Creating and Managing Tables

EX_NO:1 **DATE**:

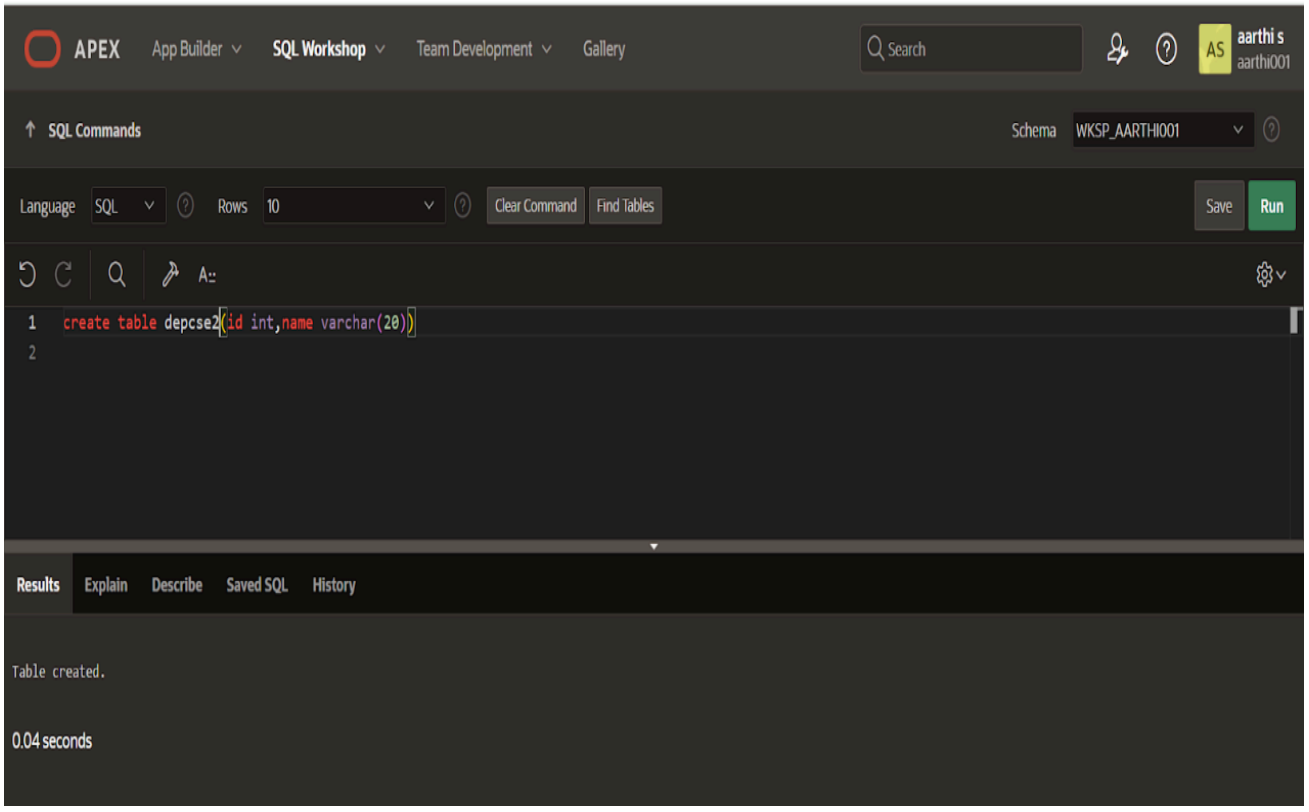
1. Create the DEPT table based on the DEPARTMENT following the table instance chart below. Confirm that the table is created.

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

QUERY:

Create table depcse(id int,name varchar(20));

OUTPUT:



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

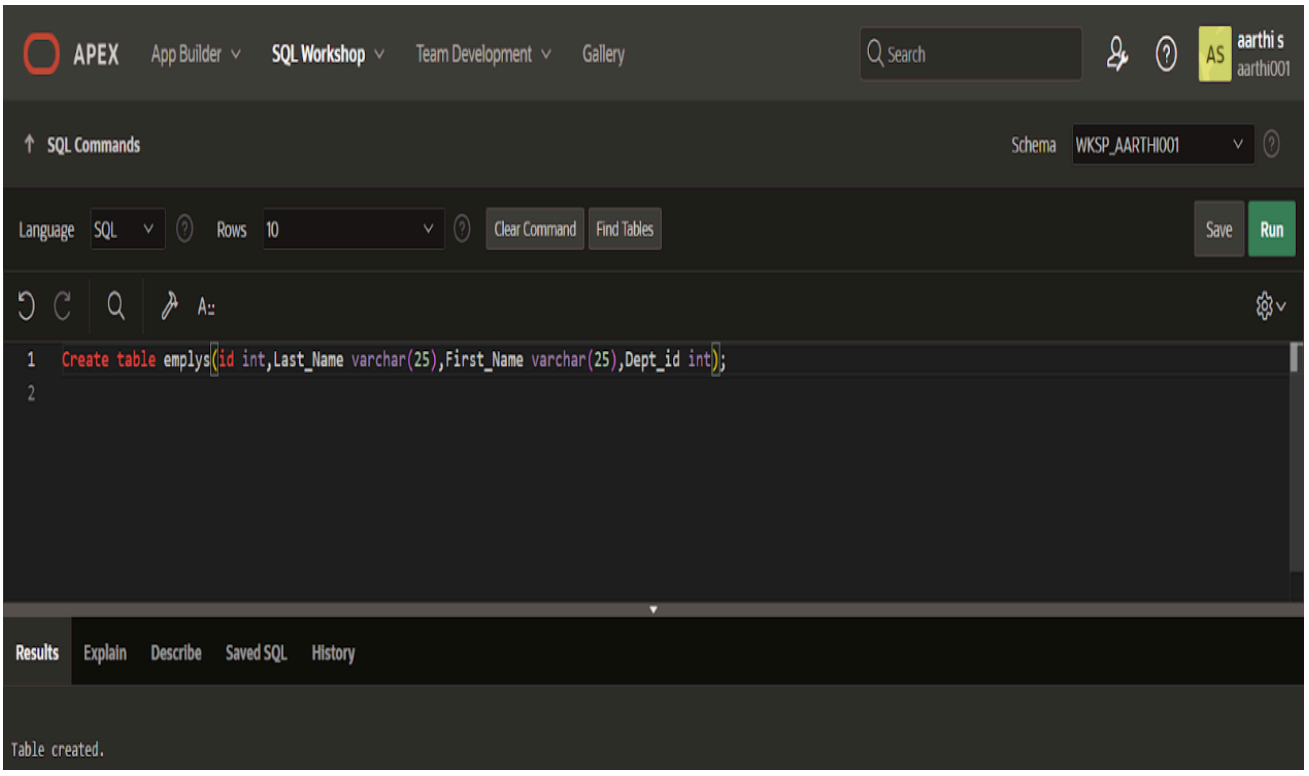
Column name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				
Nulls/Unique				
FK table				
FK column				

Data Type	Number	Varchar2	Varchar2	Number
Length	7	25	25	7

QUERY:

Create table emply(id int,Last_Name varchar(25),First_Name varchar(25),Dept_id int);

OUTPUT:

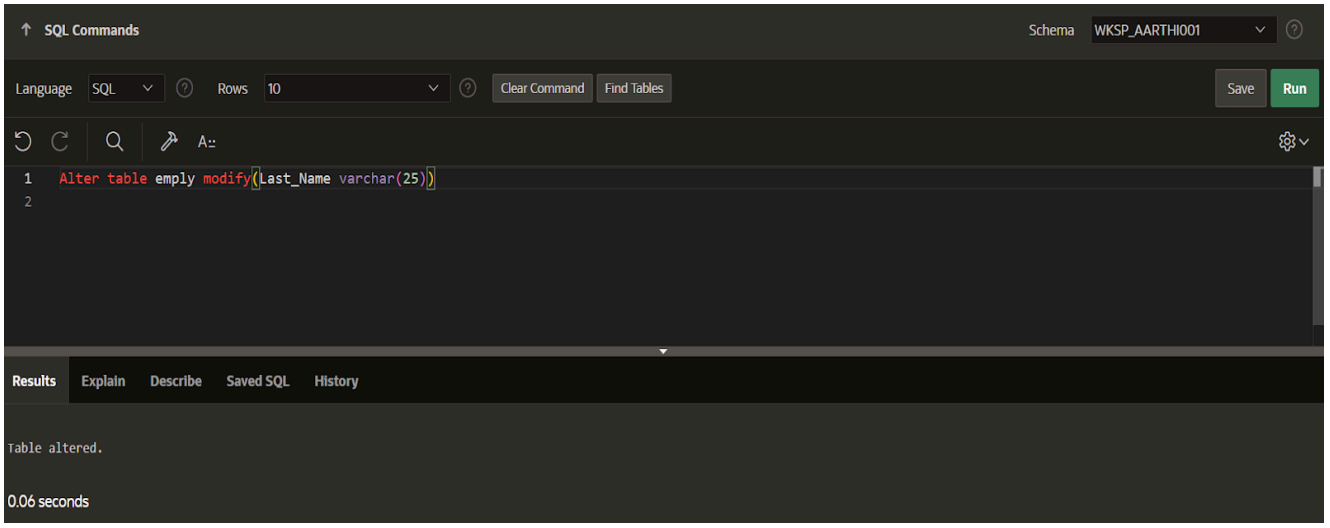


3. Modify the EMP table to allow for longer employee last names. Confirm the modification.(Hint: Increase the size to 50)

QUERY:

Alter table emply modify(Last_Name varchar(25))

OUTPUT:

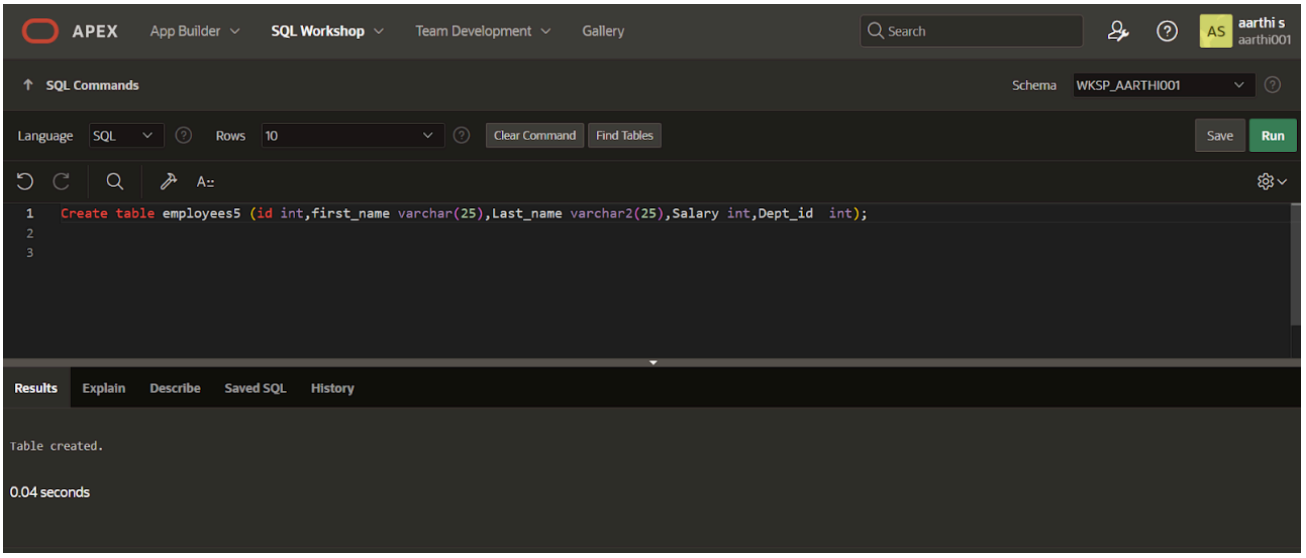


4. Create the EMPLOYEES2 table based on the structure of EMPLOYEES table. Include Only the Employee_id, First_name, Last_name, Salary and Dept_id coloumns. Name the columns Id, First_name, Last_name, salary and Dept_id respectively.

QUERY:

Create table employees3 (id int,first_name varchar(25),Last_name varchar2(25),Salary int,Dept_id int);

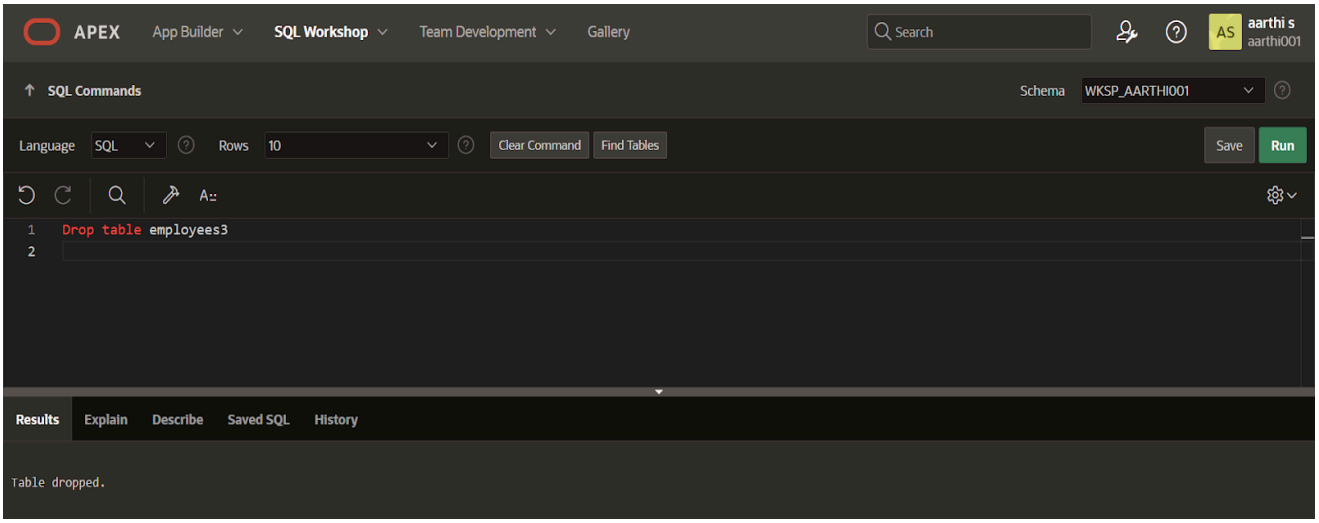
OUTPUT:



5. Drop the EMP table.

QUERY:
Drop table employees3;

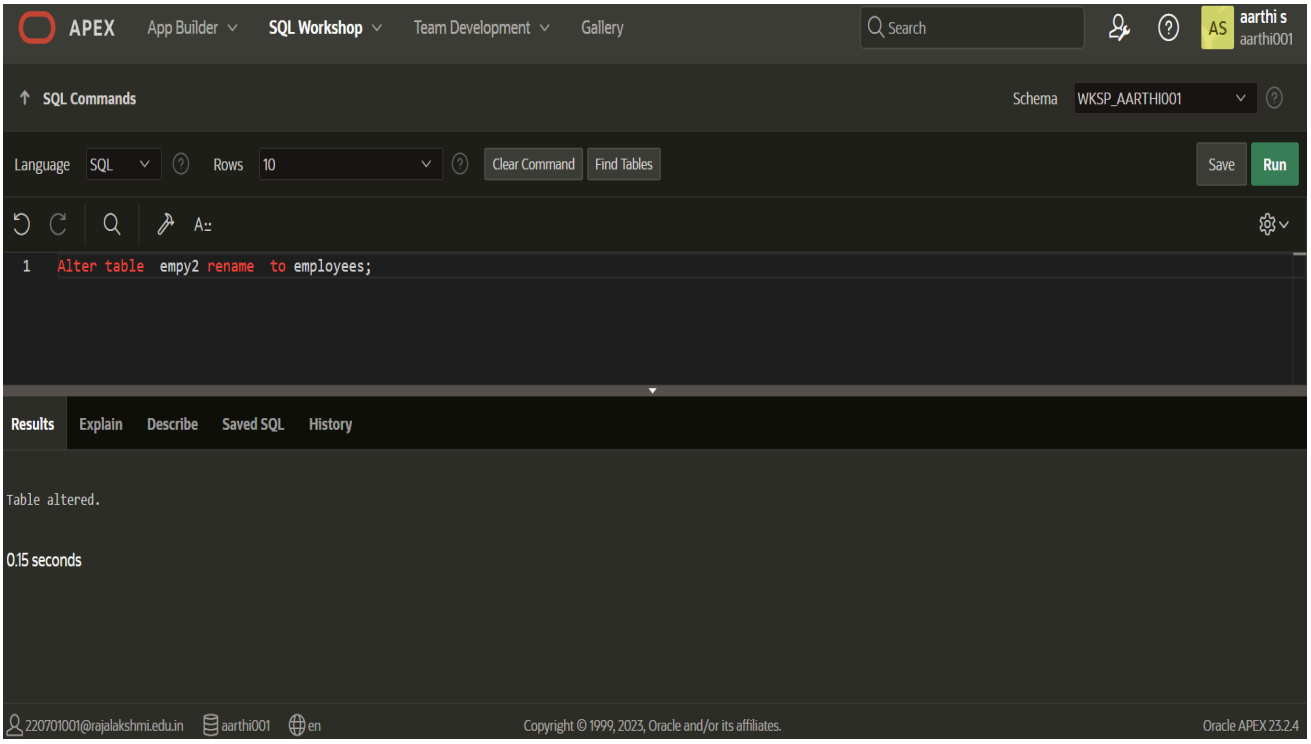
OUTPUT:



6. Rename the EMPLOYEES2 table as EMP.

QUERY:
Alter table emply2 rename to em2;

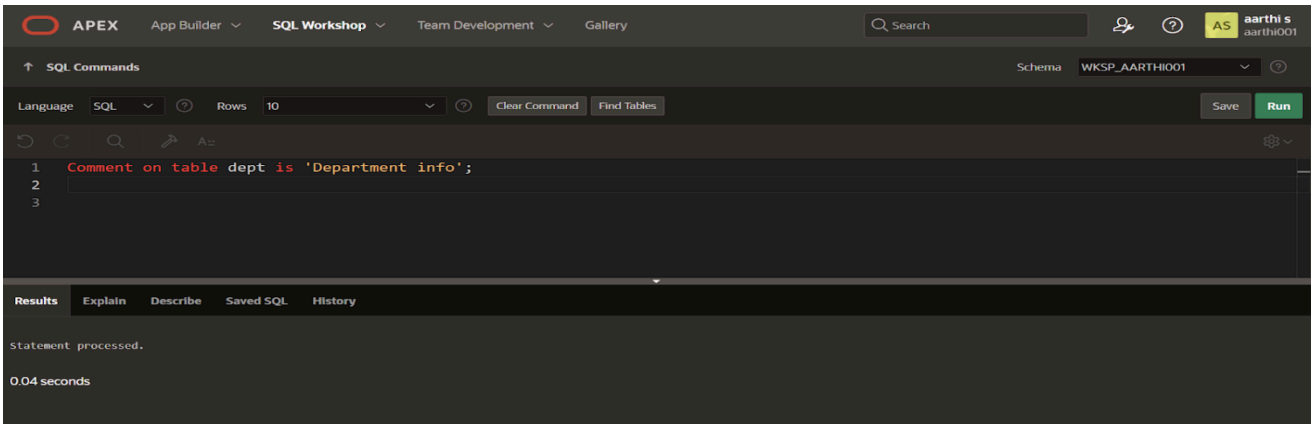
OUTPUT:



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';

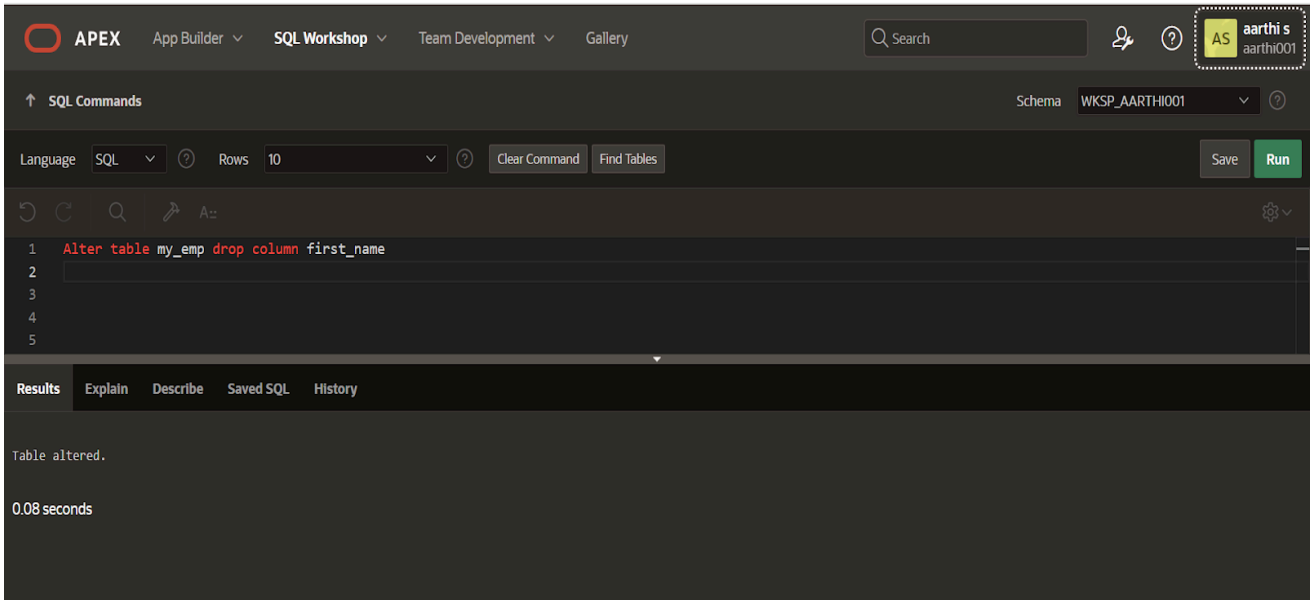
OUTPUT:



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

QUERY:
Alter table my_emp drop column first_name

Output:



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

MANIPULATING DATA

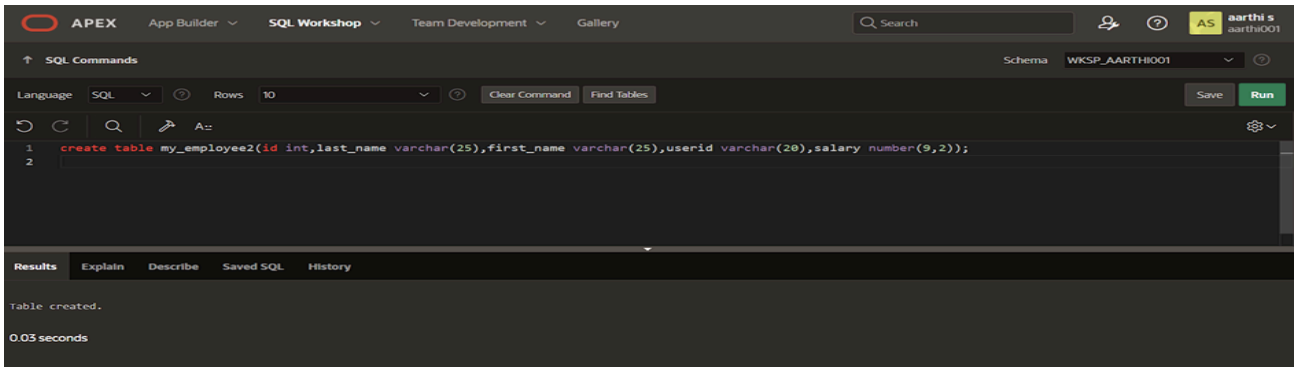
EX_NO:2 **DATE:**

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 my_employee2(id int,last_name varchar(25),first_name varchar(25),userid varchar(20),salary number(9,2));

OUTPUT:

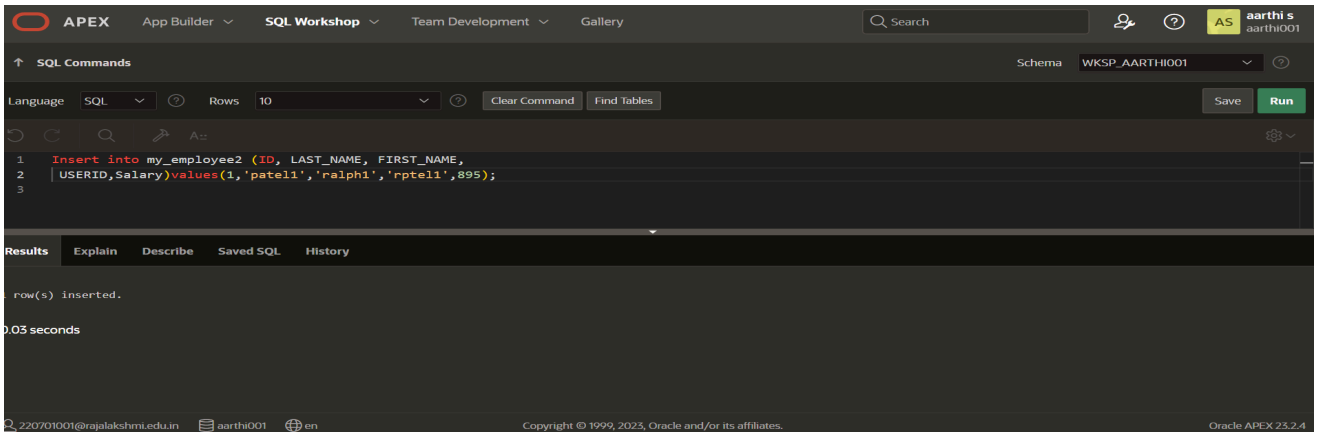


2. Add the first and second rows data to MY_EMPLOYEE table from the following sample data.

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

QUERY:
Insert into my_employee2 values(1,'patel','ralph','rptel',895),
(2,'dancs','betty','bdancs',860);

OUTPUT:

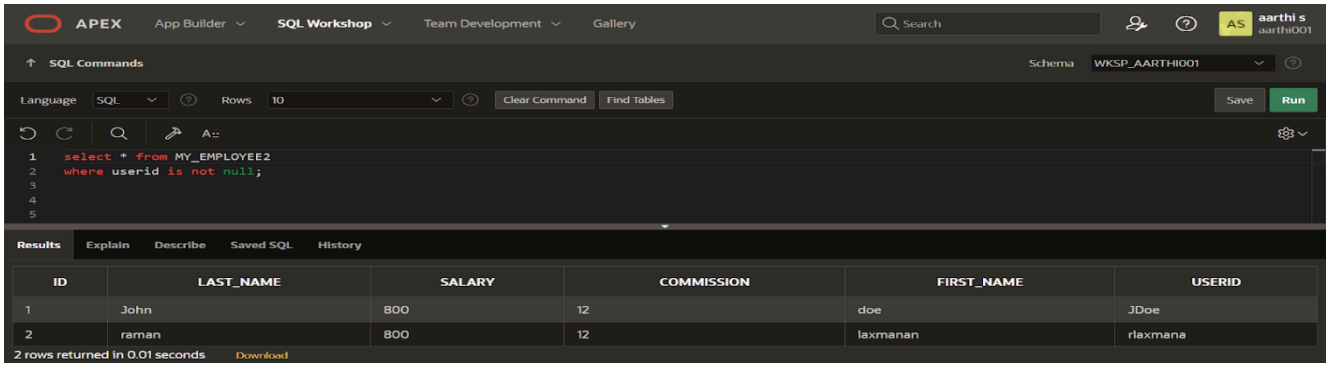


3. Populate the next two rows of data from the sample data. Concatenate the first letter of the first_name with the first seven characters of the last_name to produce Userid.

QUERY:

INSERT INTO MY_EMPLOYEE2
(ID,LAST_NAME,USERID,SALARY,COMMISSION,FIRST_NAME) VALUES (002,'raman',
SUBSTR('raman', 1, 1) || SUBSTR('laxmanan', 1, 7),800,12, 'laxmanan');

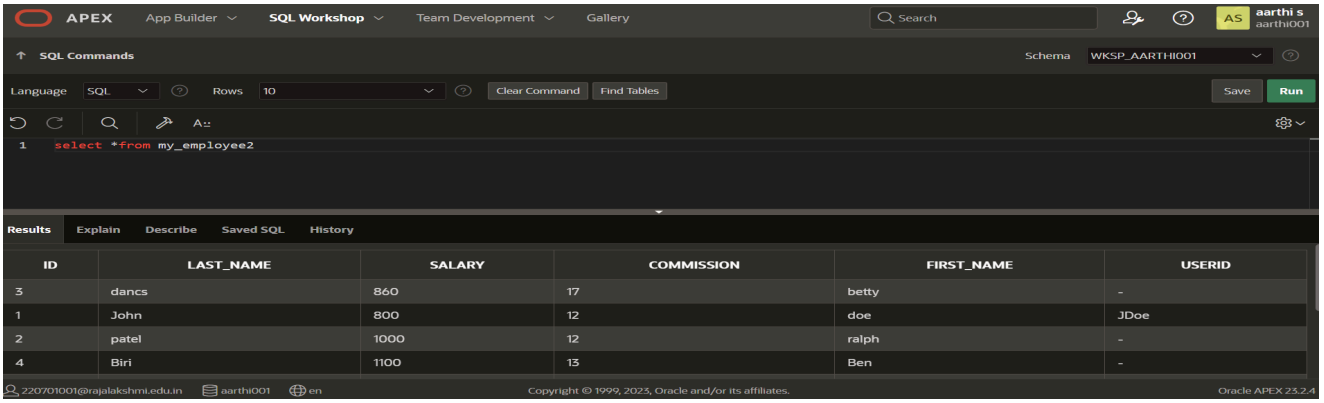
OUTPUT:



4. Display the table with values.

QUERY:
select *from my_employee2;

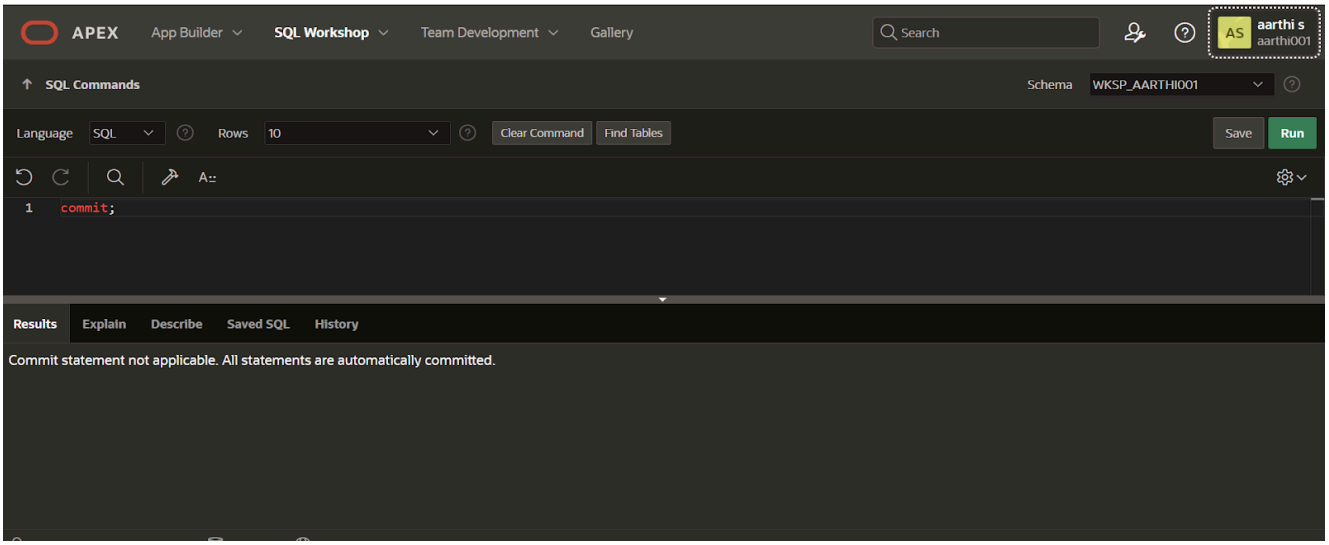
OUTPUT:



5. Make the data additions permanent.

QUERY:
commit;

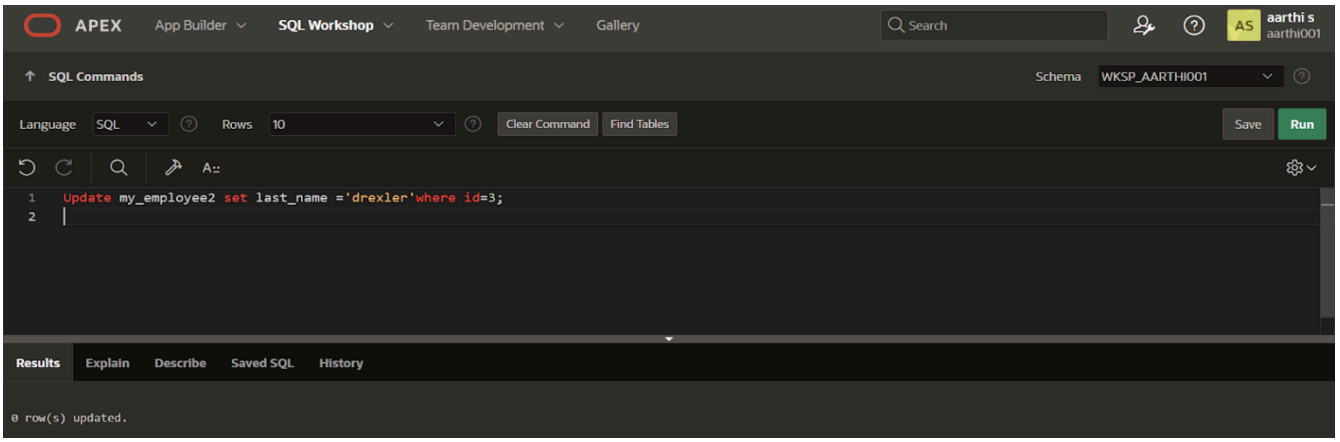
OUTPUT:



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

QUERY:
Update my_employee2 set last_name ='drexler' where id=3;

OUTPUT:

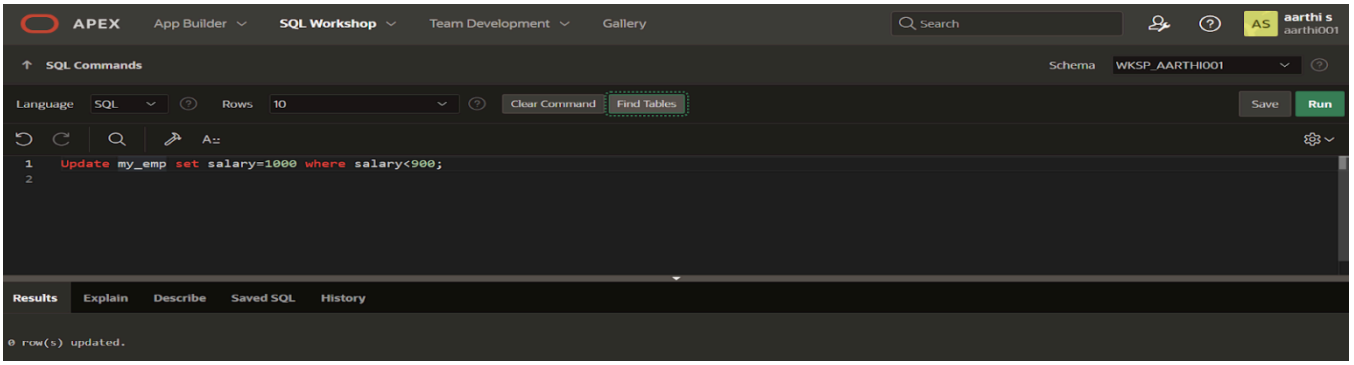


7. Change the salary to 1000 for all the employees with a salary less than 900.

QUERY:

Update my_emp set salary=1000 where salary<900;

OUTPUT:

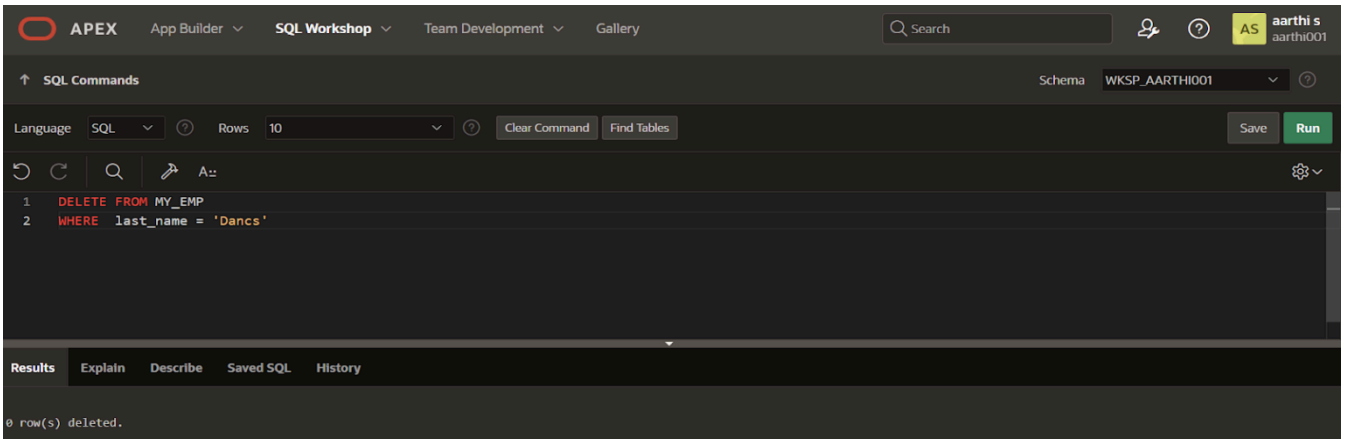


8. Delete Betty dancs from MY _EMPLOYEE table.

QUERY:

Delete from my emp where first name=’chad’;

OUTPUT:



Evaluation Procedure	Marks awarded
Query (5)	

INCLUDING CONSTRAINTS

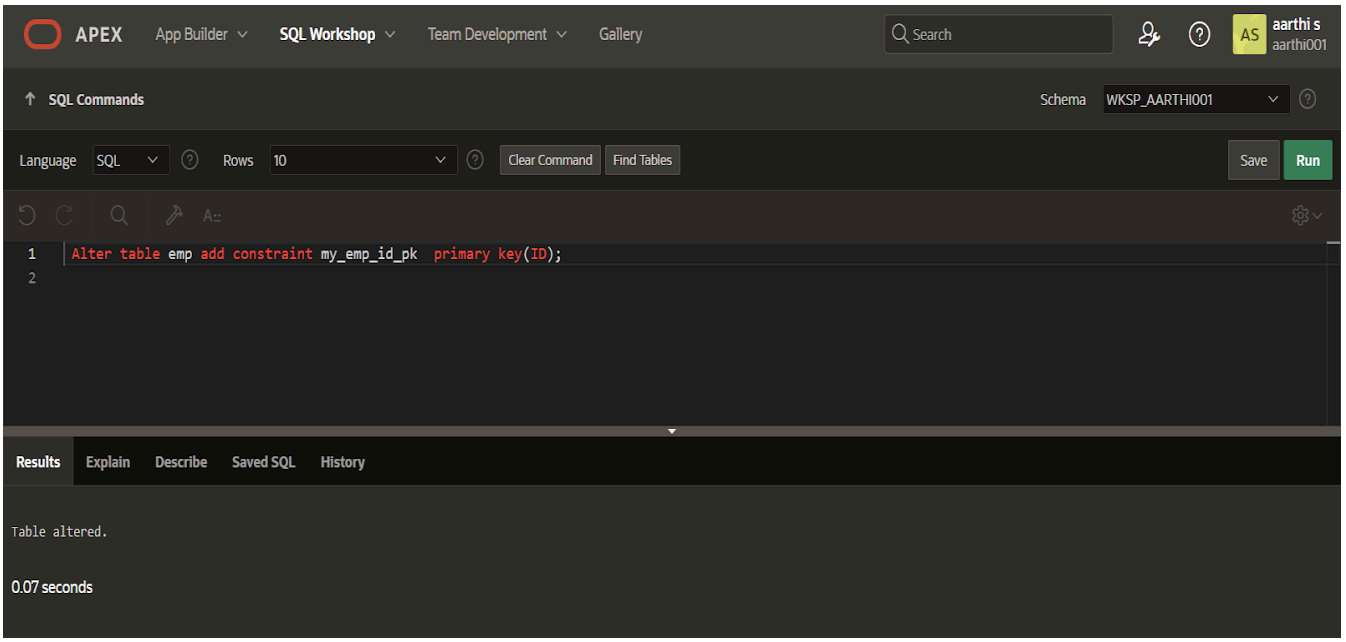
EX_NO:3 **DATE:**

---

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(ID);

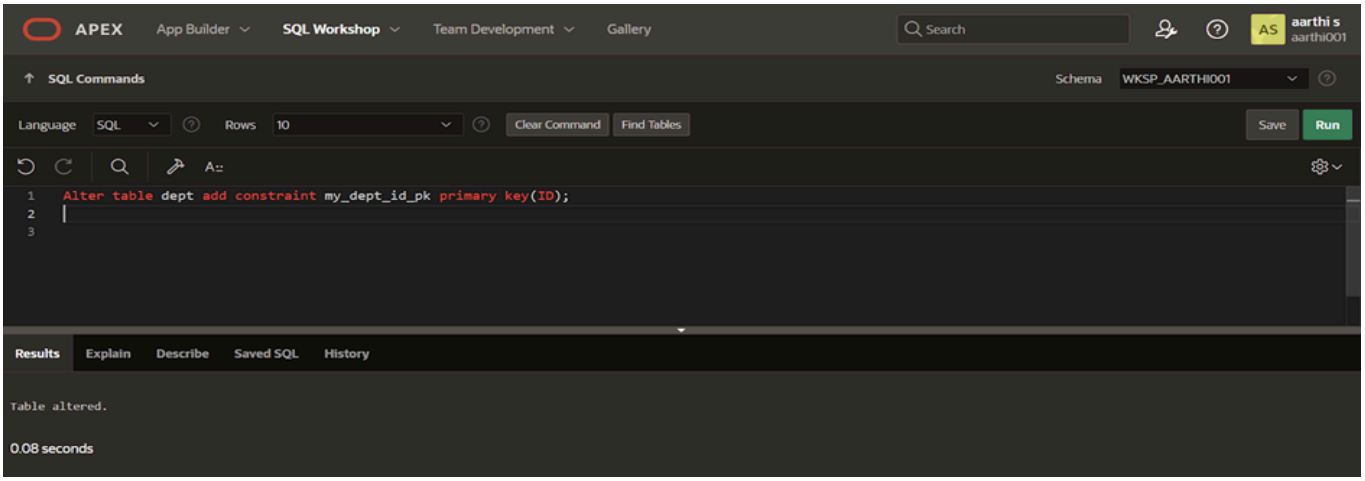
   OUTPUT:



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(ID);

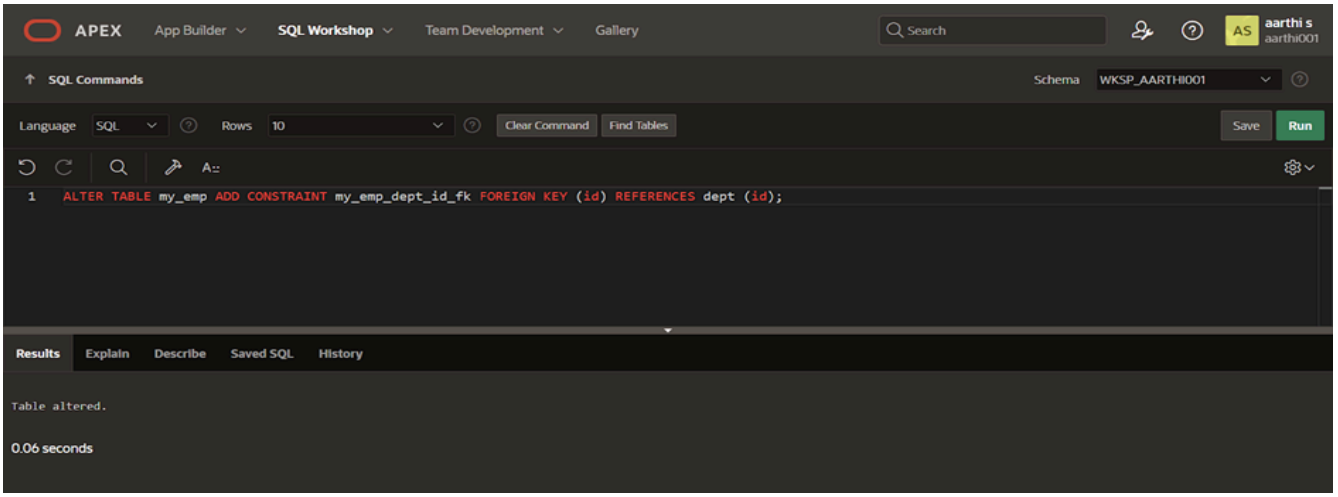
   OUTPUT:



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 non existent deparment. Name the constraint my_emp_dept_id_fk.

   QUERY:
   ALTER TABLE my_emp ADD CONSTRAINT my_emp_dept_id_fk FOREIGN KEY (id) REFERENCES dept (id);

   OUTPUT:



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

   Query:
   Alter table emp3 add commission ck check(commision>0);

   OUTPUT:


Gmail
in:sent
Active
1 alter table emp3 add(constraints ck check(comission>0))
Results
Explain
Describe
Saved SQL
History
Table altered.
0.07 seconds

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

INCLUDING CONSTRAINTS

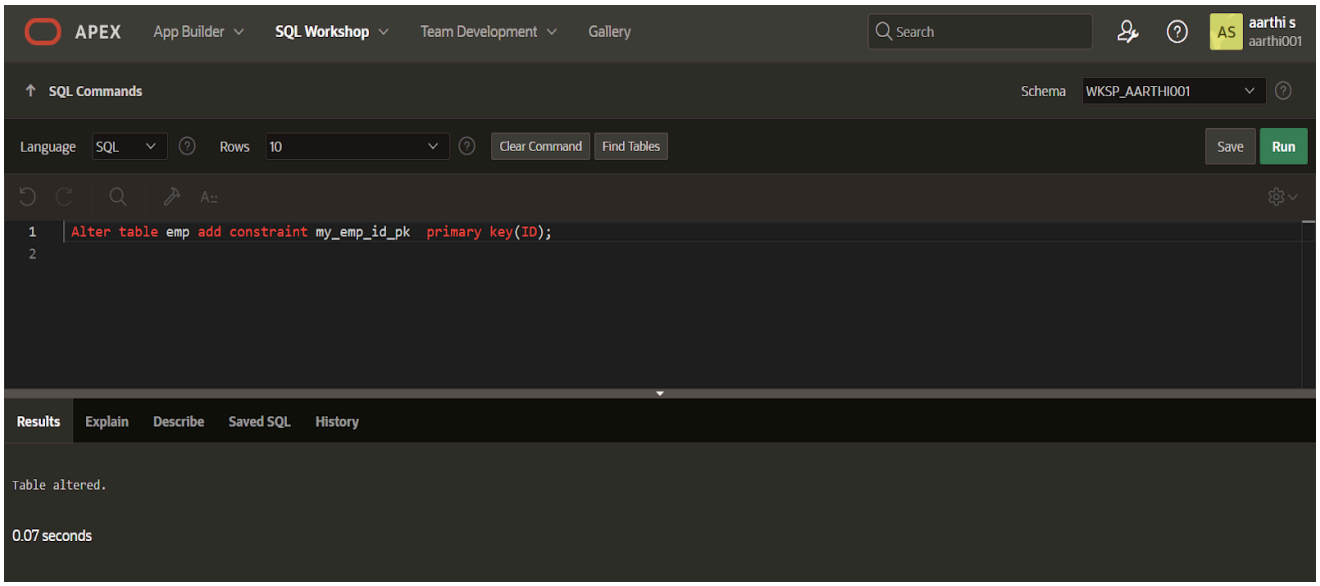
EX_NO: **DATE:**

---

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

   QUERY:
   Alter table emp add constraint my _emp_id_pk primary key(ID);

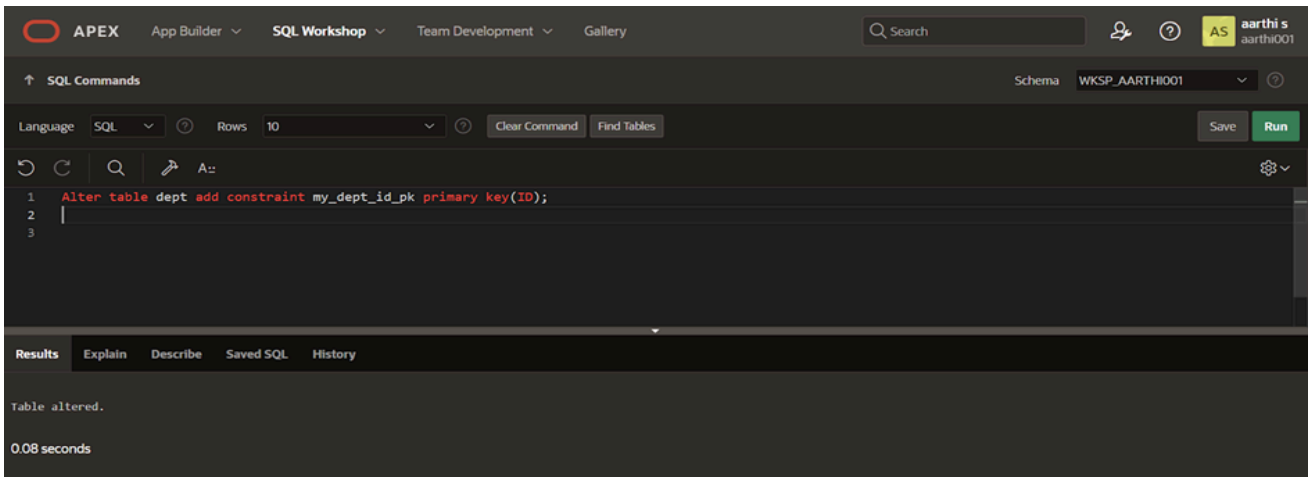
   OUTPUT:



2. Create a PRIMAY KEY constraint to the DEPT table using the ID 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(ID);

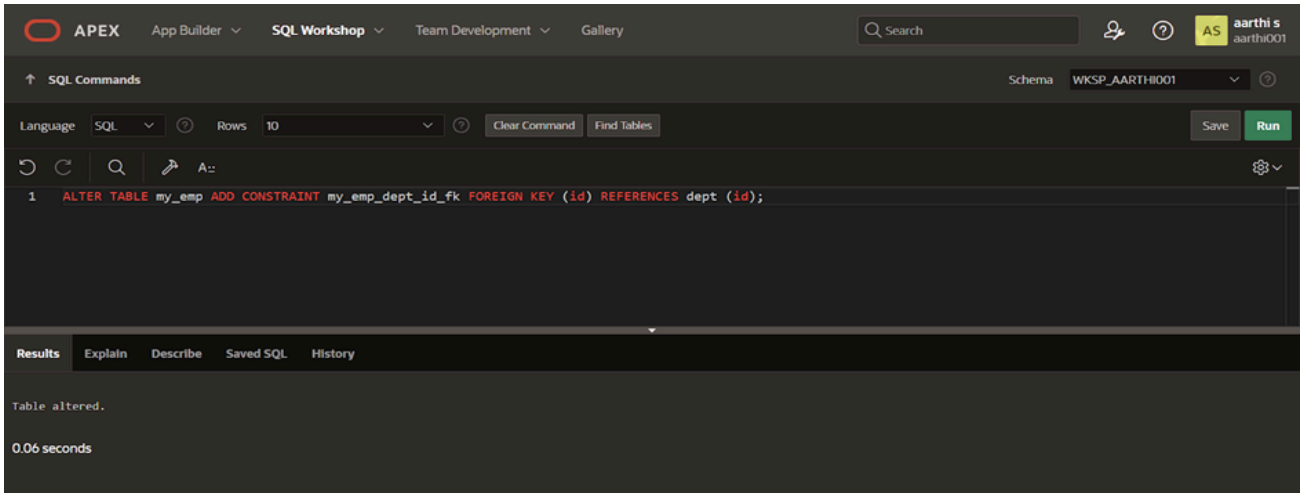
OUTPUT:



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 non existent deparment. Name the constraint my_emp_dept_id_fk.

   QUERY:
   ALTER TABLE my_emp ADD CONSTRAINT my_emp_dept_id_fk FOREIGN KEY (id) REFERENCES dept (id);

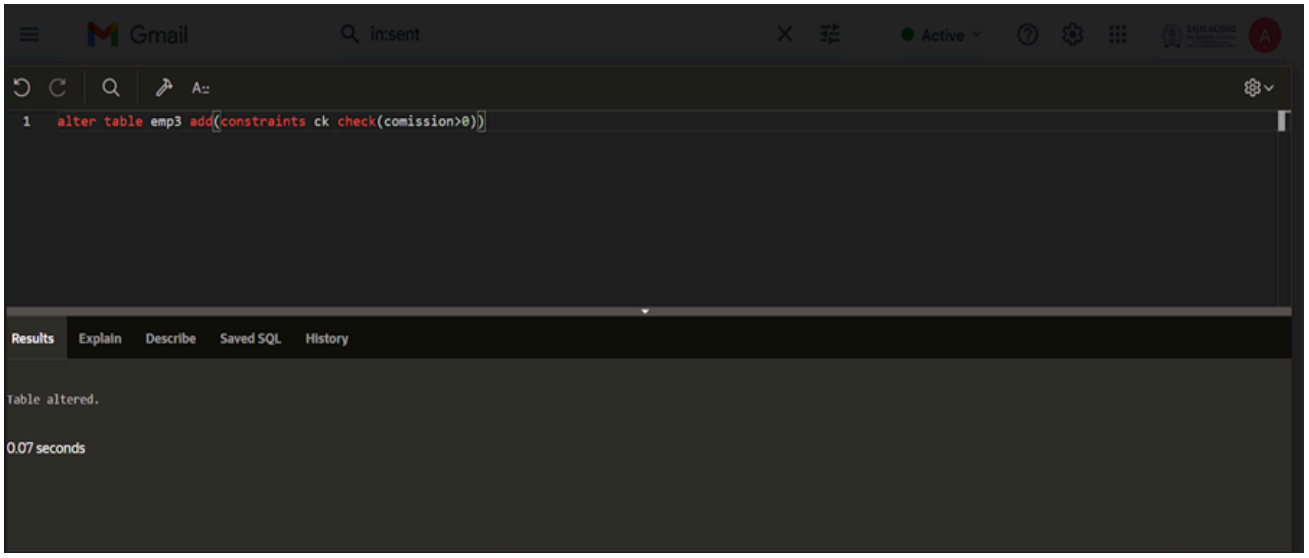
   OUTPUT:



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

   QUERY:
   Alter table emp3 add commission ck check(commision>0);

   OUTPUT:



Restricting and Sorting data

EX_NO: 5 **DATE:**

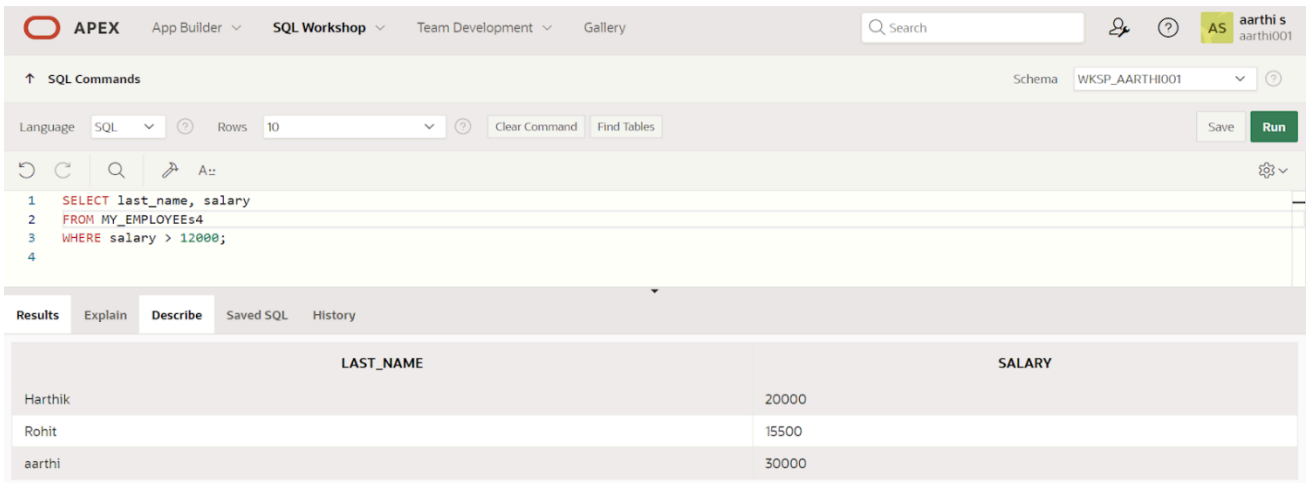
---

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

   QUERY:

   SELECT last_name, salary FROM MY_EMPLOYEES4 WHERE salary > 12000;

   OUTPUT:

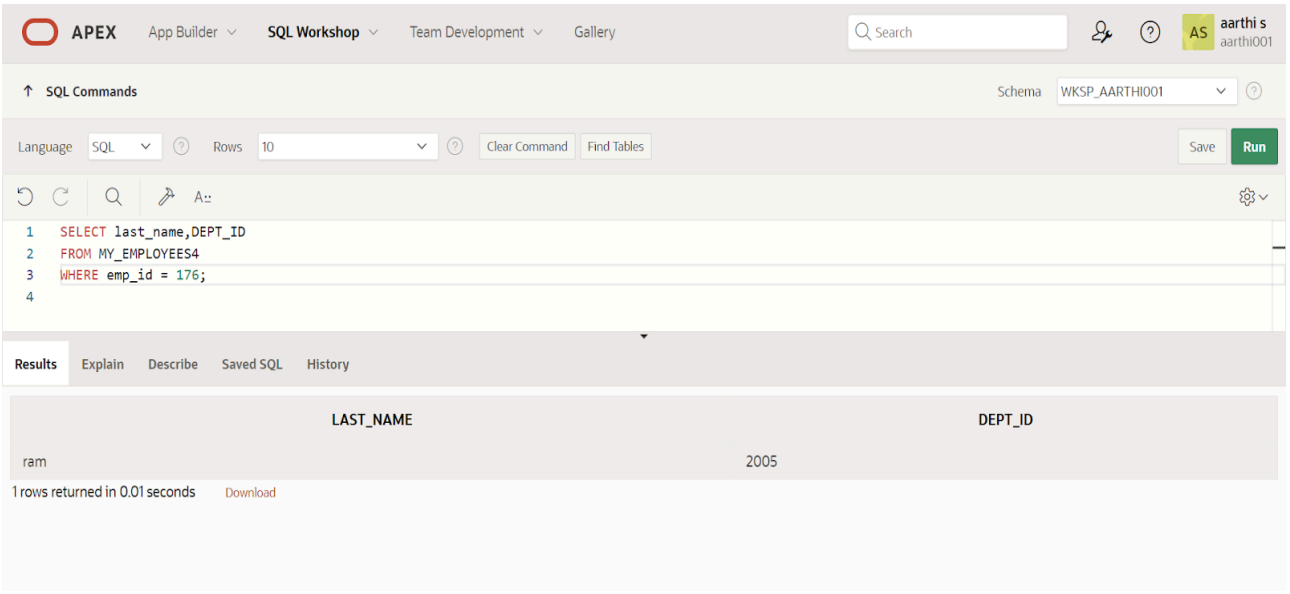


2. Create a query to display the employee last name and department number for employee number 176

   QUERY:

   SELECT last_name,DEPT_ID FROM MY_EMPLOYEES4 WHERE emp_ id = 176;

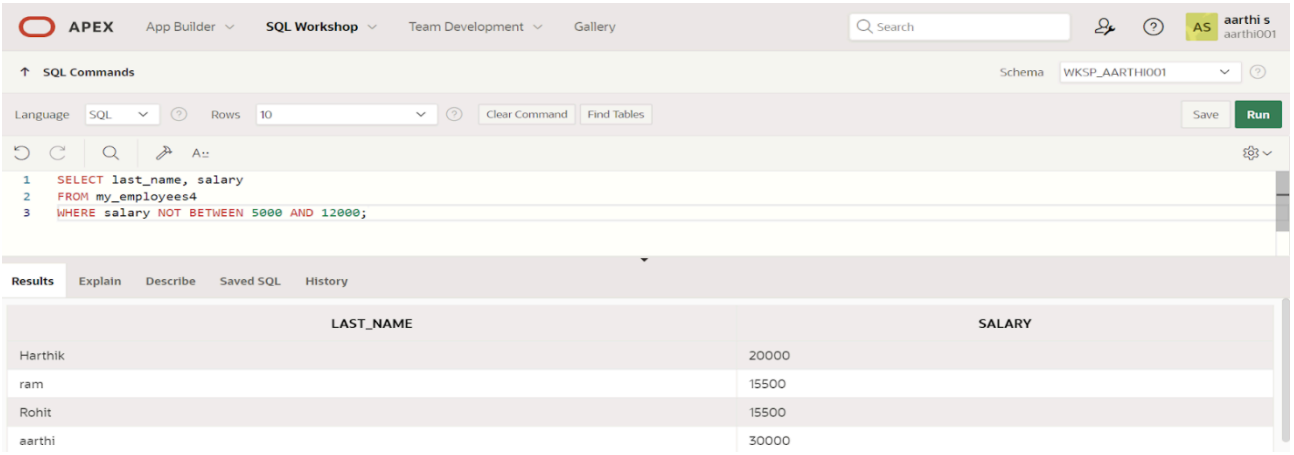
   OUTPUT:



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

   QUERY:
   SELECT last_name, salary FROM my_employees4 WHERE salary NOT BETWEEN 5000 AND 12000;

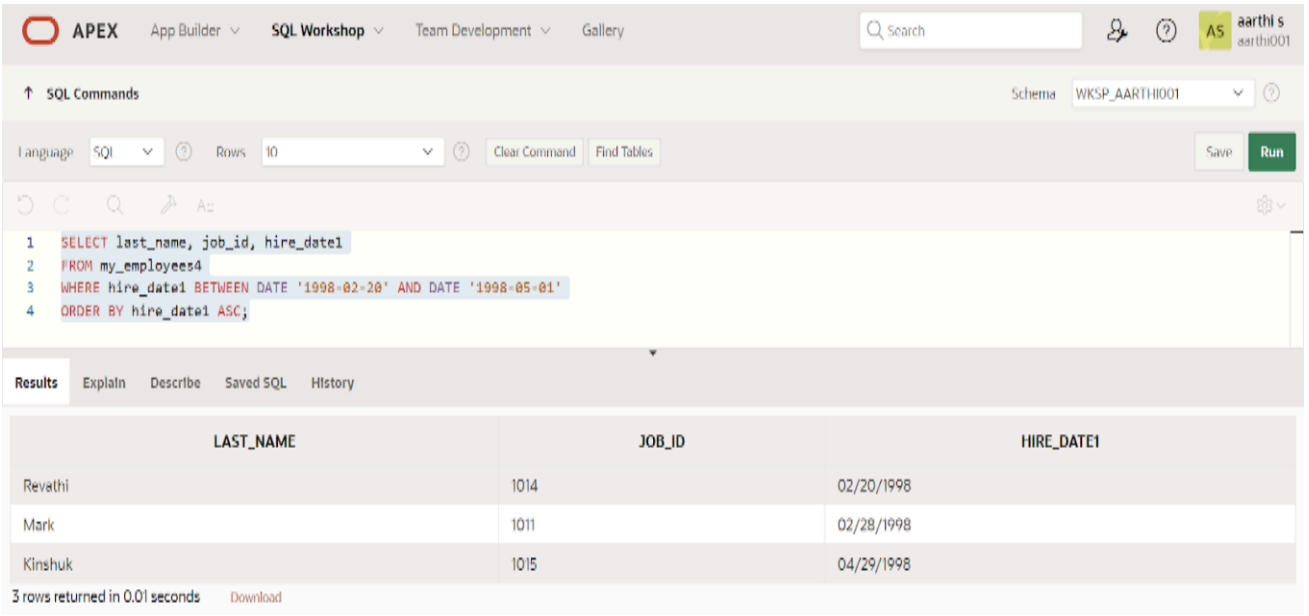
   OUTPUT:



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_date1 FROM my_employees4 WHERE hire_date1 BETWEEN DATE '1998-02-20' AND DATE '1998-05-01' ORDER BY hire_date1 ASC;

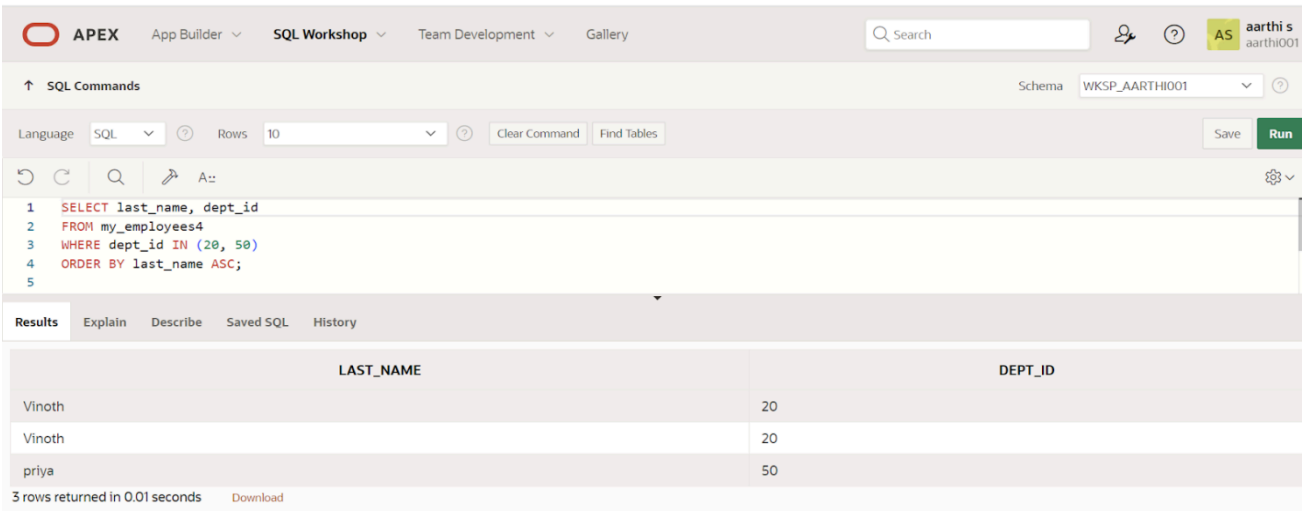
   OUTPUT:



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 my_employees4 WHERE dept_id IN (20, 50)ORDER BY last_name ASC;

OUTPUT:

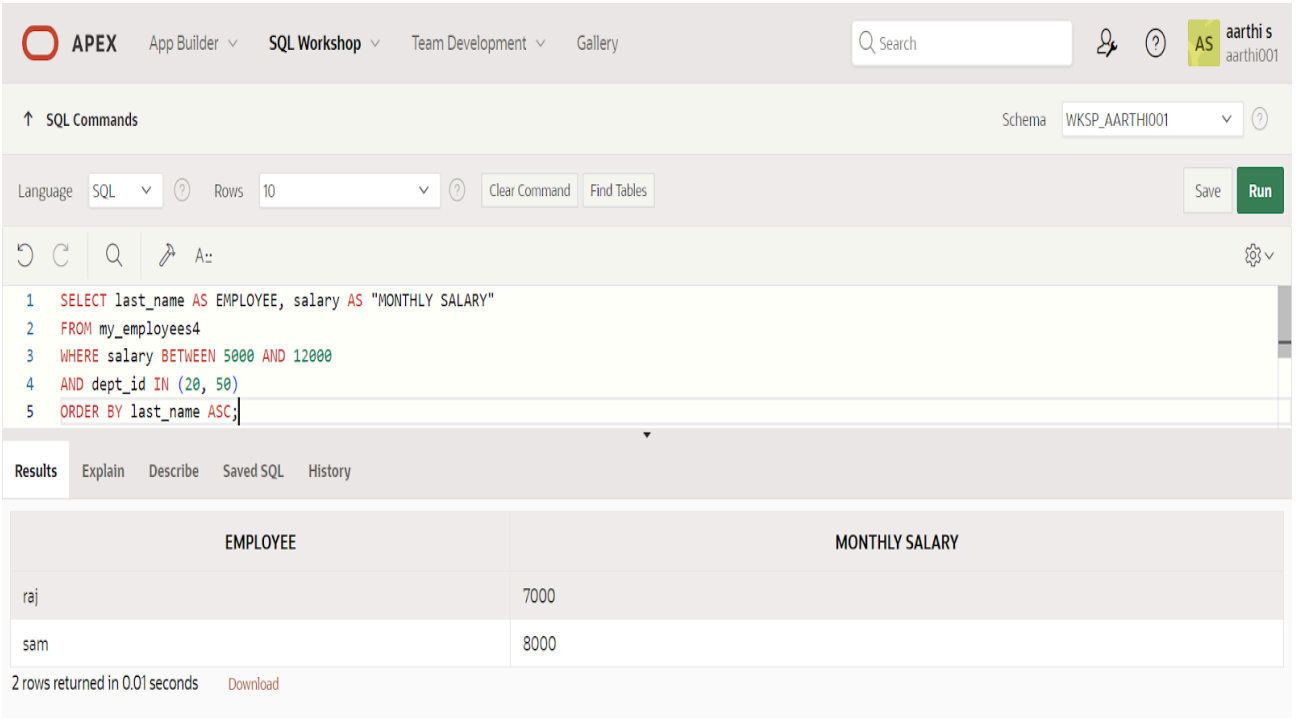


6. Display the last name and salary of all employees who earn between 5000 and 12000 and are in departments 20 and 50 in alphabetical order by name. Label the columns EMPLOYEE, MONTHLY SALARY respectively.(hints: between, in)

QUERY:
SELECT last_name AS EMPLOYEE, salary AS "MONTHLY SALARY" FROM my_employees4

WHERE salary BETWEEN 5000 AND 12000 AND dept_id IN (20, 50)
ORDER BY last_name ASC;

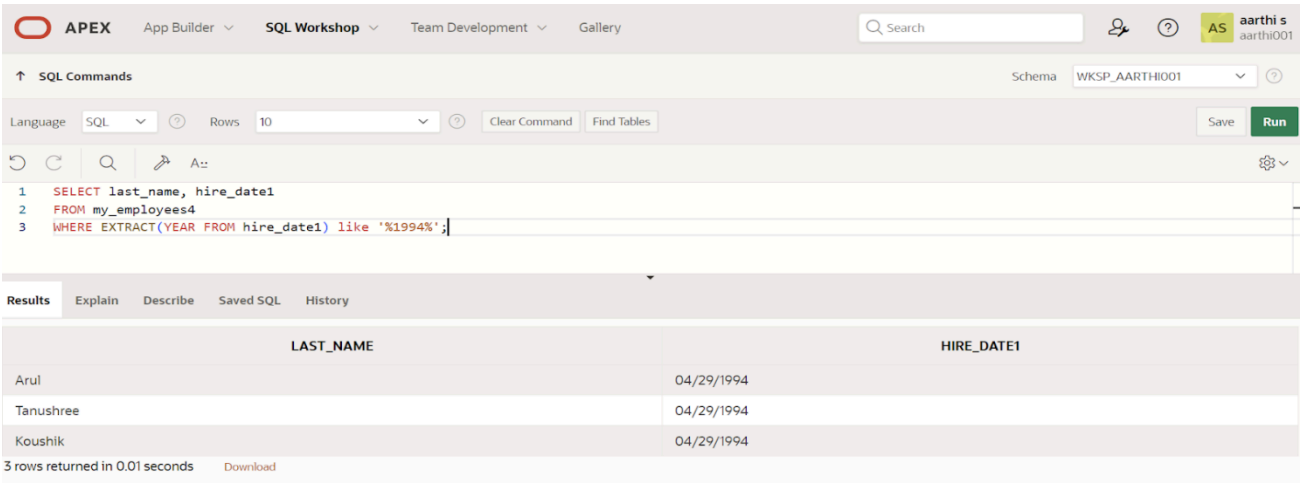
OUTPUT:



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

QUERY:
SELECT last_name, hire_date1 FROM my_employees4 WHERE EXTRACT(YEAR FROM hire_date1) like '%1994%';

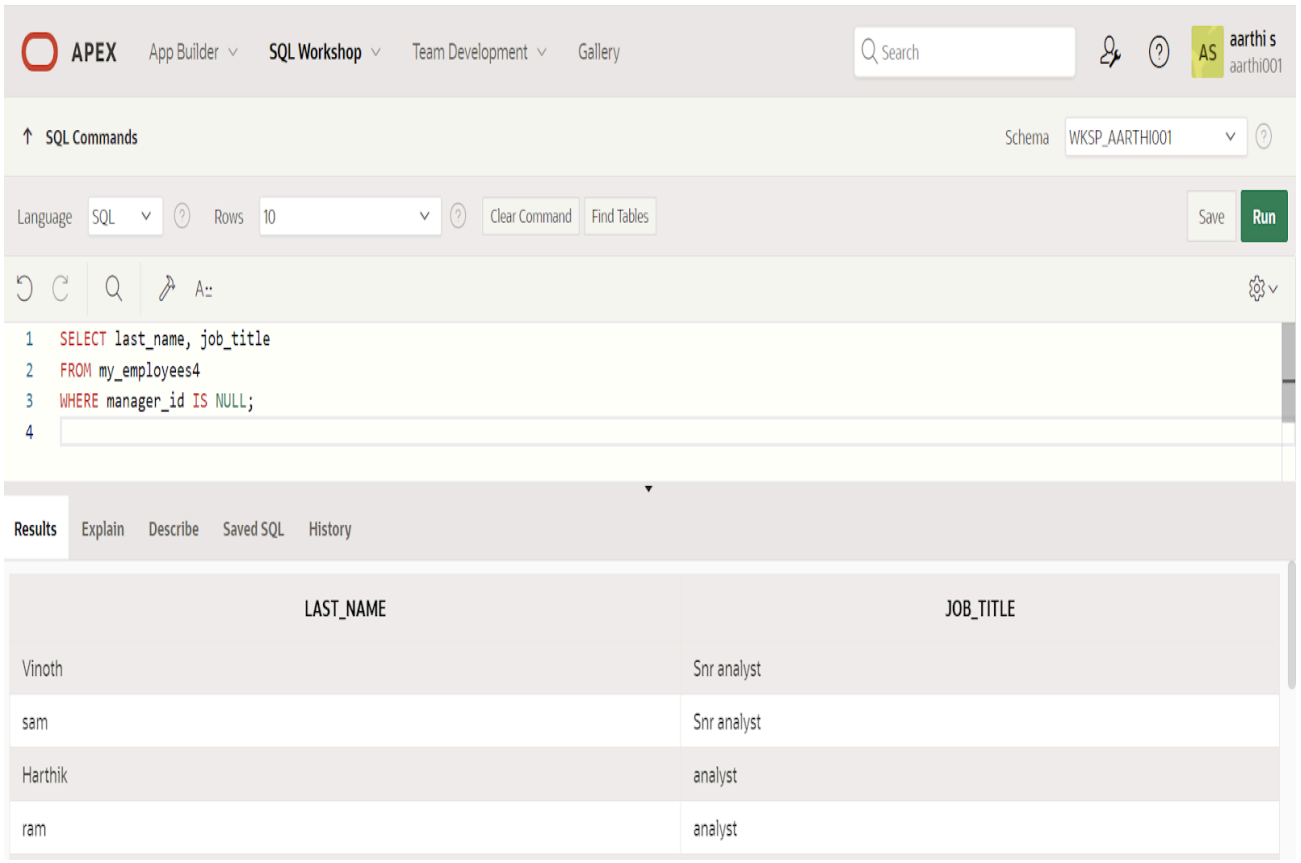
OUTPUT:



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_title FROM my_employees4 WHERE manager_id IS NULL;

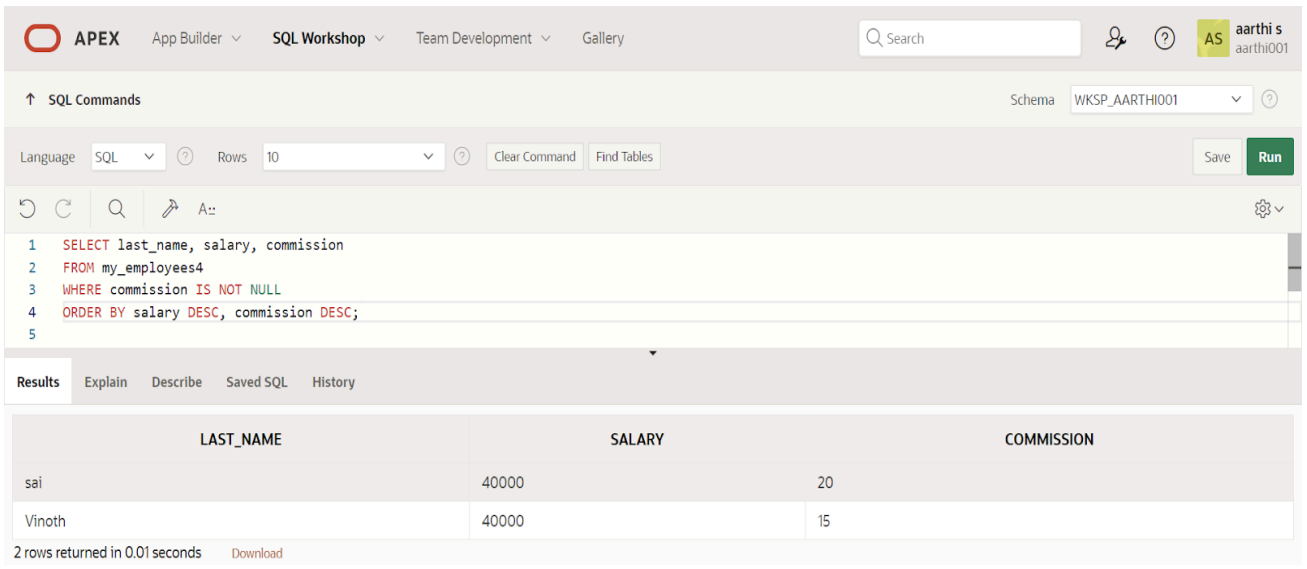
OUTPUT:



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 null,orderby)

QUERY:
SELECT last_name, salary, commission FROM my_employees4 WHERE commission IS NOT NULL ORDER BY salary DESC, commission DESC;

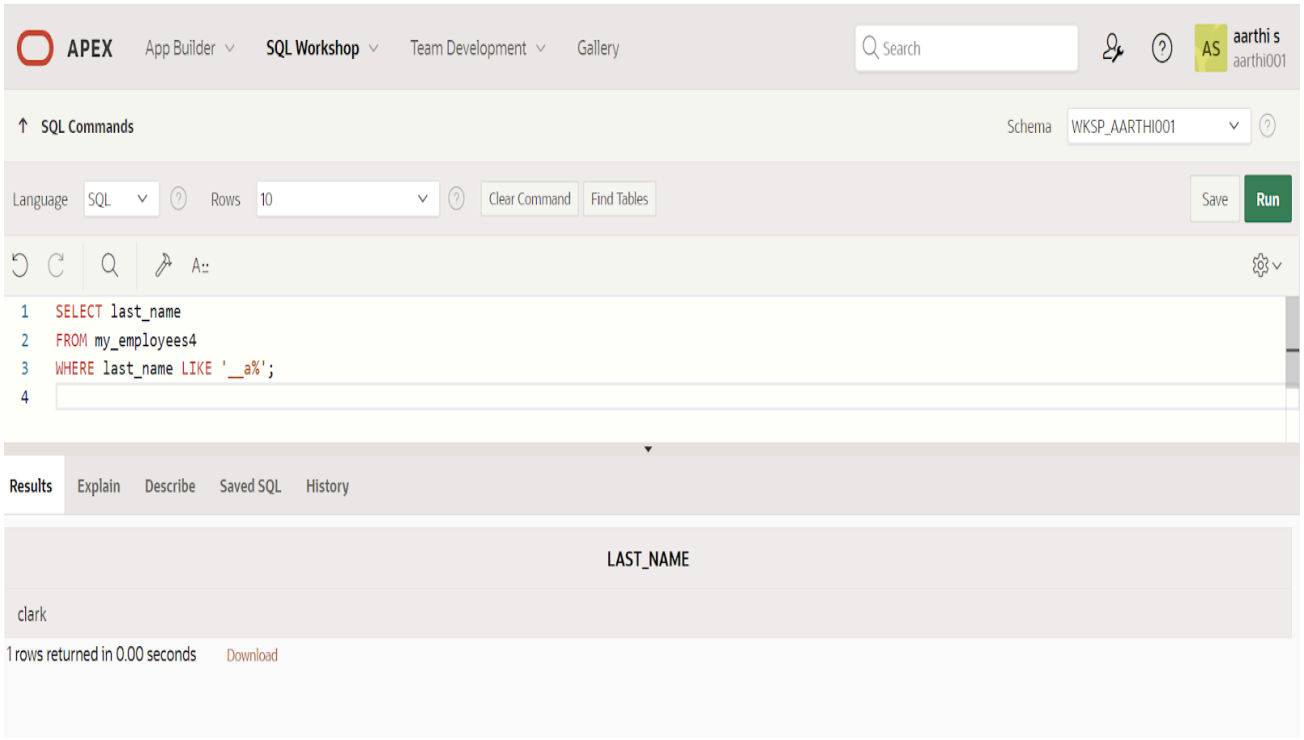
OUTPUT:



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

QUERY:
SELECT last_name FROM my_employees4 WHERE last_name LIKE '__a%';

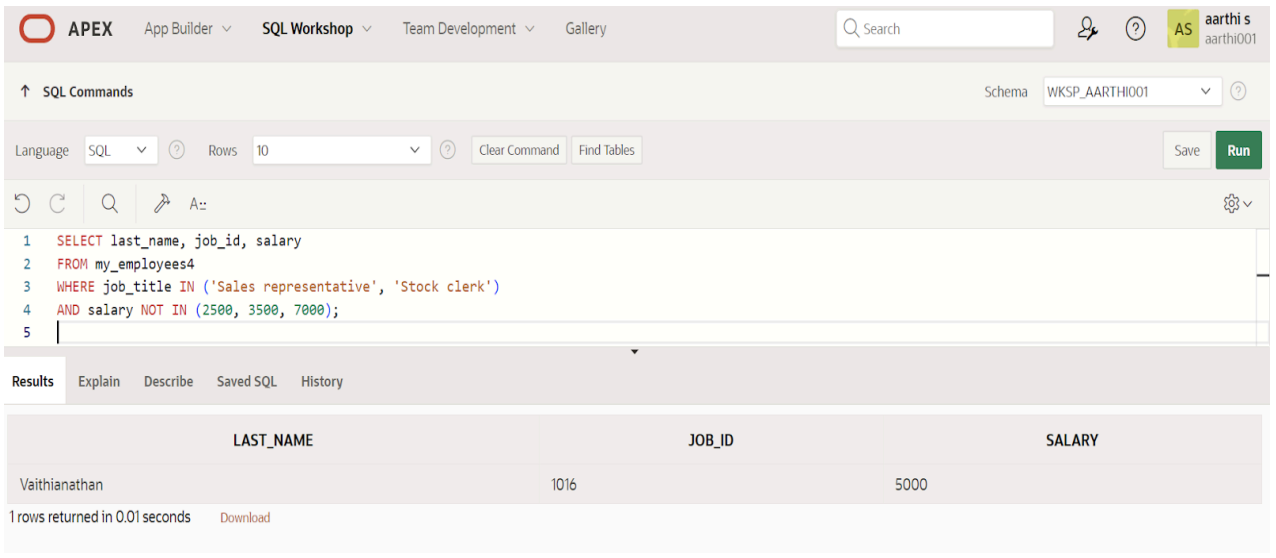
OUTPUT:



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

QUERY:
SELECT last_name, job_id, salary FROM my_employees4 WHERE job_title IN ('Sales representative', 'Stock clerk') AND salary NOT IN (2500, 3500, 7000);

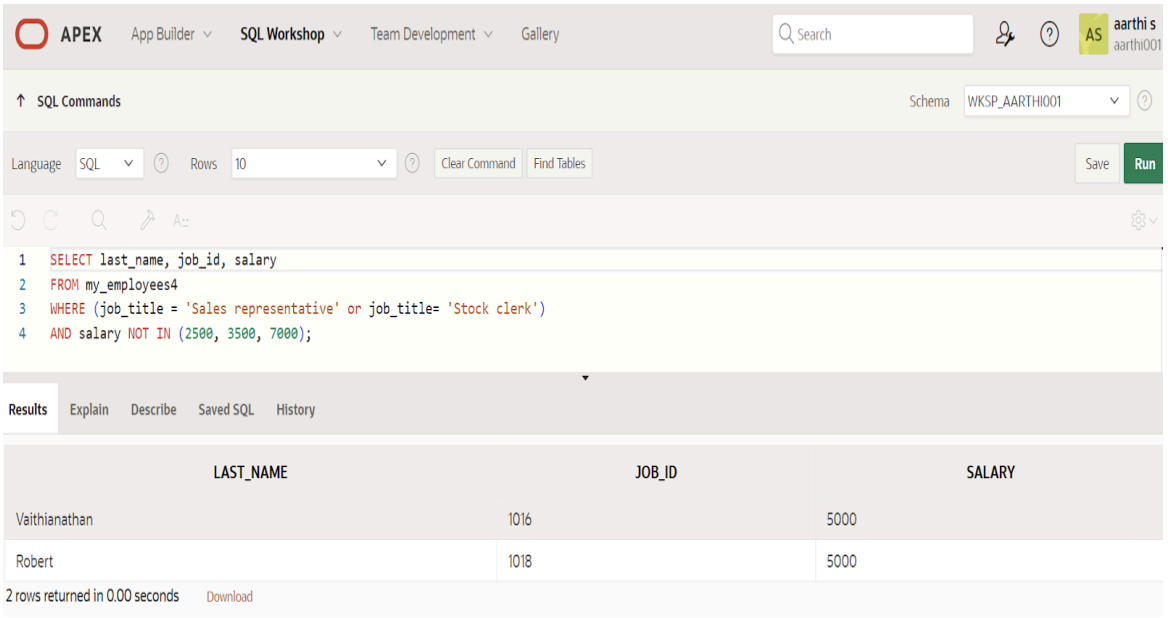
OUTPUT:



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 my_employees4 WHERE (job_title = 'Sales representative' or job_title= 'Stock clerk') AND salary NOT IN (2500, 3500, 7000);

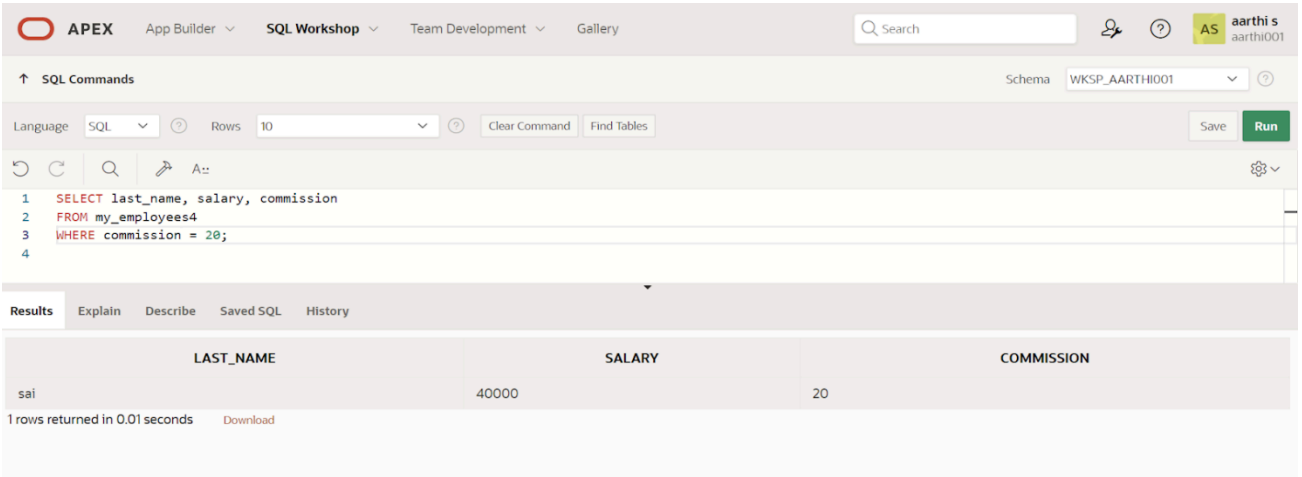
OUTPUT:



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

QUERY:
SELECT last_name, salary, commission FROM my_employees4 WHERE commission = 20;

OUTPUT:



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

SINGLE ROW FUNCTIONS

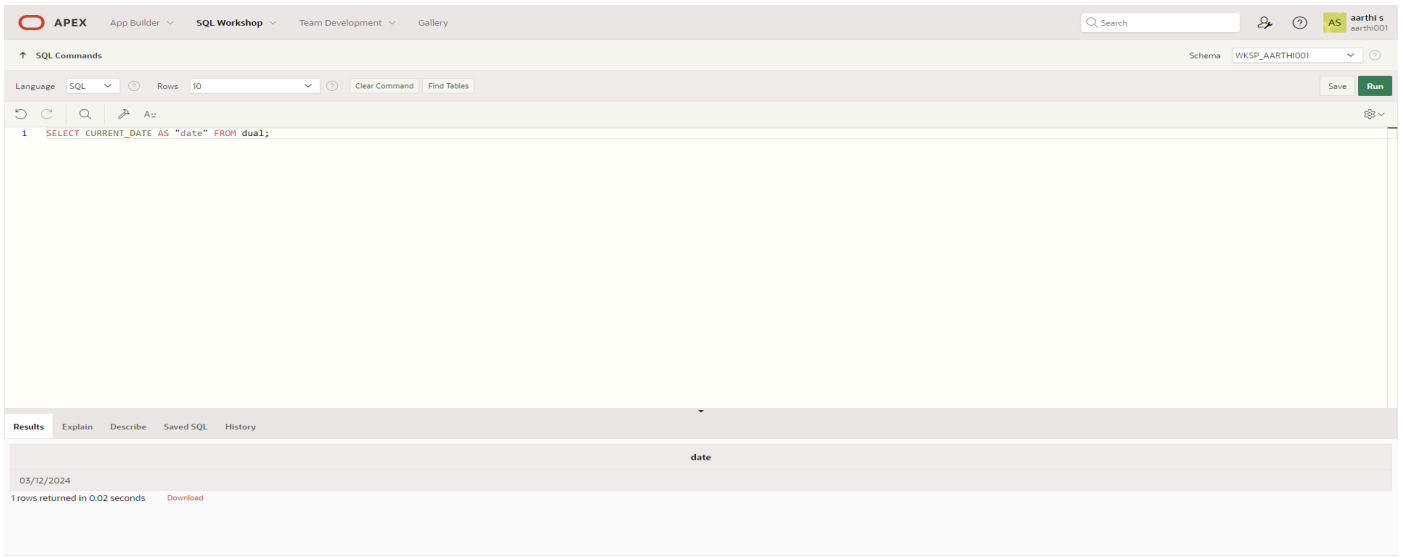
EX_NO:6 **DATE:**

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

QUERY:

select sysdate from dual;

OUTPUT:



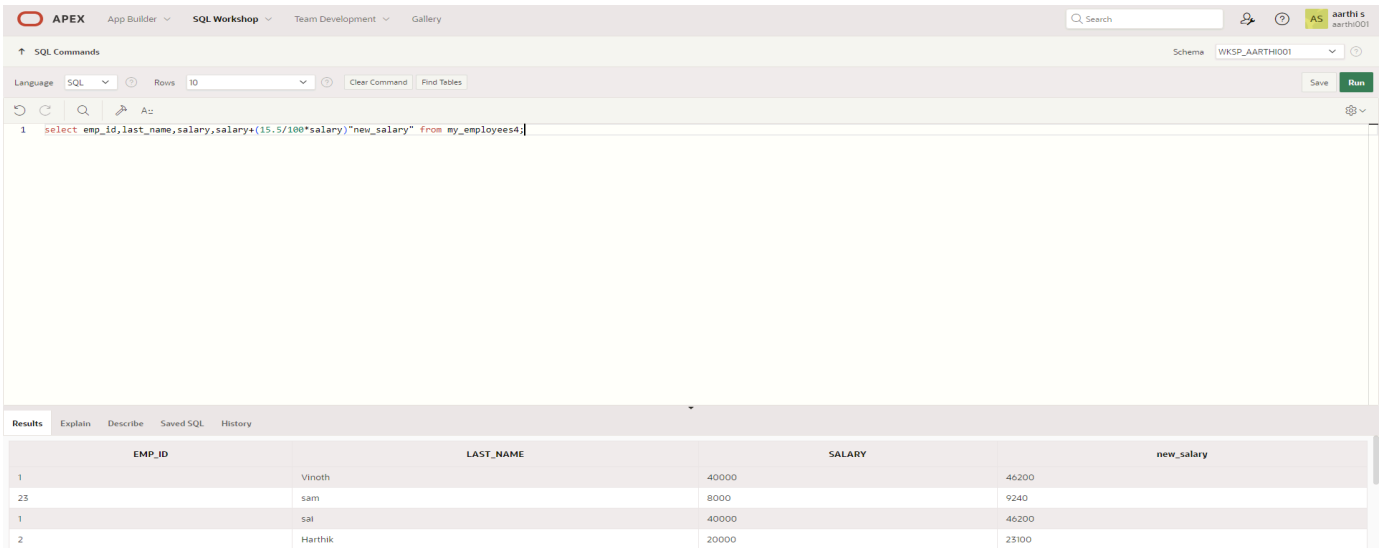
2. The HR department needs a report to display the employee number, last name, salary, and increased by

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 my_employees4;

OUTPUT:

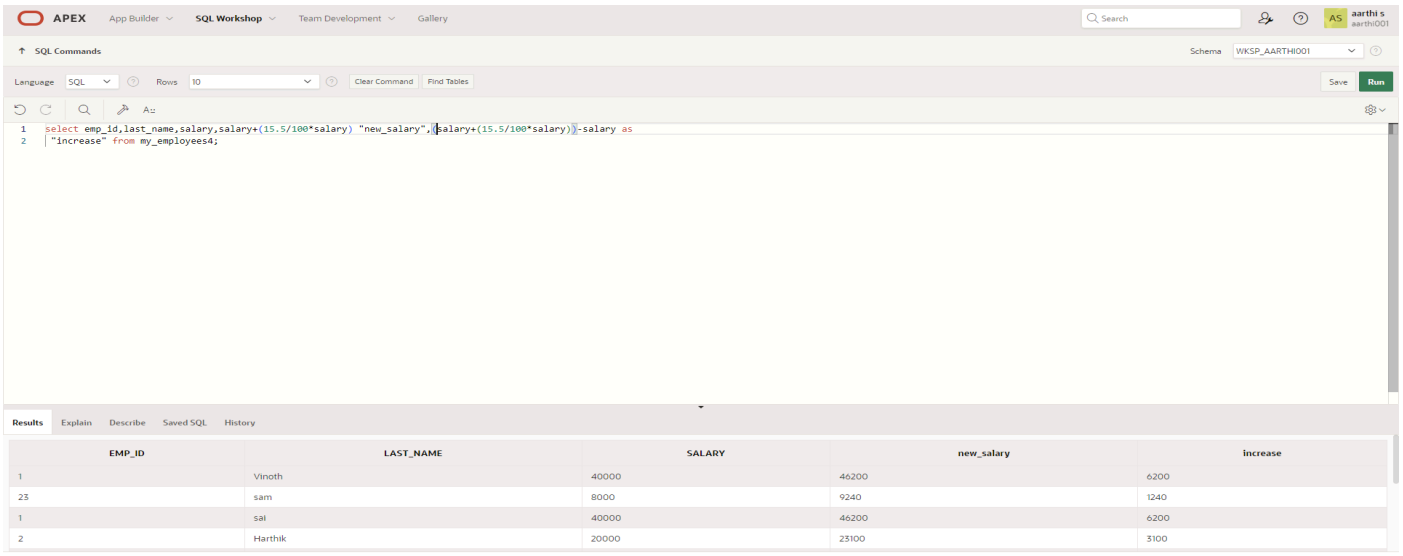


3. Modify your query lab_03_02.sql to add a column that subtracts the old salary from the new salary. Label the column Increase.

QUERY:

select employee_id,last_name,salary,salary+(15.5/100*salary)
"new_salary",salary+(15.5/100*salary)- salary as "Increase" from my_employees4;

OUTPUT:

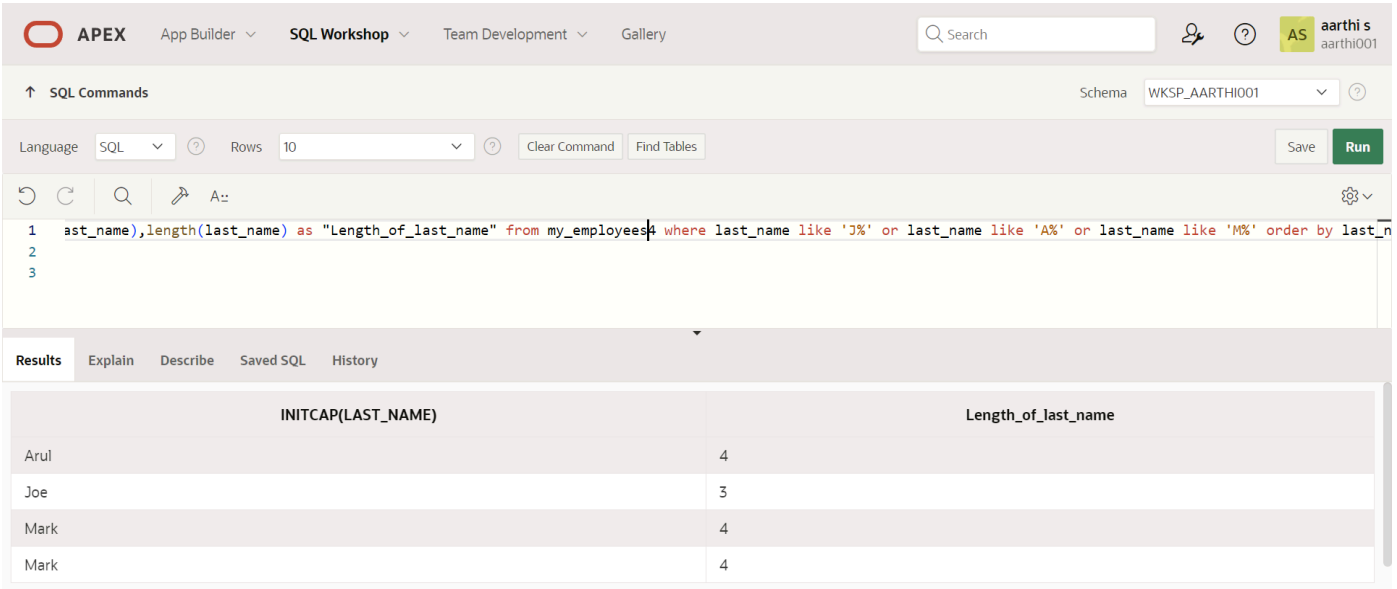


4. Write a query that displays the last name (with the first letter uppercase and all other letters lowercase) and the length of the last name for all employees_table whose name starts with the letters J, A, or M. Give each column an appropriate label. Sort the results by the employees_table' last names.

QUERY:

Select initcap(last_name),length(last_name) as "Length_of_last_name" from my _employees4 where last_name like 'J%' or last_name like 'A%' or last_name like 'M%' order by last_name asc;

OUTPUT:

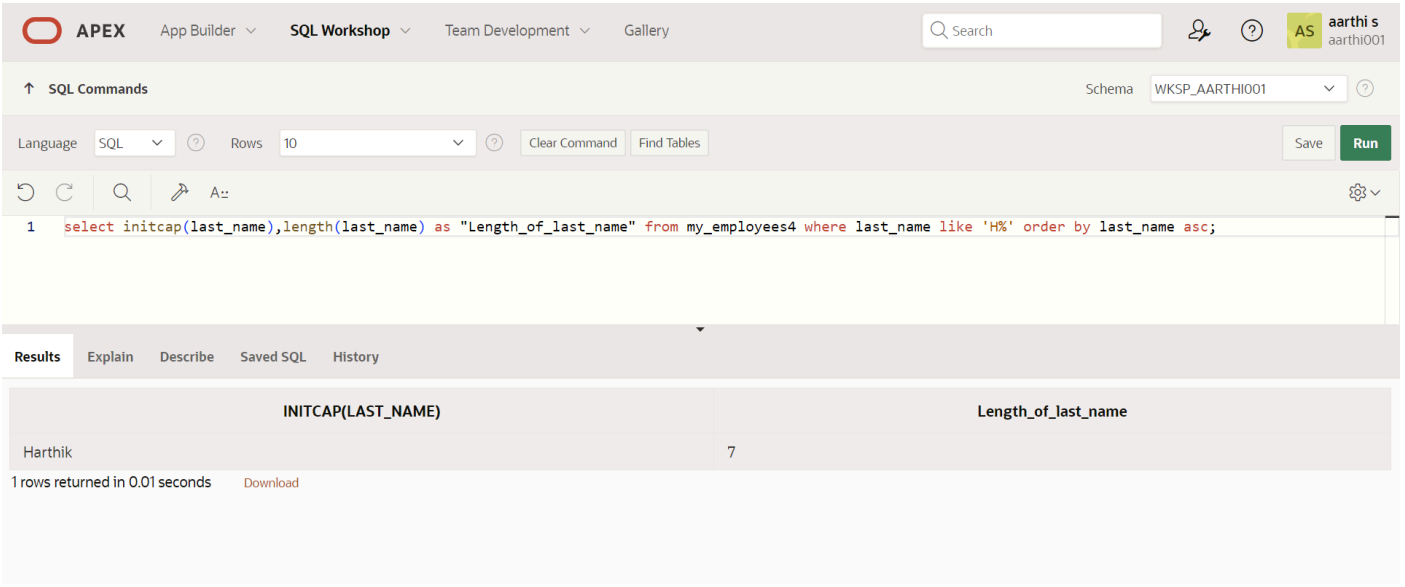


5. Rewrite the query so that the user is prompted to enter a letter that starts the last name. For example, if the user enters H when prompted for a letter, then the output should show all employees_table whose last name starts with the letter H.

QUERY:

select initcap(last_name),length(last_name) as "Length_of_last_name" from my_ employees4 where last_name like 'H%' order by last_name asc;

OUTPUT:

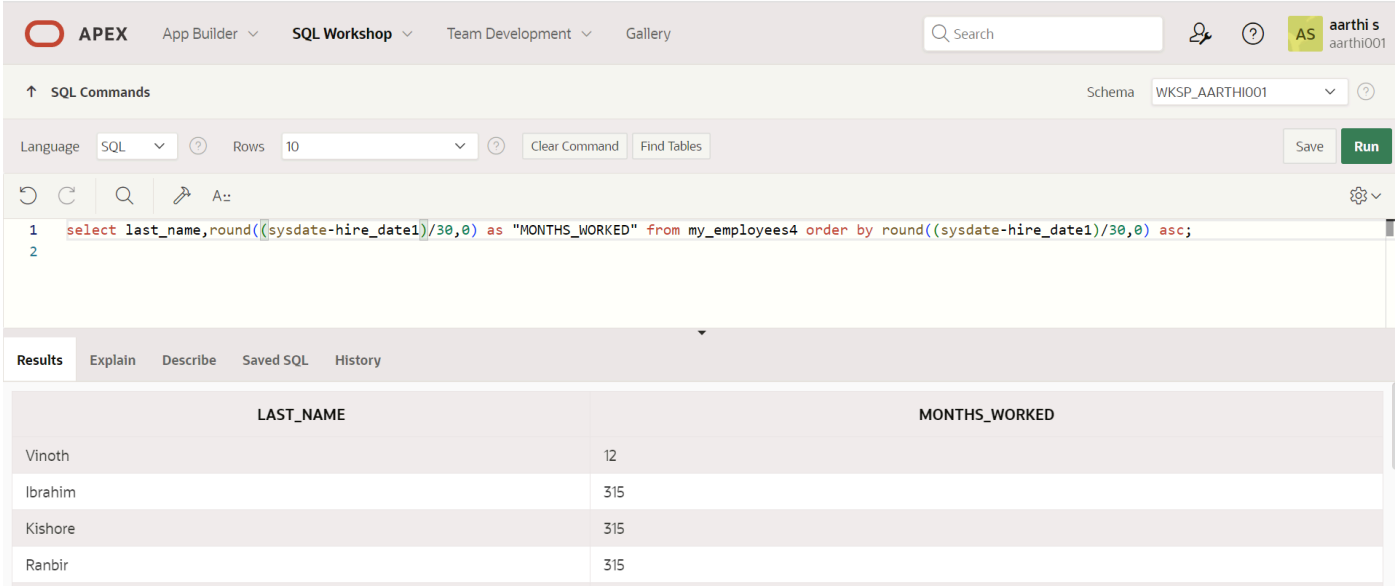


6. The HR department wants to find the length of employment for each employee. For each employee, display the last name and calculate the number of months between today and the date on which the employee was hired. Label the column MONTHS_WORKED. Order your results by the number of months employed. Round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

QUERY:

select last_name,round((sysdate-hire_date)/30,0) as "MONTHS_WORKED" from my_ employees4 order by round((sysdate-hire_date)/30,0) asc;

OUTPUT:

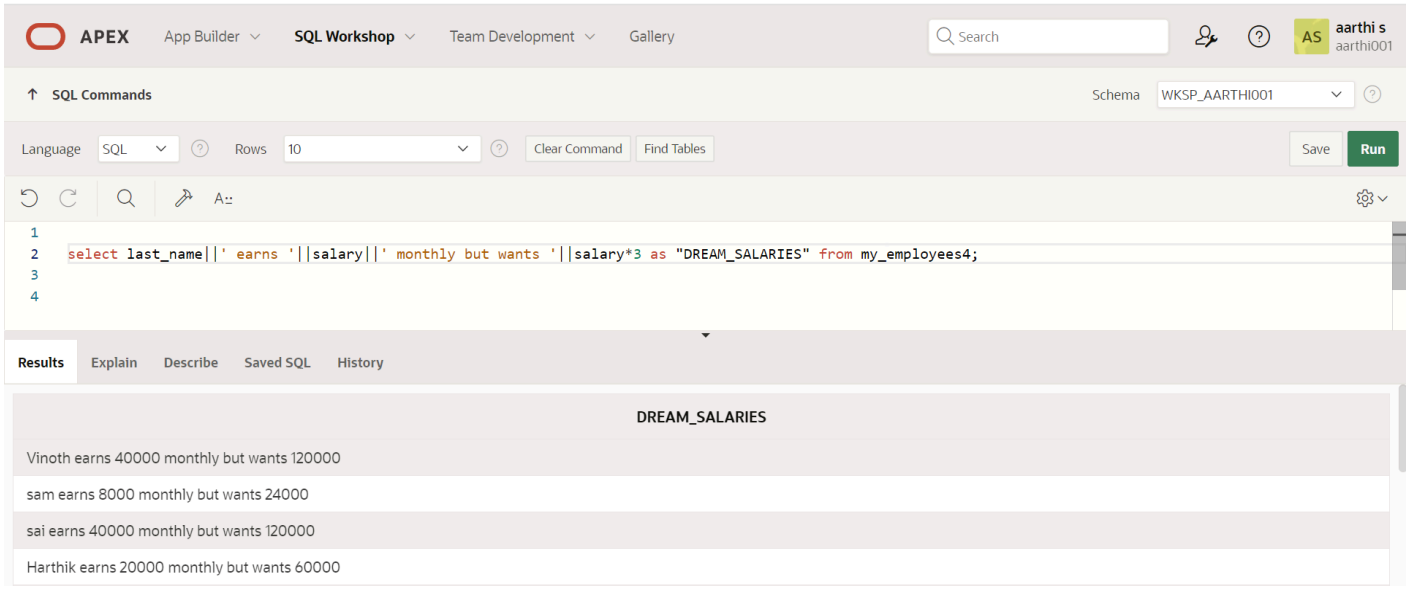


7. Create a report that produces the following for each employee:
<employee last name> earns<salary>monthly but wants <3 times salary>.Label the column Dream Salaries.

QUERY:

```
select last_name||' earns '||salary||' monthly but wants '||salary*3 as "DREAM_SALARIES" from my_employees4
```

OUTPUT:

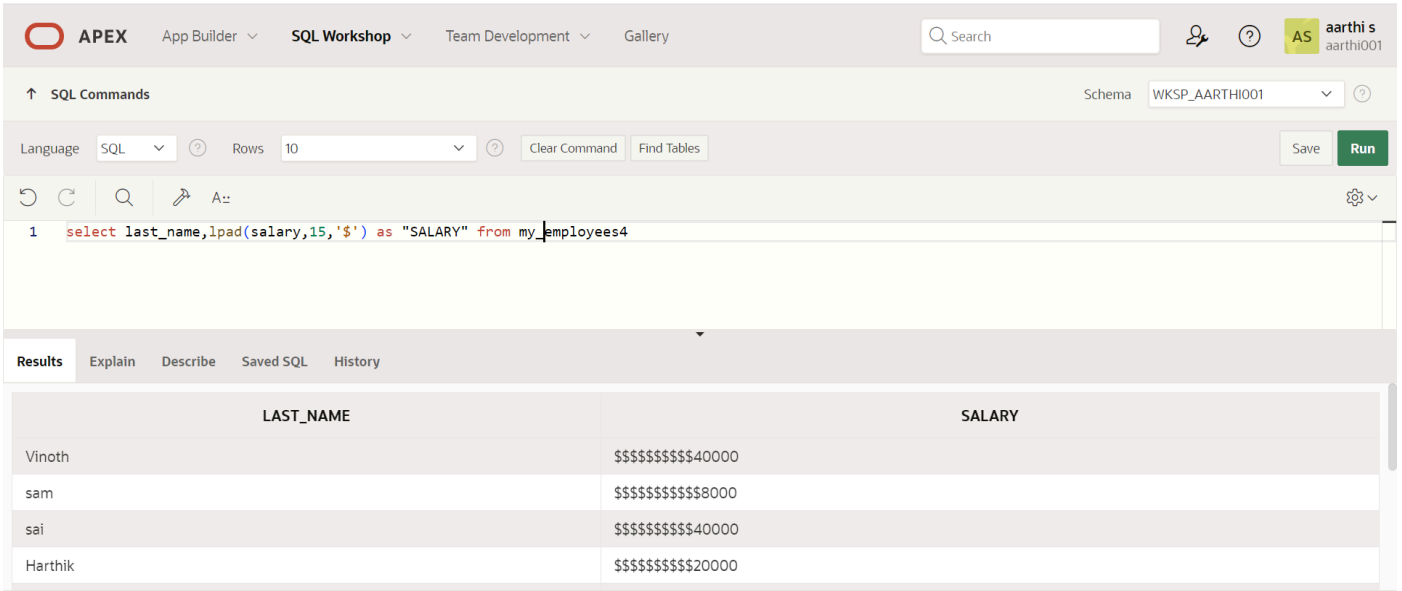


8. Create a query to display the last name and salary for all employees_table. Format the salary to be 15 characters long, left-padded with the $ symbol. Label the column SALARY.

QUERY:

```
select last_name,lpad(salary,15,'$') as "SALARY" from my_  employees4;
```

OUTPUT:

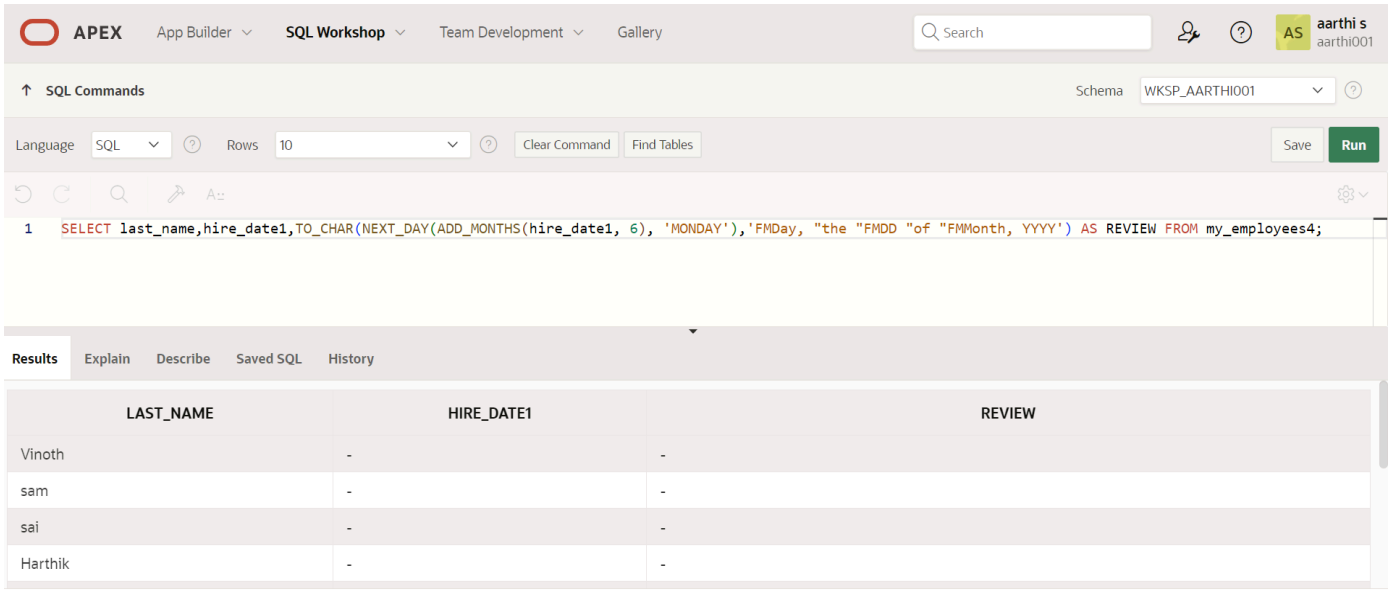


9. Display each employee's last name, hire date, and salary review date, which is the first Monday after six months of service. Label the column REVIEW. Format the dates to appear in the format similar to "Monday, the Thirty-First of July, 2000."

QUERY:

SELECT last_name,hire_date,TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6), 'MONDAY'),'FMDay, "the "FMDD "of "FMMonth, YYYY') AS REVIEW FROM my_employees4;

OUTPUT:

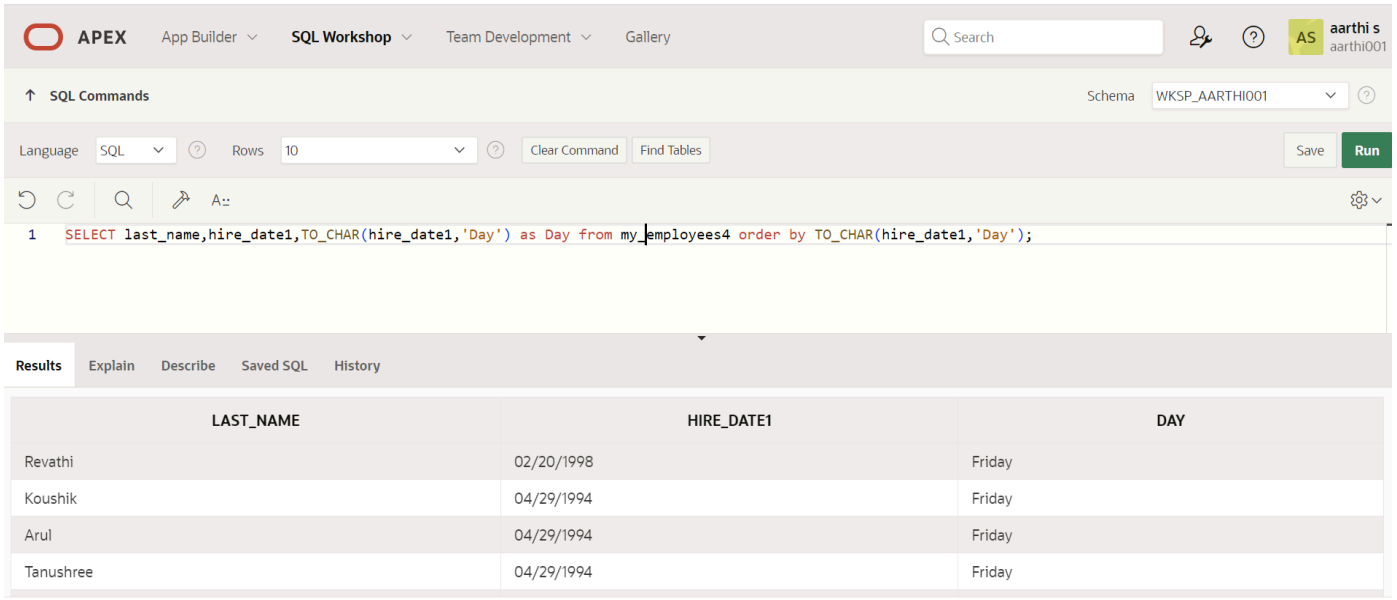


10. Display the last name, hire date, and day of the week on which the employee started. Label the column DAY. Order the results by the day of the week, starting with Monday.

QUERY:

SELECT last_name,hire_date,TO_CHAR(hire_date,'Day') as Day from my_ employees4 order by TO_CHAR(hire_date,'Day');

OUTPUT:



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

RESULT:

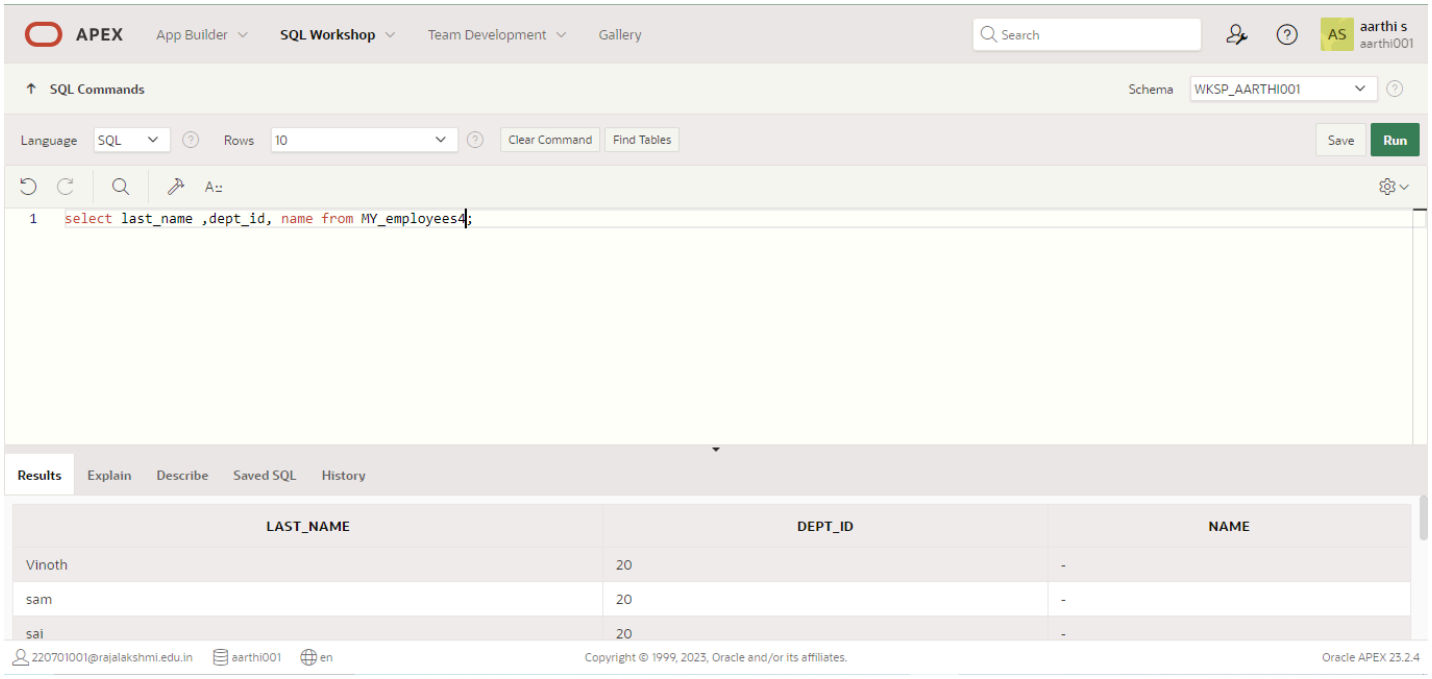
Evaluation Procedure	Marks awarded
Query (5)	
Execution (5)	
Viva (5)	
Total (5)	
Faculty Signature	

EXNO:7 DISPLAYING DATA FROM MULTIPLE TABLES **DATE**:

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

QUERY: select lname ,dept_id, dept_name from MY_emp;

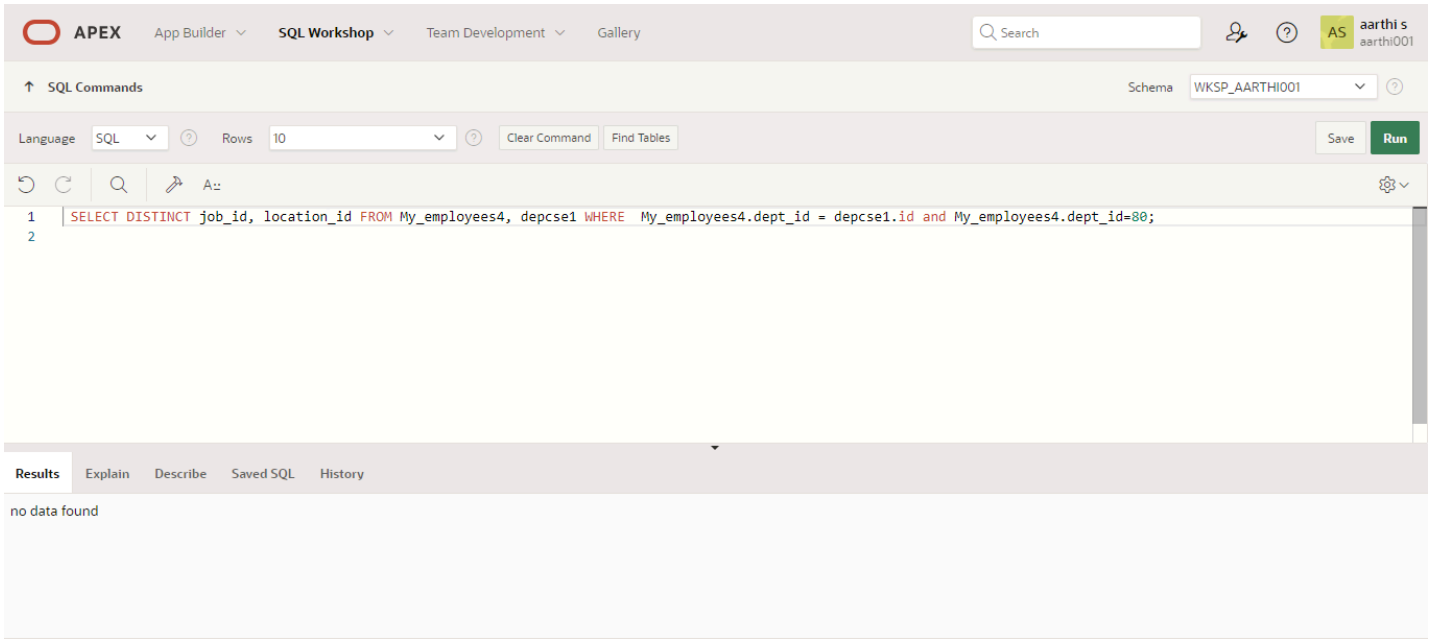
OUTPUT:



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

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

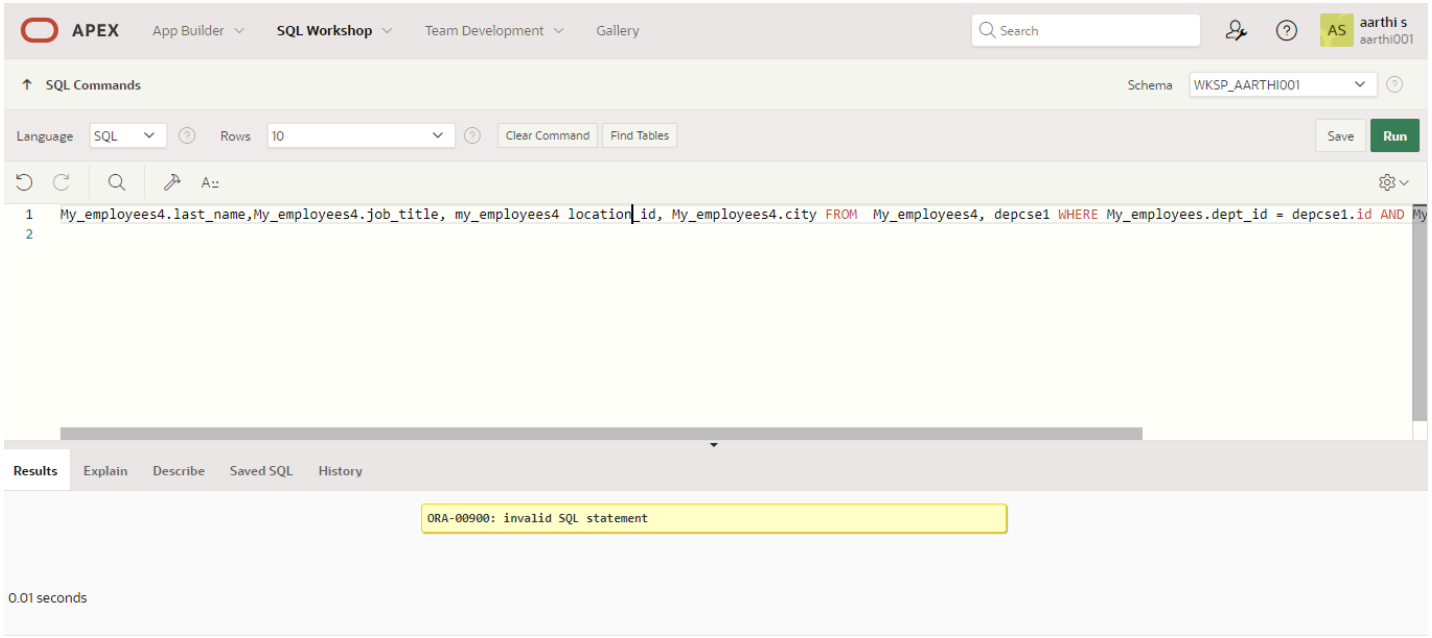
OUTPUT:



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

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

OUTPUT:



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

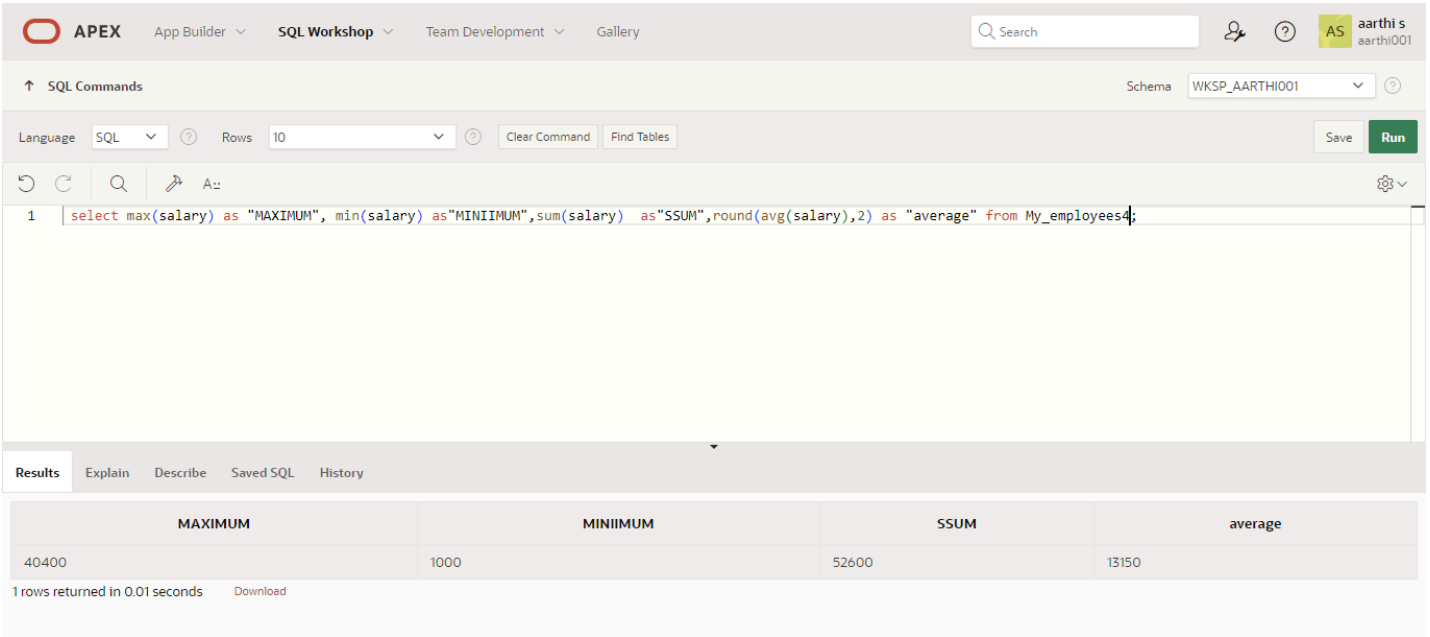
Faculty Signature	

EXNO:8 AGGREGATING DATA USING GROUP FUNCTIONS **DATE**:

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

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

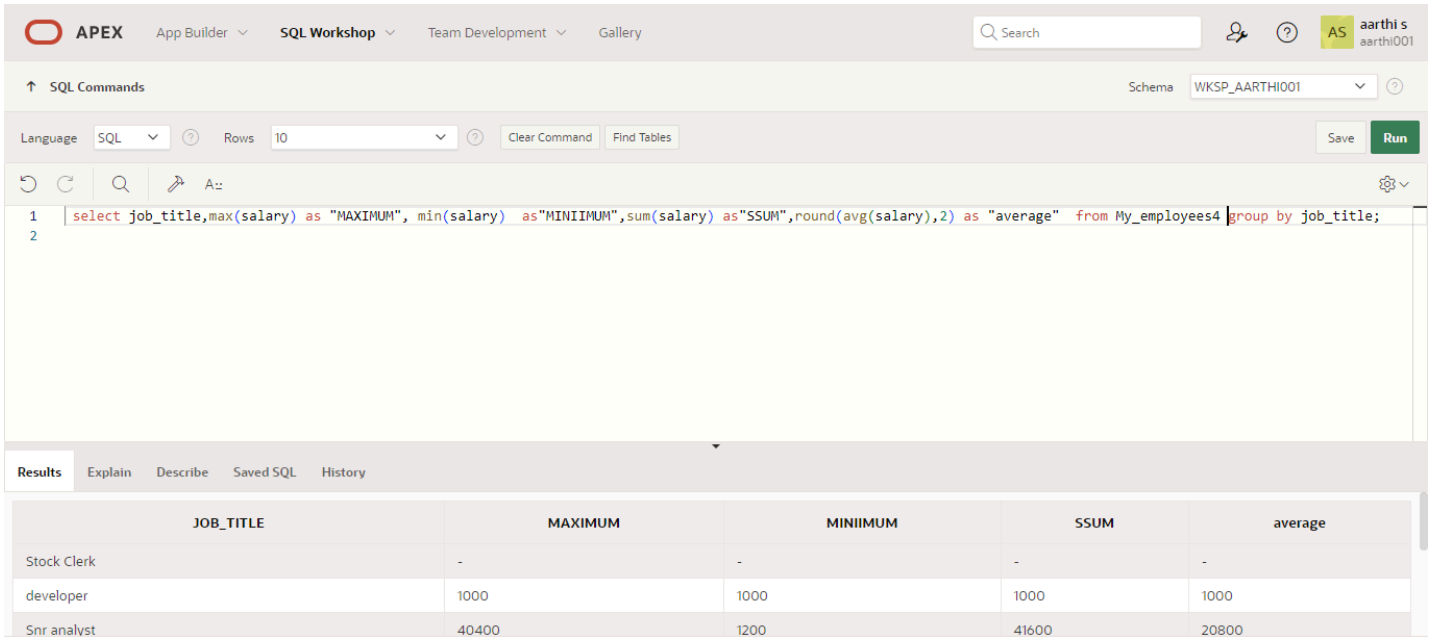
OUTPUT:



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

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

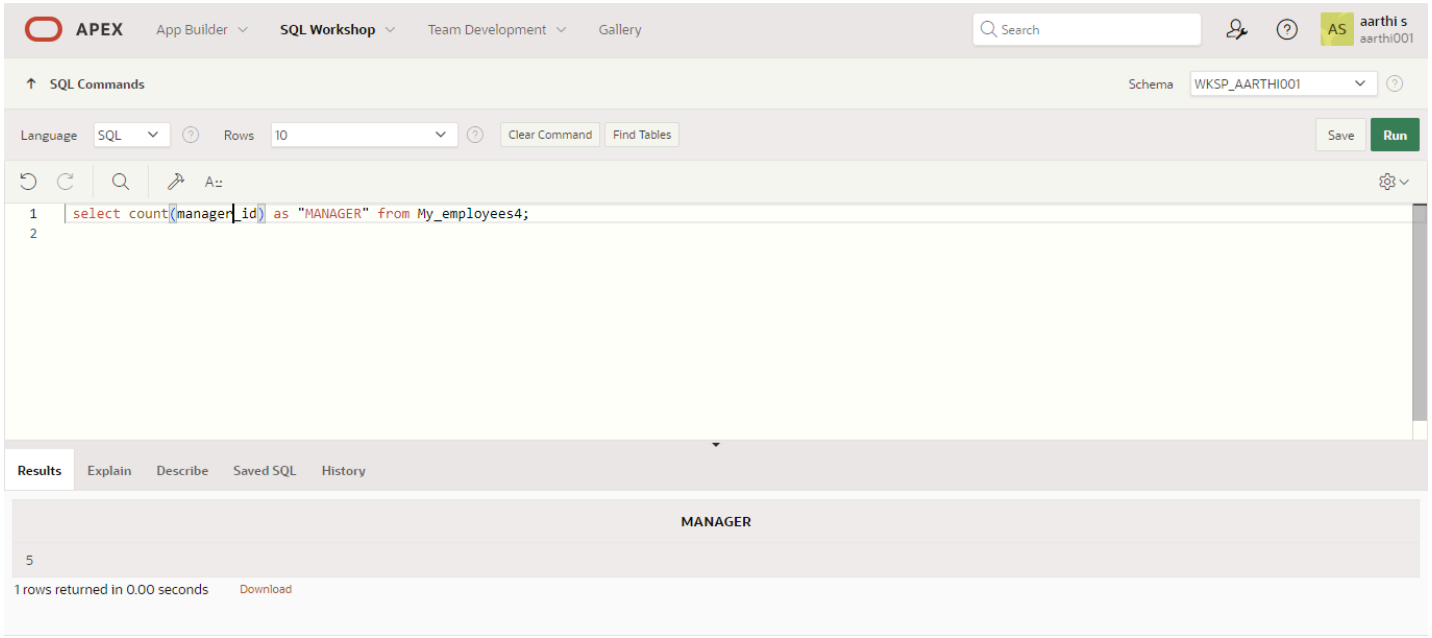
OUTPUT:



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

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

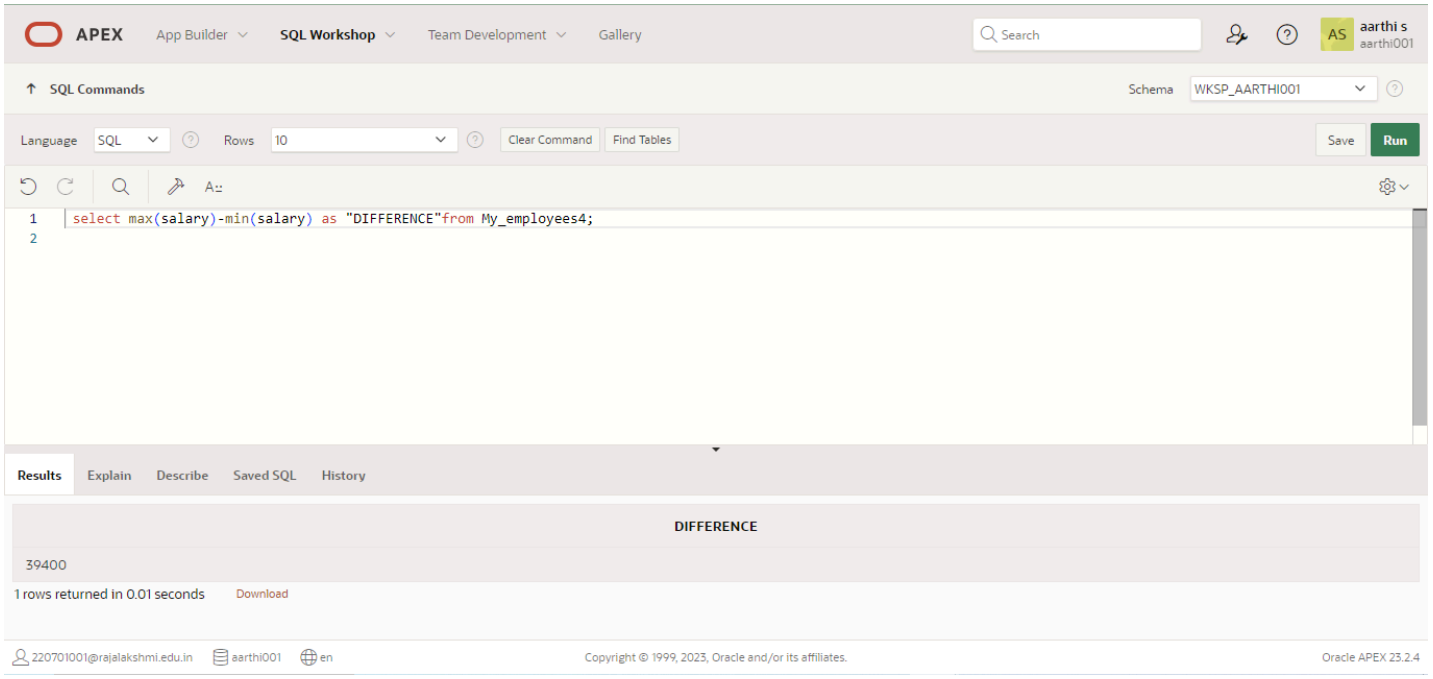
OUTPUT:



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

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

OUTPUT:

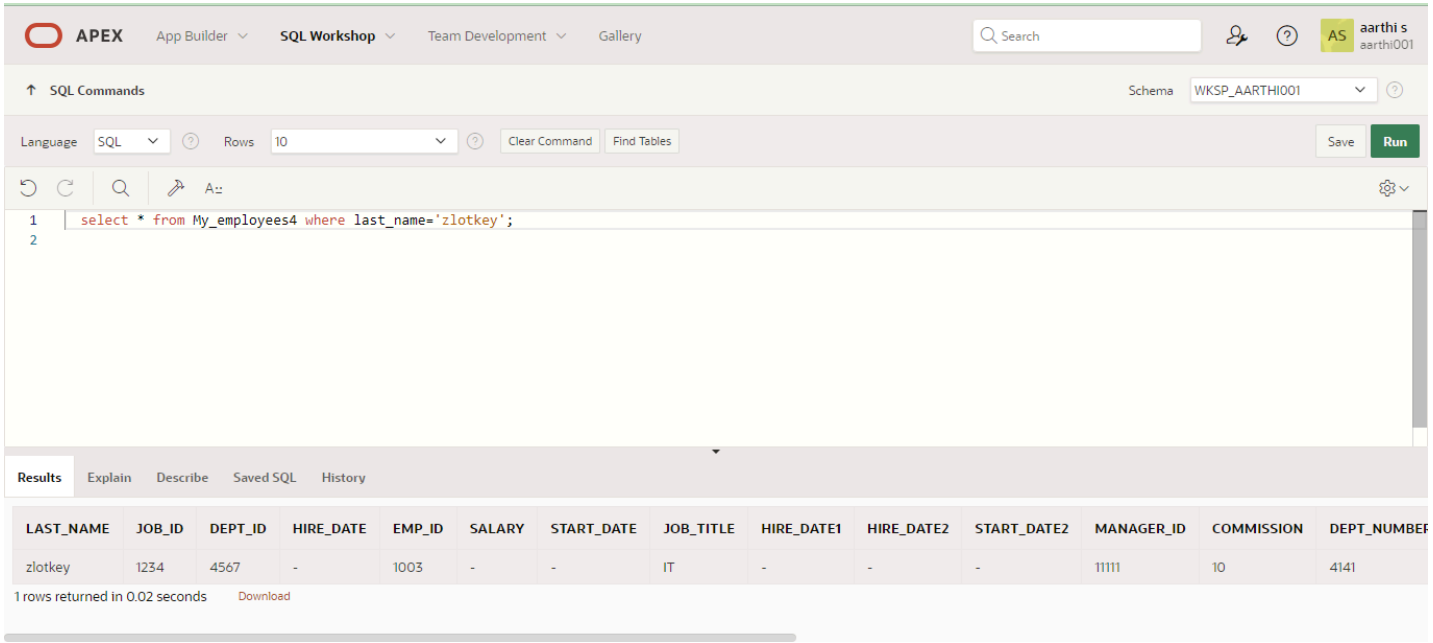


EXNO:9 SUB QUERIES DATE:

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

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

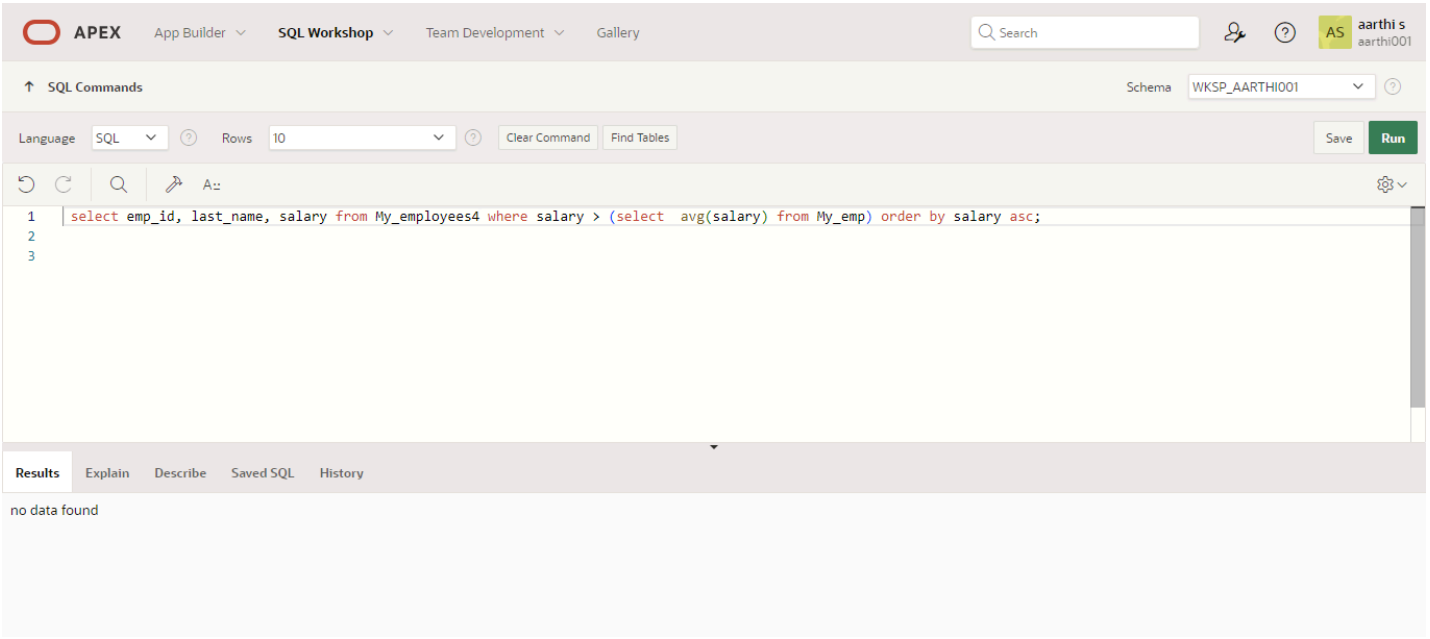
OUTPUT:



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

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

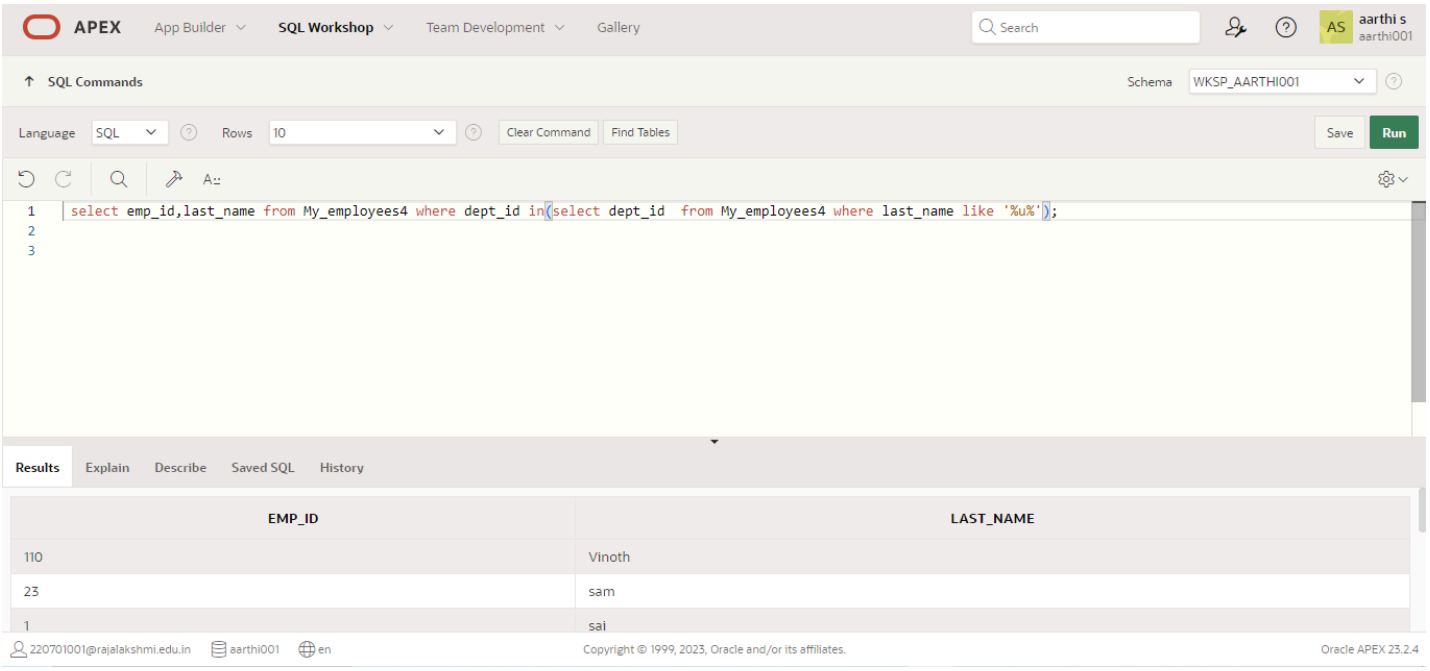
OUTPUT:



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

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

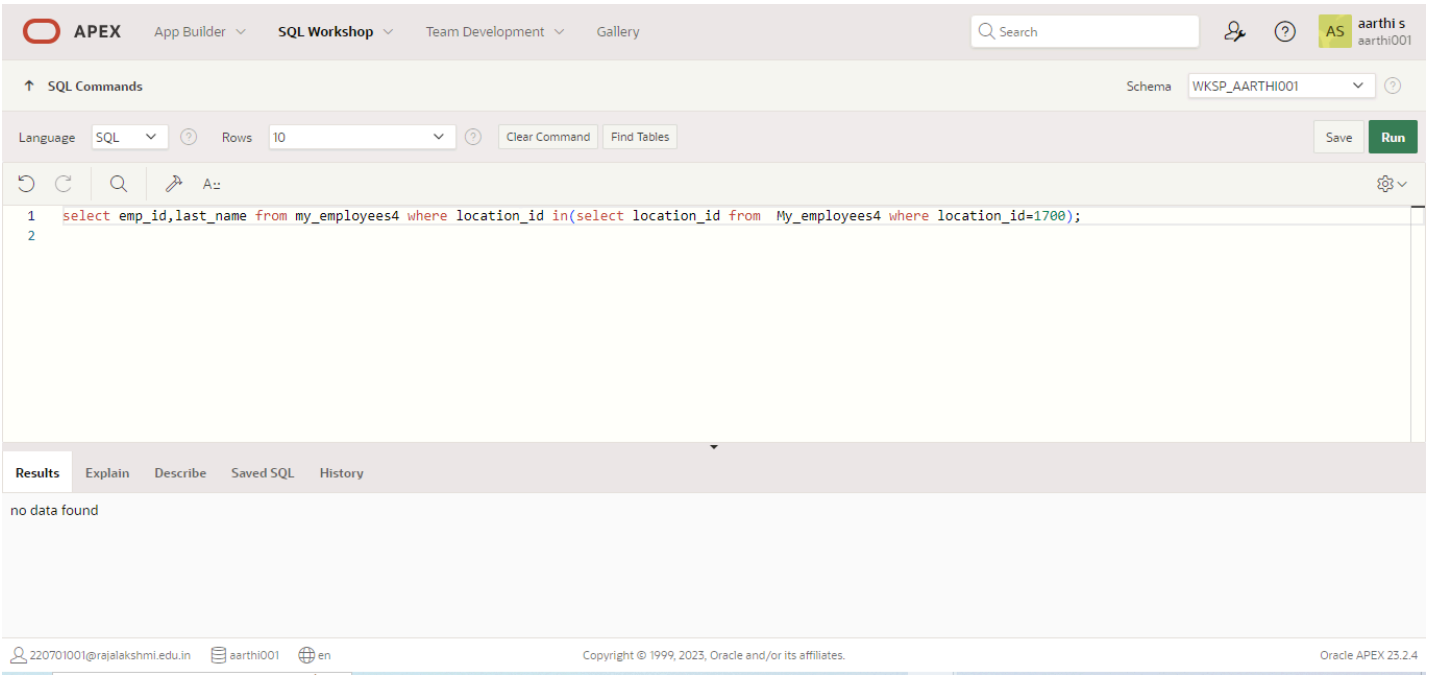
OUTPUT:



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

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

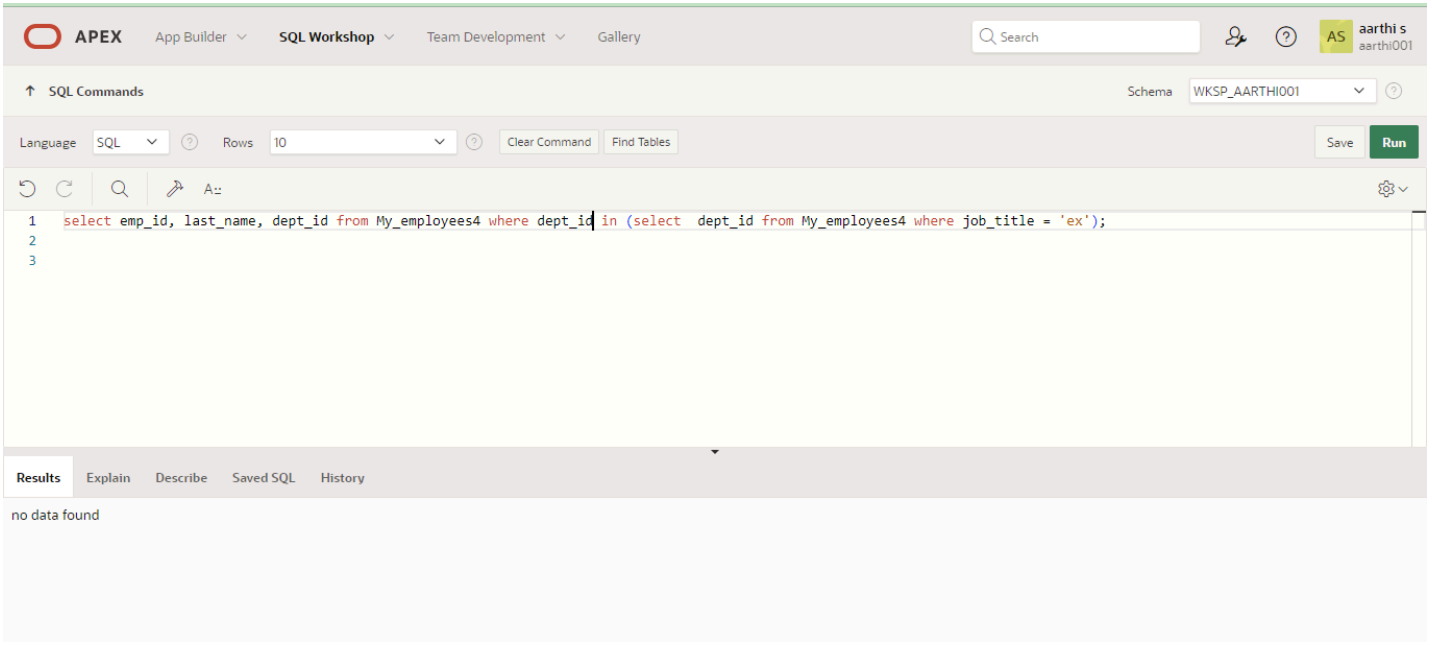
OUTPUT:



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

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

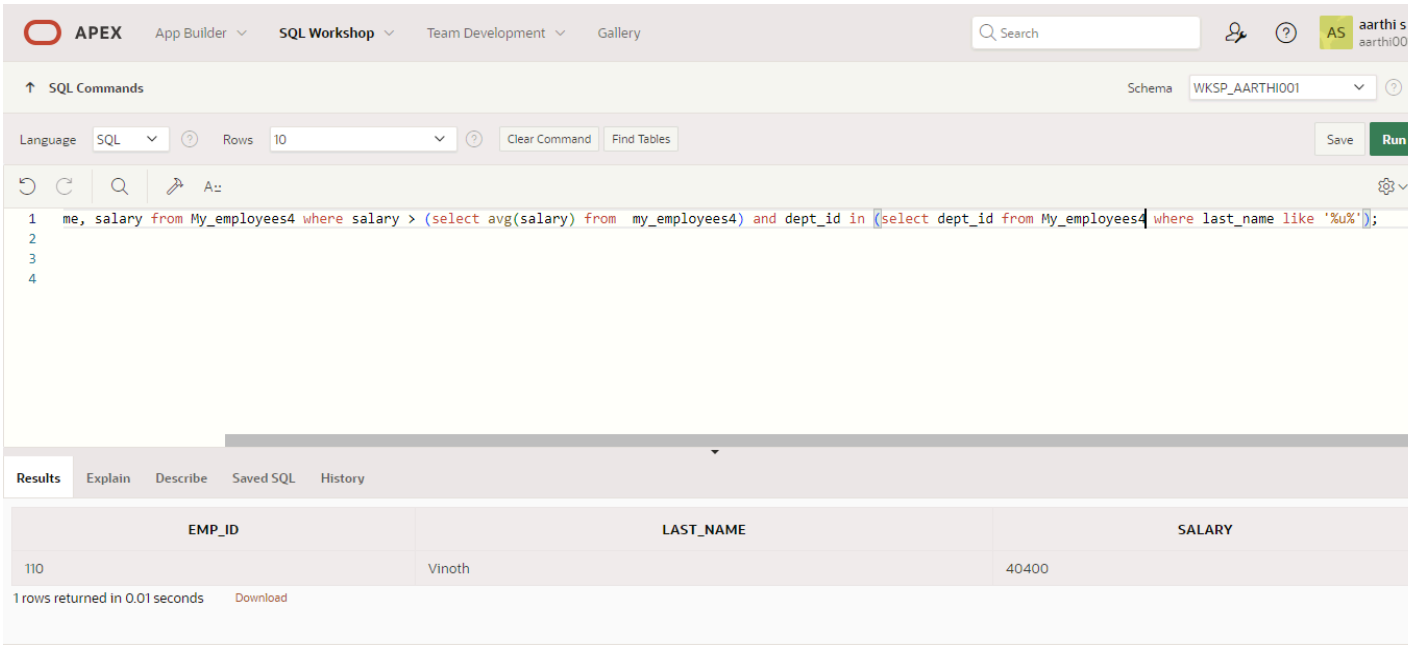
OUTPUT:



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

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

OUTPUT:

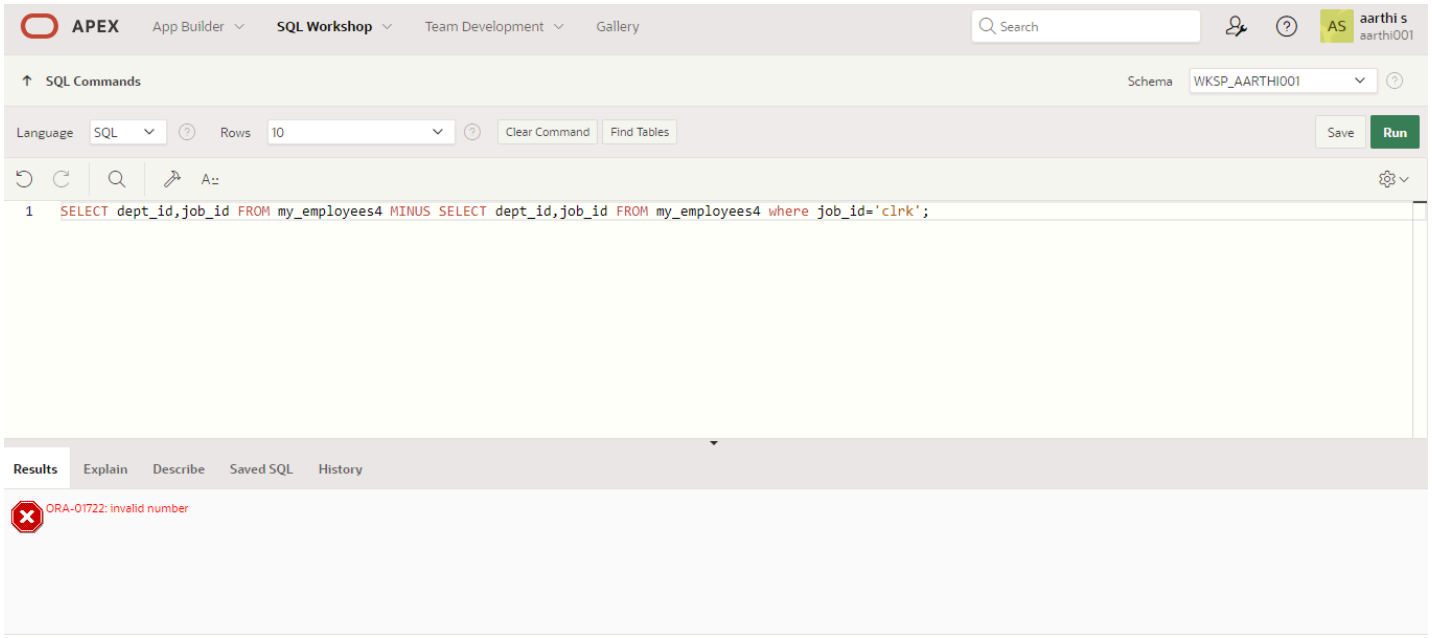


EXNO:10 USING THE SET OPERATORS **DATE**:

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

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

OUTPUT:

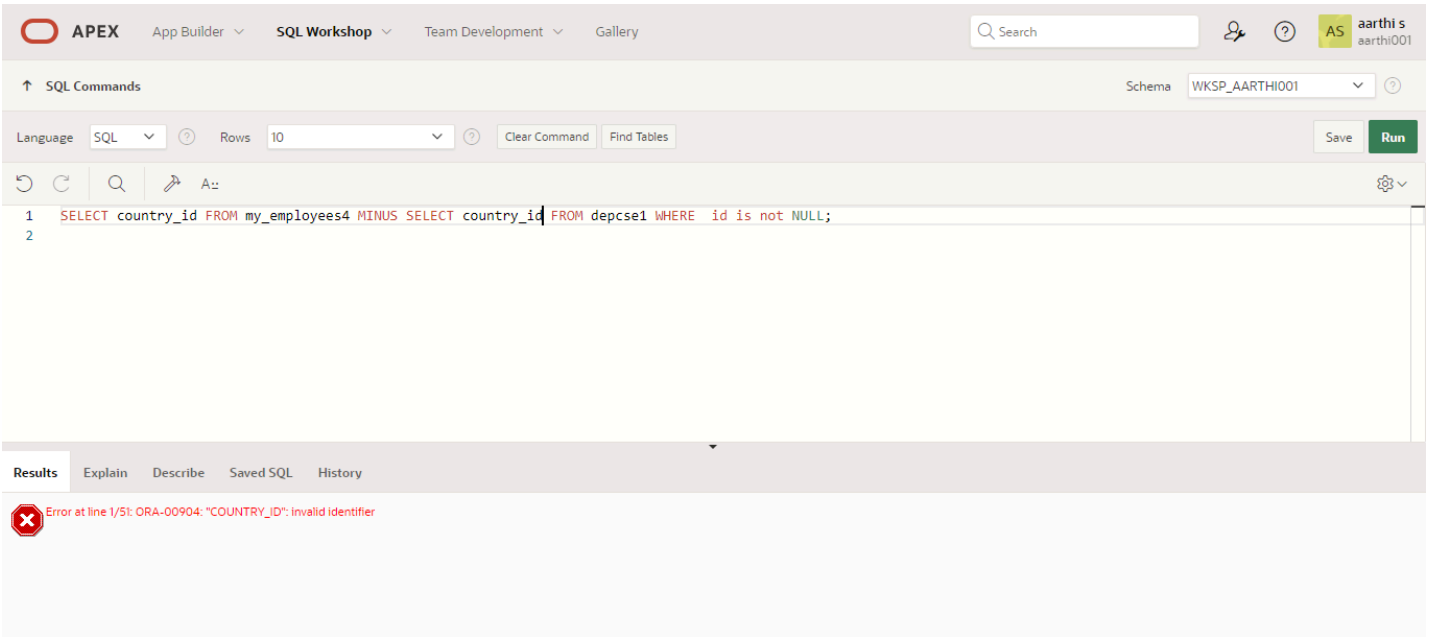


2. The HR department needs a list of countries that have no departments located in them.

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

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

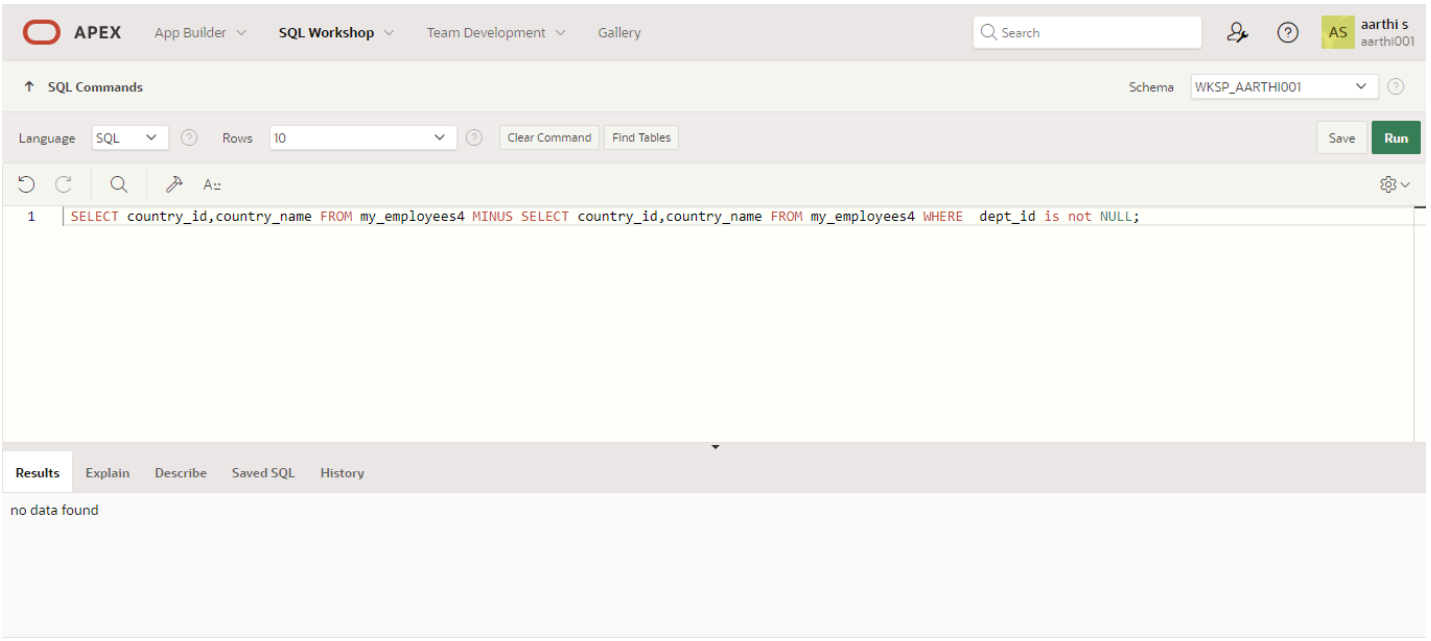
OUTPUT:



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

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

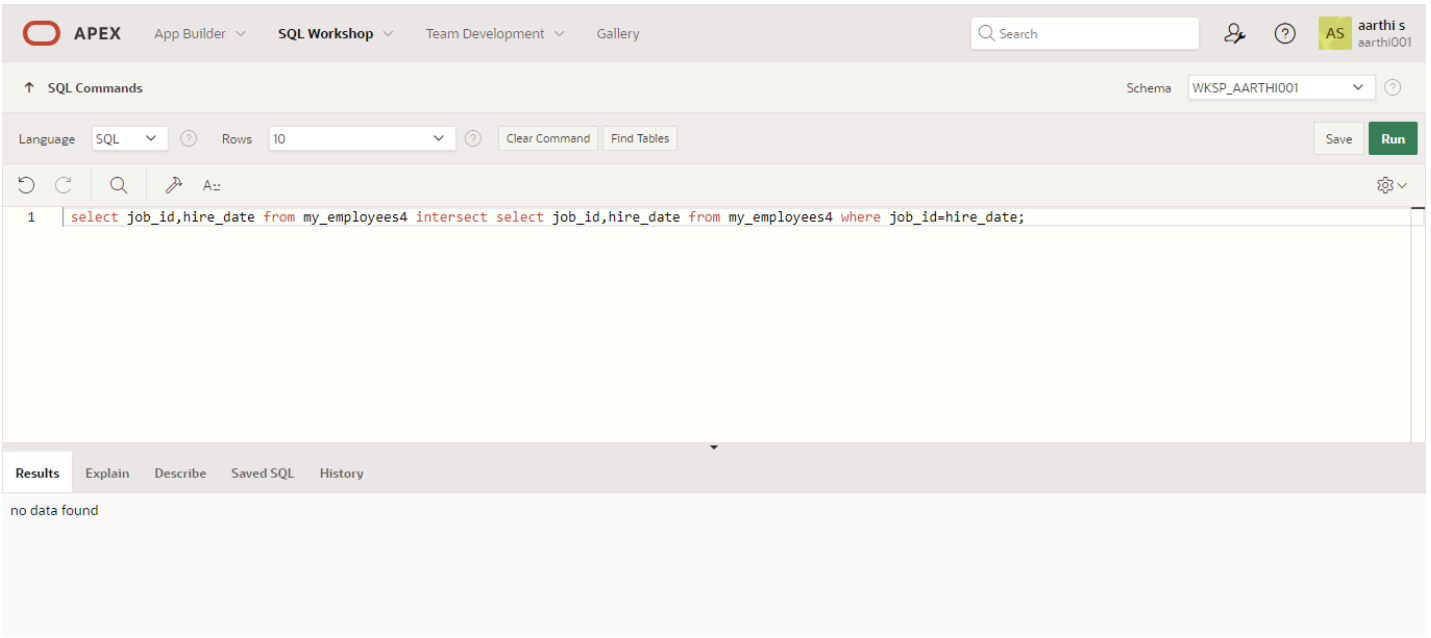
OUTPUT:



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

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

OUTPUT:



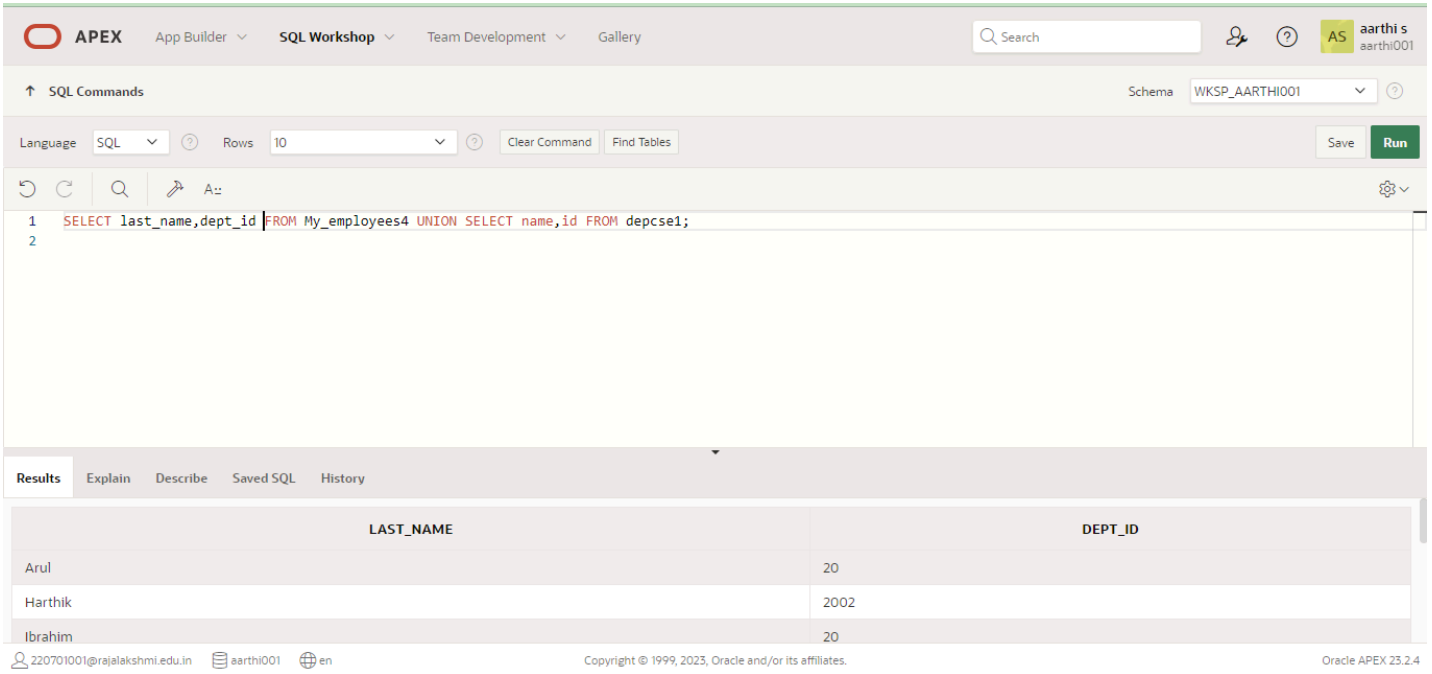
5. The HR department needs a report with the following specifications:

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

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

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

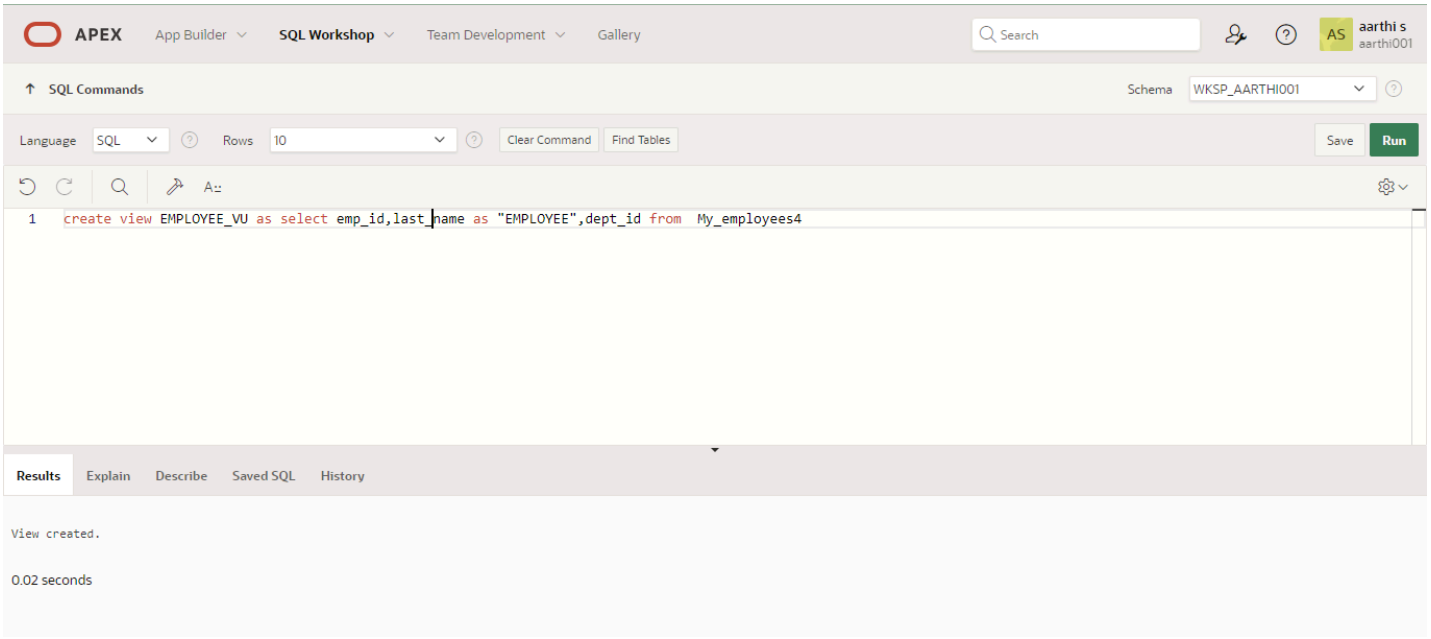
OUTPUT:



EXNO:11 CREATING VIEWS **DATE: 1. Create a view called EMPLOYEE_VU based on the employee numbers, employee names and department numbers from the EMPLOYEES table. Change the heading for the employee name to EMPLOYEE.**

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

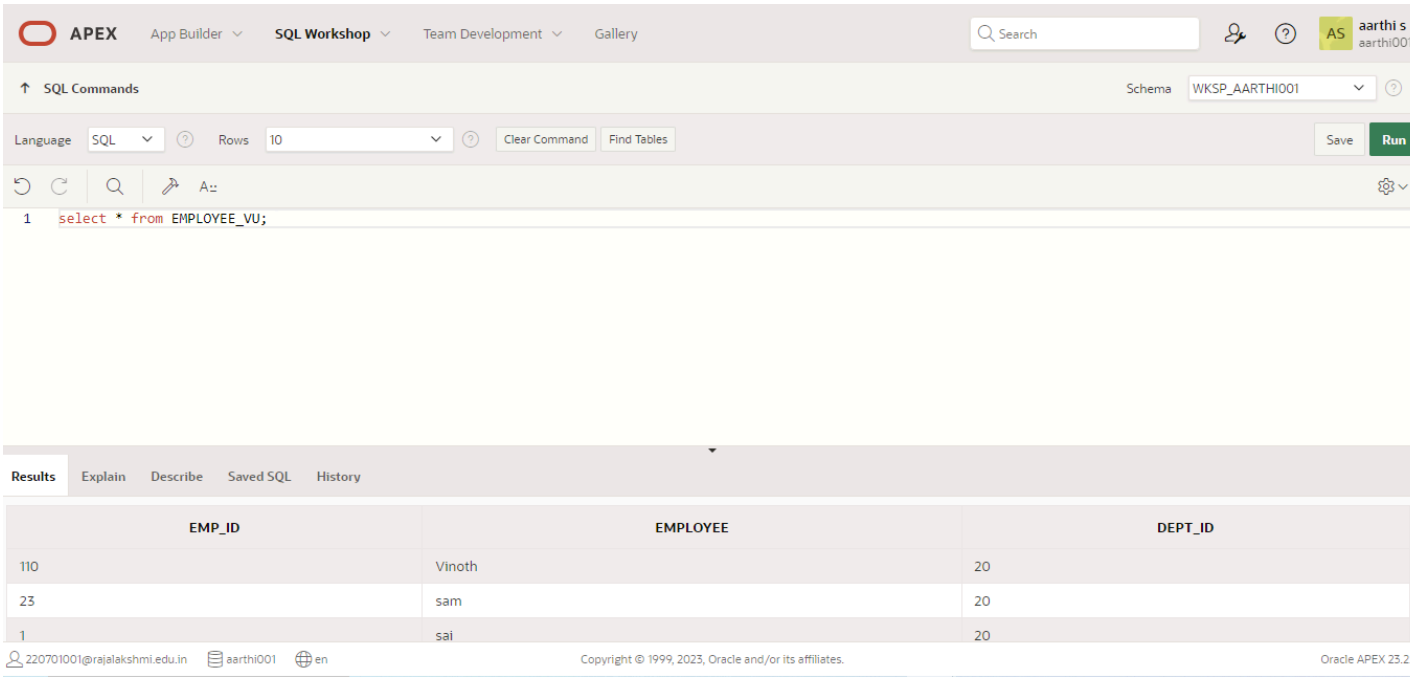
OUTPUT:



2. Display the contents of the EMPLOYEES_VU view.

Query: select * from EMPLOYEE_VU;

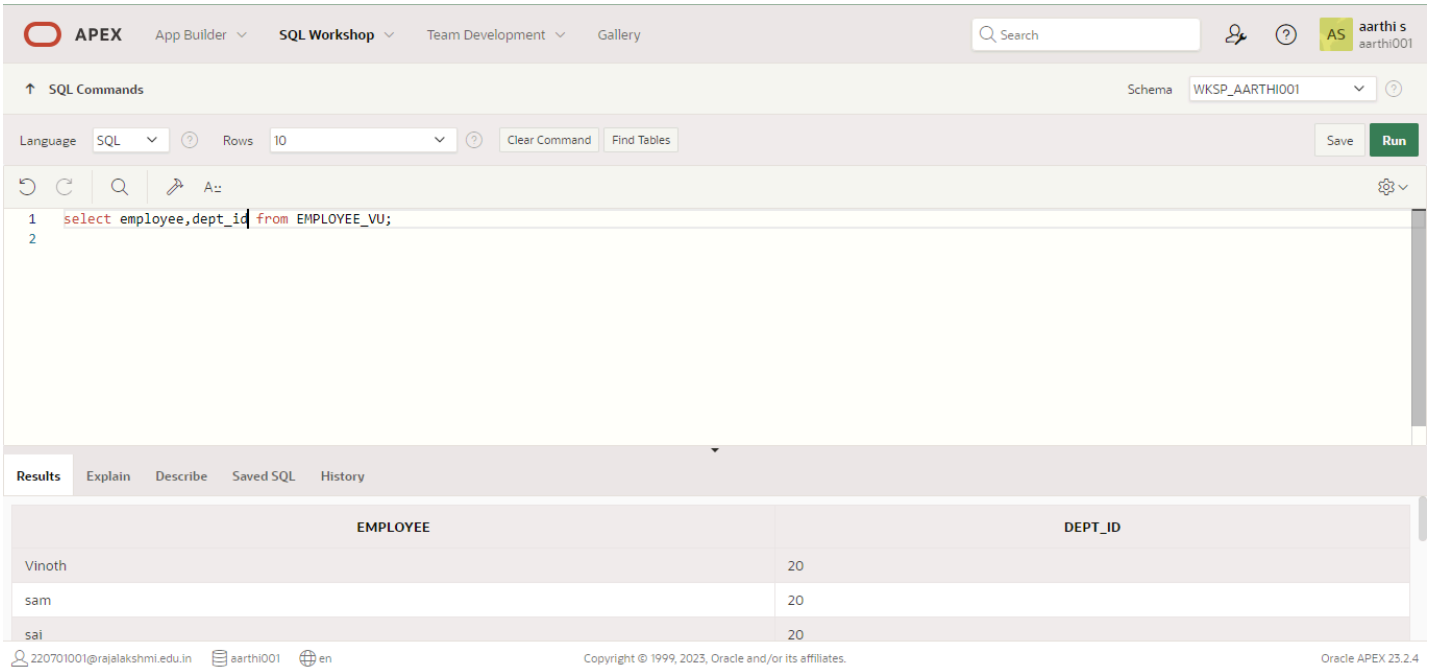
OUTPUT:



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

Query: select employee,id_dept from EMPLOYEE_VU;

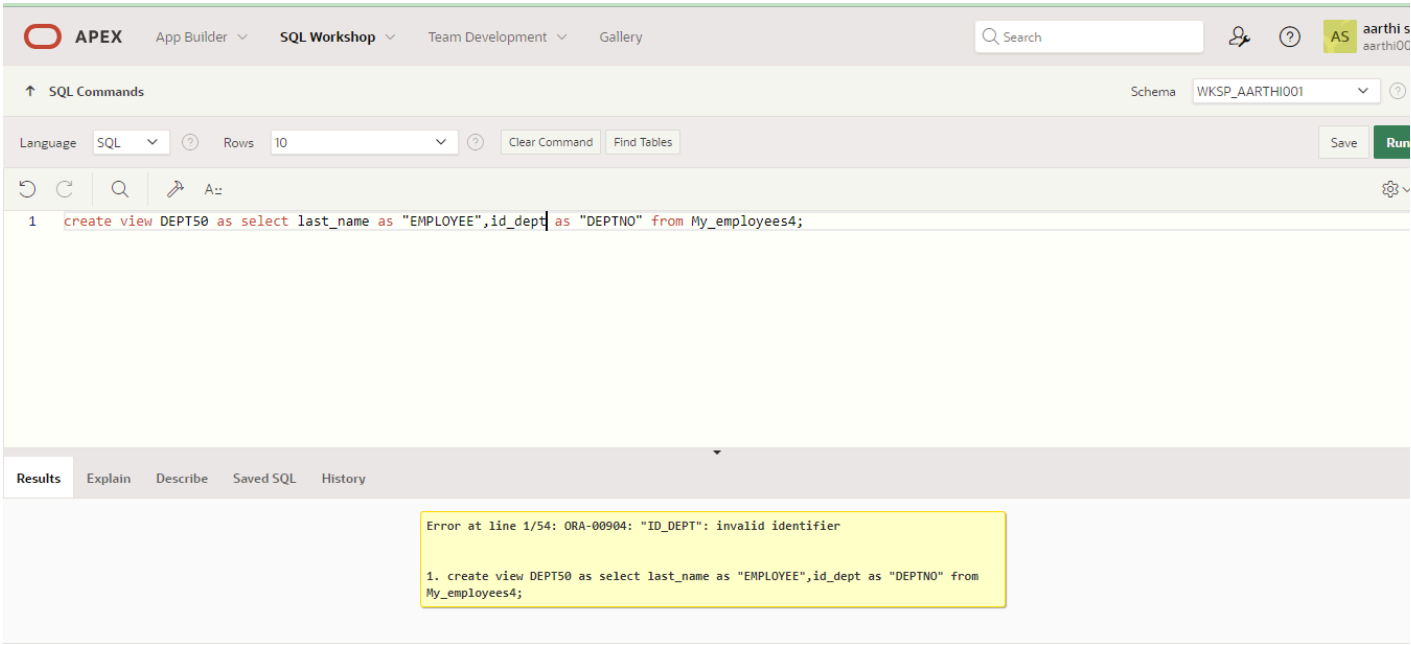
OUTPUT:



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 lname as "EMPLOYEE",id_dept as "DEPTNO" from My_emp;

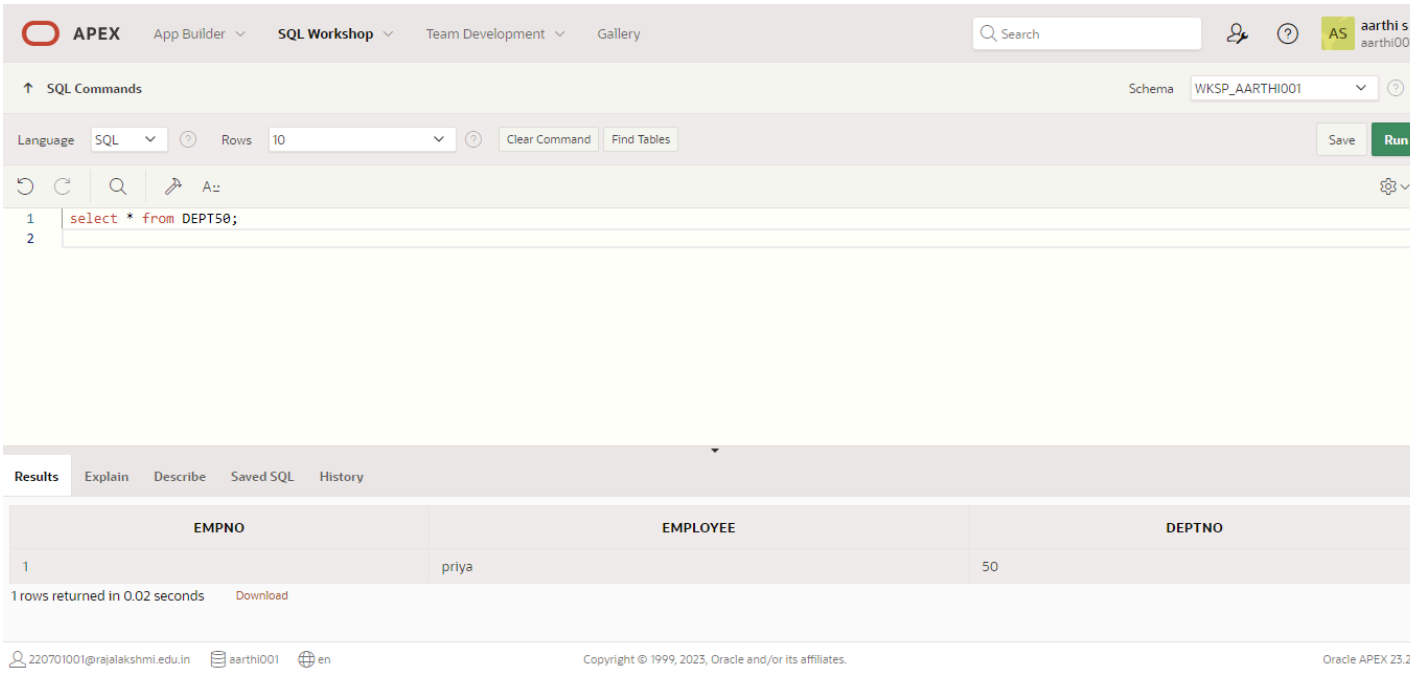
OUTPUT:



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

Query: select * from DEPT50;

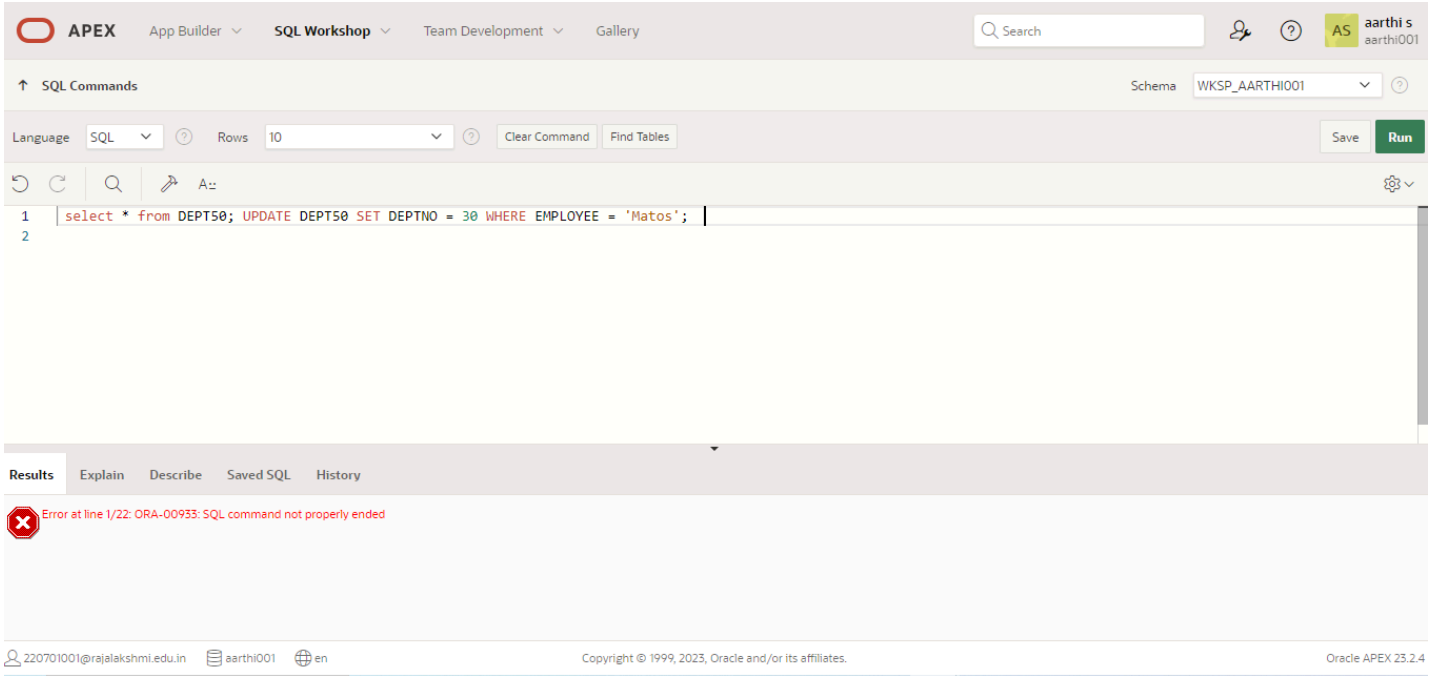
OUTPUT:



6. Attempt to reassign Matos to department 80.

Query: UPDATE DEPT50 SET DEPTNO = 30 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 as "employee",job_title as "DEPARTMENT", salary from My_emp;

OUTPUT:

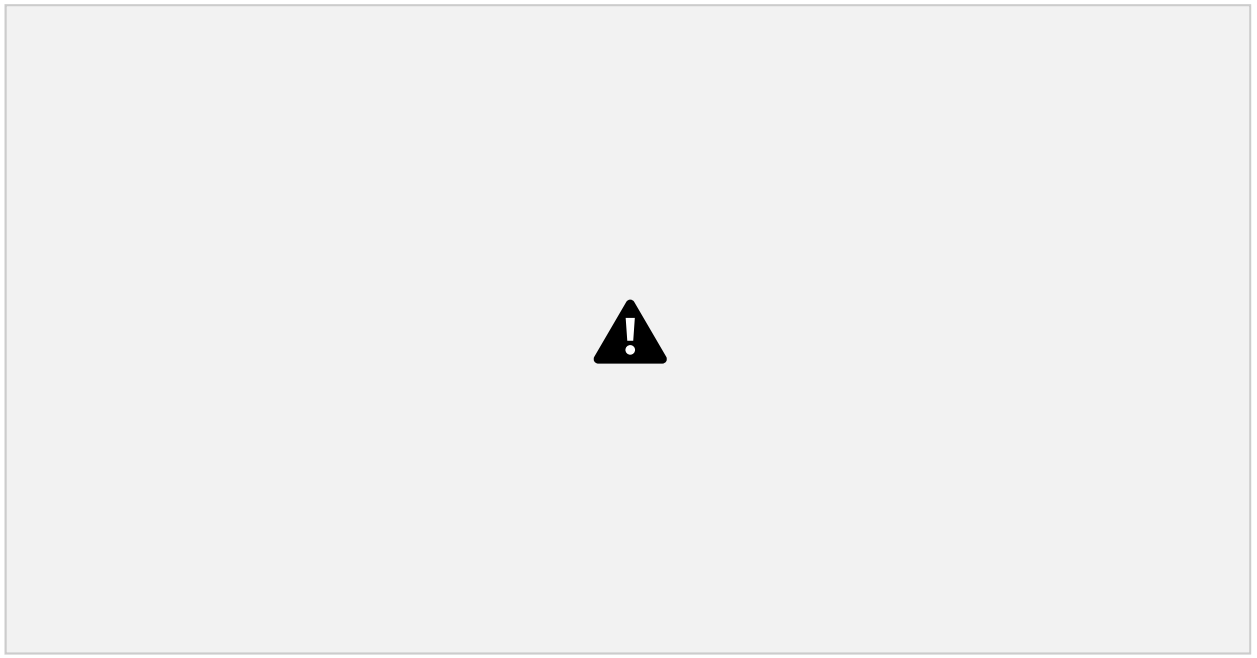
EXNO:13 CREATING VIEWS **DATE:**

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

column.

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

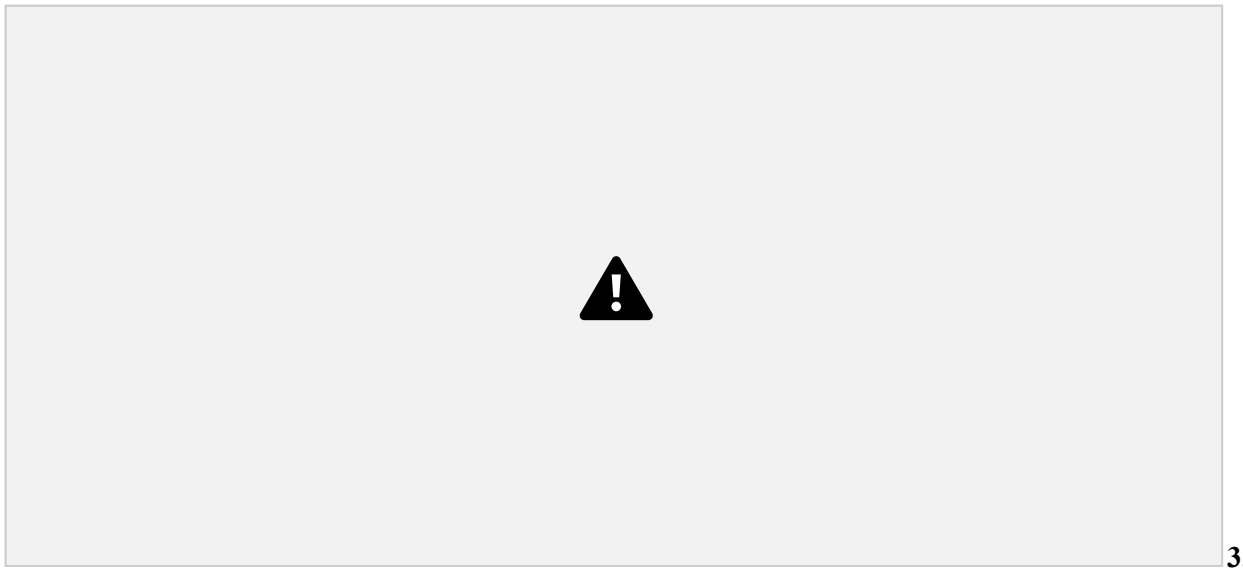
OUTPUT:



2. SELECT * FROM view_d_songs. What was returned?

QUERY: select * from d_songs;

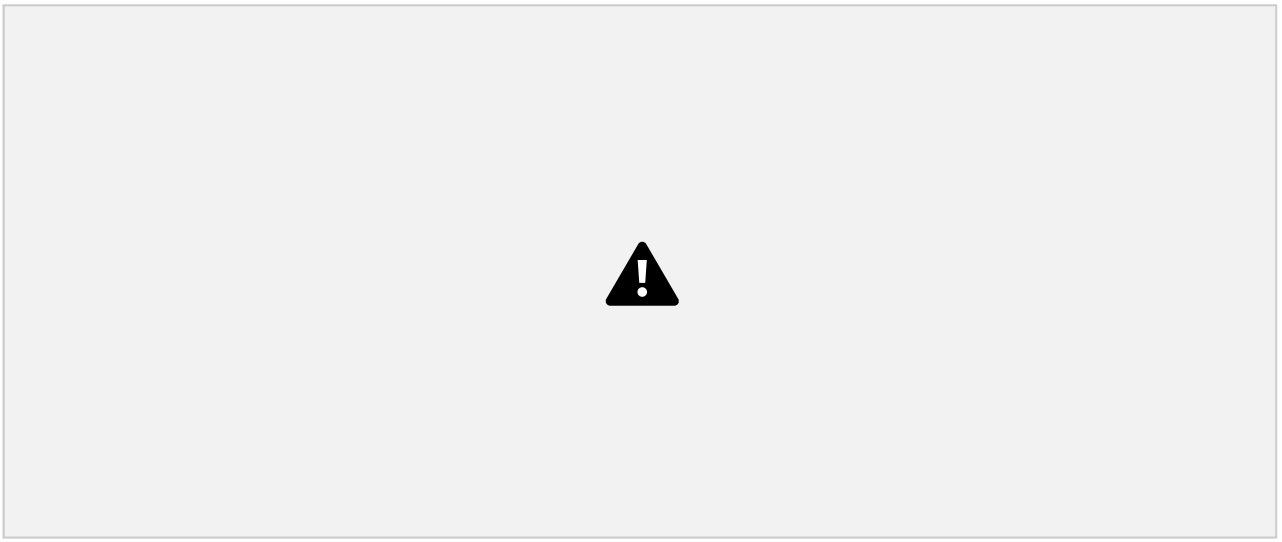
OUTPUT:



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

QUERY: CREATE OR REPLACE VIEW view_d_songs AS SELECT d_songs.id, d_songs.title "Song Title", d_songs.artist, d_songs.type_code from d_songs;

OUTPUT:



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

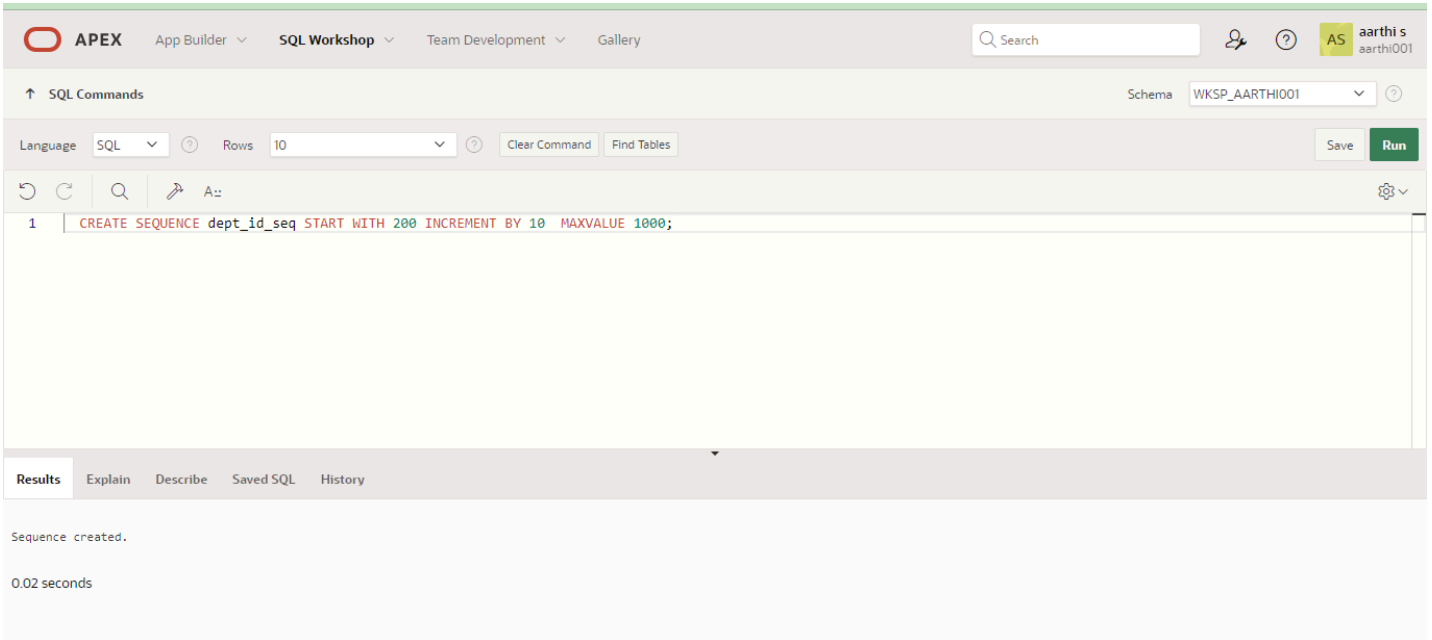
QUERY: CREATE OR REPLACE VIEW view_d_events_pkgs AS SELECT d_name as "Name of Event", TO_CHAR(d_date, 'dd-Month-yyyy') as "Event date", theme as "Theme description" FROM d_events;

OUTPUT:

EXNO:14 OTHER DATABASE OBJECTS **DATE:**

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

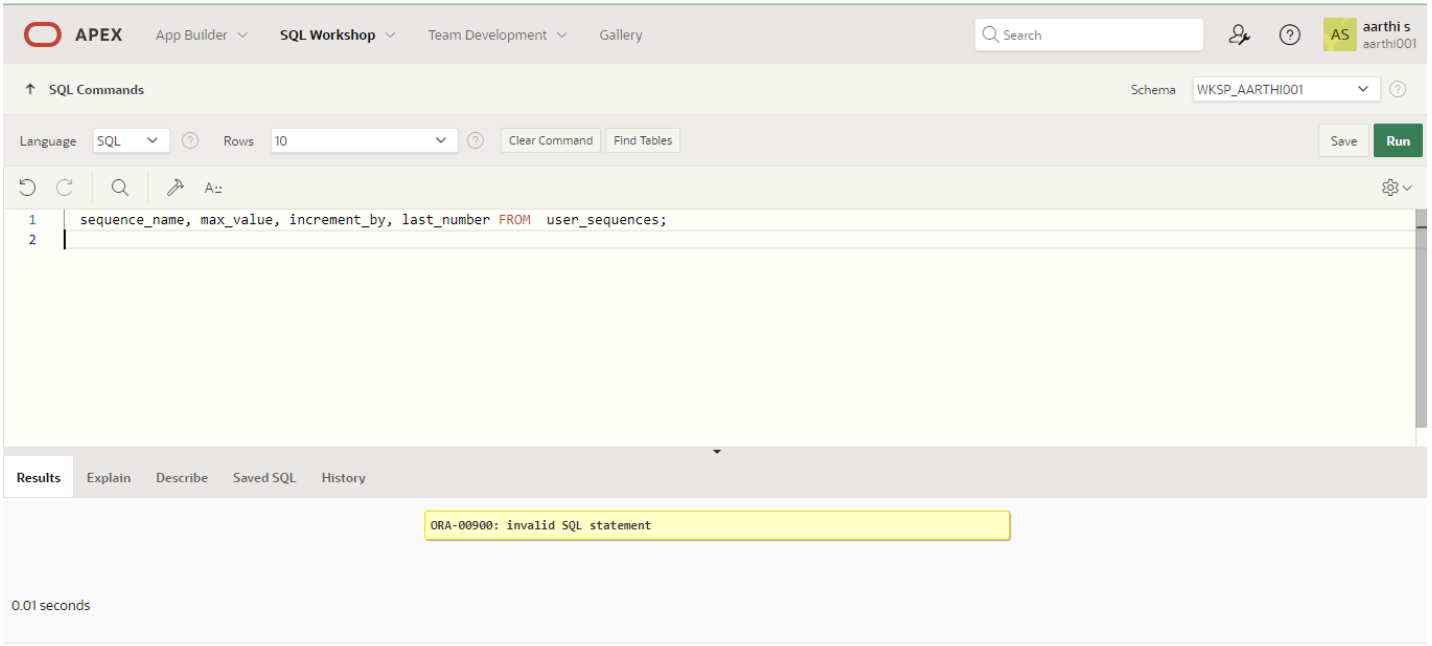
Query: CREATE SEQUENCE dept_id_seq START WITH 200 INCREMENT BY 10 MAXVALUE 1000;
OUTPUT:



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

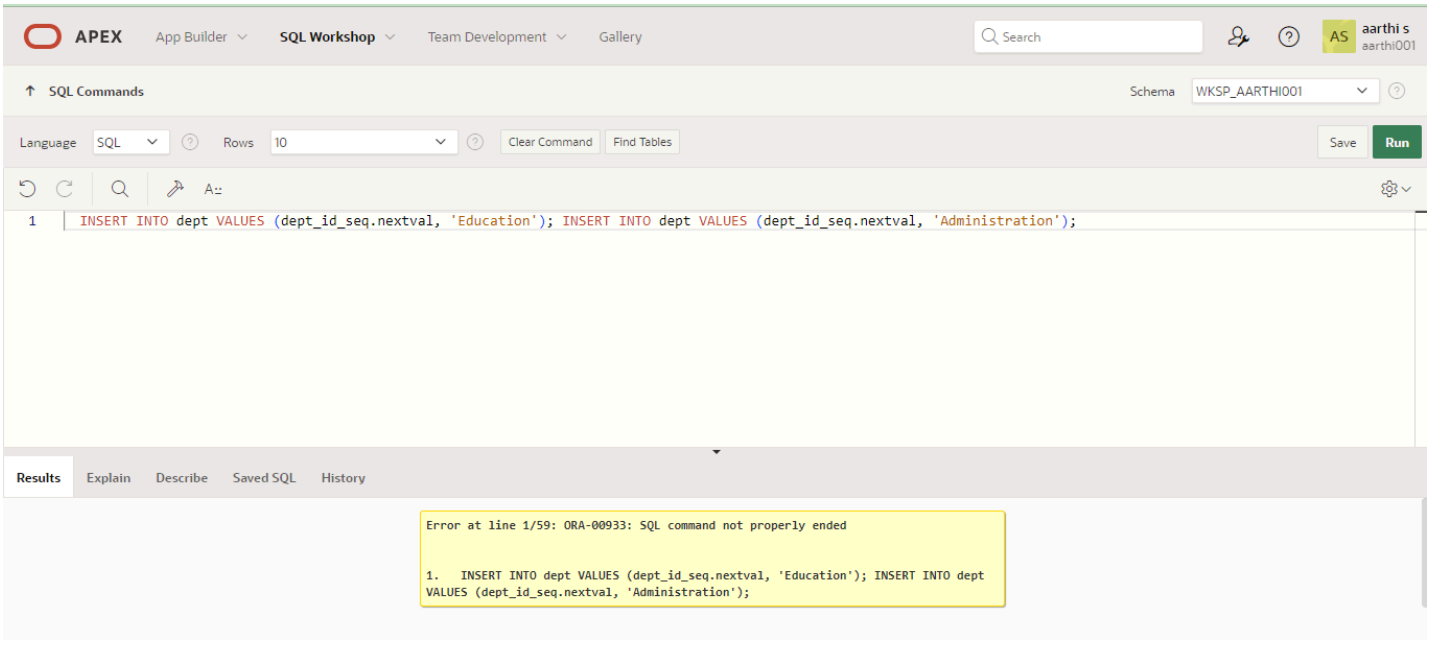
QUERY: SELECT sequence_name, max_value, increment_by, last_number FROM

user_sequences;
OUTPUT:

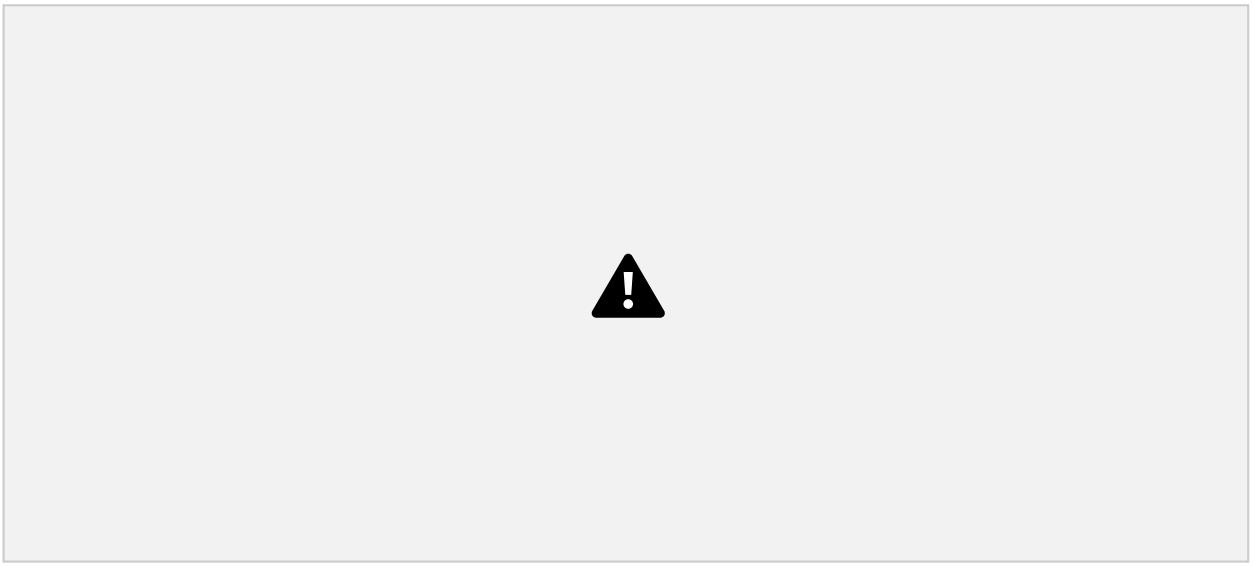


3. Write a script to insert two rows into the DEPT table. Name your script lab12_3.sql. Be sure to use the sequence that you created for the ID column. Add two departments named Education and Administration. Confirm your additions. Run the commands in your script.

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 emp (dept_id); **OUTPUT:**



5. Display the indexes and uniqueness that exist in the data dictionary for the EMP table. QUERY:
SELECT index_name, table_name, uniqueness FROM user_indexes WHERE table_name = 'EMP';

OUTPUT:

Exno:15 CONTROLLING USER ACCESS date:

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

2. What privilege should a user be given to create tables?
Ans : The CREATE TABLE privilege

3. If you create a table, who can pass along privileges to other users on your table?
Ans: You can, or anyone you have given those privileges to by using the WITH GRANT OPTION.

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

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

5. What command do you use to change your password?
Ans: The ALTER USER statement

6. Grant another user access to your DEPARTMENTS table. Have the user grant you query access to his or her DEPARTMENTS table. Ans: Team 2 executes the GRANT statement.
GRANT select
ON departments
TO <user1>;

Team 1 executes the GRANT statement.
GRANT select
ON departments
TO <user2>;
WHERE user1 is the name of team 1 and user2 is the name of team 2. **7. Query all the rows in your DEPARTMENTS table.**
Ans: SELECT * FROM departments;
8. Add a new row to your DEPARTMENTS table. Team 1 should add Education as department number 500. Team 2 should add Human Resources department number 510. Query the other team's table.

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

Team 2 executes this INSERT statement.
INSERT INTO departments(department_id, department_name) VALUES (510,
'Administration');
COMMIT;
9. Create a synonym for the other team's DEPARTMENTS table.

Ans: Team 1 creates a synonym named team2.
CREATE SYNONYM team2
FOR <user2>.DEPARTMENTS;

Team 2 creates a synonym named team1.
CREATE SYNONYM team1
FOR <user1>. DEPARTMENTS;

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

Ans: Team 1 executes this SELECT statement.
SELECT *
FROM team2;

Team 2 executes this SELECT statement.
SELECT *
FROM team1;

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

Ans: SELECT table_name FROM user_tables;
12. Query the ALL_TABLES data dictionary view to see information about all the tables that you
can access. Exclude tables that you own.

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

13. Revoke the SELECT privilege from the other team.

Ans: Team 1 revokes the privilege.

REVOKE select
ON departments
FROM user2;

Team 2 revokes the privilege.

REVOKE select
ON departments
FROM user1;

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

Team 1 executes this INSERT statement.
Ans:
DELETE FROM departments
WHERE department_id = 500;
COMMIT;

Team 2 executes this INSERT statement.

DELETE FROM departments
WHERE department_id = 510;
COMMIT;

EXNO:16 PL/SQL DATE: CONTROL STRUCTURES

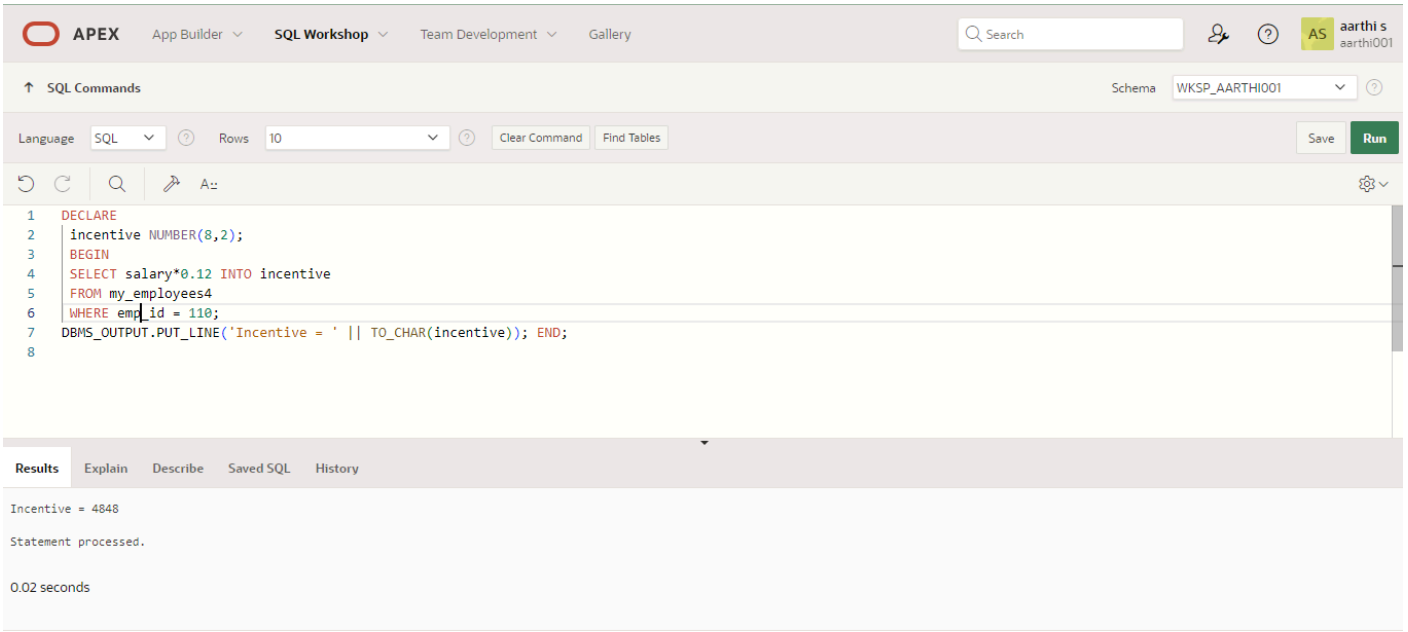
1.Write a PL/SQL block to calculate the incentive of an employee whose ID is 110.

QUERY:

```
 DECLARE
incentive NUMBER(8,2);
BEGIN
SELECT salary*0.12 INTO incentive
FROM employees
WHERE employee_id = 110;
```

DBMS_OUTPUT.PUT_LINE('Incentive = ' || TO_CHAR(incentive));
END;

OUTPUT:



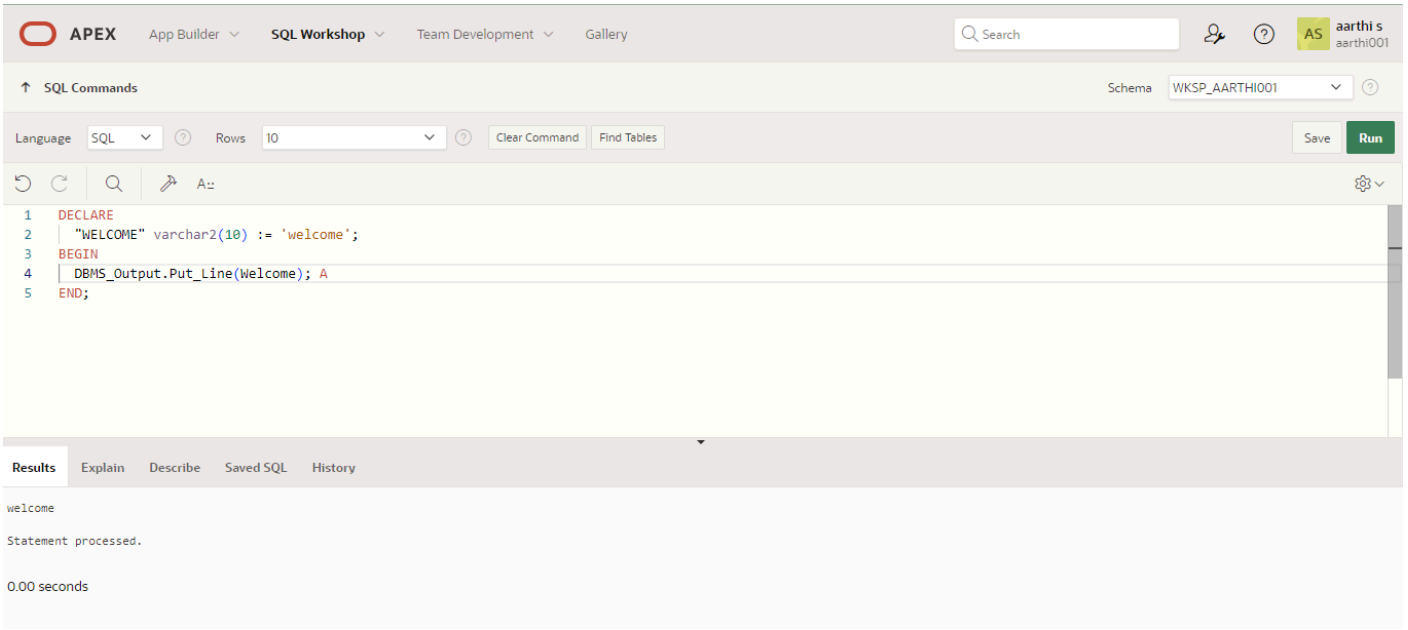
2.) Write a PL/SQL block to show an invalid case-insensitive reference to a quoted and without quoted user-defined identifier

QUERY:

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

/

OUTPUT:

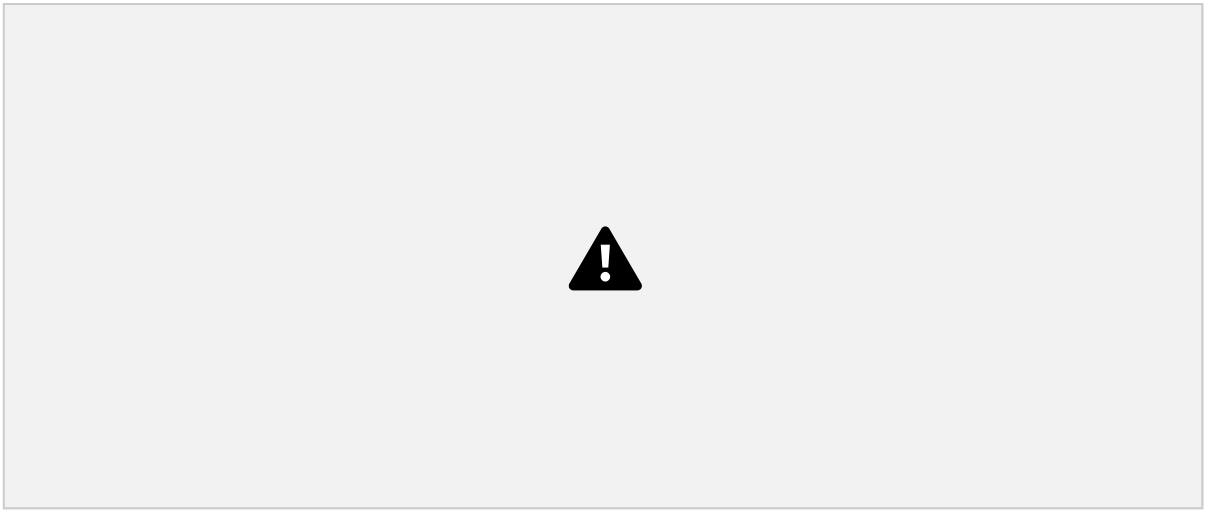


3.) Write a PL/SQL block to adjust the salary of the employee whose ID 122.

QUERY:

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

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

OUTPUT:

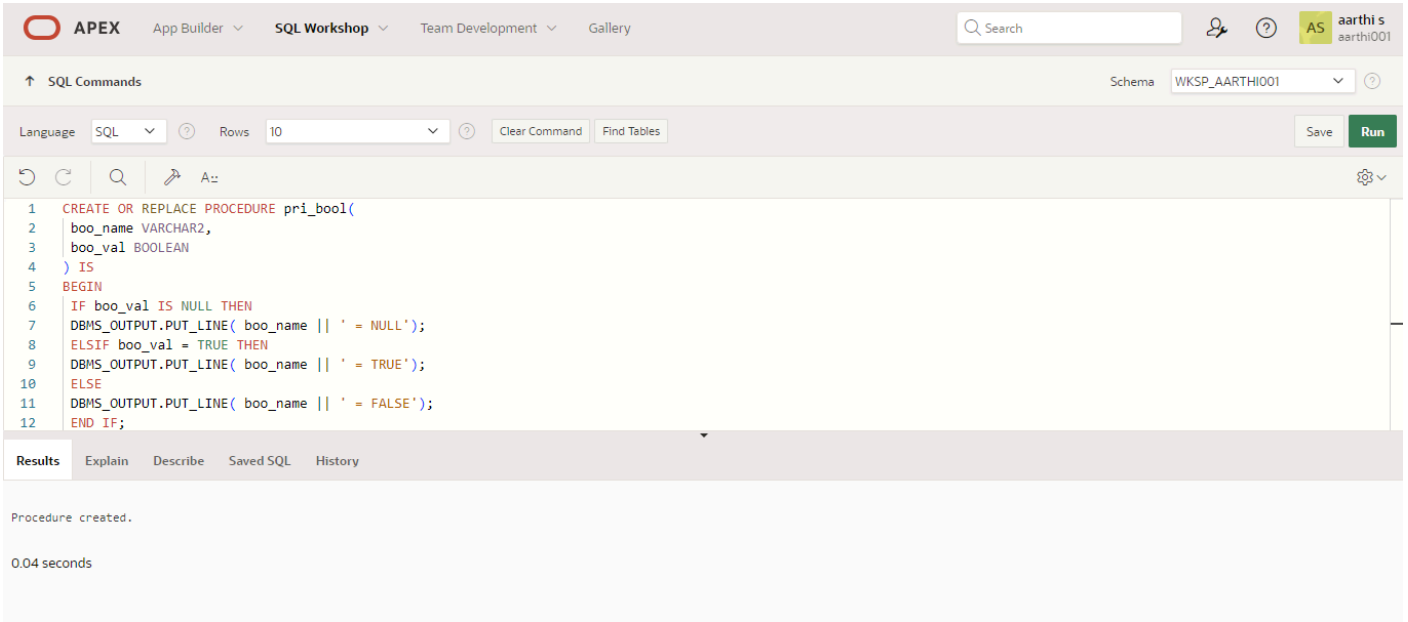


4.) Write a PL/SQL block to create a procedure using the "IS [NOT] NULL Operator" and show AND operator returns TRUE if and only if both operands are TRUE.

QUERY:

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

OUTPUT:

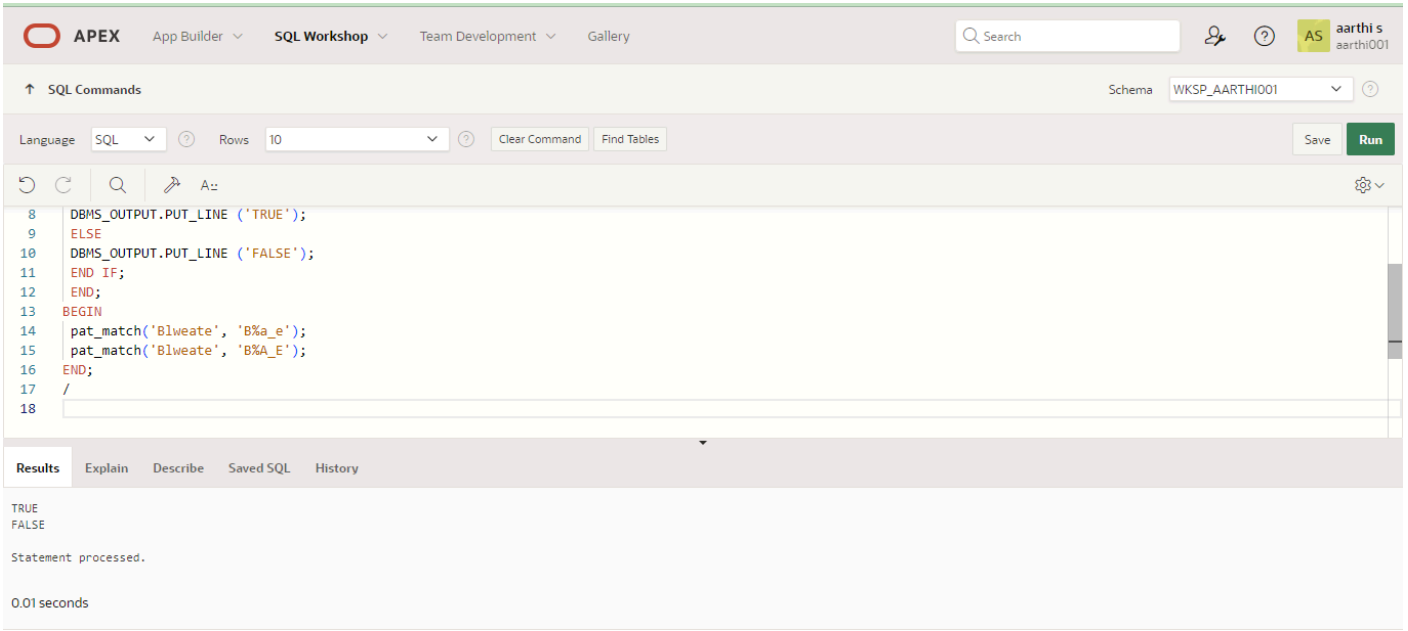


5.) Write a PL/SQL block to describe the usage of LIKE operator including wildcard characters and escape character.

QUERY:

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

OUTPUT:

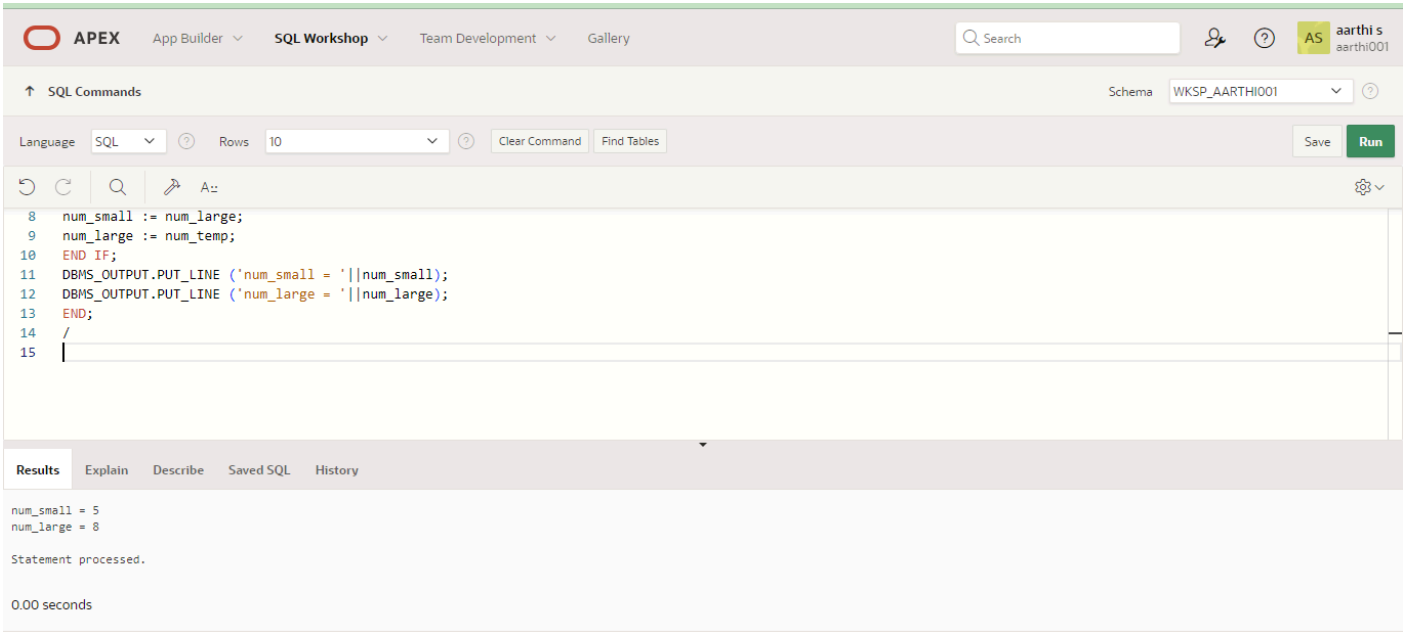


6.) Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num_small variable and large number will store in num_large variable

QUERY:

```
DECLARE
num_small NUMBER := 8;
num_large NUMBER := 5;
num_temp NUMBER;
BEGIN

IF num_small > num_large THEN
num_temp := num_small;
num_small := num_large;
num_large := num_temp;
END IF;

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

OUTPUT:



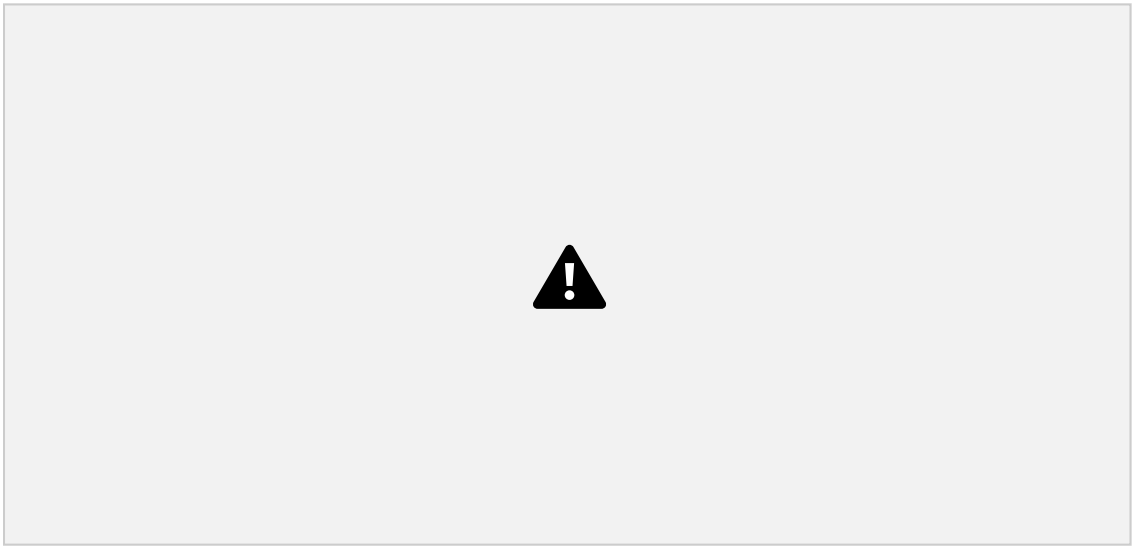
7.) Write a PL/SQL procedure to calculate the incentive on a target achieved and display the message either the record updated or not.

QUERY:

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

/

OUTPUT:

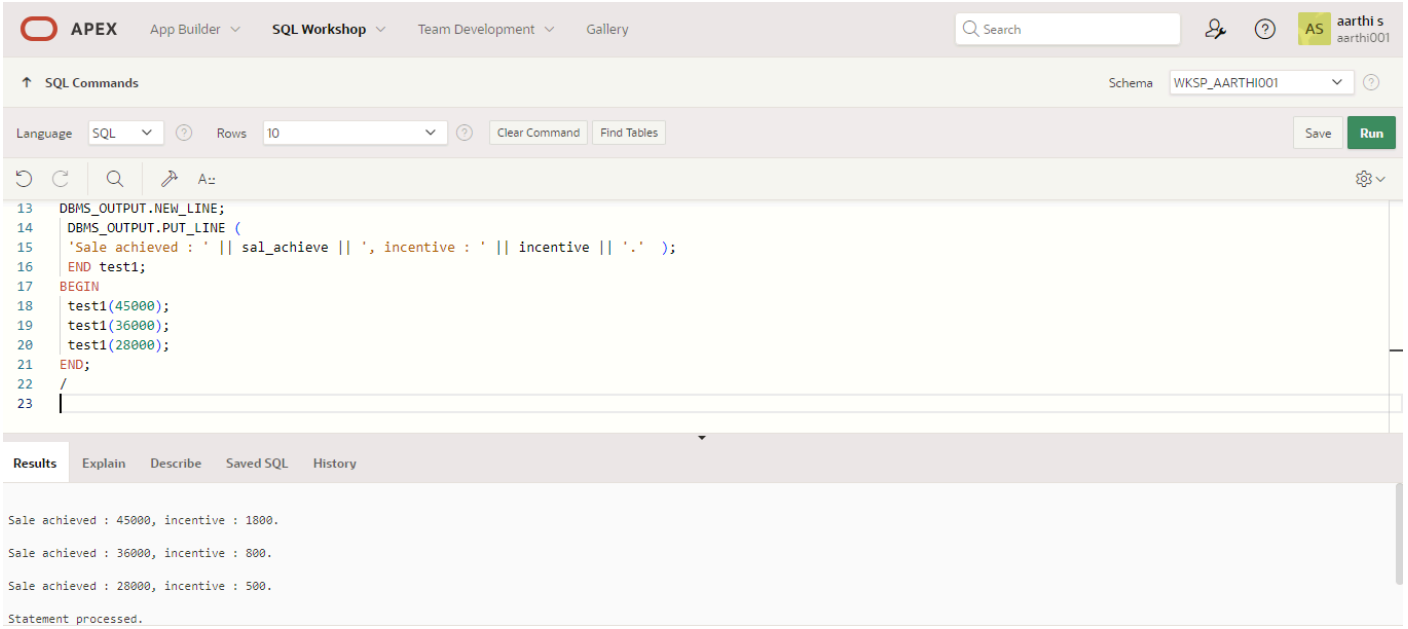


8.) Write a PL/SQL procedure to calculate incentive achieved according to the specific sale limit

QUERY:

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

OUTPUT:



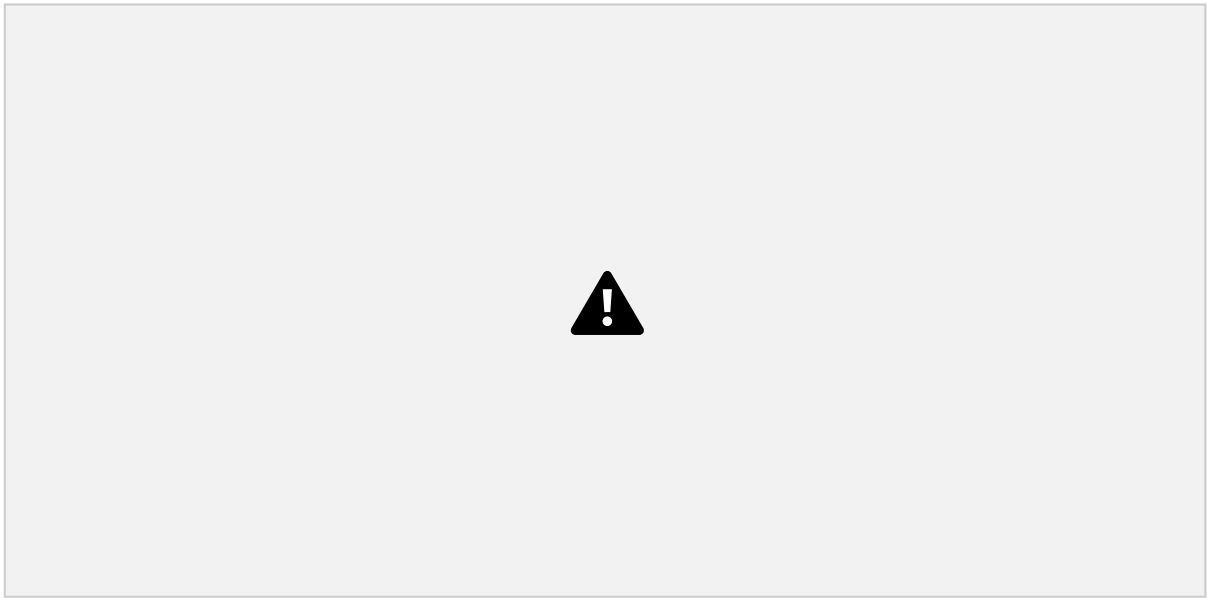
9.) Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

QUERY:

```
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:



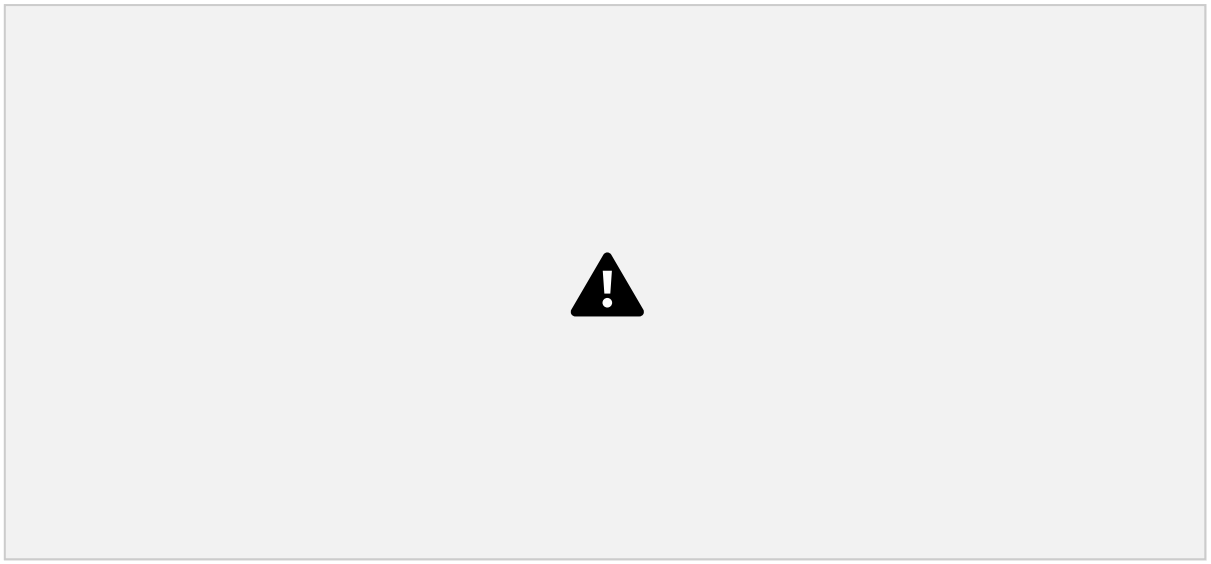
10.) Write a PL/SQL program to count number of employees in a specific department and check whether this department have any vacancies or not. If any vacancies, how many vacancies are in that department.

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:



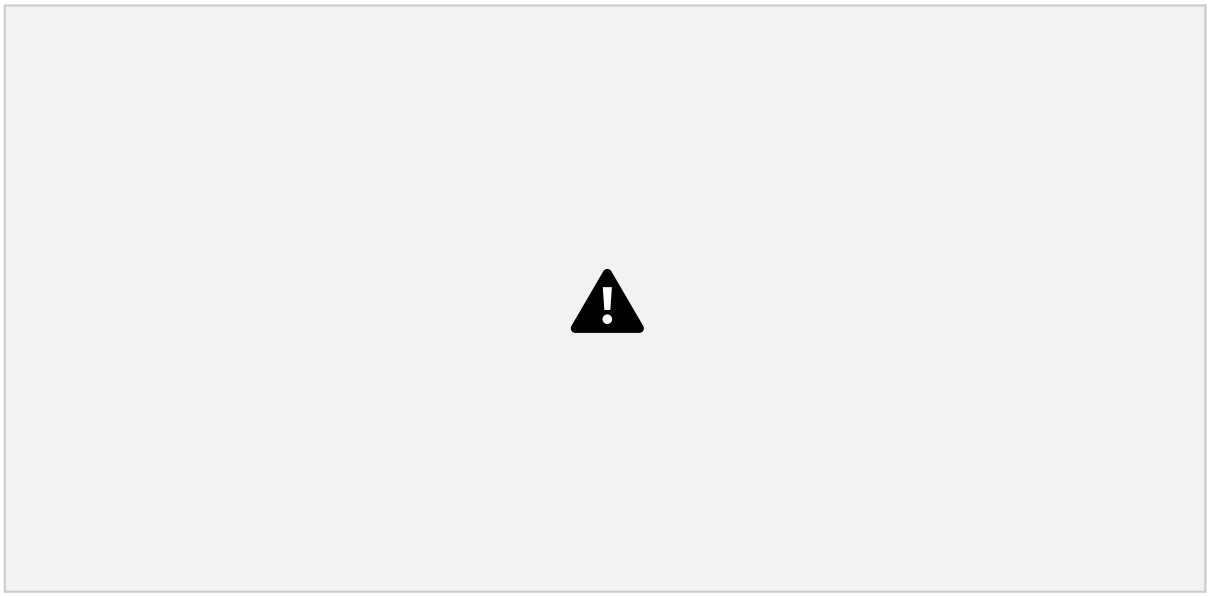
11.) Write a PL/SQL program to display the employee IDs, names, job titles, hire dates, and salaries of all employees

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:

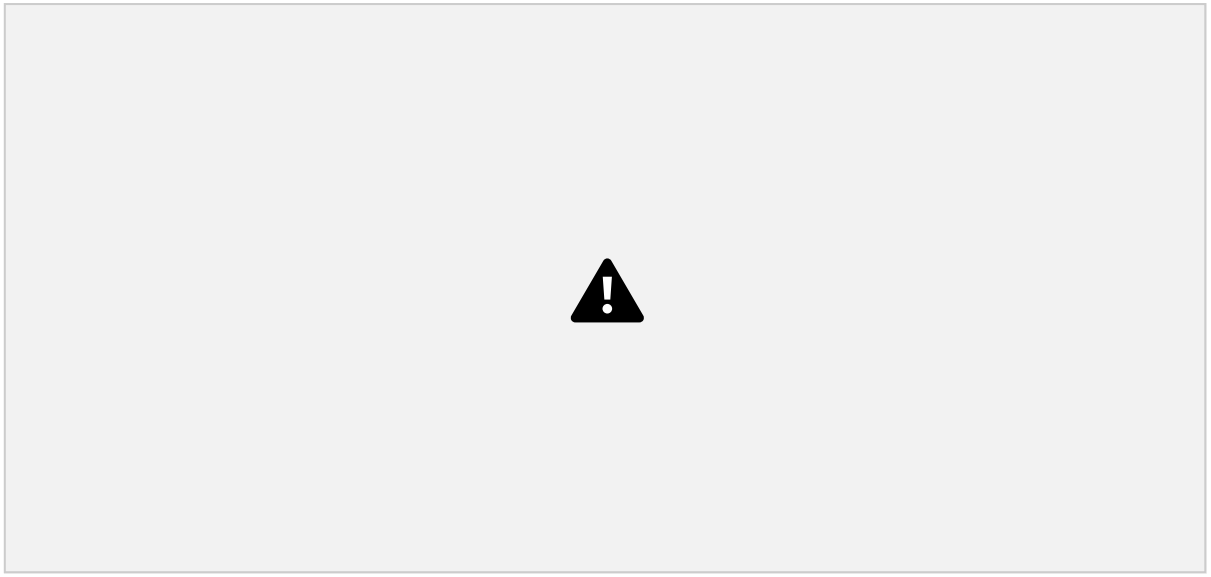


12.) Write a PL/SQL program to display the employee IDs, names, and department names of all employees.

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:



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;
```

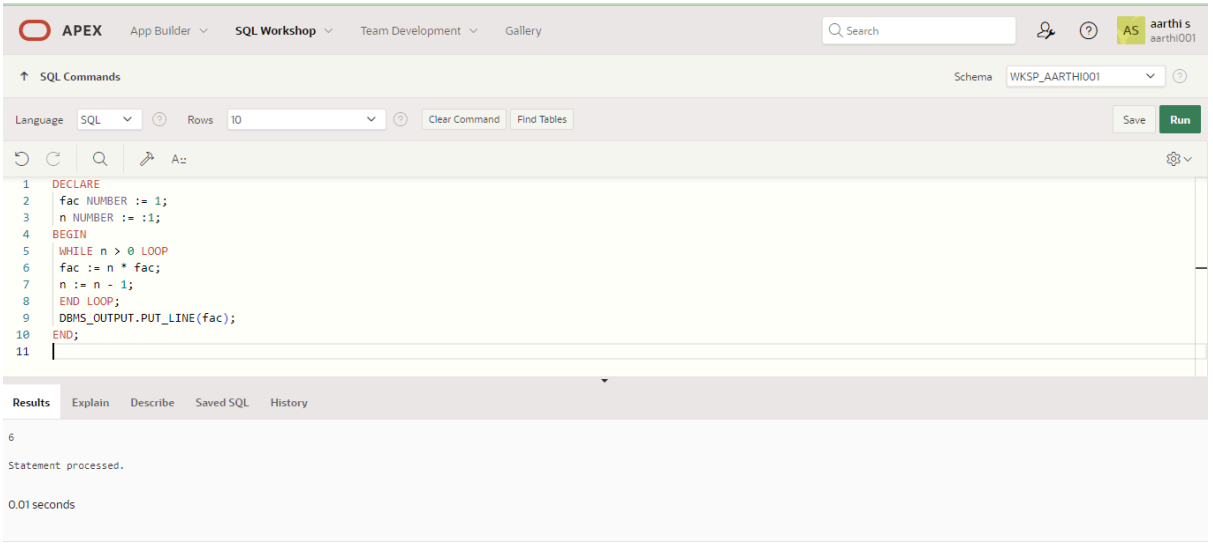
/OUTPUT:

EXNO:17 PROCEDURES AND FUNCTIONS DATE: **1.)Factorial of a number using function.**

QUERY:

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

OUTPUT:

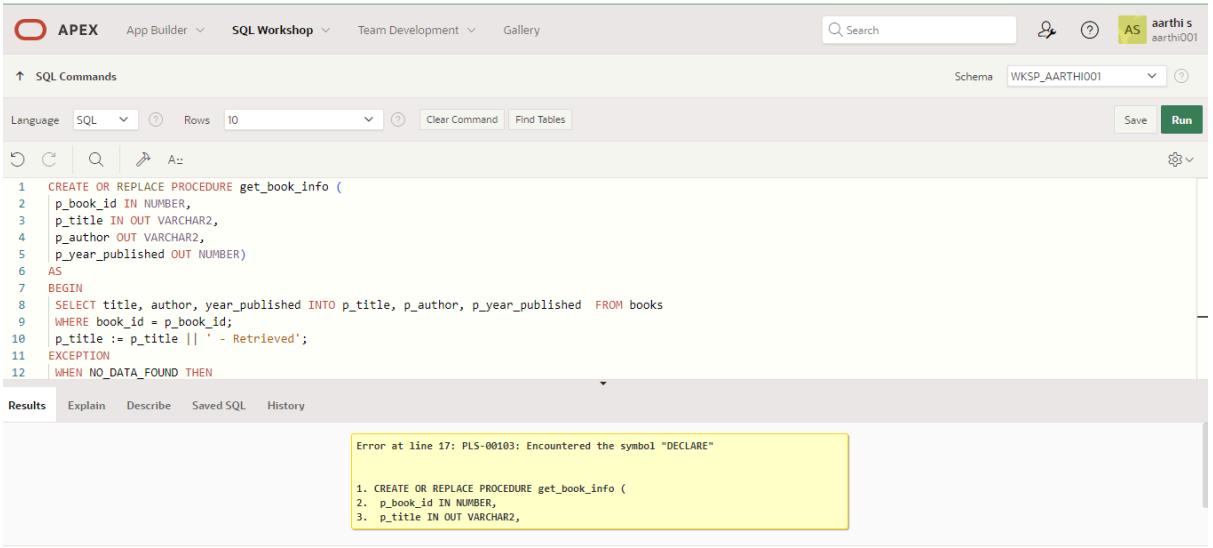


2.) Write a PL/SQL program using Procedures IN,INOUT,OUT parameters to retrieve the corresponding book information in library.

QUERY:

```
CREATE OR REPLACE PROCEDURE get_book_info (
 p_book_id IN NUMBER,
 p_title IN OUT VARCHAR2,
 p_author OUT VARCHAR2,
 p_year_published OUT NUMBER)
AS
BEGIN
 SELECT title, author, year_published INTO p_title, p_author, p_year_published
FROM books
 WHERE book_id = p_book_id;
 p_title := p_title || ' - Retrieved';
EXCEPTION
 WHEN NO_DATA_FOUND THEN
 p_title := NULL;
 p_author := NULL;
 p_year_published := NULL;
END;
DECLARE
 v_book_id NUMBER := 1;
 v_title VARCHAR2(100);
 v_author VARCHAR2(100);
 v_year_published NUMBER;
BEGIN
 v_title := 'Initial Title';
 get_book_info(p_book_id => v_book_id, p_title => v_title, p_author =>
v_author, p_year_published => v_year_published);
 DBMS_OUTPUT.PUT_LINE('Title: ' || v_title);
 DBMS_OUTPUT.PUT_LINE('Author: ' || v_author);
 DBMS_OUTPUT.PUT_LINE('Year Published: ' || v_year_published); END;
```

OUTPUT:



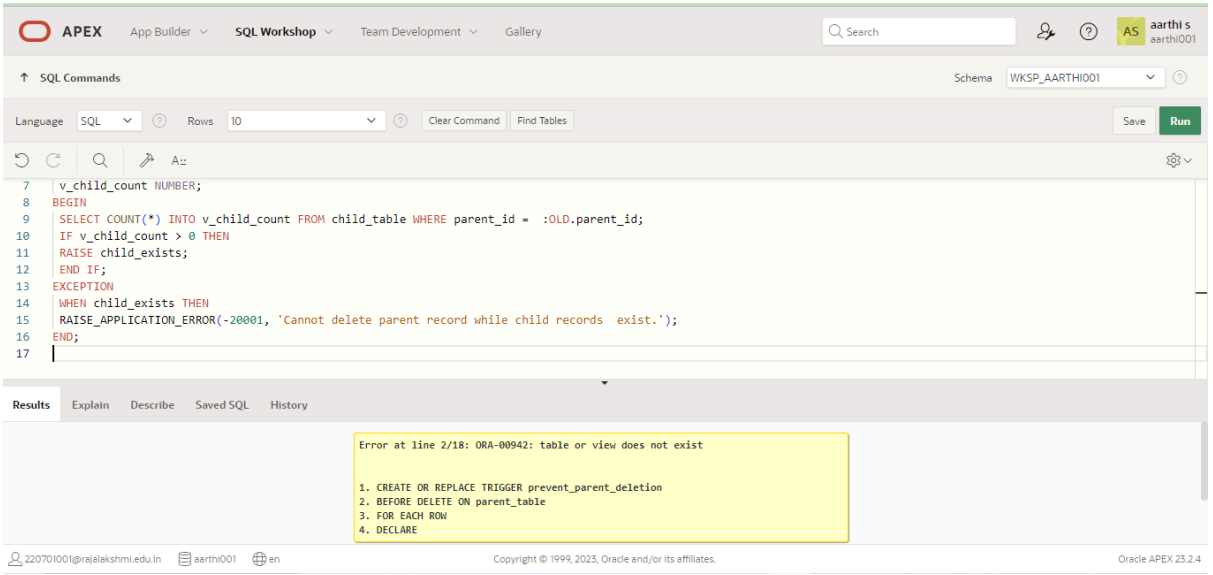
EXNO:18 TRIGGER **DATE:**

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

QUERY:

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

OUTPUT:

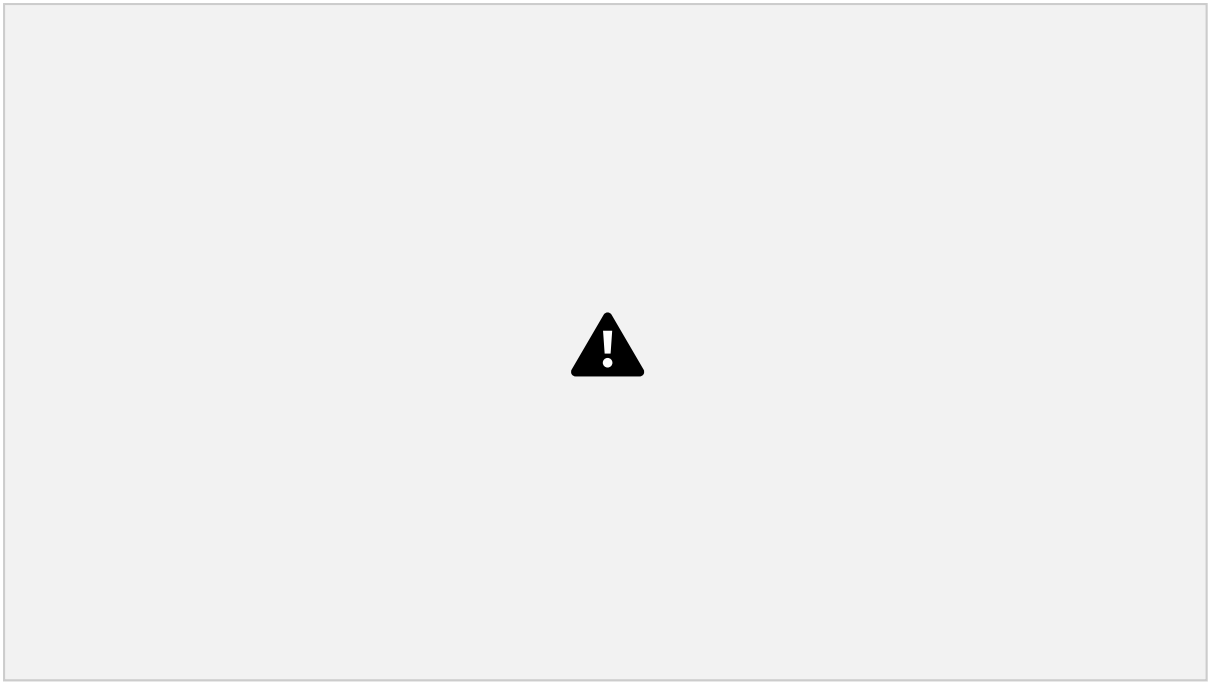


2.) Write a code in PL/SQL to create a trigger that checks for duplicate values in a specific column and raises an exception if found

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:

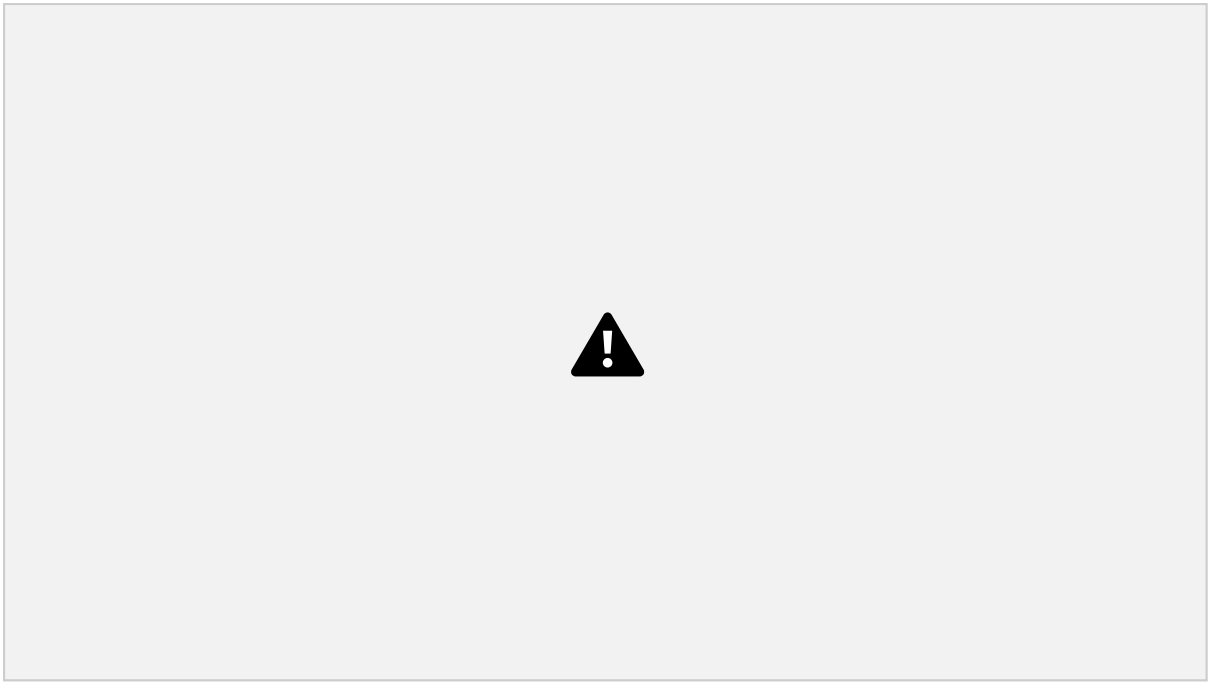


3.) Write a code in PL/SQL to create a trigger that restricts the insertion of new rows if the total of a column's values exceeds a certain threshold

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:

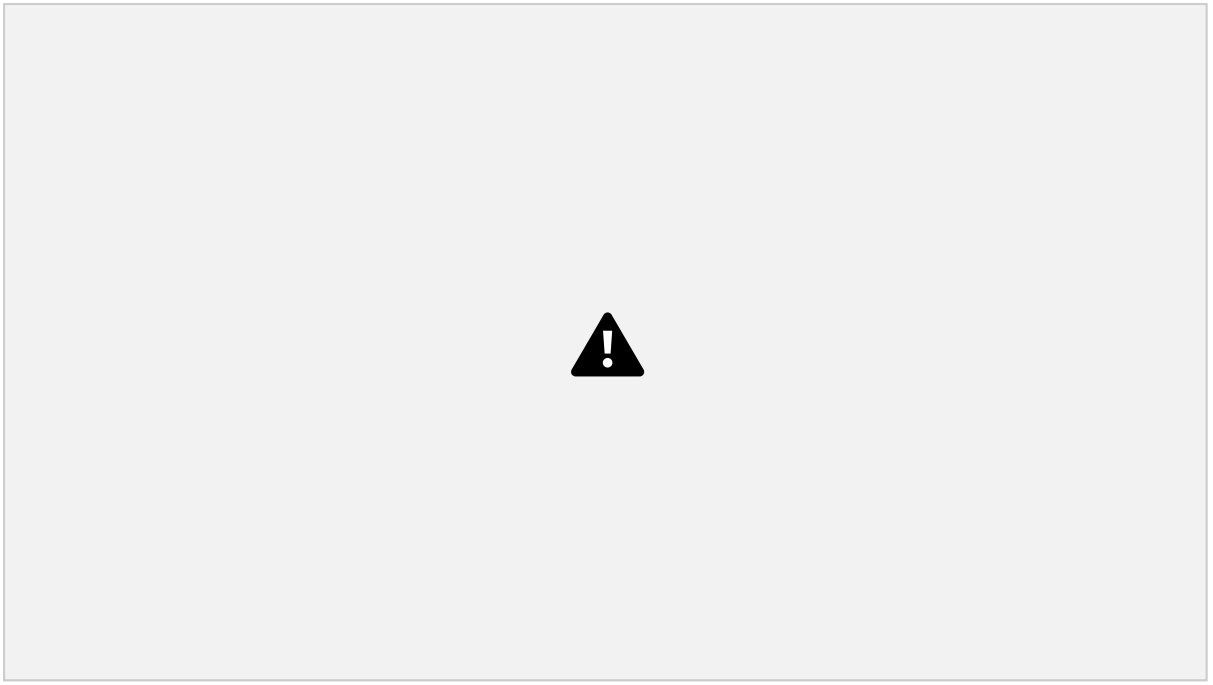


4.) Write a code in PL/SQL to design a trigger that captures changes made to specific columns and logs them in an audit table.

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:

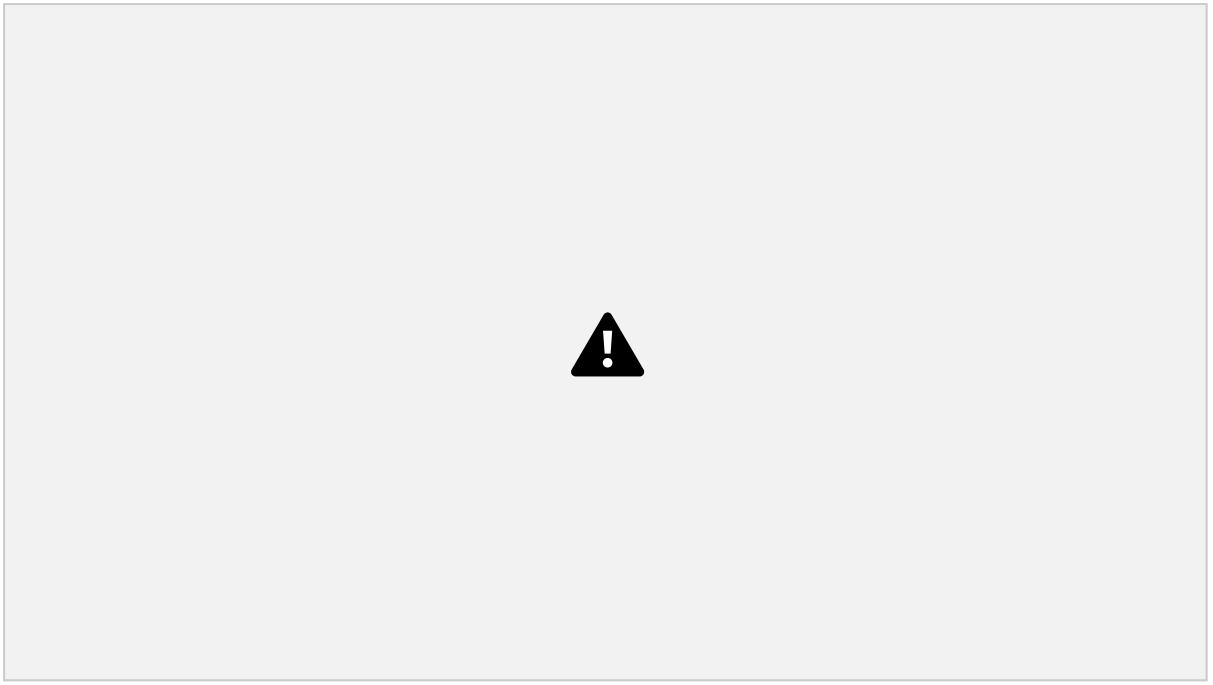


5.) Write a code in PL/SQL to implement a trigger that records user activity (inserts, updates, deletes) in an audit log for a given set of tables.

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:

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 });

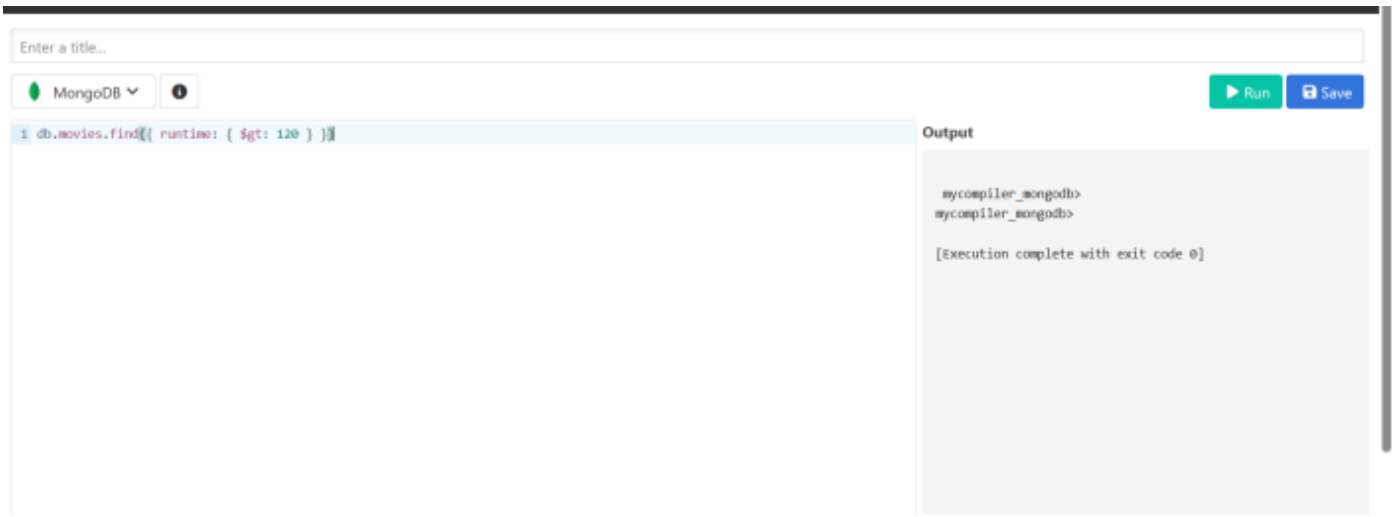
OUTPUT:

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:

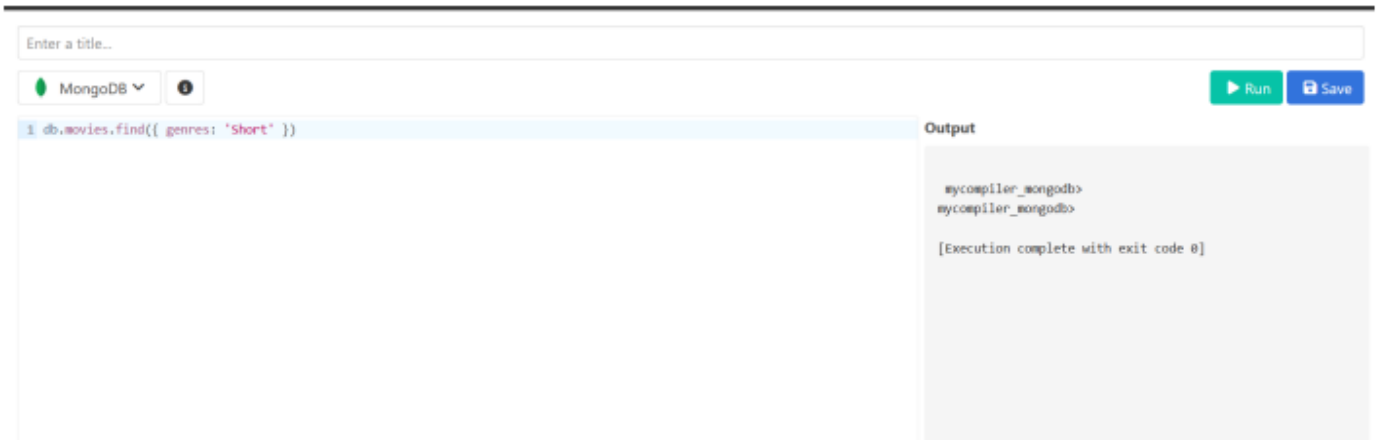


3.) Write a MongoDB query to find the restaurant Id, name and grades for those restaurants where the 2nd element of grades array contains a grade of "A" and score 9 on an ISODate "2014-08-11T00:00:00Z".

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:

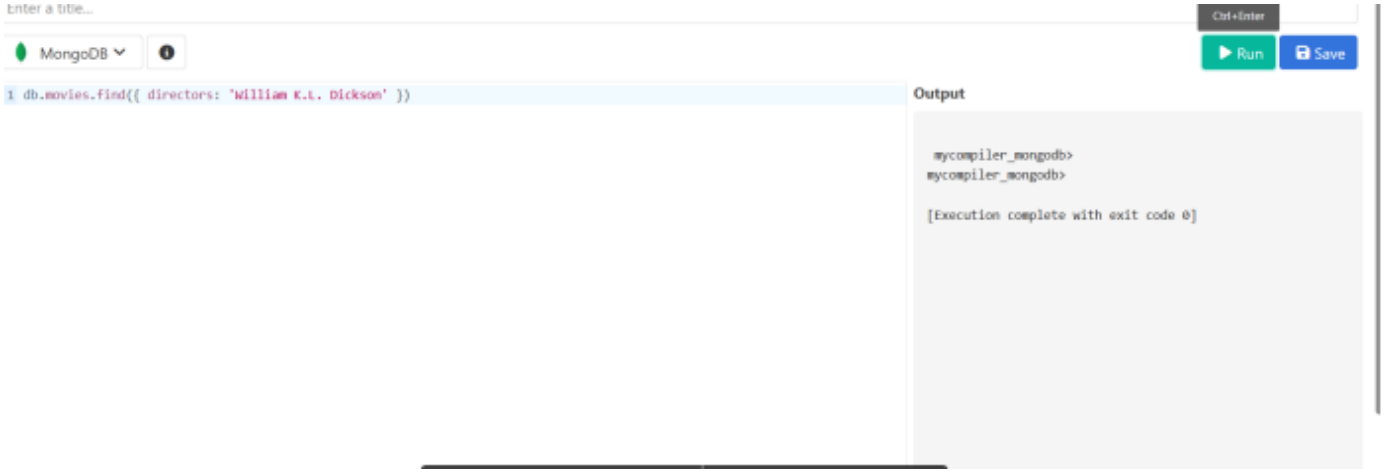


4.) Write a MongoDB query to find the restaurant Id, name, address and geographical location for those restaurants where 2nd element of coord array contains a value which is more than 42 and upto 52

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 });


Enter a title...
Ctrl+S
MongoDB
Run
Save
1 db.movies.find({ countries: 'USA' })
Output
mycompiler_mongodb>
mycompiler_mongodb>
[Execution complete with exit code 0]