

ORGANIZING DATA FOR ANALYTICS

FINAL PROJECT



Bala Durga Sridevi Kesavarapu

**Table of Contents**

**Executive Summary** 3

**Detail Project description**

Problem Statement 4

Identification of all entities, attributes, and relationships 4

Functional requirements 6

**Conceptual Database Design**

ER Diagram 7

Description of the entities, attributes, and relationships 8

Normalization 10

**Database Implementation**

Instance charts 11

Create Tables 14

Insert values 20

Script to delete Database 20

Queries 21

**Executive Summary**

The Supplemental Instruction Program (SIP) is a peer-led study group program for undergraduate students. SIP is an academic support program that utilizes collaborative learning strategies to assist students on the learning process. Historically speaking, the SIP is generally offered for the more advanced courses (i.e. higher difficulty). The program is staffed by one university employee, who is also the manager, Graduate Assistants who administer the program, and undergraduate student SIP Leaders. Each course may have more than one section and SIP offers distinct sessions for each section of the course. These sessions are led by SIP Leaders. These leaders are undergraduate students who have taken the course before and have been successful; earning a letter grade of B+ or higher. Leaders facilitate three review sessions each week, during which they incorporate learning strategies with instructional content specific to current coursework. Leaders communicate with the professor of the associated course to be aware of the current course material. Students are welcome to come and go at any point during the session. The sessions are open to any student enrolled in that particular course section. Students do not have to register for the SIP sessions in order to attend. Participants earn a reward if they attend 2 or more sessions, across courses, within a semester. Participants’ grades are recorded. Reports are generated for participants who attend a minimum of 3 sessions to display whether attendance results in a change in participants’ grades. The goal of SIP is:

* To demonstrate increased self-sufficiency, self-confidence and course knowledge compared to non-participants.
* To help participants earn higher test grades, course grades and persist more often than non-participants.
* To develop confidence in leaders as group facilitators as a result of SIP.

**Detail Project description**

* + **Problem statement**

Supplemental Instruction Program (SIP) needs an improved database to perform the following operations:

1. To document all individuals involved in the program
2. To track attendance every week.
3. To store the data specific to students, SIP leaders, classroom schedule.
4. To find students who attended 2 or more SIP sessions to compare their grades.
5. To find students who attended more than 2 SIP sessions to reward them.
6. Include location of sessions.
7. Create and deliver a streamlined database.
8. To provide meaningful information to determine the effectiveness of the SIP program.

**Identify all entities, attributes, and relationships that are commonly encountered with organization.**

|  |  |
| --- | --- |
| **Entity** | **Type** |
| Person |  |
| Department |  |
| Staff Member | Sub-type of Person |
| Administrator | Sub-type of Staff Member |
| Graduate Assistant | Sub-type of Staff Member |
| SIP Leader | Sub-type of Staff Member |
| Professor | Sub-type of Staff Member |
| Student | Sub-type of Person |
| Participant | Sub-type of Student |
| Student Athlete | Sub-type of Student |
| Course |  |
| Section |  |
| Session |  |
| Location |  |
| Attendance | Associative between Session and Participant |
| Reward |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity A** | **Relationship** | **Cardinality** | **Entity B** |
| SIP Leader | facilitates | 1:1 | section |
| Course offering | contains | 1:N | section |
| section | meets in | 1:1 | location |
| location | houses | 1:N | section |
| section | has | 1:N | session |
| participant | attends | 1:N | session |
| session | attended by | 1:N | participant |
| professor | associated with | 1:N | section |
| section | associated with | 1:1 | professor |
| SIP Leader | assigned to | 1:1 | professor |
| professor | assigned to | 1:N | SIP leader |
| participant | earns | 1:1 | reward |
| reward | earned by | 1:N | participant |
| staff member | may be a | complete disjoint | manager |
| staff member | may be a | complete disjoint | graduate assistant |
| staff member | may be a | complete disjoint | SIP leader |
| department | has | 1:1 | manager |
| manager | has | 1:N | graduate assistant |
| manager | has | 1:N | SIP leader |
| participant | may be a | incomplete overlapping | student athlete |
| student athlete | is a |  | participant |

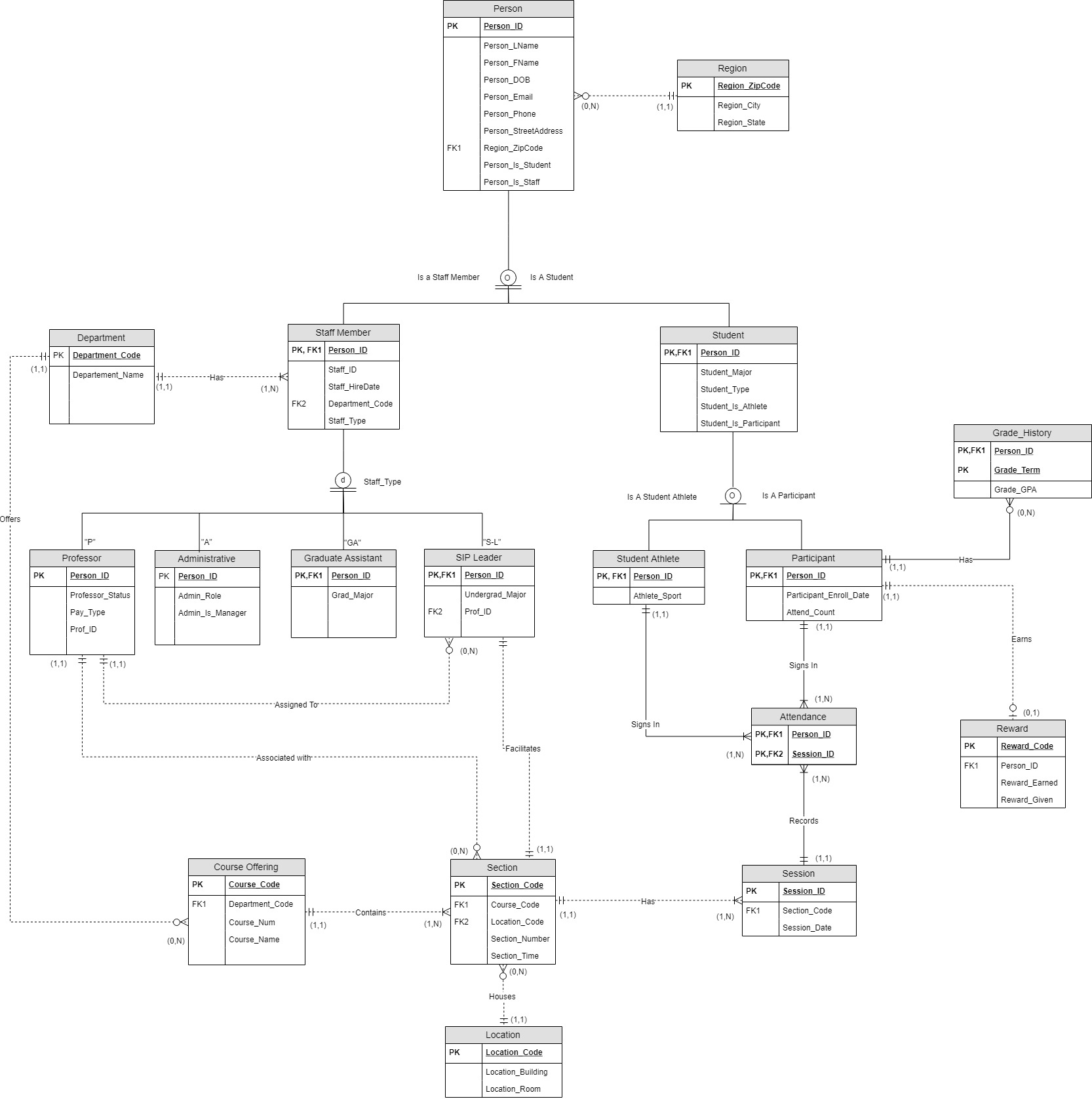
**Collect and document the functional requirements: what is the database required to do?**

Typical functional requirements include:

1. The database stores all the Entities, Attributes and their Relationships constraints.
2. Querying the database and getting the required information is easy.
3. With increase in the size of data, the size of the database can be increased.
4. The data can be loaded easily from the flat files (.CSV, Excel) to the database.
5. Implementation of the Business rules based on the requirement is feasible.

**Conceptual Database Design**

* + **ER Diagram**
    - **Draw a complete entity relationship diagram that depicts your database. You can use any software to draw the ER diagram.**



**Describe the entities, relationships, their respective attributes, the primary keys and constraints. State clearly any additional assumptions you have made.**

**Region:**

REGION has the REGION\_ZIPCODE as the Primary Key and additional attributes REGION\_CITY and REGION\_STATE.

**Person:**

PERSON has the PERSON\_ID as the Primary Key and has REGION\_ZIPCODE as a Foreign Key. Additional attributes are PERSON\_LNAME, PERSON\_FNAME, PERSON\_DOB, PERSON\_EMAIL, PERSON\_PHONE, PERSON\_STREETADDRESS, PERSON\_IS\_STUDEN, PERSON\_IS\_STAFF, and PERSON\_STATE. PERSON is a supertype in a specialization hierarchy representation, in a total overlapping relationship to STAFF\_MEMBER and STUDENT.

**Staff Member:**

STAFF\_MEMBER has the PERSON\_ID as the Primary Key as well as a Foreign Key, and DEPARTMENT\_CODE as a Foreign Key. Additional attributes are STAFF\_ID, STAFF\_HIREDATE, and STAFF\_TYPE. STAFF\_MEMBER is a subtype of PERSON, thus inheriting attributes of the PERSON table. STAFF\_MEMBER also has a total disjoint relationship with subtypes PROFESSOR, ADMINISTRATIVE, GRADUATE\_ASSISTANT, and SIP\_LEADER.

**Department:**

DEPARTMENT has the DEPARTMENT\_CODE as the Primary Key and additional attribute DEPARTMENT\_NAME.

**Student:**

STUDENT has the PERSON\_ID as a Primary Key as well as a Foreign Key, and it has inherited attributes from the PERSON table. Additional attributes are STUDENT\_MAJOR, STUDENT\_TYPE, STUDENT\_IS\_ATHLETE, and STUDENT\_IS\_PARTICIPANT)

**Professor:**

PROFESSOR has PERSON\_ID as the Primary Key and has inherited attributes from the STAFF MEMBER table. Additional attributes are PERSON\_ID, PROFESSOR\_STATUS, PAY\_TYPE, PROF\_ID.

**Administrative:**

ADMINISTRATIVE has PERSON\_ID as the Primary Key and has inherited attributes from the STAFF MEMBER table. Additional attributes are PERSON\_ID, ADMIN\_ROLE, and ADMIN\_IS\_MANAGER.

**Graduate Assistant:**

GRADUATE\_ASSISTANT has PERSON\_ID as the Primary Key and has inherited attributes from the STAFF MEMBER table. Additional attributes are PERSON\_ID, and GRAD\_MAJOR.

**SIP Leader:**

SIP\_LEADER has PERSON\_ID has both primary key as well as foreign key and PROF\_ID as foreign key. It also has UNDERGRAD\_MAJOR as an attribute which will have a list of undergrad majors of the students.

**Student Athlete:**

STUDENT\_ATHLETE has PERSON\_ID which is both Primary and foreign key of the table. It has ATHLETE\_SPORT as an attribute which says weather a student is an athlete or not.

**Participant:**

PARTICIPANT has PERSON\_ID which is both Primary and foreign key of the table. It has an attribute PARTICIPANT\_ENROLL\_DATE which has the dates of which the participants enrolled. It also has another attribute PARTICIPANT\_ATTENDANCE\_COUNT which has the number of sessions a participant attended.

**Attendance:**

ATTENDANCE has a composite key of PERSON\_ID, SESSION\_ID which are also foreign keys of the table.

**Grade History:**

GRADE\_HISTORY has PERSON\_ID attribute as both Primary Key and Foreign Key, thus making GRADE\_HISTORY a weak entity. Other attributes are GRADE\_TERM and GRADE\_GPA.

**Reward:**

REWARD is a strong entity having REWARD\_CODE as Primary Key and PERSON\_ID as the Foreign Key. Additional attributes are PERSON\_ID which is FK to Participant entity, REWARD\_EARNED\_DATE and REWARD\_GIVEN\_DATE.

**Course Offering:**

COURSE\_OFFERING is a strong entity having COURSE\_CODE as Primary Key and DEPARTMENT\_CODE as the Foreign Key. Additional attributes are DEPARTMENT\_CODE which is FK to Department entity, Course Num and COURSE\_NAME.

**Section:**

SECTION is a strong entity having SECTION\_CODE as the Primary Key, COURSE\_CODE and LOCATION\_CODE as Foreign Keys. SECTION participates in one-to-many relationship with SESSION. Other attributes are SECTION\_NUMBER and SECTION\_TIME.

**Session:**

SESSION has SESSION\_ID as Primary Key and SECTION\_CODE as Foreign Key and SESSION\_DATE.

**Location:**

LOCATION is a clustered entity which groups Room and Buildings and relationships. LOCATION has LOCATION\_CODE as Primary Key and is a strong entity. Other attributes are LOCATION\_BUILDING and LOCATION\_ROOM.

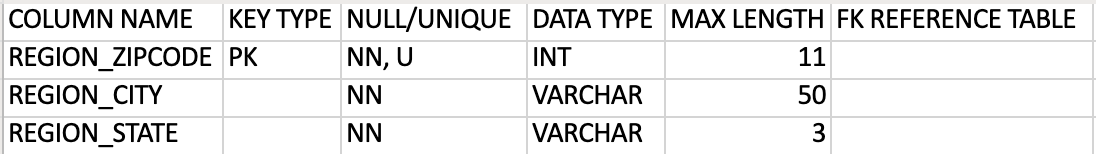
**Normalization - Decompose the relations into 3NF.**

All the tables are in 3NF. The normalized tables are described in the instance charts below.

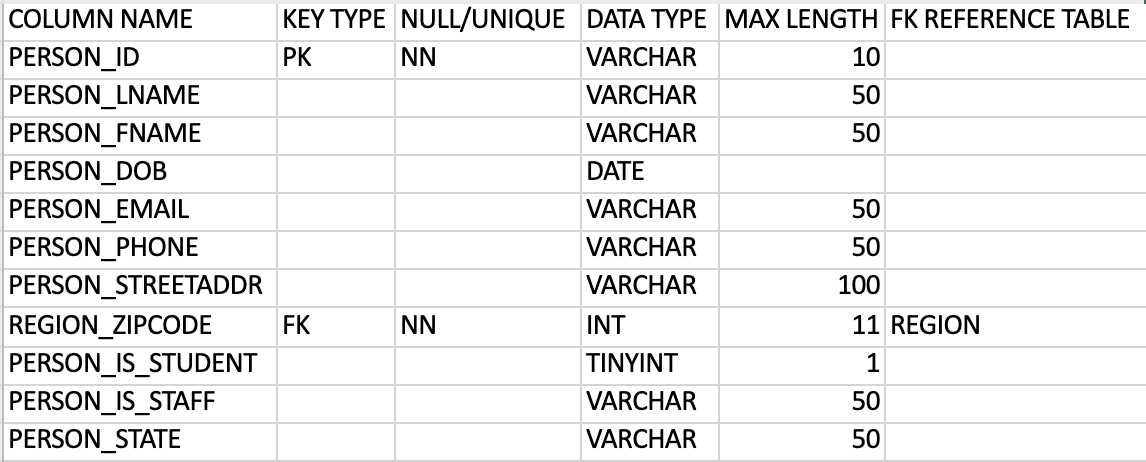
**Database Implementation**

**Develop instance charts for all tables in your database.**

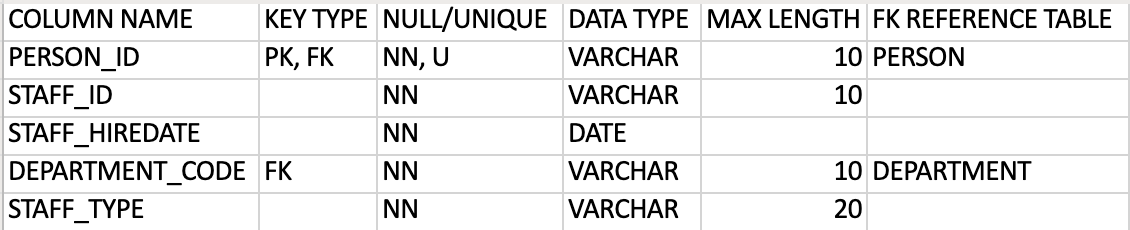
1. REGION (REGION\_ZIPCODE, REGION\_CITY, REGION\_STATE)



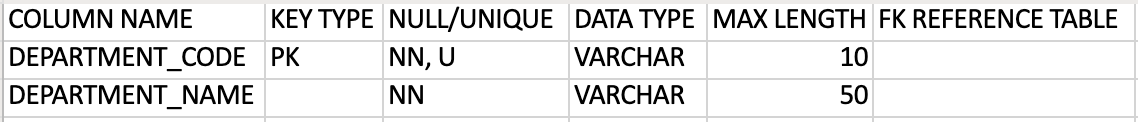
2. PERSON (PERSON\_ID, PERSON\_LNAME, PERSON\_FNAME, PERSON\_DOB, PERSON\_EMAIL, PERSON\_PHONE, PERSON\_STREETADDR, REGION\_ZIPCODE, PERSON\_IS\_STUDENT, PERSON\_IS\_STAFF)



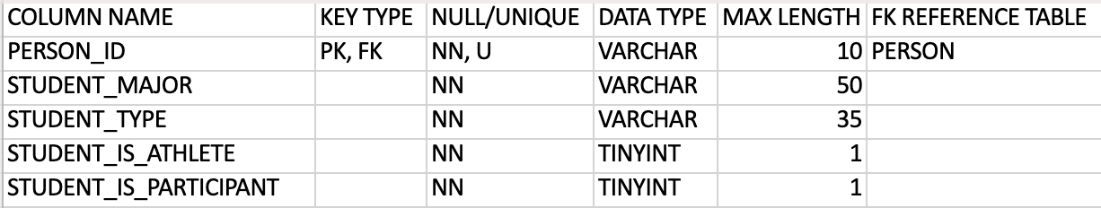
3. STAFF\_MEMBER (PERSON\_ID, STAFF\_ID, STAFF\_HIREDATE, DEPARTMENT\_CODE, STAFF\_TYPE)



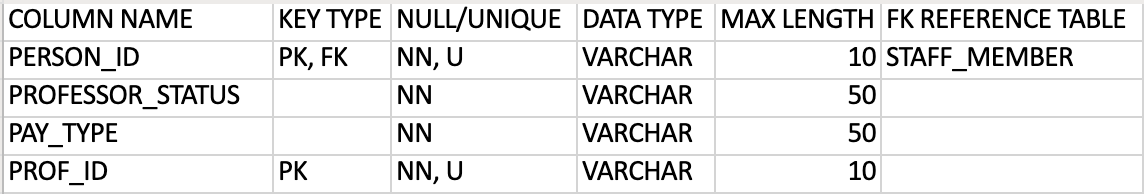
4. DEPARTMENT (DEPARTMENT\_CODE, DEPARTMENT\_NAME)



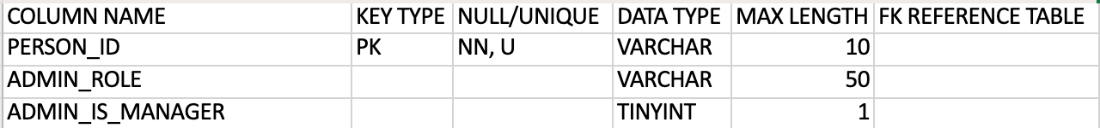
5. STUDENT (PERSON\_ID, STUDENT\_MAJOR, STUDENT\_TYPE, STUDENT\_IS\_ATHLETE, STUDENT\_IS\_PARTICIPANT)



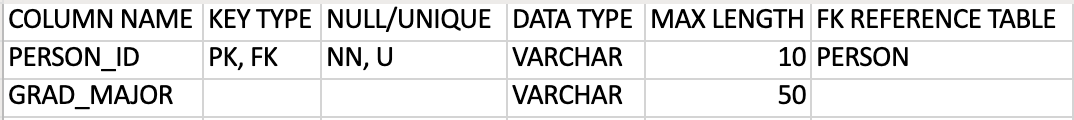
6. PROFESSOR (PERSON\_ID, PROF\_ID, PROFESSOR\_STATUS, PAY\_TYPE)



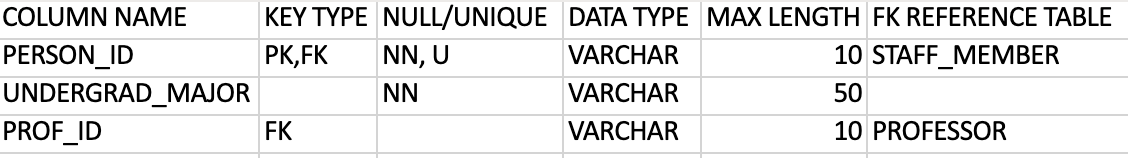
7. ADMINISTRATIVE (PERSON\_ID, ADMIN\_ROLE, ADMIN\_IS\_MANAGER)



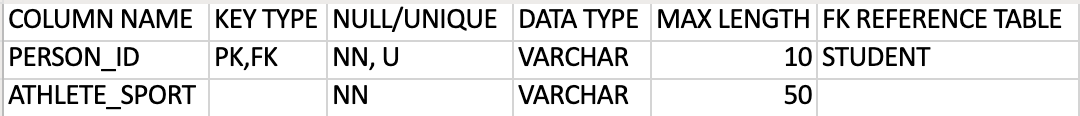
8. GRADUATE\_ASSISTANT (PERSON\_ID, GRAD\_MAJOR)



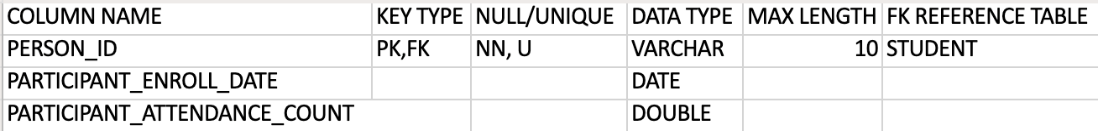
9. SIP\_LEADER (PERSON\_ID, UNDERGRAD\_MAJOR, PROF\_ID)



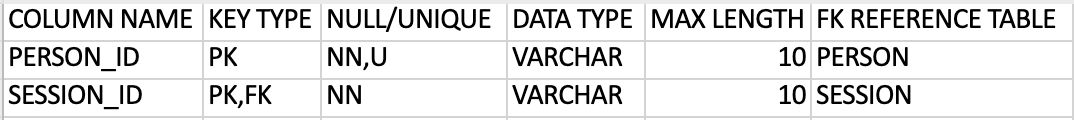
10. STUDENT\_ATHLETE (PERSON\_ID, ATHLETE\_SPORT)



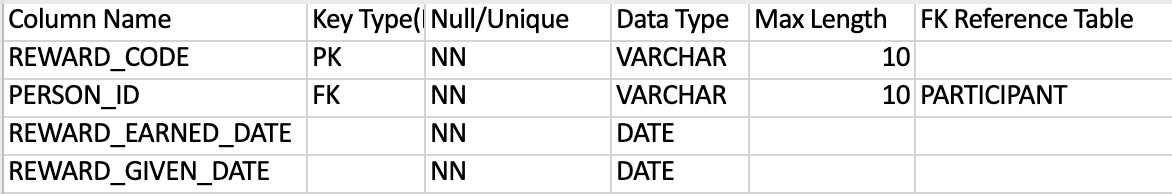
11. PARTICIPANT (PERSON\_ID, PARTICIPANT\_ENROLL\_DATE, PARTICIPANT\_ATTENDANCE\_COUNT)



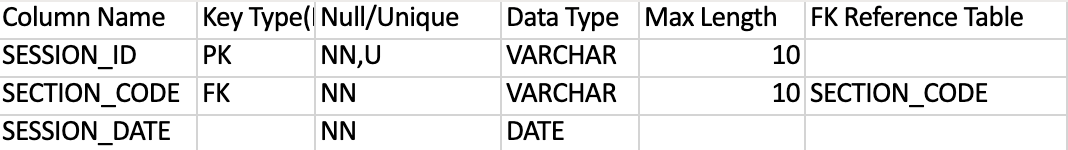
12. ATTENDANCE (PERSON\_ID, SESSION\_ID)



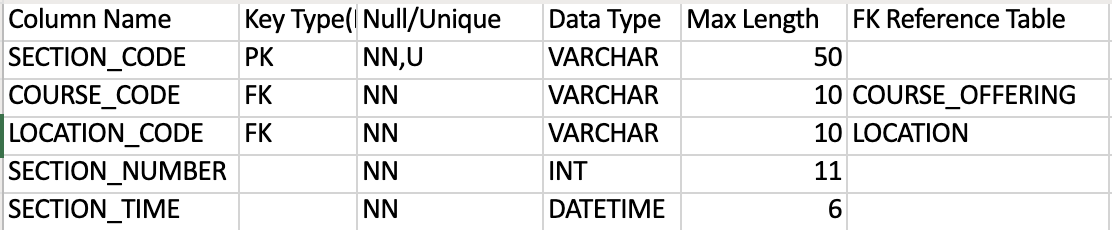
13. REWARD (REWARD\_CODE, PERSON\_ID, REWARD\_EARNED\_DATE, REWARD\_GIVEN\_DATE, REWARD\_TYPE)



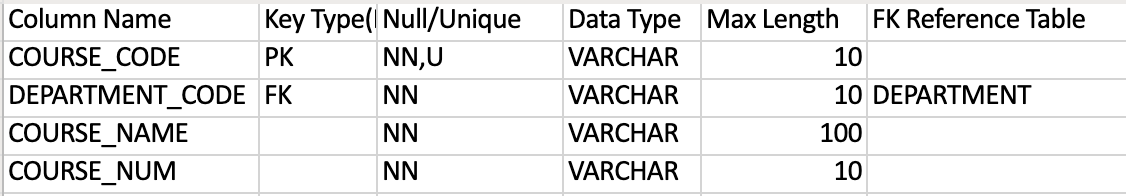
14. SESSION (SESSION\_ID, SECTION\_CODE, SESSION\_DATE)



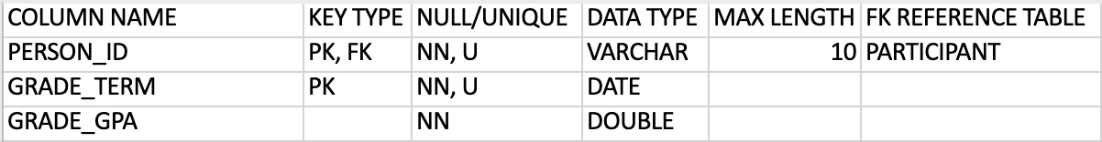
15. SECTION (SECTION\_CODE, COURSE\_CODE, LOCATION\_CODE, SECTION\_NUMBER, SECTION\_TIME)



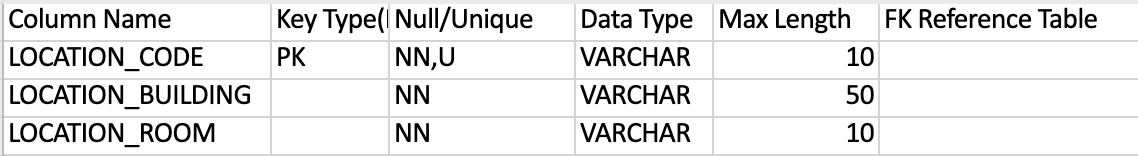
16. COURSE\_OFFERING (COURSE\_CODE, DEPARTMENT\_CODE, COURSE\_NAME, COURSE\_NUM)



17. GRADE\_HISTORY (PERSON\_ID, GRADE\_TERM, GRADE\_GPA, GRADE\_GPA\_CHANGE)



18. LOCATION (LOCATION\_CODE, LOCATION\_BUILDING, LOCATION\_ROOM)



**Implement the database on a MySQL server. Specifically, write a SQL script to do the following:**

1. **Create the Tables. Each Table must have a Primary Key constraint, Foreign Key constraints where applicable, and other constraints as necessary.**

/\* ISM-671 Spring2019

Group Project

Group 1 \*/

/\* Create Statements \*/

/\* Region; Jeremiah: \*/

CREATE TABLE REGION (

REGION\_ZIPCODE VARCHAR (11) NOT NULL UNIQUE,

REGION\_CITY VARCHAR (50),

REGION\_STATE CHAR (3),

PRIMARY KEY(REGION\_ZIPCODE)

);

/\* Person; Jeremiah: \*/

CREATE TABLE PERSON (

PERSON\_ID VARCHAR (10) NOT NULL UNIQUE,

PERSON\_LNAME VARCHAR (50) NOT NULL,

PERSON\_FNAME VARCHAR (50) NOT NULL,

PERSON\_DOB DATE NOT NULL,

PERSON\_EMAIL VARCHAR (50) NOT NULL,

PERSON\_PHONE VARCHAR (50) NOT NULL,

PERSON\_STREETADDR VARCHAR (100)

REGION\_ZIPCODE INT (11) NOT NULL,

PERSON\_IS\_STUDENT TINYINT (1) NOT NULL,

PERSON\_IS\_STAFF VARCHAR (50) NOT NULL,

PERSON\_STATE VARCHAR (50)

PRIMARY KEY(PERSON\_ID),

FOREIGN KEY(REGION\_ZIPCODE) REFERENCES REGION(REGION\_ZIPCODE) ON UPDATE CASCADE

);

/\* Department; Jeremiah: \*/

CREATE TABLE DEPARTMENT (

DEPARTMENT\_CODE VARCHAR (10) NOT NULL UNIQUE,

DEPARTMENT\_NAME VARCHAR (50) NOT NULL,

PRIMARY KEY(DEPARTMENT\_CODE)

);

/\* Staff Member; Jeremiah: \*/

CREATE TABLE STAFF\_MEMBER (

PERSON\_ID VARCHAR (10) NOT NULL UNIQUE,

STAFF\_ID VARCHAR (10) NOT NULL UNIQUE,

STAFF\_HIREDATE DATE NOT NULL,

DEPARTMENT\_CODE VARCHAR (10) NOT NULL,

STAFF\_TYPE VARCHAR (20) NOT NULL,

PRIMARY KEY(PERSON\_ID),

FOREIGN KEY(PERSON\_ID) REFERENCES PERSON(PERSON\_ID) ON UPDATE CASCADE ON DELETE RESTRICT,

FOREIGN KEY(DEPARTMENT\_CODE) REFERENCES DEPARTMENT(DEPARTMENT\_CODE) ON UPDATE CASCADE ON DELETE RESTRICT

);

/\* Student; Prithvi: \*/

CREATE TABLE STUDENT (

PERSON\_ID VARCHAR (10) NOT NULL UNIQUE,

STUDENT\_ID VARCHAR (10) NOT NULL UNIQUE,

STUDENT\_MAJOR VARCHAR (35),

STUDENT\_TYPE CHAR (10),

STUDENT\_IS\_ATHLETE TINYINT (1),

STUDENT\_IS\_PARTICIPANT TINYINT (1),

PRIMARY KEY (PERSON\_ID),

FOREIGN KEY (PERSON\_ID) REFERENCES PERSON (PERSON\_ID) ON UPDATE CASCADE

);

/\* Professor; Prithvi: \*/

CREATE TABLE PROFESSOR (

PERSON\_ID VARCHAR (10) NOT NULL UNIQUE,

PROFESSOR\_STATUS VARCHAR (10),

PAY\_TYPE VARCHAR (50),

PROF\_ID VARCHAR (10) NOT NULL

PRIMARY KEY (PERSON\_ID)

);

/\* Administrative; Prithvi: \*/

CREATE TABLE ADMINISTRATIVE (

PERSON\_ID VARCHAR (10) NOT NULL UNIQUE,

ADMIN\_ROLE VARCHAR (10),

ADMIN\_IS\_MANAGER TINYINT (1),

PRIMARY KEY (PERSON\_ID)

);

/\* Graduate Assistant; Prithvi: \*/

CREATE TABLE GRADUATE\_ASSISTANT (

PERSON\_ID VARCHAR (10) NOT NULL UNIQUE,

GRAD\_MAJOR VARCHAR (50),

PRIMARY KEY (PERSON\_ID),

FOREIGN KEY (PERSON\_ID) REFERENCES PERSON (PERSON\_ID) ON UPDATE CASCADE,

FOREIGN KEY (STUDENT\_ID) REFERENCES STUDENT (STUDENT\_ID) ON UPDATE CASCADE

);

/\* SIP Leader; Sridevi: \*/

CREATE TABLE SIP\_LEADER (

PERSON\_ID VARCHAR (10) NOT NULL UNIQUE,

UNDERGRAD\_MAJOR VARCHAR (50),

PROF\_ID VARCHAR (10)

PRIMARY KEY (PERSON\_ID),

FOREIGN KEY (PERSON\_ID) REFERENCES PERSON (PERSON\_ID) ON UPDATE CASCADE,

FOREIGN KEY (STUDENT\_ID) REFERENCES STUDENT (STUDENT\_ID) ON UPDATE CASCADE

);

/\* Student Athlete; Sridevi: \*/

CREATE TABLE STUDENT\_ATHLETE (

PERSON\_ID CHAR (11) NOT NULL UNIQUE,

ATHLETE\_SPORT CHAR (10),

PRIMARY KEY (PERSON\_ID),

FOREIGN KEY (PERSON\_ID) REFERENCES PERSON (PERSON\_ID) ON UPDATE CASCADE

);

/\* Participant; Sridevi: \*/

CREATE TABLE PARTICIPANT (

PERSON\_ID VARCHAR (10) NOT NULL UNIQUE,

PARTICIPANT\_ENROLL\_DATE DATE,

PARTICIPANT\_ATTENDANCE\_COUNT DOUBLE,

PRIMARY KEY (PERSON\_ID),

FOREIGN KEY (PERSON\_ID) REFERENCES PERSON (PERSON\_ID) ON UPDATE CASCADE

);

/\* Attendance; Sridevi: \*/

CREATE TABLE ATTENDANCE (

PERSON\_ID VARCHAR (10) NOT NULL UNIQUE,

SESSION\_ID VARCHAR (10) NOT NULL UNIQUE,

PRIMARY KEY (PERSON\_ID, SESSION\_ID),

FOREIGN KEY(PERSON\_ID) REFERENCES PERSON,

FOREIGN KEY(SESSION\_ID) REFERENCES SESSION ON UPDATE CASCADE

);

/\* Reward; Anita: \*/

CREATE TABLE REWARD (

REWARD\_CODE VARCHAR (10) NOT NULL UNIQUE,

PERSON\_ID VARCHAR (10) NOT NULL UNIQUE,

REWARD\_EARNED\_DATE DATE,

REWARD\_GIVEN\_DATE DATE,

PRIMARY KEY (REWARD\_CODE),

FOREIGN KEY (PERSON\_ID) REFERENCES PARTICIPANT (PERSON\_ID) ON UPDATE CASCADE ON DELETE RESTRICT);

/\* Session; Anita: \*/

CREATE TABLE SESSION (

SESSION\_ID VARCHAR (50) NOT NULL UNIQUE,

SECTION\_CODE VARCHAR (10) NOT NULL UNIQUE,

COURSE\_CODE VARCHAR (10) NOT NULL,

LOCATION\_CODE VARCHAR (10) NOT NULL,

SECTION\_NUMBER INT (11) NOT NULL,

SECTION\_TIME DATETIME (6) NOT NULL,

PRIMARY KEY (SESSION\_ID),

FOREIGN KEY (SECTION\_CODE) REFERENCES SECTION (SECTION\_CODE) ON UPDATE CASCADE ON DELETE RESTRICT);

/\* Section; Anita: \*/

CREATE TABLE SECTION (

SECTION\_CODE CHAR (10) NOT NULL UNIQUE,

COURSE\_CODE CHAR (10) NOT NULL UNIQUE,

LOCATION\_CODE CHAR (10) NOT NULL UNIQUE,

SECTION\_NUMBER VARCHAR (10) NOT NULL,

SECTION\_TIME VARCHAR (10) NOT NULL,

PRIMARY KEY (SECTION\_CODE),

FOREIGN KEY (COURSE\_CODE) REFERENCES COURSE\_OFFERING (COURSE\_CODE) ON UPDATE CASCADE ON DELETE RESTRICT,

FOREIGN KEY (LOCATION\_CODE) REFERENCES LOCATION (LOCATION\_CODE) ON UPDATE CASCADE ON DELETE RESTRICT);

/\* Course Offering; Anita: \*/

CREATE TABLE COURSE\_OFFERING (

COURSE\_CODE VARCHAR (10) NOT NULL UNIQUE,

DEPARTMENT\_CODE VARCHAR (10) NOT NULL UNIQUE,

COURSE\_NAME VARCHAR (100) NOT NULL,

COURSE\_NUM VARCHAR (10) NOT NULL,

PRIMARY KEY (COURSE\_CODE),

FOREIGN KEY (DEPARTMENT\_CODE) REFERENCES DEPARTMENT (DEPARTMENT\_CODE) ON UPDATE CASCADE ON DELETE RESTRICT);

/\* Location; Anita: \*/

CREATE TABLE LOCATION (

LOCATION\_CODE VARCHAR (10) NOT NULL UNIQUE,

LOCATION\_BUILDING VARCHAR (50) NOT NULL,

LOCATION\_ROOM VARCHAR (10) NOT NULL,

PRIMARY KEY (LOCATION\_CODE));

/\* Grade\_History; Anita: \*/

CREATE TABLE GRADE\_HISTORY (

PERSON\_ID VARCHAR (10) NOT NULL UNIQUE,

GRADE\_TERM DATE NOT NULL,

GRADE\_GPA DOUBLE NOT NULL,

PRIMARY KEY(PERSON\_ID),

FOREIGN KEY (PERSON\_ID) REFERENCES PARTICIPANT (PERSON\_ID) ON UPDATE CASCADE ON DELETE RESTRICT);

1. **Insert rows into each Table. Make up your own data. Make sure that each table has between 2 and 10 rows.**

We have inserted at least two rows for all the tables using realistic fictional values**.**

1. **Write a SQL script to delete the entire database, i.e., drop all Tables.**

DROP DATABSE GROUP\_PROJECT;

**(OR)**

DROP TABLE PERSON

DROP TABLE REGION

DROP TABLE DEPARTMENT

DROP TABLE STAFF\_MEMBER

DROP TABLE STUDENT

DROP TABLE PROFESSOR

DROP TABLE ADMINISTRATIVE

DROP TABLE GRADUATE\_ASSISTANT

DROP TABLE SIP\_LEADER

DROP TABLE STUDENT\_ATHLETE

DROP TABLE PARTICIPANT

DROP TABLE REWARD

DROP TABLE SESSION

DROP TABLE SECTION

DROP TABLE COURSE\_OFFERING

DROP TABLE LOCATION

DROP TABLE ATTENDANCE

DROP TABLE GRADE\_HISTORY

1. **Develop five queries to retrieve some meaningful data from the database.**
2. Provide a list of Professors who are assigned to the SIP Leaders.

**QUERY:**

SELECT T1. PERSON\_LNAME AS PROF\_FNAME,

T1. PERSON\_FNAME AS PROF\_LNAME,

T2. PERSON\_LNAME AS SIPLEADER\_LNAME,

T2. PERSON\_FNAME AS SIPLEADER\_FNAME

FROM

(SELECT P. [PERSON\_ID], [PERSON\_LNAME],[PERSON\_FNAME], PROF\_ID

FROM [DBO]. [PERSON] P JOIN [DBO]. [PROFESSOR] PR

ON P. PERSON\_ID = PR. PERSON\_ID) T1

LEFT JOIN

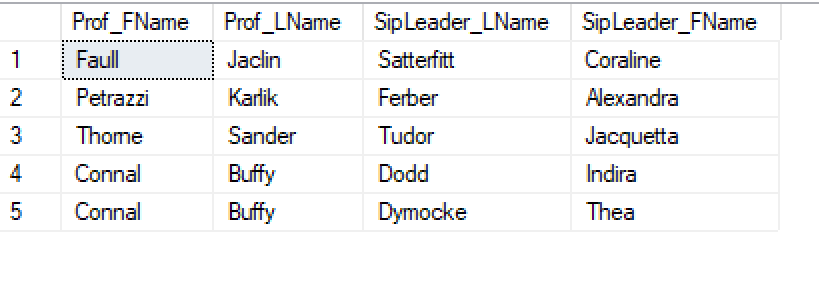
(SELECT P. [PERSON\_ID],[PERSON\_LNAME],[PERSON\_FNAME] , PROF\_ID

FROM [DBO].[PERSON] P JOIN [DBO].[SIP\_LEADER] S

ON P. PERSON\_ID = S. PERSON\_ID) T2

ON T1. PROF\_ID = T2. PROF\_ID

**OUTPUT:**



1. Provide a list of Students who received rewards.

**QUERY:**

SELECT P.PERSON\_FNAME,

P.PERSON\_LNAME,

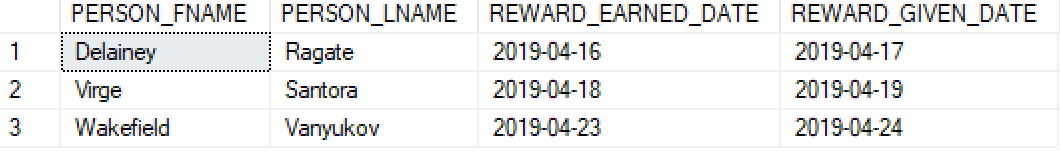
[REWARD\_EARNED\_DATE],

[REWARD\_GIVEN\_DATE]

FROM [dbo].[REWARD] R

JOIN [dbo].[PERSON] P ON R.PERSON\_ID = P.PERSON\_ID

**OUTPUT:**



1. Provide a report of Students who got an increase in the GPA by attending the program.

**QUERY:**

SELECT P.PERSON\_FNAME AS 'STUDENT FIRST NAME',P.PERSON\_LNAME AS 'STUDENT LAST NAME'

FROM PERSON P JOIN STUDENT S ON P.PERSON\_ID = S.PERSON\_ID

JOIN PARTICIPANT PA ON S.PERSON\_ID = PA.PERSON\_ID

JOIN GRADE\_HISTORY G ON PA.PERSON\_ID = G.PERSON\_ID

JOIN (

SELECT ST.PERSON\_ID, ST.GRADE\_GPA AS 'START GPA', FN.GRADE\_GPA AS 'END GPA'

FROM (SELECT PERSON\_ID, GRADE\_GPA FROM GRADE\_HISTORY WHERE (PERSON\_ID, GRADE\_TERM) IN

(SELECT PERSON\_ID, MIN(GRADE\_TERM) FROM GRADE\_HISTORY GROUP BY PERSON\_ID))ST

JOIN

(SELECT PERSON\_ID, GRADE\_GPA FROM GRADE\_HISTORY WHERE (PERSON\_ID, GRADE\_TERM) IN

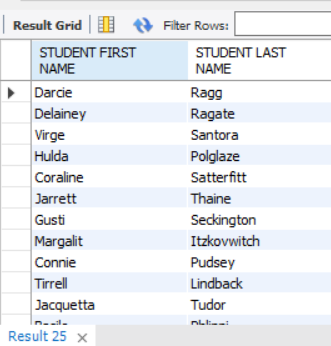
(SELECT PERSON\_ID, MAX(GRADE\_TERM) FROM GRADE\_HISTORY GROUP BY PERSON\_ID))FN

ON ST.PERSON\_ID = FN.PERSON\_ID

WHERE FN.GRADE\_GPA > ST.GRADE\_GPA)CH ON G.PERSON\_ID = CH.PERSON\_ID

GROUP BY P.PERSON\_ID;

**OUTPUT:**



1. Provide listing of Staff Members who are from NC and sort the list by Last Name.

**QUERY:**

SELECT PERSON\_FNAME AS 'STAFF FIRST NAME', PERSON\_LNAME AS 'STAFF LAST NAME', PERSON\_EMAIL AS EMAIL, DEPARTMENT\_NAME AS DEPARTMENT

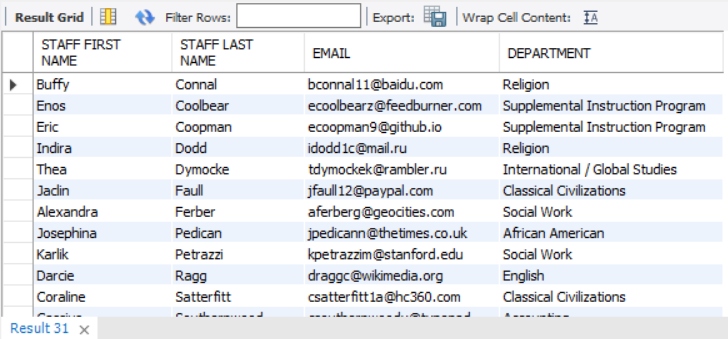
FROM REGION R JOIN PERSON P ON R.REGION\_ZIPCODE = P.REGION\_ZIPCODE

JOIN STAFF\_MEMBER ST ON P.PERSON\_ID = ST.PERSON\_ID

JOIN DEPARTMENT D ON ST.DEPARTMENT\_CODE = D.DEPARTMENT\_CODE

ORDER BY PERSON\_LNAME;

**OUTPUT:**



1. Provide a list of Students who are doing Graduate Assistantship.

**QUERY:**

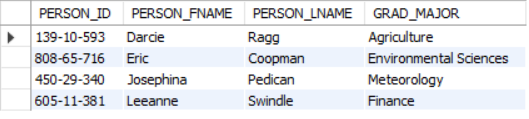
SELECT P.PERSON\_ID, P.PERSON\_FNAME, P.PERSON\_LNAME, G.GRAD\_MAJOR

FROM PERSON P, GRADUATE\_ASSISTANT G

WHERE P.PERSON\_ID = G.PERSON\_ID

ORDER BY P.PERSON\_FNAME

**OUTPUT:**



1. Provide a list of Professors who have more than one SIP Leader.

**QUERY:**

SELECT P.PERSON\_LNAME, P.PERSON\_FNAME, S.PROF\_ID

FROM PROFESSOR PR

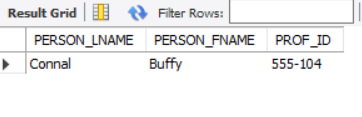
JOIN PERSON P ON P.PERSON\_ID= PR.PERSON\_ID

JOIN SIP\_LEADER S ON

S.PROF\_ID= PR.PROF\_ID

GROUP BY P.PERSON\_ID

HAVING COUNT(S.PROF\_ID) > 1



1. Provide a list of Students who participated in 2 or more SIP sessions.

**QUERY:**

SELECT STUDENT.PERSON\_ID, PERSON.PERSON\_FNAME, PERSON.PERSON\_LNAME

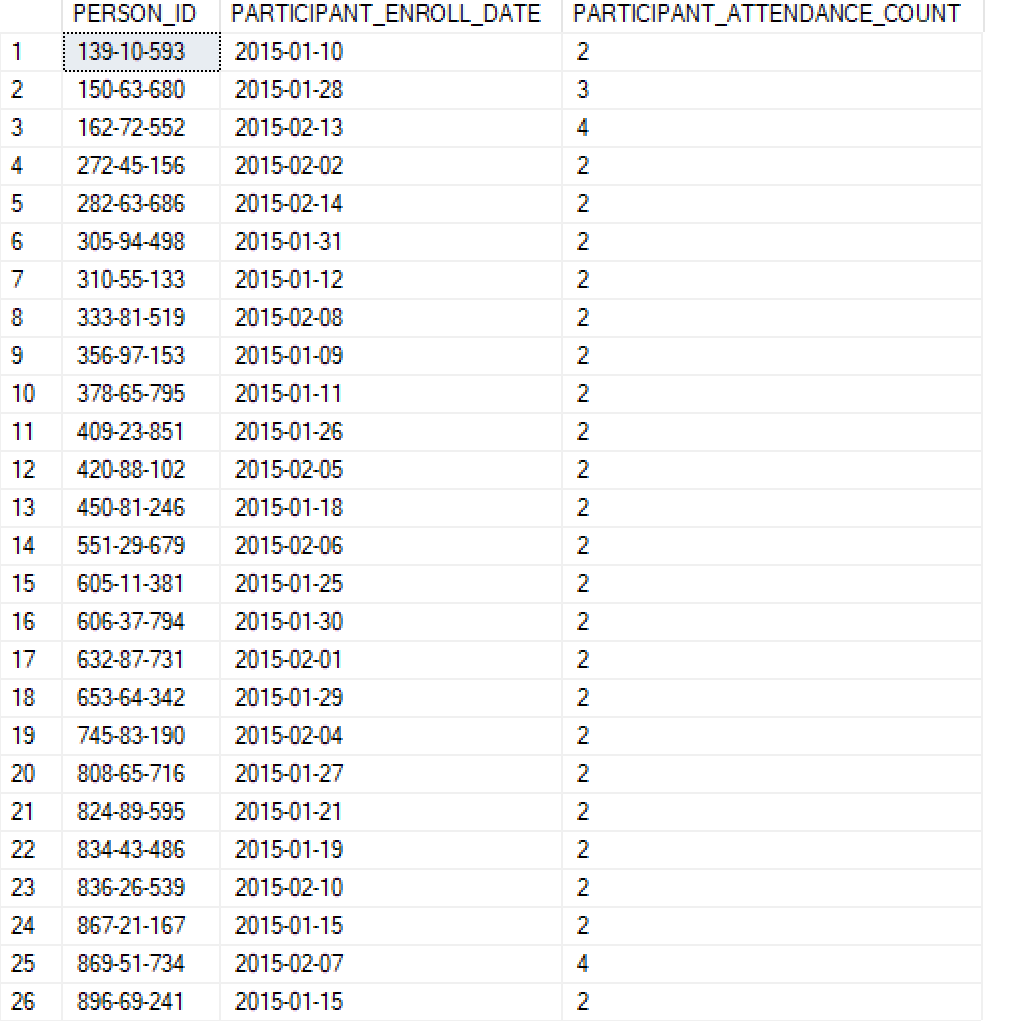
FROM STUDENT, PERSON, PARTICIPANT

WHERE STUDENT.PERSON\_ID = PERSON.PERSON\_ID AND PERSON.PERSON\_ID = PARTICIPANT.PERSON\_ID

GROUP BY PERSON.PERSON\_ID, PARTICIPANT.PARTICIPANT\_ATTENDANCE\_COUNT, STUDENT.PERSON\_ID, PERSON.PERSON\_FNAME, PERSON.PERSON\_LNAME

HAVING PARTICIPANT\_ATTENDANCE\_COUNT >=2;

**OUTPUT:**



1. Provide a list of Students who are undergrads.

SELECT STUDENT.PERSON\_ID, PERSON.PERSON\_FNAME, PERSON.PERSON\_LNAME

FROM PERSON, STUDENT

WHERE STUDENT.PERSON\_ID = PERSON.PERSON\_ID AND PERSON.PERSON\_IS\_STUDENT = 1 AND STUDENT.STUDENT\_TYPE = 'Undergrad'

GROUP BY PERSON.PERSON\_ID, STUDENT.PERSON\_ID, PERSON.PERSON\_FNAME, PERSON.PERSON\_LNAME

ORDER BY PERSON.PERSON\_FNAME, PERSON.PERSON\_LNAME;

**OUTPUT:**

