

Main Project Report

Group ID: krc353_1

40068000	David-Étienne Pigeon
26242219	Marc Hegedus
40032171	Iana Belichtka
40152713	Liam Blount
26943799	Luiz Goncalves
26251978	Deniz Akcal

COMP353

Instructor: Khaled Jababo

Concordia University

August 12th 2019

Group Account: KRC353_1

Faculty of Engineering and Computer Science Expectations of Originality

This form sets out the requirements for originality for work submitted by students in the Faculty of Engineering and Computer Science. Submissions such as assignments, lab reports, project reports, computer programs and take-home exams must conform to the requirements stated on this form and to the Academic Code of Conduct. The course outline may stipulate additional requirements for the course.

1. Your submissions must be your own original work. Group submissions must be the original work of the students in the group.
2. Direct quotations must not exceed 5% of the content of a report, must be enclosed in quotation marks, and must be attributed to the source by a numerical reference citation¹. Note that engineering reports rarely contain direct quotations.
3. Material paraphrased or taken from a source must be attributed to the source by a numerical reference citation.
4. Text that is inserted from a web site must be enclosed in quotation marks and attributed to the web site by numerical reference citation.
5. Drawings, diagrams, photos, maps or other visual material taken from a source must be attributed to that source by a numerical reference citation.
6. No part of any assignment, lab report or project report submitted for this course can be submitted for any other course.
7. In preparing your submissions, the work of other past or present students cannot be consulted, used, copied, paraphrased or relied upon in any manner whatsoever.
8. Your submissions must consist entirely of your own or your group's ideas, observations, calculations, information and conclusions, except for statements attributed to sources by numerical citation.
9. Your submissions cannot be edited or revised by any other student.
10. For lab reports, the data must be obtained from your own or your lab group's experimental work.
11. For software, the code must be composed by you or by the group submitting the work, except for code that is attributed to its sources by numerical reference.

You must write one of the following statements on each piece of work that you submit:

For individual work: **"I certify that this submission is my original work and meets the Faculty's Expectations of Originality"**, with your signature, I.D. #, and the date.

For group work: **"We certify that this submission is the original work of members of the group and meets the Faculty's Expectations of Originality"**, with the signatures and I.D. #s of all the team members and the date.

A signed copy of this form must be submitted to the instructor at the beginning of the semester in each course.

I certify that I have read the requirements set out on this form, and that I am aware of these requirements. I certify that all the work I will submit for this course will comply with these requirements and with additional requirements stated in the course outline.

Course Number: COMP 353
Name: Deniz Akca
Signature: Deniz Akca

Instructor: Khaled Jabado
I.D. # 26252978
Date: August 12th 2019

Iana Belitchka 40032171

¹ Rules for reference citation can be found in "Form and Style" by Patrich MacDonagh and Jack Bordan, fourth edition, May, 2000, available at <http://www.encs.concordia.ca/scs/Forms/Form&Style.pdf>.

Approved by the ENCS Faculty Council February 10, 2012

David-Etienne Pigeon

40068000

David K...

Luiz Goncalves

26943799

Luiz

Liam Blount-Castagnay

40152713

Liam

E/R DIAGRAM

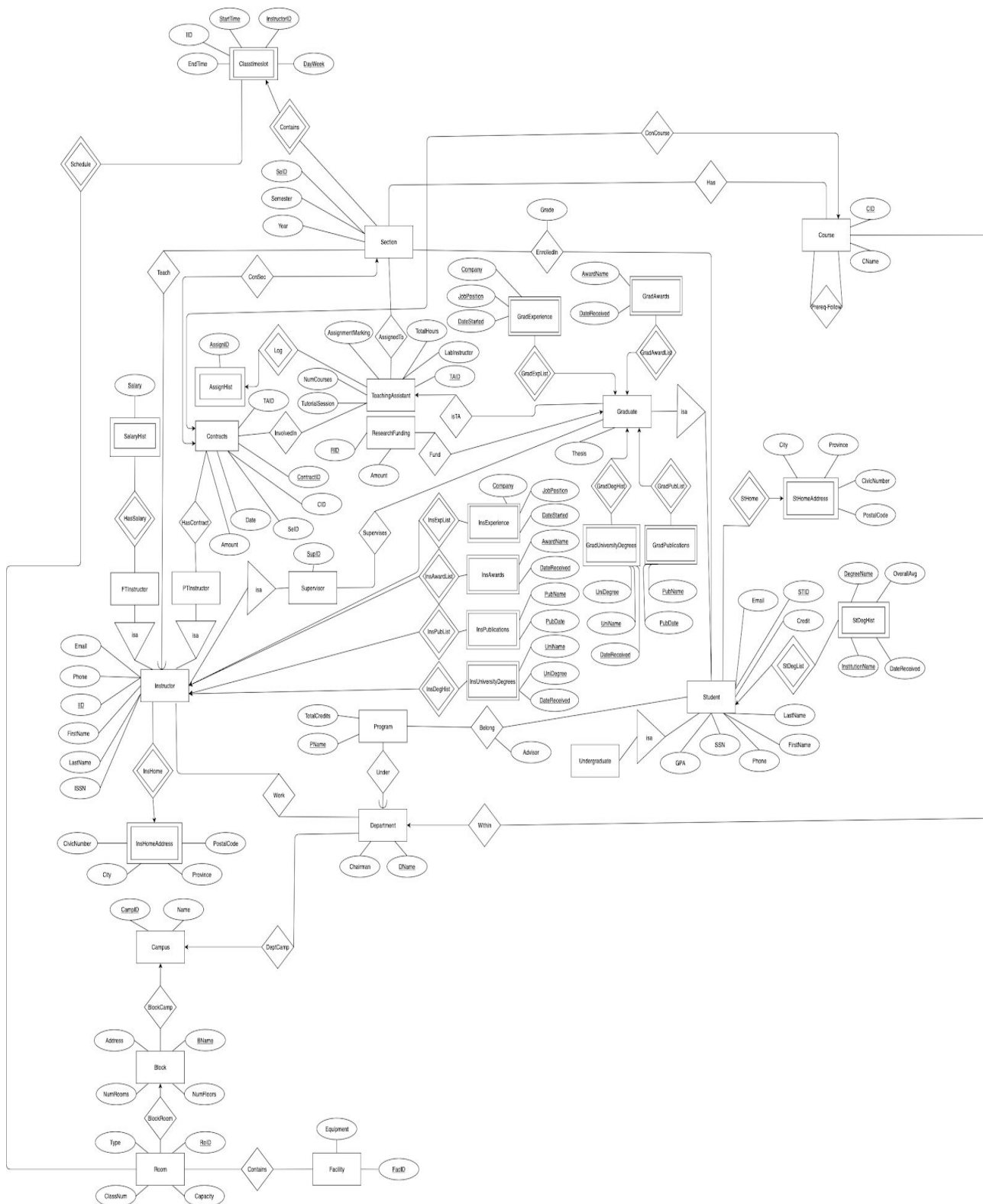


Figure 1: The Database E/R Diagram. A .JPEG of the E/R Diagram has been added for clarity in the .ZIP file.

E/R RELATION SCHEMA

```
create table AssignHist
(
    AssignID int,
    TAID int,
    primary key (AssignID, TAID),
    foreign key (TAID) references TeachingAssistant (TAID)
);
```

```
create table AssignTo
(
    SelD int,
    TAID int,
    primary key (SelD, TAID),
    foreign key (SelD) references Section (SelD),
    foreign key (TAID) references TeachingAssistant (TAID)
);
```

```
create table Belong
(
    STID int,
    PName char(30),
    Advisor char(30),
    primary key (STID, PName),
    foreign key (STID) references Student (STID),
    foreign key (PName) references Program (PName)
);
```

```
Create table Block
(
    BName char(30) primary key,
    Address varchar(30),
    NumFloors int,
    NumRooms int
);
```

```
create table BlockCamp
(
    CampID int,
    BName varchar(30) primary key,
    foreign key (BName) references Block (BName),
    foreign key (CampID) references Campus (CampID)
);
```

```

create table BlockRoom
(
    RoID int,
    BName varchar(30),
    primary key (BName, RoID),
    foreign key (BName) references Block (BName),
    foreign key (RoID) references Room (RoID)
);

```

```

create table Campus
(
    CampID Int Primary Key,
    Name char(30)
);

```

```

create table ClassTimeslot
(
    StartTime time,
    DayWeek varchar(30),
    SeID int,
    EndTime time,
    RoID int,
    primary key (StartTime, DayWeek, SeID, RoID),
    foreign key (RoID) references Room (RoID),
    foreign key (SeID) references Section (SeID)
);

```

```

create table Contains
(
    facID int,
    RoID int,
    primary key (facID, RoID),
    foreign key (facID) references Facility (facID)
    foreign key (RoID) references Room (RoID)
);

```

```

create table Contracts
(
    CID int,
    Date date,
    Amount float,
    SeID int,
    TAID int,
    ContractID int primary key,
    foreign key (CID) references Course (CID),
    foreign key (SeID) references Section (SeID)
);

```

```

create table Course
(
    CID int primary key,
    CName char(30)
);

create table Department
(
    DName char(30) primary key,
    Chairman char(30)
);

create table DeptCamp
(
    CampID int,
    DName varchar(30) primary key,
    foreign key (CampID) references Campus (CampID),
    foreign key (DName) references Department (DName)
);

create table EnrolledIn
(
    STID int,
    SeID int,
    Grade char(2),
    primary key (STID, SeID),
    foreign key (STID) references Student (STID),
    foreign key (SeID) references Section (SeID)
);

create table Facility
(
    equipment varchar(30),
    facID INT Primary key,
);

create table Fund
(
    STID int,
    RID int primary key,
    foreign key (STID) references Graduate (STID),
    foreign key (RID) references ResearchFunding (RID)
);

```

```

create table FTInstructor
(
    IID int primary key,
    foreign key (IID) references Instructor (IID)
);

create table GradAwards
(
    AwardName char(30),
    DateReceived date,
    STID int,
    primary key (AwardName, DateReceived, STID),
    foreign key (STID) references Graduate (STID)
);

create table GradExperience
(
    JobPosition char(30),
    STID int,
    DateStarted date,
    Company varchar(30),
    primary key (JobPosition, Company, DateStarted, STID),
    foreign key (STID) references Graduate (STID)
);

create table GradPublications
(
    PubName char(30),
    PubDate date,
    STID int,
    primary key (STID, PubName, PubDate),
    foreign key (STID) references Graduate (STID)
);

create table Graduate
(
    STID int primary key,
    thesis int,
    foreign key (STID) references Student (STID)
);

```

```

create table GradUniversityDegrees
(
    UniDegree char(30),
    UniName char(30),
    DateReceived date,
    STID int,
    primary key (UniDegree, STID, UniName, DateReceived),
    foreign key (STID) references Graduate (STID)
);

create table Has
(
    CID int,
    SeID int primary key,
    foreign key (CID) references Course (CID),
    foreign key (SeID) references Section (SeID)
);

create table HasContract
(
    IID int,
    ContractID int,
    primary key (IID, ContractID),
    foreign key (ContractID) references Contracts (ContractID),
    foreign key (IID) references PTInstructor (IID)
);

create table InsAwards
(
    AwardName char(30),
    DateReceived date,
    IID int,
    primary key (AwardName, DateReceived, IID),
    foreign key (IID) references Instructor (IID)
);

create table InsExperience
(
    JobPosition varchar(30),
    DateStarted date,
    Company varchar(30),
    IID int,
    primary key (JobPosition, DateStarted, Company, IID),
    foreign key (IID) references Instructor (IID)
);

```



```

create table InsHomeAddress
(
    City char(30),
    Province char(30),
    CivicNumber int,
    PostalCode int,
    IID int primary key,
    foreign key (IID) references Instructor (IID)
);

create table InsPublications
(
    PubName varchar(30),
    PubDate date,
    IID int,
    primary key (PubName, PubDate, IID),
    foreign key (IID) references Instructor (IID)
);

create table Instructor
(
    IID int primary key,
    ISSN int,
    Phone varchar(30),
    FirstName varchar(30),
    SupID int,
    Email varchar(30),
    LastName varchar(30),
    foreign key (SupID) references Supervisor (SupID)
);

create table IsTA
(
    STID int primary key,
    TAID int,
    foreign key (STID) references Graduate (STID)
    foreign key (TAID) references TeachingAssistant (TAID)
);

create table InsUniversityDegrees
(
    UniName varchar(30),
    UniDegree varchar(30),
    IID int,
    DateReceived date,
    primary key (UniName, UniDegree, DateReceived, IID),
    foreign key (IID) references Instructor (IID)
);

```

```
create table InvolvedIn
(
    ContractID int,
    TAID int,
    primary key (ContractID, TAID),
    foreign key (ContractID) references Contracts (ContractID)
    foreign key (TAID) references TeachingAssistant (TAID)
);
```

```
create table `PrereqFollow`
(
    CID1 int,
    CID2 int,
    primary key (CID1, CID2),
    foreign key (CID1) references Course (CID),
    foreign key (CID2) references Course (CID)
);
```

```
create table Program
(
    PName char(30) primary key,
    TotalCredits int
);
```

```
create table PTInstructor
(
    IID int primary key
);
```

```
create table ResearchFunding
(
    RID int primary key,
    Amount int
);
```

```
create table Room
(
    RoID int primary key,
    ClassNum int,
    Capacity int,
    Type varchar(30)
);
```

```

create table SalaryHist
(
    IID int primary key,
    Salary float primary key,
    foreign key (IID) references FTInstructor (IID)
);

create table Section
(
    SelD int primary key,
    Semester char(30),
    Year int,
    foreign key (instructorID) references Instructor (IID)
);

create table StDegHist
(
    DegreeName char(30),
    OverallAvg float,
    InstitutionName char(30),
    DateReceived date,
    STID int,
    primary key (DegreeName, InstitutionName, STID),
    foreign key (STID) references Student (STID)
);

create table StHomeAddress
(
    City char(30),
    Province char(30),
    CivicNumber int,
    PostalCode int,
    STID int primary key,
    foreign key (STID) references Student (STID)
);

create table Student
(
    STID int primary key,
    Credit int,
    FirstName char(30),
    LastName char(30),
    GPA float,
    SSN int,
    Phone varchar(30),
    Email varchar(30)
);

```

```

create table Supervises
(
    SupID int,
    STID int,
    primary key (STID, SupID),
    foreign key (STID) references Graduate (STID)
    foreign key (SupID) references Supervisor (SupID)
);

```

```

create table Supervisor
(
    SupID int primary key
);

```

```

create table Teach
(
    SelD int default,
    IID int default,
    primary key (IID, SelD),
    foreign key (IID) references Instructor (IID)
    foreign key (SelD) references Section (SelD)
);

```

```

create table TeachingAssistant
(
    TAID int primary key,
    TotalHours int,
    AssignmentMarking char,
    LabInstructor char,
    NumCourses int,
    TutorialSession char
);

```

```

create table Under
(
    DName char(30),
    PName char(30) primary key,
    foreign key (DName) references Department (DName),
    foreign key (PName) references Program (PName)
);

```

```

create table Undergraduate
(
    STID int primary key,
    foreign key (STID) references Student (STID)
);

```

```
create table Within
(
    CID int primary key,
    DName char(30),
    foreign key (DName) references Department (DName),
    foreign key (CID) references Course (CID)
);
```

```
create table Work
(
    DName varchar(30),
    IID int,
    primary key (DName, IID),
    foreign key (DName) references Department (DName),
    foreign key (IID) references Instructor (IID)
);
```

QUERIES

1)

```
INSERT INTO Instructor
VALUES(33, 100, '41866446637', 'Fabien', 33, 'Fabien@Junde.com', 'Junde');
```

Output:

New tuple created:

33	100	41866446637	Fabien	33	fabien@Junde.com	Junde
----	-----	-------------	--------	----	------------------	-------

```
DELETE FROM Instructor
WHERE IID = 33;
```

Output:

Tuple deleted.

```
UPDATE Instructor
SET LastName = 'Funs'
WHERE IID = 33;
```

Output:

From creating, we get:

33	100	41866446637	Fabien	33	fabien@Junde.com	Funs
----	-----	-------------	--------	----	------------------	------

```
SELECT *
FROM Instructor;
```

Output:

```
mysql> mysql> select * from Instructor;
```

IID	ISSN	Phone	FirstName	SupID	Email	LastName
1	901000	5141220000	Lisa	NULL	lisa@cranterson.com	Cranterson
2	901001	5141220001	Ernest	2	ernest@steig.com	Steig
3	901002	5141220002	Melissa	NULL	melissa@roberts.com	Roberts
4	901003	5141220003	Jake	5	jake@ralph.com	Ralph
5	901004	5141220004	Claire	1	Claire@devons.com	Devons
6	901005	5141220005	Jeremy	NULL	jeremy@kudo.com	Kudo
7	901006	5141220006	Linda	3	linda@torrents.com	Torrents
8	901007	5141220007	Jessica	8	Jessica@stevenson.com	Stevenson
9	901008	5141220008	Issac	5	Isaac@harrison.com	Harrison
10	901009	5141220009	Nicholas	7	Nicholas@larsen.com	Larsen
11	901010	5141220010	Christiano	NULL	christiano@ronaldo.com	Ronaldo
12	901011	4184559000	Bob	1	bob@saggette.com	Sagette
13	9010112	4184559001	Kevin	1	kevin@luu.com	Luu
14	9010113	4184559002	Kevin	8	kevin@michel.com	Michel
15	9010114	4380093300	Davidoid	NULL	davidoid@junky.com	Junky
16	9010115	4380093301	Lola	NULL	lola@beefo.com	Beefo
17	9010116	4380093302	George	3	Georgea@mooser.com	Mooser
18	9010117	4380093303	Eaton	6	Eaton@joe.com	Joe
19	9010118	4380093304	Chem	6	Chem@xenon.com	Xenon
20	9010119	4380093305	Xavier	6	xavier@naura.com	Naura
21	9010120	4380093306	Natasha	NULL	natasha@britney.com	Britney

21 rows in set (0.00 sec)

2)

```
INSERT INTO Student
VALUES(56, 90, 'Liam', 'Beaulieu', 2.6, 120, '4185602233', 'Liam@Beaulieu.com');
```

Output:

56	90	Liam	Beaulieu	2.6	120	4185602233	Liam@Beaulieu.com
----	----	------	----------	-----	-----	------------	-------------------

```
DELETE FROM Student
WHERE STID = 56;
```

Output:
Tuple deleted.

```
UPDATE Student
SET FirstName = 'David'
WHERE STID = 56;
```

Output:

56	90	David	Beaulieu	2.6	120	4185602233	Liam@Beaulieu.com
----	----	-------	----------	-----	-----	------------	-------------------

```
SELECT *
FROM Student;
```

Output:

```
mysql> select * from Student;
```

STID	Credit	FirstName	LastName	GPA	SSN	Phone	Email
1	90	Felix	Harris	3.1	453	5144443450	felix@harrix.com
2	90	James	Watson	2	9536	5144443451	james@watson.com
3	90	Marcus	Morris	2.7	86234	5144443452	marcus@morris.com
4	90	Johnny	Howard	3.4	34986	5144443453	johnny@howard.com
5	90	John	Smith	3.1	393	5144443454	john@smith.com
6	44	Phil	Newton	2.1	9856	5144443455	phil@newton.com
7	44	Andrew	Morrison	2	25806	5144443456	andrew@morrison.com
8	44	Mackenzie	Johnson	3.2	259	5144443457	mackenzie@johnson.com
9	44	Max	Phillips	3	5274	5144443458	max@phillips.com
10	44	Linus	Torvards	4	2494	5144443459	linus@torvards
11	44	Khaled	Jababo	4	348	5144443460	khaled@jababo.com
12	44	Aiman	Hanna	4	7238734	5144443461	aiman@hanna.com
13	44	Messi	Lionel	1	1283	5144443462	messi@lionel.com
14	44	Luis	Loni	3	19	5144443463	luis@loni.com
15	44	Goun	Lio	2	18	5144443464	goun@lio.com
16	44	Harves	Ri	2	171	5144443465	harves@ri.com
17	44	Jonathan	Izzmifirtre	3.9	16	5144443466	jonathan@izzmifirtre.com
18	44	Nicolas	Tabourette	3.8	5744	5144443467	nicolas@tabourette.com
19	44	Nathan	Mackinnon	3.8	54272	5144443468	nathan@mackinnon.com
20	44	Sydney	Crosby	4	2576	5144443469	sydney@crosby.com
21	44	David	Izzmifartre	3.9	572	5144443470	david@izzmifartre.com
22	44	Poche	Jigyuan	4	647	5144443471	poche@jigyuan.com
23	44	Larry	Simpson	2.7	62	5144443472	larry@simpson.com
24	44	Foki	Ruki	2	56	5144443473	foki@ruki.com
25	44	Smity	Sam	4	6253	5144443474	smity@sam.com
26	44	Rawl	Sol	3	236	5144443475	rawl@sol.com
27	44	Sara	Flore	4	6	5144443476	sara@flore.com
28	90	Joey	Looker	3.2	42	5144443477	joey@looker.com
29	90	Lucienne	Bouchard	1.5	124	5144443478	lucienne@bouchard.com
30	90	Esmeralda	Bobo	2.7	76474	5144443479	esmeralda@bobo.com
31	90	Laurent	Voyer	2.8	5633	5144443480	laurent@voyer.com
32	90	Eddey	Boucher	3	63	5144443481	eddey@boucher.com

32 rows in set (0.00 sec)

3)

```
INSERT INTO TeachingAssistant  
VALUES(45, 200, 'y', 'n', 2,'y');
```

Output:

45	200	y	n	2	y
----	-----	---	---	---	---

```
DELETE FROM TeachingAssistant  
WHERE TAID = 45;
```

Output:

Tuple deleted.

```
UPDATE TeachingAssistant  
SET TotalHours = 259  
WHERE TAID = 45;
```

Output:

45	259	y	n	2	y
----	-----	---	---	---	---

```
SELECT S.FirstName, S.LastName, S.Email, S.Credit, S.GPA  
FROM Student S INNER JOIN IsTA TA on TA.STID = S.STID  
WHERE S.GPA > 3.2;
```

Output:

```
mysql> SELECT S.FirstName, S.LastName, S.Email, S.Credit, S.GPA  
-> FROM Student S INNER JOIN IsTA TA on TA.STID = S.STID  
-> WHERE S.GPA > 3.2;
```

FirstName	LastName	Email	Credit	GPA
Smity	Sam	smity@sam.com	44	4
Mackenzie	Johnson	mackenzie@johnson.com	44	3.2
Linus	Torvards	linus@torvards	44	4
Poche	Jigyuan	poche@jigyuan.com	44	4
Khaled	Jababo	khaled@jababo.com	44	4
Sydney	Crosby	sydney@crosby.com	44	4
Aiman	Hanna	aiman@hanna.com	44	4
Nathan	Mackinnon	nathan@mackinnon.com	44	3.8
David	Izzmifartre	david@izzmifartre.com	44	3.9
Nicolas	Tabourette	nicolas@tabourette.com	44	3.8
Jonathan	Izzmifirtre	jonathan@izzmifirtre.com	44	3.9
Sara	Flore	sara@flore.com	44	4

12 rows in set (0.00 sec)

4)

```
SELECT distinct Name  
FROM Campus  
GROUP BY Name;
```

Output:

```
mysql> SELECT distinct Name  
-> FROM Campus  
-> GROUP BY Name;  
+-----+  
| Name |  
+-----+  
| Billie Holiday Campus |  
| EastWest Campus |  
| Germane Campus |  
| Hill Campus |  
| King James Campus |  
| Loyola Campus |  
| Oxford Campus |  
| St George Campus |  
| West Campus |  
| Zimmerman Campus |  
+-----+  
10 rows in set (0.00 sec)
```

5)

```
SELECT BName  
FROM BlockCamp  
WHERE CampID = 7;
```

Output:

```
mysql> SELECT BName  
-> FROM BlockCamp  
-> WHERE CampID = 7;  
+-----+  
| BName |  
+-----+  
| A Block |  
| B Block |  
| C Block |  
| D Block |  
| E Block |  
+-----+  
5 rows in set (0.00 sec)
```

6)

```
SELECT B.Address, B.NumFloors, B.NumRooms, R.type, CASE
  WHEN R.Type='Labroom' or R.Type='Classroom' THEN R.Capacity
  END as 'Capacity', CASE
  WHEN R.Type='Labroom' or R.Type='Classroom' THEN F.equipment
  END as 'Equipment'
FROM Block B
  INNER JOIN BlockRoom BR on B.BName=BR.BName
  INNER JOIN Room R on R.RoID=BR.RoID
  INNER JOIN Contains C on R.RoID=C.RoID
  INNER JOIN Facility F on F.facID=C.facID
WHERE B.BName = 'B Block';
```

Output:

```
mysql> SELECT B.Address, B.NumFloors, B.NumRooms, R.type, CASE  WHEN R.Type='Labroom' or R.T
.Type='Labroom' or R.Type='Classroom' THEN F.equipment  END as 'Equipment' FROM Block B  IN
.RoID=BR.RoID  INNER JOIN Contains C on R.RoID=C.RoID  INNER JOIN Facility F on F.facID=C.f
+-----+-----+-----+-----+-----+-----+
| Address          | NumFloors | NumRooms | type   | Capacity | Equipment          |
+-----+-----+-----+-----+-----+-----+
| 235 westSide street | 3         | 23       | Labroom | 130      | auditorium         |
| 235 westSide street | 3         | 23       | Labroom | 130      | microphone         |
| 235 westSide street | 3         | 23       | Labroom | 130      | overhead projector |
| 235 westSide street | 3         | 23       | Labroom | 45       | projector          |
| 235 westSide street | 3         | 23       | Labroom | 45       | air conditioning   |
| 235 westSide street | 3         | 23       | Labroom | 45       | black board        |
| 235 westSide street | 3         | 23       | classroom | 122     | computers          |
| 235 westSide street | 3         | 23       | classroom | 122     | overhead projector |
| 235 westSide street | 3         | 23       | classroom | 122     | black board        |
| 235 westSide street | 3         | 23       | classroom | 122     | auditorium         |
| 235 westSide street | 3         | 23       | classroom | 122     | microphone         |
| 235 westSide street | 3         | 23       | classroom | 122     | speakers           |
| 235 westSide street | 3         | 23       | classroom | 123     | auditorium         |
| 235 westSide street | 3         | 23       | classroom | 123     | microphone         |
| 235 westSide street | 3         | 23       | classroom | 123     | speakers           |
| 235 westSide street | 3         | 23       | classroom | 124     | projector          |
| 235 westSide street | 3         | 23       | classroom | 124     | air conditioning   |
| 235 westSide street | 3         | 23       | classroom | 124     | tablet             |
| 235 westSide street | 3         | 23       | classroom | 125     | projector          |
| 235 westSide street | 3         | 23       | classroom | 125     | air conditioning   |
| 235 westSide street | 3         | 23       | classroom | 125     | tablet             |
+-----+-----+-----+-----+-----+-----+
21 rows in set (0.00 sec)
```

7)

```
SELECT P.Pname, P.TotalCredits
FROM Program P
  INNER JOIN Under U on P.PName = U.PName
  INNER JOIN Department D on D.DName=U.DName
WHERE D.DName='Judaism';
```

Output:

Pname	TotalCredits
Business	22
Economics	6
English	12
Law	14
Religion	10
Rule the World	30
Understanding Money	90

8)

```
SELECT C.Cname, P.Pname
FROM Section S
  INNER JOIN Has H on S.SeID = H.SeID
  INNER JOIN Course C on C.CID = H.CID
  INNER JOIN Within W on W.CID = C.CID
  INNER JOIN Department D on W.Dname = D.Dname
  INNER JOIN Under U on U.Dname = D.Dname
  INNER JOIN Program P on P.Pname = U.Pname
WHERE S.Semester='Fall' AND P.Pname = 'Canadian History'
GROUP BY P.Pname;
```

Output:

Cname	Pname
Hist300	Canadian History

1 row in set (0.00 sec)

9)

```
SELECT distinct C.Cname,T.IID ,I.FirstName,I.LastName, S.SeID, S.Year,
S.Semester, CT.DayWeek, CT.StartTime, CT.EndTime,CT.RoID, R.Capacity,
B.Address
FROM Instructor I
INNER JOIN Teach T on I.IID = T.IID
INNER JOIN Section S on T.SeID = S.SeID
INNER JOIN Has H on S.SeID = H.SeID
INNER JOIN Course C on H.CID = C.CID
INNER JOIN Within W on C.CID = W.CID
INNER JOIN Department D on D.Dname = W.Dname
INNER JOIN DeptCamp DC on DC.Dname = D.Dname
INNER JOIN ClassTimeslot CT on S.SeID = CT.SeID
INNER JOIN Room R on R.RoID=CT.RoID
INNER JOIN BlockRoom on R.RoID = BlockRoom.RoID
INNER JOIN Block B on BlockRoom.BName = B.BName
WHERE D.Dname = 'Physics' AND S.Semester = 'Winter';
```

Output:

Cname	IID	FirstName	LastName	SeID	Year	Semester	DayWeek	StartTime	EndTime	RoID	Capacity	Address
MECH300	1	Lisa	Cranterson	2	2020	Winter	Wednesday	09:30:00	10:45:00	2	130	235 westSide street

1 row in set (0.01 sec)

10)

```
SELECT DISTINCT S.STID, S.FirstName, S.LastName
FROM Section SC
INNER JOIN EnrolledIn E on SC.SeID = E.SeID
INNER JOIN Student S on E.STID = S.STID
INNER JOIN Belong B on B.STID = S.STID
INNER JOIN Program P on P.PName = B.PName
WHERE B.Pname='Arts' AND Semester='Summer';
```

Output:

STID	FirstName	LastName
1	Felix	Harris
2	James	Watson
14	Luis	Loni
20	Sydney	Crosby
31	Laurent	Voyer

11)

```
SELECT I.FirstName, I.LastName
FROM Course C
  INNER JOIN Has H on H.CID = C.CID
  INNER JOIN Section S on H.SeID = S.SeID
  INNER JOIN Teach T on T.SeID = S.SeID
  INNER JOIN Instructor I on T.IID = I.IID
WHERE C.CName = 'COMP249' AND S.Semester = 'Fall';
```

Output:

FirstName	LastName
Eaton	Joe
Chem	Xenon
Jessica	Stevenson
Issac	Harrison
Nicholas	Larsen

12)

```
SELECT I.FirstName, I.LastName
FROM Instructor I
  INNER JOIN Work W on I.IID = W.IID
  INNER JOIN Department D on D.DName = W.DName
Where D.DName='Physics' AND SupID IS NOT NULL
GROUP BY I.FirstName;
```

Output:

FirstName	LastName
Chem	Xenon
Eaton	Joe
George	Mooser
Nicholas	Larsen
Xavier	Naura

13)

```
SELECT B.Advisor
FROM Belong B
  INNER JOIN Program P on P.PName = B.PName
  INNER JOIN Under U on U.PName = P.PName
  INNER JOIN Department D on U.DName = D.DName
WHERE D.DName = 'Anthropology'
GROUP BY B.Advisor;
```

Output:

```
+-----+
| Advisor |
+-----+
| Elma Aveiro |
| Jack |
| Katia Aveiro |
| Marc Luver |
| Marc Zuter |
+-----+
5 rows in set (0.00 sec)
```

14)

```
SELECT ST.FirstName, ST.LastName, ST.STID
FROM Student ST
  INNER JOIN Graduate G on ST.STID = G.STID
  INNER JOIN Supervises SP on G.STID = SP.STID
  INNER JOIN Supervisor S on SP.SupID = S.SupID
  INNER JOIN Instructor I on I.IID= S.SupID
WHERE I.IID = 1;
```

Output:

```
+-----+-----+-----+
| FirstName | LastName | STID |
+-----+-----+-----+
| Phil      | Newton   | 6     |
| Khaled    | Jababo   | 11    |
| Aiman     | Hanna    | 12    |
| Luis      | Loni     | 14    |
| Smity     | Sam      | 25    |
| Rawl      | Sol      | 26    |
| Sara      | Flore    | 27    |
+-----+-----+-----+
```


15)

```
SELECT Student.STID, Student.FirstName, Student.LastName,
TeachingAssistant.AssignmentMarking
FROM Student
INNER JOIN Graduate on Student.STID = Graduate.STID
INNER JOIN IsTA on IsTA.STID=Graduate.STID
INNER JOIN TeachingAssistant on TeachingAssistant.TAID=IsTA.TAID
INNER JOIN AssignTo on AssignTo.TAID=TeachingAssistant.TAID
INNER JOIN Section S on AssignTo.SeID = S.SeID
INNER JOIN Has H on S.SeID = H.SeID
INNER JOIN Course C on H.CID = C.CID
WHERE C.CID='1' and Semester='Summer';
```

Output:

STID	FirstName	LastName	AssignmentMarking
13	Messi	Lionel	n
16	Harves	Ri	n
14	Luis	Loni	y
27	Sara	Flore	y
15	Goun	Lio	n

16)

```
SELECT S.STID, S.FirstName, S.LastName, SUM(RF.Amount)
FROM Graduate G , Student S, Fund F, ResearchFunding RF
WHERE S.STID=G.STID AND G.STID = F.STID AND F.RID = RF.RID
GROUP BY F.STID;
```

Output:

STID	FirstName	LastName	SUM(RF.Amount)
8	Mackenzie	Johnson	11000
10	Linus	Torvards	8000
14	Luis	Loni	200
15	Goun	Lio	100
16	Harves	Ri	6000
17	Jonathan	Izzmifirtre	3000
18	Nicolas	Tabourette	1500
19	Nathan	Mackinnon	100
20	Sydney	Crosby	20000
21	David	Izzmifartre	1000
22	Poche	Jigyuan	100
23	Larry	Simpson	200
24	Foki	Ruki	300
25	Smity	Sam	100
26	Rawl	Sol	500
27	Sara	Flore	2000

17)

```
SELECT D.DName, D.Chairman, Count(*)
FROM Department D, Within W
WHERE D.DName=W.DName
GROUP BY W.DName;
```

Output:

DName	Chairman	Count(*)
Biology	Juan	4
Computer Science	Ryan	6
Engineering	Max	2
French	Suzy	4
Gender Studies	Loretta	4
History	Juju	5
Physics	Greg	2

18)

```
SELECT P.PName, Count(*)
FROM Program P, Under U
WHERE P.PName=U.PName
GROUP BY U.Dname;
```

Output:

PName	Count(*)
Arts	2
Computer Science	1
Computer Architecture	3
Canadian History	1
Business	7
Mathematics	1
Rockets	1

19)

```
SELECT C.CName
FROM Course C
INNER JOIN Has H ON C.CID = H.CID
INNER JOIN Section Se ON Se.SeID = H.SeID
INNER JOIN EnrolledIn E ON Se.SeID = E.SeID
INNER JOIN Student S ON S.STID = E.STID
WHERE S.STID = 2 AND Se.Semester = 'Summer';
```

Output:

CName
COMP218
FR100
COMP108
MECH300
FR400
BI0100

20)

First, we get the section ID we want the student to be registered in (39 is the output):

```
SELECT Se.SeID
FROM Course C
  INNER JOIN Has H ON C.CID = H.CID
  INNER JOIN Section Se ON Se.SeID = H.SeID
WHERE C.CName = 'COMP249' AND Se.Semester = 'Winter' AND Se.Year = 2020;
```

Then, we get the student ID with a specific SSN (1 is the output):

```
SELECT St.STID
FROM Student St
WHERE St.SSN = '453';
```

Finally, we insert the above section ID and student ID into table EnrolledIn:

```
INSERT INTO EnrolledIn
VALUES
  (1, 39, NULL);
```

Output:

1	39	NULL
---	----	------

21)

First, we get the student ID (1 is the output):

```
SELECT St.STID
FROM Student St
WHERE St.SSN = '453';
```

Then, we get the section ID (32 is the output):

```
SELECT Se.SeID
FROM Course C
INNER JOIN Has H ON C.CID = H.CID
INNER JOIN Section Se ON Se.SeID = H.SeID
WHERE C.CName = 'COMP248' AND Se.Semester = 'Fall' AND Se.Year = 2020;
```

```
DELETE FROM EnrolledIn
WHERE STID = 'St.STID' AND SeID = 'Se.SeID';
```

Finally, we drop the student with section ID output and student ID output:

```
DELETE FROM EnrolledIn
WHERE STID = 1 AND SeID = 32
```

Output:
Tuple deleted.

22)

```
SELECT SHA.City, SHA.Province, SHA.CivicNumber, SHA.PostalCode,
SDH.DegreeName, SDH.OverallAvg, SDH.InstitutionName, SDH.DateReceived,
E.Grade, C.CName
FROM StHomeAddress SHA, StDegHist SDH, Student S
INNER JOIN EnrolledIn E ON S.STID = E.STID
INNER JOIN Section Se ON Se.SeID = E.SeID
INNER JOIN Has H ON Se.SeID = H.SeID
INNER JOIN Course C ON C.CID = H.CID
WHERE S.STID = 7 AND SHA.STID = 7 AND SDH.STID = 7
GROUP BY S.STID;
```

Output:

City	Province	CivicNumber	PostalCode	DegreeName	OverallAvg	InstitutionName	DateReceived	Grade	CName
Montreal	QC	1456	T3T 8H8	Finance	90	JMSB	2018-05-31	D+	BI0100

Functional Dependencies

Weak entity AssignHist:

$(\text{AssignID}, \text{TAID}) \rightarrow \{\text{AssignID}, \text{TAID}\}$

Relationship AssignTo:

$(\text{SeID}, \text{TAID}) \rightarrow \{\text{SeID}, \text{TAID}\}$

Relationship Belong:

$(\text{STID}, \text{PName}) \rightarrow \{\text{STID}, \text{PName}, \text{Advisor}\}$

Strong entity Block:

$(\text{BName}) \rightarrow \{\text{BName}, \text{Address}, \text{NumFloors}, \text{NumRooms}\}$

Relationship BlockCamp:

$(\text{BName}) \rightarrow \{\text{BName}, \text{CampID}\}$

Relationship BlockRoom:

$(\text{BName}, \text{RoID}) \rightarrow \{\text{BName}, \text{RoID}\}$

Strong entity Campus:

$(\text{CampID}) \rightarrow \{\text{CampID}, \text{Name}\}$

Weak entity ClassTimeslot:

$(\text{StartTime}, \text{DayWeek}, \text{SeID}, \text{RoID}) \rightarrow \{\text{StartTime}, \text{DayWeek}, \text{SeID}, \text{RoID}, \text{EndTime}\}$

Relationship Contains:

$(\text{facID}, \text{RoID}) \rightarrow \{\text{facID}, \text{RoID}\}$

Strong entity Contracts:

$(\text{ContractID}) \rightarrow \{\text{ContractID}, \text{CID}, \text{SeID}, \text{TAID}, \text{Date}, \text{Amount}\}$

Strong entity Course:

$(\text{CID}) \rightarrow \{\text{CID}, \text{CName}\}$

Strong entity Department:

$(\text{DName}) \rightarrow \{\text{DName}, \text{Chairman}\}$

Relationship DeptCamp:

$(\text{DName}) \rightarrow \{\text{DName}, \text{CampID}\}$

Relationship EnrolledIn:

$(STID, SeID) \rightarrow \{STID, SeID, Grade\}$

Strong entity Facility:

$(facID) \rightarrow \{facID, equipment\}$

Relationship Fund:

$(STID) \rightarrow \{STID, RID\}$

Strong entity FTInstructor:

$(IID) \rightarrow \{IID\}$

Weak entity GradAwards:

$(AwardName, DateReceived, STID) \rightarrow \{AwardName, DateReceived, STID\}$

Weak entity GradExperience:

$(JobPosition, Company, DateStarted, STID) \rightarrow \{JobPosition, Company, DateStarted, STID\}$

Weak entity GradPublications:

$(STID, PubName, PubDate) \rightarrow \{STID, PubName, PubDate\}$

Strong entity Graduate:

$(STID) \rightarrow \{STID, thesis\}$

Weak entity GradUniversityDegrees:

$(UniDegree, STID, UniName, DateReceived) \rightarrow \{UniDegree, STID, UniName, DateReceived\}$

Relationship Has:

$(SeID) \rightarrow \{SeID, CID\}$

Relationship HasContract:

$(IID, ContractID) \rightarrow \{IID, ContractID\}$

Weak entity InsAwards:

$(AwardName, DateReceived, IID) \rightarrow \{AwardName, DateReceived, IID\}$

Weak entity InsExperience:

$(JobPosition, DateStarted, Company, IID) \rightarrow \{JobPosition, DateStarted, Company, IID\}$

Weak entity InsHomeAddress:

$(IID) \rightarrow \{IID, City, Province, CivicNumber, PostalCode\}$

Weak entity InsPublications:

(PubName, PubDate, IID) \rightarrow {PubName, PubDate, IID}

Strong entity Instructor:

(IID) \rightarrow {IID, ISSN, Phone, FirstName, SupID, Email, LastName}

Weak entity InsUniversityDegrees:

(UniName, UniDegree, DateReceived, IID) \rightarrow {UniName, UniDegree, DateReceived, IID}

Relationship IsTA:

(STID) \rightarrow {STID, TAID}

Relationship InvolvedIn:

(ContractID, TAID) \rightarrow {ContractID, TAID}

Relationship PrereqFollow:

(CID1, CID2) \rightarrow {CID1, CID2}

Strong entity Program:

(PName) \rightarrow {PName, TotalCredits}

Strong entity PTInstructor:

(IID) \rightarrow {IID}

Strong entity ResearchFunding:

(RID) \rightarrow {RID, Amount}

Strong entity Room:

(RoID) \rightarrow {RoID, ClassNum, Capacity, Type}

Weak entity SalaryHist:

(IID, Salary) \rightarrow {IID, Salary}

Strong entity Section:

(SeID) \rightarrow {SeID, Semester, Year}

Weak entity StDegHist:

(DegreeName, InstitutionName, STID) \rightarrow {DegreeName, InstitutionName, STID, OverallAvg, DateReceived}

Weak entity StHomeAddress:

(STID) \rightarrow {STID, City, Province, CivicNumber, PostalCode}

Strong entity Student:

$(STID) \rightarrow \{STID, Credit, FirstName, LastName, GPA, SSN, Phone, Email\}$

Relationship Supervises:

$(STID, SupID) \rightarrow \{STID, SupID\}$

Strong entity Supervisor:

$(SupID) \rightarrow \{SupID\}$

Relationship Teach:

$(IID, SelD) \rightarrow \{IID, SelD\}$

Strong entity TeachingAssistant:

$(TAID) \rightarrow \{TAID, TotalHours, AssignmentMarking, LabInstructor, NumCourses, TutorialSession\}$

Relationship Under:

$(PName) \rightarrow \{PName, DName\}$

Strong entity Undergraduate:

$(STID) \rightarrow \{STID\}$

Relationship Within:

$(CID) \rightarrow \{CID, DName\}$

Relationship Work:

$(DName, IID) \rightarrow \{DName, IID\}$

NORMALIZATION

3NF: LHS is a key OR RHS has key attribute

BCNF: LHS is a key

All BCNF are in 3NF.

Weak entity AssignHist:

$(\text{AssignID}, \text{TAID}) \rightarrow \{\text{AssignID}, \text{TAID}\}$

=> Trivial. BCNF.

Relationship AssignTo:

$(\text{SeID}, \text{TAID}) \rightarrow \{\text{SeID}, \text{TAID}\}$

=> Trivial. BCNF.

Relationship Belong:

$(\text{STID}, \text{PName}) \rightarrow \{\text{STID}, \text{PName}, \text{Advisor}\}$

=> LHS key; BCNF.

Strong entity Block:

$(\text{BName}) \rightarrow \{\text{BName}, \text{Address}, \text{NumFloors}, \text{NumRooms}\}$

=> LHS key; BCNF.

Relationship BlockCamp:

$(\text{BName}) \rightarrow \{\text{BName}, \text{CampID}\}$

=> LHS key; BCNF.

Relationship BlockRoom:

$(\text{BName}, \text{RoID}) \rightarrow \{\text{BName}, \text{RoID}\}$

=> Trivial. BCNF.

Strong entity Campus:

$(\text{CampID}) \rightarrow \{\text{CampID}, \text{Name}\}$

=> LHS key; BCNF.

Weak entity ClassTimeslot:

$(\text{StartTime}, \text{DayWeek}, \text{SeID}, \text{RoID}) \rightarrow \{\text{StartTime}, \text{DayWeek}, \text{SeID}, \text{RoID}, \text{EndTime}\}$

=> LHS key; BCNF.

Relationship Contains:

$(\text{facID}, \text{RoID}) \rightarrow \{\text{facID}, \text{RoID}\}$

=> Trivial. BCNF.

Strong entity Contracts:

$(\text{ContractID}) \rightarrow \{\text{ContractID}, \text{CID}, \text{SeID}, \text{TAID}, \text{Date}, \text{Amount}\}$

=> LHS key; BCNF.

Strong entity Course:

$(\text{CID}) \rightarrow \{\text{CID}, \text{CName}\}$

=> LHS key; BCNF.

Strong entity Department:

$(\text{DName}) \rightarrow \{\text{DName}, \text{Chairman}\}$

=> LHS key; BCNF.

Relationship DeptCamp:

$(\text{DName}) \rightarrow \{\text{DName}, \text{CampID}\}$

=> LHS key; BCNF.

Relationship EnrolledIn:

$(\text{STID}, \text{SeID}) \rightarrow \{\text{STID}, \text{SeID}, \text{Grade}\}$

=> LHS key; BCNF.

Strong entity Facility:

$(\text{facID}) \rightarrow \{\text{facID}, \text{equipment}\}$

=> LHS key; BCNF.

Relationship Fund:

$(\text{STID}) \rightarrow \{\text{STID}, \text{RID}\}$

=> LHS key; BCNF.

Strong entity FTInstructor:

$(\text{IID}) \rightarrow \{\text{IID}\}$

=> Trivial. BCNF.

Weak entity GradAwards:

$(\text{AwardName}, \text{DateReceived}, \text{STID}) \rightarrow \{\text{AwardName}, \text{DateReceived}, \text{STID}\}$

=> Trivial. BCNF.

Weak entity GradExperience:

$(\text{JobPosition}, \text{Company}, \text{DateStarted}, \text{STID}) \rightarrow \{\text{JobPosition}, \text{Company}, \text{DateStarted}, \text{STID}\}$

=> Trivial. BCNF.

Weak entity GradPublications:

$(\text{STID}, \text{PubName}, \text{PubDate}) \rightarrow \{\text{STID}, \text{PubName}, \text{PubDate}\}$

=> Trivial. BCNF.

Strong entity Graduate:

$(\text{STID}) \rightarrow \{\text{STID}, \text{thesis}\}$

=> LHS key; BCNF.

Weak entity GradUniversityDegrees:

$(\text{UniDegree}, \text{STID}, \text{UniName}, \text{DateReceived}) \rightarrow \{\text{UniDegree}, \text{STID}, \text{UniName}, \text{DateReceived}\}$

=> Trivial. BCNF.

Relationship Has:

$(\text{SeID}) \rightarrow \{\text{SeID}, \text{CID}\}$

=> LHS key; BCNF.

Relationship HasContract:

$(\text{IID}, \text{ContractID}) \rightarrow \{\text{IID}, \text{ContractID}\}$

=> Trivial. BCNF.

Weak entity InsAwards:

$(\text{AwardName}, \text{DateReceived}, \text{IID}) \rightarrow \{\text{AwardName}, \text{DateReceived}, \text{IID}\}$

=> Trivial. BCNF.

Weak entity InsExperience:

$(\text{JobPosition}, \text{DateStarted}, \text{Company}, \text{IID}) \rightarrow \{\text{JobPosition}, \text{DateStarted}, \text{Company}, \text{IID}\}$

=> Trivial. BCNF.

Weak entity InsHomeAddress:

$(IID) \rightarrow \{IID, City, Province, CivicNumber, PostalCode\}$

=> LHS key; BCNF.

*Postal Code determines Province, City, and Civic Number. As such, InsHomeAddress could be decomposed, however for simplicity, it has been left unaltered.

Weak entity InsPublications:

$(PubName, PubDate, IID) \rightarrow \{PubName, PubDate, IID\}$

=> Trivial. BCNF.

Strong entity Instructor:

$(IID) \rightarrow \{IID, ISSN, Phone, FirstName, SupID, Email, LastName\}$

=> LHS key; BCNF. (ISSN is also a candidate key).

Weak entity InsUniversityDegrees:

$(UniName, UniDegree, DateReceived, IID) \rightarrow \{UniName, UniDegree, DateReceived, IID\}$

=> Trivial. BCNF.

Relationship IsTA:

$(STID) \rightarrow \{STID, TAID\}$

=> LHS key; BCNF.

Relationship InvolvedIn:

$(ContractID, TAID) \rightarrow \{ContractID, TAID\}$

=> Trivial. BCNF.

Relationship PrereqFollow:

$(CID1, CID2) \rightarrow \{CID1, CID2\}$

=> Trivial. BCNF.

Strong entity Program:

$(PName) \rightarrow \{PName, TotalCredits\}$

=> LHS key; BCNF.

Strong entity PTInstructor:

$(IID) \rightarrow \{IID\}$

=> Trivial. BCNF.

Strong entity ResearchFunding:

$(RID) \rightarrow \{RID, Amount\}$

=> LHS key; BCNF.

Strong entity Room:

$(RoID) \rightarrow \{RoID, ClassNum, Capacity, Type\}$

=> LHS key; BCNF.

Weak entity SalaryHist:

$(IID, Salary) \rightarrow \{IID, Salary\}$

=> Trivial. BCNF.

Strong entity Section:

$(SeID) \rightarrow \{SeID, Semester, Year\}$

=> LHS key; BCNF.

Weak entity StDegHist:

$(DegreeName, InstitutionName, STID) \rightarrow \{DegreeName, InstitutionName, STID, OverallAvg, DateReceived\}$

=> Trivial. BCNF.

Weak entity StHomeAddress:

$(STID) \rightarrow \{STID, City, Province, CivicNumber, PostalCode\}$

=> LHS key; BCNF.

*Postal Code determines Province, City, and Civic Number. As such, InsHomeAddress could be decomposed, however for simplicity, it has been left unaltered.

Strong entity Student:

$(STID) \rightarrow \{STID, Credit, FirstName, LastName, GPA, SSN, Phone, Email\}$

=> LHS key; BCNF. (SSN also a candidate key).

Relationship Supervises:

$(STID, SupID) \rightarrow \{STID, SupID\}$

=> Trivial. BCNF.

Strong entity Supervisor:

$(SupID) \rightarrow \{SupID\}$

=> Trivial. BCNF.

Relationship Teach:

$(IID, SeID) \rightarrow \{IID, SeID\}$

=> Trivial. BCNF.

Strong entity TeachingAssistant:

$(TAID) \rightarrow \{TAID, TotalHours, AssignmentMarking, LabInstructor, NumCourses, TutorialSession\}$

=> LHS key; BCNF.

Relationship Under:

$(PName) \rightarrow \{PName, DName\}$

=> LHS key; BCNF.

Strong entity Undergraduate:

$(STID) \rightarrow \{STID\}$

=> Trivial. BCNF.

Relationship Within:

$(CID) \rightarrow \{CID, DName\}$

=> LHS key; BCNF.

Relationship Work:

$(DName, IID) \rightarrow \{DName, IID\}$

=> Trivial. BCNF.

FUNCTIONALITIES IMPLEMENTED

We have implemented all the queries from number I to XXII on our UI located at <http://krc353.encs.concordia.ca> as well as 3 additional queries:

1. Check prerequisite(s) of a class
2. Check follow-up(s) of a class
3. Check the FD of Course ID $\rightarrow \{\text{CourseID}, \text{CourseName}\}$

ADDITIONAL FEATURES

- Before-Insert Trigger to check for ClassTimeslot conflict:

```
CREATE TRIGGER test_timeslot_before_insert
```

```
BEFORE INSERT ON ClassTimeslot FOR EACH ROW
```

```
BEGIN
```

```
IF (Select COUNT(*)
```

```
from ClassTimeslot
```

```
where (NEW.DayWeek, SeID) IN (Select DayWeek, ClassTimeslot.SeID
```

```
from (Select SeID
```

```
from Section
```

```
where (semester,year) IN (Select semester,year
```

```
from Section
```

```
where Section.SeID=NEW.SeID)
```

```
AND Section.SeID<>NEW.SeID) A
```

```
inner join ClassTimeslot
```

```
on A.SeID=ClassTimeslot.SeID)
```

```
AND ((NEW.StartTime >= StartTime AND NEW.StartTime <= EndTime AND NEW.RoID= RoID) OR  
(NEW.StartTime >= StartTime AND NEW.StartTime <= EndTime AND NEW.IID=IID))) > 0
```

```
THEN SIGNAL SQLSTATE '12345' SET MESSAGE_TEXT = 'conflicting timeslot' ;
```

```
END IF;
```

```
END
```

- Prerequisite Queries:

```
SELECT C.Cname
FROM Course C
WHERE CID IN (Select P.CID1
              FROM Course C, PrereqFollow P
              WHERE Cname='COMP249' AND C.CID=P.CID2);
```

Output:
COMP248

```
SELECT C.Cname
FROM Course C
WHERE CID IN (Select P.CID2
              FROM Course C, PrereqFollow P
              WHERE Cname='COMP248' AND C.CID=P.CID1);
```

Output:
COMP249

- FD Check Query (if outputs an empty set, it means that the FD holds)
For FD $CID \rightarrow \{CID, CName\}$
SELECT C.CID
FROM Course C
GROUP BY C.CID
HAVING COUNT(DISTINCT C.CName) > 1;
Output: Empty set

- Display schedule of a specific student:
Select S2.STID, S2.FirstName, C.Cname, ClassTimeslot.StartTime,
ClassTimeslot.DayWeek, ClassTimeslot.EndTime
from ClassTimeslot inner join Section S on ClassTimeslot.SeID =
S.SeID inner join Has H on S.SeID = H.SeID inner join Course C on
H.CID = C.CID inner join EnrolledIn EI on S.SeID = EI.SeID inner join
Student S2 on EI.STID = S2.STID
WHERE S2.STID=1 and S.Year=2020 and S.Semester='Winter';
Output:

STID	FirstName	Cname	StartTime	DayWeek	1	EndTime
1	Felix	MECH300	09:30:00	Wednesday		10:45:00
1	Felix	ENGR300	12:45:00	Friday		13:45:00

CONTRIBUTIONS

We forgot