# Analysis

## RESULTS OF TESTING

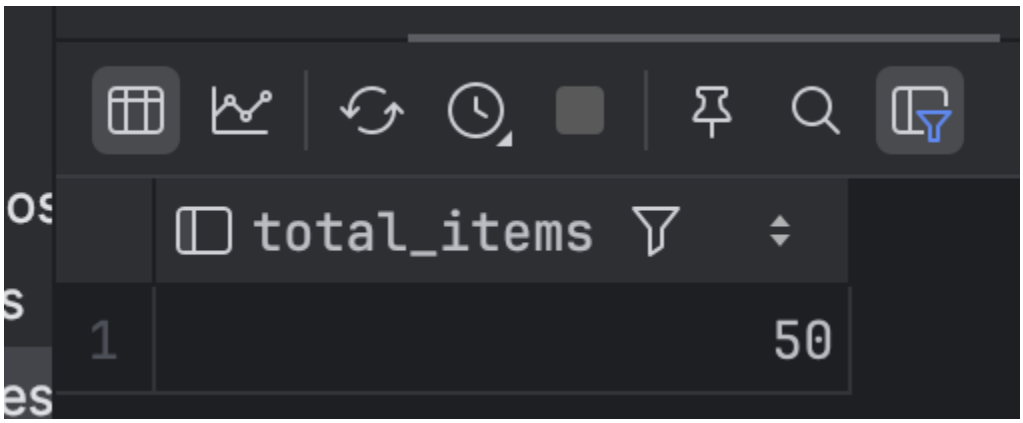
1. **TRG_WatchList_Cap**

**Unit Testing**

**Expected: count = 50**

```
-- Insert 50 watchlist entries for idUser = 1
INSERT INTO Watchlist (idUser, idContent, time_added) VALUES ( idUser 1, idContent 1001, time_added '2024-01-01 01:00:00');
INSERT INTO Watchlist (idUser, idContent, time_added) VALUES ( idUser 1, idContent 1002, time_added '2024-01-01 01:01:00');
INSERT INTO Watchlist (idUser, idContent, time_added) VALUES ( idUser 1, idContent 1003, time_added '2024-01-01 01:02:00');
INSERT INTO Watchlist (idUser, idContent, time_added) VALUES ( idUser 1, idContent 1004, time_added '2024-01-01 01:03:00');
INSERT INTO Watchlist (idUser, idContent, time_added) VALUES ( idUser 1, idContent 1005, time_added '2024-01-01 01:04:00');
INSERT INTO Watchlist (idUser, idContent, time_added) VALUES ( idUser 1, idContent 1006, time_added '2024-01-01 01:05:00');
INSERT INTO Watchlist (idUser, idContent, time_added) VALUES ( idUser 1, idContent 1007, time_added '2024-01-01 01:06:00');
INSERT INTO Watchlist (idUser, idContent, time_added) VALUES ( idUser 1, idContent 1008, time_added '2024-01-01 01:07:00');
INSERT INTO Watchlist (idUser, idContent, time_added) VALUES ( idUser 1, idContent 1009, time_added '2024-01-01 01:08:00');
INSERT INTO Watchlist (idUser, idContent, time_added) VALUES ( idUser 1, idContent 1010, time_added '2024-01-01 01:09:00');
INSERT INTO Watchlist (idUser, idContent, time_added) VALUES ( idUser 1, idContent 1011, time_added '2024-01-01 01:10:00');
INSERT INTO Watchlist (idUser, idContent, time_added) VALUES ( idUser 1, idContent 1012, time_added '2024-01-01 01:11:00');
INSERT INTO Watchlist (idUser, idContent, time_added) VALUES ( idUser 1, idContent 1013, time_added '2024-01-01 01:12:00');
INSERT INTO Watchlist (idUser, idContent, time_added) VALUES ( idUser 1, idContent 1014, time_added '2024-01-01 01:13:00');
INSERT INTO Watchlist (idUser, idContent, time_added) VALUES ( idUser 1, idContent 1015, time_added '2024-01-01 01:14:00');
INSERT INTO Watchlist (idUser, idContent, time_added) VALUES ( idUser 1, idContent 1016, time_added '2024-01-01 01:15:00');
INSERT INTO Watchlist (idUser, idContent, time_added) VALUES ( idUser 1, idContent 1017, time_added '2024-01-01 01:16:00');
INSERT INTO Watchlist (idUser, idContent, time_added) VALUES ( idUser 1, idContent 1018, time_added '2024-01-01 01:17:00');
INSERT INTO Watchlist (idUser, idContent, time_added) VALUES ( idUser 1, idContent 1019, time_added '2024-01-01 01:18:00');
```

```
-- Insert the 51st entry (should trigger oldest removal)
INSERT INTO Watchlist_Request (idUser, idContent) VALUES ( idUser 1, idContent 1051);

-- Check result: should still be 50 items
SELECT COUNT(*) AS total_items FROM Watchlist WHERE idUser = 1;
```




**This is correct.**

**Integration Testing**

**Expected : idContent = 1099 exist in Watchlist and count remains 50**

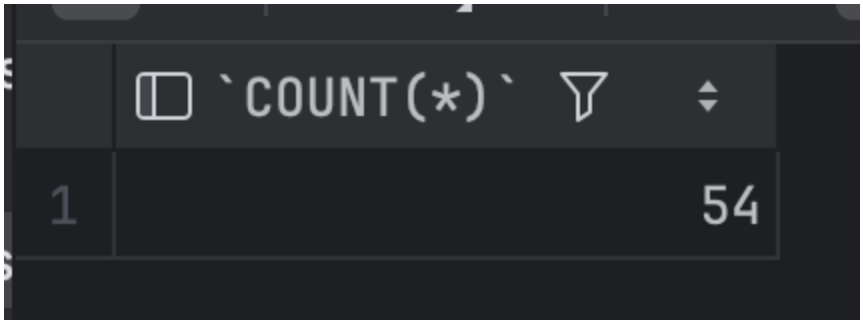
```
INSERT INTO Watchlist_Request (idUser, idContent) VALUES ( idUser 1, idContent 1099);

-- Check if it is inserted correctly
SELECT * FROM Watchlist WHERE idUser = 1 AND idContent = 1099;

SELECT COUNT(*) AS total_items FROM Watchlist WHERE idUser = 1;
```

| | idUser | idContent | time_added |
|---|---|---|---|
| 1 | 1 | 1099 | 2025-05-12 12:42:24 |

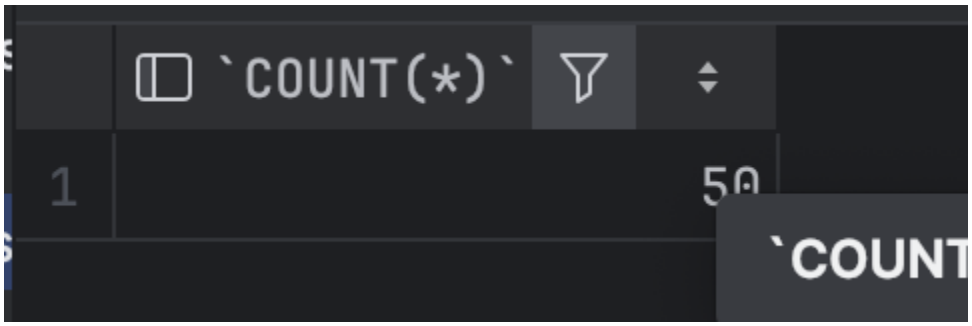
| total_items |
|---|
| 50 |

**Data Integrity Testing**
**Expected: total watchlist count = more than 50.**
**Total watchlist count for idUser = 1 = 50**

```
INSERT INTO Watchlist_Request (idUser, idContent) VALUES ( idUser 2, idContent 1099);

-- Check if it is 51 since the user is different
SELECT COUNT(*) FROM Watchlist;

-- Check if it is 50.
SELECT COUNT(*) FROM Watchlist WHERE idUser = 1;
```




**For watchlist (total)**



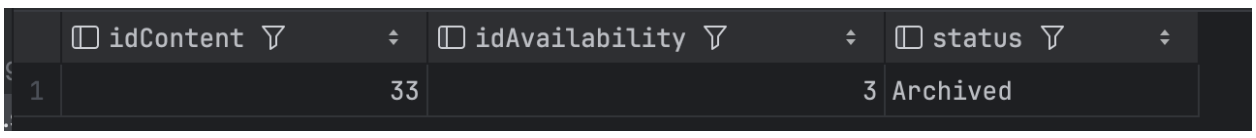

**For watchlist where idUser = 1.**

2. **SET_ARCHIEVED**

**Unit Testing**

**Expected idAvailability is changed to 3**

```
UPDATE Formatted_Content
SET idAvailability = 1
WHERE idContent = 33;

SELECT idContent, idAvailability FROM Formatted_Content WHERE idContent = 33;

-- Insert a new low rating
INSERT INTO Review(idUser, idContent, rating_star)
VALUES ( idUser 2, idContent 33, rating_star 1);

-- Check if the idContent is set to idAvailability = 3 (Archived)
SELECT idContent, Formatted_Content.idAvailability, CA.status FROM Formatted_Content
JOIN MultiMediaDB.Content_Availability CA 1..n<->1: on CA.idAvailability = Formatted_Content.idAvailability
WHERE idContent = 33;
```




**This works.**

**Integration Testing**
**Expected : changed to archived after 1 star ratings**

```
UPDATE Formatted_Content SET idAvailability = 1 WHERE idContent  = 1;
INSERT INTO Review(idUser, idContent, rating_star) VALUES ( idUser 4, idContent 1, rating_star 5);

INSERT INTO Review(idUser, idContent, rating_star) VALUES ( idUser 4, idContent 1, rating_star 1);
INSERT INTO Review(idUser, idContent, rating_star) VALUES ( idUser 4, idContent 1, rating_star 1);
INSERT INTO Review(idUser, idContent, rating_star) VALUES ( idUser 4, idContent 1, rating_star 1);
INSERT INTO Review(idUser, idContent, rating_star) VALUES ( idUser 4, idContent 1, rating_star 1);
INSERT INTO Review(idUser, idContent, rating_star) VALUES ( idUser 4, idContent 1, rating_star 1);

-- check content is now archived
SELECT idContent, Formatted_Content.idAvailability, CA.status FROM Formatted_Content
JOIN MultiMediaDB.Content_Availability CA 1..n<->1: on CA.idAvailability = Formatted_Content.idAvailability
WHERE idContent = 1;
```

| | idContent | idAvailability | status |
|---|---|---|---|
| 1 | 1 | 3 | Archived |
| 2 | 1 | 3 | Archived |

**Date Integrity Testing**
**Expected: idContent = 77 is available**

```
-- DATA INTEGRITY TESTING
UPDATE Formatted_Content SET idAvailability = 1 WHERE idContent = 77;
DELETE FROM Review WHERE idContent = 77;
INSERT INTO Review(idUser, idContent, rating_star) VALUES ( idUser 4, idContent 77, rating_star 2);

check content is now available
SELECT idContent, Formatted_Content.idAvailability, CA.status FROM Formatted_Content
JOIN MultiMediaDB.Content_Availability CA 1..n<->1: on CA.idAvailability = Formatted_Content.idAvailability
WHERE idContent = 77;
```

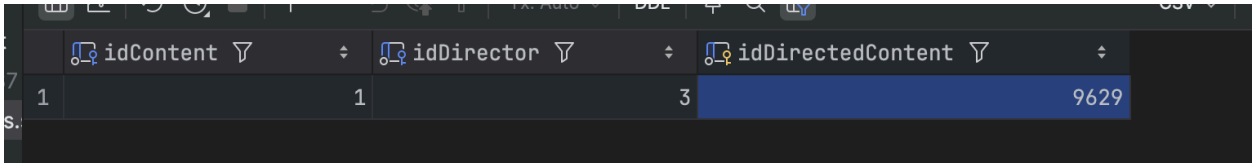
| | idContent | idAvailability | status |
|---|---|---|---|
| 1 | 77 | 1 | Available |

3. **TRG_Duplicate_Director**

**Unit Testing**

**Expected: inserting correctly**

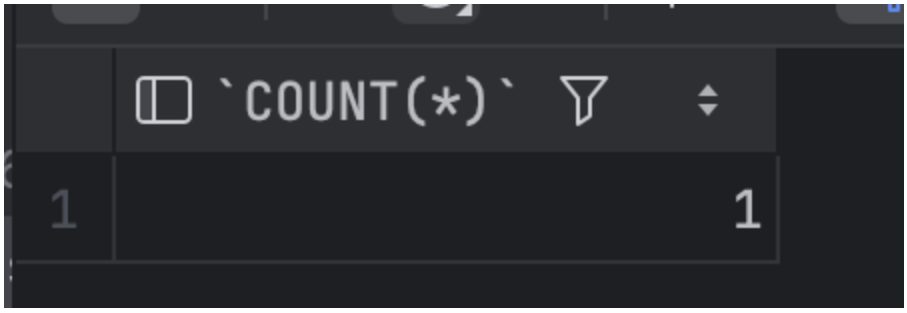
```sql
DELETE FROM Directed_Content
WHERE idContent = 1 AND idDirector = 3;

INSERT INTO Directed_Content_Request(idContent, idDirector)
VALUES ( idContent 1, idDirector 3);

-- Check if the pair is correctly inserted
SELECT * FROM Directed_Content
WHERE idContent = 1 AND idDirector = 3;
```

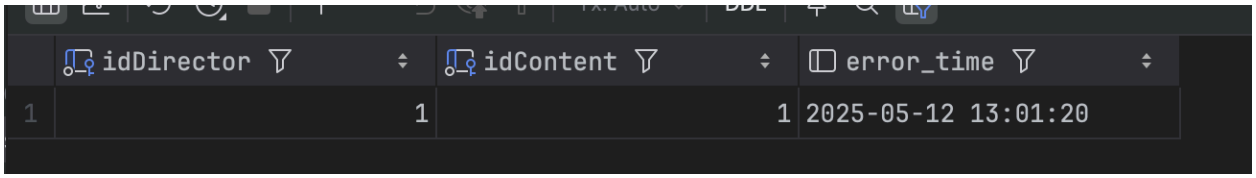


| | idContent | idDirector | idDirectedContent |
|---|---|---|---|
| 1 | 1 | 3 | 9629 |

**Integration Testing**
**Expected : Count = 1, and the insert appears in the Director_Assignment_Errors table**

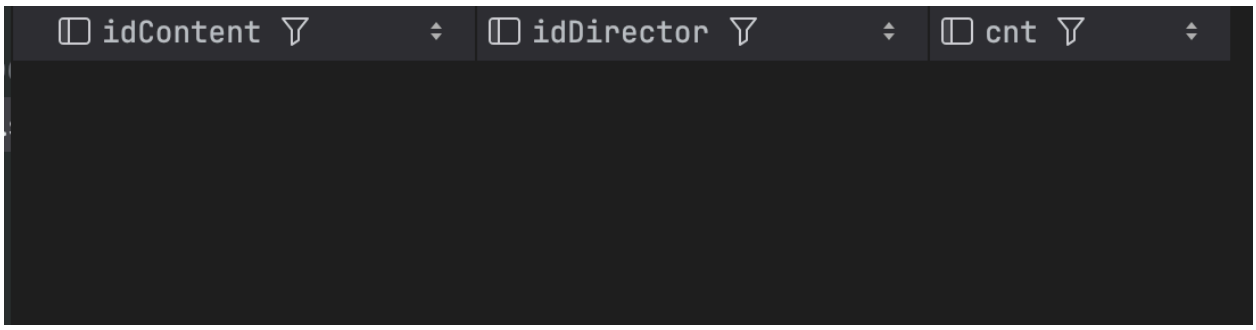
```sql
-- Insert duplicate
INSERT INTO Directed_Content_Request(idContent, idDirector)
VALUES ( idContent 1, idDirector 1);

-- Check that it's still only once in Directed_Content
SELECT COUNT(*) FROM Directed_Content
WHERE idContent = 1 AND idDirector = 1;

-- Check if it was logged as an error with the current timestamp
SELECT * FROM Director_Assignment_Errors
WHERE idContent = 1 AND idDirector = 1;
```




**Data Integrity Testing**
**Expected: empty**

```
SELECT idContent, idDirector, COUNT(*) AS cnt
FROM Directed_Content
GROUP BY idContent, idDirector
HAVING cnt > 1;
```

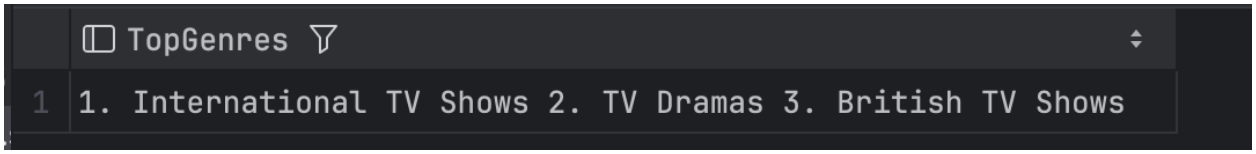



**Functions**

4. **Rank_Genres_By_Hours()**

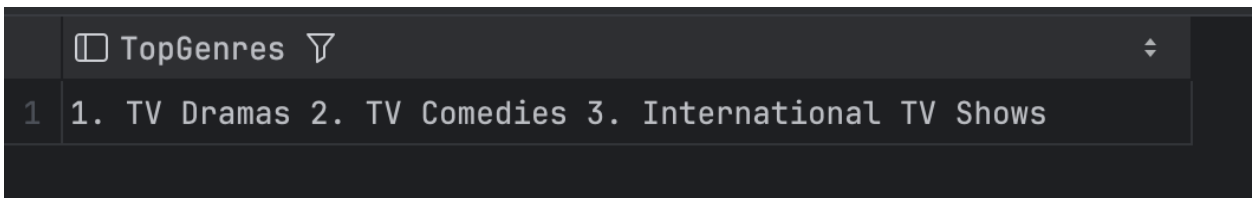
**Unit Testing**
**Expected: Ranked top 3 genres**

```
SELECT Rank_Genres_BY_Hours() AS TopGenres;
```




**Integration Testing**
**Expected : Order to change from the earlier. TV Comedies to be in Top 3.**
**Unexpected: TV Drama to be in top 3. But makes sense because one content can have multiple genres.**

```
-- Insert new watch activity where Genre = TV Comedies
INSERT IGNORE INTO WatchHistory (idUser, idContent)
VALUES ( idUser 1,  idContent 5),
       ( idUser 2, idContent 5),
       ( idUser 2, idContent 18),
       ( idUser 3, idContent 34),
        ( idUser 4, idContent 34),
        ( idUser 1, idContent 18),
        ( idUser 2, idContent 109),
        ( idUser 3, idContent 109),
        ( idUser 4, idContent 109),
        ( idUser 1, idContent 85);

-- CHECK TV Comedies is in Top 3
SELECT Rank_Genres_BY_Hours() AS TopGenres;
```




**Data Integrity Testing**
**Expected : empty**

```sql
-- Delete all watch history for the past month
DELETE FROM WatchHistory WHERE watch_time >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH);

-- Check if it is returns a empty response and not crash
SELECT Rank_Genres_BY_Hours() AS TopGenres;
```

| | TopGenres |
|---|---|
| 1 | <null> |

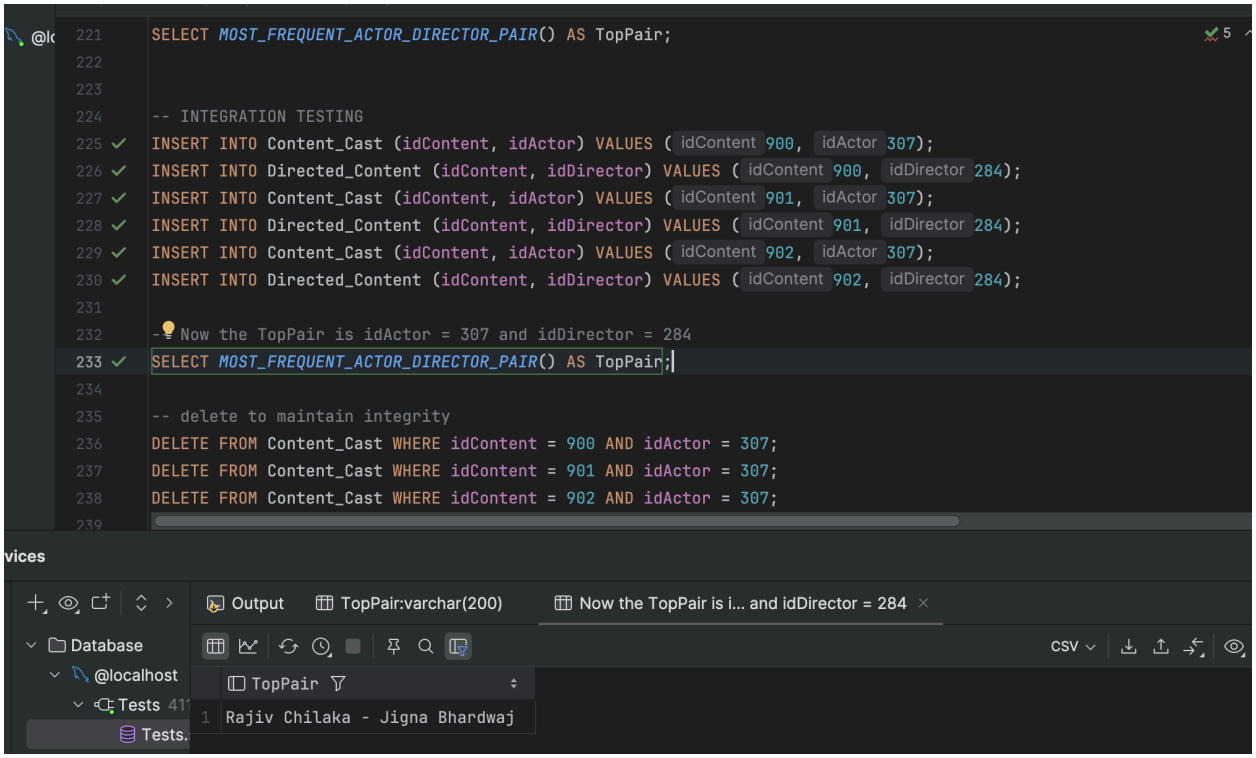
5. **MOST_FREQUENT_ACTOR_DIRECTOR_PAIR**

**Unit Testing**
**Expected: Director and Actor pair**

```sql
SELECT MOST_FREQUENT_ACTOR_DIRECTOR_PAIR() AS TopPair;
```

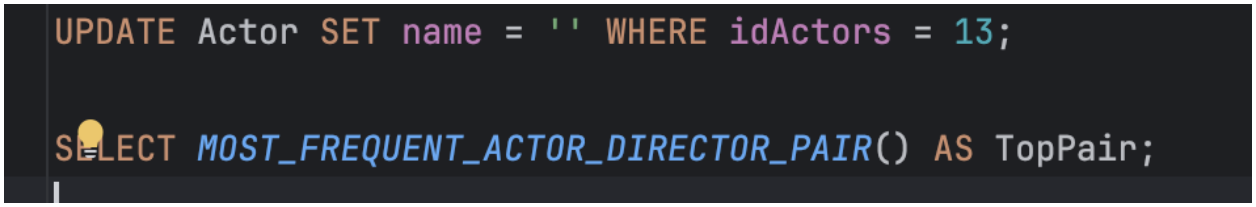
| | TopPair |
|---|---|
| 1 | Rajiv Chilaka - Julie Tejwani |

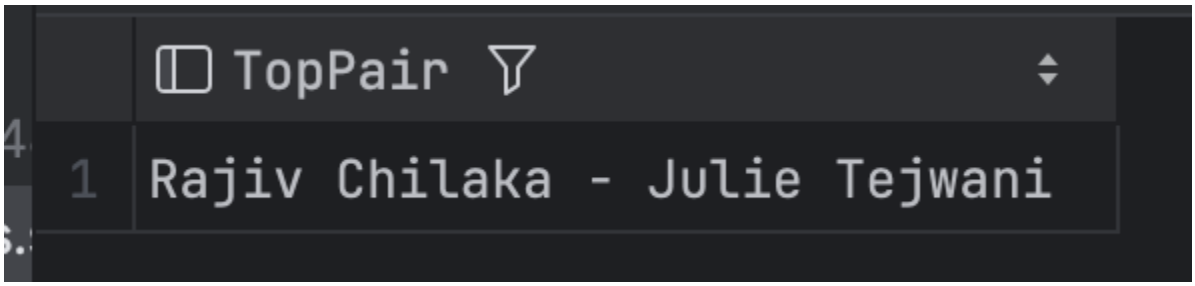
**Integration Testing**
**Expected: a new pairing**




**Data Integrity Testing**

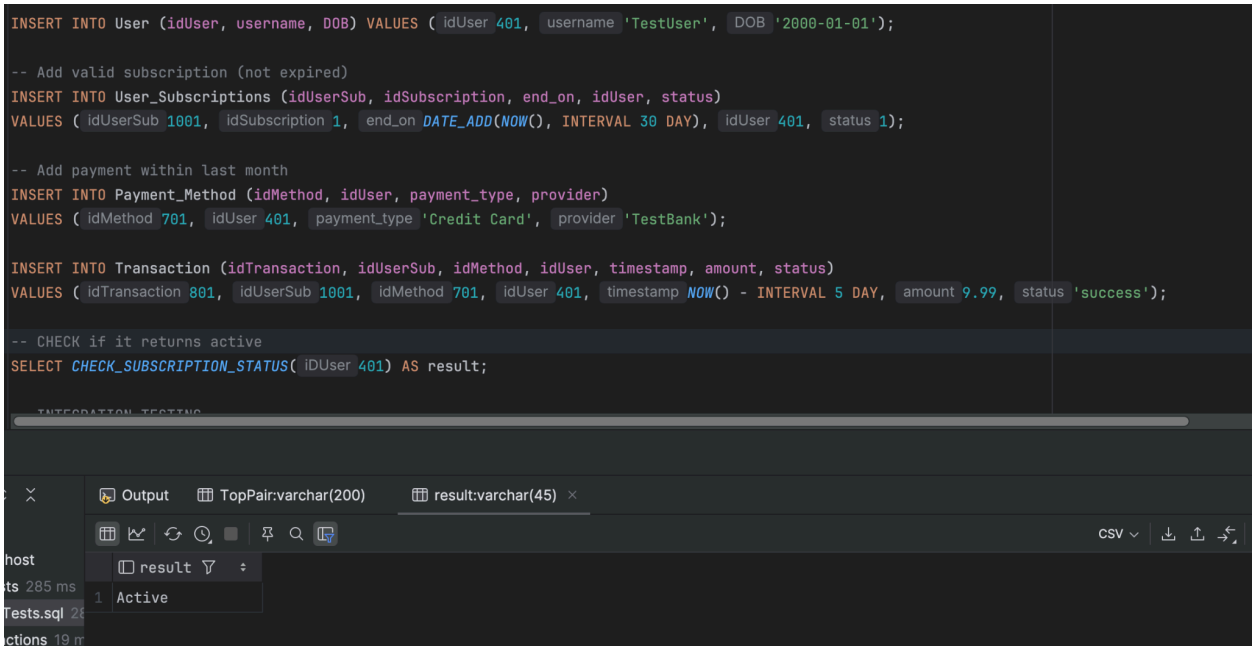
**Expected: same output**



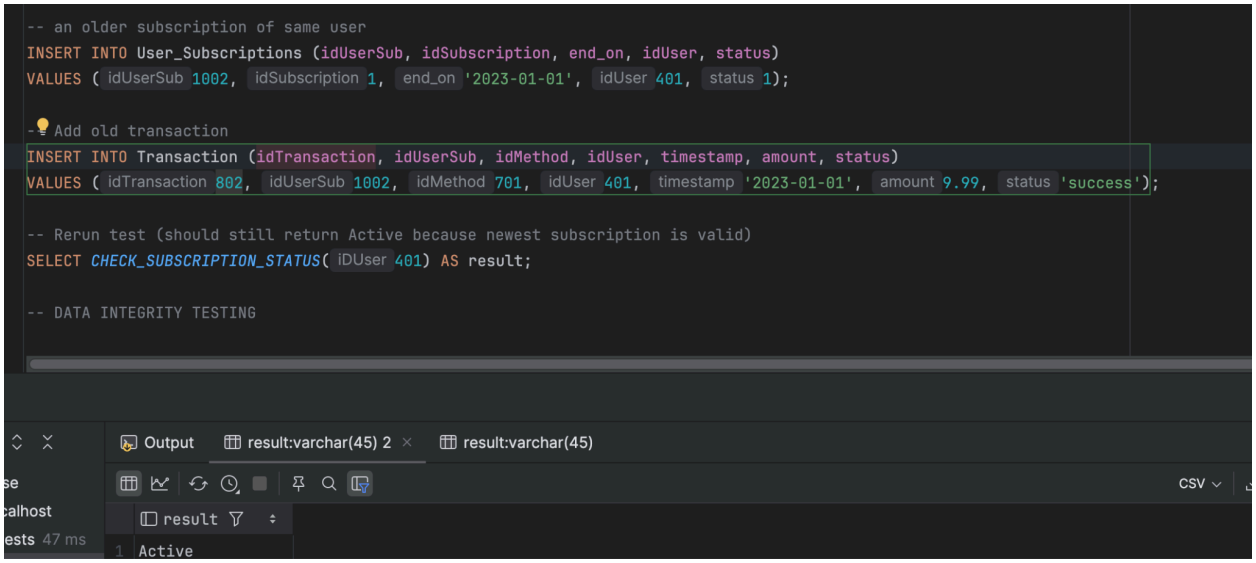




6. **CHECK_SUBSCRIPTION_STATUS**

**Unit Testing**
**Expected: active**




**Integration Testing**
**Expected: active**




**Data Integrity Testing**
**Expected : Expired**

```
-- Add user with no subscription
INSERT INTO User (idUser, username, DOB) VALUES ( idUser 402, username 'NoSubUser', DOB '1995-01-01');

-- CHECK return Expired
SELECT CHECK_SUBSCRIPTION_STATUS( iDUser 402) AS result;
```

Output | result:varchar(45) 2 | result:varchar(45)

| | result |
|---|---|
| 1 | Expired |

7. **USER_ACTIVITY_REPORT**

**Unit Testing**

**Expected: Format of output contain everything needed**

```
CHECK output and format
CALL USER_ACTIVITY_REPORT( idUser 1);
```

Output | Result 74 | Result 75

| | NUMBER_OF_CONTENT | TIME_SPENT | AVERAGE_RATING |
|---|---|---|---|
| 1 | 0 | <null> | 2.60 |

**Integration Testing**

**Expected: everything is updated and displayed**

```
INSERT INTO WatchHistory(idUser, idContent) VALUES ( idUser 5, idContent 33),
                                                   ( idUser 5, idContent 22),
                                                   ( idUser 5, idContent 77);

INSERT INTO Review(idUser, idContent, rating_star) VALUES ( idUser 5, idContent 22, rating_star 4),
                                                          ( idUser 5, idContent 77, rating_star 3);

-- CHECKS that average rating is correct and time spent is increased
CALL USER_ACTIVITY_REPORT( idUser 5);

-- DATA INTEGRITY TESTING
```

Output | CHECKS that average ...me spent is increased | Result 75

| | NUMBER_OF_CONTENT | TIME_SPENT | AVERAGE_RATING |
|---|---|---|---|
| 1 | 3 | 4.85 | 3.67 |

**Data Integrity Testing**
**Expected: Display empty content and parts**

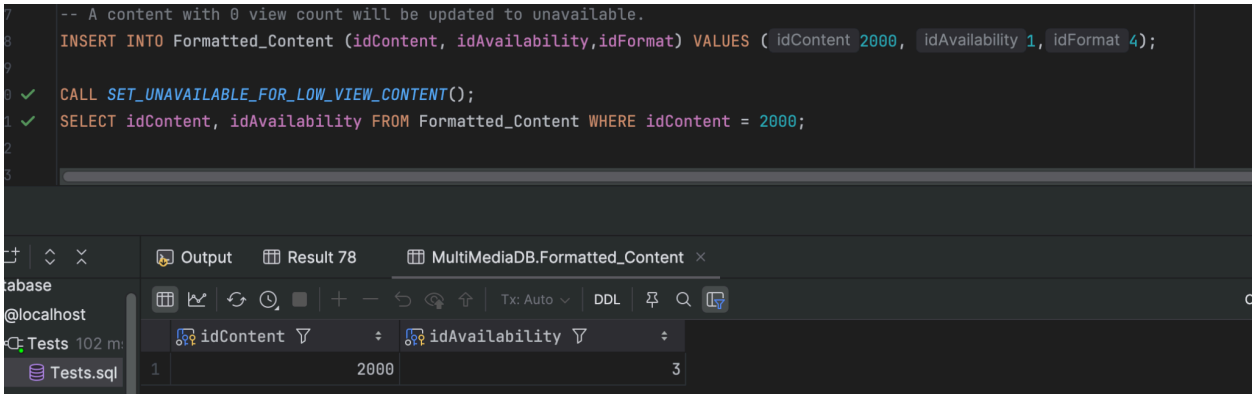
```
-- CHECK returns 0 or nulls when the user doesn't have any activity
CALL USER_ACTIVITY_REPORT( idUser 2);
```

Output | Result 78 | Result 77

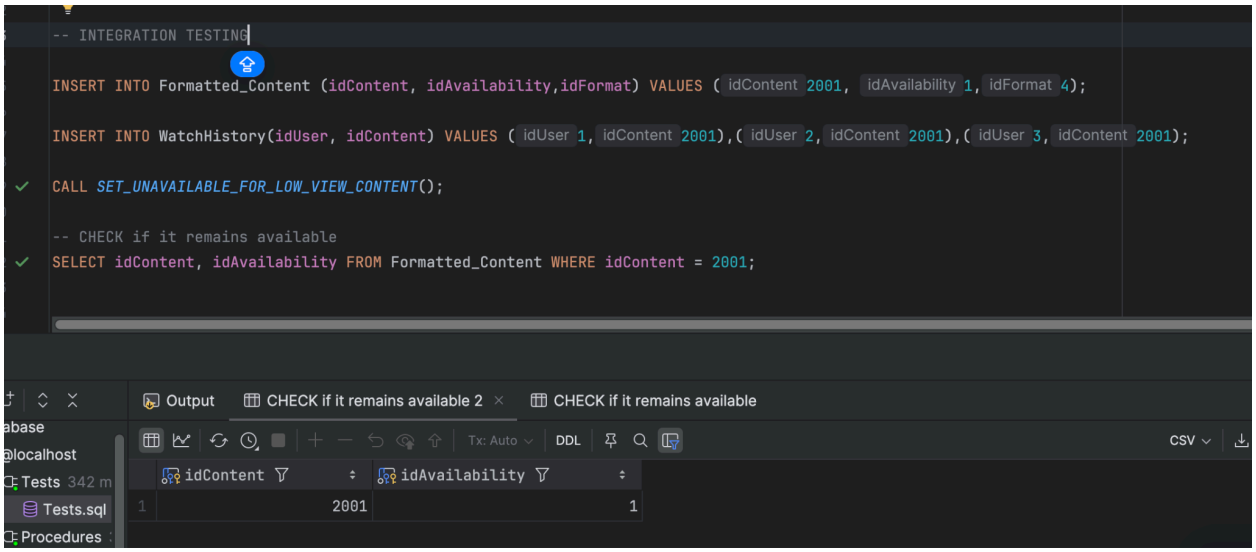
| | NUMBER_OF_CONTENT | TIME_SPENT | AVERAGE_RATING |
|---|---|---|---|
| 1 | 0 | <null> | 1.86 |

8. **SET_UNAVAILABLE_FOR_LOW_VIEW_CONTENT**

**Unit Testing**
**Expected: idAvailability is 3.**

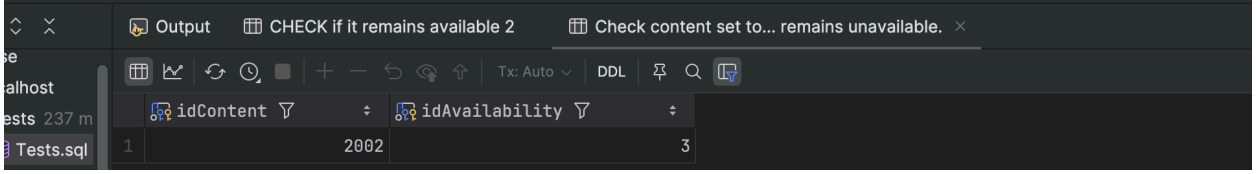



**Integration Testing**
**Expected: Content is still available.**




**Data Integrity Testing**
**Expected: Content is still unavailable**

```
INSERT INTO Formatted_Content (idContent, idAvailability,idFormat) VALUES ( idContent 2002, idAvailability 3, idFormat 4);

CALL SET_UNAVAILABLE_FOR_LOW_VIEW_CONTENT();

-- Check content set to unavailable remains unavailable.
SELECT idContent, idAvailability FROM Formatted_Content WHERE idContent = 2002;


-- 9.LOG_FAILED_PAYMENTS
```




### 9. LOG_FAILED_PAYMENTS

```
-- Create user
INSERT INTO User (idUser, username, DOB) VALUES ( idUser 101, username 'FailUser', DOB '1990-01-01');

-- Create subscription
INSERT INTO User_Subscriptions (idUserSub, idSubscription, end_on, idUser, status)
VALUES ( idUserSub 301, idSubscription 1, end_on NOW() + INTERVAL 30 DAY, idUser 101, status 1);

-- Add payment method
INSERT INTO Payment_Method (idMethod, idUser, payment_type, provider)
VALUES ( idMethod 501, idUser 101, payment_type 'Credit Card', provider 'Visa');
```

**Unit Testing**
**Expected: transaction logged into payment error and notification**

```
-- Insert a failed transaction for user 101
INSERT INTO Transaction (idUserSub, idMethod, idUser, timestamp, amount, status)
VALUES ( idUserSub 301, idMethod 501, idUser 101, timestamp NOW(), amount 19.99, status 'fail');

-- Run the logging procedure
CALL LOG_FAILED_PAYMENTS();

-- Check the payment error was logged
SELECT * FROM Payment_Errors WHERE idUser = 101;
SELECT * FROM NOTIFICATIONS WHERE idUser = 101;
```

| | idError | idTransaction | idUser | error_message | created_at |
|---|---|---|---|---|---|
| 1 | 1 | 9 | 101 | FailUser payment of $19.99 has failed | 2025-05-12 02:21:46 |
| 2 | 2 | 12 | 101 | FailUser payment of $8.99 has failed | 2025-05-12 02:25:10 |
| 3 | 3 | 13 | 101 | FailUser payment of $9.99 has failed | 2025-05-12 02:25:10 |

Output | MultiMediaDB.Payment_Errors | MultiMediaDB.NOTIFICATIONS

| | idNotification | message | created_at | idUser |
|---|---|---|---|---|
| 1 | 10 | FailUser payment of $19.99 has fail | 2025-05-12 02:21:46 | 101 |
| 2 | 11 | FailUser payment of $8.99 has fail | 2025-05-12 02:25:10 | 101 |
| 3 | 12 | FailUser payment of $9.99 has failed | 2025-05-12 02:25:10 | 101 |

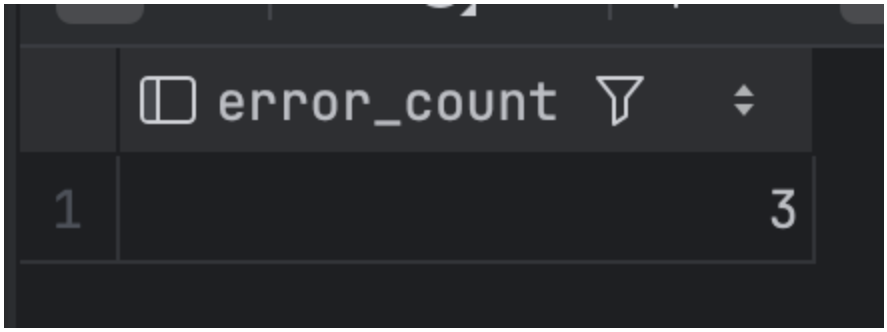
message: text

3 rows

## Integration Testing

```
-- Insert more failed transactions for same user
INSERT INTO Transaction (idUserSub, idMethod, idUser, timestamp, amount, status)
VALUES
( idUserSub 301, idMethod 501, idUser 101, timestamp NOW(), amount 8.99, status 'fail'),
( idUserSub 301, idMethod 501, idUser 101, timestamp NOW(), amount 9.99, status 'fail');

-- First call
CALL LOG_FAILED_PAYMENTS();

-- Call again to test duplicate handling
CALL LOG_FAILED_PAYMENTS();

-- CHECK Only 3 unique payment errors and notifications
SELECT COUNT(*) AS error_count FROM Payment_Errors WHERE idUser = 101;
SELECT COUNT(*) AS notif_count FROM NOTIFICATIONS WHERE idUser = 101;
```

| notif_count |
|---|
| 3 |




**Data Integrity Testing**

```
412
413  -- CHECK for error since idUser doesn't exist.
414  INSERT INTO Transaction (idUserSub, idMethod, idUser, timestamp, amount, status)
415  VALUES ( idUserSub 401, idMethod 601, idUser 9999, timestamp NOW(), amount 12.99, status 'fail');
416
417
```

[23000][1452] Cannot add or update a child row: a foreign key constraint fails (`multimediadb`.`transaction`, CONSTRAINT `Fk_Transaction_UserSub` FOREIGN KEY (`idUserSub`) REFERENCES `user_subscriptions` (`idUserSub`) ON DELETE CASCADE)

**10. REMOVE_EXPIRED_SUBSCRIPTIONS**

**Setup**

```
-- Setup: create test user and expired subscription
INSERT INTO User (idUser, username, DOB) VALUES ( idUser 301, username 'ExpireUser', DOB '2000-01-01');
INSERT INTO User_Subscriptions (idSubscription, end_on, idUser, status)
VALUES ( idSubscription 2, end_on '2024-01-01', idUser 301, status 1);
```

**Unit Testing**

**Expected: user_subscription is deleted from the table and a new notification is inserted**

```sql
-- Manually simulating the event logic
INSERT INTO NOTIFICATIONS(message, idUser)
    SELECT message CONCAT('The subscription of ', U.username, ' has expired.' ),U.idUser
    FROM User_Subscriptions
    JOIN MultiMediaDB.User U 1..n<->1: on U.idUser = User_Subscriptions.idUser
    WHERE NOW() > end_on OR end_on IS NULL OR status = 0;

DELETE FROM User_Subscriptions
WHERE NOW() > end_on OR end_on IS NULL OR status = 0;

-- CHECK that subscription is deleted
SELECT * FROM User_Subscriptions WHERE idUser = 301;

-- Check that notification about the deletion exists.
SELECT * FROM NOTIFICATIONS WHERE idUser = 301;
```

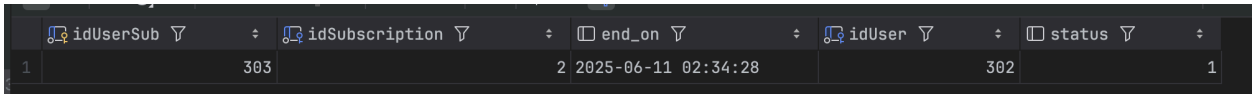
| idUserSub | idSubscription | end_on | idUser | status |
|---|---|---|---|---|

| | idNotification | message | created_at | idUser |
|---|---|---|---|---|
| 1 | 14 | The subscription of ExpireUser has expired. | 2025-05-12 02:32:05 | 301 |

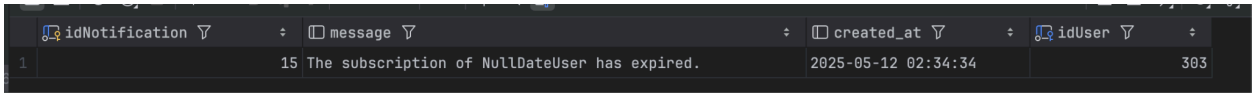
**Integration Testing**

**Expected: user 302 is remain and 303 is removed and notification is inserted**

```sql
-- INTEGRATION TESTING

-- Add 2 test users
INSERT INTO User (idUser, username, DOB) VALUES ( idUser 302, username 'ActiveUser', DOB '2000-02-02'), ( idUser 303, username 'NullDateUser', DOB '2000-03-03');

-- Active subscription
INSERT INTO User_Subscriptions (idSubscription, end_on, idUser, status)
VALUES ( idSubscription 2, end_on DATE_ADD(NOW(), INTERVAL 30 DAY), idUser 302, status 1);

-- Expired subscription with NULL date
INSERT INTO User_Subscriptions (idSubscription, end_on, idUser, status)
VALUES ( idSubscription 2, end_on NULL, idUser 303, status 1);

-- Run event logic
INSERT INTO NOTIFICATIONS(message, idUser)
SELECT message CONCAT('The subscription of ', U.username, ' has expired.'), U.idUser
FROM User_Subscriptions
        JOIN MultiMediaDB.User U 1..n<->1: on U.idUser = User_Subscriptions.idUser
WHERE NOW() > end_on OR end_on IS NULL OR status = 0;

DELETE FROM User_Subscriptions
WHERE NOW() > end_on OR end_on IS NULL OR status = 0;

-- Check: user 302 should remain, 303 should be removed
SELECT * FROM User_Subscriptions WHERE idUser IN (302, 303);
SELECT * FROM NOTIFICATIONS WHERE idUser IN (302, 303);
```



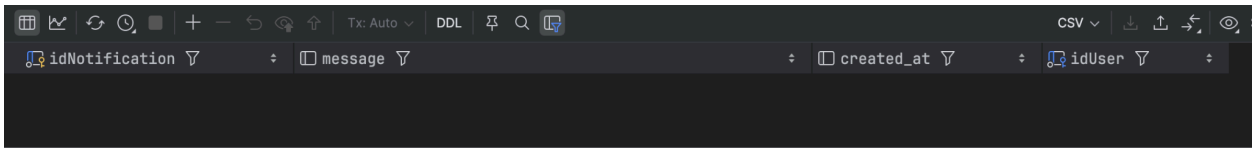
| | idUserSub | idSubscription | end_on | idUser | status |
|---|---|---|---|---|---|
| 1 | 303 | 2 | 2025-06-11 02:34:28 | 302 | 1 |



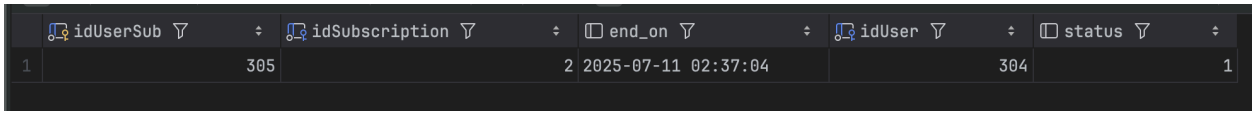
| | idNotification | message | created_at | idUser |
|---|---|---|---|---|
| 1 | 15 | The subscription of NullDateUser has expired. | 2025-05-12 02:34:34 | 303 |

**Data Integrity Testing**
**Expected: user_subscription for idUser = 304 is not deleted and notification does not exist.**

```
-- Insert another valid subscription
INSERT INTO User (idUser, username, DOB) VALUES ( idUser 304, username 'SafeUser', DOB '2001-01-01');
INSERT INTO User_Subscriptions (idSubscription, end_on, idUser, status)
VALUES ( idSubscription 2, end_on DATE_ADD(NOW(), INTERVAL 60 DAY), idUser 304, status 1);

-- Run event logic again
INSERT INTO NOTIFICATIONS(message, idUser)
SELECT  message CONCAT('The subscription of ', U.username, ' has expired.'), U.idUser
FROM User_Subscriptions
        JOIN MultiMediaDB.User U  1..n<->1: on U.idUser = User_Subscriptions.idUser
WHERE NOW() > end_on OR end_on IS NULL OR status = 0;

DELETE FROM User_Subscriptions
WHERE NOW() > end_on OR end_on IS NULL OR status = 0;

-- CHECK THAT it is not deleted
SELECT * FROM User_Subscriptions WHERE idUser = 304;
SELECT * FROM NOTIFICATIONS WHERE idUser = 304;
```



| idNotification | message | created_at | idUser |
|---|---|---|---|



| | idUserSub | idSubscription | end_on | idUser | status |
|---|---|---|---|---|---|
| 1 | 305 | 2 | 2025-07-11 02:37:04 | 304 | 1 |

**11. TOP_10_POPULAR_CONTENT_TODAY**
**Unit Testing**
**Expected: TestGenre is added with TestMovie**

```sql
-- Setup test genre and content
INSERT INTO Genre (idGenre, name) VALUES ( idGenre 1001, name 'TestGenre');
INSERT INTO Content (idContent, title, type, date_added) VALUES ( idContent 9001, title 'TestMovie', type 'Movie', date_added NOW());
INSERT INTO Content_Genre (idContent, idGenre) VALUES ( idContent 9001, idGenre 1001);

-- Simulate 5 views
INSERT INTO WatchHistory (idUser, idContent)
VALUES
    ( idUser 1, idContent 9001),
    ( idUser 2, idContent 9001),
    ( idUser 3, idContent 9001),
    ( idUser 4, idContent 9001);


select * FROM WatchHistory;
-- Check the new genre appears in the event
SELECT * FROM TOP_10_POPULAR_CONTENT_TODAY WHERE genre = 'TestGenre';
```

Output | MultiMediaDB.TOP_10_...POPULAR_CONTENT_TODAY | MultiMediaDB.TOP_10_...POPULAR_CONTENT_TODAY 2

Tx: Auto | DDL | CSV

| | idContent | name | genre | total_views |
|---|---|---|---|---|
| 1 | 9001 | TestMovie | TestGenre | 4 |

## Integration Testing
## Expected: both genres exist

```sql
-- INTEGRATION TESTING

-- Add second genre and another movie
INSERT INTO Genre (idGenre, name) VALUES ( idGenre 1002, name 'OtherGenre');
INSERT INTO Content (idContent, title, type, date_added) VALUES ( idContent 9002, title 'OtherMovie', type 'Movie', date_added NOW());
INSERT INTO Content_Genre (idContent, idGenre) VALUES ( idContent 9002, idGenre 1002);

-- Simulate 8 views for OtherMovie
INSERT INTO WatchHistory (idUser, idContent)
VALUES
    ( idUser 1, idContent 9002),
    ( idUser 2, idContent 9002),
    ( idUser 3, idContent 9002),
    ( idUser 4, idContent 9002);

-- Check for both genres exist
SELECT * FROM TOP_10_POPULAR_CONTENT_TODAY WHERE genre IN ('TestGenre', 'OtherGenre');
```

Output | MultiMediaDB.TOP_10_...POPULAR_CONTENT_TODAY | MultiMediaDB.TOP_10_...POPULAR_CONTENT_TODAY 2

Tx: Auto | DDL | CSV

| | idContent | name | genre | total_views |
|---|---|---|---|---|
| 1 | 9002 | OtherMovie | OtherGenre | 4 |
| 2 | 9001 | TestMovie | TestGenre | 4 |

## Data Integrity Testing
## Expected: the new test movie does not exist in the table


## Performance Insights

There are no noticeable delays or latency issues, even when working with large datasets. The current implementation of triggers, functions, procedures, and events executes efficiently and responds quickly, indicating that the system is well-optimized for performance under the existing workload. However, it is to be noted that the system has not been stress tested.

## Improvements and Recommendations

From a performance perspective, there is currently no significant latency observed during the query executions. However, from a design and optimization perspective, Transaction table could be improved.

Currently, Transaction table contains both idUser_Subcription and idUser, while the User_Subscription already contains the idUser. This introduces partial redundancy and violate Third Normal Form (3NF). However this set up makes sense within the context of the system as Payment Method table use a composite primary key (idMethod, idUser) to ensure the each payment method is unique to a single user and prevent a user from accessing another user's payment method. idUser is needed in the Transaction table to reference this primary keys from the Payment Method.

**Recommendation:**

Use before insert on Transaction trigger to signal an error when the user is inserted into the Transaction with an idMethod that doesn't belong to them.