Final Project:

Netflix Database

Names of Participants:
Daaron McFarling Bao-Tran Phan Shrinath Rao Shurong Tian Adam Uribe Jerry Zeng

Class: CS4347.0U1 || **Professor**: Dr. Cankaya || **Term**: Summer 2018

Table of Contents

Netflix Database: Deliverable I

Introduction

Movies and TV have always been a major topic of discussion among millenials. Previously, big movie rental companies, such as RedBox and Blockbuster, have played major roles in the sense of socialization, however, the use of technology and the internet has become more widespread among the developed world. At this point, we have progressed to a point where streaming media is a more viable solution than playing media through physical disks.

Netflix is a video streaming service that has become very popular among millennials because of its usability and its extensive library of various movies and TV shows.Since we all use Netflix, our team thought it would be interesting to try and develop a database to understand the potential workings behind the scenes.

If our team implements this database, it will be mainly for our education and to help us understand the processes of creating a functioning database. By having the necessary information in a database needed for a user to watch videos in a simplistic way, once a GUI is created, a person would have a practical video streaming application.

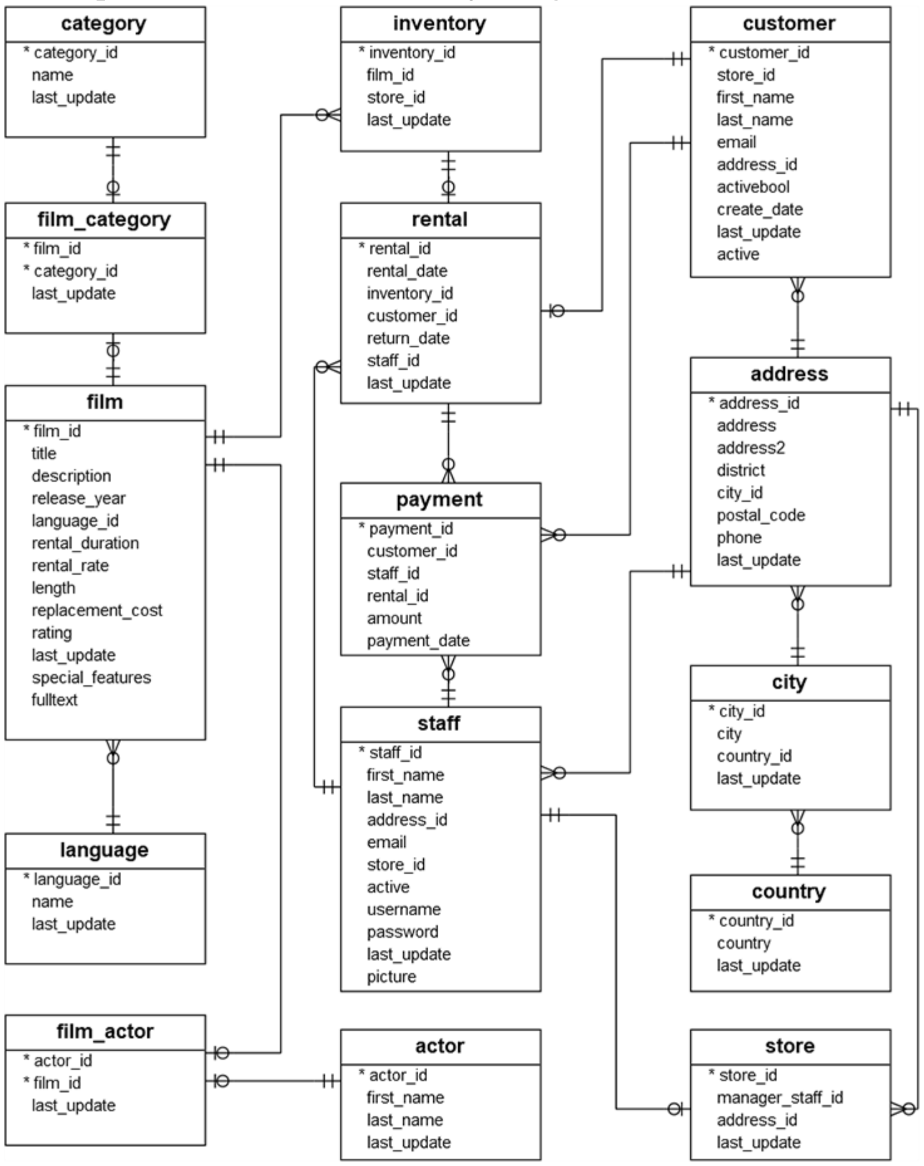
Delegation of Tasks for Deliverable I

Member Name	Primary Role
Daaron McFarling	Comparison to Similar Works
Bao-Tran Phan	Documentation Editing
Shurong Tian	EER and Schema diagrams
Shrinath Rao	Editing and Implementation
Adam Uribe	Implementation of Database in MS SQL
Jerry Zeng	Diagram creation

Research on Similar Pre-Existing Databases

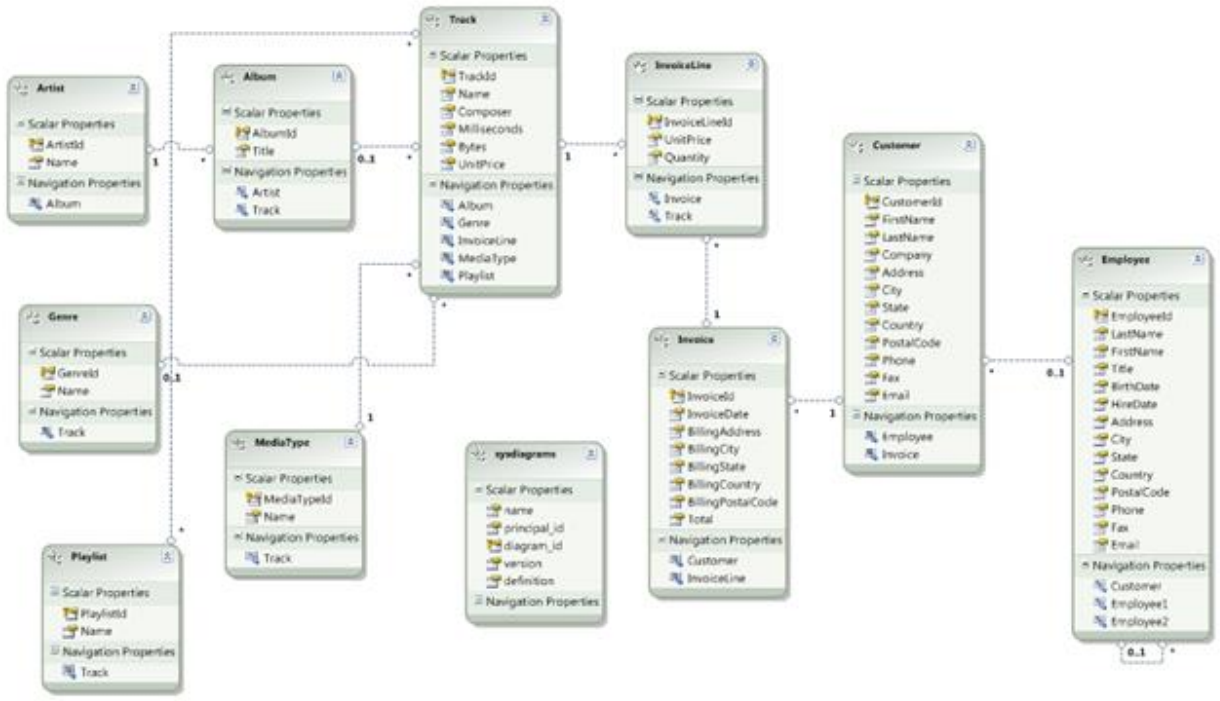
Our team did some research to see what aspects we could potentially utilize in our own Netflix database. Other than looking at other Netflix databases, we checked examples for entities that would provide similar services to Netflix, such as iTunes and DVD Rentals. Our

1. Sample DVD Rental Database *by PostgreSQL*



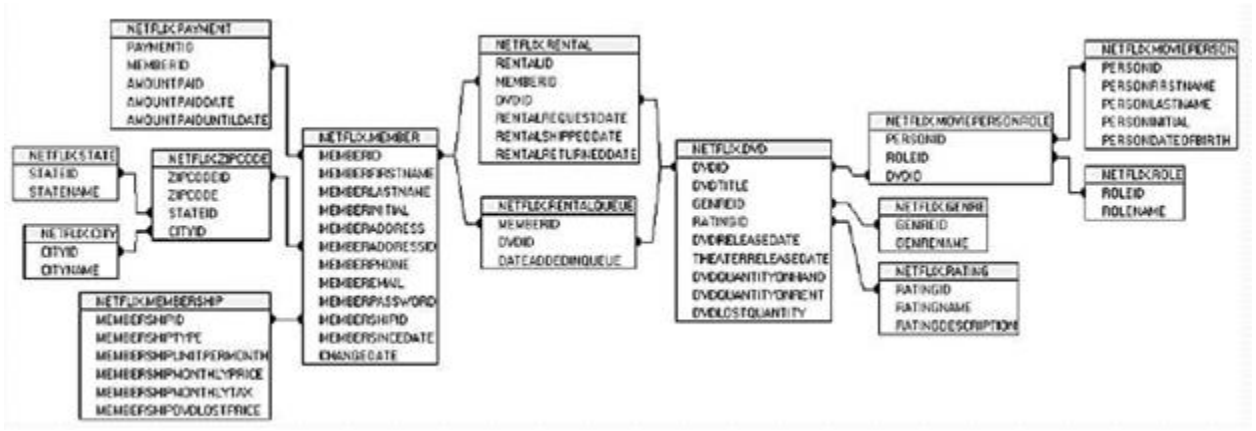
[1] About this Database: This sample database found on SQL tutorial website. It is designed as a DVD rental service, which Netflix used to be. Many attributes we had not originally considered when creating our own database, such as language, actor and director IDs, and many other different ways to keep track of content besides a large database of *all* films [1]. While this database accounts for stores ours does not since we are considering a video streaming service.

2. Popular UI Volume 2- iTunes *by Grapecity*



[2] About This Database: Another similarly designed database would be one designed for another media platform for music, such as iTunes. This database model introduced us to idea for multiple platforms that could be running the same application of Netflix. For example, we could have users that have Netflix running on various devices, such as their smart TVs, their Phones, tablets, consoles, PCs, etc... [2] While this database accounts for devices ours doesn't' since ours focuses more on Netflix's capability to recommend and track statistics.

3. OLTP Netflix Database *by Rona Charlene Lao*



[3] About This Database: This database keeps track of online transaction processing. It has a large number of attributes to keep track of payment information, which we plan to implement something similar in our own database to keep track of a customer's payment information [3]. This database is quite similar to our own since we too will want to keep track of the user's payment info.

Sources

[1] Postgresqltutorial.com. (2018). PostgreSQL Sample Database. [online] Available at: https://bit.ly/2dnqtnr [Accessed 25 Jun. 2018].

[2] B, C. (2018). Popular UI Volume 2 – iTunes. [online] GrapeCity. Available at: https://bit.ly/2tIyyfN [Accessed 28 Jun. 2018].

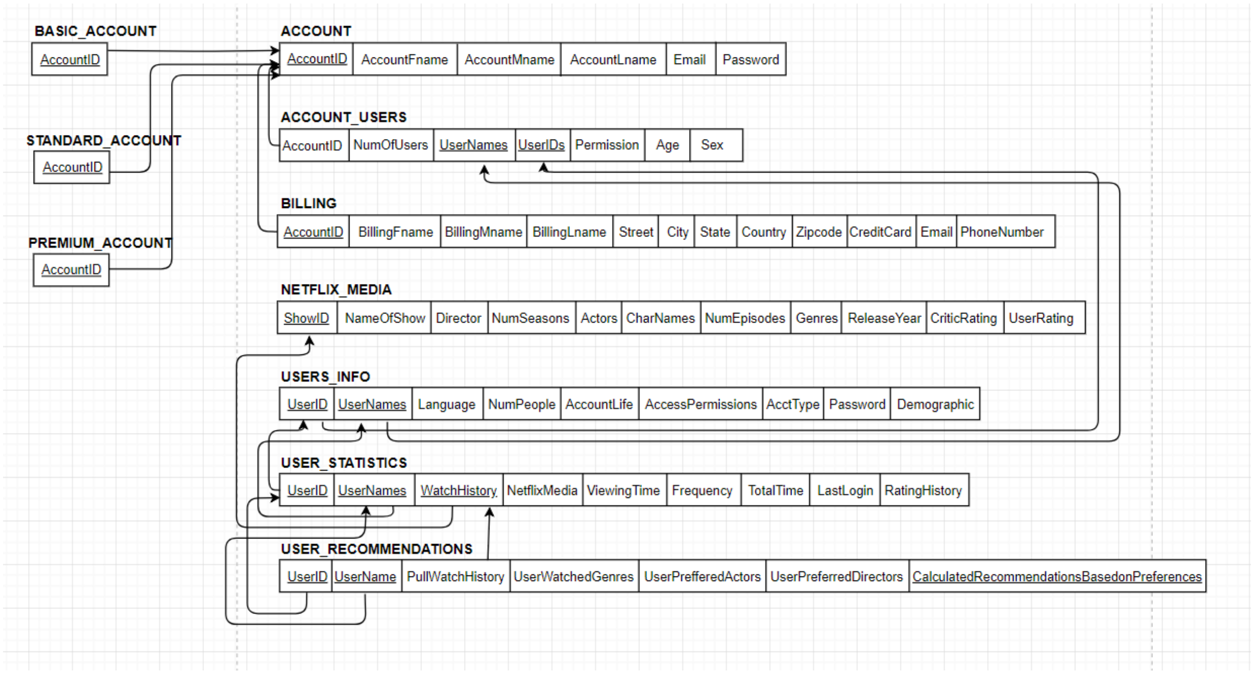
[3] Lao, R. (2018). Ronalao termpresent. [online] Slideshare.net. Available at: https://bit.ly/2KoN72z [Accessed 27 Jun. 2018].

EER Diagram v. 1

See attached file "v1_EER+Schema Diagrams.pdf" in deliverable 1 folder.
To get a better view for both diagrams, zoom in to at least 125%.

Schema Diagram

To get a better view, please see the attached file "v1_EER+Schema Diagrams.pdf" in deliverable 1 folder and zoom in to at least 125%.


BASIC_ACCOUNT
AccountID
STANDARD_ACCOUNT
AccountID
PREMIUM_ACCOUNT
AccountID
ACCOUNT
AccountID | AccountFname | AccountMname | AccountLname | Email | Password
ACCOUNT_USERS
AccountID | NumOfUsers | UserNames | UserIDs | Permission | Age | Sex
BILLING
AccountID | BillingFname | BillingMname | BillingLname | Street | City | State | Country | Zipcode | CreditCard | Email | PhoneNumber
NETFLIX_MEDIA
ShowID | NameOfShow | Director | NumSeasons | Actors | CharNames | NumEpisodes | Genres | ReleaseYear | CriticRating | UserRating
USERS_INFO
UserID | UserNames | Language | NumPeople | AccountLife | AccessPermissions | AcctType | Password | Demographic
USER_STATISTICS
UserID | UserNames | WatchHistory | NetflixMedia | ViewingTime | Frequency | TotalTime | LastLogin | RatingHistory
USER_RECOMMENDATIONS
UserID | UserName | PullWatchHistory | UserWatchedGenres | UserPrefferedActors | UserPreferredDirectors | CalculatedRecommendationsBasedonPreferences

https://drive.google.com/file/d/1XMl9-Hld8NtNq-RA3TRe-1E9-FEn9Xcq/view?usp=sharing

SQL Code with Insert Statements

See attached file, "v1_Netflix.sql" in deliverable 1 folder.

Netflix Database: Deliverable II

Delegation of Tasks for Deliverable II

Member Name	Primary Role
Daaron McFarling	Documentation, Diagram Updates
Bao-Tran Phan	Documentation, Diagram Creation and Updates
Shurong Tian	Back-end GUI
Shrinath Rao	Front-end GUI
Adam Uribe	Create view statement, Updated SQL Code, Updated Schema Diagram
Jerry Zeng	Diagram Updates

Schema Diagram v.2

See attached "*Schema Diagram v2.png*" in deliverable 2 folder.

EER Diagram v.2

See attached "*EERv2.png*" in deliverable 2 folder.
To get a better view of the diagram, zoom in to at least 125%.

Changes made since v.1:
While creating the dependency diagram for our database, we noticed to some redundant and/or unnecessary attributes, such as for the following information:

- Removed TotalTime since it is essentially the same thing as ViewingTime from UserStatistics
- Removed Username since it was redundant to userID information. Since we changed our scope to more back-end statistics, there is no use for the database user to use userName.
- Removed password from UserInfo since each user does not have a unique password. (Every user under the same main account will have one password.)
- Removed Username from User_REcommendations since recommendations will be linked to a specific userID anyway.

After redefining a few definitions, which caused some changes in PKs and FKs, such as for the following:

- Username is no longer a PK in Account_Users since users may have the same username but their associated UserID would still be unique to a specific user.

We also shifted some attributes around, realizing that they fit better elsewhere, such as the following:

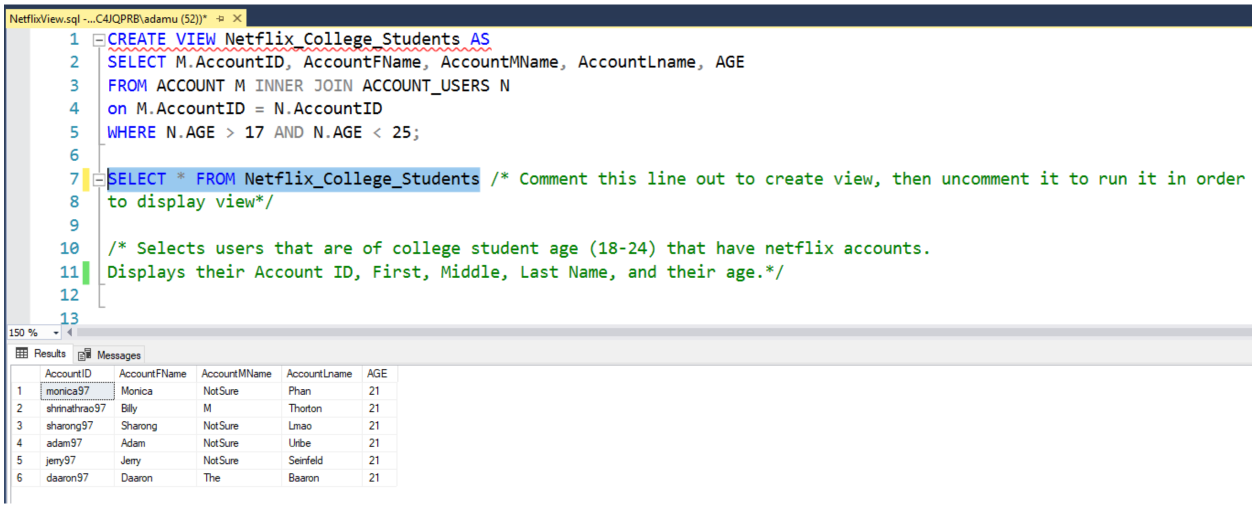
- NumPeople went from Account_Users to Account since the the numbers of users using the account is more applicable to users as a whole and not for every individual user.

Dependency Diagram

The diagram was created using draw.io: https://goo.gl/oj6wLU
To see the full diagram, click the link or see the external file "*dependencyDiagram.png*" in deliverable 2 folder.

CREATE VIEW Statement

To see Create View statement, see "NetflixView.sql" in deliverable 2 folder.


NetflixView.sql -...C4JQPRB\adamu (52))*

```
CREATE VIEW Netflix_College_Students AS
SELECT M.AccountID, AccountFName, AccountMName, AccountLname, AGE
FROM ACCOUNT M INNER JOIN ACCOUNT_USERS N
on M.AccountID = N.AccountID
WHERE N.AGE > 17 AND N.AGE < 25;

SELECT * FROM Netflix_College_Students /* Comment this line out to create view, then uncomment it to run it in order
to display view*/

/* Selects users that are of college student age (18-24) that have netflix accounts.
Displays their Account ID, First, Middle, Last Name, and their age.*/
```

	AccountID	AccountFName	AccountMName	AccountLname	AGE
1	monica97	Monica	NotSure	Phan	21
2	shrinathrao97	Billy	M	Thorton	21
3	sharong97	Sharong	NotSure	Lmao	21
4	adam97	Adam	NotSure	Uribe	21
5	jerry97	Jerry	NotSure	Seinfeld	21
6	daaron97	Daaron	The	Baaron	21

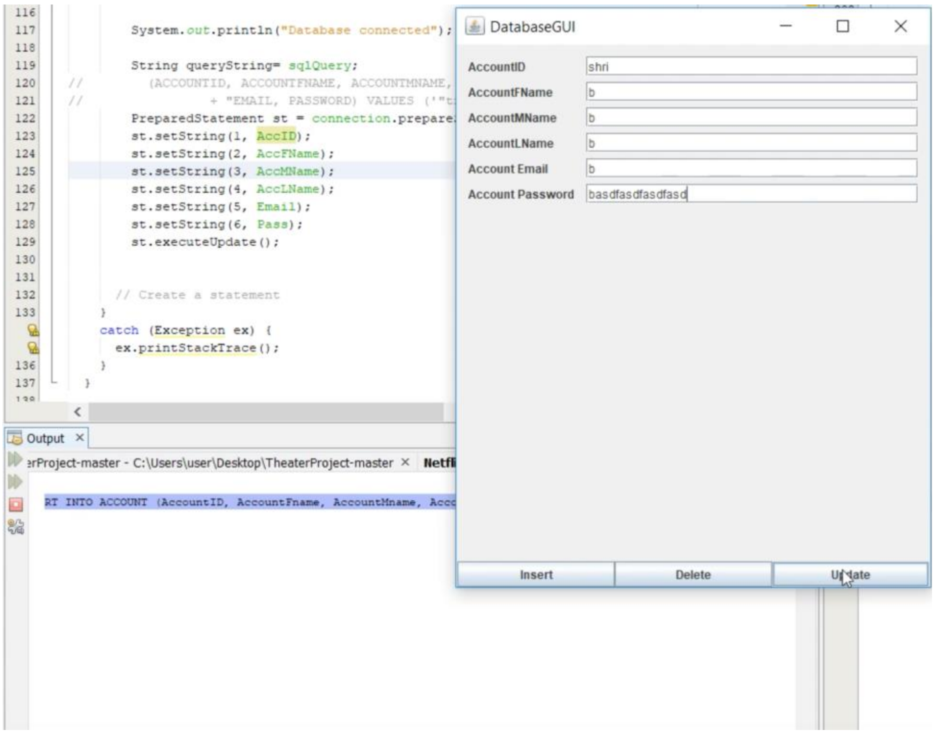
Updated SQL Code can be found in Deliverable 2 folder in file "v2_Netflix.sql"

Database User Interface

Our team used Java 1.8 to create the user interface.

See video titled "GUI Video.mp4" in Deliverable 2 folder to see GUI in action. All screenshots below are taken from the video.
(All screenshots can be found in Deliverable 2 folder if they are too small to see in the document as shown below)

Modify Operation on Account

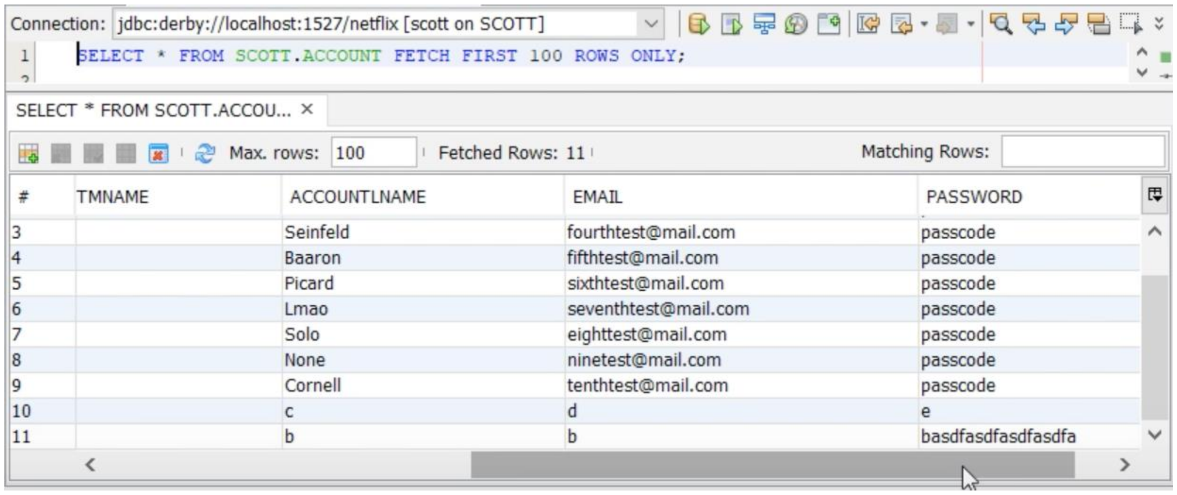


```
        System.out.println("Database connected");

        String queryString= sqlQuery;
//        (ACCOUNTID, ACCOUNTFNAME, ACCOUNTMNAME,
//                + "EMAIL, PASSWORD) VALUES ('"t
        PreparedStatement st = connection.prepare
        st.setString(1, AccID);
        st.setString(2, AccFName);
        st.setString(3, AccMName);
        st.setString(4, AccLName);
        st.setString(5, Email);
        st.setString(6, Pass);
        st.executeUpdate();


      // Create a statement
    }
    catch (Exception ex) {
      ex.printStackTrace();
    }
  }
```

```
erProject-master - C:\Users\user\Desktop\TheaterProject-master

RT INTO ACCOUNT (AccountID, AccountFname, AccountMname, Acc
```

DatabaseGUI

AccountID	shri
AccountFName	b
AccountMName	b
AccountLName	b
Account Email	b
Account Password	basdfasdfasdfasd



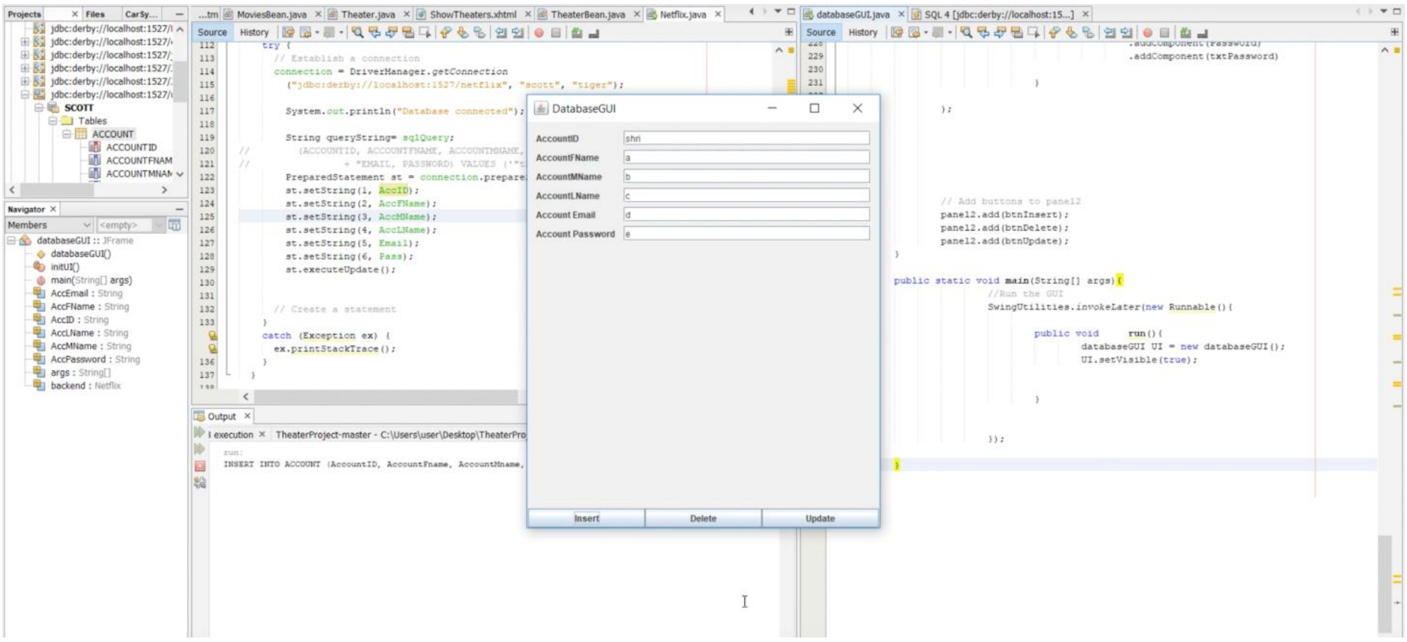
Connection: jdbc:derby://localhost:1527/netflix [scott on SCOTT]

```
SELECT * FROM SCOTT.ACCOUNT FETCH FIRST 100 ROWS ONLY;
```

Max. rows: 100 Fetched Rows: 11 Matching Rows:

#	TMNAME	ACCOUNTLNAME	EMAIL	PASSWORD
3		Seinfeld	fourthtest@mail.com	passcode
4		Baaron	fifthtest@mail.com	passcode
5		Picard	sixthtest@mail.com	passcode
6		Lmao	seventhtest@mail.com	passcode
7		Solo	eighttest@mail.com	passcode
8		None	ninetest@mail.com	passcode
9		Cornell	tenthtest@mail.com	passcode
10		c	d	e
11		b	b	basdfasdfasdfasdfa

Insert Operation on Account

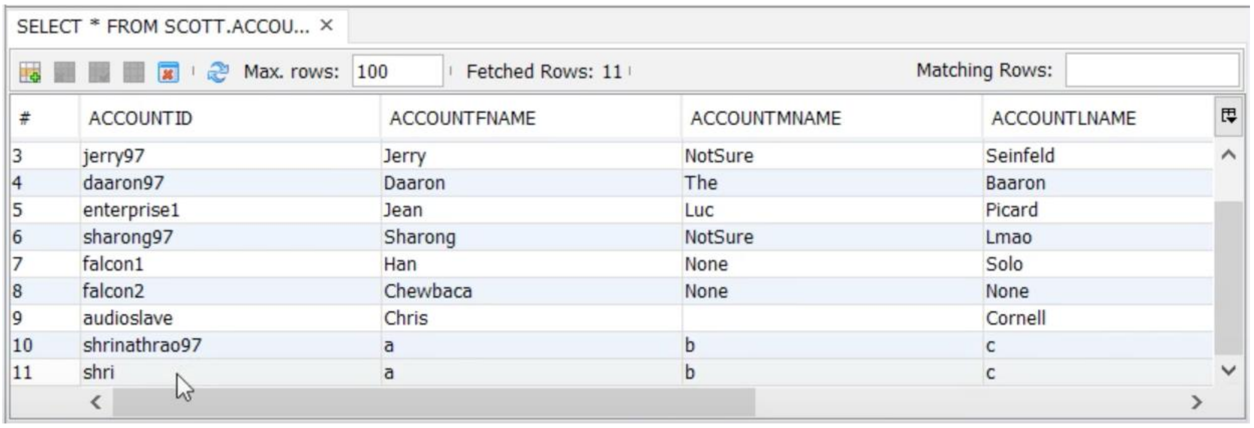


```
try {
  // Establish a connection
  connection = DriverManager.getConnection
    ("jdbc:derby://localhost:1527/netflix", "scott", "tiger");

    System.out.println("Database connected");

    String queryString= sqlQuery;
//        (ACCOUNTID, ACCOUNTFNAME, ACCOUNTMNAME,
//                  + "EMAIL, PASSWORD) VALUES ('"t
    PreparedStatement st = connection.prepare
    st.setString(1, AccID);
    st.setString(2, AccFName);
    st.setString(3, AccMName);
    st.setString(4, AccLName);
    st.setString(5, Email);
    st.setString(6, Pass);
    st.executeUpdate();


  // Create a statement
}
catch (Exception ex) {
  ex.printStackTrace();
}
}
```

```
run:
INSERT INTO ACCOUNT (AccountID, AccountFname, AccountMname,
```

AccountID	shri
AccountFName	a
AccountMName	b
AccountLName	c
Account Email	d
Account Password	e

```
                .addComponent(txtPassword)

          )

    );




    // Add buttons to panel2
    panel2.add(btnInsert);
    panel2.add(btnDelete);
    panel2.add(btnUpdate);
}

public static void main(String[] args){
        //Run the GUI
        SwingUtilities.invokeLater(new Runnable(){

            public void     run(){
                databaseGUI UI = new databaseGUI();
                UI.setVisible(true);


            }


        });

}
```

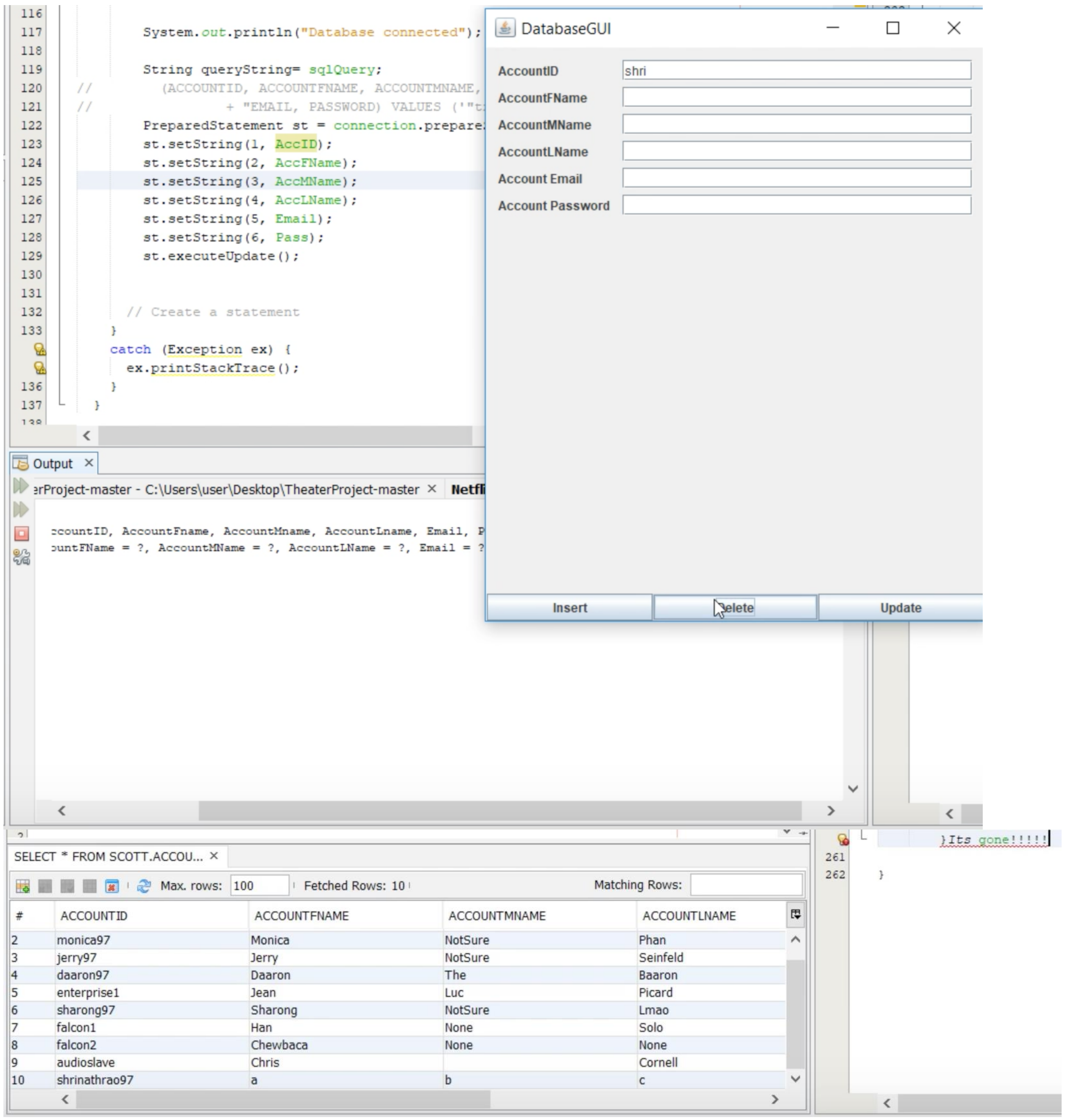


SELECT * FROM SCOTT.ACCOU...

Max. rows: 100 Fetched Rows: 11 Matching Rows:

#	ACCOUNTID	ACCOUNTFNAME	ACCOUNTMNAME	ACCOUNTLNAME
3	jerry97	Jerry	NotSure	Seinfeld
4	daaron97	Daaron	The	Baaron
5	enterprise1	Jean	Luc	Picard
6	sharong97	Sharong	NotSure	Lmao
7	falcon1	Han	None	Solo
8	falcon2	Chewbaca	None	None
9	audioslave	Chris		Cornell
10	shrinathrao97	a	b	c
11	shri	a	b	c

Delete Operation on Account



```
            System.out.println("Database connected");

            String queryString= sqlQuery;
//            (ACCOUNTID, ACCOUNTFNAME, ACCOUNTMNAME,
//                    + "EMAIL, PASSWORD) VALUES ('"t
            PreparedStatement st = connection.prepare
            st.setString(1, AccID);
            st.setString(2, AccFName);
            st.setString(3, AccMName);
            st.setString(4, AccLName);
            st.setString(5, Email);
            st.setString(6, Pass);
            st.executeUpdate();


        // Create a statement
        }
        catch (Exception ex) {
          ex.printStackTrace();
        }
    }
```

Output
erProject-master - C:\Users\user\Desktop\TheaterProject-master
Netfli

```
ccountID, AccountFname, AccountMname, AccountLname, Email, P
ountFName = ?, AccountMName = ?, AccountLName = ?, Email = ?
```

AccountID	shri
AccountFName	
AccountMName	
AccountLName	
Account Email	
Account Password	

```
}Its gone!!!!!
}
```

#	ACCOUNTID	ACCOUNTFNAME	ACCOUNTMNAME	ACCOUNTLNAME
2	monica97	Monica	NotSure	Phan
3	jerry97	Jerry	NotSure	Seinfeld
4	daaron97	Daaron	The	Baaron
5	enterprise1	Jean	Luc	Picard
6	sharong97	Sharong	NotSure	Lmao
7	falcon1	Han	None	Solo
8	falcon2	Chewbaca	None	None
9	audioslave	Chris		Cornell
10	shrinathrao97	a	b	c