Indian Institute of Technology Jodhpur
BDM

Assignment 2
BDM

Submitted by: Jojo Joseph (G23AI2100)

Google Colab :
https://colab.research.google.com/drive/1U6phW2lEz6SHlu99qoVvTRlyzg1FlxnT?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

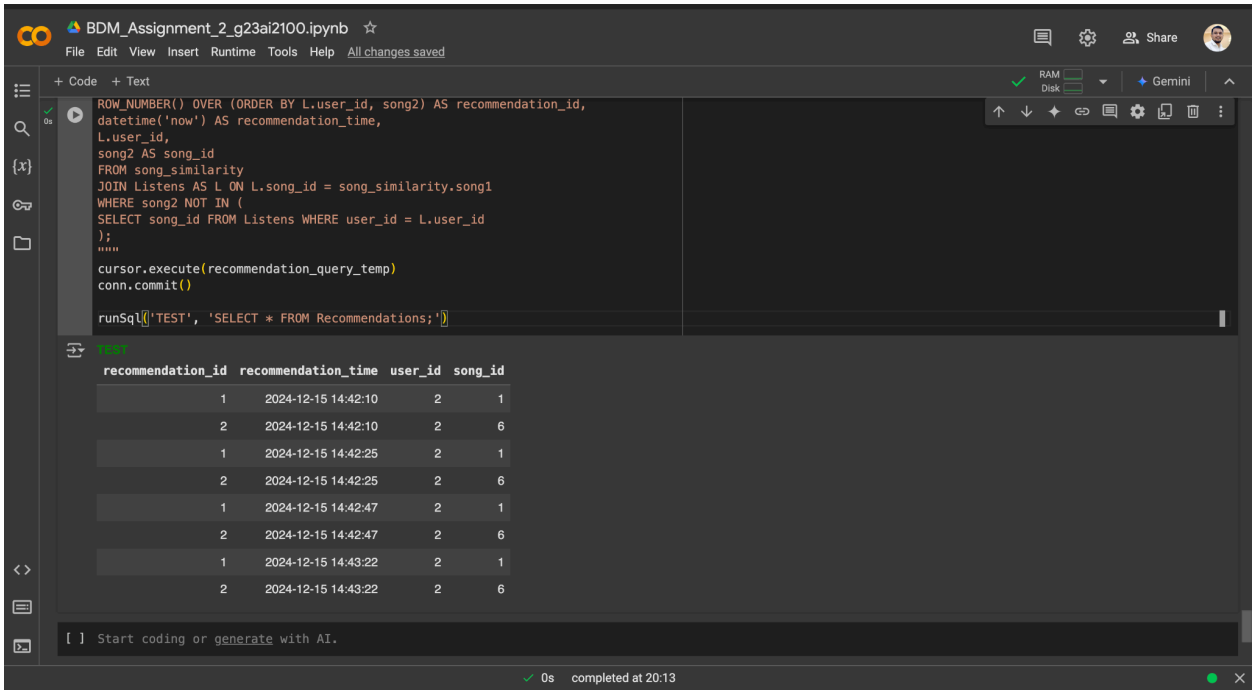
Ans

```
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
```

```
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 :

Submitted by: Jojo Joseph (G23AI2100)



#2 Generate the recommendaions 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

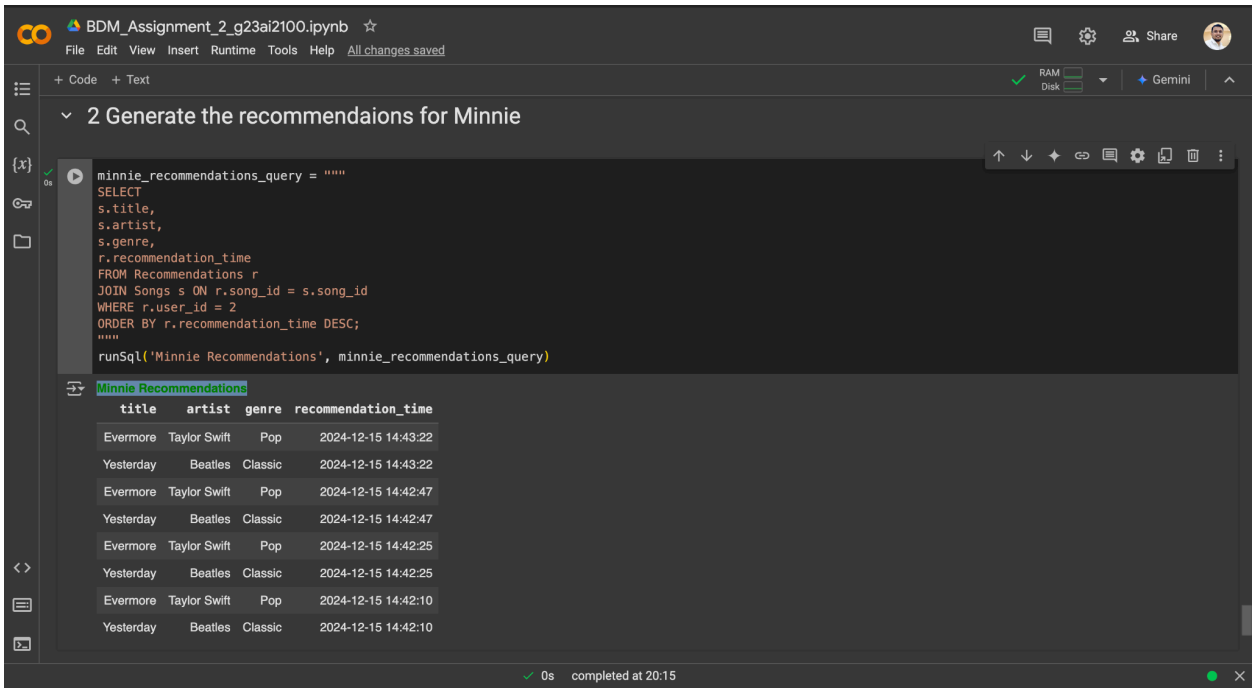
WHERE r.user_id = 2

ORDER BY r.recommendation_time DESC;

"""

runSql('Minnie Recommendations', minnie_recommendations_query)

Submitted by: Jojo Joseph (G23AI2100)



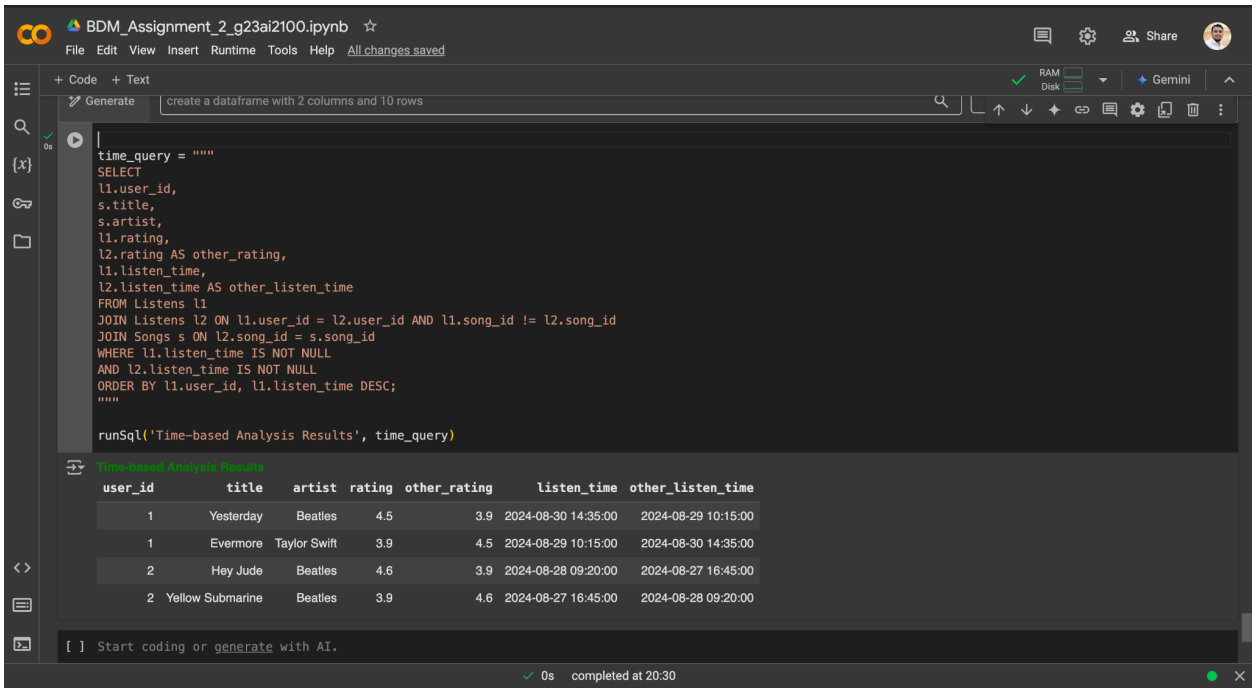
#3 Re-do the generation of recommendations now on the basis of listen time

Ans

```
time_query = """
SELECT
l1.user_id,
s.title,
s.artist,
l1.rating,
l2.rating AS other_rating,
l1.listen_time,
```

Submitted by: Jojo Joseph (G23AI2100)

```
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)
```



#4 Generate new recommendations

Ans

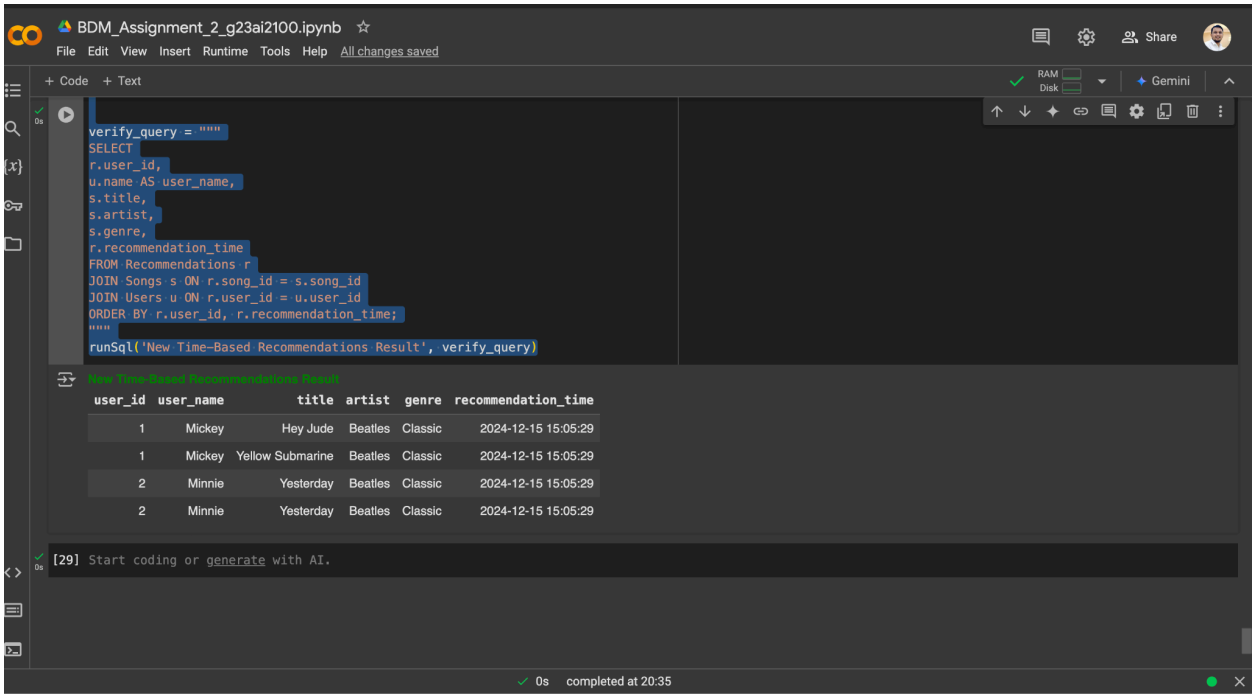
```

```

Submitted by: Jojo Joseph (G23AI2100)

```
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)
```



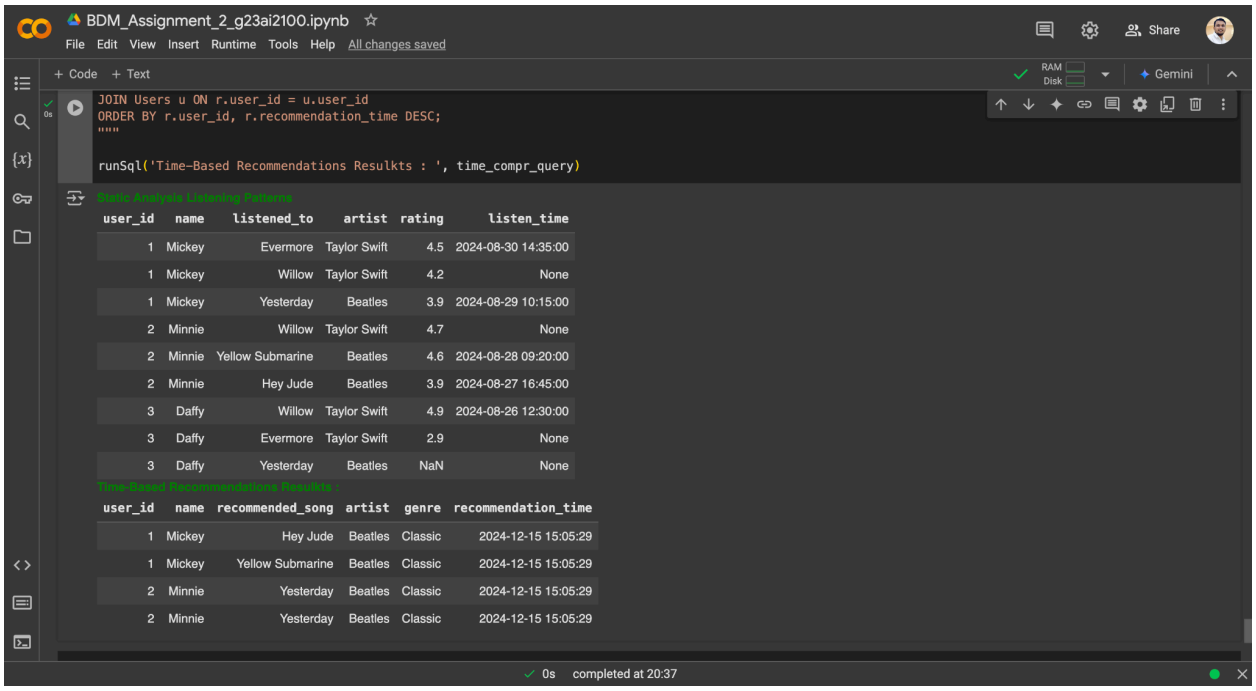
#5 What are the differences with the static method on #2 above

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;
```

Submitted by: Jojo Joseph (G23AI2100)

```
"""



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 Resulkts : ', time_compr_query)
```



Submitted by: Jojo Joseph (G23AI2100)