**STUDENT MANAGEMENT PROJECT (DBMS)**

1. **TABLES**

1. Student\_official table

**create table student\_official**(

 roll\_no varchar(20), sectionId varchar(20), deptId varchar(20), semester varchar(20), join\_year int, pass\_year int,

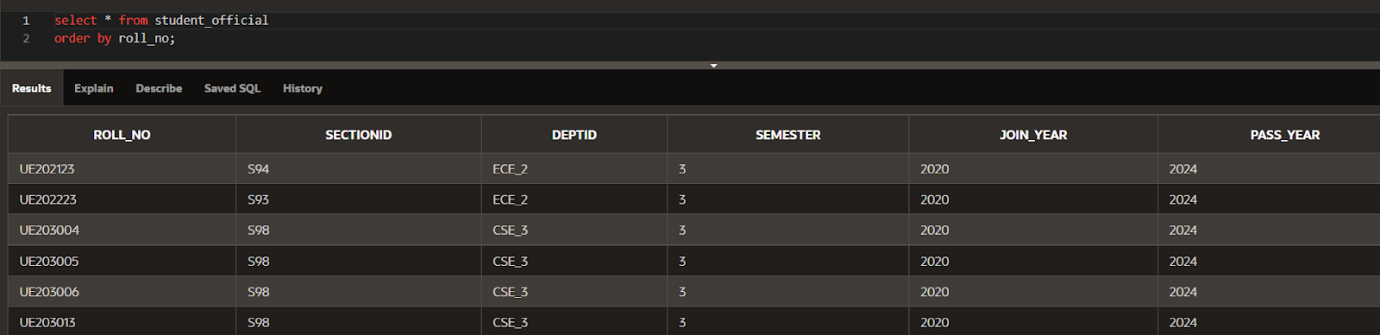
 CONSTRAINT "PK\_roll\_no\_student\_official" PRIMARY KEY (roll\_no),   CONSTRAINT "FK\_deptId\_student\_official" FOREIGN KEY (deptId) REFERENCES department(deptId),

 CONSTRAINT "FK\_sectionId\_student\_official" FOREIGN KEY (sectionId) REFERENCES section(sectionId)

)

**Select** \* from student\_official

Order by roll\_no;



1. Student\_personal table

**CREATE TABLE student\_personal**(

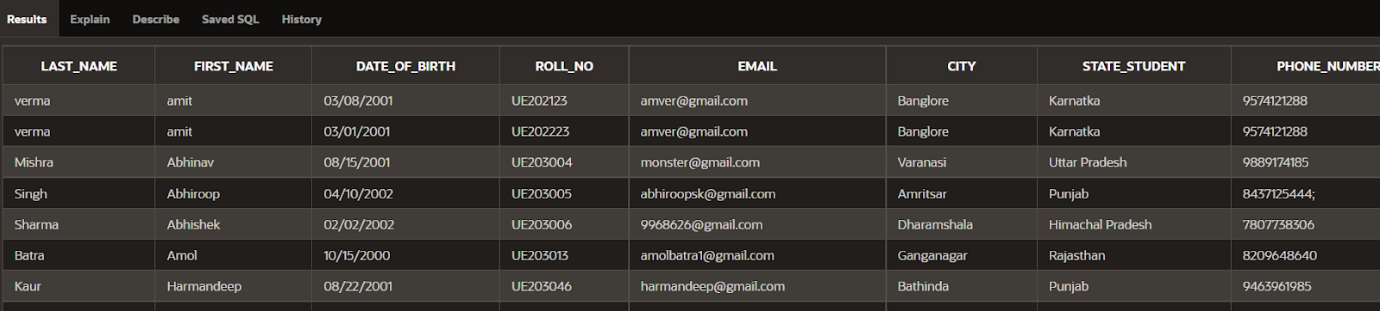
 last\_name VARCHAR(20), first\_name VARCHAR(20), date\_of\_birth date, roll\_no varchar(10), email varchar(20),city varchar(20), state\_student varchar(20), phone\_number varchar(12),

 CONSTRAINT "FK\_roll\_no\_student\_personal" FOREIGN KEY (roll\_no) REFERENCES student\_official(roll\_no)

);

**select** \* from student\_personal

Order by roll\_no;



1. Results table

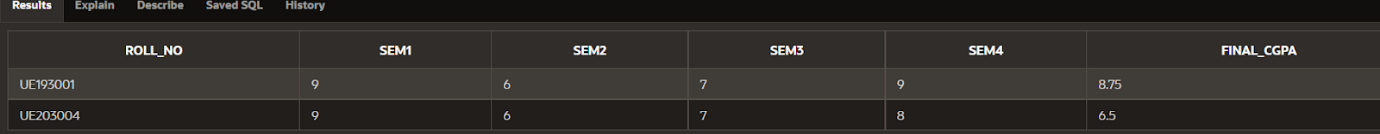
**create table results** (

    roll\_no varchar(20), sem1 int, sem2 int, sem3 int, sem4 int, final\_cgpa int,

    CONSTRAINT "FK\_roll\_no\_results" FOREIGN KEY (roll\_no) REFERENCES student\_official(roll\_no));

**select** \* from results

Order by roll\_no;



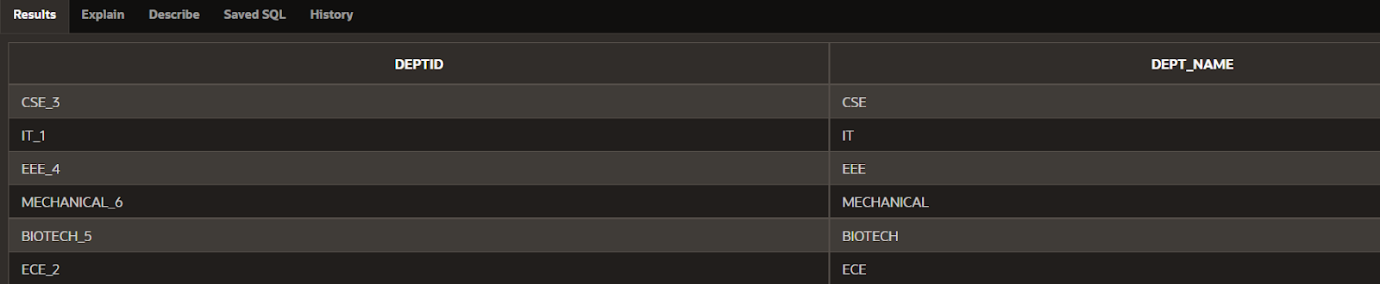
1. Department table

**CREATE TABLE department**(

    deptId VARCHAR(20), dept\_name VARCHAR(50)

);

**Select** \* from department;



1. Course table

**CREATE TABLE course**(

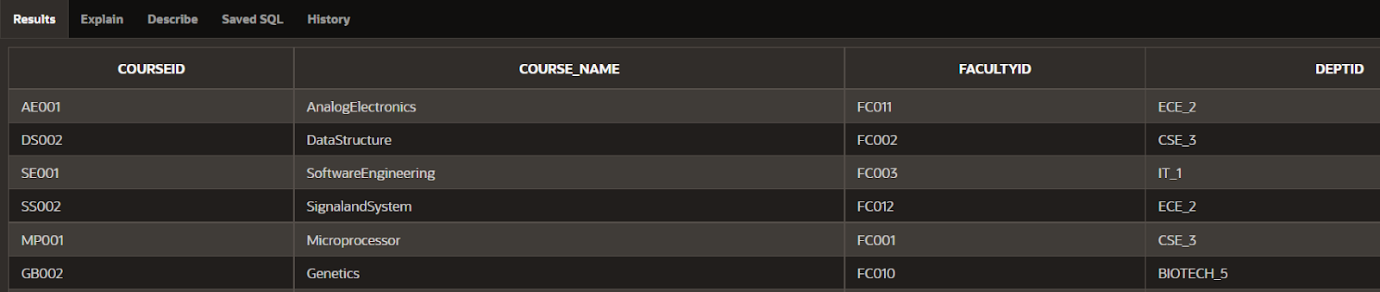
    courseId varchar(20), course\_name varchar(40), facultyID varchar(20) , deptId varchar(20),

    CONSTRAINT "PK\_courseId\_course" PRIMARY KEY (courseID),

    CONSTRAINT "FK\_deptId\_course" FOREIGN KEY (deptId) REFERENCES department(deptId)

);

**Select** \* from course;



1. Exam table

**CREATE TABLE exam** (

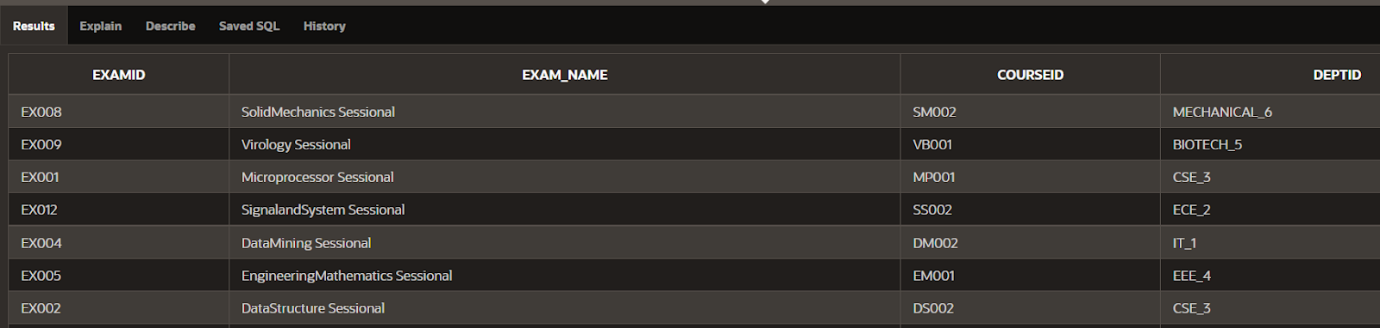
deptId VARCHAR(20), examId varchar(20), exam\_name varchar(40),courseId varchar(20),

    CONSTRAINT "PK\_examId\_exam" PRIMARY KEY (examId),

    CONSTRAINT "FK\_deptId\_exam" FOREIGN KEY (deptId) REFERENCES department(deptId)

);

**Select** \* from exam;



1. Section table

**create table section** (

    sectionId varchar(20), section\_name VARCHAR(20), deptId varchar(20), no\_of\_students int,

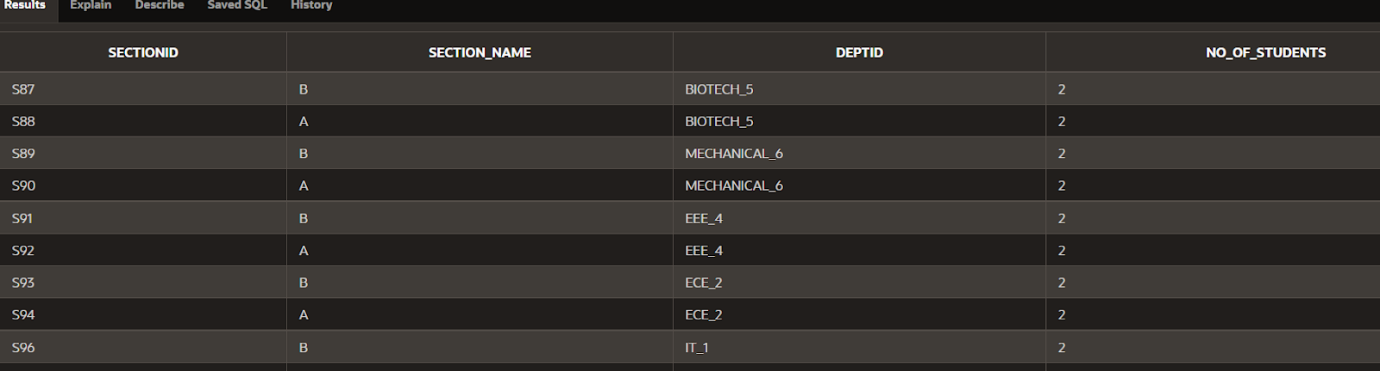
    CONSTRAINT "PK\_sectionId\_section" PRIMARY KEY (sectionId),

    CONSTRAINT "FK\_deptId\_section" FOREIGN KEY (deptId) REFERENCES department(deptId)

);

**Select** \* from section

Order by sectionId;

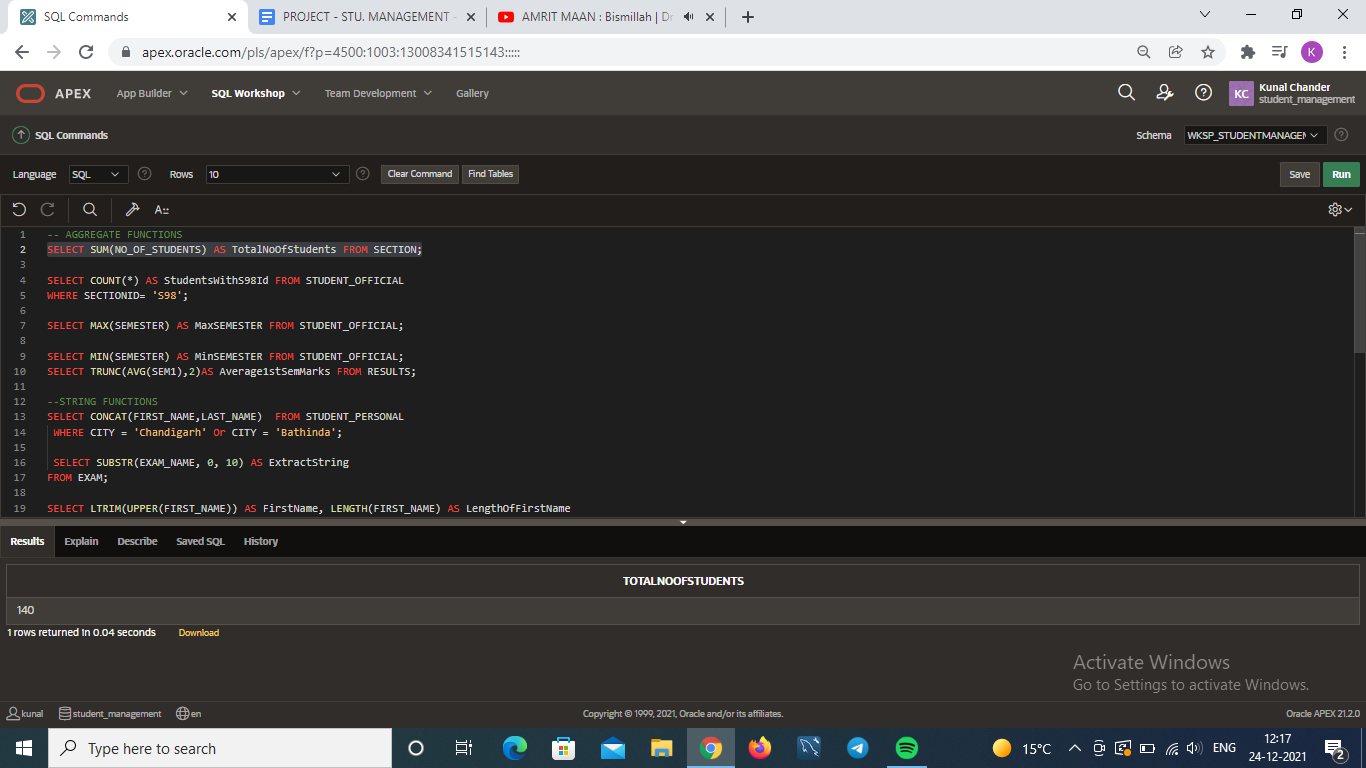


**2.   OPERATORS AND FUNCTIONS :**

**Aggregate functions :**

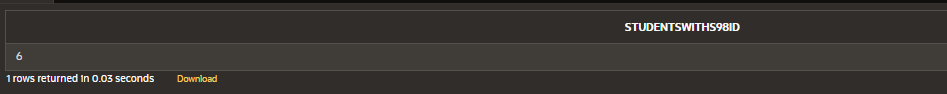
Sum() :

SELECT SUM(NO\_OF\_STUDENTS) AS TotalNoOfStudents FROM SECTION;



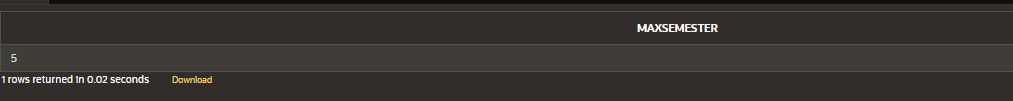
Count() :

SELECT COUNT(\*) AS StudentsWithS98Id FROM STUDENT\_OFFICIAL

WHERE SECTIONID= 'S98';

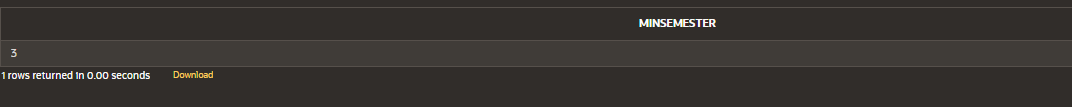
Max():

SELECT MAX(SEMESTER) AS MaxSEMESTER FROM STUDENT\_OFFICIAL;

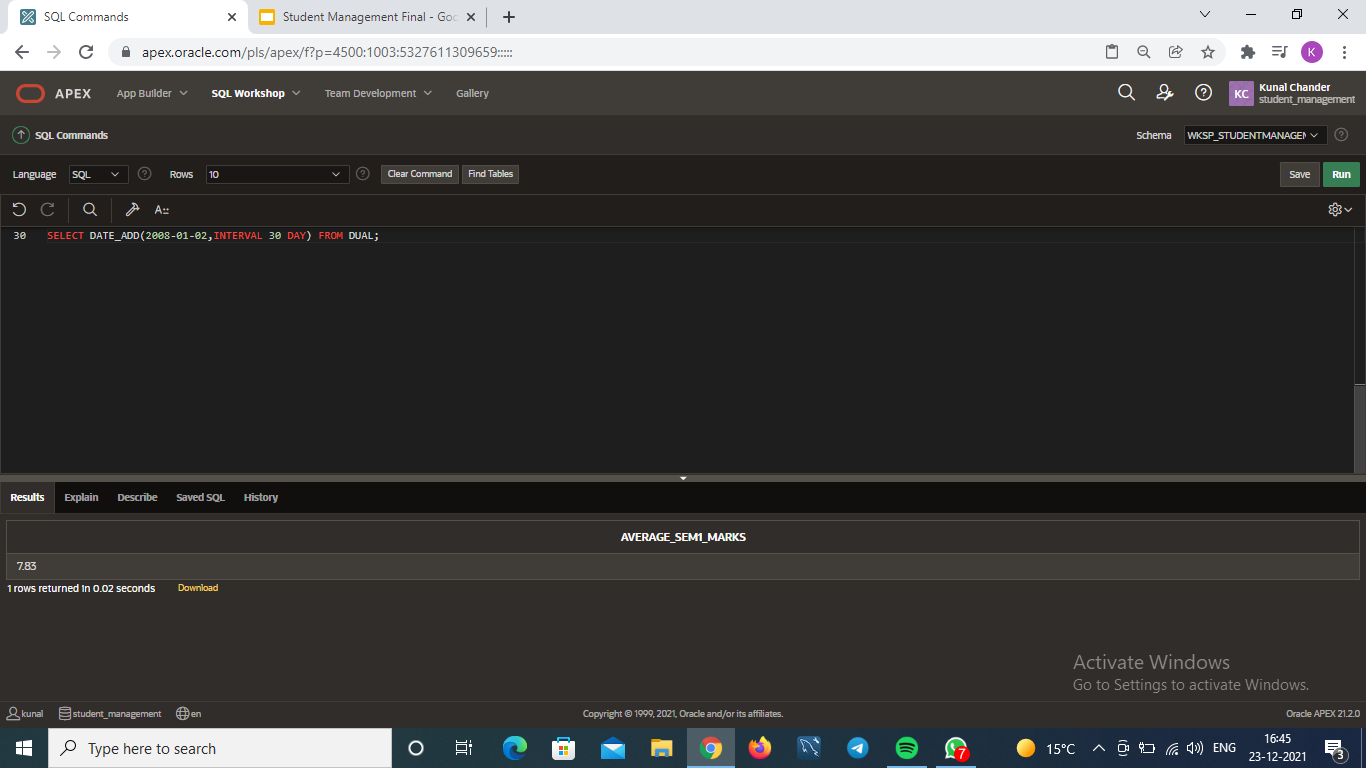


Min():

SELECT MIN(SEMESTER) AS MinSEMESTER FROM

STUDENT\_OFFICIAL;

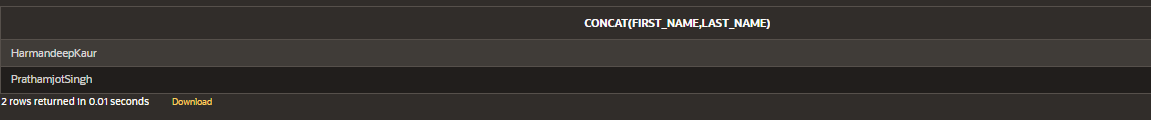
Avg():

SELECT AVG(SEM1) AS AVERAGE\_SEM1\_MARKS FROM RESULTS;

**String functions :**

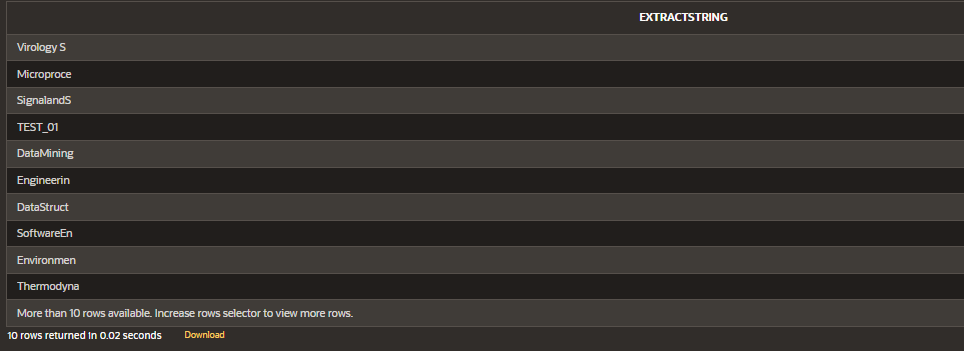
Concat():

SELECT CONCAT(FIRST\_NAME,LAST\_NAME)  FROM STUDENT\_PERSONAL

WHERE CITY = 'Chandigarh' Or CITY = 'Bathinda';

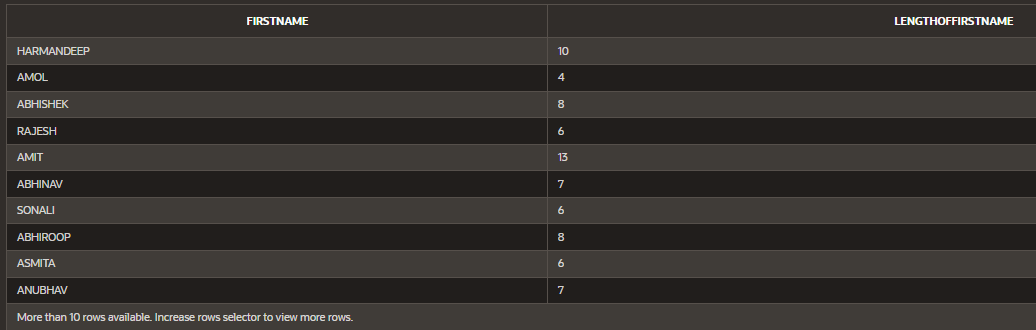
Substr():

SELECT SUBSTR(EXAM\_NAME, 0, 10) AS ExtractString

FROM EXAM;

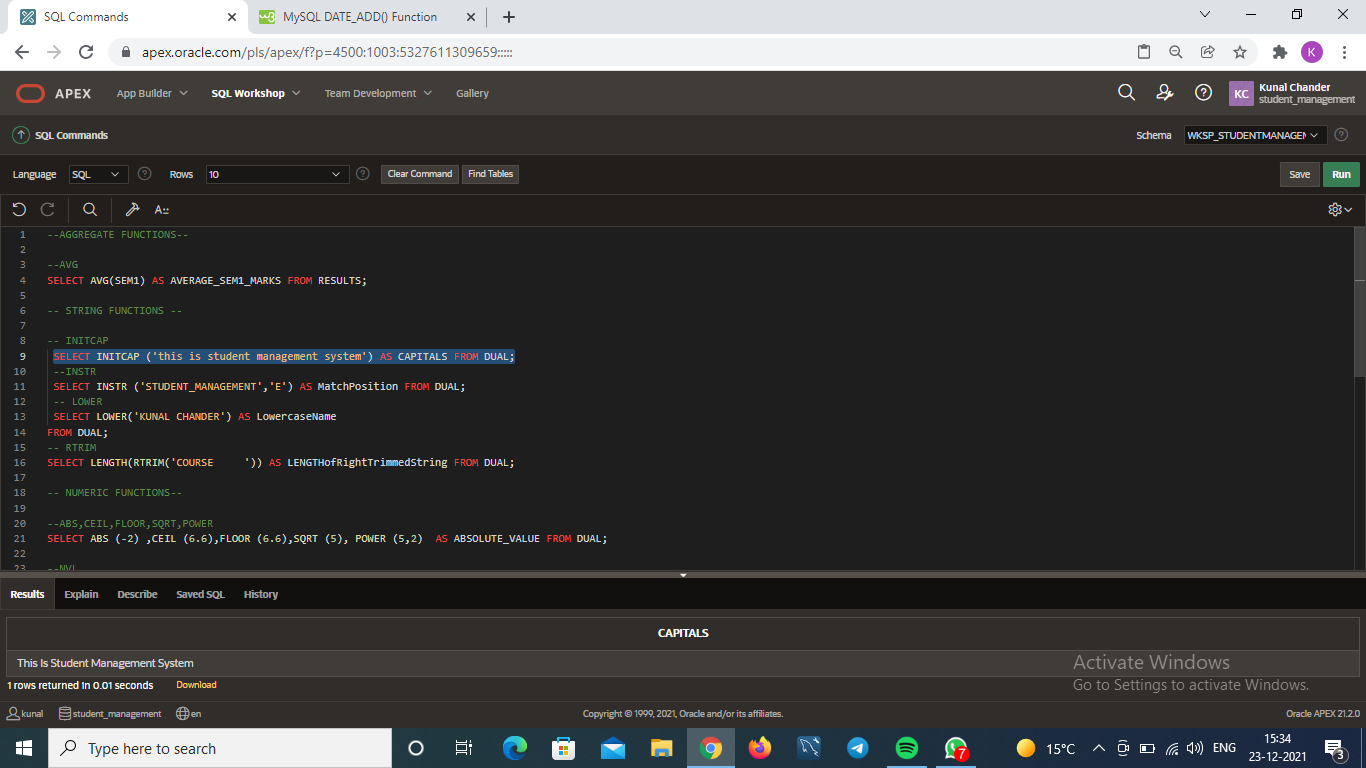
Ltrim(),Upper(),Length():

SELECT LTRIM(UPPER(FIRST\_NAME)) AS FirstName, LENGTH(FIRST\_NAME) AS LengthOfFirstName

FROM STUDENT\_PERSONAL;

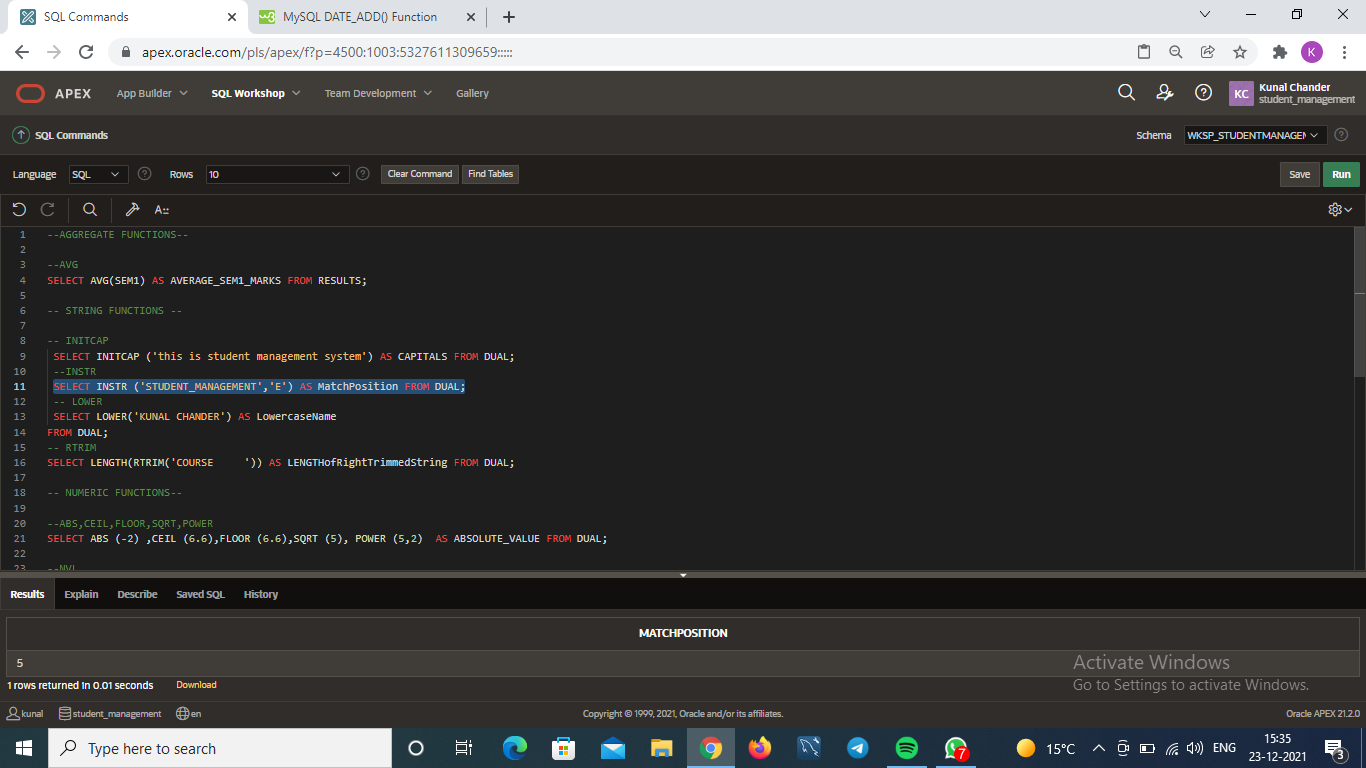
Initcap():

SELECT INITCAP ('this is student management system') AS CAPITALS FROM DUAL;



Instr():

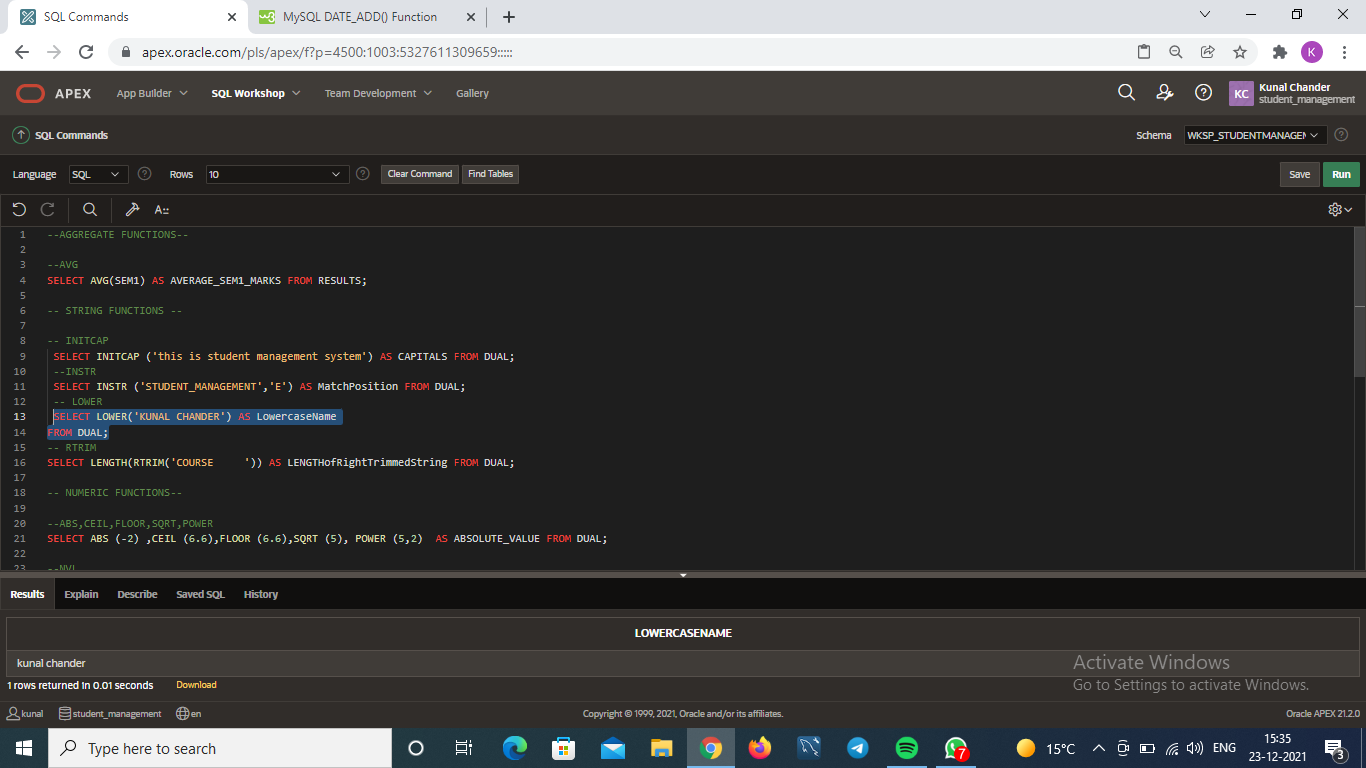
SELECT INSTR ('STUDENT\_MANAGEMENT','E') AS MatchPosition FROM DUAL;



Lower():

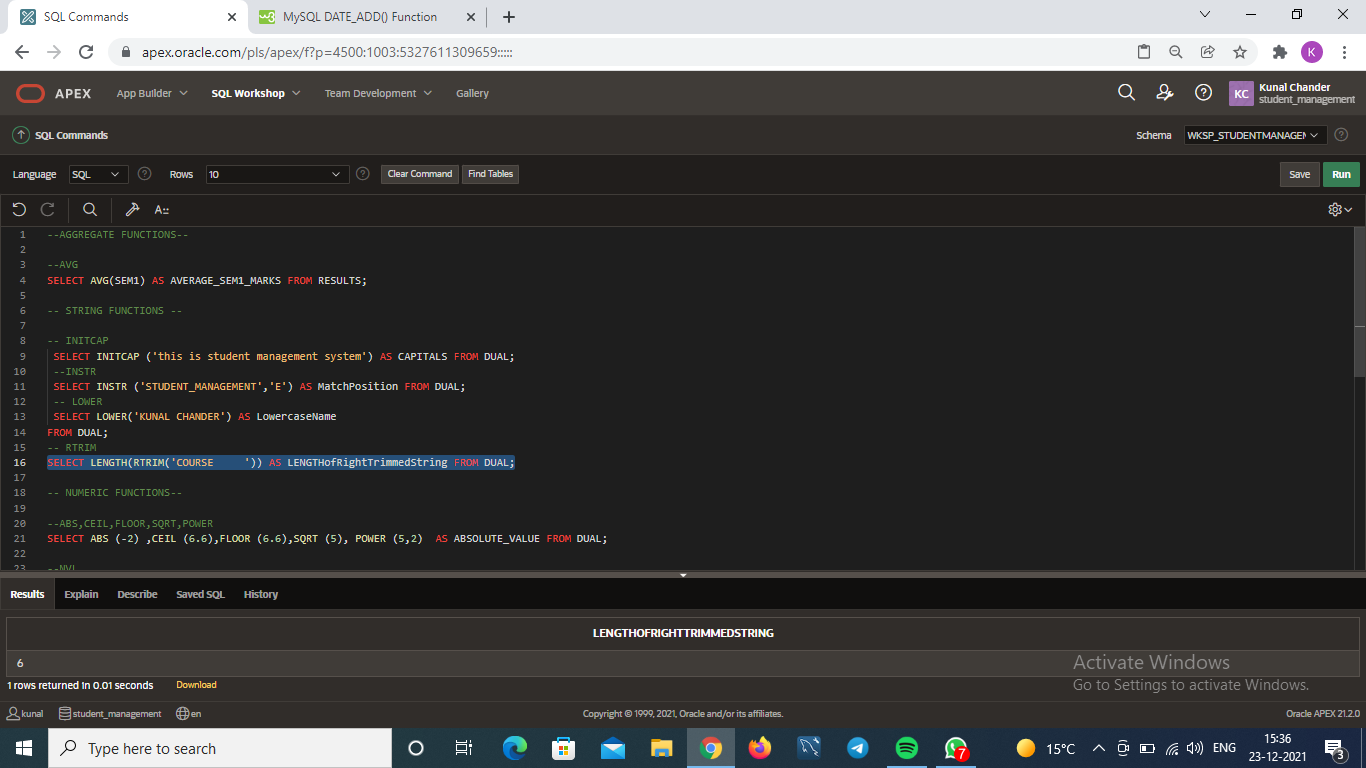
SELECT LOWER('KUNAL CHANDER') AS LowercaseName

FROM DUAL;



Rtrim():

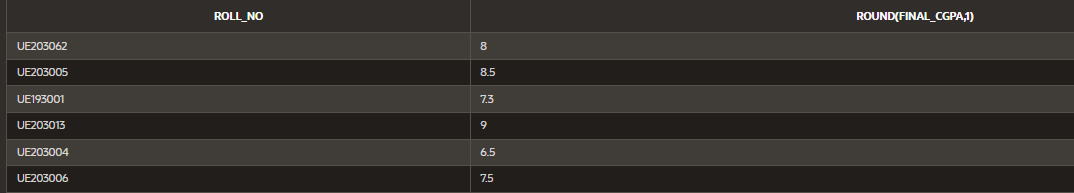
SELECT LENGTH(RTRIM('COURSE     ')) AS LENGTHofRightTrimmedString FROM DUAL;



**Numeric functions :**

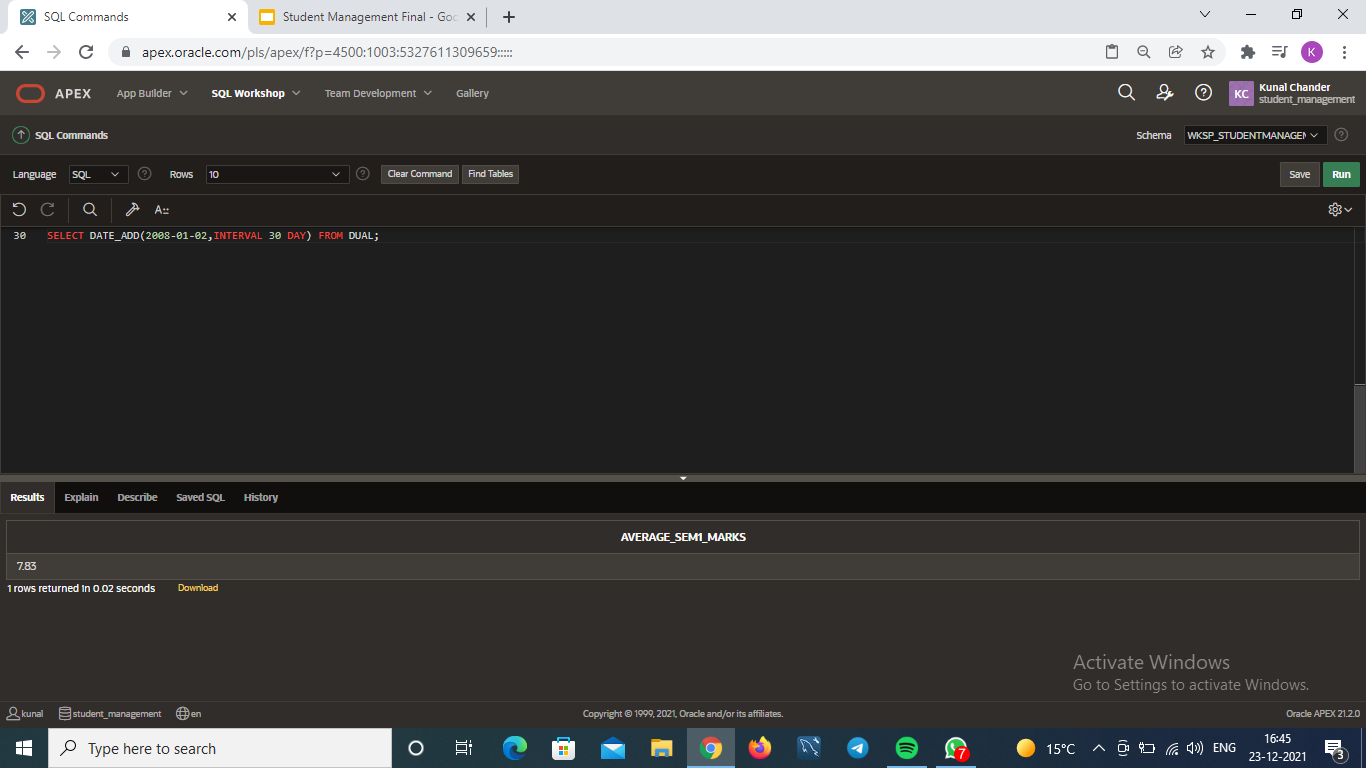
Round():

SELECT ROLL\_NO, ROUND(FINAL\_CGPA,1) FROM RESULTS;



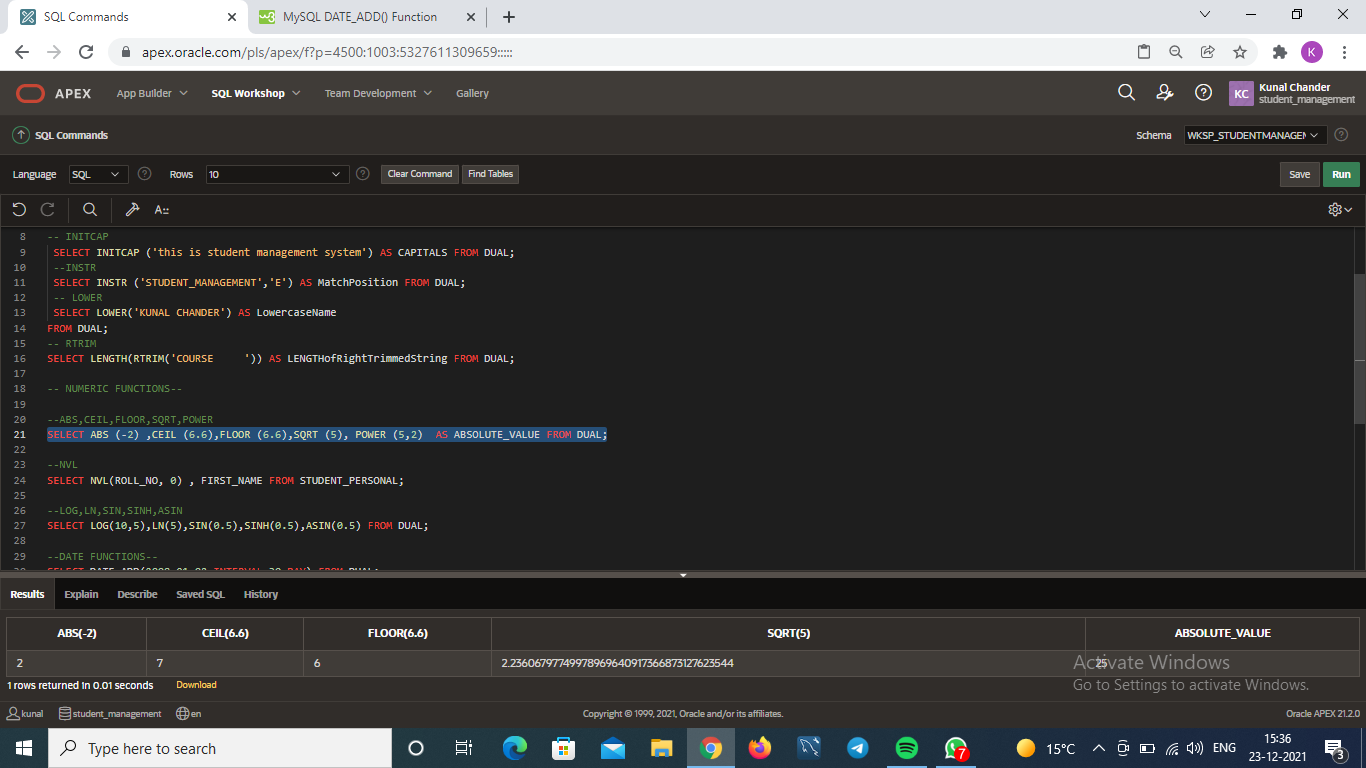
Trunc():

SELECT TRUNC(AVG(SEM1),2)AS Average1stSemMarks FROM RESULTS;



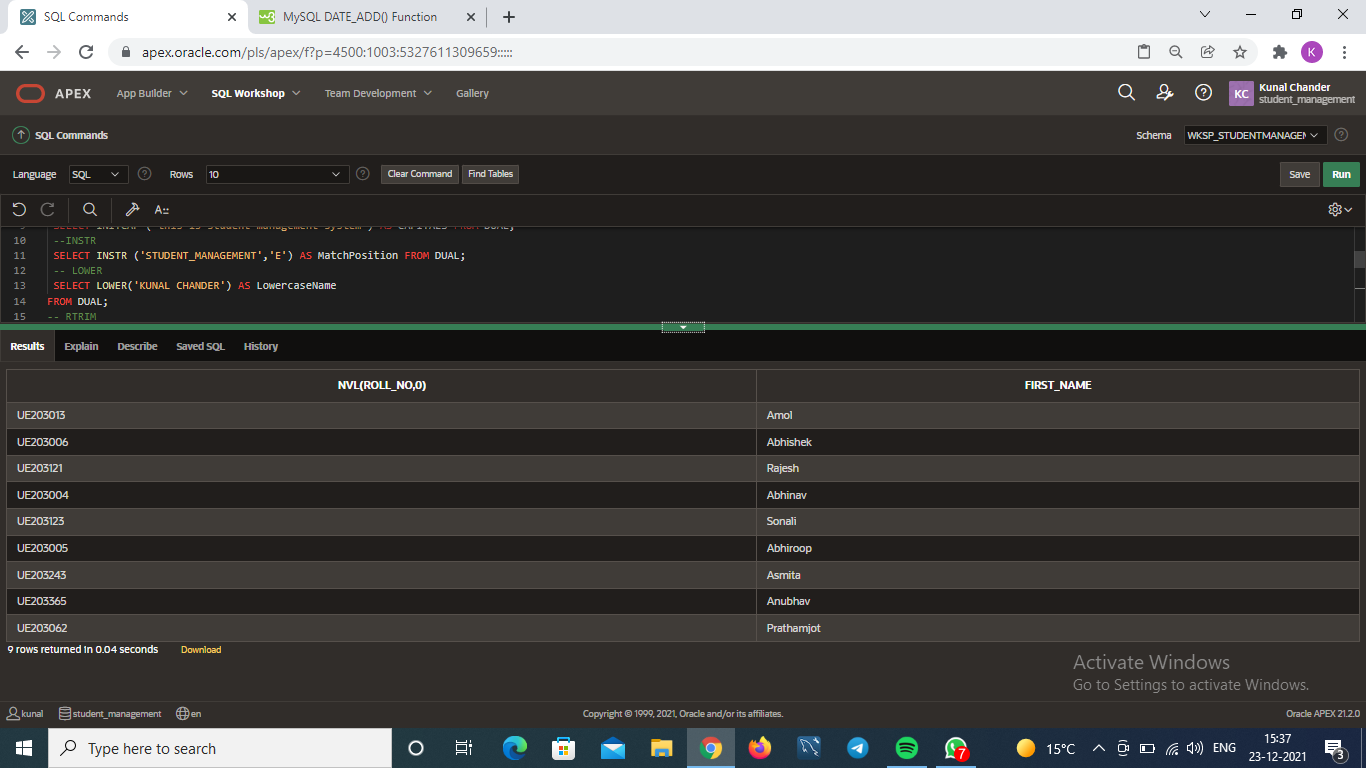
Abs(),Ceil(),Floor(),Sqrt(),Power():

SELECT ABS (-2) ,CEIL (6.6),FLOOR (6.6),SQRT (5), POWER (5,2)  AS ABSOLUTE\_VALUE FROM DUAL;



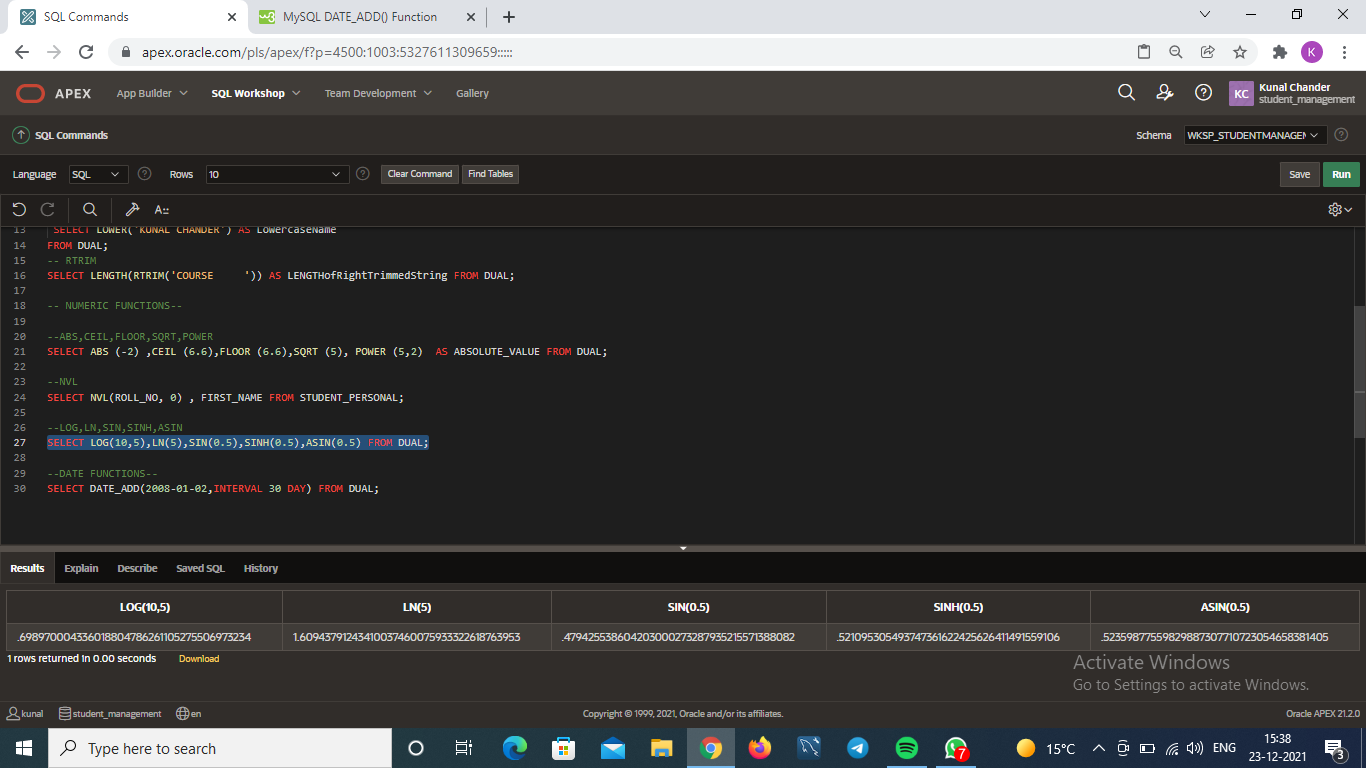
Nvl():

SELECT NVL(ROLL\_NO, 0) , FIRST\_NAME FROM STUDENT\_PERSONAL;



Log(),Ln(),Sin(),Sinh(),Asin():

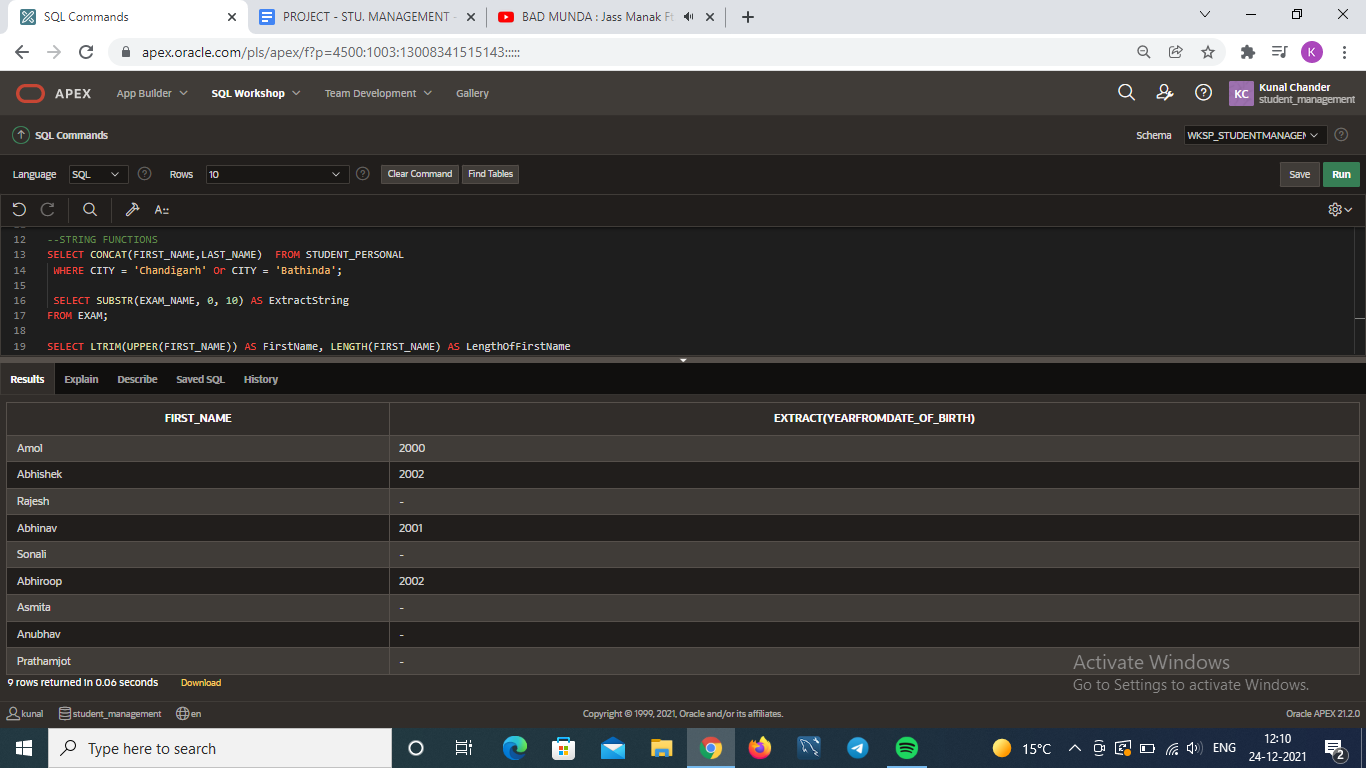
SELECT LOG(10,5),LN(5),SIN(0.5),SINH(0.5),ASIN(0.5) FROM DUAL;



**Date functions :**

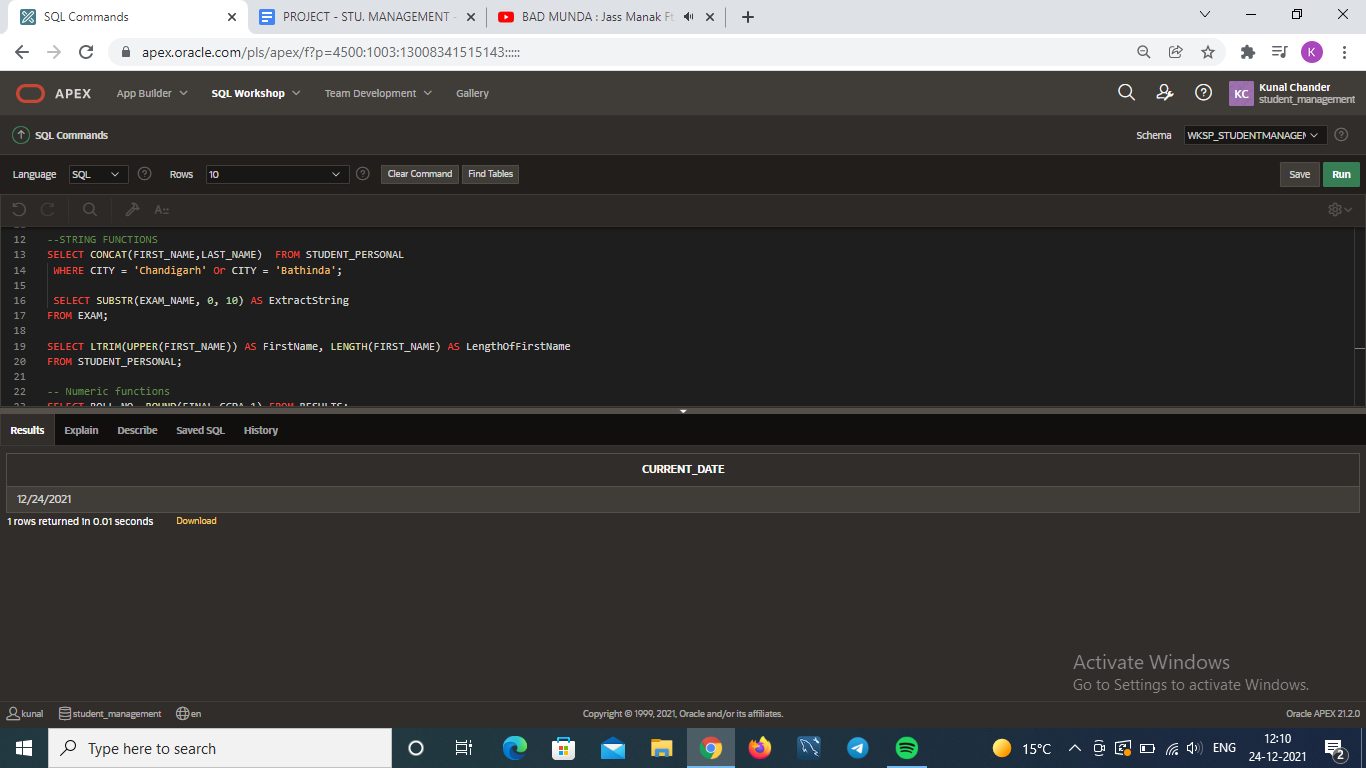
Extract():

SELECT FIRST\_NAME, EXTRACT(YEAR FROM DATE\_OF\_BIRTH) FROM STUDENT\_PERSONAL;



Current\_date():

SELECT CURRENT\_DATE FROM dual;

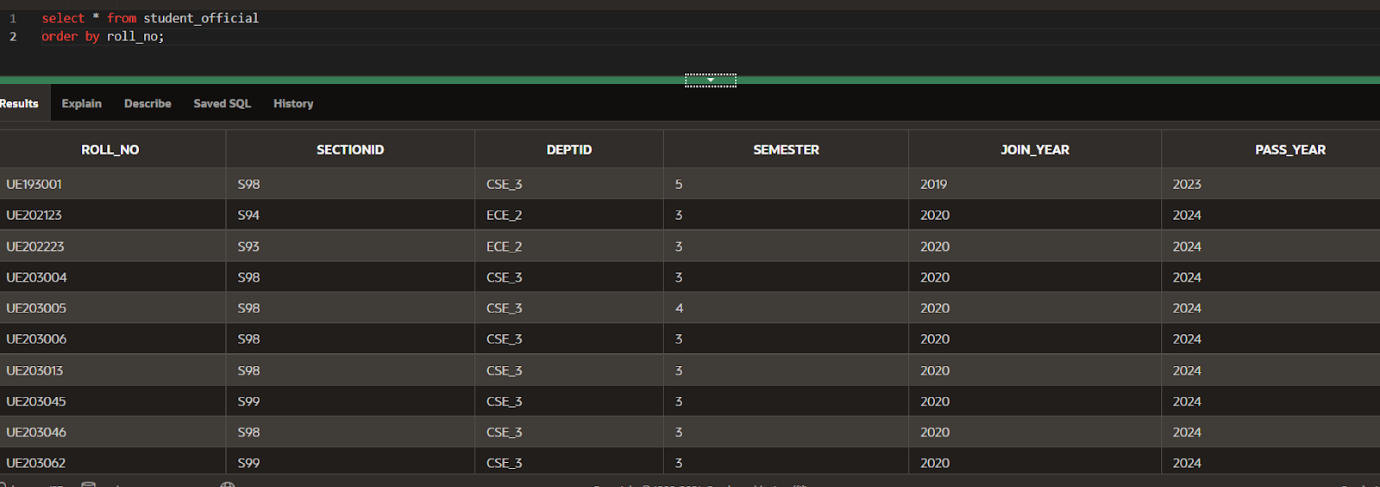
****

1. **Order by ,group by , having :**

Order by :

Select \* from student\_official

Order by roll\_no;

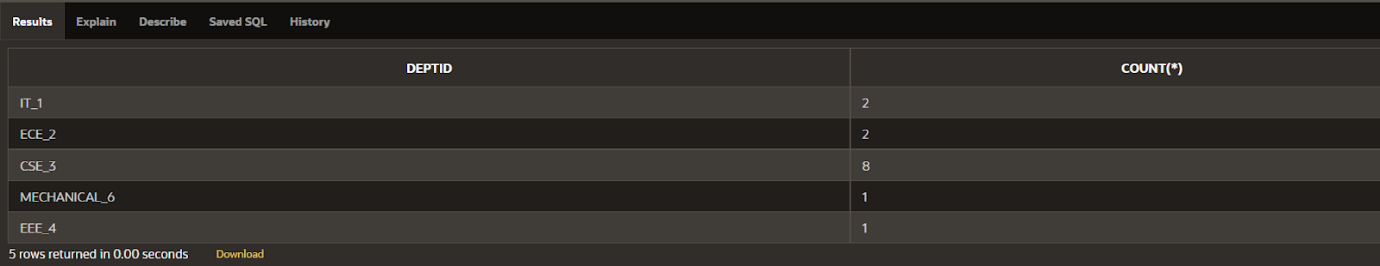


         Group by :

      select deptId ,count(\*)

      from student\_official

      group by deptId;



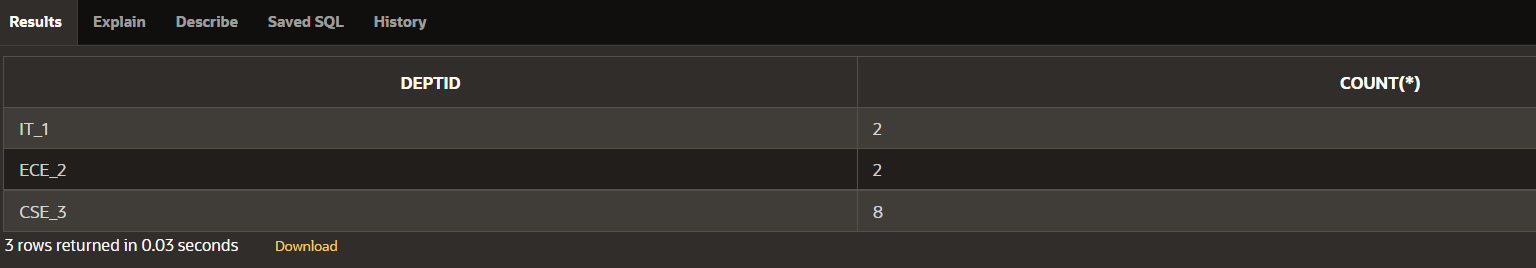
         Having :

      select deptId ,count(\*)

      from student\_official

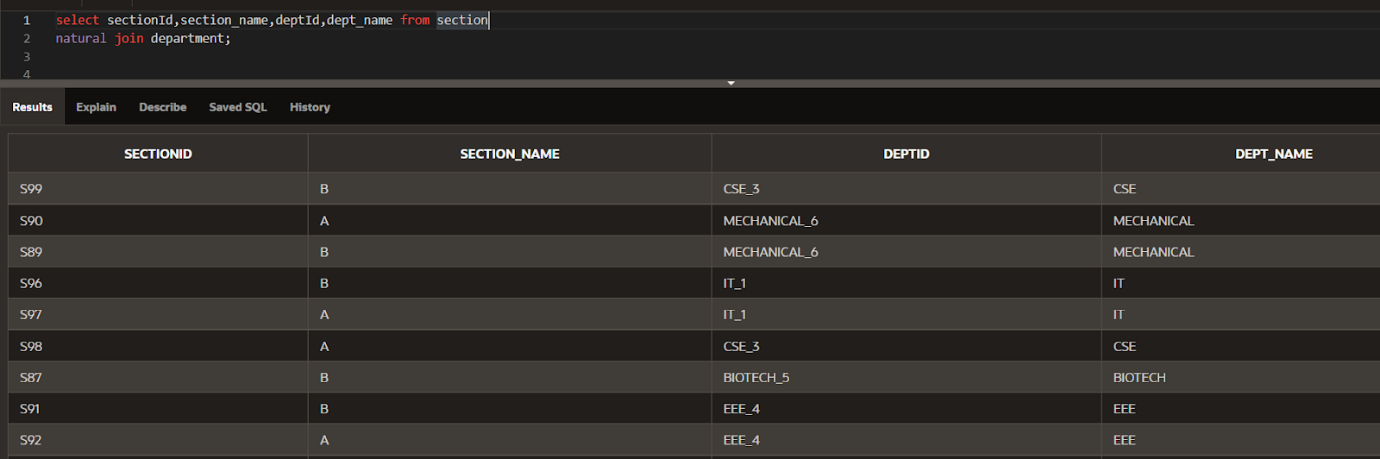
      group by deptId

      having count(\*)>1;

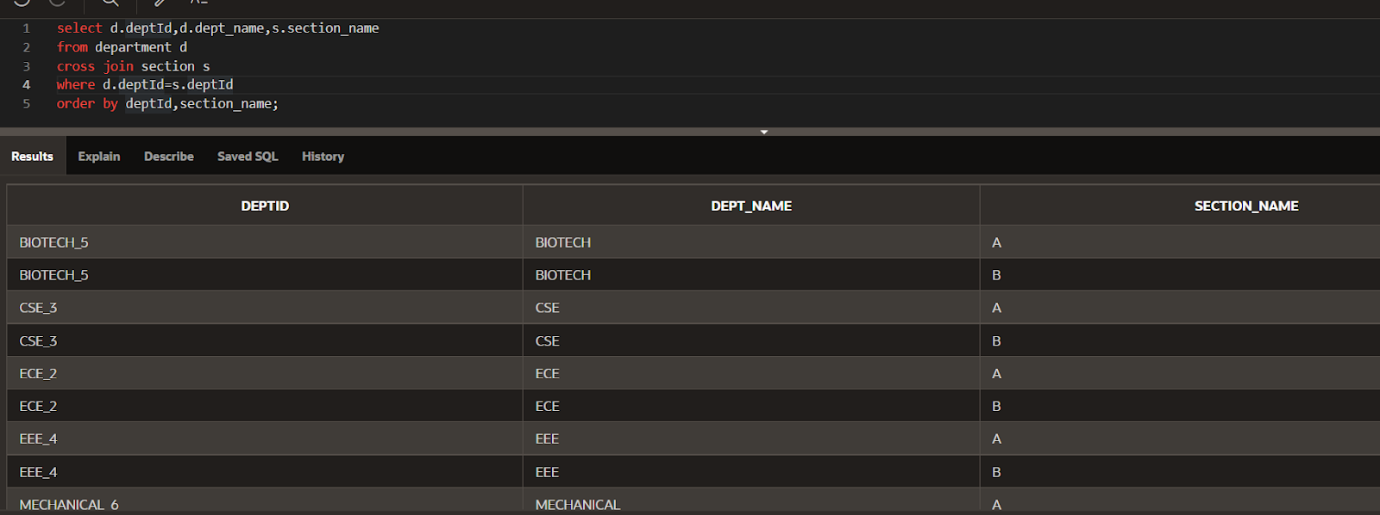


1. **JOINS:**

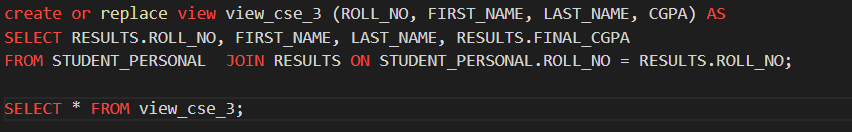
1. **NATURAL JOIN**

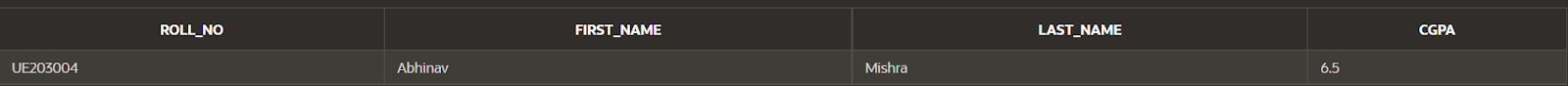
****

1. **CROSS JOIN**

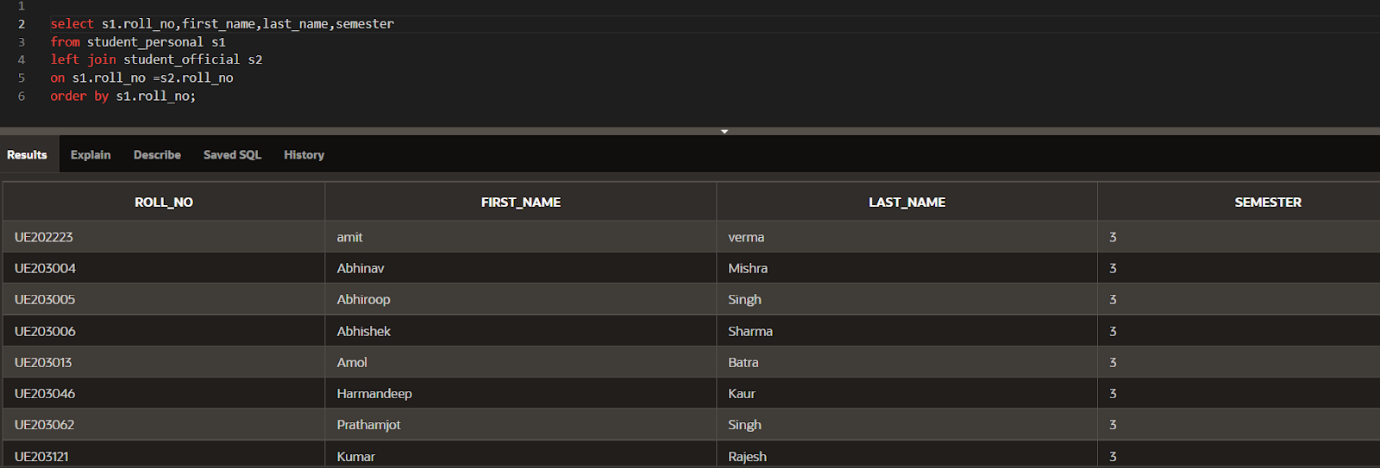
****

1. **INNER JOIN**

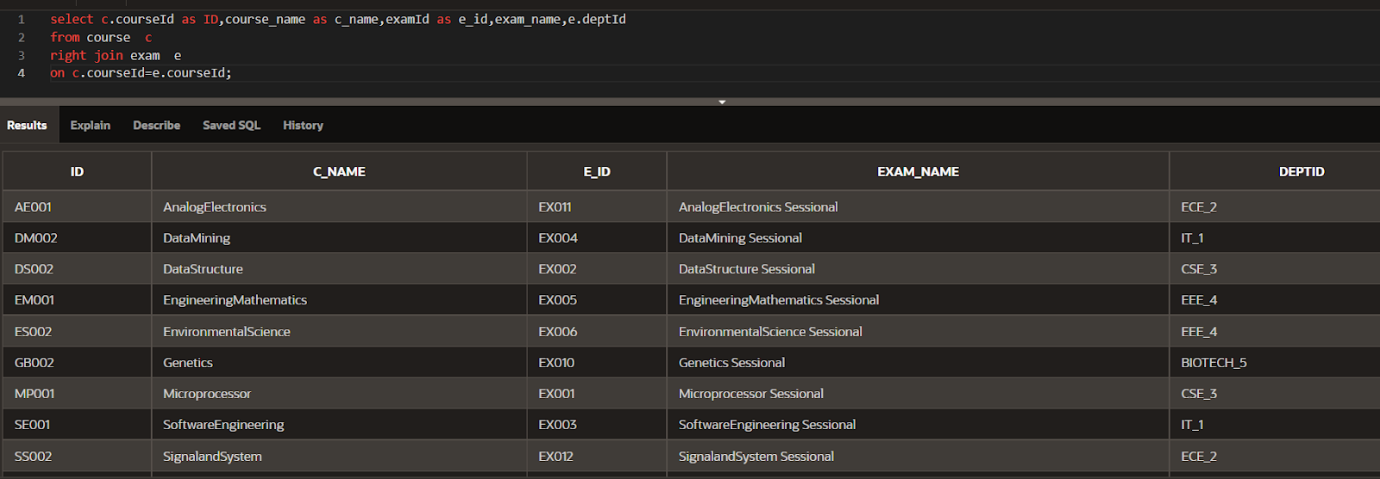
****

****

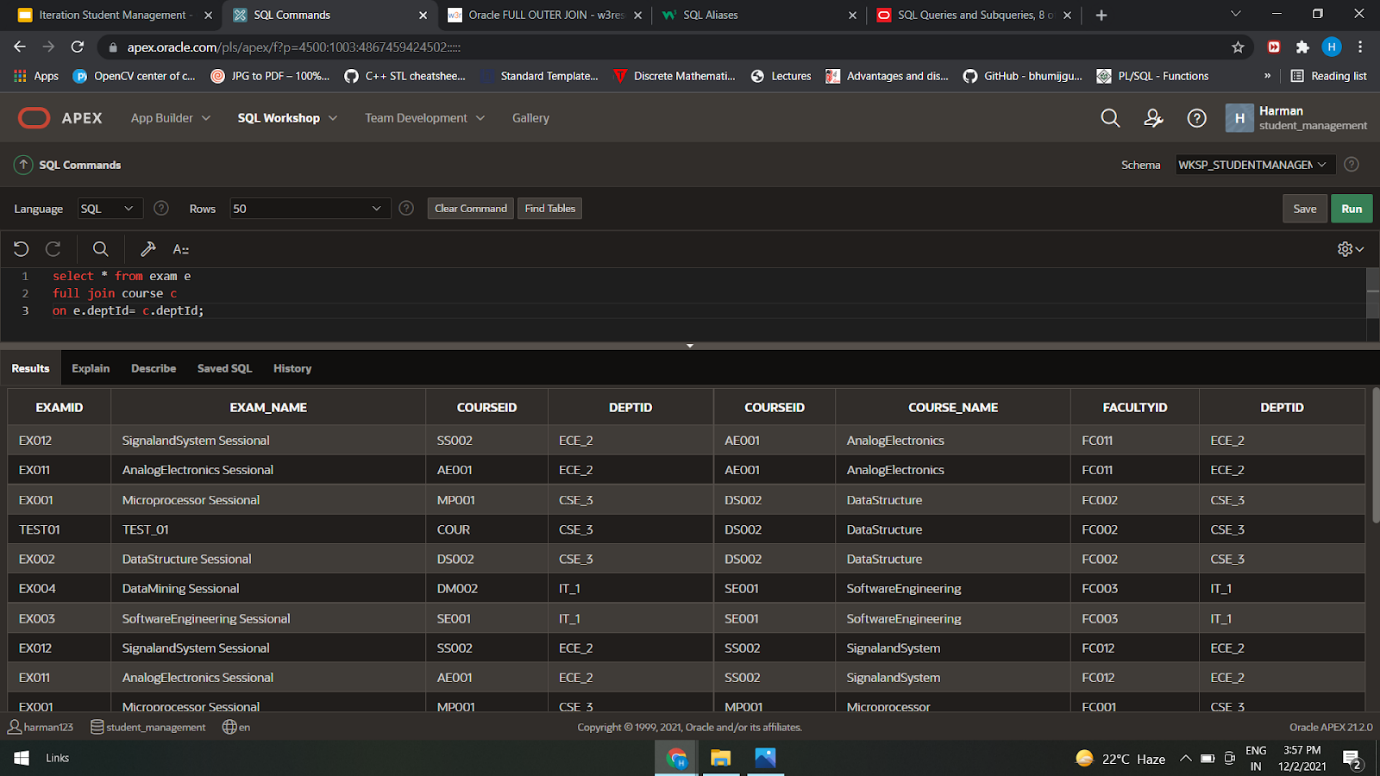
1. **LEFT JOIN**

****

1. **RIGHT JOIN**

****

1. **FULL JOIN**

****

1. **VIEWS :**

**VIEW FROM SINGLE TABLE:**

create view stu\_details

as

select roll\_no,first\_name,last\_name,email,phone\_number

from student\_personal;

**MULTI-TABLE VIEW :**

CREATE OR REPLACE VIEW "STUDENT\_INFO" ("ROLL\_NO", "FIRST\_NAME", "LAST\_NAME", "SEMESTER", "JOIN\_YEAR", "PASS\_YEAR") AS

  select s1.roll\_no,first\_name,last\_name,semester,join\_year,pass\_year

from student\_official s1,student\_personal s2

where s1.roll\_no= s2.roll\_no

order by s1.roll\_no;

1. **VIEWS AND JOINS**

**create** or replace view stu\_info

as

select s1.roll\_no,first\_name,last\_name,semester,deptId

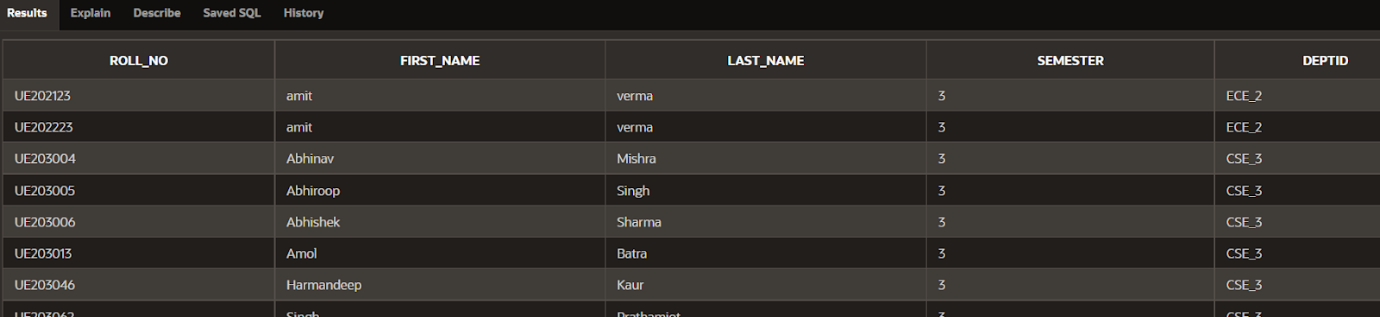
from student\_personal s1

left join student\_official s2

on s1.roll\_no=s2.roll\_no

order by s1.roll\_no;

**select** \* from stu\_info;



**create** or replace VIEW stu\_bysec as

 select s.roll\_no,semester,sec.sectionID,section\_name,sec.deptId

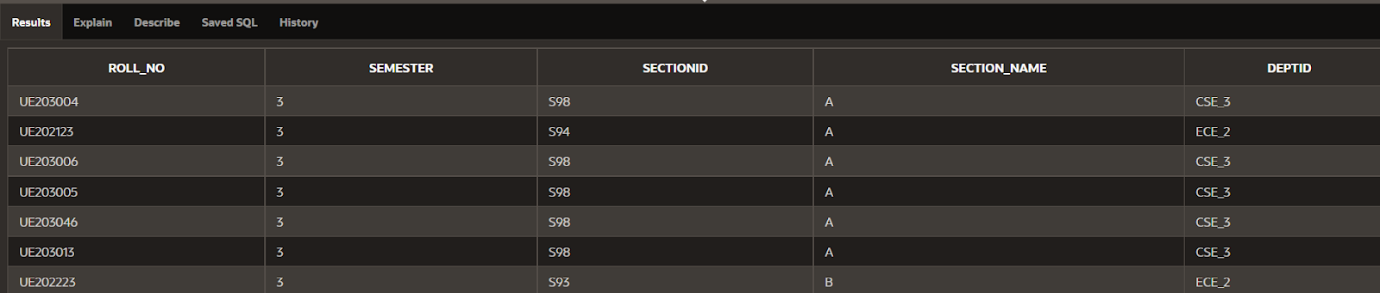
 from student\_official s

 full join section sec

 on s.sectionId=sec.sectionId

 order by sec.section\_name;

**select** \* from stu\_bysec;



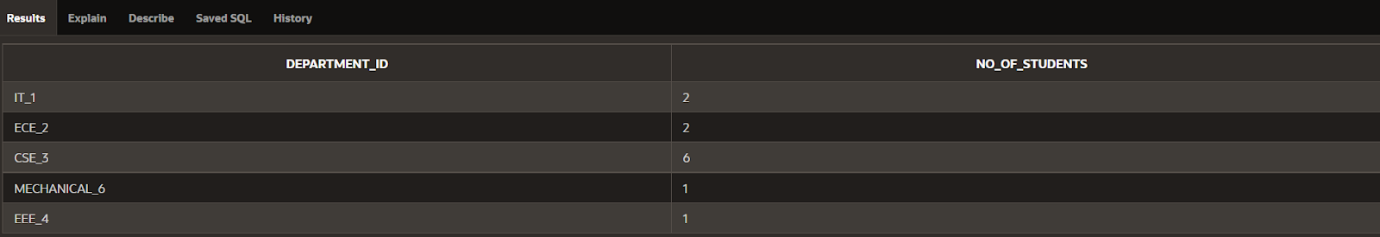
**CREATE** OR REPLACE VIEW student\_by\_department(DEPARTMENT\_ID, NO\_OF\_STUDENTS) AS

SELECT DEPTID, COUNT(\*)

FROM STUDENT\_OFFICIAL

GROUP BY DEPTID;

**Select** \* from student\_by\_department;

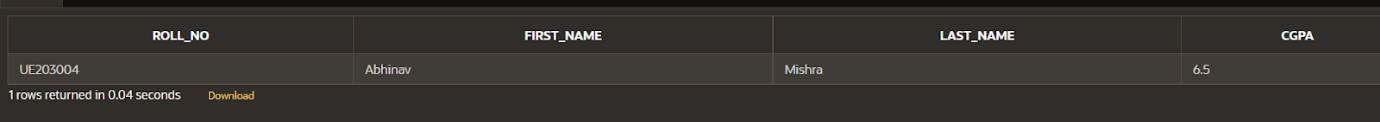


**create** or replace view view\_cse\_3 (ROLL\_NO, FIRST\_NAME, LAST\_NAME, CGPA) AS

SELECT RESULTS.ROLL\_NO, FIRST\_NAME, LAST\_NAME, RESULTS.FINAL\_CGPA

FROM STUDENT\_PERSONAL  JOIN RESULTS ON STUDENT\_PERSONAL.ROLL\_NO = RESULTS.ROLL\_NO;

**SELECT** \* FROM view\_cse\_3;



**create** or replace FORCE EDITIONABLE VIEW stu\_dept as

 select roll\_no,d.deptId,dept\_name

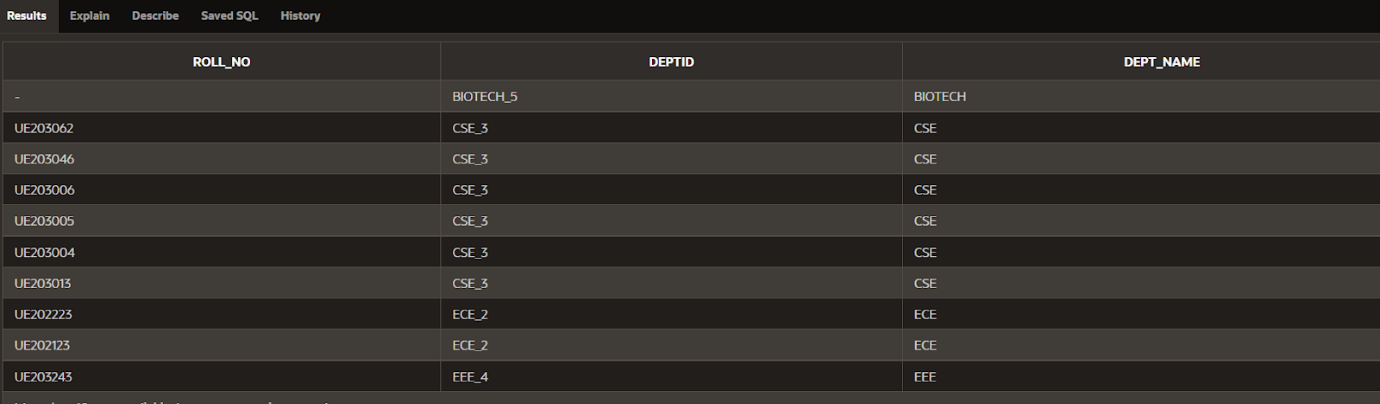
 from student\_official s

 full join department d

 on s.deptId=d.deptId

 order by d.deptId;

**select** \* from stu\_dept;



1. **INDEX**

**SINGLE-COLUMN INDEX :**

**create index** semester\_idx

on student\_official(semester);

(on semester column of table student\_official)

(FOR queries regarding semester)

**COMPOSITE INDEX:**

**UNIQUE INDEX:**

1. **SEQUENCE :**

1. **TRIGGERS:**

create or replace trigger trigger\_student\_personal\_delete\_before

BEFORE DELETE on student\_personal

begin

dbms\_output.put\_line('Data in student\_personal table successfully deleted');

End;

create or replace trigger trigger\_student\_personal\_insert\_after

after insert on student\_personal

begin

dbms\_output.put\_line('Data successfully inserted into student\_personal table  ');

end;

create or replace trigger trigger\_student\_personal\_update\_date\_of\_birth

after update of date\_of\_birth

on student\_personal

referencing OLD as old

NEW as new

for each row

begin

dbms\_output.put\_line('Data in student\_personal table successfully updated');

dbms\_output.put\_line('Old dob: '|| :OLD.date\_of\_birth);

dbms\_output.put\_line('NEW dob: '|| :NEW.date\_of\_birth);

End;

create or replace trigger trigger\_student\_personal\_update\_email

after update of email

on student\_personal

referencing OLD as old

NEW as new

for each row

begin

dbms\_output.put\_line('Data in student\_personal table successfully updated');

dbms\_output.put\_line('Old email: '|| :OLD.email);

dbms\_output.put\_line('NEW email: '|| :NEW.email);

end;

create or replace trigger trigger\_student\_personal\_update\_first\_name

after update of first\_name

on student\_personal

referencing OLD as old

NEW as new

for each row

begin

dbms\_output.put\_line('Data in student\_personal table successfully updated');

dbms\_output.put\_line('Old first\_name: '|| :OLD.first\_name);

dbms\_output.put\_line('NEW first\_name: '|| :NEW.first\_name);

End;

create or replace trigger trigger\_student\_personal\_update\_last\_name

after update of last\_name

on student\_personal

referencing OLD as old

NEW as new

for each row

begin

dbms\_output.put\_line('Data in student\_personal table successfully updated');

dbms\_output.put\_line('Old l\_name: '|| :OLD.last\_name);

dbms\_output.put\_line('NEW l\_name: '|| :NEW.last\_name);

End;

create or replace trigger trigger\_student\_personal\_update\_phone\_number

after update of phone\_number

on student\_personal

referencing OLD as old

NEW as new

for each row

begin

dbms\_output.put\_line('Data in student\_personal table successfully updated');

dbms\_output.put\_line('Old phn: '|| :OLD.phone\_number);

dbms\_output.put\_line('NEW phn: '|| :NEW.phone\_number);

end;

create or replace trigger trigger\_student\_personal\_update\_state\_student

after update of state\_student

on student\_personal

referencing OLD as old

NEW as new

for each row

begin

dbms\_output.put\_line('Data in student\_personal table successfully updated');

dbms\_output.put\_line('Old state: '|| :OLD.state\_student);

dbms\_output.put\_line('NEW state: '|| :NEW.state\_student);

end;

1. **AUDIT TRAIL**

1. **RECORDS:**

1. **FUNCTIONS**

1. FUNCTION TO FIND NUMBER OF STUDENTS HAVING CGPA GREATER THAN GIVEN CGPA

create or replace FUNCTION studentsGreaterThan (

    cg RESULTS.FINAL\_CGPA%TYPE

)

RETURN NUMBER

IS

    temp\_stuCount NUMBER:=0;

    temp RESULTS.FINAL\_CGPA%TYPE;

BEGIN

    SELECT COUNT(\*) INTO temp\_stuCount FROM RESULTS

    WHERE cg >= (

        SELECT FINAL\_CGPA FROM RESULTS

    );

    IF temp\_stuCount = NULL THEN

        temp\_stuCount := 0;

    end if;

    RETURN temp\_stuCount;

END;

-- using function to get number of students having cgpa greater than entered number n

declare

n number;

BEGIN

 n:= :n;

    dbms\_output.put\_line(studentsGreaterThan(n));

end;

After insertion in results insert ss here

1. FUNCTION TO FIND NUMBER OF STUDENTS HAVING CGPA LESSER THAN GIVEN CGPA

create or replace FUNCTION studentsLessThan (

    cg RESULTS.FINAL\_CGPA%TYPE

)

RETURN NUMBER

IS

    temp\_stuCount NUMBER:=0;

    temp RESULTS.FINAL\_CGPA%TYPE;

BEGIN

    SELECT COUNT(\*) INTO temp\_stuCount FROM RESULTS

    WHERE cg < (

        SELECT FINAL\_CGPA FROM RESULTS

    );

    IF temp\_stuCount = NULL THEN

        temp\_stuCount := 0;

    end if;

    RETURN temp\_stuCount;

END;

-- using function to get number of students having cgpa lesser than entered number n

declare

n number;

BEGIN

 n:= :n;

    dbms\_output.put\_line(studentsLessThan(n));

end;

After insertion in results insert ss here

1. **PACKAGE/PROCEDURE**

a. Student Official Operations Pack

create or replace package student\_official\_pack as

PROCEDURE student\_official\_insert(

            v\_roll\_no  IN VARCHAR2,

            v\_sectionId IN VARCHAR2,

            v\_deptId  IN VARCHAR2,

            V\_semester IN VARCHAR2,

            v\_join\_year IN int,

            v\_pass\_year IN int

           );

procedure search\_by\_roll\_no\_in\_student\_official(r\_no in varchar2);

procedure delete\_student\_official\_data(

        v\_roll\_no in varchar2);

procedure update\_student\_official\_sectionID(

        v\_roll\_no in varchar2,

        v\_sectionID in varchar2);

procedure update\_student\_official\_deptID(

        v\_roll\_no in varchar2,

        v\_deptID in varchar2);

procedure update\_student\_official\_semester(

        v\_roll\_no in varchar2,

        v\_semester in varchar2);

end student\_official\_pack;

create or replace package body student\_official\_pack

as PROCEDURE student\_official\_insert(

            v\_roll\_no  IN VARCHAR2,

            v\_sectionId IN VARCHAR2,

            v\_deptId  IN VARCHAR2,

            V\_semester IN VARCHAR2,

            v\_join\_year IN int,

            v\_pass\_year IN int

           )

           is

           BEGIN INSERT into student\_official

           values(v\_roll\_no,v\_sectionId,v\_deptId,V\_semester,v\_join\_year,v\_pass\_year);

END student\_official\_insert;

procedure search\_by\_roll\_no\_in\_student\_official(r\_no in varchar2)

        is

        v\_roll\_no varchar2(20);

        v\_sectionID varchar2(20);

        v\_deptID varchar2(10);

        v\_semester varchar2(20);

        v\_join\_year int;

        v\_pass\_year int;

        begin

        select roll\_no into v\_roll\_no from student\_official where roll\_no=r\_no;

        select sectionID into v\_sectionID from student\_official where roll\_no=r\_no;

        select deptID into v\_deptID from student\_official where roll\_no=r\_no;

        select semester into v\_semester from student\_official where roll\_no=r\_no;

        select join\_year into v\_join\_year from student\_official where roll\_no=r\_no;

        select pass\_year into v\_pass\_year from student\_official where roll\_no=r\_no;

        dbms\_output.put\_line(v\_roll\_no||'  '||v\_sectionID||' '||v\_deptID||'  '||v\_semester||'  '||v\_join\_year||'  '||v\_pass\_year);

        IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1000,'Invalid Roll no.');

        END IF;

end search\_by\_roll\_no\_in\_student\_official;

procedure delete\_student\_official\_data(

        v\_roll\_no in varchar2)

        is

        begin

        delete from student\_official

        where roll\_no=v\_roll\_no;

        IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1001,'Invalid Roll no.');

        END IF;

        end delete\_student\_official\_data;

procedure update\_student\_official\_sectionID(

        v\_roll\_no in varchar2,

        v\_sectionID in varchar2)

        is

        invalid\_official\_sectionID exception;

        begin

        if (v\_sectionID != 'S99' or v\_sectionID != 'S90' or v\_sectionID != 'S89' or v\_sectionID != 'S96' or v\_sectionID != 'S97' or v\_sectionID != 'S98'

        or v\_sectionID != 'S87' or v\_sectionID != 'S91' or v\_sectionID != 'S92' or v\_sectionID != 'S94' or v\_sectionID != 'S93' or v\_sectionID != 'S88')

        then

    raise invalid\_official\_sectionID;

end if;

        update student\_official

        set sectionID=v\_sectionID

        where roll\_no=v\_roll\_no;

        IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1004,'Error');

        END IF;

        exception

    when invalid\_official\_sectionID then

    dbms\_output.put\_line('Section Id does not exist');

        end update\_student\_official\_sectionID;

procedure update\_student\_official\_deptID(

        v\_roll\_no in varchar2,

        v\_deptID in varchar2)

        is

        invalid\_official\_deptId exception;

        begin

        if (v\_deptID != 'CSE\_3' or v\_deptID != 'IT\_1' or v\_deptID != 'EEE\_4' or v\_deptID != 'MECHANICAL\_6' or v\_deptID != 'BIOTECH\_5' or v\_deptID != 'ECE\_2') then

    raise invalid\_official\_deptId;

end if;

        update student\_official

        set deptID=v\_deptID

        where roll\_no=v\_roll\_no;

        IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1003,'Error');

        END IF;

        exception

    when invalid\_official\_deptId then

    dbms\_output.put\_line('Department Id does not exist');

        end update\_student\_official\_deptID;

procedure update\_student\_official\_semester(

        v\_roll\_no in varchar2,

        v\_semester in varchar2)

        is

        begin

        update student\_official

        set semester=v\_semester

        where roll\_no=v\_roll\_no;

        IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1003,'Error');

        END IF;

        end update\_student\_official\_semester;

end student\_official\_pack;

b. Student Personal Operations Pack

create or replace package student\_personal\_operations\_pack as

   PROCEDURE student\_insert(

            v\_last\_name     IN VARCHAR2,

            v\_first\_name  IN VARCHAR2,

            v\_date\_of\_birth IN date,

            v\_roll\_no  IN VARCHAR2,

            v\_email  IN VARCHAR2,

            v\_city  IN VARCHAR2,

            v\_state\_student IN VARCHAR2,

            v\_phone\_number  IN VARCHAR2

           );

    procedure search\_by\_roll\_no(r\_no in varchar2);

    procedure delete\_stu\_data(

        v\_roll\_no in varchar2);

    procedure update\_stu\_name(

        v\_roll\_no in varchar2,

        v\_last\_name in varchar2,

        v\_first\_name in varchar2);

    procedure update\_stu\_email(

        v\_roll\_no in varchar2,

        v\_email in varchar2);

    procedure update\_stu\_phn(

        v\_roll\_no in varchar2,

        v\_phn in varchar2);

    procedure update\_state\_student(

        v\_roll\_no in varchar2,

        v\_state\_student in varchar2);

procedure update\_city(

        v\_roll\_no in varchar2,

        v\_city in varchar2);

procedure update\_date\_of\_birth(

        v\_roll\_no in varchar2,

        v\_date\_of\_birth in varchar2);

end student\_personal\_operations\_pack;

create or replace package body student\_personal\_operations\_pack as

   PROCEDURE student\_insert(

            v\_last\_name     IN VARCHAR2,

            v\_first\_name  IN VARCHAR2,

            v\_date\_of\_birth IN date,

            v\_roll\_no  IN VARCHAR2,

            v\_email  IN VARCHAR2,

            v\_city  IN VARCHAR2,

            v\_state\_student IN VARCHAR2,

            v\_phone\_number  IN VARCHAR2

           )

        is

        BEGIN

            INSERT INTO student\_personal values(v\_last\_name,v\_first\_name,v\_date\_of\_birth,v\_roll\_no,v\_email,v\_city,v\_state\_student,v\_phone\_number);

    END student\_insert;

    procedure search\_by\_roll\_no(r\_no in varchar2)

        is

        v\_fname varchar2(20);

        v\_lname varchar2(20);

        v\_date date;

        v\_rollno varchar2(10);

        v\_email varchar2(20);

        v\_phone varchar2(12);

        v\_state\_student VARCHAR2(20);

        begin

        select first\_name into v\_fname from student\_personal where roll\_no=r\_no;

        select last\_name into v\_lname from student\_personal where roll\_no=r\_no;

        select date\_of\_birth into v\_date from student\_personal where roll\_no=r\_no;

        select roll\_no into v\_rollno from student\_personal where roll\_no=r\_no;

        select email into v\_email from student\_personal where roll\_no=r\_no;

        select phone\_number into v\_phone from student\_personal where roll\_no=r\_no;

        select state\_student into v\_state\_student from student\_personal where roll\_no=r\_no;

        dbms\_output.put\_line(v\_rollno||'  '||v\_fname||' '||v\_lname||'  '||v\_date||'  '||v\_email||'  '||v\_phone||'  '||v\_state\_student);

        IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1000,'Error');

        END IF;

    end search\_by\_roll\_no;

    procedure delete\_stu\_data(

        v\_roll\_no in varchar2)

        is

        begin

        delete from student\_personal

        where roll\_no=v\_roll\_no;

        IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1001,'Error');

        END IF;

        end delete\_stu\_data;

    procedure update\_stu\_name(

        v\_roll\_no in varchar2,

        v\_last\_name in varchar2,

        v\_first\_name in varchar2)

        is

        begin

        update student\_personal

        set last\_name=v\_last\_name,first\_name=v\_first\_name

        where roll\_no=v\_roll\_no;

        IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1002,'Error');

        END IF;

        end update\_stu\_name;

    procedure update\_stu\_email(

        v\_roll\_no in varchar2,

        v\_email in varchar2)

        is

        begin

        update student\_personal

        set email=v\_email

        where roll\_no=v\_roll\_no;

        IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1003,'Error');

        END IF;

        end update\_stu\_email;

    procedure update\_stu\_phn(

        v\_roll\_no in varchar2,

        v\_phn in varchar2)

        is

        begin

        update student\_personal

        set phone\_number=v\_phn

        where roll\_no=v\_roll\_no;

        IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1004,'Error');

        END IF;

        end update\_stu\_phn;

    procedure update\_state\_student(

        v\_roll\_no in varchar2,

       v\_state\_student in varchar2)

        is

        begin

        update student\_personal

        set state\_student=v\_state\_student

        where roll\_no=v\_roll\_no;

        IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1005,'Error');

        END IF;

        end update\_state\_student;

procedure update\_city(

        v\_roll\_no in varchar2,

        v\_city in varchar2)

is

        begin

        update student\_personal

        set city=v\_city

        where roll\_no=v\_roll\_no;

        IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1005,'Error');

        END IF;

        end update\_city;

procedure update\_date\_of\_birth(

        v\_roll\_no in varchar2,

        v\_date\_of\_birth in varchar2)

is

        begin

        update student\_personal

        set date\_of\_birth=v\_date\_of\_birth

        where roll\_no=v\_roll\_no;

        IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1005,'Error');

        END IF;

        end update\_date\_of\_birth;

end student\_personal\_operations\_pack;

c. Exam Operations Pack

create or replace package BODY exam\_operations\_pack as

PROCEDURE exam\_insert(

            v\_deptId IN VARCHAR2,

            v\_examId  IN VARCHAR2,

            v\_exam\_name IN VARCHAR2,

            v\_courseId  IN VARCHAR2

           )

IS

BEGIN

INSERT INTO exam

values(    v\_deptId,

         v\_examId ,

            v\_exam\_name,

            v\_courseId );

END exam\_insert;

PROCEDURE search\_by\_examId(ex\_id in varchar2)

IS

v\_deptId VARCHAR2(20);

v\_examId   VARCHAR2(20);

v\_exam\_name  VARCHAR(40);

v\_courseId   VARCHAR2(20);

BEGIN

SELECT deptId INTO v\_deptId FROM exam WHERE examId =ex\_id;

SELECT examId INTO v\_examId FROM exam WHERE examId =ex\_id;

SELECT exam\_name INTO v\_exam\_name FROM exam WHERE examId =ex\_id;

SELECT courseId INTO v\_courseId FROM exam WHERE examId =ex\_id;

dbms\_output.put\_line(v\_examId || '  ' || v\_deptId ||'  ' ||   v\_exam\_name ||'  ' || v\_courseId);

IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1001,'Error'); --Error code might need to be changed

    END IF;

END search\_by\_examId;

PROCEDURE delete\_exam\_data(v\_examId in varchar2)

IS

BEGIN

DELETE FROM exam

WHERE examId = v\_examId;

IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1001,'Error');

        END IF;

END delete\_exam\_data;

PROCEDURE update\_deptId(v\_examId in varchar2,v\_deptId  IN varchar2)

IS

BEGIN

UPDATE exam

SET deptId = v\_deptId WHERE examId =V\_examId;

IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1001,'Error');

        END IF;

END update\_deptId;

PROCEDURE update\_exam\_name(v\_examId in varchar2,v\_exam\_name  IN varchar2)

IS

BEGIN

UPDATE exam

SET exam\_name = v\_exam\_name WHERE examId =V\_examId;

IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1001,'Error');

        END IF;

END update\_exam\_name;

PROCEDURE update\_courseId(v\_examId in varchar2,v\_courseId  IN varchar2)

IS

BEGIN

UPDATE exam

SET courseId= v\_courseId WHERE examId =V\_examId;

IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1001,'Error');

        END IF;

END update\_courseId;

END exam\_operations\_pack;

d. Section Operations Pack

CREATE OR REPLACE PACKAGE SECTION\_OPERATIONS\_PACK AS

   -- Adds a record

   PROCEDURE insert\_section(p\_sectionid in  SECTION.SECTIONID%type,

   p\_section\_name in SECTION.SECTION\_NAME%type,

   p\_deptid in SECTION.DEPTID%TYPE,

   p\_no\_of\_students in SECTION.NO\_OF\_STUDENTS%TYPE);

   -- Deletes a record

   PROCEDURE delete\_section(p\_sectionid in SECTION.SECTIONID%TYPE);

   --Update the section table

   PROCEDURE update\_section\_name(p\_sectionid in SECTION.SECTIONID%type,

    p\_section\_name in SECTION.SECTION\_NAME%type);

  PROCEDURE update\_no\_of\_students(p\_sectionid in SECTION.SECTIONID%type,

    p\_no\_of\_students in SECTION.NO\_OF\_STUDENTS%type);

END SECTION\_OPERATIONS\_PACK;

-- PACKAGE BODY

CREATE OR REPLACE PACKAGE BODY SECTION\_OPERATIONS\_PACK AS

   PROCEDURE insert\_section (p\_sectionid in SECTION.SECTIONID%type,

      p\_section\_name in SECTION.SECTION\_NAME%type,

      p\_deptid in SECTION.DEPTID%TYPE,

      p\_no\_of\_students in SECTION.NO\_OF\_STUDENTS%TYPE)

   IS

   BEGIN

      INSERT INTO SECTION (SECTIONID,SECTION\_NAME,DEPTID,NO\_OF\_STUDENTS)

         VALUES(p\_sectionid, p\_section\_name,p\_deptid, p\_no\_of\_students);

   END insert\_section;

   PROCEDURE delete\_section(p\_sectionid in SECTION.SECTIONID%type) IS

   BEGIN

      DELETE FROM SECTION

      WHERE SECTIONID = p\_sectionid;

      IF SQL%NOTFOUND THEN

            dbms\_output.put\_line('Invalid Section ID');

        END IF;

   END delete\_section;

   PROCEDURE update\_section\_name (p\_sectionid  in SECTION.SECTIONID%type,

    p\_section\_name in SECTION.SECTION\_NAME%type)

    IS BEGIN

    UPDATE SECTION SET SECTION\_NAME = p\_section\_name

    WHERE SECTIONID = p\_sectionid;

    IF SQL%NOTFOUND THEN

            dbms\_output.put\_line('Invalid Section ID');

        END IF;

    END update\_section\_name;

     PROCEDURE update\_no\_of\_students (p\_sectionid  in SECTION.SECTIONID%type,

    p\_no\_of\_students in SECTION.NO\_OF\_STUDENTS%type)

    IS BEGIN

    if (p\_no\_of\_students > 60) then

    dbms\_output.put\_line('No. of students should not exceed to 60');

    else

    UPDATE SECTION SET NO\_OF\_STUDENTS = p\_no\_of\_students

    WHERE SECTIONID = p\_sectionid;

    IF SQL%NOTFOUND THEN

            dbms\_output.put\_line('Invalid Section ID');

        END IF;

    end if;

    END update\_no\_of\_students;

END SECTION\_OPERATIONS\_PACK;

/

e. Course Operations Pack

CREATE OR REPLACE PACKAGE COURSE\_OPERATIONS\_PACK AS

 PROCEDURE insert\_course(p\_courseid in  COURSE.COURSEID%type,

   p\_course\_name in COURSE.COURSE\_NAME%type

   ,p\_facultyid in COURSE.FACULTYID%TYPE,

   p\_deptid in COURSE.DEPTID%TYPE

   );

    PROCEDURE delete\_course(p\_courseid in COURSE.COURSEID%TYPE);

     PROCEDURE update\_course\_name(p\_courseid in COURSE.COURSEID%type,

    p\_course\_name in COURSE.COURSE\_NAME%type);

      PROCEDURE update\_deptid (p\_courseid in COURSE.COURSEID%type,

    p\_deptid in COURSE.DEPTID%type);

    END COURSE\_OPERATIONS\_PACK;

--PACKAGE BODY

CREATE OR REPLACE PACKAGE BODY COURSE\_OPERATIONS\_PACK AS

PROCEDURE insert\_course(p\_courseid in  COURSE.COURSEID%type,

   p\_course\_name in COURSE.COURSE\_NAME%type

   ,p\_facultyid in COURSE.FACULTYID%TYPE,

   p\_deptid in COURSE.DEPTID%TYPE

   )

   IS BEGIN

   INSERT INTO COURSE (COURSEID,COURSE\_NAME,FACULTYID,DEPTID)

   VALUES (p\_courseid,p\_course\_name,p\_facultyid,p\_deptid);

   END insert\_course;

PROCEDURE delete\_course(p\_courseid in COURSE.COURSEID%TYPE)

    is begin

    DELETE FROM COURSE

    WHERE COURSEID = p\_courseid;

    IF SQL%NOTFOUND THEN

          dbms\_output.put\_line ('Invalid Course ID');

        END IF;

    end delete\_course;

PROCEDURE update\_course\_name(p\_courseid in COURSE.COURSEID%type,

    p\_course\_name in COURSE.COURSE\_NAME%type)

    is begin

    update COURSE SET COURSE\_NAME = p\_course\_name

    where COURSEID = p\_courseid;

     IF SQL%NOTFOUND THEN

          dbms\_output.put\_line ('Invalid Course ID');

        END IF;

    end update\_course\_name;

    PROCEDURE update\_deptid (p\_courseid in COURSE.COURSEID%type,

    p\_deptid in COURSE.DEPTID%type)

    is invalid\_deptId exception;

     begin

    if (p\_deptid != 'CSE\_3' or p\_deptid != 'IT\_1' or p\_deptid != 'EEE\_4' or p\_deptid != 'MECHANICAL\_6' or p\_deptid != 'BIOTECH\_5' or p\_deptid != 'ECE\_2') then

    raise invalid\_deptId;

end if;

    update COURSE SET DEPTID = p\_deptid

    where COURSEID = p\_courseid;

     IF SQL%NOTFOUND THEN

          dbms\_output.put\_line ('Invalid Course ID');

        END IF;

        exception

    when invalid\_deptId then

    dbms\_output.put\_line('Department Id does not exist');

    end update\_deptid;

END COURSE\_OPERATIONS\_PACK;

f. Result Operations Pack

create or replace package result\_operations\_pack as PROCEDURE student\_result\_insert(

            v\_roll\_no  IN VARCHAR2,

            v\_sem1 IN int,

            v\_sem2 IN int,

            v\_sem3 IN int,

            v\_sem4 IN int,

            v\_final\_cgpa IN int

           );

procedure search\_by\_roll\_no\_in\_results(r\_no in varchar2);

procedure delete\_results\_data(

        v\_roll\_no in varchar2);

procedure update\_results\_sem1(

        v\_roll\_no in varchar2,

        v\_sem1 in varchar2);

procedure update\_results\_sem2(

        v\_roll\_no in varchar2,

        v\_sem2 in varchar2);

procedure update\_results\_sem3(

        v\_roll\_no in varchar2,

        v\_sem3 in varchar2);

procedure update\_results\_sem4(

        v\_roll\_no in varchar2,

        v\_sem4 in varchar2);

end result\_operations\_pack;

create or replace package body result\_operations\_pack as

PROCEDURE student\_result\_insert(

            v\_roll\_no  IN VARCHAR2,

            v\_sem1 IN int,

            v\_sem2 IN int,

            v\_sem3 IN int,

            v\_sem4 IN int,

            v\_final\_cgpa IN int

           )

           is

           BEGIN INSERT into results

           values(v\_roll\_no,v\_sem1,v\_sem2,v\_sem3,v\_sem4,v\_final\_cgpa);

END student\_result\_insert;

procedure search\_by\_roll\_no\_in\_results(r\_no in varchar2)

        is

        v\_roll\_no varchar2(20);

        v\_sem1 int;

        v\_sem2 int;

        v\_sem3 int;

        v\_sem4 int;

        v\_final\_cgpa int;

        begin

        select roll\_no into v\_roll\_no from results where roll\_no=r\_no;

        select sem1 into v\_sem1 from results where roll\_no=r\_no;

        select sem2 into v\_sem2 from results where roll\_no=r\_no;

        select sem3 into v\_sem3 from results where roll\_no=r\_no;

        select sem4 into v\_sem4 from results where roll\_no=r\_no;

        select final\_cgpa into v\_final\_cgpa from results where roll\_no=r\_no;

        dbms\_output.put\_line(v\_roll\_no||'  '||v\_sem1||' '||v\_sem2||'  '||v\_sem3||'  '||v\_sem4||'  '||v\_final\_cgpa);

        IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1000,'Error');

        END IF;

end search\_by\_roll\_no\_in\_results;

procedure delete\_results\_data(

        v\_roll\_no in varchar2)

        is

        begin

        delete from results

        where roll\_no=v\_roll\_no;

        IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1001,'Error');

        END IF;

        end delete\_results\_data;

procedure update\_results\_sem1(

        v\_roll\_no in varchar2,

        v\_sem1 in varchar2)

        is

        begin

        update results

        set sem1=v\_sem1

        where roll\_no=v\_roll\_no;

        IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1003,'Error');

        END IF;

        end update\_results\_sem1;

procedure update\_results\_sem2(

        v\_roll\_no in varchar2,

        v\_sem2 in varchar2)

        is

        begin

        update results

        set sem2=v\_sem2

        where roll\_no=v\_roll\_no;

        IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1003,'Error');

        END IF;

        end update\_results\_sem2;

procedure update\_results\_sem3(

        v\_roll\_no in varchar2,

        v\_sem3 in varchar2)

        is

        begin

        update results

        set sem3=v\_sem3

        where roll\_no=v\_roll\_no;

        IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1003,'Error');

        END IF;

        end update\_results\_sem3;

procedure update\_results\_sem4(

        v\_roll\_no in varchar2,

        v\_sem4 in varchar2)

        is

        begin

        update results

        set sem4=v\_sem4

        where roll\_no=v\_roll\_no;

        IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1003,'Error');

        END IF;

        end update\_results\_sem4;

end result\_operations\_pack;

g. Department Operations Pack

CREATE OR REPLACE PACKAGE department\_operations\_pack AS

PROCEDURE department\_insert(v\_deptId  IN varchar2,v\_dept\_name  IN varchar2);

PROCEDURE search\_by\_deptId(departmentId IN varchar2);

PROCEDURE delete\_dept\_data(v\_deptId in varchar2);

PROCEDURE update\_dept\_name(v\_deptId in varchar2,v\_dept\_name  IN varchar2);

end department\_operations\_pack ;

create or replace PACKAGE BODY  department\_operations\_pack AS

PROCEDURE department\_insert(v\_deptId  IN varchar2,v\_dept\_name  IN varchar2)

IS

BEGIN

INSERT INTO department

values(v\_deptId,v\_dept\_name);

END department\_insert;

PROCEDURE search\_by\_deptId(departmentId IN varchar2)

IS

v\_deptId VARCHAR(20);

v\_dept\_name VARCHAR(50);

BEGIN

SELECT deptId INTO v\_deptId FROM department WHERE deptId = departmentId;

SELECT dept\_name INTO v\_dept\_name FROM department WHERE deptId = departmentId;

dbms\_output.put\_line( v\_deptId || ' ' || v\_dept\_name);

IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1000,'Error');--Errorcode might need to be changed

END IF;

END search\_by\_deptId;

PROCEDURE delete\_dept\_data(v\_deptId in varchar2)

IS

BEGIN

 delete from department

        where deptId=v\_deptId;

IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1001,'Error'); --Errorcode might need to be changed

END IF;

END delete\_dept\_data;

PROCEDURE update\_dept\_name(v\_deptId in varchar2,v\_dept\_name  IN varchar2)

IS

BEGIN

UPDATE department

Set dept\_name = v\_dept\_name WHERE deptId = v\_deptId;

IF SQL%NOTFOUND THEN

            RAISE\_APPLICATION\_ERROR(-1002,'Error');--Errorcode might need to be changed

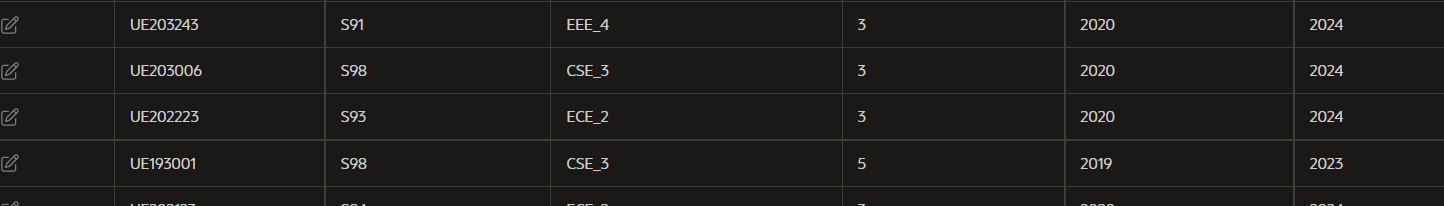
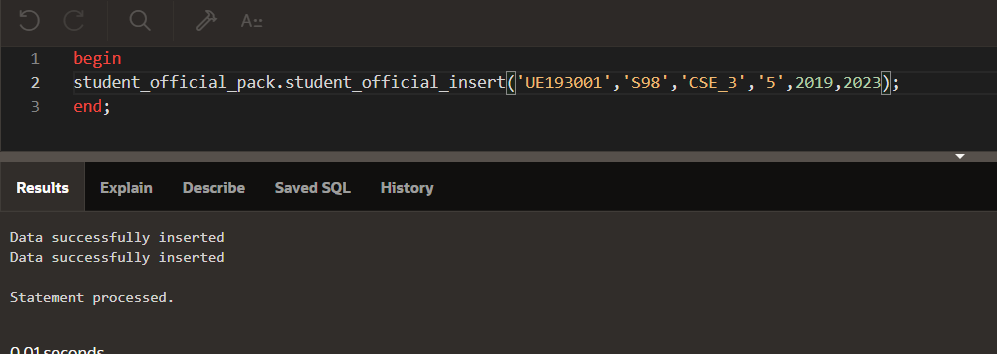
 END IF;

END  update\_dept\_name;

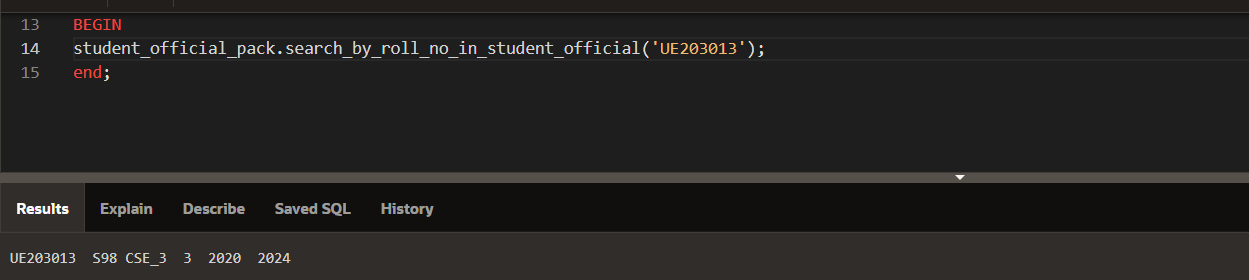
END department\_operations\_pack;

**DEMONSTRATION:**

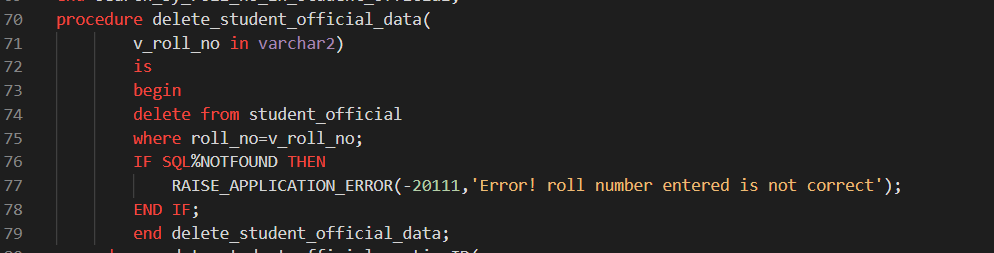
#INSERTION USING PACKAGE

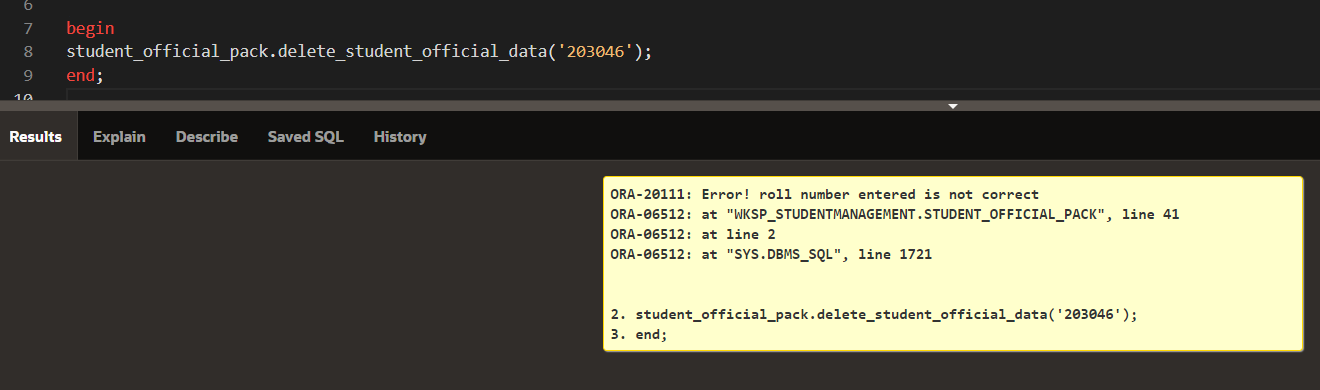


#SEARCHING USING PACKAGE

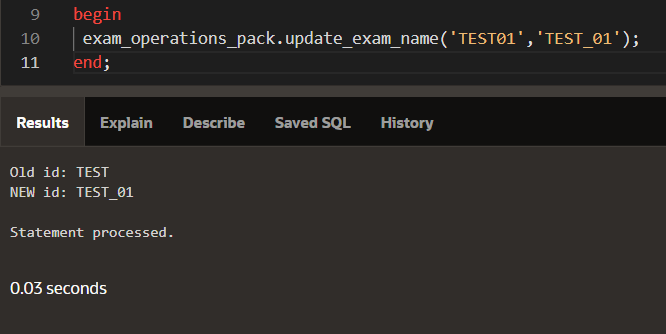
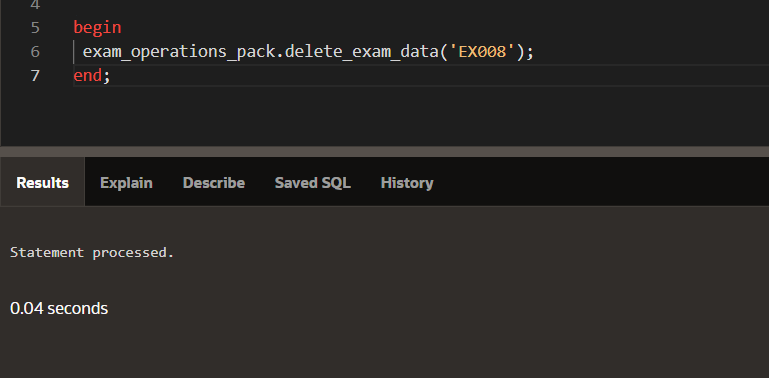


#EXCEPTION HANDLING SS

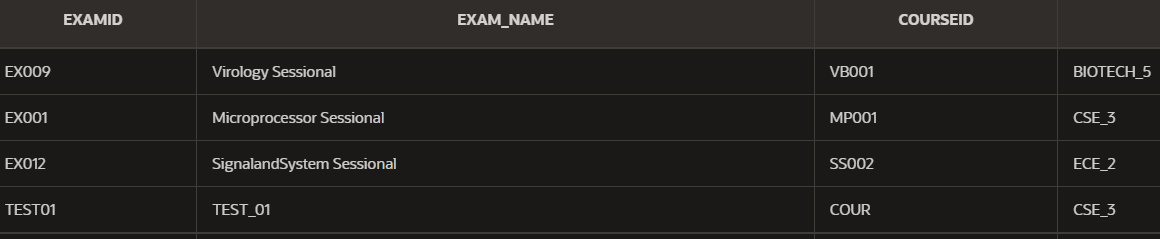




#UPDATION(EXAM NAME) AND DELETION USING PACKAGE

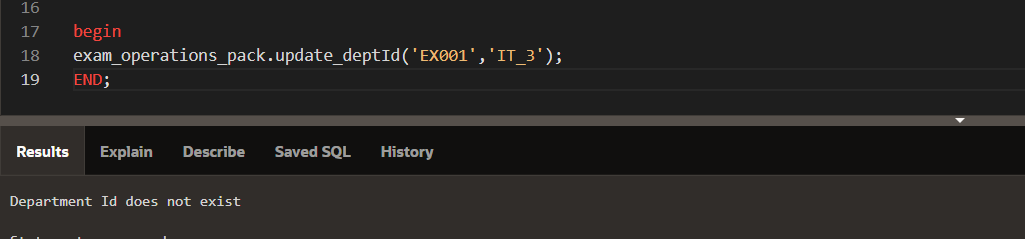


#EXAM TABLE AFTER UPDATION AND DELETION

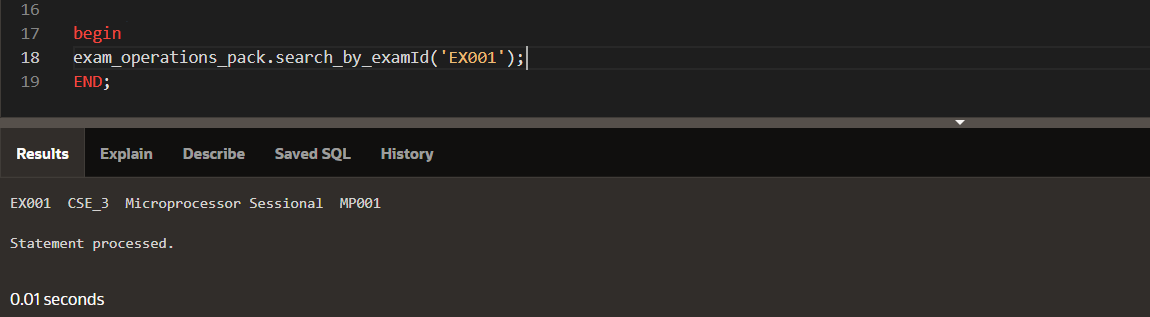


#updating deptid to new deptid which don’t exist

#EXCEPTION HANDLING SS



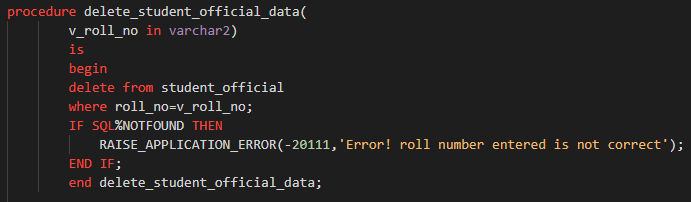
#SEARCHING



1. **CURSOR**

**Implicit Cursor:**

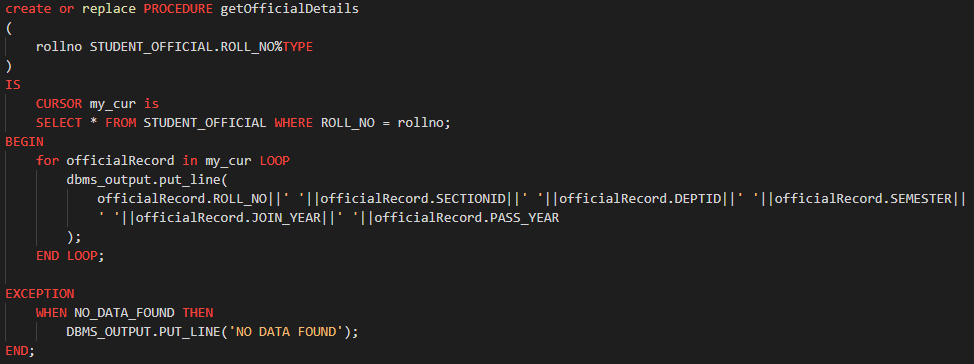
We have used inbuilt cursor (sql) and used to attributes like %FOUND, %NOTFOUND to handle errors in procedures

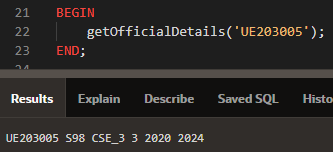
****

****

**Explicit Cursor:**

**We have used explicit cursor in the query to obtain official details of students given their roll numbers. We have avoided much use of cursors as it slows down the performance of query due to row by row processing**.





1. **SOME other QUERIES**

A.

create or replace PROCEDURE getOfficialDetails

(

    rollno STUDENT\_OFFICIAL.ROLL\_NO%TYPE

)

IS

    TYPE of\_rec IS TABLE OF STUDENT\_OFFICIAL%ROWTYPE;

    officialRecord of\_rec;

BEGIN

    SELECT \* BULK COLLECT INTO officialRecord FROM STUDENT\_OFFICIAL WHERE ROLL\_NO = rollno;

    IF officialRecord.count = 0 THEN

        dbms\_output.put\_line('NO DATA FOUND');

    ELSE

        for i in 1..officialRecord.count LOOP

            dbms\_output.put\_line(

                officialRecord(i).ROLL\_NO||' '||officialRecord(i).SECTIONID||' '||officialRecord(i).DEPTID||' '||officialRecord(i).SEMESTER||' '||officialRecord(i).JOIN\_YEAR||' '||officialRecord(i).PASS\_YEAR

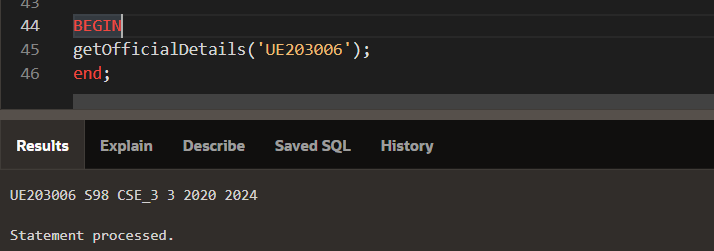
            );

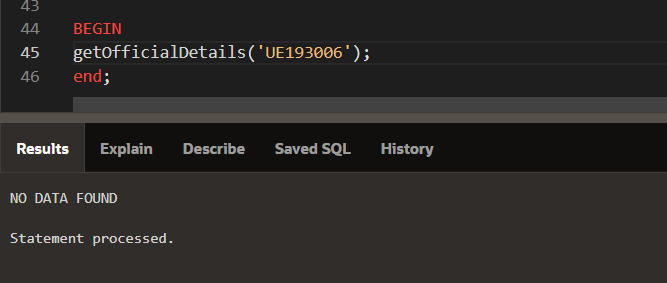
        END LOOP;

    END IF;

END;

Output of query:





B.

create or replace PROCEDURE **getStudentOfficialByDeptName** (

    d\_name DEPARTMENT.DEPT\_NAME % TYPE

)

IS

    TYPE my\_record IS TABLE OF STUDENT\_OFFICIAL%ROWTYPE;

    rec1 my\_record;

    temp\_d\_id DEPARTMENT.DEPT\_NAME%TYPE;

BEGIN

    SELECT DEPTID INTO temp\_d\_id FROM DEPARTMENT WHERE

    DEPT\_NAME = d\_name;

    SELECT \* BULK COLLECT INTO rec1 FROM STUDENT\_OFFICIAL

    WHERE DEPTID = temp\_d\_id;

        --output line required

end;

OUTPUT SS OF QUERY:

C.

create or replace PROCEDURE get\_students\_by\_section (

    sectionid SECTION.SECTIONID%TYPE

)

IS

    TYPE temp\_rec IS RECORD

    (

        f\_name STUDENT\_PERSONAL.FIRST\_NAME%type,

        l\_name STUDENT\_PERSONAL.LAST\_NAME%TYPE,

        r\_no STUDENT\_OFFICIAL.ROLL\_NO%TYPE,

        email\_id STUDENT\_PERSONAL.EMAIL%TYPE

    );

    TYPE getStudentsBySection IS TABLE OF temp\_rec;

    get\_student getStudentsBySection;

BEGIN

    select FIRST\_NAME, LAST\_NAME, STUDENT\_OFFICIAL.ROLL\_NO, EMAIL BULK COLLECT INTO get\_student FROM STUDENT\_OFFICIAL JOIN STUDENT\_PERSONAL

    ON STUDENT\_OFFICIAL.ROLL\_NO = STUDENT\_PERSONAL.ROLL\_NO

    where sectionId = sectionid;

    dbms\_output.put\_line('First\_Name                Last\_Name               Roll\_no             Email');

    for i in 1..get\_student.count LOOP

        dbms\_output.put\_line(get\_student(i).f\_name||' '||get\_student(i).l\_name||' '||get\_student(i).r\_no||' '||get\_student(i).email\_id);

    END LOOP;

end;

OUTPUT OF QUERY:

DATA IN STUDENT OFFICIAL IS WRONG FOR SOME ROLL NO (FIRST NAME AND LAST NAME INTERCHANGED )

D.

create or replace PROCEDURE get\_student\_by\_first\_letter

(

    alphabet VARCHAR2

)

is

    TYPE studentPersonalRecord is TABLE OF STUDENT\_PERSONAL%ROWTYPE;

    rec1 studentPersonalRecord;

BEGIN

    SELECT \* BULK COLLECT INTO rec1

    FROM STUDENT\_PERSONAL WHERE FIRST\_NAME LIKE (alphabet||'%');

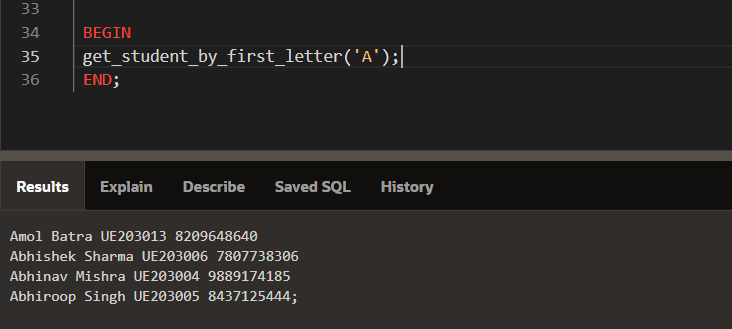
    FOR i in 1..rec1.count LOOP

        dbms\_output.put\_line(rec1(i).FIRST\_NAME||' '||rec1(i).LAST\_NAME||' '||rec1(i).ROLL\_NO||' '||rec1(i).PHONE\_NUMBER);

    END LOOP;

END;

OUTPUT OF QUERY:



E.

create or replace PROCEDURE getPersonalDetails

(

    rollno STUDENT\_PERSONAL.ROLL\_NO%TYPE

)

IS

    TYPE per\_rec IS TABLE OF STUDENT\_PERSONAL%ROWTYPE;

    personalRecord per\_rec;

BEGIN

    SELECT \* BULK COLLECT INTO personalRecord FROM STUDENT\_PERSONAL WHERE ROLL\_NO = rollno;

    for i in 1..personalRecord.count LOOP

        dbms\_output.put\_line(

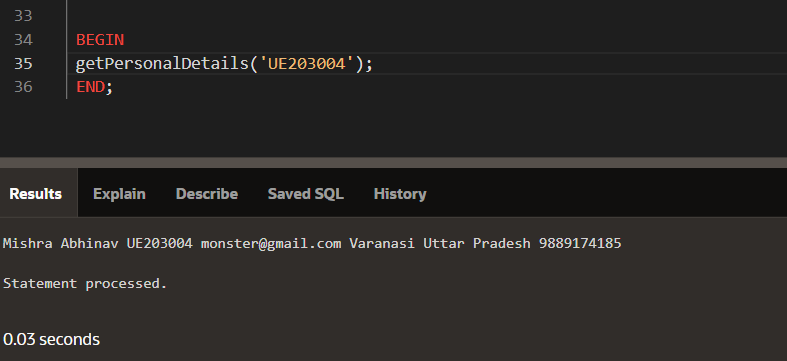
            personalRecord(i).LAST\_NAME||' '||personalRecord(i).FIRST\_NAME||' '||personalRecord(i).ROLL\_NO||' '||personalRecord(i).EMAIL||' '||personalRecord(i).CITY||' '||personalRecord(i).STATE\_STUDENT||' '||personalRecord(i).PHONE\_NUMBER

        );

    END LOOP;

END;

OUTPUT OF QUERY:



vv