NETFLIX

Database Design and Maintenance Plan

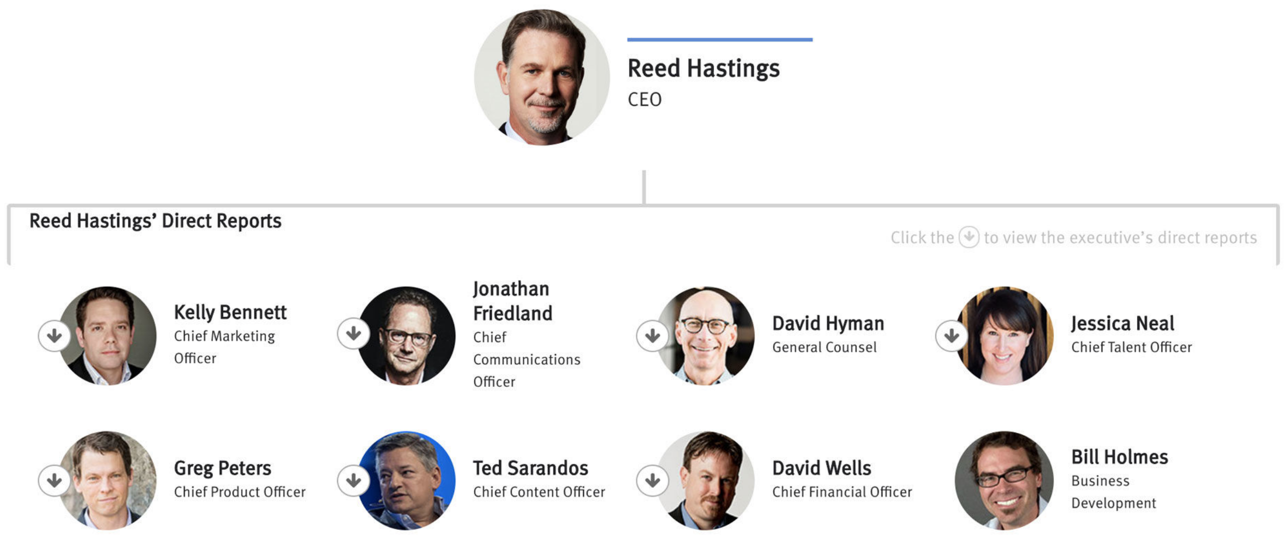
BCS390: Final Project

Marshal Multani

Netflix, Inc. is an American media-services provider based in Los Gatos, California. The company's primary business is its subscription-based streaming Over the Top service which offers online streaming of a library of movies, TV shows and documentaries, to its customers. As of April 2019, the company had over 148 million paid subscriptions total, including 60 million in the United States.

Netflix offers 3 membership plans to suit their customers' needs. The plan will determine how many people can stream Netflix content at once, and whether the subscriber can watch in Standard Definition (SD), High Definition (HD), or Ultra High Definition (UHD). As a Netflix subscriber, the customers get billed once a month on the data of their original sign up.

The following is the current organization chart of Netflix which includes its founder and CEO, Reed Hastings, and the executives directly reporting to him. Each executive has a manager in its corresponding field like Chief Marketing Officer would have a manager of market data analytics, Chief Content Officer would have a manger of the in-house production team, and so on.



Reed Hastings
CEO

Reed Hastings' Direct Reports

Click the to view the executive's direct reports

Kelly Bennett
Chief Marketing Officer

Jonathan Friedland
Chief Communications Officer

David Hyman
General Counsel

Jessica Neal
Chief Talent Officer

Greg Peters
Chief Product Officer

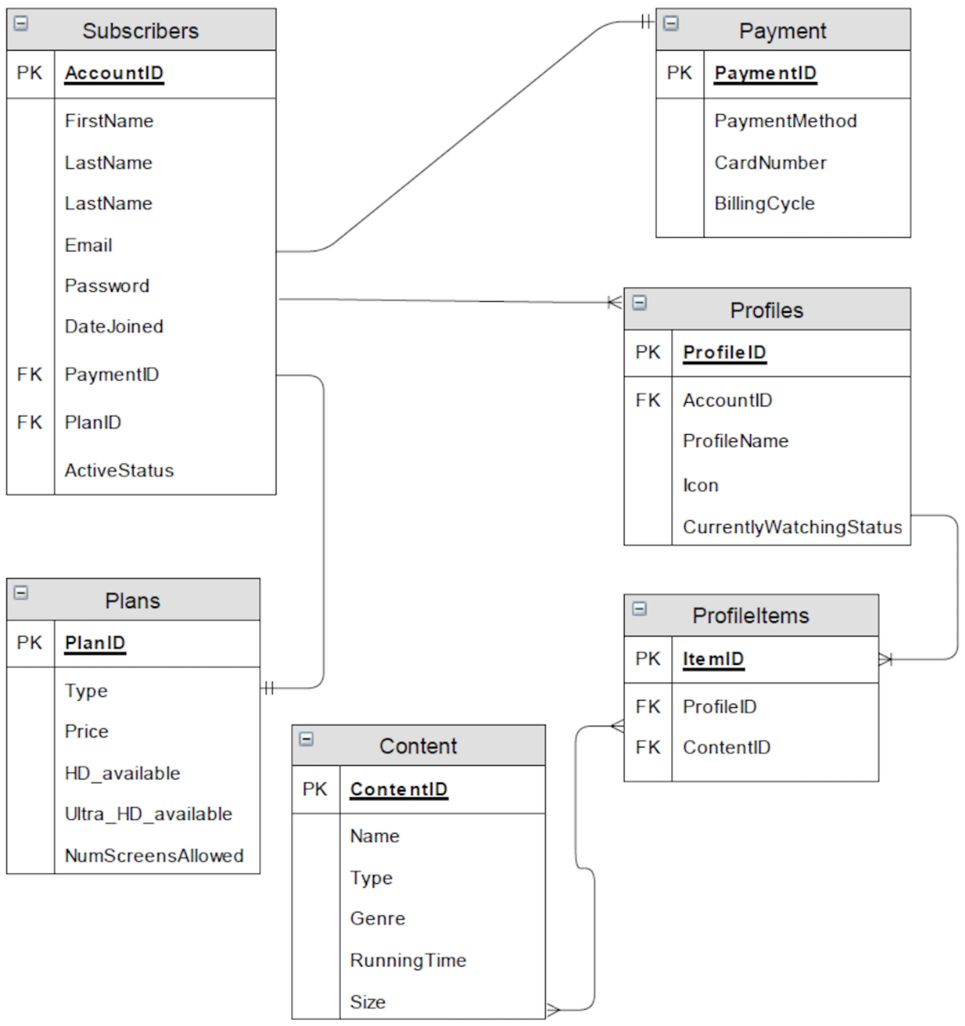
Ted Sarandos
Chief Content Officer

David Wells
Chief Financial Officer

Bill Holmes
Business Development

Database Design

- Logical Model: The following diagram is the logical model of the Netflix database. The focus will be on the U.S market only.



- Physical Model:

SUBSCRIBERS		
PK	AccountID	INT
	FirstName	VARCHAR(30)
	LastName	VARCHAR(30)
	Email	VARCHAR(60)
	Password	VARCHAR(60)
	DateJoined	DATE
FK	PaymentID	INT
	ActiveStatus	BIT
FK	PlanID	INT

PAYMENT		
PK	PaymentID	INT
	PaymentMethod	VARCHAR(30)
	CardNum	INT
	BillingCycle	INT

PLANS		
FK	PlanID	INT
	Type	VARCHAR(30)
	Price	MONEY
	HD_Available	BIT
	Ultra_HD_Available	BIT
	NumScreensAllowed	INT

CONTENT		
PK	ContentID	INT
	Name	VARCHAR(60)
	Type	VARCHAR(45)
	Genre	VARCHAR(25)
	RunningTime	TIME

PROFILES		
PK	ProfileID	INT
FK	AccountID	INT
	ProfileName	VARCHAR(30)
	Icon	VARCHAR(25)
	CurrentlyWatching	BIT

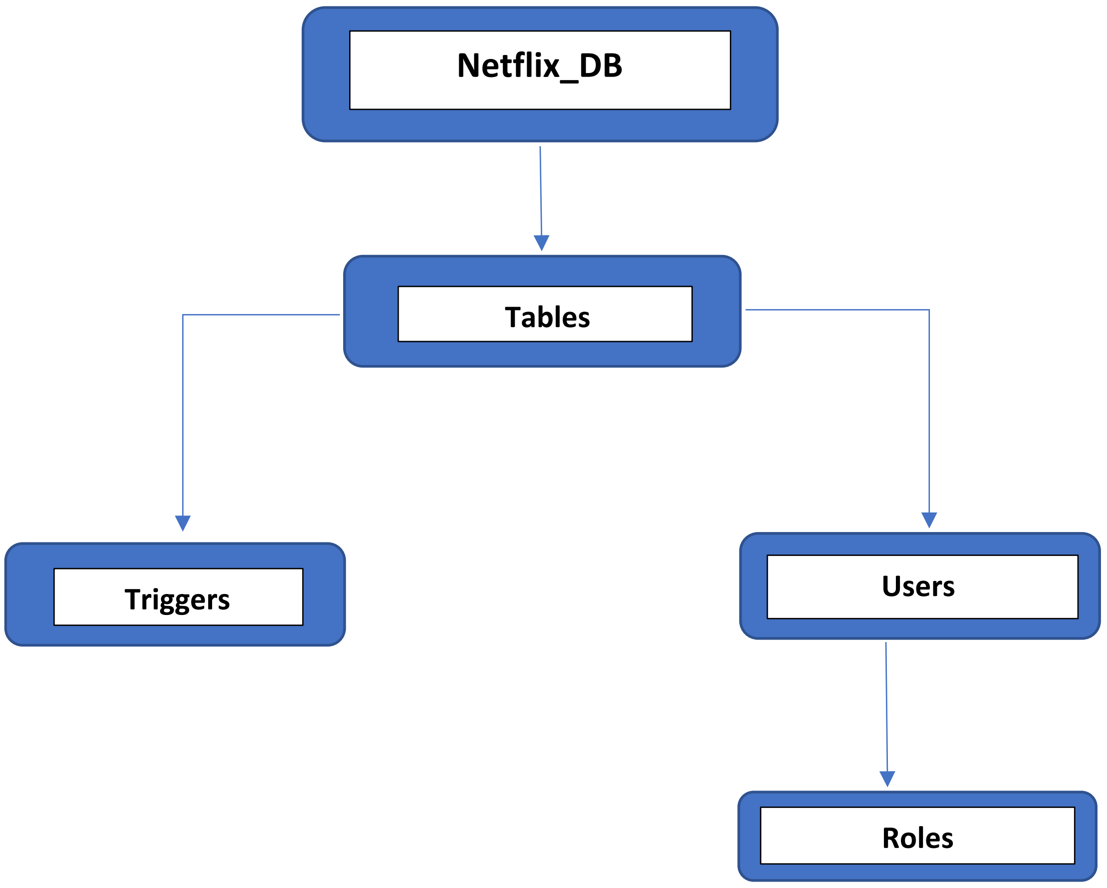
PROFILE_ITEMS		
PK	ItemID	INT
FK	ProfileID	INT
FK	ContentID	INT

Note: See Appendix A for the "CREATE TABLE" queries.

Note: See Appendix B for "SELECT" queries.

Note: "INSERT" queries are not available since the data was inserted using the wizard.

▪ Database Object Hierarchy:



▪ **Users and Permissions:**

User	Role
CEO	Db_Owner
Auditor	Db_datareader
IT manager	Db_ddladmin
App developer	Db_datareader
Db developer	Db_datawriter, Db_datareader
Network security manager	Db_securityadmin

- Storage:

A company like Netflix deals with terabytes of data on a daily basis. It would require a huge memory capacity to handle such amount of data. To store the content library that the company has, a cluster of hard drives with a total capacity of 500 TB would suffice.

Databases and data mining tools have a huge impact in the media and content industries, for both the company and the users. These industries use the advantage of being able to efficiently manage and store their content, as well as use tools towards more financial and analytical purposes such as advertisement or licensing.

We will use NoSQL Cassandra to store customer data, which is important to analyze to determine what shows would be best to license and personalize user experience. Customer data may include how many views certain shows receive, what are the most popular categories, what shows can be recommended to an individual user and even what is popular in certain regions.

- Hardware Configuration:

  - A cluster of **Stornado Turbo servers**:
    - CPU 2xE52620 v4
    - RAM: 32GB DDR4
    - Boot Drive: 250GB SSD

  - Power: The server at a minimum should have a dual power supplies connected to a UPS. For extra protection, it is good to have separate power feed as well.

- Government and Regulatory Compliance:
  - **SOX: Sarbanes-Oxley Act**: Since Netflix is a publicly traded company, it has to comply with SOX that establishes sweeping auditing and financial regulations.
  - **PCI-DSS: Payment Card Industry Data Security Standard:** Handling customers' payment information with integrity is vital. Validation of PCI-DSS compliance will be performed annually.
  - **Licensing Agreements:** Proper rights shall be acquired for the content stored in the database that is not produced by Netflix itself.

- Backup and Recovery Measures:
  - Recovery Model: **Full**
    - No data will be lost due to lost or damaged data files.
    - It will allow us to recover the data to an arbitrary point in time.
    - If the tail of the log is damaged, changes since the most recent log backup must be redone.
  - **Full database backup** shall be scheduled during the off-peak period, which is during 4:00 AM to 6:00 AM.
  - Also **schedule differential log backups**: It will reduce our restore time by reducing the number of log backups we have restore after restoring the data.
- Data Availability Plan
  - We will use **Failover Clustering** to provide high availability of the content to our customers. With database clustering, we can reach extremely high levels of availability due to its load balancing capabilities and distributed servers. In case a server got shut down the database will, however, be available.

- Database Performance Plan
  - Follow the Pareto Principle.

- **Monitor**: Scan the environment and review the output of the instrumentation facilities.
- **Analyze:** Determine solutions to any issues including Memory allocation, Logging options, I/O efficiency, and workload on the servers.
- **Optimization:**_Take corrective actions. Set an automation to make sure that appropriate measures are taken at the right time and add indexes at the avenues of improvement.

Database Maintenance Plan:

- Database shrinking: A shrink operation shall only be used after an operation that creates lots of unused space.
- WARNING: If you shrink a database repeatedly and notice that the database size grows again, this indicates that the space that was shrunk is required for regular operations. In these cases, repeatedly shrinking the database is a wasted operation.
- After the degree of fragmentation is known, use the following table to determine the best method to correct the fragmentation.

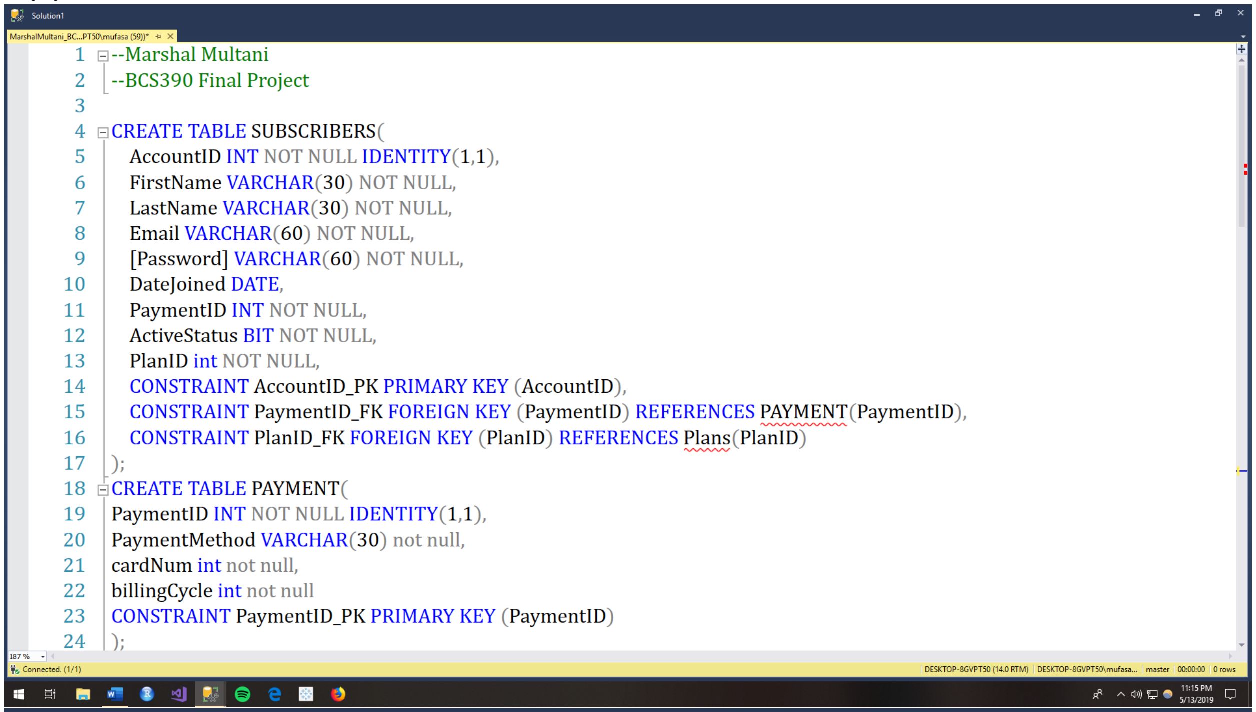
avg_fragmentation_in_percent **value**	**Corrective statement**
> 5% and < = 30%	ALTER INDEX REORGANIZE
> 30%	ALTER INDEX REBUILD WITH (ONLINE = ON)

- Indexes rebuild: Rebuilding an index can be executed online or offline. Reorganizing an index is always executed online. To achieve availability similar to the reorganize option, you should rebuild indexes online.
- Transaction locking shall be implemented to ensure the integrity of transactions in scenarios such as when multiple users are accessing/ modifying data at the same time.

- Database Integrity Plan:
    - **DB checking**: SQL Integrity Check will, on a regular basis, verify when the last known successful integrity check was performed. This helps identify any possible corruption early on, so you can act quickly instead of finding out a fix is needed after the corruption has grown even larger in scope.
    - Use **CHECKDB** for integrity check on SQL server**.**
    - The **DBCC** command can be used to monitor current memory allocation and usage.
- Disaster Recovery Plan:
    - Disaster Recovery Team Members and contacts:
        - Alvin Chipmunk – Ph#: 347-333-0898 Email- AlvinC@gmail
        - Simon Chipmunk- Ph#: 347-434-7927 Email- SimonC@gmail
        - Theodore Chipmunk- Ph# 516-909-1234 Email- TeddyC@ymail
    - Should the servers experience a failure for even a few ours, the company may expect a loss in millions of dollars in revenue. The cost of recovery would depend on the severity of the issue. Malicious attacks may prove to be much costlier than hardware or software failures. Hence, try to minimize the risk by reducing *threats* and *vulnerabilities* (targets of opportunity).
    - In case of loss of a personal, always make sure that not merely a single person is knowledgeable about a given task.
    - Levels of Criticality:
        - **Desirable**- plan to apply restoration on an appropriate date during off peak hours of operation.
        - **Essential**- schedule restoration on the as soon as possible, possibly the on the same day, during off peak hours
        - **Critical**- apply restoration immediately.
    - Schedule a self-audit for the database for security and compliance.

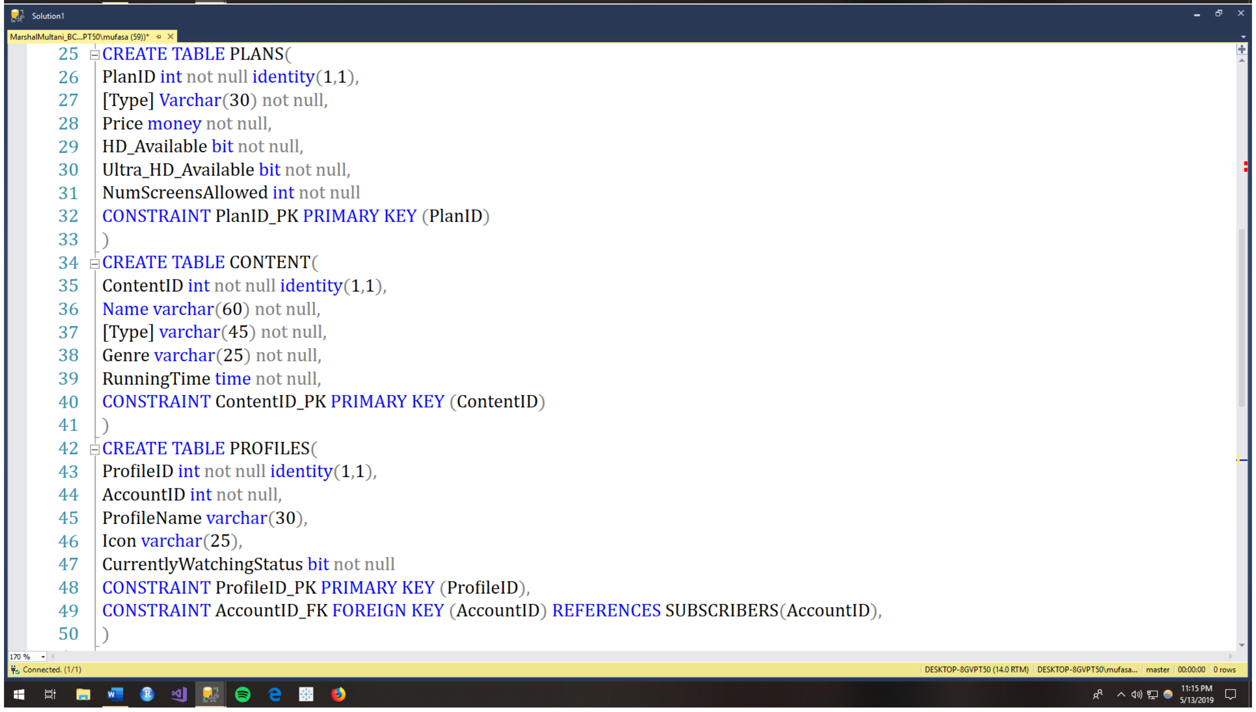
Appendices:

Appendix A:

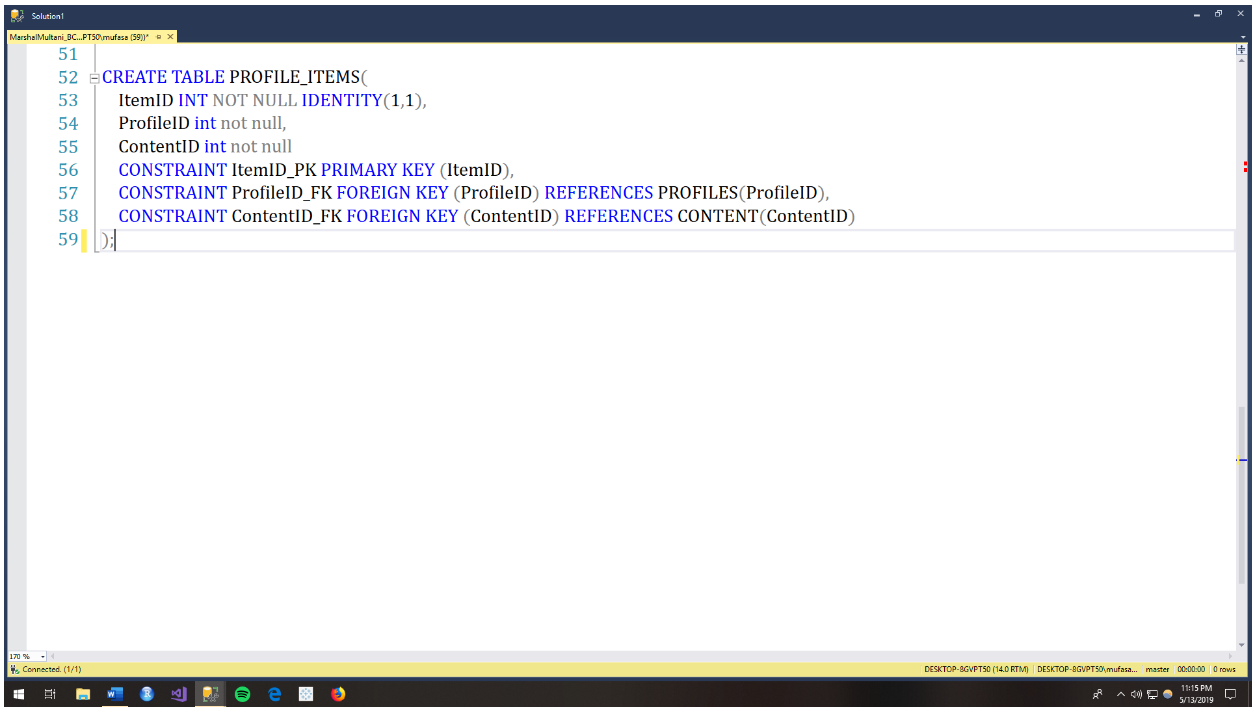


```
--Marshal Multani
--BCS390 Final Project

CREATE TABLE SUBSCRIBERS(
   AccountID INT NOT NULL IDENTITY(1,1),
   FirstName VARCHAR(30) NOT NULL,
   LastName VARCHAR(30) NOT NULL,
   Email VARCHAR(60) NOT NULL,
   [Password] VARCHAR(60) NOT NULL,
   DateJoined DATE,
   PaymentID INT NOT NULL,
   ActiveStatus BIT NOT NULL,
   PlanID int NOT NULL,
   CONSTRAINT AccountID_PK PRIMARY KEY (AccountID),
   CONSTRAINT PaymentID_FK FOREIGN KEY (PaymentID) REFERENCES PAYMENT(PaymentID),
   CONSTRAINT PlanID_FK FOREIGN KEY (PlanID) REFERENCES Plans(PlanID)
);
CREATE TABLE PAYMENT(
PaymentID INT NOT NULL IDENTITY(1,1),
PaymentMethod VARCHAR(30) not null,
cardNum int not null,
billingCycle int not null
CONSTRAINT PaymentID_PK PRIMARY KEY (PaymentID)
);
```



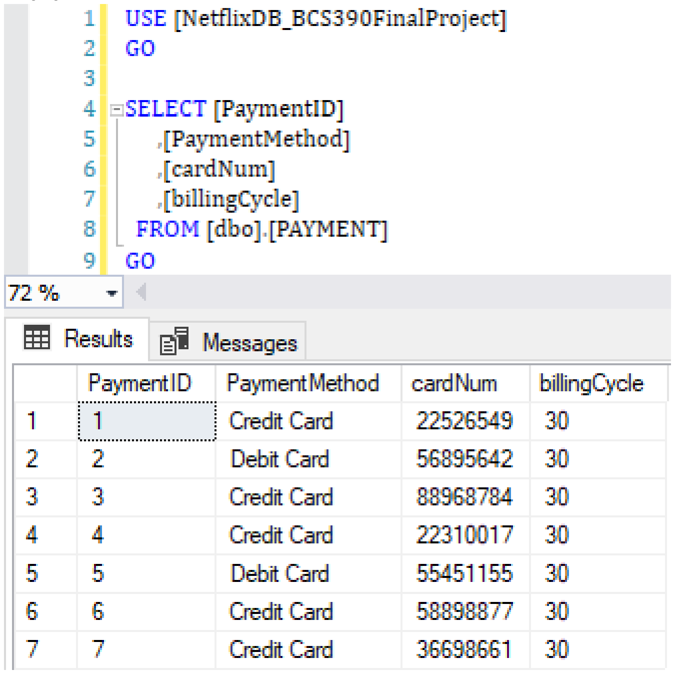
```
CREATE TABLE PLANS(
PlanID int not null identity(1,1),
[Type] Varchar(30) not null,
Price money not null,
HD_Available bit not null,
Ultra_HD_Available bit not null,
NumScreensAllowed int not null
CONSTRAINT PlanID_PK PRIMARY KEY (PlanID)
)
CREATE TABLE CONTENT(
ContentID int not null identity(1,1),
Name varchar(60) not null,
[Type] varchar(45) not null,
Genre varchar(25) not null,
RunningTime time not null,
CONSTRAINT ContentID_PK PRIMARY KEY (ContentID)
)
CREATE TABLE PROFILES(
ProfileID int not null identity(1,1),
AccountID int not null,
ProfileName varchar(30),
Icon varchar(25),
CurrentlyWatchingStatus bit not null
CONSTRAINT ProfileID_PK PRIMARY KEY (ProfileID),
CONSTRAINT AccountID_FK FOREIGN KEY (AccountID) REFERENCES SUBSCRIBERS(AccountID),
)
```



```sql

CREATE TABLE PROFILE_ITEMS(
    ItemID INT NOT NULL IDENTITY(1,1),
    ProfileID int not null,
    ContentID int not null
    CONSTRAINT ItemID_PK PRIMARY KEY (ItemID),
    CONSTRAINT ProfileID_FK FOREIGN KEY (ProfileID) REFERENCES PROFILES(ProfileID),
    CONSTRAINT ContentID_FK FOREIGN KEY (ContentID) REFERENCES CONTENT(ContentID)
);
```

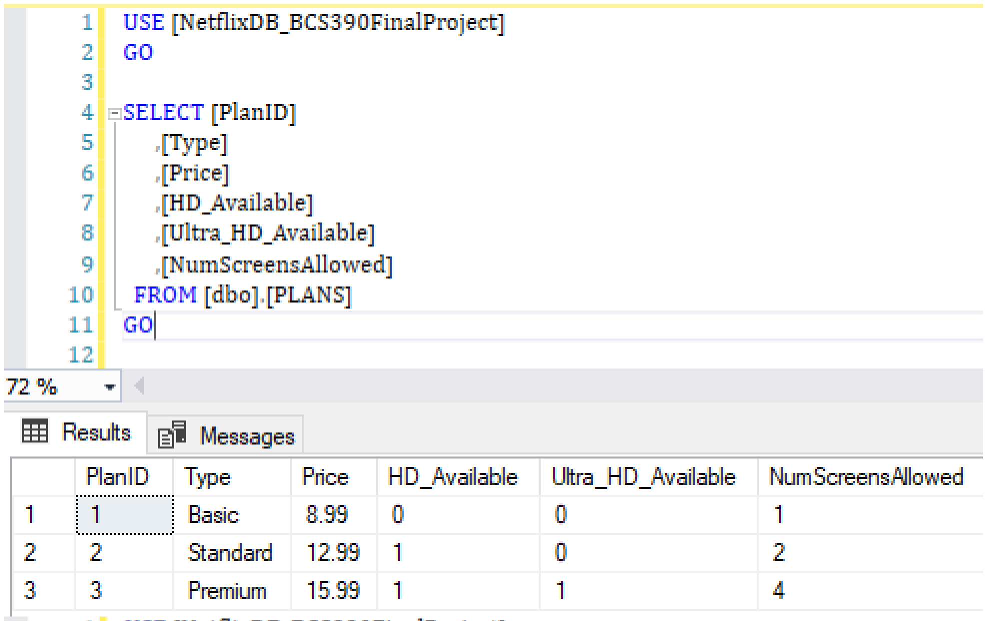
Appendix B:



```sql
USE [NetflixDB_BCS390FinalProject]
GO

SELECT [PaymentID]
      ,[PaymentMethod]
      ,[cardNum]
      ,[billingCycle]
  FROM [dbo].[PAYMENT]
GO
```

	PaymentID	PaymentMethod	cardNum	billingCycle
1	1	Credit Card	22526549	30
2	2	Debit Card	56895642	30
3	3	Credit Card	88968784	30
4	4	Credit Card	22310017	30
5	5	Debit Card	55451155	30
6	6	Credit Card	58898877	30
7	7	Credit Card	36698661	30



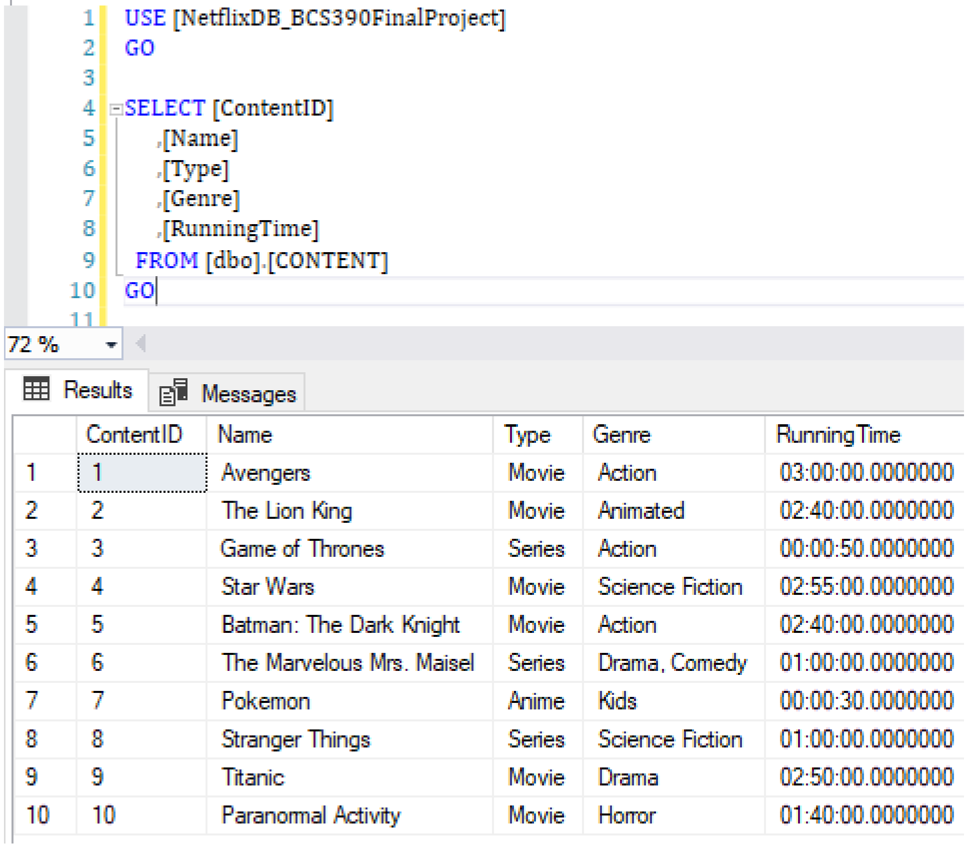
```sql
USE [NetflixDB_BCS390FinalProject]
GO

SELECT [PlanID]
      ,[Type]
      ,[Price]
      ,[HD_Available]
      ,[Ultra_HD_Available]
      ,[NumScreensAllowed]
  FROM [dbo].[PLANS]
GO
```

72 %

Results | Messages

	PlanID	Type	Price	HD_Available	Ultra_HD_Available	NumScreensAllowed
1	1	Basic	8.99	0	0	1
2	2	Standard	12.99	1	0	2
3	3	Premium	15.99	1	1	4



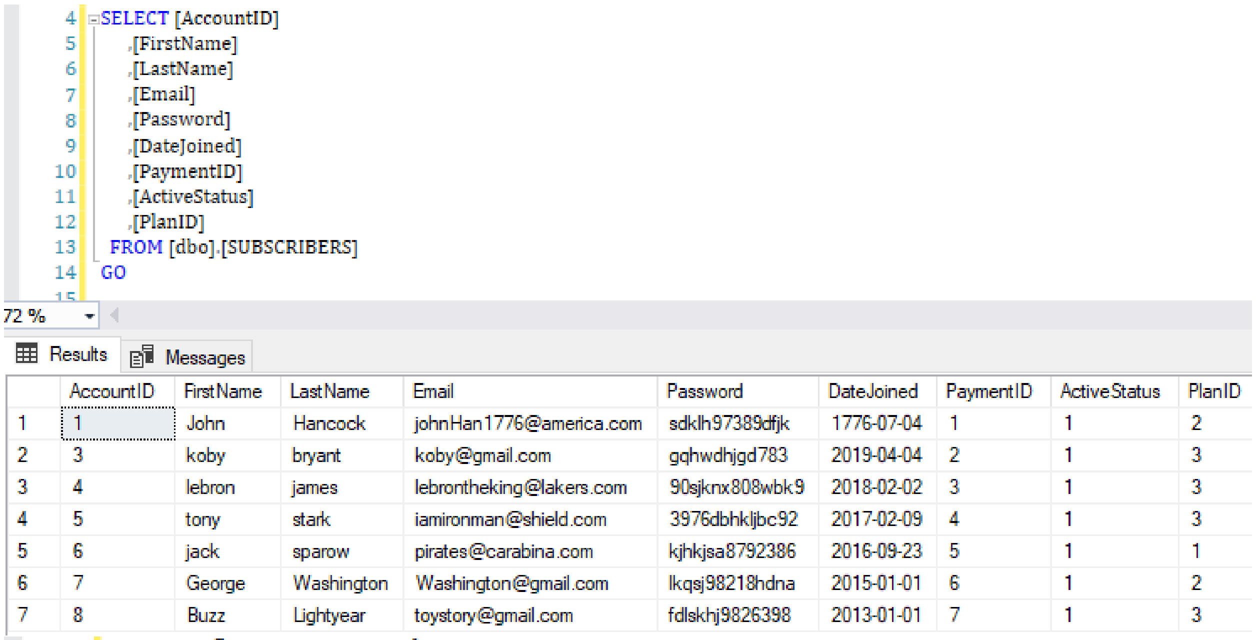
```sql
USE [NetflixDB_BCS390FinalProject]
GO

SELECT [ContentID]
      ,[Name]
      ,[Type]
      ,[Genre]
      ,[RunningTime]
  FROM [dbo].[CONTENT]
GO
```

72 %

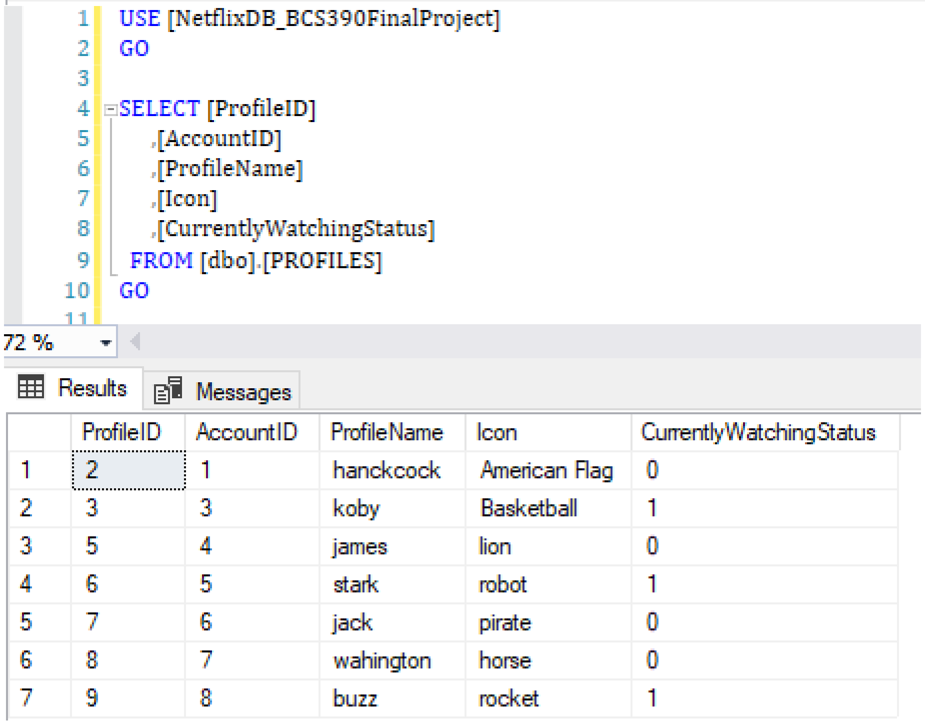
Results | Messages

	ContentID	Name	Type	Genre	RunningTime
1	1	Avengers	Movie	Action	03:00:00.0000000
2	2	The Lion King	Movie	Animated	02:40:00.0000000
3	3	Game of Thrones	Series	Action	00:00:50.0000000
4	4	Star Wars	Movie	Science Fiction	02:55:00.0000000
5	5	Batman: The Dark Knight	Movie	Action	02:40:00.0000000
6	6	The Marvelous Mrs. Maisel	Series	Drama, Comedy	01:00:00.0000000
7	7	Pokemon	Anime	Kids	00:00:30.0000000
8	8	Stranger Things	Series	Science Fiction	01:00:00.0000000
9	9	Titanic	Movie	Drama	02:50:00.0000000
10	10	Paranormal Activity	Movie	Horror	01:40:00.0000000



```sql
SELECT [AccountID]
      ,[FirstName]
      ,[LastName]
      ,[Email]
      ,[Password]
      ,[DateJoined]
      ,[PaymentID]
      ,[ActiveStatus]
      ,[PlanID]
  FROM [dbo].[SUBSCRIBERS]
GO
```

	AccountID	FirstName	LastName	Email	Password	DateJoined	PaymentID	ActiveStatus	PlanID
1	1	John	Hancock	johnHan1776@america.com	sdklh97389dfjk	1776-07-04	1	1	2
2	3	koby	bryant	koby@gmail.com	gqhwdhjgd783	2019-04-04	2	1	3
3	4	lebron	james	lebrontheking@lakers.com	90sjknx808wbk9	2018-02-02	3	1	3
4	5	tony	stark	iamironman@shield.com	3976dbhkljbc92	2017-02-09	4	1	3
5	6	jack	sparow	pirates@carabina.com	kjhkjsa8792386	2016-09-23	5	1	1
6	7	George	Washington	Washington@gmail.com	lkqsj98218hdna	2015-01-01	6	1	2
7	8	Buzz	Lightyear	toystory@gmail.com	fdlskhj9826398	2013-01-01	7	1	3



```sql
USE [NetflixDB_BCS390FinalProject]
GO

SELECT [ProfileID]
      ,[AccountID]
      ,[ProfileName]
      ,[Icon]
      ,[CurrentlyWatchingStatus]
  FROM [dbo].[PROFILES]
GO
```

	ProfileID	AccountID	ProfileName	Icon	CurrentlyWatchingStatus
1	2	1	hanckcock	American Flag	0
2	3	3	koby	Basketball	1
3	5	4	james	lion	0
4	6	5	stark	robot	1
5	7	6	jack	pirate	0
6	8	7	wahington	horse	0
7	9	8	buzz	rocket	1