Game Film Nation

Introduction To Database Programming Project

SUBMISSION 2

Group:

***Jack Teahan***

***Kenneth Malon***

***Kieran McClure***

***Jared McInerney***

**Introduction**

For the second project phase, we were required to further implement the features of our system in a more physical hands on approach, by creating the SQL script for our tables and for populating our tables with sample data.

Other files we needed to create are the triggers, sequences, views, indexes and stored procedures. Again the Whole team worked together on many parts of the project, but we all collaborated with each on the same triggers etc many times to speed up the process of solving the issues.

There is a main file for creating the database, and then there is files for populating, all the different triggers, procedures, sequences, indexes and views

**Database Changes:**

**~~----------------------------------------------------------------------------------------------------~~**

**OLD TABLE : ACCOUNT**

AccountID Numeric(8) auto-increment **PRIMARY KEY**,

Name char(30) NOT NULL,

Type char(1) NOT NULL,

Age Numeric(3),

Password char(15) NOT NULL,

Status char(1) NOT NULL,

Email char(30) NOT NULL,

Username char(30) UNIQUE NOT NULL,

MemberLevel char(15),

ReviewCount Numeric(7)

**UPDATED TABLE : ACCOUNT**

AccountID Numeric(8) auto-increment **PRIMARY KEY**,

Name char(30) **~~NOT NULL~~** ,

Type char(1**)** NOT NULL,

**DOB Numeric(3) NOT NULL**,

Password char(15) **~~NOT NULL~~**,

Status char(1) NOT NULL,

Email char(30) **UNIQUE** **~~NOT NULL~~**,

Username char(30) UNIQUE **~~NOT NULL~~**,

MemberLevel char(15), **DEFAULT('Copper'),**

~~ReviewCount Numeric(7)~~ **(MOVED TO REVIEW)**

**SavedSysTime Timestamp (ADDED TO ACCOUNT)**

**~~----------------------------------------------------------------------------------------------------~~**

**OLD TABLE : REVIEW**

ReviewID Numeric(8) auto-increment **PRIMARY KEY**,

Rating Numeric(1) NOT NULL,

Description char(200),

ReviewType char(15) NOT NULL,

Status char(1) NOT NULL,

AccountID numeric(8) NOT NULL,

MediaID numeric (8) NOT NULL,

**MediaID FOREIGN KEY references (Media),**

**AccountID FOREIGN KEY references (Account)**

**UPDATED TABLE : REVIEW**

ReviewID Numeric(8) auto-increment **PRIMARY KEY**,

Rating Numeric(1) NOT NULL,

Description char(200**) NOT NULL** ,

ReviewType char(15) NOT NULL,

Status char(1) NOT NULL,

AccountID numeric(8) NOT NULL,

MediaID numeric (8) NOT NULL,

**ReviewCount Numeric(7)** **(ADDED TO REVIEW)**

**MediaID FOREIGN KEY references (Media)**

**AccountID FOREIGN KEY references (Account)**

**~~----------------------------------------------------------------------------------------------------~~**

**OLD TABLE : MEDIA**

Media\_Id Numeric(8) auto-increment **PRIMARY KEY**,

Name char(40) NOT NULL,

Description char(200),

Year\_Released Numeric(4),

Age\_Rating char(5),

Budget Numeric(10),

Primary\_Genre char(15),

Secondary\_Genre char(15),

Media\_Type char(1) NOT NULL,

Platform char(15),

Average\_Rating Decimal(3),

**Media\_Cast\_Id FOREIGN KEY references (MediaCast)**

**UPDATED TABLE : MEDIA**

Media\_Id Numeric(8) **~~auto-increment~~** **NOT NULL PRIMARY KEY**,

Name char(40) NOT NULL,

Description char(200),

Year\_Released Numeric(4),

Age\_Rating char(5),

Budget Numeric(10),

Primary\_Genre char(15),

Secondary\_Genre char(15),

Media\_Type char(1) NOT NULL,

Platform char(15),

Average\_Rating Decimal(3) **DEFAULT(0)**,

**~~Media\_Cast\_Id FOREIGN KEY references (Media)~~**

**~~----------------------------------------------------------------------------------------------------~~**

**OLD TABLE : CAST**

Cast\_Id Numeric(8) auto-increment **PRIMARY KEY**,

Role char(20) NOT NULL,

EmployeeID numeric(8) NOT NULL

**Employee\_Id FOREIGN KEY references (Employee)**

**UPDATED TABLE : CAST**

Cast\_Id Numeric(8) **~~auto-increment~~** **NOT NULL PRIMARY KEY**,

**~~Role char(20) NOT NULL,~~**

EmployeeID numeric(8) NOT NULL,

**MediaID numeric(8),**

**Employee\_Id FOREIGN KEY references (Employee)**

**~~----------------------------------------------------------------------------------------------------~~**

**OLD TABLE : EMPLOYEE**

Employee\_Id Numeric(8) auto-increment **PRIMARY KEY**,

Birth\_Date Date,

Name char(20) NOT NULL,

Nationality char(20)

**UPDATED TABLE : EMPLOYEE**

Employee\_Id Numeric(8) **~~auto-increment~~** **PRIMARY KEY**,

**DOB** Date **NOT NULL**,

Name char(20) NOT NULL,

**Role char(20) NOT NULL,**

Nationality char(20) **NOT NULL**

***Dropped Table: MediaCast***

**Buisness Rules Implemented:**

**Account:**

* **Email address for your account must not exists in the database**
* **All fields must be entered**
* **Password must be between 5 and 15 character in length**
* **Name must be less than 30 characters long**
* **Username must be less than 30 characters long**
* **Username for your account must not exist in the database**
* **UserID must increment every time a new user is added to the system**
* **The User only has 5 attempts at login before he is locked out from the system for a day**

***(code implemented but no way of showing result as testing is impossible at the moment)***

* ***Added new Buisness rule to change member level based on amount of reviews made (This is arttempted but not fully working, code is shown later on)***

**Media:**

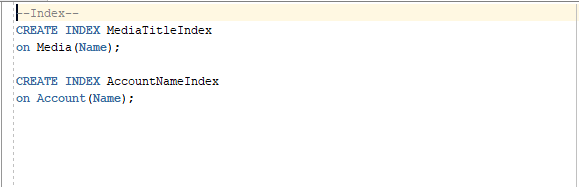
* **MediaID is incremented every time new media is added**
* **Title must be entered**
* **Title must be less than 40 characters long**
* **Description must be less than 200 characters long**
* **YearReleased must be less than4 characters long**
* **AgeRating must be less than 5 characters long**
* **Budget must be less than 10 characters long**
* **Budget must be numeric**
* **Platform must be less than 15 characters long**

**Employee:**

* **Cast Name must be less than 20 ccharacters long**
* **Cast DOB must be in date form (dd/mm/yyyy)**
* **Nationality must be less than 20 characters long**

**Indexes:**

Indexes were made for account and media tables to access these values quicker:

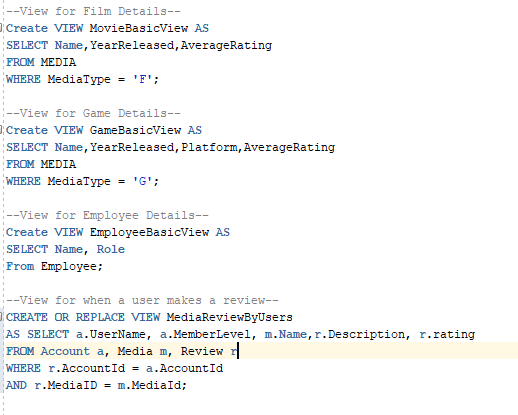


**Sequences:**

Sequences were made for all tables 

**Views made:**

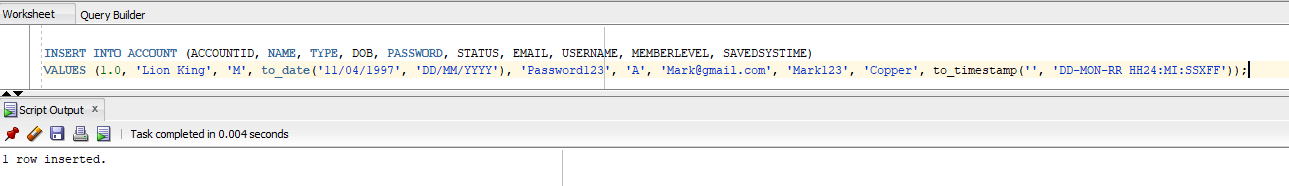
Sequences were made for all tables:



**Triggers made:**

***Register***

***This is a simple insert testing all works okay***

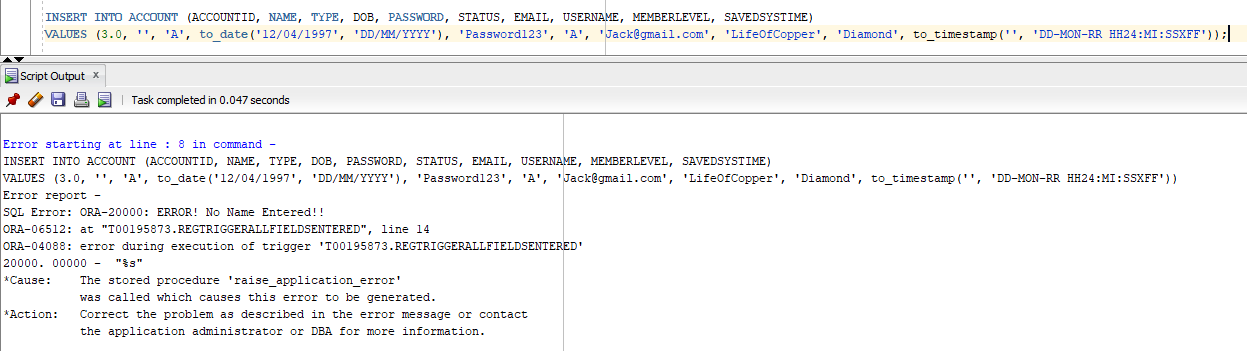


***(TriggerRegister.sql)***

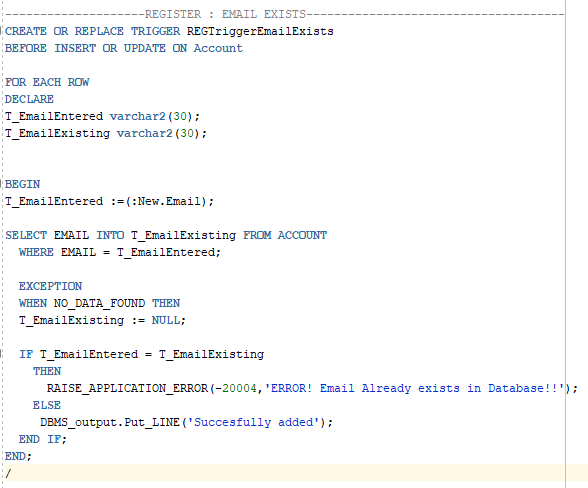
***This trigger checks if all fields are null and catches the error***



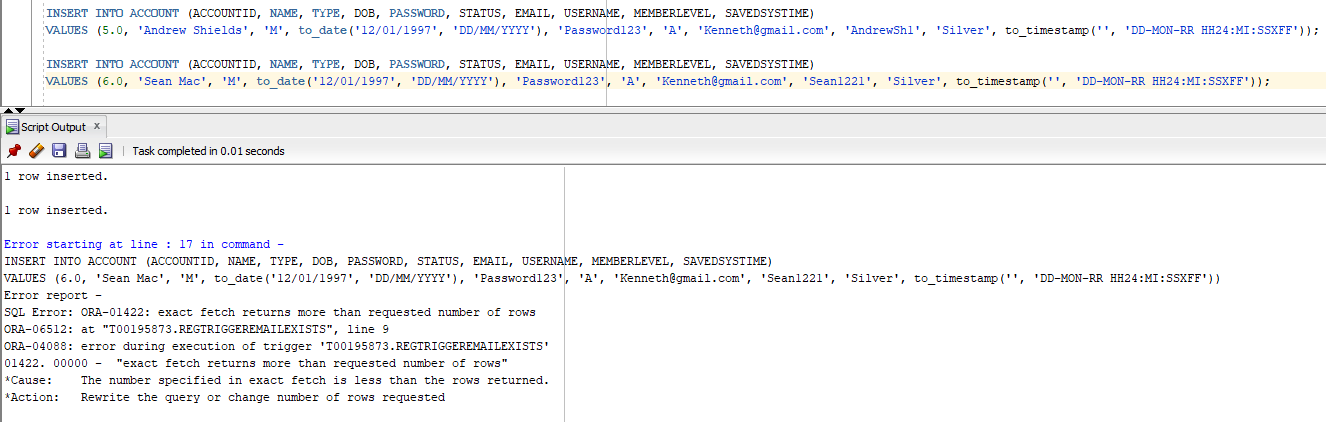
***Result:***



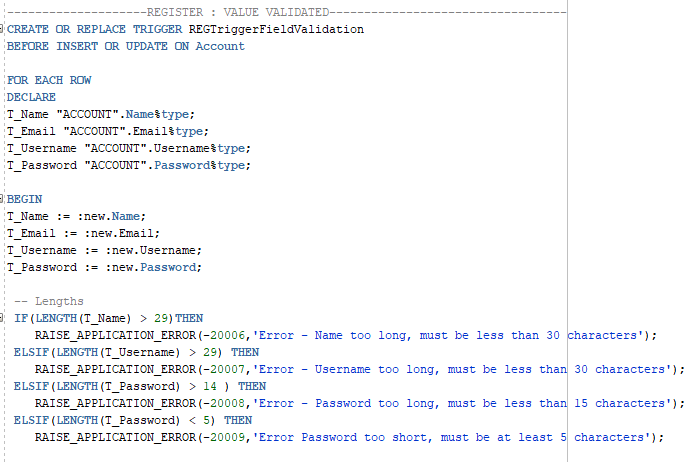
***This trigger checks if an email already exist in the database and shows appropriate error message***

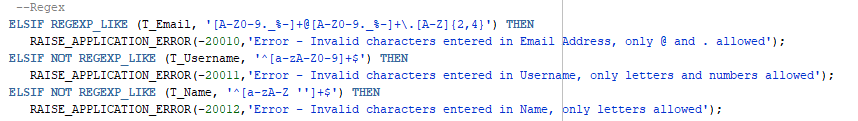


***Result:***

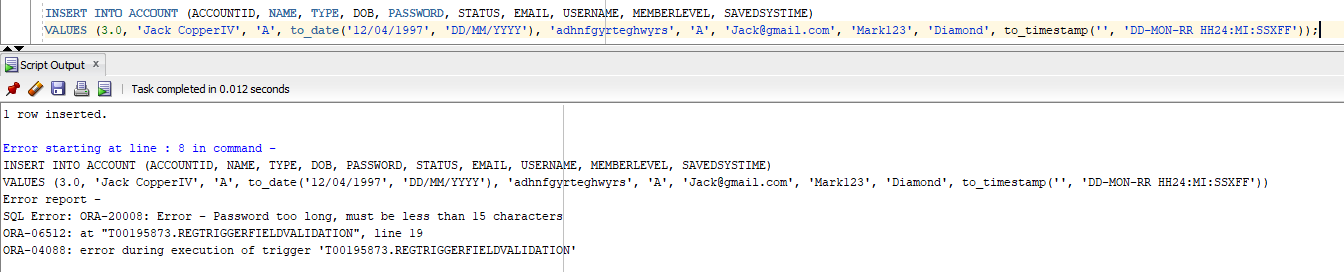


***This trigger checks if all fields are the correct length***

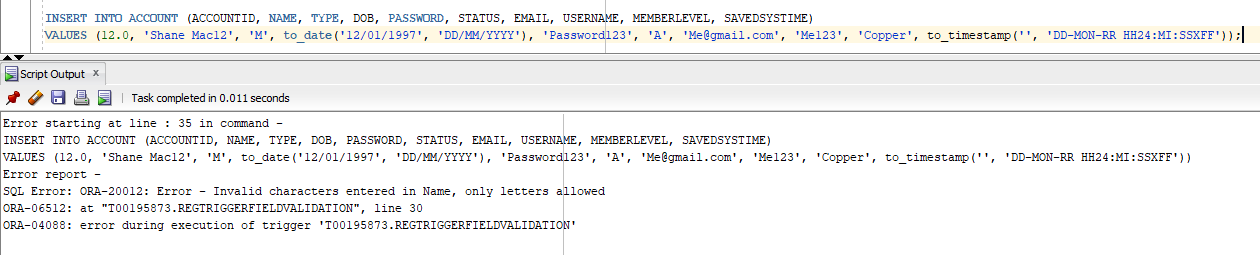


******

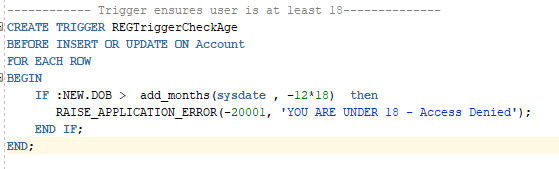
***Result for length:***

******

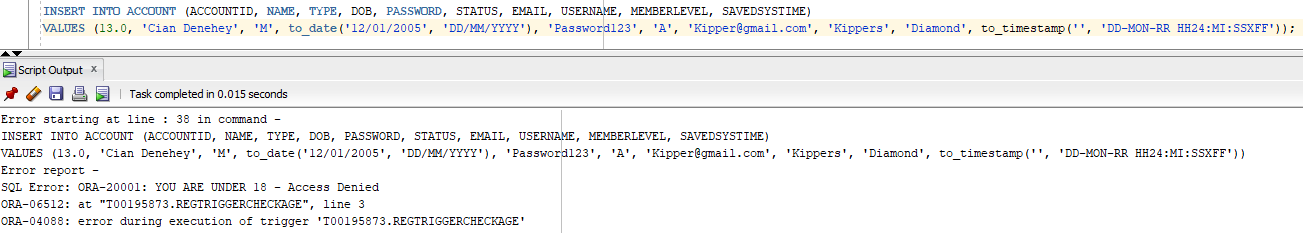
***Result for Regex:***

******

***This trigger checks if the member’s age is over 18 to register to the system***

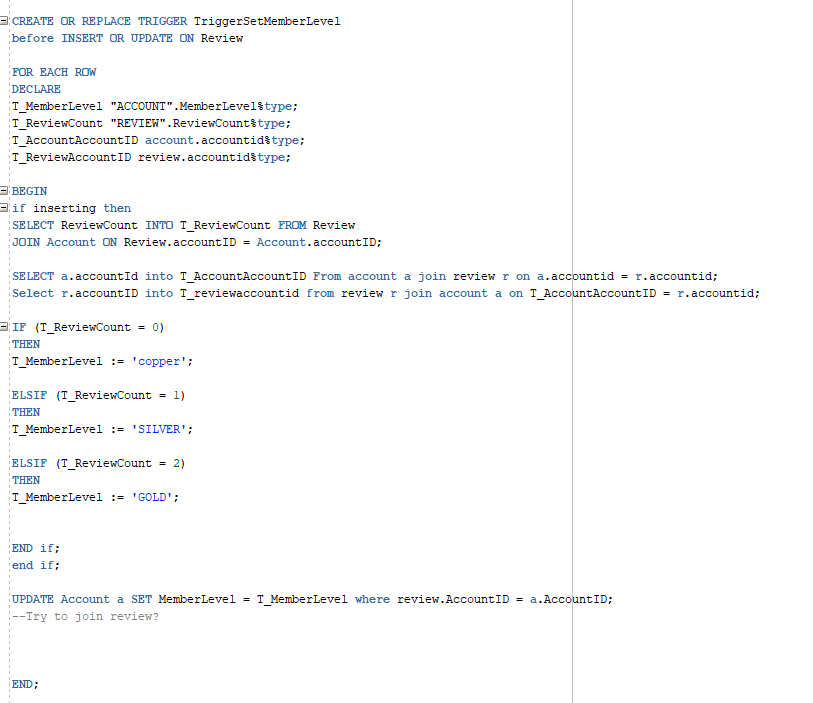


***Result:***



***This trigger is to update the members level based on the amount of reviews they have left***

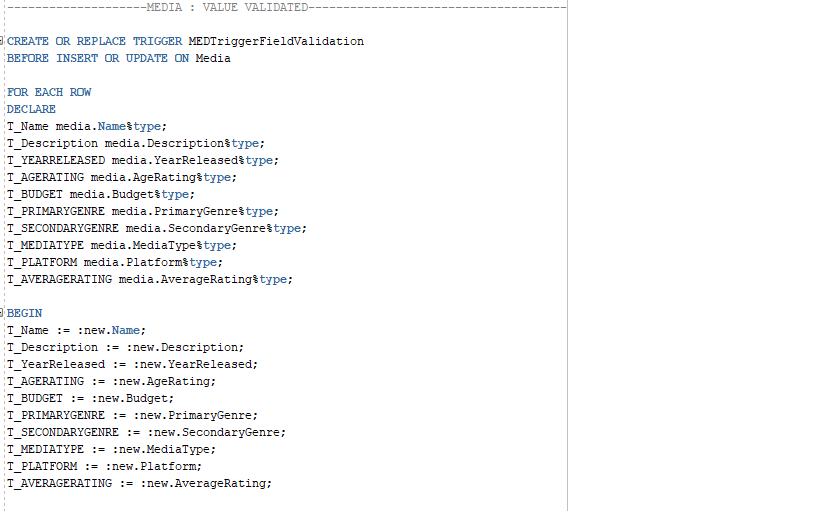
***(This trigger has been attempted but not implemented to the system as it was casuing an infinite loop because it kept being “updated, causing the trigger to keep running)***

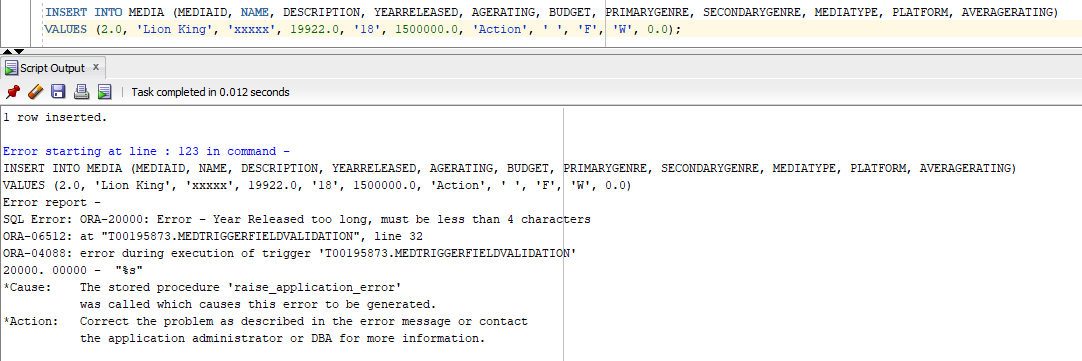


***Media***

***(TriggerMedia.sql)***

***This trigger checks if any media is too long or wrong format***

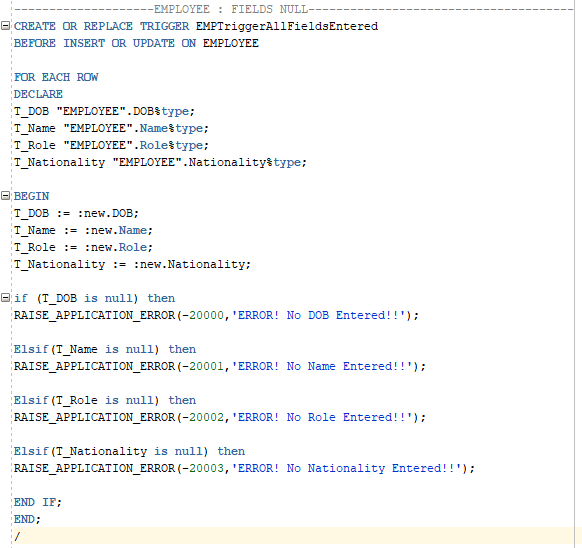
******

***Result for length:***

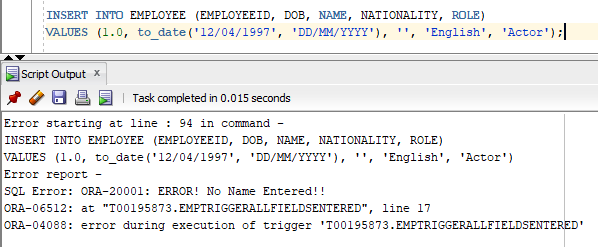
***Employee***

***(TriggerEmployee.sql)***

***This trigger checks for null entries***

******

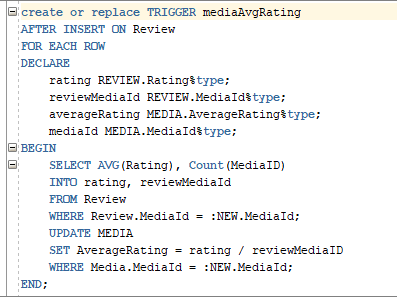
***Result:***

******

***INCOMPLETE TRIGGERS:***

***This Trigger calculates the average rating for media based on the ratings supplied by all the reviews stored***

***(this trigger was attempted but unfortunately did not work fully)***

******

***This trigger if a user fails to login after a total of 5 times, then theyr account gets locked for a certain amount of days***

***(In reflection this should more than likely be a stored procedure, but we can implement this in the future)***

***(This also was attempted but tnever implemented as there was no way of testing this code without a login GUI)***

******

***STORED PROCEDURES:***

***This procedure is used to search the database for a shortened set of details about a film or game based on the name of somebody who has worked in the cast.***

***For example, an implementation of this would allow somebody to see the average review score, name of film and genre of all films on the database featuring ‘Paul Rudd’.***

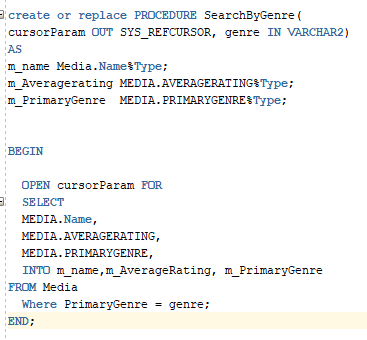
***Useful for Quick searches if the user doesn’t need a lot of information on the film except the essentials.***



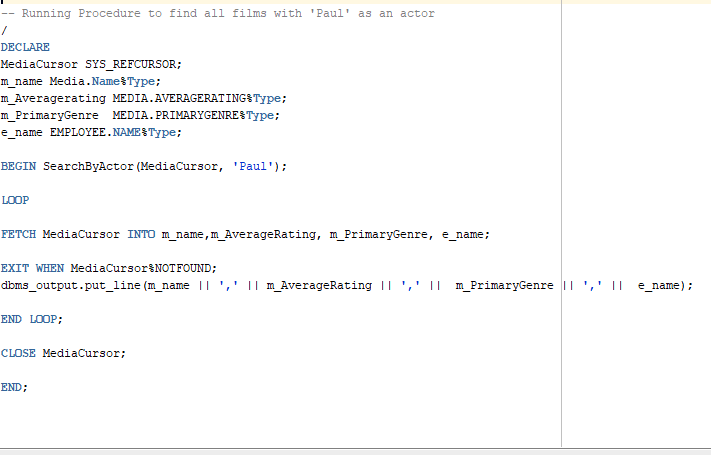
***This procedure is used to search the database for a shortened set of details about a film or game based on thegenre the user wishes to refine by.***

***For example, an implementation of this would allow somebody to see the average review score, name of film and genre of all films on the database that are classified as ‘Horror’.***

***Useful for Quick searches if the user doesn’t need a lot of information on the film except the essentials. – In rweflection, I would edit this procedure to allow for returning primary genres, then secomdary genres (It currently only looks for primary genre) along with allowing it to show highest Average Reviews first.***



This Piece of code tests that Search Actor works, It compiles however it doesn’t seem to return the result set to the DBMS output.

******

***CONCLUSION:***

After working on this stage of the project, our group found the implementation of the business rules pretty difficult. We spent a lot of our time producing pseudocode to tackle issues as we identified business rules. There were several business rules we only realisd as we began to produce code. The biggest challenge we found was communication, Using Oracle meant we didn’t have a common connection and we constantly has versions of scripts in use. We devised a system to divide up the development of scripts and they were sent into a group pool to keep everybody upto date with the current version of the database.

We were initially going to divide the group up so some people work on Triggers, some on Stored Procedures and Some on the more database side of things like creating populating scripts etc, but in general everybody helped everyone with their tasks so in the end the whole thing became one group effort, which was beneficial as some people had knowledge to help the other person based on problems they had faced in the past, and vice versa.