Project Phase 4

Radha Rungta, Ahana Talukdar, Apeksha Chandak, Bhavya Ram V

Insert Function

1. Insert Account

Inserts a new account into the `Account` table, along with optional bank details. If bank details are provided, it inserts them into the `Bank_Details` table. The function collects account information like name, email, country, active devices, and bank account number.

2. Insert Subscription

Inserts a subscription for a given account. The function first checks if the account exists, then inserts the subscription details into the `Subscription` table. Depending on the subscription type (Mobile, Basic, Standard, Premium), it also inserts specific details into the relevant table (like `Mobile`, `Basic`, etc.).

3. Insert Actor

Inserts a new actor into the `Actor` table. The function prompts for the actor's name, age, and country, and adds these details into the table. The `Actor_ID` is auto-incremented.

4. Insert Show

Inserts a new show into the `Shows` table by first adding details to the `Content` table (such as title, description, country, and rating). After that, it inserts specific details like the number of seasons into the `Shows` table.

5. Insert Movie

Inserts a new movie into the `Movies` table. The movie's content details (like title, description, country of origin, and rating) are first added to the `Content` table. Then, movie-specific information such as release year, duration, and ratings are inserted into the `Movies` table.

Retrieve Function

Selection Queries

1. **Subscription Status Report(Number of users with subscriptions that are either inactive or ending soon)**

The `subscription_status_report()` function generates a report on user subscription statuses, focusing on accounts with subscriptions that are either inactive or ending within 10 days. It executes a SQL query that categorizes subscriptions as *Inactive* or *Ending Soon* based on their status and expiration date, then counts the distinct number of

users in each category. The results, including the subscription status and the number of affected users, are displayed in a user-friendly format. If no data matches the criteria, the user is informed. The function also handles errors gracefully, providing detailed feedback in case of issues.

2. **Get Actors and Languages worked in**

The `get_actors_languages_worked_in()` function retrieves and displays the names of actors along with the languages they have worked in. The function uses a SQL query to fetch the actor names and the distinct languages associated with the content they have acted in, combining the results in a readable format. If there is data available, it lists each actor with the languages they have worked in. If no data is found, the function informs the user accordingly.

3. **Get all the movies of a particular genre**

The `get_movies_by_genre()` function retrieves all movies in the database that belong to a specific genre provided by the user. It prompts the user to input the desired genre and executes a SQL query to fetch movie details such as `Title`, `Description`, `Country_of_Origin`, `Release_Year`, `Duration`, and `Ratings`. The function joins the `Content`, `Movies`, and `Content_Genre` tables to filter and retrieve the relevant data. If matching movies are found, they are displayed; otherwise, the user is notified that no movies exist in the specified genre.

Projection Queries

4. **Account who has watched more than 3 movies or shows**

The `account_watched_three()` function retrieves and displays the account IDs of users who have watched more than three movies. It queries the database to count the number of movies watched by each account and filters those with more than three. The results are then printed, or a message is shown if no accounts meet the criteria.

5. **Get Content whose avg rating > 8.5**

The `get_highly_rated_content()` function retrieves and displays the titles of content with an average rating greater than 8.5. The content is ordered by its average rating in descending order. It executes a SQL query to calculate the average ratings for each content and filters the results based on the rating condition (greater than 8.5). If no content meets this condition, it informs the user that no highly rated content was found.

Aggregate Queries

6. **Get number of actors for a particular content**

The `get_number_of_actors_for_content()` function retrieves and displays the number of actors associated with a given content title. It takes the content title as input from the user, executes a SQL query to count the actors linked to that title, and shows the result. If no actors are found for the provided title, it notifies the user to check the title.

7. **Get account with most followers**

The `get_account_with_most_followers()` function retrieves the account with the highest number of followers from the database. It executes a SQL query that joins the `Account` and `Account_Followers` tables, counting the number of followers for each account. The results are sorted in descending order of followers, and the account with the most followers is displayed, including the account ID, name, and follower count. If no accounts have followers, the user is informed.

8. **Get total hours watched by an account**

The `get_total_hours_watched()` function calculates the total number of hours watched by a specific account in the database. It prompts the user to enter the account's email, retrieves the corresponding `Account_ID` using the `query_account_id_by_email()` function, and then executes a SQL `SELECT` query to sum the hours from the `Duration_Watched` column in the `Account_Watch_History` table for the specified account. If data exists, it displays the total hours watched; otherwise, it informs the user that no data or hours are available.

Search Queries

9. **Get accounts between birth year 2000 and 2010**

The `get_accounts_by_birth_year()` function retrieves accounts from the database where the `Date_of_Birth` falls between the years 2000 and 2010. It executes a SQL `SELECT` query with a `YEAR()` function to filter records meeting the specified date range. If matching accounts are found, their `Account_ID`, `Name`, and `Date_of_Birth` are displayed. If no matches are found, the function informs the user.

10. **Get accounts with name containing 'REDDY'**

The `get_accounts_by_name()` function retrieves all accounts from the database where the name contains the substring 'REDDY'. It executes a SQL `SELECT` query with a `LIKE` condition to filter names that match the specified pattern. If matching accounts are found, their `Account_ID` and `Name` are displayed. If no matches exist, the user is notified.

Analysis Queries

11. **Get USER WRAPPED SUMMARY**

The `get_user_wrapped_summary()` function retrieves a detailed summary of a user's activity based on their email address. The summary includes various metrics, such as total watch time, most watched content, favorite genres, total reviews given, most liked content, and watch history over the past year.
Account Retrieval: The function first retrieves the `Account_ID` and user name using the provided email. If the account does not exist, an error message is shown.Data Retrieval: The function queries various data points related to the user's activity:

a. Total watch time
b. Most watched content
c. Favorite genres

d. Total reviews given
e. Most liked content
f. Watch history over time

Displaying Summary: Once the data is retrieved, it organizes and displays a detailed summary of the user's activity in a formatted output.

12. **Get average rating by country and genre**

    The `get_average_rating_by_country_and_genre()` function retrieves the average ratings of content, grouped by their country of origin and genre. It calculates the average rating for each combination and orders the results by country and average rating in descending order. The function displays the results or a message if no data is found.

Update Function

1. **Update Email**

   Updates the email address of a user. The user provides their name, old email, and new email. The query ensures that only the email associated with the provided old email for the given name is updated.

2. **Update Account Type**

   Updates the `AccountType` of a user based on their name. The user is prompted to enter the name of the account holder and the new account type (`General` or `Kids`), which will be updated in the `Account` table.

3. **Update Bank Account Details**

   Updates the bank account details of a user based on their email address. The user provides their email and new bank details (account number, bank name, and branch code), and the function updates these fields in the `Account` table.

4. **Update Mobile Subscription Price of a Country**

   Updates the price of a subscription based on the country and subscription type. The user provides the country, subscription type (e.g., Mobile, Standard, Premium), and the new price, and the function updates the `Price` field in the respective subscription table.

Delete Function

1. **Delete Account**

   The `deleteAccount()` function facilitates the deletion of a user account from the database based on the provided email ID. It executes a SQL `DELETE` query on the `Account` table to remove the record associated with the email. If the deletion is successful, the function confirms the account's removal; otherwise, it notifies the user that no account was found with the given email.

2. **Delete Subscription**

   The `delete_subscription()` function enables the deletion of a subscription associated with a specific account identified by its email ID. It first retrieves the `Account_ID` corresponding to the provided email ID from the `Account` table. If no matching account is found, the user is notified. If a valid account exists, the function executes a SQL `DELETE` query on the `Subscription` table to remove the subscription linked to the retrieved `Account_ID`. The function confirms the deletion's success or notifies the user if no subscription was found

3. **Delete Movie**

   The `delete_movie()` function allows users to remove a specific movie from the database by providing its title. It executes a SQL `DELETE` query to delete the corresponding record in the `Content` table. The function confirms successful deletion if the title exists or notifies the user if no matching record is found.

4. **Delete Show**

   The `delete_show()` function allows users to delete a specific show from the database by providing its title. It executes a SQL `DELETE` query to remove the corresponding record from the `Content` table. If the title exists, the function confirms the deletion; otherwise, it notifies the user that no matching record was found.

5. **Delete Account Following**

   The `delete_account_following()` function is designed to remove a "following" relationship between two user accounts based on their email addresses. The function validates the existence of both accounts by fetching their `Account_ID` from the `Account` table. If the accounts are valid, it executes a SQL `DELETE` query on the `Account_Following` table to remove the relationship.