

Indian Institute of Technology Jodhpur
BDM

Assignment 2
BDM

Submitted by: **Jojo Joseph (G23AI2100)**

Google Colab :

<https://colab.research.google.com/drive/1U6phW2IEz6SHlu99qoVvTRlyzg1FlxnT?usp=sharing>

gitHub : https://github.com/IITJPGD/BDMAssignment2_g23ai2100

BDM Assignment

#ASSIGNMENT-2. To be submitted as a PDF with queries and screenshots.

#1 Insert the above into the recommendations table

Ans

```
import sqlite3

conn = sqlite3.connect(dbname)

cursor = conn.cursor()

recommendation_query_temp = """

INSERT INTO Recommendations (recommendation_id, recommendation_time,
user_id,

song_id)

WITH song_similarity AS (

SELECT u1.song_id AS song1, u2.song_id AS song2, COUNT(*) AS common_users

FROM Listens u1

JOIN Listens u2
```

Submitted by: Jojo Joseph (G23AI2100)

```

ON u1.user_id = u2.user_id AND u1.song_id != u2.song_id

GROUP BY u1.song_id, u2.song_id

HAVING COUNT(*) > 1

)

SELECT

ROW_NUMBER() OVER (ORDER BY L.user_id, song2) AS recommendation_id,

datetime('now') AS recommendation_time,

L.user_id,

song2 AS song_id

FROM song_similarity

JOIN Listens AS L ON L.song_id = song_similarity.song1

WHERE song2 NOT IN (

SELECT song_id FROM Listens WHERE user_id = L.user_id

);

"""

cursor.execute(recommendation_query_temp)

conn.commit()

runSql('TEST', 'SELECT * FROM Recommendations;')

```

OutPut :

The screenshot shows a Jupyter Notebook titled "BDM_Assignment_2_g23ai2100.ipynb". The code cell contains a SQL query that uses a window function to rank songs by recommendation time for a specific user, excluding songs they have already listened to. The query is as follows:

```
ROW_NUMBER() OVER (ORDER BY L.user_id, song2) AS recommendation_id,  
datetime('now') AS recommendation_time,  
L.user_id,  
song2 AS song_id  
FROM song_similarity  
JOIN Listens AS L ON L.song_id = song_similarity.song1  
WHERE song2 NOT IN (  
  SELECT song_id FROM Listens WHERE user_id = L.user_id  
);  
""""  
cursor.execute(recommendation_query_temp)  
conn.commit()  
  
runSql(['TEST', 'SELECT * FROM Recommendations;'])
```

The output of the query is displayed as a table with the following columns: recommendation_id, recommendation_time, user_id, and song_id. The results show 8 rows of data, representing recommendations for user 2, ranked by time.

recommendation_id	recommendation_time	user_id	song_id
1	2024-12-15 14:42:10	2	1
2	2024-12-15 14:42:10	2	6
1	2024-12-15 14:42:25	2	1
2	2024-12-15 14:42:25	2	6
1	2024-12-15 14:42:47	2	1
2	2024-12-15 14:42:47	2	6
1	2024-12-15 14:43:22	2	1
2	2024-12-15 14:43:22	2	6

The bottom status bar indicates that the code was completed at 20:13.

#2 Generate the recommendations for Minnie

Ans

```
minnie_recommendations_query = """"
```

```
SELECT
```

```
s.title,
```

```
s.artist,
```

```
s.genre,
```

```
r.recommendation_time
```

```
FROM Recommendations r
```

```
JOIN Songs s ON r.song_id = s.song_id
```

```
WHERE r.user_id = 2
```

```
ORDER BY r.recommendation_time DESC;
```

```
""""
```

```
runSql('Minnie Recommendations', minnie_recommendations_query)
```

BDM_Assignment_2_g23ai2100.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

✓ RAM
Disk

Share

2 Generate the recommendaions for Minnie

```
minnie_recommendations_query = """
SELECT
s.title,
s.artist,
s.genre,
r.recommendation_time
FROM Recommendations r
JOIN Songs s ON r.song_id = s.song_id
WHERE r.user_id = 2
ORDER BY r.recommendation_time DESC;
"""
runSql('Minnie Recommendations', minnie_recommendations_query)
```

Minnie Recommendations

title	artist	genre	recommendation_time
Evermore	Taylor Swift	Pop	2024-12-15 14:43:22
Yesterday	Beatles	Classic	2024-12-15 14:43:22
Evermore	Taylor Swift	Pop	2024-12-15 14:42:47
Yesterday	Beatles	Classic	2024-12-15 14:42:47
Evermore	Taylor Swift	Pop	2024-12-15 14:42:25
Yesterday	Beatles	Classic	2024-12-15 14:42:25
Evermore	Taylor Swift	Pop	2024-12-15 14:42:10
Yesterday	Beatles	Classic	2024-12-15 14:42:10

0s completed at 20:15

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

Ans

```
time_query = """

SELECT

l1.user_id,

s.title,

s.artist,

l1.rating,

l2.rating AS other_rating,

l1.listen_time,
```

```

12.listen_time AS other_listen_time

FROM Listens l1

JOIN Listens l2 ON l1.user_id = l2.user_id AND l1.song_id != l2.song_id

JOIN Songs s ON l2.song_id = s.song_id

WHERE l1.listen_time IS NOT NULL

AND l2.listen_time IS NOT NULL

ORDER BY l1.user_id, l1.listen_time DESC;

"""

runSql('Time-based Analysis Results', time_query)

```

The screenshot shows a Jupyter Notebook titled "BDM Assignment 2_g23ai2100.ipynb". The code cell contains a SQL query and a function call. The query selects user_id, title, artist, rating, other_rating, listen_time, and other_listen_time from the Listens table, joined with the Songs table. The results are displayed in a table below the code cell.

```

time_query = """
SELECT
l1.user_id,
s.title,
s.artist,
l1.rating,
l2.rating AS other_rating,
l1.listen_time,
l2.listen_time AS other_listen_time
FROM Listens l1
JOIN Listens l2 ON l1.user_id = l2.user_id AND l1.song_id != l2.song_id
JOIN Songs s ON l2.song_id = s.song_id
WHERE l1.listen_time IS NOT NULL
AND l2.listen_time IS NOT NULL
ORDER BY l1.user_id, l1.listen_time DESC;
"""

runSql('Time-based Analysis Results', time_query)

```

user_id	title	artist	rating	other_rating	listen_time	other_listen_time
1	Yesterday	Beatles	4.5	3.9	2024-08-30 14:35:00	2024-08-29 10:15:00
1	Evermore	Taylor Swift	3.9	4.5	2024-08-29 10:15:00	2024-08-30 14:35:00
2	Hey Jude	Beatles	4.6	3.9	2024-08-28 09:20:00	2024-08-27 16:45:00
2	Yellow Submarine	Beatles	3.9	4.6	2024-08-27 16:45:00	2024-08-28 09:20:00

[] Start coding or generate with AI.

0s completed at 20:30

#4 Generate new recommendations

Ans

```

cursor.execute("DELETE FROM Recommendations;")

conn.commit()

# ----- new recommendations based on user preferences ---

insert_new_recom_query = """

WITH user_preferences AS (

SELECT

l1.user_id,

l1.song_id,

s1.genre,

l1.rating,

l1.listen_time

FROM Listens l1

JOIN Songs s1 ON l1.song_id = s1.song_id

WHERE l1.listen_time IS NOT NULL

),

potential_recommendations AS (

SELECT

up.user_id,

s2.song_id AS recommended_song_id,

s2.title,

s2.artist,

s2.genre,

up.rating AS user_rating,

```

```

ROW_NUMBER() OVER (PARTITION BY up.user_id ORDER BY up.listen_time DESC,
up.rating DESC) AS rank

FROM user_preferences up

JOIN Songs s2 ON up.genre = s2.genre

WHERE s2.song_id NOT IN (

SELECT song_id FROM Listens WHERE user_id = up.user_id

)

)

INSERT INTO Recommendations (recommendation_id, recommendation_time,
user_id,

song_id)

SELECT

ROW_NUMBER() OVER (ORDER BY user_id, rank),

datetime('now'),

user_id,

recommended_song_id

FROM potential_recommendations

WHERE rank <= 3;

"""

cursor.execute(insert_new_recom_query)

conn.commit()

```

```
verify_query = """

SELECT

r.user_id,

u.name AS user_name,

s.title,

s.artist,

s.genre,

r.recommendation_time

FROM Recommendations r

JOIN Songs s ON r.song_id = s.song_id

JOIN Users u ON r.user_id = u.user_id

ORDER BY r.user_id, r.recommendation_time;

"""

runSql('New Time-Based Recommendations Result', verify_query)
```


The screenshot shows a Jupyter Notebook titled "BDM_Assignment_2_g23ai2100.ipynb". The code cell contains a SQL query to retrieve user information and song recommendations. The query is as follows:

```
verify_query = """
SELECT
  r.user_id,
  u.name AS user_name,
  s.title,
  s.artist,
  s.genre,
  r.recommendation_time
FROM Recommendations r
JOIN Songs s ON r.song_id = s.song_id
JOIN Users u ON r.user_id = u.user_id
ORDER BY r.user_id, r.recommendation_time;
"""

runSql('New Time-Based Recommendations Result', verify_query)
```

The output of the query is a table with the following data:

user_id	user_name	title	artist	genre	recommendation_time
1	Mickey	Hey Jude	Beatles	Classic	2024-12-15 15:05:29
1	Mickey	Yellow Submarine	Beatles	Classic	2024-12-15 15:05:29
2	Minnie	Yesterday	Beatles	Classic	2024-12-15 15:05:29
2	Minnie	Yesterday	Beatles	Classic	2024-12-15 15:05:29

The bottom of the notebook shows a status bar indicating the code was completed at 20:35.

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

Ans

```
static_analysis_query_test = """

SELECT

l.user_id,

u.name,

s.title AS listened_to,

s.artist,

l.rating,

l.listen_time

FROM Listens l

JOIN Songs s ON l.song_id = s.song_id

JOIN Users u ON l.user_id = u.user_id

ORDER BY l.user_id, l.rating DESC;
```

```

"""

runSql('Static Analysis Listening Patterns', static_analysis_query_test)

time_compr_query = """

SELECT

r.user_id,

u.name,

s.title AS recommended_song,

s.artist,

s.genre,

r.recommendation_time

FROM Recommendations r

JOIN Songs s ON r.song_id = s.song_id

JOIN Users u ON r.user_id = u.user_id

ORDER BY r.user_id, r.recommendation_time DESC;

"""

runSql('Time-Based Recommendations Results : ', time_compr_query)

```

BDM_Assign_2_g23ai2100.ipynb

File Edit View Insert Runtime Tools Help All changes saved

RAM

Disk

Gemini

+ Code + Text

0s

JOIN Users u ON r.user_id = u.user_id
ORDER BY r.user_id, r.recommendation_time DESC;

runSql('Time-Based Recommendations Resultks : ', time_compr_query)

Write a python following pattern

user_id	name	listened_to	artist	rating	listen_time
1	Mickey	Evermore	Taylor Swift	4.5	2024-08-30 14:35:00
1	Mickey	Willow	Taylor Swift	4.2	None
1	Mickey	Yesterday	Beatles	3.9	2024-08-29 10:15:00
2	Minnie	Willow	Taylor Swift	4.7	None
2	Minnie	Yellow Submarine	Beatles	4.6	2024-08-28 09:20:00
2	Minnie	Hey Jude	Beatles	3.9	2024-08-27 16:45:00
3	Daffy	Willow	Taylor Swift	4.9	2024-08-26 12:30:00
3	Daffy	Evermore	Taylor Swift	2.9	None
3	Daffy	Yesterday	Beatles	NaN	None

Time-Based Recommendations Resultks

user_id	name	recommended_song	artist	genre	recommendation_time
1	Mickey	Hey Jude	Beatles	Classic	2024-12-15 15:05:29
1	Mickey	Yellow Submarine	Beatles	Classic	2024-12-15 15:05:29
2	Minnie	Yesterday	Beatles	Classic	2024-12-15 15:05:29
2	Minnie	Yesterday	Beatles	Classic	2024-12-15 15:05:29

0s

completed at 20:37

Submitted by: Jojo Joseph (G23AI2100)