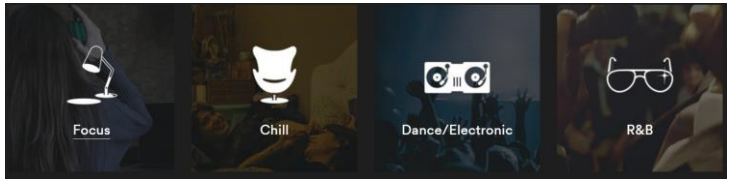
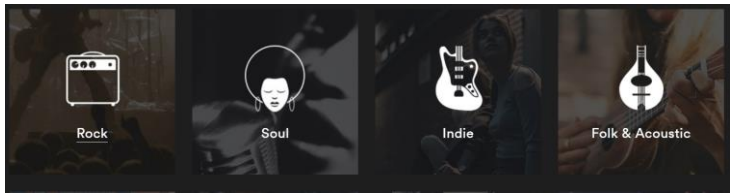
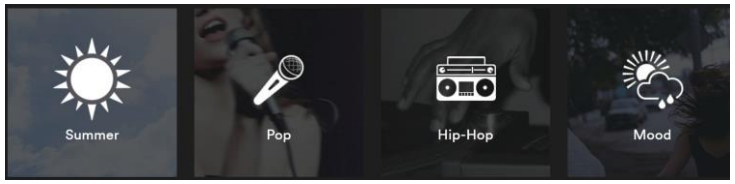






USING CUSTOMERS' MUSIC LIBRARY FOR ADVERTISING

DRIVING DECISION MAKING WITH DATABASES

Sahar Tariq
12/9/2019

Project Goal

Introduction:

- Spotify is a music streaming platform
- Spotify database contains songs, metrics of each song, artists, genres, and users that listen to them

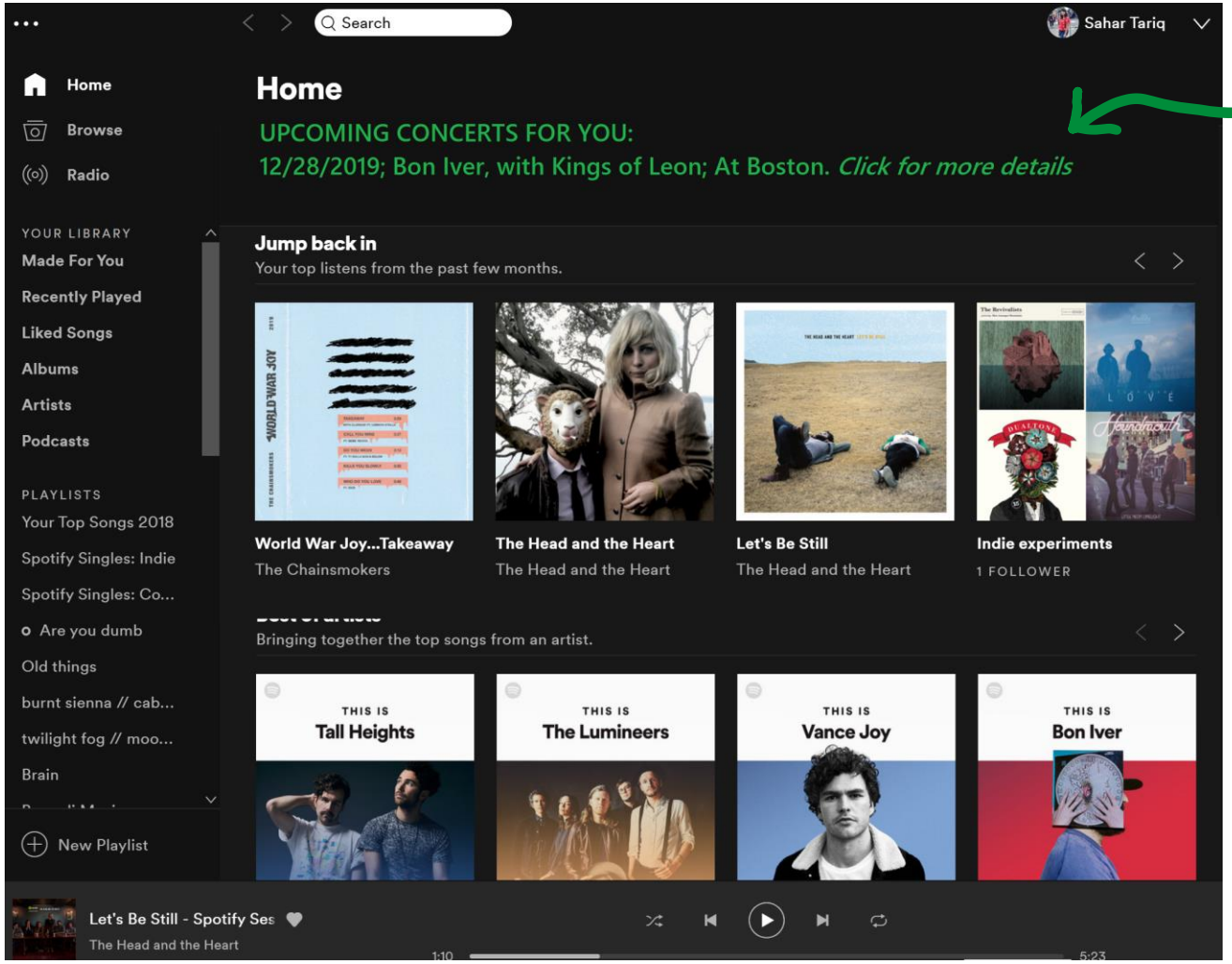
Problem:

- How can Spotify use their music database to plan and advertise concerts

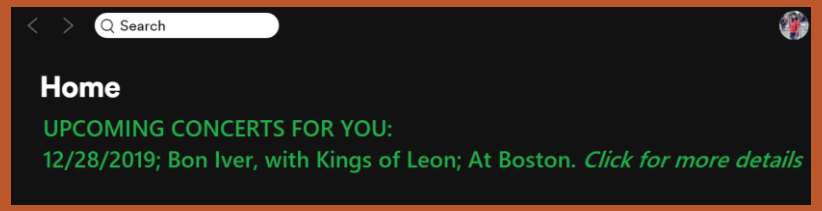
Goals:

- Goal A: Create system to harness the music and users database for planning the concerts (who will perform and where)
- Goal B: Create database for choosing which users to advertise the concert to

Idea: Banner on user's profile advertising upcoming concert in their location based on their music taste



Project Requirements



Framework to Achieve Project Goals

Step 1: Create database - network of available users, artists, songs, genres

Step 2: Query database – Choose performing artist

- Most popular artists in a location
- Most popular music genre in a location + Most popular artist in genre

Step 3: Query database - Choose supporting artists

- Find other artists who have similar music type as main artist

Step 4: Update database - Input concert details to database

Step 5: Query database - Choose users to advertise concert to

- Select users who are in same location as concert, and have listened to performing artist more than 50 times in one year

Conceptual Database Design

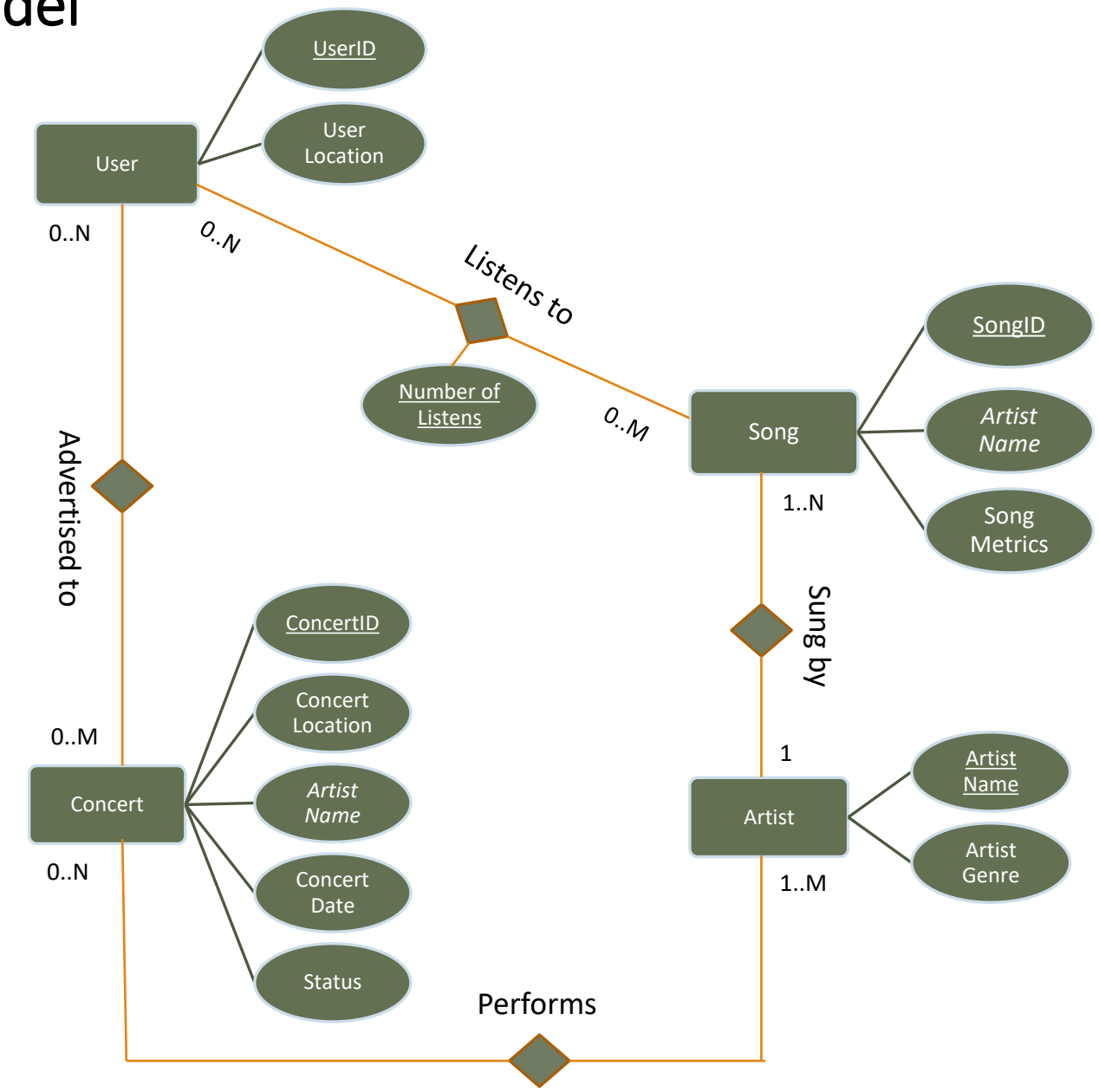
Project Summary:

Create database and relations
User → listens to → Song → sung by → Artist → performs → Concert

Update database when new shows are planned
Concert

Create connection
Concert → advertised to → User via queries

EER Model



Conceptual Database Design

Relational Model:

User(UserID, User_location)

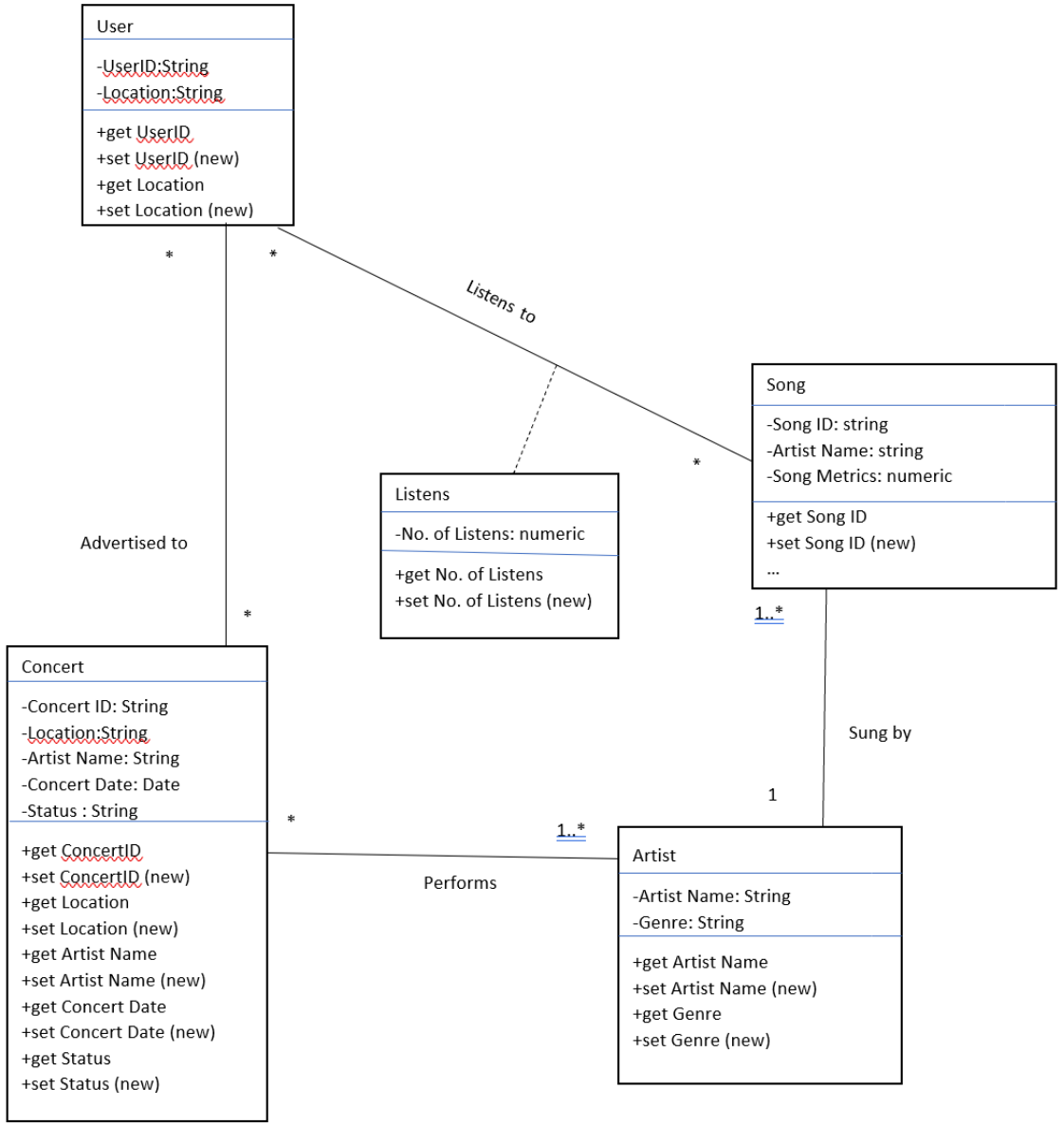
Listens(*UserID*, *TrackID*, Number_of_listens)

Songs(TrackID, *ArtistName*, TrackName, Metrics)

Artist_profile(ArtistName, Genre)

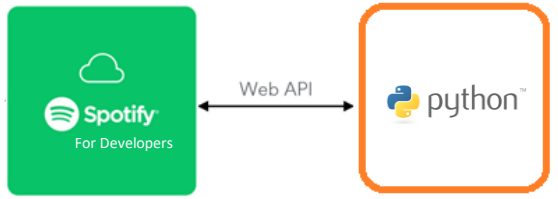
Concerts(ConcertID, *ArtistName*, Concert Location)

UML Model

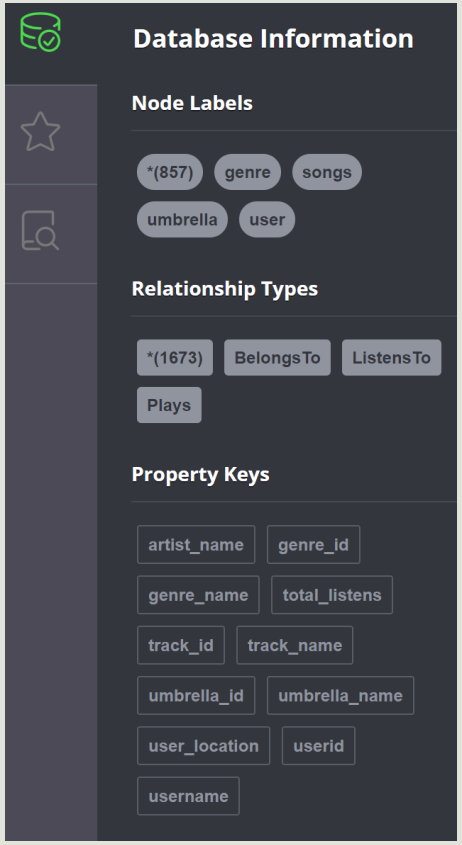


Conceptual to Implemented

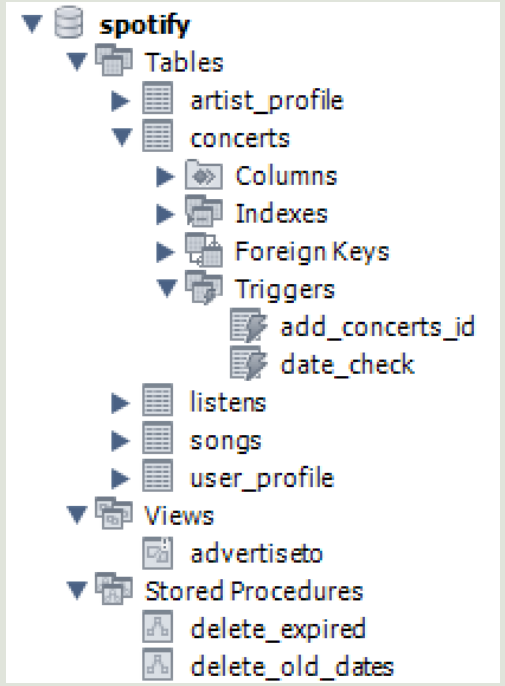
- Obtain data: create code with Spotify API and Python to scrape music library of 8 sample users + clean with R
- Created full Cypher database: For relationship network
- Created full MySQL database: To perform analytics using queries
- Created R to MongoDB connection: Be able to use Mongo's query structures
- Created central R interface to combine data from all databases in R



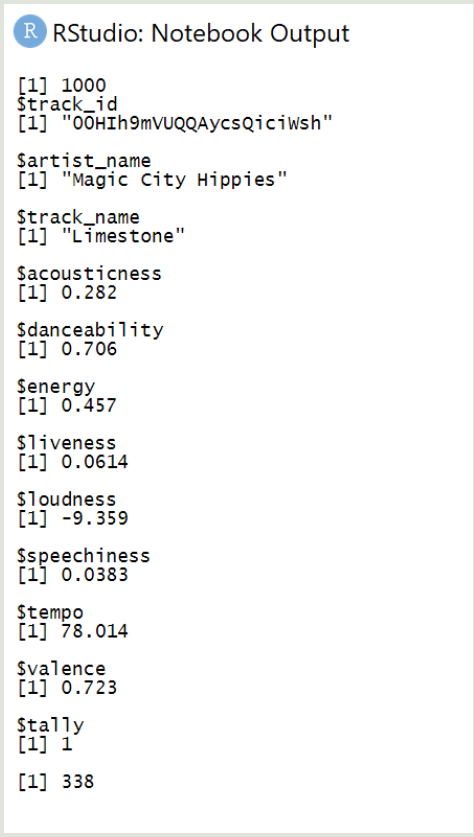
neo4j
Graph Database
&
Cypher



MySQL



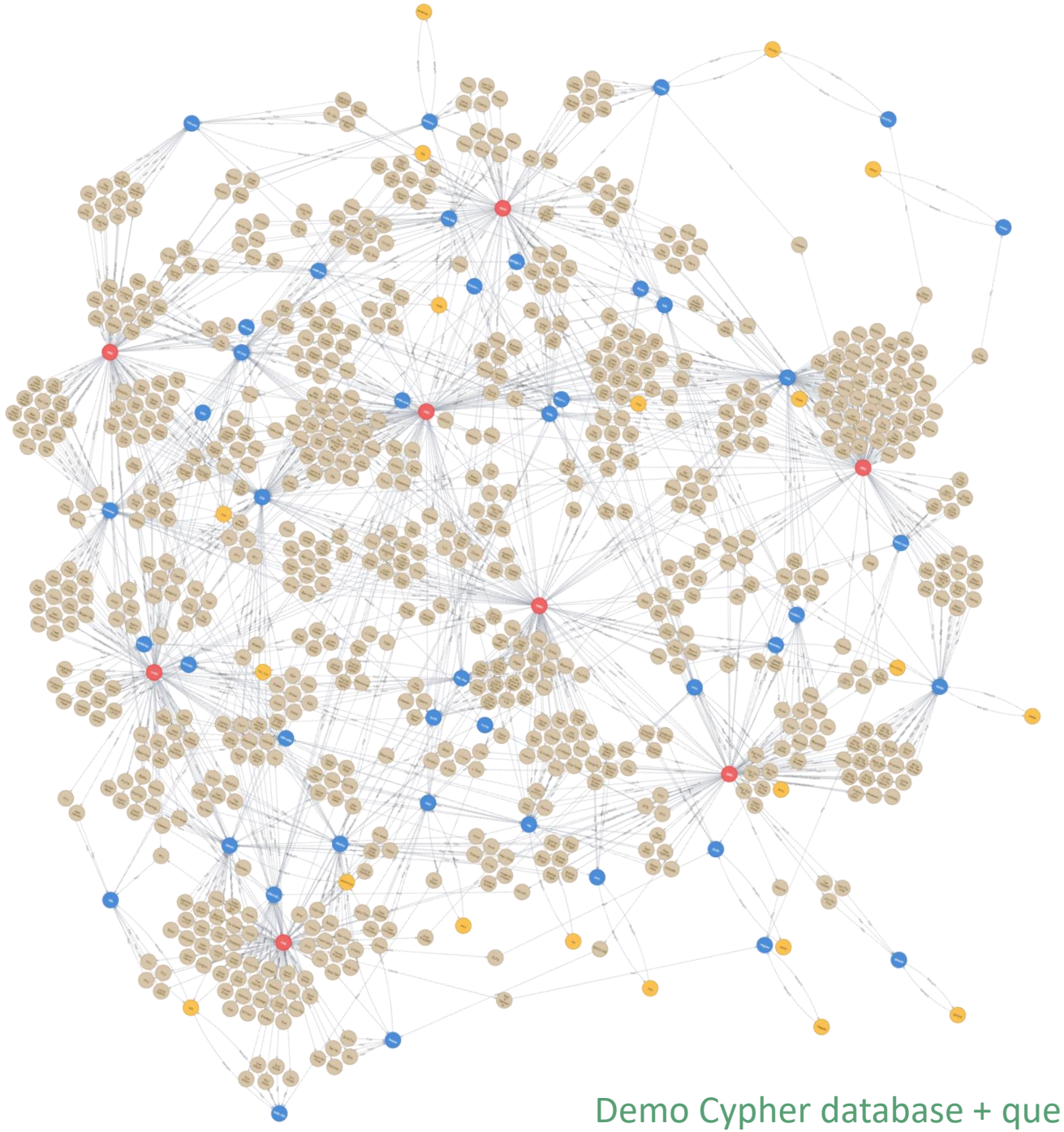
mongoDB.



Cypher Database Design

Used to:

- Visualize relationship of users, artists, songs, genres
- See what data is available
- See performance of specific artists, specific genres, and specific user's metrics
- Add / delete new users, songs, relationships

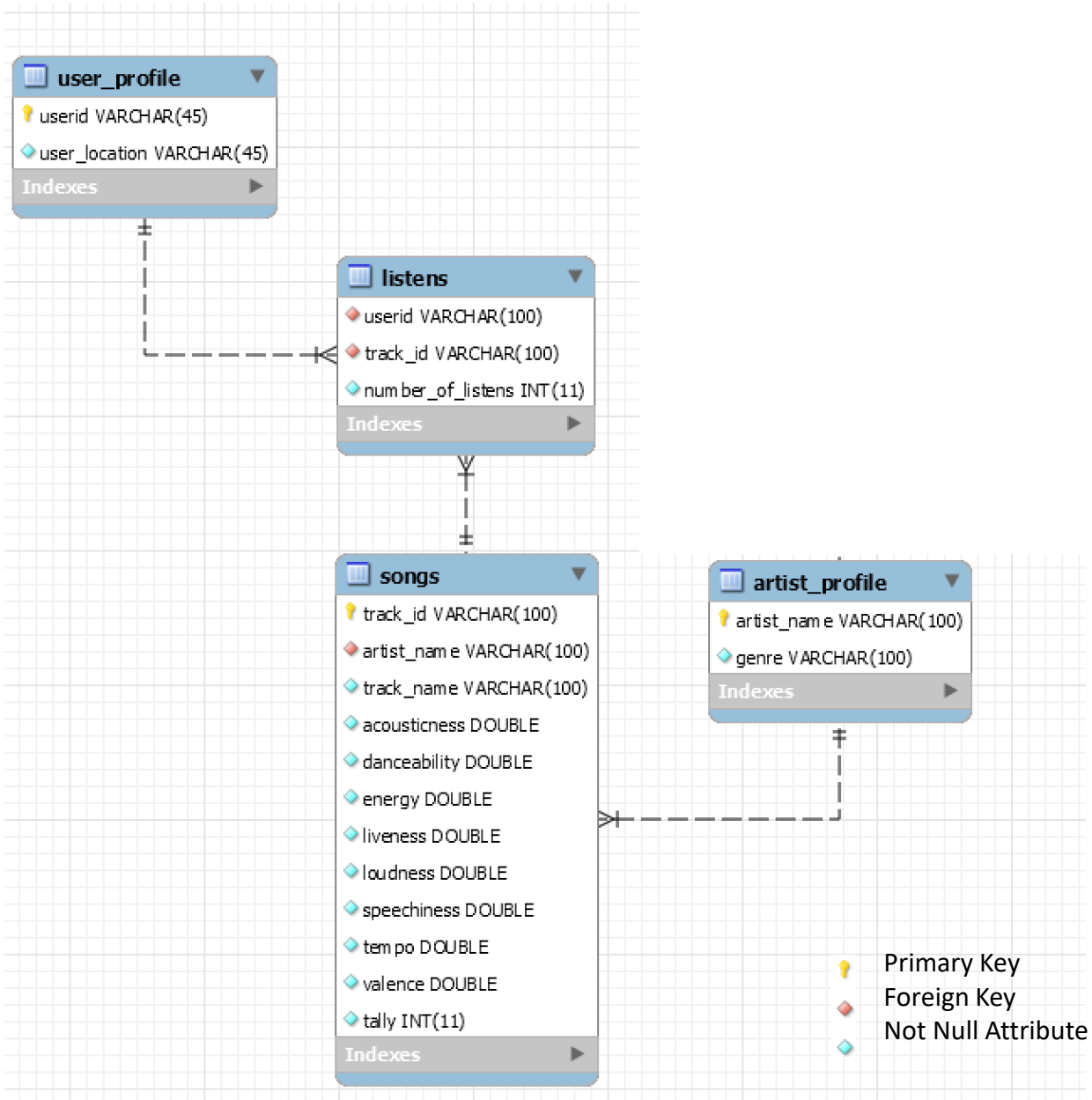


Demo Cypher database + queries →

MySQL Database Design

Used to:

- Decision making tool
- Find which artists users are listening to
- Choose performing artist by
  - Most popular artists in a location
  - Most popular music genre in a location + Most popular artist in genre



Demo mySQL database + queries →

MySQL Query Results

Artist popularity by location

artist_name	user_location	percent_users_that_listens_to_artist	number_of_songs_by_artist_users_listened_to	total_times_listened
The Head and the Heart	MA	1.0000	9	2001
Bon Iver	MA	1.0000	7	1643
The Killers	MA	1.0000	4	946
Florence + The Machine	MA	1.0000	3	767

Genre popularity by location

genre	user_location	percent_users_that_listens_to_genre	number_of_artists_in_genre_users_listen_to	total_times_listened
pop	MA	1.0000	14	4009
art-pop	MA	1.0000	13	4133
chamber-pop	MA	1.0000	9	5726
electro	MA	1.0000	5	1245

Artist popularity in each genre

genre	artist_name	percent_users_that_listens_to_artist	total_times_listened
art-pop	Maggie Rogers	50.0000	1465
art-pop	Bleachers	37.5000	360
art-pop	Glass Animals	25.0000	872
art-pop	Sylvan Esso	25.0000	775

PlayGround MongoDB Database Design

Used to:

- Create data tables
- Find average metrics of each artist using MapReduce

```
    emit(key,value);
};

var reduceFunction=function(key,value){
    reduce_val={count:0, acousticness:0, danceability:0, energy:0,

    for (var i = 0; i<value.length; i++) {
        reduce_val.count+=value[i].count;
        reduce_val.acousticness+=value[i].acousticness;
        reduce_val.danceability+=value[i].danceability;
        reduce_val.energy+=value[i].energy;
        reduce_val.liveness+=value[i].liveness;
        reduce_val.loudness+=value[i].loudness;
        reduce_val.speechless+=value[i].speechless;
        reduce_val.tempo+=value[i].tempo;
        reduce_val.valence+=value[i].valence;
    }

    return reduce_val;
};
```

Run

Result

```
' : 1.444, "avg_acousticness" : "0.1877", "avg_danceability" : "0
usticness" : "0.4677", "avg_danceability" : "0.5473", "avg_energ
 : 1.3081, "avg_acousticness" : "0.4314", "avg_danceability" : "0
```

Demo Mongo in Playground →

R + Full MongoDB Database Design

Used to:

- Find average metrics of each artist using Aggregate
- Choose supporting artists by finding other artists who have same metrics as main artist

```
#MONGO QUERY - table of artists' average metrics using Aggregate


dfm1=my_collection$aggregate('[{"$group":
                        {"_id":"$artist_name", "count": {"$sum":1}, "avg_acousticness":{"$avg":"$acousticness"},
"avg_danceability":{"$avg":"$danceability"}, "avg_energy":{"$avg":"$energy"}, "avg_liveness":{"$avg":"$liveness"},
"avg_loudness":{"$avg":"$loudness"}, "avg_speechiness":{"$avg":"$speechiness"}, "avg_tempo":{"$avg":"$tempo"},
"avg_valence":{"$avg":"$valence"}}}]')

head(dfm1)
```

	_id <chr>	count <int>	avg_acousticness <dbl>	avg_danceability <dbl>	avg_energy <dbl>	avg_liveness <dbl>	avg_loudness <dbl>	avg_speechiness <dbl>
1	Teenage Love	1	0.16500	0.7540000	0.3640000	0.2450000	-7.099000	0.05010000
2	Justin Timberlake	5	0.14636	0.6774000	0.6530000	0.1536600	-6.104400	0.05268000
3	Joss Stone	1	0.02570	0.7440000	0.6150000	0.0672000	-5.122000	0.29400000
4	BTS	3	0.13279	0.6466667	0.6720000	0.2403333	-4.998667	0.05706667
5	Bazzi	1	0.34600	0.6380000	0.7170000	0.1050000	-4.722000	0.03370000
6	Vance Joy	3	0.50900	0.6253333	0.6716667	0.1259667	-6.139000	0.03433333

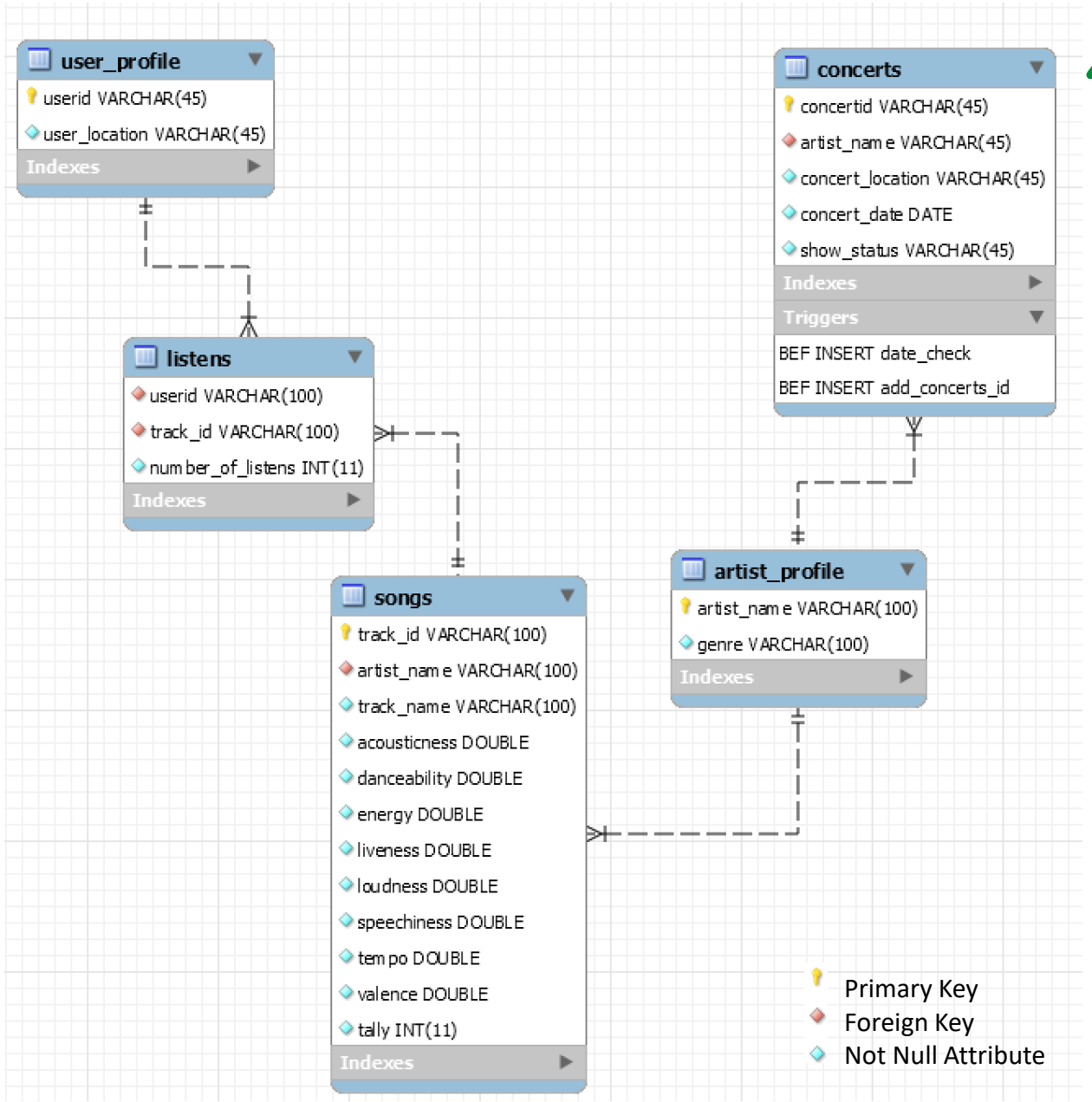
6 rows | 1-9 of 10 columns

```
#MONGO QUERY - Find supporting artists who have specific metrics

my_collection2$find('{"avg_acousticness":1,"avg_danceability":0, "avg_energy":0, "avg_liveness":0, "avg_loudness":-15,
"avg_speechiness":0, "avg_valence":0}', fields = '{"_id":1, "artist_name":1}')


```
```

	_id <chr>
1	Nathaniel Rateliff
2	Bon Iver
3	Cat Power

3 rows

Demo Mongo in R →
```

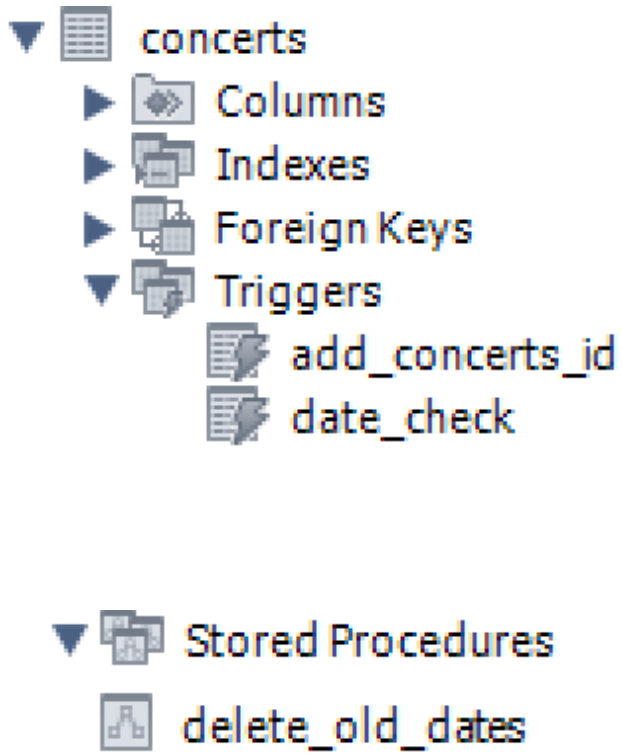
MySQL Database Design

Used to:

- Input concert details to database
- Autogenerate concert id
- Only show upcoming concerts dates
- Clean past concerts



MySQL Table Design



Enter values into table

concertid	artist_name	concert_location	concert_date	status
Alt-J-MA-2020-01-01	Alt-J	MA	2020-01-01	upcoming
Bon Iver-MA-2019-12-04	Bon Iver	MA	2019-12-04	upcoming
Maggie Rogers-MA-2019…	Maggie Rogers	MA	2019-12-11	upcoming
The Head and the Heart-…	The Head and the Heart	MA	2019-12-05	upcoming

"Trigger before insert": Autogenerates using entered values

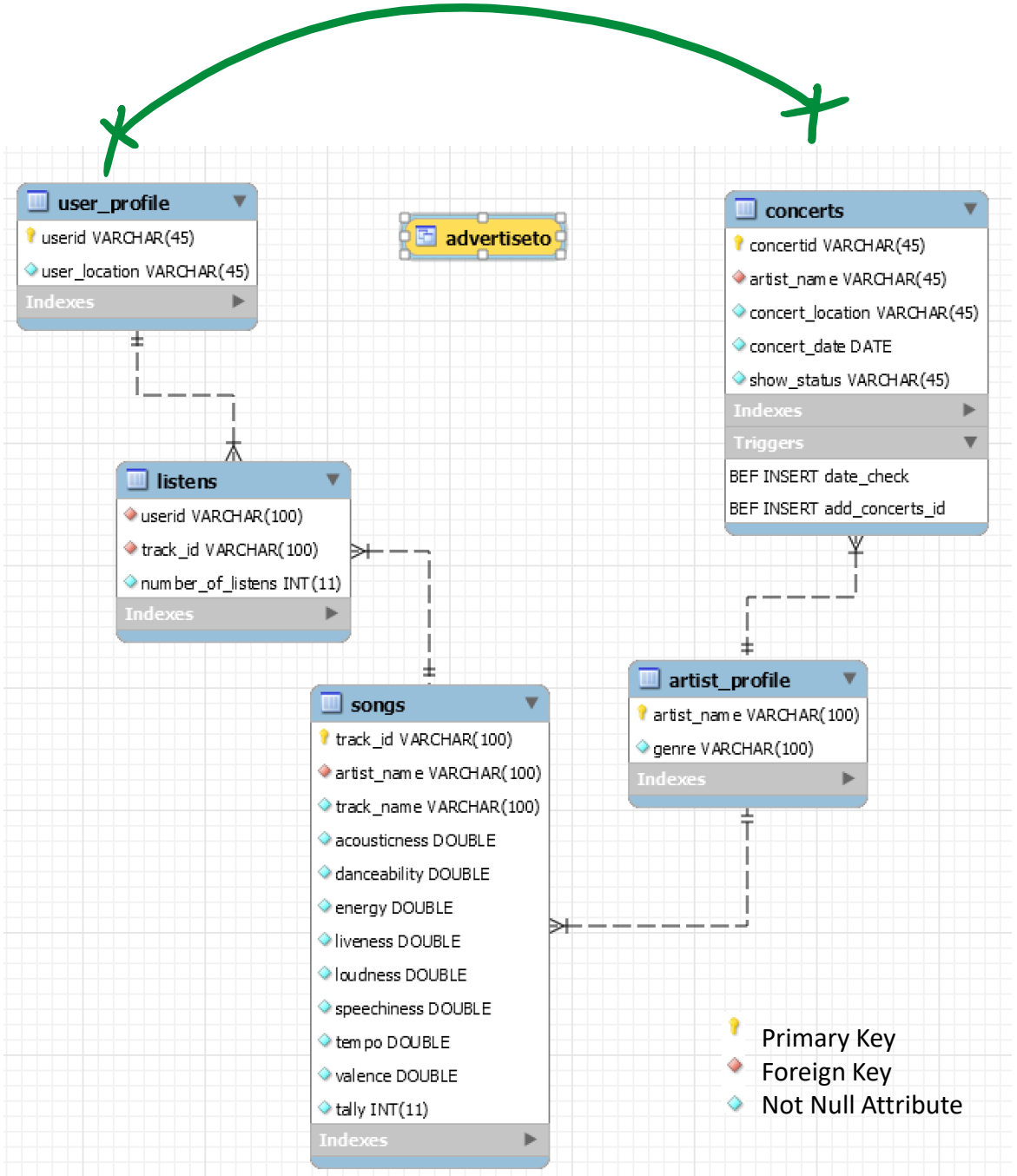
- *"Trigger before insert": status changes to "expired" when entering date before current date*
- *"Procedure": all expired and past dates are deleted*

Demo mySQL table + queries→

MySQL Table

Used to:

- Find users to advertise concert to
- Select users who are in same location as concert, and have listened to performing artist >50 times
- Connect user to concerts in a stored view

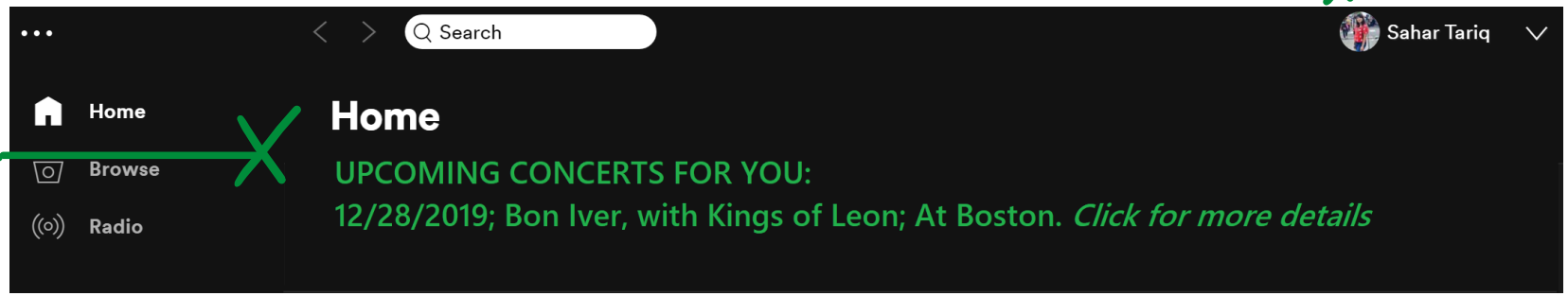


Demo mySQL view →

MySQL Results

- Views
  - advertiseto
    - concertid
    - artist_name
    - concert_location
    - userid
    - total_times_listened
    - stat

concertid	artist_name	concert_location	userid	total_times_listened	stat
Alt-J-MA-2020-01-01	Alt-J	MA	bd93	231	upcoming
Bon Iver-MA-2019-12-04	Bon Iver	MA	bd93	761	upcoming
Bon Iver-MA-2019-12-04	Bon Iver	MA	st92	882	upcoming
Maggie Rogers-MA-2019-12-11	Maggie Rogers	MA	st92	748	upcoming
The Head and the Heart-MA-2019-12-05	The Head and the Heart	MA	st92	1472	upcoming
The Head and the Heart-MA-2019-12-05	The Head and the Heart	MA	bd93	529	upcoming



Connect Centrally to R-Studio

Used to:

- Be able to get the advantages of each software
- Connect the different databases centrally to R so we access the tables made from the different softwares query abilities together
- Analyze them together
- Front end application to centrally pull data from the different databases to use them together

```
[1] "advertiseto"    "artist_profile" "concerts"       "listens"        "songs"
"user_profile"
[1] "concertid"        "artist_name"      "concert_location" "concert_date"
"status"
[1] TRUE
```

labels <chr>
user
genre
songs
umbrella

labels <chr>
BelongsTo
ListensTo
Plays

```
$track_id
[1] "00HIh9mVUQQAycsQiciWsh"

$artist_name
[1] "Magic City Hippies"

$track_name
[1] "Limestone"

$acousticness
[1] 0.282

$danceability
[1] 0.706

$energy
[1] 0.457

$liveness
[1] 0.0614

$loudness
[1] -9.359

$speechiness
[1] 0.0383

$tempo
[1] 78.014

$valence
[1] 0.723

$tally
[1] 1
```

Demo R Interface→

Thank you

Questions?