COMMANDS:

STUDENT:

CREATE TABLE “STUDENT” (

student\_university\_id numeric(10) primary key,

first\_name varchar(100) not null,

last\_name varchar(100) not null,

middle\_name varchar(100),

date\_of\_birth date not null CHECK(date\_of\_birth<current\_date),

international\_yn boolean not null,

pre\_majors\_yn boolean,

majors\_yn boolean,

under\_represented\_group\_name varchar(50),

email varchar(100) not null,

phone numeric (10),

street varchar(50),

apartment varchar(50),

city varchar(100),

state varchar(50),

zip numeric (5),

resume\_location varchar(200)

);

COURSE:

CREATE TABLE “COURSE” (

course\_code varchar(10) primary key,

course\_name varchar(100) not null,

credit\_hours numeric(2) DEFAULT 0 CHECK(course\_fee>=0),

department varchar(5),

course\_fee numeric(10,5) DEFAULT 0 CHECK(course\_fee>=0),

online\_yn boolean,

required\_for\_major\_transition\_yn boolean not null,

);

PROGRAM:

CREATE TABLE “PROGRAM” (

program\_type varchar(50),

concentration varchar(100),

description varchar(200),

total\_credits\_needed integer DEFAULT 0 CHECK(total\_credits\_needed>=0),

total\_core\_credits\_needed integer DEFAULT 0 CHECK(total\_core\_credits\_needed>=0),

primary key(program\_type, concentration),

constraint credits\_check CHECK(total\_credits\_needed>= total\_core\_credits\_needed)

);

Create domain GRADE varchar(15) NOT NULL CHECK(VALUE IN (‘A’,’B’,’C’,’D’,’F’,’W’));

CREATE TABLE “PRE\_REQUISITE” (

course\_code varchar(10),

pre\_requisite\_course\_code varchar(10) references COURSE(course\_code),

minimum\_grade\_required grade,

primary key(course\_code, pre\_requisite\_course\_code)

);

CREATE TABLE “PROGRAM\_REQUIREMENT” (

program\_type varchar(50) ,

concentration varchar(100),

course\_code varchar(10),

minimum\_grade\_required grade,

primary key(program\_type , concentration ,course\_code),

foreign key(program\_type , concentration) references PROGRAM(program\_type , concentration)

);

Create domain SEMESTER varchar(15) NOT NULL CHECK(VALUE IN (‘fall’,’spring’,’’summer”));

Create domain COURSE\_STATUS varchar(20) NOT NULL CHECK(VALUE IN (‘registered’,’completed’,’withdrawn’,’failed’));

CREATE TABLE “ENROLLED\_ON” (

student\_university\_id numeric(10) references STUDENT(student\_university\_id),

course\_code varchar(10) references COURSE(course\_code),

semester semester,

year numeric(4) CHECK(year>0),

start\_date date DEFAULT current\_date,

end\_date date,

course\_status course\_status,

grade grade,

primary key(student\_university\_id, course\_code, semester , year),

constraint course\_enrolled\_dates CHECK(start\_date<end\_date)

);

Create domain SURVEY varchar(20) NOT NULL CHECK(VALUE IN (‘pre-major’,’major’,’exit’));

CREATE TABLE “SURVEY” (

survey\_type survey,

question varchar(400),

primary key(survey\_type, question)

);

Create table “STUDENT\_PEER\_MENTOR” (

student\_university\_id numeric(10) references STUDENT(student\_university\_id),

mentor\_id numeric(10) references STUDENT(student\_university\_id),

semester semester,

year numeric(4) CHECK(year>0)

primary key(student\_university\_id, semester, year)

);

Create table “GIVES” (

student\_university\_id numeric(10) references STUDENT(student\_university\_id),

survey\_type survey\_type,

question varchar(400),

survey\_date date DEFAULT current\_date,

rating integer CHECK(0<=rating<=5),

comment varchar(200),

foreign key(survey\_type, question) references SURVEY(survey\_type, question),

primary key(student\_university\_id, survey\_type, question, survey\_date)

);

Create table “INCIDENT” (

incident\_category varchar(20),

description varchar(200),

primary key(incident\_category)

);

Create table “FOLLOW\_UP” (

student\_university\_id numeric(10) references STUDENT(student\_university\_id),

incident\_category varchar(20) references INCIDENT(incident\_category),

incident\_date date CHECK(incident\_date<=current\_date),

incident\_time time,

necessary\_action varchar(200),

primary key(student\_university\_id, incident\_category, incident\_category, incident\_time)

);

Create table “OPTS\_FOR” (

student\_university\_id numeric(10) references STUDENT(student\_university\_id),

program\_type varchar(50) ,

concentration varchar(100),

opted\_semester semester,

opted\_year numeric(4) CHECK(opted\_year>0),

start\_date date DEFAULT current\_date,

end\_date date,

overall\_grade grade,

status varchar(20) ,

primary key(student\_university\_id, program\_type, concentration, opted\_semester, opted\_year),

foreign key(program\_type , concentration) references PROGRAM(program\_type , concentration),

constraint program\_dates\_check CHECK(start\_date<end\_date)

);

INSERT COMMANDS:

STUDENT:

INSERT INTO STUDENT(student\_university\_id, first\_name, last\_name, date\_of\_birth, international\_yn, pre\_majors\_yn, majors\_yn, under\_represented\_group\_name , email,phone,street, apartment, city, state,zip, resume\_location)

VALUES(

(101960001,’Virat’,’Kohli’,02-10-1985,True,True,False,’Asian’,’vk@cs.edu’, 1505666901, “Gandhi Ave”, ‘Apt 1’, ‘Raja Apts’, ‘ABQ’,’NM’,87101,’www.cs.edu/resume/101960001.pdf’),

(101960002,’MS’,Dhoni,12-02-1981,False,True,False,’Asian’,’ms@cs.edu’, 1505621702, ‘Nehru Ave’, ‘Apt 2’, ‘Rani Apts’, ‘ABQ’,’NM’,87106,’www.cs.edu/resume/101960002.pdf’),

(101960003,’Jhon’,’Cena’,03-06-1990,False,True,True,’’,’jc@cs.edu’, 1415621602, ‘Lead Ave’, ‘Apt 9’, ‘RJ Apts’, ‘ABQ’,’NM’,87110,’www.cs.edu/resume/101960003.pdf’),

(101960004,’Randy’,’Ortan’ ,01-01-1991,True,True,True,’’,’ro@cs.edu’, 1765621602, ‘Copper Ave’, ‘Apt 9’, ‘Kings Apts’, ‘ABQ’,’NM’,87109,’www.cs.edu/resume/101960004.pdf’),

(101960005,’Aakarsh’,’Nadella’ ,01-03-1995,True,False,True,’Hispanic’,’an@cs.edu’, 1765621505, ‘Gold Ave’, ‘Apt 9’, ‘Queens Apts’, ‘ABQ’,’NM’,87100,’www.cs.edu/resume/101960005.pdf’)

);

**DATA INSERTION:**

To insert the data, we create the data for each of the tables in a comma separated value(CSV) file and import them into postgres using the below commands.

COPY STUDENT(student\_university\_id, first\_name, last\_name, date\_of\_birth, international\_yn, pre\_majors\_yn, majors\_yn, under\_represented\_group\_name , email,phone,street, apartment, city, state,zip, resume\_location)

FROM ‘./STUDENT.csv’ DELIMITER ‘,’ CSV HEADER

\copy STUDENT(student\_university\_id, first\_name, last\_name, date\_of\_birth, international\_yn, pre\_majors\_yn, majors\_yn, under\_represented\_group\_name , email,phone,street, apartment, city, state,zip, resume\_location)

FROM '/STUDENT.csv' DELIMITER ',' CSV HEADER

\copy COURSE(course\_code,course\_name,department,course\_fee,online\_yn,required\_for\_major\_transition\_yn)

FROM './COURSE.csv' DELIMITER ',' CSV HEADER encoding ‘windows-1251’

\copy PROGRAM(program\_type,concentration,description, total\_credits\_needed, total\_core\_credits\_needed )

FROM './PROGRAM.csv' DELIMITER ',' CSV HEADER encoding ‘windows-1251’

\copy PRE\_REQUISITE(course\_code,pre\_requisite\_course\_code, minimum\_grade\_required)

FROM './PRE-REQUISITE.csv' DELIMITER ',' CSV HEADER encoding ‘windows-1251’

\copy PROGRAM\_REQUIREMENT(program\_type, concentration, course\_code, minimum\_grade\_required)

FROM './PROGRAM\_REQUIREMENT.csv' DELIMITER ',' CSV HEADER encoding ‘windows-1251’

\copy ENROLLED\_ON(student\_university\_id, course\_code, semester, year, start\_date, end\_date, course\_status, grade)

FROM './ENROLLED\_ON.csv' DELIMITER ',' CSV HEADER encoding ‘windows-1251’

\copy SURVEY(survey\_type,question)

FROM './SURVEY\_ON.csv' DELIMITER ',' CSV HEADER encoding ‘windows-1251’

\copy STUDENT\_PEER\_MENTOR(student\_university\_id, mentor\_id, semester, year)

FROM './ STUDENT\_PEER\_MENTOR.csv' DELIMITER ',' CSV HEADER encoding ‘windows-1251’

\copy GIVES(student\_university\_id, survey\_type, question, survey\_date,rating,comment)

FROM './GIVES.csv' DELIMITER ',' CSV HEADER encoding ‘windows-1251’

\copy INCIDENT(incident\_category, description)

FROM './INCIDENT.csv' DELIMITER ',' CSV HEADER encoding ‘windows-1251’

\copy FOLLOW\_UP(student\_university\_id, incident\_category, incident\_date,incident\_time, necessary\_action )

FROM './FOLLOW\_UP.csv' DELIMITER ',' CSV HEADER encoding ‘windows-1251’

\copy OPTS\_FOR(student\_university\_id, program\_type, concentration, opted\_semester

,year,start\_date,end\_date, overall\_grade,status )

FROM './OPTS\_FOR.csv' DELIMITER ',' CSV HEADER encoding ‘windows-1251’

\copy ACTIVITY(activity\_name, activity\_type, organized\_by, organizer\_mail, organizer\_phone)

FROM './ ACTIVITY.csv' DELIMITER ',' CSV HEADER encoding ‘windows-1251’

\copy EMPLOYEE(employee\_id, ssn, first\_name, middle\_name, last\_name, email, work\_experience, salary)

FROM './ EMPLOYEE.csv' DELIMITER ',' CSV HEADER encoding ‘windows-1251’

copy FACULTY (employee\_id, specialization, research\_interest )

FROM './ FACULTY.csv' DELIMITER ',' CSV HEADER encoding ‘windows-1251’

\copy ADVISEMENT (student\_university\_id,employee\_id, advisement\_type, year, semester)

FROM './ADVISEMENT.csv' DELIMITER ',' CSV HEADER encoding ‘windows-1251

\copy TUTORING (course\_code,employee\_id)

FROM './TUTORING.csv' DELIMITER ',' CSV HEADER encoding ‘windows-1251

\copy TAUGHT\_BY (course\_code,employee\_id,semester, year, lecture\_hall\_name, lecture\_room\_number, start\_time, end\_time)

FROM './ TAUGHT\_BY.csv' DELIMITER ',' CSV HEADER encoding ‘windows-1251’;

\copy TUTOR\_STUDENT (student\_university\_id,employee\_id)

FROM './TUTOR\_STUDENT.csv' DELIMITER ',' CSV HEADER encoding ‘windows-1251’;

\copy STUDENT\_PARTICIPATION (student\_university\_id, activity\_name, activity\_type, activity\_date , start\_time, venue, duration)

FROM './STUDENT\_PARTICIPATION.csv' DELIMITER ',' CSV HEADER encoding ‘windows-1251’;

**QUERIES:**

**Query 1**: : Given a student, show all the necessary information to describe how far along is that student in their program of study

SELECT STUDENT.student\_university\_id AS “StudentID”

,first\_name AS “FirstName”

,last\_name AS “LastName”

,program\_type AS “ProgramType”

,concentration AS “Concentration”

,OPTS\_FOR.start\_date AS “ProgramStartDate”

,sum(COURSE.credit\_hours) AS “CreditHoursApproved”

,count(DISTINCT (

ENROLLED\_ON.semester

,ENROLLED\_ON.year

)) AS “SemestersRegistered”

,overall\_grade AS “OverallGrade”

,OPTS\_FOR.STATUS AS “ProgramStatus”

FROM STUDENT

INNER JOIN OPTS\_FOR ON STUDENT.student\_university\_id = OPTS\_FOR.student\_university\_id

INNER JOIN ENROLLED\_ON ON STUDENT.student\_university\_id = ENROLLED\_ON.student\_university\_id

INNER JOIN COURSE ON COURSE.course\_code = ENROLLED\_ON.course\_code

WHERE course\_status = ”completed”

AND STUDENT.student\_university\_id = 101960001

GROUP BY STUDENT.student\_university\_id

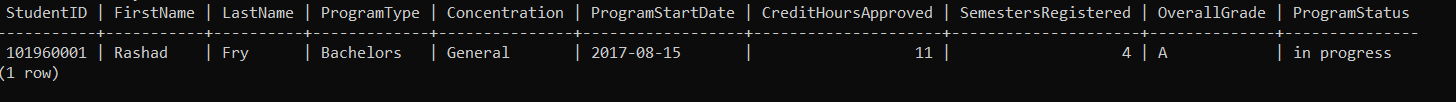
,program\_type

,concentration

,OPTS\_FOR.start\_date

,overall\_grade

,OPTS\_FOR.STATUS



**Query 2**:

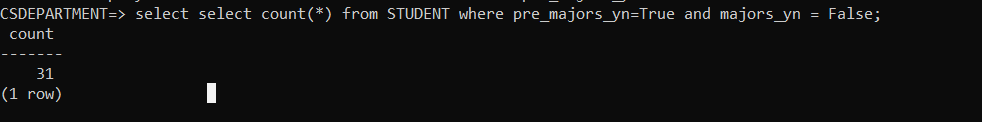
1. Total number of students who are premajors.

**SELECT count(\*)**

**FROM STUDENT**

**WHERE pre\_majors\_yn = True**

**AND majors\_yn = False**

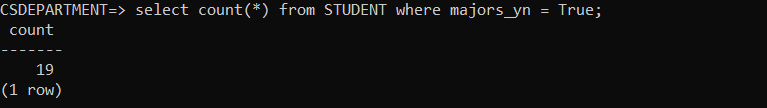


1. Total number of students who are majors (or enrolled in the MSc or the PhD program)

**SELECT count(\*)**

**FROM STUDENT**

**WHERE majors\_yn = True**

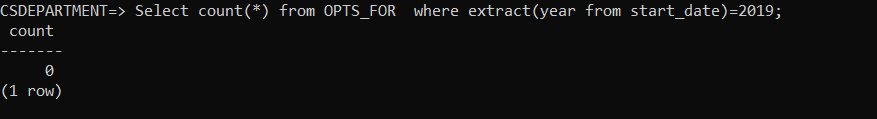


1. Total number of students who entered the program this year.

**SELECT count(\*)**

**FROM OPTS\_FOR**

**WHERE extract(year FROM start\_date) = 2019**



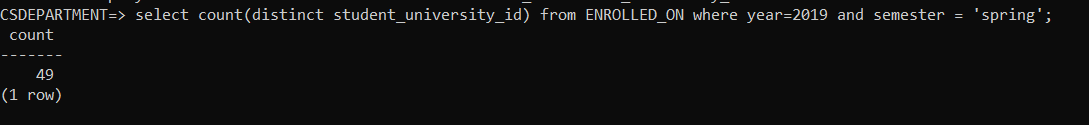
1. Total number of students who are active, i.e. enrolled in course for the current semester

**SELECT count(DISTINCT student\_univesity\_id)**

**FROM ENROLLED\_ON**

**WHERE year = 2019**

**AND semester = ‘spring’**



1. Total number of students in the program of study

**SELECT program\_type AS ProgramType**

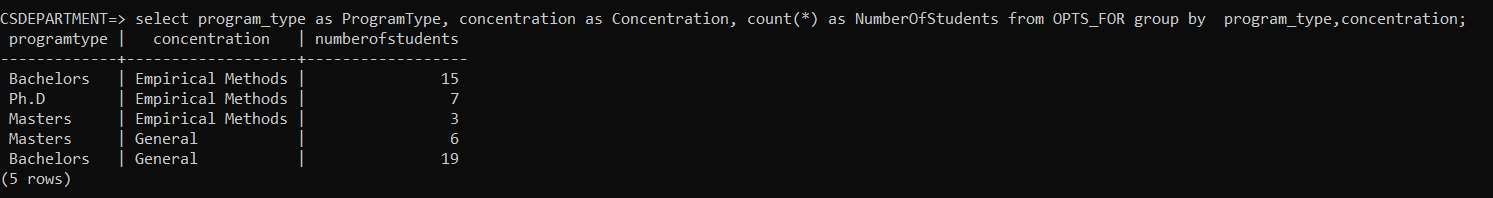
**,concentration AS Concentration**

**,count(\*) AS NumberOfStudents**

**FROM OPTS\_FOR**

**GROUP BY program\_type**

**,concentration;**

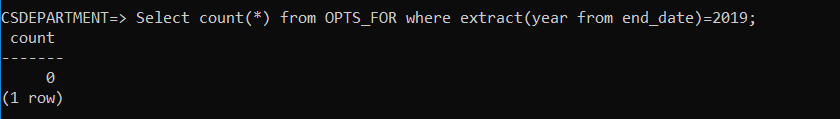


1. How many students graduated this year

**SELECT count(\*)**

**FROM OPTS\_FOR**

**WHERE extract(year FROM end\_date) = 2019**



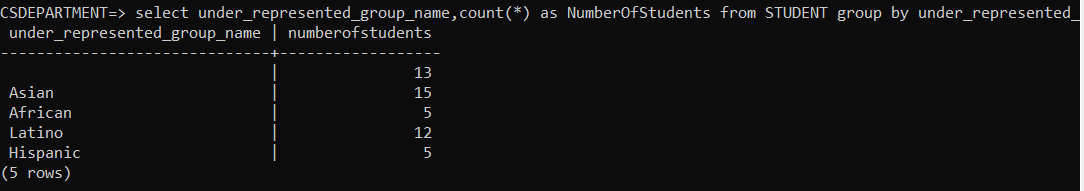
1. How many students in each category belong to under-represented groups

**SELECT under\_represented\_group\_name**

**,count(\*) AS NumberOfStudents**

**FROM STUDENT**

**GROUP BY under\_represented\_group\_name;**



**Query 3**: List of students who participated in a specific activity of recruitment/retention. Provide the total number of students by gender.

**SELECT R1.student\_university\_id AS "StudentID"**

**,first\_name AS "FirstName"**

**,last\_name AS "LastName"**

**,activity\_name AS "ActivityName"**

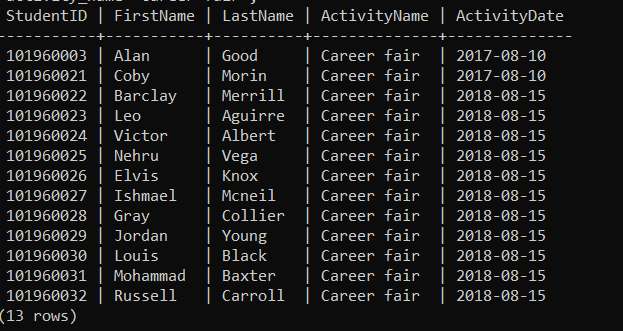
**,activity\_date AS "ActivityDate"**

**FROM STUDENT\_PARTICIPATION R1**

**INNER JOIN STUDENT R2 ON R1.student\_university\_id = R2.student\_university\_id**

**WHERE R1.activity\_type = 'recruitment'**

**AND activity\_name = 'Career fair';**



**SELECT activity\_name AS "ActivityName"**

**,activity\_date AS "ActivityDate"**

**,R2.gender AS "Gender"**

**,count(\*) AS "Count"**

**FROM STUDENT\_PARTICIPATION R1**

**INNER JOIN STUDENT R2 ON R1.student\_university\_id = R2.student\_university\_id**

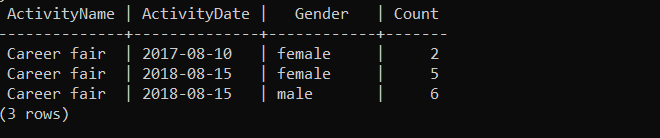
**WHERE R1.activity\_type = 'recruitment'**

**AND activity\_name = 'Career fair'**

**GROUP BY activity\_name**

**,activity\_date**

**,gender**



**Query 4**: List the names of the tutors, the student mentors, and the faculty mentors, provide the total of persons in each category. For tutors, list the courses they are able to tutor for. For mentors, indicate how many students they are mentoring and the list of students mentors.

**SELECT CONCAT (**

**first\_name**

**,last\_name**

**) AS "Name"**

**,'Tutor' AS "Type"**

**FROM TUTORING**

**INNER JOIN EMPLOYEE ON TUTORING.employee\_id = EMPLOYEE.employee\_id**

**UNION**

**SELECT CONCAT (**

**first\_name**

**,last\_name**

**) AS "Name"**

**,'Student Mentor' AS "Type"**

**FROM STUDENT\_PEER\_MENTOR**

**INNER JOIN STUDENT ON STUDENT\_PEER\_MENTOR.mentor\_id = STUDENT.student\_university\_id**

**UNION**

**SELECT CONCAT (**

**first\_name**

**,last\_name**

**) AS "Name"**

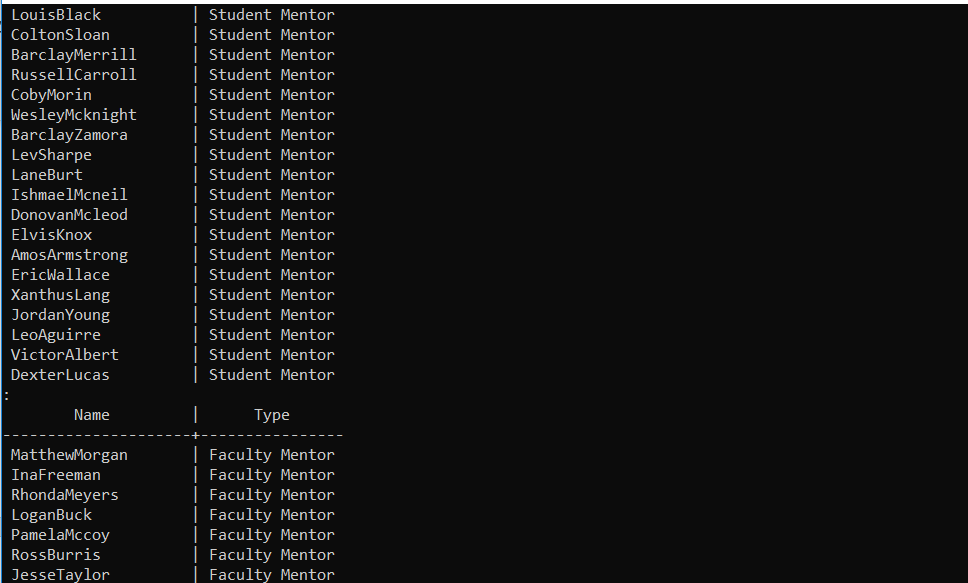
**,'Faculty Mentor' AS "Type"**

**FROM EMPLOYEE**

**INNER JOIN ADVISEMENT ON EMPLOYEE.employee\_id = ADVISEMENT.employee\_id**

**AND advisement\_type = 'mentoring'**

**ORDER BY "Type";**



**SELECT CONCAT (**

**first\_name**

**,last\_name**

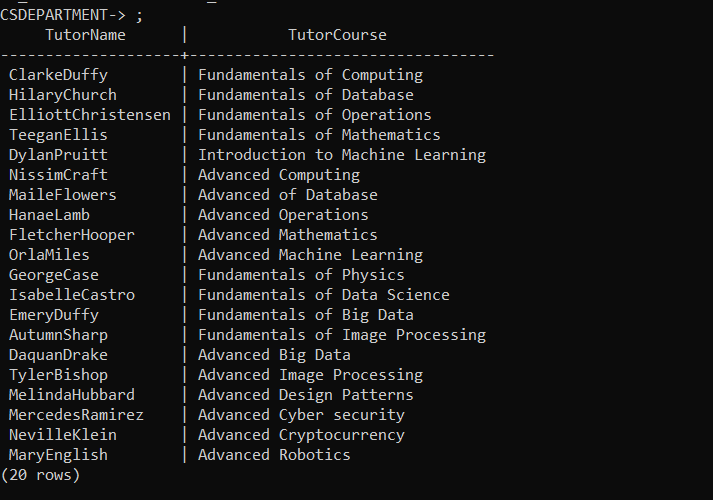
**) AS “TutorName”**

**,course\_name AS “TutorCourse”**

**FROM Tutoring**

**INNER JOIN EMPLOYEE ON TUTORING.employee\_id = EMPLOYEE.employee\_id**

**INNER JOIN COURSE ON Tutoring.course\_code = COURSE.course\_code**



**SELECT CONCAT (**

**first\_name**

**,last\_name**

**) AS "FacultyMentorName"**

**,count(\*) AS "StudentCount"**

**FROM ADVISEMENT**

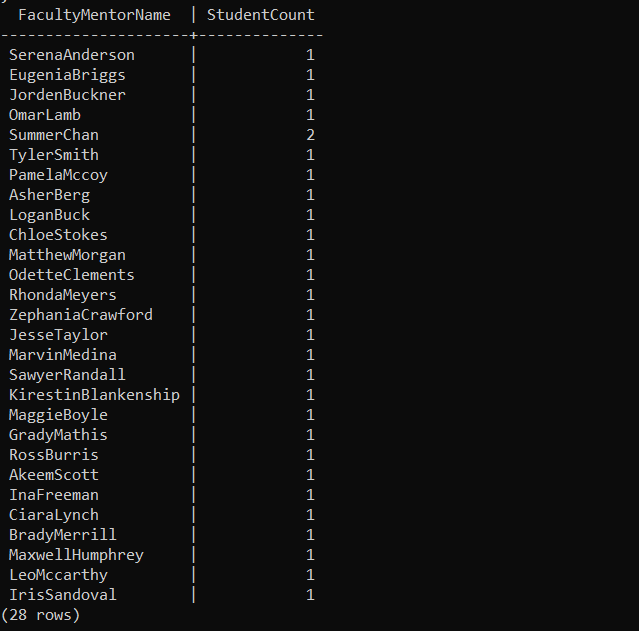
**INNER JOIN EMPLOYEE ON ADVISEMENT.employee\_id = EMPLOYEE.employee\_id**

**WHERE advisement\_type = '**

**mentoring'**

**GROUP BY first\_name**

**,last\_name;**



**SELECT CONCAT (**

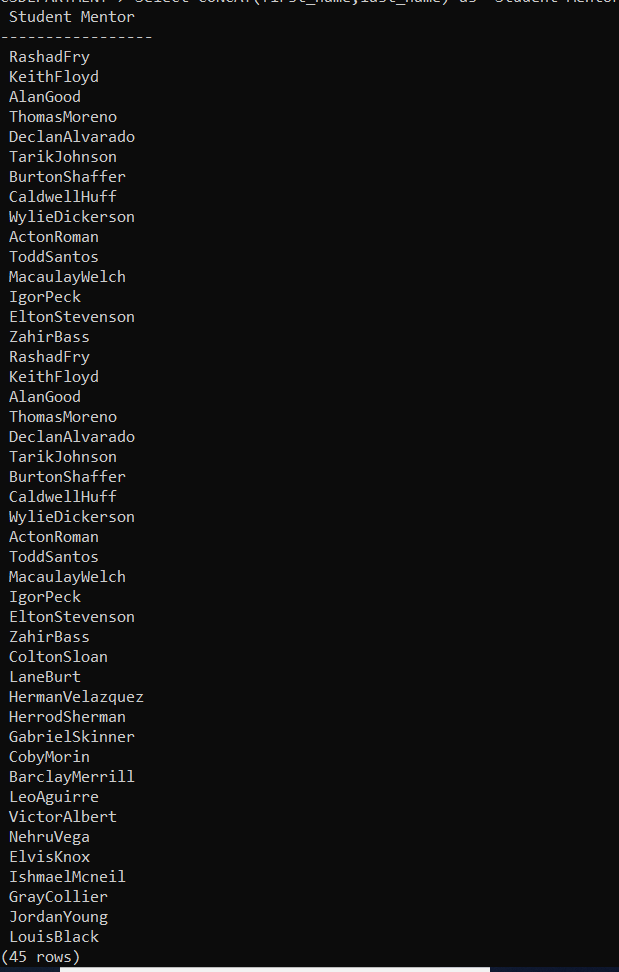
**first\_name**

**,last\_name**

**) AS “Student Mentor”**

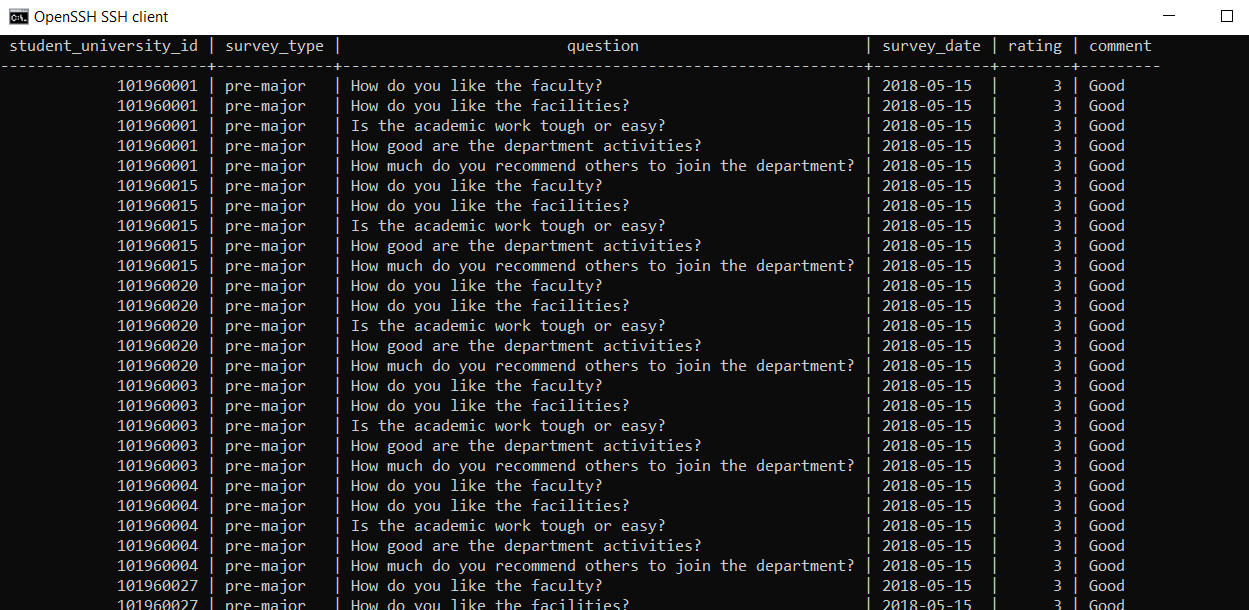
**FROM STUDENT\_PEER\_MENTOR NATURAL**

**INNER JOIN STUDENT**



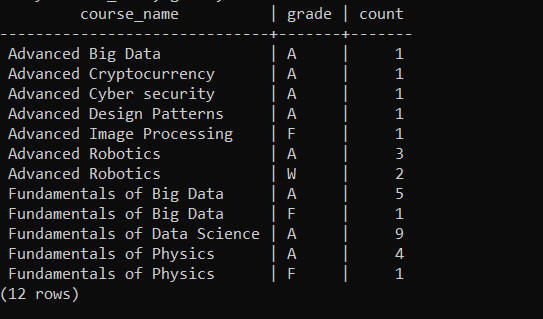
**Query 5**:

Select \* from GIVES where extract(year from survey\_date)=2018



Query 6:

Select course\_name, grade, count(grade) from ENROLLED\_ON INNER JOIN COURSE ON ENROLLED\_ON.course\_code = COURSE.course\_code where year = 2018 and semester=’spring’ group by course\_name, grade order by course\_name, grade



**Query 7** :

Given a faculty member, list the names of the students they advise and any details about the advisement given.

**SELECT DISTINCT CONCAT (**

**first\_name**

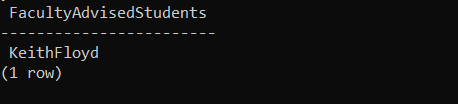
**,last\_name**

**) AS "FacultyAdvisedStudents"**

**FROM ADVISEMENT**

**INNER JOIN STUDENT ON ADVISEMENT.student\_university\_id = STUDENT.student\_university\_id**

**WHERE employee\_id = 1012345868;**



**Query 8** : List the faculty members who do not advise nor mentor any student.

**SELECT EMPLOYEE.\***

**FROM FACULTY**

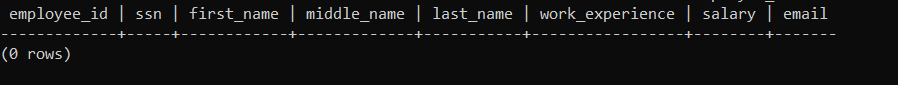
**INNER JOIN EMPLOYEE ON FACULTY.employee\_id = EMPLOYEE.employee\_id**

**WHERE EMPLOYEE.employee\_id NOT IN (**

**SELECT employee\_id**

**FROM ADVISEMENT**

**);**



**Query 9**: List the tutors with their names and courses they are able to tutor for

**SELECT CONCAT (**

**first\_name**

**,last\_name**

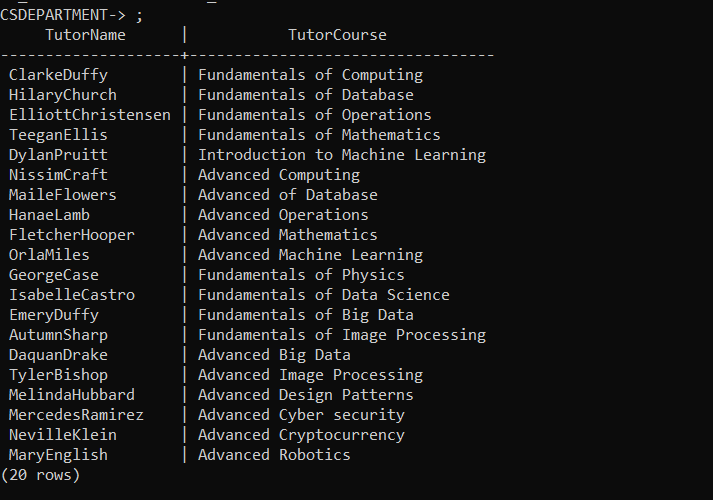
**) AS “TutorName”**

**,course\_name AS “TutorCourse”**

**FROM Tutoring**

**INNER JOIN EMPLOYEE ON TUTORING.employee\_id = EMPLOYEE.employee\_id**

**INNER JOIN COURSE ON Tutoring.course\_code = COURSE.course\_code**



**Query 10**: : List the information about the tutors who belong to under-represented groups.

**SELECT CONCAT (**

**first\_name**

**,last\_name**

**) AS "Tutor"**

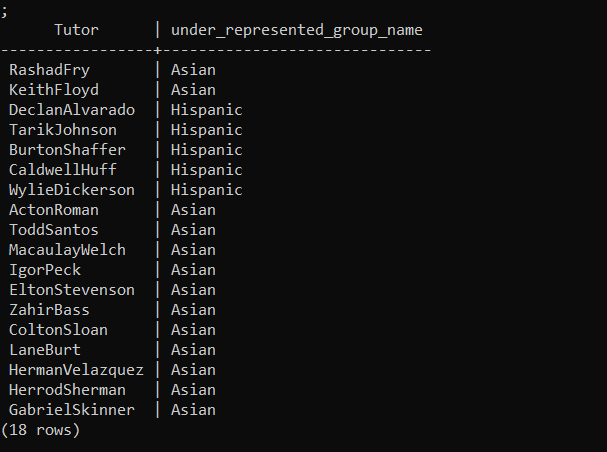
**,under\_represented\_group\_name**

**FROM TUTOR\_STUDENT**

**INNER JOIN STUDENT ON STUDENT.student\_university\_id = TUTOR\_STUDENT.student\_university\_id**

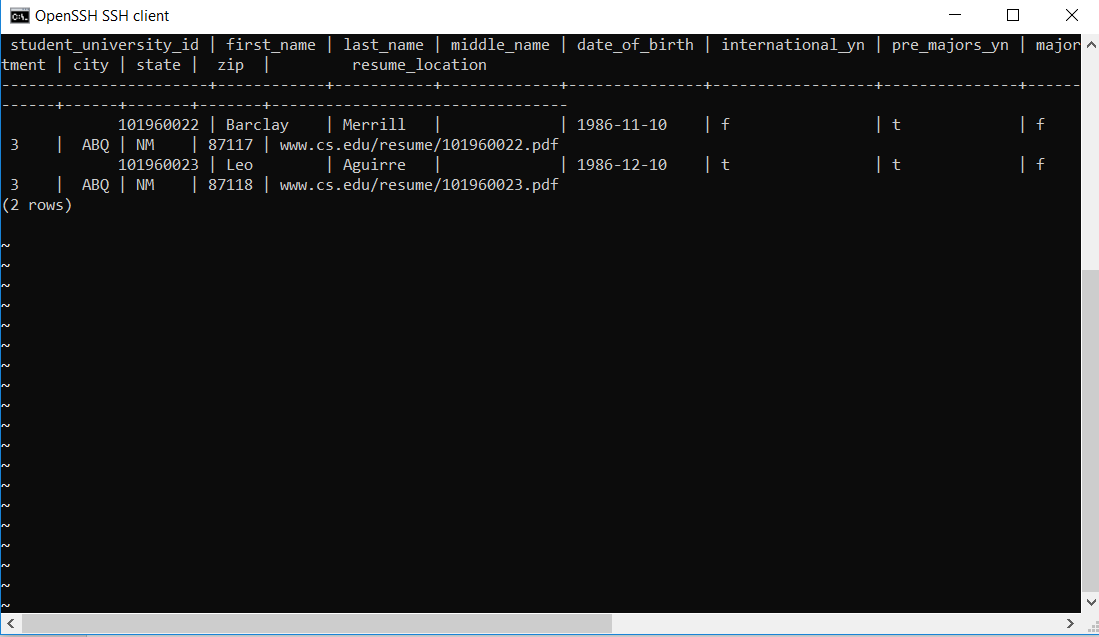
**WHERE und er\_represented\_group\_name <> NULL**

**OR under\_represented\_group\_name <> '';**



**Query 11**.

Select \* from STUDENT where student\_university\_id in (SELECT e1.student\_university\_id from ENROLLED\_ON e1 INNER JOIN ENROLLED\_ON e2 ON e1. student\_university\_id = e2. student\_university\_id where e1.semester = e2.semester and e1.year= e2.year and (e1.course\_code,e2.course\_code) in (select course\_code,pre\_requisite\_course\_code from PRE\_REQUISITE))



**CREATE TABLE “COURSE” (**

**course\_code VARCHAR(10) PRIMARY KEY**

**,course\_name VARCHAR(100) NOT NULL**

**,credit\_hours NUMERIC(2) DEFAULT 0 CHECK (course\_fee >= 0)**

**,department VARCHAR(5)**

**,course\_fee NUMERIC(10, 5) DEFAULT 0 CHECK (course\_fee >= 0)**

**,online\_yn boolean**

**,required\_for\_major\_transition\_yn boolean NOT NULL**

**,**

**);**

**CREATE TABLE “PROGRAM” (**

**program\_type VARCHAR(50)**

**,concentration VARCHAR(100)**

**,description VARCHAR(200)**

**,total\_credits\_needed INT DEFAULT 0 CHECK (total\_credits\_needed >= 0)**

**,total\_core\_credits\_needed INT DEFAULT 0 CHECK (total\_core\_credits\_needed >= 0)**

**,PRIMARY KEY (**

**program\_type**

**,concentration**

**)**

**,CONSTRAINT credits\_check CHECK (total\_credits\_needed >= total\_core\_credits\_needed)**

**);**

**CREATE domain GRADE VARCHAR(15) NOT NULL CHECK (**

**VALUE IN (**

**‘A’**

**,’B’**

**,’C’**

**,’D’**

**,’F’**

**,’W’**

**)**

**);**

**CREATE TABLE “PRE\_REQUISITE” (**

**course\_code VARCHAR(10)**

**,pre\_requisite\_course\_code VARCHAR(10) REFERENCES COURSE(course\_code)**

**,minimum\_grade\_required grade**

**,PRIMARY KEY (**

**course\_code**

**,pre\_requisite\_course\_code**

**)**

**);**

**CREATE TABLE “PROGRAM\_REQUIREMENT” (**

**program\_type VARCHAR(50)**

**,concentration VARCHAR(100)**

**,course\_code VARCHAR(10)**

**,minimum\_grade\_required grade**

**,PRIMARY KEY (**

**program\_type**

**,concentration**

**,course\_code**

**)**

**,FOREIGN KEY (**

**program\_type**

**,concentration**

**) REFERENCES PROGRAM(program\_type, concentration)**

**);**

**CREATE domain SEMESTER VARCHAR(15) NOT NULL CHECK (**

**VALUE IN (**

**‘fall’**

**,’spring’**

**,’’summer”**

**)**

**);**

**CREATE domain COURSE\_STATUS VARCHAR(20) NOT NULL CHECK (**

**VALUE IN (**

**‘registered’**

**,’completed’**

**,’withdrawn’**

**,’failed’**

**)**

**);**

**CREATE TABLE “ENROLLED\_ON” (**

**student\_university\_id NUMERIC(10) REFERENCES STUDENT(student\_university\_id)**

**,course\_code VARCHAR(10) REFERENCES COURSE(course\_code)**

**,semester semester**

**,year NUMERIC(4) CHECK (year > 0)**

**,start\_date DATE DEFAULT CURRENT\_DATE**

**,end\_date DATE**

**,course\_status course\_status**

**,grade grade**

**,PRIMARY KEY (**

**student\_university\_id**

**,course\_code**

**,semester**

**,year**

**)**

**,CONSTRAINT course\_enrolled\_dates CHECK (start\_date < end\_date)**

**);**

**CREATE domain SURVEY VARCHAR(20) NOT NULL CHECK (**

**VALUE IN (**

**‘pre - major’**

**,’major’**

**,’exit’**

**)**

**);**

**CREATE TABLE “SURVEY” (**

**survey\_type survey**

**,question VARCHAR(400)**

**,PRIMARY KEY (**

**survey\_type**

**,question**

**)**

**);**

**CREATE TABLE “STUDENT\_PEER\_MENTOR” (**

**student\_university\_id NUMERIC(10) REFERENCES STUDENT(student\_university\_id)**

**,mentor\_id NUMERIC(10) REFERENCES STUDENT(student\_university\_id)**

**,semester semester**

**,year NUMERIC(4) CHECK (year > 0) PRIMARY KEY (**

**student\_university\_id**

**,semester**

**,year**

**)**

**);**

**CREATE TABLE “GIVES” (**

**student\_university\_id NUMERIC(10) REFERENCES STUDENT(student\_university\_id)**

**,survey\_type survey\_type**

**,question VARCHAR(400)**

**,survey\_date DATE DEFAULT CURRENT\_DATE**

**,rating INT CHECK (0 <= rating <= 5)**

**,comment VARCHAR(200)**

**,FOREIGN KEY (**

**survey\_type**

**,question**

**) REFERENCES SURVEY(survey\_type, question)**

**,PRIMARY KEY (**

**student\_university\_id**

**,survey\_type**

**,question**

**,survey\_date**

**)**

**);**

**CREATE TABLE “INCIDENT” (**

**incident\_category VARCHAR(20)**

**,description VARCHAR(200)**

**,PRIMARY KEY (incident\_category)**

**);**

**CREATE TABLE “FOLLOW\_UP” (**

**student\_university\_id NUMERIC(10) REFERENCES STUDENT(student\_university\_id)**

**,incident\_category VARCHAR(20) REFERENCES INCIDENT(incident\_category)**

**,incident\_date DATE CHECK (incident\_date <= CURRENT\_DATE)**

**,incident\_time TIME**

**,necessary\_action VARCHAR(200)**

**,PRIMARY KEY (**

**student\_university\_id**

**,incident\_category**

**,incident\_category**

**,incident\_time**

**)**

**);**

**CREATE TABLE “OPTS\_FOR” (**

**student\_university\_id NUMERIC(10) REFERENCES STUDENT(student\_university\_id)**

**,program\_type VARCHAR(50)**

**,concentration VARCHAR(100)**

**,opted\_semester semester**

**,opted\_year NUMERIC(4) CHECK (opted\_year > 0)**

**,start\_date DATE DEFAULT CURRENT\_DATE**

**,end\_date DATE**

**,overall\_grade grade**

**,STATUS VARCHAR(20)**

**,PRIMARY KEY (**

**student\_university\_id**

**,program\_type**

**,concentration**

**,opted\_semester**

**,opted\_year**

**)**

**,FOREIGN KEY (**

**program\_type**

**,concentration**

**) REFERENCES PROGRAM(program\_type, concentration)**

**,CONSTRAINT program\_dates\_check CHECK (start\_date < end\_date)**

**);**