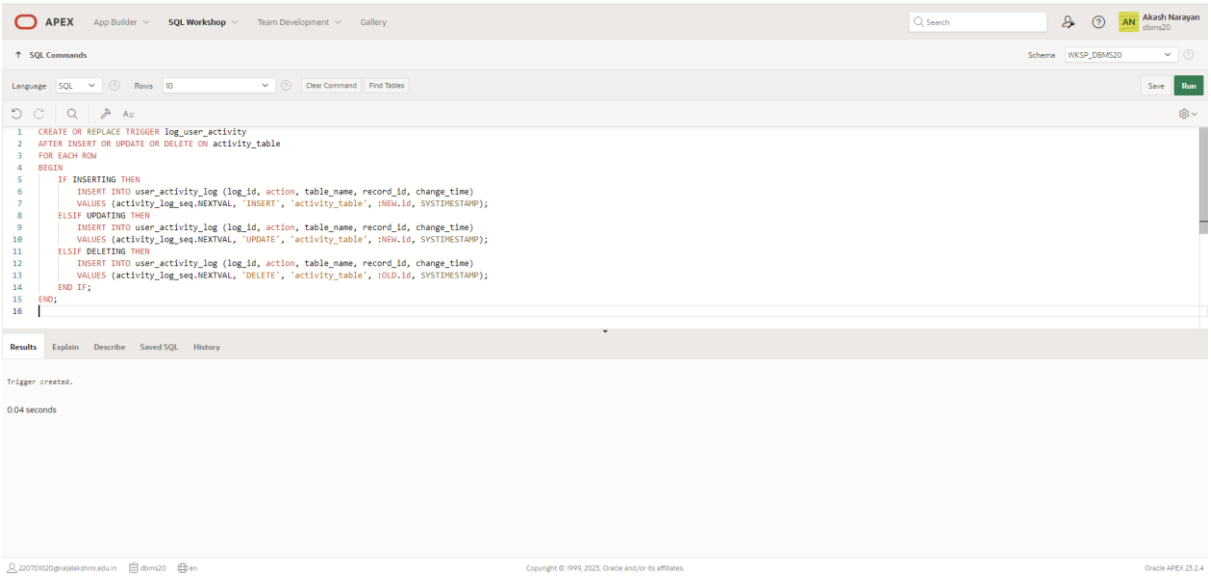


5. Write a code in PL/SQL to implement a trigger that records user activity (inserts, updates, deletes) in an audit log for a given set of tables.

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:

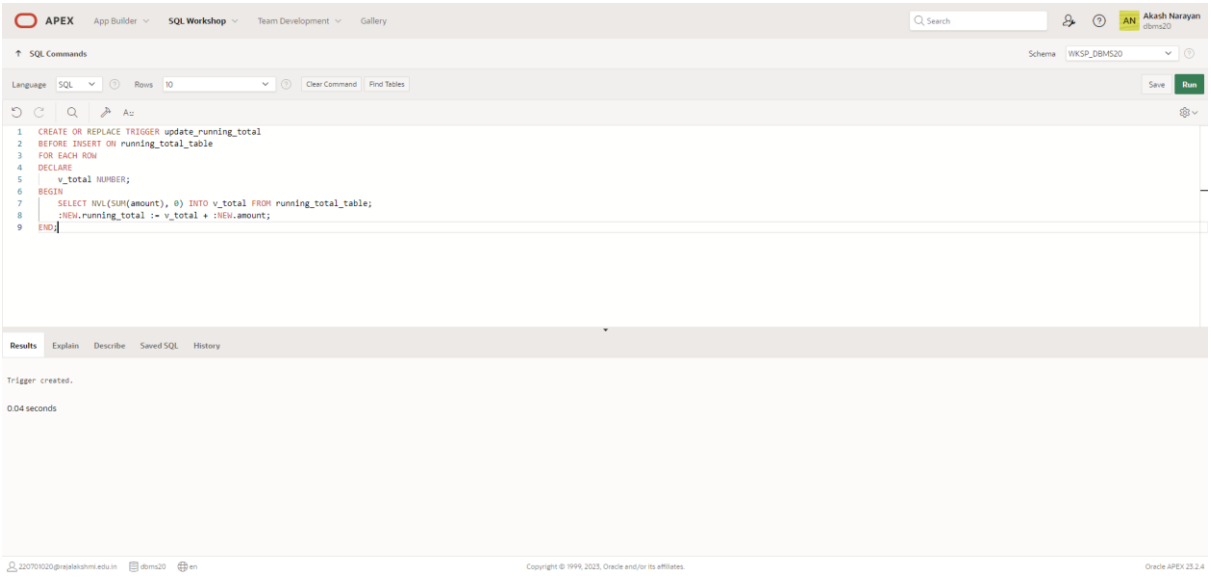


6. Write a code in PL/SQL to implement a trigger that automatically calculates and updates a running total column for a table whenever new rows are inserted.

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:

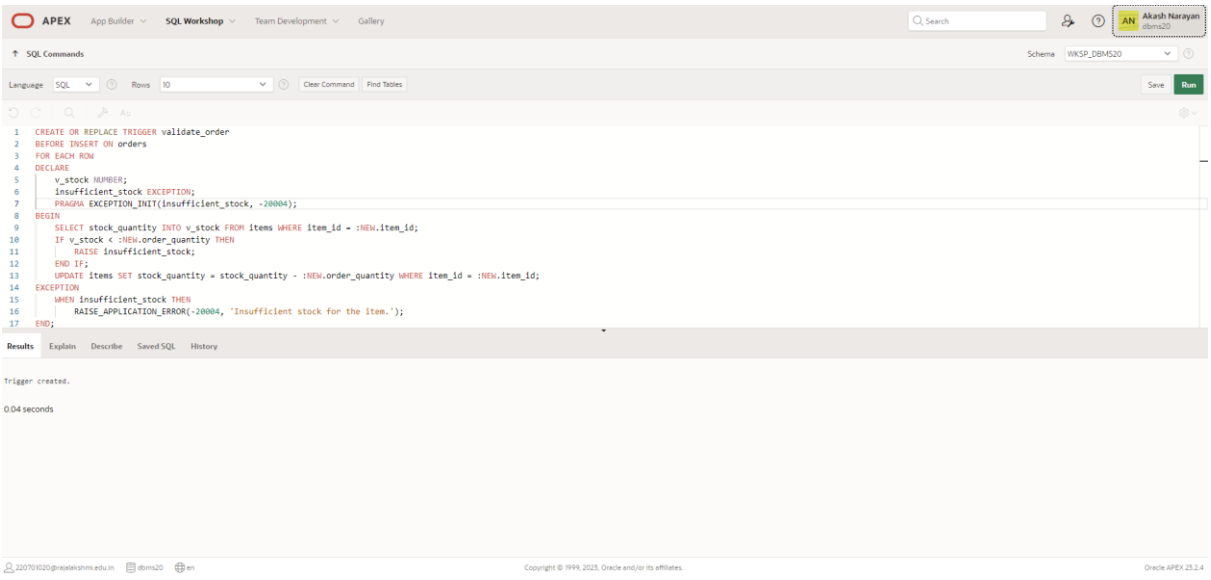


7. Write a code in PL/SQL to create a trigger that validates the availability of items before allowing an order to be placed, considering stock levels and pending orders

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:



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

RESULT:

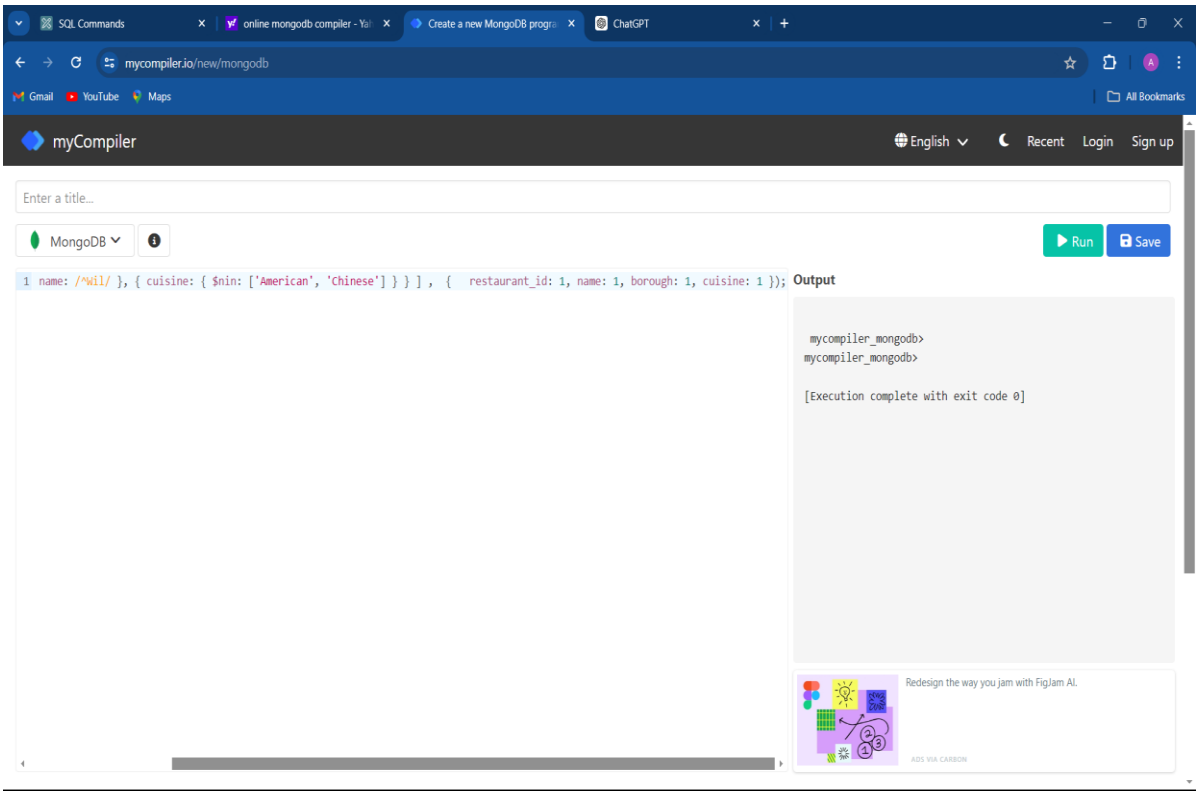
MONGO DB

EXNO: 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 });

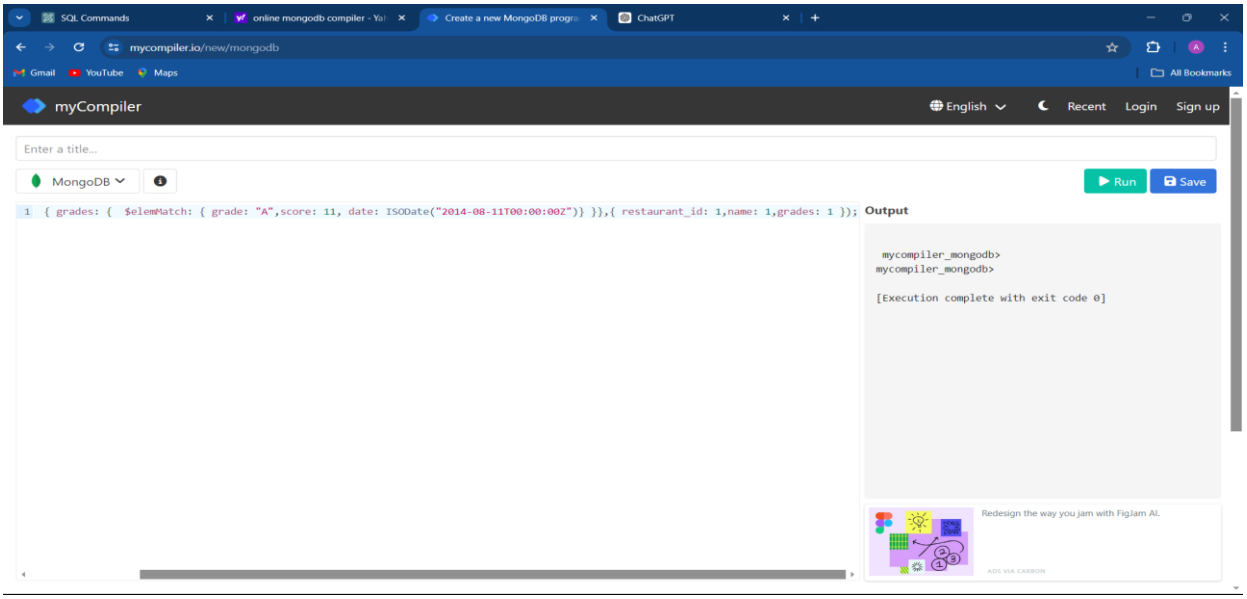
OUTPUT:



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 });

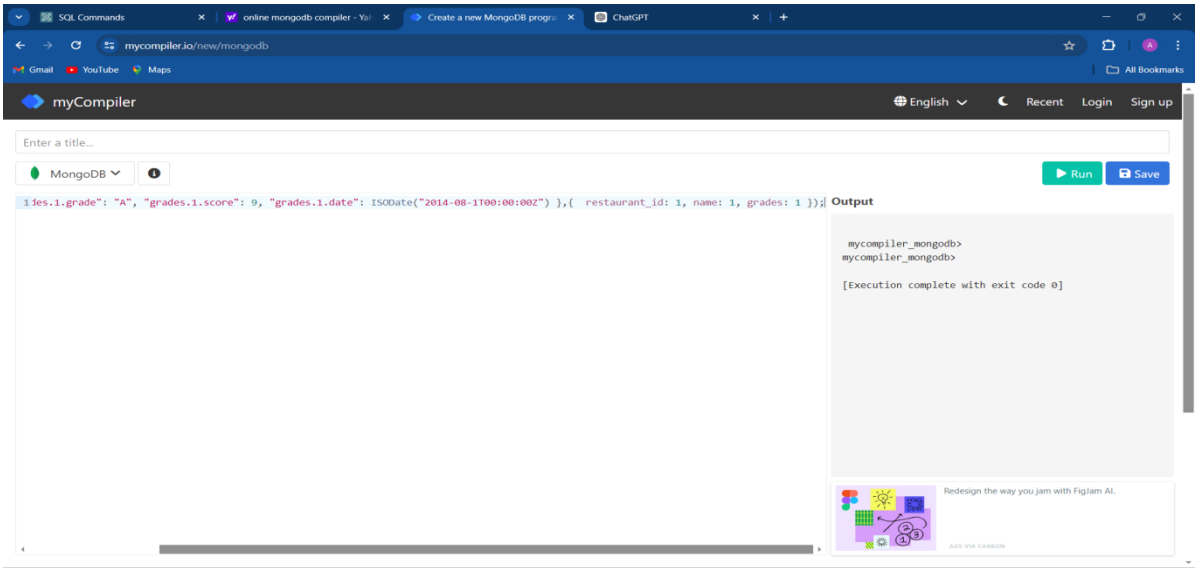
OUTPUT:



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

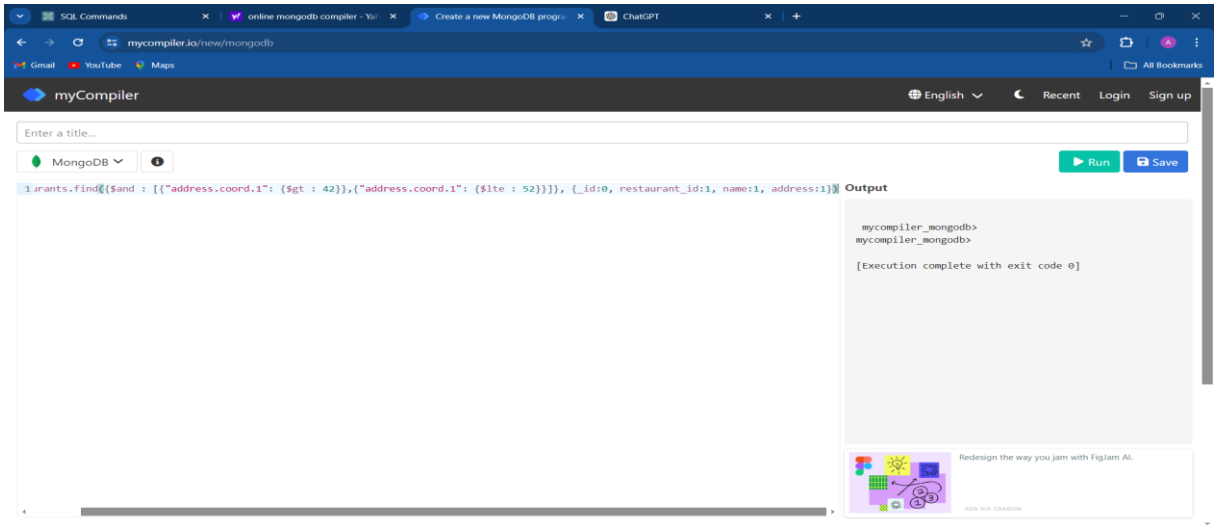
OUTPUT:



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})

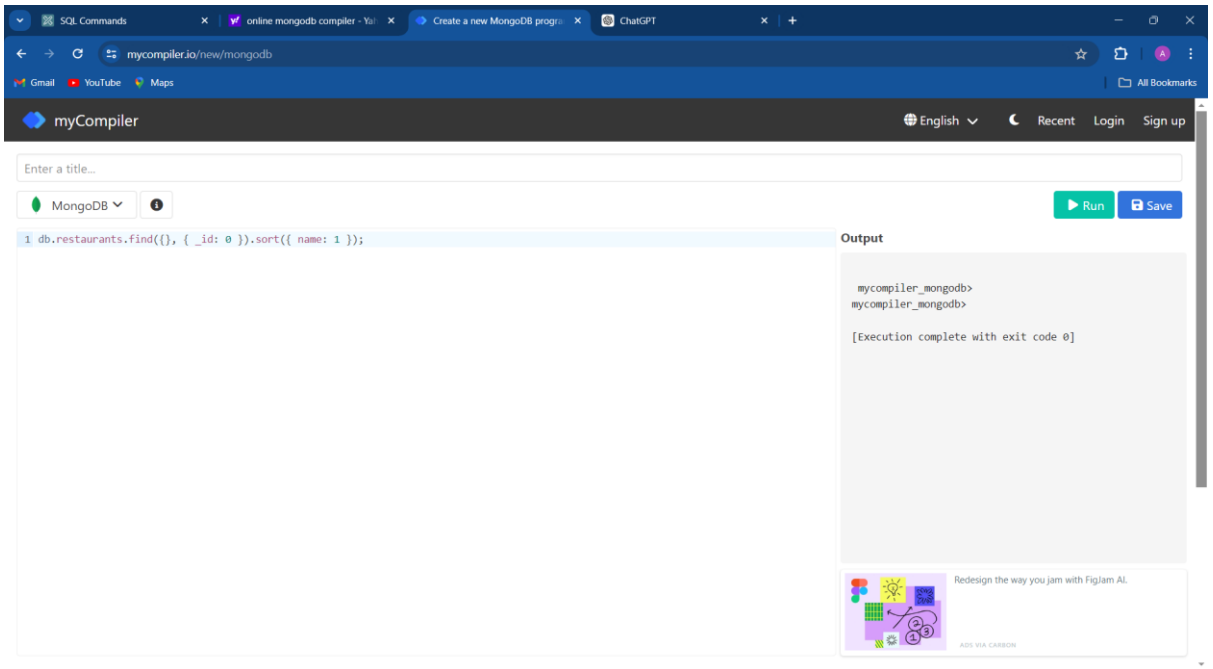
OUTPUT:



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 });

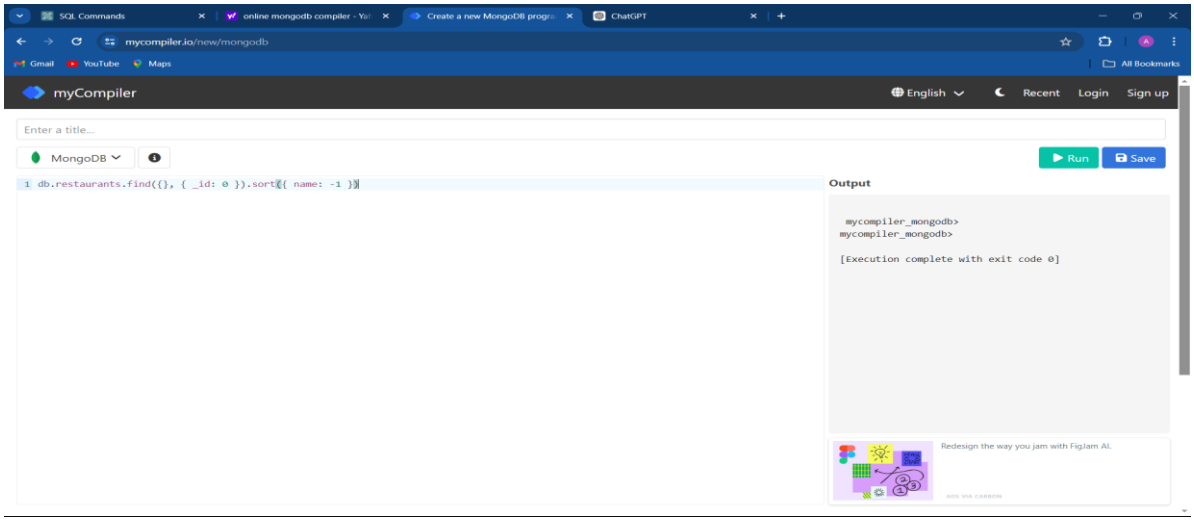
OUTPUT:



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 })

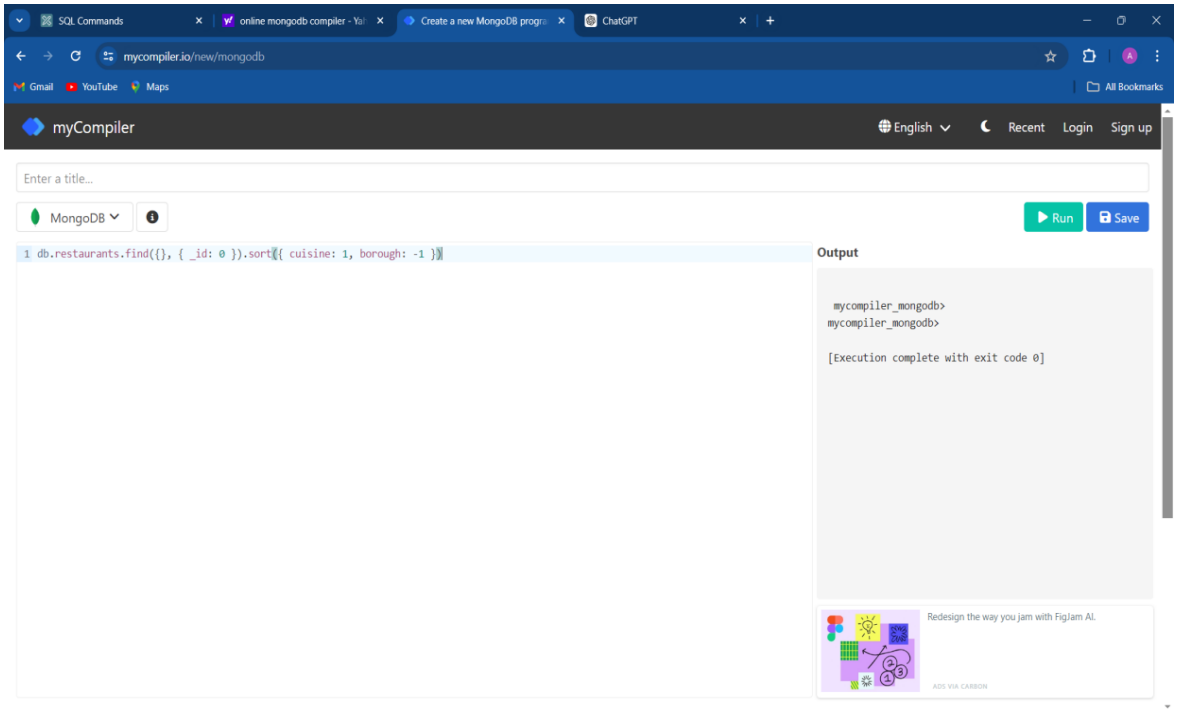
OUTPUT:



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 })

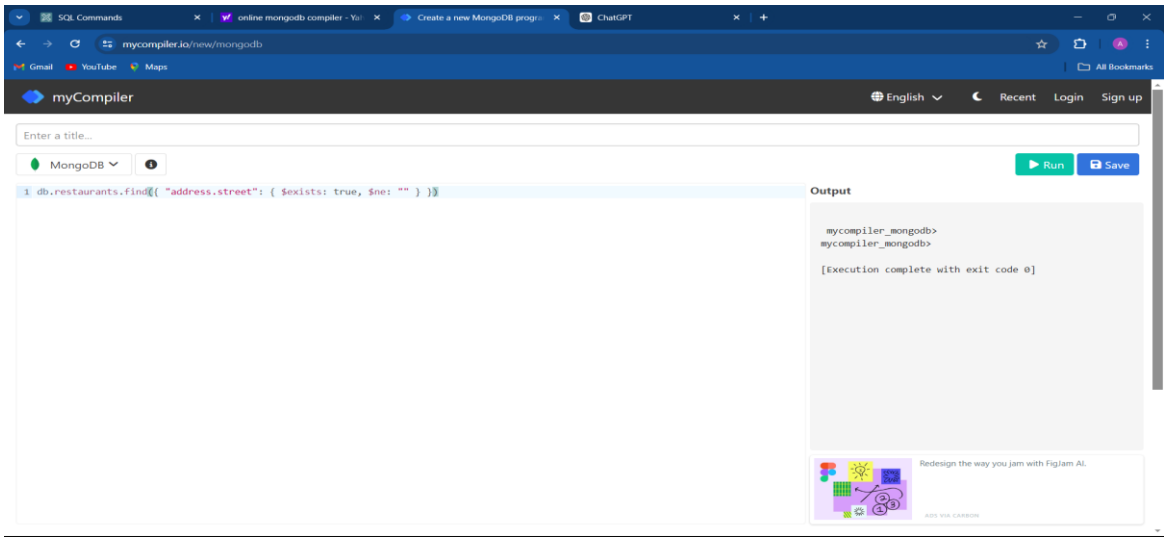
OUTPUT:



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: "" } })

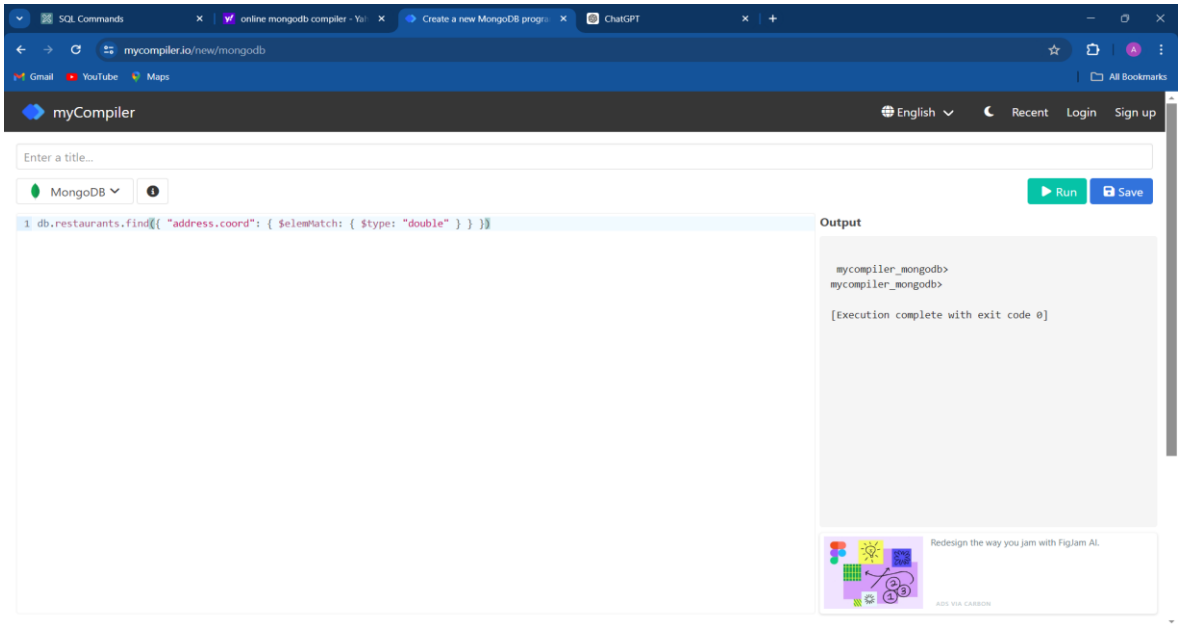
OUTPUT:



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:

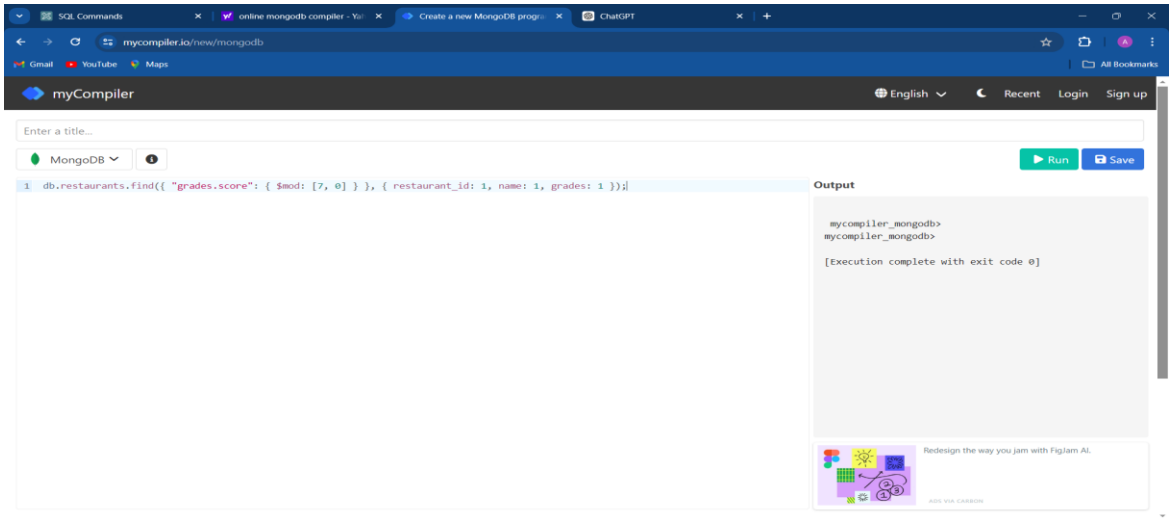


---

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 });

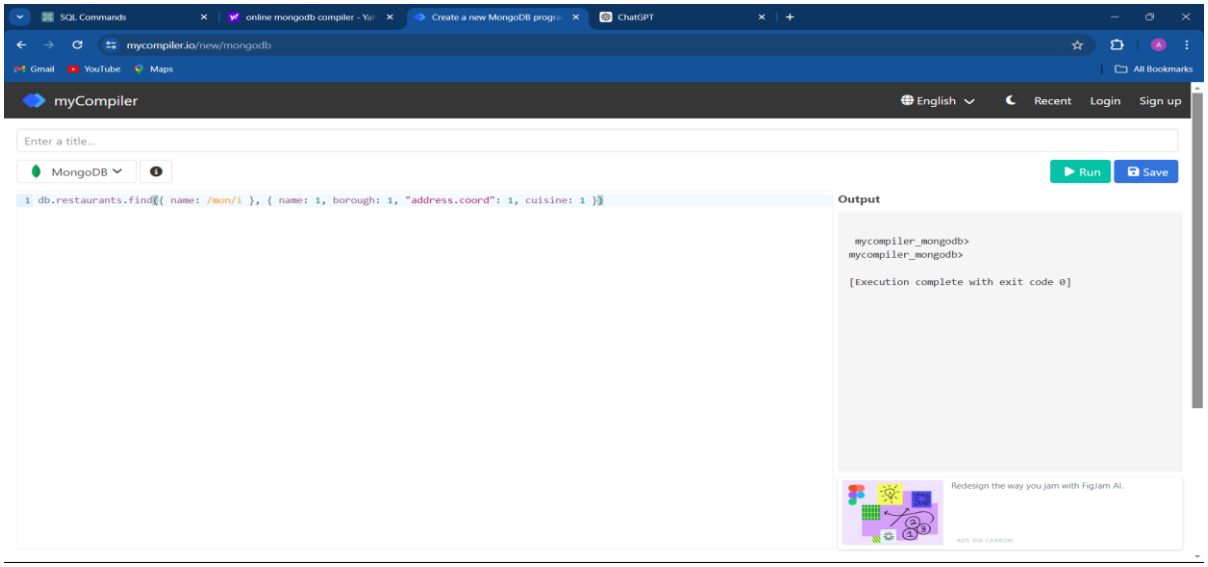
OUTPUT:



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 })

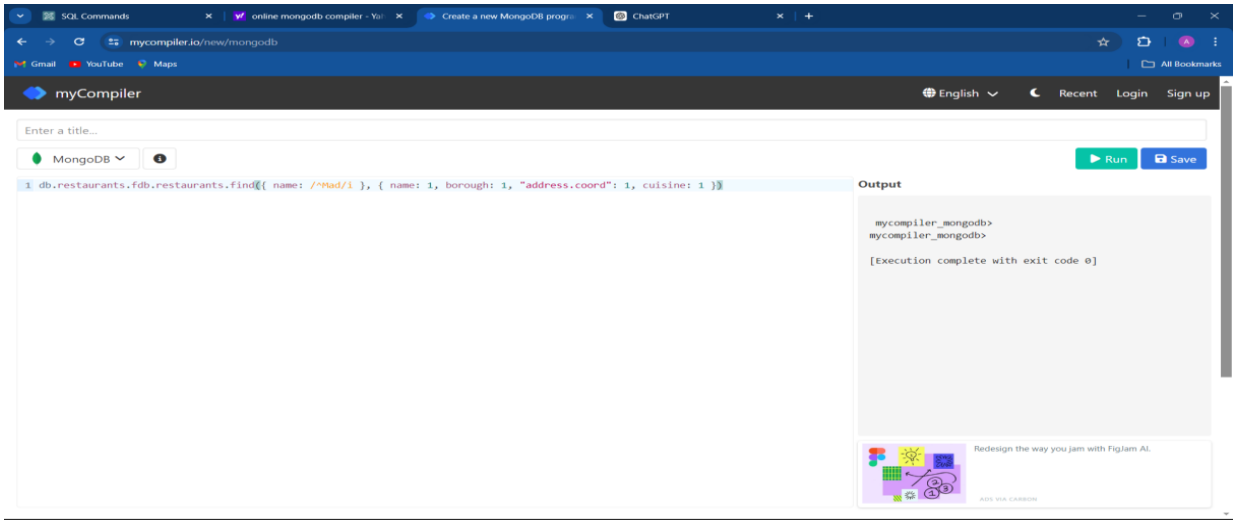
OUTPUT:



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 })

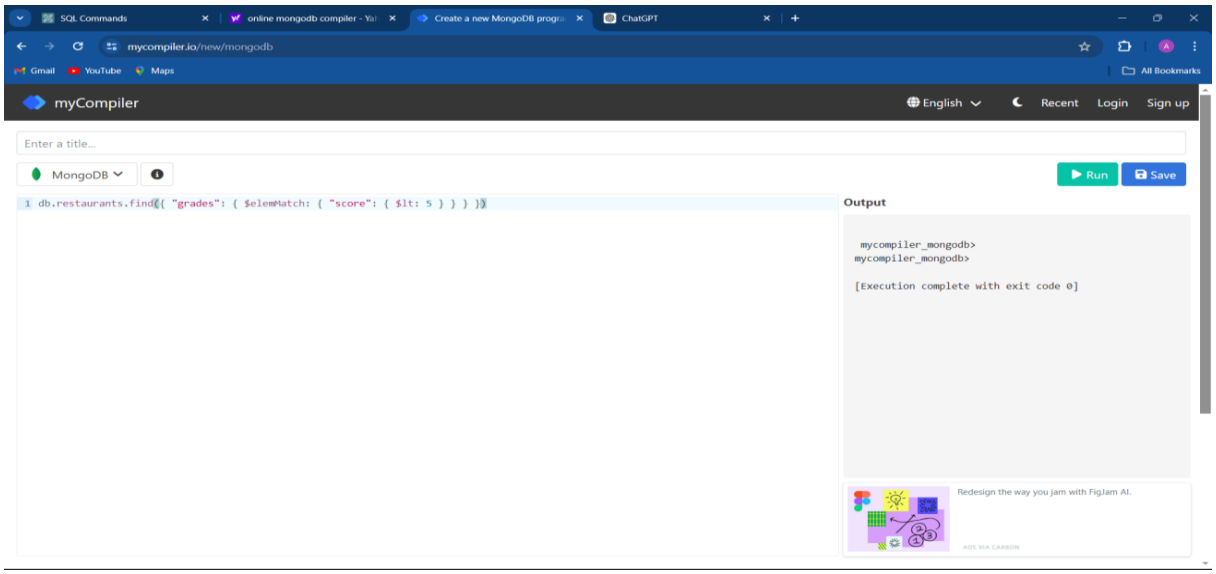
OUTPUT:



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 } } } })

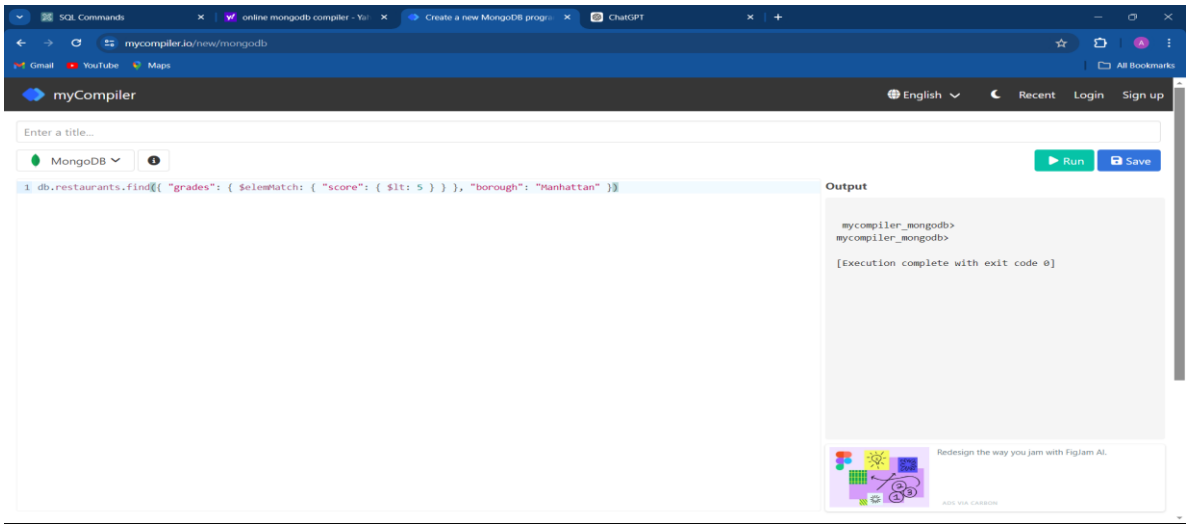
OUTPUT:



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" })

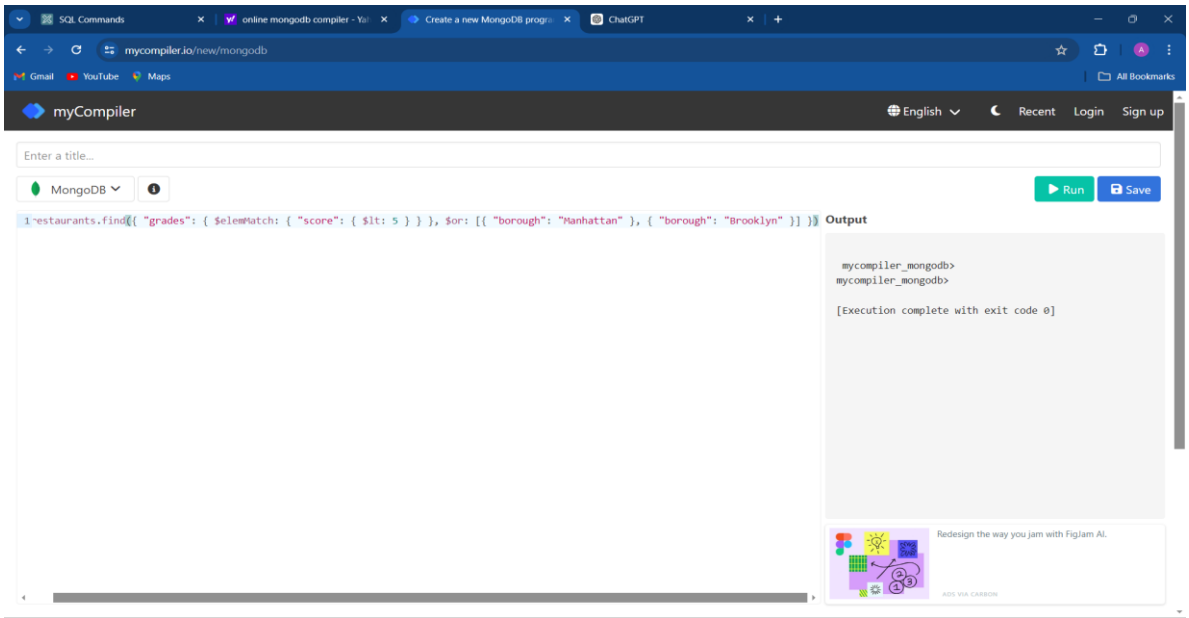
OUTPUT:



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" }] })

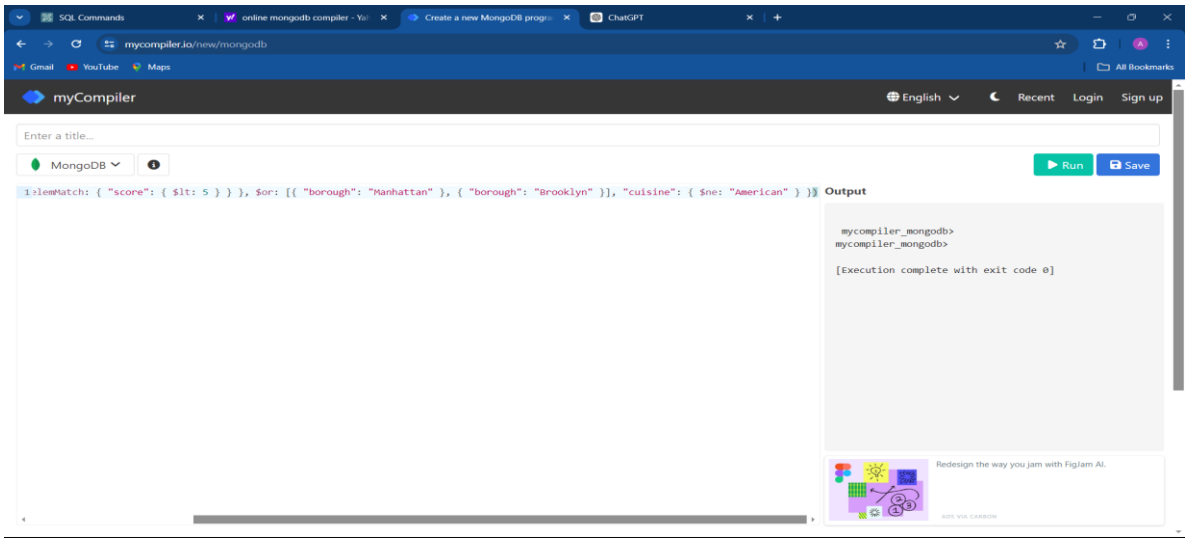
OUTPUT:



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" } })

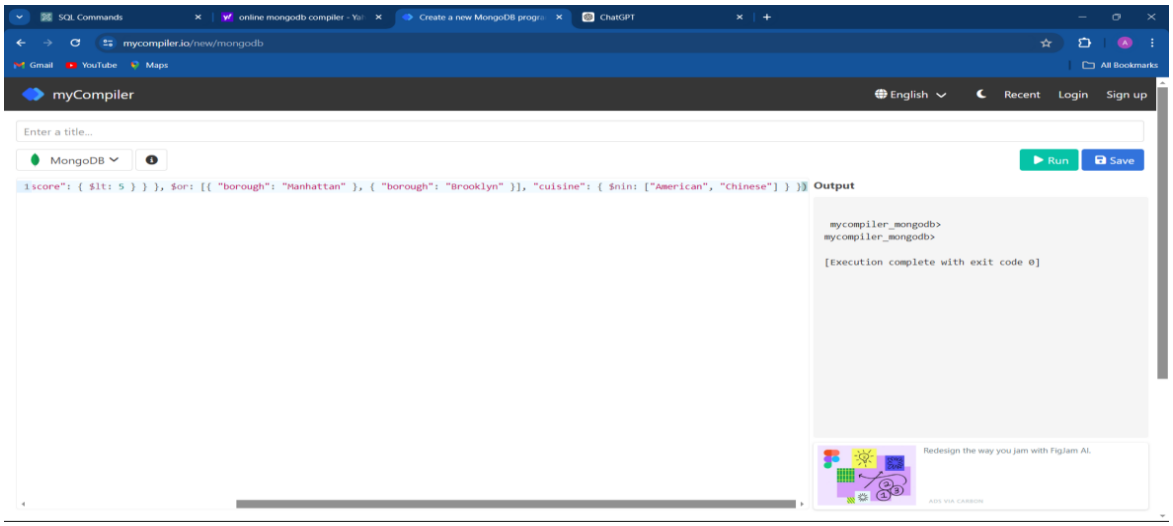
OUTPUT:



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"] } })

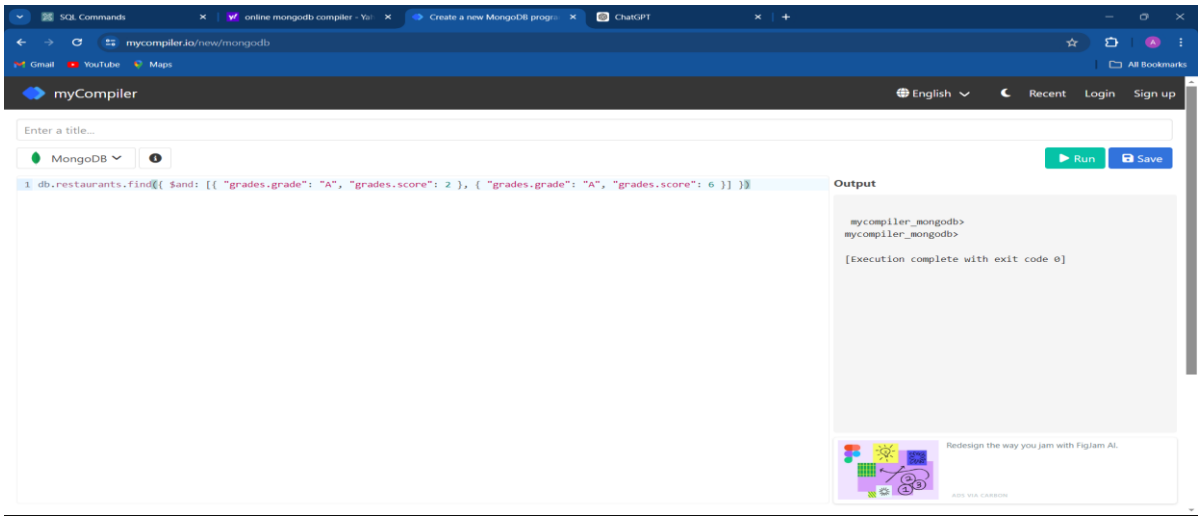
OUTPUT:



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 }] })

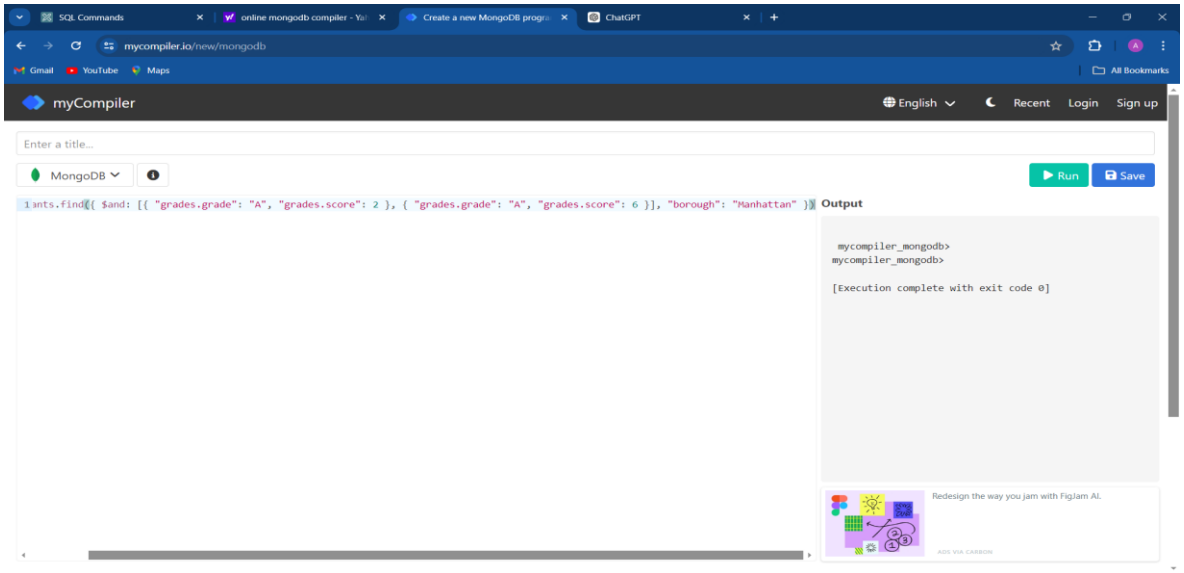
OUTPUT:



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" })

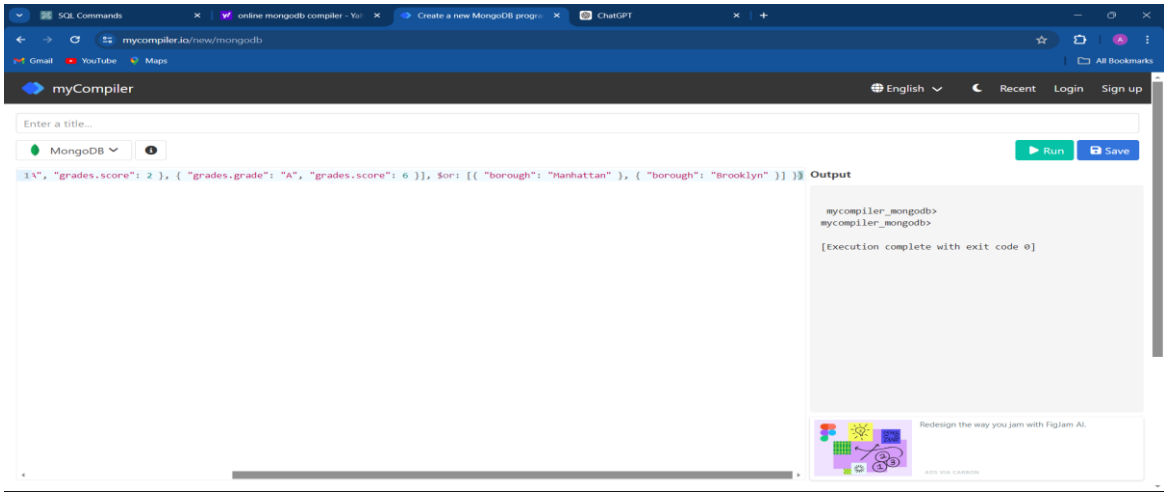
OUTPUT:



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" }] })

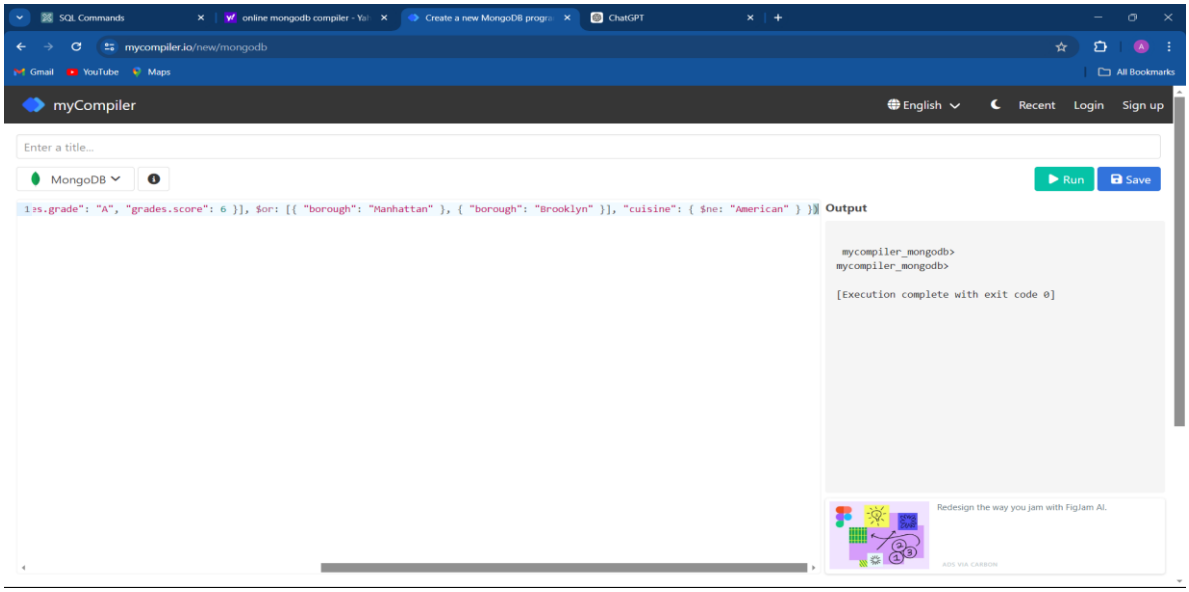
OUTPUT:



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" } })

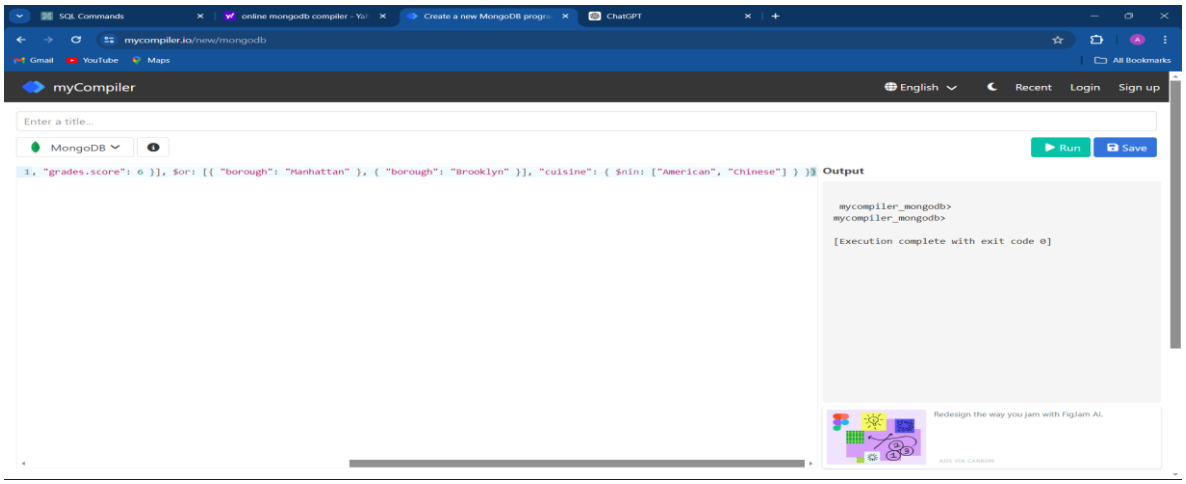
OUTPUT:



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"] } })

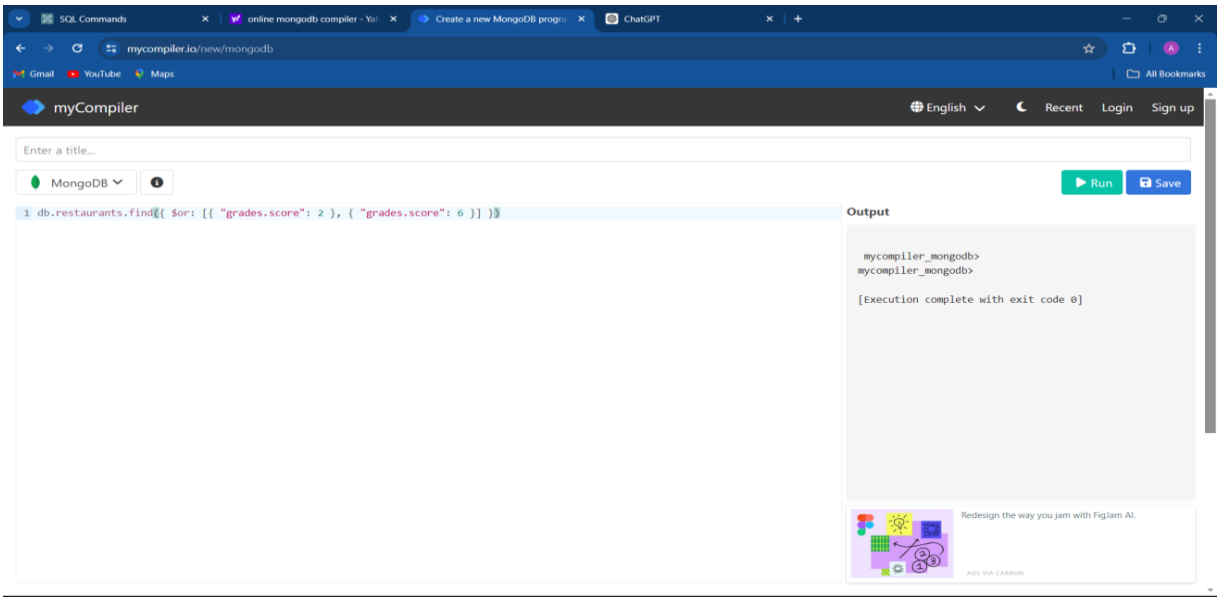
OUTPUT:



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:



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

RESULT:

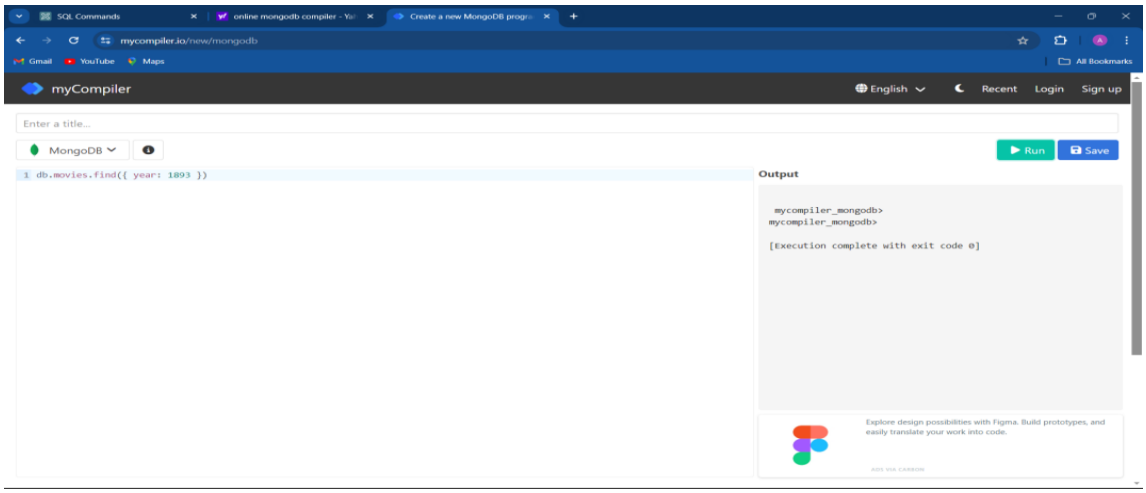
MONGO DB

EXNO: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 })

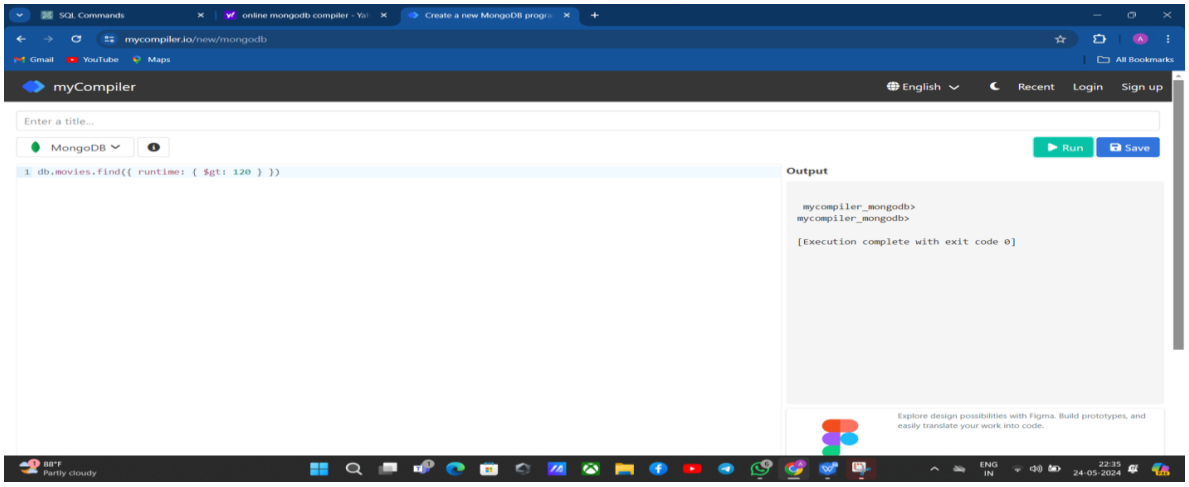
OUTPUT:



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 } })

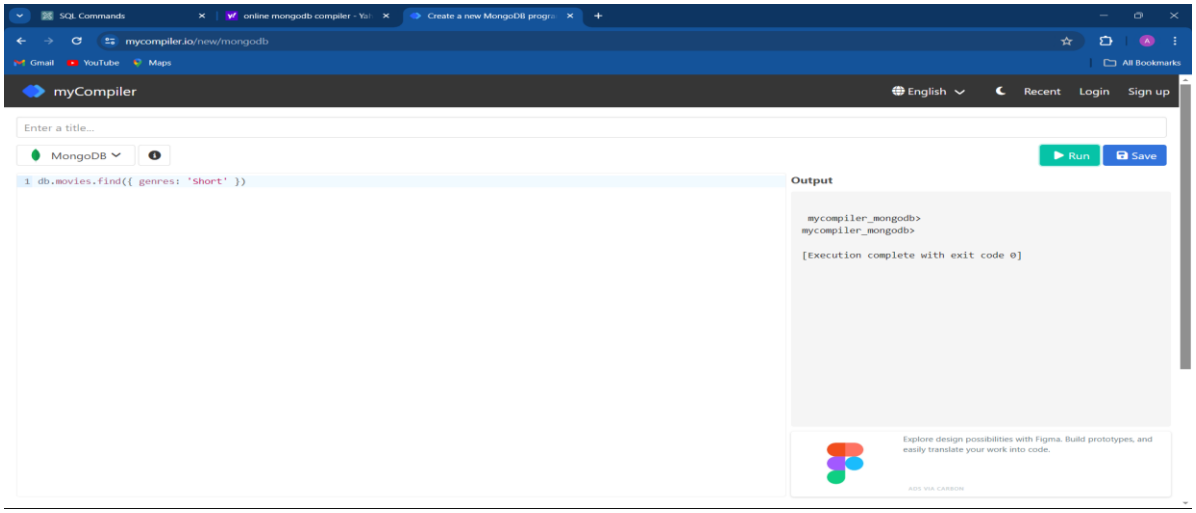
OUTPUT:



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

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

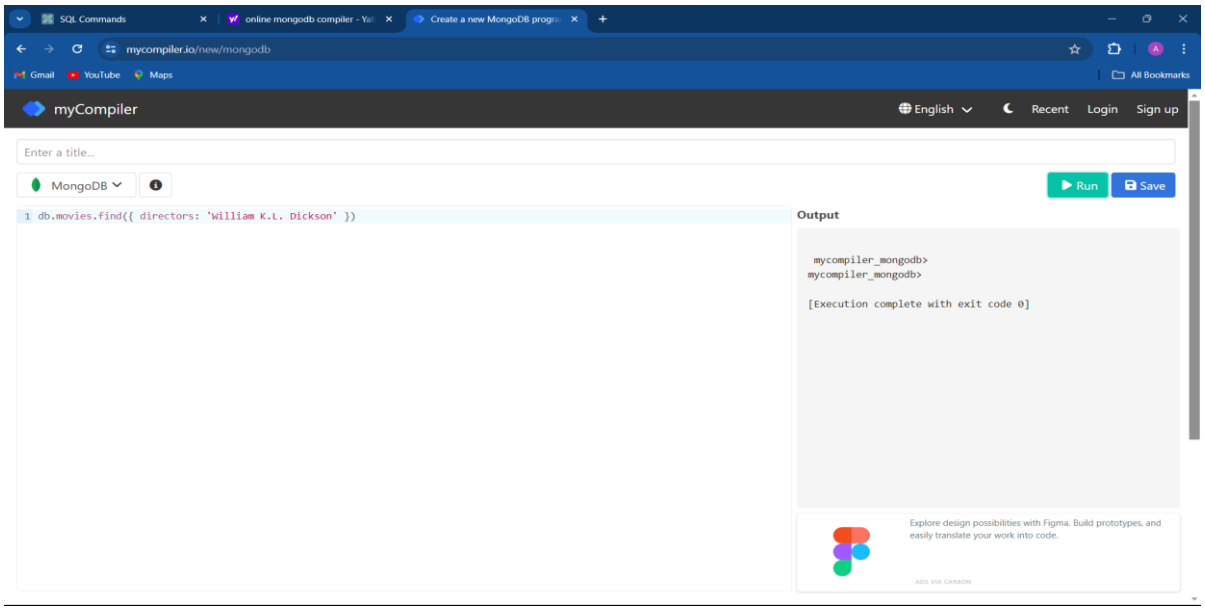
OUTPUT:



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' })

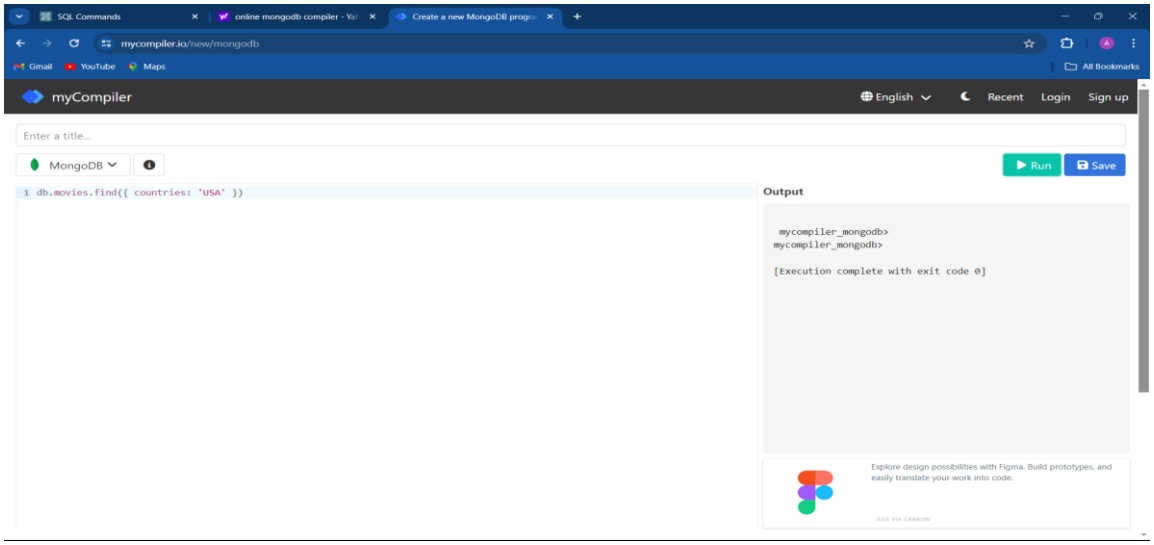
OUTPUT:



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' })

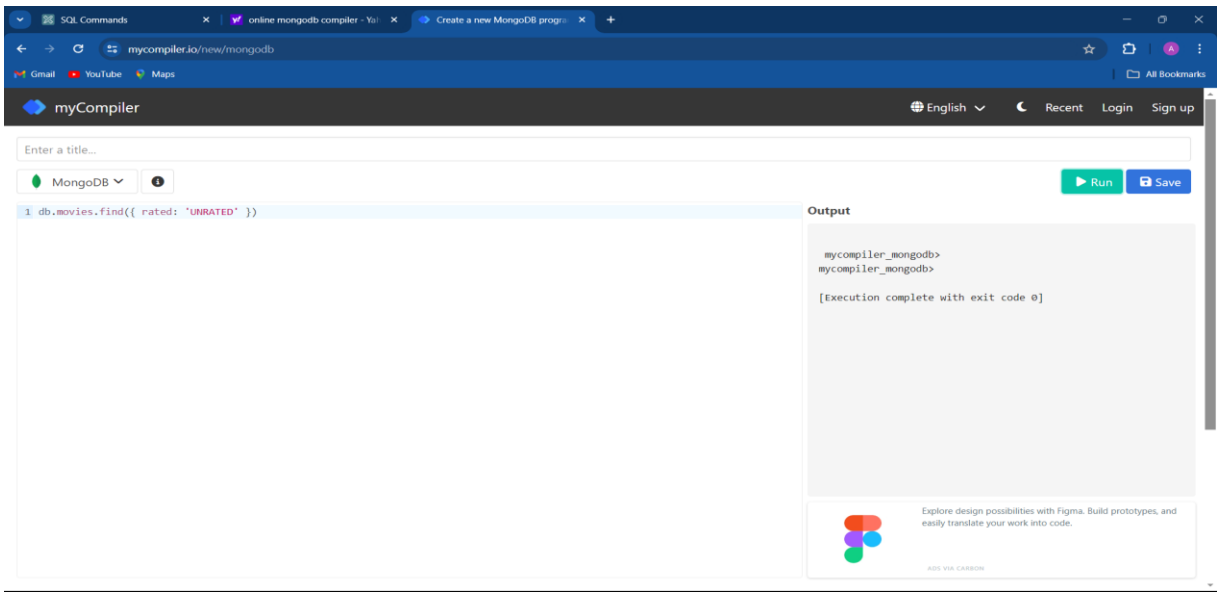
OUTPUT:



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

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

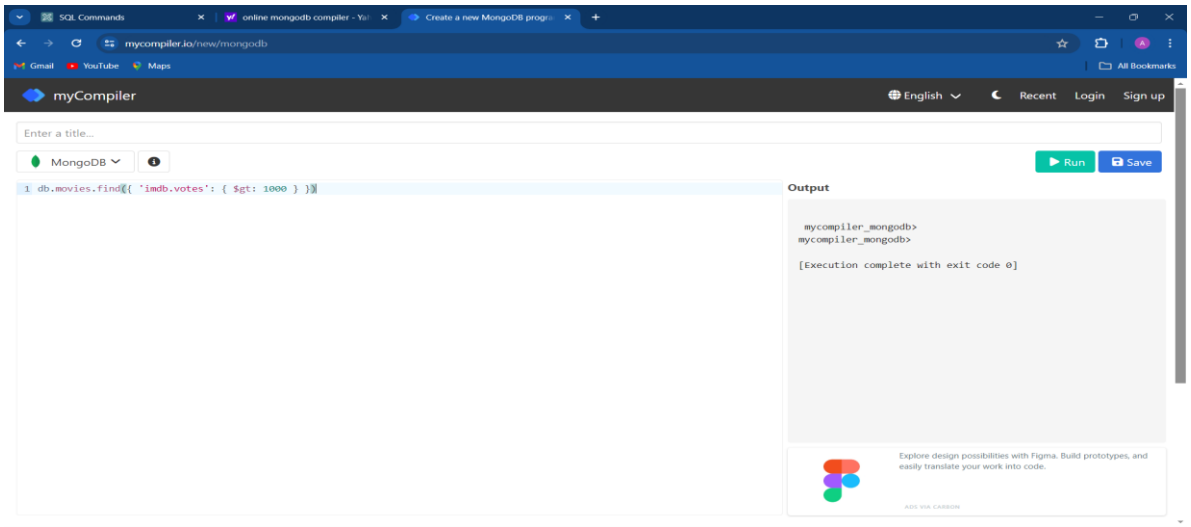
OUTPUT:



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 } })

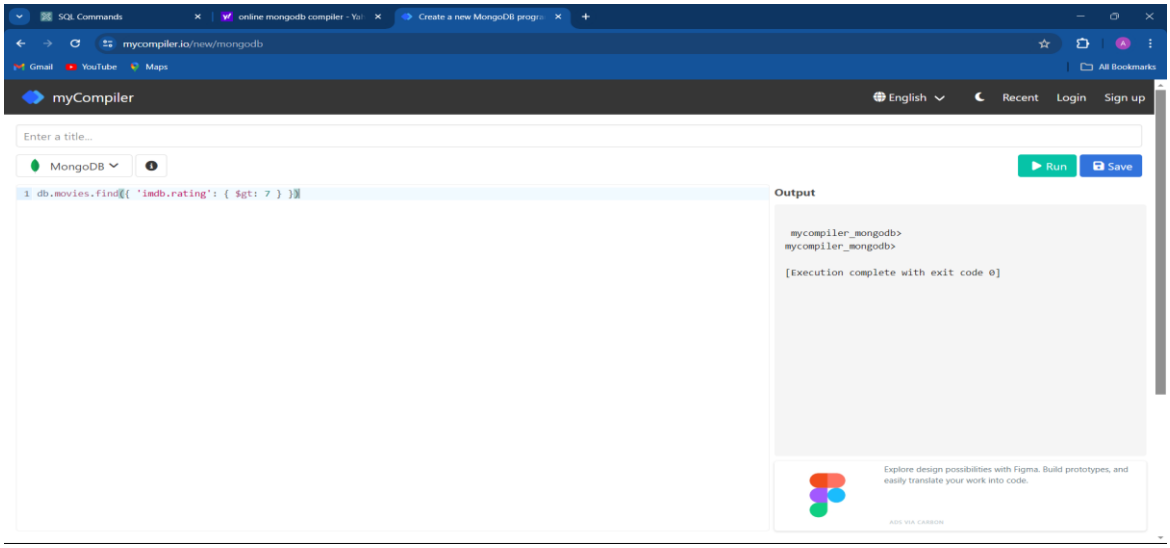
OUTPUT:



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 } })

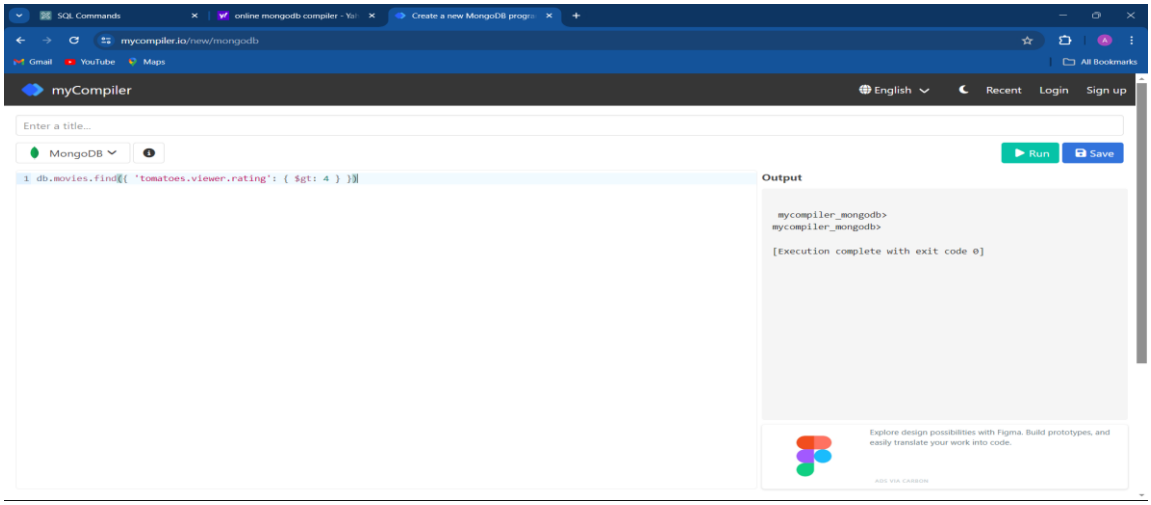
OUTPUT:



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 } })

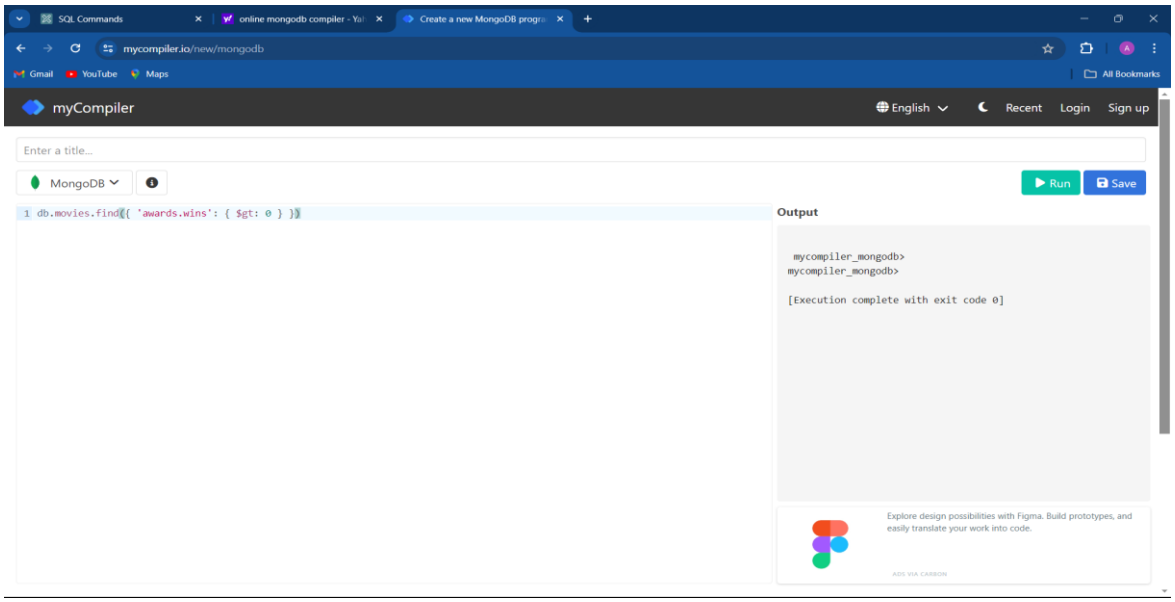
OUTPUT:



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

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

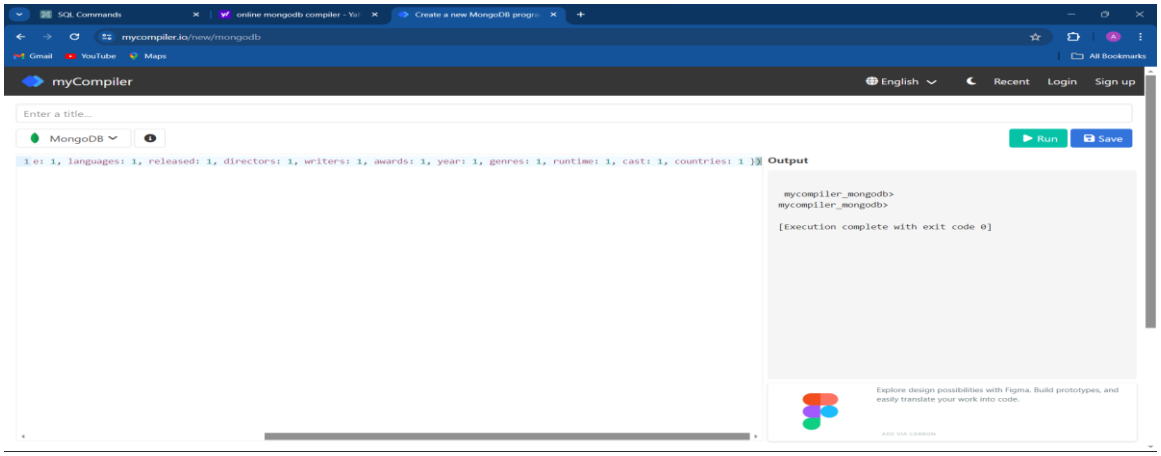
OUTPUT:



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 })

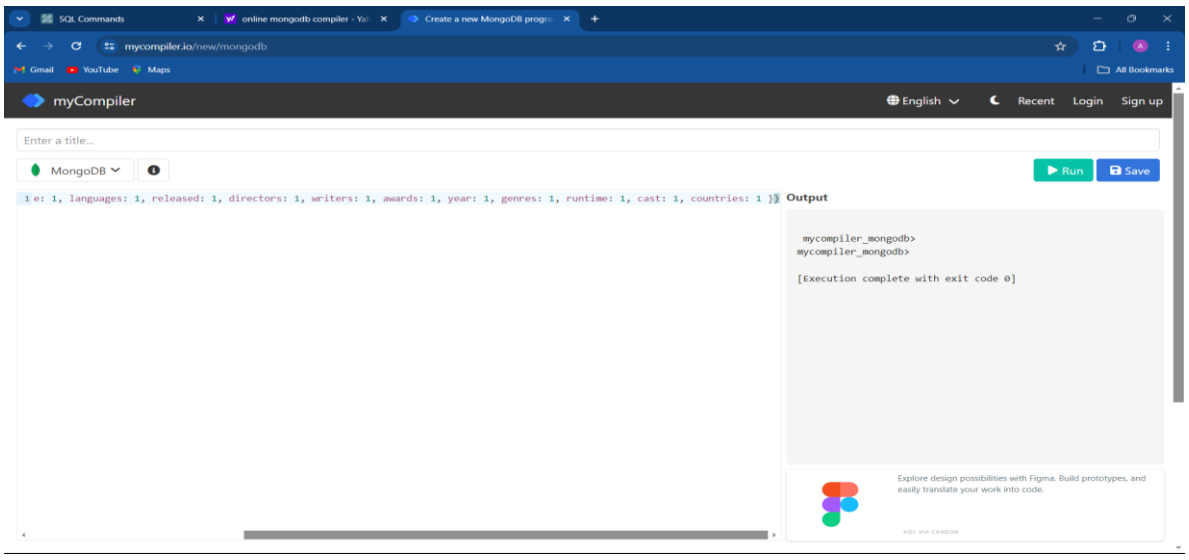
OUTPUT:



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 })

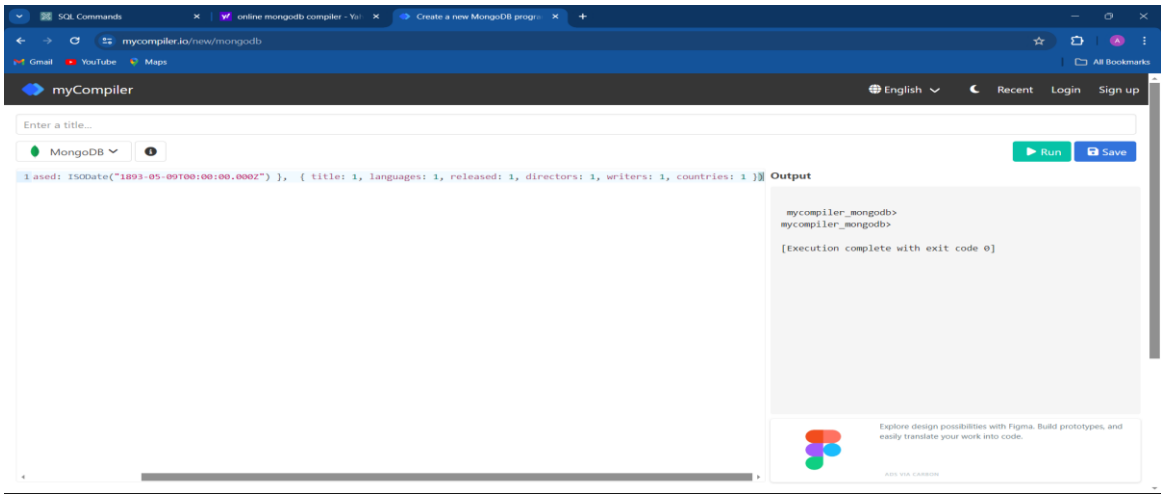
OUTPUT:



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 })

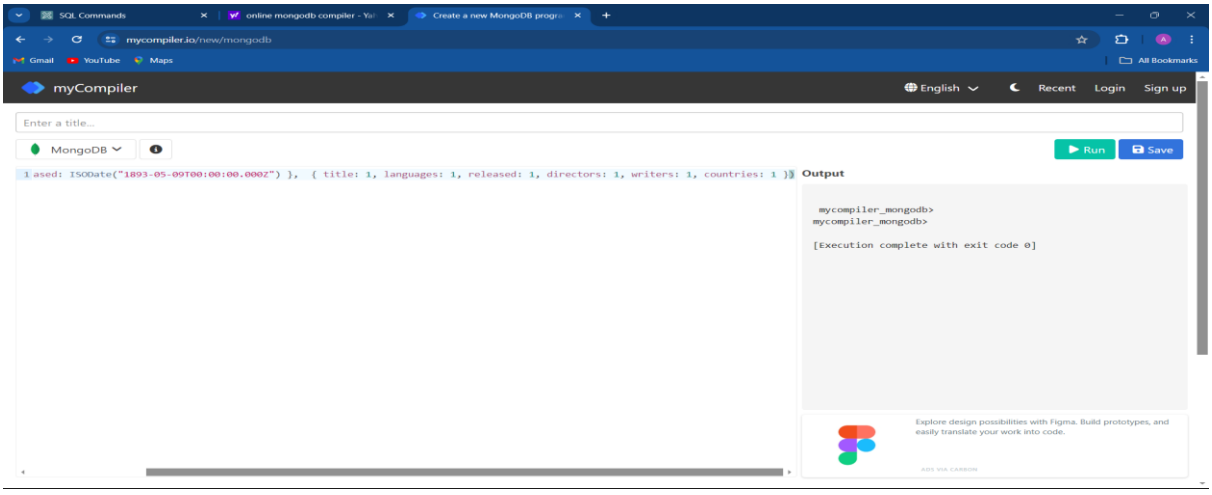
OUTPUT:



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:



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

RESULT: