Ryerson University Faculty of Science

Department of Computer Science

Course Number	510
Course Title	CPS
Semester/Year	Fall 2020
Instructor	Dr. A. Abhari
TA Name	Haytham Qushtom

Report Title	Assignment 10

Section No.	8
Submission Date	12/01/2020
Due Date	12/01/2020

Student Name	Student ID	Signature*
Mohamad Alkahil	xxxx97490	[illegible]
Michael Teshome	xxxx9929	[illegible]

(Note: remove the first 4 digits from your student ID)

Table of Contents

***Relational Algebra*:**

Query 1: List all the names of users from Toronto

```
SELECT Name
FROM Users
WHERE City = 'Toronto';
```

RA:

$$\Pi_{Name}(\sigma_{City = 'Toronto'}(Users))$$

Query 2: List all Female users who are gay

```
SELECT Name
FROM Users
WHERE gender = 'F' AND SexualOrn = 'Gay';
```

RA:

$$\Pi_{Name}(\sigma_{gender = 'F' \ AND \ SexualOrn = 'Gay'}(Users))$$

Query 3: List all male users who are straight

```
SELECT Name
FROM Users
WHERE gender = 'M' AND SexualOrn = 'Straight';
```

RA:

$$\Pi_{Name}(\sigma_{gender = 'M' \ AND \ SexualOrn = 'Straight'}(Users))$$

Query 4: List userID's for the users that have been blocked more than 5 times

```
SELECT UserID
FROM Monitors
WHERE BlockNum>5;
```

RA:

$$\Pi_{UserID}(\sigma_{BlockNum \ > 5}(Monitors))$$

Query 5: List all the matches(users who have liked each other)

```
SELECT m.name, l.name
FROM Users m, users l, likedusers
WHERE liker = m.userid AND likee = l.userid AND
EXISTS (
          SELECT liker
          FROM likedusers
          WHERE likee = m.userid AND liker = l.userid);
```

RA:

$$Original \leftarrow \rho_{LikedUsers(Liker1,\ Likee1)}(LikedUsers)$$

$\Pi_{Liker1,\ Likee1}$ (LikedUsers$_{\bowtie \text{Liker1 = Likee AND Likee1 = Liker}}$Original)

Query 6: List all the Likes for each user
SELECT e.Name AS Liker, r.Name AS Likee
FROM Users e,Users r,LikedUsers
WHERE e.UserID=Liker AND r.UserID=Likee
ORDER BY e.UserID;
RA:

LikerID ← $\Pi_{UserID,\ Name}\ (Users)$

$\rho_{LikerID(LikerID,\ LikerName)}(LikerID)$

Likers ← LikedUsers$_{\bowtie \text{Liker = LikerID}}$LikerID

LikeeID ← $\Pi_{UserID,\ Name}\ (Users)$

$\rho_{LikeeID(LikeeID,\ LikeeName)}(LikeeID)$

Likes ← LikeeID$_{\bowtie \text{LikeeID = Likee}}$Likers

Result ← $\Pi_{LikerName,\ LikeeName}\ (Likes)$

Query 7: List the number of likes for each 'liker'
SELECT e. name AS Liker, COUNT(r.name) AS NUMBER_OF_LIKES
FROM Users e, users r, LikedUsers
WHERE e.userid = liker AND r.userid = likee
GROUP BY e.name
ORDER BY e.name;
RA:

LikerID ← $\Pi_{UserID,\ Name}\ (Users)$

$\rho_{LikerID(LikerID,\ LikerName)}(LikerID)$

Likers ← LikedUsers$_{\bowtie \text{Liker = LikerID}}$LikerID

$_{LikerName}F_{COUNT(Liker)}(Likers)$

*counts the number of times the Liker appears in the table, groups by LikerName.

Query 8: List all the active users(A.K.A which users are actively liking others)
SELECT e.Name AS Active
FROM Users e, LikedUsers
WHERE e.UserID = Liker AND EXISTS(
    SELECT r.Name AS Likee

    FROM Users r, LikedUsers
    WHERE r.UserID = Likee);

RA:

$LikerID \leftarrow \Pi_{UserID,\, Name}(Users)$

$\Pi_{Name}(\text{LikedUsers} \bowtie_{\text{Liker = UserID}} \text{LikerID})$

<u>Query 9</u>: List the companies that are in the same city as the users and have a rating greater than 3.5

SELECT name, COUNT(companyname) AS NUM_OF_Dating_Spots
FROM Users, OutsideCompanies, Locations
WHERE rating > 3.5 AND users.city = locations.city AND locations.locationid = outsidecompanies.locationid
GROUP BY name
ORDER BY name;

RA:

$LocationInfo \leftarrow \Pi_{LocationID,\, city}(Locations)$

$CompanyInfo \leftarrow \Pi_{LocationID,\, rating}(OutsideCompanies)$

$UserInfo \leftarrow \Pi_{Name,\, city}(Users)$

${}_{Name}F_{COUNT(Name)}(\text{UserInfo} \bowtie (\text{CompanyInfo} \bowtie_{\text{rating} > 3.5} \text{LocationInfo}))$

*count the number of times a user's name appears in the final relation, which is equal to the number of companies within the same city as the user.

<u>Query 10</u>: List the name of the users who have been blocked too many times

SELECT u.name as Hostile_User
FROM   Users u,Monitors
WHERE   u.userid = monitors.userid
AND EXISTS
(SELECT m.userid as Hostile_User
FROM   Monitors m
WHERE m.BlockNum>7 AND u.userid = m.userid);

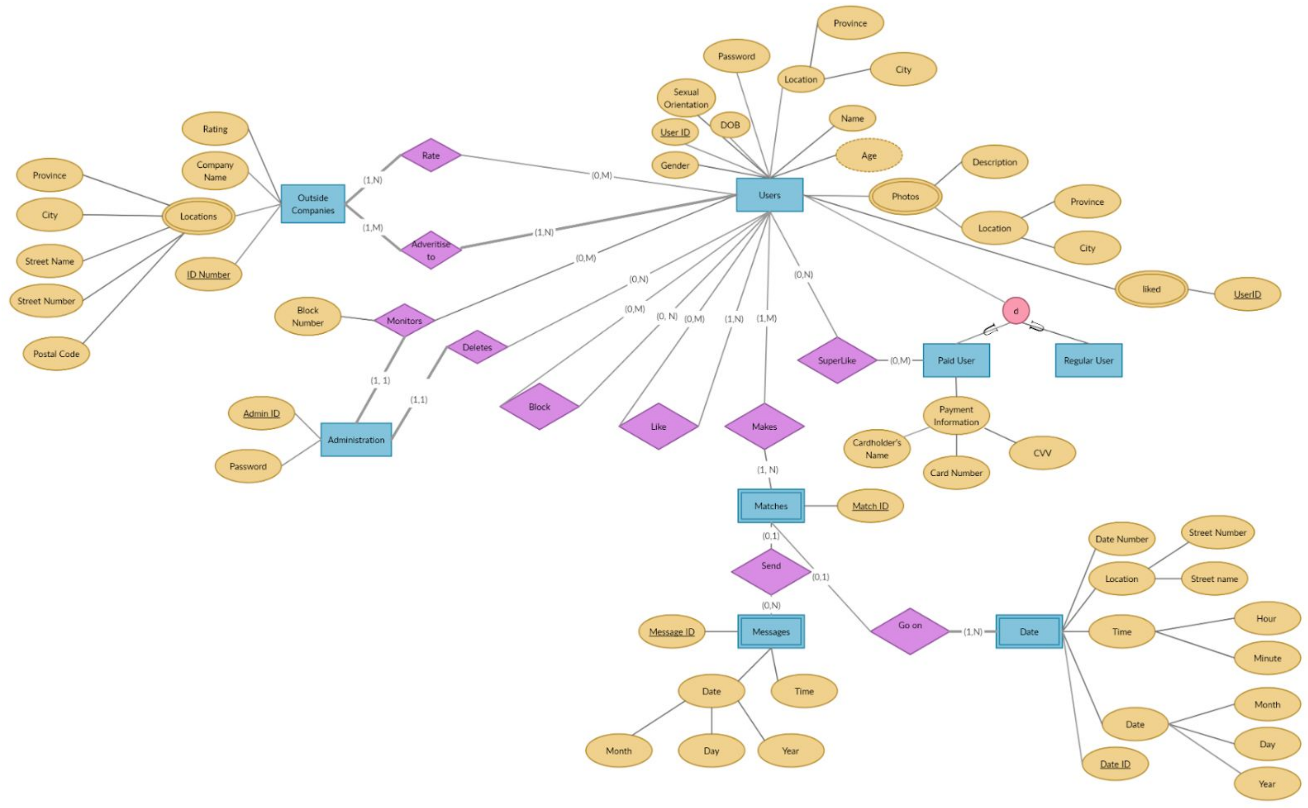
RA:

$(\Pi_{Name}((\Pi_{UserID,\, Name}(\text{Users}) \bowtie \text{Monitors}))$

<u>Project Description</u>:
The database system is for a dating application. Users can upload images of themselves to add to their profile. Users can "like" other users to create potential matches, matches can send messages, and this leads to potential dates. The application also provides the option to block other users who they no longer wish to communicate with.

The entities associated with this application are the regular users, administrators, outside companies, matches, dates, and messages. Paying users have the same functionality as regular users, but they are also allowed to "super like" another user (showing a larger interest in who they want to match with). Administrators have the ability to remove the accounts of users who have been reported for harassment. Outside Companies will be allowed to advertise their businesses on the app as potential date destinations for the users(i.e. restaurants). Users will be able to rate the outside companies in their profile. The messages between users will be time stamped. When Users want to go on dates they can set up the day, time, and location of the date through the application.

<u>Final ER Diagram</u>:



<u>Tables</u>:

```
CREATE TABLE Photos(
    PhotoID VARCHAR2(25)    NOT NULL PRIMARY KEY,
    Description    VARCHAR2(200),
    Province       VARCHAR2(25),
    City           VARCHAR2(25)
);

CREATE TABLE Users(
    UserID         VARCHAR2(25)    NOT NULL PRIMARY KEY,
    Password       VARCHAR2(30)    NOT NULL,
    Name           VARCHAR2(20),
    DateOfBirth    VARCHAR2(10),
    SexualOrn      VARCHAR2(10),
    Province       VARCHAR2(25),
    City           VARCHAR2(40),
    PhotoID VARCHAR2(25)    REFERENCES     Photos(PhotoID),
    Gender         VARCHAR2(1)
);

CREATE TABLE PaidUser(
    UserID         VARCHAR2(25)    REFERENCES Users(UserID),
    CardNumber     NUMBER          NOT NULL,
CrdName VARCHAR2(25)    NOT NULL,
CVV            NUMBER              NOT NULL
);

CREATE TABLE RegularUser(
    UserID         VARCHAR2(25)    REFERENCES Users(UserID)
);

CREATE TABLE Locations(
    LocationID     VARCHAR2(25)    NOT NULL PRIMARY KEY,
    Province       VARCHAR2(25),
    City           VARCHAR2(25),
    StreetName     VARCHAR2(25),
    StreetNumber   NUMBER,
    PostalCode     VARCHAR2(6)
);
CREATE TABLE OutsideCompanies(
    IdNumber       NUMBER         NOT NULL PRIMARY KEY,
    CompanyName    VARCHAR2(20),
    LocationID     VARCHAR2(25)   REFERENCES Locations(LocationID),
      Rating           NUMBER(3,2)
);

CREATE TABLE Administration(
    AdminID VARCHAR2(25)   NOT NULL PRIMARY KEY,
    Password       VARCHAR2(30)   NOT NULL
);

CREATE TABLE Matches(
    MatchID        VARCHAR2(25)   NOT NULL PRIMARY KEY,
    UserID              VARCHAR2(25)   REFERENCES     Users(UserID),
      AdminID        VARCHAR2(25)   REFERENCES Administration(AdminID)
);

CREATE TABLE Messages(
    MessageID      VARCHAR2(25)   NOT NULL PRIMARY KEY,
    Time           NUMBER,
    Month          VARCHAR2(9),
    Day            NUMBER,
    Year           NUMBER,
    MatchID VARCHAR2(25)   REFERENCES     Matches(MatchID)
);
CREATE TABLE Dates(
    DateID         VARCHAR2(25)   NOT NULL PRIMARY KEY,
    MessageID      VARCHAR2(25)   REFERENCES     Messages(MessageID),
    StreetName     VARCHAR2(25),
    StreetNumber   NUMBER,
    Hour           NUMBER,
    Minute         NUMBER,
    Month          VARCHAR2(9),
    Day            NUMBER,
    Year           NUMBER,
    DateNum        NUMBER
);

CREATE TABLE UserLikes(
    Liker          VARCHAR2(25)   REFERENCES     Users(UserID),
    Likee          VARCHAR2(25)    REFERENCES      Users(UserID),
    MatchID VARCHAR2(25)    REFERENCES      Matches(MatchID)
);

CREATE TABLE Rate(
      UserID         VARCHAR2(25)    REFERENCES      Users(UserID),
      IdNumber       NUMBER          REFERENCES OutsideCompanies(IdNumber)
);

CREATE TABLE AdvertiseTo(
    UserID         VARCHAR2(25)    REFERENCES      Users(UserID),
IdNumber       NUMBER          REFERENCES OutsideCompanies(IdNumber)
);

CREATE TABLE GoesOn(
    MatchID VARCHAR2(25)    REFERENCES      Matches(MatchID),
    DateID         VARCHAR2(25)    REFERENCES      Dates(DateID)
);

CREATE TABLE Monitors(
    UserID         VARCHAR2(25)    REFERENCES      Users(UserID),
    AdminID        VARCHAR2(25)    REFERENCES      Administration(AdminID),
    BlockNum       NUMBER
);

CREATE TABLE Makes(
    UserID         VARCHAR2(25)    REFERENCES      Users(UserID),
    MatchID VARCHAR2(25)    REFERENCES      Matches(MatchID)
);

CREATE TABLE Block(
    Blocker        VARCHAR(25)     REFERENCES      Users(UserID),
    Blockee        VARCHAR(25)     REFERENCES      Users(UserID)
);

CREATE TABLE SuperLike(
    Liker          VARCHAR2(25)    REFERENCES      Users(UserID),
    Likee          VARCHAR2(25)    REFERENCES      Users(UserID)
);
```

```
CREATE TABLE LikedUsers(
	Liker		VARCHAR2(25)	REFERENCES	Users(UserID),
	Likee		VARCHAR2(25)	REFERENCES	Users(UserID)
);
```

Dummy Data:

Photos

UsersID	PhotoID	Description	Province	City
daveywavey	123098	Me at the beach!	Ontario	Toronto
L_scar17	3421345	Just chillin	Ontario	Ottawa
J_doeyo	1789422	Nature calls	Ontario	Toronto
frankybby	1285435	Music is life	Ontario	Toronto
marrymac	700021	Where's Johnny?	Ontario	Toronto
jimmydee	390283	Slim Jim	Ontario	Ottawa

Users

UserID	Password	Name	DateOfBirth	SexualOrn	Province	City	PhotoID	Gender
DaveyWavey	dw123	David Jackson	23/10/95	Straight	Ontario	Toronto	123098	M
L_scar17	expelliarmus	Harry Potter	31/07/80	Straight	Ontario	Ottawa	3421345	M
J_doeyo	hey9022	Jessica Doe	04/03/83	Gay	Ontario	Toronto	1789422	F
frankybby	ratpack200	Frank Sinatra	12/12/92	Bi	Ontario	Toronto	1285435	M
marrymac	jfkily	Marilyn Monroe	01/06/93	Bi	Ontario	Toronto	700021	F
jimmydee	hmuiyr	Jimmy Ricky	03/17/99	Gay	Ontario	Ottawa	390283	M
ChezzyYezzy	scoobdypoop	Kanye West	06/08/77	Bi	Ontario	Ottawa	123678	M

OutsideCompanies

IDNumber	CompanyName	LocationID	Rating
543432892	Rudy’s Bar	839298	4.3
138458901	McDonalds	128849	3.2
1234385953	Cafe du Marc	024801	3.6
905493891	Quencher	165739	4.8

Messages

MessageID	Month	Day	Year
5345821	October	20	2020
9091212	March	13	2020
1209437	October	20	2020

Dates

DateID	DateNum	StreetNumber	StreetName	Hour	Minute	Month	Day	Year
224986	1	13	Yardon lane	3	30	September	1	2020
224987	2	12	Yellowbrick road	2	15	October	15	2020

Administration

AdminID	**Password**
admin123	admin456

Monitors

UserID	**AdminID**	**BlockNum**

frankybby	admin123	2
J_doeyo	admin123	10

Locations

Location ID	Province	City	StreetName	StreetNumber	PostalCode
839298	Ontario	Toronto	Young	53	M192W6
128849	Ontario	Ottawa	Dumb	17	M342P7
024801	Ontario	Toronto	Broke	32	M058L4
165739	Ontario	Barrie	Blue	22	M183F3

Liked_Users

Liker (userID)	Likee(userID)
frankybby	marrymac
ChezzyYezzy	jimmydee
DaveyWavey	marrymac
J_doeyo	marrymac
jimmydee	frankybby
frankybby	ChezzyYezzy
marrymac	frankybby

<u>Proof Database is 3NF:</u>
<u>Users</u>:
R(User_ID, Gender, DOB, Password, City, Province, Photo_ID, Name)
<u>Bernstein's Algorithm:</u>
Step 1: Write out all the functional dependencies

1. User_id → Gender
2. User_id → DOB
3. User_id → Password
4. User_id → City
5. User_id → Province
6. User_id → Photo_id
7. User_ID → Name
8. DOB, Name, City, Province → Gender
9. DOB, Name, City, Province → Sexual Orientation
10. DOB, Name, City, Province → Password
11. DOB, Name, City, Province → Photo_id
12. City → Province
13. Photo_id → User_id

Functional dependencies 8 through 11 were made based on the assumption that it is incredibly unlikely to have 2 people with the same name and birthday who are from the same city.

Step 2: Check for redundancies in each of the functional dependencies:

1. User_id → Gender
    a. User_id+ = {User_id, DOB, Password, city, Province, Photo_id, Name, gender}
    b. The above includes gender, so the functional dependency is redundant
2. User_id → DOB
    a. User_id+ = {User_id, gender, Password, city, Province, Photo_id, Name}
    b. The above does not include DOB, so the functional dependency is not redundant
3. User_id → Password
    a. User_id+ = {User_id, DOB, gender, city, Province, Photo_id, Name, Password}
    b. The above does include Password, so the functional dependency is redundant
4. User_id → City
    a. User_id+ = {User_id, DOB, Password, gender, Province, Photo_id, Name}
    b. The above does not include city, so the functional dependency is not redundant
5. User_id → Province
    a. User_id+ = {User_id, DOB, Password, city, gender, Photo_id, Name, Province}
    b. The above does include Province, so the functional dependency is redundant

6. User_id → Photo_id
    a. User_id+ = {User_id, DOB, Password, city, Province, Name, Photo_id}
    b. The above does include Photo_id, so the functional dependency is redundant
7. User_id → Name
    a. User_id+ = {User_id, DOB, Password, city, Province, Photo_id, gender}
    b. The above does not include Name, so the functional dependency is not redundant
8. DOB, Name, City, Province → Gender
    a. {DOB, Name, City, Province}+ = {DOB, Name, City, Province, Sexual Orientation, Password, Photo_id, User_id, gender}
    b. The above does include Gender, so the functional dependency is redundant
9. DOB, Name, City, Province → Sexual Orientation
    a. {DOB, Name, City, Province}+ = {DOB, Name, City, Province, gender, Password, Photo_id, User_id, Sexual Orientation}
    b. The above does include Sexual Orientation, so the functional dependency is redundant
10. DOB, Name, City, Province → Password
    a. {DOB, Name, City, Province}+ = {DOB, Name, City, Province, gender, Sexual Orientation, Photo_id, User_id, Password}
    b. The above does include Password, so the functional dependency is redundant
11. DOB, Name, City, Province → Photo_id
    a. {DOB, Name, City, Province}+ = {DOB, Name, City, Province, gender, Sexual Orientation, Password}
    b. The above does not include Photo_id, so the functional dependency is not redundant
12. City → Province
    a. City+ = {city}
    b. The above does not include Province, so the functional dependency is not redundant
13. Photo_id → User_id
    a. Photo_id+ = {Photo_id}
    b. The above does not include User_id, so the functional dependency is not redundant.

Note: Since city → Province, the determinants for dependencies 8 through 11 can be reduced as follows:

8. DOB, Name, City → Gender
9. DOB, Name, City → Sexual Orientation
10. DOB, Name, City → Password
11. DOB, Name, City → Photo_id

Final set of functional dependencies:
User_id → DOB
User_id → City
User_id → Name
DOB, Name, City → Photo_id
City → Province
Photo_id → User_id
Step 3: Determine the keys
Sub-steps:

1. Determine if any attributes are not used in any functional dependencies
    a. These are present in all of the keys
2. Determine if any attributes are only present on the right side of a functional dependency
    a. These are not present in any of the keys
3. Test the remaining attributes to see if they are keys.

Sub-Step 1:
No such attributes exist. All attributes are present in some functional dependency

Sub-Step 2:
{Gender , Sexual Orientation, Password)
The above attributes will not be present in any of the keys.

Sub-Step 3:
Remaining attributes: {User_id, Photo_id, DOB, name, city, Province}. Check each:

User_id + = {User_id, gender, DOB, Password, City, Province, Photo_id, Name} → User_id is a key!

Photo_id+ = { Photo_id, User_id, gender, DOB, Password, City, Province, Name} → Photo_id is a key!

Name, City, DOB, and province, by themselves are not keys.

{Name, City, DOB}+ = {Name, City, DOB, gender, Province, Photo_id, User_id, Password} → {Name, City, DOB} is a key!

Step 4: Derive the final relational schema:
R.1 (User_id, DOB, City, Name)
R.2 (DOB, Name, City, Photo_id)
R.3 (City, Province)

R.4(Photo_id, User_id)

<u>BCNF Algorithm:</u>

R.1(User_id, DOB, city, name):

User_id → DOB
User_id → city
User_id → Name

Has 1 candidate key: User_id, and since every attribute is determined by a candidate key in this relation, R.1 is in the BCNF form.

R.2(DOB, city, name, Photo_id):

DOB, city, Name → Photo_id

Has 1 candidate key: {DOB, city, Name}, and since every attribute is determined by a candidate key in this relation, R.2 is in the BCNF form.

R.3(city, Province):

City → Province

Has 1 candidate key: city,, and since every attribute is determined by a candidate key in this relation, R.3 is in the BCNF form.

R.4(Photo_id, User_id):

Photo_id → User_id

Has 1 candidate key: Photo_id, and since every attribute is determined by a candidate key in this relation, R.4 is in the BCNF form.

<u>Outside Companies</u>:
R(ID_Number, Company_Name, LocationID, Rating)
<u>Bernstein's Algorithm:</u>
Step 1: Write out all the functional dependencies

1. ID_Number → Rating
2. ID_Number → Company_Name
3. ID_Number → LocationID
4. Company_Name, Locations → Rating

Step 2: Check for redundancies in each of the functional dependencies:

1. ID_Number → Rating
    a. ID_Number+ = {ID_Number, Company_Name, LocationID, Rating}
    b. The above include rating, so the functional dependency is redundant
2. ID_Number → Company_Name

    a. ID_Number+ = {ID_Number, Rating, LocationID}
    b. The above does not include Company_Name, so the functional dependency is not redundant.
3. ID_Number → LocationID
    a. ID_Number+ = {ID_Number, Company_Name, Rating}
    b. The above does not include LocationID, so the functional dependency is not redundant.
4. Company_Name, LocationID → Rating
    a. {Company_Name, LocationID}+ = {Company_Name, LocationI}
    b. The above does not include Rating, so the functional dependency is not redundant.

Final set of functional dependencies:

ID_Number → Company_Name
ID_Number → LocationID
Company_Name, Locations → Rating

Step 3: Determine the keys
Sub-steps:

1. Determine if any attributes are not used in any functional dependencies
    a. These are present in all of the keys
2. Determine if any attributes are only present on the right side of a functional dependency
    a. These are not present in any keys
3. Test the remaining attributes to see if they are keys

Sub-Step 1:
No such attributes exist. All attributes are present in some functional dependency

Sub-Step 2:
{Rating}
The above attribute will not be present in any of the keys.

Sub-Step 3:
Remaining attributes: {ID_Number, Company_Name, LocationID}. Check each:

ID_Number+ = {ID_Number, Company_Name, LocationID, Rating}
ID_Number is a key!

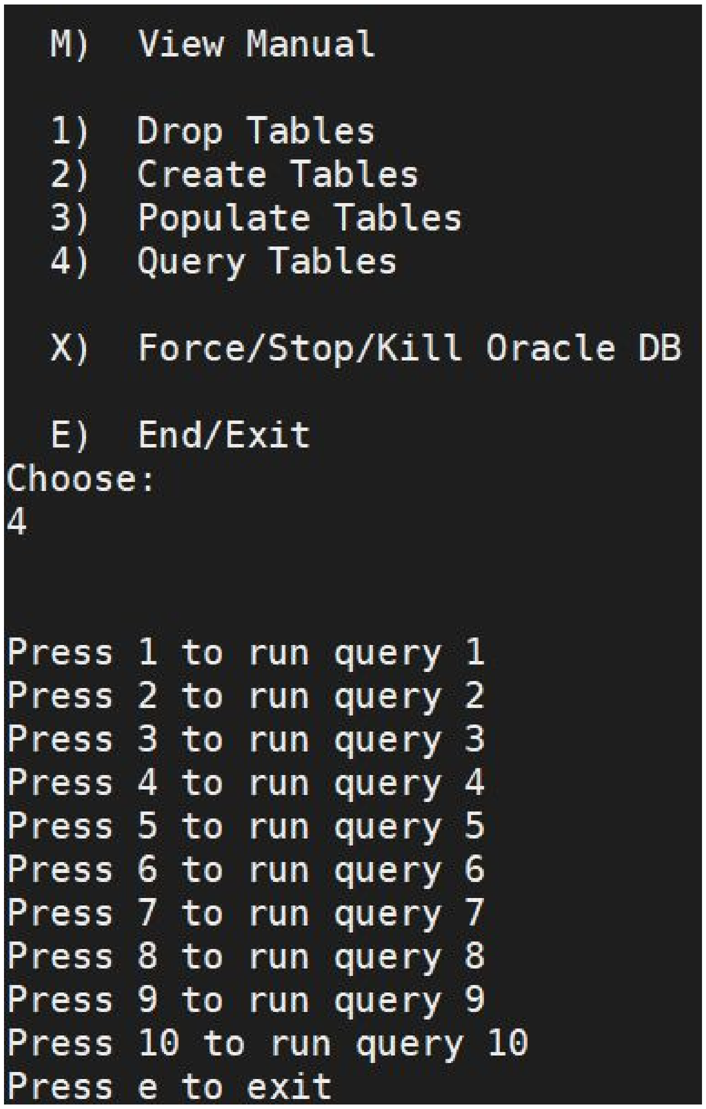
Company_Name and LocationID are not keys on their own.

{Company_Name, LocationID}+ = {Company_Name, LocationID, Rating, ID_Number}
{Company_Name, LocationID} is a key!

Step 4: Derive the final relational schema:
R.1 (ID_Number, Company_Name)
R.2 (ID_Number, LocationID)
R.3 (Company_Name, LocationID, Rating)

All the other tables in the database are in 3NF.

Screen of the User Interface:



```
  M)  View Manual

  1)  Drop Tables
  2)  Create Tables
  3)  Populate Tables
  4)  Query Tables

  X)  Force/Stop/Kill Oracle DB

  E)  End/Exit
Choose:
4


Press 1 to run query 1
Press 2 to run query 2
Press 3 to run query 3
Press 4 to run query 4
Press 5 to run query 5
Press 6 to run query 6
Press 7 to run query 7
Press 8 to run query 8
Press 9 to run query 9
Press 10 to run query 10
Press e to exit
```

Concluding Remarks

Designing the database was challenging at times, especially since none of the members of the group had worked with SQL before. Most of the assignments were straightforward if you kept up with the course material, but there were a few where we felt that we were at a disadvantage because we are both computer engineering students and the course was originally designed for computer science students. For future students it may be beneficial to include the rubrics ahead of each assignment rather than releasing some of them after the due date. All in all, the experience was fairly educational, and did expose the both of us to a new form of data management that we had not encountered in the past. It was a helpful learning experience.