Big Data Management - Assignment 1

Music Recommendation System

Akshay Kumar (G24AI1033)

Question 1 : Insert the above into the recommendations table

```
#Insert the above into the recommendations table

qry_insert_recs = """
INSERT INTO Recommendations (recommendation_id, user_id, song_id, recommendation_time)
WITH song_similarity AS (
  -- Find pairs of songs listened to by more than one user
  SELECT
    u1.song_id as song1,
    u2.song_id as song2
  FROM Listens u1
  JOIN Listens u2 ON u1.user_id = u2.user_id AND u1.song_id != u2.song_id
  GROUP BY
    u1.song_id, u2.song_id
  HAVING
    COUNT(DISTINCT u1.user_id) > 1
),
potential_recs AS (
  -- Find recommendations for users
  SELECT DISTINCT
    l.user_id,
    ss.song2 as song_id
  FROM song_similarity ss
  -- Join with all listens to find users who have listened to one song in the pair
  JOIN Listens l ON l.song_id = ss.song1
  -- Exclude songs the user has already listened to
  WHERE
    ss.song2 NOT IN (SELECT song_id FROM Listens WHERE user_id = l.user_id)
)
-- Select the recommendations and generate a new ID and timestamp
SELECT
  (SELECT COALESCE(MAX(recommendation_id), 0) FROM Recommendations) + ROW_NUMBER() OVER () as recommendation_id,
  user_id,
  song_id,
  strftime('%Y-%m-%d %H:%M:%S', 'now') as recommendation_time
FROM potential_recs;
"""


conn = sqlite3.connect(dbname)
cursor = conn.cursor()
cursor.execute("DELETE FROM Recommendations;")

cursor.execute(qry_insert_recs)
conn.commit()
conn.close()

runSql('Data in Recommendations Table', "SELECT * FROM Recommendations;")
```

Data in Recommendations Table

recommendation_id	recommendation_time	user_id	song_id
1	2025-06-23 16:18:20	2	1
2	2025-06-23 16:18:20	2	6

Question 2 : Generate the recommendations for Minnie

```
# Generate the recommendations for Minnie
qry_minnie_recs = """
SELECT
  u.name,
  s.title,
  s.artist
FROM Recommendations r
JOIN Users u ON r.user_id = u.user_id
JOIN Songs s ON r.song_id = s.song_id
WHERE u.name = 'Minnie';
"""
runSql("Recommendations for Minnie", qry_minnie_recs)
```

Recommendations for Minnie

name	title	artist
Minnie	Evermore	Taylor Swift
Minnie	Yesterday	Beatles

Question 3 : Re-do the generation of recommendations now on the basis of listen time

```
# Re-do the generation of recommendations now on the basis of listen time

qry_recs_by_sameday = """
WITH same_day_listens AS (
  -- Step 1: Find pairs of songs listened to by DIFFERENT users on the SAME DAY.
  SELECT
    l1.user_id AS user1_id,
    l1.song_id AS song1_id,
    l2.user_id AS user2_id,
    l2.song_id AS song2_id
  FROM Listens l1
  JOIN Listens l2 ON
    -- Find listens on the same day
    date(l1.listen_time) = date(l2.listen_time)
    -- Ensure we are comparing listens from two different users
    AND l1.user_id != l2.user_id
  WHERE
    -- Only consider listens with a valid timestamp
    l1.listen_time IS NOT NULL AND l2.listen_time IS NOT NULL
),
potential_recs AS (
  -- Step 2: From the pairs, determine potential recommendations.
  -- If user1 listened to song1, then song2 (listened to by user2) is a recommendation for user1.
  SELECT DISTINCT
    sdl.user1_id as user_id,
    sdl.song2_id as song_id
  FROM same_day_listens sdl
  -- Step 3: Filter out songs that the user has already listened to.
  WHERE
    sdl.song2_id NOT IN (SELECT song_id FROM Listens WHERE user_id = sdl.user1_id)
)
-- Final selection of distinct user-song recommendations
SELECT * FROM potential_recs;

"""
runSql("New Recommendations Based on Same-Day Listens", qry_recs_by_sameday)
```

Question 4 : Generate new recommendations

New Recommendations Based on Same-Day Listens

user_id	song_id

Question 5 : What are the differences with the static method on #2 above

Shared Listening Recommendation (Method 1)	Listening Time-based Recommendation (Method 2)
This method looks at all songs a person has ever listened to.	This method only looks at songs where we know the exact listen time.
It uses all the data in the Listens table, even if the listen_time is empty.	It throws away any listen that doesn't have a timestamp, using less data.
Its goal is to find users who share the same overall taste in music.	Its goal is to find users with shared taste, but only using "verified" (timestamped) listens.
Because it used all the data, it found that Mickey and Daffy both liked similar songs.	Because it threw away data, it couldn't find any two users who liked similar songs.
This method worked and gave us two good recommendations for Minnie.	This method failed and gave us zero recommendations.
It's a more robust approach when your data might be incomplete.	It's not a good approach for our dataset because too much data was missing.
It assumes that any listen, with or without a time, tells us something useful.	It assumes that only a listen with a timestamp is trustworthy enough to be used.
The recommendations are based on a user's entire listening history.	The recommendations are based on a small, filtered part of a user's history.
This method successfully found a pattern in the data.	This method failed because filtering the data removed the very pattern it needed to find.
Gave useful recommendations by looking at the bigger picture.	Gave no recommendations because it was too strict with the data it used.