Team No- 6
Team Member Name -
Aryan Sharma (BT21GCS161)
Raja Pandey (BT21GCS323)
Rishabh Gupta (BT21GCS020)
Ranjeev Singh (BT21GCS080)

Music Data Analysis For a Music App Company

Project Deliverables for Dimensional and NoSQL Databases:

- 1. To create a Data Warehouse Instance document, along with Necessary Tables.

  a. To create an Instance of Oracle
  b. Design the Tables Architecture
  c. Populate data in the Tables, as per the Project Requirement. Only 2 entries per table.
  d. Create a document for Roll-up and Drill-down, to get higher level information and granular level information, respectively. Use Slicing &amp; dicing Strategies.

- 2. prepare the Schema-Document of below SCD's. (Specify the Columns out of the above Tables for better Candidates for it).

  a. Type I changing Dimension
  b. Type II changing Dimension
  c. Type III changing Dimension

- 3. To prepare documents for Vertical and Horizontal Fragmentation. Consider 3 Nodes. - for 2 Tables

- 4. Configure REDIS, and Create the Keys and Values, needed for the Project. – like as PATH etc., suggest some Keys and the corresponding Values it can take in it.

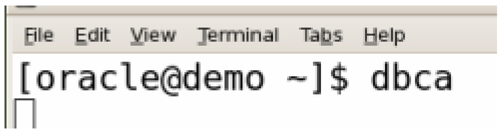
- 5. To create architecture using FACT and DIMENSIONS as per Star Schema. Consider the Key-Performance-Indicators (KPIs) – like as the Percentage of Profitability in shares, the Time takes to give those returns etc.

- 6. To create architecture using FACT and DIMENSIONS as per Snowflake Schema.

- 7. Configure the Collections in MongoDB, for Your Project Requirements – Here Create tables in such a way, that there are no Joins needed, to pull out the same Data. Write information about Variables setup, like PATH Variable etc.

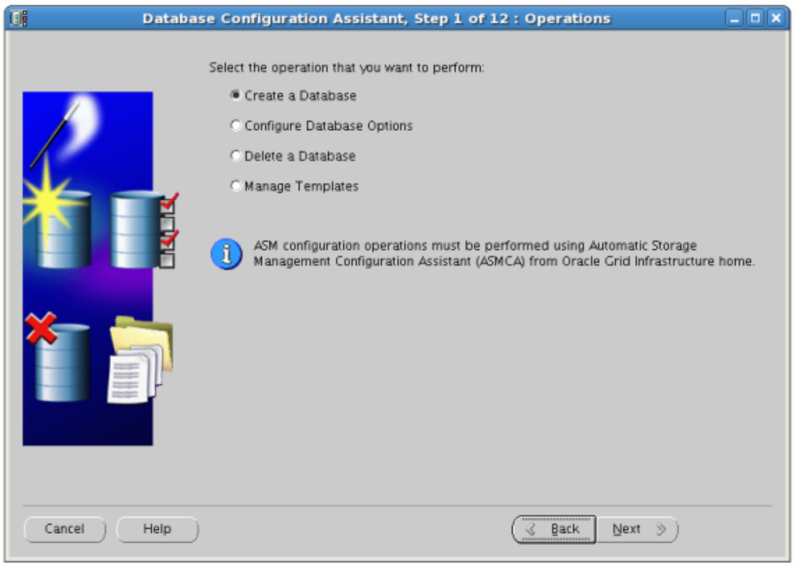
1.1) database configuration assistant is used to perform any of the following tasks:

To create the Database configuration assistant type dbca in the terminal



```
[oracle@demo ~]$ dbca
```

i) *Create a new database*

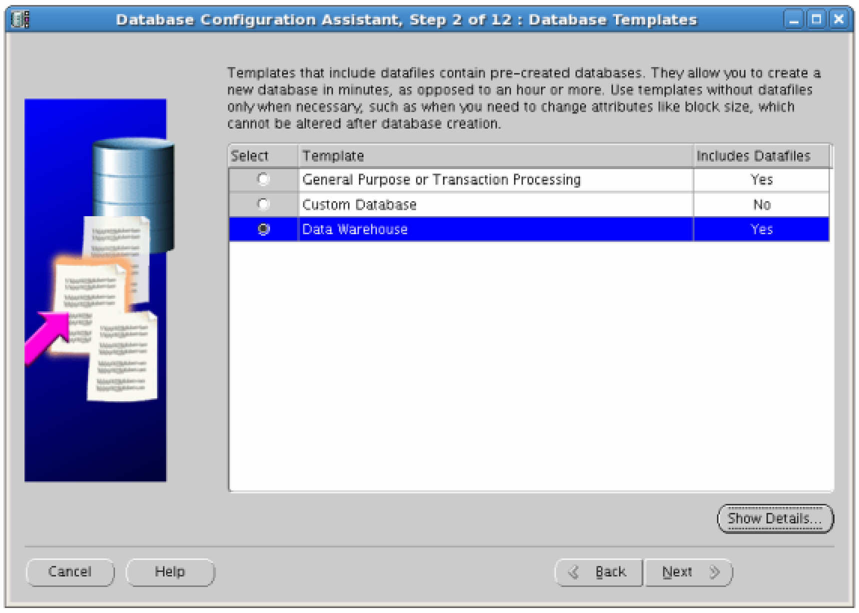

Database Configuration Assistant, Step 1 of 12 : Operations

Select the operation that you want to perform:

● Create a Database
○ Configure Database Options
○ Delete a Database
○ Manage Templates

ASM configuration operations must be performed using Automatic Storage Management Configuration Assistant (ASMCA) from Oracle Grid Infrastructure home.

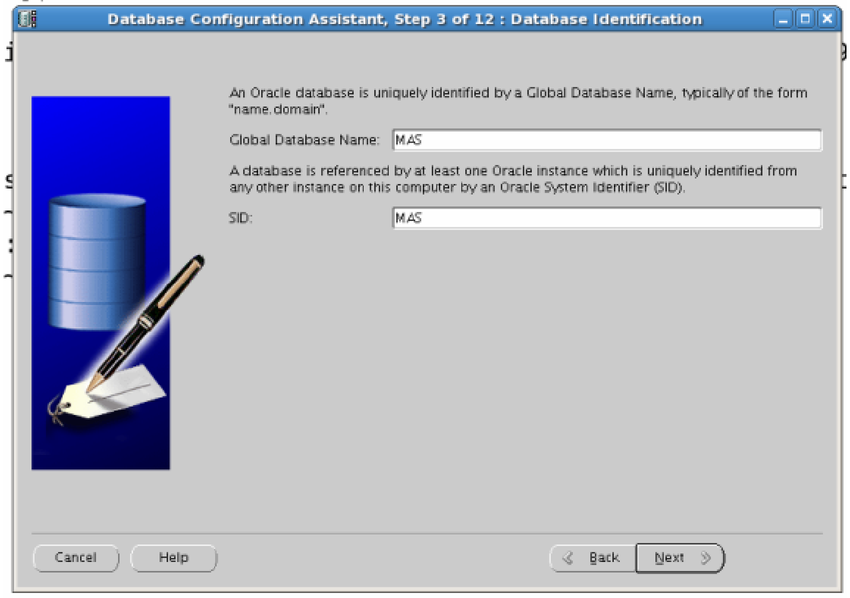
ii) *Select database warehouse*


Database Configuration Assistant, Step 2 of 12 : Database Templates

Templates that include datafiles contain pre-created databases. They allow you to create a new database in minutes, as opposed to an hour or more. Use templates without datafiles only when necessary, such as when you need to change attributes like block size, which cannot be altered after database creation.

Select	Template	Includes Datafiles
○	General Purpose or Transaction Processing	Yes
○	Custom Database	No
●	Data Warehouse	Yes

iii) *Database Identification*


Database Configuration Assistant, Step 3 of 12 : Database Identification

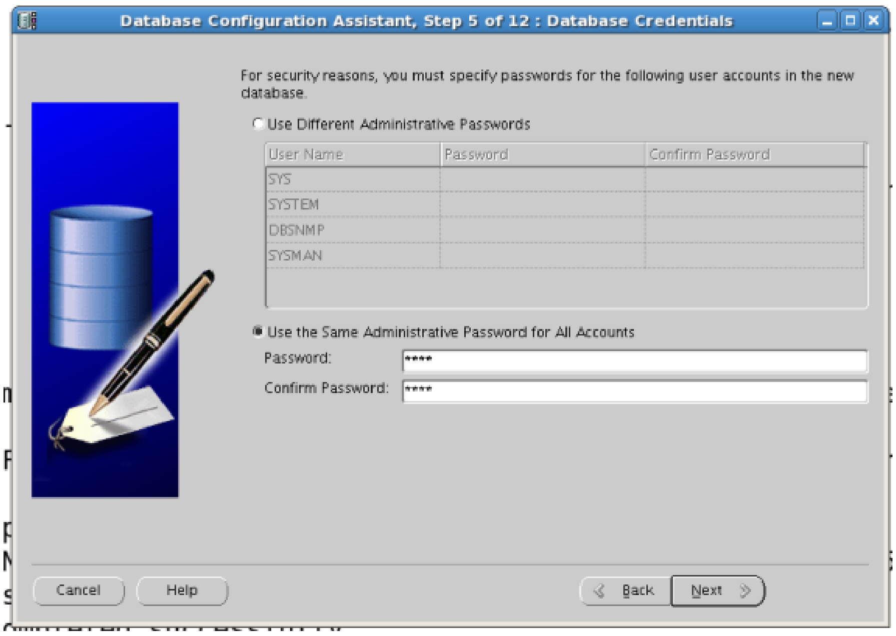
An Oracle database is uniquely identified by a Global Database Name, typically of the form "name.domain".

Global Database Name: MAS

A database is referenced by at least one Oracle instance which is uniquely identified from any other instance on this computer by an Oracle System Identifier (SID).

SID: MAS

iv) *Set the Database Credentials*



Database Configuration Assistant, Step 5 of 12 : Database Credentials

For security reasons, you must specify passwords for the following user accounts in the new database.

○ Use Different Administrative Passwords

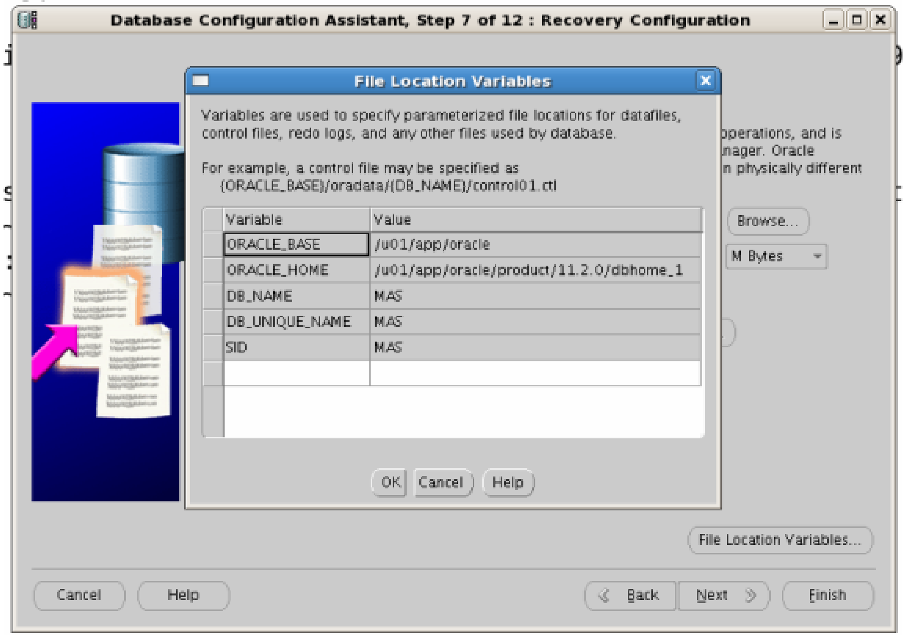
User Name	Password	Confirm Password
SYS		
SYSTEM		
DBSNMP		
SYSMAN		

● Use the Same Administrative Password for All Accounts

Password: ****

Confirm Password: ****

v) *File location Variable*



Database Configuration Assistant, Step 7 of 12 : Recovery Configuration

File Location Variables

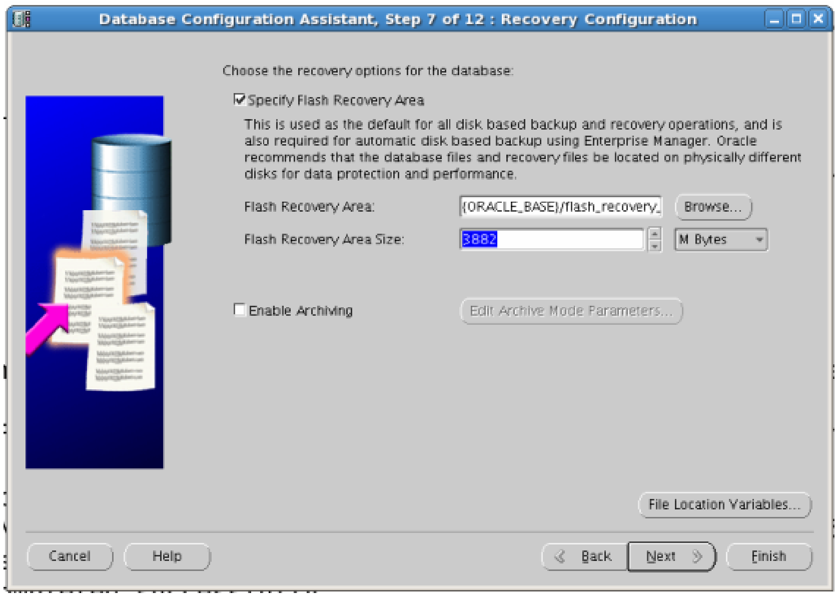
Variables are used to specify parameterized file locations for datafiles, control files, redo logs, and any other files used by database.

For example, a control file may be specified as
{ORACLE_BASE}/oradata/{DB_NAME}/control01.ctl

Variable	Value
ORACLE_BASE	/u01/app/oracle
ORACLE_HOME	/u01/app/oracle/product/11.2.0/dbhome_1
DB_NAME	MAS
DB_UNIQUE_NAME	MAS
SID	MAS

operations, and is
nager. Oracle
n physically different

vi) *Recovery Configuration*



Database Configuration Assistant, Step 7 of 12 : Recovery Configuration

Choose the recovery options for the database:

☑ Specify Flash Recovery Area

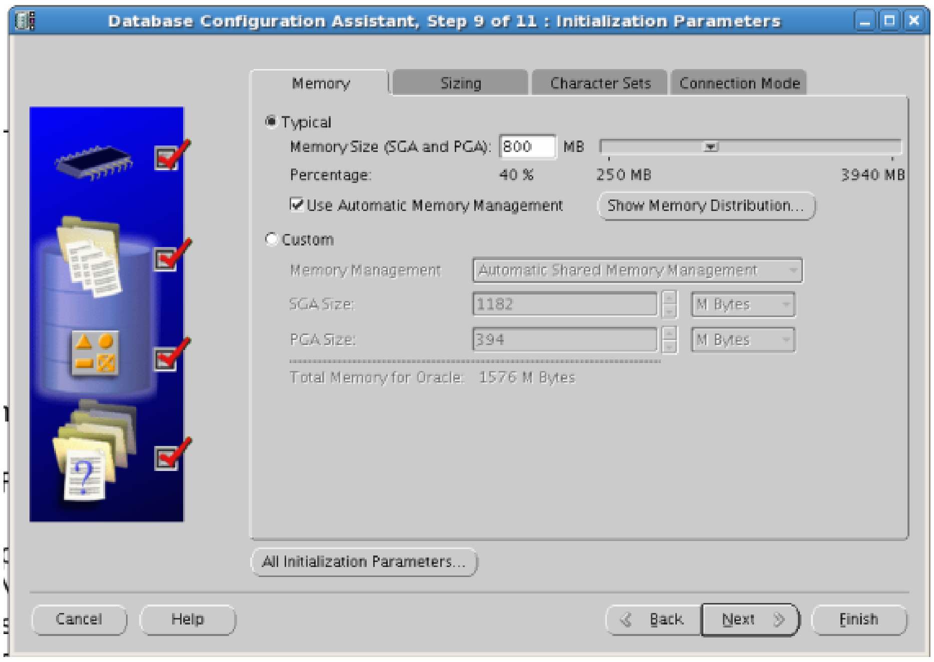
This is used as the default for all disk based backup and recovery operations, and is also required for automatic disk based backup using Enterprise Manager. Oracle recommends that the database files and recovery files be located on physically different disks for data protection and performance.

Flash Recovery Area: {ORACLE_BASE}/flash_recovery_

Flash Recovery Area Size: 3882 M Bytes

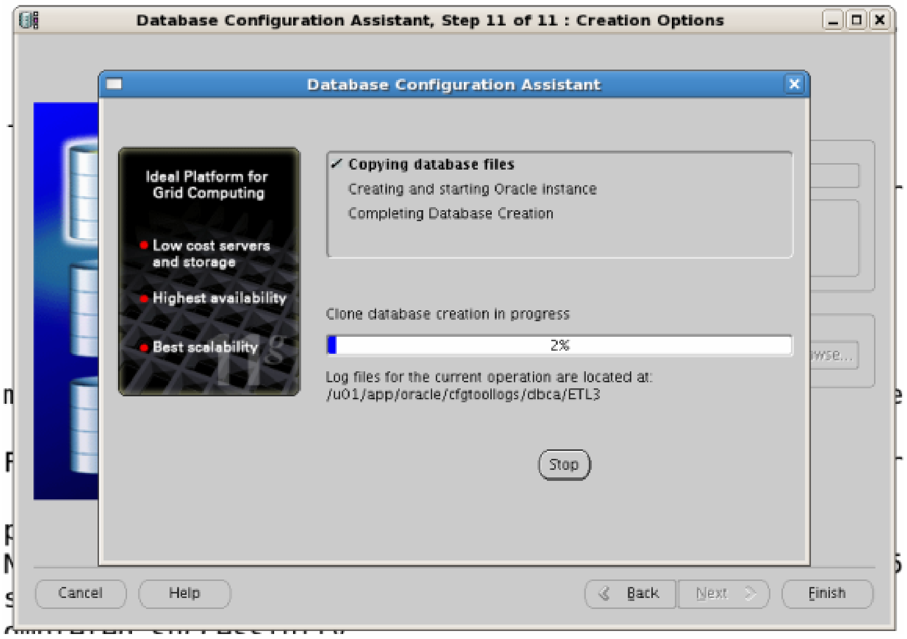
☐ Enable Archiving

vii) *Initialization Parameters – Memory*


Database Configuration Assistant, Step 9 of 11 : Initialization Parameters

● Typical
Memory Size (SGA and PGA): 800 MB
Percentage: 40 % 250 MB 3940 MB
☑ Use Automatic Memory Management
○ Custom
Memory Management: Automatic Shared Memory Management
SGA Size: 1182 M Bytes
PGA Size: 394 M Bytes
Total Memory for Oracle: 1576 M Bytes

viii) Database Creation Complete


Database Configuration Assistant, Step 11 of 11 : Creation Options

Database Configuration Assistant

Ideal Platform for Grid Computing
Low cost servers and storage
Highest availability
Best scalability

✓ Copying database files
Creating and starting Oracle instance
Completing Database Creation

Clone database creation in progress
2%
Log files for the current operation are located at:
/u01/app/oracle/cfgtoollogs/dbca/ETL3

1.a)To create Instance of Oracle

To start the Sql program before that you need to start listener - the listener control utility

i) Type - lsnrctl start

```
[oracle@demo ~]$ lsnrctl start

LSNRCTL for Linux: Version 11.2.0.1.0 - Production on 16-NOV-2023 16:06:37

Copyright (c) 1991, 2009, Oracle.  All rights reserved.

TNS-01106: Listener using listener name LISTENER has already been started
[oracle@demo ~]$ █
```

To connect the Sql program –

ii) Type sqlplus "/as sysdba"

```
[oracle@demo ~]$ sqlplus "/as sysdba"

SQL*Plus: Release 11.2.0.1.0 Production on Sat Nov 4 09:46:15 2023

Copyright (c) 1982, 2009, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit
With the Partitioning, Oracle Label Security, OLAP, Data Mining,
Oracle Database Vault and Real Application Testing options
```

To managing user access and privileges in a SQL database.

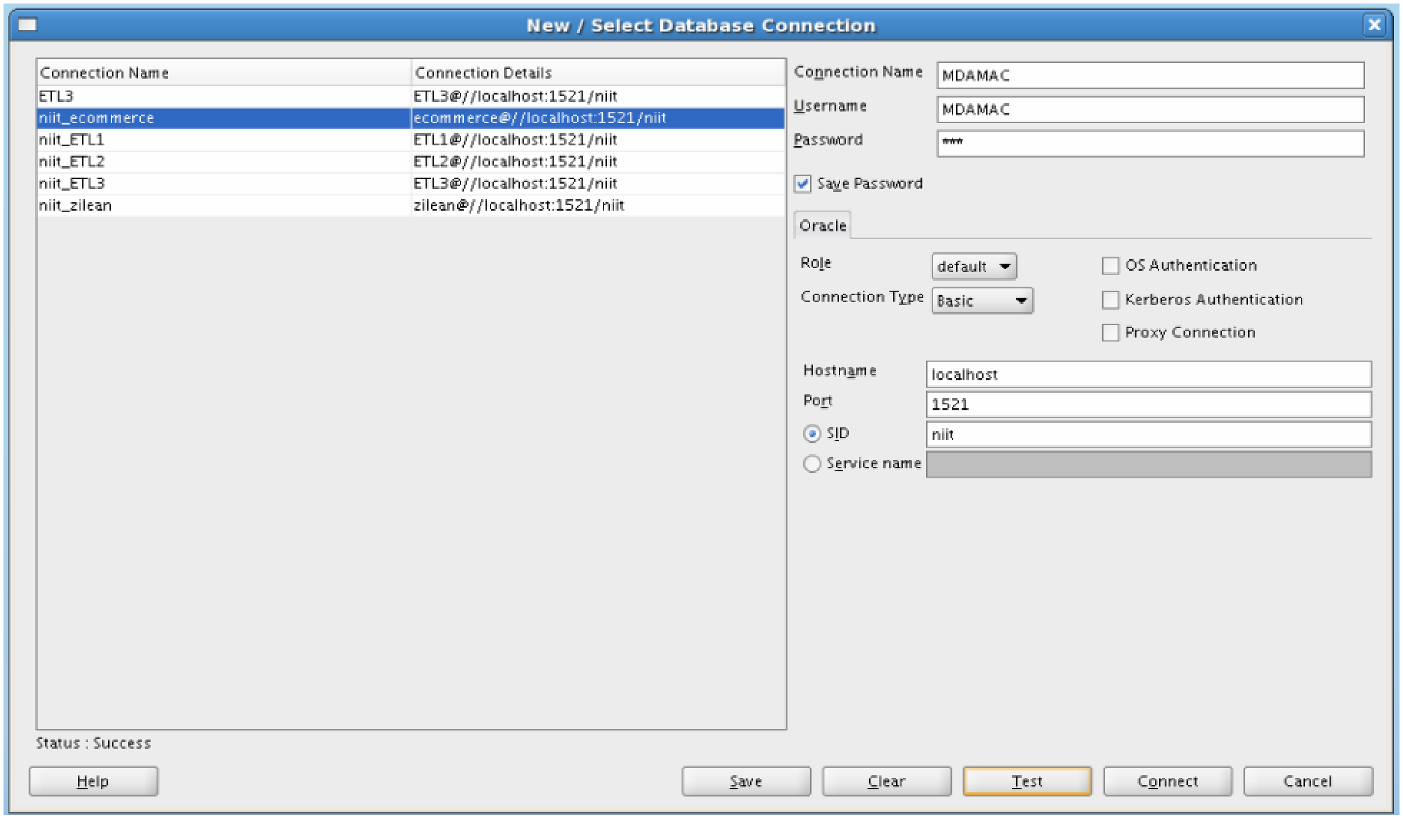
iii) Type create user <username> identified by <password>;

```
SQL> create user MDAMAC identified by MAS;

User created.
```

iv) Type grant connect, resource, dba to <username>;

```
SQL> grant connect, resource, dba to MDAMAC;

Grant succeeded.
```

1.b) Design the Tables Architecture



Connection Name	Connection Details
ETL3	ETL3@//localhost:1521/niit
niit_ecommerce	ecommerce@//localhost:1521/niit
niit_ETL1	ETL1@//localhost:1521/niit
niit_ETL2	ETL2@//localhost:1521/niit
niit_ETL3	ETL3@//localhost:1521/niit
niit_zilean	zilean@//localhost:1521/niit

Connection Name: MDAMAC
Username: MDAMAC
Password: ***
[✓] Save Password
Role: default
Connection Type: Basic
[ ] OS Authentication
[ ] Kerberos Authentication
[ ] Proxy Connection
Hostname: localhost
Port: 1521
[● SID]: niit
○ Service name

Status : Success

Once, the connection estimation successfully

User Side

1) *User Profile table -*

Column	***Data Type***
UserID	*INT*
Username	*VARCHAR (50)*
Email	*VARCHAR (100)*
First Name	*VARCHAR (50)*
Last Name	*VARCHAR (50)*
Age	*INT*
Gender	*VARCHAR (10)*
Location	*VARCHAR (100)*
Subscription Status	*VARCHAR (20)*

2) *User Play History:*

Column	***Data Type***
PlayID	*INT*
UserID	*INT*
SongID	*INT*
Timestamp	*TIMESTAMP*
Duration	*INT*

3) *Playlist:*

Column	***Data Type***
PlaylistID	*INT*
UserID	*INT*
Playlist Name	*VARCHAR (100)*
Description	*TEXT*

4) *User Liked Songs:*

Column	***Data Type***
LikeID	*INT*
UserID	*INT*
SongID	*INT*

5) *User Ratings:*

Column	***Data Type***
RatingID	INT
UserID	INT
SongID	INT
Rating	INT
Timestamp	TIMESTAMP

Admin (Music Company side)

1) *ad tracking*

Column	***Data Type***
Ad_ID	*INT*
SongID	*INT*
Advertiser	*VARCHAR (100)*
Ad Duration	*VARCHAR (20)*
Ad Revenue	*DECIMAL (10, 2)*
Timestamp	*TIMESTAMP*

2) *Artists*

Column	***Data Type***
Artist_ID	INT
Artist Name	VARCHAR (100)
Country	VARCHAR (50)
Biography	TEXT
Social Media Links	VARCHAR (200)

3) *Music Platform*

Column	***Data Type***
User_ID	INT
Username	VARCHAR (50)
Email	VARCHAR (100)
Registration Date	DATE
Subscription Status	VARCHAR (20)

4) *music studio*

Column	***Data Type***
Studio_ID	INT
Studio Name	VARCHAR (100)
Location	VARCHAR (100)
Contact Information	VARCHAR (200)

5) *playlist*

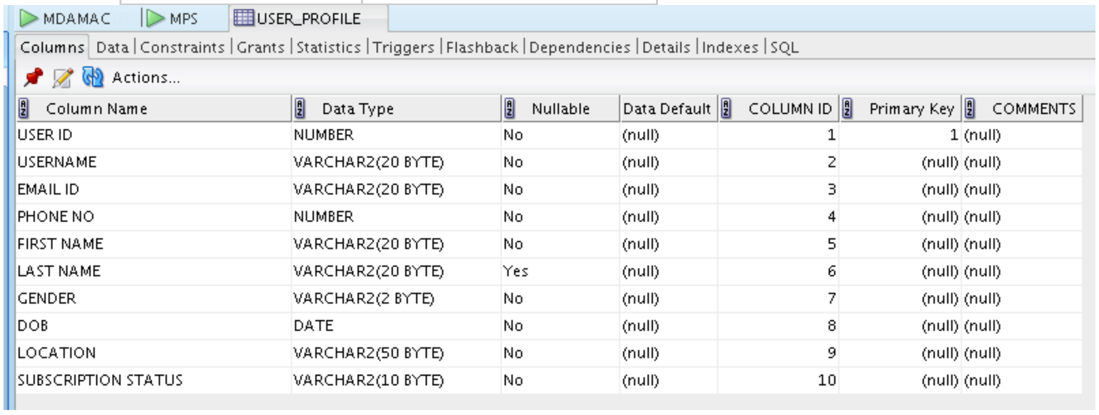
Column	***Data Type***
PlaylistID	*INT*
User_id	*INT*
Playlist Name	*VARCHAR (100)*
Description	*TEXT*

6) *revenue*

Column	***Data Type***
Revenue_ID	INT
SongID	INT
Date	DATE
Revenue Amount	DECIMAL (10, 2)

7) *song library*

Column	Data Type
SongID	*INT*
Title	*VARCHAR (100)*
ArtistID	*INT*
GenreID	*INT*
Release Date	*DATE*
Duration	*INT*
Album	*VARCHAR (100)*
Language	*VARCHAR (50)*
Play Count	*INT*
URL	*VARCHAR (200)*

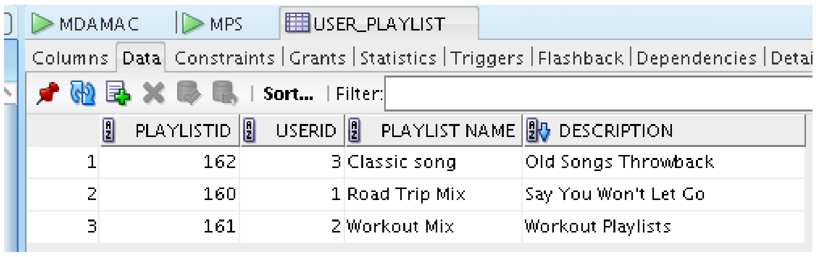

MDAMAC
MPS
USER_PROFILE
Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Indexes | SQL
Actions...

Column Name	Data Type	Nullable	Data Default	COLUMN ID	Primary Key	COMMENTS
USER ID	NUMBER	No	(null)	1	1	(null)
USERNAME	VARCHAR2(20 BYTE)	No	(null)	2	(null)	(null)
EMAIL ID	VARCHAR2(20 BYTE)	No	(null)	3	(null)	(null)
PHONE NO	NUMBER	No	(null)	4	(null)	(null)
FIRST NAME	VARCHAR2(20 BYTE)	No	(null)	5	(null)	(null)
LAST NAME	VARCHAR2(20 BYTE)	Yes	(null)	6	(null)	(null)
GENDER	VARCHAR2(2 BYTE)	No	(null)	7	(null)	(null)
DOB	DATE	No	(null)	8	(null)	(null)
LOCATION	VARCHAR2(50 BYTE)	No	(null)	9	(null)	(null)
SUBSCRIPTION STATUS	VARCHAR2(10 BYTE)	No	(null)	10	(null)	(null)

1.c) Populate data in the Tables, as per the Project Requirement. Only 2 entries per table.

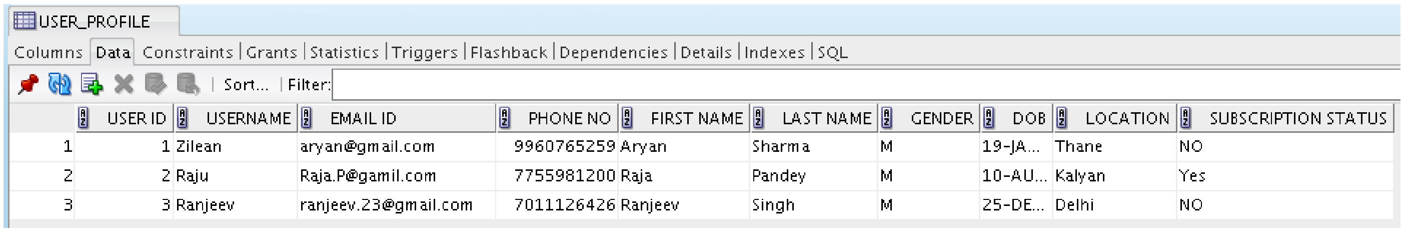
User Side –

i) User Playlist


MDAMAC
MPS
USER_PLAYLIST
Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Detai
Sort... | Filter:

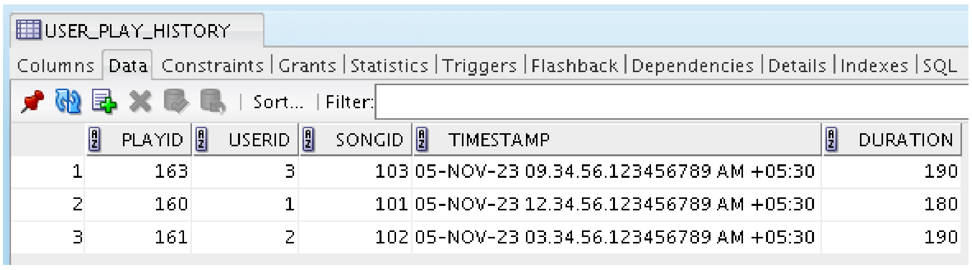
	PLAYLISTID	USERID	PLAYLIST NAME	DESCRIPTION
1	162	3	Classic song	Old Songs Throwback
2	160	1	Road Trip Mix	Say You Won't Let Go
3	161	2	Workout Mix	Workout Playlists

ii) User Profile


USER_PROFILE
Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Indexes | SQL
Sort... | Filter:

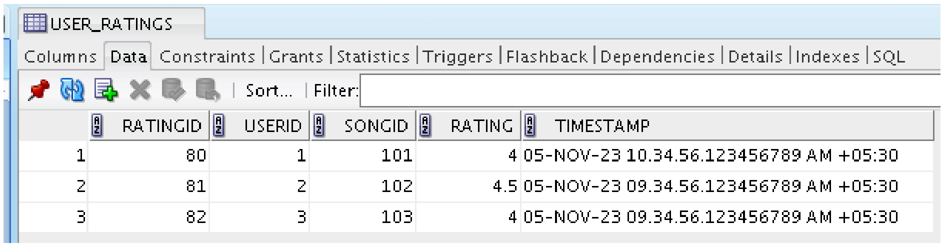
	USER ID	USERNAME	EMAIL ID	PHONE NO	FIRST NAME	LAST NAME	GENDER	DOB	LOCATION	SUBSCRIPTION STATUS
1	1	Zilean	aryan@gmail.com	9960765259	Aryan	Sharma	M	19-JA...	Thane	NO
2	2	Raju	Raja.P@gamil.com	7755981200	Raja	Pandey	M	10-AU...	Kalyan	Yes
3	3	Ranjeev	ranjeev.23@gmail.com	7011126426	Ranjeev	Singh	M	25-DE...	Delhi	NO

iii) User Play History


USER_PLAY_HISTORY
Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Indexes | SQL
Sort... | Filter:

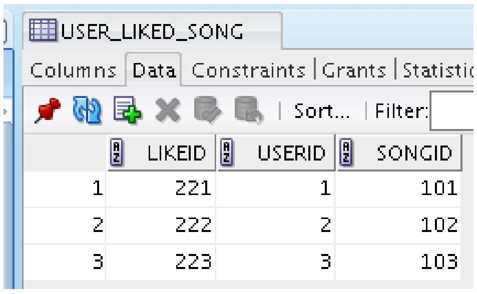
	PLAYID	USERID	SONGID	TIMESTAMP	DURATION
1	163	3	103	05-NOV-23 09.34.56.123456789 AM +05:30	190
2	160	1	101	05-NOV-23 12.34.56.123456789 AM +05:30	180
3	161	2	102	05-NOV-23 03.34.56.123456789 AM +05:30	190

iv) User Rating


USER_RATINGS
Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Indexes | SQL
Sort... | Filter:

	RATINGID	USERID	SONGID	RATING	TIMESTAMP
1	80	1	101	4	05-NOV-23 10.34.56.123456789 AM +05:30
2	81	2	102	4.5	05-NOV-23 09.34.56.123456789 AM +05:30
3	82	3	103	4	05-NOV-23 09.34.56.123456789 AM +05:30

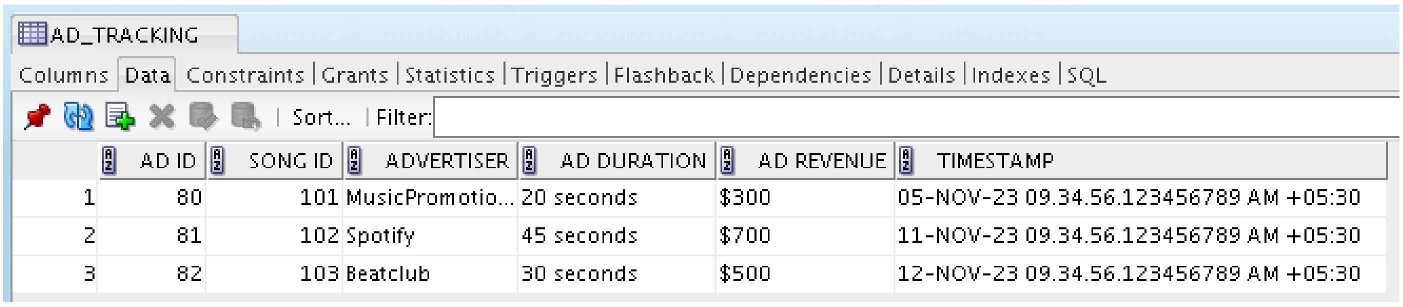
v) Liked song



	LIKEID	USERID	SONGID
1	221	1	101
2	222	2	102
3	223	3	103

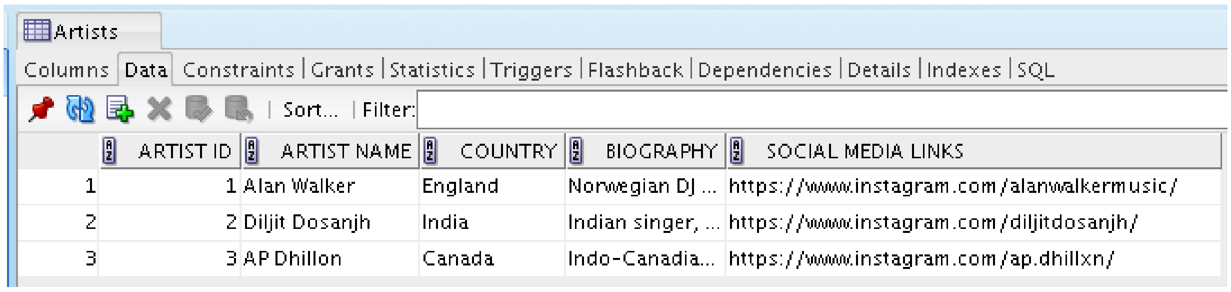
Admin (Music Company side)

i) ad tracking


AD_TRACKING
Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Indexes | SQL
Sort... | Filter:

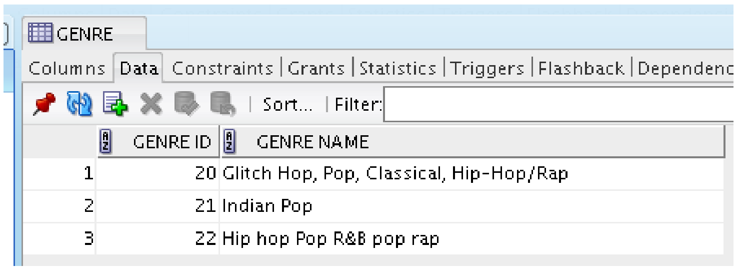
	AD ID	SONG ID	ADVERTISER	AD DURATION	AD REVENUE	TIMESTAMP
1	80	101	MusicPromotio...	20 seconds	$300	05-NOV-23 09.34.56.123456789 AM +05:30
2	81	102	Spotify	45 seconds	$700	11-NOV-23 09.34.56.123456789 AM +05:30
3	82	103	Beatclub	30 seconds	$500	12-NOV-23 09.34.56.123456789 AM +05:30

ii) Artists


Artists
Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Indexes | SQL
Sort... | Filter:

	ARTIST ID	ARTIST NAME	COUNTRY	BIOGRAPHY	SOCIAL MEDIA LINKS
1	1	Alan Walker	England	Norwegian DJ ...	https://www.instagram.com/alanwalkermusic/
2	2	Diljit Dosanjh	India	Indian singer, ...	https://www.instagram.com/diljitdosanjh/
3	3	AP Dhillon	Canada	Indo-Canadia...	https://www.instagram.com/ap.dhillxn/

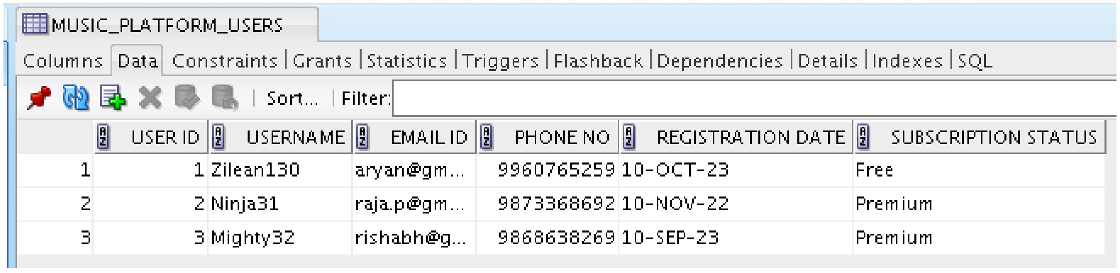
iii) Genre



GENRE

	GENRE ID	GENRE NAME
1	20	Glitch Hop, Pop, Classical, Hip-Hop/Rap
2	21	Indian Pop
3	22	Hip hop Pop R&B pop rap

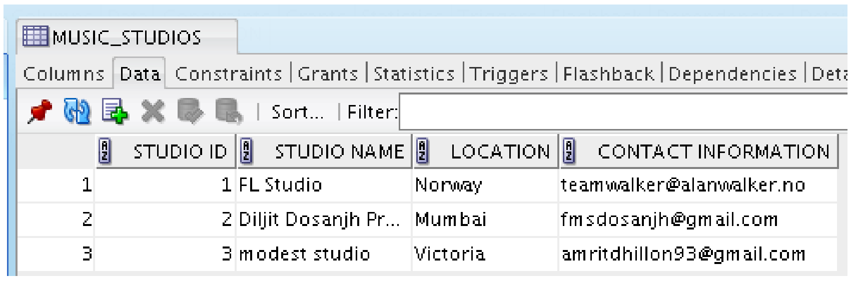
iv) music platform



MUSIC_PLATFORM_USERS

	USER ID	USERNAME	EMAIL ID	PHONE NO	REGISTRATION DATE	SUBSCRIPTION STATUS
1	1	Zilean130	aryan@gm...	9960765259	10-OCT-23	Free
2	2	Ninja31	raja.p@gm...	9873368692	10-NOV-22	Premium
3	3	Mighty32	rishabh@g...	9868638269	10-SEP-23	Premium

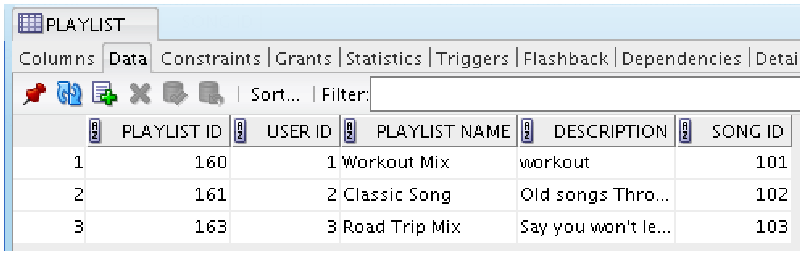
v) music studio



MUSIC_STUDIOS

	STUDIO ID	STUDIO NAME	LOCATION	CONTACT INFORMATION
1	1	FL Studio	Norway	teamwalker@alanwalker.no
2	2	Diljit Dosanjh Pr...	Mumbai	fmsdosanjh@gmail.com
3	3	modest studio	Victoria	amritdhillon93@gmail.com

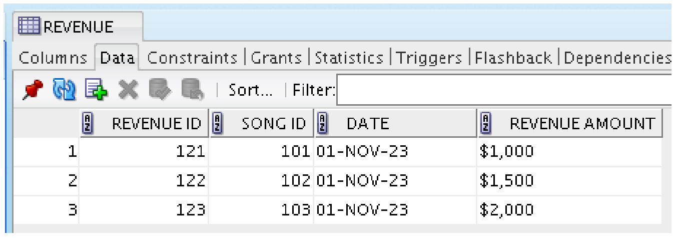
vi) playlist



PLAYLIST

	PLAYLIST ID	USER ID	PLAYLIST NAME	DESCRIPTION	SONG ID
1	160	1	Workout Mix	workout	101
2	161	2	Classic Song	Old songs Thro...	102
3	163	3	Road Trip Mix	Say you won't le...	103

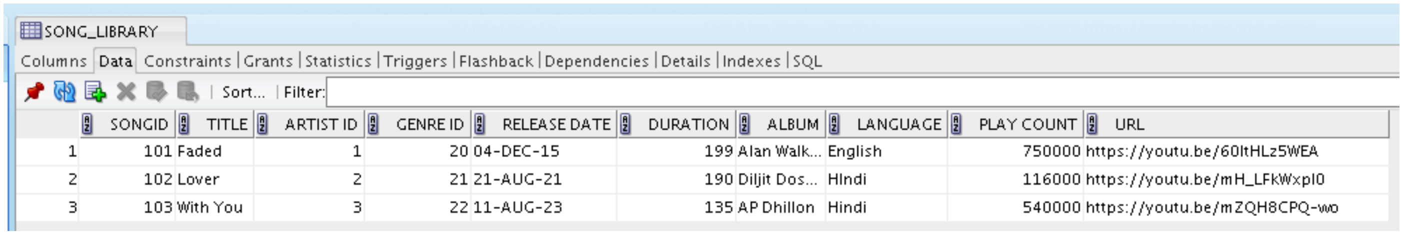
vii) revenue



REVENUE

	REVENUE ID	SONG ID	DATE	REVENUE AMOUNT
1	121	101	01-NOV-23	$1,000
2	122	102	01-NOV-23	$1,500
3	123	103	01-NOV-23	$2,000

viii)Song Library

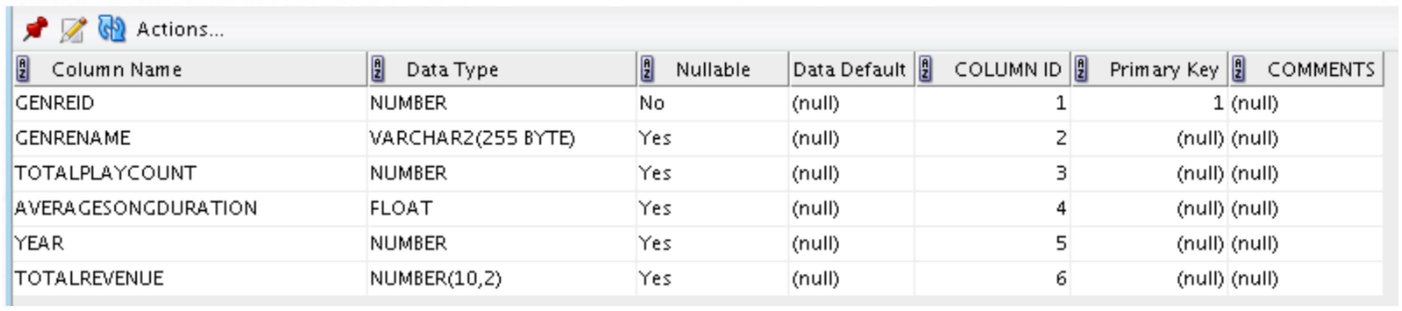


SONG_LIBRARY

	SONGID	TITLE	ARTIST ID	GENRE ID	RELEASE DATE	DURATION	ALBUM	LANGUAGE	PLAY COUNT	URL
1	101	Faded	1	20	04-DEC-15	199	Alan Walk...	English	750000	https://youtu.be/60ItHLz5WEA
2	102	Lover	2	21	21-AUG-21	190	Diljit Dos...	HIndi	116000	https://youtu.be/mH_LFkWxpl0
3	103	With You	3	22	11-AUG-23	135	AP Dhillon	Hindi	540000	https://youtu.be/mZQH8CPQ-wo

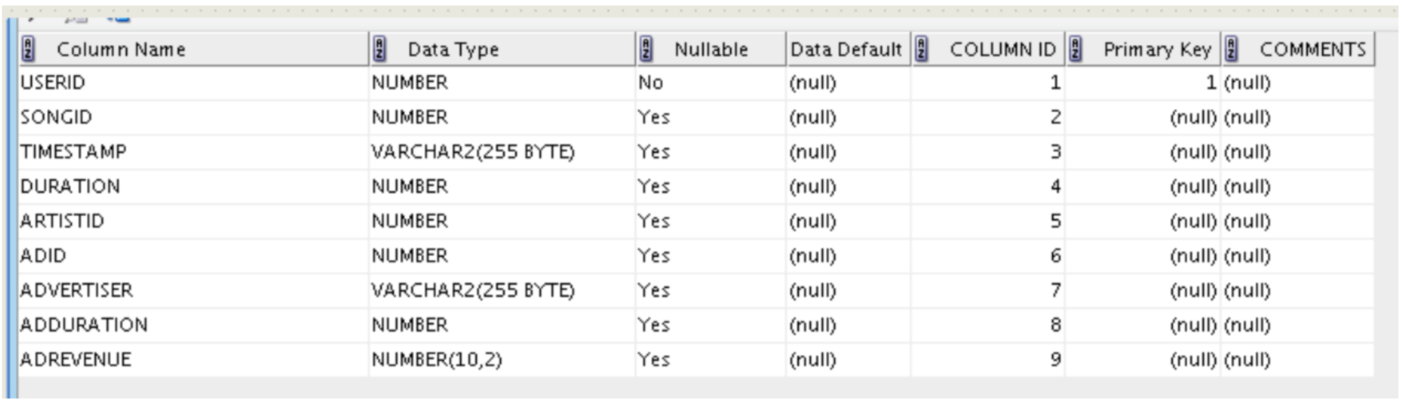
1.d) Roll up and Drill Down

Roll up


Actions...

Column Name	Data Type	Nullable	Data Default	COLUMN ID	Primary Key	COMMENTS
GENREID	NUMBER	No	(null)	1	1	(null)
GENRENAME	VARCHAR2(255 BYTE)	Yes	(null)	2	(null)	(null)
TOTALPLAYCOUNT	NUMBER	Yes	(null)	3	(null)	(null)
AVERAGESONGDURATION	FLOAT	Yes	(null)	4	(null)	(null)
YEAR	NUMBER	Yes	(null)	5	(null)	(null)
TOTALREVENUE	NUMBER(10,2)	Yes	(null)	6	(null)	(null)

Drill down


Column Name	Data Type	Nullable	Data Default	COLUMN ID	Primary Key	COMMENTS
USERID	NUMBER	No	(null)	1	1	(null)
SONGID	NUMBER	Yes	(null)	2	(null)	(null)
TIMESTAMP	VARCHAR2(255 BYTE)	Yes	(null)	3	(null)	(null)
DURATION	NUMBER	Yes	(null)	4	(null)	(null)
ARTISTID	NUMBER	Yes	(null)	5	(null)	(null)
ADID	NUMBER	Yes	(null)	6	(null)	(null)
ADVERTISER	VARCHAR2(255 BYTE)	Yes	(null)	7	(null)	(null)
ADDURATION	NUMBER	Yes	(null)	8	(null)	(null)
ADREVENUE	NUMBER(10,2)	Yes	(null)	9	(null)	(null)

2. To prepare the Schema-Document of below SCD's. (Specify the Columns out of the above Tables for better Candidate for it).
a. Type I changing Dimension
b. Type II changing Dimension
c. Type III changing Dimension

a. *<u>Type1 – SCD Type 1 changing Dimension</u>*

1) *<u>User Profile table -</u>*

Column	***Data Type***
UserID	*INT*
Username	*VARCHAR (50)*
Email	*VARCHAR (100)*
First Name	*VARCHAR (50)*
Last Name	*VARCHAR (50)*
Age	*INT*
Gender	*VARCHAR (10)*
Location	*VARCHAR (100)*
Subscription Status	*VARCHAR (20)*

2) *<u>User Play History:</u>*

Column	***Data Type***
PlayID	*INT*
UserID	*INT*
SongID	*INT*
Timestamp	*TIMESTAMP*
Duration	*INT*

3) *Playlist:*

Column	***Data Type***
LikeID	*INT*
UserID	*INT*
SongID	*INT*

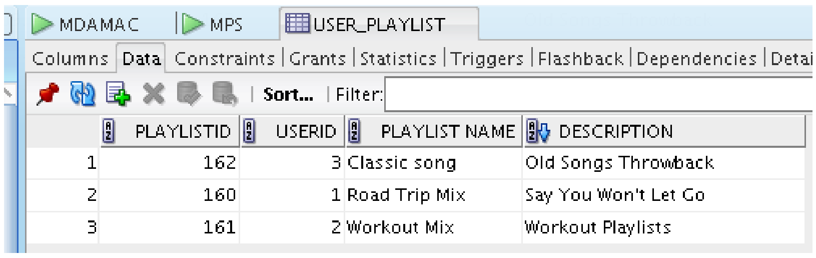
4) *User Liked Songs:*

Column	***Data Type***
PlaylistID	*INT*
UserID	*INT*
Playlist Name	*VARCHAR (100)*
Description	*TEXT*

5) *User Ratings:*

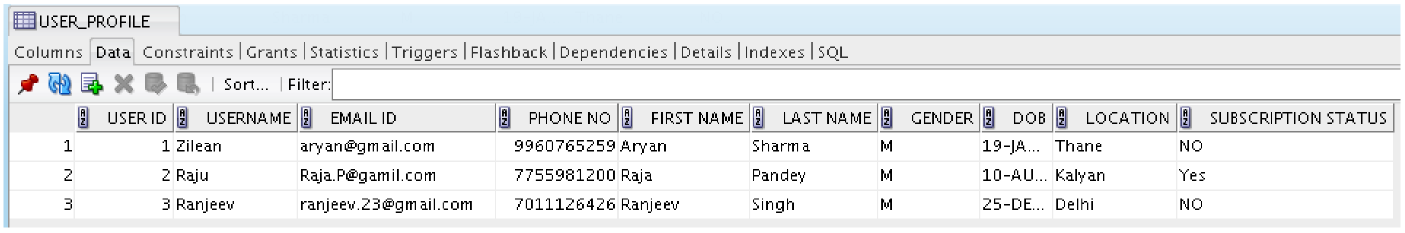
Column	***Data Type***
RatingID	INT
UserID	INT
SongID	INT
Rating	INT
Timestamp	TIMESTAMP

i) User Playlist


MDAMAC
MPS
USER_PLAYLIST
Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Detai
Sort... | Filter:

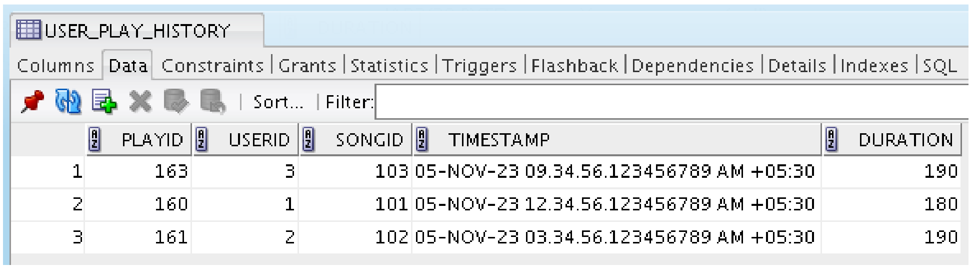
	PLAYLISTID	USERID	PLAYLIST NAME	DESCRIPTION
1	162	3	Classic song	Old Songs Throwback
2	160	1	Road Trip Mix	Say You Won't Let Go
3	161	2	Workout Mix	Workout Playlists

ii) User Profile


USER_PROFILE
Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Indexes | SQL
Sort... | Filter:

	USER ID	USERNAME	EMAIL ID	PHONE NO	FIRST NAME	LAST NAME	GENDER	DOB	LOCATION	SUBSCRIPTION STATUS
1	1	Zilean	aryan@gmail.com	9960765259	Aryan	Sharma	M	19-JA...	Thane	NO
2	2	Raju	Raja.P@gamil.com	7755981200	Raja	Pandey	M	10-AU...	Kalyan	Yes
3	3	Ranjeev	ranjeev.23@gmail.com	7011126426	Ranjeev	Singh	M	25-DE...	Delhi	NO

iii) *User Play History*

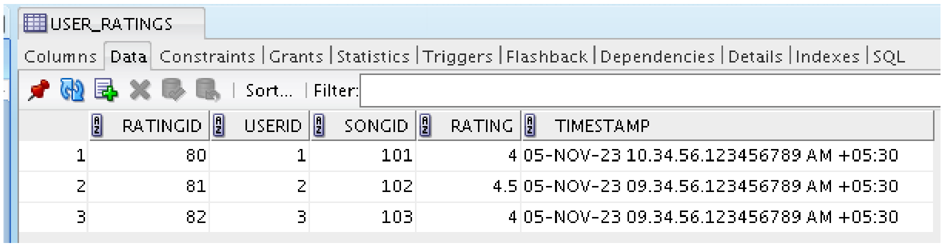


USER_PLAY_HISTORY

Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Indexes | SQL

	PLAYID	USERID	SONGID	TIMESTAMP	DURATION
1	163	3	103	05-NOV-23 09.34.56.123456789 AM +05:30	190
2	160	1	101	05-NOV-23 12.34.56.123456789 AM +05:30	180
3	161	2	102	05-NOV-23 03.34.56.123456789 AM +05:30	190

iv) *User Rating*

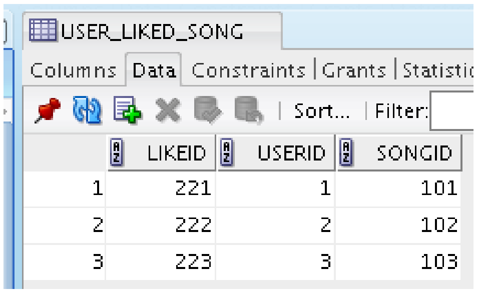


USER_RATINGS

Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Indexes | SQL

	RATINGID	USERID	SONGID	RATING	TIMESTAMP
1	80	1	101	4	05-NOV-23 10.34.56.123456789 AM +05:30
2	81	2	102	4.5	05-NOV-23 09.34.56.123456789 AM +05:30
3	82	3	103	4	05-NOV-23 09.34.56.123456789 AM +05:30

v) *Liked song*



USER_LIKED_SONG

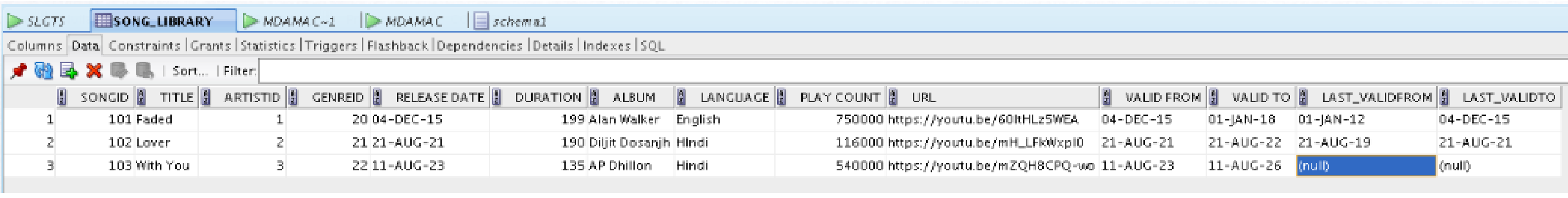
Columns | Data | Constraints | Grants | Statistic

	LIKEID	USERID	SONGID
1	221	1	101
2	222	2	102
3	223	3	103

b. ***Type 2 – SCD Type II changing Dimension***

Song Library Table

Column	**Data Type**
SongID	INT
Title	VARCHAR (100)
ArtistID	INT
GenreID	INT
Release Date	DATE
Duration	INT
Album	VARCHAR (100)
Language	VARCHAR (50)
Play Count	INT
URL	VARCHAR (200)
Valid From	DATE
Valid To	DATE



SLGTS | SONG_LIBRARY | MDAMAC~1 | MDAMAC | schema1

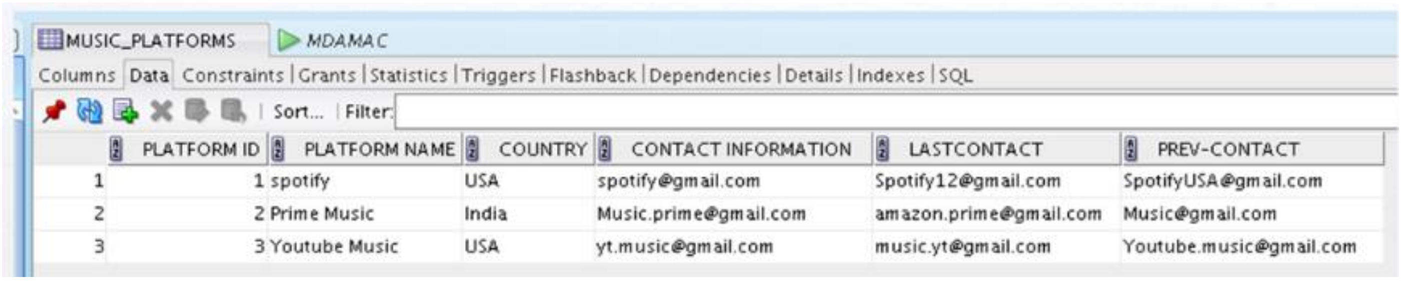
Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Indexes | SQL

	SONGID	TITLE	ARTISTID	GENREID	RELEASE DATE	DURATION	ALBUM	LANGUAGE	PLAY COUNT	URL	VALID FROM	VALID TO	LAST_VALIDFROM	LAST_VALIDTO
1	101	Faded	1	20	04-DEC-15	199	Alan Walker	English	750000	https://youtu.be/60ItHLz5WEA	04-DEC-15	01-JAN-18	01-JAN-12	04-DEC-15
2	102	Lover	2	21	21-AUG-21	190	Diljit Dosanjh	Hindi	116000	https://youtu.be/mH_LFkWxpl0	21-AUG-21	21-AUG-22	21-AUG-19	21-AUG-21
3	103	With You	3	22	11-AUG-23	135	AP Dhillon	Hindi	540000	https://youtu.be/mZQH8CPQ-wo	11-AUG-23	11-AUG-26	(null)	(null)

c. <u>Type 3- SCD Type III Changing Dimension</u>

Music Platforms Table

Column	***Data Type***
Platform ID	*INT*
Platform Name	*VARCHAR (100)*
Country	*VARCHAR (50)*
Contact Information	*VARCHAR (200)*
lastContact	*VARCHAR (200)*
Prev-Contact	*VARCHAR (200)*



	PLATFORM ID	PLATFORM NAME	COUNTRY	CONTACT INFORMATION	LASTCONTACT	PREV-CONTACT
1	1	spotify	USA	spotify@gmail.com	Spotify12@gmail.com	SpotifyUSA@gmail.com
2	2	Prime Music	India	Music.prime@gmail.com	amazon.prime@gmail.com	Music@gmail.com
3	3	Youtube Music	USA	yt.music@gmail.com	music.yt@gmail.com	Youtube.music@gmail.com

3.To prepare document for Vertical and Horizontal Fragmentation. Consider 3 Nodes. – for 2 Tables

A ***<u>Vertical Fragmentation - User Side:</u>***

"User Profile Table"

Column	***Data Type***
UserID	*INT*
Username	*VARCHAR (50)*
Email	*VARCHAR (100)*
First Name	*VARCHAR (50)*
Last Name	*VARCHAR (50)*
Age	*INT*
Gender	*VARCHAR (10)*
Location	*VARCHAR (100)*
Subscription Status	*VARCHAR (20)*

<u>Fragmentation Criteria –</u>
<u>Node 1: user identification</u>
<u>Node 2: personal details</u>
<u>Node 3: geographical location</u>

Node 1: : user identification

Column	***Data Type***	***Description***
UserID	*INT*	*Unique identifier for each user*
Username	*VARCHAR (255)*	*User's login name*
Email	*VARCHAR (255)*	*User's email address*
Subscription Status	*VARCHAR (20)*	*User's subscription status (active, inactive, etc.)*

user identification

UserID	*Username*	*Email*	*Subscription Status*
1	*Zilean*	*aryan@gmail.com*	*NO*
2	*Raj*	*Raj.P@gmail.com*	*Yes*

Node 2 :: personal details

Column	***Data Type***	***Description***
UserID	*INT*	*Unique identifier for each user*
First Name	*VARCHAR (255)*	*User's first name*
Last Name	*VARCHAR (255)*	*User's last name*
Gender	*VARCHAR (10)*	*User's Gender ("M", "F", "NA")*
Age	*INT*	*User's age*

personal details

UserID	*First Name*	*Last Name*	*Gender*	*Age*
1	*Aryan*	*Sharma*	*M*	*19*
2	*Raja*	*Pandey*	*M*	*21*

Node 3 :: geographical location

Column	***Data Type***	***Description***
UserID	INT	Unique identifier for each user
Location	VARCHAR (255)	User's geographical location

geographical location

User ID	***Location***
1	***Thane***
2	***Kalyan***

B. ***Vertical Fragmentation – Admin Side***
" Song Library Table"

Column	***Data Type***
UserID	*INT*
Username	*VARCHAR (50)*
Email	*VARCHAR (100)*
First Name	*VARCHAR (50)*
Last Name	*VARCHAR (50)*
Age	*INT*
Gender	*VARCHAR (10)*
Location	*VARCHAR (100)*
Subscription Status	*VARCHAR (20)*

Fragmentation Criteria –
Node 1: basic song information
Node 2: genre information
Node 3: detailed song information

Node 1:: Basic Song Information

Column	***Data Type***	***Description***
SongID	*INT*	*Primary key for the song.*
Title	*VARCHAR (100)*	*Title of the song.*
ArtistID	*INT*	*Foreign key referencing artists.*

Node 2:: Genre Information

Column	***Data Type***	***Description***
SongID	*INT*	*Primary key for the song.*
GenreID	*INT*	*Foreign key referencing genres.*
Release Date	*DATE*	*Date when the song was released.*

Node 3:: Detailed Song Information

Column	***Data Type***	***Description***
SongID	*INT*	*Primary key for the song.*
Duration	*INT*	*Duration of the song in seconds.*
Album	*VARCHAR (100)*	*Album to which the song belongs.*
Language	*VARCHAR (50)*	*Language of the song.*
Play Count	*INT*	*Number of times the song played.*
URL	*VARCHAR (200)*	*URL or link to the song.*

i. *Horizontal Fragmentation (Admin Side)*

Song Library Table

Column	*Data Type*	*Description*
SongID	*INT*	*Primary key for the song.*
Title	*VARCHAR (100)*	*Title of the song.*
ArtistID	*INT*	*Foreign key referencing artists.*
GenreID	*INT*	*Foreign key referencing genres.*
Release Date	*DATE*	*Date when the song was released.*
Duration	*INT*	*Duration of the song in seconds.*
Album	*VARCHAR (100)*	*Album to which the song belongs.*
Language	*VARCHAR (50)*	*Language of the song.*
Play Count	*INT*	*Number of times the song played.*
URL	*VARCHAR (200)*	*URL or link to the song.*

Fragmentation Criteria –
Node 1: Language - Hindi
Node 2: Language - English
Node 3: Language - Punjabi

Node 1: Language - Punjabi

SongID	*Title*	*ArtistID*	*GenreID*	*Release Date*	*Duration*	*Album*	*Language*	*Play Count*	*URL*
101	*With You*	*1*	*20*	*11-Aug-23*	*195*	*AP Dhillon*	*Punjabi*	*5400000*	*Youtube*
102	*Bachke Bachke*	*2*	*21*	*25-Sept-23*	*199*	*Karan Aujla*	*Punjabi*	*6400000*	*Youtube*

Node 2: Language - Hindi

SongID	*Title*	*ArtistID*	*GenreID*	*Release Date*	*Duration*	*Album*	*Language*	*Play Count*	*URL*
103	*Tum Se Hi*	*3*	*22*	*11-Aug-07*	*195*	*Mohit Chauhan*	*Hindi*	*5400000*	*Youtube*
104	*Kaise Hua*	*4*	*23*	*25-Sept-19*	*199*	*Vishal Mishra*	*Hindi*	*6400000*	*Youtube*

Node 3: Language - English

SongID	*Title*	*ArtistID*	*GenreID*	*Release Date*	*Duration*	*Album*	*Language*	*Play Count*	*URL*

104	FIFTY FIFTY	5	24	11-Aug-22	195	Cupid	English	1400000	Youtube
105	No Lie	6	25	25-Sept-17	199	Sean Paul	English	9400000	Youtube

ii. Horizontal Fragmentation (user Side)
"User Profile Table"

Column	***Data Type***
UserID	*INT*
Username	*VARCHAR (50)*
Email	*VARCHAR (100)*
First Name	*VARCHAR (50)*
Last Name	*VARCHAR (50)*
Age	*INT*
Gender	*VARCHAR (10)*
Location	*VARCHAR (100)*
Subscription Status	*VARCHAR (20)*

Fragmentation Criteria –
Node 1: Location - India
Node 2: Location - China
Node 3: Location – USA

Node 1: Location - India

UserID	Username	Email	First Name	Last Name	Age	Gender	Location	Subscription Status
1	Zilean	aryan@gmail.com	Aryan	Sharma	19	M	India	NO
2	Raj	Raj.P@gmail.com	Raja	Pandey	21	M	India	Yes

Node 2: Location - China

UserID	Username	Email	First Name	Last Name	Age	Gender	Location	Subscription Status
3	Ju	Ju.jing@gmail.com	Ju	Jing	19	F	China	yes
4	Fang	Fang@gmail.com	Fang	Chang	21	F	China	Yes

Node 3: Location - USA

UserID	Username	Email	First Name	Last Name	Age	Gender	Location	Subscription Status
5	Brown	David.@gmail.com	David	Brown	25	M	USA	NO
6	John	John@gmail.com	John	Smith	29	M	USA	Yes

4. Configure REDIS, and Create the Keys and Values, needed for the Project. – like as PATH etc., suggest some Keys and the corresponding Values it can take in it.

Configuration

- *Download the Redis Software by the link: https://github.com/microsoftarchive/redis/releases*

- *Click on the "Redis-x64-3.0.504.msi" file and download it.*

- *After downloading, install the software by accepting the license and adding the path. Select the port you want to run Redis in the next step. Go with the default port i.e. 6379*

- *Select the maximum memory limit for Redis to run and click Install*

- *Go to C drive → Program Files → Redis → redis-cli (right-click on it and select run as administrator)*

ALL SET!

These key-value pairs provide a structured way to organize and access data within Redis for your music data analysis project.

1. *User Profile:*

Key: user:profile:{UserID}
Value: JSON object containing user profile information (e.g., username, email, subscription status).

2. *User Play History:*

Key: user:playhistory:{UserID}
Value: List of JSON objects, each representing a play history entry (e.g., song ID, timestamp, duration).

3. *User Playlists:*

Key: user:playlist:{UserID}:{PlaylistID}
Value: List of song IDs representing the songs in the playlist.

4. *User Liked Songs:*

Key: user:likedsongs:{UserID}
Value: Set of song IDs representing the songs the user has liked.

5. *User Ratings:*

Key: user:ratings:{UserID}
Value: Hash map where keys are song IDs, and values are ratings given by the user.

6. *Song Information:*

Key: song:info:{SongID}
Value: JSON object containing detailed information about the song (e.g., title, artist, genre, release date).

7. *Artist Information:*

Key: artist:info:{ArtistID}
Value: JSON object containing information about the artist (e.g., name, country, biography).

8. Genre Information:

Key: genre:info:{GenreID}
Value: String representing the genre name.

9. Music Studios Information:

Key: studio:info:{StudioID}
Value: JSON object containing information about the music studio (e.g., name, location, contact information).

10. Revenue Information:

Key: revenue:{SongID}:{Date}
Value: Revenue amount for a specific song on a given date.

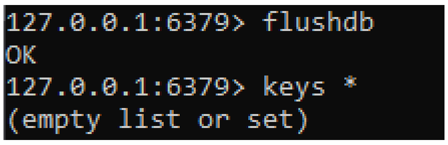
11. Ad Tracking Information:

Key: ad:tracking:{AdID}
Value: JSON object containing information about the ad (e.g., song ID, advertiser, ad duration, ad revenue).

12. Playlist Information:

Key: playlist:info:{PlaylistID}
Value: JSON object containing information about the playlist (e.g., user ID, playlist name, description).

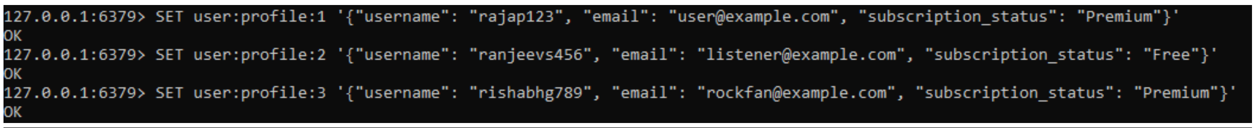
Redis CLI commands for each of the key-value pairs



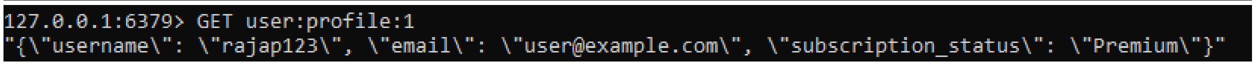
```
127.0.0.1:6379> flushdb
OK
127.0.0.1:6379> keys *
(empty list or set)
```

1. **User Profile:**

(SET -Set key to hold the string value. If the Key already holds a value, it is overwritten)
(GET -Get the value of the key. If the key does not exist the special value nil is returned.)



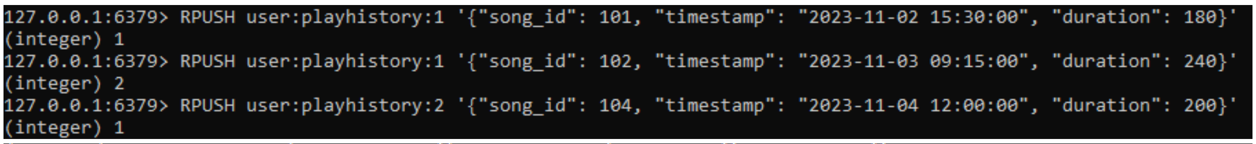
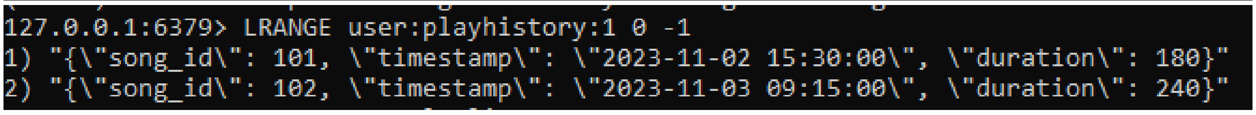
```
127.0.0.1:6379> SET user:profile:1 '{"username": "rajap123", "email": "user@example.com", "subscription_status": "Premium"}'
OK
127.0.0.1:6379> SET user:profile:2 '{"username": "ranjeevs456", "email": "listener@example.com", "subscription_status": "Free"}'
OK
127.0.0.1:6379> SET user:profile:3 '{"username": "rishabhg789", "email": "rockfan@example.com", "subscription_status": "Premium"}'
OK
```



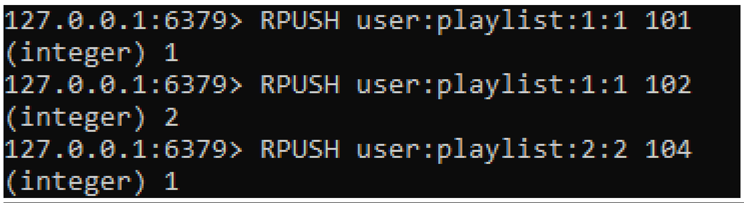
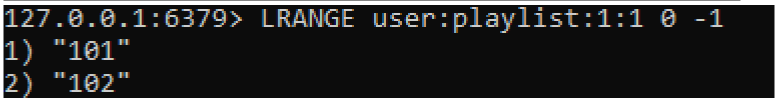
```
127.0.0.1:6379> GET user:profile:1
"{\"username\": \"rajap123\", \"email\": \"user@example.com\", \"subscription_status\": \"Premium\"}"
```

2. **User Play History:**

(RPUSH -Insert all the specified values at the tail of the list stored at Key.)
(LRANGE -Returns the specified elements of the list stored at Key)

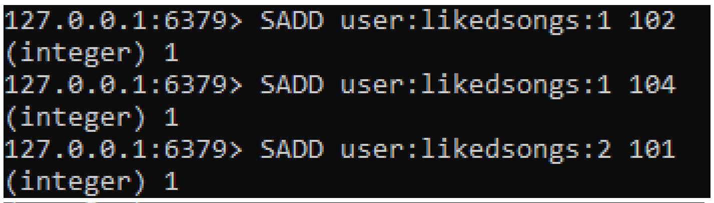
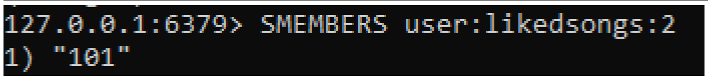

```
127.0.0.1:6379> RPUSH user:playhistory:1 '{"song_id": 101, "timestamp": "2023-11-02 15:30:00", "duration": 180}'
(integer) 1
127.0.0.1:6379> RPUSH user:playhistory:1 '{"song_id": 102, "timestamp": "2023-11-03 09:15:00", "duration": 240}'
(integer) 2
127.0.0.1:6379> RPUSH user:playhistory:2 '{"song_id": 104, "timestamp": "2023-11-04 12:00:00", "duration": 200}'
(integer) 1
```

```
127.0.0.1:6379> LRANGE user:playhistory:1 0 -1
1) "{\"song_id\": 101, \"timestamp\": \"2023-11-02 15:30:00\", \"duration\": 180}"
2) "{\"song_id\": 102, \"timestamp\": \"2023-11-03 09:15:00\", \"duration\": 240}"
```

3.User Playlists:


```
127.0.0.1:6379> RPUSH user:playlist:1:1 101
(integer) 1
127.0.0.1:6379> RPUSH user:playlist:1:1 102
(integer) 2
127.0.0.1:6379> RPUSH user:playlist:2:2 104
(integer) 1
```

```
127.0.0.1:6379> LRANGE user:playlist:1:1 0 -1
1) "101"
2) "102"
```

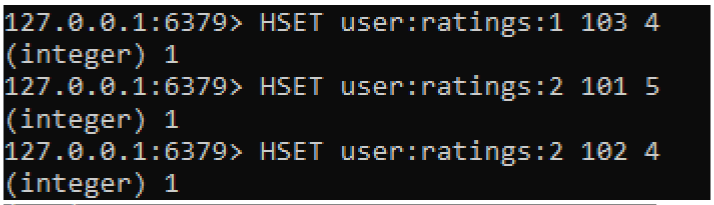
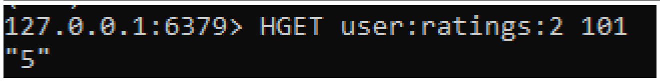
4. User Liked Songs:

(SADD -Add the specified members to the set stored at Key.)
(SMEMBERS -Returns all the members of the set value stored at Key.)

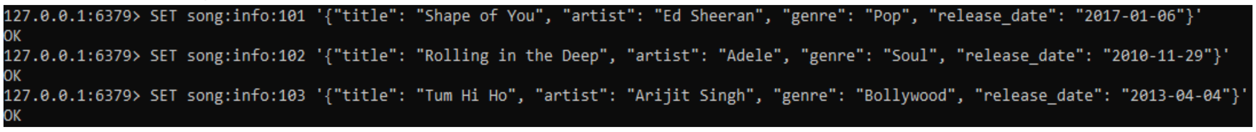
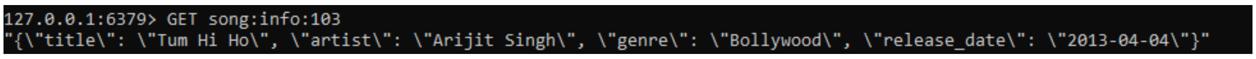

```
127.0.0.1:6379> SADD user:likedsongs:1 102
(integer) 1
127.0.0.1:6379> SADD user:likedsongs:1 104
(integer) 1
127.0.0.1:6379> SADD user:likedsongs:2 101
(integer) 1
```

```
127.0.0.1:6379> SMEMBERS user:likedsongs:2
1) "101"
```

5. User Ratings:

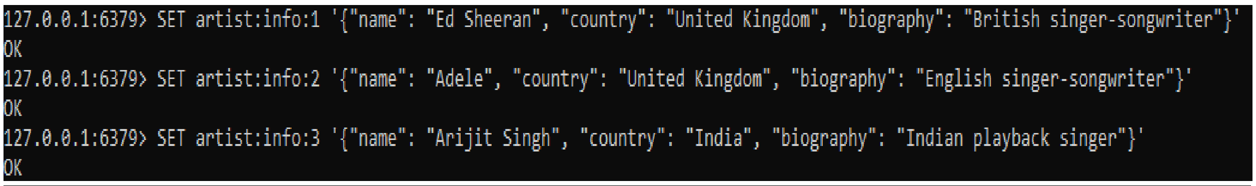
(HSET -Sets the specified fields to their respective values in the hash stored at Key.)
(HGET -Returns the value associated with the field in the hash stored at Key.)


```
127.0.0.1:6379> HSET user:ratings:1 103 4
(integer) 1
127.0.0.1:6379> HSET user:ratings:2 101 5
(integer) 1
127.0.0.1:6379> HSET user:ratings:2 102 4
(integer) 1
```

```
127.0.0.1:6379> HGET user:ratings:2 101
"5"
```

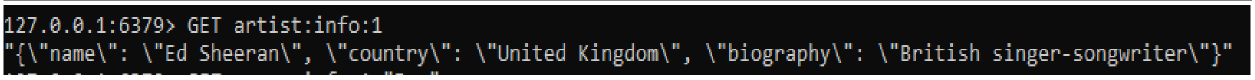
6. Song Information:


```
127.0.0.1:6379> SET song:info:101 '{"title": "Shape of You", "artist": "Ed Sheeran", "genre": "Pop", "release_date": "2017-01-06"}'
OK
127.0.0.1:6379> SET song:info:102 '{"title": "Rolling in the Deep", "artist": "Adele", "genre": "Soul", "release_date": "2010-11-29"}'
OK
127.0.0.1:6379> SET song:info:103 '{"title": "Tum Hi Ho", "artist": "Arijit Singh", "genre": "Bollywood", "release_date": "2013-04-04"}'
OK
```

```
127.0.0.1:6379> GET song:info:103
"{\"title\": \"Tum Hi Ho\", \"artist\": \"Arijit Singh\", \"genre\": \"Bollywood\", \"release_date\": \"2013-04-04\"}"
```

7. Artist Information:

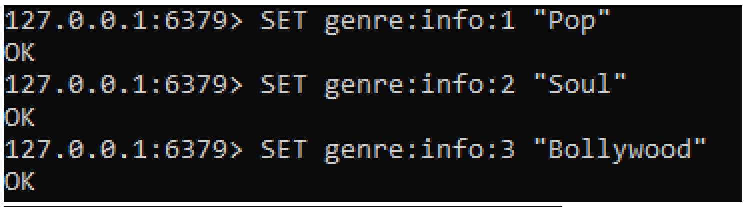


```
127.0.0.1:6379> SET artist:info:1 '{"name": "Ed Sheeran", "country": "United Kingdom", "biography": "British singer-songwriter"}'
OK
127.0.0.1:6379> SET artist:info:2 '{"name": "Adele", "country": "United Kingdom", "biography": "English singer-songwriter"}'
OK
127.0.0.1:6379> SET artist:info:3 '{"name": "Arijit Singh", "country": "India", "biography": "Indian playback singer"}'
OK
```

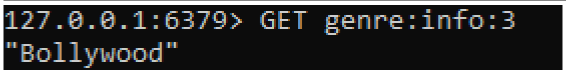


```
127.0.0.1:6379> GET artist:info:1
"{\"name\": \"Ed Sheeran\", \"country\": \"United Kingdom\", \"biography\": \"British singer-songwriter\"}"
```

8. Genre Information:

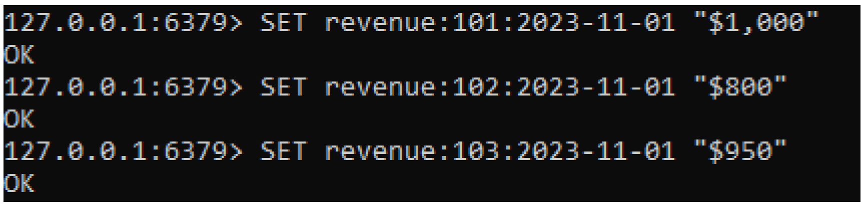


```
127.0.0.1:6379> SET genre:info:1 "Pop"
OK
127.0.0.1:6379> SET genre:info:2 "Soul"
OK
127.0.0.1:6379> SET genre:info:3 "Bollywood"
OK
```



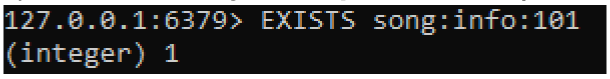
```
127.0.0.1:6379> GET genre:info:3
"Bollywood"
```

9. Revenue Information:



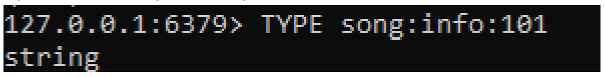
```
127.0.0.1:6379> SET revenue:101:2023-11-01 "$1,000"
OK
127.0.0.1:6379> SET revenue:102:2023-11-01 "$800"
OK
127.0.0.1:6379> SET revenue:103:2023-11-01 "$950"
OK
```

(***Whether the key-value pair exists….)***



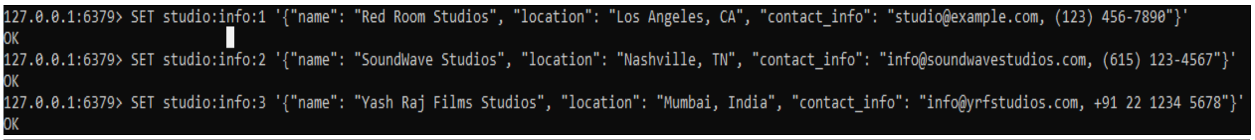
```
127.0.0.1:6379> EXISTS song:info:101
(integer) 1
```

(What its type….)

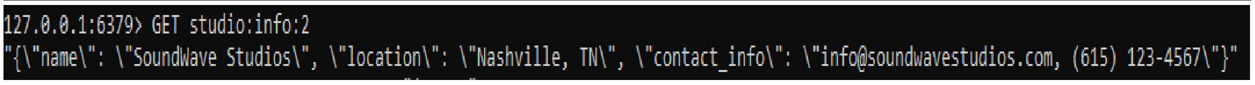


```
127.0.0.1:6379> TYPE song:info:101
string
```

10. Music Studios Information:

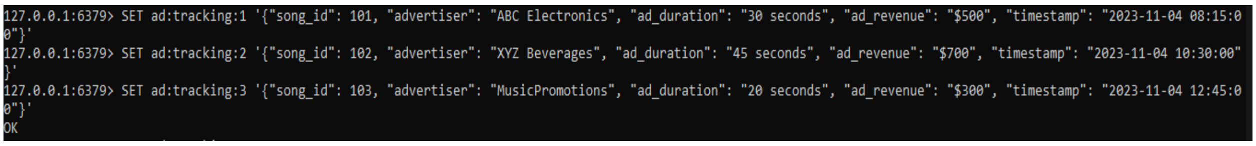


```
127.0.0.1:6379> SET studio:info:1 '{"name": "Red Room Studios", "location": "Los Angeles, CA", "contact_info": "studio@example.com, (123) 456-7890"}'
OK
127.0.0.1:6379> SET studio:info:2 '{"name": "SoundWave Studios", "location": "Nashville, TN", "contact_info": "info@soundwavestudios.com, (615) 123-4567"}'
OK
127.0.0.1:6379> SET studio:info:3 '{"name": "Yash Raj Films Studios", "location": "Mumbai, India", "contact_info": "info@yrfstudios.com, +91 22 1234 5678"}'
OK
```

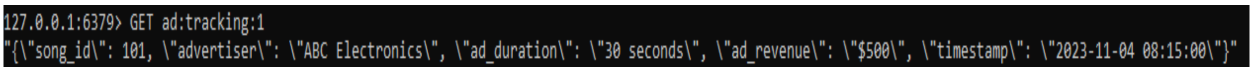


```
127.0.0.1:6379> GET studio:info:2
"{\"name\": \"SoundWave Studios\", \"location\": \"Nashville, TN\", \"contact_info\": \"info@soundwavestudios.com, (615) 123-4567\"}"
```

11. Ad Tracking Information:

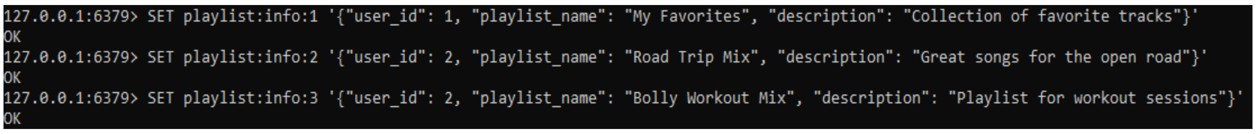


```
127.0.0.1:6379> SET ad:tracking:1 '{"song_id": 101, "advertiser": "ABC Electronics", "ad_duration": "30 seconds", "ad_revenue": "$500", "timestamp": "2023-11-04 08:15:0
0"}'
127.0.0.1:6379> SET ad:tracking:2 '{"song_id": 102, "advertiser": "XYZ Beverages", "ad_duration": "45 seconds", "ad_revenue": "$700", "timestamp": "2023-11-04 10:30:00"
}'
127.0.0.1:6379> SET ad:tracking:3 '{"song_id": 103, "advertiser": "MusicPromotions", "ad_duration": "20 seconds", "ad_revenue": "$300", "timestamp": "2023-11-04 12:45:0
0"}'
OK
```

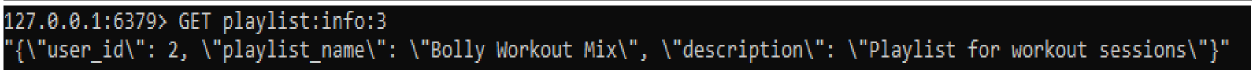


```
127.0.0.1:6379> GET ad:tracking:1
"{\"song_id\": 101, \"advertiser\": \"ABC Electronics\", \"ad_duration\": \"30 seconds\", \"ad_revenue\": \"$500\", \"timestamp\": \"2023-11-04 08:15:00\"}"
```

12. Playlist Information:

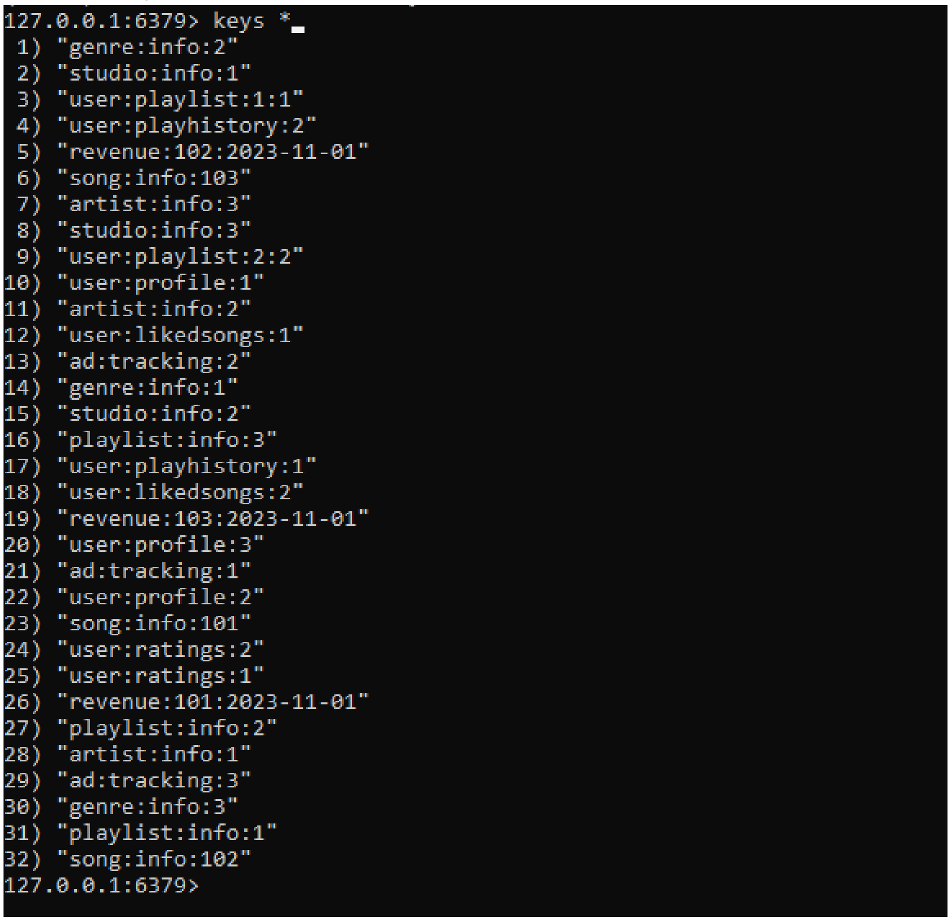


```
127.0.0.1:6379> SET playlist:info:1 '{"user_id": 1, "playlist_name": "My Favorites", "description": "Collection of favorite tracks"}'
OK
127.0.0.1:6379> SET playlist:info:2 '{"user_id": 2, "playlist_name": "Road Trip Mix", "description": "Great songs for the open road"}'
OK
127.0.0.1:6379> SET playlist:info:3 '{"user_id": 2, "playlist_name": "Bolly Workout Mix", "description": "Playlist for workout sessions"}'
OK
```



```
127.0.0.1:6379> GET playlist:info:3
"{\"user_id\": 2, \"playlist_name\": \"Bolly Workout Mix\", \"description\": \"Playlist for workout sessions\"}"
```

ALL the keys at the end



```
127.0.0.1:6379> keys *
 1) "genre:info:2"
 2) "studio:info:1"
 3) "user:playlist:1:1"
 4) "user:playhistory:2"
 5) "revenue:102:2023-11-01"
 6) "song:info:103"
 7) "artist:info:3"
 8) "studio:info:3"
 9) "user:playlist:2:2"
10) "user:profile:1"
11) "artist:info:2"
12) "user:likedsongs:1"
13) "ad:tracking:2"
14) "genre:info:1"
15) "studio:info:2"
16) "playlist:info:3"
17) "user:playhistory:1"
18) "user:likedsongs:2"
19) "revenue:103:2023-11-01"
20) "user:profile:3"
21) "ad:tracking:1"
22) "user:profile:2"
23) "song:info:101"
24) "user:ratings:2"
25) "user:ratings:1"
26) "revenue:101:2023-11-01"
27) "playlist:info:2"
28) "artist:info:1"
29) "ad:tracking:3"
30) "genre:info:3"
31) "playlist:info:1"
32) "song:info:102"
127.0.0.1:6379>
```

Analyzing music data for a music app company using Redis involves leveraging its in-memory capabilities, and data structures.

- *User Play History Analysis:*

Data Structure: Sorted Sets
Analysis Approach: Use a sorted set for each user to store their play history, where the score is the timestamp of the play.
Retrieve and analyze the most-played songs or artists over a specific time period.
Identify trends in user listening behaviour, such as peak listening hours or favourite genres.

- *Playlist Popularity:*

Data Structure: Sorted Sets or Sets
Analysis Approach: Use a sorted set to store playlists and their popularity scores based on the number of users subscribed.
Identify the most popular playlists for different genres or moods.
Analyze changes in playlist popularity over time to tailor recommendations

- *User Liked Songs Analysis:*

Data Structure: Sets
Analysis Approach: Use sets to store the songs that each user has liked.
Identify the most liked songs across the entire user base.
Analyze user preferences by looking at the intersection or union of liked songs between users.

- *Genre Preferences:*

Data Structure: Sets or Hashes
Analysis Approach: Use sets or hashes to store songs categorized by genres.
Analyze which genres are most popular among users.
Track changes in genre preferences over time.

- *Ad Tracking and Revenue Analysis:*

Data Structure: Hashes or Sorted Sets
Analysis Approach: Use hashes to store information about ad tracking, including advertiser, duration, and revenue.
Analyze the performance of different ads based on play counts and revenue.
Identify the most profitable songs or genres in terms of ad revenue.

- *Real-time Recommendations:*

Data Structure: Caching
Analysis Approach: Cache frequently accessed data like popular songs, playlists, or trending artists.
Use Pub/Sub for real-time updates on new releases, trending songs, or user-generated content.
Analyze user interactions in real-time to provide personalized recommendations.

5. To create architecture using FACT and DIMENSIONS as per Star Schema. Consider the Key-Performance-Indicators (KPIs) – like as the Percentage of Profitability in shares, the Time takes to give those returns etc.

Tables For Star Schema

Admin (Music Company side)

1) *<u>ad tracking</u>*

Column	***Data Type***
Ad_ID	*INT*
SongID	*INT*
Advertiser	*VARCHAR (100)*
Ad Duration	*VARCHAR (20)*
Ad Revenue	*DECIMAL (10, 2)*
Timestamp	*TIMESTAMP*

2) *<u>Artists</u>*

Column	***Data Type***
Artist_ID	INT
Artist Name	VARCHAR (100)
Country	VARCHAR (50)
Biography	TEXT
Social Media Links	VARCHAR (200)

3) *music platform*

Column	**Data Type**
User_ID	INT
Username	VARCHAR (50)
Email	VARCHAR (100)
Registration Date	DATE
Subscription Status	VARCHAR (20)

4) *music studio*

Column	**Data Type**
Studio_ID	INT
Studio Name	VARCHAR (100)
Location	VARCHAR (100)
Contact Information	VARCHAR (200)

5) *playlist*

Column	***Data Type***
PlaylistID	*INT*
User_id	*INT*
Playlist Name	*VARCHAR (100)*
Description	*TEXT*

6) *revenue*

Column	**Data Type**
Revenue_ID	INT
SongID	INT
Date	DATE
Revenue Amount	DECIMAL (10, 2)

7) *song library*

Column	***Data Type***
SongID	*INT*
Title	*VARCHAR (100)*
ArtistID	*INT*
GenreID	*INT*
Release Date	*DATE*
Duration	*INT*

Album	*VARCHAR (100)*
Language	*VARCHAR (50)*
Play Count	*INT*
URL	*VARCHAR (200)*

8) Cost Dimension

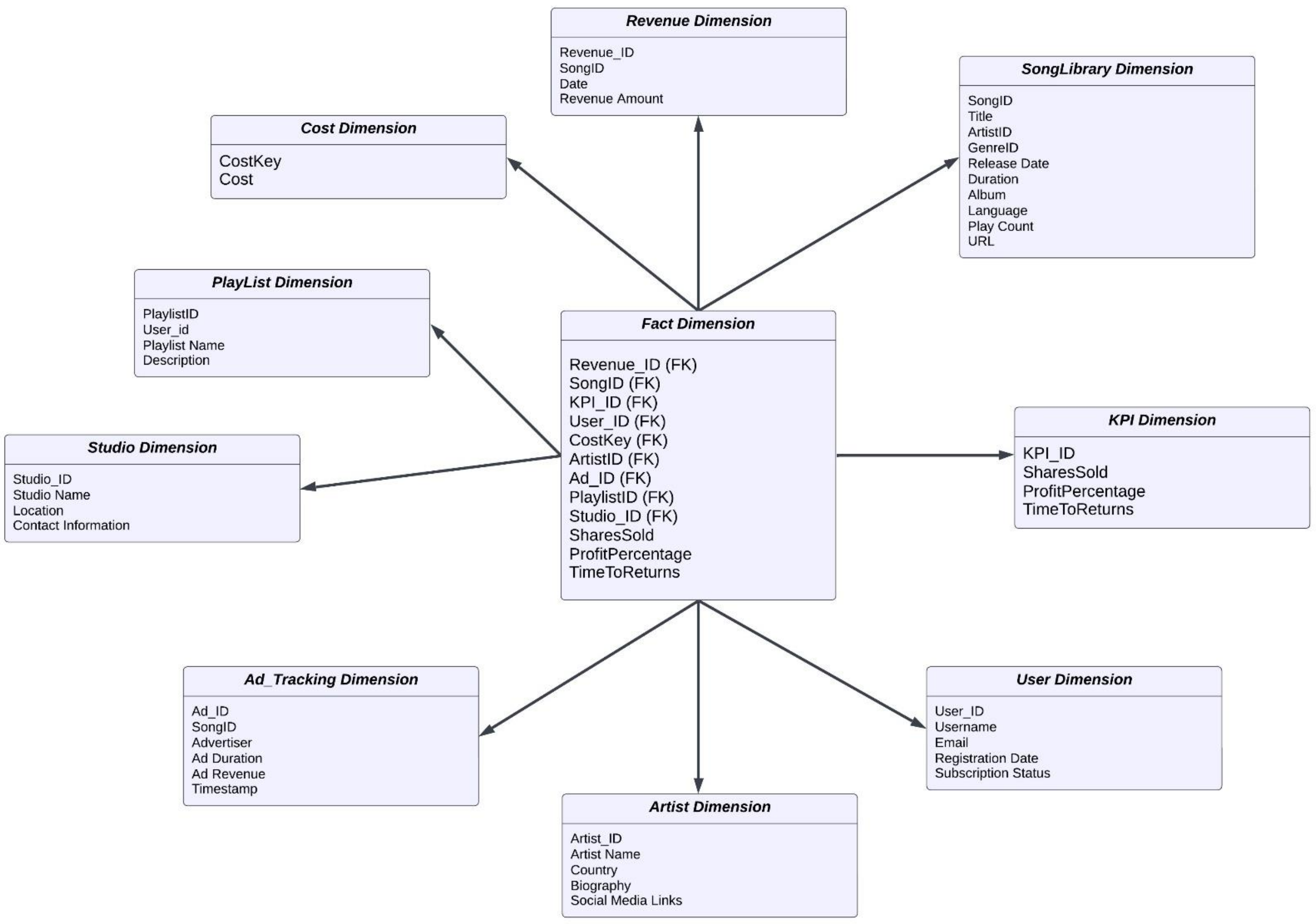
Column	***Data Type***
CostKey	*INT*
Cost	*INT*

9) KPI Dimension

Column	***Data Type***
KPI_ID	*INT*
SharesSold	*INT*
ProfitPercentage	*INT*
TimeToReturns	*Date*

10) Fact Dimension

Column	***Data Type***
Revenue_ID(FK)	*INT*
SongID (FK)	*INT*
KPI_ID (FK)	*INT*
User_ID (FK)	*INT*
CostKey (FK)	*INT*
ArtistID (FK)	*INT*
Ad_ID (FK)	*INT*
PlaylistID (FK)	*INT*
Studio_ID (FK)	*INT*
SharesSold	*INT*
ProfitPercentage	*INT*
TimeToReturns	*INT*



6) To create architecture using FACT and DIMENSIONS as per Snowflake Schema

Tables For Snowflakes Schema

1) *Fact Dimension*

Column	***Data Type***
Revenue_ID(FK)	*INT*
SongID (FK)	*INT*
KPI_ID (FK)	*INT*
User_ID (FK)	*INT*
CostKey (FK)	*INT*
ArtistID (FK)	*INT*
Ad_ID (FK)	*INT*
PlaylistID (FK)	*INT*
Studio_ID (FK)	*INT*
SharesSold	*INT*
ProfitPercentage	*INT*

TimeToReturns	*INT*

1) *ad tracking*

Column	***Data Type***
Ad_ID	*INT*
SongID	*INT*
Advertiser	*VARCHAR (100)*
Timestamp	*TIMESTAMP*

1.1) *Ad_Tracking Sub Dimension*

Column	***Data Type***
Ad_SubID	*INT*
Ad_ID	*INT*
Ad Duration	*INT*
Ad Revenue	*INT*

2) *Artists*

Column	***Data Type***
Artist_ID	INT
Artist Name	VARCHAR (100)
Genre	Varchar(200)
Social Media Links	VARCHAR (200)

2.1) Artist Sub Dimension

Column	***Data Type***
Artist_ID	*INT*
Artist_SubID	*INT*
Country	*Varchar*
Biography	*Varchar*
Instagram_ID	*Varchar*
Youtube_ID	*Varchar*
Twitter_ID	*Varchar*
Thread_ID	*Varchar*

3) *User*

Column	***Data Type***
User_ID	INT
Username	VARCHAR (50)
Email	VARCHAR (100)
Registration Date	DATE
Subscription Status	VARCHAR (20)

3.1) User Sub Dimension

Column	***Data Type***
User_ID	*INT*
User_SubID	*INT*
First_Name	*Varchar*
Last_Name	*Varchar*
Phone_no	*INT*
Subscription_Type	*Varchar*

4) <u>music studio</u>

Column	***Data Type***
Studio_ID	INT
Studio Name	VARCHAR (100)
Location	VARCHAR (100)
Contact Information	VARCHAR (200)

4.1) Studio Sub Dimension

Column	***Data Type***
Studio_ID	*INT*
Studio_SubID	*INT*
Studio_Branch	*Varchar*
Country	*Varchar*
State	*Varchar*

5) <u>playlist</u>

Column	***Data Type***
PlaylistID	*INT*
User_id	*INT*
Playlist Name	*VARCHAR (100)*
Description	*TEXT*

6) revenue

Column	**Data Type**
Revenue_ID	INT
SongID	INT
Date	DATE
Revenue Amount	DECIMAL (10, 2)

6.1) Revenue Dimension

Column	***Data Type***
Revenue_ID	*INT*
Revenue_SubID	*INT*
Ads	*INT*
SharesSold	*INT*
Royalties	*INT*

7) *song library*

Column	***Data Type***
SongID	*INT*
Title	*VARCHAR (100)*
ArtistID	*INT*
GenreID	*INT*
Release Date	*DATE*
Album	*VARCHAR (100)*
Play Count	*INT*

7.1) SongLibrary Sub Dimension

Column	***Data Type***
SongID	*INT*
Song_SubID	*INT*
Genre_Type	*Varchar*
Album	*Varchar*
Language	*Varchar*
Play_Time	*INT*
URL	*Varchar*

8) Cost Dimension

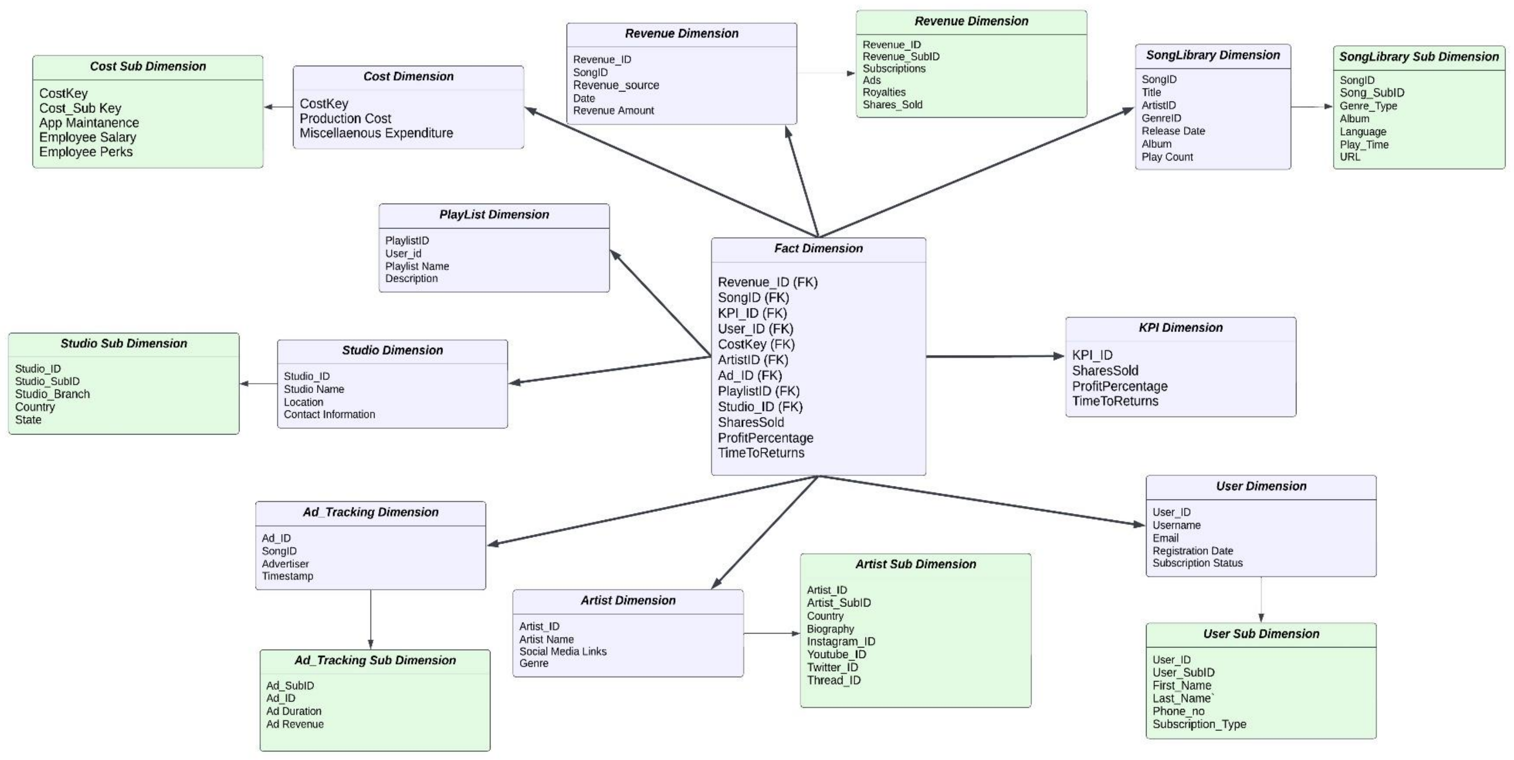
Column	***Data Type***
CostKey	*INT*
Cost	*INT*

8.1) Cost Sub Dimension

Column	***Data Type***
CostKey	*INT*
Cost_Sub Key	*INT*
App Maintanence	*INT*
Employee Salary	*INT*
Employee Perks	*INT*

9) KPI Dimension

Column	***Data Type***
KPI_ID	*INT*
SharesSold	*INT*
ProfitPercentage	*INT*

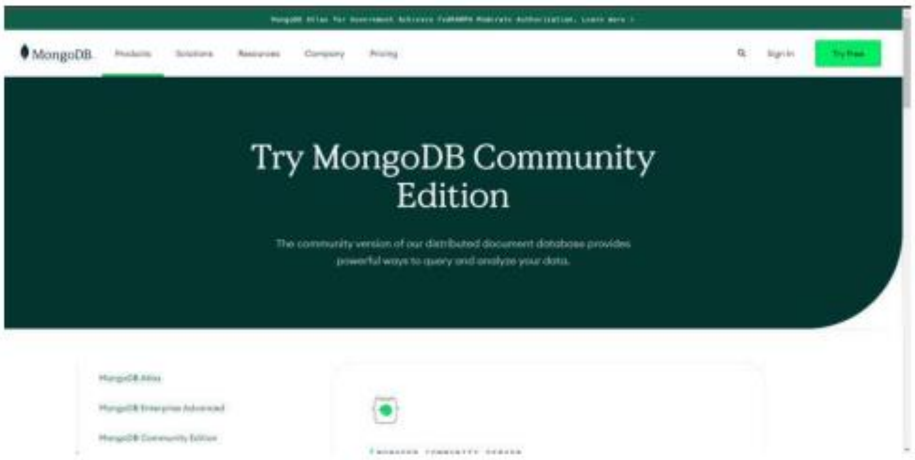


7) Configure the Collections in MongoDB, for Your Project Requirements – Here Create tables in such a way, that there are no Joins needed, to pull out the same Data. Write information about Variables setup, like PATH Variable etc.

Setting Up MongoDB And Variables

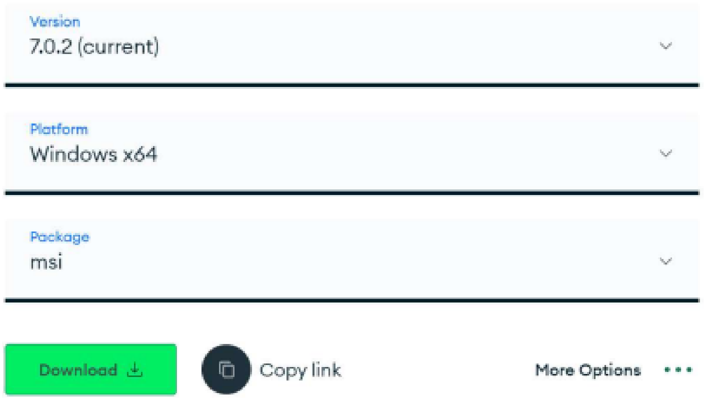
MongoDB Installation

Step1: Click on the link:
https://www.mongodb.com/try/download/community


Try MongoDB Community Edition

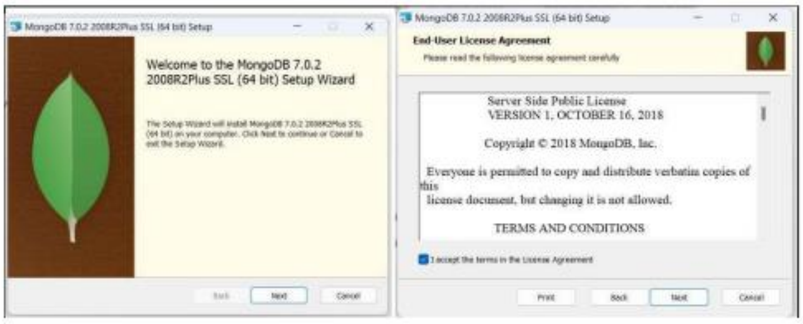
The community version of our distributed document database provides powerful ways to query and analyze your data.

Step2: Download the latest 7.0.2 Version for Windows x64 and select the msi package


Version: 7.0.2 (current)
Platform: Windows x64
Package: msi

Step3: Extract the file and launch the installer

- Click Next to start installation.
- Accept the licence agreement then click Next


MongoDB 7.0.2 2008R2Plus SSL (64 bit) Setup

Welcome to the MongoDB 7.0.2 2008R2Plus SSL (64 bit) Setup Wizard

The Setup Wizard will install MongoDB 7.0.2 2008R2Plus SSL (64 bit) on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

MongoDB 7.0.2 2008R2Plus SSL (64 bit) Setup

End-User License Agreement

Please read the following license agreement carefully

Server Side Public License
VERSION 1, OCTOBER 16, 2018

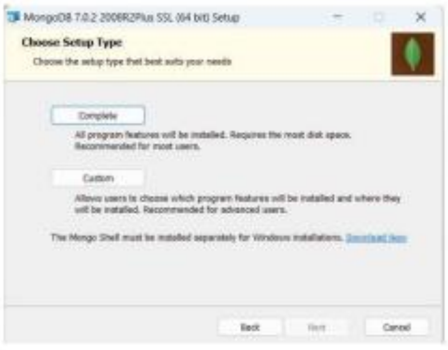
Copyright © 2018 MongoDB, Inc.

Everyone is permitted to copy and distribute verbatim copies of this
license document, but changing it is not allowed.

TERMS AND CONDITIONS

[✓] I accept the terms in the License Agreement

- Click on complete set up
- Click on run service as a network user



MongoDB 7.0.2 2008R2Plus SSL (64 bit) Setup

Choose Setup Type

Choose the setup type that best suits your needs

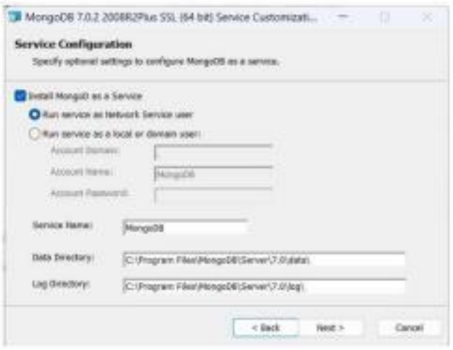
Complete

All program features will be installed. Requires the most disk space. Recommended for most users.

Custom

Allows users to choose which program features will be installed and where they will be installed. Recommended for advanced users.

The Mongo Shell must be installed separately for Windows installations. Download here



MongoDB 7.0.2 2008R2Plus SSL (64 bit) Service Customizati...

Service Configuration

Specify optional settings to configure MongoDB as a service.

[✓] Install MongoD as a Service

[✓ Run service as Network Service user]

Run service as a local or domain user:

Account Domain:

Account Name: MongoDB

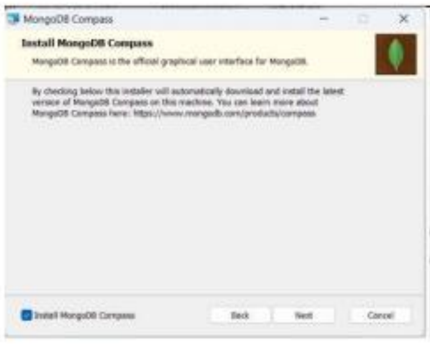
Account Password:

Service Name: MongoDB

Data Directory: C:\Program Files\MongoDB\Server\7.0\data\

Log Directory: C:\Program Files\MongoDB\Server\7.0\log\

- Install MongoDB Compass



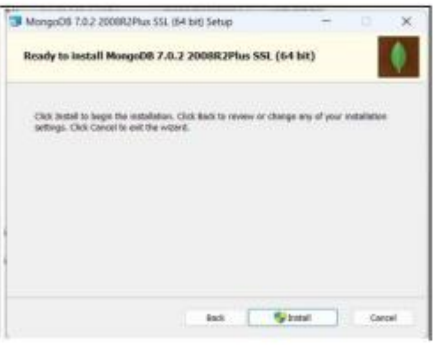
MongoDB Compass

Install MongoDB Compass

MongoDB Compass is the official graphical user interface for MongoDB.

By checking below this installer will automatically download and install the latest version of MongoDB Compass on this machine. You can learn more about MongoDB Compass here: https://www.mongodb.com/products/compass

[✓] Install MongoDB Compass

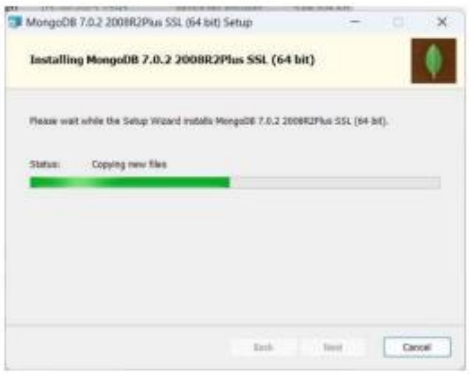


MongoDB 7.0.2 2008R2Plus SSL (64 bit) Setup

Ready to install MongoDB 7.0.2 2008R2Plus SSL (64 bit)

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.

- Start Installation



MongoDB 7.0.2 2008R2Plus SSL (64 bit) Setup

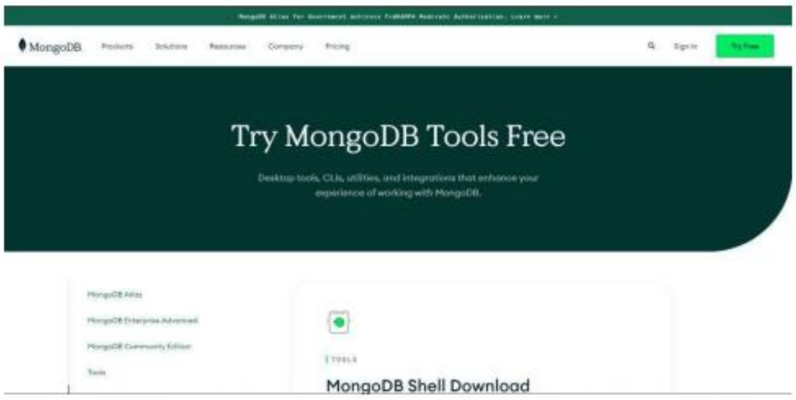
Installing MongoDB 7.0.2 2008R2Plus SSL (64 bit)

Please wait while the Setup Wizard installs MongoDB 7.0.2 2008R2Plus SSL (64 bit).

Status: Copying new files

<u>Mongo Shell Installation</u>

Step1: Click on the link:
https://www.mongodb.com/try/download/shell

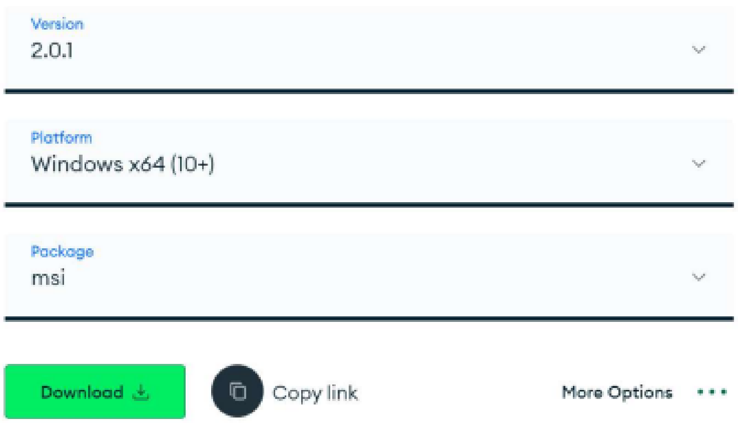


Try MongoDB Tools Free

Desktop tools, CLIs, utilities, and integrations that enhance your experience of working with MongoDB.

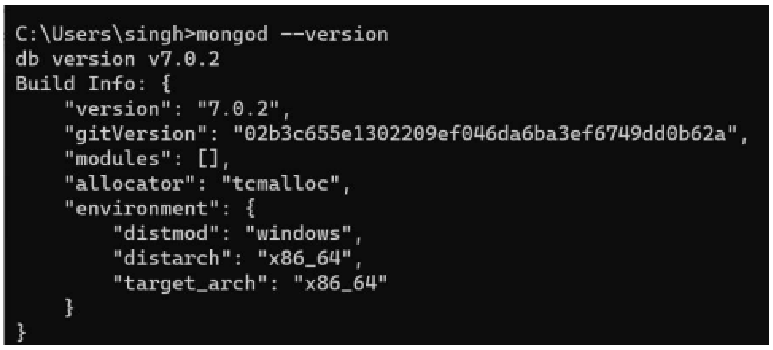
MongoDB Shell Download

Step2: Download the latest 2.0.1 for Windows x64 and select the msi package



Verifying the MongoDB Version

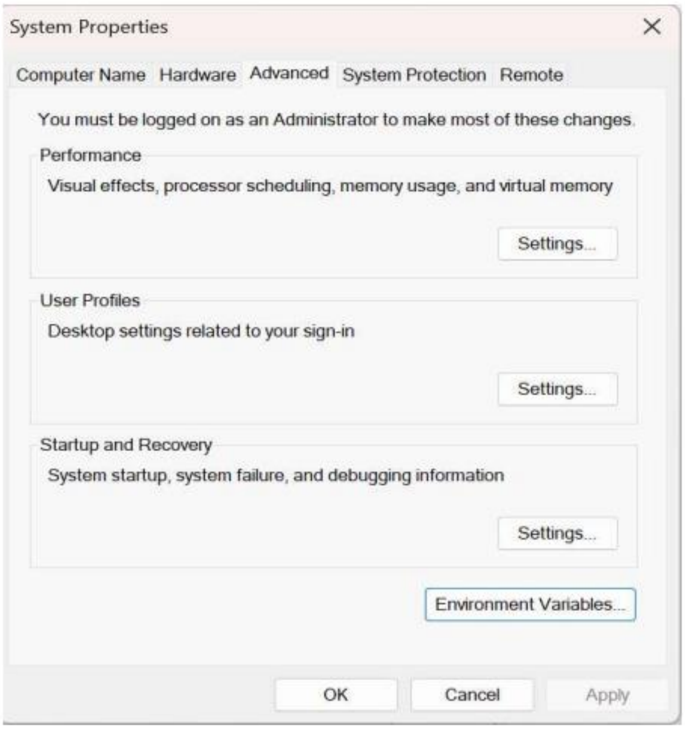
Using the command ***mongod --version***



```
C:\Users\singh>mongod --version
db version v7.0.2
Build Info: {
    "version": "7.0.2",
    "gitVersion": "02b3c655e1302209ef046da6ba3ef6749dd0b62a",
    "modules": [],
    "allocator": "tcmalloc",
    "environment": {
        "distmod": "windows",
        "distarch": "x86_64",
        "target_arch": "x86_64"
    }
}
```

Configurations

Step1: Go to environment variables in system properties



System Properties

You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

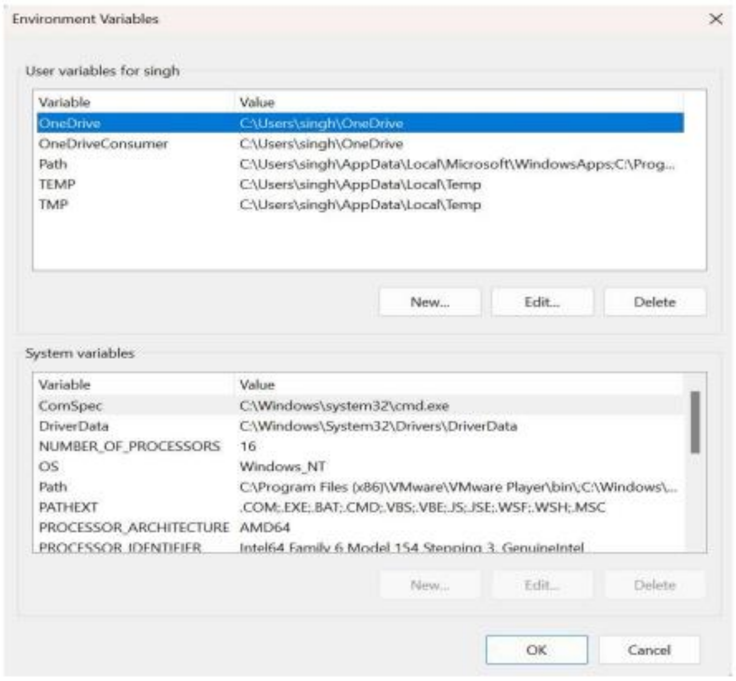
User Profiles

Desktop settings related to your sign-in

Startup and Recovery

System startup, system failure, and debugging information

Step2: Click on path in system variables



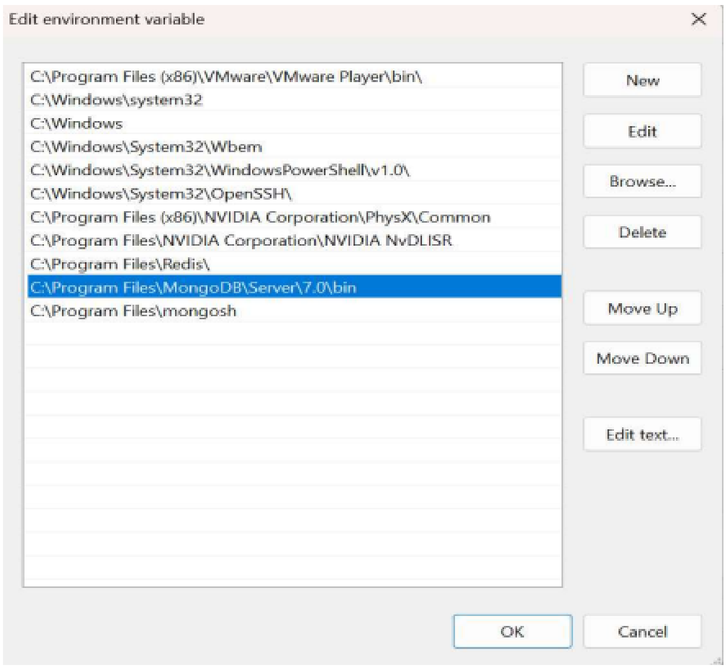
User variables for singh

Variable	Value
OneDrive	C:\Users\singh\OneDrive
OneDriveConsumer	C:\Users\singh\OneDrive
Path	C:\Users\singh\AppData\Local\Microsoft\WindowsApps;C:\Prog...
TEMP	C:\Users\singh\AppData\Local\Temp
TMP	C:\Users\singh\AppData\Local\Temp

System variables

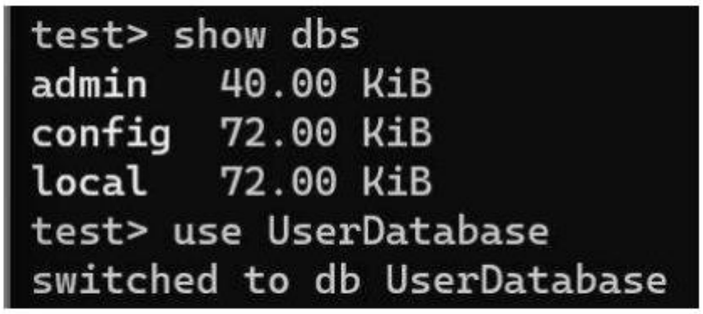
Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	16
OS	Windows_NT
Path	C:\Program Files (x86)\VMware\VMware Player\bin\;C:\Windows\...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 154 Stepping 3, GenuineIntel

Step3: Add new path where the database file is located and click on OK



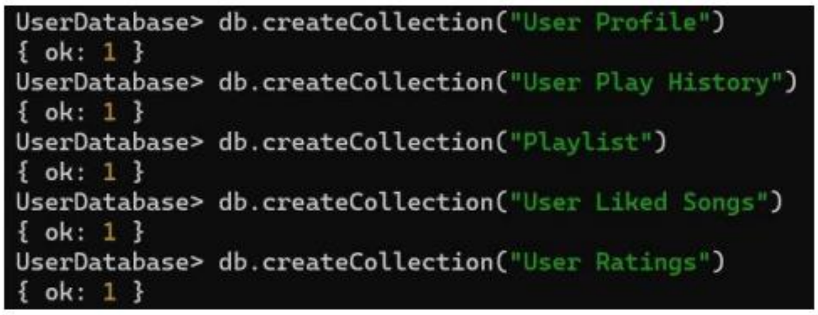
```
C:\Program Files (x86)\VMware\VMware Player\bin\
C:\Windows\system32
C:\Windows
C:\Windows\System32\Wbem
C:\Windows\System32\WindowsPowerShell\v1.0\
C:\Windows\System32\OpenSSH\
C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common
C:\Program Files\NVIDIA Corporation\NVIDIA NvDLISR
C:\Program Files\Redis\
C:\Program Files\MongoDB\Server\7.0\bin
C:\Program Files\mongosh
```

Creating A New UserDatabase



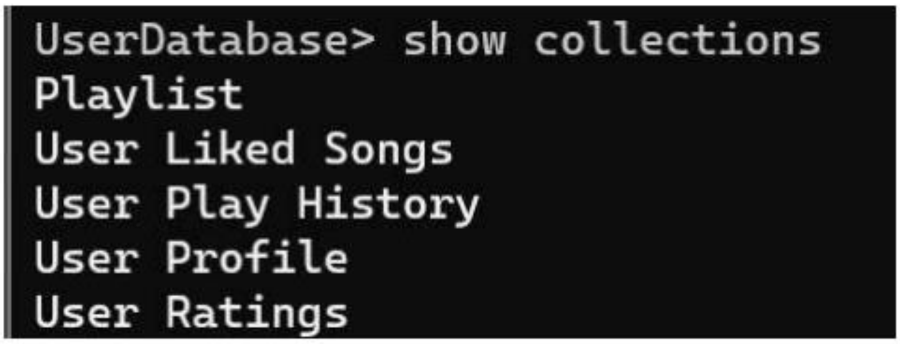
```
test> show dbs
admin    40.00 KiB
config   72.00 KiB
local    72.00 KiB
test> use UserDatabase
switched to db UserDatabase
```

Creating Collections



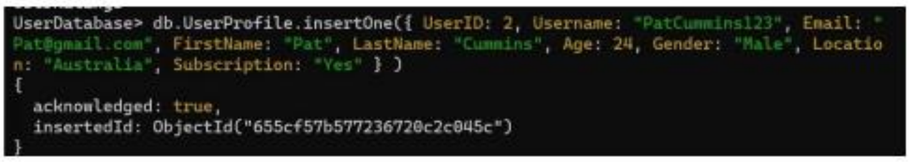
```
UserDatabase> db.createCollection("User Profile")
{ ok: 1 }
UserDatabase> db.createCollection("User Play History")
{ ok: 1 }
UserDatabase> db.createCollection("Playlist")
{ ok: 1 }
UserDatabase> db.createCollection("User Liked Songs")
{ ok: 1 }
UserDatabase> db.createCollection("User Ratings")
{ ok: 1 }
```

Show Collections

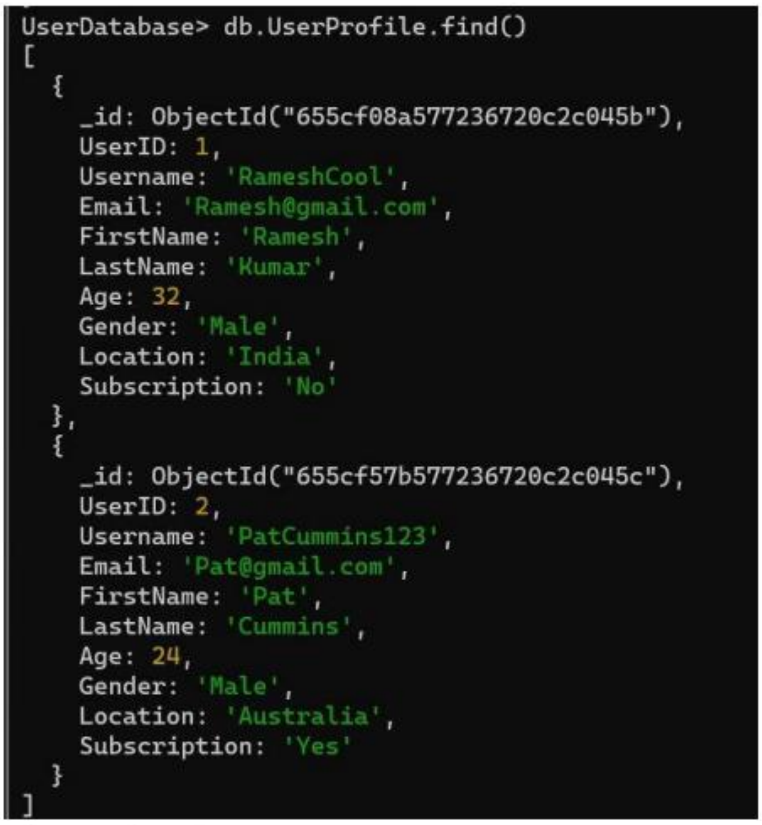


```
UserDatabase> show collections
Playlist
User Liked Songs
User Play History
User Profile
User Ratings
```

1)User Profile Table

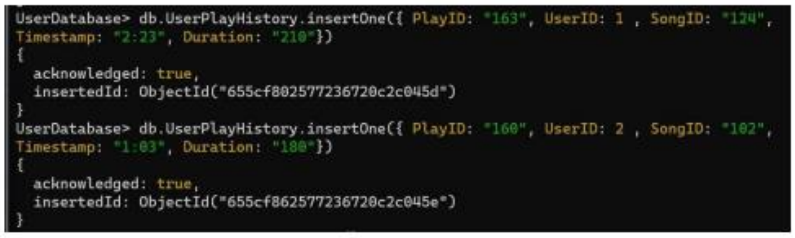


```
UserDatabase> db.UserProfile.insertOne({ UserID: 2, Username: "PatCummins123", Email: "
Pat@gmail.com", FirstName: "Pat", LastName: "Cummins", Age: 24, Gender: "Male", Locatio
n: "Australia", Subscription: "Yes" } )
{
  acknowledged: true,
  insertedId: ObjectId("655cf57b577236720c2c045c")
}
```

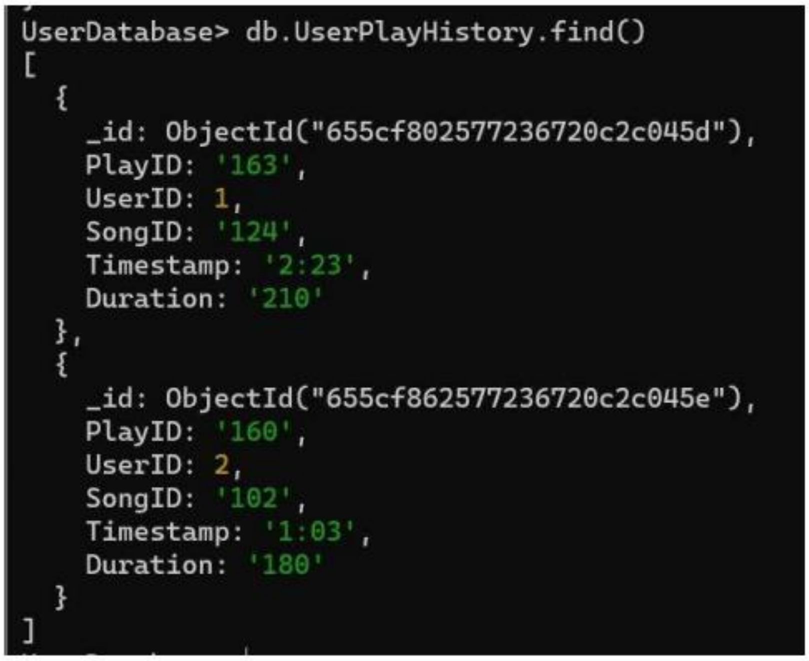


```
UserDatabase> db.UserProfile.find()
[
  {
    _id: ObjectId("655cf08a577236720c2c045b"),
    UserID: 1,
    Username: 'RameshCool',
    Email: 'Ramesh@gmail.com',
    FirstName: 'Ramesh',
    LastName: 'Kumar',
    Age: 32,
    Gender: 'Male',
    Location: 'India',
    Subscription: 'No'
  },
  {
    _id: ObjectId("655cf57b577236720c2c045c"),
    UserID: 2,
    Username: 'PatCummins123',
    Email: 'Pat@gmail.com',
    FirstName: 'Pat',
    LastName: 'Cummins',
    Age: 24,
    Gender: 'Male',
    Location: 'Australia',
    Subscription: 'Yes'
  }
]
```

2)User Play History

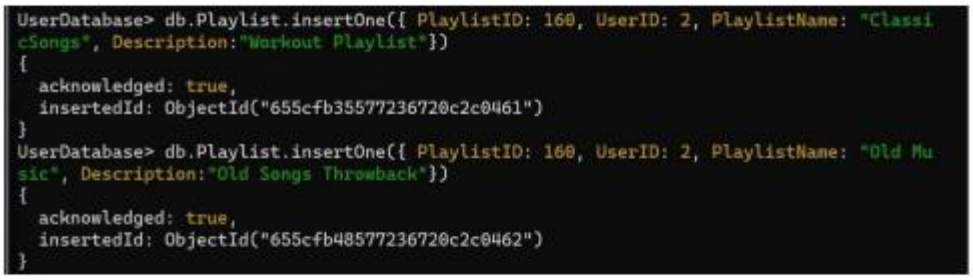


```
UserDatabase> db.UserPlayHistory.insertOne({ PlayID: "163", UserID: 1 , SongID: "124",
Timestamp: "2:23", Duration: "210"})
{
  acknowledged: true,
  insertedId: ObjectId("655cf802577236720c2c045d")
}
UserDatabase> db.UserPlayHistory.insertOne({ PlayID: "160", UserID: 2 , SongID: "102",
Timestamp: "1:03", Duration: "180"})
{
  acknowledged: true,
  insertedId: ObjectId("655cf862577236720c2c045e")
}
```

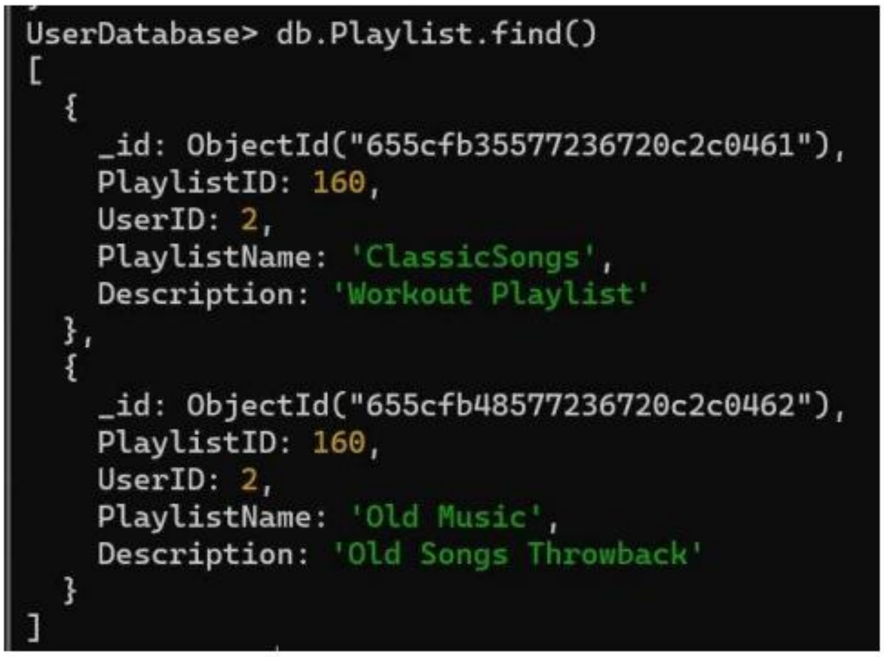


```
UserDatabase> db.UserPlayHistory.find()
[
  {
    _id: ObjectId("655cf802577236720c2c045d"),
    PlayID: '163',
    UserID: 1,
    SongID: '124',
    Timestamp: '2:23',
    Duration: '210'
  },
  {
    _id: ObjectId("655cf862577236720c2c045e"),
    PlayID: '160',
    UserID: 2,
    SongID: '102',
    Timestamp: '1:03',
    Duration: '180'
  }
]
```

3)Playlist

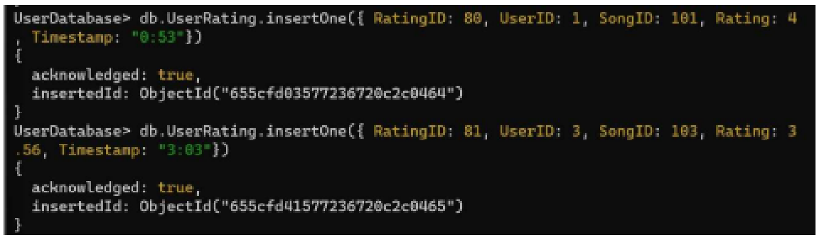


```
UserDatabase> db.Playlist.insertOne({ PlaylistID: 160, UserID: 2, PlaylistName: "Classi
cSongs", Description:"Workout Playlist"})
{
  acknowledged: true,
  insertedId: ObjectId("655cfb35577236720c2c0461")
}
UserDatabase> db.Playlist.insertOne({ PlaylistID: 160, UserID: 2, PlaylistName: "Old Mu
sic", Description:"Old Songs Throwback"})
{
  acknowledged: true,
  insertedId: ObjectId("655cfb48577236720c2c0462")
}
```

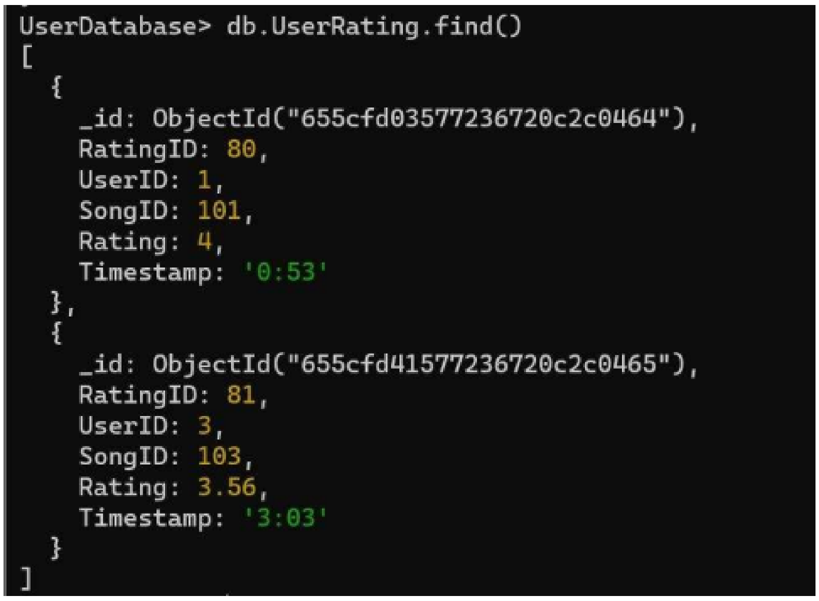


```
UserDatabase> db.Playlist.find()
[
  {
    _id: ObjectId("655cfb35577236720c2c0461"),
    PlaylistID: 160,
    UserID: 2,
    PlaylistName: 'ClassicSongs',
    Description: 'Workout Playlist'
  },
  {
    _id: ObjectId("655cfb48577236720c2c0462"),
    PlaylistID: 160,
    UserID: 2,
    PlaylistName: 'Old Music',
    Description: 'Old Songs Throwback'
  }
]
```

4)User Liked Songs

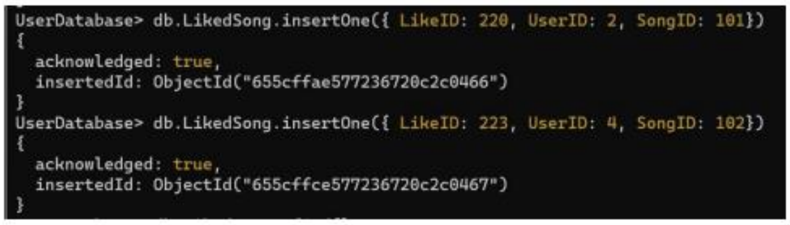


```
UserDatabase> db.UserRating.insertOne({ RatingID: 80, UserID: 1, SongID: 101, Rating: 4
, Timestamp: "0:53"})
{
  acknowledged: true,
  insertedId: ObjectId("655cfd03577236720c2c0464")
}
UserDatabase> db.UserRating.insertOne({ RatingID: 81, UserID: 3, SongID: 103, Rating: 3
.56, Timestamp: "3:03"})
{
  acknowledged: true,
  insertedId: ObjectId("655cfd41577236720c2c0465")
}
```

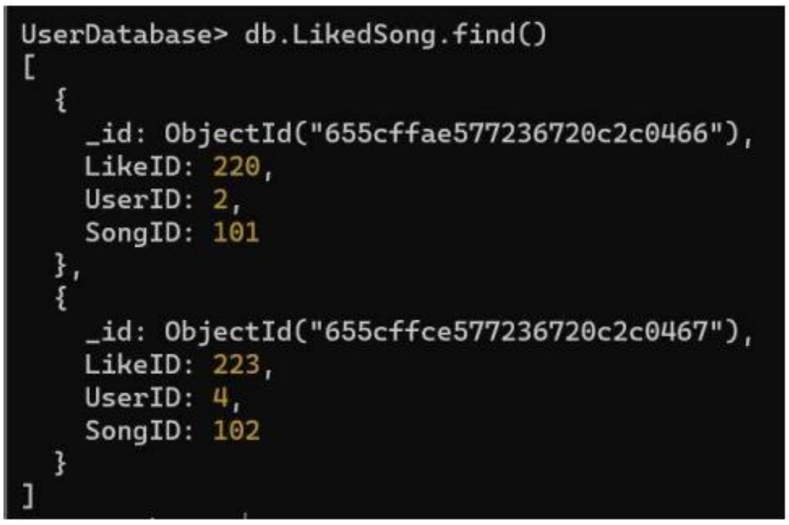


```
UserDatabase> db.UserRating.find()
[
  {
    _id: ObjectId("655cfd03577236720c2c0464"),
    RatingID: 80,
    UserID: 1,
    SongID: 101,
    Rating: 4,
    Timestamp: '0:53'
  },
  {
    _id: ObjectId("655cfd41577236720c2c0465"),
    RatingID: 81,
    UserID: 3,
    SongID: 103,
    Rating: 3.56,
    Timestamp: '3:03'
  }
]
```

5)User Ratings

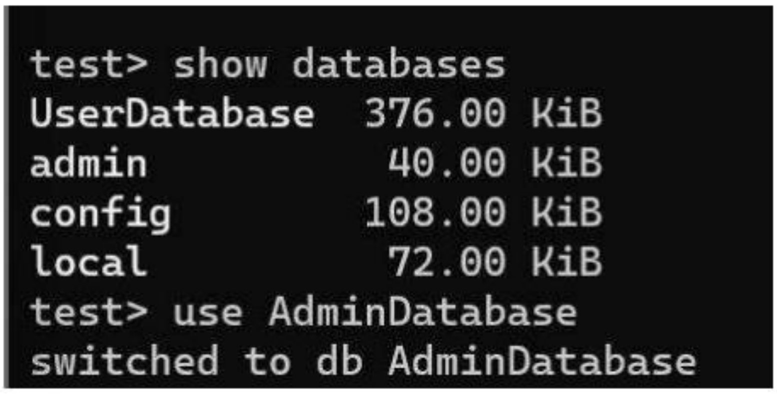


```
UserDatabase> db.LikedSong.insertOne({ LikeID: 220, UserID: 2, SongID: 101})
{
  acknowledged: true,
  insertedId: ObjectId("655cffae577236720c2c0466")
}
UserDatabase> db.LikedSong.insertOne({ LikeID: 223, UserID: 4, SongID: 102})
{
  acknowledged: true,
  insertedId: ObjectId("655cffce577236720c2c0467")
}
```



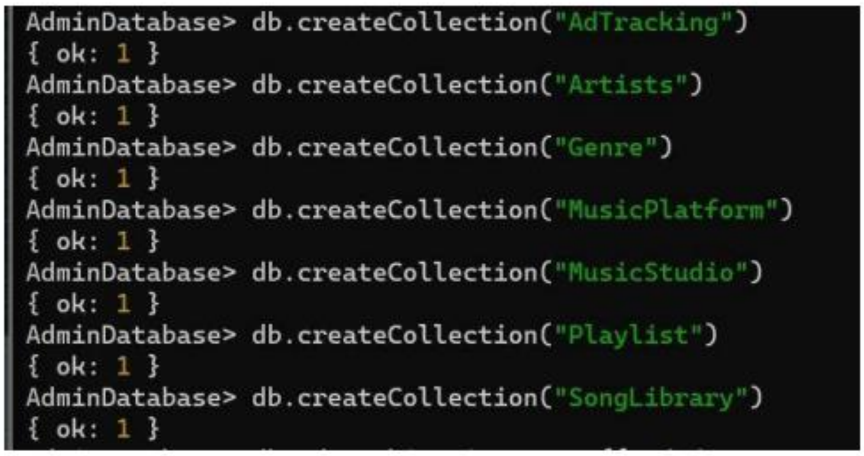
```
UserDatabase> db.LikedSong.find()
[
  {
    _id: ObjectId("655cffae577236720c2c0466"),
    LikeID: 220,
    UserID: 2,
    SongID: 101
  },
  {
    _id: ObjectId("655cffce577236720c2c0467"),
    LikeID: 223,
    UserID: 4,
    SongID: 102
  }
]
```

Making A Separate AdminDatabase



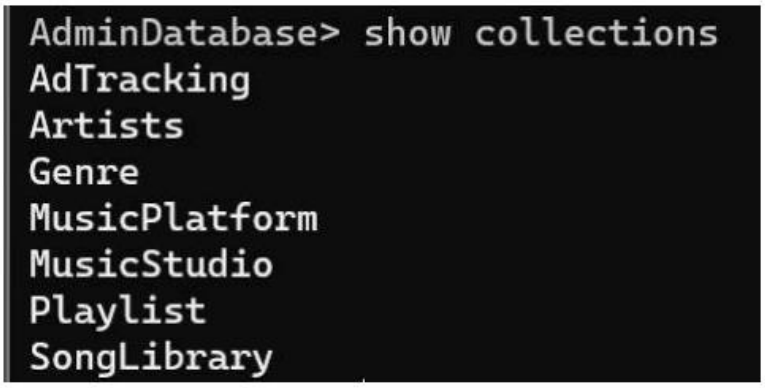
```
test> show databases
UserDatabase  376.00 KiB
admin          40.00 KiB
config        108.00 KiB
local          72.00 KiB
test> use AdminDatabase
switched to db AdminDatabase
```

Creating Collection



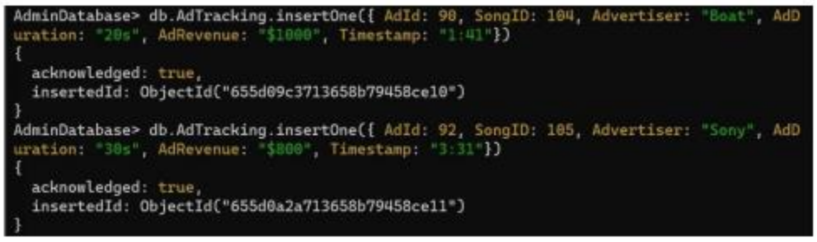
```
AdminDatabase> db.createCollection("AdTracking")
{ ok: 1 }
AdminDatabase> db.createCollection("Artists")
{ ok: 1 }
AdminDatabase> db.createCollection("Genre")
{ ok: 1 }
AdminDatabase> db.createCollection("MusicPlatform")
{ ok: 1 }
AdminDatabase> db.createCollection("MusicStudio")
{ ok: 1 }
AdminDatabase> db.createCollection("Playlist")
{ ok: 1 }
AdminDatabase> db.createCollection("SongLibrary")
{ ok: 1 }
```

Show Collections

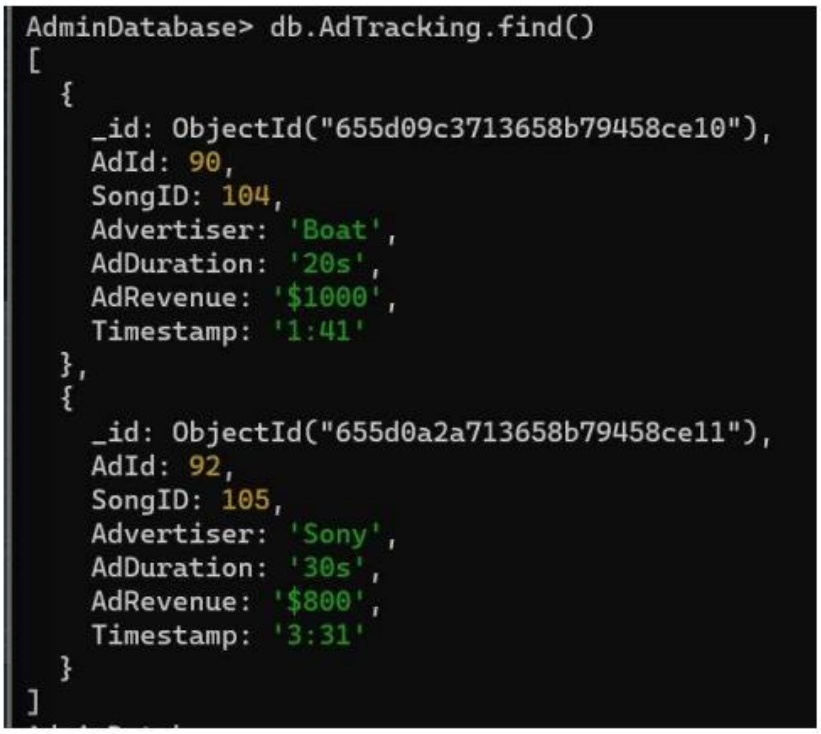


```
AdminDatabase> show collections
AdTracking
Artists
Genre
MusicPlatform
MusicStudio
Playlist
SongLibrary
```

1)AdTracking Table

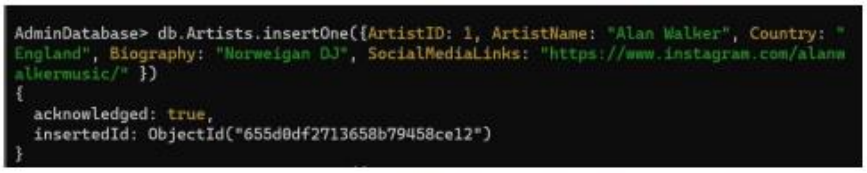


```
AdminDatabase> db.AdTracking.insertOne({ AdId: 90, SongID: 104, Advertiser: "Boat", AdD
uration: "20s", AdRevenue: "$1000", Timestamp: "1:41"})
{
  acknowledged: true,
  insertedId: ObjectId("655d09c3713658b79458ce10")
}
AdminDatabase> db.AdTracking.insertOne({ AdId: 92, SongID: 105, Advertiser: "Sony", AdD
uration: "30s", AdRevenue: "$800", Timestamp: "3:31"})
{
  acknowledged: true,
  insertedId: ObjectId("655d0a2a713658b79458ce11")
}
```

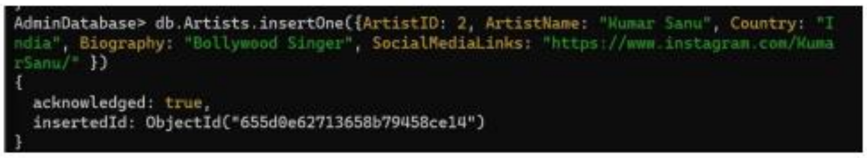


```
AdminDatabase> db.AdTracking.find()
[
  {
    _id: ObjectId("655d09c3713658b79458ce10"),
    AdId: 90,
    SongID: 104,
    Advertiser: 'Boat',
    AdDuration: '20s',
    AdRevenue: '$1000',
    Timestamp: '1:41'
  },
  {
    _id: ObjectId("655d0a2a713658b79458ce11"),
    AdId: 92,
    SongID: 105,
    Advertiser: 'Sony',
    AdDuration: '30s',
    AdRevenue: '$800',
    Timestamp: '3:31'
  }
]
```

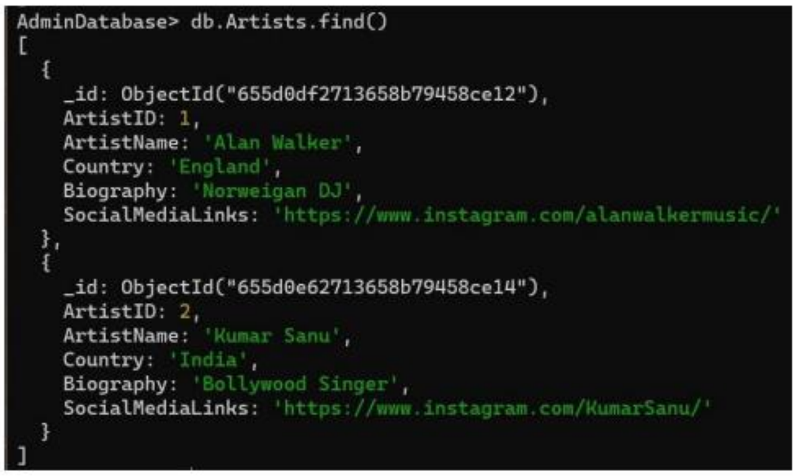
2)Artists Table



```
AdminDatabase> db.Artists.insertOne({ArtistID: 1, ArtistName: "Alan Walker", Country: "
England", Biography: "Norweigan DJ", SocialMediaLinks: "https://www.instagram.com/alanw
alkermusic/" })
{
  acknowledged: true,
  insertedId: ObjectId("655d0df2713658b79458ce12")
}
```

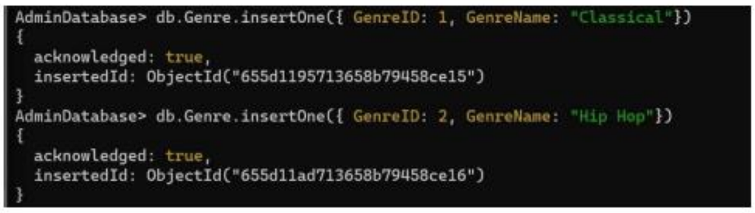


```
AdminDatabase> db.Artists.insertOne({ArtistID: 2, ArtistName: "Kumar Sanu", Country: "I
ndia", Biography: "Bollywood Singer", SocialMediaLinks: "https://www.instagram.com/Kuma
rSanu/" })
{
  acknowledged: true,
  insertedId: ObjectId("655d0e62713658b79458ce14")
}
```

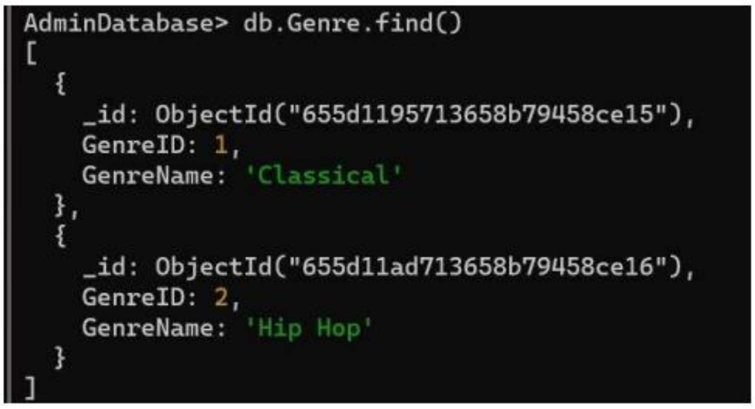


```
AdminDatabase> db.Artists.find()
[
  {
    _id: ObjectId("655d0df2713658b79458ce12"),
    ArtistID: 1,
    ArtistName: 'Alan Walker',
    Country: 'England',
    Biography: 'Norweigan DJ',
    SocialMediaLinks: 'https://www.instagram.com/alanwalkermusic/'
  },
  {
    _id: ObjectId("655d0e62713658b79458ce14"),
    ArtistID: 2,
    ArtistName: 'Kumar Sanu',
    Country: 'India',
    Biography: 'Bollywood Singer',
    SocialMediaLinks: 'https://www.instagram.com/KumarSanu/'
  }
]
```

<u>**3)Genre Table**</u>

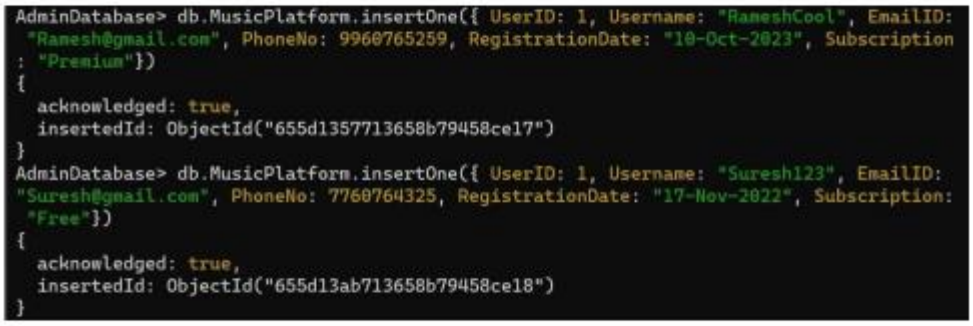


```
AdminDatabase> db.Genre.insertOne({ GenreID: 1, GenreName: "Classical"})
{
  acknowledged: true,
  insertedId: ObjectId("655d1195713658b79458ce15")
}
AdminDatabase> db.Genre.insertOne({ GenreID: 2, GenreName: "Hip Hop"})
{
  acknowledged: true,
  insertedId: ObjectId("655d11ad713658b79458ce16")
}
```

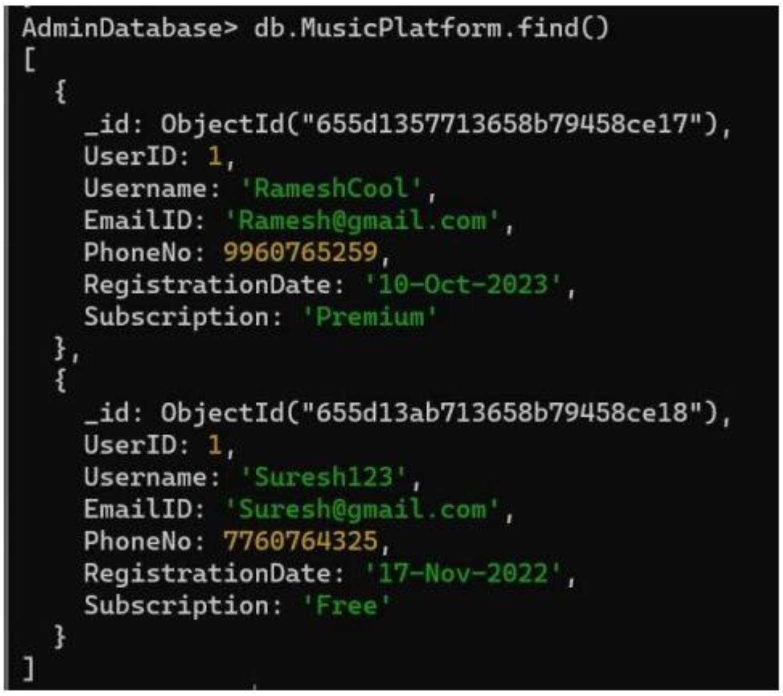


```
AdminDatabase> db.Genre.find()
[
  {
    _id: ObjectId("655d1195713658b79458ce15"),
    GenreID: 1,
    GenreName: 'Classical'
  },
  {
    _id: ObjectId("655d11ad713658b79458ce16"),
    GenreID: 2,
    GenreName: 'Hip Hop'
  }
]
```

<u>4)Music Platform Table</u>

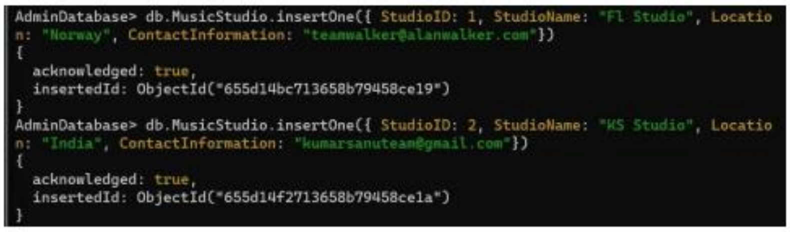


```
AdminDatabase> db.MusicPlatform.insertOne({ UserID: 1, Username: "RameshCool", EmailID:
 "Ramesh@gmail.com", PhoneNo: 9960765259, RegistrationDate: "10-Oct-2023", Subscription
: "Premium"})
{
  acknowledged: true,
  insertedId: ObjectId("655d1357713658b79458ce17")
}
AdminDatabase> db.MusicPlatform.insertOne({ UserID: 1, Username: "Suresh123", EmailID:
"Suresh@gmail.com", PhoneNo: 7760764325, RegistrationDate: "17-Nov-2022", Subscription:
 "Free"})
{
  acknowledged: true,
  insertedId: ObjectId("655d13ab713658b79458ce18")
}
```

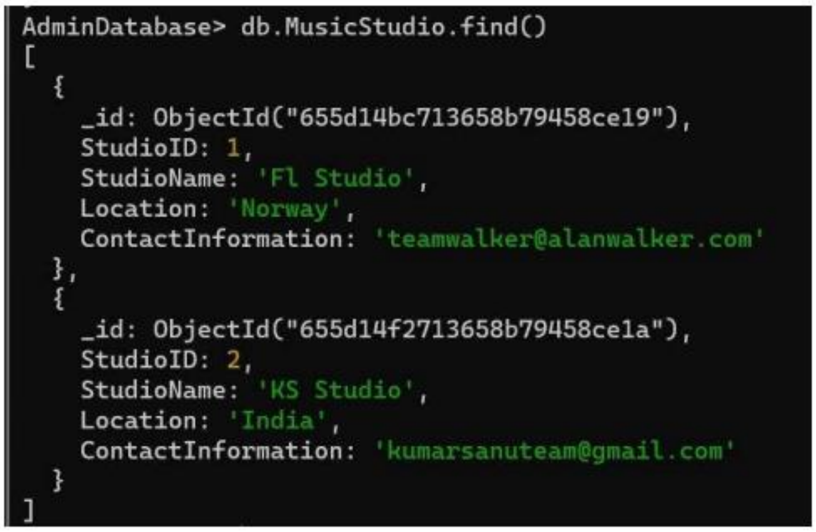


```
AdminDatabase> db.MusicPlatform.find()
[
  {
    _id: ObjectId("655d1357713658b79458ce17"),
    UserID: 1,
    Username: 'RameshCool',
    EmailID: 'Ramesh@gmail.com',
    PhoneNo: 9960765259,
    RegistrationDate: '10-Oct-2023',
    Subscription: 'Premium'
  },
  {
    _id: ObjectId("655d13ab713658b79458ce18"),
    UserID: 1,
    Username: 'Suresh123',
    EmailID: 'Suresh@gmail.com',
    PhoneNo: 7760764325,
    RegistrationDate: '17-Nov-2022',
    Subscription: 'Free'
  }
]
```

5)Music Studio Table

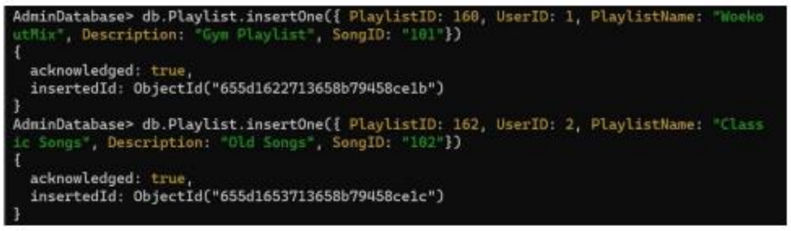


```
AdminDatabase> db.MusicStudio.insertOne({ StudioID: 1, StudioName: "Fl Studio", Locatio
n: "Norway", ContactInformation: "teamwalker@alanwalker.com"})
{
  acknowledged: true,
  insertedId: ObjectId("655d14bc713658b79458ce19")
}
AdminDatabase> db.MusicStudio.insertOne({ StudioID: 2, StudioName: "KS Studio", Locatio
n: "India", ContactInformation: "kumarsanuteam@gmail.com"})
{
  acknowledged: true,
  insertedId: ObjectId("655d14f2713658b79458ce1a")
}
```

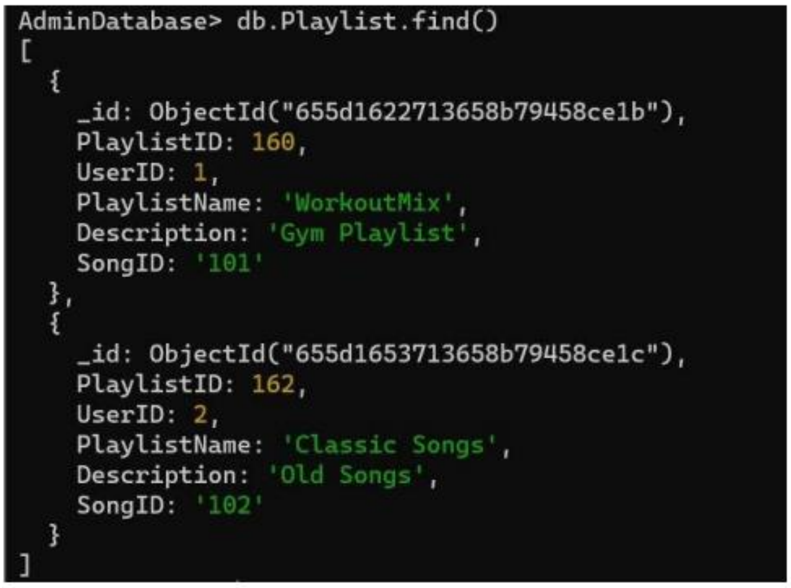


```
AdminDatabase> db.MusicStudio.find()
[
  {
    _id: ObjectId("655d14bc713658b79458ce19"),
    StudioID: 1,
    StudioName: 'Fl Studio',
    Location: 'Norway',
    ContactInformation: 'teamwalker@alanwalker.com'
  },
  {
    _id: ObjectId("655d14f2713658b79458ce1a"),
    StudioID: 2,
    StudioName: 'KS Studio',
    Location: 'India',
    ContactInformation: 'kumarsanuteam@gmail.com'
  }
]
```

6)Playlist Table

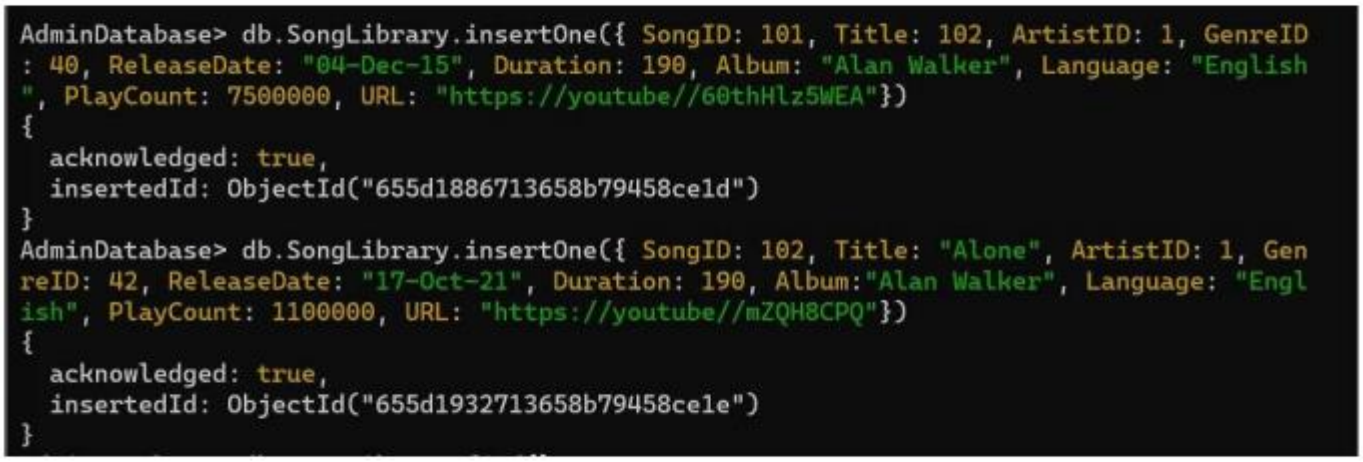


```
AdminDatabase> db.Playlist.insertOne({ PlaylistID: 160, UserID: 1, PlaylistName: "Woeko
utMix", Description: "Gym Playlist", SongID: "101"})
{
  acknowledged: true,
  insertedId: ObjectId("655d1622713658b79458ce1b")
}
AdminDatabase> db.Playlist.insertOne({ PlaylistID: 162, UserID: 2, PlaylistName: "Class
ic Songs", Description: "Old Songs", SongID: "102"})
{
  acknowledged: true,
  insertedId: ObjectId("655d1653713658b79458ce1c")
}
```

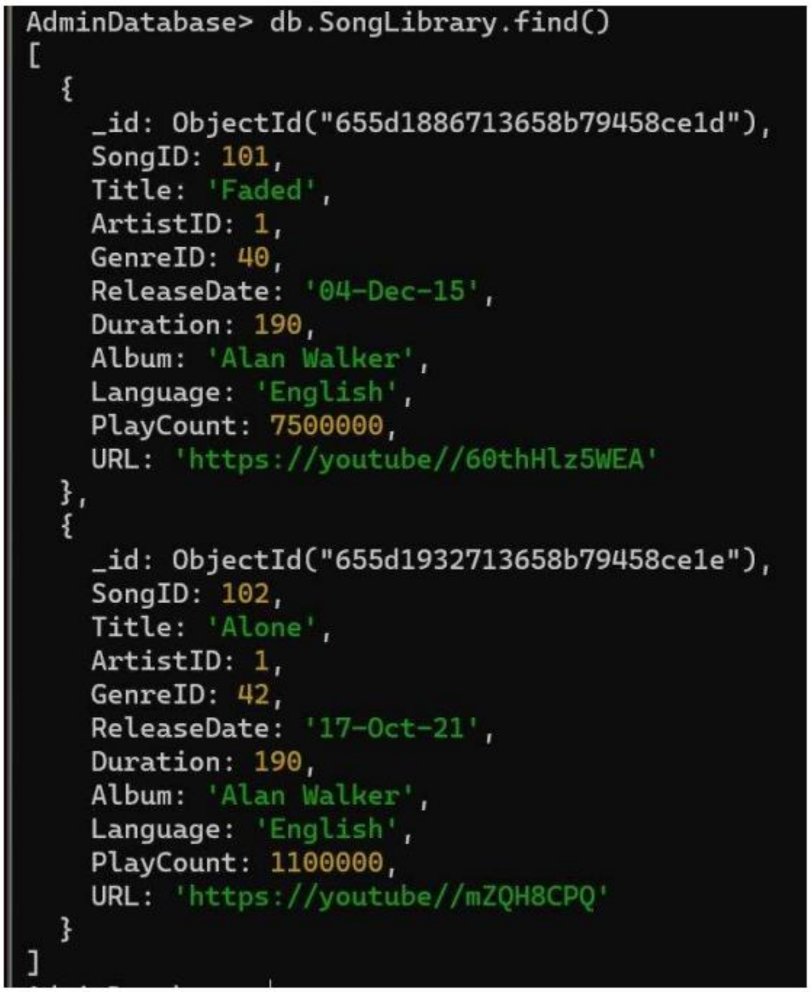


```
AdminDatabase> db.Playlist.find()
[
  {
    _id: ObjectId("655d1622713658b79458ce1b"),
    PlaylistID: 160,
    UserID: 1,
    PlaylistName: 'WorkoutMix',
    Description: 'Gym Playlist',
    SongID: '101'
  },
  {
    _id: ObjectId("655d1653713658b79458ce1c"),
    PlaylistID: 162,
    UserID: 2,
    PlaylistName: 'Classic Songs',
    Description: 'Old Songs',
    SongID: '102'
  }
]
```

7)Song Library Table

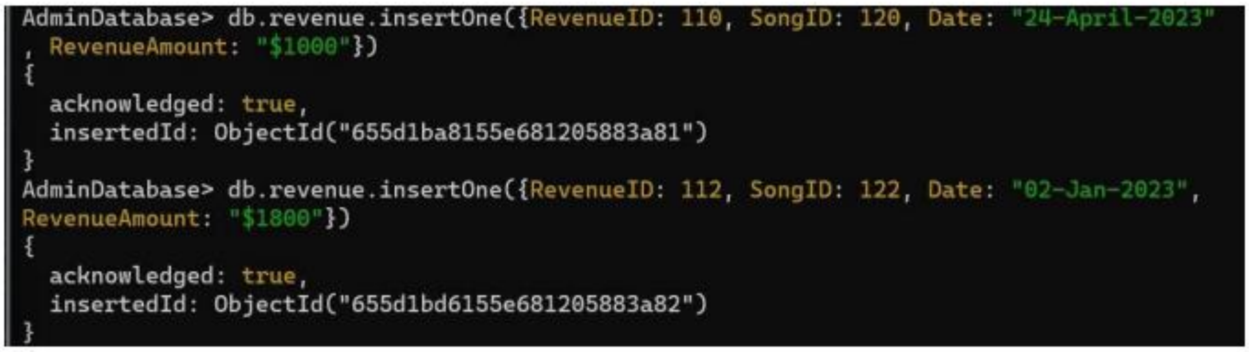


```
AdminDatabase> db.SongLibrary.insertOne({ SongID: 101, Title: 102, ArtistID: 1, GenreID
: 40, ReleaseDate: "04-Dec-15", Duration: 190, Album: "Alan Walker", Language: "English
", PlayCount: 7500000, URL: "https://youtube//60thHlz5WEA"})
{
  acknowledged: true,
  insertedId: ObjectId("655d1886713658b79458ce1d")
}
AdminDatabase> db.SongLibrary.insertOne({ SongID: 102, Title: "Alone", ArtistID: 1, Gen
reID: 42, ReleaseDate: "17-Oct-21", Duration: 190, Album:"Alan Walker", Language: "Engl
ish", PlayCount: 1100000, URL: "https://youtube//mZQH8CPQ"})
{
  acknowledged: true,
  insertedId: ObjectId("655d1932713658b79458ce1e")
}
```

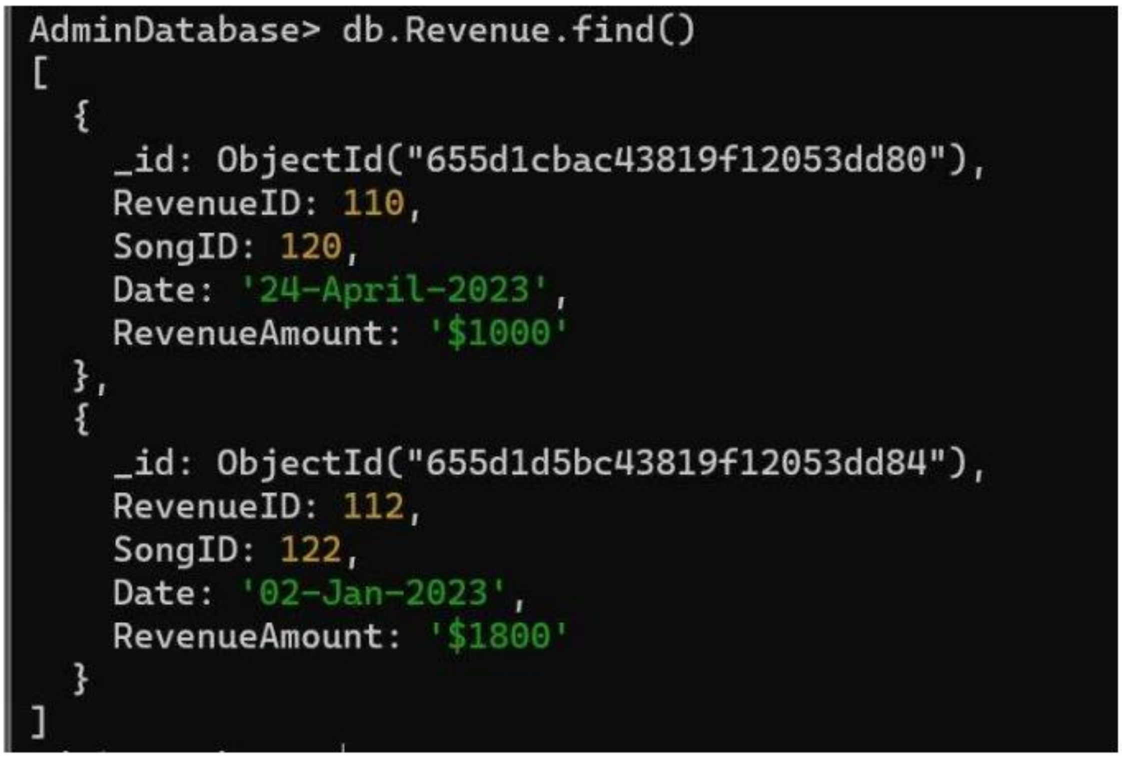


```
AdminDatabase> db.SongLibrary.find()
[
  {
    _id: ObjectId("655d1886713658b79458ce1d"),
    SongID: 101,
    Title: 'Faded',
    ArtistID: 1,
    GenreID: 40,
    ReleaseDate: '04-Dec-15',
    Duration: 190,
    Album: 'Alan Walker',
    Language: 'English',
    PlayCount: 7500000,
    URL: 'https://youtube//60thHlz5WEA'
  },
  {
    _id: ObjectId("655d1932713658b79458ce1e"),
    SongID: 102,
    Title: 'Alone',
    ArtistID: 1,
    GenreID: 42,
    ReleaseDate: '17-Oct-21',
    Duration: 190,
    Album: 'Alan Walker',
    Language: 'English',
    PlayCount: 1100000,
    URL: 'https://youtube//mZQH8CPQ'
  }
]
```

8)Revenue Table



```
AdminDatabase> db.revenue.insertOne({RevenueID: 110, SongID: 120, Date: "24-April-2023"
, RevenueAmount: "$1000"})
{
  acknowledged: true,
  insertedId: ObjectId("655d1ba8155e681205883a81")
}
AdminDatabase> db.revenue.insertOne({RevenueID: 112, SongID: 122, Date: "02-Jan-2023",
RevenueAmount: "$1800"})
{
  acknowledged: true,
  insertedId: ObjectId("655d1bd6155e681205883a82")
}
```



```
AdminDatabase> db.Revenue.find()
[
  {
    _id: ObjectId("655d1cbac43819f12053dd80"),
    RevenueID: 110,
    SongID: 120,
    Date: '24-April-2023',
    RevenueAmount: '$1000'
  },
  {
    _id: ObjectId("655d1d5bc43819f12053dd84"),
    RevenueID: 112,
    SongID: 122,
    Date: '02-Jan-2023',
    RevenueAmount: '$1800'
  }
]
```