Consider the following tables:

"`courseno`" includes a string, like 'DATA324'. The attributes `Student.major`, `Track.major`, and `TrackRequirements.major` contain just the acronym for the major, e.g., 'CptS','EE', etc.

```
CREATE TABLE Course (
      courseno    VARCHAR(7),
      credits     INTEGER NOT NULL,
      enroll_limit INTEGER,
      classroom   VARCHAR(10),
      PRIMARY KEY(courseNo),
);

CREATE TABLE Tracks (
      major       VARCHAR(7),
      trackcode   VARCHAR(10),
      title       VARCHAR(30),
      PRIMARY KEY(major, trackcode)
);

CREATE TABLE Student (
      sID         CHAR(8),
      sName       VARCHAR(30),
      major       VARCHAR(10),
      trackcode   VARCHAR(10),
      PRIMARY KEY(sID),
      FOREIGN KEY (major,trackcode) REFERENCES Tracks(major,trackcode)
);

CREATE TABLE Enroll (
      courseno    VARCHAR(7),
      sID         CHAR(8),
      grade       FLOAT NOT NULL,
      PRIMARY KEY (courseNo, sID),
      FOREIGN KEY (courseNo) REFERENCES Course(courseNo),
      FOREIGN KEY (sID) REFERENCES Student(sID)
);

CREATE TABLE Prereq (
      courseNo     VARCHAR(7),
      preCourseNo  VARCHAR(7),
      PRIMARY KEY (courseNo, preCourseNo),
      FOREIGN KEY (courseNo) REFERENCES Course(courseNo),
      FOREIGN KEY (preCourseNo) REFERENCES Course(courseNo)
);

CREATE TABLE TrackRequirements (
      major       VARCHAR(7),
      trackcode   VARCHAR(10),
      courseNo    VARCHAR(7),
      PRIMARY KEY (major,trackcode,courseNo),
      FOREIGN KEY (major,trackcode) REFERENCES Tracks(major,trackcode),
      FOREIGN KEY (courseNo) REFERENCES Course(courseNo)
);
```

Please do the following before you answer the homework questions:

1. Install PostgreSQL Server (if you haven't done so).
2. Create a database named `HW4`. (If you are working on PgAdmin, initially open a query window for the "postgres" database; create the database HW4, and then open a new query window for the new HW4 database)
3. Unzip the archive file `HW4-DB.zip`.
4. Create and populate your tables by running the given `.sql` script files (in HW4-DB.zip). You are given 6 script files: `course.sql; tracks.sql; student.sql; enroll.sql; prereq.sql; trackReq.sql`. Each file contains the CREATE TABLE statement to create the corresponding tables the `INSERT` statements to insert the sample data.
   - See the Appendix for instructions on how to execute script files on command line.
   - You may alternatively copy and paste the `CREATE TABLE` and `INSERT` statements to the PgAdmin query tool and run them.
   - Follow the order `course, tracks, student, enroll, prereq, trackrequirements` when running insert statements. Otherwise there would be violations for the foreign key constraints.
5. Check if the data is inserted correctly by running a "`SELECT * FROM <relationname>`" on each table. The number of tuples in each table should be same as number of tuples in each INSERT statement in the corresponding file.

Write the following queries in SQL:

1. Find the distinct courses that 'SYS' track students in 'CptS' major are enrolled in. Return the courseno and credits for those courses. Return results sorted based on courseno.

courseno	credits
CptS121	4
CptS122	4
CptS223	3
CptS260	3
CptS317	3
CptS322	3
~~CptS323~~	~~3~~
CptS355	3
CptS360	3
CptS421	3
~~CptS422~~	~~3~~
CptS423	3
CptS451	3
CptS460	3
MATH171	4
MATH172	4
MATH216	3
MATH220	3

2. Find the sorted names, ids, majors and track codes of the students who are enrolled in more than 18 credits (19 or above).

sname	sid	major	trackcode	sum
Aaron	12584489	ME	NULL	63
Ali	12582389	CptS	SE	30
Alice	12583589	CptS	SYS	52
Ally	12579189	CHE	NULL	24
Amir	12582989	CHE	NULL	21
Bill	12581189	CptS	SE	27
Jack	12584789	CptS	SE	52
Nick	12582689	CHE	NULL	36
Sam	12567189	ME	NULL	20

3. Find the courses that only 'SE' track students in 'CptS major have been enrolled in. Give an answer without using the set EXCEPT operator. (Final corrected version)

courseno
CptS323
CptS422
CptS484
CptS487

4. Find the students who have enrolled in the courses that Diane enrolled and earned the same grade Diane earned in those courses. Return the student name, sID, major as well as the courseno and grade for those courses.

sname	sid	major	courseno	grade
Aaron	12584489	ME	EE499	2.75
Nancy	12566189	ME	MATH115	2.75

5. Find the students in 'CptS' major who are not enrolled in any classes. Return their names and sIDs. (Note: Give a solution using OUTER JOIN)

sname	sid
Ally	12514189
Li	12576189
Mick	12565189
Sam	12254189
Tom	12354189

6. Find the courses given in the 'Sloan' building which have enrolled more students than their enrollment limit. Return the courseno, enroll_limit, and the actual enrollment for those courses.

courseno	enroll_limit	enrollnum
CptS260	3	4
CHE211	4	7
CHE321	4	5
CHE110	4	6

7. Find 'CptS' major students who enrolled in a course for which there exists a prerequisite that the student got a grade lower than "2". (For example, Alice (sid: 12583589) was enrolled in CptS355 but had a grade 1.75 in prerequisite CptS223.) Return the names and sIDs of those students and the courseno of the course (i.e., the course whose prereq had a low grade).

sname	sid	courseno
Alice	12583589	CptS317
Alice	12583589	CptS322
Alice	12583589	CptS355
Alice	12583589	CptS360
Alice	12583589	CptS451

8. For each 'CptS' course, find the percentage of the students who failed the course. Assume a passing grade is 2 or above. (Note: Assume students who didn't earn a grade in class should be excluded in average calculation. Also assume all CptS courses start with the 'CptS' prefix).

courseno	passrate
CptS451	100
CptS323	100
CptS322	100
CptS317	100
CptS360	100
CptS484	100
CptS121	100
CptS355	75
CptS460	100
CptS421	100
CptS422	100
CptS423	66
CptS260	75
CptS487	100
CptS223	66
CptS122	100

9. Consider the following relational algebra expression.
(i) explain what the expression is doing,
(ii) write an equivalent SQL query.

$\sigma_{pcount>=2}\ (\gamma_{courseno,count(preCourseNo)->pCount}\ (Course \bowtie Prereq))$

courseno	pcount
CHE398	2
CptS317	2
CptS355	2
CptS421	2
CptS422	2
CptS423	2
EE262	2
EE331	2
EE351	2
EE361	2
EE415	2
EE416	2

Submission Instructions:
In this homework, you are only allowed to use the standard SQL statements that we covered in class.
HW4 will be submitted online via the course learning management system. Create a SQL script file named D324_HW4_<yourname>.sql which includes your queries for questions 1 through 9. (For example: *D324_HW4_joecoug.sql*) Include a comment at the top of the file and write your name and WSU ID.
Note that you may lose points if you don't comply with the above submission instructions.

Appendix A – How to run a SQL script file in PostgreSQL

Create a database hw4 in PostgreSQL.

On Windows: run `cmd` to open command line window; on Linux and Mac, open a terminal.

Assuming the script file you will run is `myscript.sql`, run the following on the command line. Make sure you are at the directory where the script file is located. :

```
psql -U postgres -d hw4 < myscript.sql
```

(if `psql` is not recognized, you need to add the PostgreSQL installation path to the PATH environment variable. Alternatively, you may browse to the installation directory of PostgreSQL and then run the above command).
You have to supply a database name to connect to. The above statement assumes your database name is "hw4".
If you would be running postgreSQL with another username (other than `postgres`), replace `postgres` with that username. You will be asked to enter your password for the username you specify.

(The "yelpdb=#" here is the command prompt. Yours will look different depending on your database name.)

The above command will execute all the queries in the `myscript.sql` file.

Check http://www.postgresqlforbeginners.com/2010/11/interacting-with-postgresql-psql.html for a brief tutorial about interacting with PostgreSQL in the command line.