```
#DEMO: RUN THESE TO SHOW CONNECTION WORKS


#CONNECT TO MYSQL
library(RMySQL)
```

```
## Loading required package: DBI
```

```
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)

mydb <- dbConnect(MySQL(), user = 'testuser', password = [redacted]
                  dbname = 'spotify', host = '127.0.0.1')

#show connection
dbListTables(mydb)
```

```
## [1] "advertiseto"    "artist_profile" "concerts"       "listens"
## [5] "songs"          "user_profile"
```

```
dbListFields(mydb, 'concerts')
```

```
## [1] "concertid"        "artist_name"      "concert_location" "concert_date"
## [5] "show_status"
```

```
dbDisconnect(mydb)
```

```
## [1] TRUE
```

```
#CONNECT TO CYPHER
library(neo4r)

con <- neo4j_api$new(
  url = "http://localhost:7474",
  user = "neo4j",
  password = [redacted]
)

#show connection
con$get_labels()
```

```
## # A tibble: 4 x 1
##   labels
##   <chr>
## 1 genre
## 2 user
## 3 umbrella
## 4 songs
```

```
con$get_relationships()
```

```
## # A tibble: 3 x 1
##   labels
##   <chr>
## 1 BelongsTo
## 2 ListensTo
## 3 Plays
```

```
#CONNECT TO MONGO
library(mongolite)
mydb <- dbConnect(MySQL(), user = 'testuser', password = [redacted]
                  dbname = 'spotify', host = '127.0.0.1')

rs <- dbSendQuery(mydb,"select * from songs;")
songs=dbFetch(rs)

my_collection = mongo(collection = "songs", db = "Spotify")
my_collection$insert(songs)
```

```
## List of 5
##  $ nInserted  : num 500
##  $ nMatched   : num 0
##  $ nRemoved   : num 0
##  $ nUpserted  : num 0
##  $ writeErrors: list()
```

```
#show connection
my_collection$find()
```

```
my_collection$count()
```

```
## [1] 4501
```

```
my_collection$iterate()$one()
```

```
## $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
```

```
length(my_collection$distinct("artist_name"))
```

```
## [1] 338
```

```
#MYSQL QUERY - artist popularity by location


mydb <- dbConnect(MySQL(), user = 'testuser', password = [redacted]
                  dbname = 'spotify', host = '127.0.0.1')

rs <- dbSendQuery(mydb, "select s.artist_name, u.user_location, (count(distinct l.userid))/ux.total as percent_us
ers_that_listens_to_artist,
count(distinct s.track_id) as number_of_songs_by_artist_users_listened_to,
sum(l.number_of_listens) as total_times_listened
from user_profile u, listens l, songs s,
(select user_location, count(*) as total from user_profile group by user_location) as ux
where u.userid=l.userid and l.track_id=s.track_id and u.user_location=ux.user_location
group by s.artist_name, u.user_location
ORDER BY percent_users_that_listens_to_artist DESC,
number_of_songs_by_artist_users_listened_to DESC,
total_times_listened DESC;")
```

```
## Warning in .local(conn, statement, ...): Decimal MySQL column 2 imported as
## numeric
```

```
## Warning in .local(conn, statement, ...): Decimal MySQL column 4 imported as
## numeric
```

```
dfa=dbFetch(rs)
dbClearResult(rs)
```

```
## [1] TRUE
```

```
dbDisconnect(mydb)
```

```
## [1] TRUE
```

```
head(dfa)
```

```
##              artist_name user_location percent_users_that_listens_to_artist
## 1 The Head and the Heart            MA                                    1
## 2       Lake Street Dive            VT                                    1
## 3   Bring Me The Horizon            PA                                    1
## 4           Led Zeppelin            MD                                    1
## 5              Bon Iver            MA                                    1
## 6         Kendrick Lamar            FL                                    1
##   number_of_songs_by_artist_users_listened_to total_times_listened
## 1                                           9                 2001
## 2                                           9                 1463
## 3                                           8                 1614
## 4                                           8                 1336
## 5                                           7                 1643
## 6                                           7                 1548
```

```
#MYSQL QUERY - genre popularity by location

mydb <- dbConnect(MySQL(), user = 'testuser', password = [redacted]
                  dbname = 'spotify', host = '127.0.0.1')

rs <- dbSendQuery(mydb, "select a.genre, u.user_location, (count(distinct l.userid))/ux.total as percent_users_th
at_listens_to_genre,
count(distinct s.artist_name) as number_of_artists_in_genre_users_listen_to,
sum(l.number_of_listens) as total_times_listened
from user_profile u, listens l, songs s, artist_profile a,
(select user_location, count(*) as total from user_profile group by user_location) as ux
where u.userid=l.userid and l.track_id=s.track_id and s.artist_name=a.artist_name and u.user_location=ux.user_loc
ation
group by genre, u.user_location
ORDER BY percent_users_that_listens_to_genre DESC,
number_of_artists_in_genre_users_listen_to DESC,
total_times_listened DESC;")
```

```
## Warning in .local(conn, statement, ...): Decimal MySQL column 2 imported as
## numeric
```

```
## Warning in .local(conn, statement, ...): Decimal MySQL column 4 imported as
## numeric
```

```
dfb=dbFetch(rs)
dbClearResult(rs)
```

```
## [1] TRUE
```

```
dbDisconnect(mydb)
```

```
## [1] TRUE
```

```
head(dfb)
```

```
##             genre user_location percent_users_that_listens_to_genre
## 1            rock            MD                                   1
## 2             pop            VT                                   1
## 3           dance            NY                                   1
## 4             pop            MA                                   1
## 5         art-pop            MA                                   1
## 6 alternative-r&b            NY                                   1
##   number_of_artists_in_genre_users_listen_to total_times_listened
## 1                                         22                8726
## 2                                         20                4044
## 3                                         16                3033
## 4                                         14                4009
## 5                                         13                4133
## 6                                         12                3887
```

```
#MYSQL QUERY - artist popularity by genre

mydb <- dbConnect(MySQL(), user = 'testuser', password = [redacted]
                  dbname = 'spotify', host = '127.0.0.1')

rs <- dbSendQuery(mydb, "select a.genre, a.artist_name, (((count(distinct l.userid))/8)*100) as percent_users_tha
t_listens_to_artist,
sum(l.number_of_listens) as total_times_listened
from user_profile u, listens l, songs s, artist_profile a
where u.userid=l.userid and l.track_id=s.track_id and s.artist_name=a.artist_name
group by genre, artist_name
order by percent_users_that_listens_to_artist DESC, total_times_listened DESC;")
```

```
## Warning in .local(conn, statement, ...): Decimal MySQL column 2 imported as
## numeric
```

```
## Warning in .local(conn, statement, ...): Decimal MySQL column 3 imported as
## numeric
```

```
dfc=dbFetch(rs)
dbClearResult(rs)
```

```
## [1] TRUE
```

```
dbDisconnect(mydb)
```

```
## [1] TRUE
```

```
head(dfc)
```

```
##         genre            artist_name percent_users_that_listens_to_artist
## 1 chamber-pop               Bon Iver                                 50.0
## 2     art-pop          Maggie Rogers                                 50.0
## 3   indie-pop      Portugal. The Man                                 50.0
## 4 chamber-pop The Head and the Heart                                 37.5
## 5     hip_hop                  Drake                                 37.5
## 6       blues           Leon Bridges                                 37.5
##   total_times_listened
## 1                 2148
## 2                 1465
## 3                 1166
## 4                 2070
## 5                 1501
## 6                 1033
```

```
#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":{"$avg":"$livenes
s"}, "avg_loudness":{"$avg":"$loudness"}, "avg_speechiness":{"$avg":"$speechiness"}, "avg_valence":{"$avg":"$vale
nce"}}}]')


colnames(dfm1)=c("artist", "count", "avg_acousticness", "avg_danceability", "avg_energy", "avg_liveness", "avg_lo
udness", "avg_speechiness", "avg_valence")


head(dfm1)
```

```
##              artist count avg_acousticness avg_danceability avg_energy
## 1         Tech N9ne     9          0.00285            0.746      0.853
## 2      The Internet     9          0.53600            0.568      0.420
## 3 Cage The Elephant     9          0.08010            0.635      0.675
## 4    Paul McCartney     9          0.00710            0.590      0.781
## 5         Novo Amor     9          0.84300            0.378      0.331
## 6     Local Natives     9          0.01290            0.374      0.563
##   avg_liveness avg_loudness avg_speechiness avg_valence
## 1       0.3760       -4.452          0.1300      0.3620
## 2       0.1010      -11.169          0.2760      0.4530
## 3       0.0831       -3.445          0.0263      0.2730
## 4       0.1910       -5.420          0.0299      0.6560
## 5       0.0927      -15.463          0.0342      0.0655
## 6       0.0835       -5.290          0.0400      0.2640
```

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

dfm1[,3:9]=round(dfm1[,3:9], 0)
my_collection2 = mongo(collection = "average_metrics", db = "Spotify")
my_collection2$drop()
my_collection2$insert(dfm1)
```

```
## List of 5
##  $ nInserted  : num 338
##  $ nMatched   : num 0
##  $ nRemoved   : num 0
##  $ nUpserted  : num 0
##  $ writeErrors: list()
```

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

```
##                        _id
## 1 5dee1140653000000d0018aa
## 2 5dee1140653000000d0018ac
## 3 5dee1140653000000d0019a6
```

```{r}
#Can also do average using MapReduce

mydb <- dbConnect(MySQL(), user = 'testuser', password = '[redacted]
                  dbname = 'spotify', host = '127.0.0.1')

rs <- dbSendQuery(mydb,"select * from songs;")
songs=dbFetch(rs)
Songs_coll = mongo(collection = "songs", db="Songs")
Songs_coll$insert(songs)


#Map-reduce function to computer average metrics
avg_artist <- Songs_coll$mapreduce(

  map = "function(){key=this.artist_name;
  value={
  count:1,
  acousticness:this.acousticness,
  danceability:this.danceability,
  energy:this.energy,
  liveness:this.liveness,
  loudness:this.loudness,
  speechiness:this.speechiness,
  tempo:this.tempo,
  valence:this.valence,
  };
  emit(key,value);}",

  reduce = "function(key, value){
  reduce_val={count:0, acousticness:0, danceability:0, energy:0, liveness:0, loudness:0, speechiness:0, tempo:0, valence: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.speechiness+=value[i].speechiness;
  reduce_val.tempo+=value[i].tempo;
  reduce_val.valence+=value[i].valence+reduce_val.valence;
  }
  reduce_val.acousticness=(reduce_val.acousticness/reduce_val.count).toFixed(4);
  reduce_val.danceability=(reduce_val.danceability/reduce_val.count).toFixed(4);
  reduce_val.energy=(reduce_val.energy/reduce_val.count).toFixed(4);
  reduce_val.liveness=(reduce_val.liveness/reduce_val.count).toFixed(4);
  reduce_val.loudness=(reduce_val.loudness/reduce_val.count).toFixed(4);
  reduce_val.speechiness=(reduce_val.speechiness/reduce_val.count).toFixed(4);
  reduce_val.tempo=(reduce_val.tempo/reduce_val.count).toFixed(4);
  reduce_val.valence=(reduce_val.valence/reduce_val.count).toFixed(4);

  return reduce_val;}"
)

data.frame(avg_artist$`_id`, avg_artist$value)
```

```
#MYSQL QUERY - upcoming concerts

mydb <- dbConnect(MySQL(), user = 'testuser', password = [redacted]
                  dbname = 'spotify', host = '127.0.0.1')


rs <- dbSendQuery(mydb, "select *
from concerts c where c.show_status='upcoming' ;")

dfd=dbFetch(rs)
dbClearResult(rs)
```

```
## [1] TRUE
```

```
dbDisconnect(mydb)
```

```
## [1] TRUE
```

```
dfd
```

```
##                     concertid   artist_name concert_location concert_date
## 1         Alt-J-MA-2020-01-01         Alt-J               MA   2020-01-01
## 2      Bon Iver-MA-2019-12-25      Bon Iver               MA   2019-12-25
## 3 Maggie Rogers-MA-2019-12-11 Maggie Rogers               MA   2019-12-11
##   show_status
## 1    upcoming
## 2    upcoming
## 3    upcoming
```

```
#MYSQL QUERY - users to advertise concert to

mydb <- dbConnect(MySQL(), user = 'testuser', password = [redacted]
                  dbname = 'spotify', host = '127.0.0.1')

rs <- dbSendQuery(mydb, "select * from advertiseto;")
```

```
## Warning in .local(conn, statement, ...): Decimal MySQL column 4 imported as
## numeric
```

```
dfe=dbFetch(rs)
dbClearResult(rs)
```

```
## [1] TRUE
```

```
dbDisconnect(mydb)
```

```
## [1] TRUE
```

```
dfe
```

```
##                    concertid   artist_name concert_location userid
## 1         Alt-J-MA-2020-01-01         Alt-J               MA   bd93
## 2      Bon Iver-MA-2019-12-25      Bon Iver               MA   bd93
## 3      Bon Iver-MA-2019-12-25      Bon Iver               MA   st92
## 4 Maggie Rogers-MA-2019-12-11 Maggie Rogers               MA   st92
##   total_times_listened     stat
## 1                  231 upcoming
## 2                  761 upcoming
## 3                  882 upcoming
## 4                  748 upcoming
```

```
#CYPHER QUERY - user's genre listens

dff=as.data.frame('MATCH (u:user)-[rel:ListensTo]->(s:songs)-[pel:Plays]->(g:genre) WITH u.user_location as user_
location,g.genre_id as genre_id, u.userid as userid, sum(rel.total_listens) as total_listens
WHERE total_listens>100
RETURN genre_id, userid, total_listens
ORDER BY userid, total_listens  DESC;' %>%
  call_neo4j(con))
colnames(dff)=c("genre", "userid", "total_times_listened_above_100")

head(dff)
```

```
##         genre userid total_times_listened_above_100
## 1         rap   bd93                           2485
## 2 chamber-pop   bd93                           2480
## 3     art-pop   bd93                           1458
## 4        rock   bd93                           1402
## 5         pop   bd93                           1381
## 6     hip_hop   bd93                           1125
```

```
#CYPHER QUERY - specific user's nodes plot
library(purrr)
library(dplyr)
library(visNetwork)

G="MATCH p=(u:user{userid:'bd93'})-[r:ListensTo]->()
Return p;"%>%
  call_neo4j(con, type = "graph")


G$nodes <- G$nodes %>%
  unnest_nodes(what = "properties")
```

```
## Warning: The `.drop` argument of `unnest()` is deprecated as of tidyr 1.0.0.
## All list-columns are now preserved.
## This warning is displayed once per session.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
head(G$nodes)
```

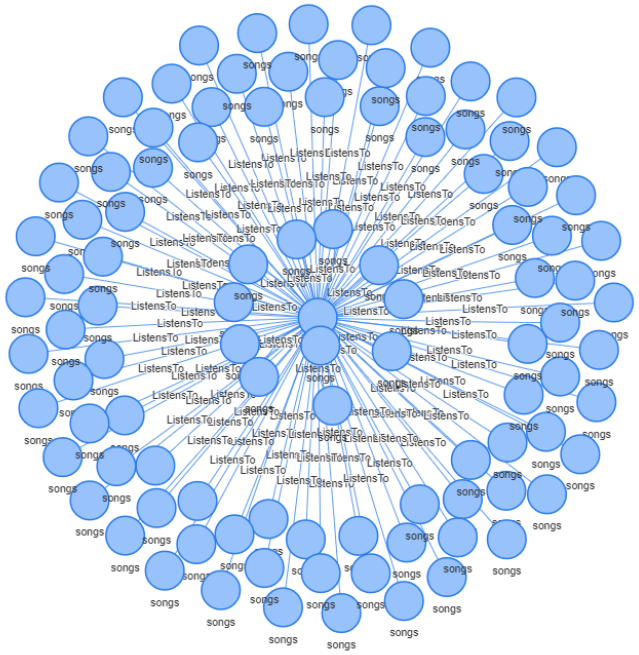
```
head(G$nodes)
```

```
## # A tibble: 6 x 8
##   id    label  user_location userid username artist_name  track_id   track_name
##   <chr> <list> <chr>         <chr>  <chr>    <chr>        <chr>      <chr>
## 1 0     <chr ~ MA            bd93   bd93     <NA>         <NA>       <NA>
## 2 562   <chr ~ <NA>          <NA>   <NA>     Paolo Nutini 5EpP1lwkA~ Iron Sky
## 3 610   <chr ~ <NA>          <NA>   <NA>     Rick Ross    1FfxSgLHU~ The Devil ~
## 4 306   <chr ~ <NA>          <NA>   <NA>     Hans Zimmer  3k2TzgUMO~ A Little P~
## 5 379   <chr ~ <NA>          <NA>   <NA>     KAYTRANADA   2vWxvpycD~ WEIGHT OFF
## 6 302   <chr ~ <NA>          <NA>   <NA>     Gunship      5VKEsChbU~ Tech Noir
```

```
G$relationships <- G$relationships %>%
  unnest_relationships() %>%
  select(from = startNode, to = endNode, label = type)
head(G$relationships)
```

```
## # A tibble: 6 x 3
##   from  to    label
##   <chr> <chr> <chr>
## 1 0     562   ListensTo
## 2 0     610   ListensTo
## 3 0     306   ListensTo
## 4 0     379   ListensTo
## 5 0     302   ListensTo
## 6 0     362   ListensTo
```

```
visNetwork::visNetwork(G$nodes, G$relationships)
```



```
#CYPHER QUERY - Number of songs listed
ga=as.data.frame('MATCH (n:songs) RETURN count(*) AS number_songs;'%>%
  call_neo4j(con))
colnames(ga)=c("Number of songs in library")


#CYPHER QUERY - Number of users
gb=as.data.frame('MATCH (n:user) RETURN count(*) AS number_users;'%>%
  call_neo4j(con))
colnames(gb)=c("Number of users in library")


#CYPHER QUERY - Number of artists
gc=as.data.frame('MATCH (n:songs) RETURN count(distinct(n.artist_name)) AS number_artists;'%>%
  call_neo4j(con))
colnames(gc)=c("Number of artists in library")

#CYPHER QUERY - Number of genres
gd=as.data.frame('MATCH (n:genre) RETURN count(n.genre_id) AS number_genres;'%>%
  call_neo4j(con))
colnames(gd)=c("Number of genres in library")


#CYHER QUERY - genres noone likes
ge=as.data.frame('MATCH (g:genre)
WHERE NOT ((:user)-[:ListensTo]->(:songs)-[:Plays]->(g))
RETURN distinct(g.genre_id);'%>%
  call_neo4j(con))
colnames(ge)=c("Genres no-one likes")


#CYPHER QUERY - genres everyone likes
gf=as.data.frame('MATCH (u:user)-[rel:ListensTo]->(s:songs)-[pel:Plays]->(g:genre)
WITH g.genre_id as genre_a, count(distinct(u.userid)) as number_of_users
WHERE number_of_users=8
RETURN genre_a;'%>%
  call_neo4j(con))
colnames(gf)=c("Genres everyone likes")


#artist everyone likes
gg=as.data.frame('MATCH (u:user)-[rel:ListensTo]->(s:songs)
WITH s.artist_name as artist, count(distinct(u.userid)) as number_of_users
WHERE number_of_users=8
RETURN artist;'%>%
  call_neo4j(con))
```

```
## No data returned.
```

```
gg[1,1]="NA"
colnames(gg)=c("Artists everyone likes")
```

```
#R FILTERS FOR PLOTS
dfaf=dplyr::filter(dfa,percent_users_that_listens_to_artist== 1)
dfaf=dplyr::filter(dfaf,user_location== "MA")


dfbf=dplyr::filter(dfb,percent_users_that_listens_to_genre== 1)
dfbf=dplyr::filter(dfbf,user_location== "MA")


dfcf=dplyr::filter(dfc,genre=="art-pop")



dfff=dplyr::filter(dff,userid=="bd93")
dfff$portion=(dfff$total_times_listened/sum(dfff$total_times_listened))*100

library(viridis)
```

```
## Loading required package: viridisLite
```

#We can take the items from the different database softwares and use them together to develop a front end decision making tool to be used by the manager

```
#R SHINY INTERFACE
library(shiny)
library(shinyWidgets)

ui = fluidPage(
  headerPanel("Spotify"),

   tabsetPanel(type = "tabs",


               tabPanel("metrics(cypher)", #page 0

               splitLayout(
    style = "border: 1px solid silver;",
    cellWidths = 250,
    cellArgs = list(style = "padding: 6px"),
    tableOutput("ga"),
    tableOutput("gb"),
    tableOutput("gc"),
    tableOutput("gd"),
    tableOutput("ge"),
    tableOutput("gf"),
    tableOutput("gg"))),


               tabPanel("artist(mysql)", #page 1

               sidebarLayout(

  sidebarPanel(
    selectizeGroupUI(
      id = "my-filters",inline = FALSE,params = list(
        user_location = list(inputId = "user_location", title = "Select location", placeholder = 'select'),
        percent_users_that_listens_to_artist = list(inputId = "percent_users_that_listens_to_artist", title = "Se
lect percent users", placeholder = 'select')
        )
      )
    ),

  mainPanel(
    h1("Artist Popularity By Location"),
    tableOutput("table"),
    plotOutput("data")
    )
  )),
```

```
  tabPanel("genre(mysql)", #page 2

               sidebarLayout(

  sidebarPanel(
    selectizeGroupUI(
      id = "my-filters2",inline = FALSE,params = list(
        user_location = list(inputId = "user_location", title = "Select location", placeholder = 'select'),
        percent_users_that_listens_to_genre = list(inputId = "percent_users_that_listens_to_genre", title = "Sele
ct percent users", placeholder = 'select')
        )
      )
    ),

  mainPanel(
    h1("Genre Popularity By Location"),
    tableOutput("table2"),
    plotOutput("data2")
    )
  )),

   tabPanel("artist by genre(mysql)", #page 3

               sidebarLayout(

  sidebarPanel(
    selectizeGroupUI(
      id = "my-filters3",inline = FALSE,params = list(
        genre = list(inputId = "genre", title = "Select genre", placeholder = 'select')
        )
      )
    ),

  mainPanel(
    h1("Artist Popularity By Genre"),
    tableOutput("table3"),
    plotOutput("data3")
    )
  )),


  tabPanel("artist(mongo)", #page 3.2

               sidebarLayout(

  sidebarPanel(
    selectizeGroupUI(
      id = "my-filters12",inline = FALSE,params = list(
        artist = list(inputId = "artist", title = "Select artist", placeholder = 'select')
        )
      ),
        selectizeGroupUI(
      id = "my-filters13",inline = FALSE,params = list(
        avg_acousticness = list(inputId = "avg_acousticness", title = "Select acousticness", placeholder = 'selec
t'),
        avg_danceability = list(inputId = "avg_danceability", title = "Select danceability", placeholder = 'selec
t'),
```

```
        avg_energy = list(inputId = "avg_energy", title = "Select energy", placeholder = 'select'),
        avg_liveness = list(inputId = "avg_liveness", title = "Select liveness", placeholder = 'select'),
        avg_loudness = list(inputId = "avg_loudness", title = "Select loudness", placeholder = 'select'),
        avg_speechiness = list(inputId = "avg_speechiness", title = "Select speechiness", placeholder = 'select'
),
        avg_valence = list(inputId = "avg_valence", title = "Select valence", placeholder = 'select')
        )
      )
    ),


  mainPanel(
    h1("Main Performing Artist"),
    tableOutput("table12"),
    h1("Supporting Artists"),
    tableOutput("table13")
    )
  )),


    tabPanel("concerts(mysql)", #page 4

                sidebarLayout(

  sidebarPanel(
    selectizeGroupUI(
      id = "my-filters4",inline = FALSE,params = list(
        concertid = list(inputId = "concertid", title = "Select concertid", placeholder = 'select')
        )
      )
    ),

  mainPanel(
    h1("Upcoming Concerts"),
    tableOutput("table4"),
    h1("Users to Advertise Concert To"),
    tableOutput("table5")
    )
  )),



      tabPanel("user profile(cypher)", #page 5

                sidebarLayout(

  sidebarPanel(
    selectizeGroupUI(
      id = "my-filters5",inline = FALSE,params = list(
        concertid = list(inputId = "userid", title = "Select userid", placeholder = 'select')
        )
      )
    ),

  mainPanel(
    h1("Listening Habits By User"),
    tableOutput("table6"),
    plotOutput("data4")
    )
  ))
```

```
server = function(input, output, session) {
  res_mod <- callModule(
    module = selectizeGroupServer,id = "my-filters",data = dfa,
    vars = c("user_location", "percent_users_that_listens_to_artist")    #page 1
    )

    res_mod1_2 <- callModule(
    module = selectizeGroupServer,id = "my-filters12",data = dfm1,
    vars = c("artist")    #page 1.2
    )

        res_mod1_3 <- callModule(
    module = selectizeGroupServer,id = "my-filters13",data = dfm1,
    vars = c("avg_acousticness", "avg_danceability", "avg_energy", "avg_liveness", "avg_loudness", "avg_speechine
ss", "avg_valence")    #page 1.3
    )

  res_mod2 <- callModule(
    module = selectizeGroupServer,id = "my-filters2",data = dfb,
    vars = c("user_location", "percent_users_that_listens_to_genre")    #page 2
    )


  res_mod3 <- callModule(
    module = selectizeGroupServer,id = "my-filters3",data = dfc,    #page 3
    vars = c("genre")
    )


   res_mod4 <- callModule(
    module = selectizeGroupServer,id = "my-filters4",data = dfd,    #page 4
    vars = c("concertid")
    )

    res_mod5 <- callModule(
    module = selectizeGroupServer,id = "my-filters4",data = dfe,    #page 4
    vars = c("concertid")
    )

    res_mod6 <- callModule(
    module = selectizeGroupServer,id = "my-filters5",data = dff,    #page 5
    vars = c("userid")
    )


    #page 0
    output$ga=renderTable(ga)
    output$gb=renderTable(gb)
    output$gc=renderTable(gc)
    output$gd=renderTable(gd)
    output$ge=renderTable(ge)
    output$gf=renderTable(gf)
    output$gg=renderTable(gg)


   #page 1
  output$table <- renderTable({
    res_mod()})
```

```
  output$data <- renderPlot({
    ggplot(data=dfaf, aes(x=reorder(artist_name,total_times_listened), y=total_times_listened))+geom_bar(stat="id
entity", position=position_dodge(), fill='seagreen')+ ggtitle("Artist popularity in location")+labs(x="Artist Nam
e", y="Total times listened")+geom_text(aes(label=total_times_listened), angle = 0, hjust=1.5, vjust=0.25, color=
"white",position = position_dodge(0.9), size=3.0)+coord_flip()})


     #page 1.2
  output$table12 <- renderTable({
    res_mod1_2()})

    output$table13 <- renderTable({
    res_mod1_3()})

  #page 2
  output$table2=renderTable({res_mod2()})

  output$data2 <- renderPlot({
    ggplot(dfbf, aes(x=user_location,y=genre ,fill=total_times_listened)) +geom_tile(state="identity")+
  ggtitle("Genre popularity in location")+scale_fill_distiller(palette ="Greens", direction = 1)+labs(x="locatio
n", y="genre")})


  #page 3
  output$table3=renderTable({res_mod3()})

  output$data3 <- renderPlot({
    ggplot(data=dfcf, aes(x=reorder(artist_name,total_times_listened), y=total_times_listened)) +
  geom_bar(stat="identity", position=position_dodge(), fill='seagreen')+coord_polar(start = 0)+ylim(-100,1500)+
  ggtitle("Artist popularity by genre")+ labs(y="Total times listened in one year", x="Artist")})


  #page 4
  output$table4=renderTable({res_mod4()})

  output$table5=renderTable({res_mod5()})


  #page 5
  output$table6=renderTable({res_mod6()})

  output$data4= renderPlot({
    ggplot(dfff, aes(x="", y=portion, fill=genre))+
  geom_bar(width = 1, stat = "identity")+coord_polar("y", start=0)+
  ggtitle("User's genre listens")+scale_fill_viridis(discrete=TRUE)})

}



shinyApp(ui = ui, server = server)
```