**Final Project Description and Requirements**

**MSIT 630 Database Systems (Summer 2019)**

**Total: 36 points**

**Due date: 7/28/2019 11:59PM**

**System description**

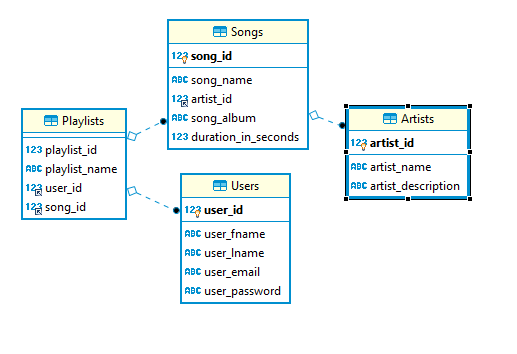
**Project Report**

**Description.**

This is a database for a music application. This application will hold users, songs, artist, and playlists.Users can add songs to their playlists. The database language is MySQL hosted from AWS.

**Design and Implementation**

**E-R Diagram.**



**Users**

The **primary key** “user\_id” will be an auto incrementing integer with a not null constraint.

Users will have a “user\_fname” which will be a varchar max of 20 characters and a not null constraint.

Users will have a “user\_lname” which will be a varchar max of 20 characters and a not null constraint.

Users will have a “email” which will be a varchar max of 20 characters and a not null constraint.

Users will have a “password” which will be a varchar max of 20 characters and a not null constraint.

One user can have many playlists. This is a one to many relationship.

This table is in 3rd normal form.

**create** **table** **if** **not** **EXISTS** Users(

user\_id **INT** **NOT** **NULL** **AUTO\_INCREMENT**,

user\_fname **VARCHAR**(20) **NOT** **NULL**,

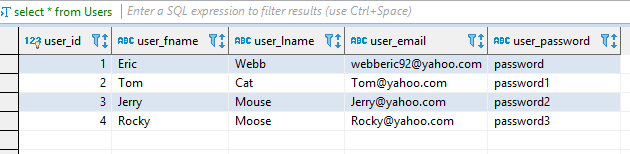
user\_lname **VARCHAR**(20) **NOT** **NULL**,

user\_email **VARCHAR**(20) **NOT** **NULL**,

user\_password **VARCHAR**(20) **NOT** **NULL**,

**PRIMARY** **KEY** ( user\_id )

)ENGINE INNODB;



**Artists**

The **primary key** “artist\_id” will be an auto incrementing integer with a not null constraint.

Artists will have a “name” which will be a varchar max of 20 characters and a not null constraint.

Artists will have a “description” which will be a varchar max of 60 characters and a not null constraint.

This table is in 3rd normal form.

**create** **table** **if** **not** **EXISTS** Artists(

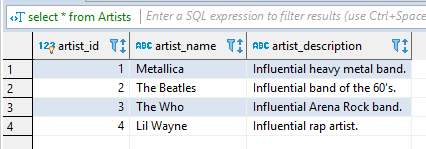
artist\_id **INT** **NOT** **NULL** **AUTO\_INCREMENT**,

artist\_name **VARCHAR**(20) **NOT** **NULL**,

artist\_description **VARCHAR**(60) **NOT** **NULL**,

**PRIMARY** **KEY** ( artist\_id )

)ENGINE INNODB;



**Songs**

The **primary key** “song\_id” will be an auto incrementing integer with a not null constraint.

Songs will have a “song\_name” which will be a varchar max of 50 characters and a not null constraint.

Songs will have a **foreign key** “artist\_id” which will be an integer with a not null constraint.

Songs will have a “song\_album” which will be a varchar max of 50 characters and a not null constraint.

Songs will have a “duration\_in\_seconds” which will be an integer with a not null constraint.

One song can have one Artist, this is a one-to-one relationship.

This table is in third normal form.

**create** **table** **if** **not** **EXISTS** Songs(

song\_id **INT** **NOT** **NULL** **AUTO\_INCREMENT**,

song\_name **VARCHAR**(50) **NOT** **NULL**,

artist\_id **INT** **NOT** **NULL**,

song\_album **VARCHAR**(50) **NOT** **NULL**,

duration\_in\_seconds **INT** **NOT** **NULL**,

**PRIMARY** **KEY** ( song\_id ),

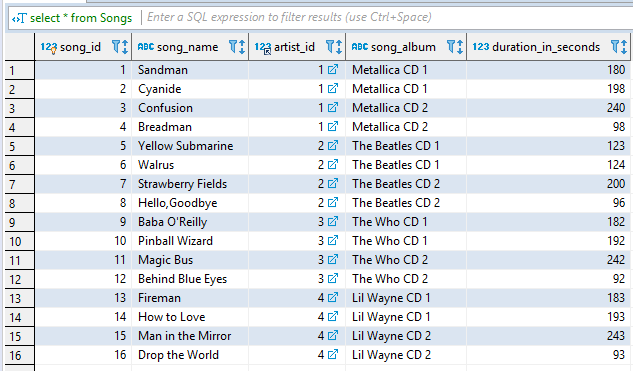
**FOREIGN** **KEY** artist\_id(artist\_id)

**REFERENCES** Artists(artist\_id)

**ON** **UPDATE** **CASCADE**

**ON** **DELETE** **RESTRICT**

)ENGINE INNODB;



**Playlists**

The **primary key** “playlist\_id” will be an auto incrementing integer with a not null constraint.

Playlists will have a “playlist\_ name” which will be a varchar max of 50 characters and a not null constraint.

Playlists will have a **foreign key** “user\_id” which will be an integer with a not null constraint.

Playlists will have a **foreign key** “song\_id” which will be an integer with a not null constraint.

Many playlists can have many songs. This is a one to many relationship.

This table is in 3rd normal form.

**create** **table** **if** **not** **EXISTS** Playlists(

playlist\_id **INT** **NOT** **NULL**,

playlist\_name **VARCHAR**(50) **NOT** **NULL**,

user\_id **int** **NOT** **NULL**,

song\_id **INT** **NOT** **NULL**,

**FOREIGN** **KEY** user\_id(user\_id)

**REFERENCES** Users(user\_id)

**ON** **UPDATE** **CASCADE**

**ON** **DELETE** **RESTRICT**,

**FOREIGN** **KEY** song\_id(song\_id)

**REFERENCES** Songs(song\_id)

**ON** **UPDATE** **CASCADE**

**ON** **DELETE** **RESTRICT**

)ENGINE INNODB;



3, for each query:

a, describe the query;

b, provide the SQL statement for it, or if you used GUI, provide the screen snapshots and the equivalent SQL statement;

c, include the results returned by the query in your final report.

Send me email if you have any further questions. Thanks.

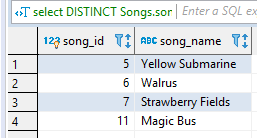
**Queries**

**You must design and develop at least 6 queries in your database application, among them,**

**1, at least 2 queries are multi-table queries;**

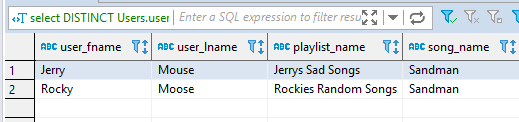
-- multi-table query, returns songs name from songs matching songId for erics playlists.

**select** **DISTINCT** Songs.song\_id,Songs.song\_name **from** Songs, Playlists **where** Songs.song\_id = Playlists.song\_id **and** Playlists.playlist\_name = "Erics Feel Good Songs" ;



-- multi-table query and group by having, returns all users names from users where playlists username and song =2 (Cyanide Metallica) and name is NOT tom

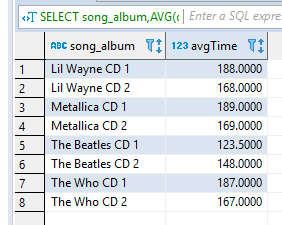
**select** **DISTINCT** Users.user\_fname,Users.user\_lname,Playlists.playlist\_name,Songs.song\_name **from** Users,Playlists,Songs **where** Playlists.user\_id = Users.user\_id **and** Playlists.song\_id =2 **group** **by** Users.user\_fname **having** **not** user\_fname= "Tom";



2, at least 2 queries use SQL aggregate functions;

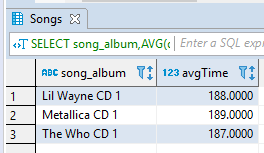
-- Aggregate function // Returns avgTime of all songs in each album.

**SELECT** song\_album,**AVG**(duration\_in\_seconds) **as** avgTime **FROM** Songs **group** **by** song\_album;



-- Aggregate, Sub-Query, group by having by. // Returns average times of albums having greater then average time of metallica cd 2. (169 seconds)

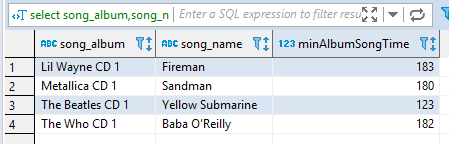
**SELECT** song\_album,**AVG**(duration\_in\_seconds) **as** avgTime **FROM** Songs **group** **by** song\_album **having** avgTime > (**select** **AVG**(duration\_in\_seconds)**from** Songs **where** song\_album = "Metallica CD 2");



3, at least 1 query uses subquery;

-- sub query // returns minumum song times of albums who are longer then the minimum time of metallicas shortest song (98 seconds.)

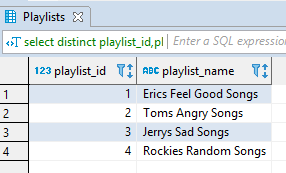
**select** song\_album,song\_name, **min**(duration\_in\_seconds)**as** minAlbumSongTime **from** Songs **group** **by** song\_album **having** minAlbumSongTime > (**select** **min**(duration\_in\_seconds) **from** Songs **where** artist\_id = 1);



4, NULL search condition should be used at least once;

-- Null select all from playlist where playlist id is not null

**select** **distinct** playlist\_id,playlist\_name **from** Playlists **where** playlist\_id **is** **not** **null**;



**insert** **into** Users ( user\_fname,user\_lname,user\_email,user\_password)

**values**

("Eric","Webb","webberic92@yahoo.com","password"),

("Tom","Cat","Tom@yahoo.com","password1"),

("Jerry","Mouse","Jerry@yahoo.com","password2"),

("Rocky","Moose","Rocky@yahoo.com","password3");

**insert** **into** Artists (artist\_name,artist\_description)

**values**

("Metallica","Influential heavy metal band."),

("The Beatles","Influential band of the 60's."),

("The Who","Influential Arena Rock band."),

("Lil Wayne","Influential rap artist.");

**insert** **into** Songs (song\_name,artist\_id,song\_album,duration\_in\_seconds)

**values**

("Sandman",1,"Metallica CD 1", 180),

("Cyanide",1,"Metallica CD 1", 198),

("Confusion",1,"Metallica CD 2", 240),

("Breadman",1,"Metallica CD 2", 98),

("Yellow Submarine",2,"The Beatles CD 1", 123),

("Walrus",2,"The Beatles CD 1", 124),

("Strawberry Fields",2,"The Beatles CD 2", 200),

("Hello,Goodbye",2,"The Beatles CD 2", 96),

("Baba O'Reilly",3,"The Who CD 1", 182),

("Pinball Wizard",3,"The Who CD 1", 192),

("Magic Bus",3,"The Who CD 2", 242),

("Behind Blue Eyes",3,"The Who CD 2", 92),

("Fireman",4,"Lil Wayne CD 1", 183),

("How to Love",4,"Lil Wayne CD 1", 193),

("Man in the Mirror",4,"Lil Wayne CD 2", 243),

("Drop the World",4,"Lil Wayne CD 2", 93)

;

**insert** **into** Playlists (playlist\_id,playlist\_name,user\_id,song\_id)

**VALUES**

(1,"Erics Feel Good Songs",1,5),

(1,"Erics Feel Good Songs",1,6),

(1,"Erics Feel Good Songs",1,7),

(1,"Erics Feel Good Songs",1,11),

(2,"Toms Angry Songs",2,1),

(2,"Toms Angry Songs",2,2),

(2,"Toms Angry Songs",2,3),

(2,"Toms Angry Songs",2,4),

(3,"Jerrys Sad Songs",3,7),

(3,"Jerrys Sad Songs",3,12),

(3,"Jerrys Sad Songs",3,14),

(3,"Jerrys Sad Songs",3,2),

(4,"Rockies Random Songs",4,5),

(4,"Rockies Random Songs",4,3),

(4,"Rockies Random Songs",4,2),

(4,"Rockies Random Songs",4,12);

**select** \* **from** Users;

**select** \* **from** Artists;

**select** \* **from** Songs;

**select** \* **from** Playlists;