In [25]:

```
import sqlite3
import pandas as pd
import pymongo
```

SQLite

1.1 Connecting to DB

In [26]:

```
# Connect to the Database - the DATABASE must already exisit
conn=sqlite3.connect('S:\\DB\\cloud_music.db')

# conn is an object of the Connection class - the next command is only for display
print(type(conn))

# The connection object (conn) has access to various methods of the Connection class.
# We are using the method cursor() and which returns a cursor object.
# The cursor object is essential to perform any operation on the database (CRUD operations)
cur=conn.cursor()

# The next command is only for display
print(type(cur))
```

```
<class 'sqlite3.Connection'>
<class 'sqlite3.Cursor'>
```

1.2 Listing tables

In [27]:

```
# Listing all tables in your database (suggest using SQLIte prompt and command .tables)
# Once we have the cursor object, we can perform all SQL operations with the help of execut

# define a function
def tables_in_sqlite_db(conn):
    cursor = conn.execute("SELECT name FROM sqlite_master WHERE type='table';")
    tables = [
        v[0] for v in cursor.fetchall()
        if v[0] != "sqlite_sequence"
    ]
    cursor.close()
    return tables
```

In [28]:

```
# call the function and pass connection object
tables = tables_in_sqlite_db(conn)

#print tables in the current database
print(tables)
```

```
['user', 'singer', 'music', 'genre']
```

1.3 Adding tables using SQL DDL Statement

In [29]:

```
# Create a new table called user
# Note: If the table is created succesfully, executing the code again will lead to exceptio
# Once we have the cursor object, we can perform all SQL operations with the help of .execu

create_user='''
create table user (
user_ID text primary key,
user_name text (30),
user_gender text (30),
user_birth date (10)
);


'''
try:
        cur.execute(create_user)
        print ('Table created successfully')
except:
# If table already exists then use the SQLite console to connect to the database (BEMM459.d
# .. Altenatively, use the SQLiteStudio GUI to delete table ECC_DEPARTMENT and execute this
        print ('Error in creating table')
```

```
Error in creating table
```

In [30]:

```
# Create four tables using the method executescript() that is defined in the cursor class.
# executescript() makes it possible to execute multiple SQL statements.

create_qry='''
create table singer (
        singer_ID text primary key,
        singer_name text (40),
        singer_gender text (40),
        singer_bio text (200)
    );

create table music (
        music_ID text primary key,
        music_name text (40),
        music_playtime integer,
        music_singer text references singer(singer_ID),
        music_comment text references comment(comment_ID),
        music_genre_ID text references genre(genre_ID)
    );

create table genre (
       genre_ID text primary key,
       genre_name text (40),
       genre_brief text (200),
       related_music text references music(music_ID)
    );

'''
try:
        cur.executescript(create_qry)
        print ('Tables created successfully')
except:
        print ('Error in creating tables')
```

```
Error in creating tables
```

In [31]:

```
def c(x):
    cur.execute(x)
    rows=cur.fetchall()
    rows=pd.DataFrame(rows)
    display(rows)
def ce(x):
    cur.execute(x)
```

In [32]:

```
c("SELECT * FROM sqlite_master WHERE type='table';")
```

	0	1	2	3	4
0	table	user	user	2	CREATE TABLE user (\nuser_ID text primary key,...
1	table	singer	singer	4	CREATE TABLE singer (\n singer_ID text ...
2	table	music	music	6	CREATE TABLE music (\n music_ID text pr...
3	table	genre	genre	8	CREATE TABLE genre (\n genre_ID text pri...

In [ ]:

```

```

In [33]:

```
#create genre-music link table
```

In [34]:

```
create_link='''
CREATE TABLE genre_music(
   genre_ID TEXT,
   music_ID TEXT,
   PRIMARY KEY (genre_ID, music_ID),
   FOREIGN KEY (genre_ID)
      REFERENCES genre (genre_ID)
         ON DELETE CASCADE
         ON UPDATE NO ACTION,
   FOREIGN KEY (music_ID)
      REFERENCES music (music_ID)
         ON DELETE CASCADE
         ON UPDATE NO ACTION
);
'''
try:
        cur.executescript(create_link)
        print ('Tables created successfully')
except:
        print ('Error in creating tables')
```

```
Tables created successfully
```

In [35]:

```
c("SELECT * FROM genre_music;")
```

In [ ]:

```
```

In [ ]:

```
```

In [36]:

```
# check to see if the new table is created
# call the function and pass connection object
tables = tables_in_sqlite_db(conn)

#print tables in the current database
print(tables)
```

```
['user', 'singer', 'music', 'genre', 'genre_music']
```

In [ ]:

```
```

In [ ]:

```
```

1.4 Inserting records using SQL DML Statement

In [37]:

```
# Insert one record in table user
# Commit only if no exception is encountered, else rollback

qry="insert into user values ('U1', 'Yizhen', 'male','1994-05-25');"

try:
        cur.execute(qry)
        conn.commit()
        print ('One record inserted successfully..commit')
except:
        print ('Error in insert operation..rollback')
        conn.rollback()
        print(conn)
```

```
One record inserted successfully..commit
```

In [38]:

```
# Insert multiple records in table user

user_value="insert into user (user_ID, user_name, user_gender,user_birth) values (?,?,?,?);
user_list=[('U2','Gavin','male','1996-08-19'),
           ('U3', 'James','male','1998-03-07'),
           ('U4','Liam', 'male','1995-04-03'),
           ('U5','Kun', 'female','1996-06-18'),
           ('U6','Leon', 'male','1960-07-07'),
           ('U7','Lucy', 'female','1919-02-06'),
           ('U8','Sky', 'male','1979-01-05'),
           ('U9','Alice', 'female','1980-01-06'),
           ('U11','Stella', 'female','1991-01-06'),
           ('U12','Miao', 'male','1972-05-07'),
           ('U13','Xue', 'male','1997-01-02'),
           ('U14','Mia', 'they','1999-11-05'),
           ('U15','Mos', 'female','1997-11-06'),
           ('U16','Ting', 'female','1996-01-06'),
           ('U17','Jack', 'male','1999-01-09'),
           ('U18','Ball', 'female','1992-02-04'),
           ('U19','Jim', 'male','1989-09-06'),
           ('U20','James', 'male','1988-01-07')]


try:
        cur.executemany(user_value, user_list)
        conn.commit()
        print ('Records inserted successfully..committed')
except:
        print ('Error in insert operation..rollback')
        conn.rollback()
```

```
Records inserted successfully..committed
```

In [39]:

```
singer_values="insert into singer (singer_ID, singer_Name, singer_gender,singer_bio) values
singer_list=[('S1','silly and idiot','They','A band from Taiwan, China'),
           ('S2', 'blur','They','One of the most greatest band of England'),
           ('S3','Calvin Harris', 'He','DJ'),
           ('S4','Rich Brian','he','a young artist of Hip-hop'),
             ('S5','Oasis','they','as good as blur'),
             ('S6','Suede','they','super awesome band'),
             ('S7','Nada Surf','they','worth to listen'),

             ]

try:
        cur.executemany(singer_values, singer_list)
        conn.commit()
        print ('Records inserted successfully..committed')
except:
        print ('Error in insert operation..rollback')
        conn.rollback()
```

```
Records inserted successfully..committed
```

In [40]:

```
c("SELECT * FROM singer;")
```

	0	1	2	3
0	S1	silly and idiot	They	A band from Taiwan, China
1	S2	blur	They	One of the most greatest band of England
2	S3	Calvin Harris	He	DJ
3	S4	Rich Brian	he	a young artist of Hip-hop
4	S5	Oasis	they	as good as blur
5	S6	Suede	they	super awesome band
6	S7	Nada Surf	they	worth to listen

In [ ]:

```

```

In [ ]:

```

```

INSERT VALUE FOR MUSIC

In [41]:

```
qry="insert into music values ('M1','HoydeA',25686,'S1','M1_C1','G1');"

try:
        cur.execute(qry)
        conn.commit()
        print ('One record inserted successfully..commit')
except:
        print ('Error in insert operation..rollback')
        conn.rollback()
        print(conn)
```

```
One record inserted successfully..commit
```

In [42]:

```
qry="insert into music values ('M2', 'Charmless Man',719430,'S2','M2_C1','G2');"

try:
        cur.execute(qry)
        conn.commit()
        print ('One record inserted successfully..commit')
except:
        print ('Error in insert operation..rollback')
        conn.rollback()
        print(conn)
```

```
One record inserted successfully..commit
```

In [ ]:

In [43]:

```
# The next command is only for display
print(type(cur))
# Preparing SQL queries to INSERT a record into the database.


cur.execute('''INSERT INTO music(
   music_ID, music_name, music_playtime, music_singer, music_comment,music_genre_ID) VALUES
    ('M4', 'Love in my pocket',297456,'S4','M4_C1','G4')''')

cur.execute('''INSERT INTO music(
   music_ID, music_name, music_playtime, music_singer, music_comment,music_genre_ID) VALUES
    ('M6','Parklife',5284667,'S2','M6_C1','G2')''')

cur.execute('''INSERT INTO music(
   music_ID, music_name, music_playtime, music_singer, music_comment,music_genre_ID) VALUES
    ('M7','Beautiful Ones',439764,'S6','S6_C1','G2')''')

cur.execute('''INSERT INTO music(
   music_ID, music_name, music_playtime, music_singer, music_comment,music_genre_ID) VALUES
    ('M5', 'Dont look back at anger',98642,'S5','M5_C1','G2')''')

cur.execute('''INSERT INTO music(
   music_ID, music_name, music_playtime, music_singer, music_comment,music_genre_ID) VALUES
   ('M8', 'wonderwall', 23457, 'S5', 'M8_C1','G2')''')

cur.execute('''INSERT INTO music(
   music_ID, music_name, music_playtime, music_singer, music_comment,music_genre_ID) VALUES
   ('M9', 'Tender', 543457, 'S2', 'M6_C1','G2')''')

cur.execute('''INSERT INTO music(
   music_ID, music_name, music_playtime, music_singer, music_comment,music_genre_ID) VALUES
   ('M10', 'Good Song', 373457, 'S2', 'M10_C1','G2')''')


# Commit your changes in the database
conn.commit()
print("Records inserted........")
```

```
<class 'sqlite3.Cursor'>
Records inserted........
```

In [44]:

```
c("SELECT * FROM music;")
```

	0	1	2	3	4	5
0	M1	HoydeA	25686	S1	M1_C1	G1
1	M2	Charmless Man	719430	S2	M2_C1	G2
2	M4	Love in my pocket	297456	S4	M4_C1	G4
3	M6	Parklife	5284667	S2	M6_C1	G2
4	M7	Beautiful Ones	439764	S6	S6_C1	G2
5	M5	Dont look back at anger	98642	S5	M5_C1	G2
6	M8	wonderwall	23457	S5	M8_C1	G2
7	M9	Tender	543457	S2	M6_C1	G2
8	M10	Good Song	373457	S2	M10_C1	G2

In [ ]:

```

```

insert values in genre

In [45]:

```
genre_values="insert into genre (genre_ID, genre_name,genre_brief, related_music) values (?
genre_list=[('G1','Indie Rock','Indie Rock','M1'),
           ('G2', 'British Pop','Pop','M2'and'M6'and'M7'and 'M5'and'M8'and 'M9'),
           ('G3','Light Music', 'Light Music','M10'),
            ('G4','Hip-pop','Hip-pop','M4')

             ]

try:
        cur.executemany(genre_values, genre_list)
        conn.commit()
        print ('Records inserted successfully..committed')
except:
        print ('Error in insert operation..rollback')
        conn.rollback()
```

```
Records inserted successfully..committed
```

In [46]:

```
c("SELECT * FROM genre;")
```

	0	1	2	3
0	G1	Indie Rock	Indie Rock	M1
1	G2	British Pop	Pop	M9
2	G3	Light Music	Light Music	M10
3	G4	Hip-pop	Hip-pop	M4

In [54]:

```
c("SELECT * FROM genre_music;")
```

	0	1
0	G1	M1
1	G2	M2
2	G2	M6
3	G2	M7
4	G2	M5
5	G2	M8
6	G2	M9
7	G3	M10
8	G4	M4

In [48]:

```
# check to see if the new table is created
# call the function and pass connection object
tables = tables_in_sqlite_db(conn)

#print tables in the current database
print(tables)
```

```
['user', 'singer', 'music', 'genre', 'genre_music']
```

this function is an interactive, to let user can type in information

In [49]:

```
# Insert one record in table user based on user input

input_user_ID=input('Enter user ID:')
input_user_Name=input('Enter use name:')
input_gender=input('Enter user gender:')
input_birth=input('Enter user birthday(year-month-day):')
qry="insert into user values (?,?,?,?);"
try:
        cur.execute(qry, (input_user_ID,input_user_Name, input_gender,input_birth))
        print ('New user added')
        conn.commit()
except:
        print ('Error in adding department .. rollback')
        conn.rollback()
```

```
Enter user ID:98
Enter use name:jack
Enter user gender:male
Enter user birthday(year-month-day):1990-08-28
New user added
```

In [50]:

```
#add a new table to calculate users' age which change with the crruent time.
alter_table = 'alter table user add column age integer;'
cur.execute(alter_table)

#query age from user
cur.execute("select cast(strftime('%Y.%m%d', 'now') - strftime('%Y.%m%d', user_birth) as in

rows=cur.fetchall()

age_list = []
for row in rows:
    age_list.append(row)


insert_age="update user set age=? where user_ID =?;"

try:
        cur.executemany(insert_age, age_list)
        conn.commit()
        print ('Records inserted successfully..committed')
except:
        print ('Error in insert operation..rollback')
        conn.rollback()
```

```
Records inserted successfully..committed
```

1.5 Querying data

In [51]:

```
# Query and display one record from the table user

# Prepare the query String
user="select * from user;"

# Execute query on SQLite
cur.execute(user)

# Fetch and display one row
row=cur.fetchone()
row
```

Out[51]:

```
('U1', 'Yizhen', 'male', '1994-05-25', 27)
```

In [52]:

```
# Fetch and display all rows
user="select * from user;"
rows=cur.execute(user).fetchall()
rows
c("SELECT * FROM user ;")
```

	0	1	2	3	4
0	U1	Yizhen	male	1994-05-25	27
1	U2	Gavin	male	1996-08-19	24
2	U3	James	male	1998-03-07	23
3	U4	Liam	male	1995-04-03	26
4	U5	Kun	female	1996-06-18	25
5	U6	Leon	male	1960-07-07	61
6	U7	Lucy	female	1919-02-06	102
7	U8	Sky	male	1979-01-05	42
8	U9	Alice	female	1980-01-06	41
9	U11	Stella	female	1991-01-06	30
10	U12	Miao	male	1972-05-07	49
11	U13	Xue	male	1997-01-02	24
12	U14	Mia	they	1999-11-05	21
13	U15	Mos	female	1997-11-06	23
14	U16	Ting	female	1996-01-06	25
15	U17	Jack	male	1999-01-09	22
16	U18	Ball	female	1992-02-04	29
17	U19	Jim	male	1989-09-06	31
18	U20	James	male	1988-01-07	33
19	98	jack	male	1990-08-28	30

In [17]:

```
# Query and display one record from the table user

# Prepare the query String
user="select * from user;"

# Execute query on SQLite
cur.execute(user)

# Fetch and display one row
row=cur.fetchall()
row
```

Out[17]:

```
[('U1', 'Yizhen', 'male', '1994-05-25', 27),
 ('U2', 'Gavin', 'male', '1996-08-19', 24),
 ('U3', 'James', 'male', '1998-03-07', 23),
 ('U4', 'Liam', 'male', '1995-04-03', 26),
 ('U5', 'Kun', 'female', '1996-06-18', 25),
 ('U6', 'Leon', 'male', '1960-07-07', 61),
 ('U7', 'Lucy', 'female', '1919-02-06', 102),
 ('U8', 'Sky', 'male', '1979-01-05', 42),
 ('U9', 'Alice', 'female', '1980-01-06', 41),
 ('U11', 'Stella', 'female', '1991-01-06', 30),
 ('U12', 'Miao', 'male', '1972-05-07', 49),
 ('U13', 'Xue', 'male', '1997-01-02', 24),
 ('U14', 'Mia', 'they', '1999-11-05', 21),
 ('U15', 'Mos', 'female', '1997-11-06', 23),
 ('U16', 'Ting', 'female', '1996-01-06', 25),
 ('U17', 'Jack', 'male', '1999-01-09', 22),
 ('U18', 'Ball', 'female', '1992-02-04', 29),
 ('U19', 'Jim', 'male', '1989-09-06', 31),
 ('U20', 'James', 'male', '1988-01-07', 33),
 ('98', 'yizhen', 'femal', '1994-95-25', None)]
```

In [31]:

```
# Query and display records from the table user based on user input

input_user_ID=input ('Enter user number:')
cur.execute("select * from user where user_ID=?", (input_user_ID,))
row=cur.fetchone()
# IMPORTANT: Individual items in the tuple can be accessed by index .. we are accessing the
print ("user name is ", row[1],",",row[2],",",row[4],"years old")
```

```
Enter user number:U2
user name is  Gavin , male , 24 years old
```

1.6 Updating data using SQL DML Statement

In [53]:

```
# Update table based on user input

input_user_ID=input('Enter user ID to update:')
input_name=input('Enter new user name:')
qry='update user set user_name=? where user_ID=?'

try:
        cur.execute(qry, (input_user_ID,input_name))
        print ('user name updated')
        conn.commit()
except:
        print ('Error in update operation .. rollback')
        conn.rollback()
```

```
Enter user ID to update:98
Enter new user name:Jack
user name updated
```

In [ ]:

```

```

In [ ]:

```

```

1.7 Deleting data using SQL DML Statement

In [55]:

```
# Delete record based on user input

input_user_ID=input('Enter name of user ID to delete:')
qry='delete from user where user_ID=?'
try:
        cur.execute(qry, (input_user_ID,))
        print ('user_ID deleted')
        conn.commit()
except:
        print ('Error in deleting user', input_user_ID)
        conn.rollback()
```

```
Enter name of user to delete:98
user_ID deleted
```

1.8 filter

In [56]:

```
#filter all the male and age>27
cur.execute("select * from user where user_gender = 'male' and age>27;")
rows=cur.fetchall()
for limit in rows:
    print (limit)
```

```
('U6', 'Leon', 'male', '1960-07-07', 61)
('U8', 'Sky', 'male', '1979-01-05', 42)
('U12', 'Miao', 'male', '1972-05-07', 49)
('U19', 'Jim', 'male', '1989-09-06', 31)
('U20', 'James', 'male', '1988-01-07', 33)
```

In [57]:

```
#join music and user and delete the null
cur.execute("select * from music left join singer on music.music_singer=singer.singer_ID wh
rows=cur.fetchall()
for limit in rows:
    print (limit)
```

```
('M1', 'HoydeA', 25686, 'S1', 'M1_C1', 'G1', 'S1', 'silly and idiot', 'The
y', 'A band from Taiwan, China')
('M5', 'Dont look back at anger', 98642, 'S5', 'M5_C1', 'G2', 'S5', 'Oasis',
'they', 'as good as blur')
('M4', 'Love in my pocket', 297456, 'S4', 'M4_C1', 'G4', 'S4', 'Rich Brian',
'he', 'a young artist of Hip-hop')
('M7', 'Beautiful Ones', 439764, 'S6', 'S6_C1', 'G2', 'S6', 'Suede', 'they',
'super awesome band')
('M2', 'Charmless Man', 719430, 'S2', 'M2_C1', 'G2', 'S2', 'blur', 'They',
'One of the most greatest band of England')
```

1.9 Database Backup and Restore

In [60]:

```
#
# Creating databse dump ... we are connected to Youtube.db
# If conn object is closed then uncomment the next line and execute code
# conn=sqlite3.connect('S\\Youtube.db')

file=open('cloud_database.sql','w')

for line in conn.iterdump():
        file.write('{}\n'.format(line))

file.close()

# Closing database connection
# conn.close()
```

In [61]:

```
# Creating new database, reading content of the dump file and executing SQL statements in i
connRestore=sqlite3.connect('S:\\DB\\cloud_database.db')
file=open('cloud_database.sql','r')
qry=file.read()
file.close()

curRestore=connRestore.cursor()
curRestore.executescript(qry)

# call function (defined earlier) and pass connection object
tables = tables_in_sqlite_db(connRestore)

#print tables in the newly restored database
print(tables)

connRestore.close()
```

```
['genre', 'genre_music', 'music', 'singer', 'user']
```

In [62]:

```
# Close database connection to cloud_database.db
conn.close()
```

In [ ]:

```

```

In [ ]:

```

```

2. Nosql

2.1 Connectivity

In [63]:

```
mongoclient = pymongo.MongoClient("mongodb://localhost:7901")

#Check what databases exist - the output is a list of database names
print(mongoclient.list_database_names())

#You can also check databases that presently exisit using a loop
dblist = mongoclient.list_database_names()
for x in dblist:
    print(x)
```

```
['Youtube_music_nosql', 'Youtube_nosql', 'admin', 'config', 'local', 'test']
Youtube_music_nosql
Youtube_nosql
admin
config
local
test
```

2.2 Defining user-defined functions for later use

In [64]:

```
#Defining a user function to check if database exists - In MongoDB, a database is not creat
def check_DatabaseExists(argDBName):
    local_dblist = mongoclient.list_database_names()
    if argDBName in local_dblist:
        print("The database ", argDBName, " exists.")
    else:
        print("The database ", argDBName, " does not exist.")

#Defining a user function to check if a collection exists - In MongoDB, a collection is not
def check_CollectionExists(argDBName, argCollName, local_mydb):
    local_mydb = mongoclient[argDBName]
    local_collist = local_mydb.list_collection_names()
    if argCollName in local_collist:
        print("The collection ",  argCollName, "exists in database ", argDBName)
    else:
        print("The collection ", argCollName, " does not exist in database ", argDBName)
```

2.3 Create new MongoDB database

In [66]:

```
#Create a new database
mydb = mongoclient["cloud_music_nosql"]
print(type(mydb))

#Check if database exists by calling function check_DatabaseExists with name of database as
check_DatabaseExists("cloud_music_nosql")

'''
#Without a function the code will be as follows
if "cloud_nosql" in dblist:
    print("The database 'cloud_music_nosql' exists.")
else:
    print("The database 'cloud_music_nosql' does not exist.")
'''

print()
```

```
<class 'pymongo.database.Database'>
The database  cloud_music_nosql  does not exist.
```

2.4 Create new collection

In [67]:

```
#Return a list of all collections in your database:
print(mydb.list_collection_names())

#Create a new collection called "customers"
mycol = mydb["comment"]
print(type(mycol))

#Check if collection exists by calling function check_CollectionExists with the following a
#Name of database as the first arguement
#Name of collection as the second arguement
#mydb as the third arguement
#In MongoDB, a collection is not created until it gets content.
check_CollectionExists("cloud_music_nosql", "comment", mydb)

'''
#Without a function the code will be as follows
collist = mydb.list_collection_names()
if "comment" in collist:
    print("The collection 'comment' exists.")
'''
```

```
[]
<class 'pymongo.collection.Collection'>
The collection  comment  does not exist in database  cloud_music_nosql
```

Out[67]:

```
'\n#Without a function the code will be as follows\ncollist = mydb.list_coll
ection_names()\nif "comment" in collist:\n    print("The collection \'commen
t\' exists.")\n'
```

In [68]:

```
mycol = mydb["comment"]
print(type(mycol))
```

```
<class 'pymongo.collection.Collection'>
```

2.5 Add documents

In [69]:

```
#Store key-values in a python dictonary object. Dictionaries (also called as associative ar
mydict = { "comment": "terrible music",
           "user_ID":"U1" }
print(type(mydict))

#Insert into collection
x = mycol.insert_one(mydict)
```

```
<class 'dict'>
```

In [70]:

```
check_DatabaseExists("cloud_music_nosql")
```

```
The database  cloud_music_nosql  exists.
```

In [72]:

```
mylist7=[

    { "_id" : "M1_C1", "comment" : "good music", "feeling" : "like", "user_ID" : "U1", "lik
    { "_id" : "M1_C2", "comment" : "nice ", "feeling" : "like", "user_ID" : "U2", "like" :
    { "_id" : "M2_C1", "comment" : "BAD music", "feeling" : "dislike", "user_ID" : "U2", "c
    { "_id" : "M2_C2", "comment" : "love it", "feeling" : "like", "user_ID" : "U1", "like"
    { "_id" : "M1_C3", "comment" : "what a good music!", "feeling" : "like", "user_ID" : "U
    { "_id" : "M1_C4", "comment" : "I have nerver heard such good music", "feeling" : "like
    { "_id" : "M2_C3", "comment" : "good", "feeling" : "like", "user_ID" : "U3", "like" : 2
    { "_id" : "M3_C2", "comment" : "such a bad song", "feeling" : "dislike", "user_ID" : "U

]

var = mycol.insert_many(mylist7)

#print list of the _id values of the inserted documents:
print(var.inserted_ids)
```

```
['M1_C1', 'M1_C2', 'M2_C1', 'M2_C2', 'M1_C3', 'M1_C4', 'M2_C3', 'M3_C2']
```

In [71]:

```
mylist3 = [
  { "_id": 'M7_C1', "comment": "good music", "feeling":"like","user_ID":"U1","like":4},
  { "_id": 'M7_C2', "comment": "such a good Chinese band", "feeling":"like","user_ID":"U2",
  { "_id": 'M5_C3', "comment": "nice ","feeling":"like", "user_ID":"U2","like":2},
  { "_id": 'M2_C5',"comment": "i love blur", "feeling":"like","user_ID":"U1","dislike":50},
  { "_id": 'M2_C9', "comment": "love it", "feeling":"like","user_ID":"U5","like":1},
  { "_id": 'M3_C5', "comment": "nice", "feeling":"like","user_ID":"U1","like":1},
  {"_id":  'M10_C2', "comment": "baddd", "feeling":"dislike","user_ID":"U3","like":1},
  { "_id": 'M4_C1', "comment": "cool", "feeling":"like","user_ID":"U1","like":4},
  { "_id": 'M18_C1', "comment": "don't like it ", "feeling":"dislike","user_ID":"U5","like"
]

var = mycol.insert_many(mylist3)

#print list of the _id values of the inserted documents:
print(var.inserted_ids)
```

```
['M7_C1', 'M7_C2', 'M5_C3', 'M2_C5', 'M2_C9', 'M3_C5', 'M10_C2', 'M4_C1', 'M
18_C1']
```

In [ ]:

In [73]:

```
#Check if Database Exists .. In MongoDB, a collection is not created until it gets content
#Note that now data has been added
check_DatabaseExists("cloud_music_nosql")

#Check if Collection Exists .. In MongoDB, a collection is not created until it gets conten
#Note that now data has been added
check_CollectionExists("cloud_music_nosql", "comment", mydb)
```

```
The database  cloud_music_nosql  exists.
The collection  comment exists in database  cloud_music_nosql
```

2.6 Query documents

In [74]:

```
#Display all documents in collection
#The find() method returns all occurrences in the selection.
#The first parameter of the find() method is a query object. In this example we use an empt

myresult = mycol.find()

#print the result:
for x in myresult:

    print(x)
```

```
{'_id': ObjectId('60e6e713e4df0d3742909fc1'), 'comment': 'terrible music',
'user_ID': 'U1'}
{'_id': 'M7_C1', 'comment': 'good music', 'feeling': 'like', 'user_ID': 'U
1', 'like': 4}
{'_id': 'M7_C2', 'comment': 'such a good Chinese band', 'feeling': 'like',
'user_ID': 'U2', 'like': 10}
{'_id': 'M5_C3', 'comment': 'nice ', 'feeling': 'like', 'user_ID': 'U2', 'li
ke': 2}
{'_id': 'M2_C5', 'comment': 'i love blur', 'feeling': 'like', 'user_ID': 'U
1', 'dislike': 50}
{'_id': 'M2_C9', 'comment': 'love it', 'feeling': 'like', 'user_ID': 'U5',
'like': 1}
{'_id': 'M3_C5', 'comment': 'nice', 'feeling': 'like', 'user_ID': 'U1', 'lik
e': 1}
{'_id': 'M10_C2', 'comment': 'baddd', 'feeling': 'dislike', 'user_ID': 'U3',
'like': 1}
{'_id': 'M4_C1', 'comment': 'cool', 'feeling': 'like', 'user_ID': 'U1', 'lik
e': 4}
{'_id': 'M18_C1', 'comment': "don't like it ", 'feeling': 'dislike', 'user_I
D': 'U5', 'like': 4}
{'_id': 'M1_C1', 'comment': 'good music', 'feeling': 'like', 'user_ID': 'U
1', 'like': 4}
{'_id': 'M1_C2', 'comment': 'nice ', 'feeling': 'like', 'user_ID': 'U2', 'li
ke': 2}
{'_id': 'M2_C1', 'comment': 'BAD music', 'feeling': 'dislike', 'user_ID': 'U
2', 'dislike': 2}
{'_id': 'M2_C2', 'comment': 'love it', 'feeling': 'like', 'user_ID': 'U1',
'like': 1}
{'_id': 'M1_C3', 'comment': 'what a good music!', 'feeling': 'like', 'user_I
D': 'U1', 'like': 2}
{'_id': 'M1_C4', 'comment': 'I have nerver heard such good music', 'feelin
g': 'like', 'user_ID': 'U5', 'like': 2}
{'_id': 'M2_C3', 'comment': 'good', 'feeling': 'like', 'user_ID': 'U3', 'lik
e': 2}
{'_id': 'M3_C2', 'comment': 'such a bad song', 'feeling': 'dislike', 'user_I
D': 'U3', 'like': 1}
```

In [75]:

```
#Python MongoDB Limit
#To limit the result in MongoDB, we use the limit() method. The limit() method takes one pa
#Limit the result to only return 5 documents:
myresult = mycol.find().limit(5)

#print the result:
for x in myresult:
    print(x)
```

```
{'_id': ObjectId('60e6e713e4df0d3742909fc1'), 'comment': 'terrible music',
'user_ID': 'U1'}
{'_id': 'M7_C1', 'comment': 'good music', 'feeling': 'like', 'user_ID': 'U
1', 'like': 4}
{'_id': 'M7_C2', 'comment': 'such a good Chinese band', 'feeling': 'like',
'user_ID': 'U2', 'like': 10}
{'_id': 'M5_C3', 'comment': 'nice ', 'feeling': 'like', 'user_ID': 'U2', 'li
ke': 2}
{'_id': 'M2_C5', 'comment': 'i love blur', 'feeling': 'like', 'user_ID': 'U
1', 'dislike': 50}
```

In [76]:

```
#The find_one() method returns the first occurrence in the selection.
x = mycol.find_one()

print(x)
```

```
{'_id': ObjectId('60e6e713e4df0d3742909fc1'), 'comment': 'terrible music',
'user_ID': 'U1'}
```

In [19]:

```
#Projection - Projection means selecting only the necessary data rather than selecting whol
#The second parameter of the find() method is an object describing which fields to include

#IMPORTANT: You are not allowed to specify both 0 and 1 values in the same object (except i
for x in mycol.find({},{ "_id":0, "comment":1, "user_ID":1,}):
    print(x)
```

```
{'comment': 'terrible music', 'user_ID': 'U1'}
{'comment': 'good music', 'user_ID': 'U1'}
{'comment': 'nice ', 'user_ID': 'U2'}
{'comment': 'BAD music', 'user_ID': 'U2'}
{'comment': 'love it', 'user_ID': 'U1'}
{'comment': 'what a good music!', 'user_ID': 'U1'}
{'comment': 'I have nerver heard such good music', 'user_ID': 'U5'}
{'comment': 'good', 'user_ID': 'U3'}
{'comment': 'such a bad song', 'user_ID': 'U3'}
{'comment': 'terrible music', 'user_ID': 'U1'}
{'comment': 'terrible music', 'user_ID': 'U1'}
{'comment': 'terrible music', 'user_ID': 'U1'}
{'comment': 'good music', 'user_ID': 'U1'}
{'comment': 'such a good Chinese band', 'user_ID': 'U2'}
{'comment': 'nice ', 'user_ID': 'U2'}
{'comment': 'i love blur', 'user_ID': 'U1'}
{'comment': 'love it', 'user_ID': 'U5'}
{'comment': 'nice', 'user_ID': 'U1'}
{'comment': 'baddd', 'user_ID': 'U3'}
{'comment': 'cool', 'user_ID': 'U1'}
{'comment': "don't like it ", 'user_ID': 'U5'}
{'comment': 'terrible music', 'user_ID': 'U1'}
```

Adding Embedded Documents

In [77]:

```
#Create a new collection called "products"
mycolED = mydb["products"]

mylist5 = [
   {"album": "blur01", "instock": [ { "warehouse": "A", "qty": 50 }, { "warehouse": "C", "c
   {"album": "blur02", "instock": [ { "warehouse": "C", "qty": 50 } ,{"user":"U1","qty":1}]
   {"album": "oasis01", "instock": [ { "warehouse": "A", "qty": 60 }, { "warehouse": "B", "
   {"album": "suede01", "instock": [ { "warehouse": "A", "qty": 40 }, { "warehouse": "B", "
   {"album": "clavin harris01", "instock": [ { "warehouse": "B", "qty": 15 }, { "warehouse"
]

var = mycolED.insert_many(mylist5)

#print list of the _id values of the inserted documents:
print(var.inserted_ids)
```

```
[ObjectId('60e6e770e4df0d3742909fc2'), ObjectId('60e6e770e4df0d3742909fc3'),
ObjectId('60e6e770e4df0d3742909fc4'), ObjectId('60e6e770e4df0d3742909fc5'),
ObjectId('60e6e770e4df0d3742909fc6')]
```

In [21]:

```
myresult = mycolED.find()

#print the result:
for x in myresult:
    print(x)
```

```
{'_id': ObjectId('60e592fdfff9457aad09c828'), 'album': 'blur01', 'instock':
[{'warehouse': 'A', 'qty': 50}, {'warehouse': 'C', 'qty': 15}]}
{'_id': ObjectId('60e592fdfff9457aad09c829'), 'album': 'blur02', 'instock':
[{'warehouse': 'C', 'qty': 50}, {'user': 'U1', 'qty': 1}]}
{'_id': ObjectId('60e592fdfff9457aad09c82a'), 'album': 'oasis01', 'instock':
[{'warehouse': 'A', 'qty': 60}, {'warehouse': 'B', 'qty': 15}]}
{'_id': ObjectId('60e592fdfff9457aad09c82b'), 'album': 'suede01', 'instock':
[{'warehouse': 'A', 'qty': 40}, {'warehouse': 'B', 'qty': 5}]}
{'_id': ObjectId('60e592fdfff9457aad09c82c'), 'album': 'clavin harris01', 'i
nstock': [{'warehouse': 'B', 'qty': 15}, {'warehouse': 'C', 'qty': 35}]}
{'_id': ObjectId('60e5a464d83d9e31d46d56df'), 'album': 'blur01', 'instock':
[{'warehouse': 'A', 'qty': 50}, {'warehouse': 'C', 'qty': 15}]}
{'_id': ObjectId('60e5a464d83d9e31d46d56e0'), 'album': 'blur02', 'instock':
[{'warehouse': 'C', 'qty': 50}, {'user': 'U1', 'qty': 1}]}
{'_id': ObjectId('60e5a464d83d9e31d46d56e1'), 'album': 'oasis01', 'instock':
[{'warehouse': 'A', 'qty': 60}, {'warehouse': 'B', 'qty': 15}]}
{'_id': ObjectId('60e5a464d83d9e31d46d56e2'), 'album': 'suede01', 'instock':
[{'warehouse': 'A', 'qty': 40}, {'warehouse': 'B', 'qty': 5}]}
{'_id': ObjectId('60e5a464d83d9e31d46d56e3'), 'album': 'clavin harris01', 'i
nstock': [{'warehouse': 'B', 'qty': 15}, {'warehouse': 'C', 'qty': 35}]}
{'_id': ObjectId('60e5bb0f1fa9043903e4c1a6'), 'album': 'blur01', 'instock':
[{'warehouse': 'A', 'qty': 50}, {'warehouse': 'C', 'qty': 15}]}
{'_id': ObjectId('60e5bb0f1fa9043903e4c1a7'), 'album': 'blur02', 'instock':
[{'warehouse': 'C', 'qty': 50}, {'user': 'U1', 'qty': 1}]}
{'_id': ObjectId('60e5bb0f1fa9043903e4c1a8'), 'album': 'oasis01', 'instock':
[{'warehouse': 'A', 'qty': 60}, {'warehouse': 'B', 'qty': 15}]}
{'_id': ObjectId('60e5bb0f1fa9043903e4c1a9'), 'album': 'suede01', 'instock':
[{'warehouse': 'A', 'qty': 40}, {'warehouse': 'B', 'qty': 5}]}
{'_id': ObjectId('60e5bb0f1fa9043903e4c1aa'), 'album': 'clavin harris01', 'i
nstock': [{'warehouse': 'B', 'qty': 15}, {'warehouse': 'C', 'qty': 35}]}
{'_id': ObjectId('60e5bee419735f8890269e25'), 'album': 'blur01', 'instock':
[{'warehouse': 'A', 'qty': 50}, {'warehouse': 'C', 'qty': 15}]}
{'_id': ObjectId('60e5bee419735f8890269e26'), 'album': 'blur02', 'instock':
[{'warehouse': 'C', 'qty': 50}, {'user': 'U1', 'qty': 1}]}
{'_id': ObjectId('60e5bee419735f8890269e27'), 'album': 'oasis01', 'instock':
[{'warehouse': 'A', 'qty': 60}, {'warehouse': 'B', 'qty': 15}]}
{'_id': ObjectId('60e5bee419735f8890269e28'), 'album': 'suede01', 'instock':
[{'warehouse': 'A', 'qty': 40}, {'warehouse': 'B', 'qty': 5}]}
{'_id': ObjectId('60e5bee419735f8890269e29'), 'album': 'clavin harris01', 'i
nstock': [{'warehouse': 'B', 'qty': 15}, {'warehouse': 'C', 'qty': 35}]}
```

In [78]:

```
#Filter With Regular Expressions
#To find only the documents where the "coomment" field starts with the letter "bad", use the

myquery = { "comment": { "$regex": 'good'} }
mydoc = mycol.find(myquery)

for x in mydoc:
    print(x)
```

```
{'_id': 'M7_C1', 'comment': 'good music', 'feeling': 'like', 'user_ID': 'U
1', 'like': 4}
{'_id': 'M7_C2', 'comment': 'such a good Chinese band', 'feeling': 'like',
'user_ID': 'U2', 'like': 10}
{'_id': 'M1_C1', 'comment': 'good music', 'feeling': 'like', 'user_ID': 'U
1', 'like': 4}
{'_id': 'M1_C3', 'comment': 'what a good music!', 'feeling': 'like', 'user_I
D': 'U1', 'like': 2}
{'_id': 'M1_C4', 'comment': 'I have nerver heard such good music', 'feelin
g': 'like', 'user_ID': 'U5', 'like': 2}
{'_id': 'M2_C3', 'comment': 'good', 'feeling': 'like', 'user_ID': 'U3', 'lik
e': 2}
```

In [79]:

```
#Filter With Regular Expressions
#To find only the documents where the "coomment" field starts with the letter "bad", use the

myquery = { "comment": { "$regex": 'love'} }
mydoc = mycol.find(myquery)

for x in mydoc:
    print(x)
```

```
{'_id': 'M2_C5', 'comment': 'i love blur', 'feeling': 'like', 'user_ID': 'U
1', 'dislike': 50}
{'_id': 'M2_C9', 'comment': 'love it', 'feeling': 'like', 'user_ID': 'U5',
'like': 1}
{'_id': 'M2_C2', 'comment': 'love it', 'feeling': 'like', 'user_ID': 'U1',
'like': 1}
```

In [80]:

```
#Filter With Regular Expressions
#To find only the documents where the "coomment" field starts with the letter "bad", use the

myquery = { "comment": { "$regex": 'nice'} }
mydoc = mycol.find(myquery)

for x in mydoc:
    print(x)
```

```
{'_id': 'M5_C3', 'comment': 'nice ', 'feeling': 'like', 'user_ID': 'U2', 'li
ke': 2}
{'_id': 'M3_C5', 'comment': 'nice', 'feeling': 'like', 'user_ID': 'U1', 'lik
e': 1}
{'_id': 'M1_C2', 'comment': 'nice ', 'feeling': 'like', 'user_ID': 'U2', 'li
ke': 2}
```

In [81]:

```
#Filter all value of comments which include 'bad'.
myquery = { "comment": {"$regex":"bad"} }
mydoc = mycol.find(myquery)

for x in mydoc:
    print(x)
```

```
{'_id': 'M10_C2', 'comment': 'baddd', 'feeling': 'dislike', 'user_ID': 'U3',
'like': 1}
{'_id': 'M3_C2', 'comment': 'such a bad song', 'feeling': 'dislike', 'user_I
D': 'U3', 'like': 1}
```

In [82]:

```
myquery = { "comment": {"$regex":"BAD"} }
mydoc = mycol.find(myquery)

for x in mydoc:
    print(x)
```

```
{'_id': 'M2_C1', 'comment': 'BAD music', 'feeling': 'dislike', 'user_ID': 'U
2', 'dislike': 2}
```

In [83]:

```
#Python MongoDB Sort - Use the sort() method to sort the result in ascending or descending
#sort("name", 1) #ascending
#sort("name", -1) #descending

mydoc = mycol.find().sort("like", -1)

for x in mydoc:
     print(x)
```

```
{'_id': 'M7_C2', 'comment': 'such a good Chinese band', 'feeling': 'like',
'user_ID': 'U2', 'like': 10}
{'_id': 'M7_C1', 'comment': 'good music', 'feeling': 'like', 'user_ID': 'U
1', 'like': 4}
{'_id': 'M4_C1', 'comment': 'cool', 'feeling': 'like', 'user_ID': 'U1', 'lik
e': 4}
{'_id': 'M18_C1', 'comment': "don't like it ", 'feeling': 'dislike', 'user_I
D': 'U5', 'like': 4}
{'_id': 'M1_C1', 'comment': 'good music', 'feeling': 'like', 'user_ID': 'U
1', 'like': 4}
{'_id': 'M5_C3', 'comment': 'nice ', 'feeling': 'like', 'user_ID': 'U2', 'li
ke': 2}
{'_id': 'M1_C2', 'comment': 'nice ', 'feeling': 'like', 'user_ID': 'U2', 'li
ke': 2}
{'_id': 'M1_C3', 'comment': 'what a good music!', 'feeling': 'like', 'user_I
D': 'U1', 'like': 2}
{'_id': 'M1_C4', 'comment': 'I have nerver heard such good music', 'feelin
g': 'like', 'user_ID': 'U5', 'like': 2}
{'_id': 'M2_C3', 'comment': 'good', 'feeling': 'like', 'user_ID': 'U3', 'lik
e': 2}
{'_id': 'M2_C9', 'comment': 'love it', 'feeling': 'like', 'user_ID': 'U5',
'like': 1}
{'_id': 'M3_C5', 'comment': 'nice', 'feeling': 'like', 'user_ID': 'U1', 'lik
e': 1}
{'_id': 'M10_C2', 'comment': 'baddd', 'feeling': 'dislike', 'user_ID': 'U3',
'like': 1}
{'_id': 'M2_C2', 'comment': 'love it', 'feeling': 'like', 'user_ID': 'U1',
'like': 1}
{'_id': 'M3_C2', 'comment': 'such a bad song', 'feeling': 'dislike', 'user_I
D': 'U3', 'like': 1}
{'_id': ObjectId('60e6e713e4df0d3742909fc1'), 'comment': 'terrible music',
'user_ID': 'U1'}
{'_id': 'M2_C5', 'comment': 'i love blur', 'feeling': 'like', 'user_ID': 'U
1', 'dislike': 50}
{'_id': 'M2_C1', 'comment': 'BAD music', 'feeling': 'dislike', 'user_ID': 'U
2', 'dislike': 2}
```

2.7 Updating documents

In [84]:

```
#Update documents
# You can update a record, or document as it is called in MongoDB, by using the update_one(

myquery = { "_id": "M3_C1" }
newvalues = { "$set": { "comment": "very good music" } }

mycol.update_one(myquery, newvalues)

#print documents in the collection after the update:
for x in mycol.find():
    print(x)
```

```
{'_id': ObjectId('60e6e713e4df0d3742909fc1'), 'comment': 'terrible music',
'user_ID': 'U1'}
{'_id': 'M7_C1', 'comment': 'good music', 'feeling': 'like', 'user_ID': 'U
1', 'like': 4}
{'_id': 'M7_C2', 'comment': 'such a good Chinese band', 'feeling': 'like',
'user_ID': 'U2', 'like': 10}
{'_id': 'M5_C3', 'comment': 'nice ', 'feeling': 'like', 'user_ID': 'U2', 'li
ke': 2}
{'_id': 'M2_C5', 'comment': 'i love blur', 'feeling': 'like', 'user_ID': 'U
1', 'dislike': 50}
{'_id': 'M2_C9', 'comment': 'love it', 'feeling': 'like', 'user_ID': 'U5',
'like': 1}
{'_id': 'M3_C5', 'comment': 'nice', 'feeling': 'like', 'user_ID': 'U1', 'lik
e': 1}
{'_id': 'M10_C2', 'comment': 'baddd', 'feeling': 'dislike', 'user_ID': 'U3',
'like': 1}
{'_id': 'M4_C1', 'comment': 'cool', 'feeling': 'like', 'user_ID': 'U1', 'lik
e': 4}
{'_id': 'M18_C1', 'comment': "don't like it ", 'feeling': 'dislike', 'user_I
D': 'U5', 'like': 4}
{'_id': 'M1_C1', 'comment': 'good music', 'feeling': 'like', 'user_ID': 'U
1', 'like': 4}
{'_id': 'M1_C2', 'comment': 'nice ', 'feeling': 'like', 'user_ID': 'U2', 'li
ke': 2}
{'_id': 'M2_C1', 'comment': 'BAD music', 'feeling': 'dislike', 'user_ID': 'U
2', 'dislike': 2}
{'_id': 'M2_C2', 'comment': 'love it', 'feeling': 'like', 'user_ID': 'U1',
'like': 1}
{'_id': 'M1_C3', 'comment': 'what a good music!', 'feeling': 'like', 'user_I
D': 'U1', 'like': 2}
{'_id': 'M1_C4', 'comment': 'I have nerver heard such good music', 'feelin
g': 'like', 'user_ID': 'U5', 'like': 2}
{'_id': 'M2_C3', 'comment': 'good', 'feeling': 'like', 'user_ID': 'U3', 'lik
e': 2}
{'_id': 'M3_C2', 'comment': 'such a bad song', 'feeling': 'dislike', 'user_I
D': 'U3', 'like': 1}
```

In [ ]:

2.8 Deleting documents

In [68]:

```
#Deleting one at a time...
myquery = { "_id": 'M3_C14' }
mycol.delete_one(myquery)

#...and querying the documents (one reduced after each command)
mydoc = mycol.find(myquery)

for x in mydoc:
    print(x)
```

In [69]:

```
#Delete multiple documents
myquery = { "_id": {"$regex": "^M4"} }

x = mycol.delete_many(myquery)

print(x.deleted_count, " documents deleted.")
```

```
0  documents deleted.
```

In [70]:

```
#Delete all documents in a collection
#To delete all documents in a collection, pass an empty query object to the delete_many() m

x = mycol.delete_many({})

print(x.deleted_count, " documents deleted.")
```

```
9  documents deleted.
```

In [74]:

```
#Python MongoDB Drop Collection - You can delete a collection by using the drop() method. A
mycol.drop()
#Check if the collection exists by calling the user function defined earlier
check_CollectionExists("Youtube_music_nosql", "comment", mydb)
```

```
The collection  comment  does not exist in database  Youtube_music_nosql
```

In [75]:

```
#Drop database using instance of MongoClient
mongoclient.drop_database("Youtube_music_nosql")
#Check to see if database exists by calling the user function defined earlier
check_DatabaseExists("Youtube_music_nosql")
```

```
The database  Youtube_music_nosql  does not exist.
```

In [ ]:

```
```

Application

In [78]:

```
import matplotlib.pyplot as plt
```

In [85]:

```
conn=sqlite3.connect('S:\\DB\\cloud_music.db')
cur=conn.cursor()
```

In [86]:

```
mongoclient = pymongo.MongoClient("mongodb://localhost:7901/")

#Check what databases exist - the output is a list of database names
print(mongoclient.list_database_names())

#You can also check databases that presently exisit using a loop
dblist = mongoclient.list_database_names()
for x in dblist:
    print(x)
```

```
['Youtube_music_nosql', 'Youtube_nosql', 'admin', 'cloud_music_nosql', 'conf
ig', 'local', 'test']
Youtube_music_nosql
Youtube_nosql
admin
cloud_music_nosql
config
local
test
```

In [87]:

```
mydb = mongoclient["cloud_music_nosql"]
print(type(mydb))
check_DatabaseExists("cloud_music_nosql")
```

```
<class 'pymongo.database.Database'>
The database  cloud_music_nosql  exists.
```

In [88]:

```
query="SELECT singer_Name from singer"
```

In [89]:

```
#use pandas to build dataframe
df = pd.read_sql_query(query,conn)

df
```

Out[89]:

	singer_name
0	silly and idiot
1	blur
2	Calvin Harris
3	Rich Brian
4	Oasis
5	Suede
6	Nada Surf

In [ ]:

```

```

In [90]:

```
left_join="select * from music left join singer on music.music_singer=singer.singer_ID wher
```

In [91]:

```
#use pandas to build dataframe
df = pd.read_sql_query(left_join,conn)

df
```

Out[91]:

	music_ID	music_name	music_playtime	music_singer	music_comment	music_genre_ID	sin
0	M1	HoydeA	25686	S1	M1_C1	G1	
1	M5	Dont look back at anger	98642	S5	M5_C1	G2	
2	M4	Love in my pocket	297456	S4	M4_C1	G4	
3	M7	Beautiful Ones	439764	S6	S6_C1	G2	
4	M2	Charmless Man	719430	S2	M2_C1	G2	

In [ ]:

```

```

In [92]:

```
# pandas also has the groupy function.
playtime=df[['singer_ID','singer_name','music_name','music_playtime']]
playtime_table=playtime.groupby(['music_name']).sum()
playtime
```

Out[92]:

	singer_ID	singer_name	music_name	music_playtime
0	S1	silly and idiot	HoydeA	25686
1	S5	Oasis	Dont look back at anger	98642
2	S4	Rich Brian	Love in my pocket	297456
3	S6	Suede	Beautiful Ones	439764
4	S2	blur	Charmless Man	719430

In [95]:

```
import matplotlib.pyplot as plt
```

In [96]:

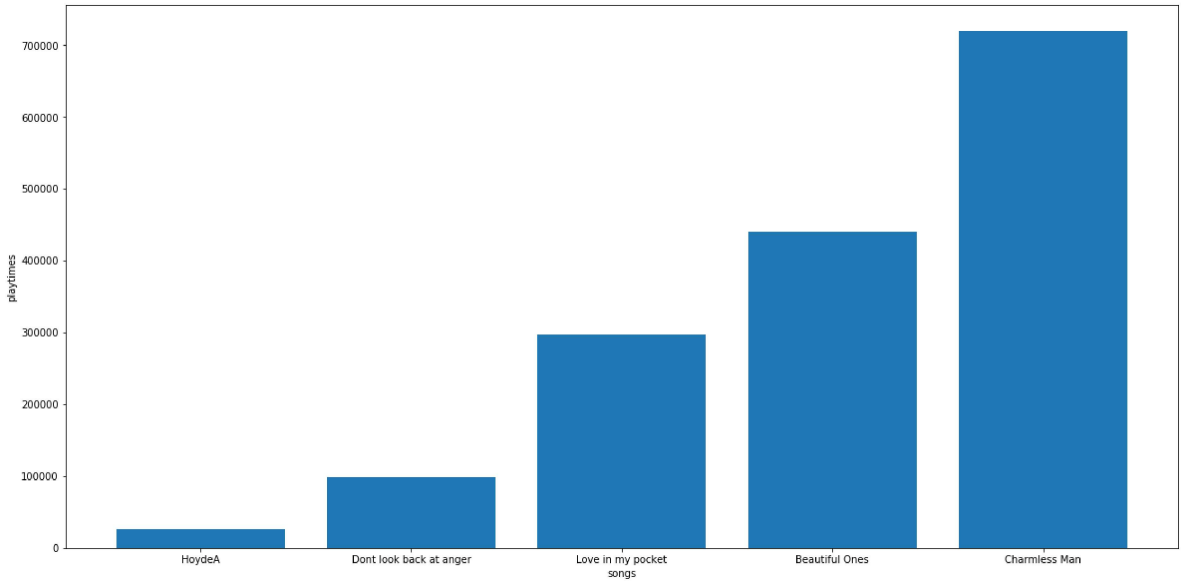
```
data=playtime['music_playtime']
label=playtime['music_name']
```

In [97]:

```
plt.figure(figsize=(20,10))
plt.xlabel('songs')
plt.ylabel('playtimes')
plt.bar(label,data)
```

Out[97]:

```
<BarContainer object of 5 artists>
```



In [110]:

```
! pip install wordcloud
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

```
Requirement already satisfied: wordcloud in e:\anaconda\lib\site-packages
(1.8.1)
Requirement already satisfied: matplotlib in e:\anaconda\lib\site-packages
(from wordcloud) (3.0.3)
Requirement already satisfied: pillow in e:\anaconda\lib\site-packages (from
wordcloud) (5.4.1)
Requirement already satisfied: numpy>=1.6.1 in e:\anaconda\lib\site-packages
(from wordcloud) (1.16.2)
Requirement already satisfied: cycler>=0.10 in e:\anaconda\lib\site-packages
(from matplotlib->wordcloud) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in e:\anaconda\lib\site-pac
kages (from matplotlib->wordcloud) (1.0.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
e:\anaconda\lib\site-packages (from matplotlib->wordcloud) (2.3.1)
Requirement already satisfied: python-dateutil>=2.1 in e:\anaconda\lib\site-
packages (from matplotlib->wordcloud) (2.8.0)
Requirement already satisfied: six in e:\anaconda\lib\site-packages (from cy
cler>=0.10->matplotlib->wordcloud) (1.12.0)
Requirement already satisfied: setuptools in e:\anaconda\lib\site-packages
(from kiwisolver>=1.0.1->matplotlib->wordcloud) (40.8.0)
```

In [98]:

```
list=[]
#Display all documents in collection
#The find() method returns all occurrences in the selection.
#The first parameter of the find() method is a query object. In this example we use an empt

myresult = mycol.find()

#print the result:
for x in myresult:
    list.append(x)

    print(x)
list
```

```
{'_id': ObjectId('60e6e713e4df0d3742909fc1'), 'comment': 'terrible music',
'user_ID': 'U1'}
{'_id': 'M7_C1', 'comment': 'good music', 'feeling': 'like', 'user_ID': 'U
1', 'like': 4}
{'_id': 'M7_C2', 'comment': 'such a good Chinese band', 'feeling': 'like',
'user_ID': 'U2', 'like': 10}
{'_id': 'M5_C3', 'comment': 'nice ', 'feeling': 'like', 'user_ID': 'U2', 'li
ke': 2}
{'_id': 'M2_C5', 'comment': 'i love blur', 'feeling': 'like', 'user_ID': 'U
1', 'dislike': 50}
{'_id': 'M2_C9', 'comment': 'love it', 'feeling': 'like', 'user_ID': 'U5',
'like': 1}
{'_id': 'M3_C5', 'comment': 'nice', 'feeling': 'like', 'user_ID': 'U1', 'lik
e': 1}
{'_id': 'M10_C2', 'comment': 'baddd', 'feeling': 'dislike', 'user_ID': 'U3',
'like': 1}
{'_id': 'M4_C1', 'comment': 'cool', 'feeling': 'like', 'user_ID': 'U1', 'lik
e': 4}
{'_id': 'M18_C1', 'comment': "don't like it ", 'feeling': 'dislike', 'user_I
D': 'U5', 'like': 4}
{'_id': 'M1_C1', 'comment': 'good music', 'feeling': 'like', 'user_ID': 'U
1', 'like': 4}
{'_id': 'M1_C2', 'comment': 'nice ', 'feeling': 'like', 'user_ID': 'U2', 'li
ke': 2}
{'_id': 'M2_C1', 'comment': 'BAD music', 'feeling': 'dislike', 'user_ID': 'U
2', 'dislike': 2}
{'_id': 'M2_C2', 'comment': 'love it', 'feeling': 'like', 'user_ID': 'U1',
'like': 1}
{'_id': 'M1_C3', 'comment': 'what a good music!', 'feeling': 'like', 'user_I
D': 'U1', 'like': 2}
{'_id': 'M1_C4', 'comment': 'I have nerver heard such good music', 'feelin
g': 'like', 'user_ID': 'U5', 'like': 2}
{'_id': 'M2_C3', 'comment': 'good', 'feeling': 'like', 'user_ID': 'U3', 'lik
e': 2}
{'_id': 'M3_C2', 'comment': 'such a bad song', 'feeling': 'dislike', 'user_I
D': 'U3', 'like': 1}
```

Out[98]:

```
[{'_id': ObjectId('60e6e713e4df0d3742909fc1'),
  'comment': 'terrible music',
  'user_ID': 'U1'},
 {'_id': 'M7_C1',
```

```
 'comment': 'good music',
 'feeling': 'like',
 'user_ID': 'U1',
 'like': 4},
{'_id': 'M7_C2',
 'comment': 'such a good Chinese band',
 'feeling': 'like',
 'user_ID': 'U2',
 'like': 10},
{'_id': 'M5_C3',
 'comment': 'nice ',
 'feeling': 'like',
 'user_ID': 'U2',
 'like': 2},
{'_id': 'M2_C5',
 'comment': 'i love blur',
 'feeling': 'like',
 'user_ID': 'U1',
 'dislike': 50},
{'_id': 'M2_C9',
 'comment': 'love it',
 'feeling': 'like',
 'user_ID': 'U5',
 'like': 1},
{'_id': 'M3_C5',
 'comment': 'nice',
 'feeling': 'like',
 'user_ID': 'U1',
 'like': 1},
{'_id': 'M10_C2',
 'comment': 'baddd',
 'feeling': 'dislike',
 'user_ID': 'U3',
 'like': 1},
{'_id': 'M4_C1',
 'comment': 'cool',
 'feeling': 'like',
 'user_ID': 'U1',
 'like': 4},
{'_id': 'M18_C1',
 'comment': "don't like it ",
 'feeling': 'dislike',
 'user_ID': 'U5',
 'like': 4},
{'_id': 'M1_C1',
 'comment': 'good music',
 'feeling': 'like',
 'user_ID': 'U1',
 'like': 4},
{'_id': 'M1_C2',
 'comment': 'nice ',
 'feeling': 'like',
 'user_ID': 'U2',
 'like': 2},
{'_id': 'M2_C1',
 'comment': 'BAD music',
 'feeling': 'dislike',
 'user_ID': 'U2',
 'dislike': 2},
{'_id': 'M2_C2',
 'comment': 'love it',
```

```
    'feeling': 'like',
    'user_ID': 'U1',
    'like': 1},
   {'_id': 'M1_C3',
    'comment': 'what a good music!',
    'feeling': 'like',
    'user_ID': 'U1',
    'like': 2},
   {'_id': 'M1_C4',
    'comment': 'I have nerver heard such good music',
    'feeling': 'like',
    'user_ID': 'U5',
    'like': 2},
   {'_id': 'M2_C3',
    'comment': 'good',
    'feeling': 'like',
    'user_ID': 'U3',
    'like': 2},
   {'_id': 'M3_C2',
    'comment': 'such a bad song',
    'feeling': 'dislike',
    'user_ID': 'U3',
    'like': 1}]
```

In [99]:

```
df2=pd.DataFrame(list)
df2
```

Out[99]:

	_id	comment	user_ID	feeling	like	dislike
0	60e6e713e4df0d3742909fc1	terrible music	U1	NaN	NaN	NaN
1	M7_C1	good music	U1	like	4.0	NaN
2	M7_C2	such a good Chinese band	U2	like	10.0	NaN
3	M5_C3	nice	U2	like	2.0	NaN
4	M2_C5	i love blur	U1	like	NaN	50.0
5	M2_C9	love it	U5	like	1.0	NaN
6	M3_C5	nice	U1	like	1.0	NaN
7	M10_C2	baddd	U3	dislike	1.0	NaN
8	M4_C1	cool	U1	like	4.0	NaN
9	M18_C1	don't like it	U5	dislike	4.0	NaN
10	M1_C1	good music	U1	like	4.0	NaN
11	M1_C2	nice	U2	like	2.0	NaN
12	M2_C1	BAD music	U2	dislike	NaN	2.0
13	M2_C2	love it	U1	like	1.0	NaN
14	M1_C3	what a good music!	U1	like	2.0	NaN
15	M1_C4	I have nerver heard such good music	U5	like	2.0	NaN
16	M2_C3	good	U3	like	2.0	NaN
17	M3_C2	such a bad song	U3	dislike	1.0	NaN

In [104]:

```
myquery = { "comment": { "$regex": 'like'} }
cursor = mydb['comment']
cursor
```

Out[104]:

```
Collection(Database(MongoClient(host=['localhost:7901'], document_class=dic
t, tz_aware=False, connect=True), 'cloud_music_nosql'), 'comment')
```

In [111]:

```
user_comment=df2[['_id','user_ID','comment']]
user_comment
```

Out[111]:

	_id	user_ID	comment
0	60e6e713e4df0d3742909fc1	U1	terrible music
1	M7_C1	U1	good music
2	M7_C2	U2	such a good Chinese band
3	M5_C3	U2	nice
4	M2_C5	U1	i love blur
5	M2_C9	U5	love it
6	M3_C5	U1	nice
7	M10_C2	U3	baddd
8	M4_C1	U1	cool
9	M18_C1	U5	don't like it
10	M1_C1	U1	good music
11	M1_C2	U2	nice
12	M2_C1	U2	BAD music
13	M2_C2	U1	love it
14	M1_C3	U1	what a good music!
15	M1_C4	U5	I have nerver heard such good music
16	M2_C3	U3	good
17	M3_C2	U3	such a bad song

In [114]:

```
user_comment.groupby('comment')['user_ID'].sum()
```

Out[114]:

```
comment
BAD music                                  U2
I have nerver heard such good music      U5
baddd                                      U3
cool                                       U1
don't like it                              U5
good                                       U3
good music                               U1U1
i love blur                                U1
love it                                  U5U1
nice                                       U1
nice                                     U2U2
such a bad song                            U3
such a good Chinese band                   U2
terrible music                             U1
what a good music!                         U1
Name: user_ID, dtype: object
```

In [125]:

```
!pip install wordcloud
```

```
Collecting wordcloud
  Using cached https://files.pythonhosted.org/packages/a7/f0/f7384c323c1fc71
49573455f9633ef063c7b4d85c64d419b711bbca9ed29/wordcloud-1.8.1-cp37-cp37m-win
_amd64.whl (https://files.pythonhosted.org/packages/a7/f0/f7384c323c1fc71495
73455f9633ef063c7b4d85c64d419b711bbca9ed29/wordcloud-1.8.1-cp37-cp37m-win_am
d64.whl)
Requirement already satisfied: pillow in e:\anaconda\lib\site-packages (from
wordcloud) (5.4.1)
Requirement already satisfied: matplotlib in e:\anaconda\lib\site-packages
 (from wordcloud) (3.0.3)
Requirement already satisfied: numpy>=1.6.1 in e:\anaconda\lib\site-packages
(from wordcloud) (1.16.2)
Requirement already satisfied: cycler>=0.10 in e:\anaconda\lib\site-packages
(from matplotlib->wordcloud) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in e:\anaconda\lib\site-pac
kages (from matplotlib->wordcloud) (1.0.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
 e:\anaconda\lib\site-packages (from matplotlib->wordcloud) (2.3.1)
Requirement already satisfied: python-dateutil>=2.1 in e:\anaconda\lib\site-
packages (from matplotlib->wordcloud) (2.8.0)
Requirement already satisfied: six in e:\anaconda\lib\site-packages (from cy
cler>=0.10->matplotlib->wordcloud) (1.12.0)
Requirement already satisfied: setuptools in e:\anaconda\lib\site-packages
 (from kiwisolver>=1.0.1->matplotlib->wordcloud) (40.8.0)
Installing collected packages: wordcloud
Successfully installed wordcloud-1.8.1
```

In [129]:

```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

use two method to generate wordcloud

In [123]:

```
user_comment['comment'].values
```

Out[123]:

```
array(['terrible music', 'good music', 'such a good Chinese band',
       'nice ', 'i love blur', 'love it', 'nice', 'baddd', 'cool',
       "don't like it ", 'good music', 'nice ', 'BAD music', 'love it',
       'what a good music!', 'I have nerver heard such good music',
       'good', 'such a bad song'], dtype=object)
```

In [128]:

```
user_comment['comment'].values.tolist()
```

Out[128]:

```
['terrible music',
 'good music',
 'such a good Chinese band',
 'nice ',
 'i love blur',
 'love it',
 'nice',
 'baddd',
 'cool',
 "don't like it ",
 'good music',
 'nice ',
 'BAD music',
 'love it',
 'what a good music!',
 'I have nerver heard such good music',
 'good',
 'such a bad song']
```

In [143]:

```
list=user_comment['comment'].values.tolist() #make array into list
```

In [157]:

```
#list convert into string
string=""
for i in list:
    string +=i
print(string)
```

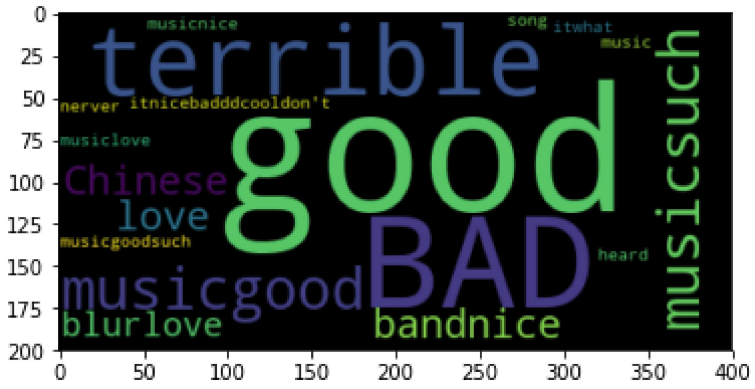
```
terrible musicgood musicsuch a good Chinese bandnice i love blurlove itniceb
adddcooldon't like it good musicnice BAD musiclove itwhat a good music!I hav
e nerver heard such good musicgoodsuch a bad song
```

In [158]:

```
wordcloud = WordCloud().generate(string)
plt.imshow(wordcloud, interpolation='bilinear')
```

Out[158]:

```
<matplotlib.image.AxesImage at 0x1cd6e811f98>
```



In [159]:

```
#another way to convert list into string
```

In [160]:

```
text=user_comment['comment']
text='.'.join(text)
type(text)
```

Out[160]:

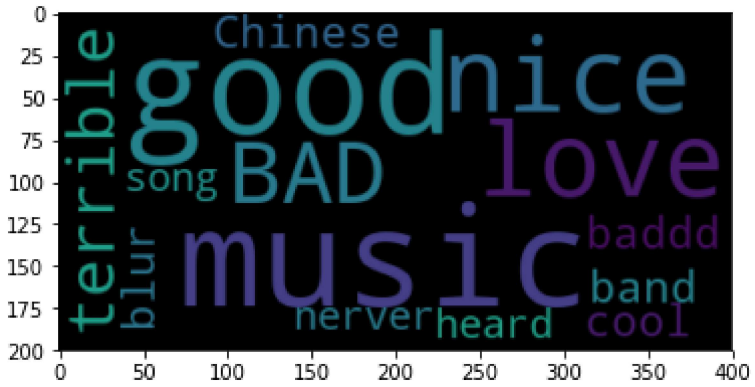
```
str
```

In [162]:

```
wordcloud = WordCloud().generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
```

Out[162]:

```
<matplotlib.image.AxesImage at 0x1cd6e8e44a8>
```



In [ ]:

In [ ]:

In [ ]: