

GROUP_34 (SPOTIFY 12-M SONGS DATASET)

Anushka Churi (ac2224)
Meeti Dixit (md2335)
Neha Kulkarni (nk4349)
Shreya Sakpal (ss7807)

BIGDATA PROJECT PHASE - III

GROUP 34 - (SPOTIFY 12-M SONGS DATASET)

CLEANING AND INTEGRATION OF DATA

```
def pre_artists():
    r_artists = 'C:\\\\Users\\\\anush\\\\Documents\\\\CSCI-620 Big ' \
                'Data\\\\Movies\\\\title.basics.tsv '
    artists = pd.read_table(r_artists)
    artists = artists.drop_duplicates() # drop duplicates
    artists = artists.dropna() # filter null values
    artists.to_csv('C:\\\\Users\\\\anush\\\\Documents\\\\CSCI-620 Big ' \
                   'Data\\\\Assignment 2\\\\director.csv', index=False,
                   header=False)
    print('Actor table preprocessed successfully')
```

```
def pre_albums():
    r_albums = 'C:\\\\Users\\\\anush\\\\Documents\\\\CSCI-620 Big ' \
               'Data\\\\Movies\\\\title.basics.tsv '
    albums = pd.read_table(r_albums)
    albums = albums.drop(['album_group'], axis=1) # drop columns
    albums = albums.drop_duplicates() # drop duplicates
    albums = albums.dropna() # filter null values
    albums.to_csv('C:\\\\Users\\\\anush\\\\Documents\\\\CSCI-620 Big ' \
                  'Data\\\\Assignment 2\\\\director.csv', index=False,
                  header=False)
    print('Albums table preprocessed successfully')
```

GROUP_34 (SPOTIFY 12-M SONGS DATASET)

Anushka Churi (ac2224)
Meeti Dixit (md2335)
Neha Kulkarni (nk4349)
Shreya Sakpal (ss7807)

```
def pre_tracks():
    r_tracks = 'C:\\\\Users\\\\anush\\\\Documents\\\\CSCI-620 Big ' \
               'Data\\\\Movies\\\\title.basics.tsv '
    tracks = pd.read_table(r_tracks)
    tracks = tracks.drop(['disc_number', 'preview_url'], axis=1) # drop columns
    tracks = tracks.drop_duplicates() # drop duplicates
    tracks = tracks.dropna() # filter null values
    tracks.to_csv('C:\\\\Users\\\\anush\\\\Documents\\\\CSCI-620 Big ' \
                  'Data\\\\Assignment 2\\\\director.csv', index=False,
                  header=False)
    print('Tracks table preprocessed successfully')
```

```
def pre_audio_features():
    r_audio_features = 'C:\\\\Users\\\\anush\\\\Documents\\\\CSCI-620 Big ' \
                       'Data\\\\Movies\\\\title.basics.tsv '
    audio_features = pd.read_table(r_audio_features)
    # drop columns
    audio_features = audio_features.drop(['analysis_url', 'key', 'mode',
                                           'speechiness', 'tempo',
                                           'time_signature', 'valence'], axis=1)
    audio_features = audio_features.drop_duplicates() # drop duplicates
    audio_features = audio_features.dropna() # filter null values
    audio_features.to_csv('C:\\\\Users\\\\anush\\\\Documents\\\\CSCI-620 Big ' \
                          'Data\\\\Assignment 2\\\\director.csv', index=False,
                          header=False)
    print('Audio Features table preprocessed successfully')
```

GROUP_34 (SPOTIFY 12-M SONGS DATASET)

Anushka Churi (ac2224)
Meeti Dixit (md2335)
Neha Kulkarni (nk4349)
Shreya Sakpal (ss7807)

```
def pre_genres():
    r_genres = 'C:\\\\Users\\\\anush\\\\Documents\\\\CSCI-620 Big ' \
               'Data\\\\Movies\\\\title.basics.tsv'
    genres = pd.read_table(r_genres)
    genres = genres.drop_duplicates() # drop duplicates
    genres = genres.dropna() # filter null values
    genres.to_csv('C:\\\\Users\\\\anush\\\\Documents\\\\CSCI-620 Big ' \
                  'Data\\\\Assignment 2\\\\director.csv', index=False,
                  header=False)
    print('Genres table preprocessed successfully')
```

The program defines several functions, each of which preprocesses a specific table. These functions perform the following tasks:

- `pre_artists()`: Reads the TSV file for the artists table, drops any duplicate rows, and removes any rows with missing values. The resulting cleaned data is saved to a new CSV file.
- `pre_albums()`: Reads the TSV file for the albums table, drops the album_group column, drops any duplicate rows, and removes any rows with missing values. The resulting cleaned data is saved to a new CSV file.
- `pre_tracks()`: Reads the TSV file for the tracks table, drops several columns, drops any duplicate rows, and removes any rows with missing values. The resulting cleaned data is saved to a new CSV file.
- `pre_audio_features()`: Reads the TSV file for the audio features table, drops several columns, drops any duplicate rows, and removes any rows with missing values. The resulting cleaned data is saved to a new CSV file.
- `pre_genres()`: Reads the TSV file for the genres table, drops any duplicate rows, and removes any rows with missing values. The resulting cleaned data is saved to a new CSV file.

The main function calls each of these preprocessing functions in turn. The cleaned data is saved to CSV files, which can be used for further analysis.

Overall, the program uses the pandas library to read and manipulate the data, dropping duplicate rows and removing any rows with missing values. The resulting cleaned data is saved to new CSV files, which can be used for further analysis.

GROUP_34 (SPOTIFY 12-M SONGS DATASET)

Anushka Churi (ac2224)
Meeti Dixit (md2335)
Neha Kulkarni (nk4349)
Shreya Sakpal (ss7807)

LATTICE CREATION USING RELATIONAL MODEL

First, we create a table by joining our existing tables in order to perform Itemset Mining. Here, we have created a table entitled “Artist Album Collaboration” where we display the artists, their respective ids, their popularity and the albums they have created along with the album ids.

TABLE ARTIST ALBUM COLLABORATION

```
CREATE TABLE artist_album_collaboration AS
SELECT a.name AS artist, a.id AS artist_id, a.popularity, raa.album_id,
albums.name AS album_name
FROM artists a
JOIN r_albums_artists raa ON a.id = raa.artist_id
JOIN albums ON albums.id = raa.album_id;
```

The screenshot shows the Database Explorer and a Result grid. The Database Explorer pane displays the structure of the 'spotify' database, including the 'artist_album_collaboration' table. The Result grid shows the data for the first 19 rows of the table, with columns: artist, artist_id, popularity, album_id, and album_name. The data includes various artists like Xzibit, Erick Sermon, and Tash, along with their album details such as 'Restless' and 'Serial Killers, Vol. 1'.

artist	artist_id	popularity	album_id	album_name
Xzibit	4tuJQJicOnuZRLi8Fdp30u	69	6Fb0JTOKgRKWSn8kcV9NG3	Restless
Xzibit	4tuJQJicOnuZRLi8Fdp30u	69	7nQbnHqXAgUzA2fs1Rtc	Man VS Machine
Xzibit	4tuJQJicOnuZRLi8Fdp30u	69	1jlrb51uTL95Vqmw2G8rpA	40 Dayz & 40 Nightz
Xzibit	4tuJQJicOnuZRLi8Fdp30u	69	2LmMr54k3mm7ZF7ngOGG	At The Speed Of Life
Xzibit	4tuJQJicOnuZRLi8Fdp30u	69	8QZjFobMgxafxEfDMbMUU	Muthaf*cka
Xzibit	4tuJQJicOnuZRLi8Fdp30u	69	1Qdgtff31uNS6JjGgzkf6	Weapons of Mass Destruction (Explicit)
Xzibit	4tuJQJicOnuZRLi8Fdp30u	69	5AsGSM7HRgRThiSIVmZzIj	Napalm
Xzibit	4tuJQJicOnuZRLi8Fdp30u	69	2MyTu1cSabuF1vgrejeKH	Napalm (Deluxe)
Xzibit	4tuJQJicOnuZRLi8Fdp30u	69	3WfzCcfFtmEKfPpS8fViw	Full Circle
Xzibit	4tuJQJicOnuZRLi8Fdp30u	69	2SXIV2wTCAE7psLBHX7Vm	Full Circle
Xzibit	4tuJQJicOnuZRLi8Fdp30u	69	71UDegEFXN1Co7xeG0nza	Serial Killers, Vol. 1
Xzibit	4tuJQJicOnuZRLi8Fdp30u	69	40Crh0NopLR7vy05qKv97c	Man VS Machine
Xzibit	4tuJQJicOnuZRLi8Fdp30u	69	47oLQsf5KbPGz4lw5mrwIj	Man VS Machine
Xzibit	4tuJQJicOnuZRLi8Fdp30u	69	4069uZx5GLgsxjHX77tl50	Man VS Machine
Xzibit	4tuJQJicOnuZRLi8Fdp30u	69	3KDwwqSAAdAJqu7UDMLBT8H	Man VS Machine
Erick Sermon	2VX0o9LDIVmK1gpnwjdjp0J	54	5tUzxkBLeB7QivUw6wJsu	Music
Erick Sermon	2VX0o9LDIVmK1gpnwjdjp0J	54	118XHvNdaXRhoQW01BycA	No Pressure
Erick Sermon	2VX0o9LDIVmK1gpnwjdjp0J	54	0dIb0Kj24nJ6q7NWn966Zj	React
Erick Sermon	2VX0o9LDIVmK1gpnwjdjp0J	54	122VGCzTTh6600Iu8e4lka	Chilltown New York
Tash	22qf8cJRzBjIW02Jc4Je0r	48	7kdjXZ8zJkerK8A2caNLZD	Rap Life
Tash	22qf8cJRzBjIW02Jc4Je0r	48	7h6h6gtTh73jxzHijqbqYV	Rap Life
Tash	22qf8cJRzBjIW02Jc4Je0r	48	1h4vWPKTzYE8G13gNaXmWI	Rap Life
Tash	22qf8cJRzBjIW02Jc4Je0r	48	80RKAYR6IfxeFMyo7Hv5IZ	Rap Life
Craig Mack	4akj4uteQQrrGxhX9Rjuvf	55	7Myc3B0faiA5IvhDCdypl	Flava In Ya Ear Remix
Craig Mack	4akj4uteQrrGxhX9Rjuvf	55	6btFnL80q39o00csBrogwy	Project: Funk Da World
Craig Mack	4akj4ute00rrGxhX9Riuvf	55	7avi4oXSlvXS0r90RuFdJ	Get Down (Remix)

GROUP_34 (SPOTIFY 12-M SONGS DATASET)

Anushka Churi (ac2224)
Meeti Dixit (md2335)
Neha Kulkarni (nk4349)
Shreya Sakpal (ss7807)

LATTICE L1

```
create table L1 as
select artist_id as artist1, album_name, count(*) as count
from artist_album_collaboration
where popularity > 50
group by artist
having count(*) > 5;
```

The above query creates a table named "L1" using the "create table" command in SQL. The table has three columns: "artist1", "album_name", and "count". The "artist1" column contains the artist IDs, the "album_name" column contains the names of the albums created by the artists, and the "count" column contains the number of times the artist has collaborated on an album with other artists.

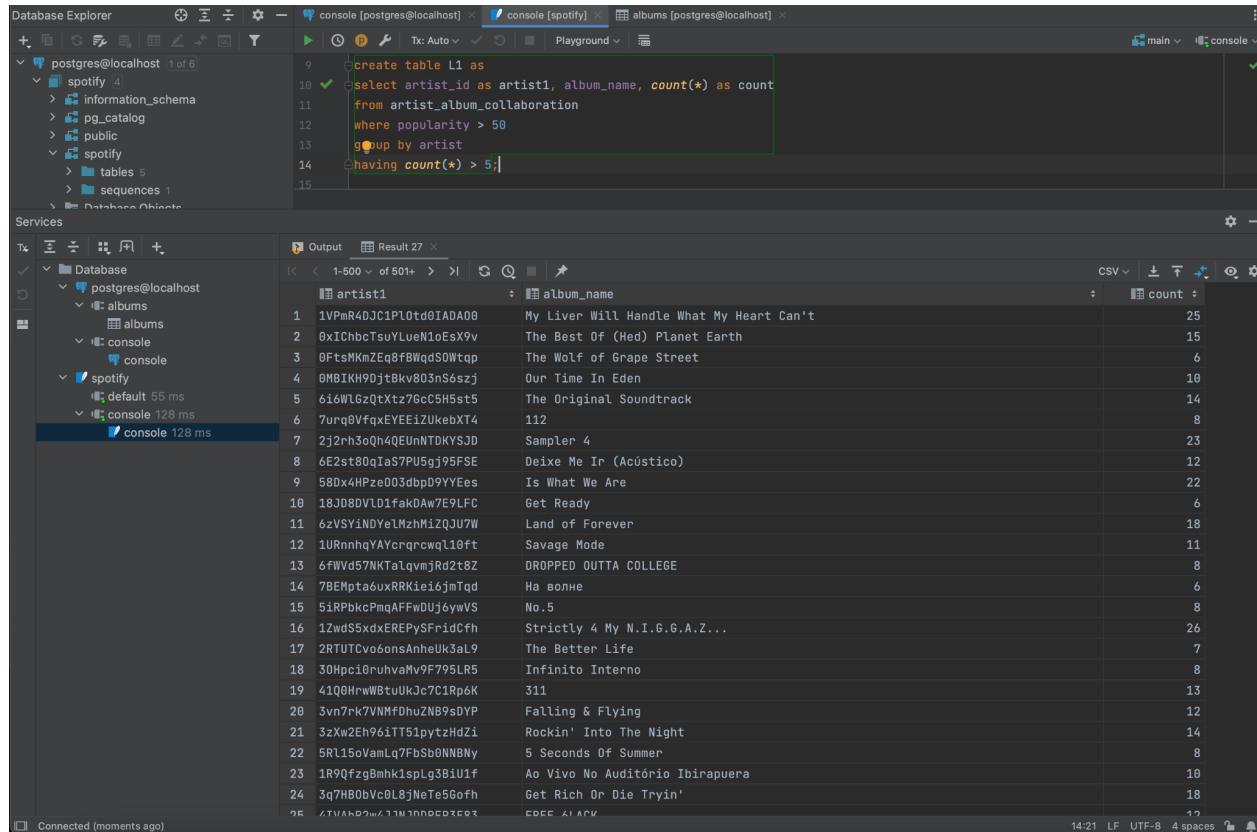
The query selects data from the "artist_album_collaboration" table where the popularity of the artist is greater than 50 and groups the data by the artist ID. **It then selects only those artists who have collaborated on at least 6 albums (count(*) > 5)** and displays the artist ID, album name, and count in the L1 table.

To create lattices using the above query, we need to perform itemset mining on the "L1" table. Itemset mining is a data mining technique that involves finding frequent itemsets in a transactional database. In this case, we can consider each album as a transaction, and the artists who collaborated on the album as items.

We can create lattices using the frequent itemsets obtained from the itemset mining process.

GROUP_34 (SPOTIFY 12-M SONGS DATASET)

Anushka Churi (ac2224)
 Meeti Dixit (md2335)
 Neha Kulkarni (nk4349)
 Shreya Sakpal (ss7807)



The screenshot shows a PostgreSQL database interface with multiple tabs open. The 'Database Explorer' tab shows a connection to 'postgres@localhost' with a schema named 'spotify'. The 'console [postgres@localhost]' tab contains a SQL query:

```

create table L1 as
select artist_id as artist1, album_name, count(*) as count
from artist_album_collaboration
where popularity > 50
group by artist
having count(*) > 5;

```

The 'Output' tab displays the results of this query, which lists albums with more than 5 collaborations, ordered by count in descending order. The results are as follows:

artist1	album_name	count
1VPMr4D3C1PL0td0IADA00	My Liver Will Handle What My Heart Can't	25
0xICbhcTsYtLueNloEsX9v	The Best Of (Hed) Planet Earth	15
0FtsMKmZEq8fBWqqSOwTqP	The Wolf of Grape Street	6
0MBIKH9djBkv803nS6szj	Our Time In Eden	10
5i6WL6zQxttz76cc5H5st5	The Original Soundtrack	14
7urq8VfqxEYEEiZUkebXT4	112	8
2j2rh3oQh4QEunNTDKYSJ0	Sampler 4	23
6E2st80qTaS7PU6gj95FSE	Deixe Me Ir (Acústico)	12
580x4HPze03dbp9YyEes	Is What We Are	22
18JD8VDL0fakDAw7E9LFC	Get Ready	6
6zSVYiNDYeIhMzIQJU7W	Land of Forever	18
1URnnhqYAYcrqrcwql10ft	Savage Mode	11
6fWWd57KtaQwmjRd2t8Z	DROPPED OUTTA COLLEGE	8
7BEMpta6uxRRKie16jmTqd	На волне	6
5iRPbkcpmqAFwDUjywVS	No.5	8
1ZwdS5xdxEREPySFrldCfh	Strictly 4 My N.I.G.G.A.Z...	26
2RTUTCvo6onsAnheUK3aL9	The Better Life	7
30Hpc10ruhvaMw9f795LR5	Infinito Interno	8
41Q0HrwBtuUkjc7C1Rp6K	311	13
3vnTrk7VNmfhuZNBy9sDYP	Falling & Flying	12
3zXw2Eh96iT51pytzHdZi	Rockin' Into The Night	14
5RL15oVamLq7FbSb0NNBNy	5 Seconds Of Summer	8
1R9QfzgBmhk1spLg3BiUif	Ao Vivo No Auditório Ibirapuera	10
3g7HB0bvCcLBjN6te5Gofh	Get Rich Or Die Tryin'	18
95	CODE XIAOMI	12

LATTICE L2

```

create table L2 as
select aac1.artist_id as artist1, aac2.artist_id as artist2, aac1.album_name,
count(*) as count
from artist_album_collaboration aac1
join artist_album_collaboration aac2
on aac1.album_id = aac2.album_id
where artist1 < artist2
and aac1.popularity > 50 and aac2.popularity > 50
group by artist1, artist2
having count(*) > 5;

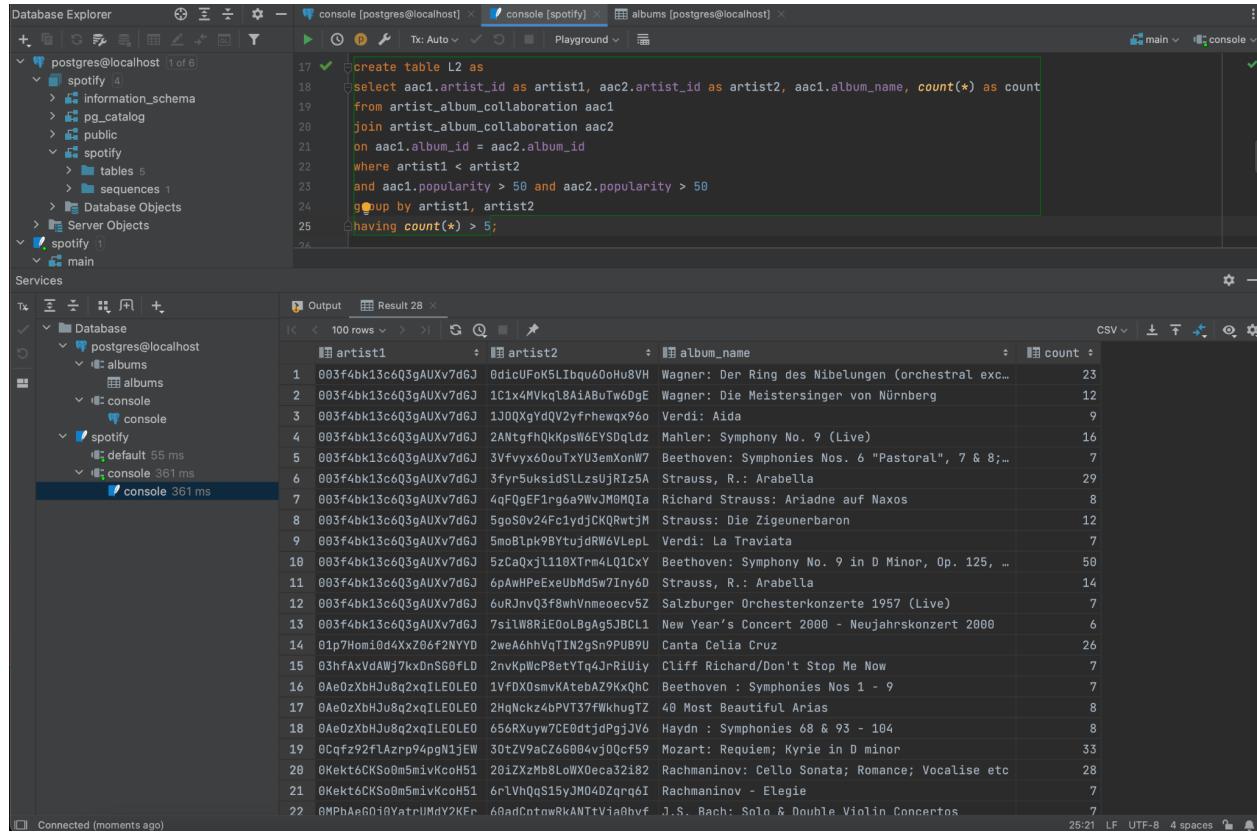
```

The query selects data from the "artist_album_collaboration" table by joining the table to itself using the "join" command. The join condition is that both rows being joined have the same "album_id", which means they are both collaborations on the same album. The query also filters the results to only include collaborations between two artists with a popularity greater than 50.

GROUP_34 (SPOTIFY 12-M SONGS DATASET)

Anushka Churi (ac2224)
 Meeti Dixit (md2335)
 Neha Kulkarni (nk4349)
 Shreya Sakpal (ss7807)

The resulting data is then grouped by the IDs of the two artists, and only those pairs of artists who have collaborated on at least 6 albums are selected (having count(*) > 5). The resulting data is then inserted into the "L2" table.



The screenshot shows the pgAdmin interface with the Database Explorer and a query editor window. The query editor contains the SQL code for creating table L2:

```

create table L2 as
select aac1.artist_id as artist1, aac2.artist_id as artist2, aac1.album_name, count(*) as count
from artist_album_collaboration aac1
join artist_album_collaboration aac2 on aac1.album_id = aac2.album_id
join artist_album_collaboration aac3 on aac2.album_id = aac3.album_id
where artist1 < artist2
and aac1.popularity > 50 and aac2.popularity > 50
group by artist1, artist2
having count(*) > 5;

```

The output window displays the results of the query, showing 28 rows of data with columns: artist1, artist2, album_name, and count. The data includes various classical compositions by different artists like Wagner, Verdi, Mahler, Beethoven, Strauss, and Rachmaninov.

	artist1	artist2	album_name	count
1	003f4bk13c6Q3gAUxV7dGJ	0dicUFoK5LIBqu60oHu8VH	Wagner: Der Ring des Nibelungen (orchestral exc...	23
2	003f4bk13c6Q3gAUxV7dGJ	1C1x4NVkql8AiABu7w0dg	Wagner: Die Meistersinger von Nürnberg	12
3	003f4bk13c6Q3gAUxV7dGJ	1J0QXgYdQV2yfrhewqx96o	Verdi: Aida	9
4	003f4bk13c6Q3gAUxV7dGJ	2ANtgefhnkkpSw6EYSDqldz	Mahler: Symphony No. 9 (Live)	16
5	003f4bk13c6Q3gAUxV7dGJ	3Vfvyx60ouTxU3emXonW7	Beethoven: Symphonies Nos. 6 "Pastorale", 7 & 8;...	7
6	003f4bk13c6Q3gAUxV7dGJ	3Tyr5uksidSLzsUjRiz5A	Strauss, R.: Arabella	29
7	003f4bk13c6Q3gAUxV7dGJ	4qFQgEF1rg6a9NvJM8MQiA	Richard Strauss: Ariadne auf Naxos	8
8	003f4bk13c6Q3gAUxV7dGJ	5go5v0v24FcIydyjCQQRwtfjM	Strauss: Die Zigeunerbaron	12
9	003f4bk13c6Q3gAUxV7dGJ	5m0blpk9BYtujdrW6VLepL	Verdi: La Traviata	7
10	003f4bk13c6Q3gAUxV7dGJ	5zCaQxjll110XTm4LQ1cXY	Beethoven: Symphony No. 9 in D Minor, Op. 125, ...	50
11	003f4bk13c6Q3gAUxV7dGJ	6pAwHPeExeUMd5w7Iny6D	Strauss, R.: Arabella	14
12	003f4bk13c6Q3gAUxV7dGJ	6uRJnvQ3f8whVmmeocv5Z	Salzburger Orchesterkonzerte 1957 (Live)	7
13	003f4bk13c6Q3gAUxV7dGJ	7silWBRiE0oLbgAg5JBCl1	New Year's Concert 2000 - Neujahrskonzert 2000	6
14	01p7Homi0d4XzX06f2NYyD	2weA6nhVqTIN2gSn9PU89U	Canta Celia Cruz	26
15	03hfAxVdAWJ7kx0nSG0fLD	2nvkpwCp8etvTy4qJrRiUiY	Cliff Richard/Don't Stop Me Now	7
16	0Ae0zXbHJu8q2xqILEOLEO	1VfdDX0smvKAtabAZ9KxQhC	Beethoven : Symphonies Nos 1 - 9	7
17	0Ae0zXbHJu8q2xqILEOLEO	2hqCckzabPVT37fWkhugTZ	Most Beautiful Arias	8
18	0Ae0zXbHJu8q2xqILEOLEO	656RXuyw7CE0dtjdPgjJV6	Haydn : Symphonies 68 & 93 - 104	8
19	0Cqfz92flzrp94pgn1jEW	30tzV9aCZ6G004vjoQcf59	Mozart: Requiem; Kyrie in D minor	33
20	0Kekt6CKSo0m5mivKcoH51	28izXzMb8loWX0eca32i82	Rachmaninov: Cello Sonata; Romance; Vocalise etc	28
21	0Kekt6CKSo0m5mivKcoH51	6rlvhqd51syJM04D2qrq6I	Rachmaninov - Elegie	7
22	6MPHa6G0i0YatnIIMy2KFr	60anCtowRkANTtIViaRhvf	J. S. Bach: Solo & Double Violin Concertos	7

LATTICE L3

```

create table L3 as
select aac1.artist_id as artist1, aac2.artist_id as artist2, aac3.artist_id as artist3, aac1.album_name, count(*) as count
from artist_album_collaboration aac1
join artist_album_collaboration aac2 on aac1.album_id = aac2.album_id
join artist_album_collaboration aac3 on aac2.album_id = aac3.album_id
where artist1 < artist2 and artist2 < artist3
and aac1.popularity > 50 and aac2.popularity > 50 and aac3.popularity > 50
group by artist1, artist2, artist3
having count(*) > 5;

```

GROUP_34 (SPOTIFY 12-M SONGS DATASET)

Anushka Churi (ac2224)

Meeti Dixit (md2335)

Neha Kulkarni (nk4349)

Shreya Sakpal (ss7807)

The "artist1", "artist2", and "artist3" columns from the above query contain the IDs of three collaborating artists who have collaborated on an album. The "album_name" column contains the name of the album they collaborated on, and the "count" column contains the number of times the three artists collaborated on an album.

The query selects data from the "artist_album_collaboration" table by joining it to itself three times using the "join" command. The join conditions are that each row being joined has the same "album_id", which means they are all collaborations on the same album, and that the IDs of the three artists are in increasing order.

The query also filters the results to only include collaborations between three artists with a popularity greater than 50. The resulting data is then grouped by the IDs of the three artists, and only those triplets of artists who have collaborated on at least 6 albums are selected (having count(*) > 5). The resulting data is then inserted into the "L3" table.

The screenshot shows a Database Explorer interface with several tabs open. The main tab displays a SQL query for creating a table L3 and selecting data from the artist_album_collaboration table. The results tab shows 17 rows of data, each containing artist1, artist2, artist3, album_name, and count. The data includes various classical compositions by Wagner, Strauss, Beethoven, and Mozart, among others.

artist1	artist2	artist3	album_name	count
1 003f4bk13c6Q3gAUXv7dGJ	0dicUFoK5L1bqu0oHu8VH	1C1x4MVkql8AiAbuTw6DgE	Wagner: Der Ring des Nibelungen (orchestral e...	18
2 003f4bk13c6Q3gAUXv7dGJ	0dicUFoK5L1bqu0oHu8VH	6pAwHPeExeUbMd5w7Iny60	Strauss, R.: Der Rosenkavalier	7
3 003f4bk13c6Q3gAUXv7dGJ	3Vfvxy60ouTxYU3emXonW7	3fyrl5ksidSLLzsUjRlZ5A	Beethoven: Symphonies Nos. 6 "Pastoral", 7 & ...	7
4 0NK0gy9j6n30AiAbuTw6DgE	5MspfQqdbdw6ax36Xqum	7gc0Q1lkfkbuL5Mt0jBqkg	Sweet Dreams (Are Made Of This)	15
5 1C1x4MVkql8AiAbuTw6DgE	szCaQxjl110XTrm4LQ1CXY	6uRJnvQ3f8whVnmeoevcv5Z	Wagner: Parsifal	7
6 2ANTgfhQKPswe6YSQdldz	5zCaQxjl110XTrm4LQ1CXY	6uRJnvQ3f8whVnmeoevcv5Z	Mahler: Symphony No.9	7
7 5vBh0nve44zwvVF5KtCwA	6NUhQz7eEsZvjEHTKHux9	6qp886A2aBjnRF5tEs4Ht	Mozart: The Piano Concertos (12 CDs, Vol.7 of...	20
8 5vBh0nve44zwvVF5KtCwA	6NUhQz7eEsZvjEHTKHux9	6uRJnvQ3f8whVnmeoevcv5Z	Mozart: The Piano Concertos (12 CDs, Vol.7 of...	20
9 5vBh0nve44zwvVF5KtCwA	6NUhQz7eEsZvjEHTKHux9	77CaCn32H4m0MQA7UELzfF	Mozart: Piano Concertos Nos. 20 & 24; Concert...	21
10 5vBh0nve44zwvVF5KtCwA	6qp886A2aBjnRF5tEs4Ht	6uRJnvQ3f8whVnmeoevcv5Z	Mozart: The Piano Concertos (12 CDs, Vol.7 of...	20
11 5vBh0nve44zwvVF5KtCwA	6qp886A2aBjnRF5tEs4Ht	77CaCn32H4m0MQA7UELzfF	Mozart: The Piano Concertos (12 CDs, Vol.7 of...	20
12 5vBh0nve44zwvVF5KtCwA	6uRJnvQ3f8whVnmeoevcv5Z	77CaCn32H4m0MQA7UELzfF	Mozart: The Piano Concertos (12 CDs, Vol.7 of...	20
13 5zCaQxjl110XTrm4LQ1CXY	6pAwHPeExeUbMd5w7Iny60	6uRJnvQ3f8whVnmeoevcv5Z	Strauss, R.: Also sprach Zarathustra; Till Eu...	9
14 6NUhQz7eEsZvjEHTKHux9	6qp886A2aBjnRF5tEs4Ht	6uRJnvQ3f8whVnmeoevcv5Z	Mozart: The Piano Concertos (12 CDs, Vol.7 of...	20
15 6NUhQz7eEsZvjEHTKHux9	6qp886A2aBjnRF5tEs4Ht	77CaCn32H4m0MQA7UELzfF	Mozart: The Piano Concertos (12 CDs, Vol.7 of...	20
16 6NUhQz7eEsZvjEHTKHux9	6uRJnvQ3f8whVnmeoevcv5Z	77CaCn32H4m0MQA7UELzfF	Mozart: The Piano Concertos (12 CDs, Vol.7 of...	20
17 6qp886A2aBjnRF5tEs4Ht	6uRJnvQ3f8whVnmeoevcv5Z	77CaCn32H4m0MQA7UELzfF	Mozart: The Piano Concertos (12 CDs, Vol.7 of...	20

GROUP_34 (SPOTIFY 12-M SONGS DATASET)

Anushka Churi (ac2224)
Meeti Dixit (md2335)
Neha Kulkarni (nk4349)
Shreya Sakpal (ss7807)

APRIORI ALGORITHM TO GENERATE LATTICES

The screenshot shows the PyCharm IDE interface. The top navigation bar includes 'Project', 'File', 'Edit', 'Run', 'View', 'Tools', 'Help', and 'PyCharm Help'. Below the navigation bar, the 'Project' tool window displays a 'pythonProject' folder containing files like 'bigdata.py', 'bd9.py', and 'parking.py'. The main code editor window contains the Apriori algorithm code. The bottom terminal window shows the execution of the script, outputting frequent itemsets from L1 to L6, and a final message indicating the process finished with exit code 0.

```
while True:
    cur.execute(f"DROP TABLE IF EXISTS l{n};")
    curr_level = f"l{n}"
    current_query = f"CREATE TABLE {curr_level} AS SELECT "
    for i in range(1, n + 1):
        current_query += f" aac{i}.artist_id AS artist{i},"
    current_query += f" aac1.album_name, COUNT(*) AS count FROM "
    for i in range(0, n):
        if i == 0:
            current_query += f" artist_album_collaboration AS aac{i + 1} JOIN"
        else:
            current_query += f" artist_album_collaboration AS aac{i + 1} ON aac{i}.album_id = aac{i + 1}.album_id AND artist{i} < artist{i + 1}"
    current_query += f" GROUP BY "
    for i in range(1, n + 1):
        if i != n:
            current_query += f" artist{i},"
        else:
            current_query += f" artist{i}"
    current_query += f" HAVING count(*) > {min_support};"
    cur.execute(current_query)
```

```
Connected!
L1: 3964 frequent itemsets
L2: 180 frequent itemsets
L3: 17 frequent itemsets
L4: 5 frequent itemsets
L5: 1 frequent itemsets
L6: 0 frequent itemsets

Process finished with exit code 0
```

ARTISTS IN THE LAST NON-ZERO LATTICE (L5)

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema, including the 'spotify' database and its tables: 'albums', 'artists', 'audio_features', 'genres', and 'tracks'. The main query editor window contains a SQL query to select names from five joined artist tables based on a common album ID. The results pane shows a single row of data corresponding to the last non-zero lattice itemset (L5).

```
1 ✓  select a.name as actor1_name,
2      a2.name as actor2_name,
3      a3.name as actor3_name,
4      a4.name as actor4_name,
5      a5.name as actor5_name,
6      l.album_name
7      from l5 l
8      join artists a on l.artist1 = a.id
9      join artists a2 on l.artist2 = a2.id
10     join artists a3 on l.artist3 = a3.id
11     join artists a4 on l.artist4 = a4.id
12     join artists a5 on l.artist5 = a5.id;
```

actor1_name	actor2_name	actor3_name	actor4_name	actor5_name	album_name
Alfred Brendel	Sir Neville Marriner	Semyon Bychkov	Berliner Philharmoniker	Academy of St. Martin in the Fields	Mozart: The Piano Concertos (12 CDs, Vol.7 of 45)

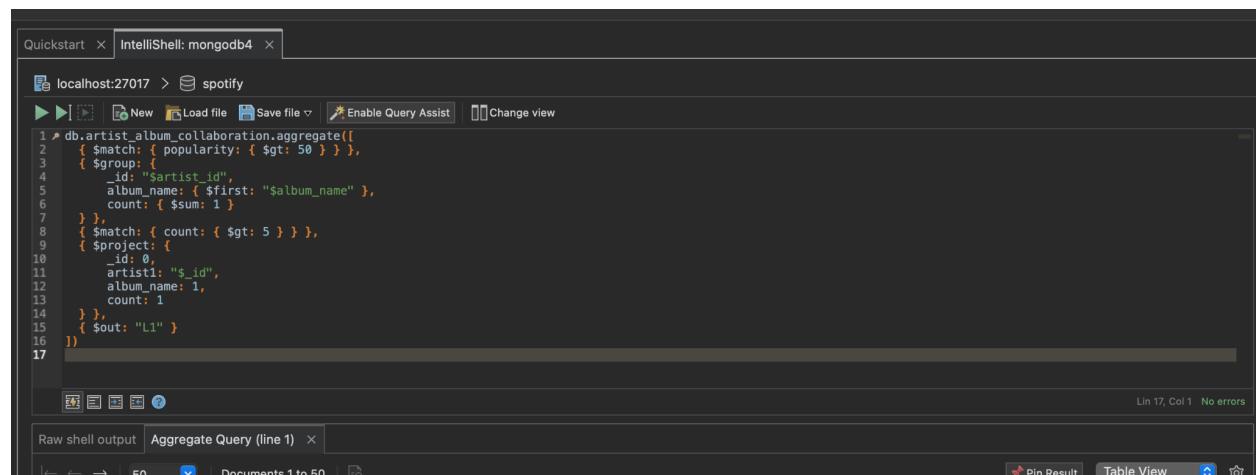
GROUP_34 (SPOTIFY 12-M SONGS DATASET)

Anushka Churi (ac2224)
Meeti Dixit (md2335)
Neha Kulkarni (nk4349)
Shreya Sakpal (ss7807)

LATTICE CREATION USING DATA-ORIENTED MODEL

LATTICE L1

```
db.artist_album_collaboration.aggregate([
  { $match: { popularity: { $gt: 50 } } },
  { $group: {
    _id: "$artist_id",
    album_name: { $first: "$album_name" },
    count: { $sum: 1 }
  }},
  { $match: { count: { $gt: 5 } } },
  { $project: {
    _id: 0,
    artist1: "$_id",
    album_name: 1,
    count: 1
  }},
  { $out: "L1"
})
```



The screenshot shows the MongoDB shell interface in IntelliJShell: mongodb4. The query is being run against the 'spotify' database on port 27017. The code is identical to the one in the previous block, defining an aggregate pipeline to find artists with more than 5 albums and output it to a collection named 'L1'. The status bar at the bottom right indicates 'Line 17, Col 1 No errors'.

```
localhost:27017 > db.artist_album_collaboration.aggregate([
  { $match: { popularity: { $gt: 50 } } },
  { $group: {
    _id: "$artist_id",
    album_name: { $first: "$album_name" },
    count: { $sum: 1 }
  }},
  { $match: { count: { $gt: 5 } } },
  { $project: {
    _id: 0,
    artist1: "$_id",
    album_name: 1,
    count: 1
  }},
  { $out: "L1"
})
```

GROUP_34 (SPOTIFY 12-M SONGS DATASET)

Anushka Churi (ac2224)
Meeti Dixit (md2335)
Neha Kulkarni (nk4349)
Shreya Sakpal (ss7807)

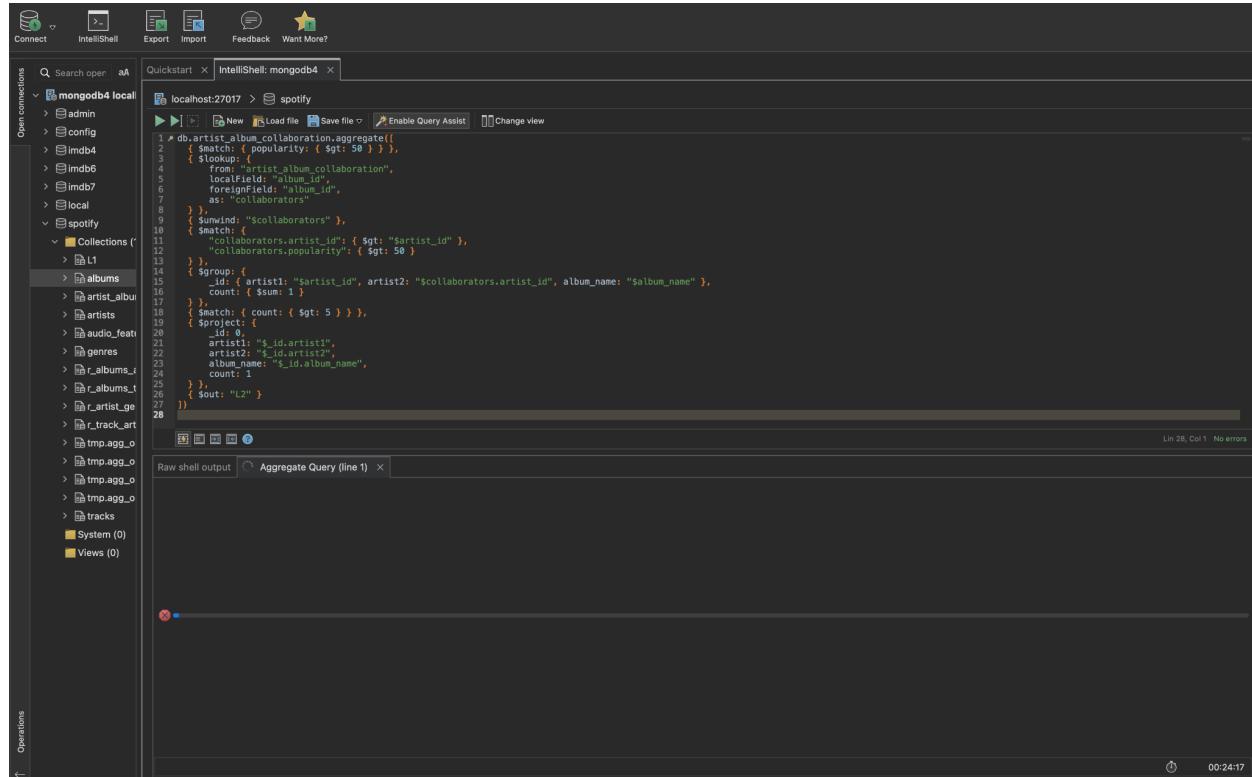
LATTICE L2

```
db.artist_album_collaboration.aggregate([  
  { $match: { popularity: { $gt: 50 } } },  
  { $lookup: {  
    from: "artist_album_collaboration",  
    localField: "album_id",  
    foreignField: "album_id",  
    as: "collaborators"  
  } },  
  { $unwind: "$collaborators" },  
  { $match: {  
    "collaborators.artist_id": { $gt: "$artist_id" },  
    "collaborators.popularity": { $gt: 50 }  
  } },  
  { $group: {  
    _id: { artist1: "$artist_id", artist2: "$collaborators.artist_id", album_name: "$album_name" },  
    count: { $sum: 1 }  
  } },  
  { $match: { count: { $gt: 5 } } },  
  { $project: {  
    _id: 0,  
    artist1: "$_id.artist1",  
    artist2: "$_id.artist2",  
    album_name: "$_id.album_name",  
  } }
```

GROUP_34 (SPOTIFY 12-M SONGS DATASET)

Anushka Churi (ac2224)
Meeti Dixit (md2335)
Neha Kulkarni (nk4349)
Shreya Sakpal (ss7807)

```
count: 1  
  
},  
  
{ $out: "L2" }  
  
])
```



In the above lattice creation, we see that the query takes a lot of time for execution. From this we observe that itemset mining in a relational model is typically faster than in a document-oriented model due to the relational model's emphasis on structured data and built-in support for join operations. This is because itemset mining often requires comparing items across different records or collections, and relational databases are designed to efficiently execute these types of operations.

On the other hand, document-oriented databases like MongoDB are optimized for flexibility and scalability, rather than structured data and join operations. This means that executing itemset mining operations in MongoDB can be more time-consuming and

GROUP_34 (SPOTIFY 12-M SONGS DATASET)

Anushka Churi (ac2224)
Meeti Dixit (md2335)
Neha Kulkarni (nk4349)
Shreya Sakpal (ss7807)

resource-intensive than in a relational database. MongoDB can struggle with large datasets, especially when the data has complex nested structures or requires significant preprocessing to make it suitable for mining.

One reason MongoDB may take longer to execute lattices than a relational database is because MongoDB requires more complex aggregation pipelines to group and aggregate data. These pipelines may involve multiple stages of filtering, sorting, and grouping, which can be time-consuming to execute. Additionally, MongoDB may require more memory and disk space to store and process the data than a relational database, which can slow down the mining process.

Overall, while MongoDB and other document-oriented databases can be used for itemset mining, they may not be the most efficient choice for this task compared to a relational database. However, the choice of database depends on the specific needs and requirements of the mining task, and it is always important to carefully evaluate and compare the performance of different databases before making a final decision.