Prof. Jose Ortiz Costa

## **CSC 675 Final Project Report**

# 1. Limit Watchlist Capacity

This trigger was very hard since the rollup issue was an error I was never able to find a work around for. I tried switching the engine and also using a helper table and still was not able to accomplish the Trigger Requirement.

I tried to test on a limit of 5 and never could get the Trigger to function properly.

Prof. Jose Ortiz Costa

### A. Results of Testing

```
-- INSERTS FOR TRIGGER TEST 1.
INSERT INTO Watchlist (user, content) VALUES (user 1, content 1);
INSERT INTO Watchlist (user, content) VALUES (user 2,
                                                        content 2):
INSERT INTO Watchlist (user, content) VALUES (user 2,
                                                        content 3);
INSERT INTO Watchlist (user, content) VALUES (user 2,
                                                        content 4);
INSERT INTO Watchlist (user, content) VALUES (user 2,
                                                        content 5);
INSERT INTO Watchlist (user, content) VALUES (user 2,
                                                        content 2);
INSERT INTO Watchlist (user, content) VALUES (user 2,
                                                        content 6);
INSERT INTO Watchlist (user, content) VALUES (user 2,
                                                        content 7);
INSERT INTO Watchlist (user, content) VALUES (user 2,
                                                        content 1);
INSERT INTO Watchlist (user, content) VALUES (user 2,
                                                        content 8);
INSERT INTO Watchlist (user, content) VALUES (user 2,
                                                        content 9);
INSERT INTO Watchlist (user, content) VALUES (user 2,
                                                        content 10);
INSERT INTO Watchlist (user, content) VALUES (user 2,
                                                        content 11);
INSERT INTO Watchlist (user, content) VALUES (user 2,
                                                        content 12);
INSERT INTO Watchlist (user, content) VALUES (user 2,
                                                        content 13);
INSERT INTO Watchlist (user, content) VALUES (user 2,
                                                        content 14);
INSERT INTO Watchlist (user, content) VALUES (user 2,
                                                        content 15);
INSERT INTO Watchlist (user, content) VALUES (user 2,
                                                       content 16);
```

Prof. Jose Ortiz Costa

## **EXPECTED OUTPUT:**

WatchList stays at 50 only keeping the latest 50 Watchlist Items.

**ACTUAL OUTPUT: FAIL** 

## **B. Performance Insights**

My queries in this trigger are very short and require little to no runtime, even the query inside the trigger only has 1 join and is a simple Select Statement.

## C. Improvements and Recommendations

I think my queries are sufficient enough, this was a pretty simple Trigger.

Prof. Jose Ortiz Costa

### 2. Rating Impact on Content Availability

### A. Results of Testing

```
-- I first run this query to make sure all content is available at the start.

SELECT *

FROM Content

JOIN Content_Availability ON Content_Availability.content = Content.contentID

WHERE Content_Availability.availability != 1;

-- INPUT

INSERT INTO Review (user, content, rating_value, review_text)

VALUES ( user 2, content 1, rating_value 1, review_text 'Test.');

INSERT INTO Review (user, content, rating_value, review_text)

VALUES ( user 1, content 1, rating_value 1, review_text 'Test.');
```

Prof. Jose Ortiz Costa

# EXPECTED OUTPUT & ACTUAL OUTPUT: PASS



## **B. Performance Insights**

My queries in this trigger are very short and require little to no runtime, even the query inside the trigger only has 1 join and is a simple Select Statement.

# C. Improvements and Recommendations

I think my queries are sufficient enough, this was a pretty simple Trigger.

Prof. Jose Ortiz Costa

### 3. Ensure Unique Director for Content

My issue with this trigger was I wasn't able to fulfill both parts of the trigger. Either I was able to log the non-unique result into the log table but leave the row inside ContentDirectors. Or I was able to stop the insert from happening in ContentDirectors and not log the failed attempt in the log table.

I never reached a conclusion of how to finish either trigger.

### A. Results of Testing

```
-- TESTS FOR 3. X

INSERT INTO ContentDirectors (content, director) VALUES (content 1, director 1);

INSERT INTO ContentDirectors (content, director) VALUES (content 1, director 1);

SELECT * FROM ContentDirectors;

SELECT * FROM Director_Assignment_Errors;
```

Prof. Jose Ortiz Costa

#### **EXPECTED OUTPUT:**

LOG INTO Director\_Assignment\_Errors and Stop INSERT.

# **ACTUAL OUTPUT: FAIL**

### **B. Performance Insights**

I left my trigger to simply just stop the insert and not log so I would say the performance is quite efficient because any duplicate insert is automatically stopped.

### C. Improvements and Recommendations

I would love to improve the approach of the trigger since I wasn't able to fulfill the requirements.

Prof. Jose Ortiz Costa

# 4. Rank Top Genres By Watch Hours

A little disclaimer for testing this section, In my project I only allow a piece of content to be associated with one genre. So in my ReadData I assign each content a random genre, So the output for this function changes depending on what genres are associated with these content.

But I assign the "listed\_in" CSV values from Data.CSV as tags and associate that with Genre and Content.

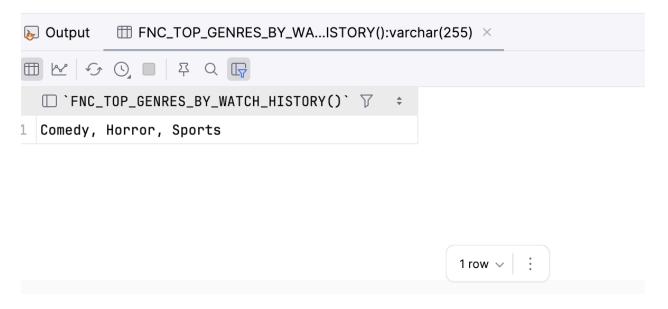
Prof. Jose Ortiz Costa

#### A. Results of Testing

```
-> Essential for 4. Rank Top Genres By Watch Hours Testing
   -> These inserts are only movies.
   NO WAY OF KNOWING THE DURATION OF A TV SHOW'S SEASONS.
   ONLY WAY YOU CAN PROPERLY MEASURE DURATION IS IF THE CONTENT IS A MOVIE.
   POPULATE THE Watch_History Table but make the date last month's for testing.
*/
-- WATCH_HISTORY INSERTS
INSERT INTO Watch_History (user, content, watch_date)
VALUES ( user 1, content 1, watch_date CURRENT_TIMESTAMP - INTERVAL 1 MONTH );
INSERT INTO Watch_History (user, content, watch_date)
VALUES ( user 2, content 12, watch_date CURRENT_TIMESTAMP - INTERVAL 1 MONTH );
INSERT INTO Watch_History (user, content, watch_date)
VALUES ( user 1, content 10, watch_date CURRENT_TIMESTAMP - INTERVAL 1 MONTH );
INSERT INTO Watch_History (user, content, watch_date)
VALUES ( user 3, content 16, watch_date CURRENT_TIMESTAMP - INTERVAL 1 MONTH );
INSERT INTO Watch_History (user, content, watch_date)
VALUES ( user 1, content 20, watch_date CURRENT_TIMESTAMP - INTERVAL 1 MONTH );
INSERT INTO Watch_History (user, content, watch_date)
VALUES ( user 2, content 21, watch_date CURRENT_TIMESTAMP - INTERVAL 1 MONTH );
INSERT INTO Watch_History (user, content, watch_date)
VALUES ( user 1, content 22, watch_date CURRENT_TIMESTAMP - INTERVAL 1 MONTH );
INSERT INTO Watch_History (user, content, watch_date)
VALUES ( user 3, content 23, watch_date CURRENT_TIMESTAMP - INTERVAL 1 MONTH );
INSERT INTO Watch_History (user, content, watch_date)
VALUES ( user 1, content 300, watch_date CURRENT_TIMESTAMP - INTERVAL 1 MONTH );
INSERT INTO Watch_History (user, content, watch_date)
VALUES ( user 2, content 301, watch_date CURRENT_TIMESTAMP - INTERVAL 1 MONTH );
INSERT INTO Watch_History (user, content, watch_date)
VALUES ( user 1, content 302, watch_date CURRENT_TIMESTAMP - INTERVAL 1 MONTH );
```

Prof. Jose Ortiz Costa

# EXPECTED OUTPUT & ACTUAL OUTPUT: PASS



### **B. Performance Insights**

My queries inside my function are very long with multiple JOINS. It is also 3 different SELECT Statements to get 1, 2, & 3 top genres. Which are then concat together and returned. Considering the fact the DB already controls thousands of lines of data it may be sufficient enough.

### C. Improvements and Recommendations

I think my approach can definitely be optimized better but I wanted to do it the way I already knew and focus on harder requirements.

Prof. Jose Ortiz Costa

### 5. Find Most Frequent Collaborators

This was one of the last requirements I completed since I was struggling where to start, it wasn't until I realized you can GROUP BY with 2 values.

Once I perfected that I was able to complete 5 and 11 which I struggled with as well.

### A. Results of Testing

### Input:

```
-$\bigsize 5.

SELECT FNC_MOST_FREQUENT_ACTOR_DIRECTORS();
```

# EXPECTED OUTPUT & ACTUAL OUTPUT: PASS

```
☐ `FNC_MOST_FREQUENT_ACTOR_DIRECTORS()` ▽ 

1 Top Actor, Director Pair: Julie Tejwani, Rajiv Chilaka
```

## **B. Performance Insights**

This function uses 2 SELECT statements where I get the top Actor from the duo. Then I get the director from the duo in the 2nd SELECT. I concat them and return.

## C. Improvements and Recommendations

I think my approach was the best for using a function since you are only

Prof. Jose Ortiz Costa

supposed to return 1 value in a function so you must workaround the proper way.

The way to improve this would be simply to use a procedure.

Prof. Jose Ortiz Costa

### 6. A. Results of Testing

#### MY WAY OF CHECKING FOR ANYTHING IN THE LAST MONTH:

# Table.Date + INTERVAL 1 MONTH > CURRENT\_TIMESTAMP;

```
DROP TRIGGER IF EXISTS TRG_AUTO_INSERT_TRANSACTIONS_FOR_SUBS$$
CREATE TRIGGER IF NOT EXISTS TRG_AUTO_INSERT_TRANSACTIONS_FOR_SUBS AFTER INSERT ON Transaction FOR EACH ROW
BEGIN
    DECLARE user_id INT DEFAULT 0;
    SELECT User.userID INTO user_id
    FROM Transaction
            JOIN Payment_Method ON Payment_Method.payment_methodID = Transaction.payment_method
             JOIN User ON Payment_Method.user = User.userID
    WHERE Payment_Method.payment_methodID = NEW.payment_method;
    IF NEW.status = 'successful' THEN
        -- ENTER the valid subscription using the sub they purchased.
       INSERT INTO User_Subscription (user, subscription)
       VALUES ( user user_id, NEW.subscription);
    END IF:
 PEND$$
DELIMITER;
```

Above is my Trigger that I implemented anytime a user purchases a subscription, after a Transaction is triggered I add the information to User\_Subscription.

Then depending on the time of the Transaction.Date I will validate if the subscription is active or expired.

Prof. Jose Ortiz Costa

### Input:

My queries inside the function are based upon if the latest transaction date for the user is more than a month old. So to test I add transactions with later dates and 2 with current time.

```
INSERT INTO Payment_Method (user, card_details) VALUES ( user 1, card_details 2772);
INSERT INTO Transaction ( payment_method, status, transaction_date)
VALUES ( payment_method 1, status 'successful', transaction_date CURRENT_TIMESTAMP - INTERVAL 2 MONTH);

INSERT INTO Payment_Method (user, card_details) VALUES ( user 2, card_details 2772);
INSERT INTO Transaction ( payment_method, status)
VALUES ( payment_method 2, status 'successful');

INSERT INTO Payment_Method (user, card_details) VALUES ( user 3, card_details 2772);
INSERT INTO Transaction ( payment_method, status, transaction_date)
VALUES ( payment_method 3, status 'successful', transaction_date CURRENT_TIMESTAMP - INTERVAL 2 MONTH);
INSERT INTO Payment_Method (user, card_details) VALUES ( user 4, card_details 2772);
INSERT INTO Transaction ( payment_method, status)
VALUES ( payment_method 4, status 'successful');
```

Prof. Jose Ortiz Costa

# **EXPECTED OUTPUT:**

User 1 subscription should expired.

User 2 subscription should be active.

ACTUAL OUTPUT: PASS

✓

<pre>□ `FNC_VALIDATE_USER_SUBSCRIPTION(1)` ♥</pre>	
1 User has an expired subscription.	
	1 row > :
☐ `FNC_VALIDATE_USER_SUBSCRIPTION(2)` ▽	<b>&gt;</b>
1 User has an active subscription.	
	1 row >   :

Prof. Jose Ortiz Costa

# **B. Performance Insights**

My function is just a SELECT statement with a few joins and returning a statement. This function will be impacted by events since subscriptions are removed in the event #10.

# C. Improvements and Recommendations

I think my function is pretty optimized.

Prof. Jose Ortiz Costa

### 7. Generate Monthly User Activity Report

### A. Results of Testing

```
-- WATCH_HISTORY INSERTS FOR PROCEDURE #7
INSERT INTO Watch_History (user, content, watch_date) VALUES
( user 1, content 1, watch_date CURRENT_TIMESTAMP - INTERVAL 1 DAY );
INSERT INTO Watch_History (user, content, watch_date) VALUES
   ( user 1, content 300, watch_date CURRENT_TIMESTAMP - INTERVAL 1 DAY );
INSERT INTO Watch_History (user, content, watch_date) VALUES
   ( user 2, content 303, watch_date CURRENT_TIMESTAMP - INTERVAL 1 DAY );
INSERT INTO Watch_History (user, content, watch_date) VALUES
   ( user 2, content 301, watch_date CURRENT_TIMESTAMP - INTERVAL 1 DAY );
-- REVIEW INSERTS (FIRST 2 FOR TRIGGER #2 TEST & LAST 2 FOR PROCEDURE #7.)
INSERT INTO Review (user, content, rating_value, review_text)
VALUES ( user 1, content 1, rating_value 2, review_text 'Boring.');
INSERT INTO Review (user, content, rating_value, review_text)
VALUES ( user 2, content 1, rating_value 2, review_text 'Boring.');
INSERT INTO Review (user, content, rating_value, review_text, review_date)
VALUES ( user 1, content 300, rating_value 6, review_text 'Decent.', review_date CURRENT_TIMESTAMP - INTERVAL 5 DAY);
INSERT INTO Review (user, content, rating_value, review_text, review_date)
VALUES ( user 2, content 300, rating_value 6, review_text 'Decent.', review_date CURRENT_TIMESTAMP - INTERVAL 5 DAY);
    ----- PROCEDURE TESTING BELOW -----
 -- 7.
CALL PRC_GENERATE_USER_REPORT( user_id 1);
```

Prof. Jose Ortiz Costa

# EXPECTED OUTPUT & ACTUAL OUTPUT: PASS



# **B. Performance Insights**

My procedure has 3 different selects where I insert Into 1 variable. No variables in these selects should be impacted by the current scheduled events. These values will change though since it depends on the user's activity.

# C. Improvements and Recommendations

My procedure has 3 different selects, so I can definitely optimize it better for bigger data.

Prof. Jose Ortiz Costa

### 8. Process Batch Content Updates

I struggled with this requirement because I did not understand what the Content\_Release table was. After asking Prof. Jose for help I understand what I must do.

My criteria for a Content\_Availability change is if the Content was released more than 5 years ago and if the Content has zero Watch\_History.

I created a function in java that inserted a watch history for half the content and left the other half unwatched.

### A. Results of Testing

### Input:

-- 8. Process Batch Content Updates
CALL PRC\_UPDATE\_CONTENT\_AVAILABILITY();

Prof. Jose Ortiz Costa

# **EXPECTED OUTPUT & ACTUAL OUTPUT: PASS**

	$\square$ content_availabilityID $ abla$	\$	$\square$ content $ eg$	\$	$\square$ availability $ abla$
1		2545		2545	AT RISK
2		2546		2546	S AT RISK
3		2547		2547	7 AT RISK
4		2548		2548	B AT RISK
5		2549		2549	AT RISK
6		2550		2550	O AT RISK
7		2551		2551	L AT RISK
8		2552		2552	2 AT RISK
9		2553		2553	3 AT RISK
10		2554		2554	AT RISK
11		2555		2555	5 AT RISK
12		2556		2556	AT RISK
13		2557		2557	7 AT RISK
14		2558		2558	B AT RISK
15		2559		2559	AT RISK
16		2560		2560	O AT RISK
17		2561		2561	L AT RISK
18		2562		2562	2 AT RISK
19		2563		2563	AT RISK
20		2564		2564	A AT RISK

# **B. Performance Insights**

My function was a little crazy this time around. I decided to get all Content that met my availability change criteria and use a for loop to assign the 'AT RISK' status.

This was a very simple task for me since I saw it in class but there is probably a better way to optimize this procedure.

# C. Improvements and Recommendations

Jaylin Jack

# CSC 675 - INTRO TO DATABASES

Prof. Jose Ortiz Costa

I know this procedure is more harsher than others on runtime so if I find a better way to run this I would gladly do that.

Prof. Jose Ortiz Costa

## 9. Handle Failed Payments

### A. Results of Testing

```
-- Transaction TABLE

DROP TABLE IF EXISTS Transaction;

CREATE TABLE IF NOT EXISTS Transaction

(
    transactionID INT PRIMARY KEY AUTO_INCREMENT,
    payment_method INT NOT NULL, -- FK
    subscription TINYINT NOT NULL, -- FK
    transaction_date DATETIME DEFAULT CURRENT_TIMESTAMP NOT NULL,
    status ENUM('successful', 'failed') NOT NULL,

CONSTRAINT fk_transaction_paymentMethod
    FOREIGN KEY (payment_method) REFERENCES Payment_Method(payment_methodID),
    CONSTRAINT fk_transaction_subscription
    FOREIGN KEY (subscription) REFERENCES Subscription_Plan(subscription_planID)
);
```

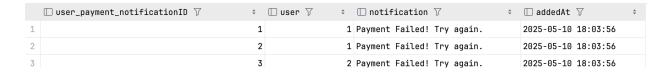
## Input:

Transaction.status is an Enum where 1= 'successful', 2 = 'failed'

```
-- FAILED TRANSACTION INSERTS FOR PROCEDURE #9
INSERT INTO Transaction (payment_method, subscription, status)
VALUES (payment_method 1, subscription 1, status 2);
INSERT INTO Transaction (payment_method, subscription, status)
VALUES (payment_method 1, subscription 1, status 2);
INSERT INTO Transaction (payment_method, subscription, status)
VALUES (payment_method 2, subscription 2, status 2);
```

Prof. Jose Ortiz Costa

# **EXPECTED OUTPUT & ACTUAL OUTPUT: PASS**



## **B. Performance Insights**

In this procedure I have 2 temporary tables, I know this impacts optimization & runtime. This is the best way I currently know how to achieve the requirements.

## C. Improvements and Recommendations

I think this procedure should have a trigger to know when the user has fixed their payment info to make the transaction not fail anymore.

Prof. Jose Ortiz Costa

# 10. Remove Expired Subscriptions

# A. Results of Testing

```
-- 10. Remove Expired Subscriptions 
-- Wait about 5 minutes from testing 6.
-- The Event is ran every 5 minutes so there's enough time to test 6. and 10.

SELECT ★ FROM User_Subscription;

SELECT ★ FROM User_Notification;
```

Prof. Jose Ortiz Costa

# EXPECTED OUTPUT & ACTUAL OUTPUT:PASS

As you can see user 1 is not inside User\_Subscription, along with user 3.



### Notify users



# **B. Performance Insights**

This requirement might have been one of the easiest. Simply delete the expired subs and insert what you deleted into the log table.

### C. Improvements and Recommendations

I don't know any way of optimizing this event any better. Maybe creating

Prof. Jose Ortiz Costa

## 11. Refresh Popular Content Rankings

Like I mentioned in Requirement 5, I had never used GROUP BY with 2 attributes until Data Grip autofilled it for me and the query worked.

But there was another issue before handling the query, I had to figure out how I would have 10 content per genre that have been watched. So in my ReadData file I did 2 INSERTS for Watch\_History for Half of the content list.

### A. Results of Testing

```
-- 11. Refresh Popular Content Rankings
```

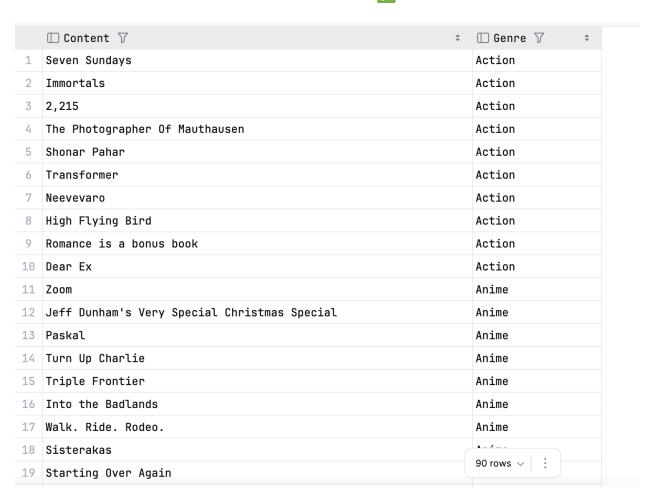
```
SELECT Content.title AS 'Content', Genre.description AS 'Genre'
FROM Popular_Content_And_Genres

JEIN Content ON Popular_Content_And_Genres.content = Content.contentID

JOIN Genre ON Popular_Content_And_Genres.genre = Genre.genreID;
```

Prof. Jose Ortiz Costa

## **EXPECTED OUTPUT & ACTUAL OUTPUT: PASS**



### **B. Performance Insights**

This event is a loop that gets the top 10 content for each genre so 10 SELECT statements are triggered daily. This event may cause a time discrepancy for a larger dataset.

### C. Improvements and Recommendations

I don't know of a better way to complete this requirement so if I find any better way I would do that.