CREATING AND MANAGING TABLES

EX_NO:1 **DATE:**

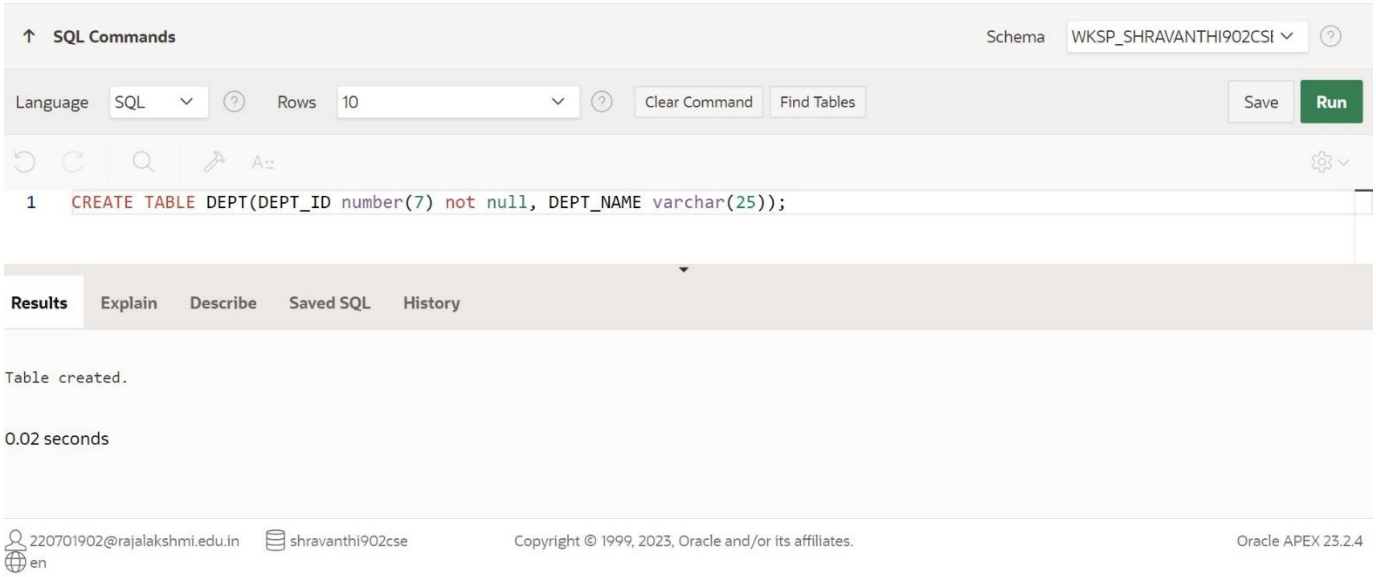
1. Create the DEPT table based on the DEPARTMENT following the table instance chart below. Confirm that the table is created.

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

QUERY:

```
CREATE TABLE DEPT (DEPT_ID number(7) not null, DEPT_NAME varchar(25));
```

OUTPUT:



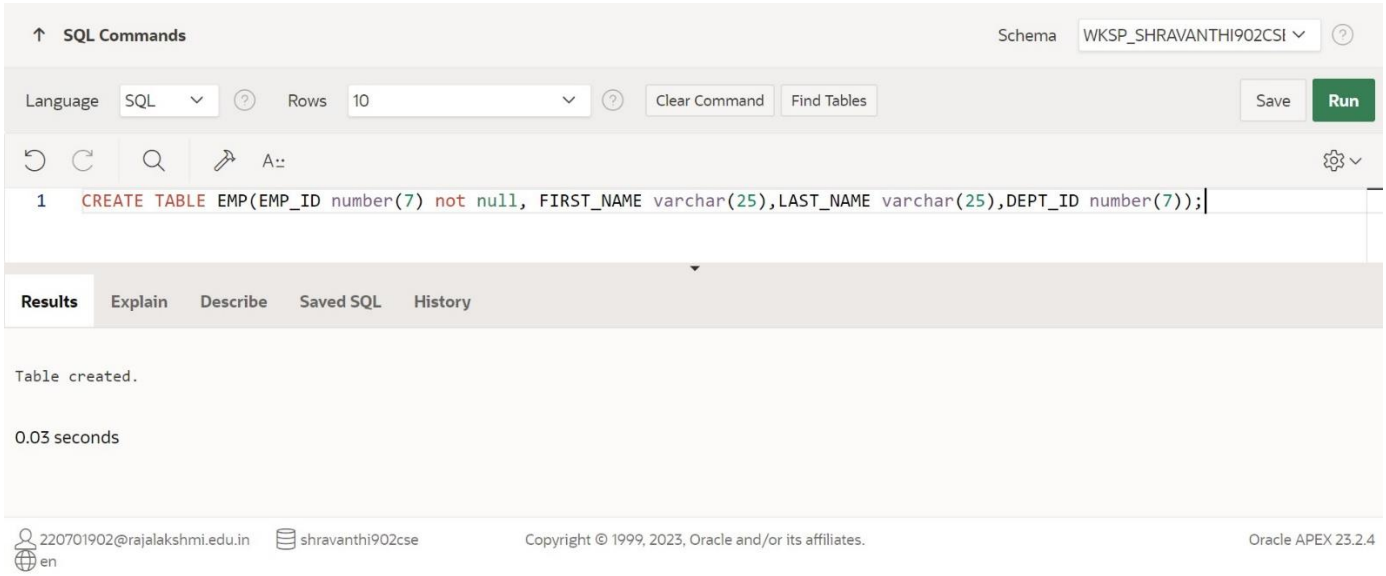
2.Create the EMP table based on the following instance chart. Confirm that the table is created.

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 (EMP_ID number(7) not null, FIRST_NAME
 varchar(25),LAST_NAME varchar(25),DEPT_ID number(7));
```

OUTPUT:

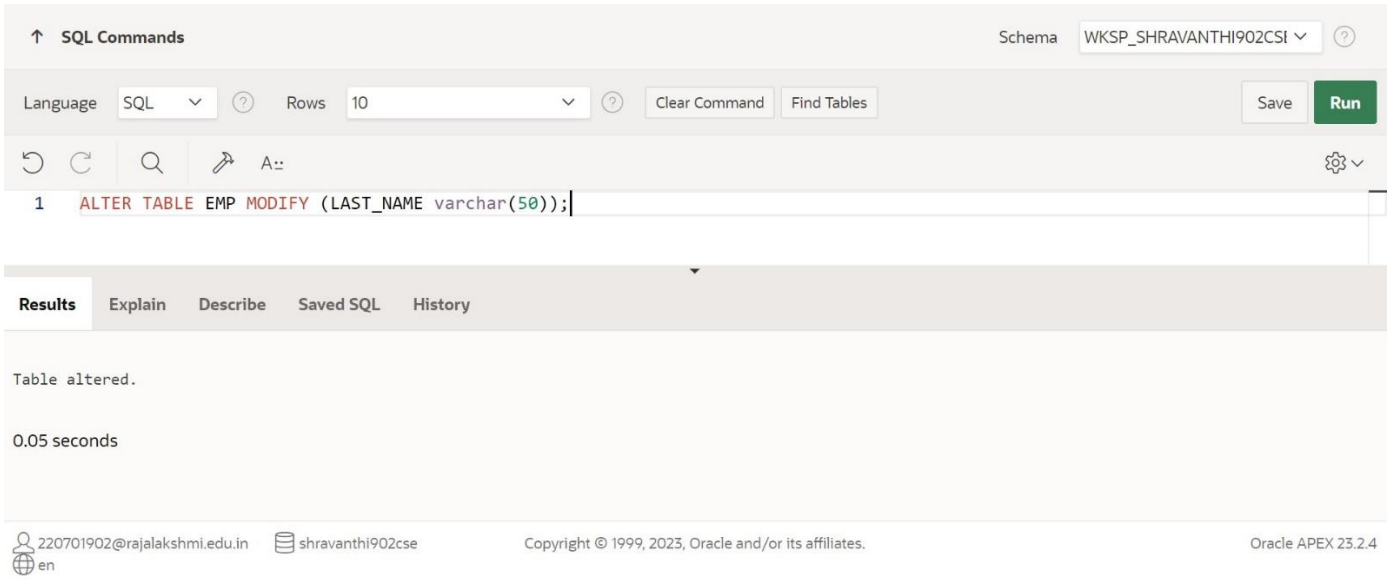


3.Modify the EMP table to allow for longer employee last names. Confirm the modification.(Hint: Increase the size to 50)

QUERY:

```
ALTER TABLE EMP MODIFY(LAST_NAME varchar(50));
```

OUTPUT:

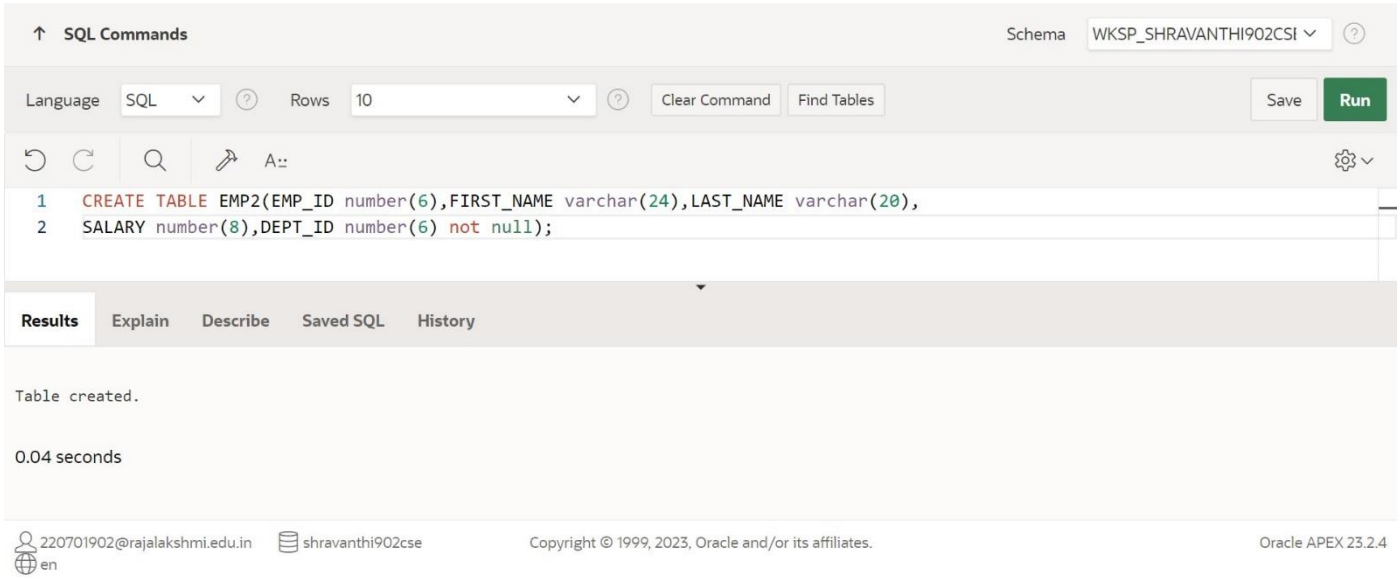


4.Create the EMPLOYEES2 table based on the structure of EMPLOYEES table. Include Only the Employee_id, First_name, Last_name, Salary and Dept_id coloumns. Name the columns Id, First_name, Last_name, salary and Dept_id respectively.

QUERY:

```
CREATE TABLE EMPLOYEE2(EMPLOYEEID number(6)not null, FIRSTNAME
 varchar(25), LASTNAME varchar(25),SALARY number(10), DEPTID number(6)
 not null);
```

OUTPUT:

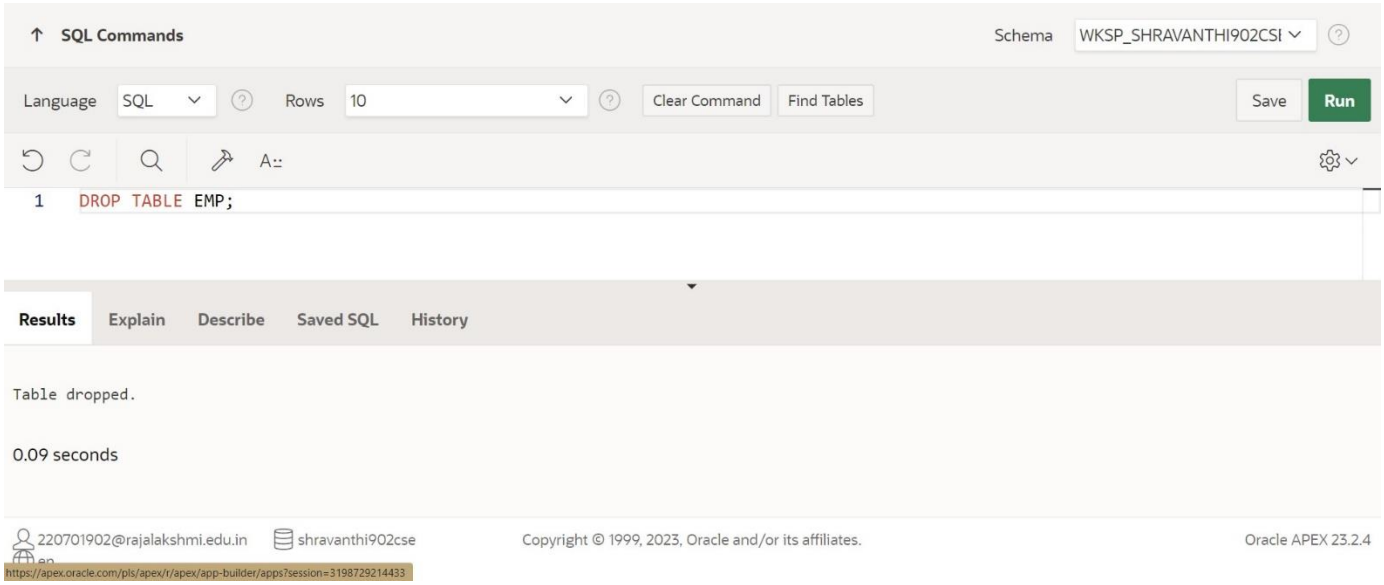


5.Drop the EMP table.

QUERY:

```
DROP TABLE EMP;
```

OUTPUT:

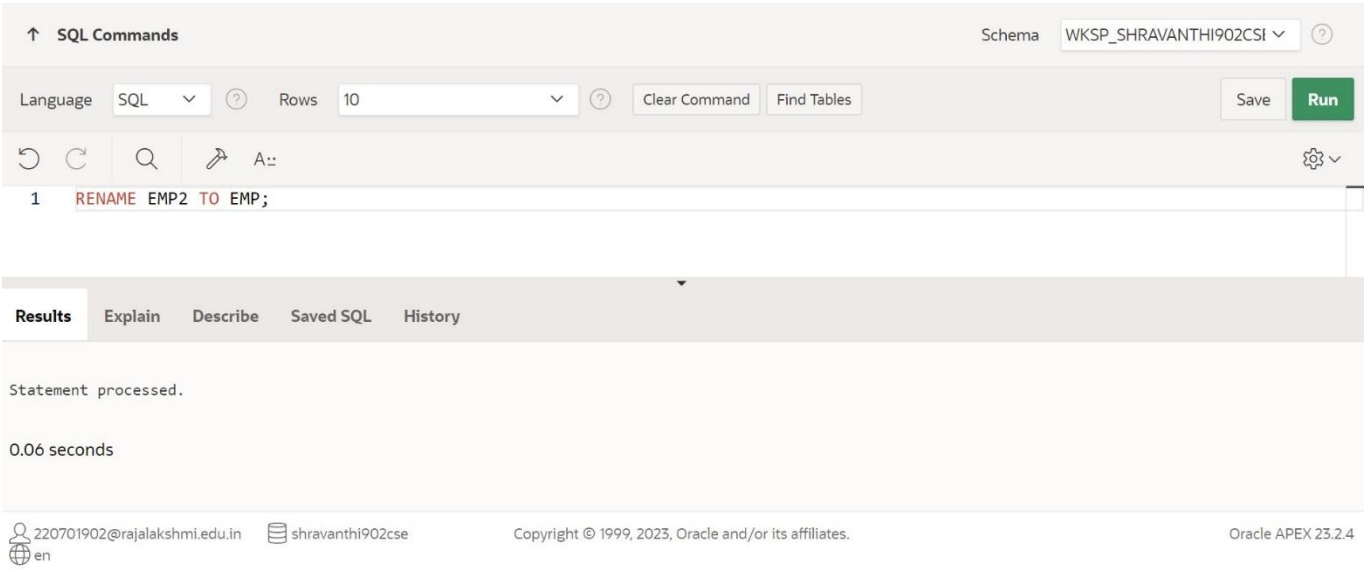


6. Rename the EMPLOYEES2 table as EMP.

QUERY:

```
RENAME EMP2 TO EMP;
```

OUTPUT:

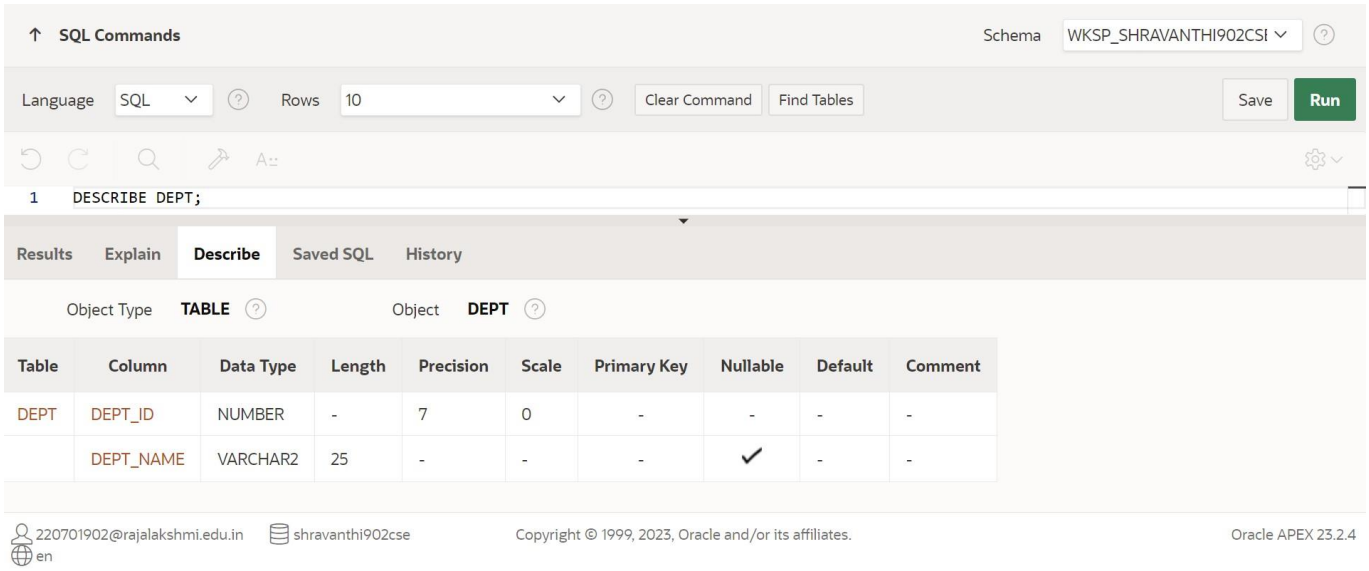


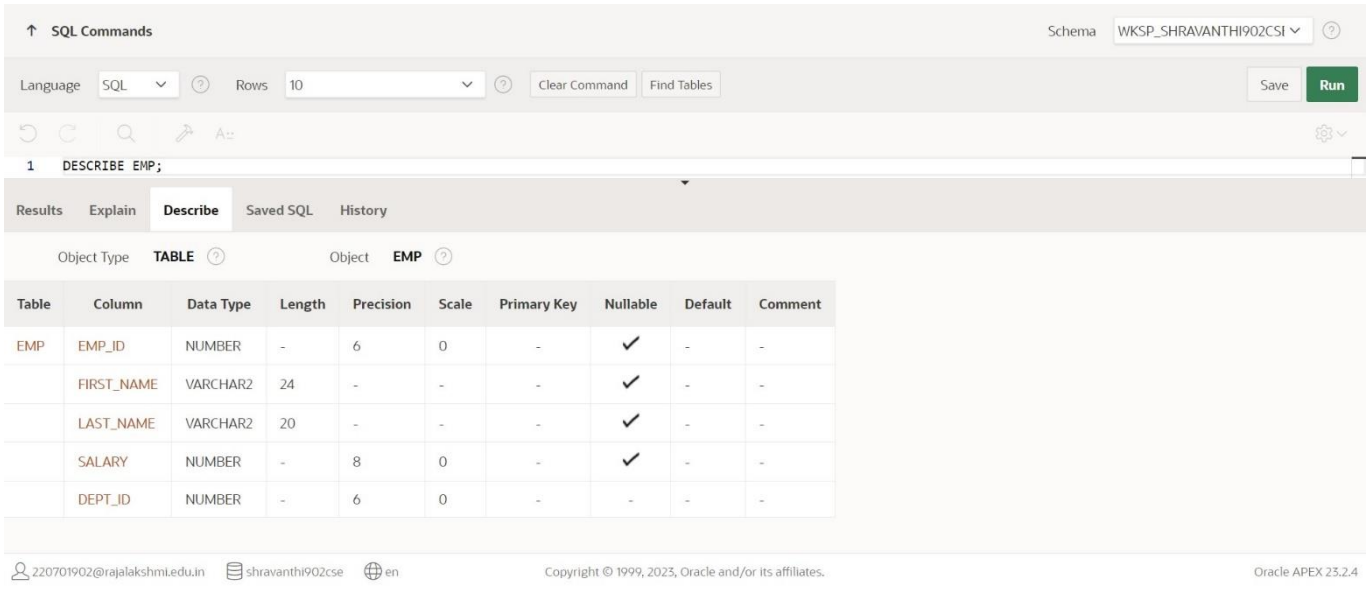
7.Add a comment on DEPT and EMP tables. Confirm the modification by describing the table.

QUERY:

comment on table dept is 'Department info';
comment on table emp is Employee info';

OUTPUT:



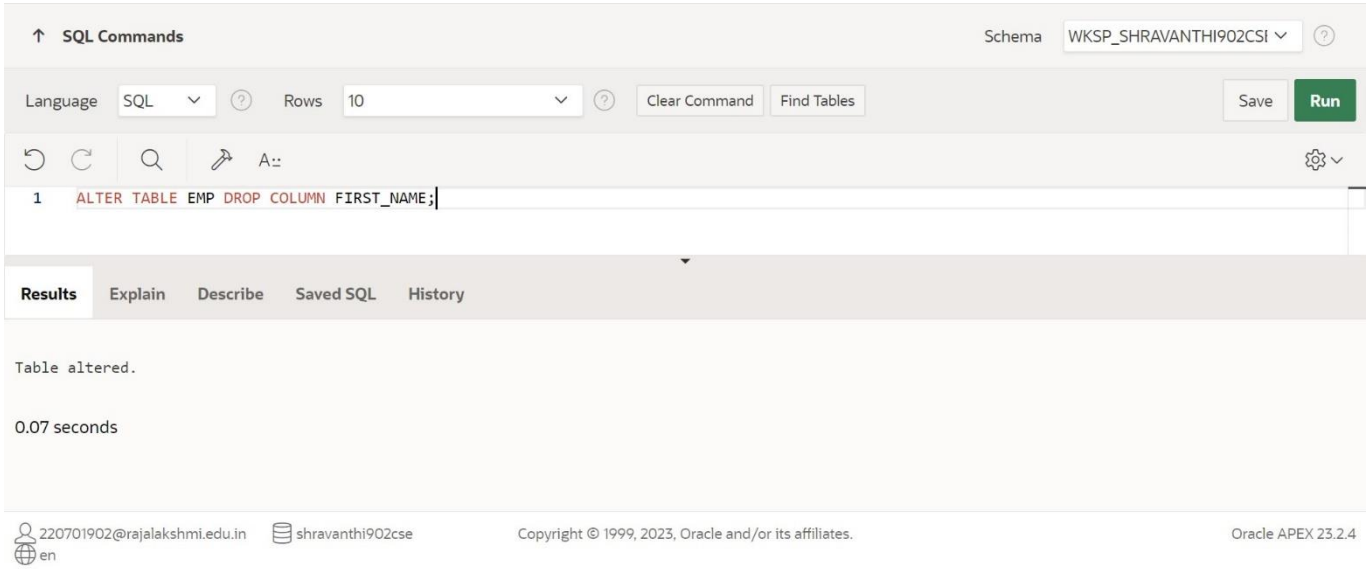


8.Drop the First_name column from the EMP table and confirm it.

QUERY:

```
ALTER TABLE EMP DROP COLUMN FIRST_NAME;
```

OUTPUT:



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

RESULT:

MANIPULATING DATA

EX_NO:2 **DATE:**

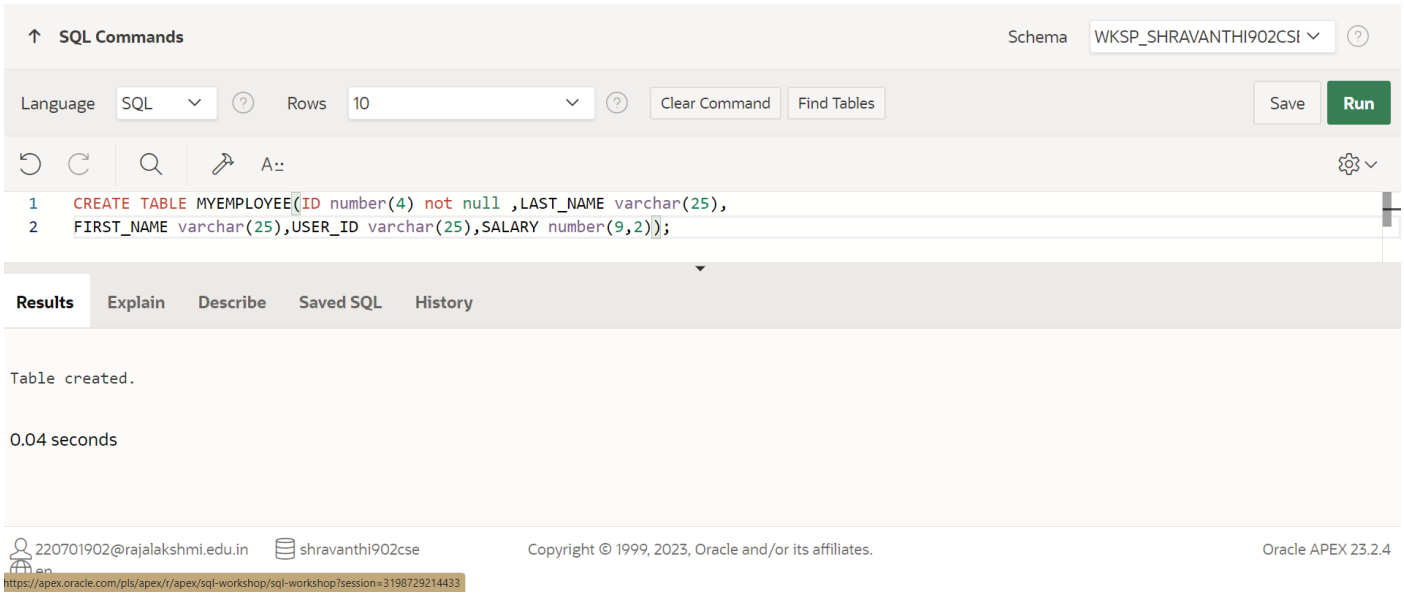
1.Create MY_EMPLOYEE table with the following structure

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 MYEMPLOYEE(ID number(4) not null ,LAST_NAME
 varchar(25),FIRST_NAME varchar(25),USER_ID varchar(25),SALARY
 number(9,2));
```

OUTPUT:



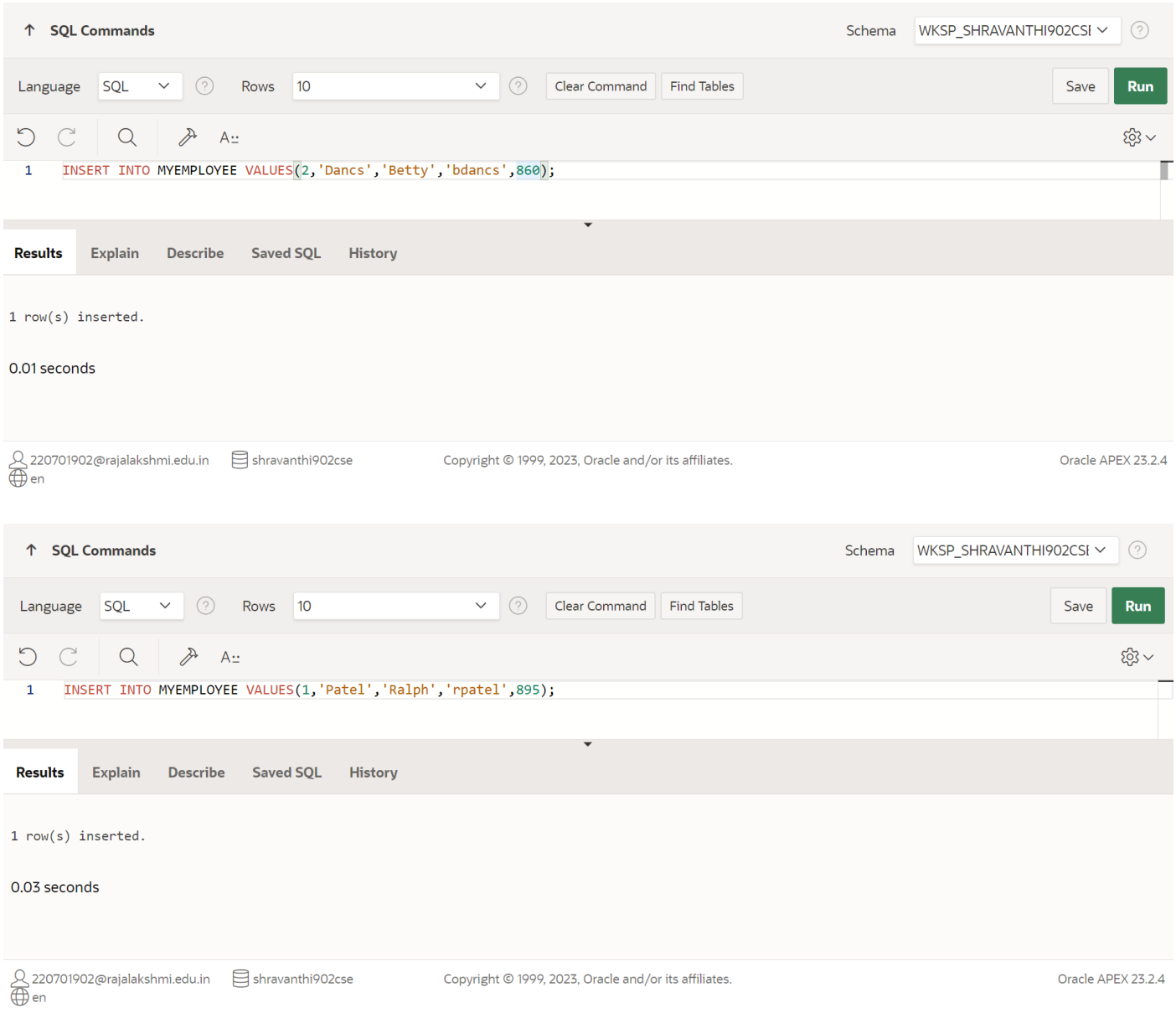
2.Add the first and second rows data to MY_EMPLOYEE table from the following sample data.

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

QUERY:

```
INSERT INTO MYEMPLOYEE VALUES(2,’Dancs’,’Betty’,’bdancs’,860);
INSERT INTO MYEMPLOYEE VALUES(1,'Patel','Ralph','rpatel',895);
```

OUTPUT:

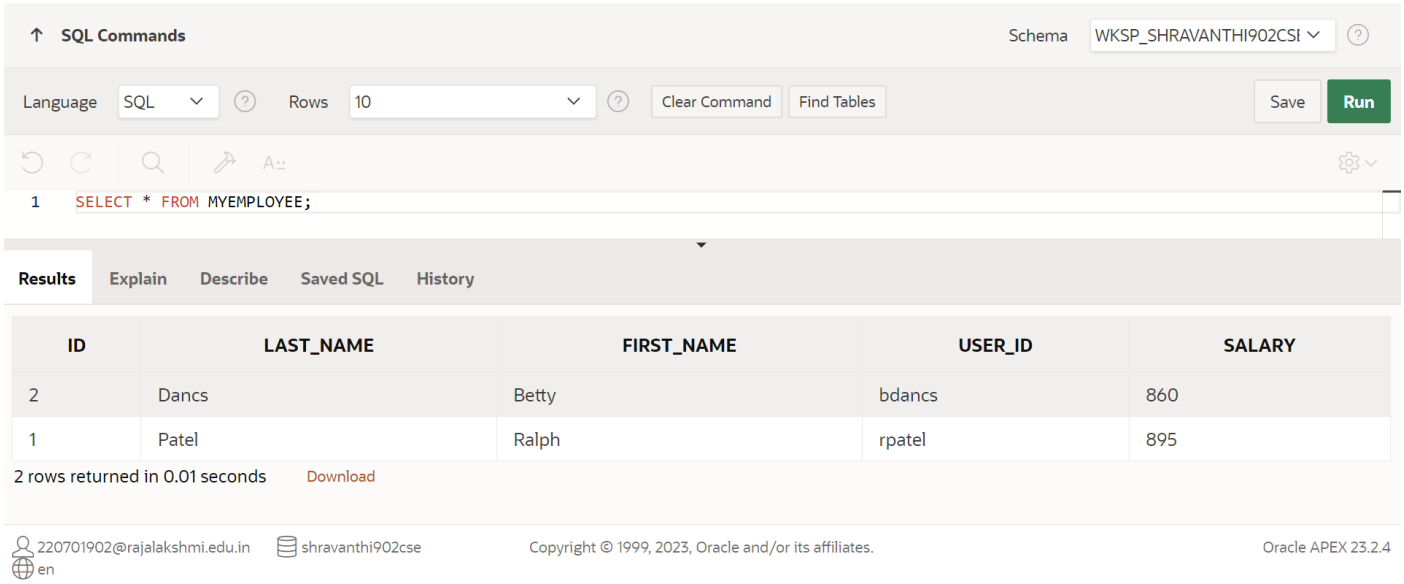


3.Display the table with values.

QUERY:

```
SELECT * FROM MYEMPLOYEE;
```

OUTPUT:

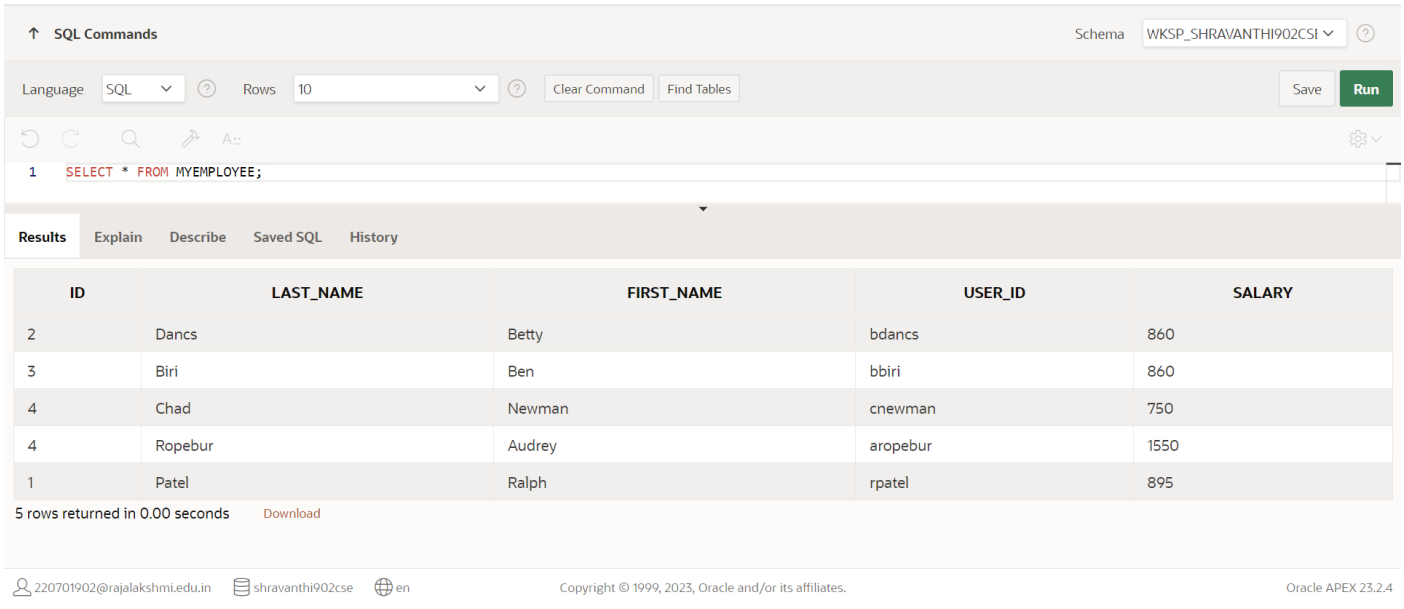


4.Populate the next two rows of data from the sample data. Concatenate the first letter of the first_name with the first seven characters of the last_name to produce Userid.

QUERY:

```
INSERT INTO MYEMPLOYEE VALUES (4,'Newman','Chad','cnewman',750);
INSERT INTO MYEMPLOYEE VALUES (5,'Ropebur','Audrey','aropebur',1550);
```

OUTPUT:

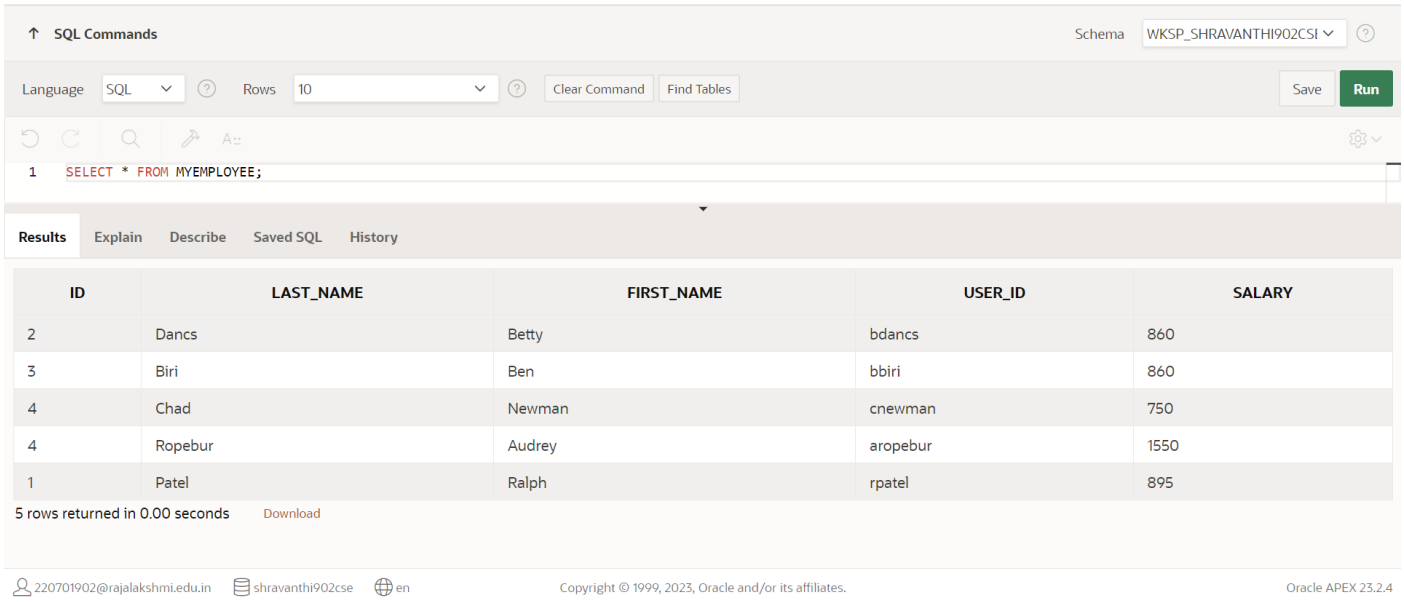


5.Make the data additions permanent.

QUERY:

```
SELECT * FROM MYEMPLOYEE;
```

OUTPUT:

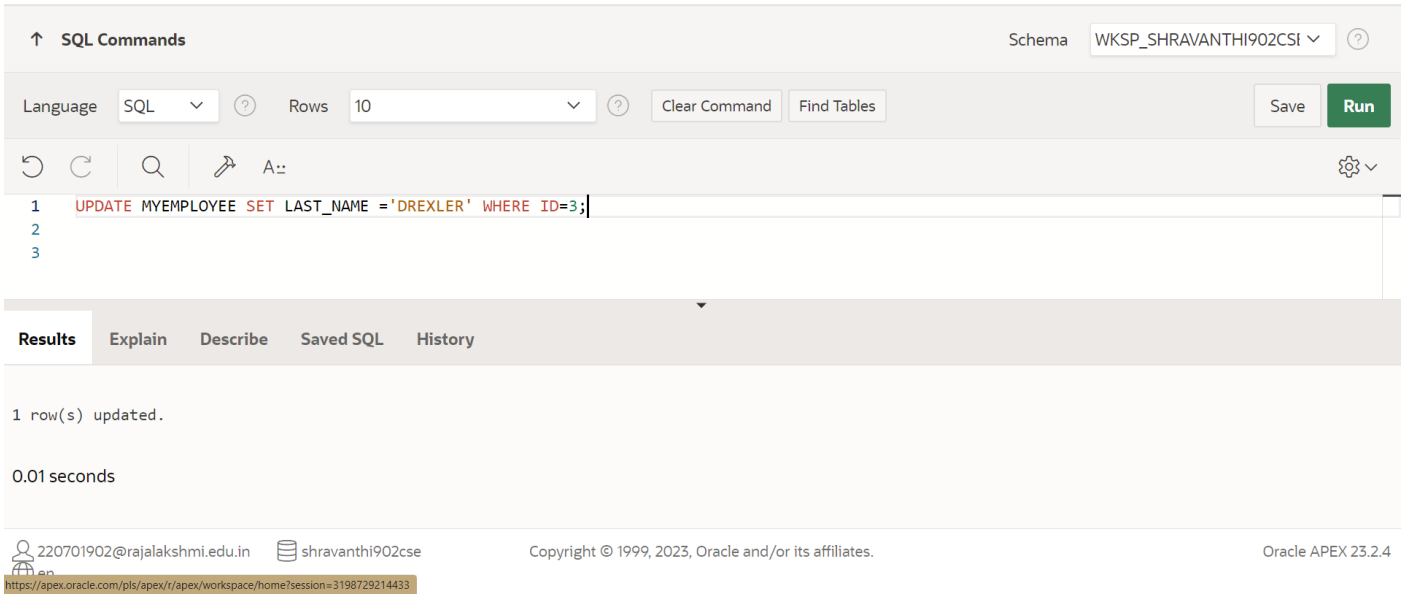


6.Change the last name of employee 3 to Drexler.

QUERY:

```
UPDATE MYEMPLOYEE SET LAST_NAME =’DREXLER’ WHERE ID=3;
```

OUTPUT:

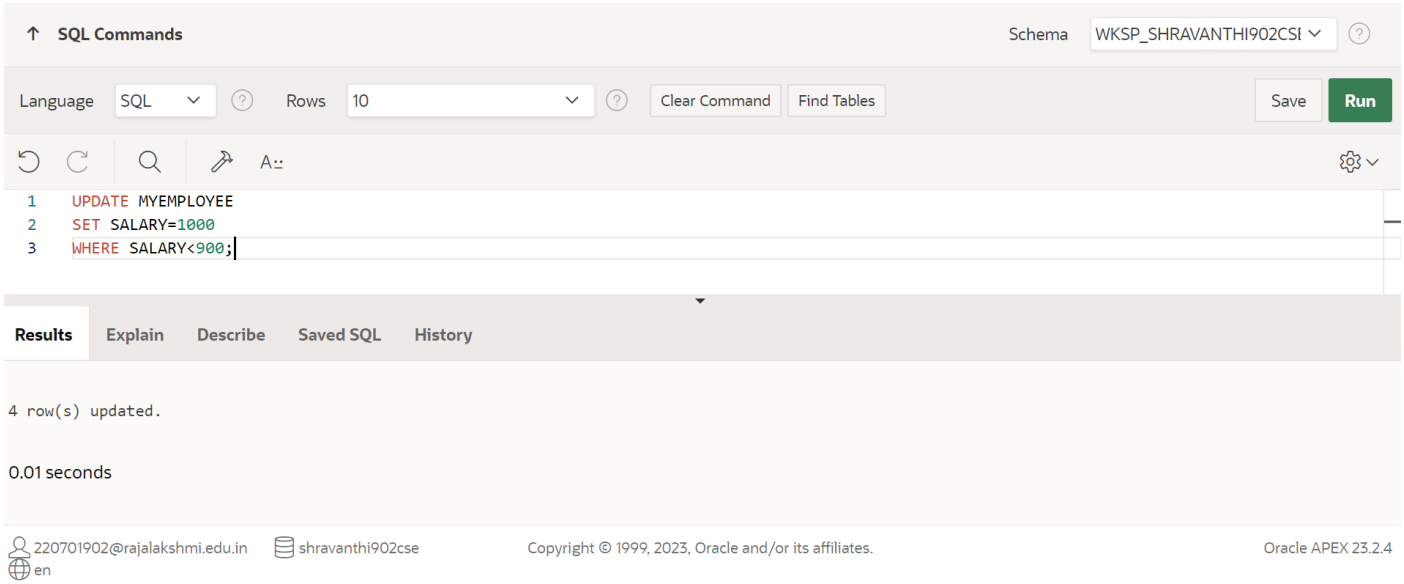


7.Change the salary to 1000 for all the employees with a salary less than 900.

QUERY:

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

OUTPUT:

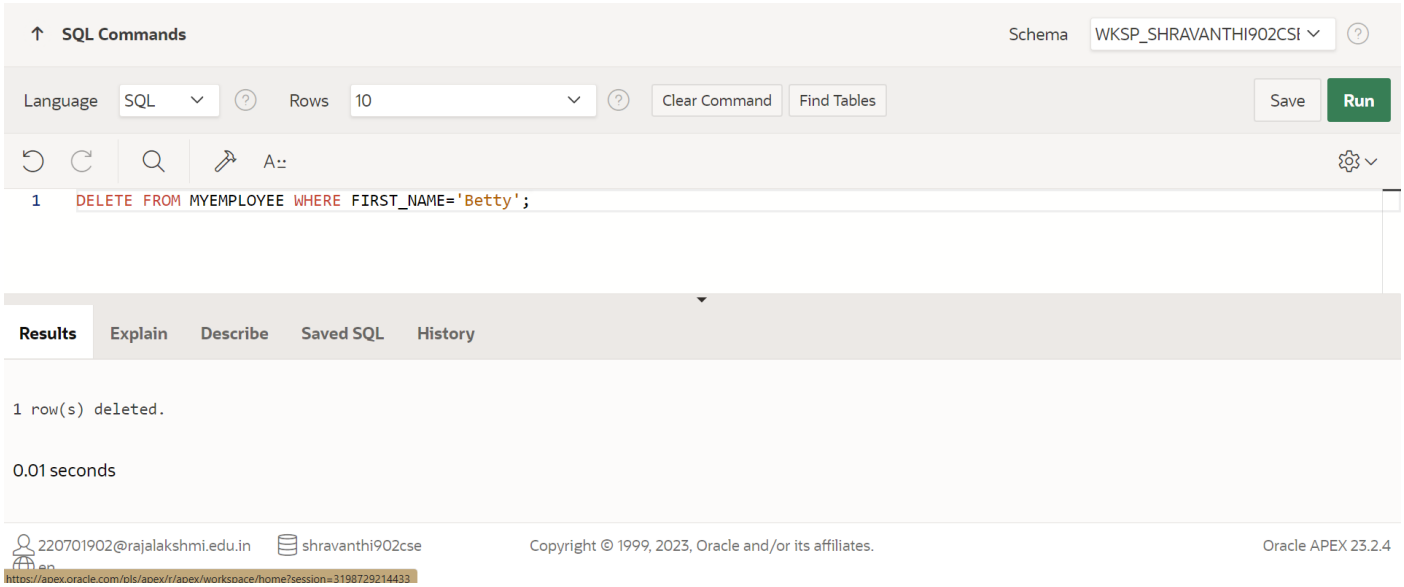


8.Delete Betty dancs from MY _EMPLOYEE table.

QUERY:

```
DELETE FROM MYEMPLOYEE WHERE FIRST_NAME=’Betty’;
```

OUTPUT:

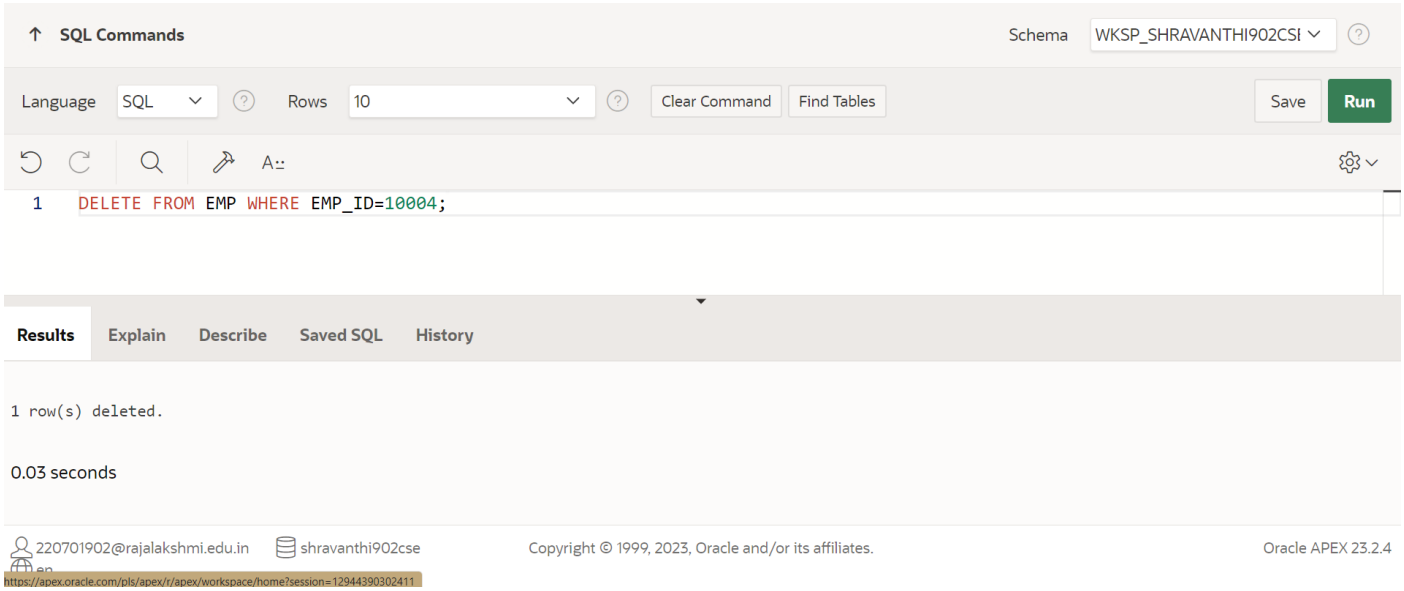


9.Empty the fourth row of the emp table.

QUERY:

```
DELETE FROM EMP WHERE EMP_ID=10004;
```

OUTPUT:



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

RESULT:

INCLUDING CONSTRAINTS

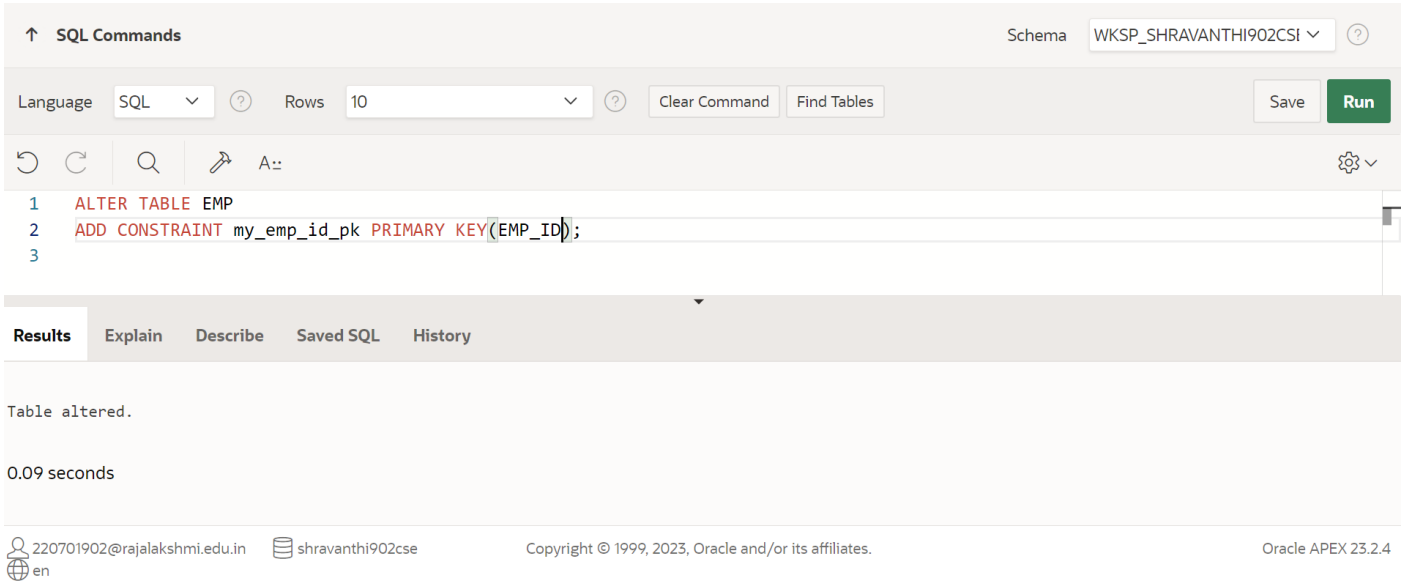
EX_NO:3 **DATE:**

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(EMP_ID);
```

OUTPUT:

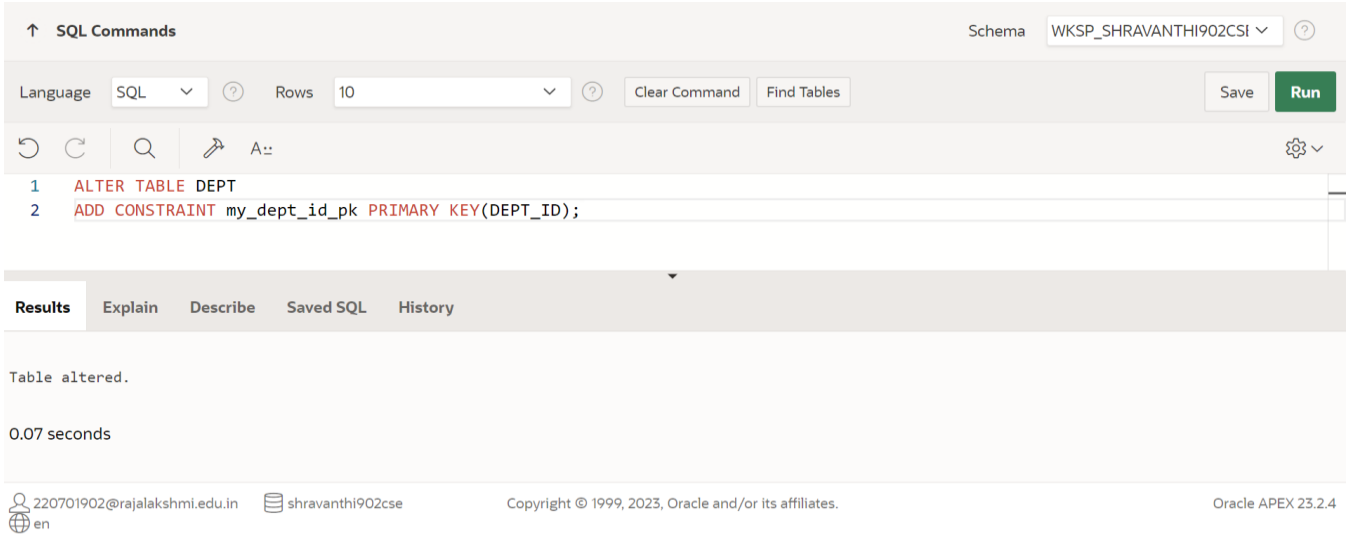


2.Create a PRIMAY KEY constraint to the DEPT table using the ID colum. The constraint should be named at creation. Name the constraint my_dept_id_pk.

QUERY:

```
ALTER TABLE DEPT
ADD CONSTRAINT my_dept_id_pk PRIMARY KEY(DEPT_ID);
```

OUTPUT:

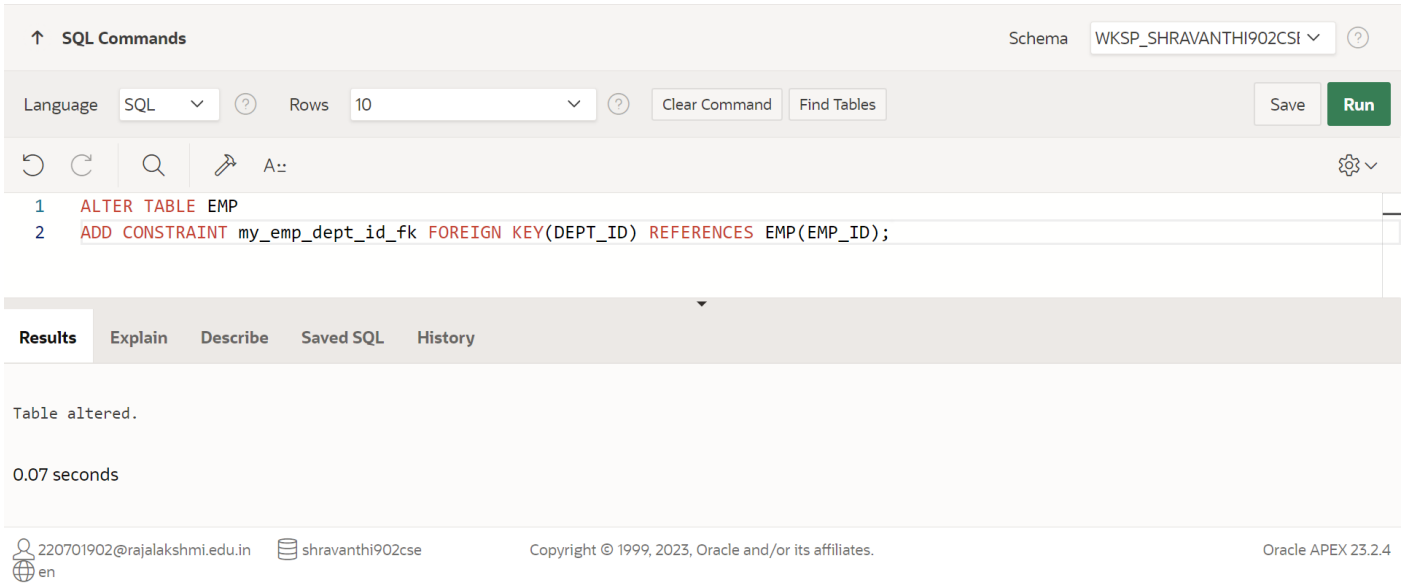


3.Add a column DEPT_ID to the EMP table. Add a foreign key reference on the EMP table that ensures that the employee is not assigned to nonexistent deparment. Name the constraint my_emp_dept_id_fk.

QUERY:

```
ALTER TABLE EMP ADD CONSTRAINT my_emp_dept_id_fk FOREIGN KEY(DEPT_ID)
  REFERENCES EMP(EMP_ID);
```

OUTPUT:

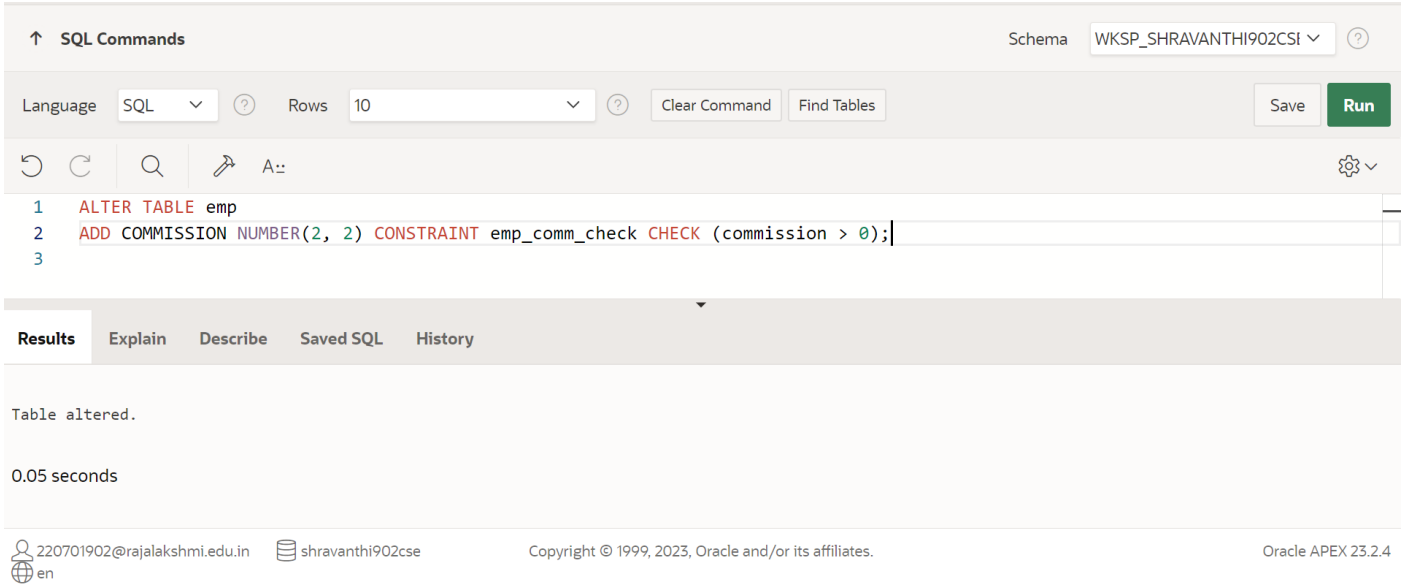


4.Modify the EMP table. Add a COMMISSION column of NUMBER data type, precision 2, scale 2. Add a constraint to the commission column that ensures that a commission value is greater than zero.

QUERY:

```
ALTER TABLE EMP ADD COMMISSION NUMBER(2, 2) CONSTRAINT emp_comm_check
  CHECK (commission > 0);
```

OUTPUT:



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

RESULT:

WRITING BASIC SQL SELECT STATEMENTS

EX_NO:4 **DATE:**

1.The following statement executes successfully.

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

QUERY:

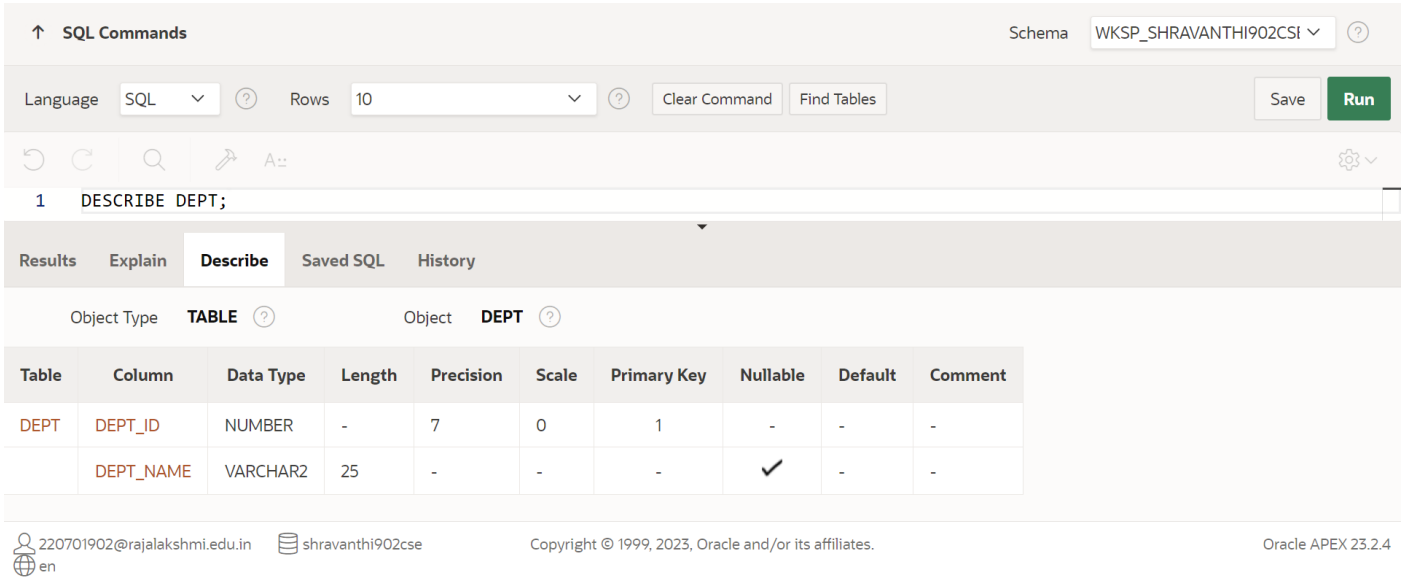
```
SELECT employeeid, last_name ,Salary*12 as Annual salary
From employees;
```

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

QUERY:

```
DESCRIBE DEPT;
```

OUTPUT:

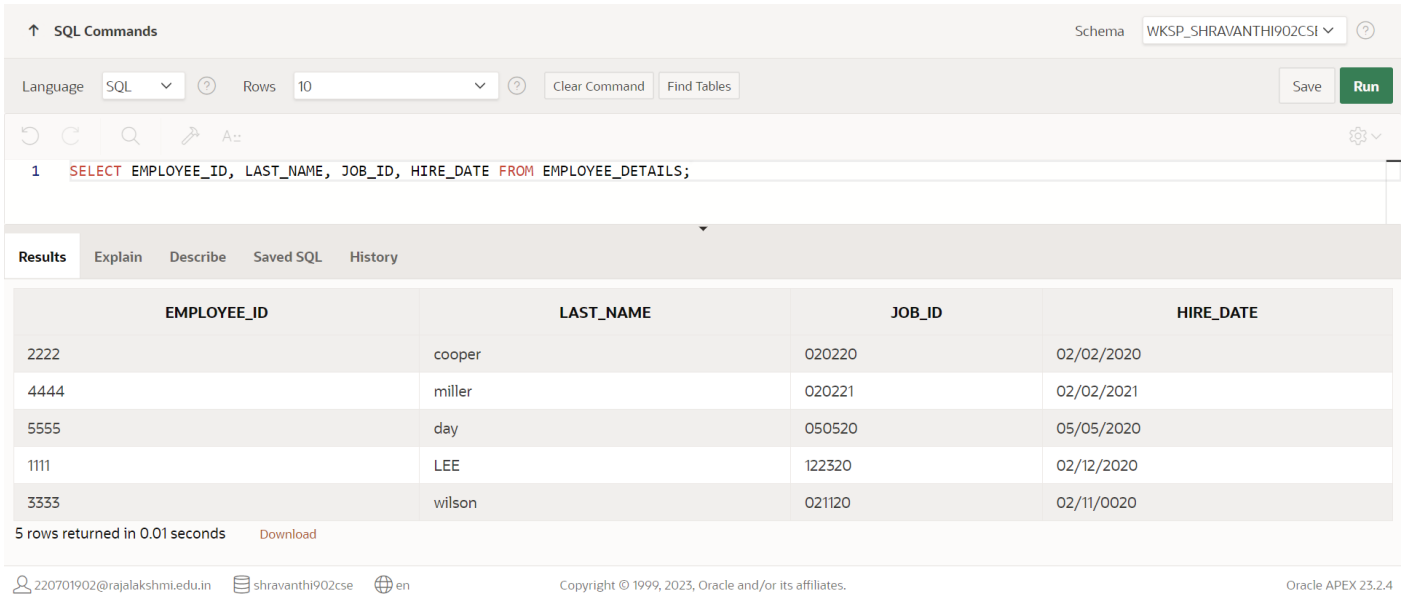


3.Create a query to display the last name, job code, hire date, and employee number for each employee, with employee number appearing first.

QUERY:

```
SELECT EMPLOYEE_ID, LAST_NAME, JOB_ID, HIRE_DATE FROM EMPLOYEE_DETAILS;
```

OUTPUT:

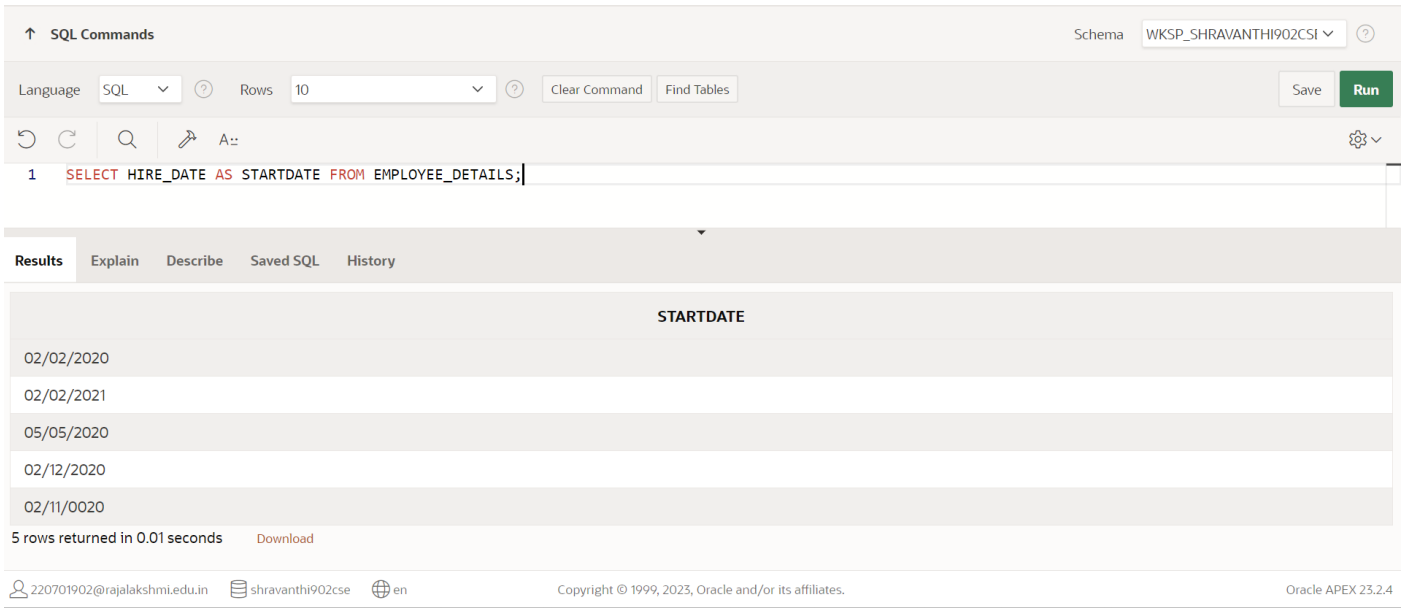


4. Provide an alias STARTDATE for the hire date.

QUERY:

```
SELECT HIRE_DATE AS STARTDATE FROM EMPLOYEE_DETAILS;
```

OUTPUT:

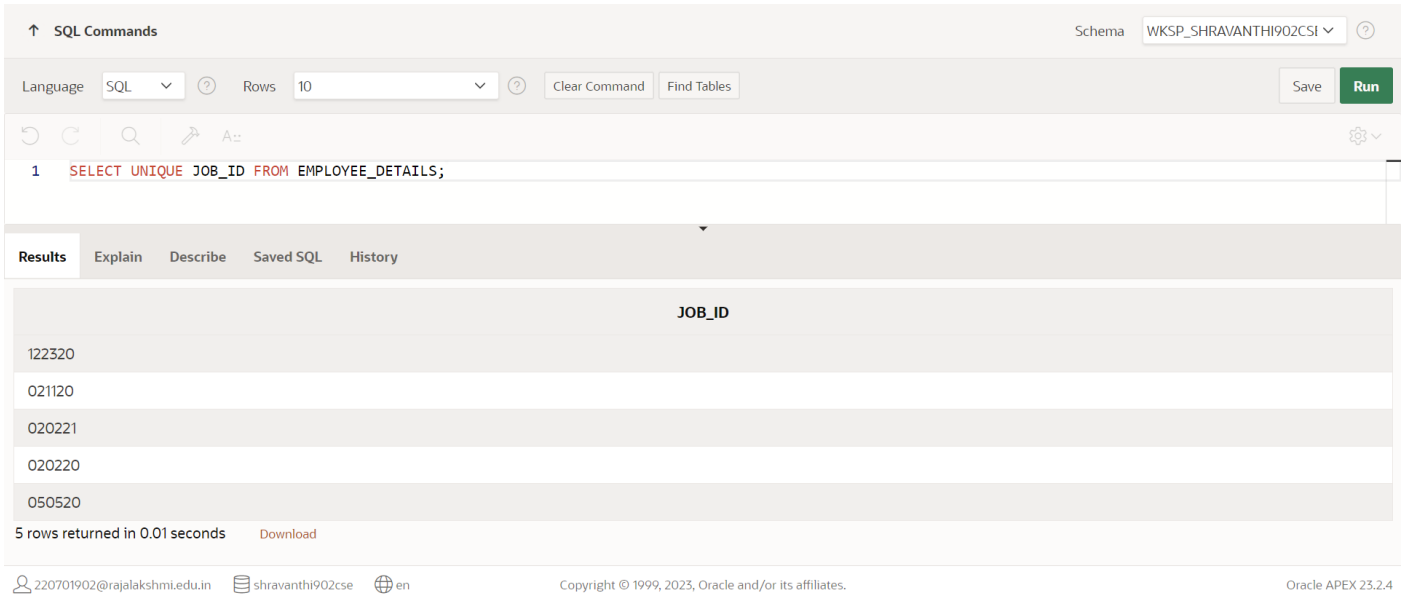


5.Create a query to display unique job codes from the employee table:

QUERY:

```
SELECT UNIQUE JOB_ID FROM EMPLOYEE_DETAILS;
```

OUTPUT:

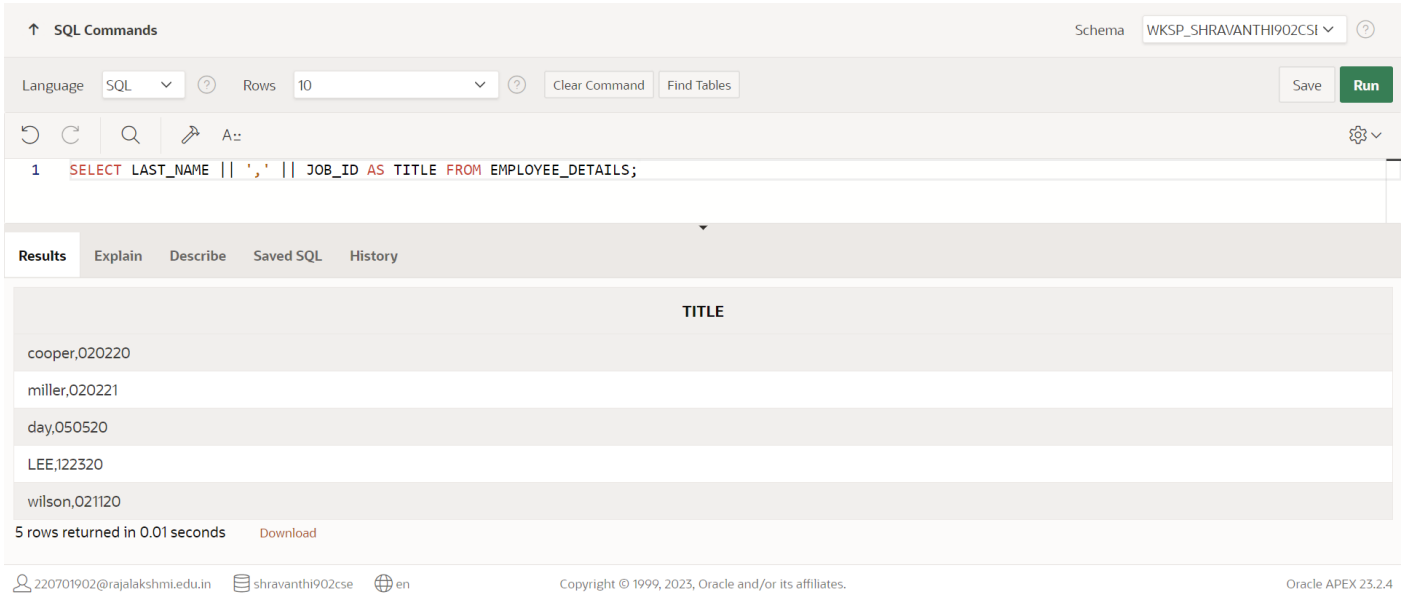


6.Display the last name concatenated with the job ID , separated by a comma and space, and name the column EMPLOYEE and TITLE.

QUERY:

```
SELECT LAST_NAME || ',' || JOB_ID AS TITLE FROM EMPLOYEE_DETAILS;
```

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 EMP_ID ||','|| LAST_NAME ||','|| JOB_ID ||','|| EMAIL_ID ||','||
 SALARY ||','|| HIRE_DATE AS "THE_OUTPUT" FROM EMPLOYEE_DETAILS;
```

OUTPUT:

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

RESULT:

RESTRICTING AND SORTING DATA

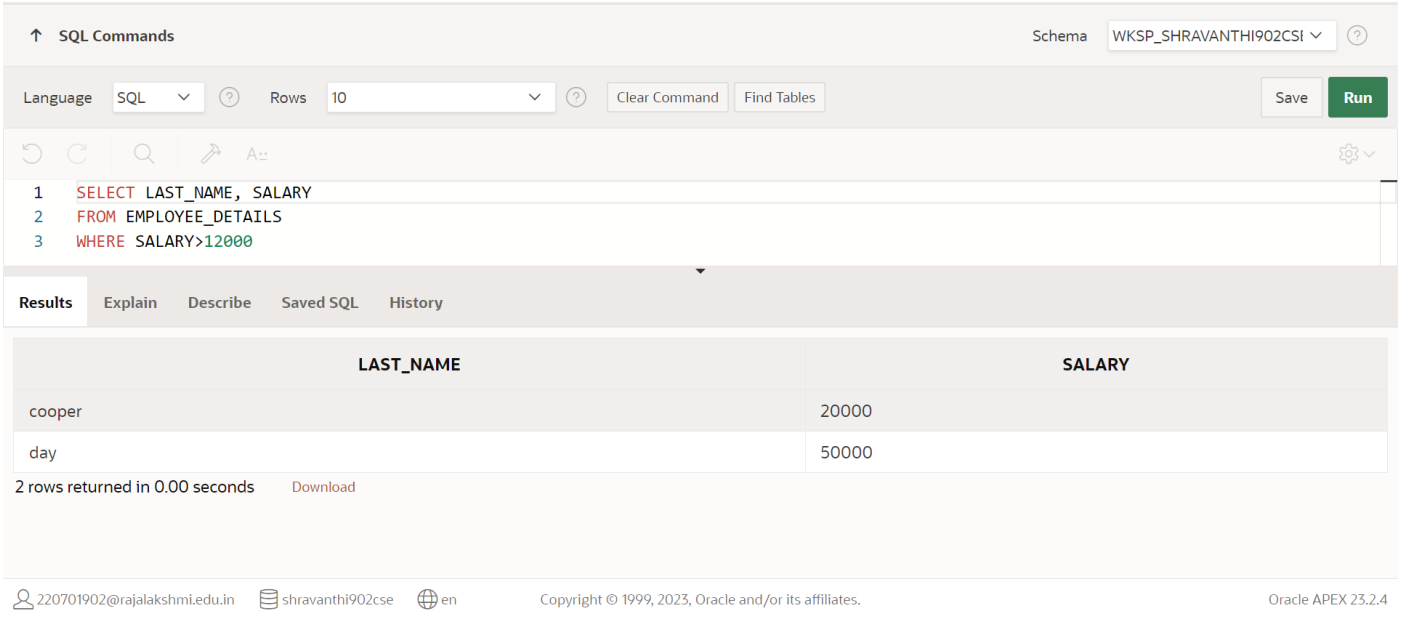
EX_NO:5 **DATE:**

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

QUERY:

```
SELECT LAST_NAME,SALARY FROM EMPLOYEE_DETAILS WHERE SALARY>12000;
```

OUTPUT:

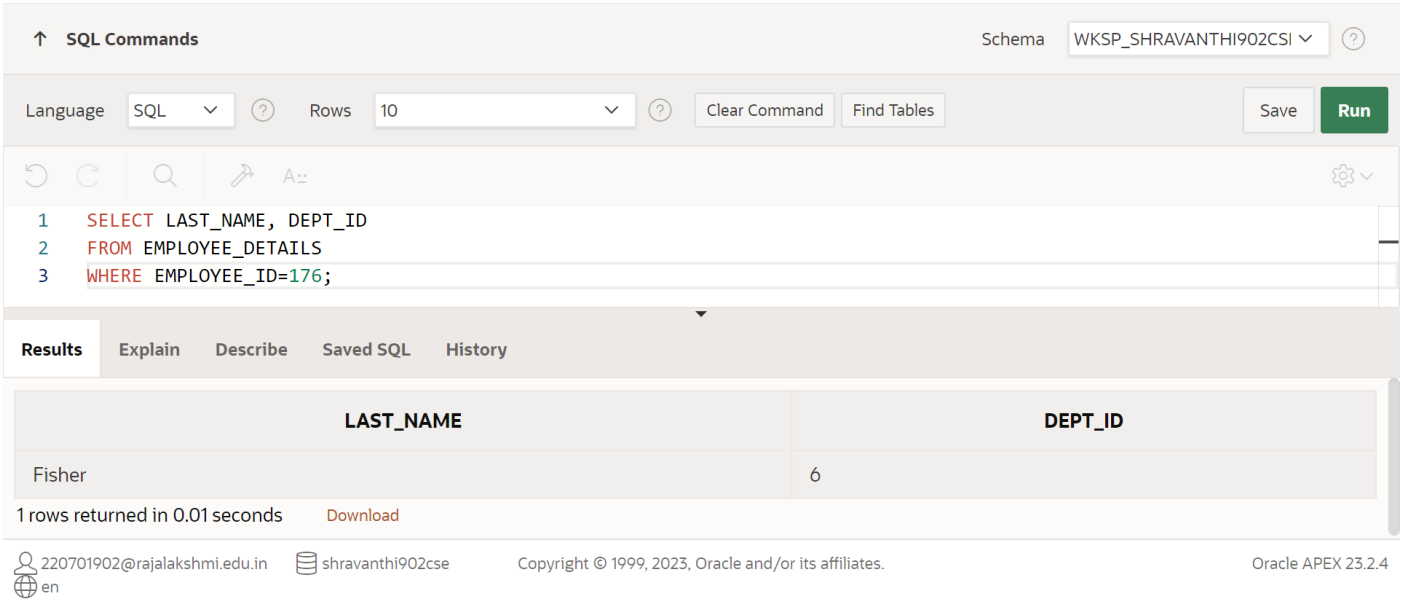


2. Create a query to display the employee last name and department number for employee number 176.

QUERY:

```
SELECT LAST_NAME,SALARY FROM EMPLOYEE_DETAILS WHERE EMPLOYEE_ID=176;
```

OUTPUT:

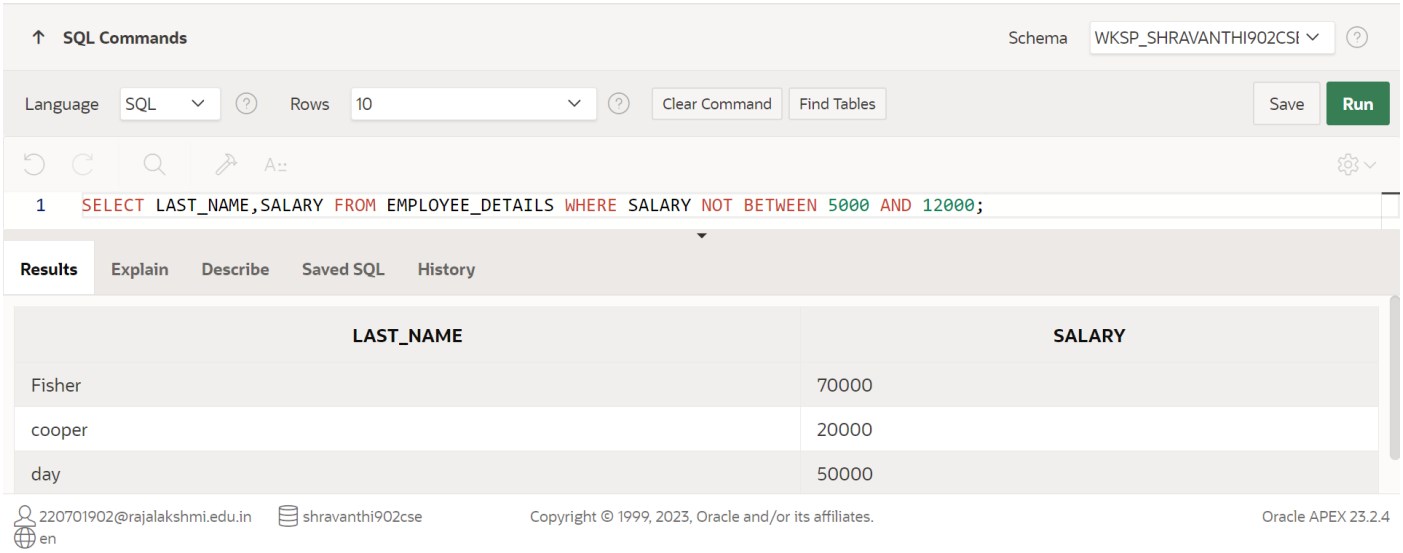


3. Create a query to display the last name and salary of employees whose salary is not in the range of 5000 and 12000. (hints: not between)

QUERY:

```
SELECT LAST_NAME,SALARY FROM EMPLOYEE_DETAILS WHERE SALARY NOT BETWEEN 5000 AND 12000;
```

OUTPUT:

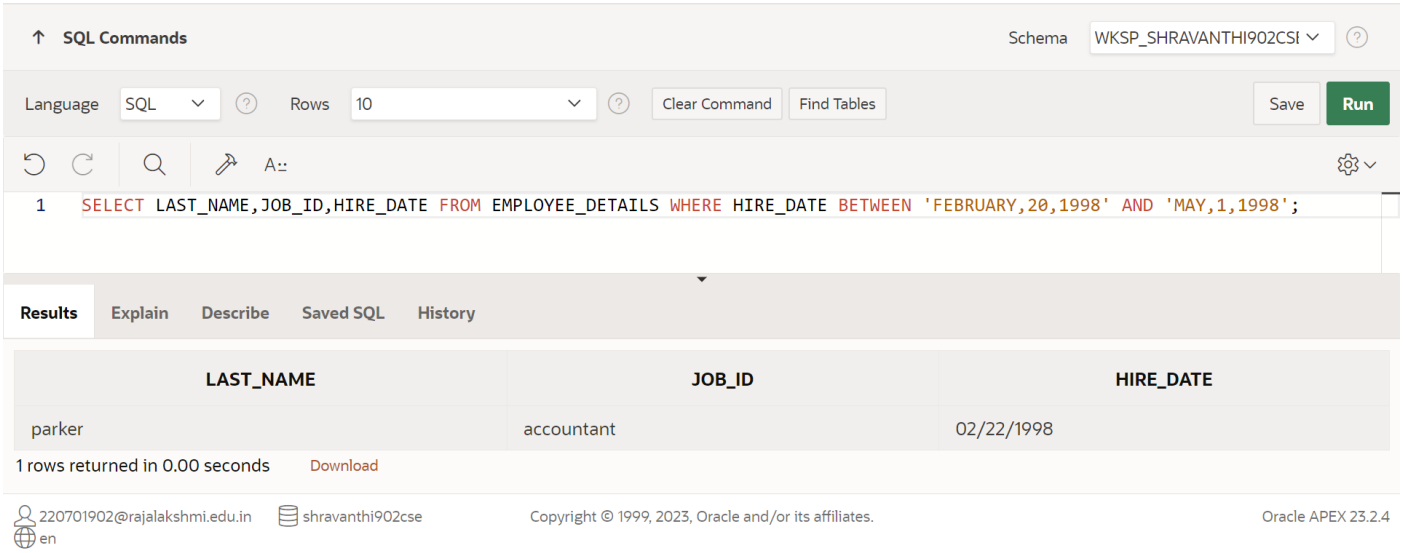


4. Display the employee last name, job ID, and start date of employees hired between February 20,1998 and May 1,1998.order the query in ascending order by start date.(hints: between)

QUERY:

```
SELECT LAST_NAME,JOB_ID,HIRE_DATE FROM EMPLOYEE_DETAILS WHERE HIRE_DATE BETWEEN
‘FEBRUARY,20,1998’ AND ‘MAY,1,1998’;
```

OUTPUT:

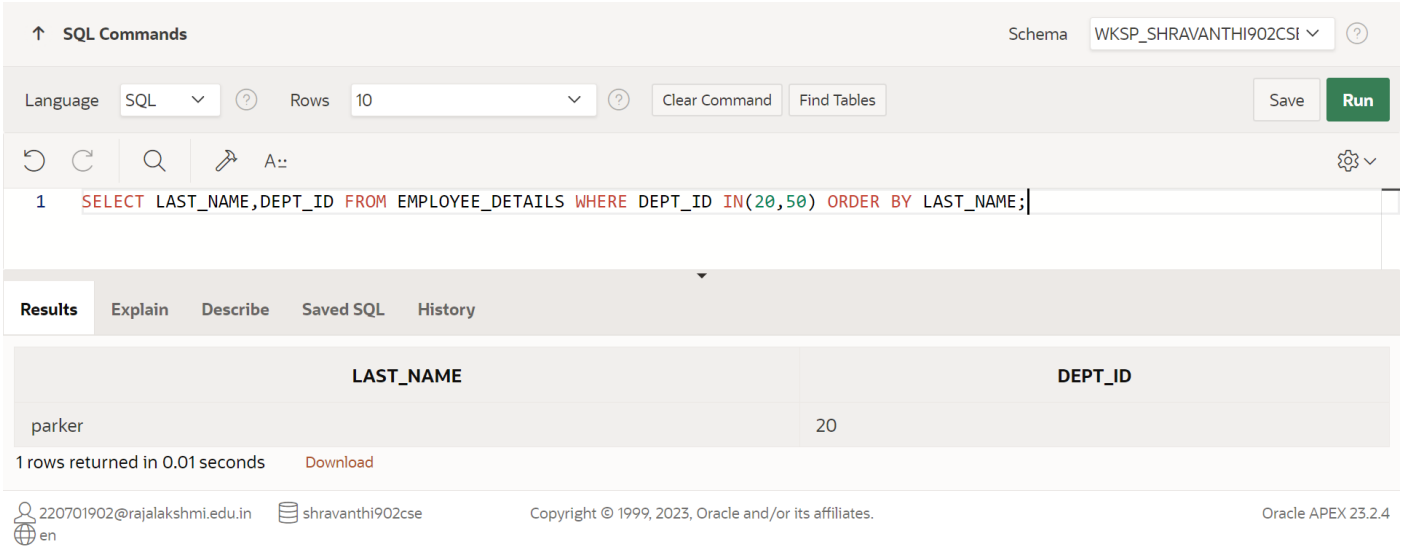


5. Display the last name and department number of all employees in departments 20 and 50 in alphabetical order by name.(hints: in, orderby)

QUERY:

```
SELECT LAST_NAME,DEPT_ID FROM EMPLOYEE_DETAILS WHERE DEPT_ID IN(20,50)
ORDER BY LAST_NAME;
```

OUTPUT:

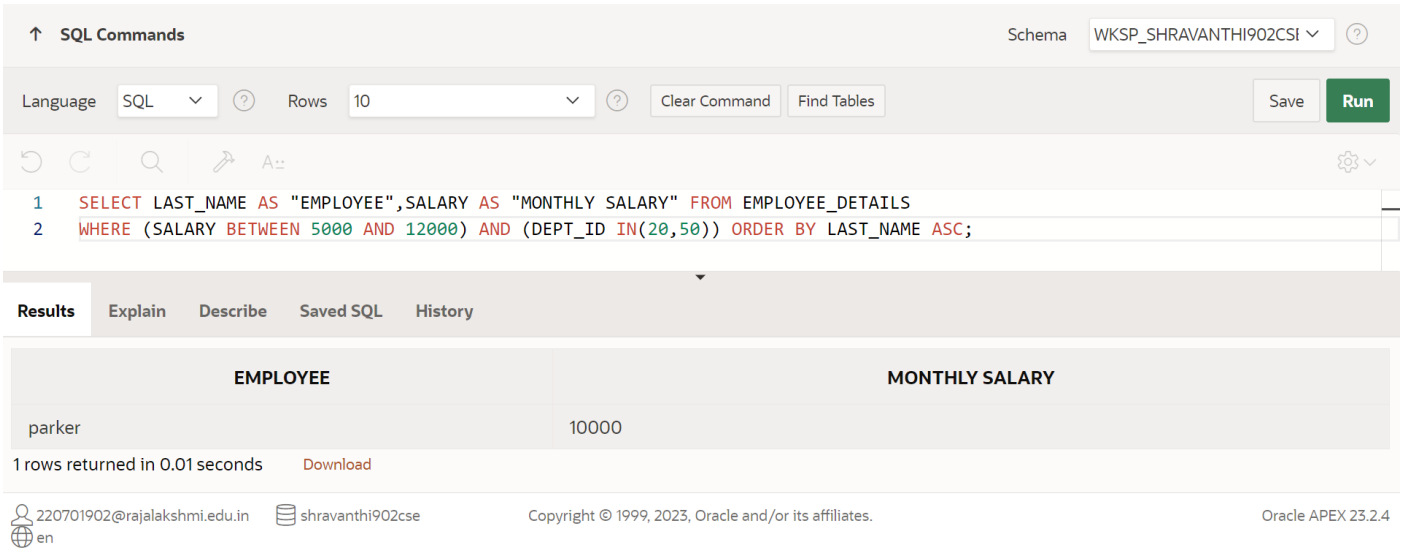


6. Display the last name and salary of all employees who earn between 5000 and 12000 and are in departments 20 and 50 in alphabetical order by name. Label the columns EMPLOYEE, MONTHLY SALARY respectively.(hints: between, in)

QUERY:

```
SELECT LAST_NAME AS "EMPLOYEE",SALARY AS "MONTHLY SALARY" FROM EMPLOYEE_DETAILS
WHERE (SALARY BETWEEN 5000 AND 12000) AND (DEPT_ID IN(20,50)) ORDER BY LAST_NAME
ASC;
```

OUTPUT:

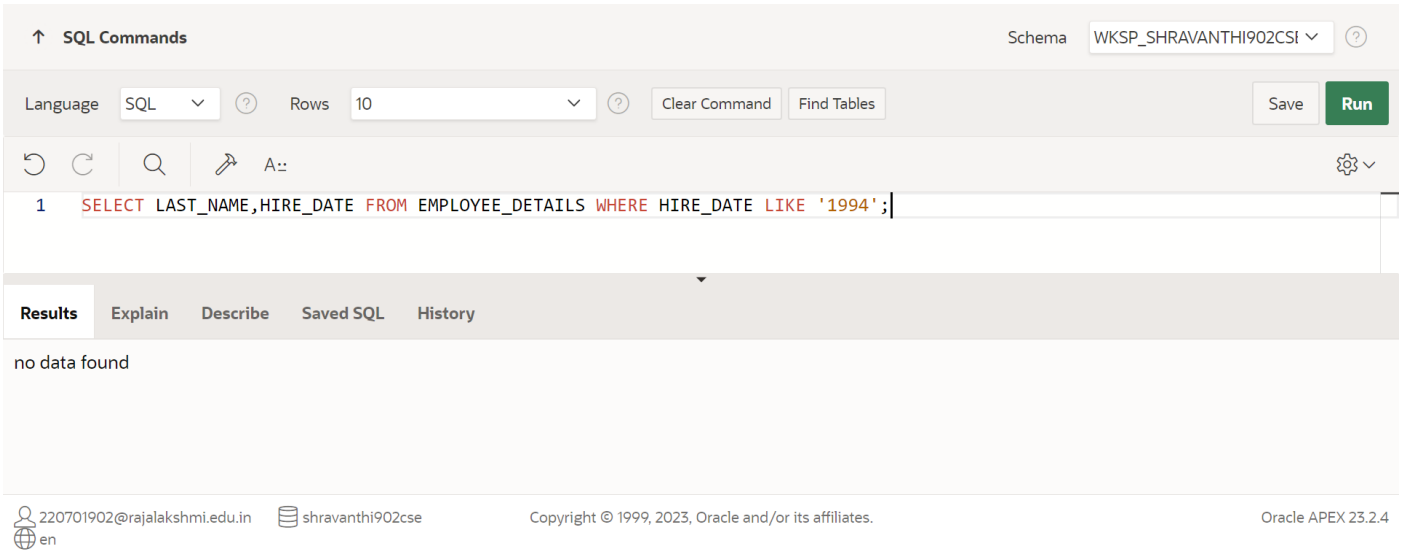


7. Display the last name and hire date of every employee who was hired in 1994.(hints: like)

QUERY:

```
SELECT LAST_NAME,HIRE_DATE FROM EMPLOYEE_DETAILS WHERE HIRE_DATE LIKE '1994';
```

OUTPUT:

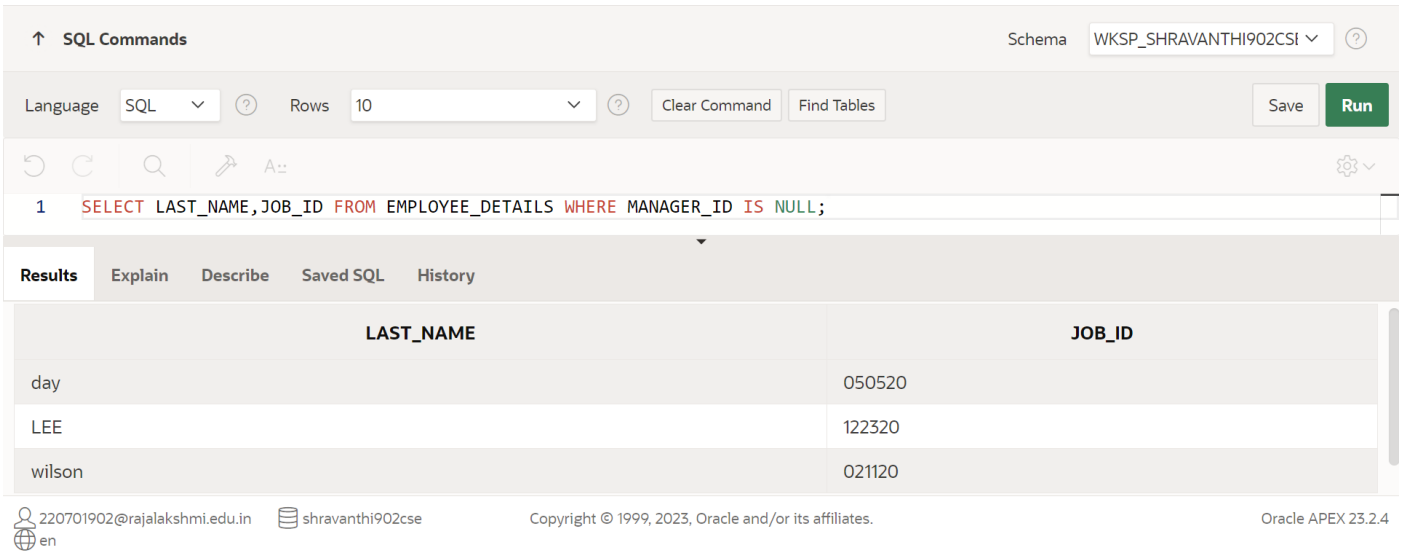


8. Display the last name and job title of all employees who do not have a manager.(hints: is null)

QUERY:

```
SELECT LAST_NAME,JOB_ID FROM EMPLOYEE_DETAILS WHERE MANAGER_ID IS NULL;
```

OUTPUT:

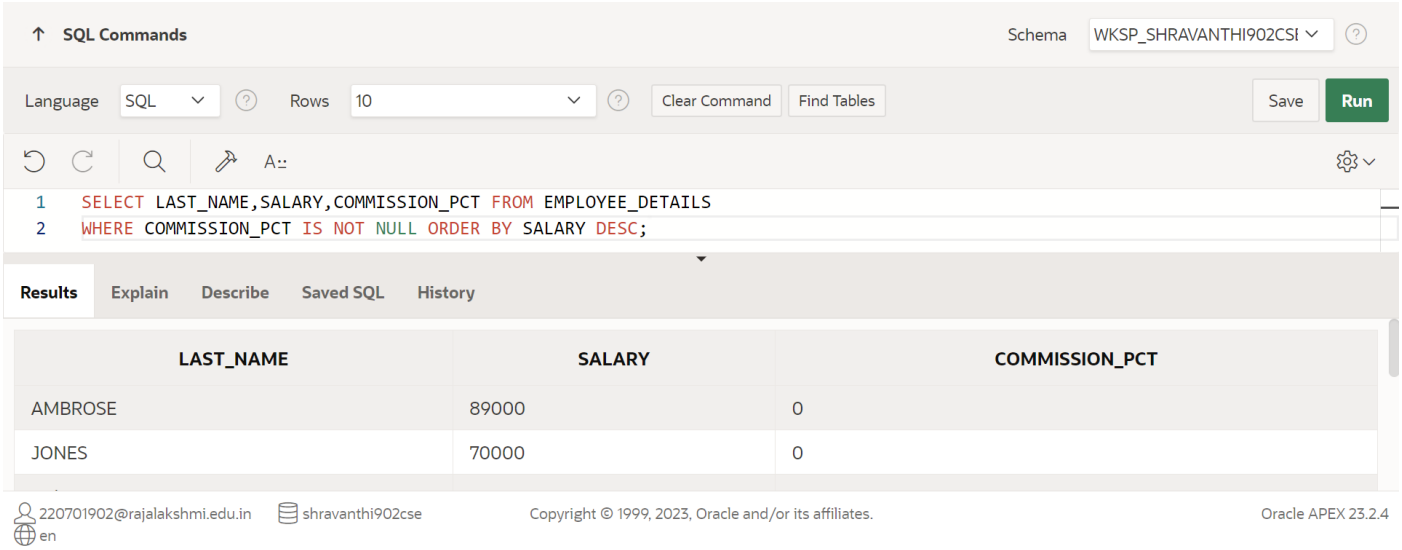


9. Display the last name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions.(hints: is not nul,orderby)

QUERY:

```
SELECT LAST_NAME,SALARY,COMMISSION_PCT FROM EMPLOYEE_DETAILS
WHERE COMMISSION_PCT IS NOT NULL ORDER BY SALARY DESC;
```

OUTPUT:

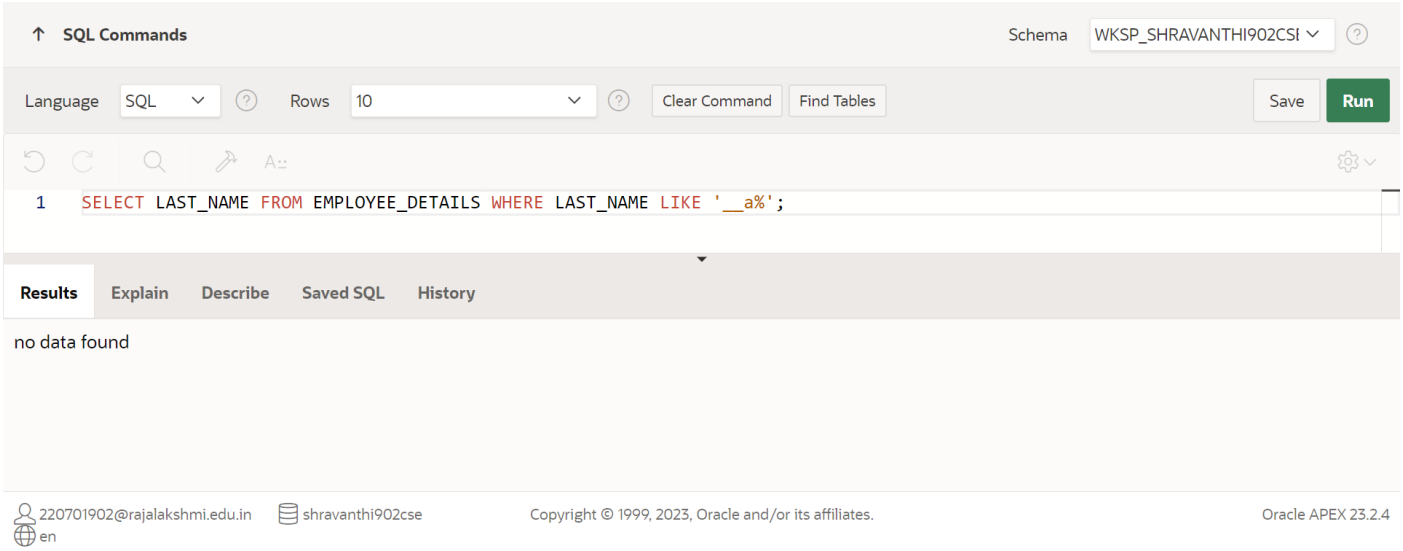


10. Display the last name of all employees where the third letter of the name is ***a***.(hints:like)

QUERY:

```
SELECT LAST_NAME FROM EMPLOYEE_DETAILS WHERE LAST_NAME LIKE '__a%';
```

OUTPUT:

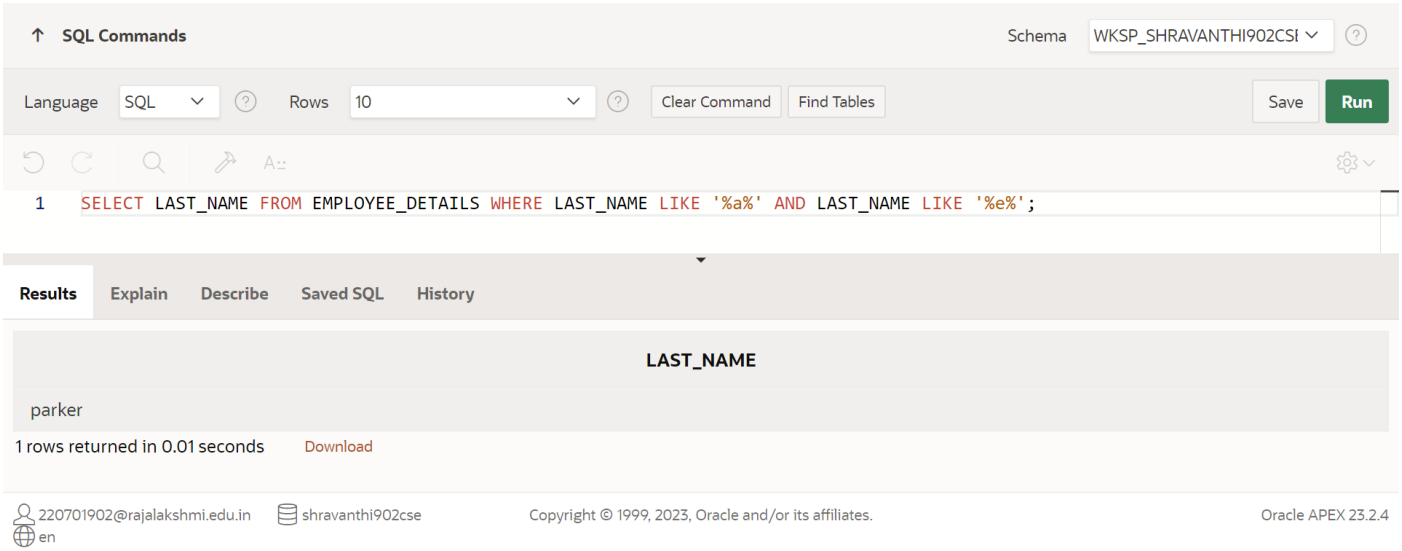


11. Display the last name of all employees who have an a and an ***e*** in their last name.(hints: like)

QUERY:

```
SELECT LAST_NAME FROM EMPLOYEE_DETAILS WHERE LAST_NAME LIKE '%a%' AND
LAST_NAME LIKE '%e%';
```

OUTPUT:

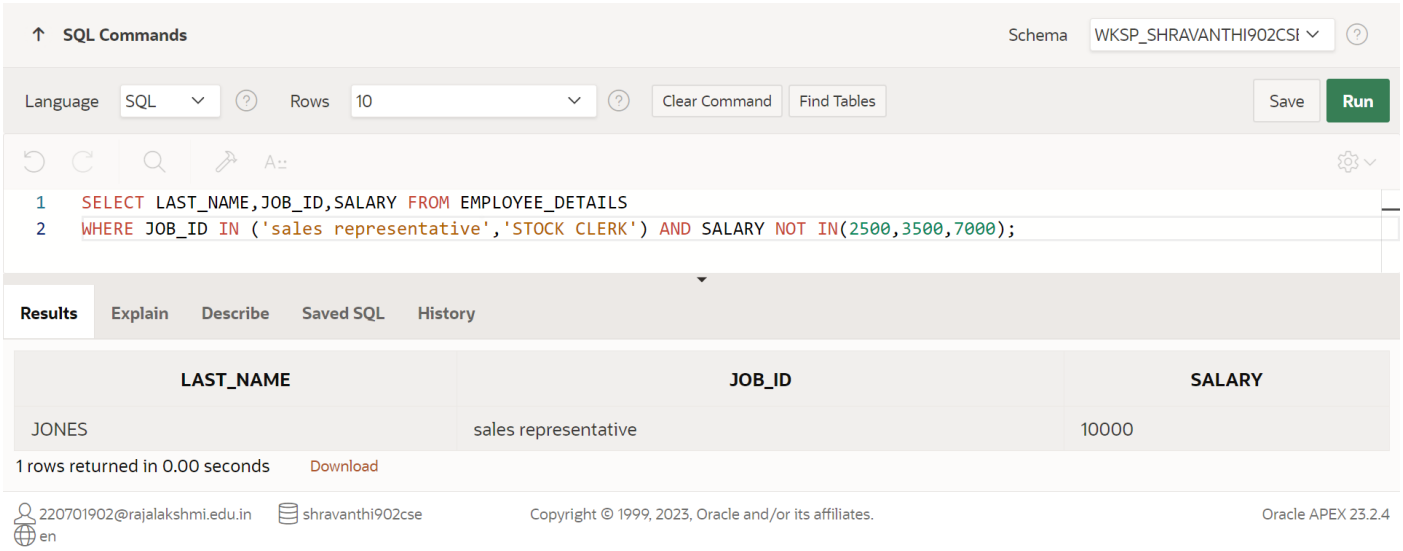


12. Display the last name and job and salary for all employees whose job is sales representative or stock clerk and whose salary is not equal to 2500 ,3500 or 7000.(hints:in,not in)

QUERY:

```
SELECT LAST_NAME,JOB_ID,SALARY FROM EMPLOYEE_DETAILS WHERE JOB_ID IN ('SALES
REPRESENTATIVE','STOCK CLERK') AND SALARY NOT IN(2500,3500,7000);
```

OUTPUT:

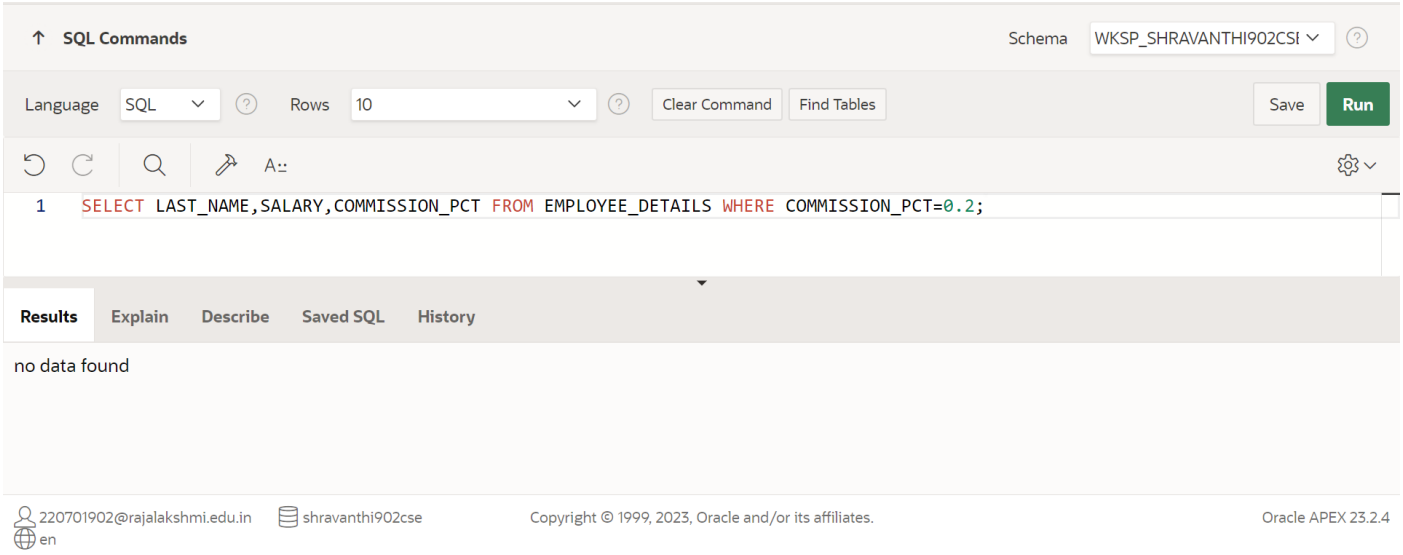


13. Display the last name, salary, and commission for all employees whose commission amount is 20%.(hints:use predicate logic)

QUERY:

```
SELECT LAST_NAME,SALARY,COMMISSION_PCT FROM EMPLOYEE_DETAILS WHERE
COMMISSION_PCT=0.2;
```

OUTPUT:



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

RESULT:

SINGLE ROW FUNCTIONS

EX.NO.6 **DATE:**

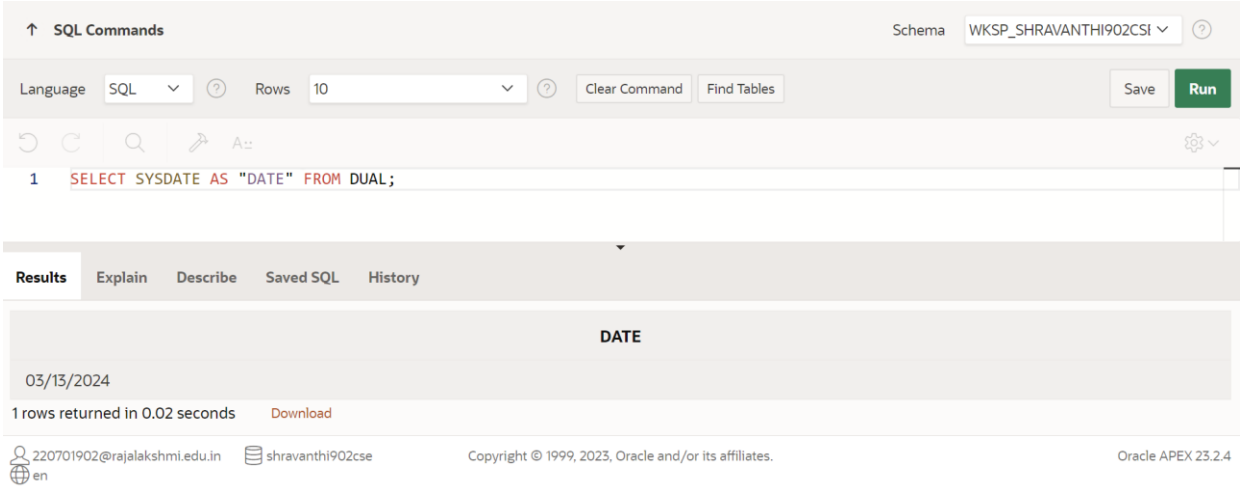
Find the Solution for the following:

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

QUERY:

```
SELECT SYSDATE AS “DATE” FROM DUAL;
```

OUTPUT:

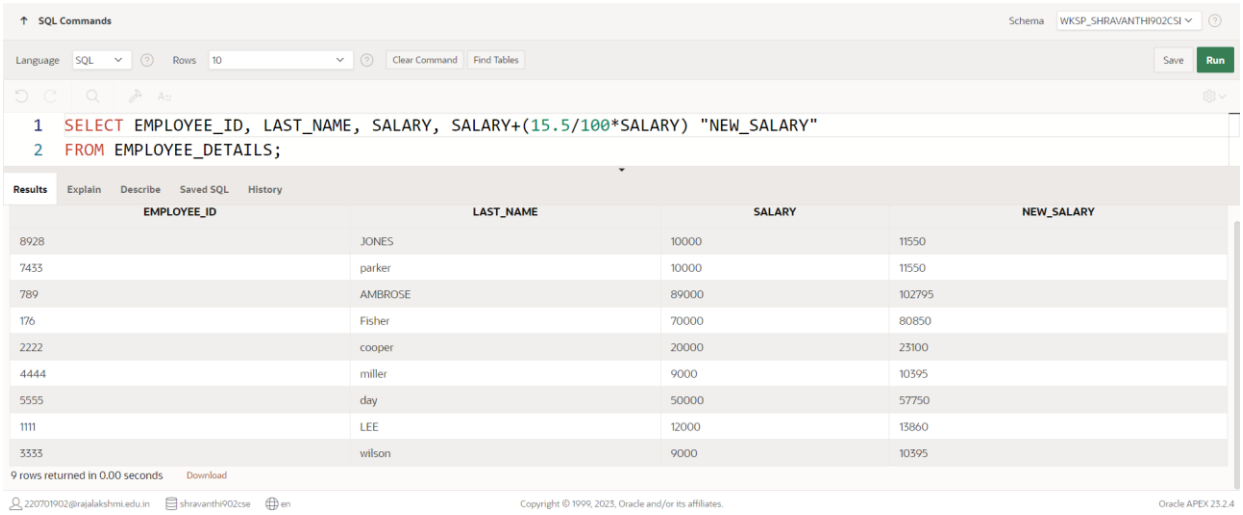


2. The HR department needs a report to display the employee number, last name, salary,and increased by 15.5% (expressed as a whole number) for each employee. Label the column New Salary.

QUERY:

```
SELECT EMPLOYEE_ID, LAST_NAME, SALARY,
SALARY+(15.5/100*SALARY) “NEW_SALARY” FROM
EMPLOYEE_DETAILS;
```

OUTPUT:

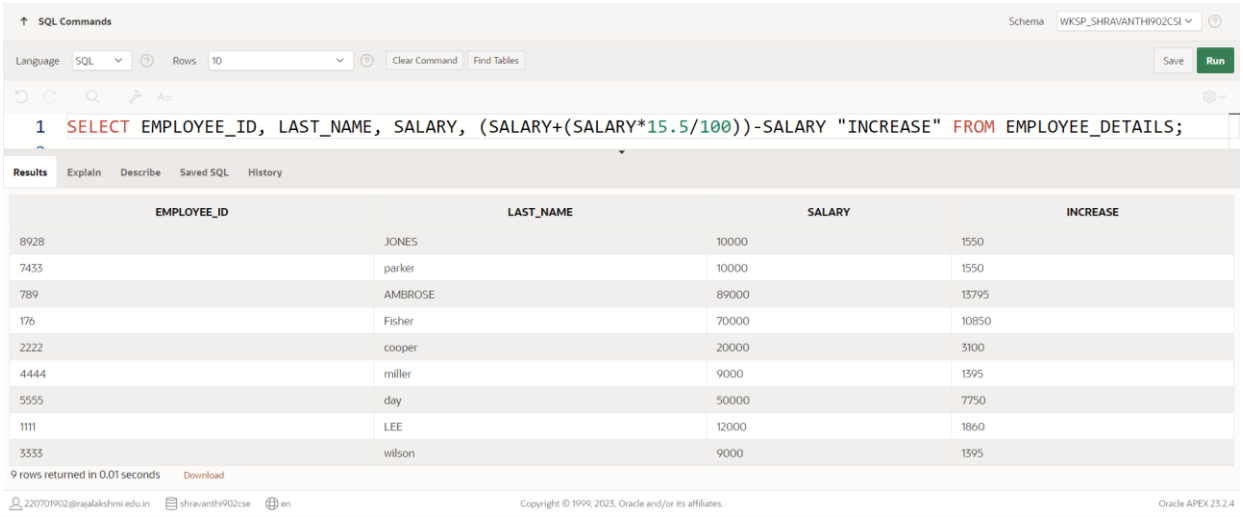


3.Modify your query lab_03_02.sql to add a column that subtracts the old salary fromthe new salary. Label the column Increase.

QUERY:

```
SELECT EMPLOYEE_ID, LAST_NAME, SALARY,
(SALARY+(SALARY*15.5/100))-SALARY “INCREASE” FROM
EMPLOYEE_DETAILS;
```

OUTPUT:



SQL Commands — Schema: WKSP_SHRAVANTHI902CSI

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

```
1  SELECT EMPLOYEE_ID, LAST_NAME, SALARY, (SALARY+(SALARY*15.5/100))-SALARY "INCREASE" FROM EMPLOYEE_DETAILS;
```

Results | Explain | Describe | Saved SQL | History

EMPLOYEE_ID	LAST_NAME	SALARY	INCREASE
8928	JONES	10000	1550
7433	parker	10000	1550
789	AMBROSE	89000	13795
176	Fisher	70000	10850
2222	cooper	20000	3100
4444	miller	9000	1395
5555	day	50000	7750
1111	LEE	12000	1860
3333	wilson	9000	1395

9 rows returned in 0.01 seconds Download

220701902@rajalakshmi.edu.in | shravanthi902cse | en

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

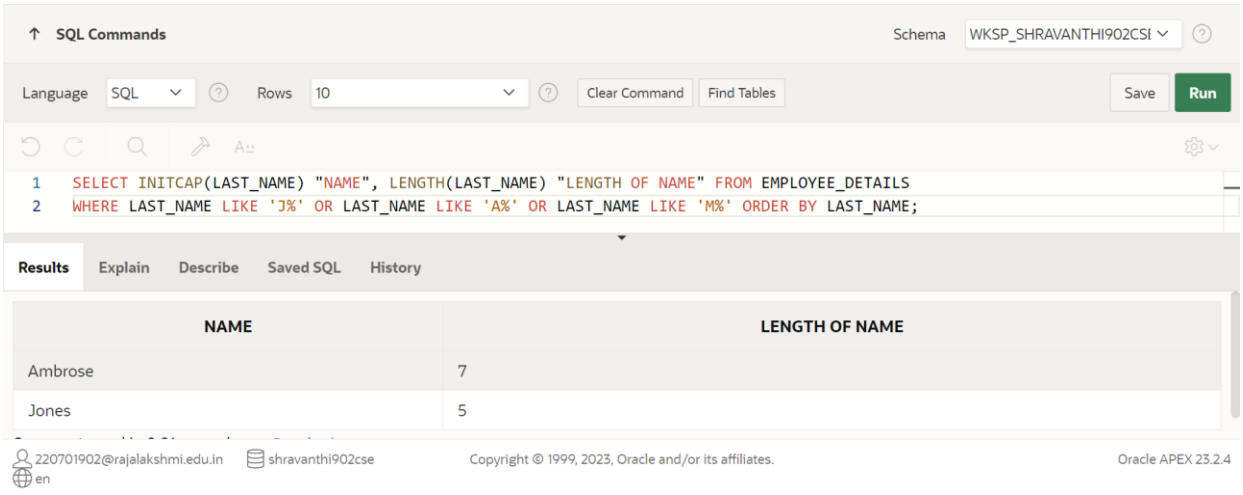
Oracle APEX 23.2.4

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

QUERY:

```
SELECT INITCAP(LAST_NAME) "NAME", LENGTH(LAST_NAME)
"LENGTH OF NAME" FROM EMPLOYEE_DETAILS
WHERE LAST_NAME LIKE 'J%' OR LAST_NAME LIKE 'A%' OR
LAST_NAME LIKE 'M%' ORDER BY LAST_NAME;
```

OUTPUT:



SQL Commands — Schema: WKSP_SHRAVANTHI902CSI

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

```
1  SELECT INITCAP(LAST_NAME) "NAME", LENGTH(LAST_NAME) "LENGTH OF NAME" FROM EMPLOYEE_DETAILS
2  WHERE LAST_NAME LIKE 'J%' OR LAST_NAME LIKE 'A%' OR LAST_NAME LIKE 'M%' ORDER BY LAST_NAME;
```

Results | Explain | Describe | Saved SQL | History

NAME	LENGTH OF NAME
Ambrose	7
Jones	5

220701902@rajalakshmi.edu.in | shravanthi902cse | en

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

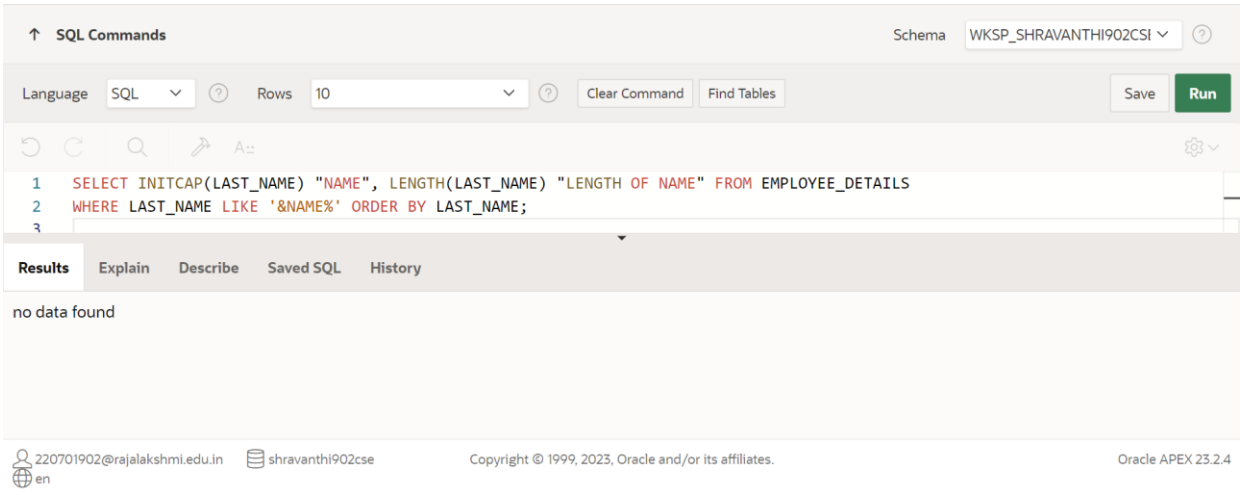
Oracle APEX 23.2.4

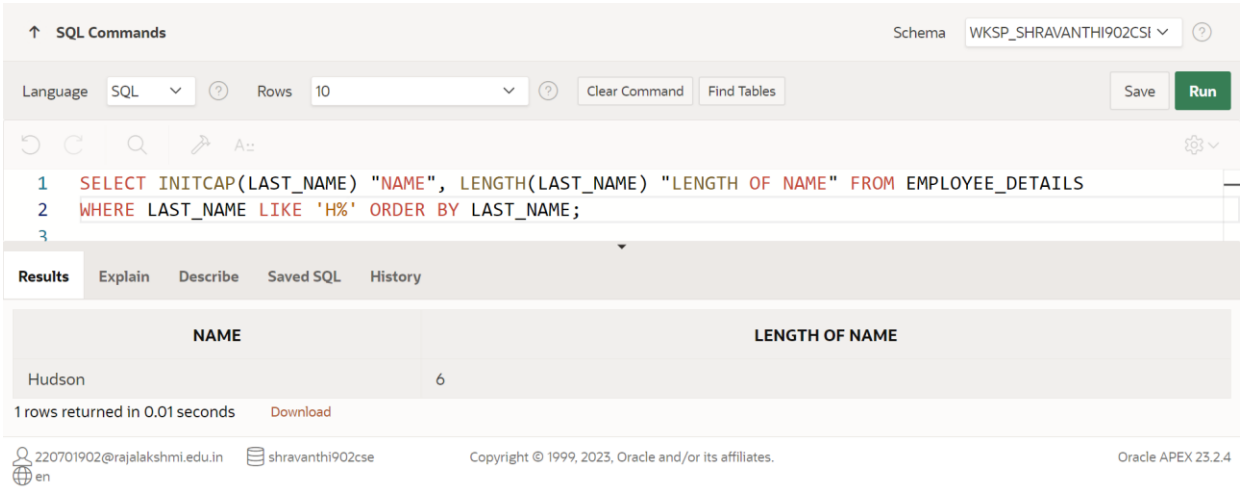
5.Rewrite the query so that the user is prompted to enter a letter that starts the last name. For example, if the user enters H when prompted for a letter, then the output should show all employees whose last name starts with the letter H.

QUERY:

```
SELECT INITCAP(LAST_NAME) "NAME", LENGTH(LAST_NAME)
"LENGTH OF NAME" FROM EMPLOYEE_DETAILS
WHERE LAST_NAME LIKE
'&NAME%' ORDER BY
LAST_NAME;
```

OUTPUT:



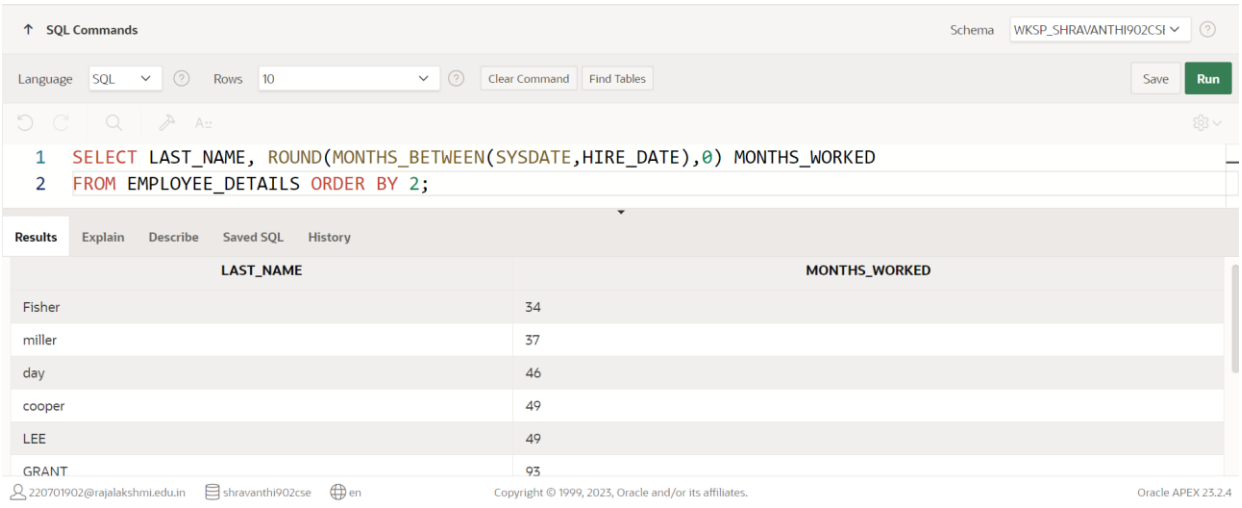


6.The HR department wants to find the length of employment for each employee. For each employee, display the last name and calculate the number of months between today andthe date on which the employee was hired. Label the column MONTHS_WORKED. Order your results by the number of months employed. Round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

QUERY:

```
SELECT LAST_NAME,
ROUND(MONTHS_BETWEEN(SYSDATE,HIRE_DATE),0) MONTHS_WORKED
FROM EMPLOYEE_DETAILS ORDER BY 2;
```

OUTPUT:

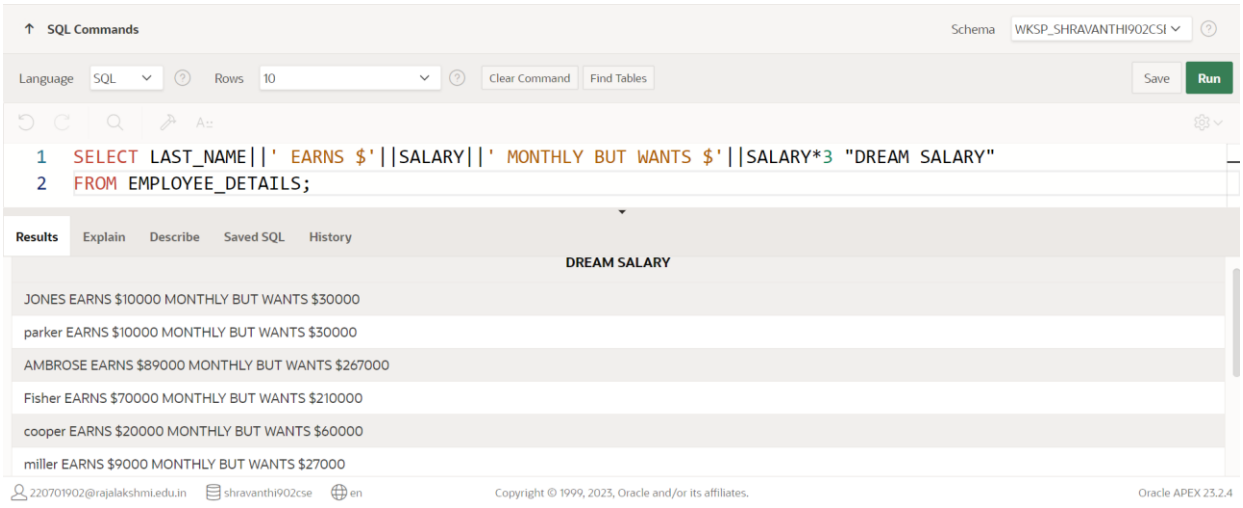


7.Create a report that produces the following for each employee:
<employee last name> earns <salary> monthly but wants <3 times salary>. Label the column Dream Salaries.

QUERY:

```
SELECT LAST_NAME||' EARNS $'||SALARY||' MONTHLY BUT WANTS
$'||SALARY*3 "DREAM SALARY" FROM EMPLOYEE_DETAILS;
```

OUTPUT:

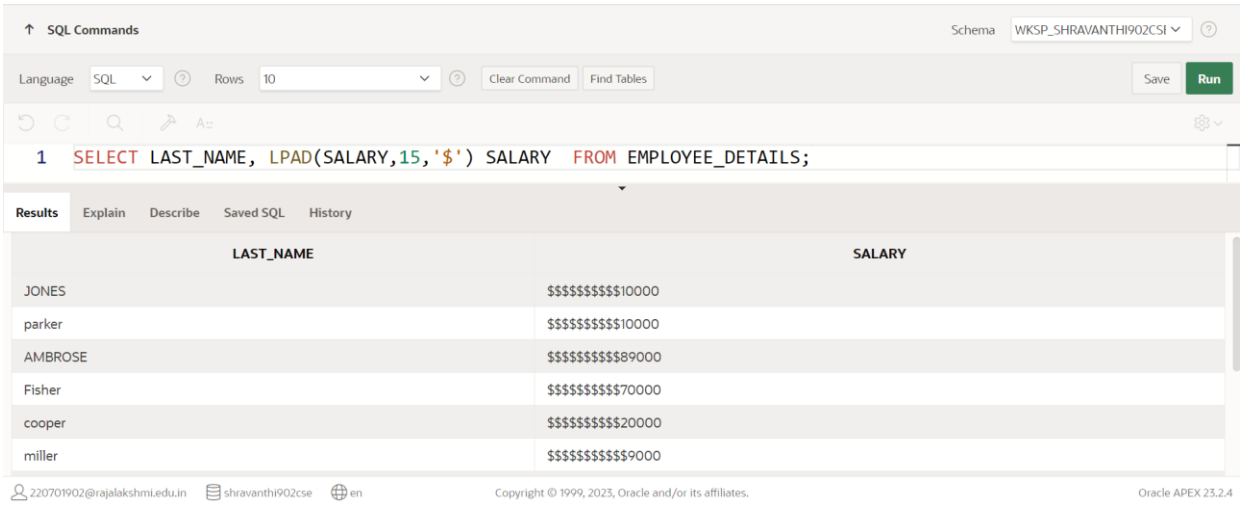


8.Create a query to display the last name and salary for all employees. Format the salaryto be 15 characters long, left-padded with the $ symbol. Label the column SALARY.

QUERY:

```
SELECT LAST_NAME,
LPAD(SALARY,15,'$') SALARY
FROM EMPLOYEE_DETAILS;
```

OUTPUT:

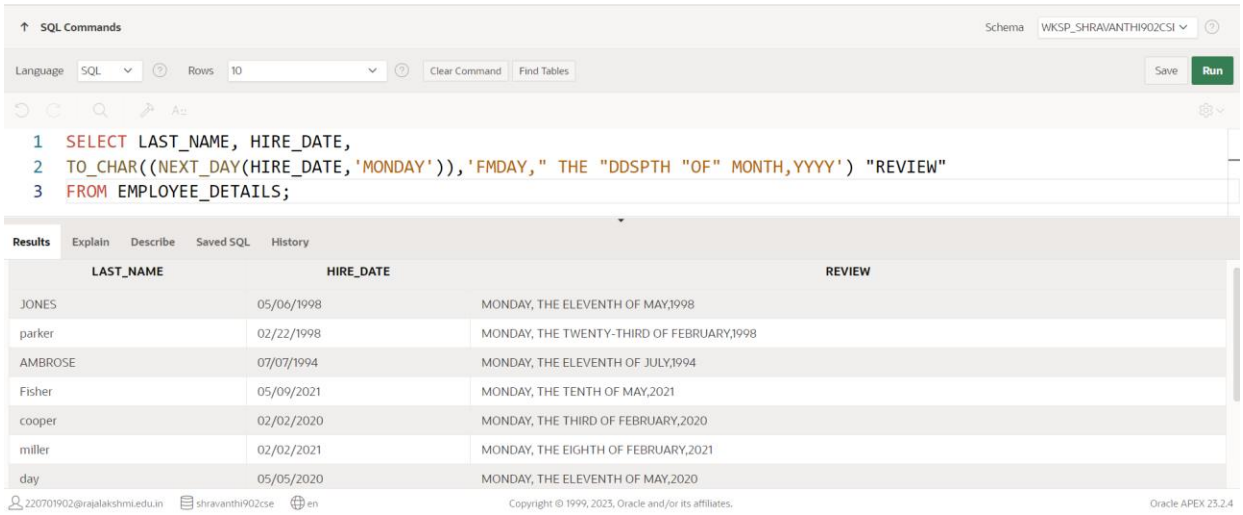


9.Display each employee's last name, hire date, and salary review date, which is the first Monday after six months of service. Label the column REVIEW. Format the dates to appear in the format similar to "Monday, the Thirty-First of July, 2000."

QUERY:

```
SELECT LAST_NAME, HIRE_DATE,
TO_CHAR((NEXT_DAY(HIRE_DATE,'MONDAY')),'FMDAY," THE "DDSPTH
"OF" MONTH,YYYY') "REVIEW" FROM EMPLOYEE_DETAILS;
```

OUTPUT:

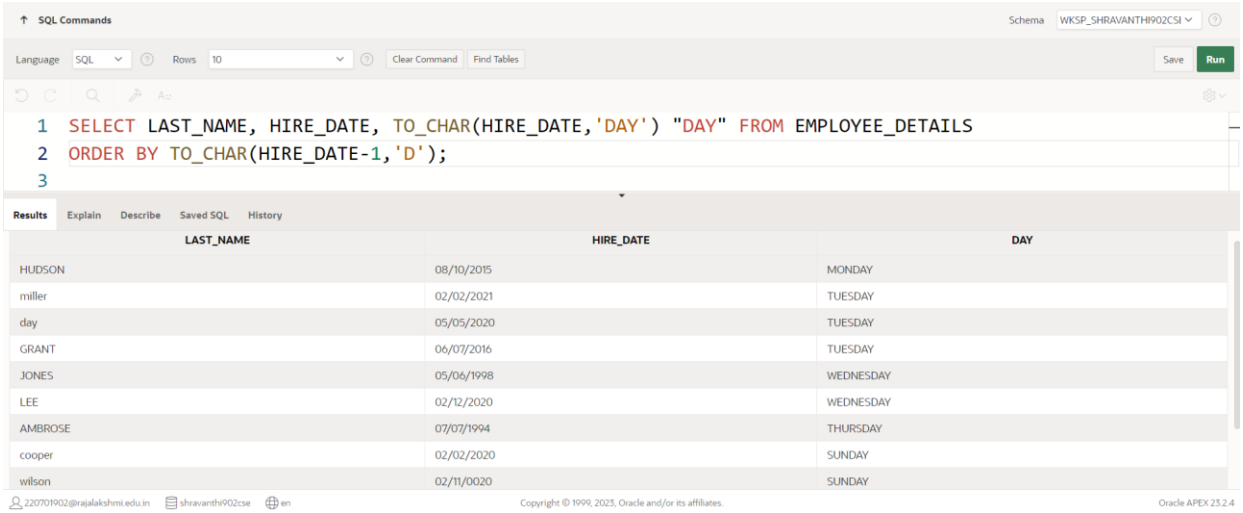


10.Display the last name, hire date, and day of the week on which the employee started. Label the column DAY. Order the results by the day of the week, starting with Monday.

QUERY:

```
SELECT LAST_NAME, HIRE_DATE, TO_CHAR(HIRE_DATE,'DAY') "DAY"
FROM EMPLOYEE_DETAILS
ORDER BY TO_CHAR(HIRE_DATE-1,'D');
```

OUTPUT:



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

RESULT:

DISPLAYING DATA FROM MULTIPLE TABLES

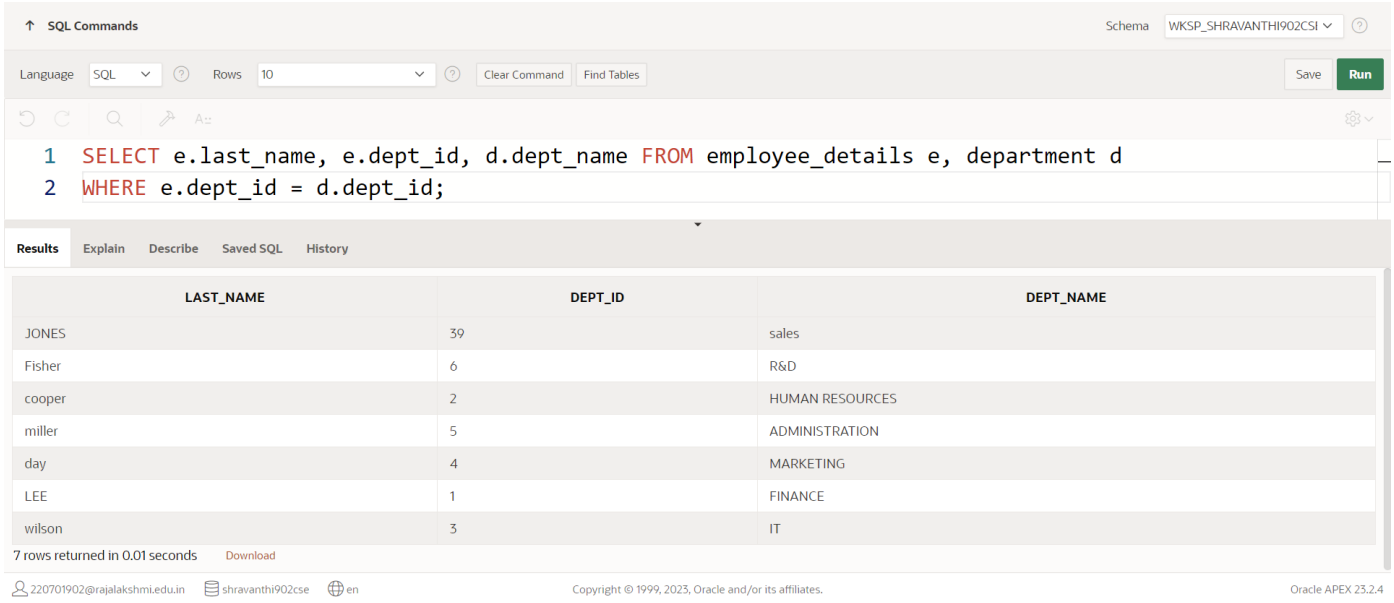
EX_NO:7 **DATE:**

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

QUERY:

```
SELECT e.last_name, e.dept_id, d.dept_name FROM employee_details e,
department d
WHERE e.dept_id = d.dept_id;
```

OUTPUT:

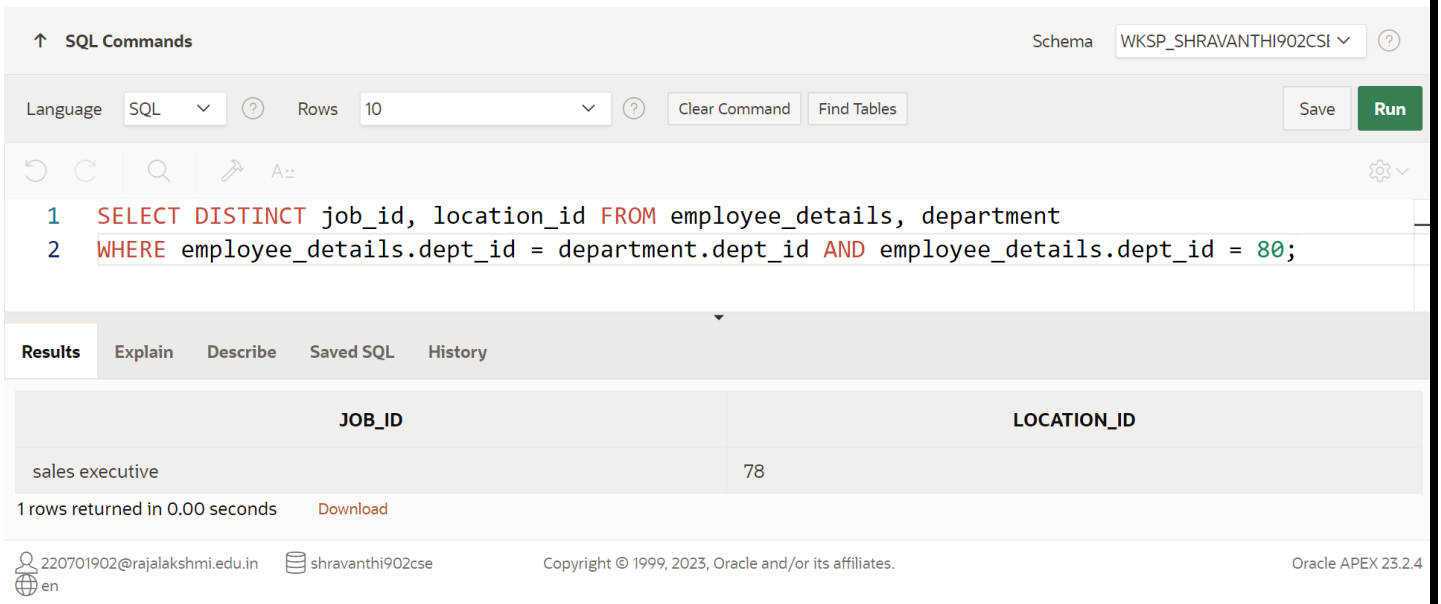


2. Create a unique listing of all jobs that are in department 80. Include the location of the department in the output.

QUERY:

```
SELECT DISTINCT job_id, location_id FROM employee_details, department
WHERE employee_details.dept_id = department.dept_id AND
employee_details.dept_id = 80;
```

OUTPUT:

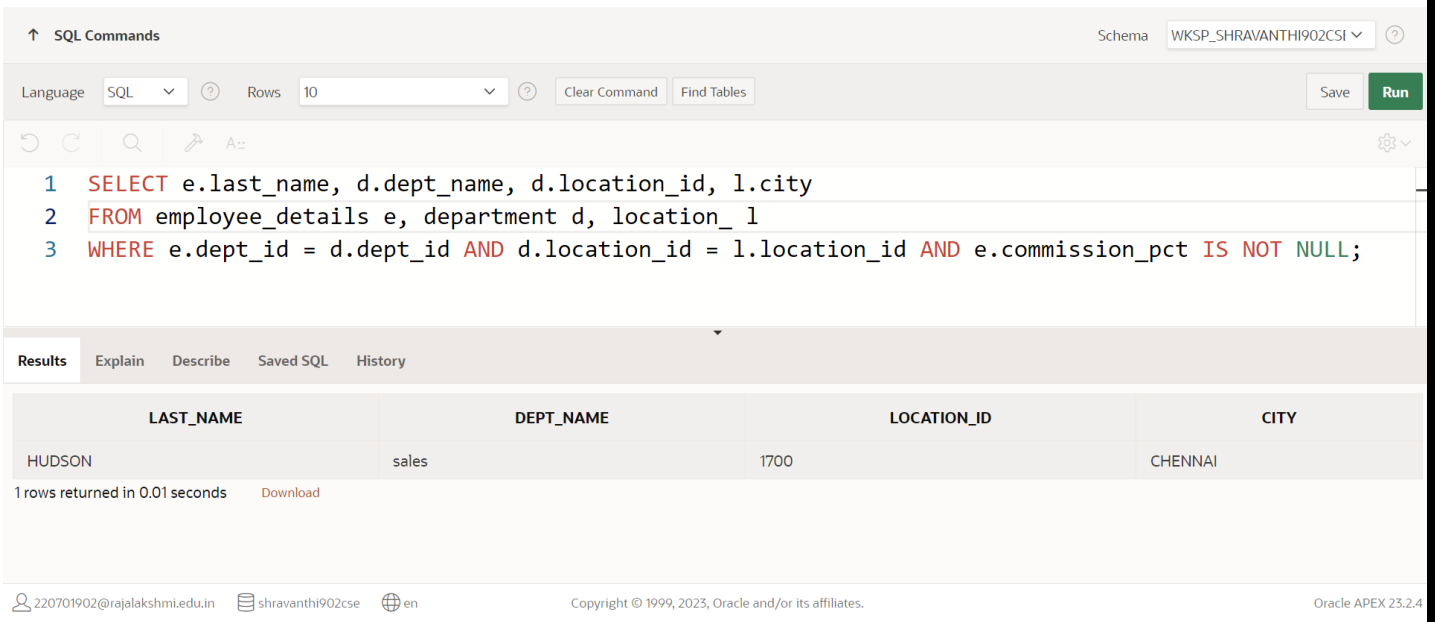


3. Write a query to display the employee last name, department name, location ID, and city of all employees who earn a commission

QUERY:

```
SELECT e.last_name, d.dept_name, d.location_id, l.city FROM
employee_details e, department d, location_ l WHERE e.dept_id
= d.dept_id AND d.location_id = l.location_id AND
e.commission_pct IS NOT NULL;
```

OUTPUT:

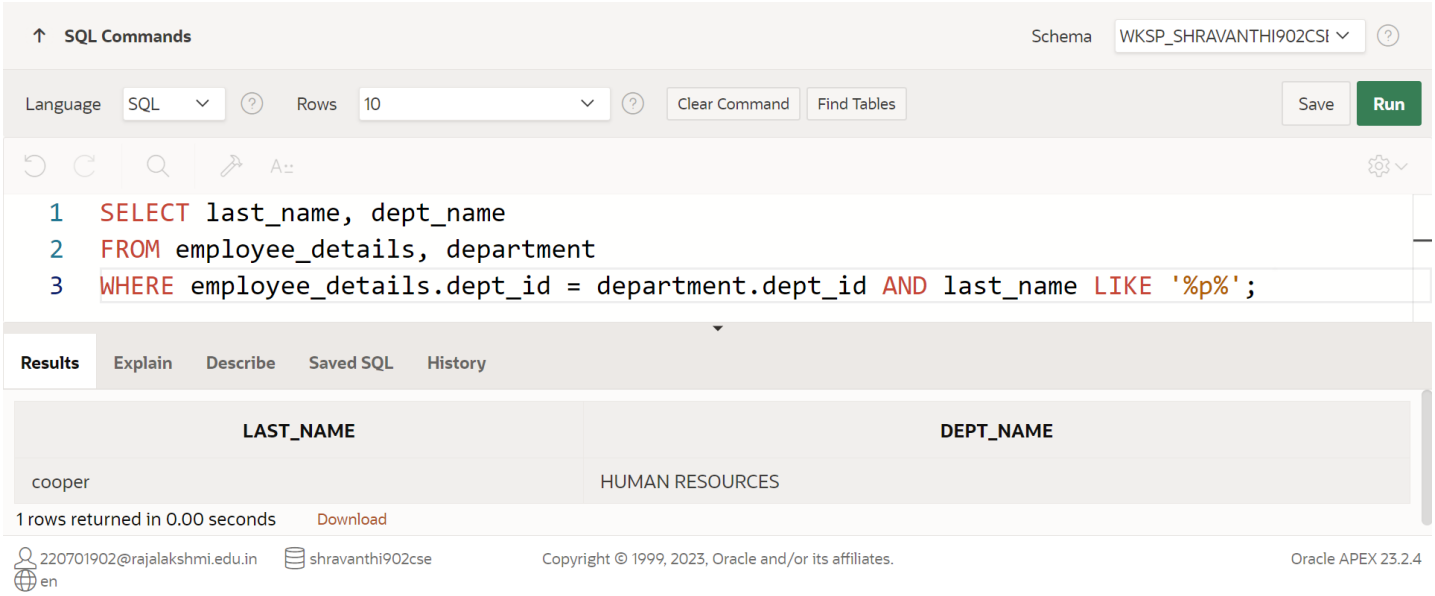


4. Display the employee last name and department name for all employees who have an a(lowercase) in their last names.

QUERY:

```
SELECT last_name, dept_name
FROM employee_details, department
WHERE employee_details.dept_id = department.dept_id AND last_name LIKE
'%p%';
```

OUTPUT:



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

QUERY:

```
SELECT e.last_name, e.job_id, e.dept_id,d.dept_name
FROM employee_details e JOIN department d ON (e.dept_id = d.dept_id)
JOIN location_ l ON (d.location_id = l.location_id)
WHERE LOWER(l.city) = 'toronto';
```

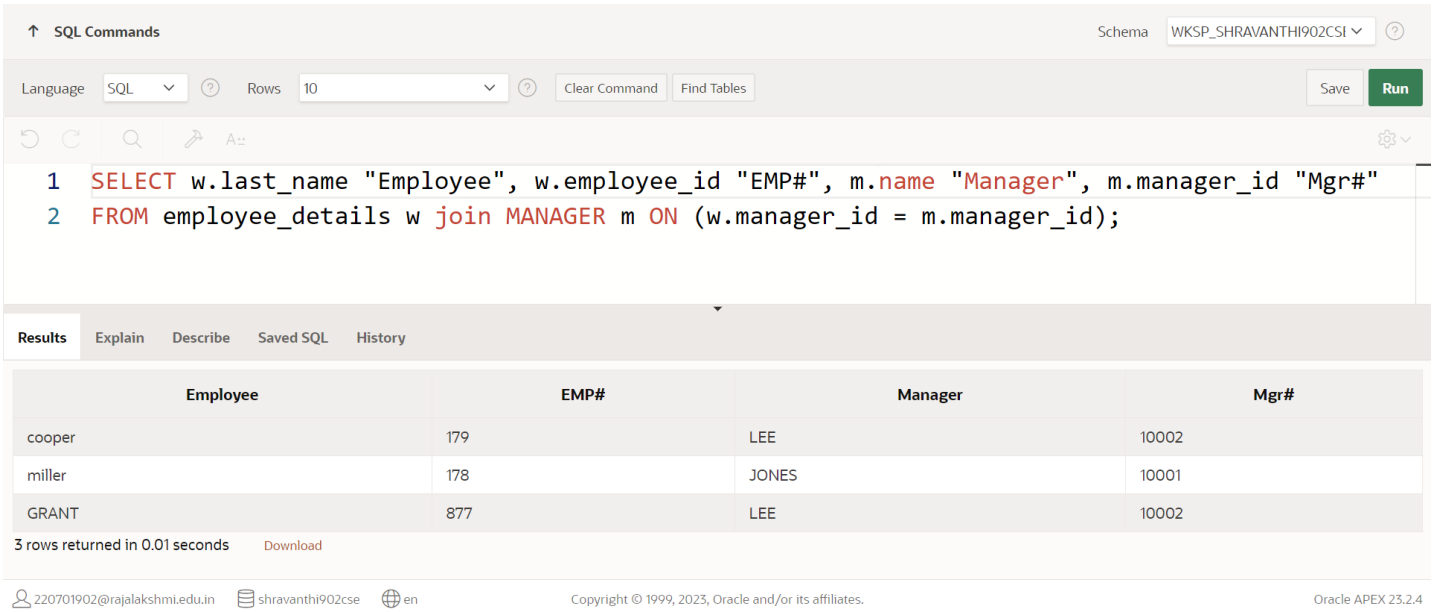
OUTPUT:

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

QUERY:

```
SELECT w.last_name "Employee", w.employee_id "EMP#", m.name "Manager",
m.manager_id "Mgr#"
FROM employee_details w join MANAGER m ON (w.manager_id = m.manager_id);
```

OUTPUT:

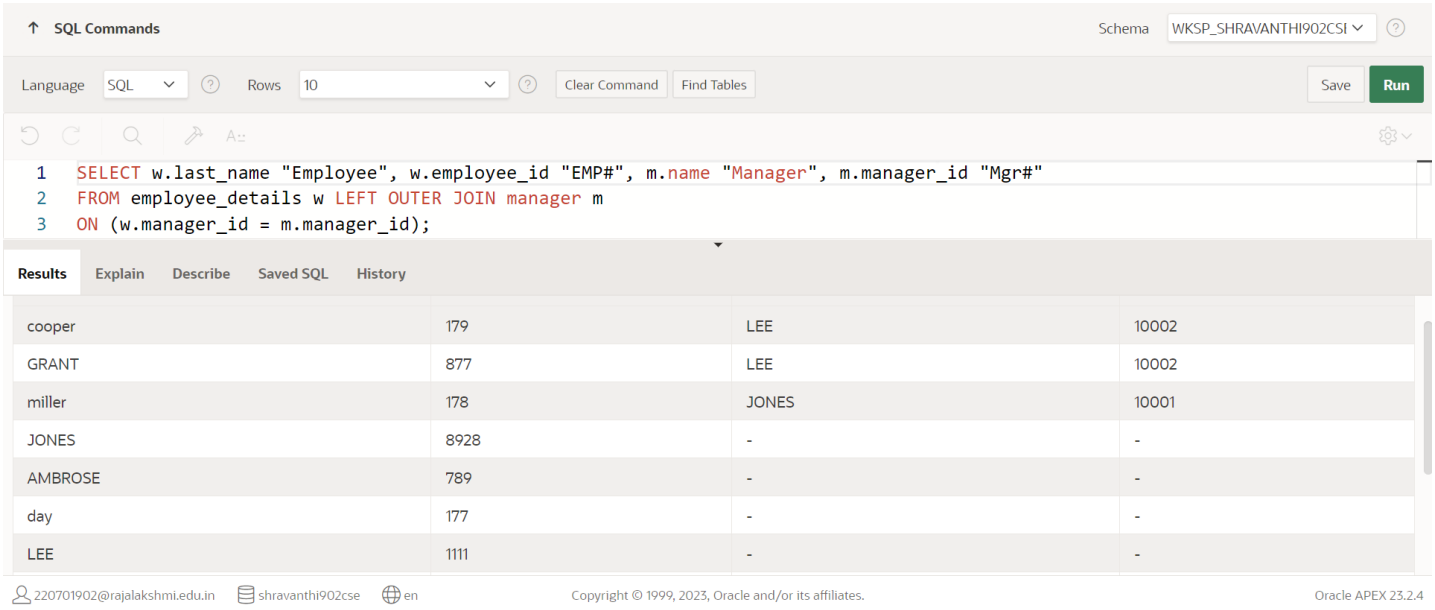


7. Modify lab4_6.sql to display all employees including King, who has no manager. Order the results by the employee number.

QUERY:

```
SELECT w.last_name "Employee", w.employee_id "EMP#", m.name "Manager",
m.manager_id "Mgr#"

FROM employee_details w LEFT OUTER JOIN manager m

ON (w.manager_id = m.manager_id);
```

OUTPUT:

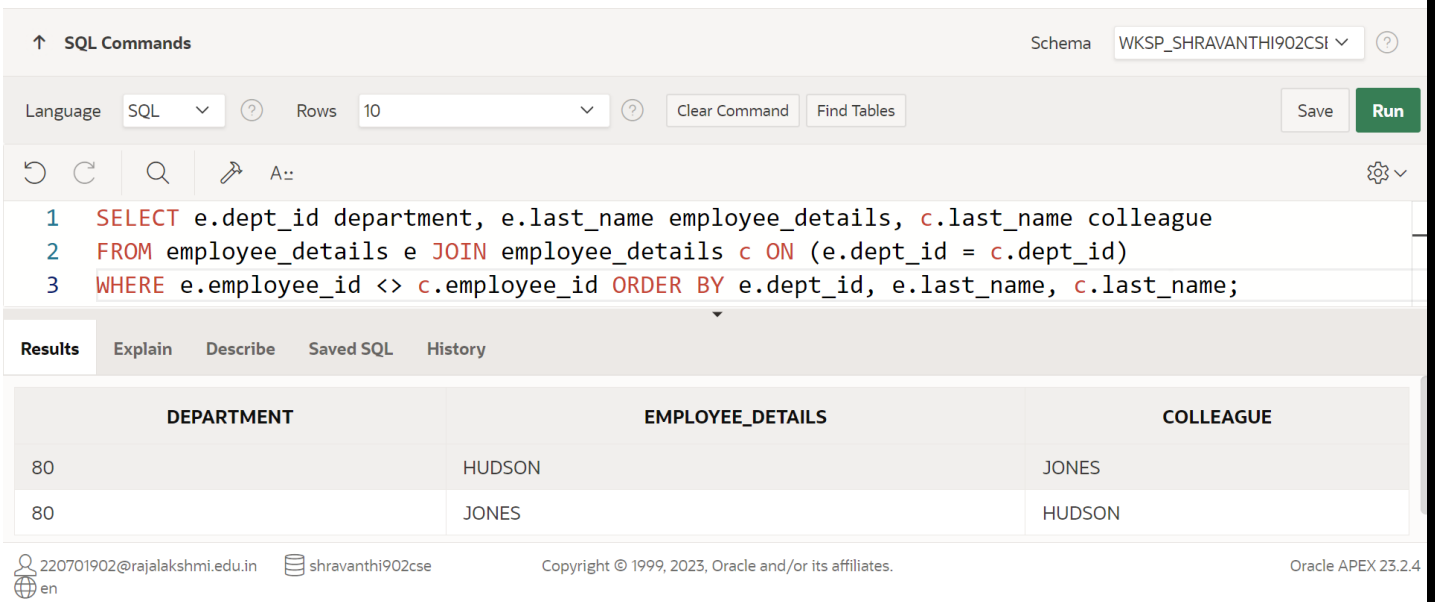


8. Create a query that displays employee last names, department numbers, and all the employees who work in the same department as a given employee. Give each column an appropriate label

QUERY:

```
SELECT e.dept_id department, e.last_name employee_details, c.last_name
colleague FROM employee_details e JOIN employee_details c ON (e.dept_id
= c.dept_id) WHERE e.employee_id <> c.employee_id ORDER BY e.dept_id,
e.last_name, c.last_name;
```

OUTPUT:

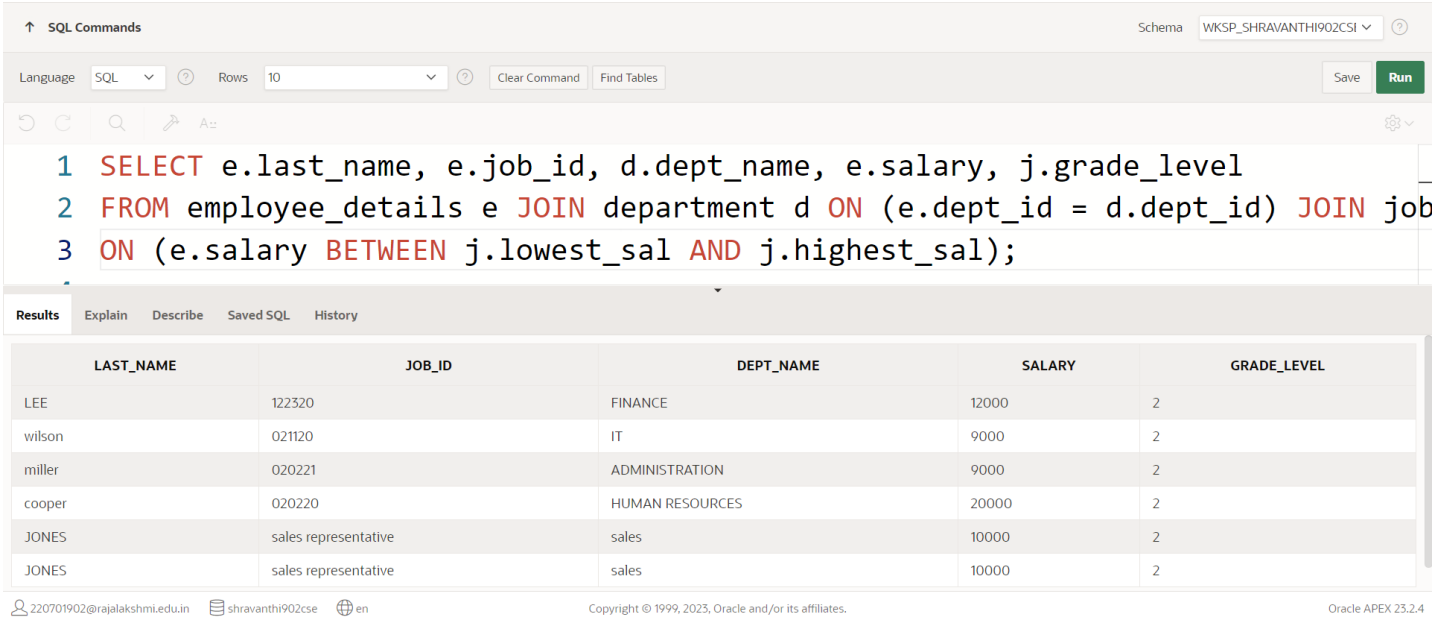


9. Show the structure of the JOB_GRADES table. Create a query that displays the name, job, department name, salary, and grade for all employees

QUERY:

```
SELECT e.last_name, e.job_id, d.dept_name, e.salary, j.grade_level
 FROM employee_details e JOIN department d ON (e.dept_id = d.dept_id)
 JOIN jobgrade j ON (e.salary BETWEEN j.lowest_sal AND j.highest_sal);
```

OUTPUT:



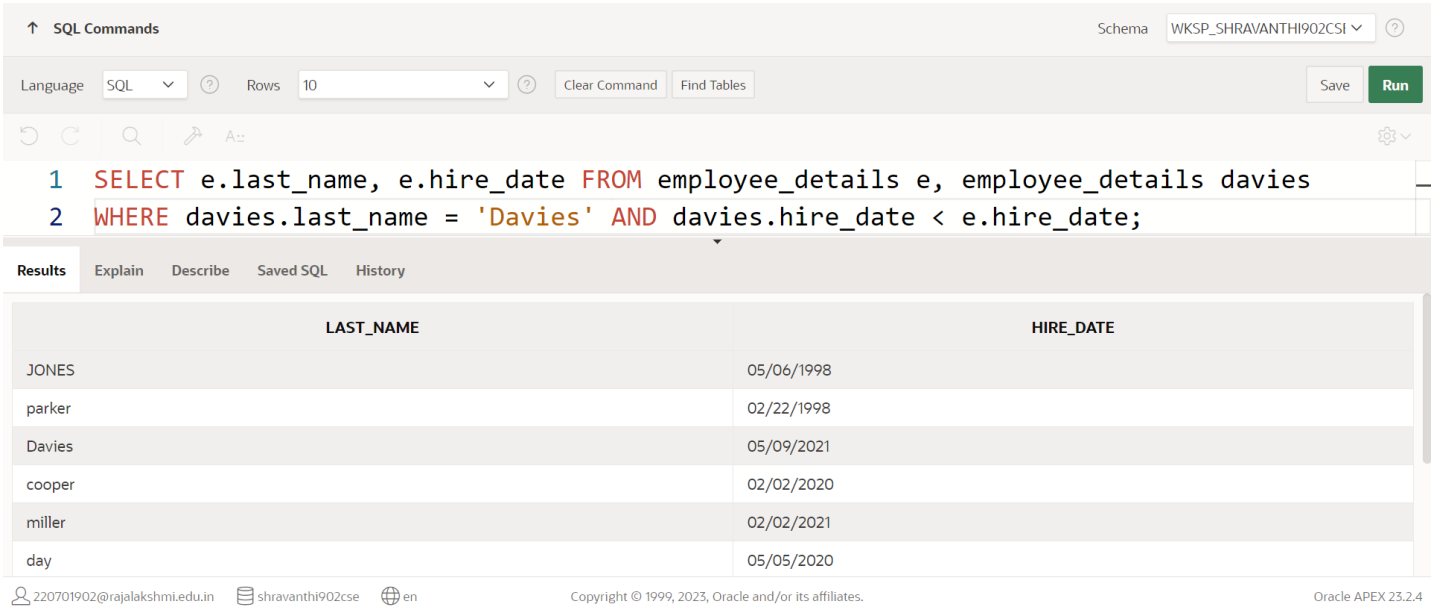
LAST_NAME	JOB_ID	DEPT_NAME	SALARY	GRADE_LEVEL
LEE	122320	FINANCE	12000	2
wilson	021120	IT	9000	2
miller	020221	ADMINISTRATION	9000	2
cooper	020220	HUMAN RESOURCES	20000	2
JONES	sales representative	sales	10000	2
JONES	sales representative	sales	10000	2

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

QUERY:

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

OUTPUT:



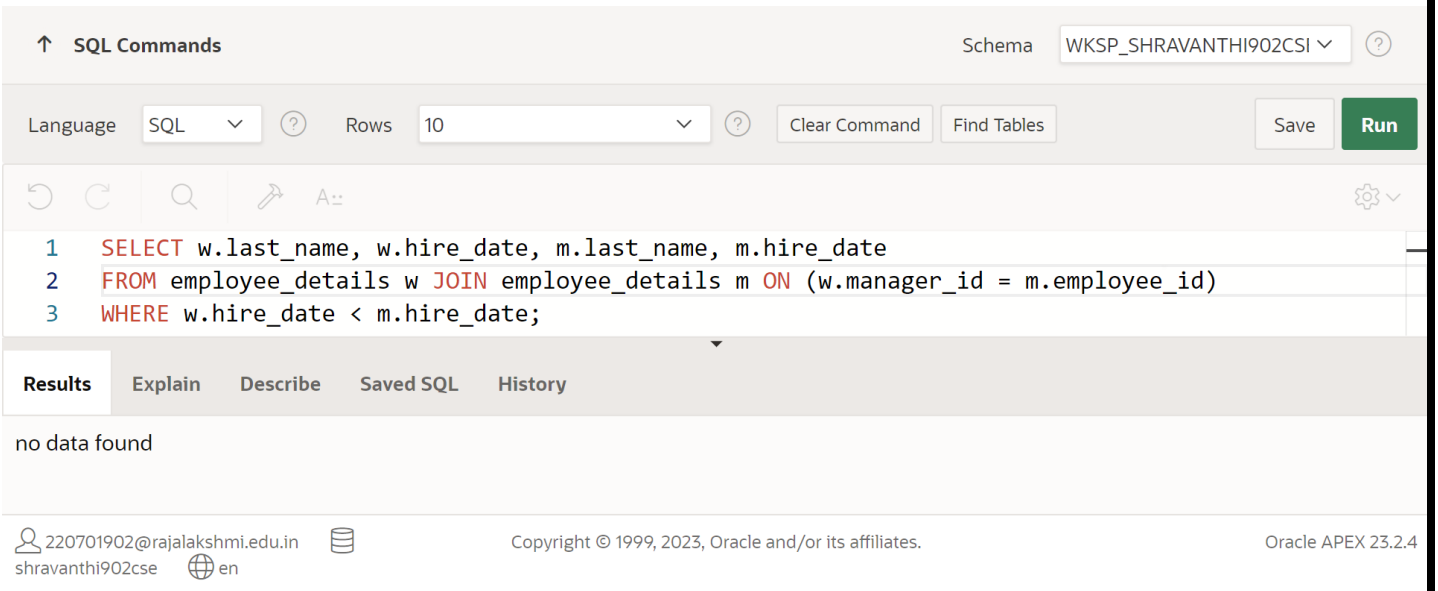
LAST_NAME	HIRE_DATE
JONES	05/06/1998
parker	02/22/1998
Davies	05/09/2021
cooper	02/02/2020
miller	02/02/2021
day	05/05/2020

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

QUERY:

```
SELECT w.last_name, w.hire_date, m.last_name, m.hire_date
FROM employee_details w JOIN employee_details m ON (w.manager_id =
m.employee_id)
WHERE w.hire_date < m.hire_date;
```

OUTPUT:



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

RESULT:

AGGREGATING DATA USING GROUP FUNCTIONS

EX_NO:8 **DATE:**

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

TRUE

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

FALSE

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

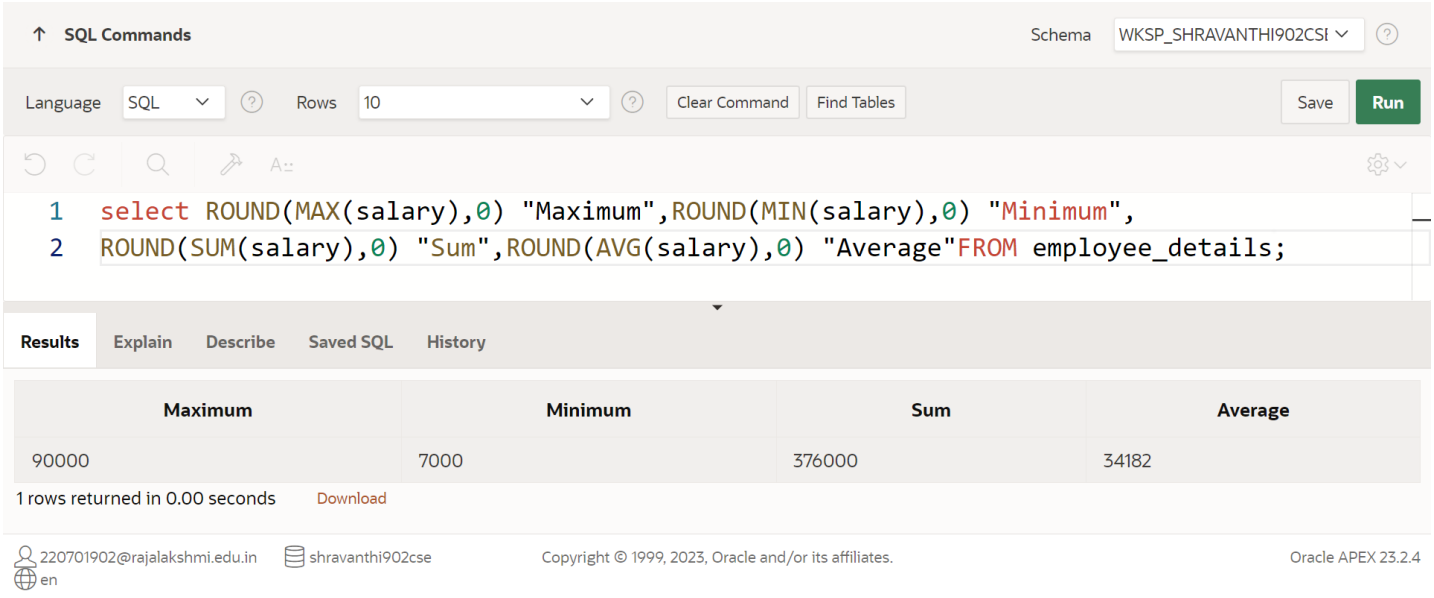
FALSE

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

QUERY:

```
select ROUND(MAX(salary),0) "Maximum",ROUND(MIN(salary),0) "Minimum",
ROUND(SUM(salary),0) "Sum",ROUND(AVG(salary),0) "Average"FROM
employee_details;
```

OUTPUT:

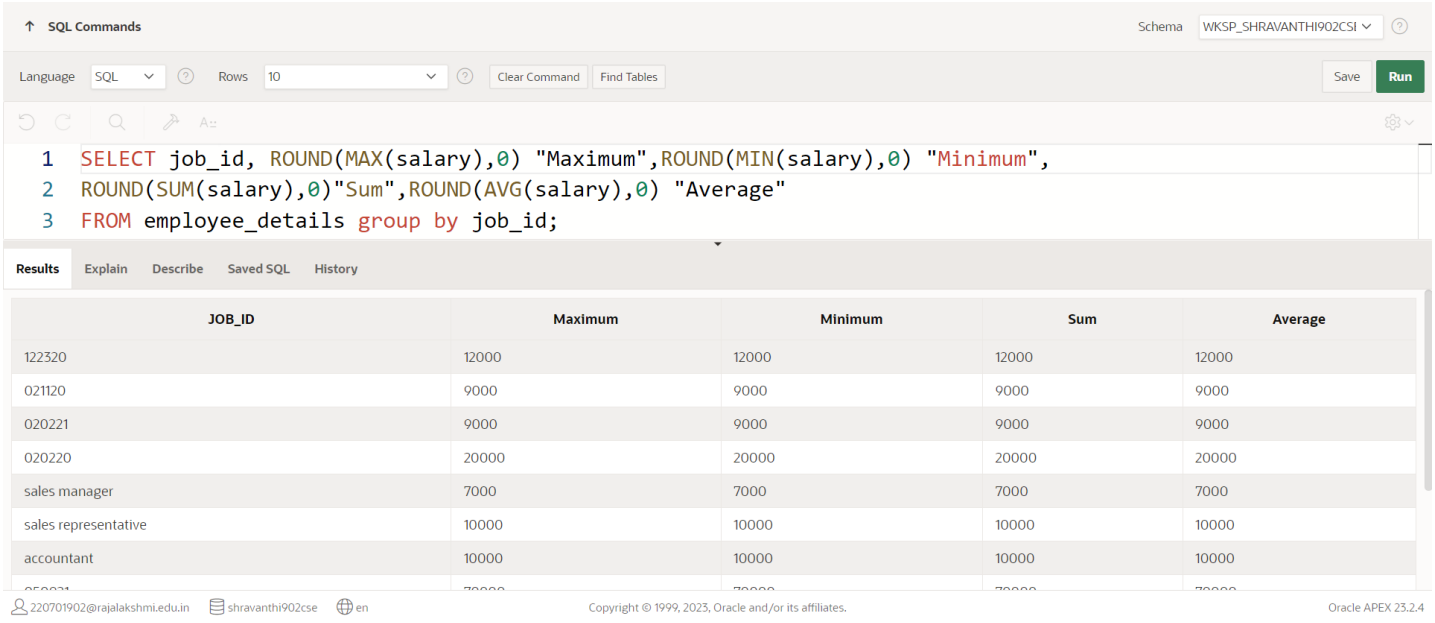


5. Modify the above query to display the minimum, maximum, sum, and average salary for each job type.

QUERY:

```
SELECT job_id, ROUND(MAX(salary),0) "Maximum",ROUND(MIN(salary),0)
"Minimum",
ROUND(SUM(salary),0)"Sum",ROUND(AVG(salary),0) "Average"
FROM employee_details group by job_id;
```

OUTPUT:

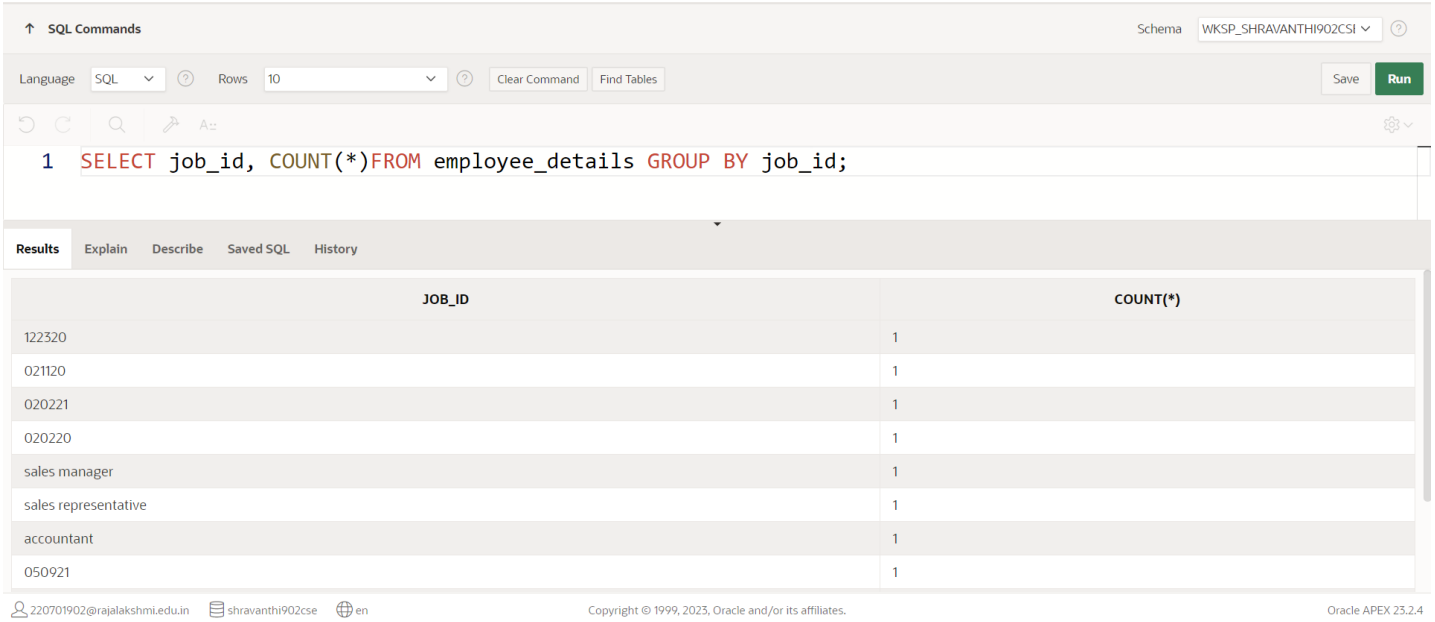


6. Write a query to display the number of people with the same job. Generalize the query so that the user in the HR department is prompted for a job title.

QUERY:

```
SELECT job_id, COUNT(*)FROM employee_details GROUP BY job_id;
```

OUTPUT:

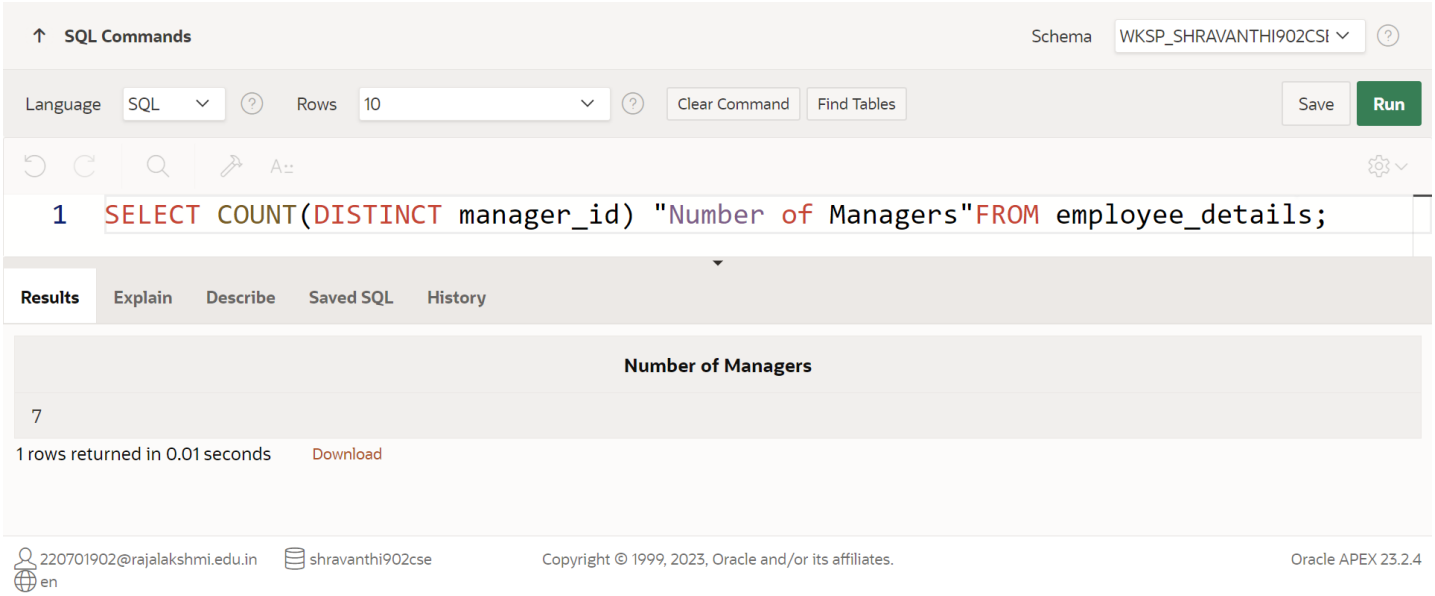


7. Determine the number of managers without listing them. Label the column Number of Managers. Hint: Use the MANAGER_ID column to determine the number of managers.

QUERY:

```
SELECT COUNT(DISTINCT manager_id) "Number of Managers"FROM employee_details;
```

OUTPUT:

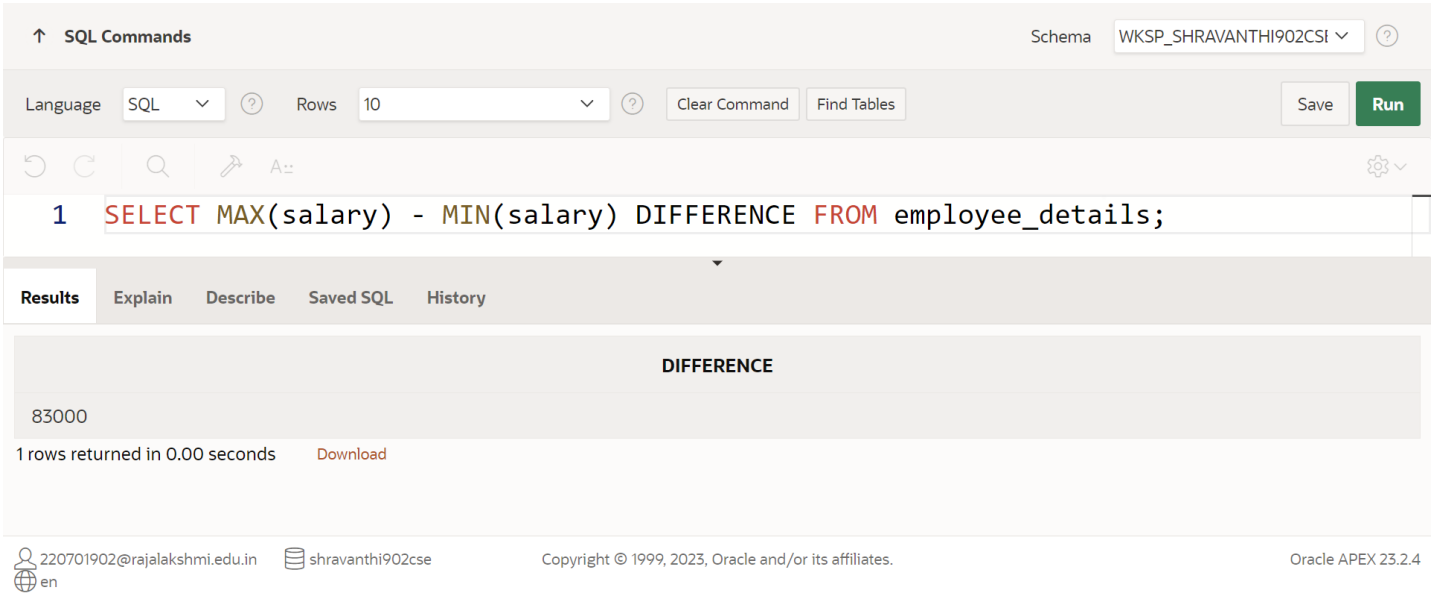


8. Find the difference between the highest and lowest salaries. Label the column DIFFERENCE

QUERY:

```
SELECT MAX(salary) - MIN(salary) DIFFERENCE FROM employee_details;
```

OUTPUT:

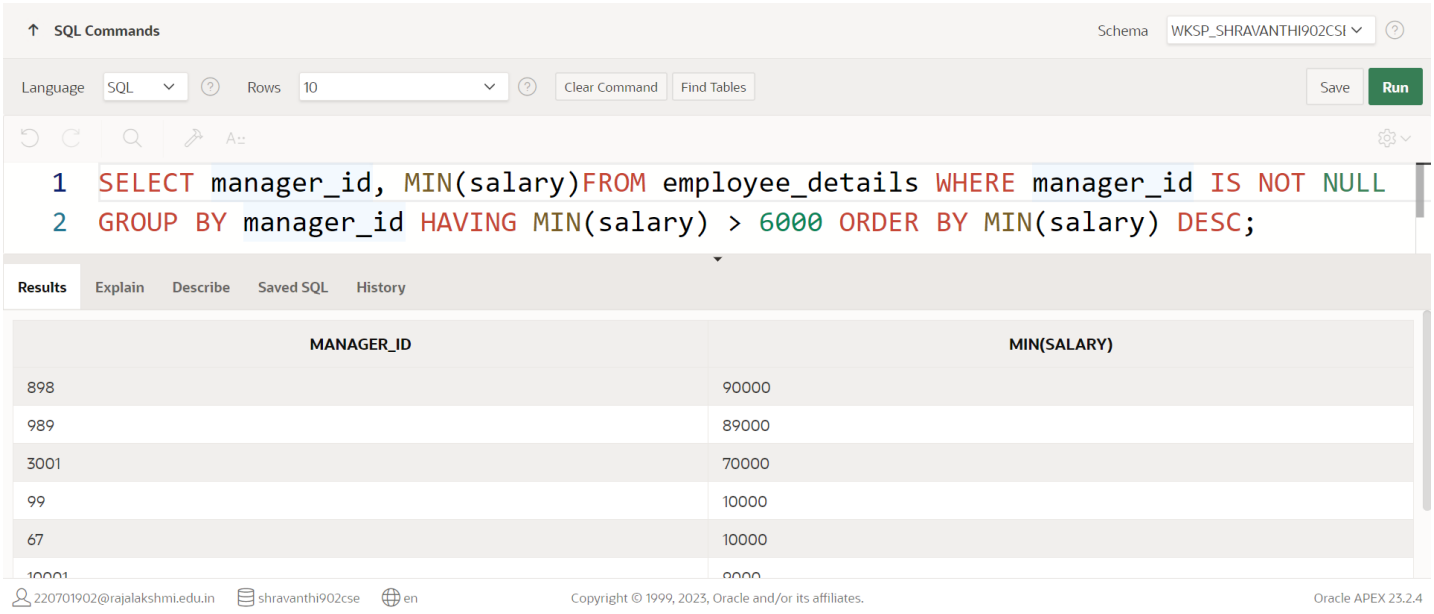


9. Create a report to display the manager number and the salary of the lowest-paid employee for that manager. Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is $6,000 or less. Sort the output in descending order of salary.

QUERY:

```
SELECT manager_id, MIN(salary)FROM employee_details WHERE manager_id IS
NOT NULL
GROUP BY manager_id HAVING MIN(salary) > 6000 ORDER BY MIN(salary) DESC;
```

OUTPUT:

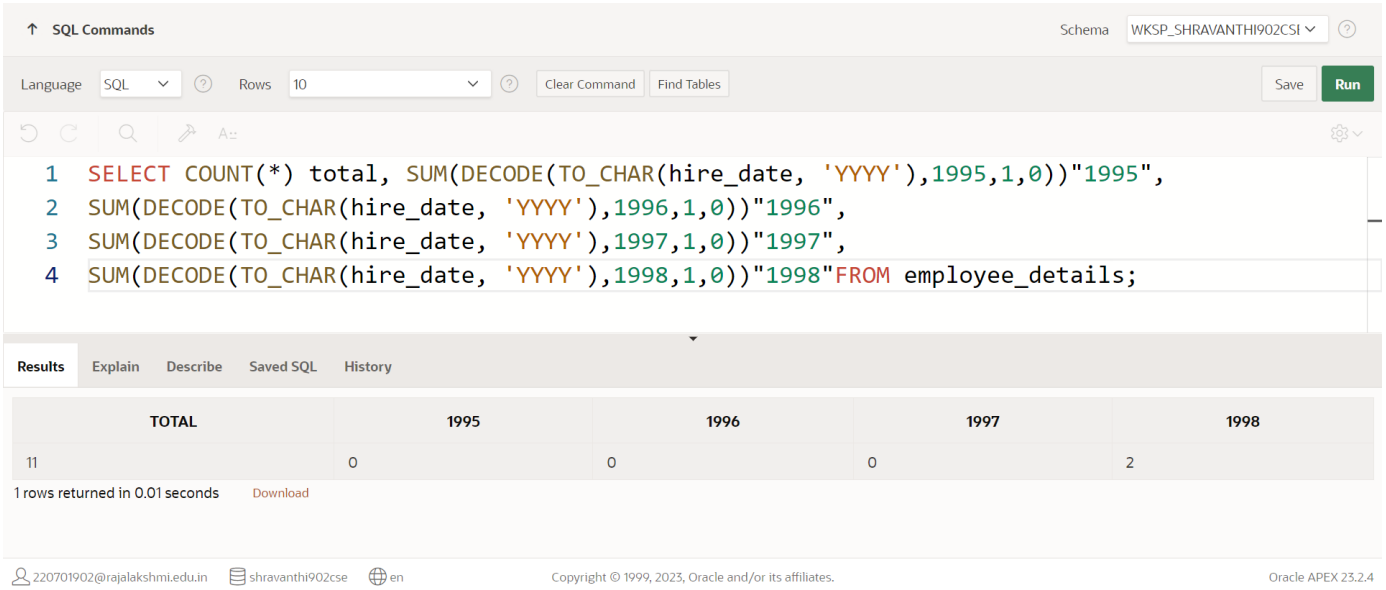


10. Create a query to display the total number of employees and, of that total, the number of employees hired in 1995, 1996, 1997, and 1998. Create appropriate column headings

QUERY:

```
SELECT COUNT(*) total, SUM(DECODE(TO_CHAR(hire_date,
'YYYY'),1995,1,0))"1995", SUM(DECODE(TO_CHAR(hire_date,
'YYYY'),1996,1,0))"1996", SUM(DECODE(TO_CHAR(hire_date,
'YYYY'),1997,1,0))"1997",SUM(DECODE(TO_CHAR(hire_date,
'YYYY'),1998,1,0))"1998"FROM employee_details;
```

OUTPUT:

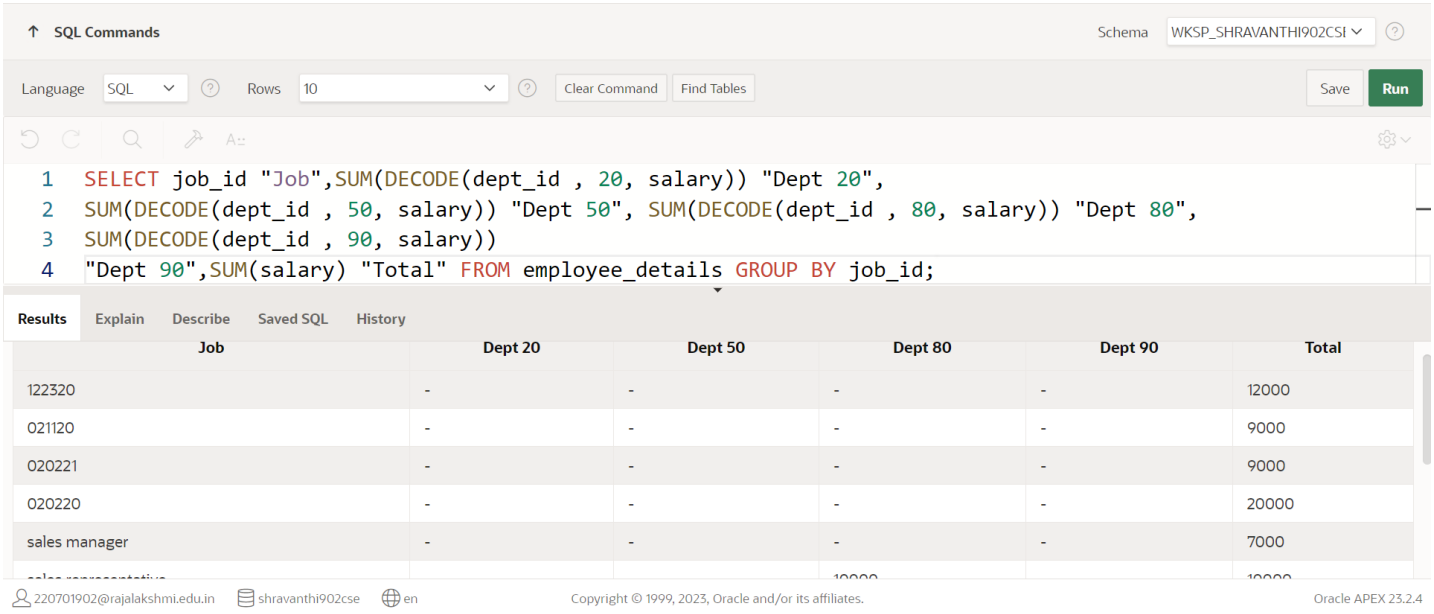


11. Create a matrix query to display the job, the salary for that job based on department number, and the total salary for that job, for departments 20, 50, 80, and 90, giving each column an appropriate heading

QUERY:

```
SELECT job_id "Job",SUM(DECODE(dept_id , 20, salary)) "Dept
20",SUM(DECODE(dept_id , 50, salary)) "Dept 50", SUM(DECODE(dept_id , 80,
salary)) "Dept 80",SUM(DECODE(dept_id , 90, salary))"Dept 90",SUM(salary)
"Total" FROM employee_details GROUP BY job_id;
```

OUTPUT:

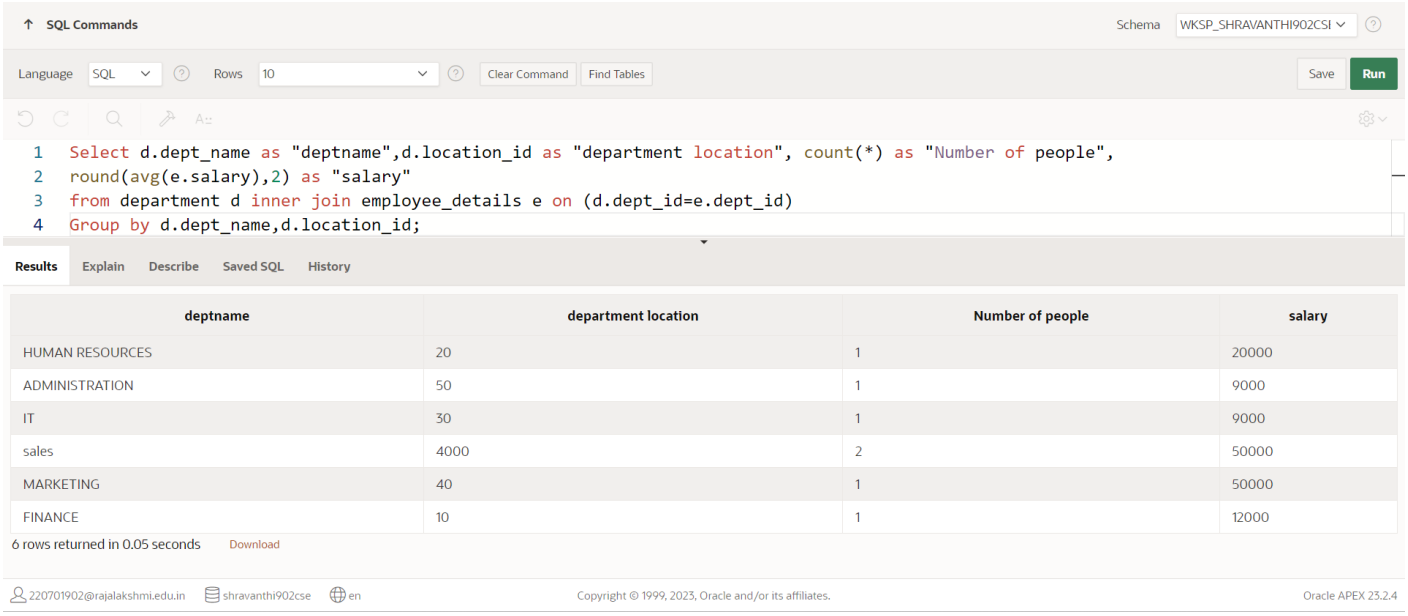


12. Write a query to display each department‘s name, location, number of employees, and the average salary for all the employees in that department. Label the column name-Location, Number of people, and salary respectively. Round the average salary to two decimal places.

QUERY:

```
Select d.dept_name as "deptname",d.location_id as "department location",
count(*) as "Number of people",
round(avg(e.salary),2) as "salary"
from department d inner join employee_details e on (d.dept_id=e.dept_id)
Group by d.dept_name,d.location_id;
```

OUTPUT:



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

RESULT:

SUB-QUERIES

EX.NO:9 **DATE:**

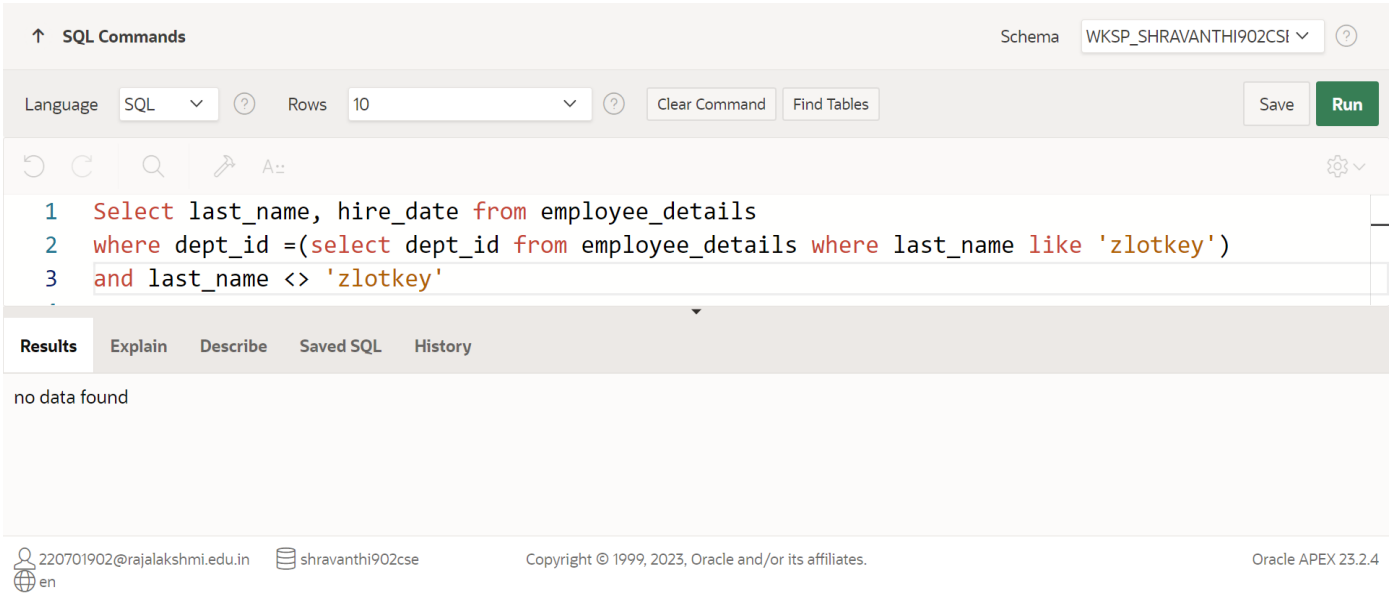
Find the Solution for the following:

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

QUERY:

```
Select last_name, hire_date from employee_details
where dept_id =(select dept_id from employee_details where last_name
like 'zlotkey')
and last_name <> 'zlotkey';
```

OUTPUT:

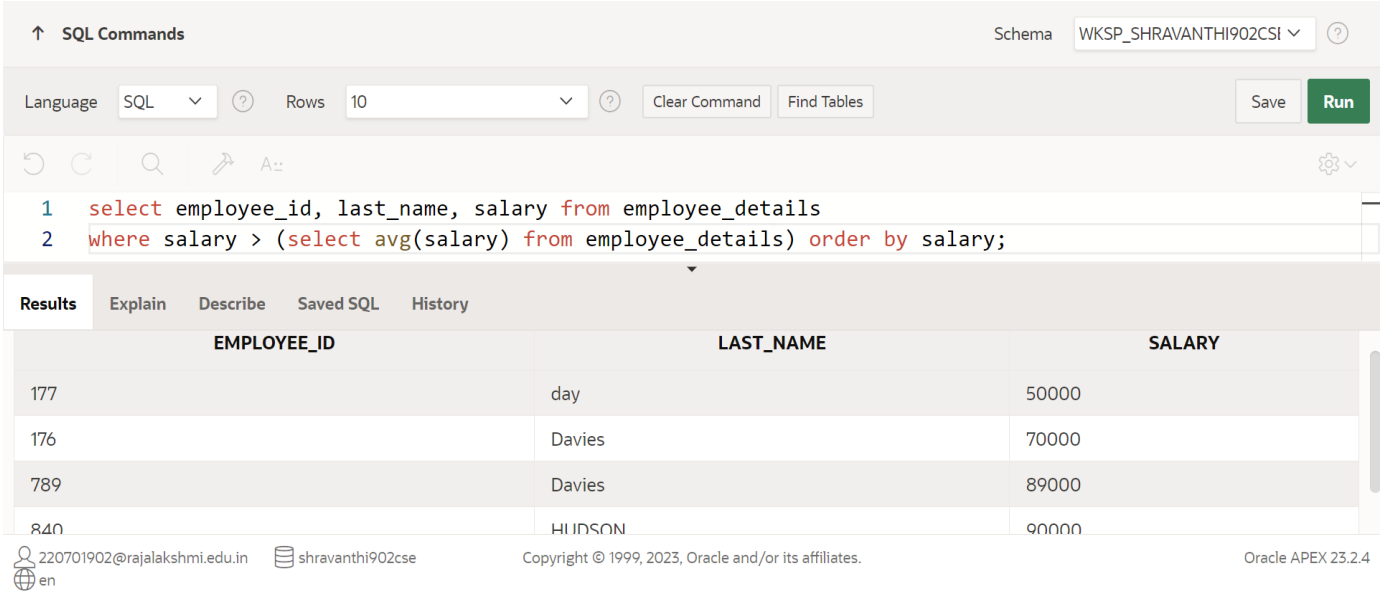


2. Create a report that displays the employee number, last name, and salary of all employeeswho earn more than the average salary. Sort the results in order of ascending salary.

QUERY:

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

OUTPUT:

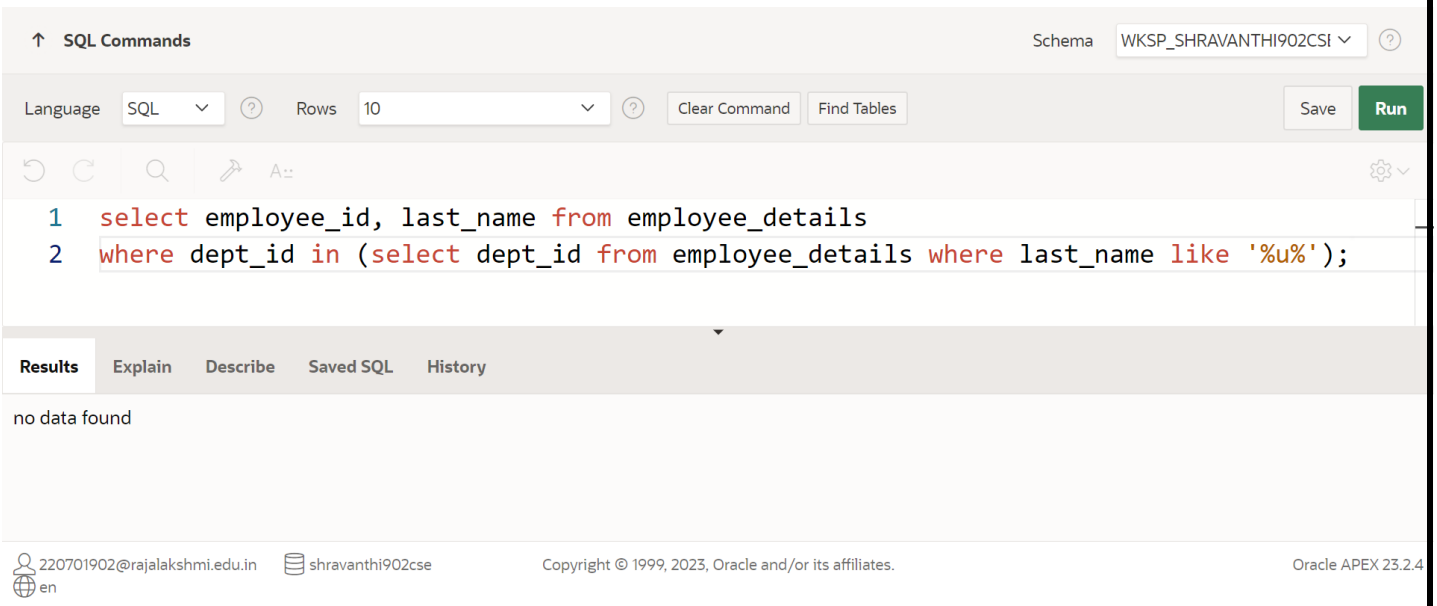


3. Write a query that displays the employee number and last name of all employees whowork in a department with any employee whose last name contains a u.

QUERY:

```
select employee_id, last_name from employee_details
where dept_id in (select dept_id from employee_details where
last_name like '%u%');
```

OUTPUT:

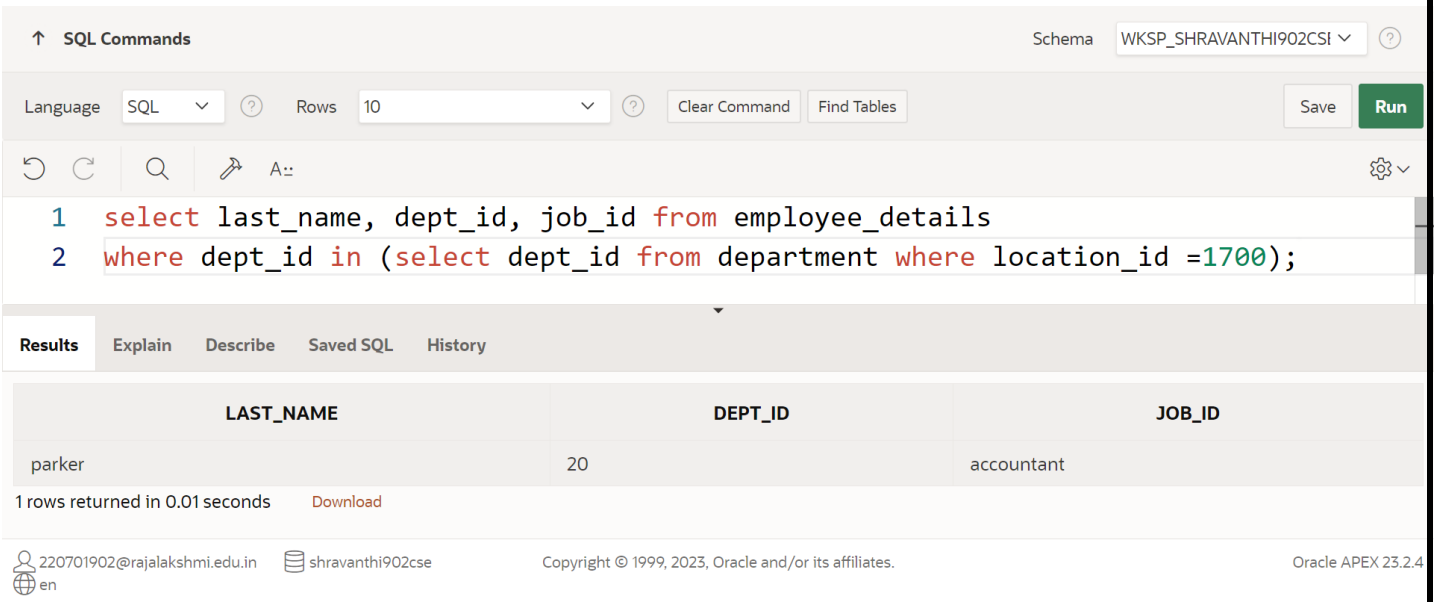


4. The HR department needs a report that displays the last name, department number, and jobID of all employees whose department location ID is 1700.

QUERY:

```
select last_name, dept_id, job_id from
employee_details
where dept_id in (select dept_id from department where
location_id =1700);
```

OUTPUT:



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

QUERY:

```
 select last_name, salary from employee_details
where manager_id in (select employee_id from employee_details
where last_name='King');
```

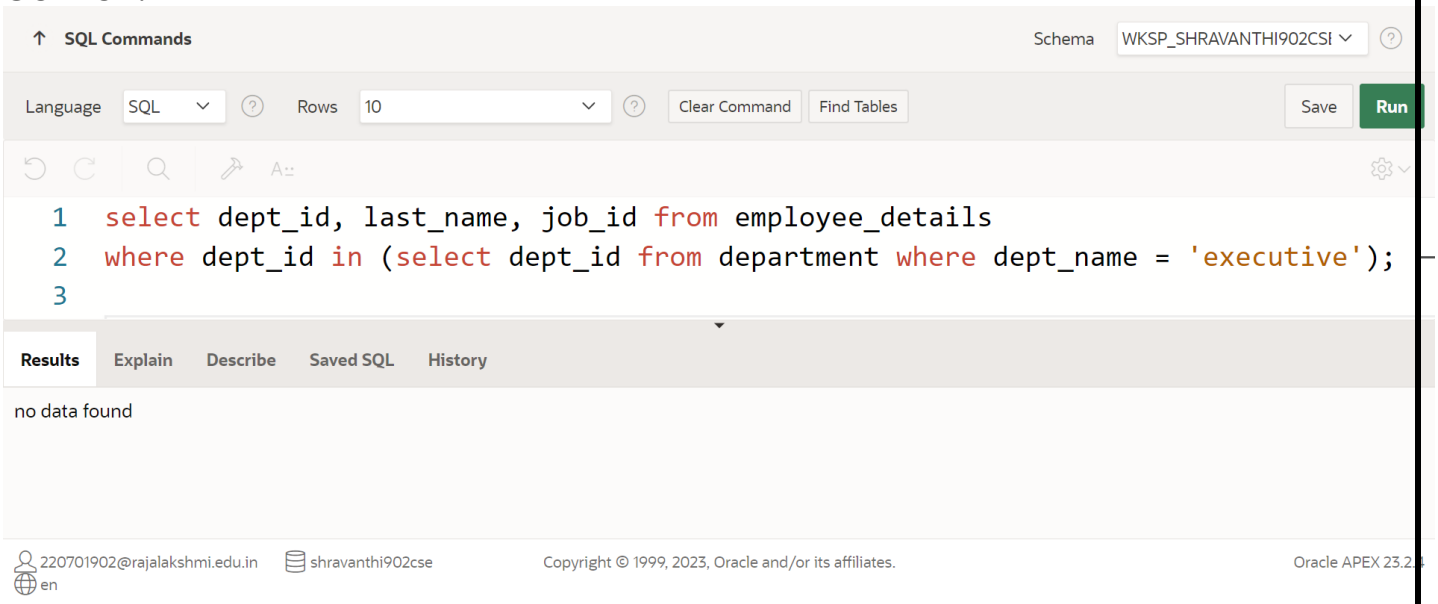
OUTPUT:

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

QUERY:

```
select dept_id, last_name, job_id from employee_details where dept_id in
(select dept_id from department where dept_name = 'executive');
```

OUTPUT:

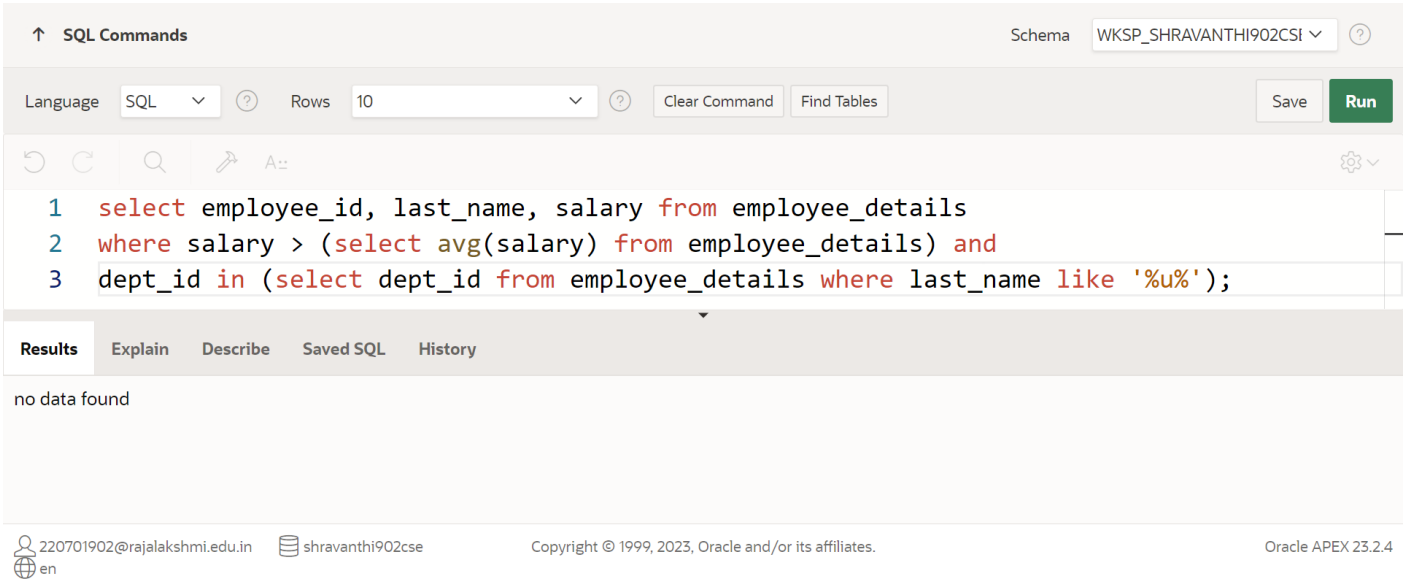


7. Modify the query 3 to display the employee number, last name, and salary of all employees who earn more than the average salary and who work in a department with anyemployee whose last name contains a u.

QUERY:

```
select employee_id, last_name, salary from employee_details
where salary > (select avg(salary) from employee_details) and
dept_id in (select dept_id from employee_details where last_name like
'%u%');
```

OUTPUT:



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

RESULT:

USING THE SET OPERATORS

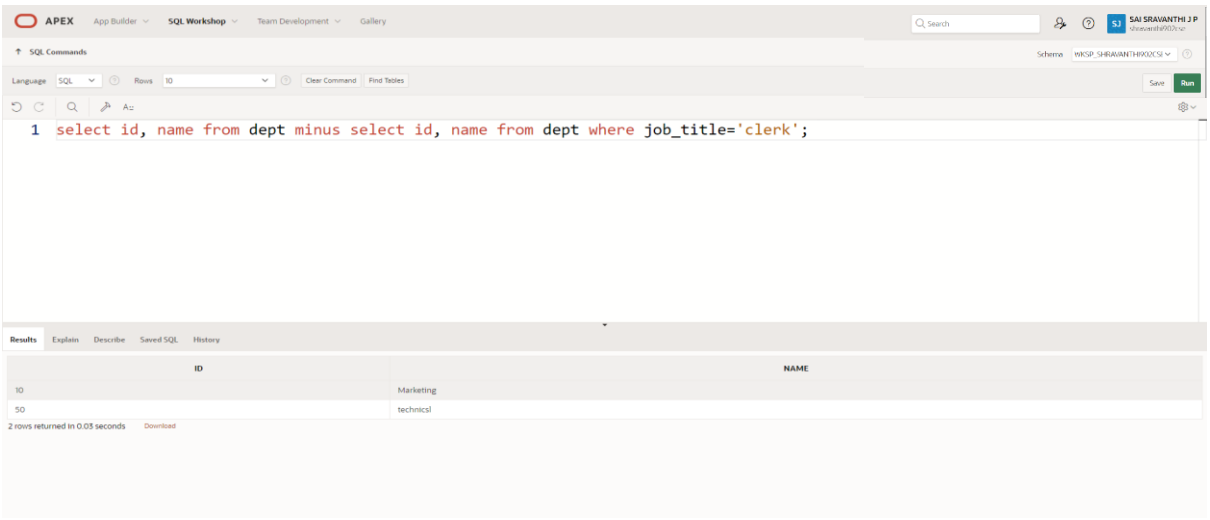
EXNO: 10 **DATE:**

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

QUERY:

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

OUTPUT:

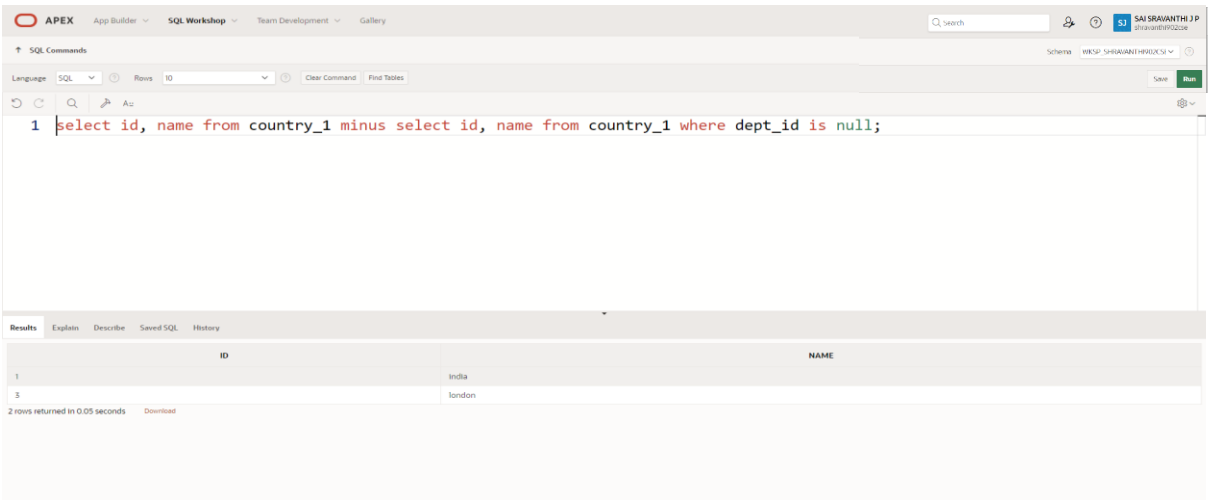


2.The HR department needs a list of countries that have no departments located in them. Display the country ID and the name of the countries. Use set operators to create this report.

QUERY:

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

OUTPUT:

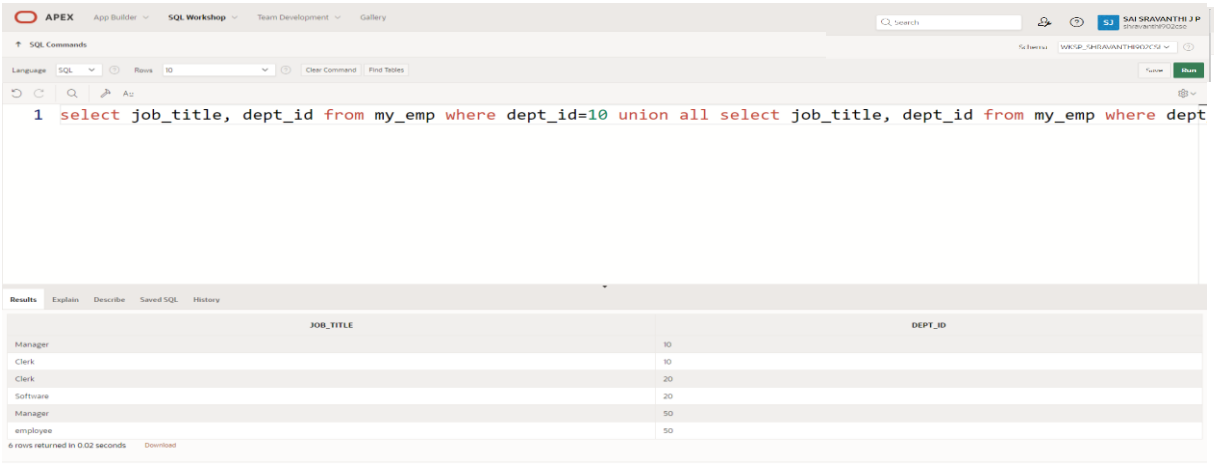


3. Produce a list of jobs for departments 10, 50, and 20, in that order. Display job ID and department ID using set operators.

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:

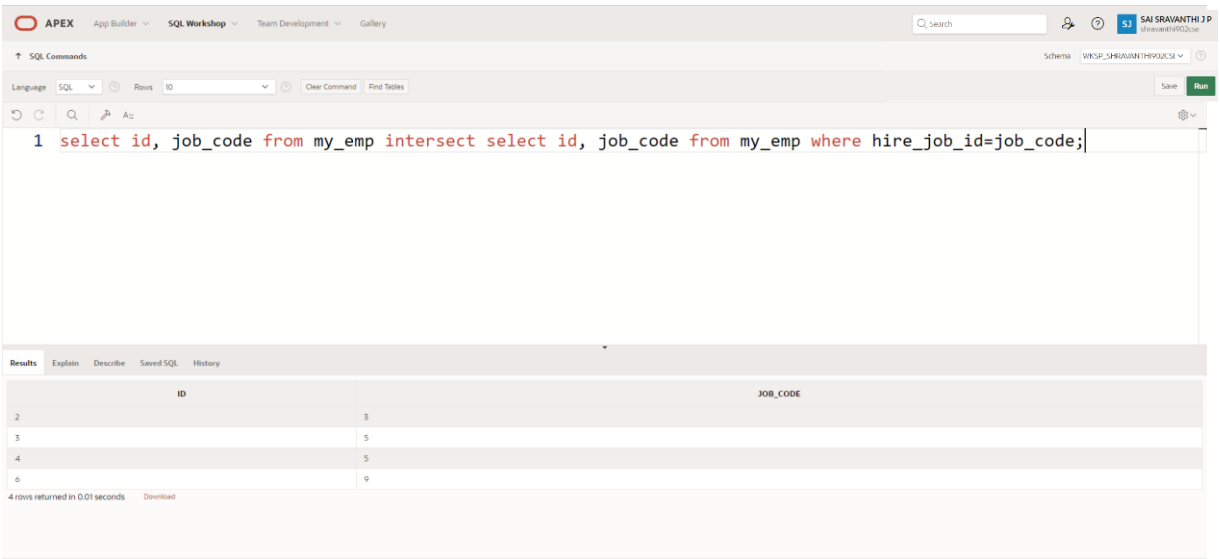


4. Create a report that lists the employee IDs and job IDs of those employees who currently have a job title that is the same as their job title when they were initially hired by the company (that is, they changed jobs but have now gone back to doing their original job).

QUERY:

```
Select id,job_code from my_emp intersect select id,job_code from
my_emp where hire_job_id=job_code;
```

OUTPUT:

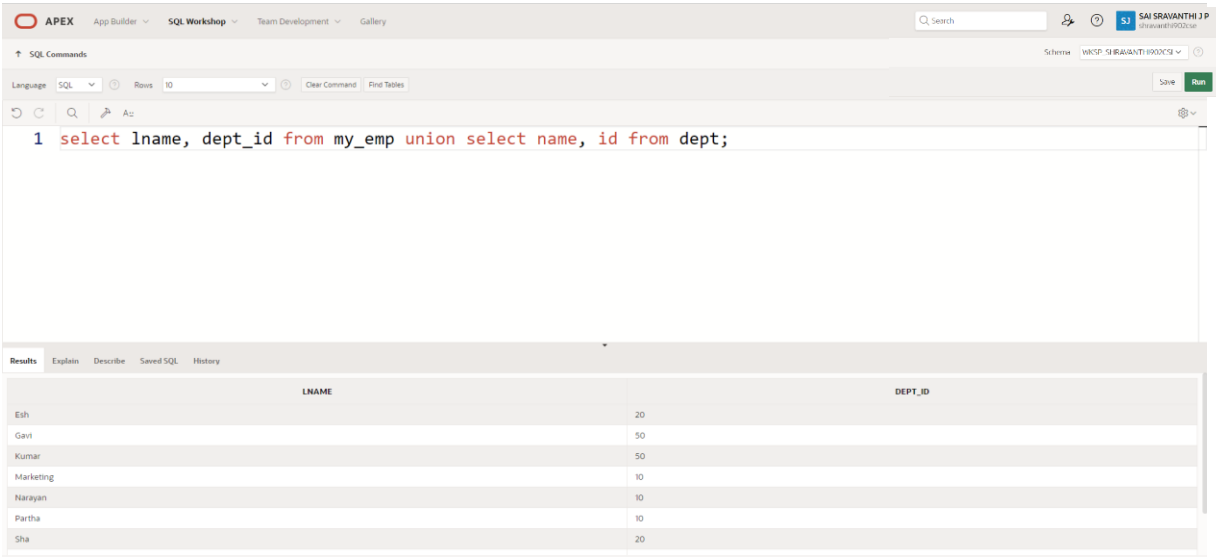


5. The HR department needs a report with the following specifications:
- Last name and department ID of all the employees from the EMPLOYEES table, regardless of whether or not they belong to a department.
- Department ID and department name of all the departments from the DEPARTMENTS table, regardless 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:

CREATING VIEWS

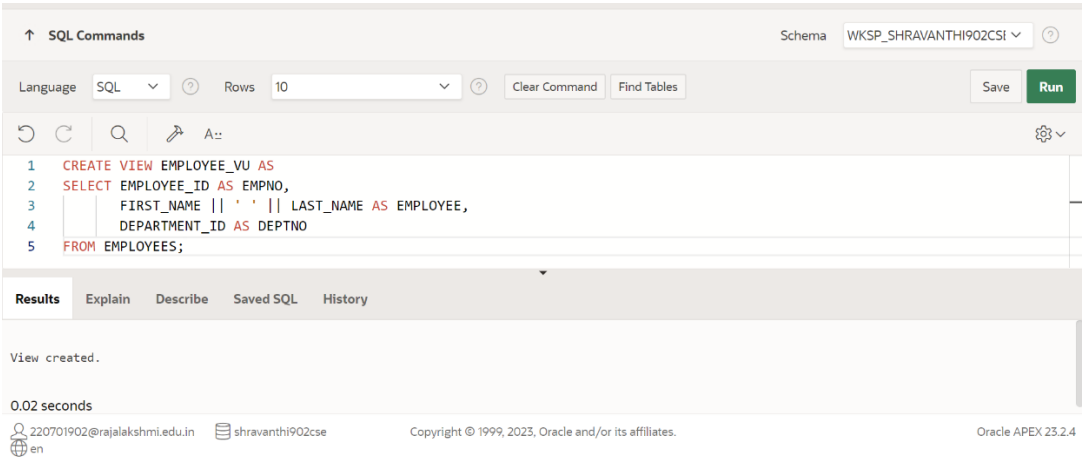
EXNO:11 **DATE:**

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

QUERY:

```
Create view employee_vu as
Select employee_id as empno, first_name || ' ' || last_name as
employee, department_id as deptno
From employees;
```

OUTPUT:

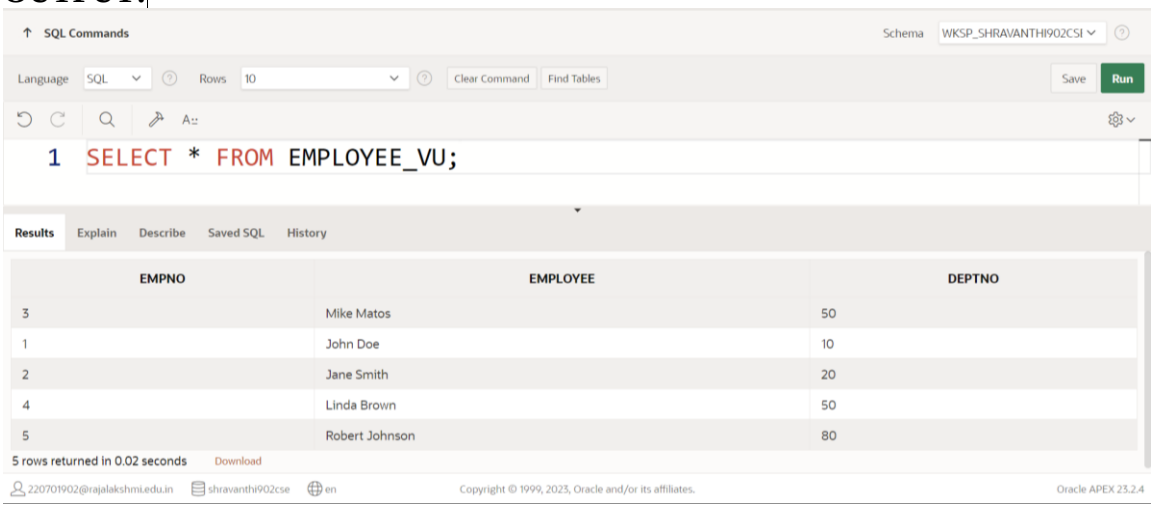


2. Display the contents of the EMPLOYEES_VU view.

QUERY:

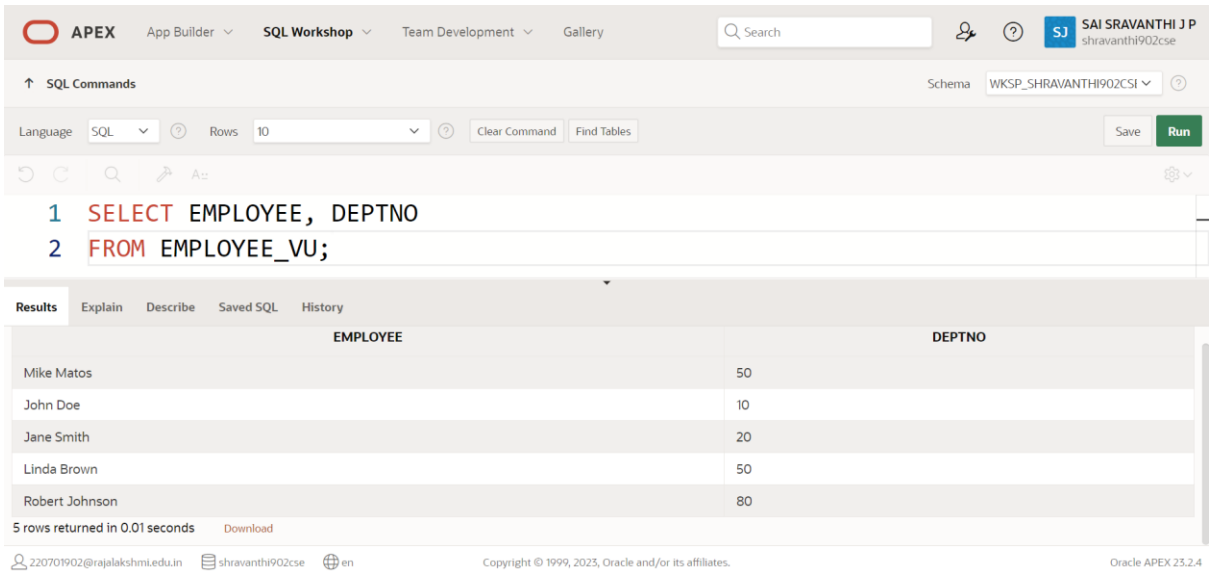
```
SELECT * FROM EMPLOYEE_VU;
```

OUTPUT:



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

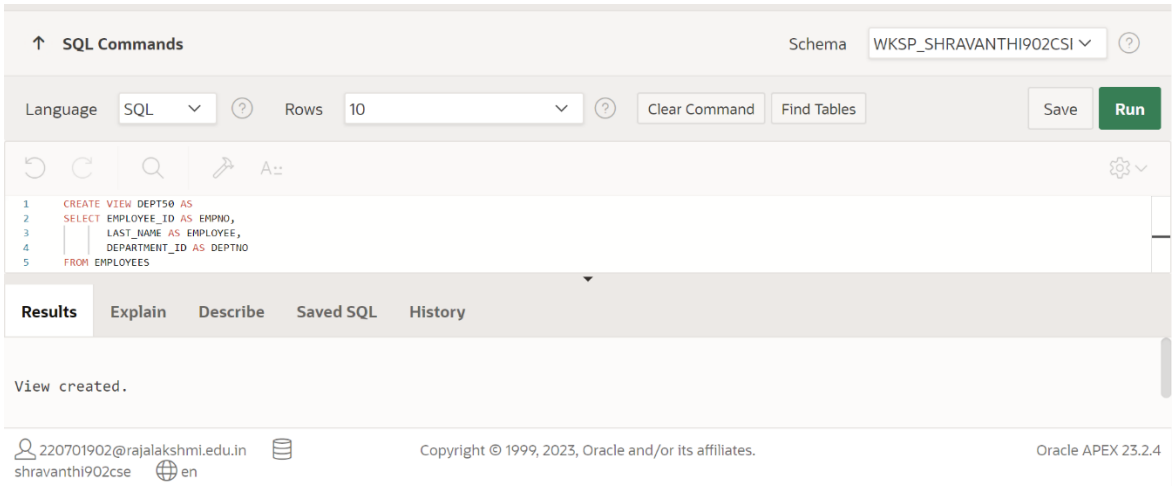
QUERY:
```
SELECT EMPLOYEE,DEPTNO
    FROM EMPLOYEE_VU;
```
OUTPUT:



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:

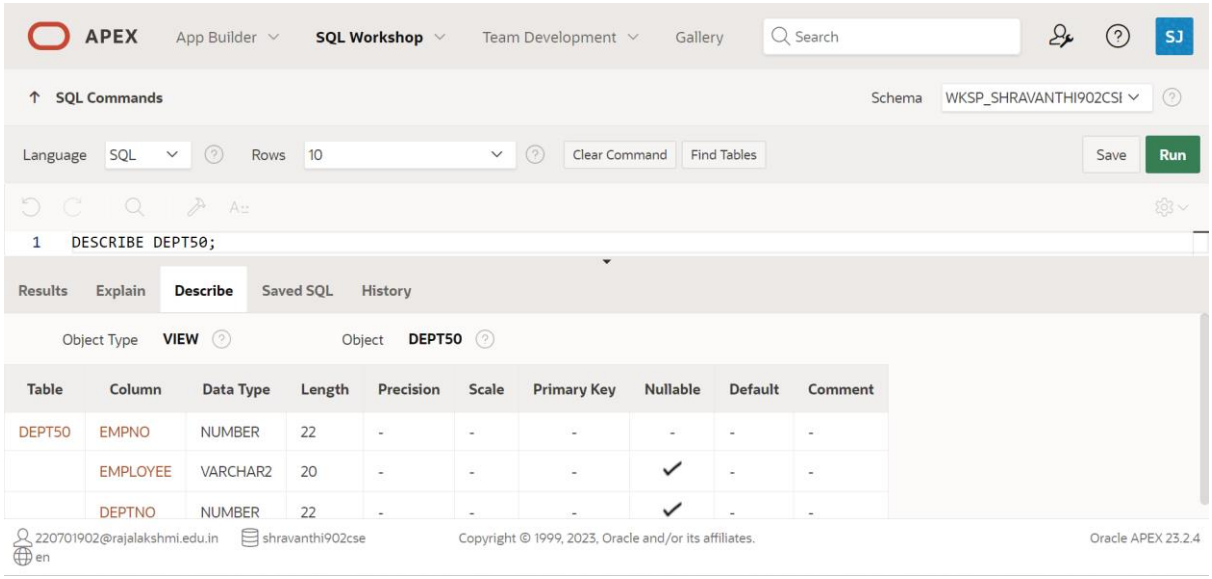


5. Display the structure and contents of the DEPT50 view.

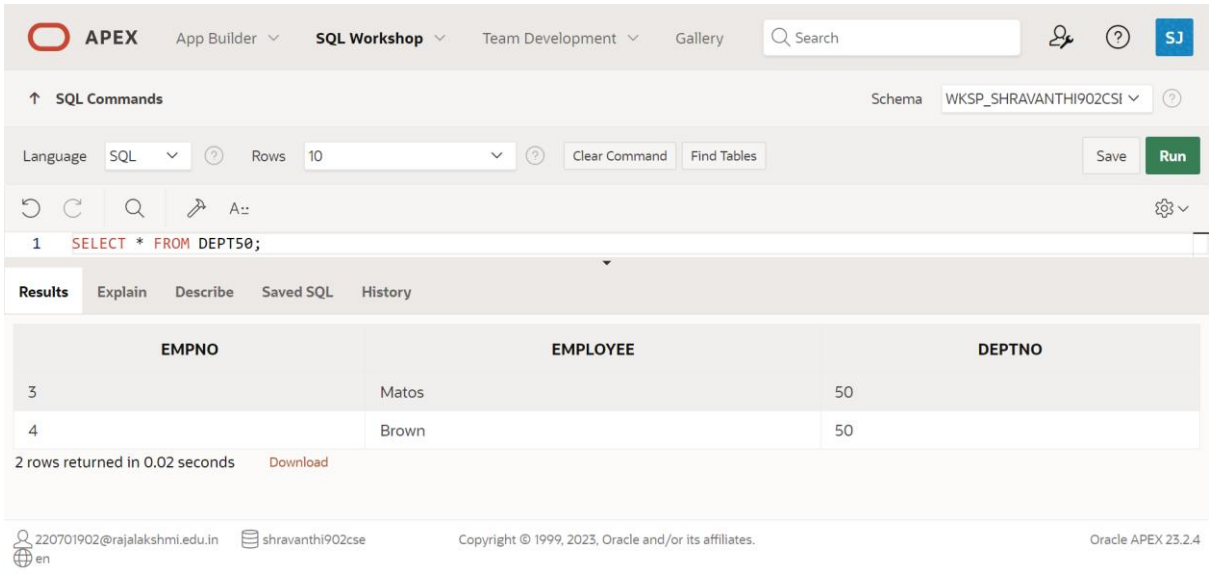
QUERY:

```
DESCRIBE DEPT50;
```

OUTPUT:



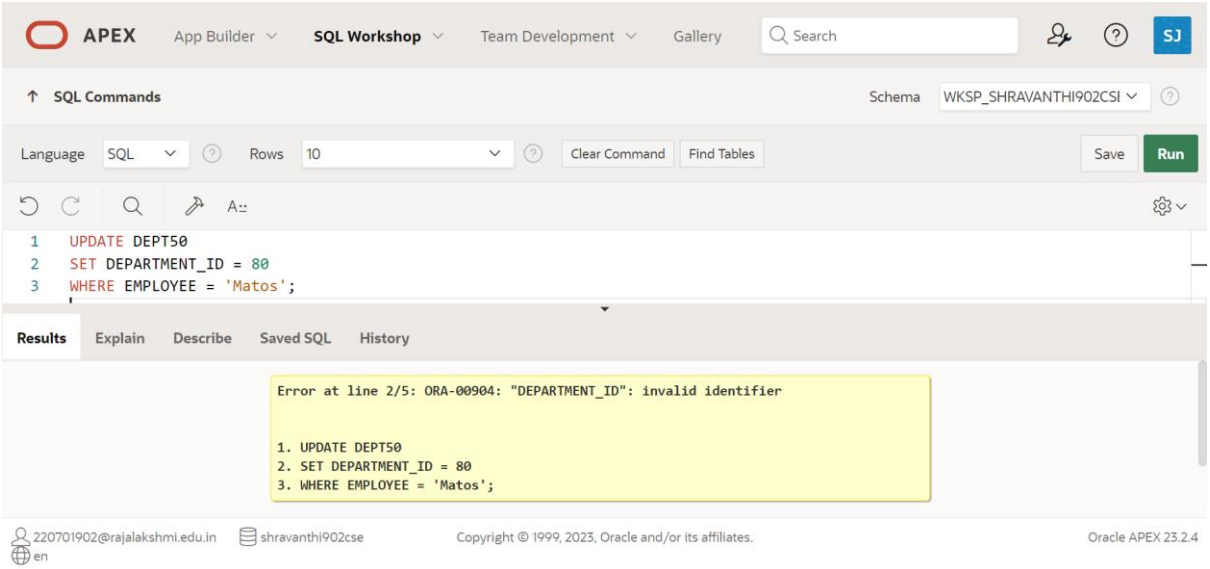

```
Select * from dept50;
```



6.Attempt to reassign Matos to department 80.

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

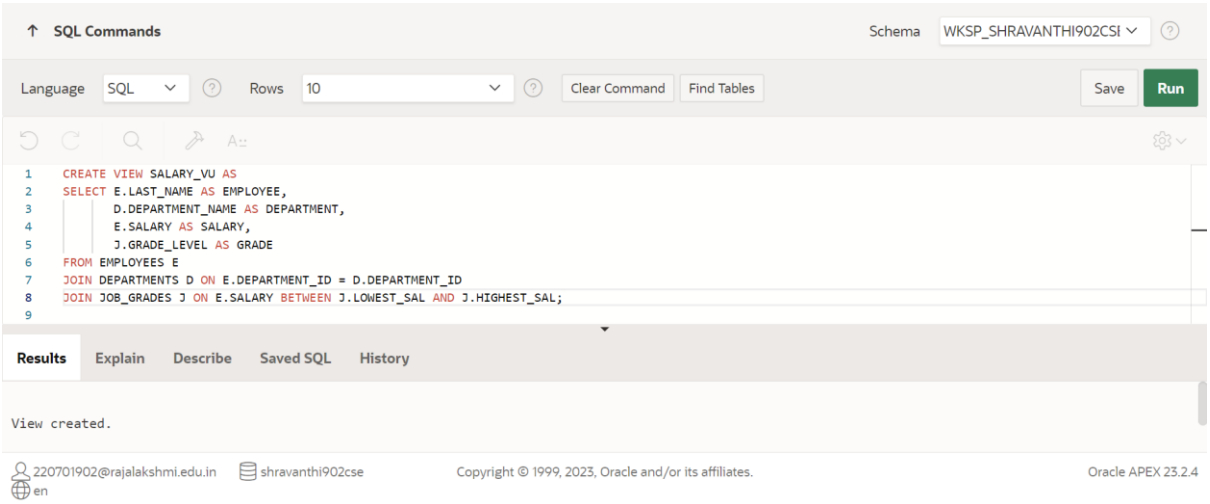
OUTPUT:



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:

PRIMARY KEY, FOREIGN KEY AND CHECK CONSTRAINTS

EXNO:12 **DATE:**

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

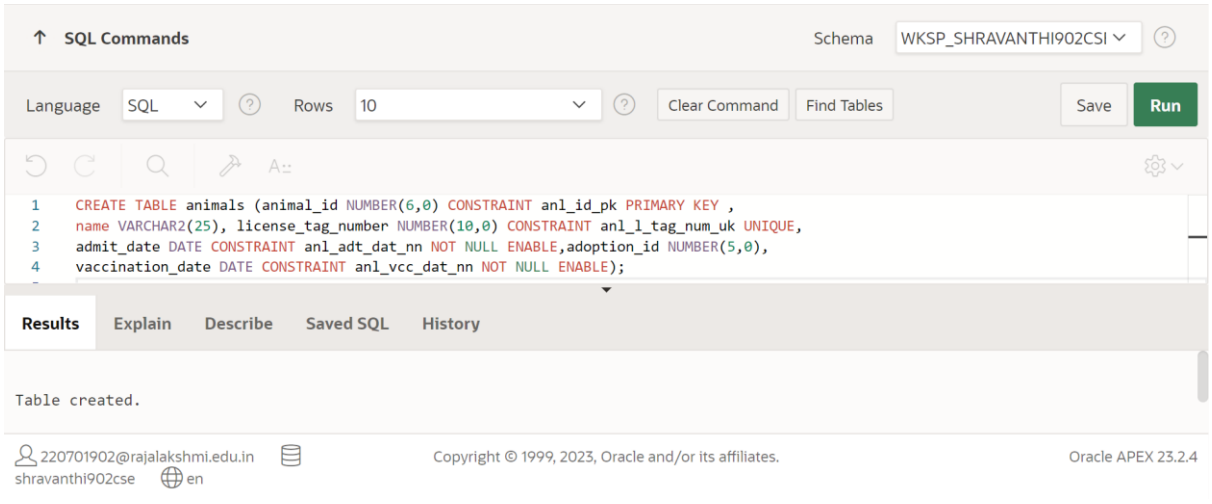
2. Using the column information for the animals table below, name constraints where applicable at the table level, otherwise name them at the column level. Define the primary key (animal_id). The license_tag_number must be unique. The admit_date and vaccination_date columns cannot contain null values.
animal_id NUMBER(6) - PRIMARY KEY
name VARCHAR2(25)
license_tag_number NUMBER(10)- UNIQUE
admit_date DATE- NOT NULL
adoption_id NUMBER(5),
vaccination_date DATE- NOT NULL

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

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:

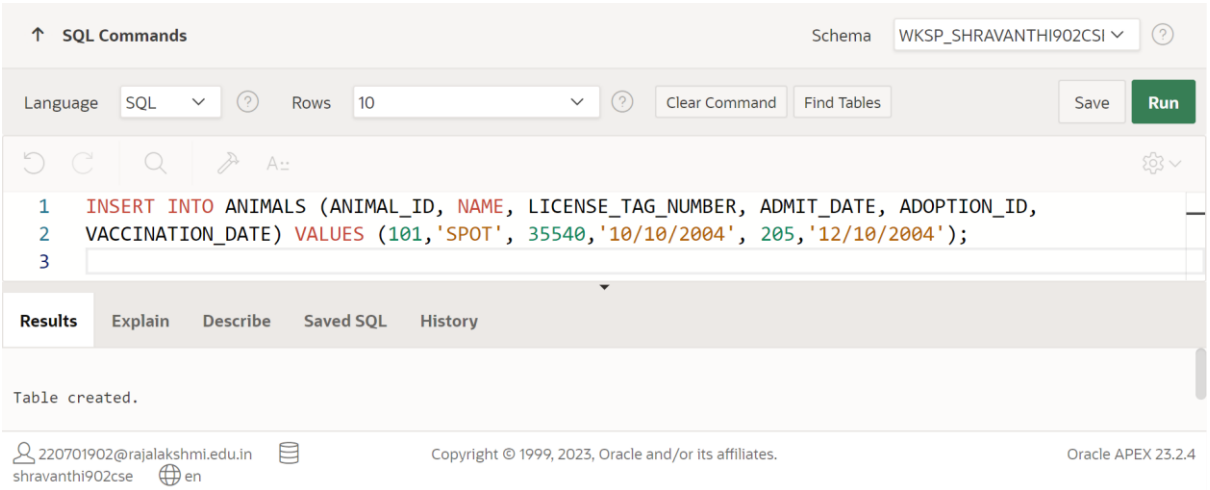


4. Enter one row into the table. Execute a SELECT * statement to verify your input.

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:



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

Ans: COLUMN LEVEL STATEMENT:

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

TABLE LEVEL STATEMENT:

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

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

a. ON DELETE CASCADE

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

b. ON DELETE SET NULL

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

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

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

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

RESULT:

CREATING VIEWS

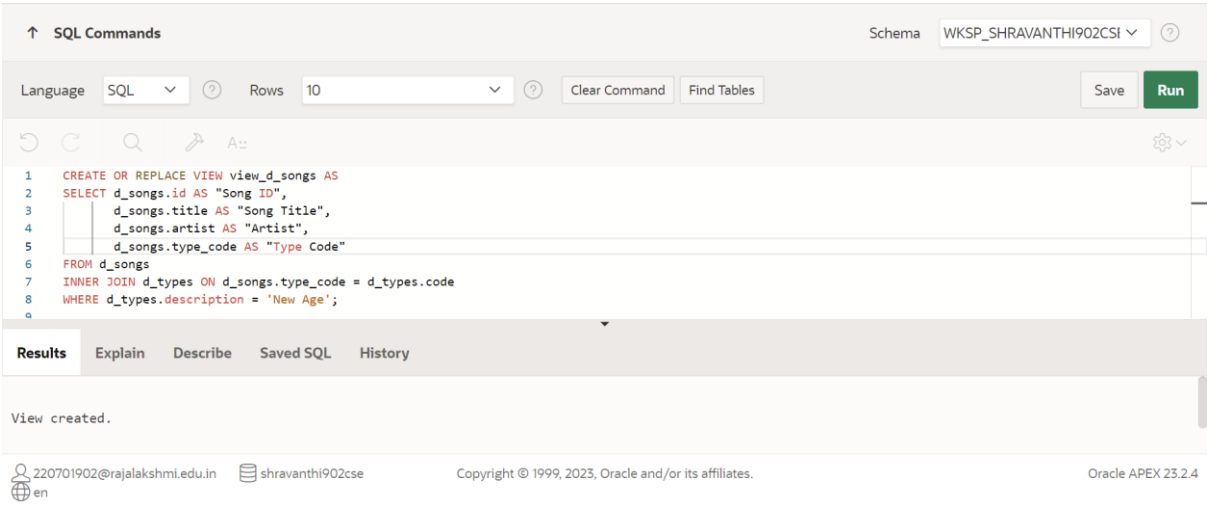
EX.NO.13 **DATE:**

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

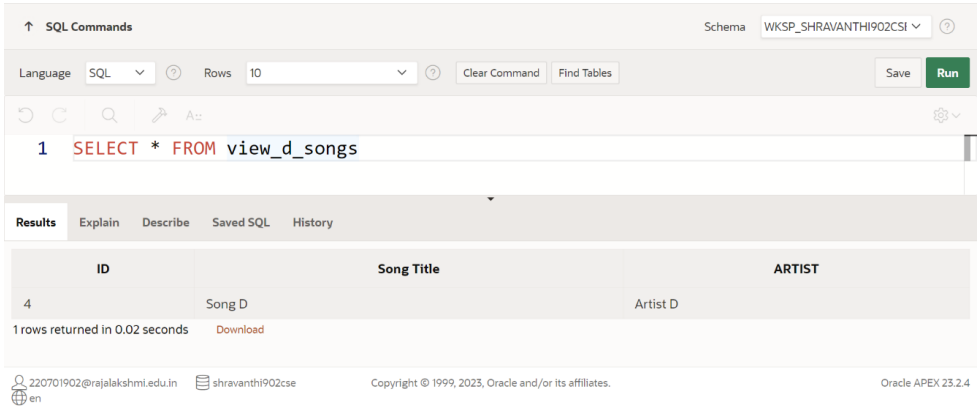
- **Restrict access and display selective columns**
- **Reduce complexity of queries from other internal systems. So, providing a way to view same data in a different manner.**
- **Let the app code rely on views and allow the internal implementation of tables to be modified later.**

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

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



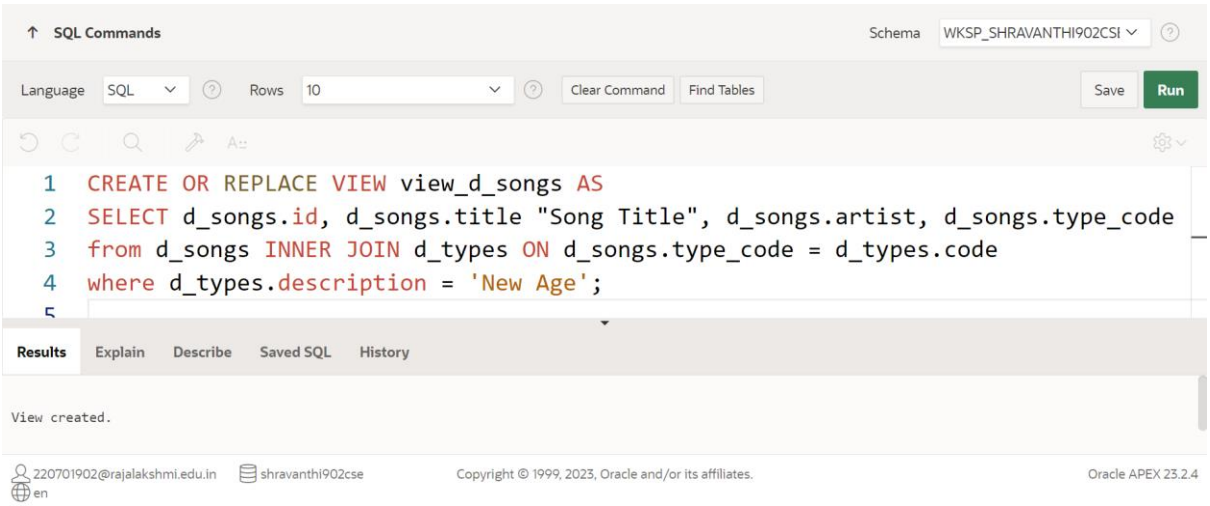
3. SELECT * FROM view_d_songs. What was returned?



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

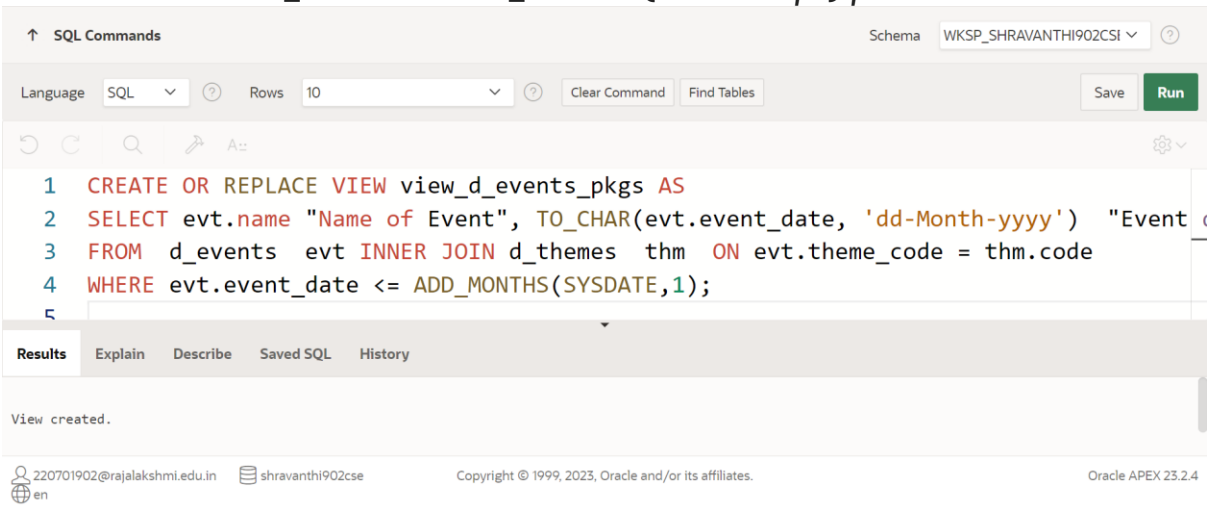
Or use alias after the CREATE statement as shown.

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



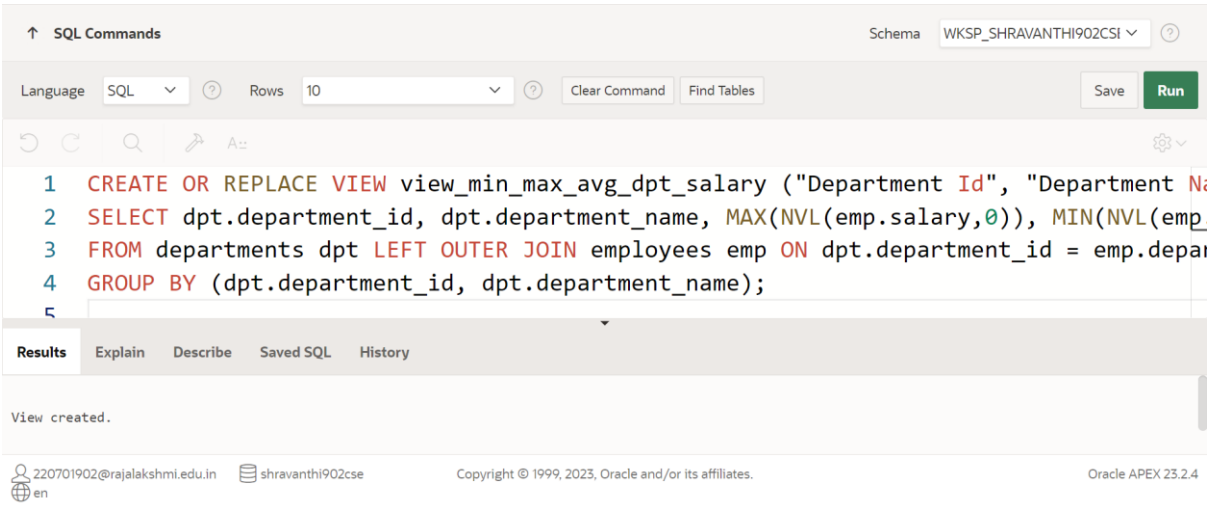
5. Jason Tsang, the disk jockey for DJs on Demand, needs a list of the past events and those planned for the coming months so he can make arrangements for each event's equipment setup. As the company manager, you do not want him to have access to the price that clients paid for their events. Create a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

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



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

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

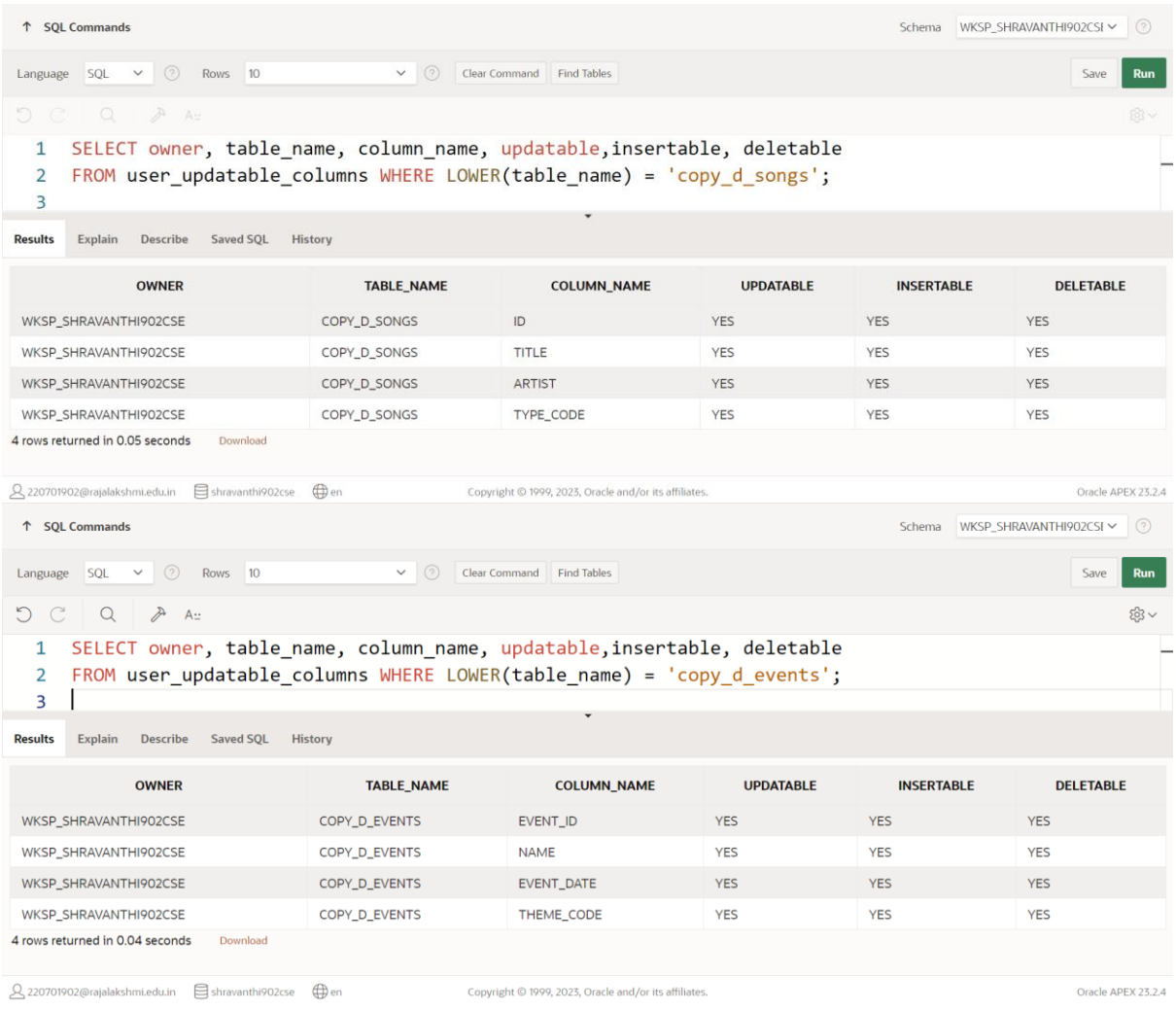


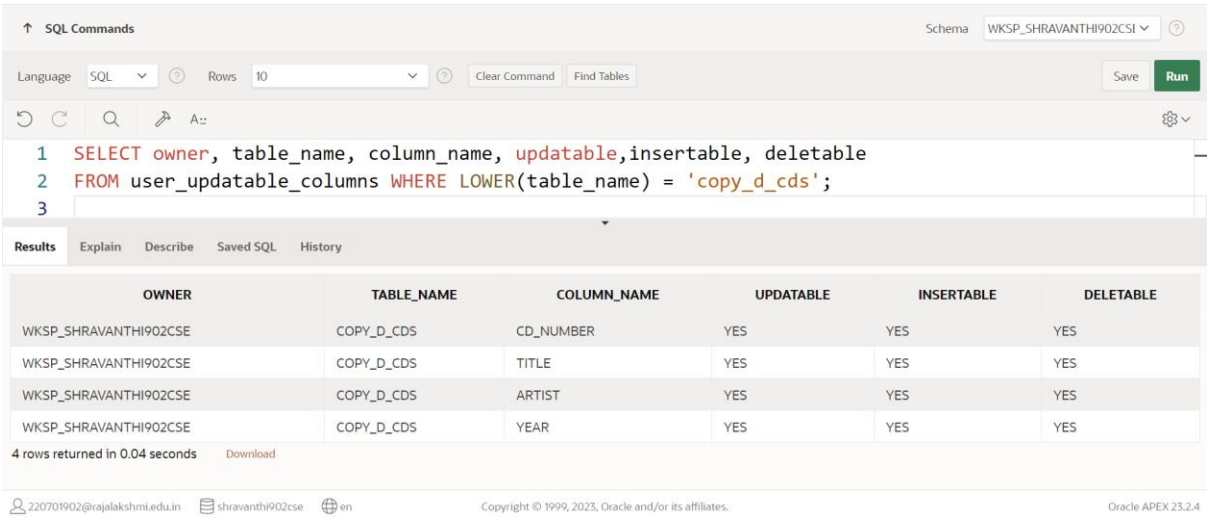
DML Operations and Views

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

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

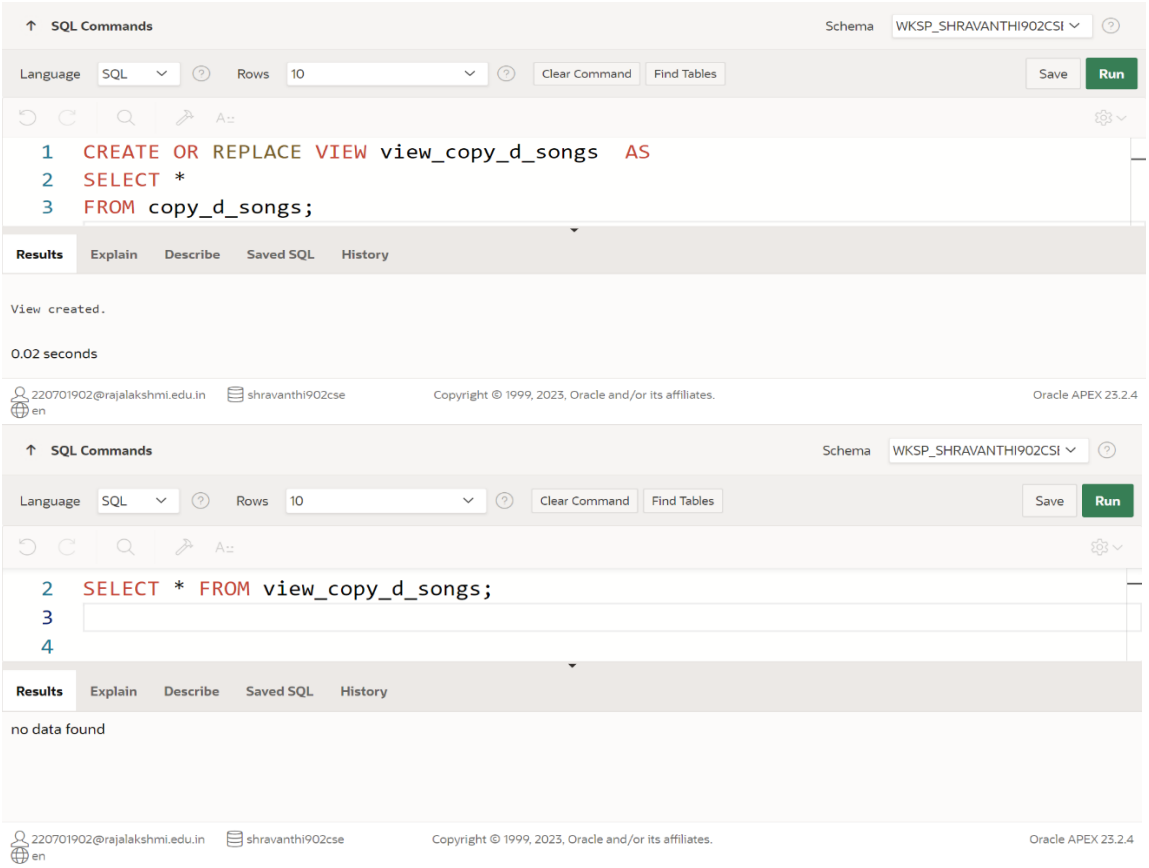
```
SELECT owner, table_name, column_name, updatable,insertable, deletable
FROM user_updatable_columns WHERE LOWER(table_name) = 'copy_d_songs';

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

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





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

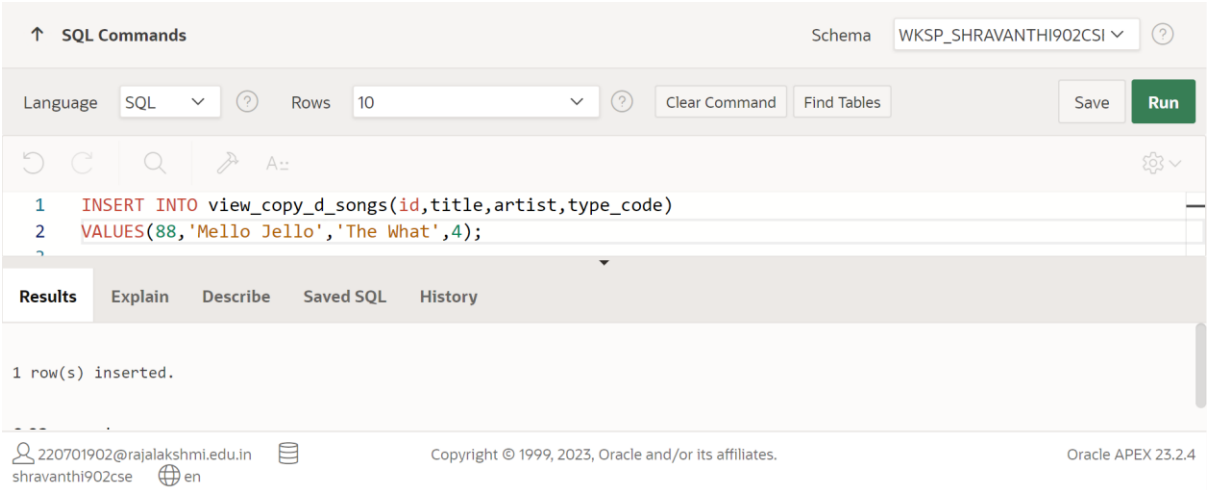
```
CREATE OR REPLACE VIEW view_copy_d_songs  AS
SELECT *
FROM copy_d_songs;

SELECT * FROM view_copy_d_songs;
```



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

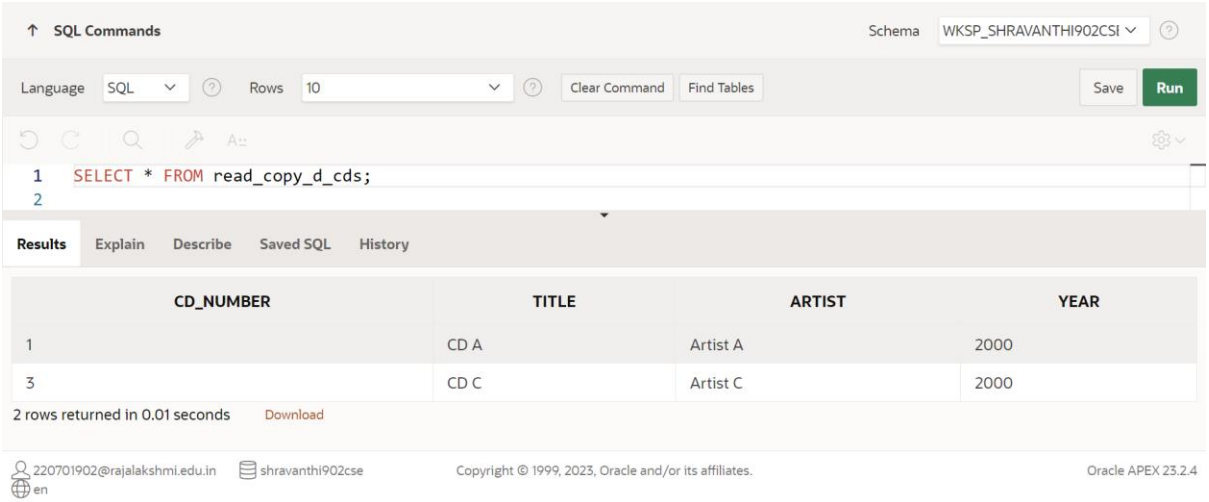
ID	TITLE	DURATION	ARTIST	TYPE_CODE
88	Mello Jello	2	The What	4

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



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

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

SELECT * FROM read_copy_d_cds;
```

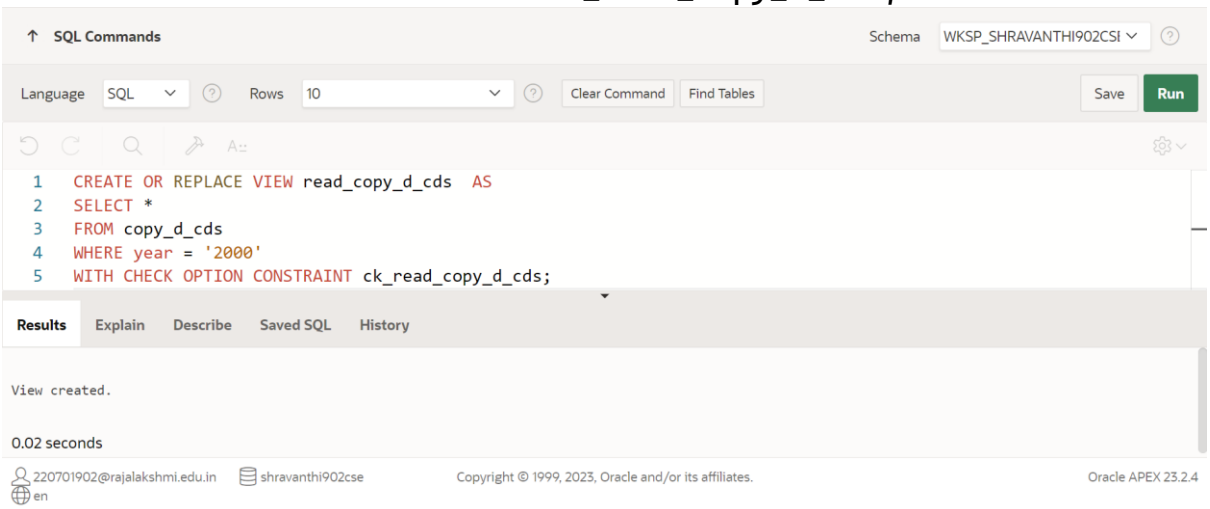


5. Using the read_copy_d_cds view, execute a DELETE FROM read_copy_d_cds WHERE cd_number = 90;

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

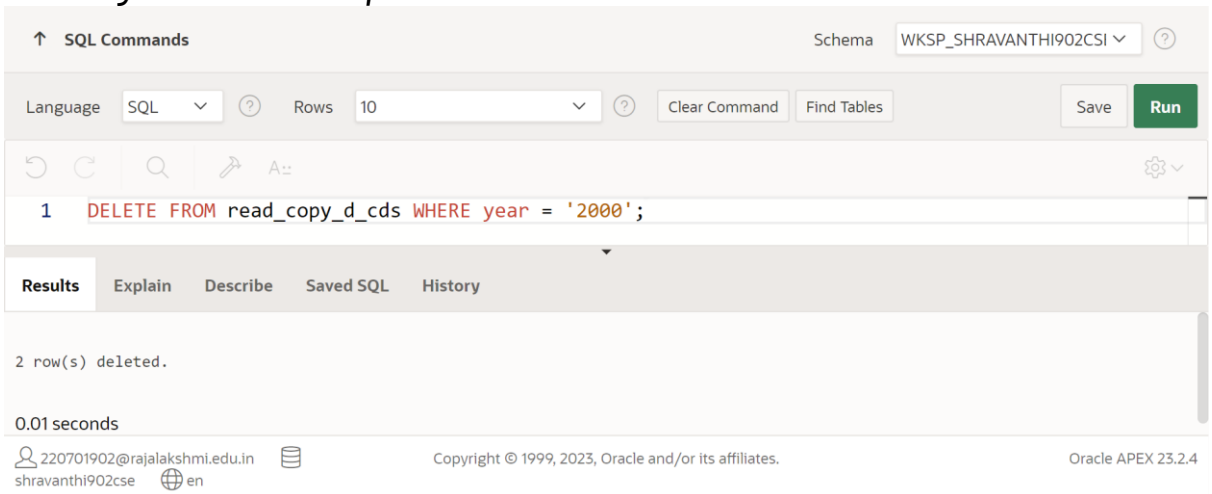
6. Use REPLACE to modify read_copy_d_cds. Replace the READ ONLY option with WITH CHECK OPTION CONSTRAINT ck_read_copy_d_cds. Execute a SELECT * statement to verify that the view exists.

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



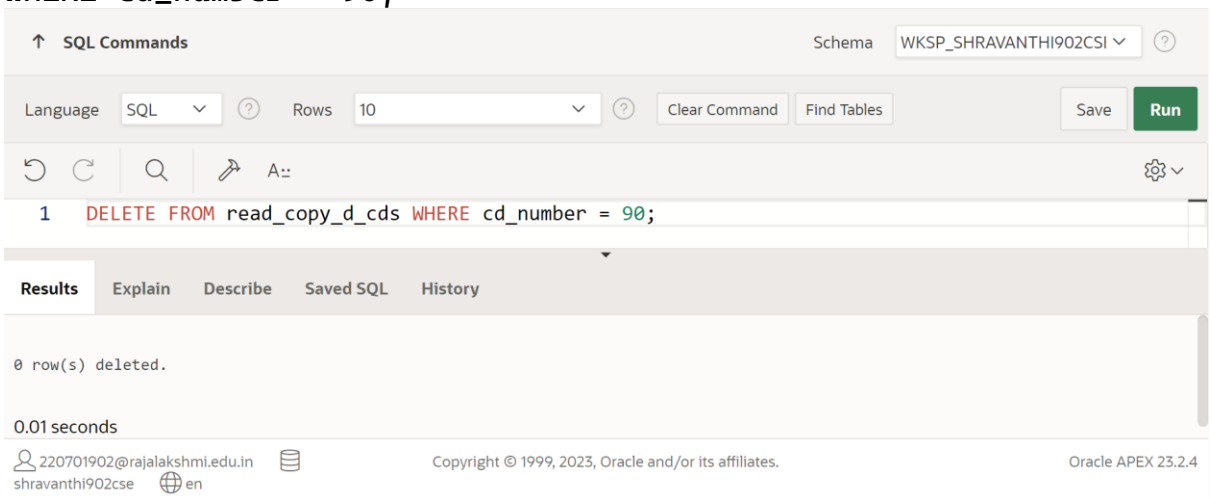
7. Use the read_copy_d_cds view to delete any CD of year 2000 from the underlying copy_d_cds.

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



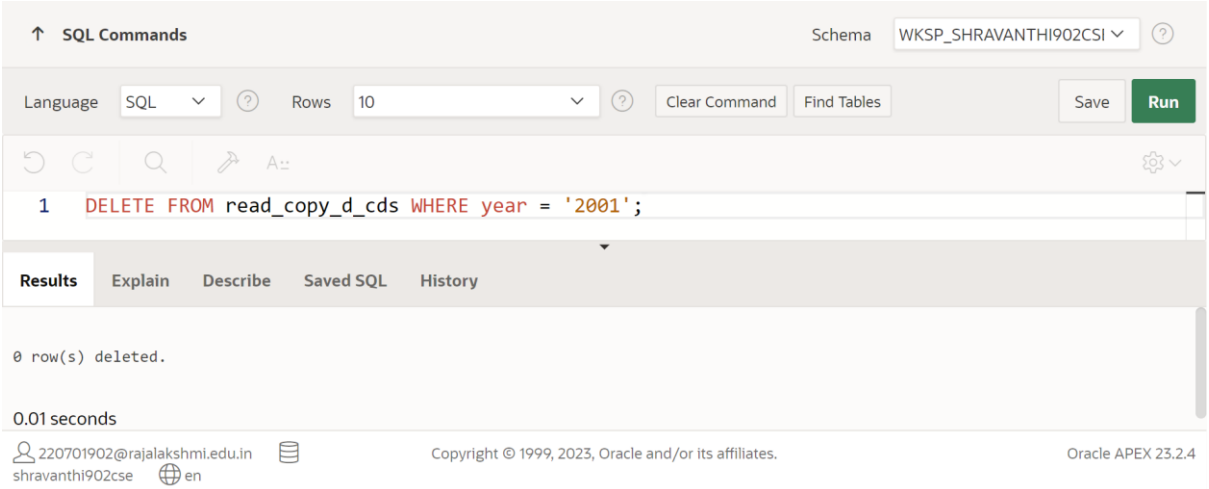
8. Use the read_copy_d_cds view to delete cd_number 90 from the underlying copy_d_cds table.

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



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

DELETE FROM read_copy_d_cds
WHERE year = '2001';



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

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

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

```
DELETE.INSERT,MODIFY restricted if it contains:

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

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

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

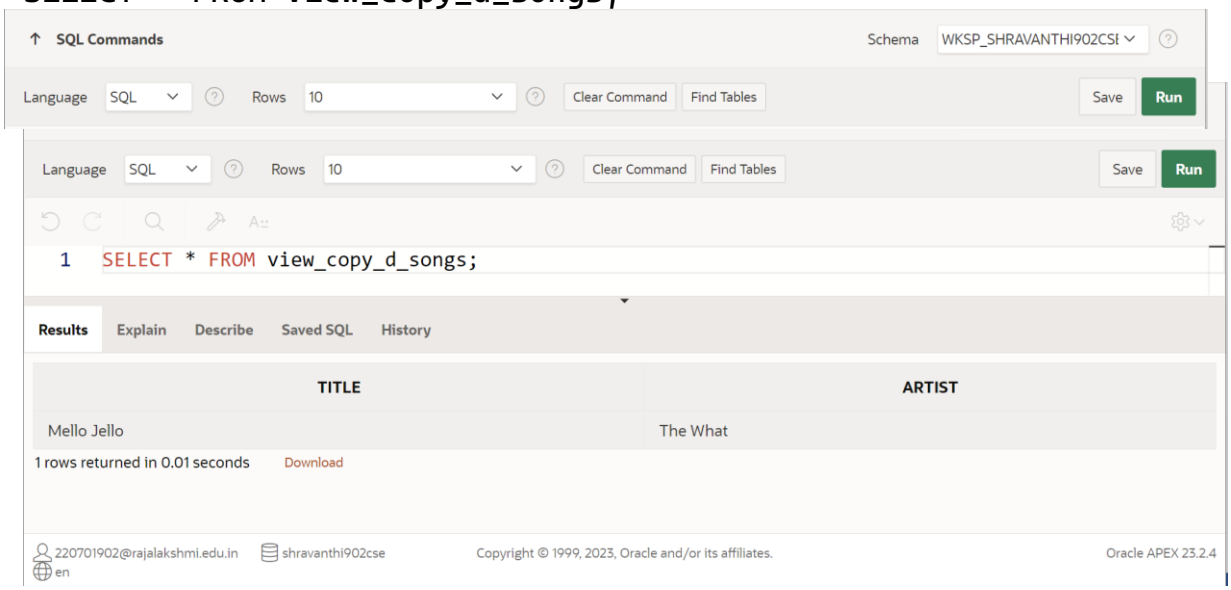
13. What is the "singularity" in terms of computing?

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

Managing Views

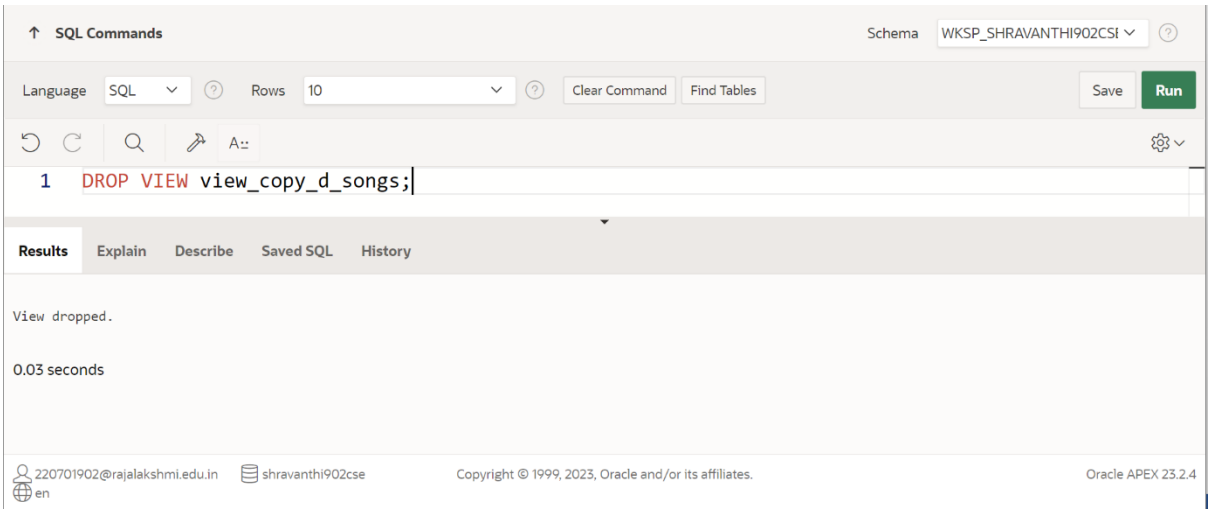
1. Create a view from the copy_d_songs table called view_copy_d_songs that includes only the title and artist. Execute a SELECT * statement to verify that the view exists.

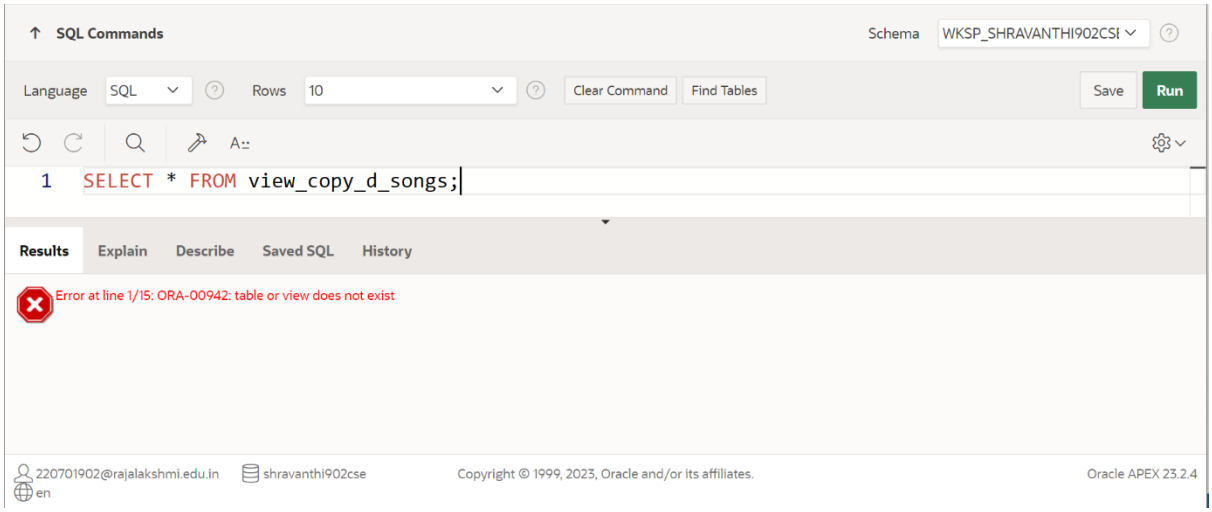
```
CREATE OR REPLACE VIEW view_copy_d_songs  AS
SELECT title, artist
FROM copy_d_songs;

SELECT * FROM view_copy_d_songs;
```



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

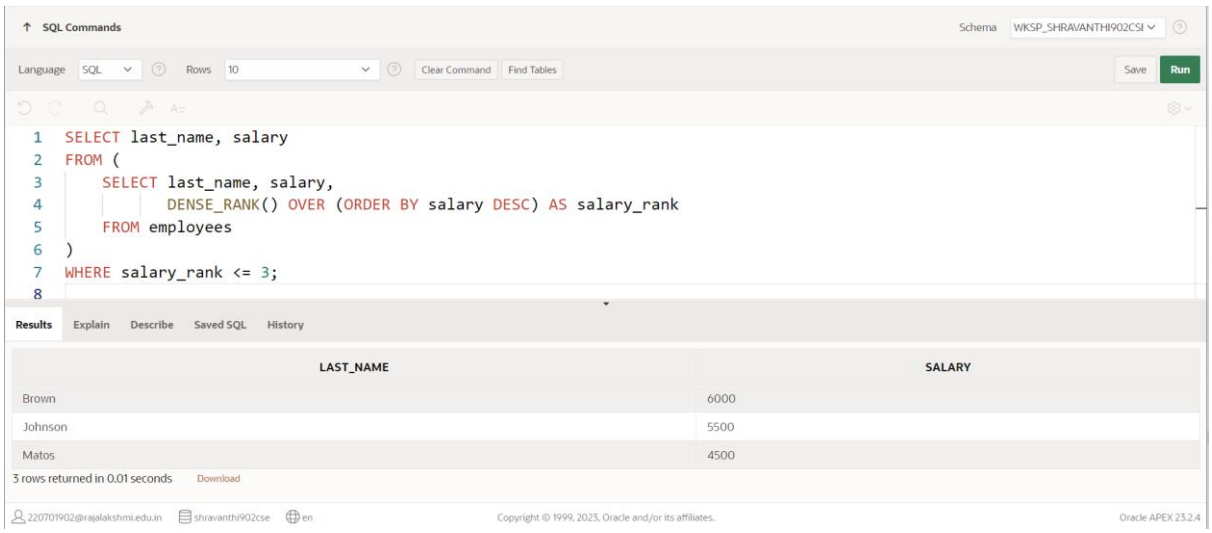
```
DROP VIEW view_copy_d_songs;
SELECT * FROM view_copy_d_songs;
```





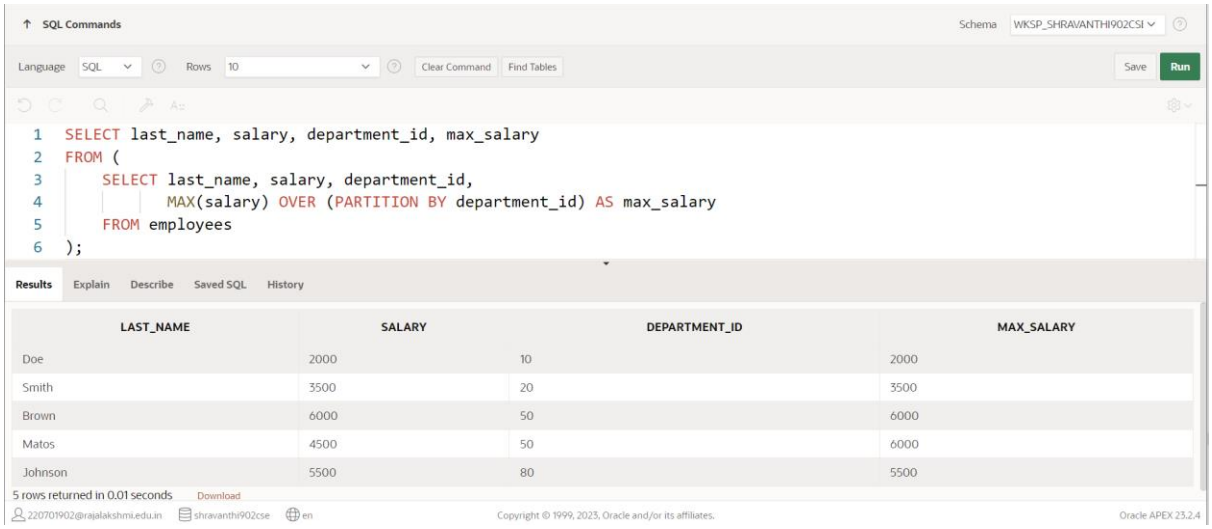
3. Create a query that selects the last name and salary from the Oracle database. Rank the salaries from highest to lowest for the top three employees.

```
SELECT last_name, salary
FROM (
    SELECT last_name, salary,
           DENSE_RANK() OVER (ORDER BY salary DESC) AS salary_rank
    FROM employees
)
WHERE salary_rank <= 3;
```



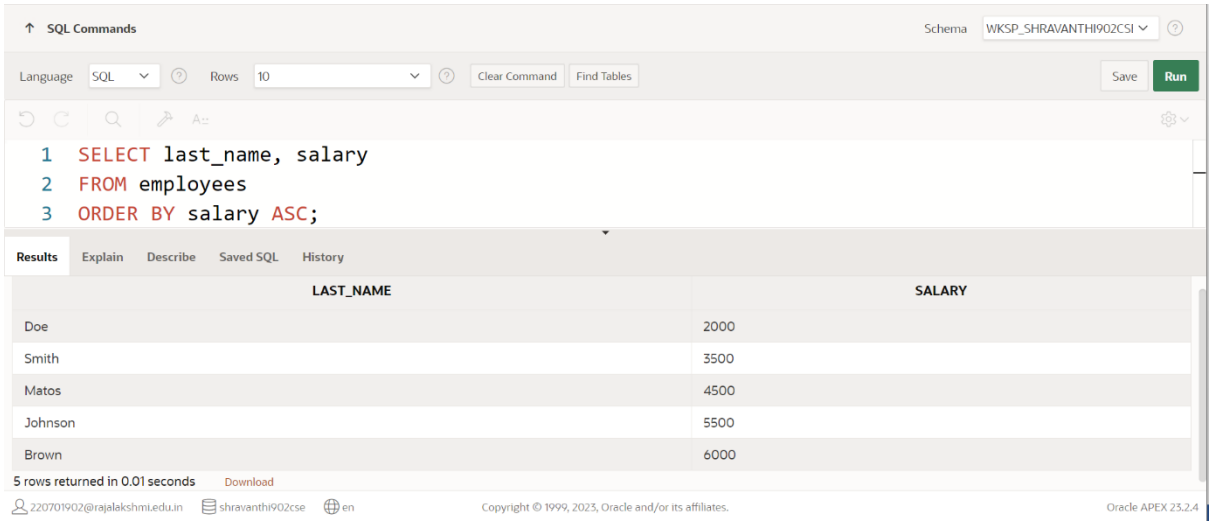
4. Construct an inline view from the Oracle database that lists the last name, salary, department ID, and maximum salary for each department. Hint: One query will need to calculate maximum salary by department ID.

```
SELECT last_name, salary, department_id, max_salary
FROM (
    SELECT last_name, salary, department_id,
           MAX(salary) OVER (PARTITION BY department_id) AS max_salary
    FROM employees
);
```



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

```
SELECT last_name, salary
FROM employees
ORDER BY salary ASC;
```



INDEXES AND SYNONYMS

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

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

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

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

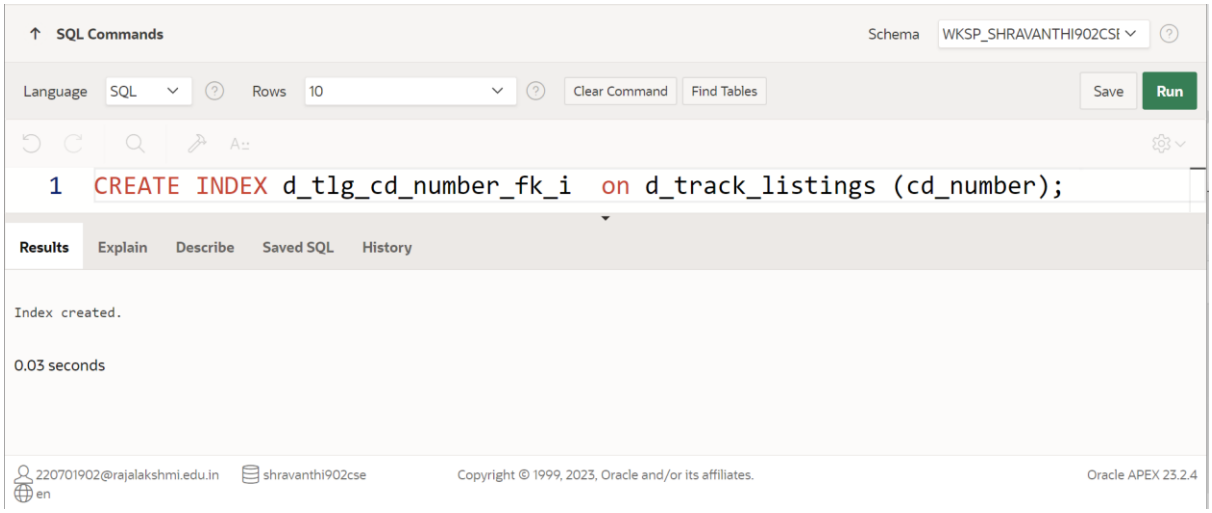
Indexes use ROWID's (base 64 string representation of the row address containing block identifier, row location in the block and the database file identifier) which is the fastest way to access any particular row.

3. When will an index be created automatically?

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

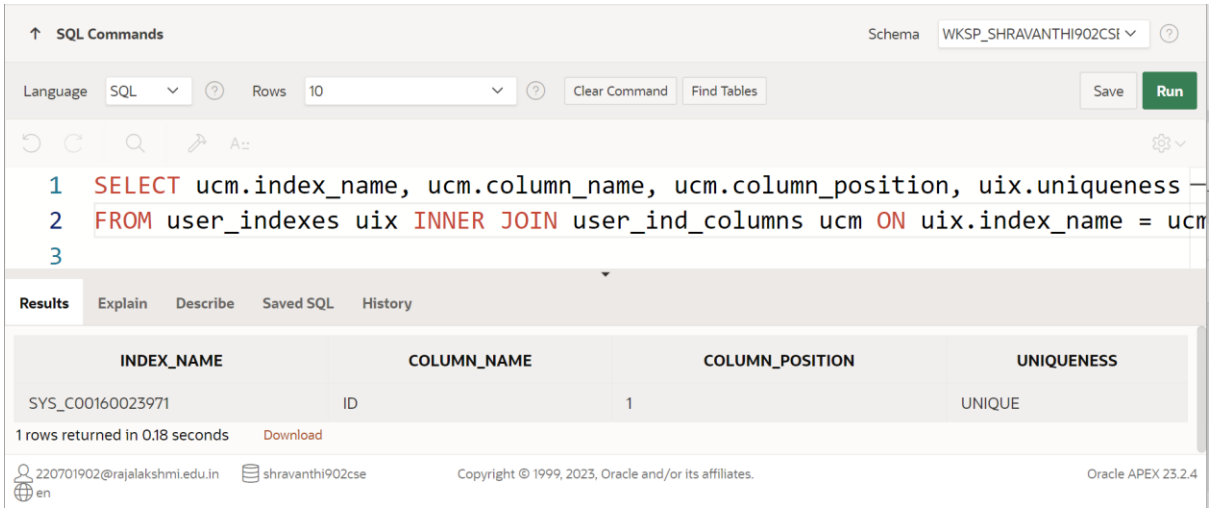
4. Create a nonunique index (foreign key) for the DJs on Demand column (cd_number) in the D_TRACK_LISTINGS table. Use the Oracle Application Express SQL Workshop Data Browser to confirm that the index was created.

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



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

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

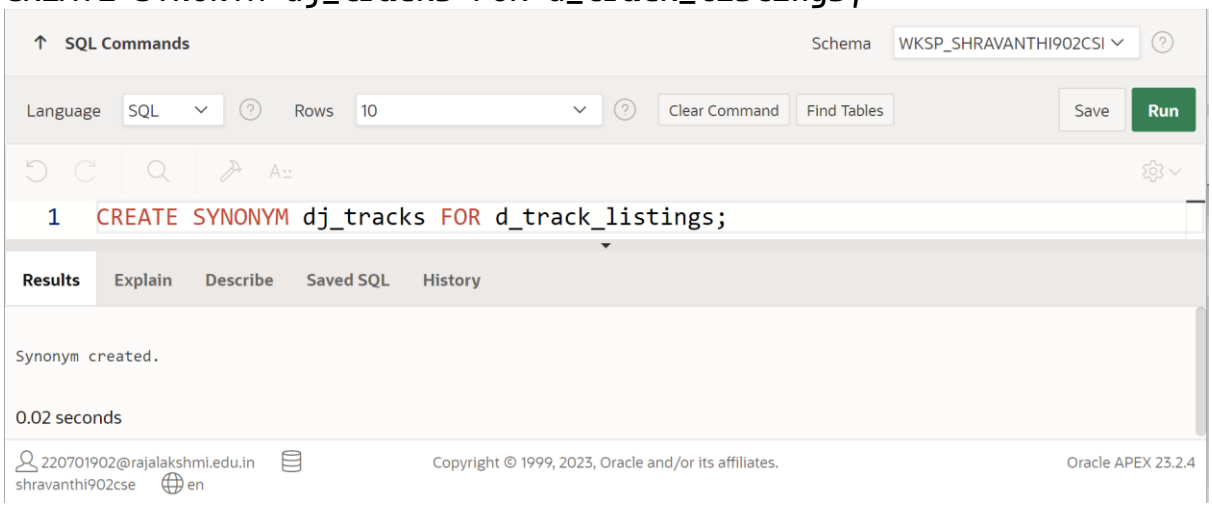


6. Use a SELECT statement to display the index_name, table_name, and uniqueness from the data dictionary USER_INDEXES for the DJs on Demand D_EVENTS table.

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

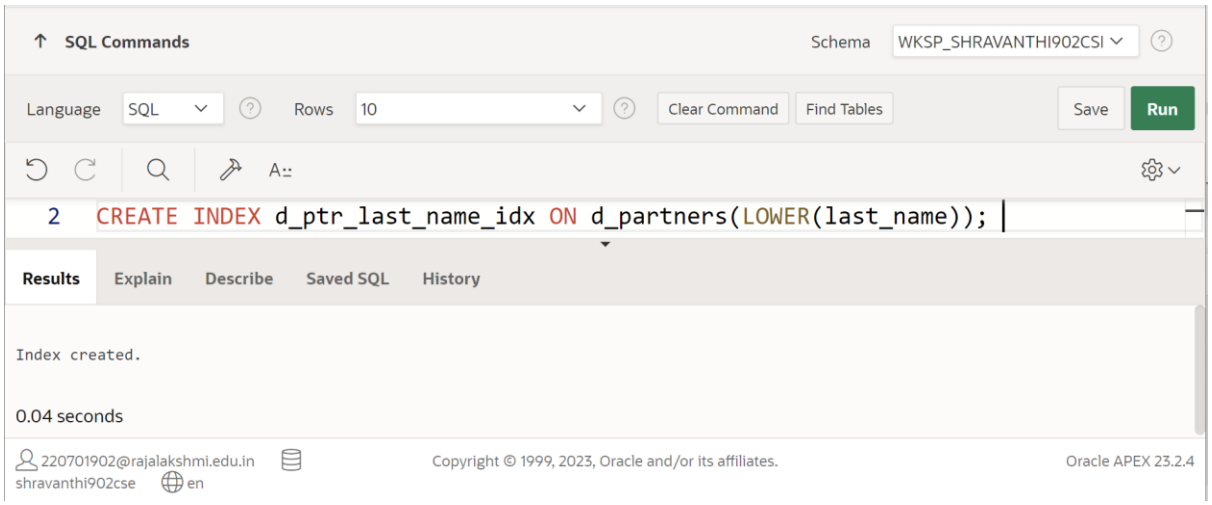
7. Write a query to create a synonym called dj_tracks for the DJs on Demand d_track_listings table.

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



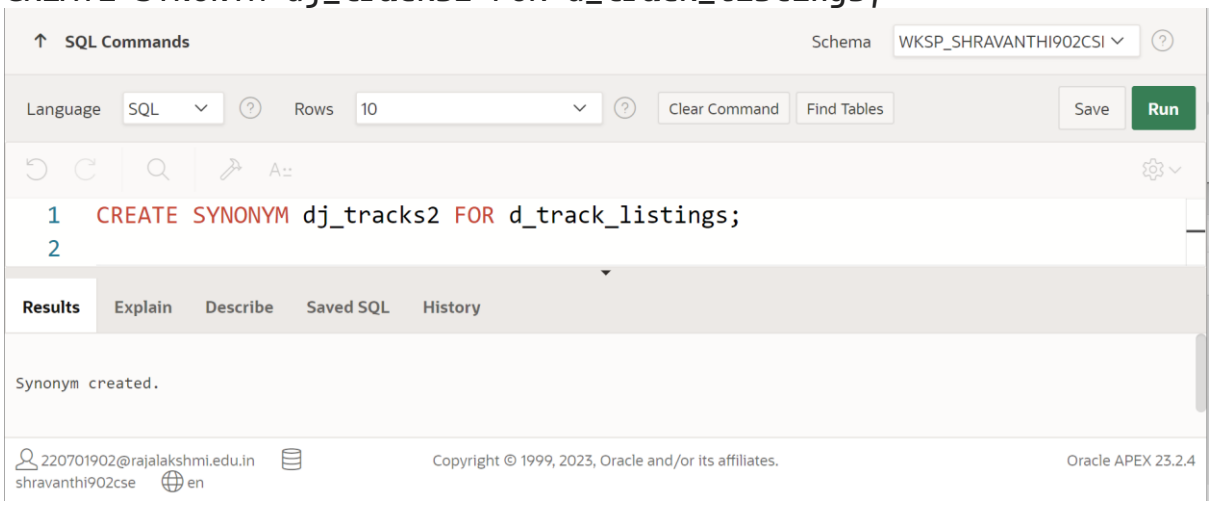
8. Create a function-based index for the last_name column in DJs on Demand D_PARTNERS table that makes it possible not to have to capitalize the table name for searches. Write a SELECT statement that would use this index.

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

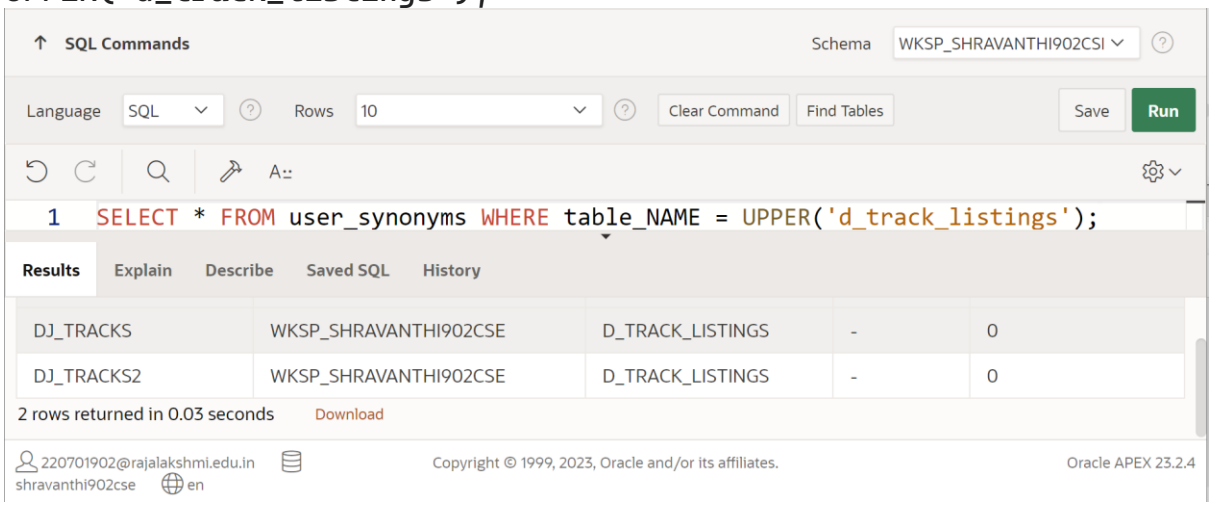


9. Create a synonym for the D_TRACK_LISTINGS table. Confirm that it has been created by querying the data dictionary.

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

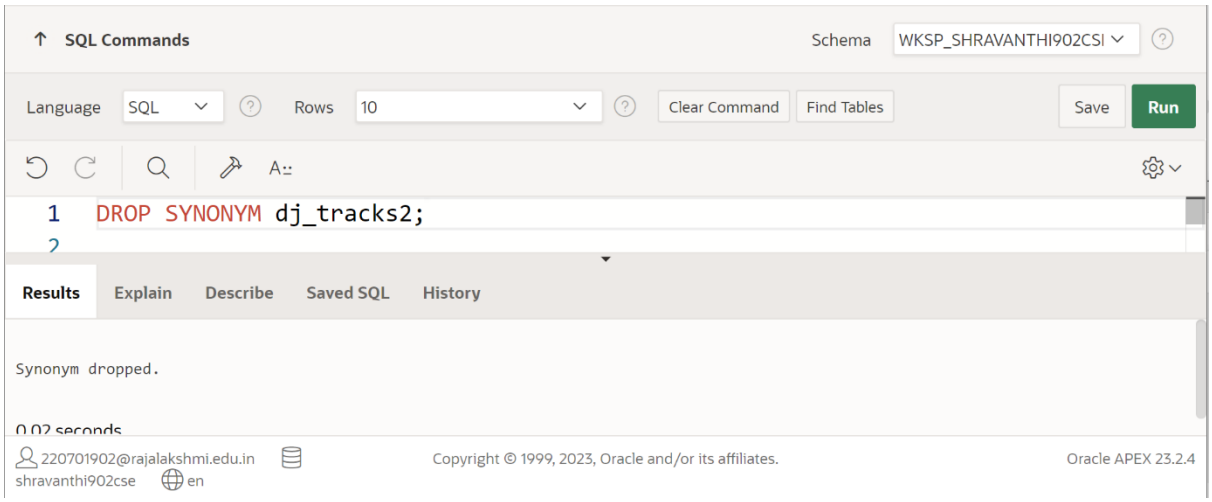



```
SELECT * FROM user_synonyms WHERE table_NAME =
UPPER('d_track_listings');
```



10. Drop the synonym that you created in question :

```
DROP SYNONYM dj_tracks2;
```



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

RESULT:

OTHER DATABASE OBJECTS

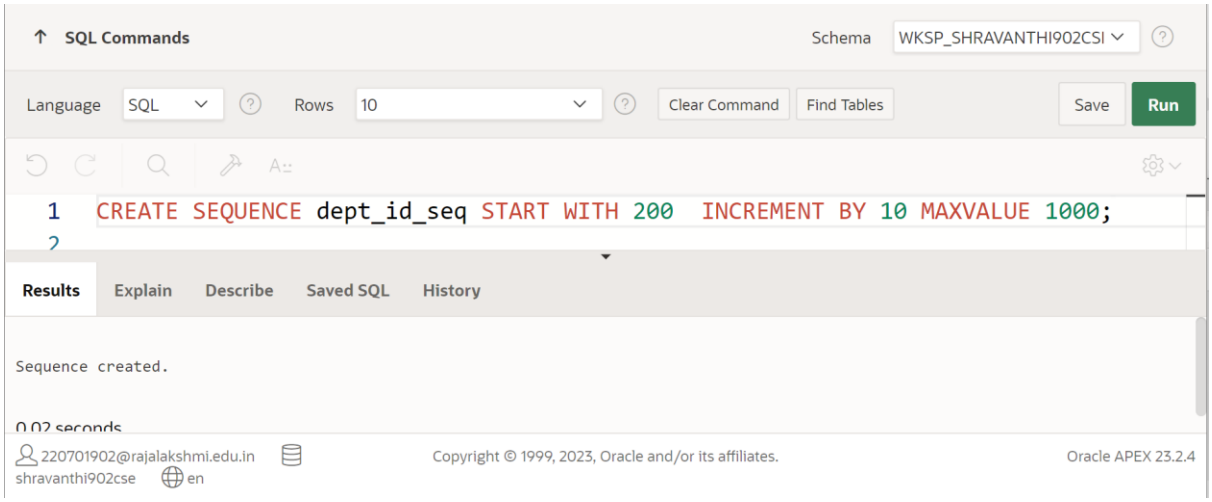
EX.NO:14 **DATE:**

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

QUERY:

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

OUTPUT:

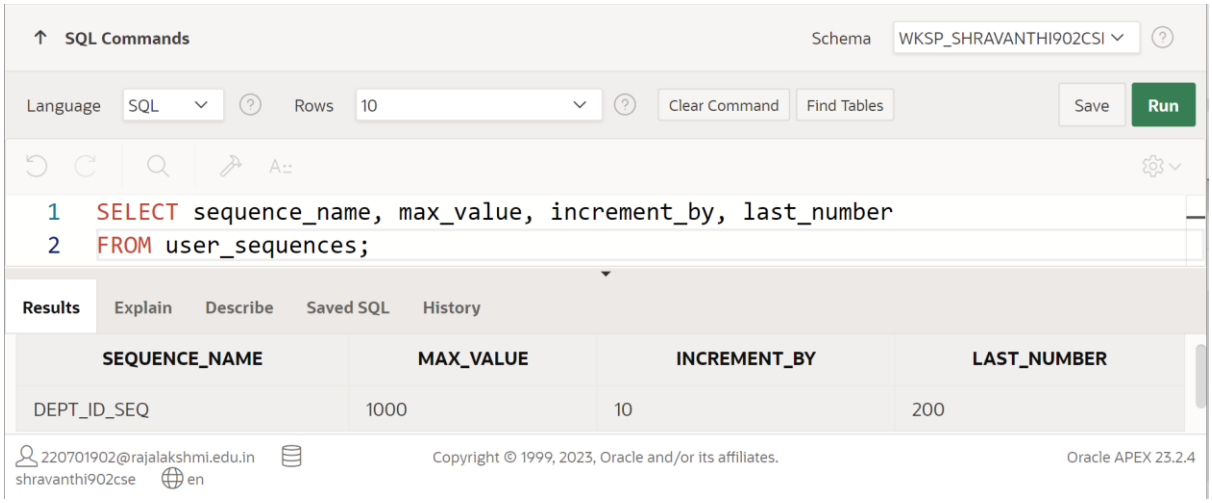


2.)Write a query in a script to display the following information about your sequences: sequence name, maximum value, increment size, and last number

QUERY:

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

OUTPUT:

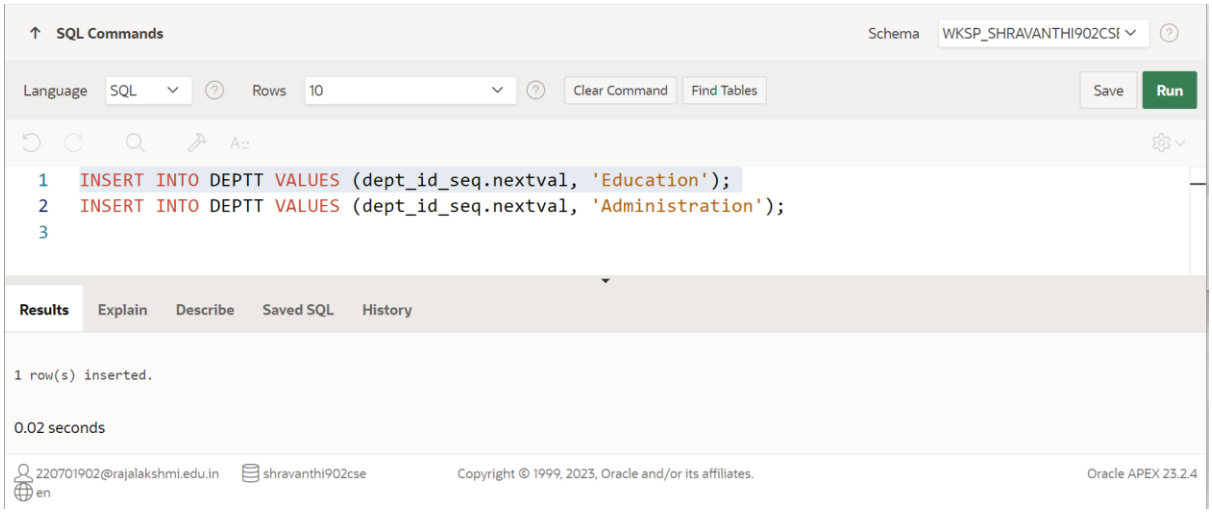


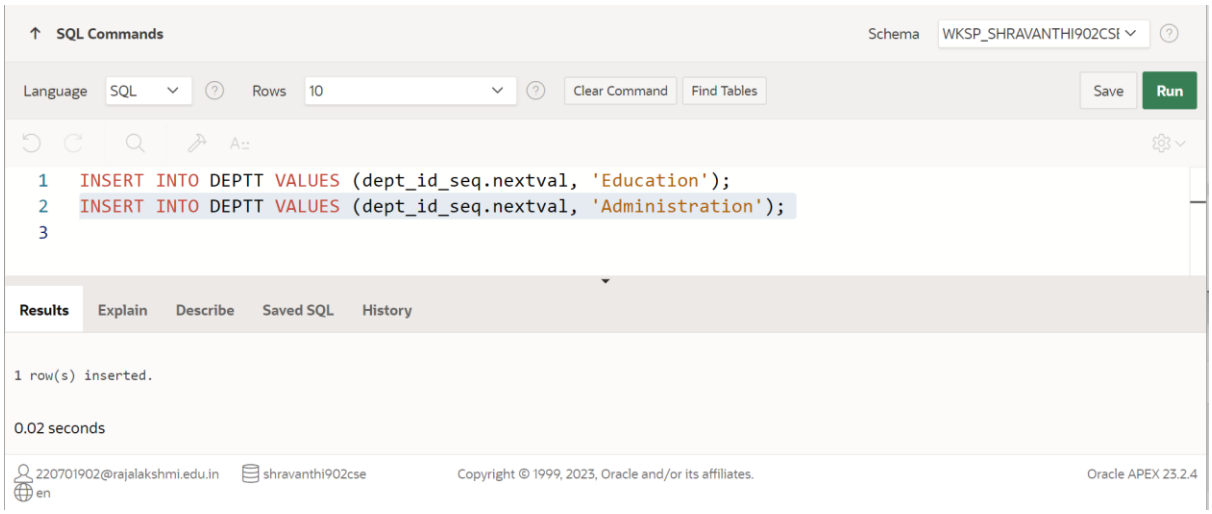
3.) Write a script to insert two rows into the DEPT table. Name your script lab12_3.sql. Be sure to use the sequence that you created for the ID column. Add two departments named Education and Administration. Confirm your additions. Run the commands in your script.

QUERY:

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

OUTPUT:



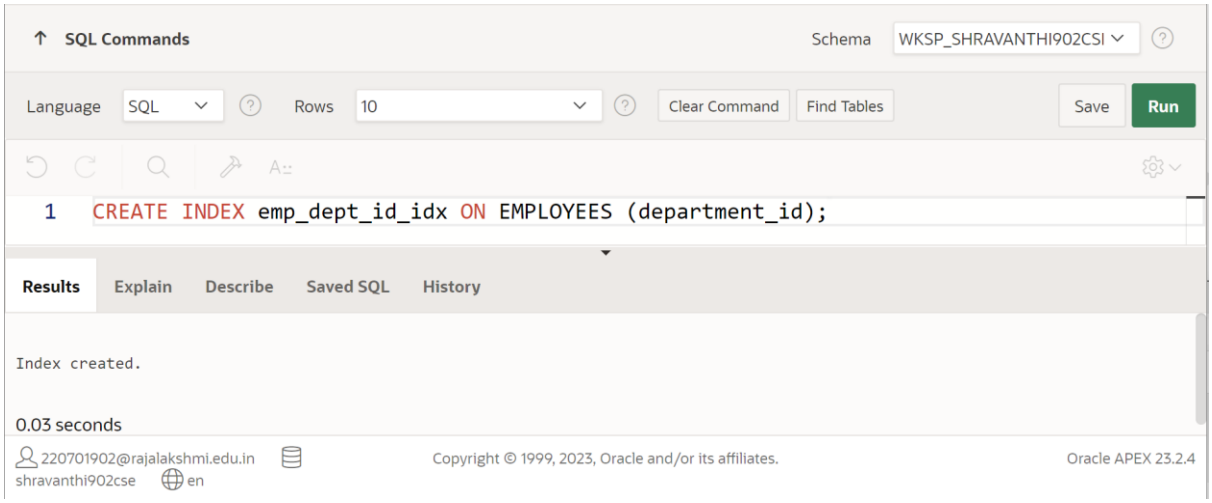


4.)Create a nonunique index on the foreign key column (DEPT_ID) in the EMP table.

QUERY:

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

OUTPUT:

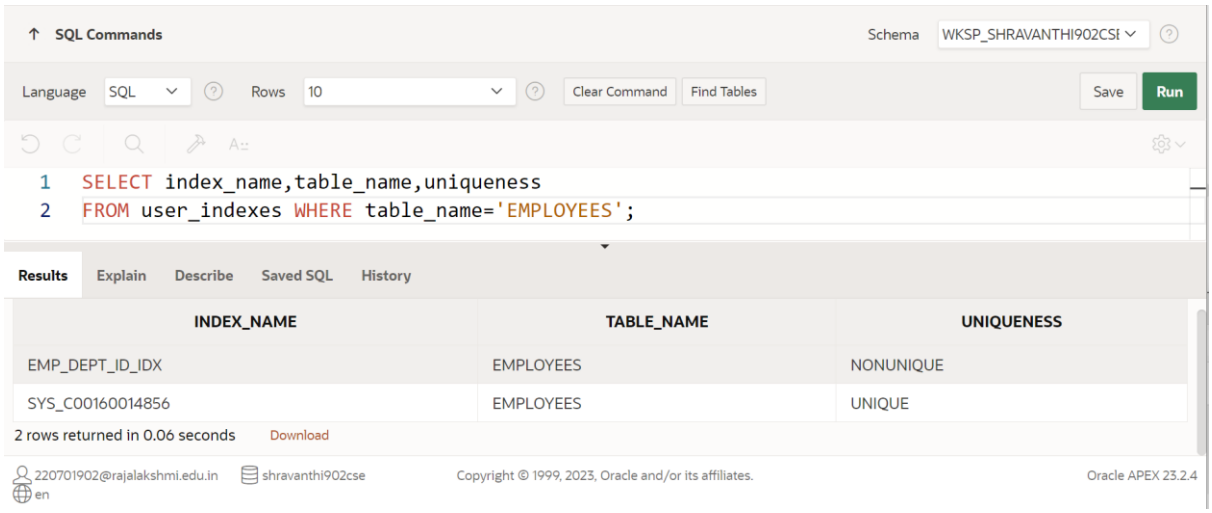


5.)Display the indexes and uniqueness that exist in the data dictionary for the EMP table.

QUERY:

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

OUTPUT:



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

RESULT:

CONTROLLING USER ACCESS

EX.NO:15 **DATE:**

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

The CREATE SESSION system privilege

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

The CREATE TABLE privilege

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

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

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

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

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

The ALTER USER statement

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

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

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

7. Query all the rows in your DEPARTMENTS table.

SELECT * FROM departments;

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

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

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

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

SELECT table_name FROM user_tables;

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

Team 1 revokes the privilege.

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

Team 2 revokes the privilege.

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

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

Team 1 executes this INSERT statement.

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

Team 2 executes this INSERT statement.

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

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

RESULT:

PL/SQL
CONTROL STRUCTURES

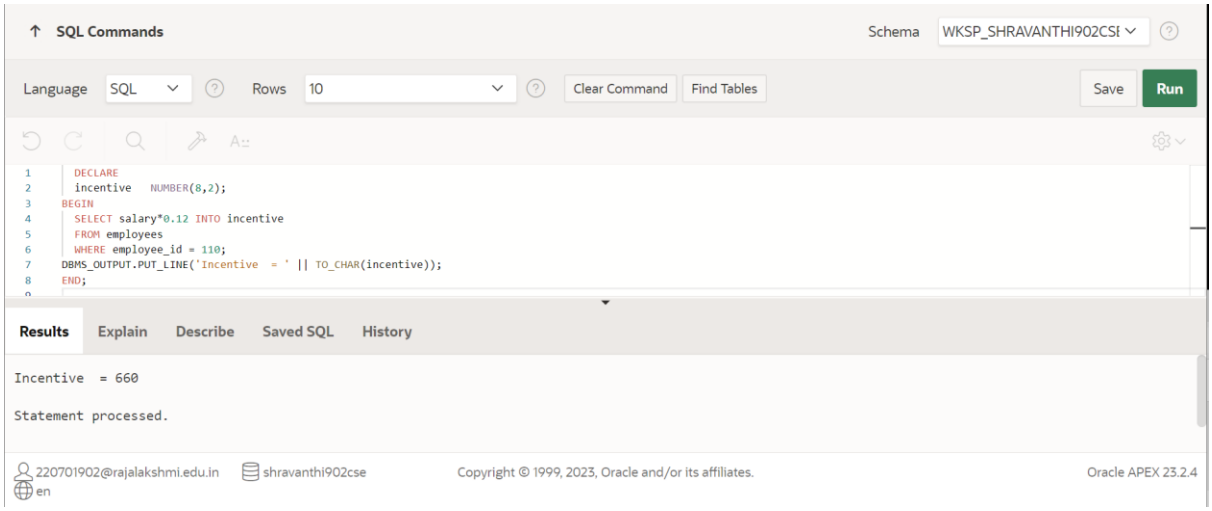
EX.NO:16 **DATE:**

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:



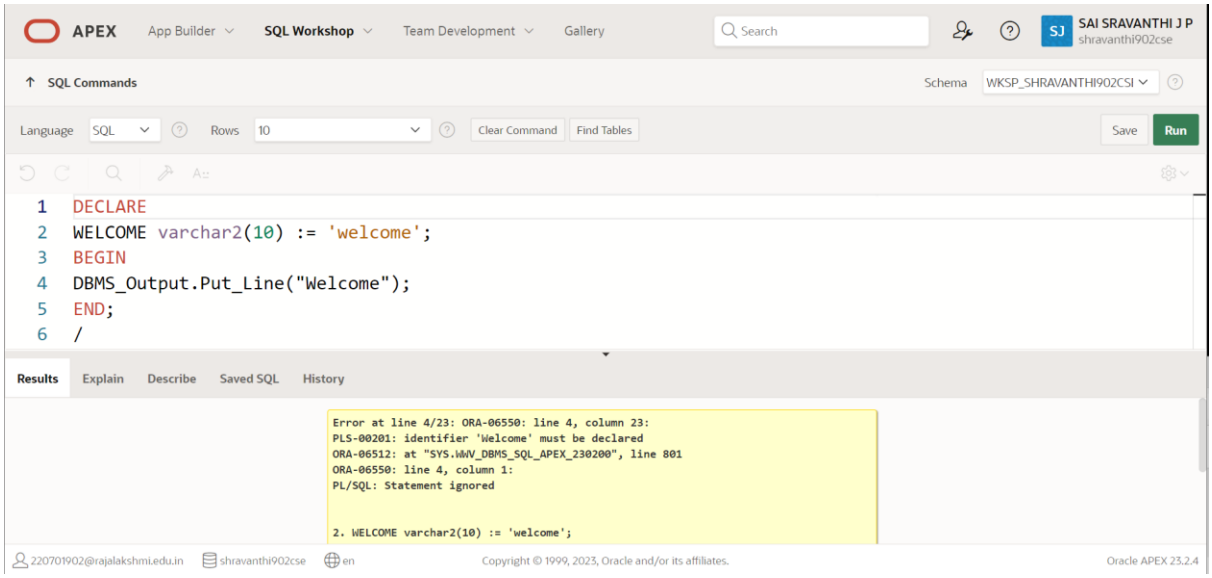
2.) Write a PL/SQL block to show an invalid case-insensitive reference to a quoted and without quoted user-defined identifier

QUERY:

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

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

OUTPUT:

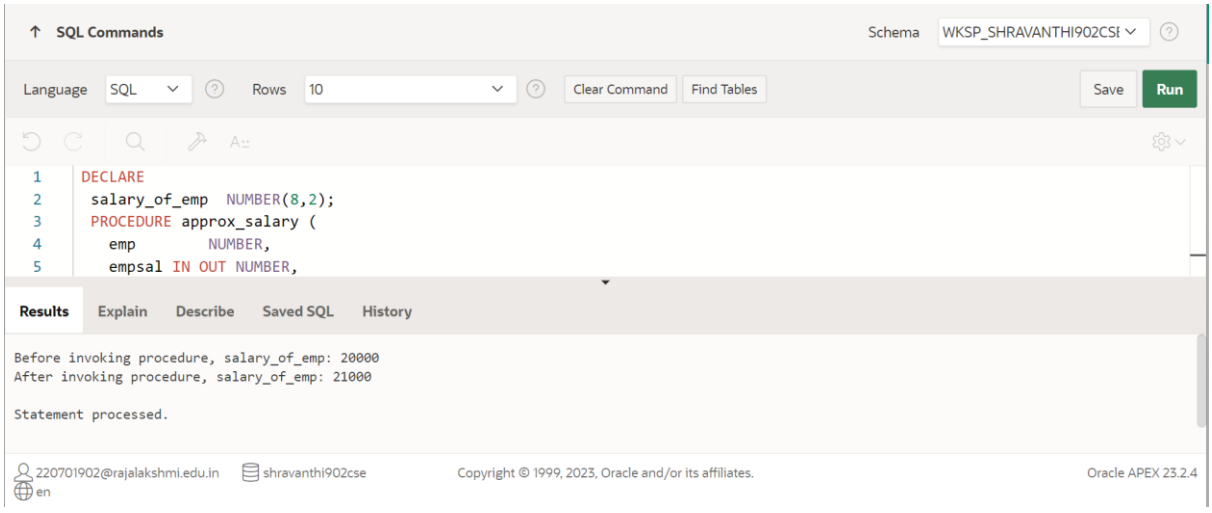


3.) Write a PL/SQL block to adjust the salary of the employee whose ID 122.

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:

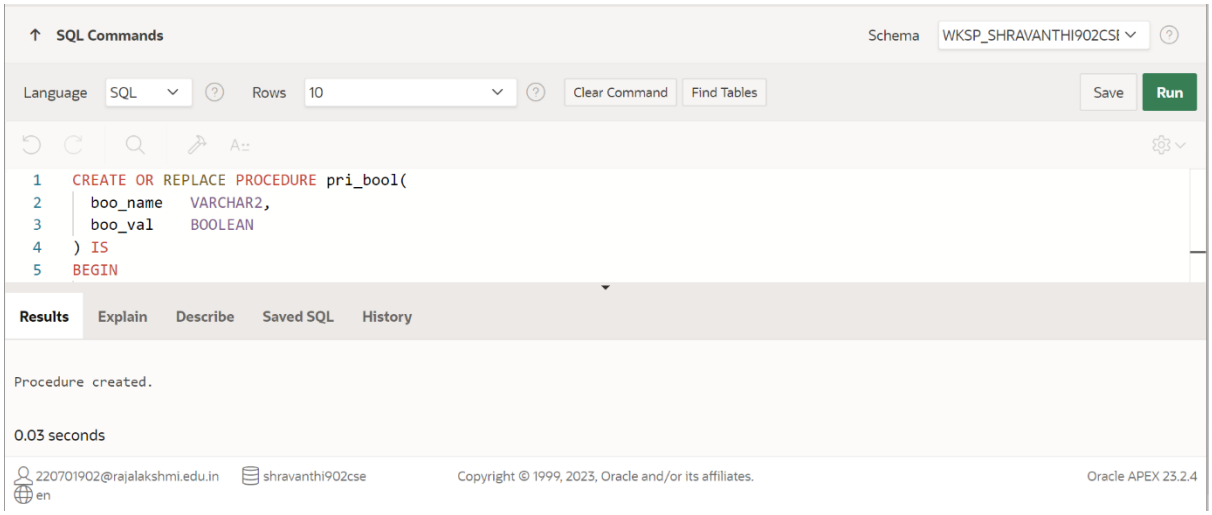


4.) Write a PL/SQL block to create a procedure using the "IS [NOT] NULL Operator" and show AND operator returns TRUE if and only if both operands are TRUE.

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:

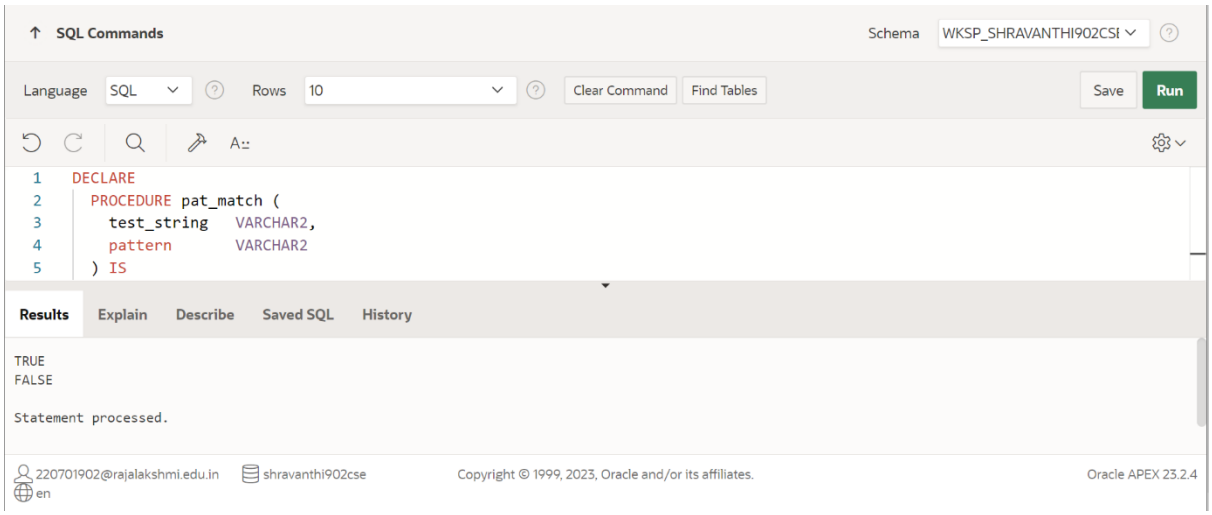


5.) Write a PL/SQL block to describe the usage of LIKE operator including wildcard characters and escape character.

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:

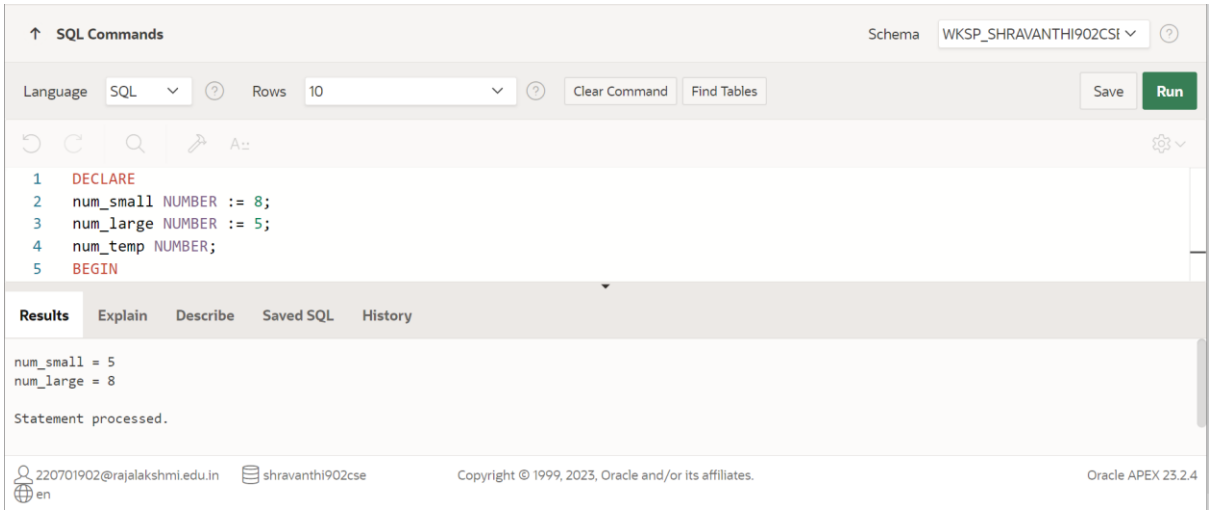


6.) Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num_small variable and large number will store in num_large variable

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:

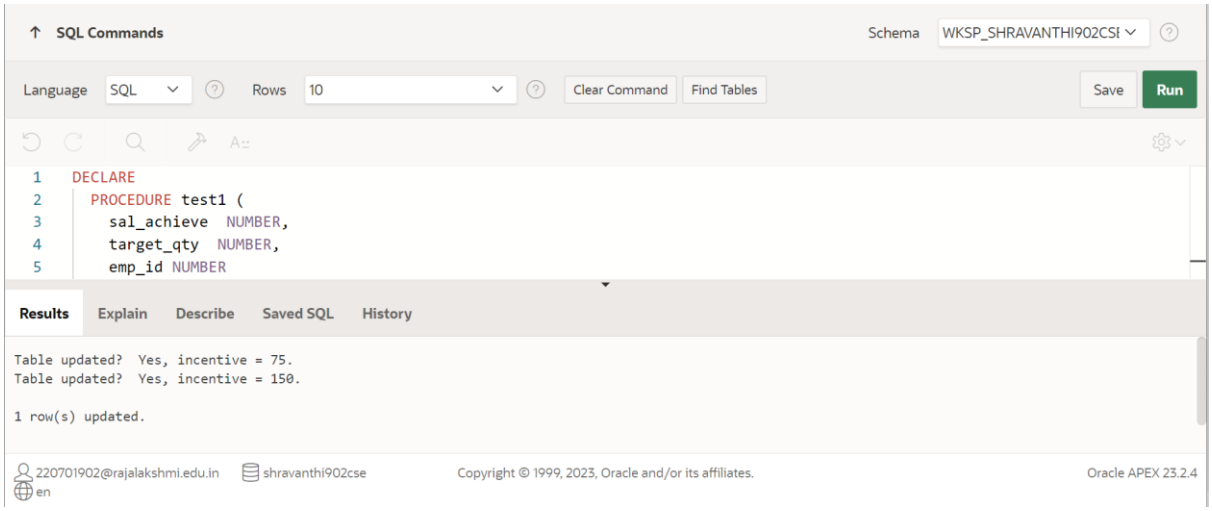


7.) Write a PL/SQL procedure to calculate the incentive on a target achieved and display the message either the record updated or not.

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:

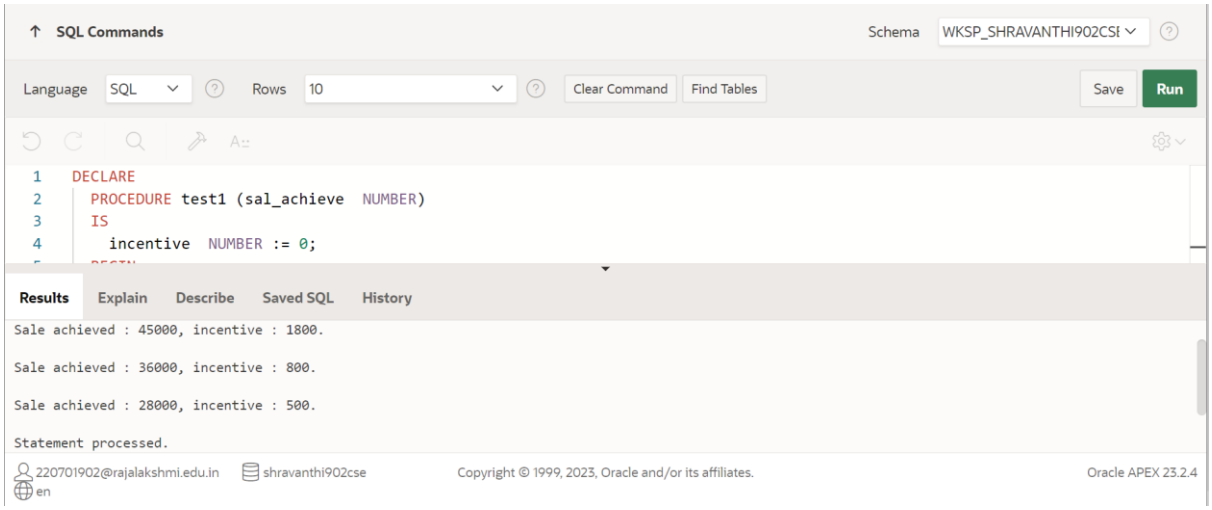


8.) Write a PL/SQL procedure to calculate incentive achieved according to the specific sale limit

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:

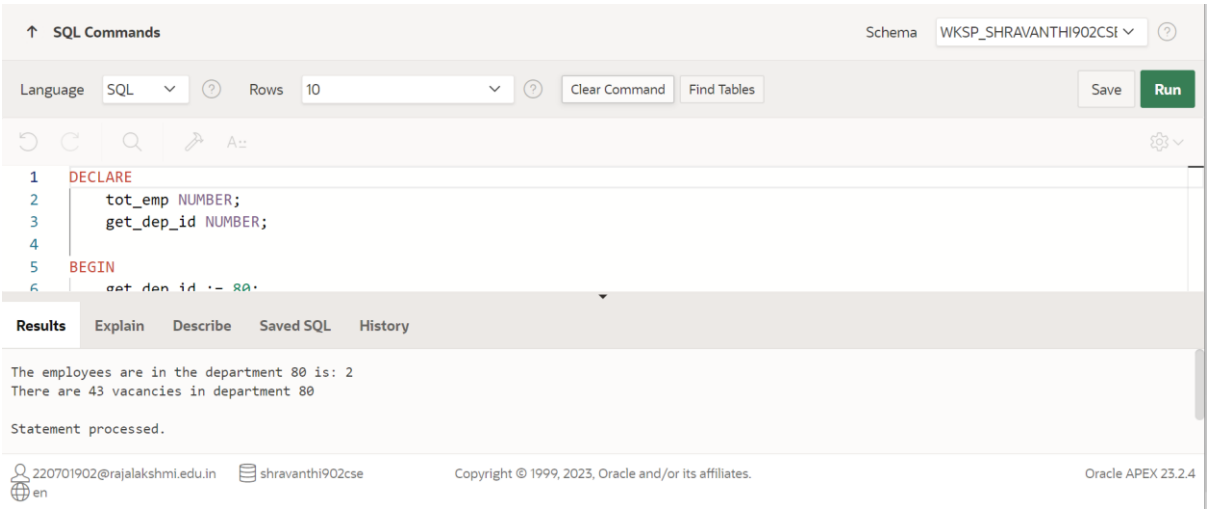


9.) Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

QUERY:

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

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

OUTPUT:

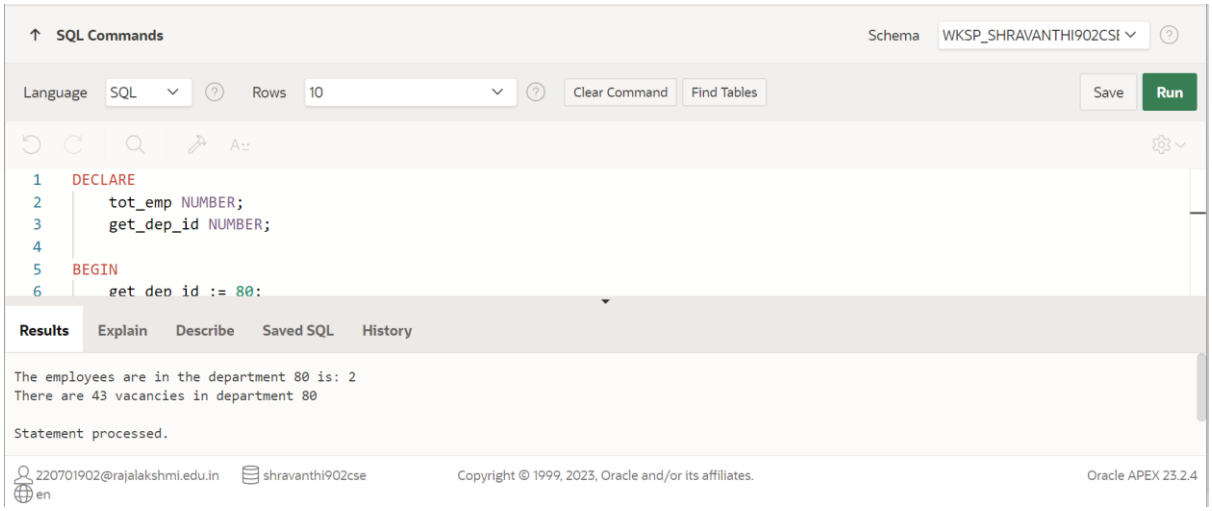


10.) Write a PL/SQL program to count number of employees in a specific department and check whether this department have any vacancies or not. If any vacancies, how many vacancies are in that department.

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:

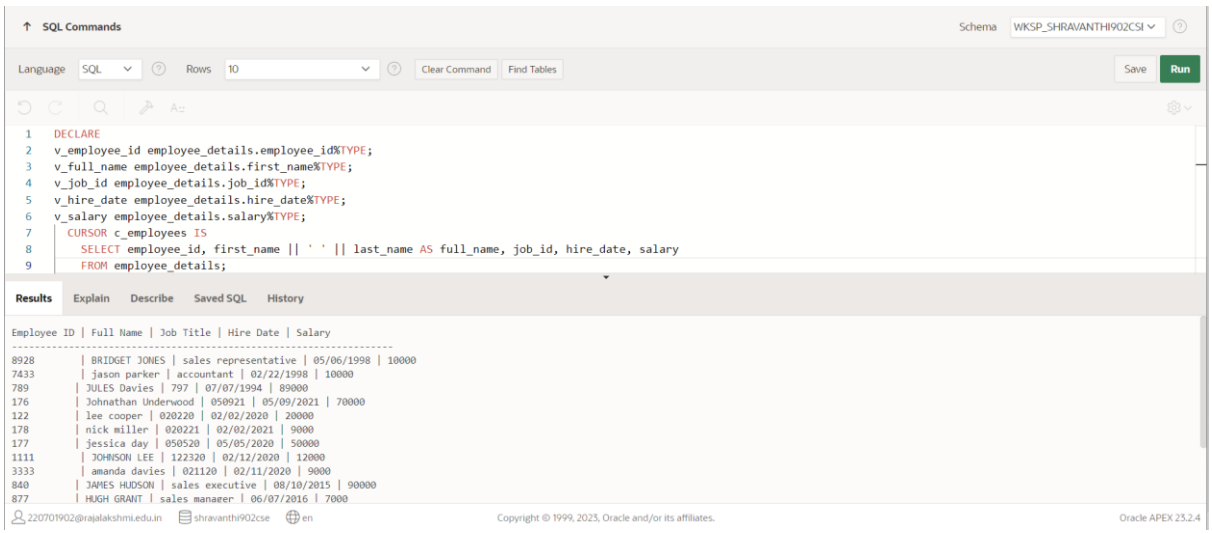


11.) Write a PL/SQL program to display the employee IDs, names, job titles, hire dates, and salaries of all employees

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:

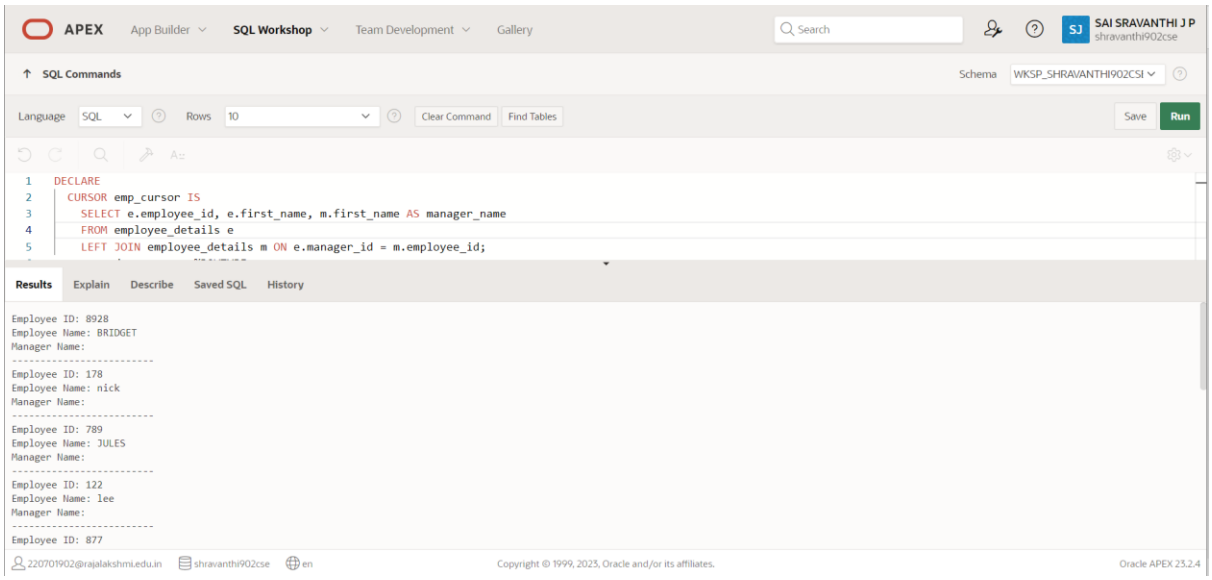


12.) Write a PL/SQL program to display the employee IDs, names, and department names of all employees.

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:

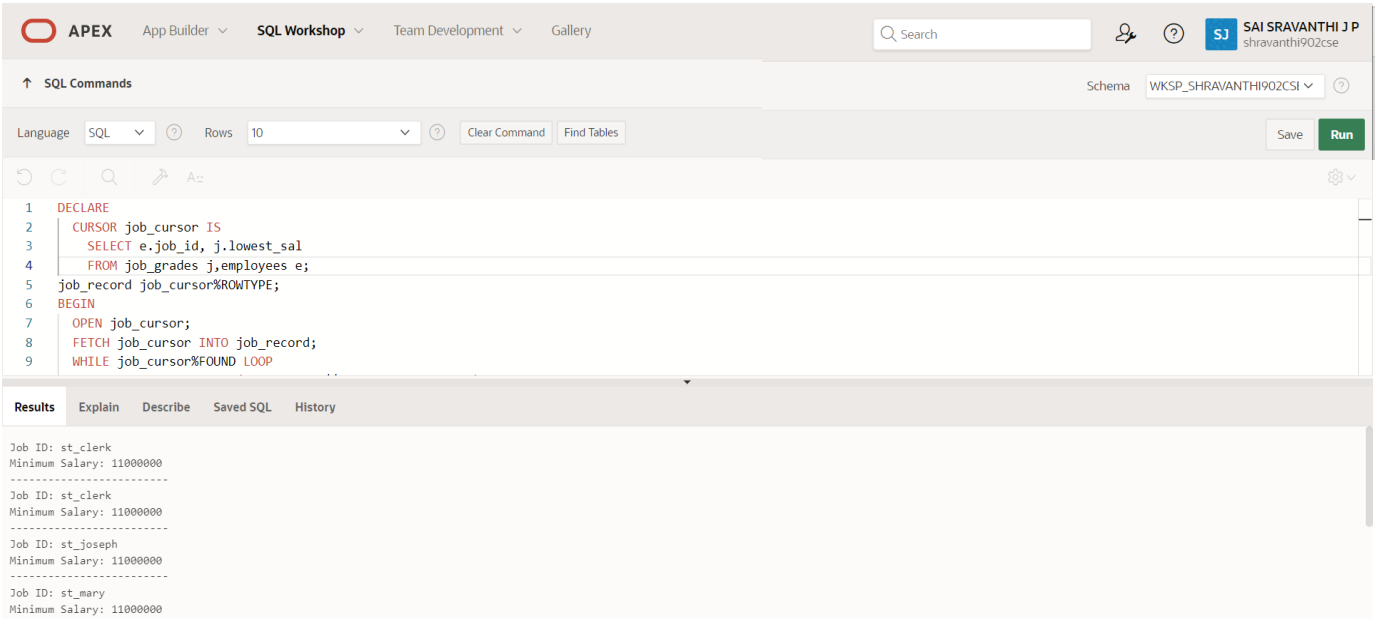


13.) Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all jobs

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:

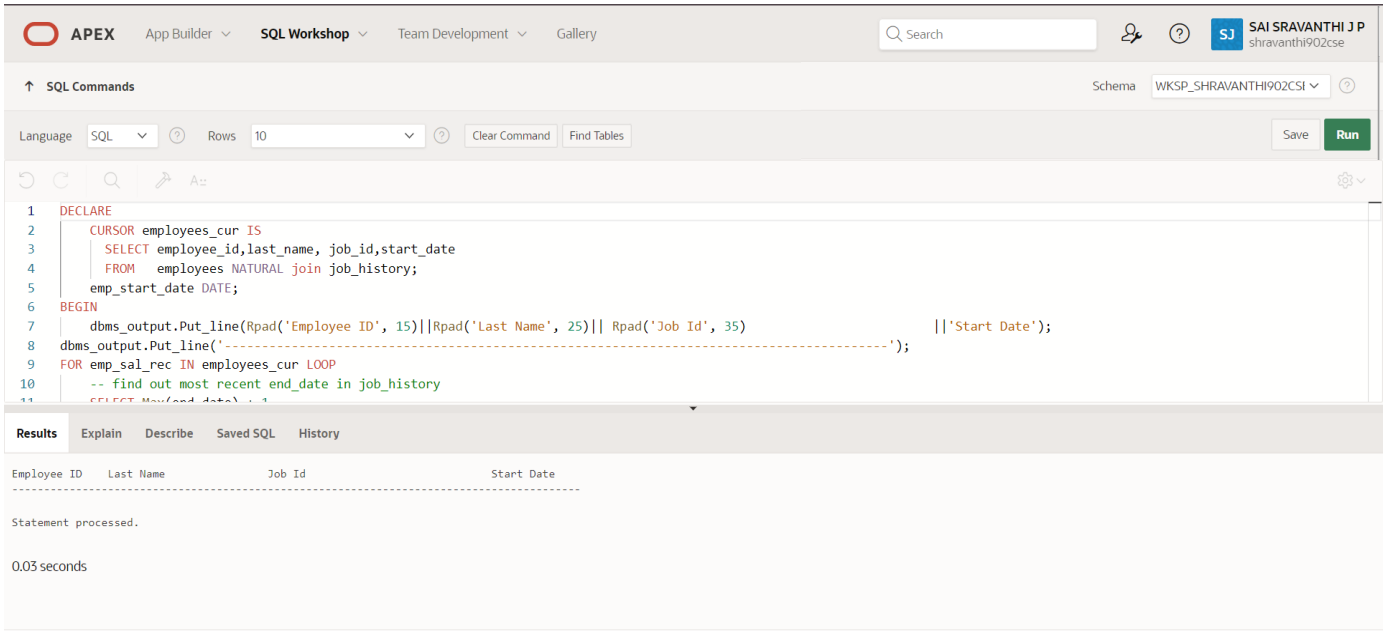


14.) Write a PL/SQL program to display the employee IDs, names, and job history start dates of all employees.

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:

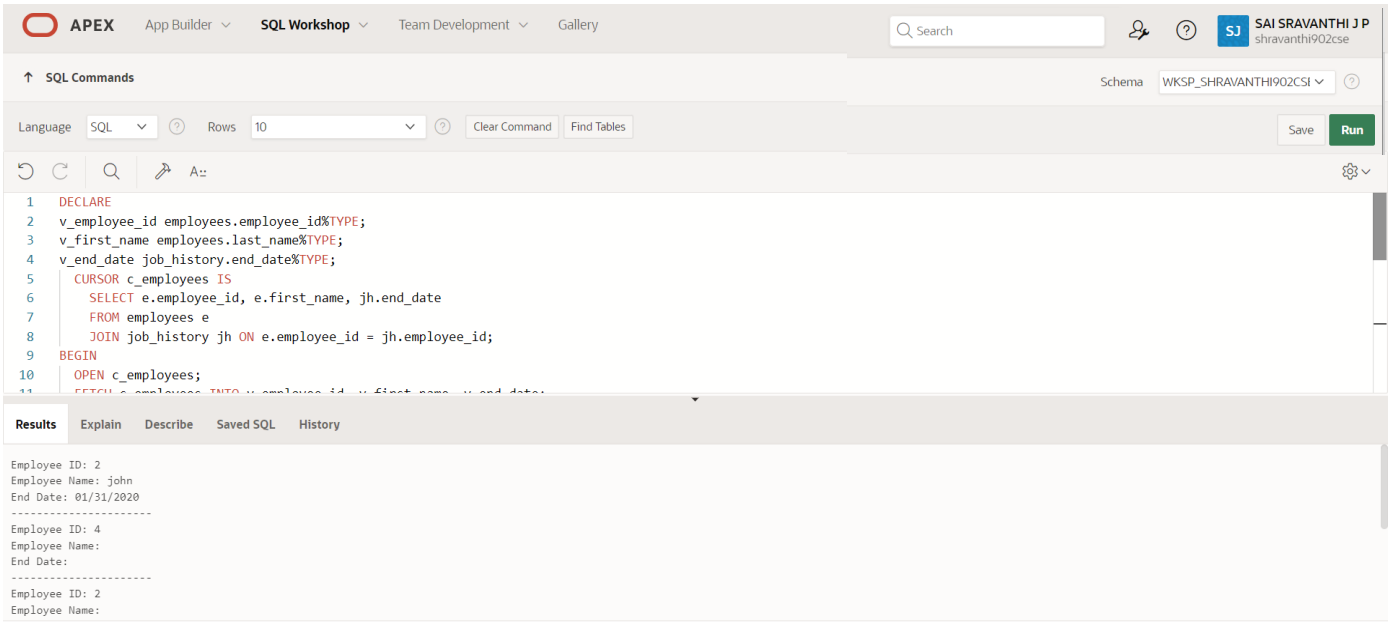


15.) Write a PL/SQL program to display the employee IDs, names, and job history end dates of all employees.

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

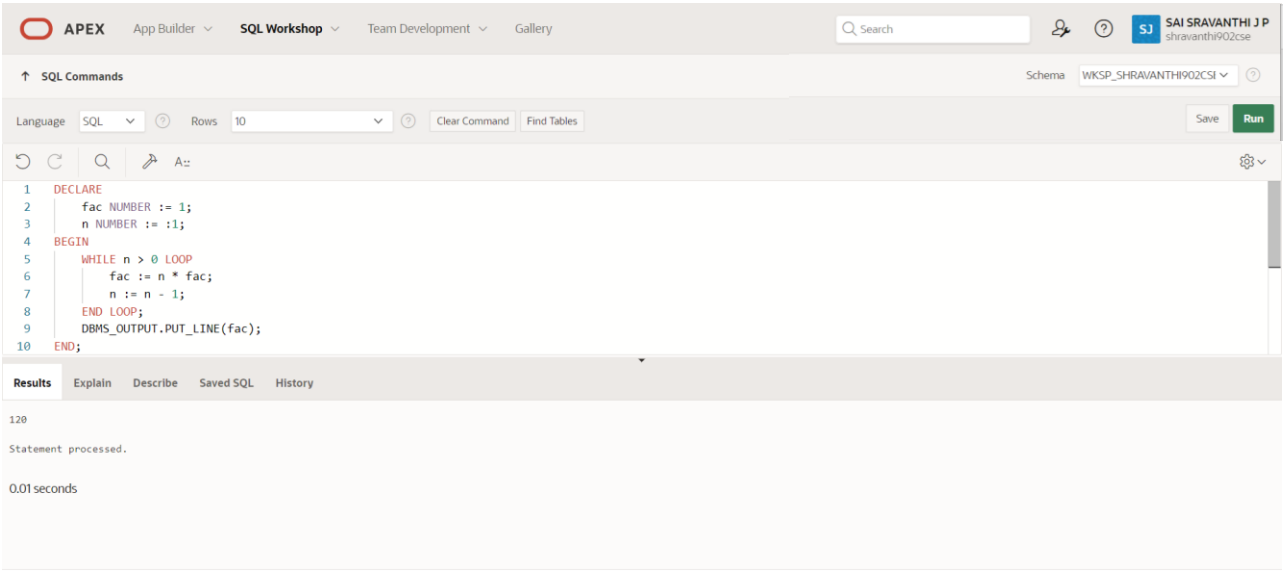
EX.NO: 17 **DATE:**

1.)Factorial of a number using function.

QUERY:

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

OUTPUT:

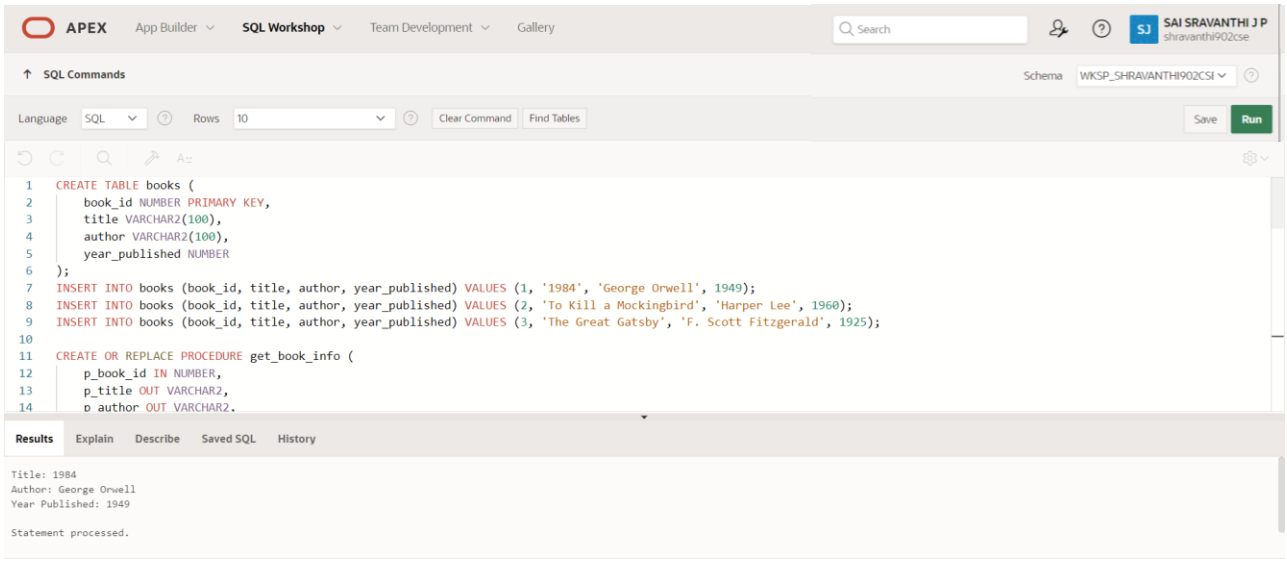


2.) Write a PL/SQL program using Procedures IN,IN,OUT,OUT parameters to retrieve the corresponding book information in library.

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

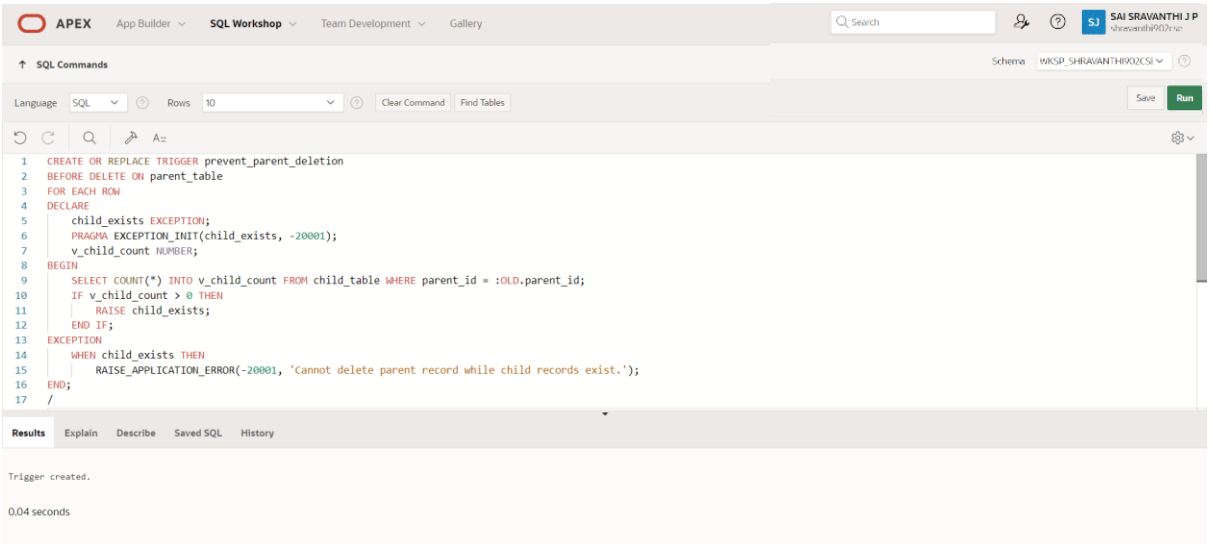
EX_NO: 18 **DATE:**

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

QUERY:

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

OUTPUT:

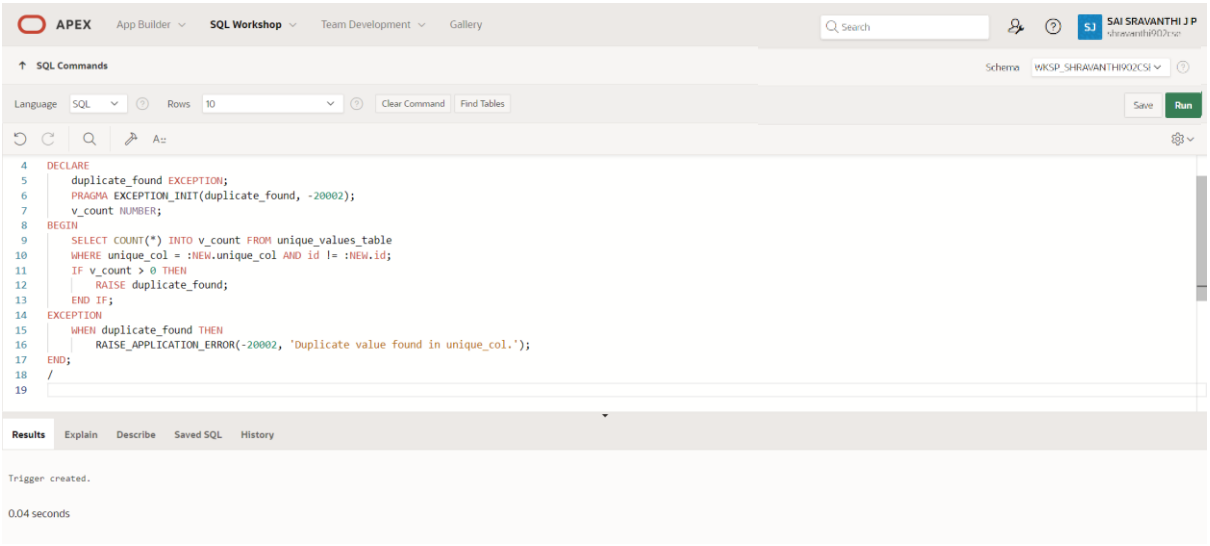


2.) Write a code in PL/SQL to create a trigger that checks for duplicate values in a specific column and raises an exception if found

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:

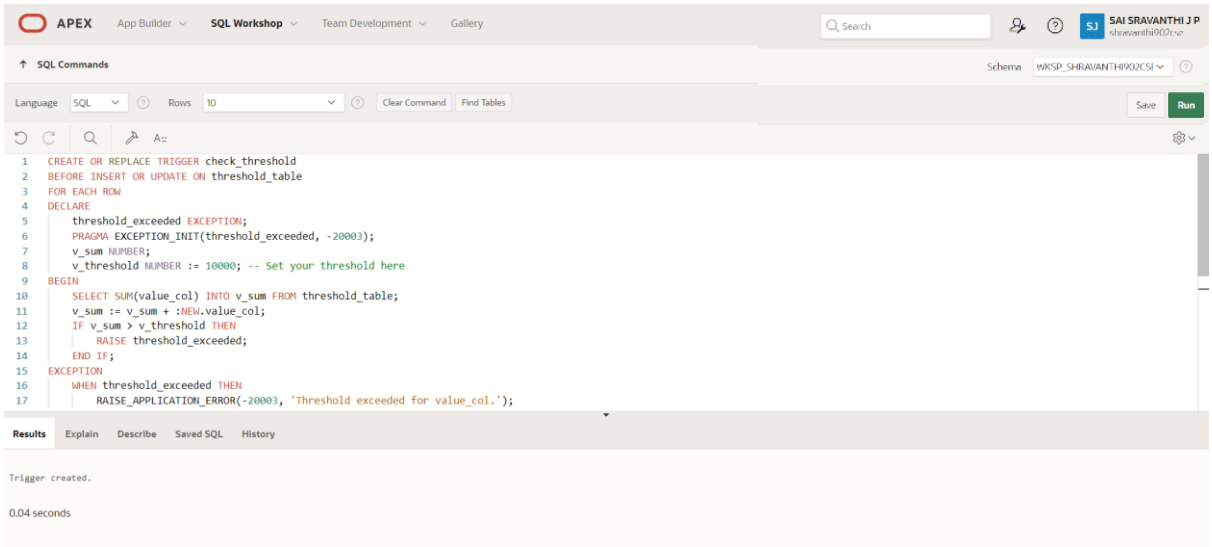


3.) Write a code in PL/SQL to create a trigger that restricts the insertion of new rows if the total of a column's values exceeds a certain threshold

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:

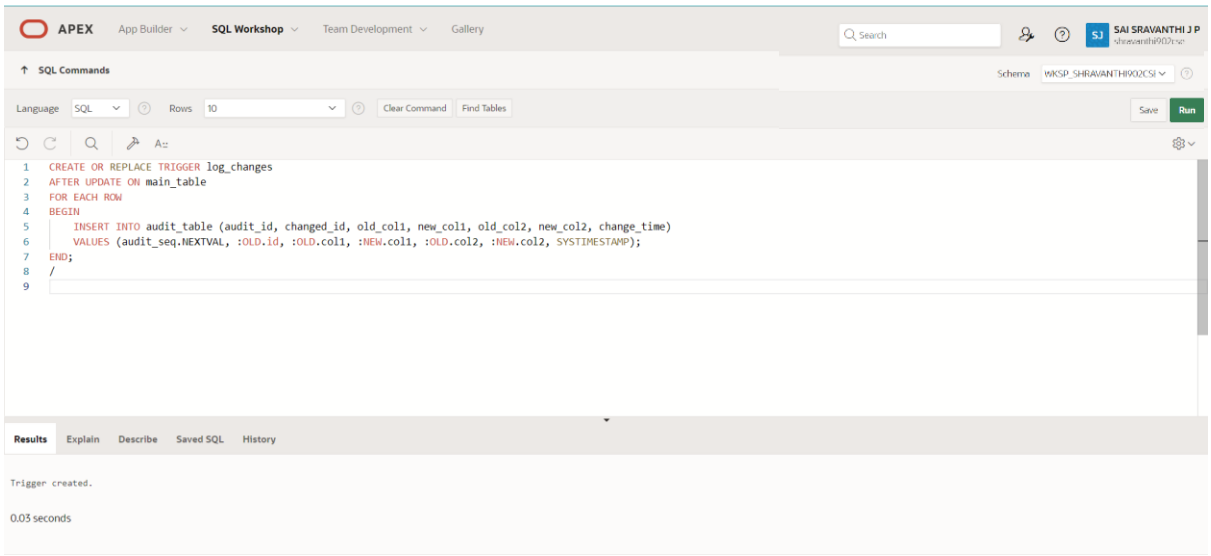


4.) Write a code in PL/SQL to design a trigger that captures changes made to specific columns and logs them in an audit table.

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:

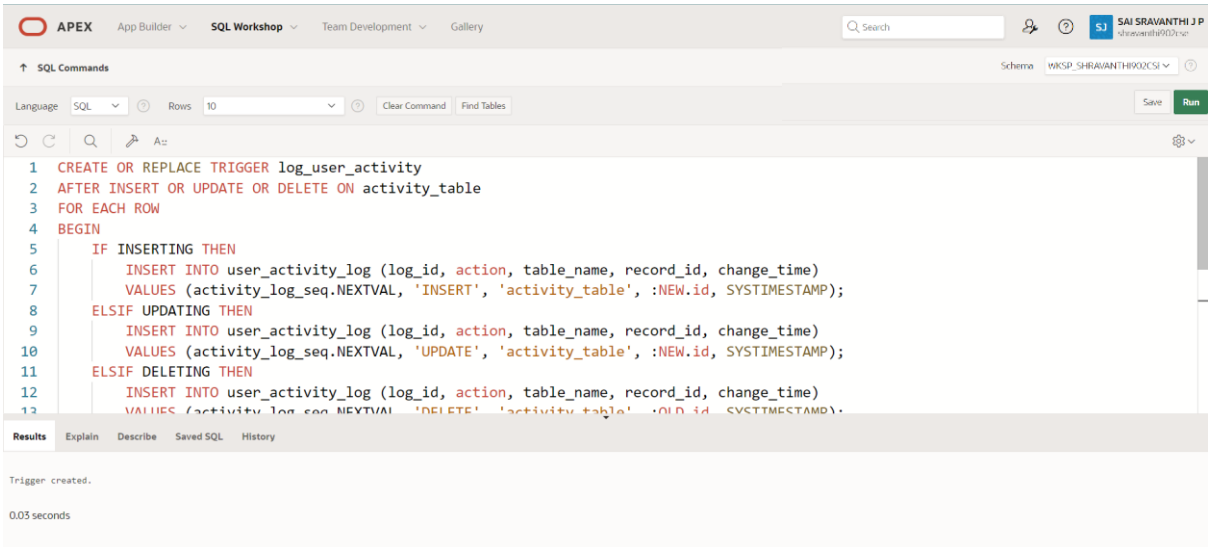


5.) Write a code in PL/SQL to implement a trigger that records user activity (inserts, updates, deletes) in an audit log for a given set of tables.

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:

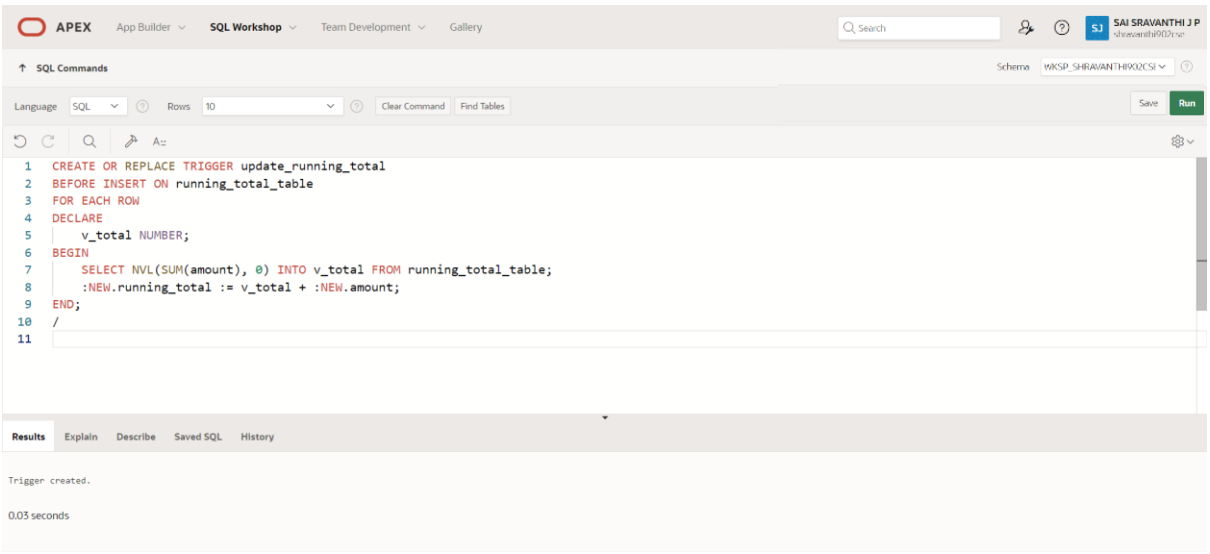


6.) Write a code in PL/SQL to implement a trigger that automatically calculates and updates a running total column for a table whenever new rows are inserted

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:

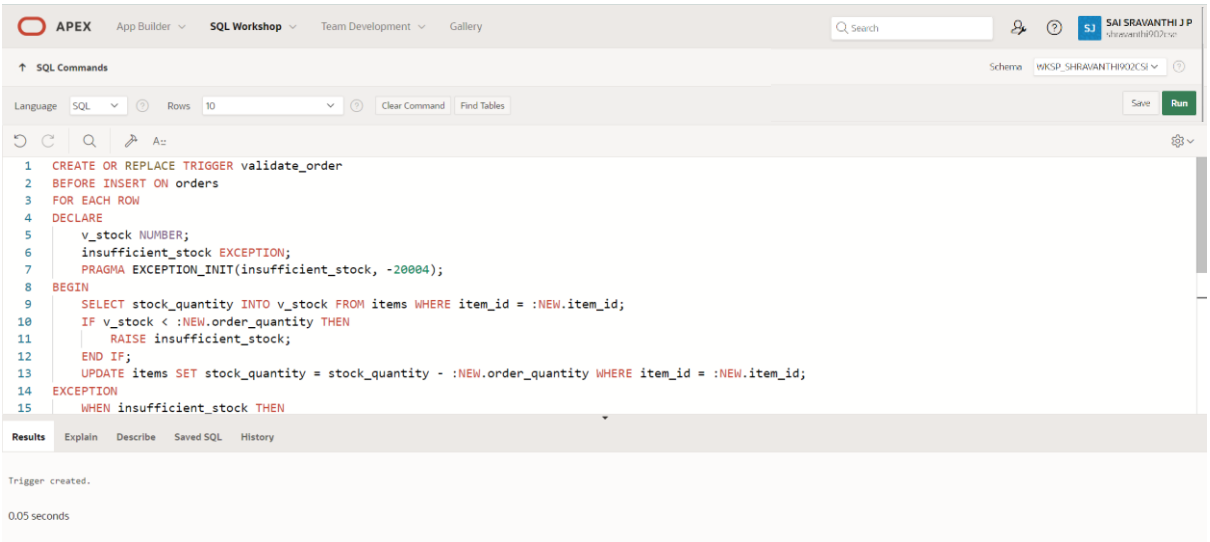


7.) Write a code in PL/SQL to create a trigger that validates the availability of items before allowing an order to be placed, considering stock levels and pending orders

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

EX_NO: 19 **DATE:**

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

QUERY:

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

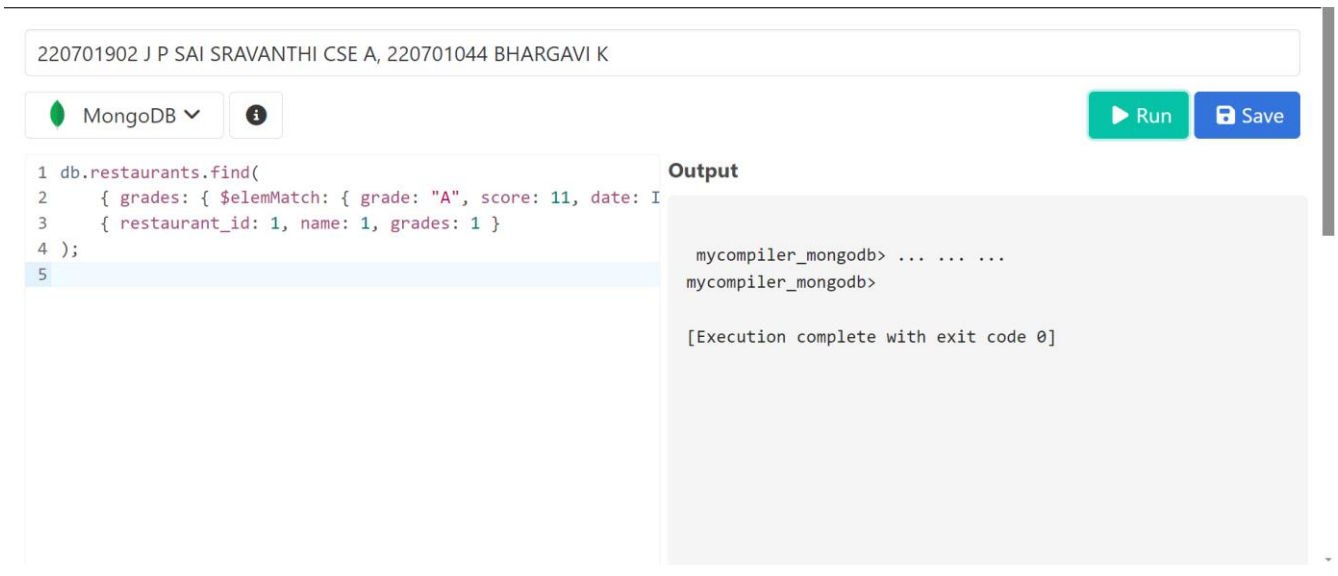
OUTPUT:

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

QUERY:

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

OUTPUT:



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

QUERY:

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

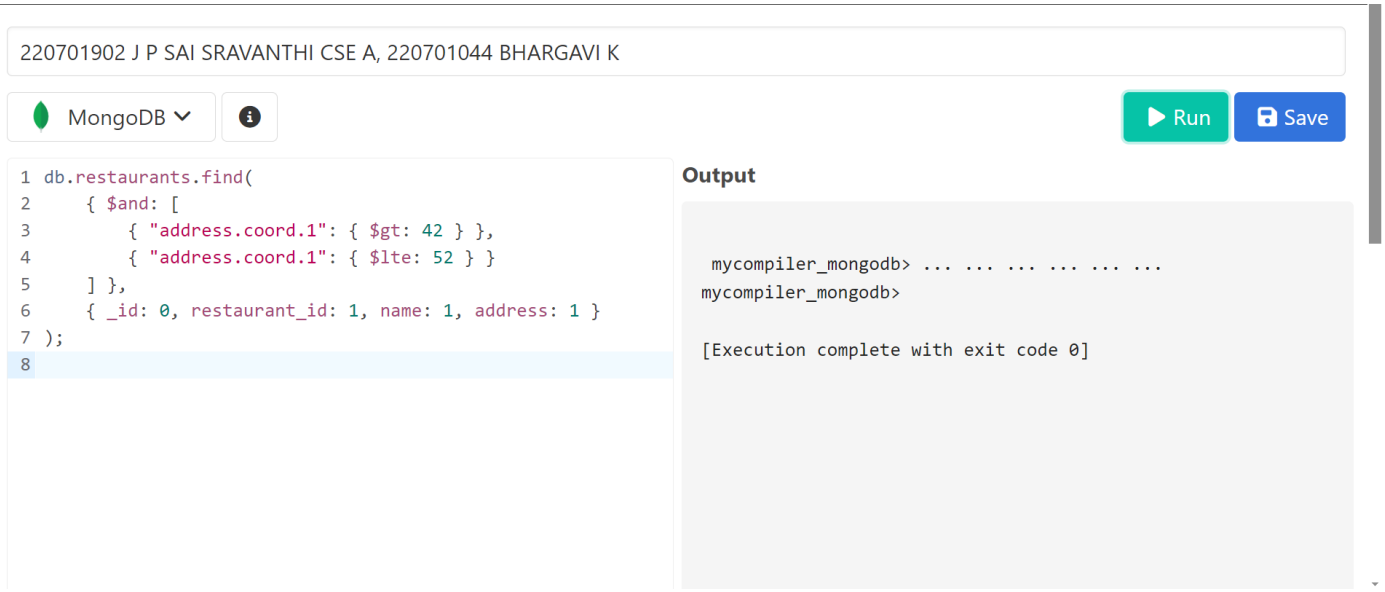
OUTPUT:

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" } } });
```

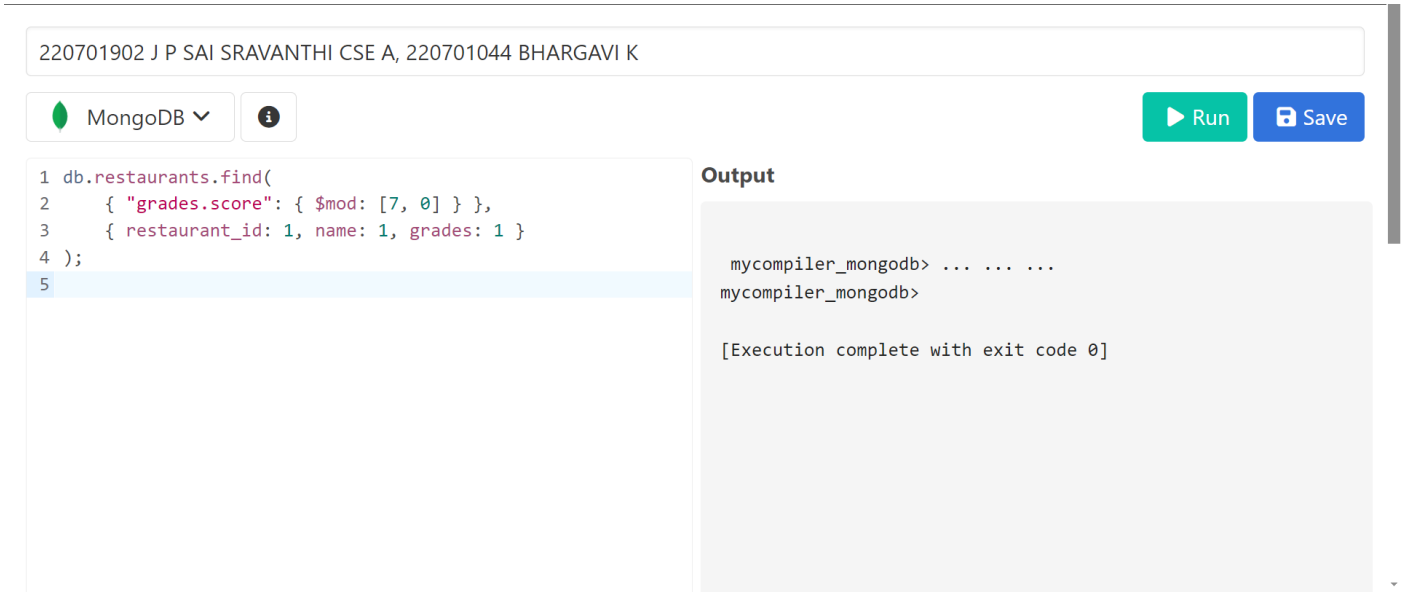
OUTPUT:

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 });
```

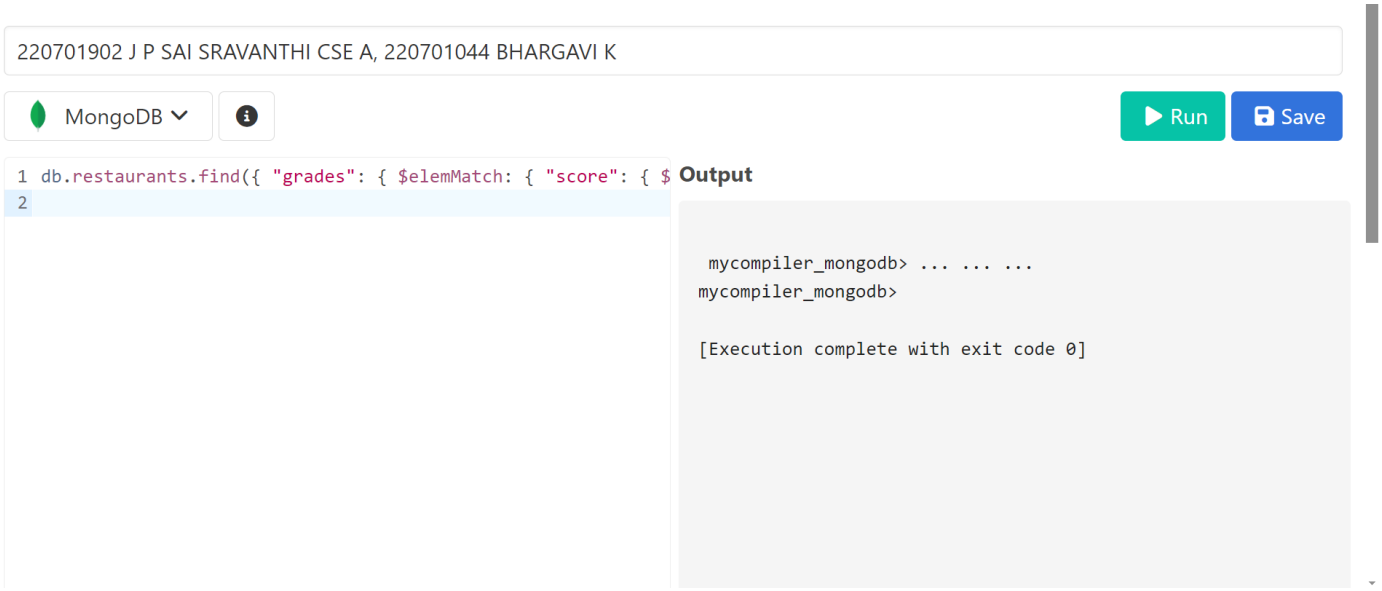
OUTPUT:

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" }] });
```

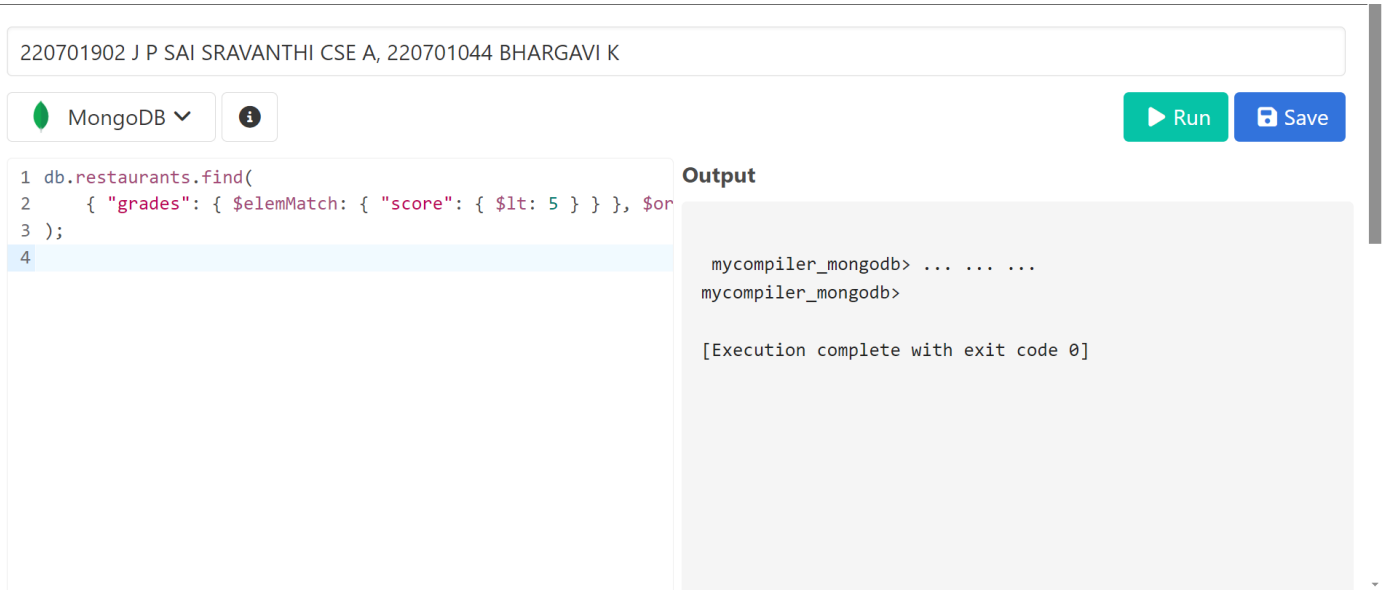
OUTPUT:

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:

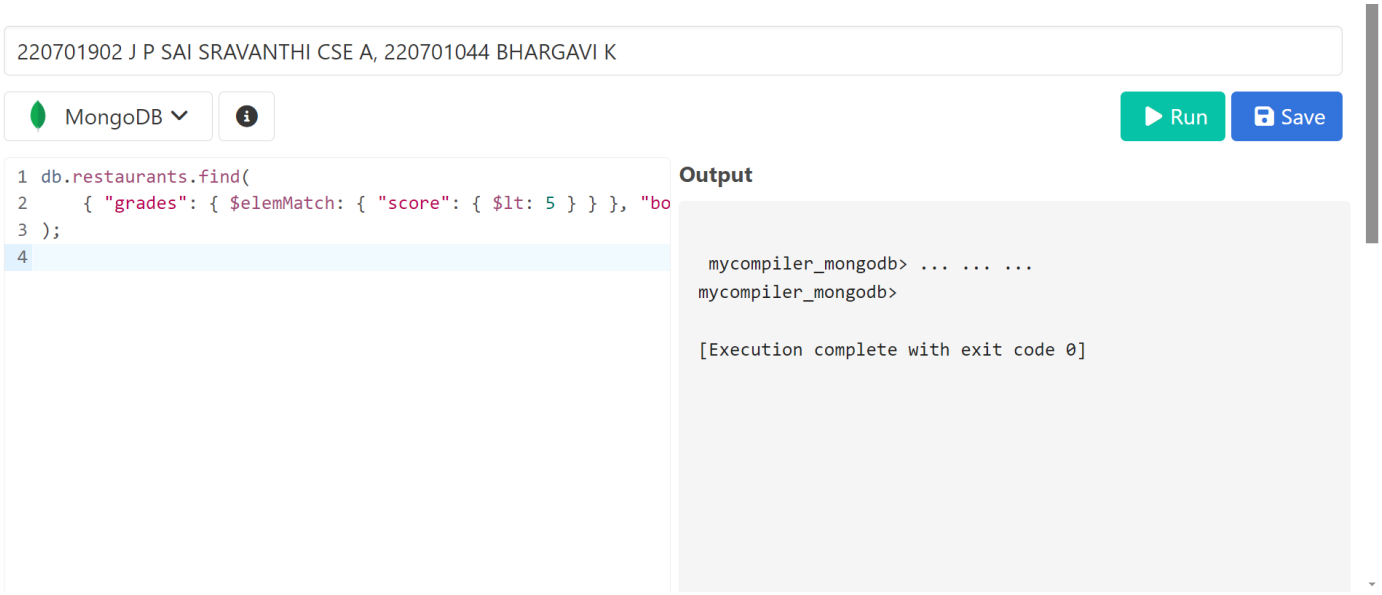


17. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn, and their cuisine is not American or Chinese.

QUERY:

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

OUTPUT:

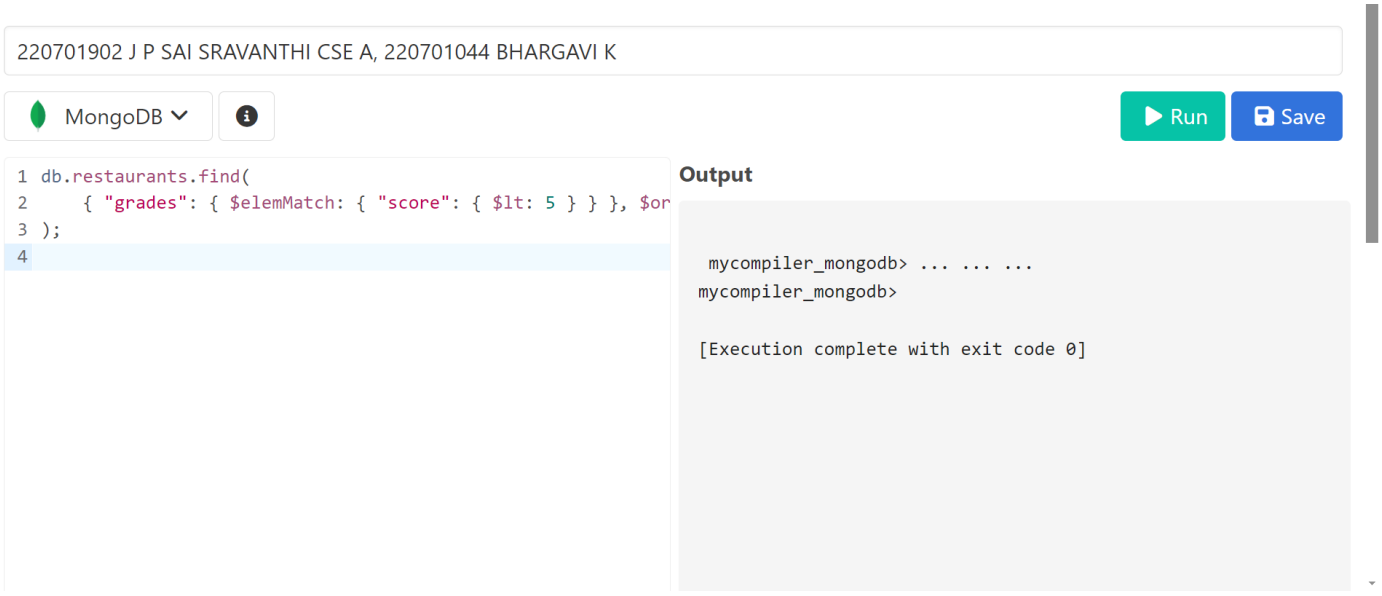


18. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6.

QUERY:

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

OUTPUT:

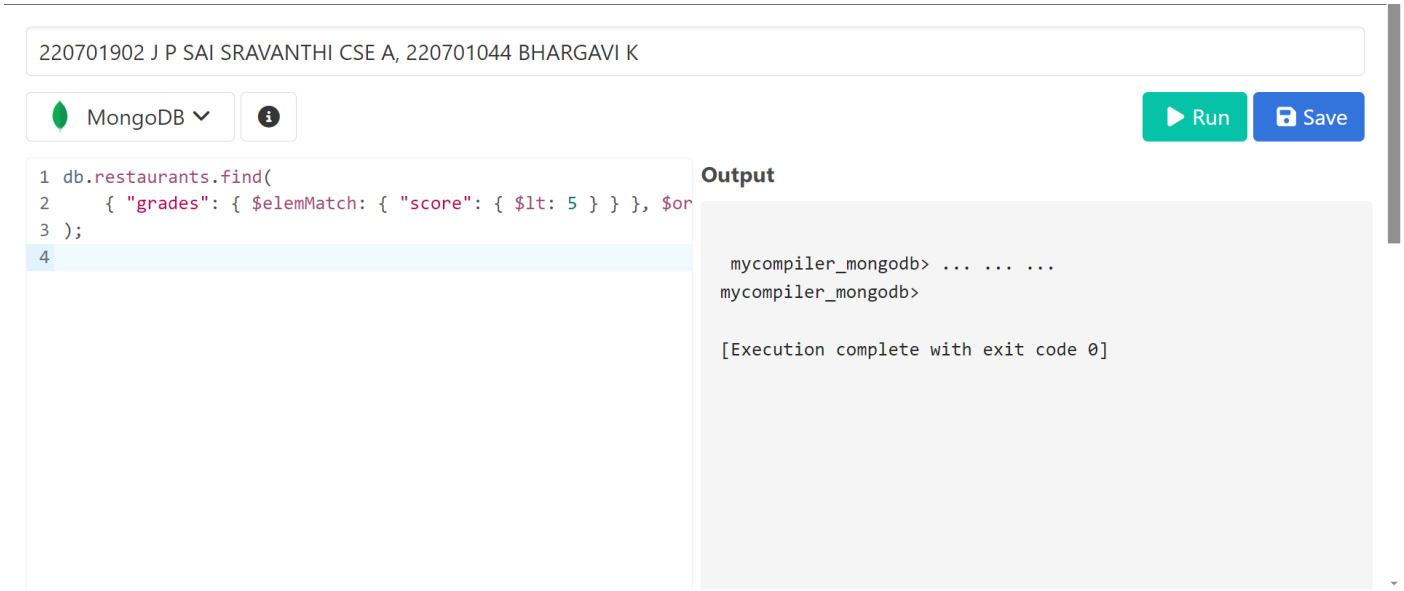


19. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan.

QUERY:

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

OUTPUT:

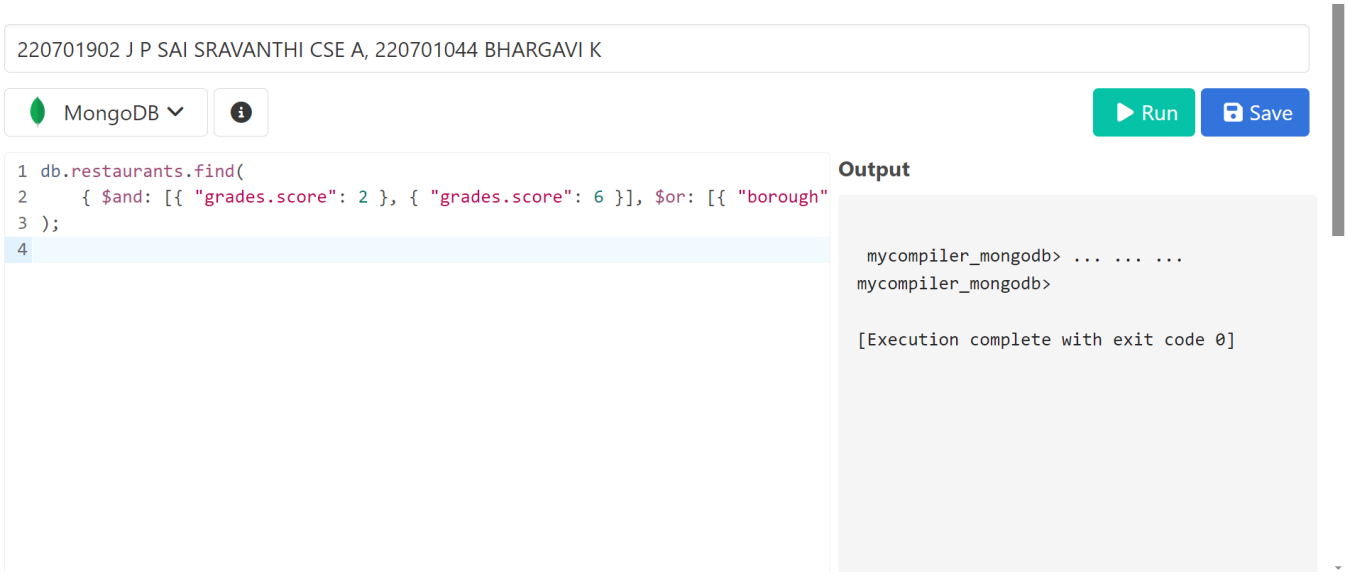


20. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn.

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" } });
```

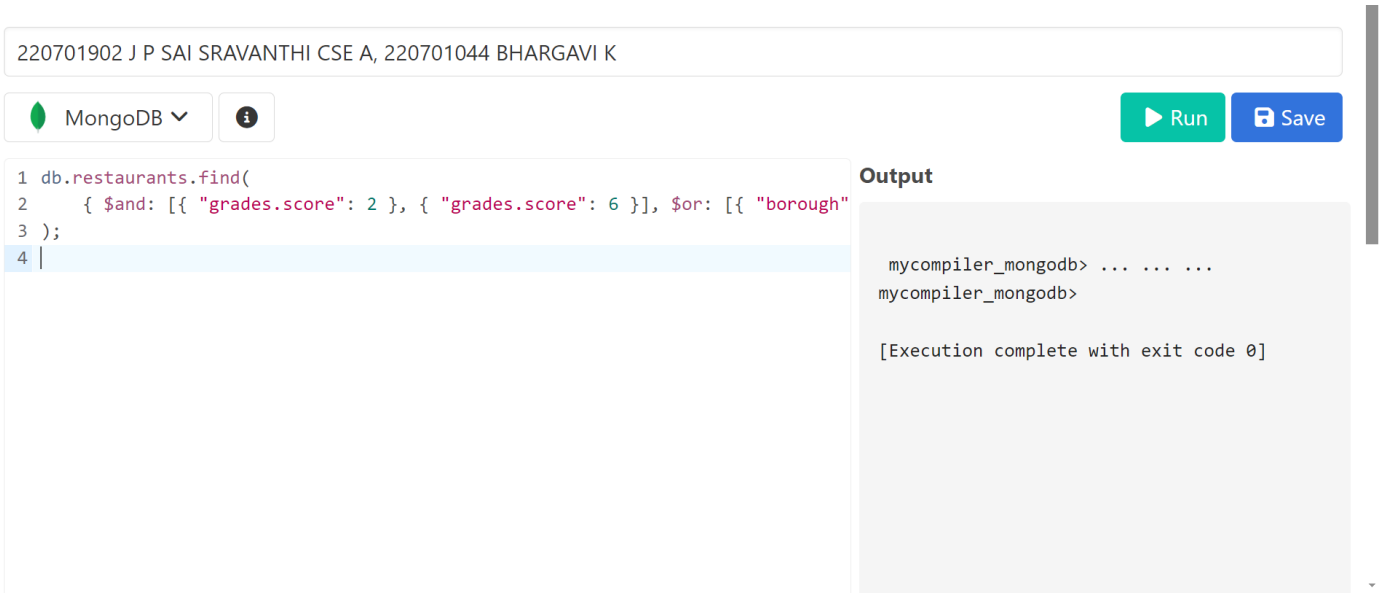
OUTPUT:

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

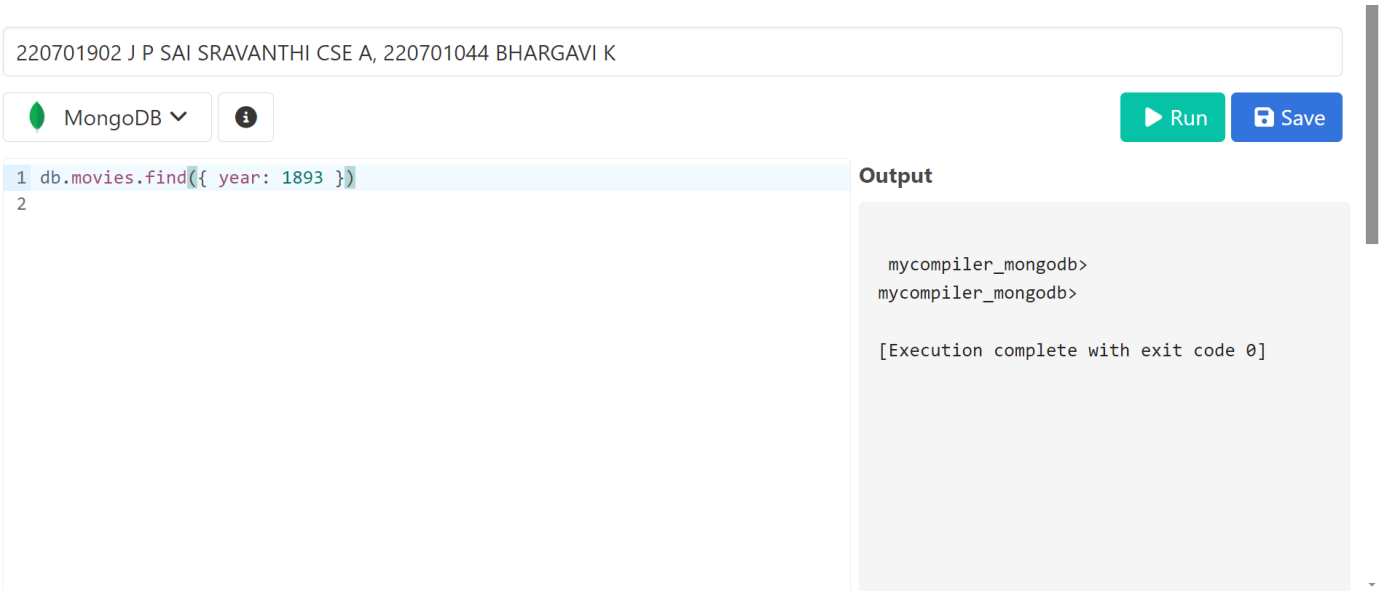
EX_NO: 20 **DATE:**

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

QUERY:

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

OUTPUT:



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

QUERY:

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

OUTPUT:

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

QUERY:

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

OUTPUT:

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' });
```

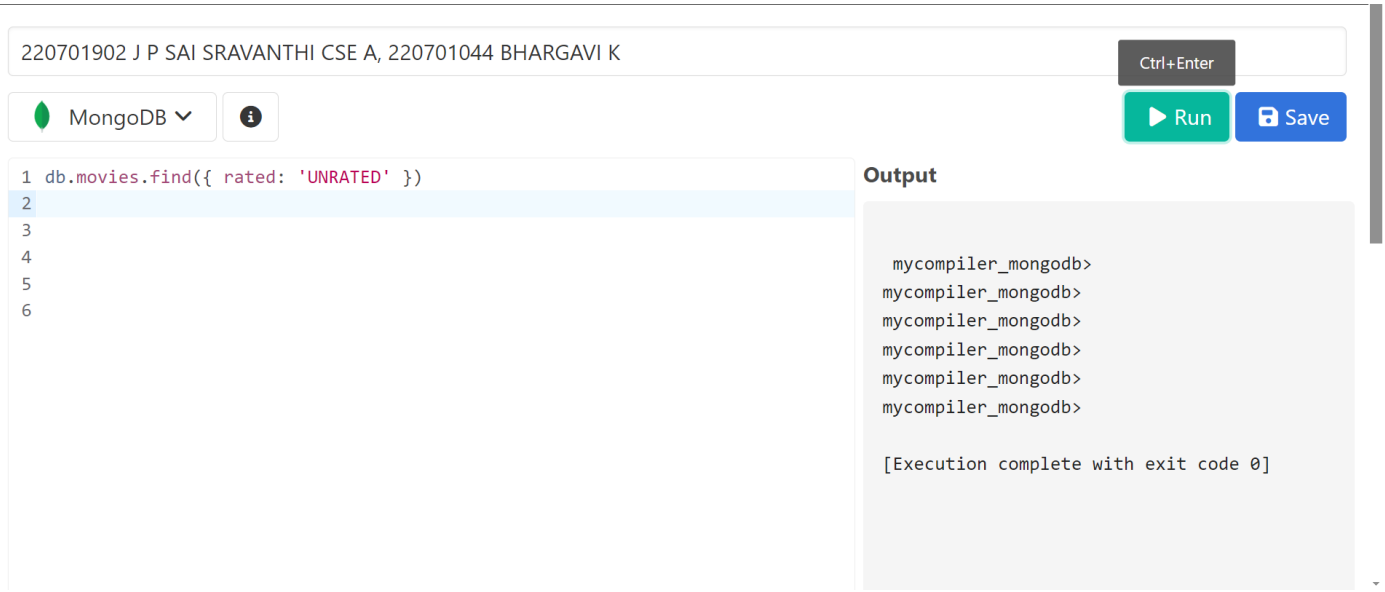
OUTPUT:

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:

220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K

MongoDB | Run | Save

```
1  db.movies.find(
2      { 'awards.nominations': { $gt: 0 } },
3      {
4          title: 1,
5          languages: 1,
6          released: 1,
7          directors: 1,
8          writers: 1,
9          awards: 1,
10         year: 1,
11         genres: 1,
12         runtime: 1,
13         cast: 1,
14         countries: 1
15     }
16 )
17
```

Output

```
 mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

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

QUERY:

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

OUTPUT:

220701902 J P SAI SRAVANTHI CSE A, 220701044 BHARGAVI K

MongoDB | Run | Save

```
1  db.movies.find(
2      { cast: 'Charles Kayser' },
3      {
4          title: 1,
5          languages: 1,
6          released: 1,
7          directors: 1,
8          writers: 1,
9          awards: 1,
10         year: 1,
11         genres: 1,
12         runtime: 1,
13         cast: 1,
14         countries: 1
15     }
16 )
17
```

Output

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

[Execution complete with exit code 0]
```

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

QUERY:

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

OUTPUT:

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: