Final Project

CISC 660 Database Management Systems

Project Option II
University Database

Under the guidance of
Dr. Junping Sun

Chintan Shashiantbhai Jariwala
NSU ID: N01841692

Date: 16th April, 2017

1. PARTITION THE SENTENCES

[1]. University (Common Sentence)
- In a university, we represent data about both students and employees.

[2]. Student
- The university keeps track of each student's name, student number, social security number, address, phone, birth date, sex, class (freshman, graduate), major department, minor department (if any), and degree program (B.A., B.S., M.A., M.S., ..., Ph.D.). Some user applications need to access the city, state, and zip code of the student's address and the student last name. Both social security number and student number have unique values for each student.
- Students may have a transcript for all the courses they have taken. For graduate students, the student's advisor should be included in the database.

[3]. Department
- Each department is described by a name, department number, office number, office phone, and college. Both department name and department number have unique values for each department. Each department has a Chairperson or a Dean in charge of that department.

[4]. Course
- Some courses have prerequisites (please pay attention here). Each course has the day, meeting time, place where the class is held.
- Each section has an instructor, semester, year, course, and section number. The section number distinguishes different sections of the same course that is taught during the same semester/year (may be at the same time), its values are 1, 2, 3, ..., up to the number of sections taught during each semester.
- A grade report for a course has student names, section number, and grades.

[5]. Employees
- Employees are classified into faculty and staff, both of them have dependents, the database stores the information of employees' dependents for the insurance and benefit purposes.
- Faculty could be full-time or part-time employees. Professors have ranks (Lecturer, Assistant Professor, Associate Professor, Full Professor) and salaries. Faculties (Professors) may hold different degree (highest degree is only considered here). Each professor belongs to at least one department. Professors may have joint appointments from other department(s).
- Staff are secretaries, program coordinators, assistant directors, directors, deans, vice presidents, and president

2. ER Diagram and CSDL
[1]. ER Diagram
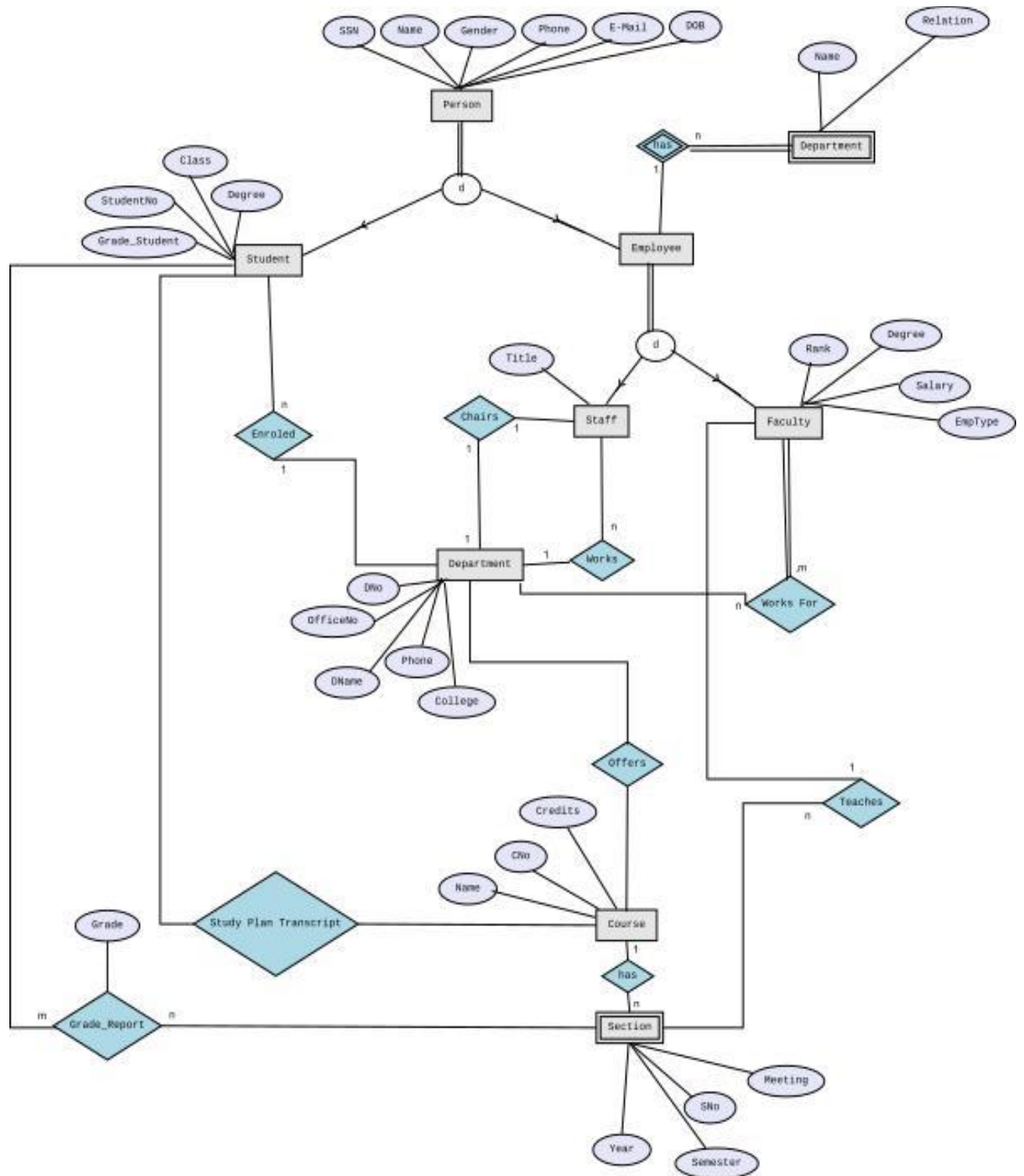
[2]. Conceptual Schema in CSDL

1) Entity Person

Attribute Name	Type	Description
SSN	VARCHAR	Unique identifier for each user
Gender	Char	Gender
Name	VARCHAR	Composite Attribute – FirstName, MiddleName, LastName
DataOfBirth	Date	Date of Birth
Phone	VARCHAR	Phone Number
Email	VARCHAR	Email
Address	VARCHAR	Composite Attribute – Street, City, State, Zip

```
/* Defination of entity PERSON */
Entity:         PERSON
Attributes:         SSN            VARCHAR(10)
                    GENDER         CHAR
                    NAME           VARCHAR2(15)
                    DOB            DATE
                    NAME           VARCHAR2(20)
                    ADDRESS        VARCHAR2(20)

Identifiers:        SSN
```

2) Entity Student

Attribute Name	Type	Description
StudentNo	VARCHAR	Unique identifier for Student
Class	VARCAHR	Student's class
DegreeProg	VARCHAR	Degree Program in which student is enrolled.

```
/* Defination of entity STUDENT */
Entity:         STUDENT
Attributes:         STUDENTNO   VARCHAR(10)
                    CLASS       VARCHAR(10)
                    DEGREEPRO   VARCHAR2(15)

Identifiers:        STUDENTNO
```

3) Entity extending from Student – Grad_Student
4) Entity extending from Employee – Faculty and Staff

5) Entity Staff

Attribute Name	Type	Description
Title	**VARCHAR**	**Staff's title**

/* Defination of entity STAFF */
Entity: STAFF
Attributes: TITLE   VARCHAR(10)

Identifiers: TITLE

6) Entity Faculty

Attribute Name	Type	Description
Rank	VARCHAR	Faculty's Rank
Degree	VARCHAR	Faculty's Degree
Salary	Number	Faculty's Salary
EmpType	VARCHAR	Full Time/Part Time

/* Defination of entity FACULTY */
Entity: FACULTY
Attributes: RAB   VARCHAR(10)
                       DEGREE   VARCHAR(10)
                       SALARY   VARCHAR2(15)
                       EMPTYPE   VARCHAR(10)

7) Entity Dependent

Attribute Name	Type	Description
Name	VARCHAR	Dependent's Name
Relation	VARCHAR	Dependent's Relation with Employee

/* Defination of entity DEPENDENT */
Entity: DEPENDENT
Attributes: NAME   VARCHAR(10)
                       RELATION   VARCHAR(10)

8) Entity Department

Attribute Name	Type	Description
DName	VARCHAR	Department Name, Unique Identifier
DNo	VARCHAR	Department Number, Unique Identifier

College	VARCHAR	College in which department is present
Phone	VARCHAR	Department Phone No
OfficeNo	VARCHAR	Office No.

/* Defination of entity DEPARTMENT */
Entity: DEPARTMENT
Attributes: DNAME   VARCHAR(10)
                    DNO   VARCHAR(10)
                    COLLEGE   VARCHAR2(15)
                    PHONE   VARCHAR(10)

Identifiers: DNAME

9) Entity Course

Attribute Name	Type	Description
CNo	VARCHAR	Course Number, Unique Identifier
Name	VARCHAR	Course Name
Credits	VARCHAR	Course Credits

/* Defination of entity COURSE */
Entity: COURSE
Attributes: CNO   VARCHAR(10)
                    NAME   VARCHAR(10)
                    CREDITS   VARCHAR2(15)

Identifiers: CNO

10) Entity Semester

Attribute Name	Type	Description
Sno	VARCHAR	Serial Number, Part of Primary Key
Year	VARCHAR	Semester year, Part of Primary Key
Semester	VARCHAR	Semester No, Part of Primary Key
Meeting	VARCHAR	Composite Attribute - Time, Day and Room

/* Defination of entity SEMESTER */
Entity: SEMESTER
Attributes: SNO   VARCHAR(10)

```
YEAR       VARCHAR(10)
SEMESTER   VARCHAR2(15)
MEETING    VARCHAR(10)
```

Identifiers: SNO

Following generalization exists:
- Person is super type of Employee and Student
- Employee is super type of Faculty and Staff
- Student is super type of Grad_Student

Following Relationship exists:
- Employee has dependents. There exists one to many relationship between Employee and Dependent.
- Staff Chairs Department. There exists one to one relationship between Staff and Department as only 1 staff member can chair the department at one time.
- Staff works for department. There exists one to many relationship between Department and Staff.
- Faculty works for department. There exists many to many relationship between Faculty and Department.
- Faculty teaches Section. There exists one to many relationship between Faculty and Section
- Student is enrolled in Department. There exists one to many relationship between Department and Student.
- Many –to-many relationship exist between Course and Student for Study Plan and Transcript
- Many-to-many relationship exists between Student and Section.
- A Department can offer one or many courses at a time. So there exists one to many relationship between Department and Course.

3. Relational Database Schema

1. **Person (SSN** VARCHAR, **FIRSTNAME** VARCHAR, **MIDDLENAME** VARCHAR, **LASTNAME** VARCHAR, **GENDER** CHAR, **PHONE** VARCHAR, **EMAIL** VARCHAR, **STREET** VARCHAR, **CITY** VARCHAR, **STATE** VARCHAR, **ZIP** VARCHAR, **DATEOFBIRTH** DATE**)**

   - **Functional Dependencies**
   a) SSN → FIRSTNAME, MIDDLENAME, LASTNAME, GENDER, PHONE, EMAIL, DATEOFBIRTH
   b) ZIP → STREET, CITY, STATE
   - **Primary Key** → SSN

2. **Employee (SSN** VARCHAR)

   - **PRIMARY KEY** → SSN
   - **FOREIGN KEY** → SSN
   - Here SSN refer to SSN in Person table for maintaining referential integrity.

6

3. **DEPENDENT (name** VARCHAR**, EESSN** VARCHAR**, Relation)**

   - **Primary Key →** NAME**,** EESSN
   - **FOREIGN KEY →** EESSN
   - Here EESSN refers to SSN in Employee table, thus maintain referential integrity.

4. **STUDENT (<u>SSN</u>** VARCHAR**, STUDENTNO** VARCHAR**, CLASS** VARCHAR**, DEGREEPROG** VARCHAR**, MINORDEPT** VARCHAR**, MAJORDEPT** VARCHAR**)**
   - **Functional Dependencies**
   a) SSN → STUDENTNO, CLASS, DEGREEPROG, MINORDEPT, MAJORDEPT
   b) STUDENTNO → SSN, CLASS, DEGREEPROG, MINORDEPT, MAJORDEPT
   - **Primary Key →** SSN
   - **Foreign Key →** SSN**,** MINORDEPT**,** MAJORDEPT
   - Here SSN is the foreign key referring to SSN in Person table and MINORDEPT and MAJORDEPT are referring to DEPTNO in Department table.

5. **STAFF (<u>SSN</u>** VARCHAR**, title** VARCHAR**, DEPTNO)**
   - **Functional Dependency**
   a) SSN → title, DEPTNO
   - **Primary Key →** SSN
   - **Foreign Key →** SSN
   - Here SSN is the foreign Key referring to SSN in Person table and DEPTNO is the foreign key referring to DNO in Department table.

6. **FACULTY (<u>SSN</u>** VARCHAR**, RANK** VARCHAR**, DEGREE** VARCHAR**, SALARY** NUMBER (10,2)**, EMPTYPE** VARCHAR**)**
   - **Functional Dependency**
   a) SSN → RANK, DEGREE, SALARY, EMPTYPE
   - **Primary Key →** SSN
   - **Foreign Key →** SSN
   - Here SSN is the foreign Key referring to SSN in Person table.

7. **DEPARTMENT (<u>DNO</u> VARCHAR, DNAME VARCHAR, COLLEGE VARCHAR, OFFICENO** VARCHAR**, PHONE** VARCHAR**, CHAIRPERSON** VARCHAR**)**
   - **Functional Dependences**
   a) DNO → DNAME, COLLEGE, OFFICENO, PHONE, CHAIRPERSON
   b) DNAME → DNO, COLLEGE, OFFICENO, PHONE, CHAIRPERSON
   - **Candidate Keys →** DNO**,** DNAME
   - **Primary Key →** DNO
   - **Foreign Key →** CHAIRPERSON
   - Chairperson is the SSN referring to SSN in Person table.

8. **WORKS_FOR (<u>FAC_SSN</u>** VARCHAR**, <u>DEPTNO</u>** VARCHAR**)**
   - **Primary Key →** FAC_SSN, DEPTNO

- **Foreign Key** → FAC_SSN, DEPTNO
- Here FAC_SSN is the foreign Key referring to SSN in Faculty table and DEPTNO is the foreign key referring to DNO in Department table.

9. **COURSE (CNO** VARCHAR**, COURSENAME** VARCHAR**, CREDITS** NUMBER**, DEPTNO** VARCHAR**)**
   - **Functional Dependencies**
   a) CNO → COURSENAME, CREDITS, DEPTNO
   - **Primary Key** → CNO
   - **Foreign Key** → DEPTNO
   - Here DEPTNO is the foreign key referring to DNO in Department table.

10. **STUDYPLAN (STU_SSN** VARCHAR**, COURSENO** VARCHAR**)**
    - **Primary Key** → STU_SSN**,** COURSENO
    - **Foreign Key** → STU_SSN**,** COURSENO
    - STU_SSN is the foreign key referring to SSN in Student table and COURSENO is the foreign key referring to CNO in Course table.

11. **TRANSCRIPT (STU_SSN** VARCHAR**, COURSENO** VARCHAR**, GRADE** VARCHAR**)**
    - **Functional Dependency**
    a) STU_SSN, COURSENO → GRADE
    - **Primary Key** → STU_SSN, COURSENO
    - **Foreign Key** → STU_SSN, COURSENO

12. **SECTION (SNO** VARCHAR**, SEMESTER** VARCHAR**, YEAR** VARCHAR**, DAY** VARCHAR**, TIME** VARCHAR**, ROOM** VARCHAR**, FACULTY** VARCHAR**, COURSENO** VARCHAR**)**
    - **Functional Dependency**
    a) SNO, SEMESTER, YEAR, COURSENO →DAY, TIME, ROOM, FACULTY
    - **Primary Key** → SNO
    - **Foreign Key** → FACULTY**,** COURSENO
    - Here Faculty is the foreign key referring to SSN in Faculty table and CourseNo is the foreign key referring to CNO in Course table.

13. **GRADE_REPORT (STU_SSN** VARCHAR**, SNO** VARCHAR**, SEMESTER** VARCHAR**, YEAR** NUMBER**, COURSENO** VARCHAR**, GRADE** VARCHAR**)**
    - **Functional Dependency**
    a) STU_SSN, SNO, SEMESTER, YEAR, COURSENO → GRADE
    - **Primary Key**→STU_SSN, SNO, SEMESTER, YEAR, COURSENO
    - **Foreign Key** → STU_SSN, { SNO, SEMESTER, YEAR, COURSENO}
    - Here STU_SSN is the foreign key referring to SSN in Student table and {SNO, SEMESTER, YEAR, COURSENO} is referring to primary key of Semester table.

4. Join Paths

- Join Condition between Person and Student is through **SSN. Person.SSN = Student.SSN.**
- Join Condition between Person and Employee is through **SSN. Person.SSN = Employee.SSN.**
- Join Condition between Employee and Staff is through **SSN. Employee.SSN = Staff.SSN.**
- Join Condition between Employee and Faculty is through **SSN. Employee.SSN = Faculty.SSN.**
- Join Condition between Employee and Dependent is through **SSN. Employee.SSN = Dependent.EESSN.**
- Join Condition between Student and Department is through MINORDEPT and MAJORDEPT. **Student.MINORDEPT = Department.MINORDEPT and Student.MAJORDEPT = Department.MAJORDEPT**.
- Join Condition between Staff and Department is through CHAIRPERSON for chairs relation. **Department.CHAIRPERSON = Staff.SSN..**
- Join Condition between Staff and Department is through DNO for Works relation. **Staff.SSN=Department.SSN**
- Join Condition between Faculty and Department is through **DNO. Faculty.DEPTNO = Department.DNO.**
- Join Condition between Faculty and WORKS_FOR is through **SSN. Faculty.SSN = WORKS_FOR.FAC_SSN.**
- Join Condition between Department and WORKS_FOR is through **DNO. Department.DNO= WORKS_FOR.DEPTNO.**
- Join Condition between Department and Course is through **DNO. Department.DNO = Course.DEPTNO.**
- Join Condition between Student and Transcript is through **SSN. Student.SSN = Transcript.STU_SSN.**
- Join Condition between Course and Transcript is through CNO. **Course.CNO=Transcript.CourseNo.**
- Join Condition between Student and Study Plan is through **SSN. Student.SSN = STUDYPLAN.STU_SSN.**
- Join Condition between Course and Study Plan is through CNO. **Course.CNO=STUDYPLAN.CourseNo.**
- Join Condition between Section and Faculty is through faculty **ssn. Faculty.SSN = Section.Faculty.**
- Join Condition between Student and GRADE_REPORT is through Student **SSN. Student.SSN = GRADE_REPORT.STU_SSN.**
- Join Condition between Section and Grade_Report is through Section's primary key **Section.SNO = Grade_Report.SNO, Section.Semester = Grade_Report.Semester, Section.year = Grade_Report.year and Section.CourseNo = Grade_Report.CourseNo.**

5. Normalization

1. **Person (**<u>SSN</u> VARCHAR, FIRSTNAME VARCHAR, MIDDLENAME VARCHAR, LASTNAME VARCHAR, GENDER CHAR, PHONE VARCHAR, EMAIL VARCHAR,DATEOFBIRTH DATE, ZIP VARCHAR**)**
   - In Person table, There exists a functional dependency of Street, City, State on Zip Code, so Person table can be split into Address tables in order to be in 3 NF.

2. **Address** (<u>ZIP</u> VARCHAR ,STREET VARCHAR, CITY VARCHAR,STATE VARCHAR)
   - Primary Key for Address table is Zip.
   - Zip in Person table is the foreign key referring to Zip in Address table for maintaining referential integrity.
   - All other entities described in relational database schema are in 3NF. There exists functional dependency and no transitive dependency.

3. **DEPENDENT (name** VARCHAR**, EESSN** VARCHAR**, Relation)**
   - **Primary Key** → NAME**,** EESSN
   - **FOREIGN KEY** → EESSN
   - Here EESSN refers to SSN in Employee table, thus maintain referential integrity.
   - There are no multivalued attributes, all the nonkey attributes are functionally dependent the primary key attribute and there is no transitive property. Therefore, there is no need to normalize the table. It is in 3NF.

4. **STUDENT (**<u>SSN</u> VARCHAR**, STUDENTNO** VARCHAR**, CLASS** VARCHAR**, DEGREEPROG** VARCHAR**, MINORDEPT** VARCHAR**, MAJORDEPT** VARCHAR**)**
   - **Primary Key** → SSN
   - **Foreign Key** → SSN**,** MINORDEPT**,** MAJORDEPT
   - Here SSN is the foreign key referring to SSN in Person table and MINORDEPT and MAJORDEPT are referring to DEPTNO in Department table.
   - There are no multivalued attributes, all the nonkey attributes are functionally dependent the primary key attribute and there is no transitive property. Therefore, there is no need to normalize the table. It is in 3NF.

5. **STAFF (**<u>SSN</u> VARCHAR**, title** VARCHAR**, DEPTNO)**
   - **Primary Key** → SSN
   - **Foreign Key** → SSN
   - Here SSN is the foreign Key referring to SSN in Person table and DEPTNO is the foreign key referring to DNO in Department table.
   - There are no multivalued attributes, all the nonkey attributes are functionally dependent the primary key attribute and there is no transitive property. Therefore, there is no need to normalize the table. It is in 3NF.

6. **FACULTY (**<u>SSN</u> VARCHAR**, RANK** VARCHAR**, DEGREE** VARCHAR**, SALARY** NUMBER(10,2)**, EMPTYPE** VARCHAR**)**

- **Primary Key** → SSN
- **Foreign Key** → SSN
- Here SSN is the foreign Key referring to SSN in Person table.
- There are no multivalued attributes, all the nonkey attributes are functionally dependent the primary key attribute and there is no transitive property. Therefore, there is no need to normalize the table. It is in 3NF.

7. **DEPARTMENT (DNO** VARCHAR**, DNAME** VARCHAR**, COLLEGE** VARCHAR**, OFFICENO** **VARCHAR,** PHONE **VARCHAR, CHAIRPERSON** VARCHAR**)**
   - **Primary Key** → DNO
   - **Foreign Key** → CHAIRPERSON
   - Chairperson is the SSN referring to SSN in Person table.
   - There are no multivalued attributes, all the nonkey attributes are functionally dependent the primary key attribute and there is no transitive property. Therefore, there is no need to normalize the table. It is in 3NF.

8. **WORKS_FOR (FAC_SSN** VARCHAR**, DEPTNO** VARCHAR**)**
   - **Primary Key** → FAC_SSN**,** DEPTNO
   - **Foreign Key** → FAC_SSN, DEPTNO
   - Here FAC_SSN is the foreign Key referring to SSN in Faculty table and DEPTNO is the foreign key referring to DNO in Department table.
   - There are no multivalued attributes, all the nonkey attributes are functionally dependent the primary key attribute and there is no transitive property. Therefore, there is no need to normalize the table. It is in 3NF.

9. **COURSE (CNO** VARCHAR**, COURSENAME** VARCHAR**, CREDITS** NUMBER**, DEPTNO** VARCHAR**)**
   - **Primary Key** → CNO
   - **Foreign Key** → DEPTNO
   - Here DEPTNO is the foreign key referring to DNO in Department table.
   - There are no multivalued attributes, all the nonkey attributes are functionally dependent the primary key attribute and there is no transitive property. Therefore, there is no need to normalize the table. It is in 3NF.

10. **STUDYPLAN (STU_SSN** VARCHAR**, COURSENO** VARCHAR**)**
    - **Primary Key** → STU_SSN, COURSENO
    - **Foreign Key** → STU_SSN, COURSENO
    - STU_SSN is the foreign key referring to SSN in Student table and COURSENO is the foreign key referring to CNO in Course table.
    - There are no multivalued attributes, all the nonkey attributes are functionally dependent the primary key attribute and there is no transitive property. Therefore, there is no need to normalize the table. It is in 3NF.

11. **TRANSCRIPT (STU_SSN** VARCHAR**, COURSENO** VARCHAR**, GRADE** VARCHAR**)**
    - **Primary Key** → STU_SSN, COURSENO

- **Foreign Key →** STU_SSN, COURSENO
- There are no multivalued attributes, all the nonkey attributes are functionally dependent the primary key attribute and there is no transitive property. Therefore, there is no need to normalize the table. It is in 3NF.

12. **SECTION (<u>SNO</u>** VARCHAR**, <u>SEMESTER</u>** VARCHAR**, <u>YEAR</u>** VARCHAR**, DAY** VARCHAR**, TIME** VARCHAR**, ROOM** VARCHAR**, FACULTY** VARCHAR**, <u>COURSENO</u>** VARCHAR**)**
    - **Primary Key →** SNO
    - **Foreign Key →** FACULTY**,** COURSENO
    - Here Faculty is the foreign key referring to SSN in Faculty table and CourseNo is the foreign key referring to CNO in Course table.
    - There are no multivalued attributes, all the nonkey attributes are functionally dependent the primary key attribute and there is no transitive property. Therefore, there is no need to normalize the table. It is in 3NF.

13. **GRADE_REPORT (<u>STU_SSN</u>** VARCHAR**, <u>SNO</u>** VARCHAR**, <u>SEMESTER</u>** VARCHAR**, <u>YEAR</u>** NUMBER**, <u>COURSENO</u>** VARCHAR**, GRADE** VARCHAR**)**
    - **Primary Key→**STU_SSN, SNO, SEMESTER, YEAR, COURSENO
    - **Foreign Key →** STU_SSN, { SNO, SEMESTER, YEAR, COURSENO}
    - Here STU_SSN is the foreign key referring to SSN in Student table and {SNO, SEMESTER, YEAR, COURSENO} is referring to primary key of Semester table.
    - There are no multivalued attributes, all the nonkey attributes are functionally dependent the primary key attribute and there is no transitive property. Therefore, there is no need to normalize the table. It is in 3NF.

6. Implement the Relational Database
[1]. Creating tables

```
SQL> create table Address
  2  (
  3  zip varchar(5) primary key,
  4  street varchar(30),
  5  city varchar(30),
  6  state varchar(20)
  7  );

Table created.

SQL> create table Person
  2  (
  3  ssn varchar(10) primary key,
  4  firstName varchar(20) not null,
  5  middleName char(20),
  6  lastName varchar(20) not null,Gender varchar(1),
  7  dob date,
  8  zip varchar(5),
  9  phone varchar(10),
 10  FOREIGN KEY (zip) REFERENCES Address (zip)
```

```
 11  );

Table created.

SQL> create table employee
  2  (
  3  ssn varchar(10) not null,
  4  FOREIGN KEY (ssn) REFERENCES Person (ssn)
  5  );

Table created.

SQL> create table Dependent
  2  (
  3  name varchar(20) not null,
  4  essn varchar(10) not null,
  5  relation varchar(20),
  6  constraint depPK primary key (name,essn)
  7  );

Table created.
SQL> create table Department
  2  (
  3  dno varchar(10) primary key,
  4  dname varchar(10) not null,
  5  college varchar(10),
  6  officeNo varchar(10),
  7  chairperson varchar(10),
  8  constraint dnameUK unique (dname)
  9  );

Table created.
SQL> create table Staff
  2  (
  3  ssn varchar(10) primary key,
  4  title varchar(10),
  5  deptno varchar(10),
  6  FOREIGN KEY (ssn) REFERENCES Employee (ssn),
  7  FOREIGN KEY (deptno) REFERENCES Department (dno)
  8  );

Table created.

SQL> create table Faculty
  2  (
  3  ssn varchar(10) primary key,
  4  degree varchar(10),
  5  rank varchar(20),
  6  EmpType varchar(10),
  7  deptNo varchar(10),
  8  FOREIGN KEY (ssn) REFERENCES Employee (ssn)
  9  );

Table created.

SQL> create table Student
  2  (
  3  ssn varchar(10) primary key,
```

```
  4   studentNo varchar(10) not null,
  5   degreeProg varchar(5),
  6   minorDept varchar(10),
  7   majorDept varchar(10),
  8   class varchar(10),
  9   FOREIGN KEY (ssn) REFERENCES Person(ssn),
 10   FOREIGN KEY (minorDept) REFERENCES Department (dno),
 11   FOREIGN KEY (majorDept) REFERENCES Department (dno),
 12   constraint snoUK unique (studentNo)
 13   );

Table created.
SQL> create table Works_For
  2   (
  3   facSSN varchar(10) not null,
  4   deptNo varchar(10) not null,
  5   constraint worksForPK primary key (facSSN,deptNo),
  6   FOREIGN KEY (facSSN) REFERENCES Faculty (ssn),
  7   FOREIGN KEY (deptNO) REFERENCES DEpartment (dno)
  8   );

Table created.
SQL> create table Course
  2   (
  3   cno varchar2(10) not null,
  4   cname varchar2(50),
  5   credits number(5),
  6   deptNo varchar(10) not null,
  7   FOREIGN KEY (deptNo) REFERENCES Department (dno)
  8   );

Table created.
SQL> create table Study_Plan
  2   (
  3   stuSSN varchar(10) not null,
  4   courseNo varchar(10) not null,
  5   constraint studyplanPK primary key (stuSSN,courseNO),
  6   FOREIGN KEY (stuSSN) REFERENCES Student (ssn),
  7   FOREIGN KEY (courseNO) REFERENCES Course(cno)
  8   );

Table created.
SQL> create table Transcript
  2   (
  3   stuSSN varchar(10) not null,
  4   courseNo varchar(10) not null,
  5   grade varchar(2) not null,
  6   constraint transcriptPK primary key (stuSSN,courseNo),
  7   FOREIGN KEY (stuSSN) REFERENCES Student(ssn),
  8   FOREIGN KEY (courseNo) REFERENCES Course (cno)
  9   );

Table created.
SQL> create table Section
  2   (
  3   sno varchar(10) not null,
  4   semester varchar(10) not null,
  5   year number(4) not null,
```

```
  6  meetingDay varchar(10),
  7  meetingTime varchar(10),
  8  meetingRoom varchar(10),
  9  instructor varchar(10),
 10  courseNo varchar(10) not null,
 11  constraint sectionPK primary key (sno,semester,year,courseNo),
 12  FOREIGN KEY (instructor) REFERENCES Faculty (ssn),
 13  FOREIGN KEY (courseNo) REFERENCES Course (cno)
 14  );

Table created.
SQL> create table Grade_Report
  2  (
  3  stuSSN varchar(10) not null,
  4  sno varchar(10) not null,
  5  semester varchar(10) not null,
  6  year number(4) not null,
  7  courseNo varchar(10) not null,
  8  grade varchar(2),
  9           constraint    gradereportPK    primary    key
(stuSSN,sno,semester,year,courseNo),
 10  FOREIGN KEY (stuSSN) REFERENCES Student (ssn),
 11   FOREIGN KEY (sno,semester,year,courseNo) REFERENCES  Section
(sno,semester,year,courseNo)
 12  );

Table created.
SQL> create table Salary_Scale
  2  (
  3  rank varchar(20) not null,
  4  empType varchar(9) not null,
  5  salary number(8,2),
  6  constraint ssPK primary key (rank,empType)
  7  );

Table created.

SQL> create table Grad_Student
  2  (
  3  stuSSN varchar(10) primary key,
  4  facSSN varchar(10) not null,
  5  FOREIGN KEY (stuSSN) REFERENCES Faculty (ssn),
  6  FOREIGN KEY (facSSN) REFERENCES Student (ssn)
  7  );

Table created.
SQL> create table Course_Prerequisite
  2  (
  3  cno varchar(10) not null,
  4  precno varchar(10) not null,
  5  constraint cprPK primary key (cno,precno),
  6  FOREIGN KEY (cno) REFERENCES Course (cno),
  7  FOREIGN KEY (precno) REFERENCES Course (cno)
  8  );

Table created.
```

[2]. Entering values into the tables
1. Address

```
SQL>   insert   into   Address   values('65432',   '2001   M
Rent','Hialeah','SF');

1 row created.

SQL>            insert            into            Person
values('765439876','Mike','S','Smith','M','11-MAR-
85','65432','8765554334');

1 row created.

SQL> insert into Address values('23121','2121 B Street','Belle
Grove','NY');

1 row created.

SQL> insert into Address values('09867','2    North
     Street','Coconut Beach','NY');

1 row created.

SQL>   insert   into   Address   values('98765','31   Westside
street','Parkgate',   'JK');

1 row created.

SQL> insert into Address values('09809','71   MC
     Street','Markland','FY');

1 row created.
SQL> insert into Address values('33328', 'Apt 14', 'Davie','FL');

1 row created.

SQL>   insert   into   Address   values('33389','8210   SW   41
CT','Davie','FL');

1 row created.

SQL> insert into Address values('09956','8130 NW 21 ST','Coconut
Beach', 'FL');

1 row created.

SQL> insert into Address values('98764','Ap 21','Coral Springs',
     'FL');

1 row created.

SQL> insert into Address values('09803','71   MC
     Street','Miami','FL');

1 row created.
```

University Database

2. Person

```
SQL>                 insert                 into                 Person
values('865361231','Patty','P','Preston','F','01-APR-
72','23121','0988342133');

1 row created.

SQL> insert into Person
values('875435123','Megan','M','Woods','F','21-OCT-
89','09867','5674398764');

1 row created.

SQL> insert into Person
values('875676767','Chris','V','Seilr','M','26-JUL-
78','98765','6775545422');

1 row created.

SQL> insert into Person
values('864542278','John','L','Carry','M','10-AUG-
81','09809','652457732');

1 row created.
SQL>                 insert                 into                 Person
values('975164333','Shaival','P','Shah','M','12-DEC-
93','33328','9856712572');

1 row created.

SQL>                 insert                 into                 Person
values('743678246','Gaurav','P','Jain','M','01-JUN-
94','33389','0988342781');

1 row created.

SQL> insert into Person
values('952561853','Shreeraj','F','Pawar','M','21-OCT-
93','09956','5674398757');

1 row created.

SQL>
SQL> insert into Person
values('942567194','Jaimin','V','Patel','M','26-JUL-
94','98764','6775543674');

1 row created.

SQL> insert into Person
values('864169257','Priyana','L','Chwada','F','10-AUG
94','09803','6524569843');

1 row created.
```

University Database

3. Employee

```
SQL> insert into Employee values('865361231');

1 row created.

SQL> insert into Employee values('875435123');

1 row created.

SQL> insert into Employee values('875676767');

1 row created.

SQL> insert into Employee values('864542278');

1 row created.
```

4. Department

```
SQL> insert into Department          values(3,'Pharmacy','PHMD',
'098762286','CN543');

1 row created.

SQL> insert into Department  values(1,'Computer  Science','CISC',
'779646655','CS123');

1 row created.

SQL> insert into Department          values(2,'Information
Technology','MSIT','658537372','IT456');

1 row created.
```

5. Staff

```
SQL> insert into Staff values('765439876','Dean',1);

1 row created.

SQL> insert into Staff values('865361231','Coordinator',2);

1 row created.

SQL> insert into Staff values('875435123','Dean',3);

1 row created.

SQL> insert into Staff values('875676767','Coordinator',3);

1 row created.

SQL> insert into Staff values('864542278','Dean',2);
```

```
    1 row created.
```

6. Faculty

```
SQL> insert into Faculty     values('865361231',
      'MCA','Assistant Professor','PART-TIME',1);

1 row created.

SQL> insert into Faculty     values('875676767',
      'MPhil','Assistant Professor','PART-TIME',2);

1 row created.

SQL> insert into Faculty     values('875435123',
      'PHD','Permanent Professor','FULL-TIME',3);

1 row created.

SQL> insert into Faculty     values('765439876',
      'PHD','Permanent Professor','FULL-TIME',1);

1 row created.
```

7. Course

```
SQL> insert into Course values('CS2','Computer Networking',3,1);

1 row created.

SQL> insert into Course values('CS3','Operating Systems',3,2);

1 row created.

SQL> insert into Course values('CS4','Computer Architecture',3,2);

1 row created.

SQL> insert into Course values('CS5','Data Strucutures',3,2);

1 row created.
```

8. COURSE_PREREQUISITE

```
SQL> insert into COURSE_PREREQUISITE values   ('CS1','CS3');

1 row created.

SQL> insert into COURSE_PREREQUISITE values   ('CS2','CS4');

1 row created.
```

9. SALARY_SCALE

```
SQL> insert into SALARY_SCALE values ('Lecturer',   'part-time',
      110000);
```

```
1 row created.

SQL> insert into SALARY_SCALE values ('Lecturer',    'full-time',
      123431.54);

1 row created.

SQL> insert into   SALARY_SCALE   values   ('Associate   Professor',
      'full-time',     753532.45);

1 row created.
```

10. Student

```
SQL> insert into Student                    values('975164333',
      'SN201','BCA',1,3,'graduate');

1 row created.

SQL> insert into Student    values('743678246',
      'SN202','MCA',3,1,'graduate');

1 row created.

SQL> insert into Student    values('952561853',
      'SN203','PHD',2,3,'phd');

1 row created.

SQL> insert into Student    values('942567194',
      'SN204','BSC',1,3,'freshman');

1 row created.

SQL> insert into Student    values('864169257',
      'SN205','PHD',null,1,'phd');

1 row created.
```

11. Dependent

```
SQL> insert into Dependent values('Richard','098762286','Spouse');

1 row created.

SQL> insert into Dependent values('Michel','779646655',  'Son');

1 row created.

SQL> insert into Dependent                    values('Kim
maggio','753338862','Spouse');

1 row created.

SQL> insert into Dependent values('Garry','962371274','daughter');

1 row created.
```

```
SQL> insert into Dependent values('Sheron','658537372','Son');

1 row created.
```

12. WORKS_FOR

```
SQL> insert into WORKS_FOR values('765439876',1);

1 row created.

SQL> insert into WORKS_FOR values('865361231',3);

1 row created.

SQL> insert into WORKS_FOR values('875676767',2);

1 row created.

SQL> insert into WORKS_FOR values('875435123',1);

1 row created.
```

13. Section

```
SQL> insert into Section
values(1,'summer',2017,'monday','11:00AM','Desantis301',
'865361231','CS2');

1 row created.

SQL> insert into Section
     values(2,'winter',2017,'tuesday','10:00AM','Desantis401',
'875676767','CS2');

1 row created.

SQL> insert into Section
     values(1,'summer',2017,'fridat','09:00AM',    'Desantis401',
'865361231','CS1');

1 row created.

SQL> insert into Section    values(1,'summer',2017,
     'thursday','04:30PM',  'Desantis401', '875435123','CS4');

1 row created.

SQL> insert into Section    values(1,'summer',2017,
     'monday','06:30PM', 'Desantis401','765439876','CS5');

1 row created.
```

14. STUDY_PLAN

```
SQL> insert into STUDY_PLAN values('864169257','CS1');

1 row created.
```

```
SQL> insert into STUDY_PLAN values('942567194','CS2');

1 row created.

SQL> insert into STUDY_PLAN values('952561853','CS3');

1 row created.

SQL> insert into STUDY_PLAN values('743678246','CS4');

1 row created.
```

15. Transcript

```
SQL> insert into Transcript values('952561853','CS1','A');

1 row created.

SQL> insert into Transcript values('743678246','CS2','A+');

1 row created.

SQL> insert into Transcript values('942567194','CS3','B');

1 row created.

SQL> insert into Transcript values('864169257','CS1','B+');

1 row created.
```

16. GRADE_STUDENT

```
SQL> insert into Grad_Student values('975164333','765439876');

1 row created.

SQL> insert into Grad_Student values('743678246','765439876');

1 row created.

SQL> insert into Grad_Student values('952561853','875435123');

1 row created.

SQL> insert into Grad_Student values('942567194','875676767');

1 row created.

SQL> insert into Grad_Student values('864169257','875676767');

1 row created.
```

7. SQL Queries
1. Query 1

```
SQL> @u1
SQL> run
  1* select d.dname, count(*) from Department d left join Student
        s on  d.dno =      s.majorDept or    d.dno =       minorDept
group by    dname

DNAME                                COUNT(*)
------------------------------ ----------
Computer Science                        4
Information Technology                  1
Pharmacy                                4
```

2. Query 2

```
SQL> @u2
SQL> run
  1  select d.dname, wf.facSSN     as    ssn,  count(s.sno)      as
       NoOfcourses from  Department  d left     join Works_For wf on
       d.dno =      wf.deptNo
  2* left   join  Section    s on  wf.facSSN  =     s.instructor
group by    d.dname, wf.facSSN     order by    d.dname     asc

DNAME                                SSN         NOOFCOURSES
------------------------------ ---------- -----------
Computer Science               765439876            4
Computer Science               875435123            1
Information Technology         875676767            1
Pharmacy                       865361231            0
```

3. Query 3

```
SQL> @u3
SQL> run
  1  select d.dname, c.cno, c.cname from Department d
  2* left   join  Course     c on  d.dno =      c.deptNo group    by
       d.dname, c.cno, c.cname order     by    d.dname      asc

DNAME                                CNO
------------------------------ ----------
CNAME
----------------------------------------------------
Computer Science               CS1
Database

Computer Science               CS2
Computer Networking

Information Technology         CS3
Operating Systems


DNAME                                CNO
------------------------------ ----------
CNAME
```

```
      --------------------------------------------------
      Information Technology        CS4
      Computer Architecture

      Information Technology        CS5
      Data Strucutures

      Pharmacy



      6 rows selected.
```

4. Query 4

```
SQL> @u4
SQL> run
  1  select c.cno,       p.precno    from Course      c
  2  left  join  Course_Prerequisite   p on c.cno = p.cno  group
     by    c.cno,        p.precno
  3* order by   c.cno asc

CNO        PRECNO
---------- ----------
CS1        CS3
CS2        CS4
CS3
CS4
CS5
```

5. Query 5

```
SQL> @u5
SQL> run
  1  select d.dname,    count(distinct    wf.facSSN)    as    prof_ct,
     count(load)/count(distinct   wf.facSSN) as    avgLoad    from
     Department  d
  2  left   join (select     deptNo, facSSN from Works_For)     wf
     on   wf. deptNo =    d.dno left join
  3* (select     count(*) as load, instructor from Section  group
     by instructor)   t on wf.facSSN =    t.instructor    group
     by    d.dname

DNAME                            PROF_CT    AVGLOAD
------------------------------ ---------- ----------
Information Technology              1          1
Pharmacy                            1          0
Computer Science                    2          1
```

6. Query 6

```
SQL> @u6
SQL> run
  1  select d.dname as  stu_ct,     sum(t2.credits)   as     tl_cr,
     sum(t2.credits)/count(distinct t1.ssn) as  avg_cr
  2  from Department   d,(select   ssn, majorDept, minorDept from
     Student) t1,(select    c.credits, t.stuSSN    as    ssn  from
```

```
         Transcript t,     Course     c      where t.courseNo = c.cno)
         t2
    3* where (t1.majorDept=d.dno     or    t1.minorDept=d.dno)     and
         (t2.ssn=t1.ssn)  group by d.dname

STU_CT                             TL_CR     AVG_CR
------------------------------ ---------- ----------
Information Technology               3          3
Pharmacy                             9          3
Computer Science                     9          3
```

7. Query 7

```
SQL> @u7
SQL> run
  1  select f.ssn,     stu.studentNo    from    Faculty    f    left
     join Section s on f.ssn      = s.instructor left    join
  Grade_Report g
  2  on s.sno     =    g.sno and   s.semester  =    g.semester and
     s.year      =    g.year      and    s.courseNo  =
     g.courseNo left  join Student stu on    stu.ssn      =
     stuSSN
  3* group by   f.ssn,     stu.studentNo order    by    f.ssn asc

SSN          STUDENTNO
---------- ----------
765439876  SN203
765439876  SN204
765439876  SN205
765439876
865361231
875435123  SN202
875676767

7 rows selected.
```

8. Query 8

```
SQL> @u8
SQL> run
  1  select f.ssn,      d.dname     from Faculty    f     left
     join  Department d
  2  on     exists     ((select    cno   from Course      where
     deptNo    =    d.dno)     intersect
  3  (select       courseNo   from Section s    where s.instructor
     =    f.ssn))
  4* group by    f.ssn,      d.dname    order by    f.ssn asc

SSN          DNAME
---------- ------------------------------
765439876  Computer Science
765439876  Information Technology
865361231
875435123  Information Technology
875676767  Computer Science
```

9. Query 9

```
SQL> @u9
SQL> run
  1  select d.dname, f.ssn from   Department d      left      join
     Works_For wf
  2  on    d.dno =     wf.deptNo
  3  left  join Faculty    f
  4  on    wf.facSSN  =     f.ssn
  5  left  join Salary_Scale    ss
  6  on    f.rank    =     ss.rank   and   f.empType  =
     ss.empType where (select    count(*)   from Section
     where instructor =     f.ssn)    >     2
  7* and   ss.salary  <    (select   avg(sal.salary)  from
     Faculty    fac, Salary_Scale   sal, Works_For      wf
     where fac.rank   =     sal.rank   and   fac.empType =
sal.empType and   wf.facSSN  =     fac.ssn   and   wf.deptNo  =
     d.dno) group by  d.dname, f.ssn

no rows selected
```

10. Query 10

```
SQL> @u10
SQL> run
  1  select ssn, count(stuSSN)     from Faculty
  2* left join Grad_Student on    ssn  =     facSSN    group  by
     ssn

SSN         COUNT(STUSSN)
---------- -------------
765439876              2
865361231              0
875435123              1
875676767              2
```

11. Query 11

```
SQL> @u11
SQL> run
  1  select dname,     count(*) from    Department inner join
Student     on   dno  =     majorDept  or    dno  =
     minorDept group  by    dname
  2* having count(*)   >    (select   avg(count) from (select
     count(*) as count from Department   inner join Student    on
     dno  =     majorDept or    dno  =     minorDept group  by
     dno))

DNAME                          COUNT(*)
------------------------------ ----------
Computer Science                      4
Pharmacy                              4
```

12. Query 12

```
SQL> @u12
SQL> run
```

26

```
  1  select d.dname,    sum(s.salary)    as    avg_salary          from
       Department  d
  2  inner join  Works_For  wf on d.dno =      wf.deptNo inner join
       Faculty     f
  3  on     wf.facSSN  =     f.ssn inner join  Salary_Scale  s    on
       f.rank     =      s.rank      and   f.empType    =
       s.empType group   by    d.dname
  4  having sum(s.salary)    >     (select     avg(avg_salary)
       from (select    d.dname,    sum(s.salary)     as
       avg_salary from  Department  d
  5  inner join  Works_For  wf    on    d.dno =      wf.deptNo
       inner join  faculty     f on wf.facSSN  =      f.ssn inner
       join  Salary_Scale     s
  6* on     f.rank     =      s.rank      and   f.empType    =
       s.empType   group by    d.dname))

no rows selected
```

13. Query 13

```
SQL> @u13
SQL> run
  1  select d.dname, wf.facSSN,    count(g.facSSN)  as   count from
       Department  d left     join Works_For wfon     d.dno =
       wf.deptNo
  2  inner join  (Grad_Student    g     inner join  Student   s   on
       g.stuSSN   =     s.ssn and   s.degreeProg     =     'PHD')
       on    wf.facSSN  =     g.facSSN group   by    d.dname,
       wf.facSSN, d.dno
  3  having count(g.facSSN)  >     (select     avg(phd_count)  from
       (select     count(*)   as    phd_count
  4* from  Works_For   wf1, Grad_Student     gs,    Student     stu
       where wf1.deptNo =     d.dno and   wf1.facSSN =
       gs.facSSN and    gs.stuSSN  =     stu.ssn          and
       stu.degreeProg='PHD')) order by    d.dname    asc

no rows selected
```

14. Query 14

```
SQL> @u14
SQL> run
  1  select s.ssn from Student    s where    not  exists ((select
       distinct c.cno
  2  from  Course    c,   Course_Prerequisite   p      where
       c.deptNo=s.majorDept and
  3* c.cno =    p.precno) minus (select    courseNo    from
       Transcript where stuSSN    =    s.ssn))

SSN
----------
975164333
743678246
952561853
942567194
864169257
```

15. Query 15

```
SQL> @u15
SQL> run
  1  select stu.ssn    from Student stu where not   exists
  2  ((select    s.courseNo from Person      p, Section  s  where
     p.ssn =    s.instructor and p.lastname =     'Smith')
  3* minus (select    courseNo   from Transcript where stuSSN
     =    stu.ssn))

no rows selected
```

16. Query 16

```
SQL> @u16
SQL> run
  1  select distinct    stu.ssn
  2  from   Student    stu, Transcript t
  3  where stu.ssn    =    t.stuSSN   and  not  exists ((select
     courseNo    from
  4  Transcript  where  stuSSN=stu.ssn) minus (select  s.courseNo
      from
  5* Person p,    Section   s where   p.ssn =    s.instructor
     and  p.lastName =    'Smith'))

SSN
----------
743678246
864169257
942567194
952561853
```

17. Query 17

```
SQL> @u17
SQL> run
  1  select stu.ssn    from  Student    stu   where not   exists
  2  ((select    t.courseNo from  Person p, Transcript   t where
  3  p.ssn =    t.stuSSN   and   p.firstName =    'Shaival')
  4* minus (select    courseNo   from Transcript where stuSSN
     =    stu.ssn))

SSN
----------
743678246
864169257
942567194
952561853
975164333
```

18. Query 18

```
SQL> @u18
SQL> run
  1  select stu.ssn    from  Student    stu   where not   exists
```

28

```
   2  ((select courseNO from Study_Plan  where stuSSN       =
      stu.ssn) minus
   3* (select     courseNo   from Transcript  where stuSSN      =
      stu.ssn    and   grade in    ('A', 'A-', 'B+', 'B', 'B-',
      'C+')))

SSN
----------
864169257
975164333
```

19. Query 19

```
SQL> @u19
SQL> run
  1  select stu.ssn    from Student     stu   where not   exists
  2  ((select courseNo from Study_Plan where  stuSSN      =
     stu.ssn) minus
  3* (select      courseNo   from  Transcript where stuSSN     =
     stu.ssn))

SSN
----------
864169257
975164333

SQL> spool off
```