# CSCI411 Project 2

Cameron Nieters

Donivan Anderson

Suresh Shrestha

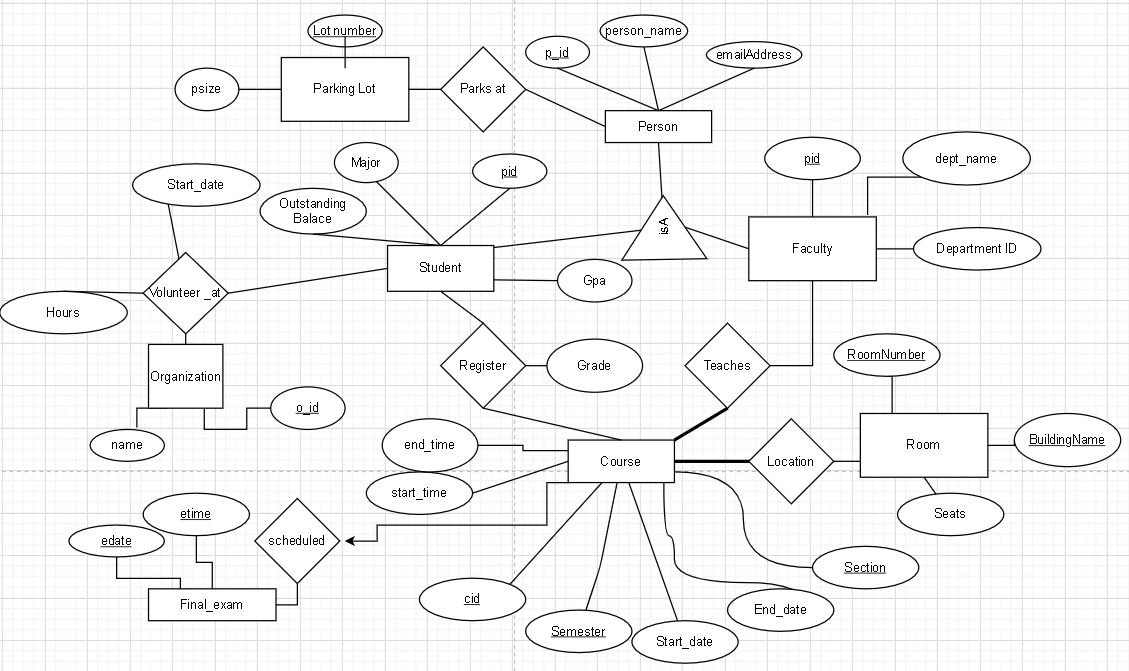
Sajan Khatri

Rabindra Adhikari

**Oracle account: cs411234**

*St. Cloud State University*

ER Diagram



# 

## Tables + Data

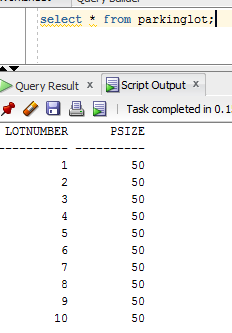
CREATE TABLE ParkingLot(

lotNumber INTEGER,

psize INTEGER,

primary key(lotNumber)

);



CREATE TABLE ROOM(

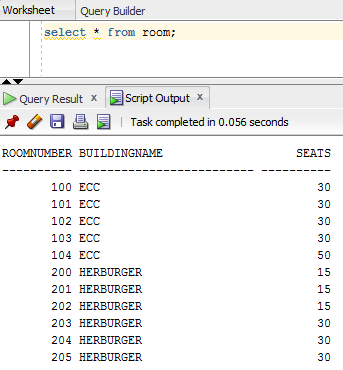
roomNUMBER INTEGER,

buildingNAME CHAR(25),

seats INTEGER,

PRIMARY KEY(roomNUMBER, buildingNAME)

);



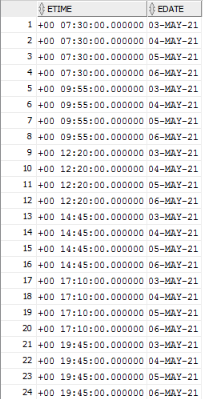
CREATE TABLE final\_exam(

etime INTERVAL DAY(0) TO SECOND(0),

edate DATE,

PRIMARY KEY (etime, edate)

);



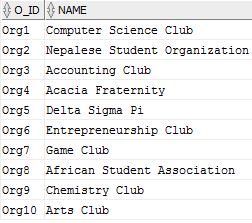
Create Table Organization(

o\_id VARCHAR(50),

name VARCHAR(50),

PRIMARY KEY(o\_id)

);



CREATE TABLE course(

cid VARCHAR(10),

semester VARCHAR(20),

start\_time INTERVAL DAY(0) TO SECOND(0),

end\_time INTERVAL DAY(0) TO SECOND(0),

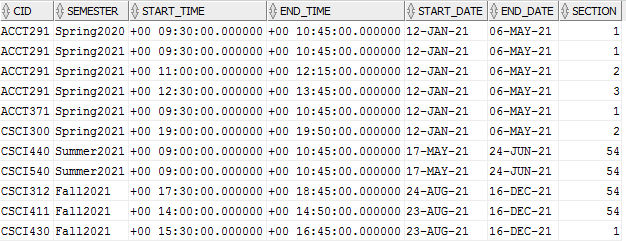
start\_date DATE,

end\_date DATE,

section INTEGER,

PRIMARY KEY (cid, semester, section)

);



CREATE TABLE Person(

pid INTEGER,

person\_name VARCHAR(20),

email VARCHAR (50),

PRIMARY KEY(pid)

);



CREATE TABLE Faculty(

pid INTEGER,

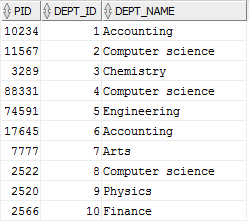
dept\_id INTEGER,

dept\_name CHAR(20),

PRIMARY KEY(pid),

FOREIGN KEY(pid) REFERENCES Person(pid)

);



Create Table Student(

pid INTEGER,

major VARCHAR(50),

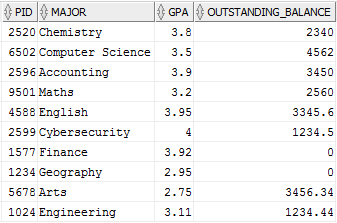
gpa REAL,

outstanding\_balance REAL,

PRIMARY KEY(pid),

FOREIGN KEY(pid) REFERENCES person(pid)

);



Create Table Volunteer\_at(

pid INTEGER,

o\_id VARCHAR(50),

hours REAL,

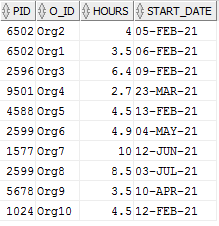
start\_date DATE,

PRIMARY KEY(pid, o\_id),

FOREIGN KEY(pid) References Student(pid),

FOREIGN KEY(o\_id) References Organization(o\_id)

);



CREATE TABLE scheduled(

cid VARCHAR(10),

semester varchar(20),

section INTEGER,

FOREIGN KEY (cid, semester, section)

REFERENCES course(cid, semester, section),

etime INTERVAL DAY(0) TO SECOND(0),

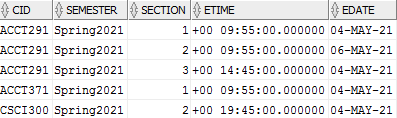
edate DATE,

FOREIGN KEY (etime, edate) REFERENCES final\_exam(etime, edate),

PRIMARY KEY (cid, semester, etime, edate),

UNIQUE (cid, semester, section)

);



CREATE TABLE Teaches(

pid INTEGER,

cid VARCHAR(10),

semester VARCHAR(20),

section INTEGER,

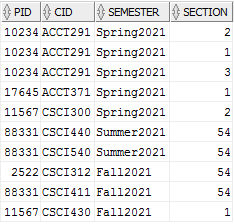
PRIMARY KEY(pid,cid,semester,section),

FOREIGN KEY (pid) REFERENCES Faculty(pid),

FOREIGN KEY (cid,semester,section)

REFERENCES course(cid, semester,section)

);



CREATE TABLE LOCATION(

cid VARCHAR(10),

semester varchar(20),

section INTEGER,

roomNUMBER INTEGER,

buildingNAME CHAR(25),

PRIMARY KEY (cid, semester, section, roomNUMBER, buildingNAME),

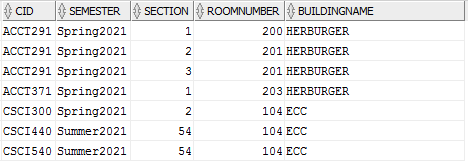
FOREIGN KEY (cid, semester, section)

REFERENCES Course (cid, semester, section),

FOREIGN KEY (roomNUMBER, buildingNAME)

REFERENCES ROOM (roomNUMBER, buildingNAME)

);



CREATE TABLE ParksAt(

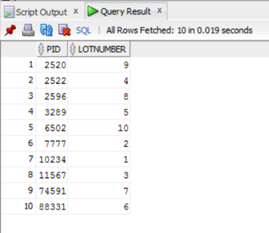
pid INTEGER,

lotNumber INTEGER,

primary key(pid, lotNumber),

FOREIGN KEY(pid) REFERENCES Person(pid)

);



CREATE TABLE Register(

cid varchar(10),

semester varchar(20),

section INTEGER,

pid INTEGER,

grade REAL,

primary key(cid, semester, section, pid),

foreign key(pid) references Student(pid),

foreign key(cid, semester, section)

references course(cid, semester, section)

);

## 

## Queries + Stored Procedures

\* - **Stored procedure**

\*1. List students with an outstanding balance

\* 2. Print transcript for student

3. List students without volunteer work

4. List students with parking permits

5. Find the Courses taught in 'Fall2021' also list the name of the professor teaching that class

\*6. Given department name, Find the parking lot number and parking size of all faculty in a specific department (same department might have different lot numbers).

\* 7. List students with >= user-specified GPA

8. Print average grade of all courses

\* 9. List the students volunteered hours and the organization name with user specified student Id.

10. Find the total number of courses in ‘Spring2021’. Display the semester name and total count.

11. Print student names who have conflicting class times in their schedule

\*12. Print email addresses of students who did volunteer work for specific organizations with user specified organization ID.

\* 13. List students with <= user-specified grade for course

14. List parking lots with available spaces

\*15. Given the pid, List students taking a specific major & their gpa

16. List students who retook a course

17.Find the course schedule for ‘CSCI440’ for ‘Summer2021’. List the course id, room number , building name and class time.

18. Print final exam time for all courses

\*19. List faculty & their contact information in specific department

\*20. List courses that happen between 2 user-specified times

\*1. List students with an outstanding balance

CREATE PROCEDURE outstandingBalance AS

sname CHAR(25);

balance INTEGER;

CURSOR stud IS SELECT P.person\_name, S.outstanding\_balance

FROM student S, Person P

WHERE S.pid = P.pid AND

S.outstanding\_balance > 0;

BEGIN

OPEN stud;

LOOP

FETCH stud INTO sname, balance;

IF stud%NOTFOUND THEN EXIT; END IF;

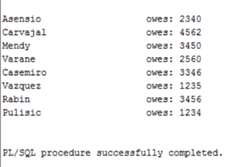
DBMS\_OUTPUT.PUT\_LINE(sname || ' owes: ' || balance);

END LOOP;

CLOSE stud;

END;

**Result**:



**Justification**: There are only 8 students in the data that we entered with an outstanding balance.

2. Print transcript for student

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE showTranscript (studentID in INTEGER)

AS gradeS REAL;

courseID char;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('STUDENT ID:' || studentID);

FOR rec IN

(

SELECT grade, cid INTO gradeS, courseid

FROM

Register R

where R.pid = studentID)

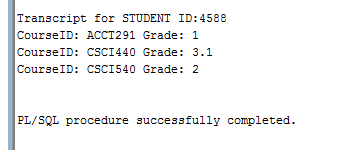
LOOP

DBMS\_OUTPUT.PUT\_LINE('CourseID: ' || rec.cid || ' Grade: ' || rec.grade);

END LOOP;

END;

**Result**:



**Justification**: The student with ID 4588 had 3 courses registered and the grades were shown for the courses too.

3. List students without volunteer work

SELECT \* FROM Person

WHERE

pid IN(

SELECT pid FROM Student

WHERE

pid NOT IN (

SELECT pid

from volunteer\_at));

**Result**:



**Justification**: There are only two students without volunteer work in the database.

4. List students with parking permits

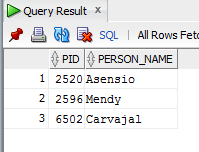
SELECT S.pid, P.person\_name

From student S, Person P, parksAt PA

WHERE S.pid= P.pid AND

P.pid = PA.pid ;

**Result**:



**Justification**: There are all together 3 students who have parking permit

5. .Find the Courses taught in 'Fall2021' also list the name of the professor teaching that class

select c.cid, p.person\_name

from Course C, Teaches T, Faculty F, Person P

Where P.pid= T.pid and

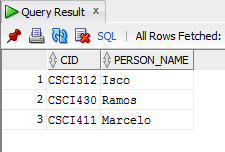
c.cid= T.cid and

c.semester= T.semester and

T.pid= F.pid and

c.semester='Fall2021';

**Result**:

****

**Justification**: There are altogether 3 courses taught in “fall2021” with specific professor related to the particular courses

\*6. Given department name, Find the parking lot number and parking size of all faculty in a specific department(same department might have different lot number).

create or replace PROCEDURE department\_parking( dep\_name In CHAR)

IS

BEGIN

FOR var IN

(SELECT F.dept\_name as deptname, Pa.lotnumber AS Lotnumber, pa.psize AS Parking\_size

FROM PERSON P, FACULTY F,parksat Pr, parkinglot pa

WHERE F.dept\_name = dep\_name AND

P.PID = F.PID and

p.pid=pr.pid and

pr.lotnumber= pa.lotnumber)

LOOP

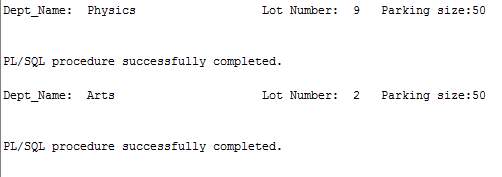
DBMS\_OUTPUT.PUT\_LINE('Dept\_Name: '|| VAR.deptname || ' Lot Number: ' || VAR.Lotnumber || ' Parking size:'|| VAR.Parking\_size);

END LOOP;

END;



**Result**:



**Justification**: from the result we can clearly seen that, we can get total number of parking size and it’s lots number for their department assigned.

\* 7. List students with >= user-specified GPA

CREATE OR REPLACE PROCEDURE list\_gpa\_students(

targ\_gpa IN REAL

) AS

sname VARCHAR(20);

sgpa REAL;

CURSOR stud IS

SELECT P.person\_name, S.gpa

FROM Person P, Student S

WHERE P.pid = S.pid AND S.gpa >= targ\_gpa;

BEGIN

OPEN stud;

LOOP

FETCH stud INTO sname, sgpa;

IF stud%NOTFOUND THEN EXIT; END IF;

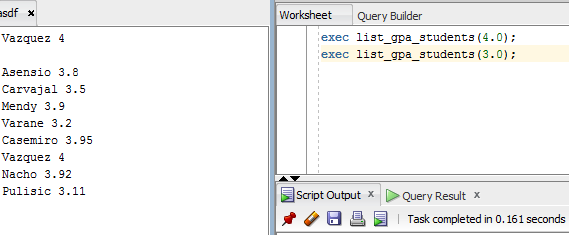
DBMS\_OUTPUT.PUT\_LINE(sname || ' ' || sgpa);

END LOOP;

CLOSE stud;

END;

**Result**:



**Justification**: The outputs of the two commands are separated by a blank line in the image. The first command only shows the one student with a 4.0 GPA, while the next one shows all students with a GPA greater than or equal to 3.0.

8. Print average grade of all courses

SELECT C.cid, C.semester, AVG(COALESCE(R.grade,0))

FROM course C

LEFT OUTER JOIN Register R ON

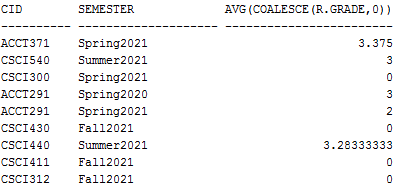
C.cid = R.cid

AND C.semester = R.semester

AND C.section = R.section

GROUP BY C.cid, C.semester;

**Result**:



**Justification**: The grades of the courses are averaged and courses that nobody took are still listed with a grade of 0. The sections are considered together, so only the course ID and semester are grouped.

\* 9. List the students volunteered hours and the organization name with user specified student id . Display student id , organization they volunteered and the hours they volunteered.

**Store Procedure:**

create or replace PROCEDURE volunteer\_hours(p\_id in INTEGER) as

student\_id INTEGER;

org\_name VARCHAR(50);

vol\_hours Real;

cursor volunteer is

select S.pid , Org.name, v.hours

From Student S, Volunteer\_at V, Organization Org

Where S.pid=p\_id and

S.pid=V.pid and

V.o\_id=Org.o\_id;

Begin

open volunteer;

Loop

fetch volunteer into student\_id, org\_name, vol\_hours;

if volunteer%notfound then exit; end if;

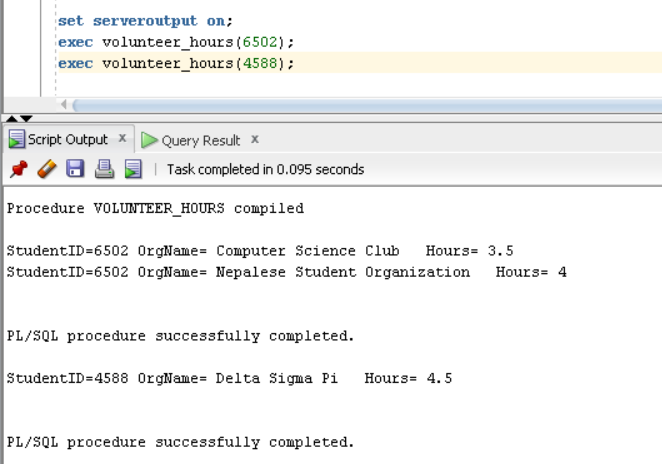
dbms\_output.put\_line('StudentID='|| Student\_id || ' OrgName= ' || org\_name || ' Hours= ' || vol\_hours);

END loop;

close volunteer;

end;

**Result:**

****

**Justification:** Since students can volunteer for more than one student organization. Students with id 6502 volunteered for two organizations and the volunteer hours are shown. But the student with student id 4588 volunteered for only one organization for certain hours . This gives the result of our store procedure as expected.

10. Find the total number of courses in ‘Spring2021’. Display the semester name and total count.

**Query**

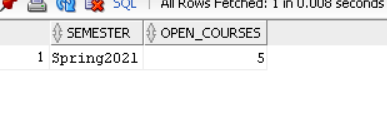
Select c.semester, count(c.cid) as total\_courses

From Course C

Where c.semester='Spring2021'

group by c.semester;

**Result**



**Justification:** There are 5 courses taught in Spring 2021 under the course. The result displays the semester name and the total no of course offered in that semester.

11. Print student names who have conflicting class times in their schedule

Query:

SELECT person\_name

from person

where pid in(

SELECT DISTINCT r.pid

FROM Register R, REGISTER S, course a, course b

where R.pid = S.pid

AND R.cid = a.cid

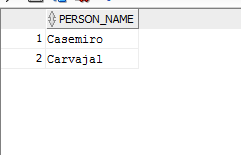
AND S.cid = b.cid

AND a.cid != b.cid

AND a.START\_TIME =b.START\_TIME

AND a.START\_DATE = b.START\_DATE);

**Result**:



**Justification**: Two students whose name was Casemiro and Carvajal had classes with conflicting time.

\*12. Print email addresses of students who did volunteer work for specific organizations given the organization Id. Display organization id and email address of student volunteers.

**Store Procedure:**

create or replace PROCEDURE volunteer\_email( org\_id in VARCHAR) as

orgz\_id varchar(50);

student\_email varchar(50);

cursor email\_address is

Select V.o\_id, P.email into orgz\_id, student\_email

From Person P, Student S, Volunteer\_at V

Where V.o\_id= org\_id and

S.pid= P.pid and

V.pid= S.pid;

Begin

open email\_address;

Loop

fetch email\_address into orgz\_id, student\_email ;

if email\_address%notfound then exit; end if;

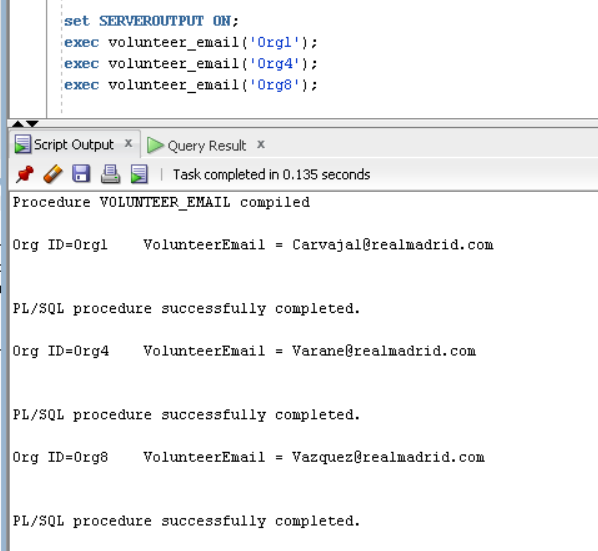
dbms\_output.put\_line('Org ID='|| orgz\_id ||' VolunteerEmail = '|| student\_email) ;

end loop;

close email\_address;

end;

**Result:**



**Justification**:Every person has an email. Student is also a person.Some student volunteers at a specific organization . For ‘Org4’ and ‘Org8’ the same student volunteered so the same volunteer email is shown. ‘Org1’ had different students who volunteered for them so different emails are shown. All these organizations had only one student volunteering for them.

\* 13. List students with <= user-specified grade for course

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE getStudentsLESSTHAN (X in REAL, courseID in char)

AS

studentID NUMBER;

BEGIN

FOR rec IN

( SELECT R.pid INTO studentID

FROM Register R

Where

((R.grade <= X) AND (R.cid = courseID) ))

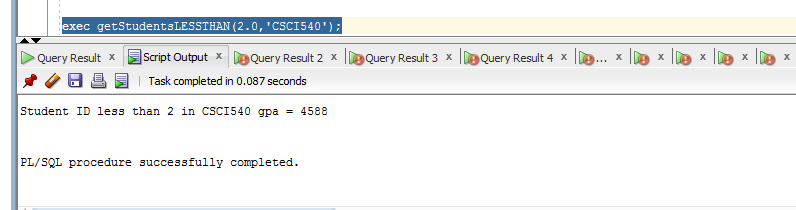
LOOP

DBMS\_OUTPUT.PUT\_LINE('Student ID less than '|| X || ' in ' || courseID || ' gpa = ' || rec.pid);

END LOOP;

END;

**Result**:



**Justification**: Student ID was 4588 who had less than 2.0 gpa in CourseID CSCI 540.

14. List parking lots with available spaces

SELECT Pl.lotNumber, Pl.psize, COUNT(Pe.pid)

FROM ParkingLot Pl, Person Pe, ParksAt Pa

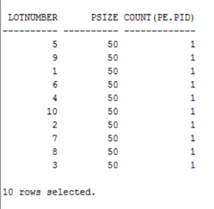
WHERE Pe.pid = Pa.pid AND

Pa.lotNumber = Pl.LotNumber

GROUP BY Pl.lotNumber, Pl.psize

HAVING COUNT(Pe.pid)<Pl.psize;

**Result**:



**Justification**: All of our parking lots have one entry and there are 10 parking lots.

\*15. Given the pid, List students taking a specific major & their gpa

create or replace PROCEDURE GRADES\_FOR\_SEMESTER( p\_pid In INTEGER)

as

PersonName Varchar(50);

MajorName Varchar(50);

Stugrade REAL;

BEGIN

SELECT S.MAJOR, S.gpa , P.PERSON\_NAME into MajorName,Stugrade, PersonName

FROM STUDENT S, PERSON P

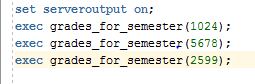
WHERE S.PID = p\_pid AND

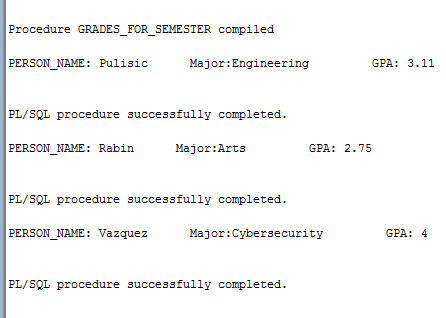
P.pid = s.pid;

DBMS\_OUTPUT.PUT\_LINE(' PERSON\_NAME: '||PersonName || ' Major:' || MajorName || ' GPA: '||Stugrade);

END;

**Result**:





**Justification**: as we saw in the result, we can get the information of the students including their NAME, MAJOR and GPA by providing specific Person ID

16. List students who retook a course

SELECT P.person\_name, C.cid, COUNT(\*)

FROM Person P, Student S, Register R, course C

WHERE P.pid = S.pid AND

S.pid = R.pid AND

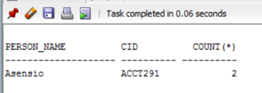
R.cid = C.cid AND

R.section = C.section

GROUP BY P.person\_name, C.cid

HAVING COUNT(\*)>1;

**Result**:



**Justification**: We entered one duplicate course, if the student has the course listed twice then that means they have retaken the course.

17.Find the course schedule for ‘CSCI440’ for ‘Summer2021’. Display the course id, room number , building name and class time.

**Query**

Select C.cid, R.roomnumber, R.buildingname, C.start\_time

From Course C, Location L, Room R

Where C.cid=L.cid and

c.semester=L.semester and

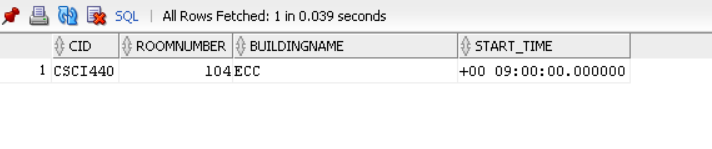
L.roomnumber=R.roomnumber and

L.buildingname=R.buildingname and

c.cid='CSCI440' and

c.semester='Summer2021';

**Result:**



**Justification:**

‘CSCI440’ has only one section. And the course schedule for ‘Summer2021’. It is taught in room 104 in ECC as is shown in the above result

18. Print final exam time for all courses

SELECT C.cid, C.section, E.etime, E.edate

FROM course C, scheduled S, final\_exam E

WHERE C.cid = S.cid

AND C.semester = S.semester

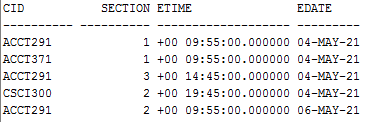
AND C.section = S.section

AND S.etime = E.etime

AND S.edate = E.edate

ORDER BY E.edate, E.etime;

**Result**:



**Justification**: The Spring2021 courses are the only courses with scheduled finals. All those courses are printed for their respective dates as described by the *scheduled* table.

\*19. List faculty & their contact information in specific department

CREATE OR REPLACE PROCEDURE contact\_faculty(

department\_name IN CHAR

) AS

fname VARCHAR(20);

femail VARCHAR(50);

CURSOR fac IS

SELECT P.person\_name, P.email

FROM Person P, Faculty F

WHERE P.pid = F.pid AND F.dept\_name = department\_name

ORDER BY P.person\_name;

BEGIN

OPEN fac;

LOOP

FETCH fac INTO fname, femail;

IF fac%NOTFOUND THEN EXIT; END IF;

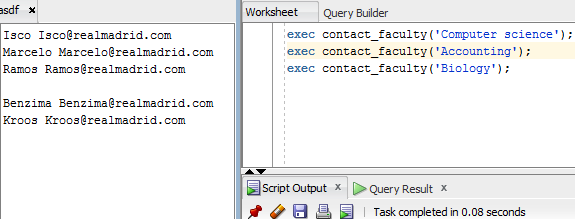
DBMS\_OUTPUT.PUT\_LINE(fname || ' ' || femail);

END LOOP;

CLOSE fac;

END;

**Result**:



**Justification**: The output of each exec is separated by a blank line. The first printed the entire computer science department, the second the accounting department, and the last one printed nothing because there is no biology department in our database.

\*20. List courses that happen between 2 user-specified times

CREATE PROCEDURE classes\_between\_times (lowerbound IN CHAR, upperbound IN CHAR) AS

classsection INTEGER;

classnumber CHAR(10);

classstart INTERVAL DAY(0) TO SECOND(0);

classend INTERVAL DAY(0) TO SECOND(0);

CURSOR Time\_manager is SELECT C.section, C.cid, C.start\_time, C.end\_time

FROM Course C

WHERE lowerbound <= C.start\_time AND

upperbound >= C.end\_time;

BEGIN

OPEN Time\_manager;

LOOP

FETCH Time\_manager INTO classsection, classnumber, classstart, classend;

IF Time\_manager%NOTFOUND THEN EXIT; END IF;

DBMS\_OUTPUT.PUT\_LINE(classnumber || ' Section ' || classsection || ' ' || classstart || ' TO ' || classend);

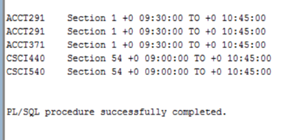
END LOOP;

CLOSE Time\_manager;

END;

exec classes\_between\_times(TO\_DSINTERVAL('0 9:00:00'), TO\_DSINTERVAL('0 10:45:00'));

**Result**:



**Justification**: These are all the classes that have times >= 9:00 and <= 10:45.