Jaylin Jack

CSC 675 - INTRO TO DATABASES

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 we had to simply break the rules for. Instead of using a trigger we had the options to either change from MySQL or store the logic in a procedure.

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

Jaylin Jack

CSC 675 - INTRO TO DATABASES

Prof. Jose Ortiz Costa

A. Results of Testing: Input:

```
# -- INSERTS FOR TRIGGER TEST 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);
INSERT INTO Watchlist (user, content) VALUES ( user 2, content 17);
INSERT INTO Watchlist (user, content) VALUES ( user 2, content 18);
INSERT INTO Watchlist (user, content) VALUES ( user 3, content 7016)
INSERT INTO Watchlist (user, content) VALUES ( user 4, content 183);
INSERT INTO Watchlist (user, content) VALUES ( user 2, content 183);
INSERT INTO Watchlist (user, content) VALUES ( user 2, content 184);
INSERT INTO Watchlist (user, content) VALUES ( user 2, content 185);
INSERT INTO Watchlist (user, content) VALUES ( user 2, content 186);
INSERT INTO Watchlist (user, content) VALUES ( user 2, content 187);
INSERT INTO Watchlist (user, content) VALUES ( user 2, content 188);
INSERT INTO Watchlist (user, content) VALUES ( user 2, content 189);
INSERT INTO Watchlist (user, content) VALUES ( user 2, content 190);
```

Jaylin Jack

CSC 675 - INTRO TO DATABASES

Prof. Jose Ortiz Costa

EXPECTED OUTPUT & ACTUAL OUTPUT: PASS✅

As you can see the top elements are no longer at the top as there was 55 Watchlist entries for user 2.

	watchlistID	user	content	addedAt
1	6	2	7	2025-06-02 23:17:04
2	7	2	1	2025-06-02 23:17:04
3	8	2	8	2025-06-02 23:17:04
4	9	2	9	2025-06-02 23:17:04
5	10	2	10	2025-06-02 23:17:04
6	11	2	11	2025-06-02 23:17:04
7	12	2	12	2025-06-02 23:17:04
8	13	2	13	2025-06-02 23:17:04
9	14	2	14	2025-06-02 23:17:04
10	15	2	15	2025-06-02 23:17:04
11	16	2	16	2025-06-02 23:17:04
12	17	2	17	2025-06-02 23:17:04
13	18	2	18	2025-06-02 23:17:04
14	19	2	7016	2025-06-02 23:17:04
15	20	2	183	2025-06-02 23:17:04
16	21	2	183	2025-06-02 23:17:04
17	22	2	184	2025-06-02 23:17:04
18	23	2	185	2025-06-02 23:17:04
19	24	2	186	2025-06-02 23:17:04
20	25	2	187	2025-06-02 23:17:04
21	26	2	[illegible]	23:17:04

50 rows

Jaylin Jack

CSC 675 - INTRO TO DATABASES

Prof. Jose Ortiz Costa

B. Performance Insights

My queries in this procedure are very short and require little to no runtime.

C. Improvements and Recommendations

I think an honest improvement could be to use PostgreSQL to achieve it using triggers. I am actually going to update this procedure to be hosted by an event that runs through each user in the DB, that way you won't need to call it to remove older entries.

Jaylin Jack

CSC 675 - INTRO TO DATABASES

Prof. Jose Ortiz Costa

2. **Rating Impact on Content_Availability**

A. Results of Testing

Input:

```sql
-- 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.');
```

Jaylin Jack

CSC 675 - INTRO TO DATABASES

Prof. Jose Ortiz Costa

EXPECTED OUTPUT & ACTUAL OUTPUT: PASS✅

	`Content ID`	Title	status
1	1	Dick Johnson Is Dead	archived

1 row

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.

Jaylin Jack

CSC 675 - INTRO TO DATABASES

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

Input:

```
-- TESTS FOR 3. ✅
CALL PRC_INSERT_CONTENTDIRECTORS_OR_LOG_ERROR( content_id 4,  director_id 2);
SELECT * FROM Content WHERE contentID = 4;
CALL PRC_INSERT_CONTENTDIRECTORS_OR_LOG_ERROR( content_id 4,  director_id 2);
SELECT * FROM Content WHERE contentID = 4;
 💡
SELECT * FROM Director_Assignment_Errors;
```

Jaylin Jack

CSC 675 - INTRO TO DATABASES

Prof. Jose Ortiz Costa

EXPECTED OUTPUT & ACTUAL OUTPUT: PASS✅

LOG INTO Director_Assignment_Errors and Stop INSERT.

	FNC_RETURN_CONTENT_DIRECTOR_ERROR(error_msg)
1	Director can not be associated with the same content twice.

	director_assignment_errorID	content	director	createdAt
1	1	4	2	2025-06-02 23:23:44

Jaylin Jack

CSC 675 - INTRO TO DATABASES

Prof. Jose Ortiz Costa

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.

Jaylin Jack

CSC 675 - INTRO TO DATABASES

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.

Jaylin Jack

CSC 675 - INTRO TO DATABASES

Prof. Jose Ortiz Costa

A. Results of Testing

Input:

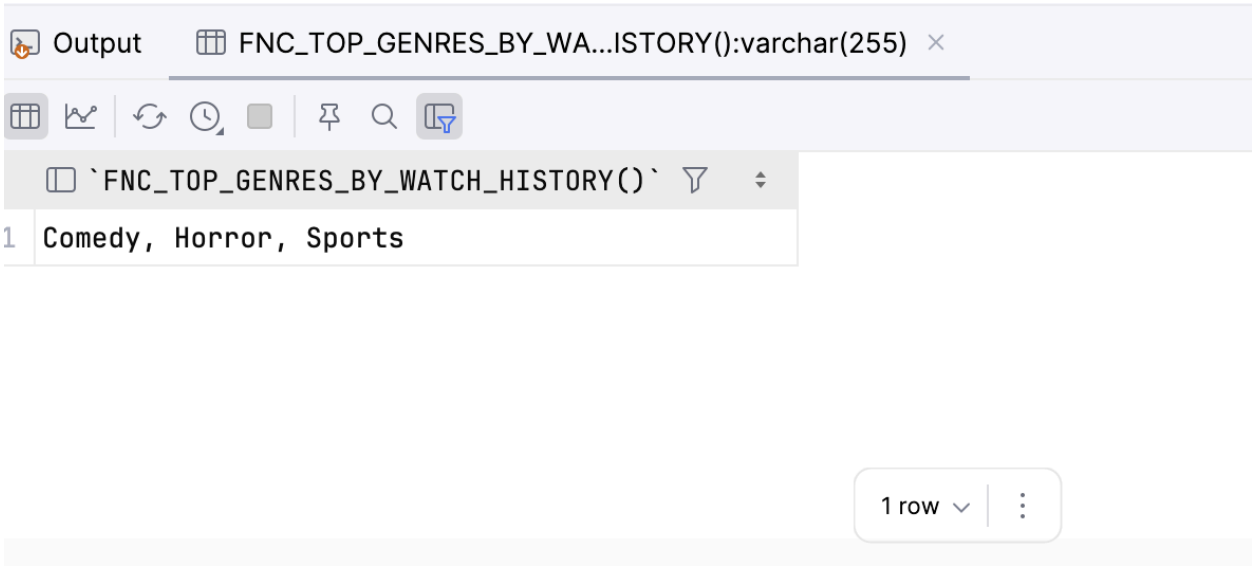
```
/*
    -> Essential for 4. Rank Top Genres By Watch Hours Testing
    -> These inserts are only movies.
    NO WAY OF KNOWING THE DURATION OF A TV SHOW'S SEASONS.
    ONLY WAY YOU CAN PROPERLY MEASURE DURATION IS IF THE CONTENT IS A MOVIE.

    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 );
```

Jaylin Jack

CSC 675 - INTRO TO DATABASES

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.

Jaylin Jack

CSC 675 - INTRO TO DATABASES

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:

```
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

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.

Jaylin Jack

CSC 675 - INTRO TO DATABASES

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;


    END$$

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.

Jaylin Jack

CSC 675 - INTRO TO DATABASES

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');
```

Jaylin Jack

CSC 675 - INTRO TO DATABASES

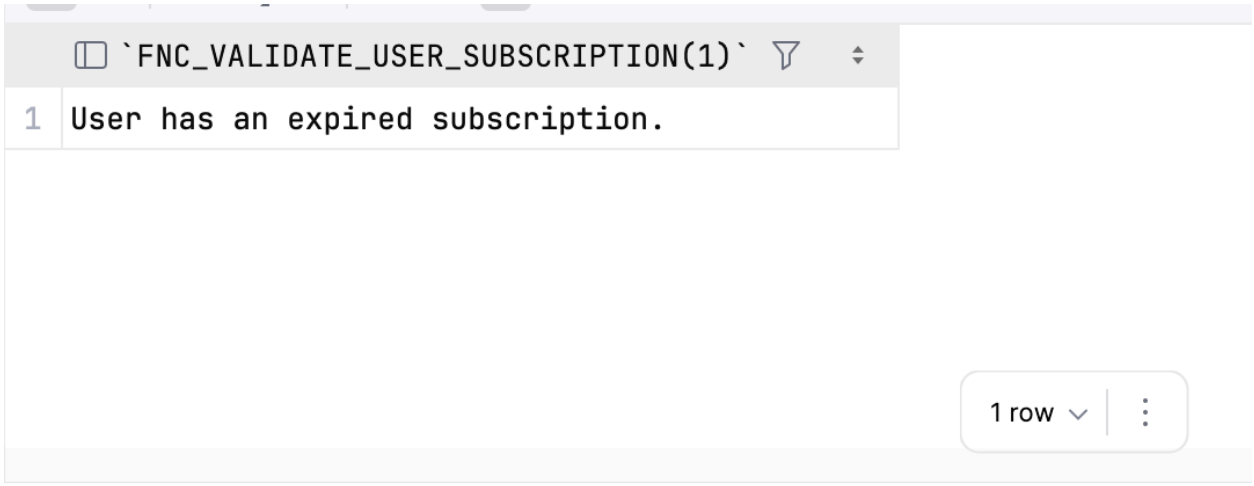
Prof. Jose Ortiz Costa

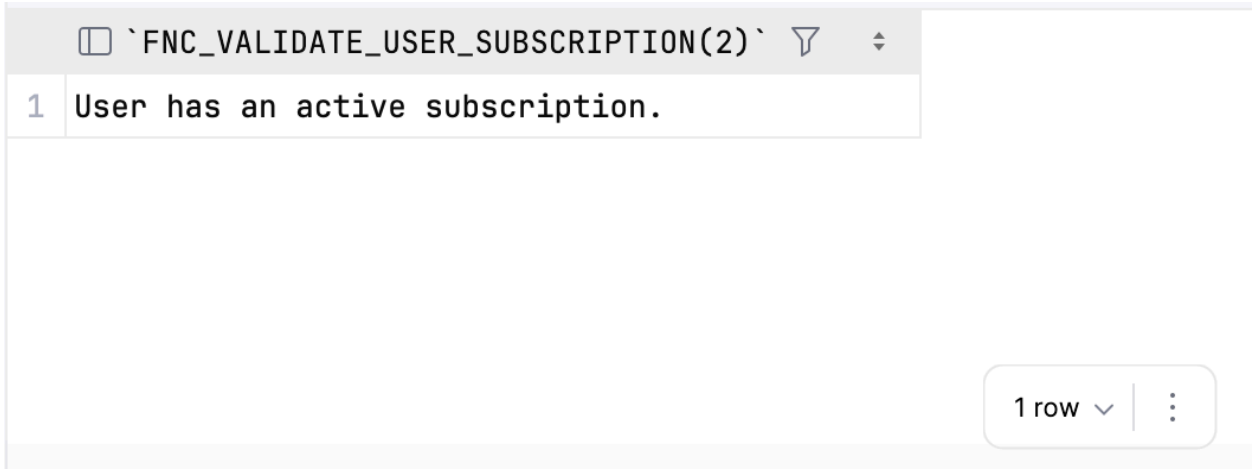
EXPECTED OUTPUT:

User 1 subscription should expired.

User 2 subscription should be active.

ACTUAL OUTPUT: PASS✅





Jaylin Jack

CSC 675 - INTRO TO DATABASES

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.

Jaylin Jack

CSC 675 - INTRO TO DATABASES

Prof. Jose Ortiz Costa

7. **Generate Monthly User Activity Report**

A. Results of Testing

Input:

```
-- 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);
```

Jaylin Jack

CSC 675 - INTRO TO DATABASES

Prof. Jose Ortiz Costa

EXPECTED OUTPUT & ACTUAL OUTPUT: PASS✅

	user_reportID	user	content_watched	avg_ratings	hours_watched	addedAt
1	1	1	2	3.50	3.20	2025-05-10 17:54:34

1 row

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.

Jaylin Jack

CSC 675 - INTRO TO DATABASES

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();
```

Jaylin Jack

CSC 675 - INTRO TO DATABASES

Prof. Jose Ortiz Costa

EXPECTED OUTPUT & ACTUAL OUTPUT: PASS✅

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

Jaylin Jack

CSC 675 - INTRO TO DATABASES

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);
```

Jaylin Jack

CSC 675 - INTRO TO DATABASES

Prof. Jose Ortiz Costa

EXPECTED OUTPUT & ACTUAL OUTPUT: PASS✅

	user_payment_notificationID	user	notification	addedAt
1	1	1	Payment Failed! Try again.	2025-05-10 18:03:56
2	2	1	Payment Failed! Try again.	2025-05-10 18:03:56
3	3	2	Payment Failed! Try again.	2025-05-10 18:03:56

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.

Jaylin Jack

CSC 675 - INTRO TO DATABASES

Prof. Jose Ortiz Costa

10. **Remove Expired Subscriptions**

A. Results of Testing

Input:

```
-- 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;
```

Jaylin Jack

CSC 675 - INTRO TO DATABASES

Prof. Jose Ortiz Costa

EXPECTED OUTPUT & ACTUAL OUTPUT:PASS✅

As you can see user 1 is not inside User_Subscription, along with user 3.

	user	subscription	status
1	2	3	active
2	4	3	active

Notify users

	user_notificationID	user	notification	addedAt
1	1	1	Your subscription has expired. Please Renew!	2025-05-12 09:17:20
2	2	3	Your subscription has expired. Please Renew!	2025-05-12 09:17:20

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

Jaylin Jack

CSC 675 - INTRO TO DATABASES

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

Input:

```
-- 11. Refresh Popular Content Rankings
SELECT Content.title AS 'Content', Genre.description AS 'Genre'
FROM Popular_Content_And_Genres
JOIN Content ON Popular_Content_And_Genres.content = Content.contentID
JOIN Genre ON Popular_Content_And_Genres.genre = Genre.genreID;
```

Jaylin Jack

CSC 675 - INTRO TO DATABASES

Prof. Jose Ortiz Costa

EXPECTED OUTPUT & ACTUAL OUTPUT: PASS✅

	Content	Genre
1	Seven Sundays	Action
2	Immortals	Action
3	2,215	Action
4	The Photographer Of Mauthausen	Action
5	Shonar Pahar	Action
6	Transformer	Action
7	Neevevaro	Action
8	High Flying Bird	Action
9	Romance is a bonus book	Action
10	Dear Ex	Action
11	Zoom	Anime
12	Jeff Dunham's Very Special Christmas Special	Anime
13	Paskal	Anime
14	Turn Up Charlie	Anime
15	Triple Frontier	Anime
16	Into the Badlands	Anime
17	Walk. Ride. Rodeo.	Anime
18	Sisterakas	
19	Starting Over Again	

90 rows

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.