# Using SQL from R

*Due Sunday November 12, 11:30pm*

In yesterday's class, you learned how to access a postgres database running on the Amazon cloud. You're going to use the skills you gained yesterday as well as your prior knowledge of R in this assignment.

Please note that if you get an error along the following lines, you should reconnect to the database. All you would need to do is establish the connection again.

```
> Error in postgresqlExecStatement(conn, statement, ...) :
  RS-DBI driver: (could not Retrieve the result : server closed the connection unexpectedly
    This probably means the server terminated abnormally
    before or while processing the request.
)
```

Now, let's connect to the postgres database with R.

To get started, set up the connection parameters you will need. These are the same parameters you used yesterday, however, we will be connecting to a different database.

```
library(RPostgreSQL)

host <- "analyticsga-east2.c20gkj5cvu3l.us-east-1.rds.amazonaws.com"
port <- "5432"
username <- "analytics_student"
password <- "analyticsga"
dbname <- "nyc_taxi_trips_database"
drv <- dbDriver("PostgreSQL")

#  establish the connection
con <- dbConnect(drv, user = username, password = password,
                 dbname = dbname, port = port, host = host)
```

Test the connection with some simple commands.

```
# explore tables in the database
dbListTables(con)
```

```
## [1] "trip_data"  "trip_fares"
```

```
# explore column names in each table
dbListFields(con, "trip_data")
```

```
##  [1] "medallion"          "hack_license"       "vendor_id"
##  [4] "rate_code"          "store_and_fwd_flag" "pickup_datetime"
##  [7] "dropoff_datetime"   "passenger_count"    "trip_time_in_secs"
## [10] "trip_distance"      "pickup_longitude"   "pickup_latitude"
## [13] "dropoff_longitude"  "dropoff_latitude"
```

```
dbListFields(con, "trip_fares")
```

```
##  [1] "medallion"       "hack_license"    "vendor_id"
##  [4] "pickup_datetime" "payment_type"    "fare_amount"
##  [7] "surcharge"       "mta_tax"         "tip_amount"
## [10] "tolls_amount"    "total_amount"
```

The tables in the NYC taxi trips database are: "trip_data" and "trip_fares".

Now, you're ready to use SQL on the NYC taxi trips database. Start by exploring the data.

```
# explore trip_data
dbGetQuery(con, statement = "
           SELECT *
           FROM trip_data LIMIT 5")
```

```
##                          medallion                    hack_license
## 1 2C76BBF34A0F42FAEDA5F04E4D3B6501 CD9D0B4429613F1B62D1D4FBB72ACF2B
## 2 0AAB83929EE412F9F79BF2E0E9333D62 DD7FD411455A5E634661D3FC203DB12B
## 3 B6C04BAA6D9BAE1FB7656C78A3DE2943 48904C3579CAFCA72F005604DCBEC4FE
## 4 40D3DBC22110395F244E5148C68B583A AC150298A6C847C21D4EAF9D9FB1637C
## 5 CD4F6FCC28F932E66CD09DF340FFF16F DDA281B05AA8B3510F21FB260BACA444
##   vendor_id rate_code store_and_fwd_flag     pickup_datetime
## 1       CMT         1                  N 2013-01-22 18:53:34
## 2       CMT         1                  N 2013-01-22 19:56:21
## 3       CMT         1                  N 2013-01-21 19:26:01
## 4       CMT         1                  N 2013-01-22 00:04:09
## 5       CMT         1                  N 2013-01-21 23:50:04
##     dropoff_datetime passenger_count trip_time_in_secs trip_distance
## 1 2013-01-22 19:10:20               1              1006           2.6
## 2 2013-01-22 20:01:17               1               296           0.5
## 3 2013-01-21 19:33:37               2               455           2.0
## 4 2013-01-22 00:09:54               1               345           1.4
## 5 2013-01-22 00:09:52               1              1188           4.7
##   pickup_longitude pickup_latitude dropoff_longitude dropoff_latitude
## 1        -73.98256        40.76379         -73.98613         40.73420
## 2        -73.99966        40.71799         -73.99355         40.72129
## 3        -74.00696        40.71616         -73.98840         40.73145
## 4        -73.99333        40.74733         -73.98177         40.73425
## 5        -74.00070        40.73465         -73.94416         40.70276
```

```
# how many rows are we dealing with? this may take 10+ seconds...
dbGetQuery(con, statement = "
           SELECT COUNT(*)
           FROM trip_data")
```

```
##      count
## 1 14776615
```

We are dealing with 14.78 million rows! This is also the case for trip_fares table. Depending on the queries you do moving forward, add a LIMIT statement, to limit the number of rows in the output.

Now let's do some statistical exploration. Look at trip_distance data. That should be easy, right? It's just a single number from each record in the trip_data table. We'll add a limit of 20000 so that we're just dealing with the first 20K rows. Feel free to adjust this number and query as you see fit [ex: you may want to remove null values and/or adjust the LIMIT].

```
trip_distance <- dbGetQuery(con, statement = "
                 SELECT trip_distance
                 FROM trip_data
                 LIMIT 20000")
```

You can see that trip_distance is a data frame:

```
str(trip_distance)
```

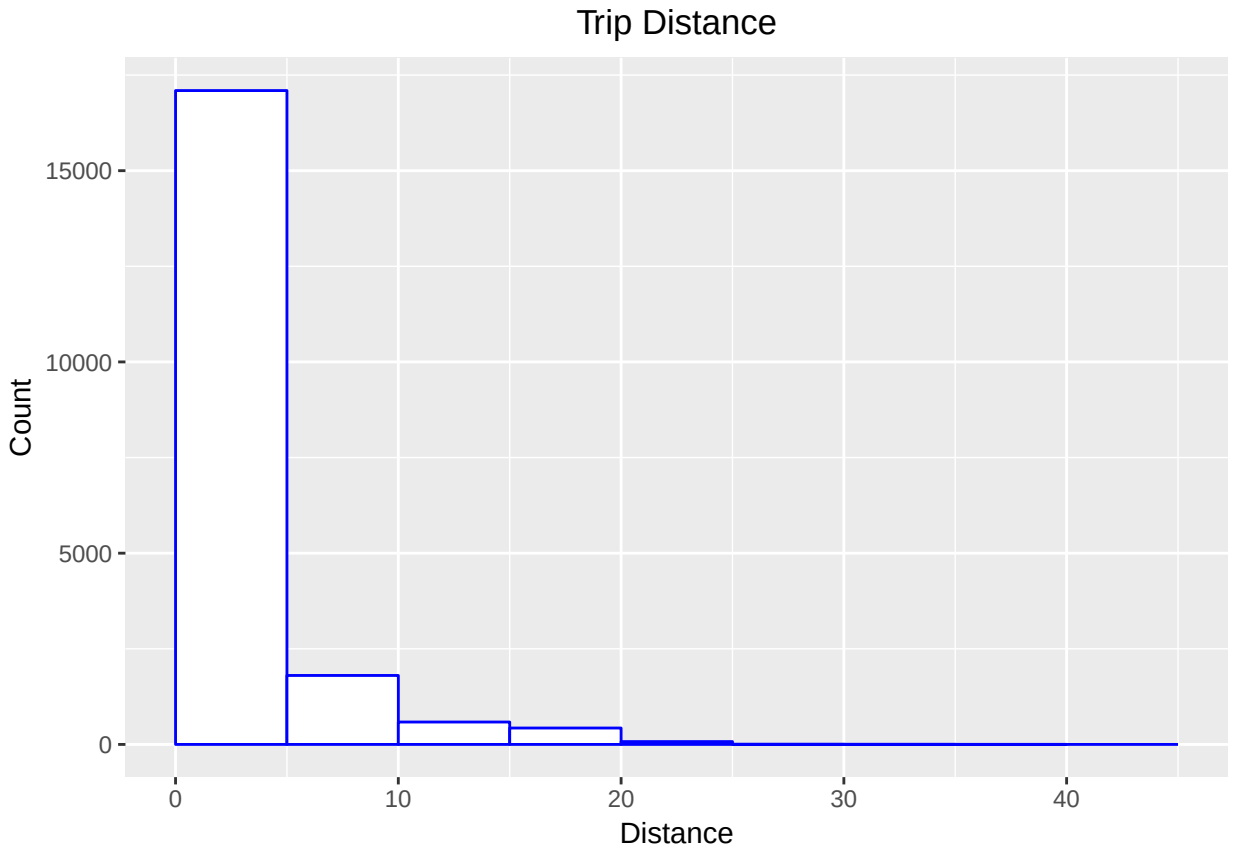
```
## 'data.frame':    20000 obs. of  1 variable:
```

```
##  $ trip_distance: num  2.6 0.5 2 1.4 4.7 2.2 1.6 2.8 1 20.8 ...
```

Explore the trip_distance data by...
- creating a histogram in ggplot2
- calculate summary statistics
    - mean
    - median
    - min
    - max
    - anything else you think could be interesting

Note any meaningful insights you find.

```
library(ggplot2)

# add ggplot code + summary stats + insights in this chunk!
c <- ggplot(data = trip_distance, aes(trip_distance)) +
  geom_histogram(breaks = seq (0,45, by =5),
                 col = "blue",
                 fill = "white") +
  labs (title = "Trip Distance") +
  labs (x = "Distance", y = "Count")+
  theme(plot.title = element_text(hjust = 0.5))

#printing the graph
print(c)
```



```
#Summarize states with igonring the NA values
analysis <- data.matrix(trip_distance, rownames.force = NA)
mean <- mean(analysis)
median <- median(analysis)
max <- max(analysis)
min <- min(analysis)

#doing printing
print(mean)
```

```
## [1] 3.08657
```

```
print(median)
```

```
## [1] 1.9
```

```
print(max)
```

```
## [1] 57.9
```

```
print(min)
```

```
## [1] 0
```

```
#Insight: From the histogram of "Trip Distance", we can see that more than 15 thousands people (almost
```

There may be variables in trip_data that can help explain the trip_distance data, like *passenger_count* and *trip_time_in_secs*.

Rework our original query for *trip_distance* to add passenger_count and trip_time_in_secs in our dataframe. HINT: you just need to adjust the SELECT statement. Don't forget to add a LIMIT!

If you'd like, you can convert trip_time_in_secs to minutes within the query.

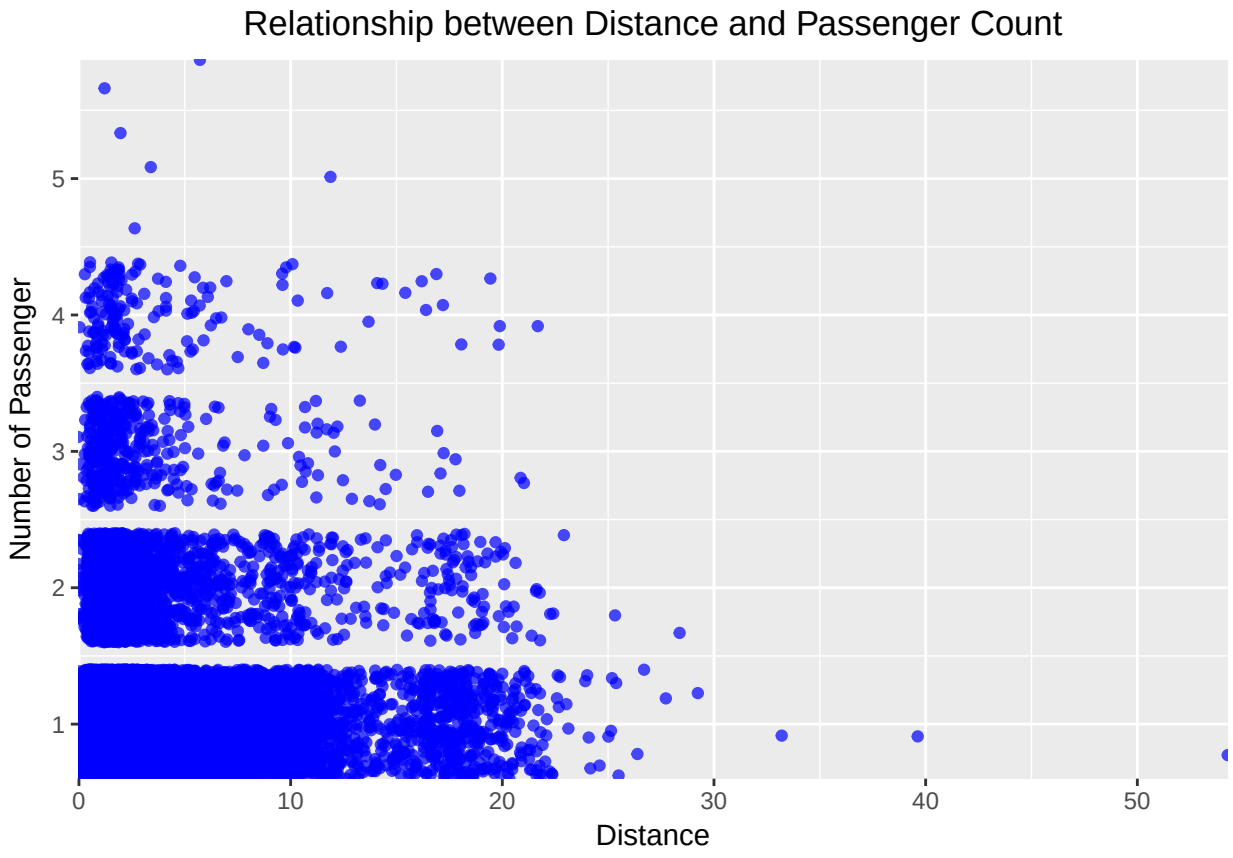
Example: *SELECT trip_time_in_secs/60 AS trip_time_in_mins FROM trip_data LIMIT 20000*

```
#Adding passenger_count and trip_time_in_secs
trip_distance <- dbGetQuery(con, statement = "
                 SELECT trip_distance,passenger_count,trip_time_in_secs
                 FROM trip_data
                 LIMIT 20000")

#convert trip_time_in_secs to minutes within the query.
trip_time_in_mins <- dbGetQuery(con, statement = "
                 SELECT trip_time_in_secs/60 AS trip_time_in_mins FROM trip_data LIMIT 20000
                 ")

#countting only passengers for future use
passenger_count <- dbGetQuery(con, statement = "
                 SELECT passenger_count
                 FROM trip_data
                 LIMIT 20000")
```

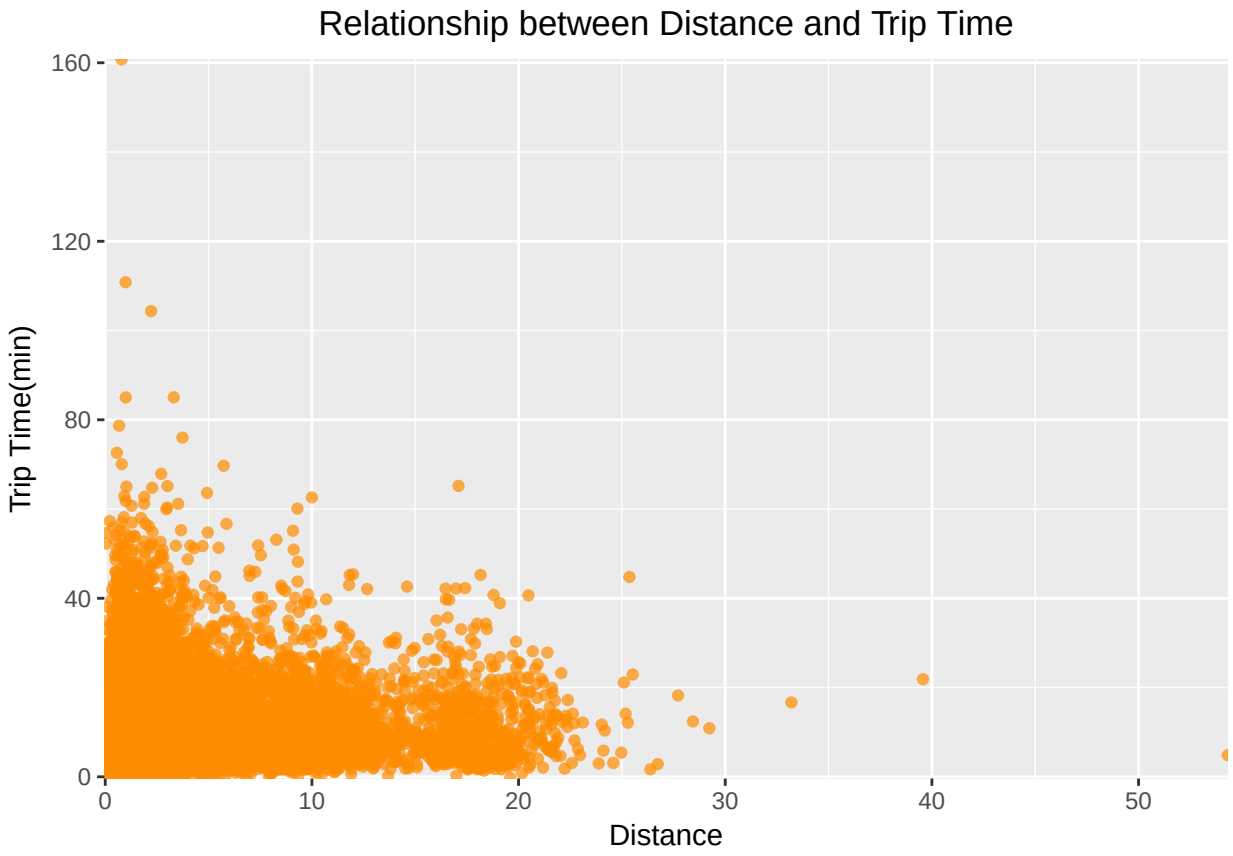
Visually explore the relationship between *trip_distance* and *passenger_count*. Also, visualize the relationship between *trip_distance* and *trip_time_in_secs* [or trip_time_in_mins]. Use whatever graph style you think is best.

One would expect that a high distance value would likely be associated with a high time value. Is this the case? Note any insights you find.

```
# add ggplot code + insights in this chunk!
ggplot(data = trip_distance, aes(x=trip_distance,y=passenger_count)) +
  geom_jitter(color = "blue",alpha=0.7) +
  scale_x_continuous(expand = c(0,0))+
  scale_y_continuous(expand = c(0,0))+
  labs (title="Relationship between Distance and Passenger Count")+
  labs (x = "Distance", y = "Number of Passenger")+
  theme(plot.title = element_text(hjust = 0.5))
```




```
#Insight: From the graph, we can see that there's a trend that the less number of passenger are, the lo

ggplot(data = trip_distance, aes(x = trip_distance, y=trip_time_in_mins)) +
  geom_jitter(color="dark orange",alpha=0.7) +
  scale_x_continuous(expand = c(0,0))+
  scale_y_continuous(expand = c(0,0))+
  labs(title="Relationship between Distance and Trip Time")+
  labs (x = "Distance", y = "Trip Time(min)")+
  theme(plot.title = element_text(hjust = 0.5))
```




```
#Insight: We can see there's a trend that indicate longer distance brings less trip time. This seems un
```

Awesome! Now let's check out the trip_fares data.

```
# explore trip_fares
dbGetQuery(con, statement = "
           SELECT *
           FROM trip_fares LIMIT 5")
```

```
##                          medallion                    hack_license
## 1 D8804787E720AA0CC644E477AF1DE7D9 B14692CA528C6A75D9D9001FF70C2B27
## 2 1DFC3445A6990A629C2F80FC902B812F 75F47AD5D2DB5761EF1EC4F3EEFEEA59
## 3 8B34DFA5A85FB4FBCD2EE2C785FFBBA3 D26018924985E5D576BAD6B7DB0F5DD7
## 4 71AEE82D433FCC6A428B0744D5D2D3D8 BFEF80A36340A1EDBB9943B08EC771E1
## 5 DFEE873CDD1DA7F38CA42A35CF3C003B 8F94188915C508E7FB4FB33EBEC09701
##   vendor_id     pickup_datetime payment_type fare_amount surcharge mta_tax
## 1       VTS 2013-01-07 07:49:00          CRD         8.0         0     0.5
## 2       VTS 2013-01-07 07:51:00          CSH         7.5         0     0.5
## 3       VTS 2013-01-07 07:53:00          CSH         6.0         0     0.5
## 4       VTS 2013-01-07 09:40:00          CSH         6.0         0     0.5
## 5       VTS 2013-01-07 09:35:00          CRD         9.0         0     0.5
##   tip_amount tolls_amount total_amount
## 1        1.6            0         10.1
## 2        0.0            0          8.0
## 3        0.0            0          6.5
## 4        0.0            0          6.5
## 5        1.8            0         11.3
```

Now, let's get the number of transactions/trips (essentially the number of rows) for each payment type. You

could order by payment_type in descending order.

You will need a COUNT(*) in the SELECT statement. You will also need a GROUP BY statement. If you get stuck, go back to the 615_SQL_starter script and look for the question "how many vendors are there per county" – use that query as a template!

Note: you do not need to use a LIMIT for this query.

```
Info on the payment types
- "CRD" -- card, debit or credit
- "CSH" -- cash
- "DIS" -- disputed fare
- "NOC" -- no charge
- "UNK" -- unknown
```

```
dbGetQuery(con, statement = "
SELECT payment_type, count(*) as num_transactions
FROM trip_fares
GROUP BY payment_type
ORDER BY payment_type DESC
           ")
```

```
##   payment_type num_transactions
## 1          UNK             6434
## 2          NOC            32783
## 3          DIS            11171
## 4          CSH          6982383
## 5          CRD          7743844
```

```
# what does the query output tell us?
# Anwser: The debit/credit card has the biggest amount of transactions, while the cash transaction is a
```

What's the average (AVG) total_amount by payment_type? You will need a GROUP BY statement. An ORDER BY statement may be helpful too. Don't use a LIMIT this time.

What does the output tell us? Are there any insights you can draw?

```
dbGetQuery(con, statement = "
SELECT payment_type, avg(total_amount) as ave_amount
FROM trip_fares
GROUP BY payment_type
ORDER BY ave_amount DESC
           ")
```

```
##   payment_type ave_amount
## 1          UNK   20.47972
## 2          CRD   16.05762
## 3          DIS   13.28042
## 4          NOC   11.72977
## 5          CSH   11.62433
```

```
# what does the query output tell us?
# Answer: The unknown type of transaction made the biggest average amount of payment, while the cash is
```

How about the average (AVG) tip_amount by payment_type? You will need a GROUP BY statement. An ORDER BY statement may be helpful too. Don't use a LIMIT this time.

What does the output tell us? Are there any insights you can draw?

```
dbGetQuery(con, statement = "
SELECT payment_type, avg(tip_amount) as ave_tip
FROM trip_fares
GROUP BY payment_type
ORDER BY ave_tip DESC
           ")
```

```
##   payment_type      ave_tip
## 1          UNK 3.2002766553
## 2          CRD 2.4152425449
## 3          NOC 0.0138318641
## 4          DIS 0.0133792857
## 5          CSH 0.0007205635
```

```
# what does the query output tell us?
# Answer: The Unknown type of transaction get the biggest average tip among all the five transaction ty
```

Create a chart or two for payment_type.

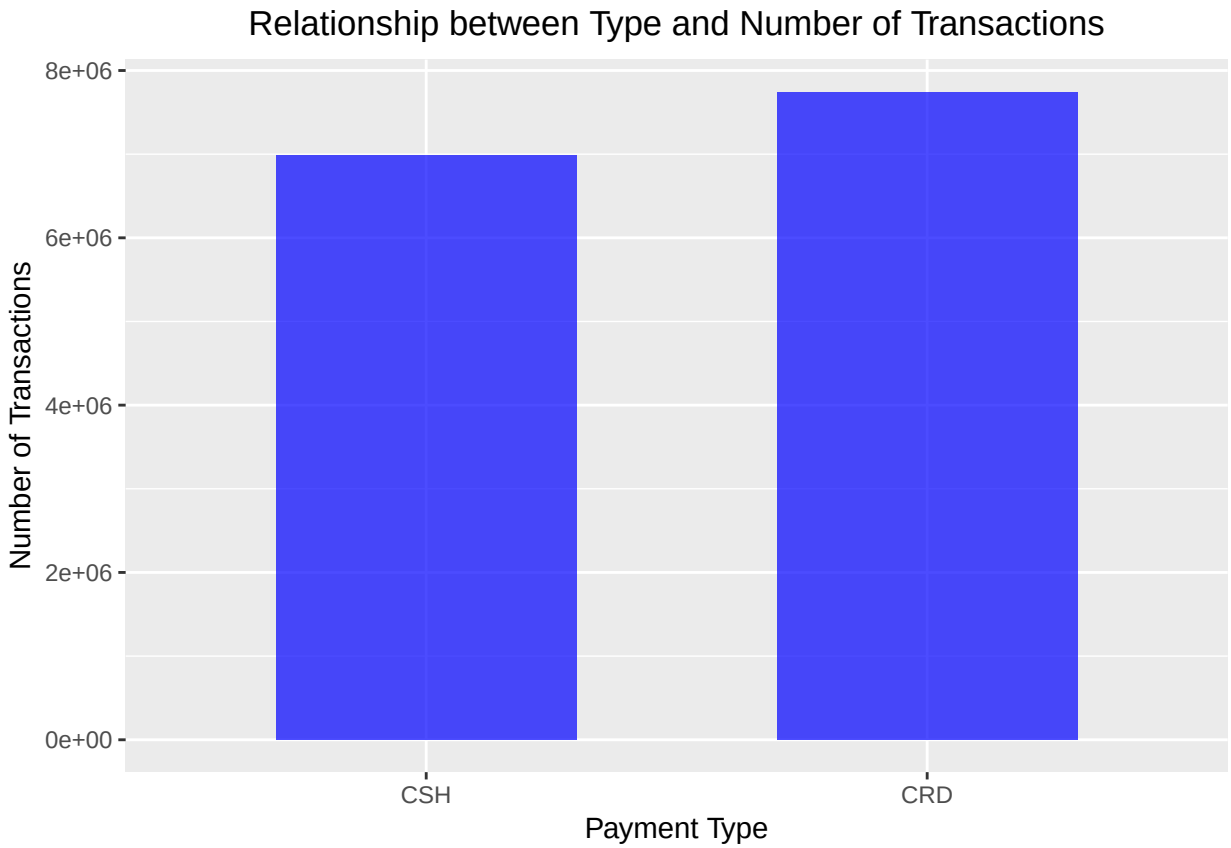
Ideas: visualize any of the queries you just ran. The charts can be as simple or complex as you'd like. Perhaps you want to compare *just* cash with card data points? If yes, you'd need to manipulate the query with a WHERE statement.

Example of a WHERE statement query from the Iowa Liquors Database: *SELECT county, FROM sales WHERE county = 'Polk' or county = 'Linn'*

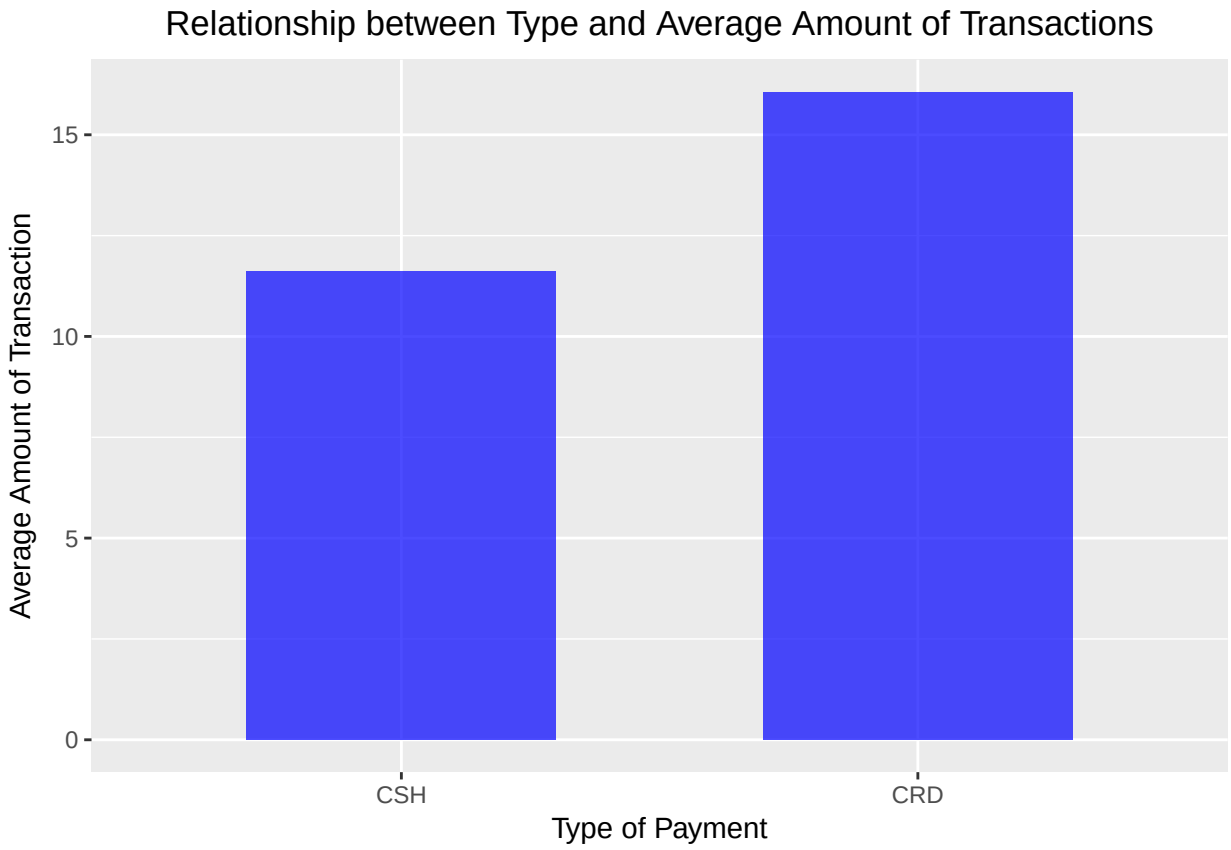
Please note the chart(s) you create tell us. Note any meaningful insights.

```
data_for_chart <- dbGetQuery(con, statement = "
SELECT payment_type, count(*) as num_transactions, avg(total_amount) as ave_amount, avg(tip_amount) as a
FROM trip_fares
WHERE payment_type = 'CSH' or payment_type = 'CRD'
GROUP BY payment_type
ORDER BY payment_type DESC
           ")

#Create another chunk here in order to skip the loading time of "data_for_chart"
```

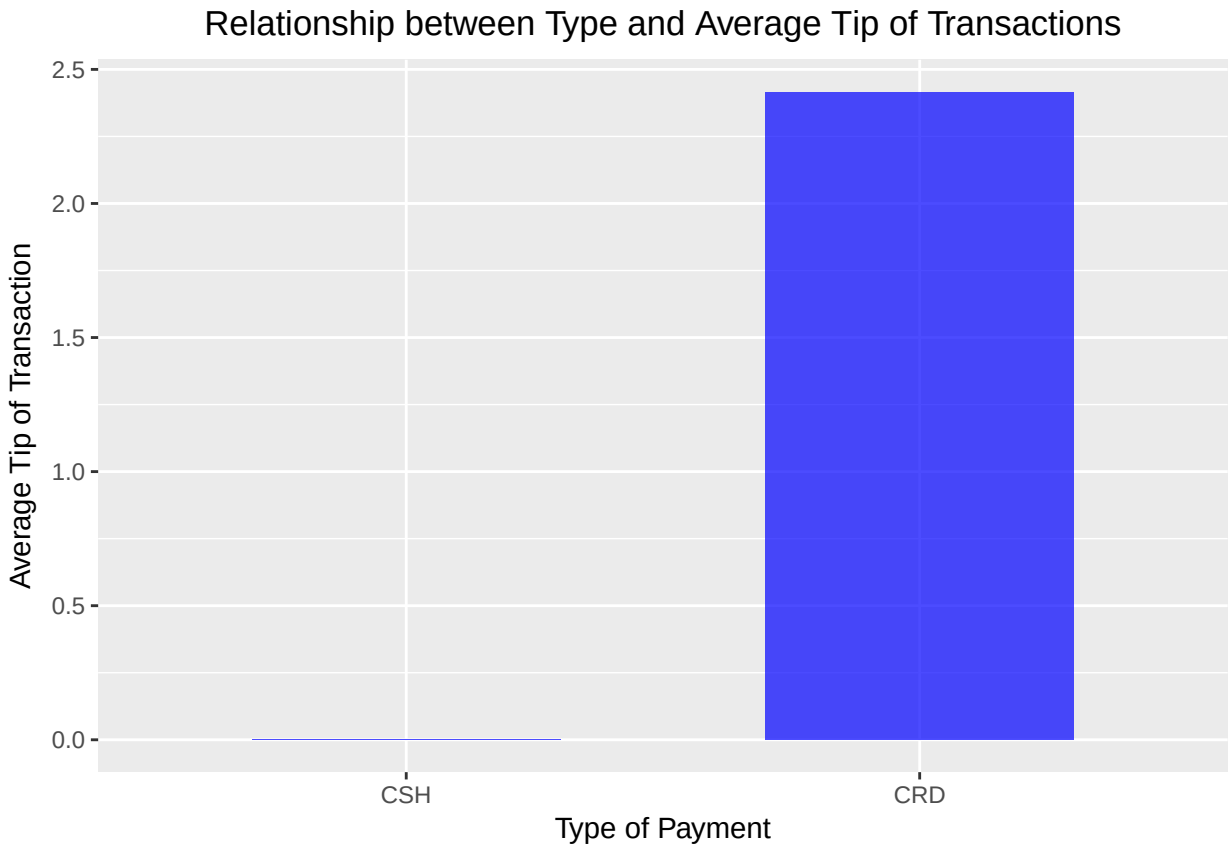
```
# add ggplot code + insights here!

ggplot(data_for_chart, aes(x=reorder(payment_type, num_transactions), y=num_transactions)) +
  geom_bar(stat="identity",fill="blue",alpha=0.7,width=0.6) +
  labs(title = "Relationship between Type and Number of Transactions") +
  labs (x = "Payment Type", y = "Number of Transactions")+
  theme(plot.title = element_text(hjust = 0.5))
```




```
ggplot(data_for_chart, aes(y=ave_amount, x=reorder(payment_type, ave_amount))) +
  geom_bar(stat="identity",fill="blue",alpha=0.7,width=0.6) +
  labs(title = "Relationship between Type and Average Amount of Transactions") +
  labs (x = "Type of Payment", y = "Average Amount of Transaction")+
  theme(plot.title = element_text(hjust = 0.5))
```




```
ggplot(data_for_chart, aes(y=ave_tip, x=reorder(payment_type, ave_tip))) +
  geom_bar(stat="identity",fill="blue",alpha=0.7,width=0.6) +
  labs(title = "Relationship between Type and Average Tip of Transactions") +
  labs (x = "Type of Payment", y = "Average Tip of Transaction")+
  theme(plot.title = element_text(hjust = 0.5))
```




```
# Insight: From the three graphs above, we can see that the biggest variation between two types of paym
```

Let's revisit the trip_data table.

```
dbGetQuery(con, statement = "
           SELECT *
           FROM trip_data LIMIT 5")
```

```
##                          medallion                      hack_license
## 1 8CE5E32B1FE6F1A5E1D4DC10DED8F77D D64A765F3F6DE8712550C09807C5FE1A
## 2 004593A2C9FEBC352B58C438F73FB435 DFE5B4139DD63453754D93CC299A05EA
## 3 176772787D2EA298752417427CB6520F EE147AB4BA38DC068FA7AB5F3A40CFBF
## 4 60CD2CAE8AF2CA0E7A9CE5895EF6FC5D CDEE1D89036C514DD92715FA4F6FE03C
## 5 E1E2675F46577A598CC83F08D52A02AE 900B6A0BFE2326C10C624D61BB0E9BE0
##   vendor_id rate_code store_and_fwd_flag     pickup_datetime
## 1       CMT         1                  N 2013-01-23 09:14:03
## 2       CMT         1                  N 2013-01-23 14:37:09
## 3       CMT         1                  N 2013-01-23 14:33:51
## 4       CMT         1                  N 2013-01-22 21:04:44
## 5       CMT         1                  N 2013-01-23 14:37:12
##      dropoff_datetime passenger_count trip_time_in_secs trip_distance
## 1 2013-01-23 09:50:32               1              2189           8.4
## 2 2013-01-23 14:59:11               1              1322           8.3
## 3 2013-01-23 15:05:19               1              1888           8.9
## 4 2013-01-22 21:07:28               1               164           0.9
## 5 2013-01-23 15:05:13               1              1681           4.5
##   pickup_longitude pickup_latitude dropoff_longitude dropoff_latitude
## 1        -73.88525        40.77011         -73.98632         40.75859
```

```
## 2        -73.86306        40.76908        -73.95270        40.77664
## 3        -73.99181        40.75587        -73.88538        40.77319
## 4        -73.98218        40.76337        -73.98789        40.75341
## 5        -74.00764        40.70834        -73.98661        40.76220
```

As a bonus, try your best to create a map! You can either visualize pickup_longitude & pickup_latitude OR dropoff_longitude & dropoff_latitude. Although, if you'd like to, feel free to create more than 1 map.

Here's a sample query you can run to get the data you need. You will need to add a LIMIT statement.

```
map_data_pickup <- dbGetQuery(con, statement = "
                        SELECT pickup_longitude, pickup_latitude
                        FROM trip_data
                        LIMIT 5000")

map_data_dropoff <- dbGetQuery(con, statement = "
                        SELECT dropoff_longitude, dropoff_latitude
                        FROM trip_data
                        LIMIT 5000")
```

Now to actually visualize your data... I suggest looking into *ggmap*. Some example code can be found here: https://blog.dominodatalab.com/geographic-visualization-with-rs-ggmaps/

What does your map tell us? Are there any insights you can draw from your map?

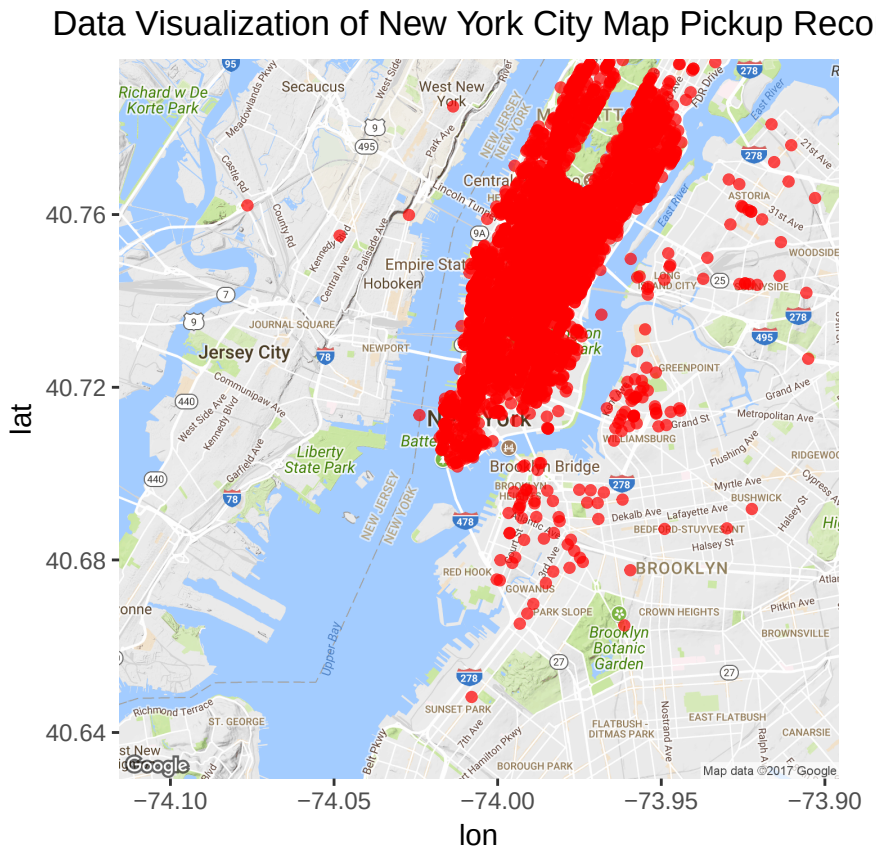
```
# install.packages("ggmap")
library(ggmap)


# add ggmap code + insights in this chunk!

Map <- get_map(location = "New York City",
                   zoom = 12,
                   crop = TRUE)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=New+York+City&zoom=12&size=640x6
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=New%20York%20City&se
```

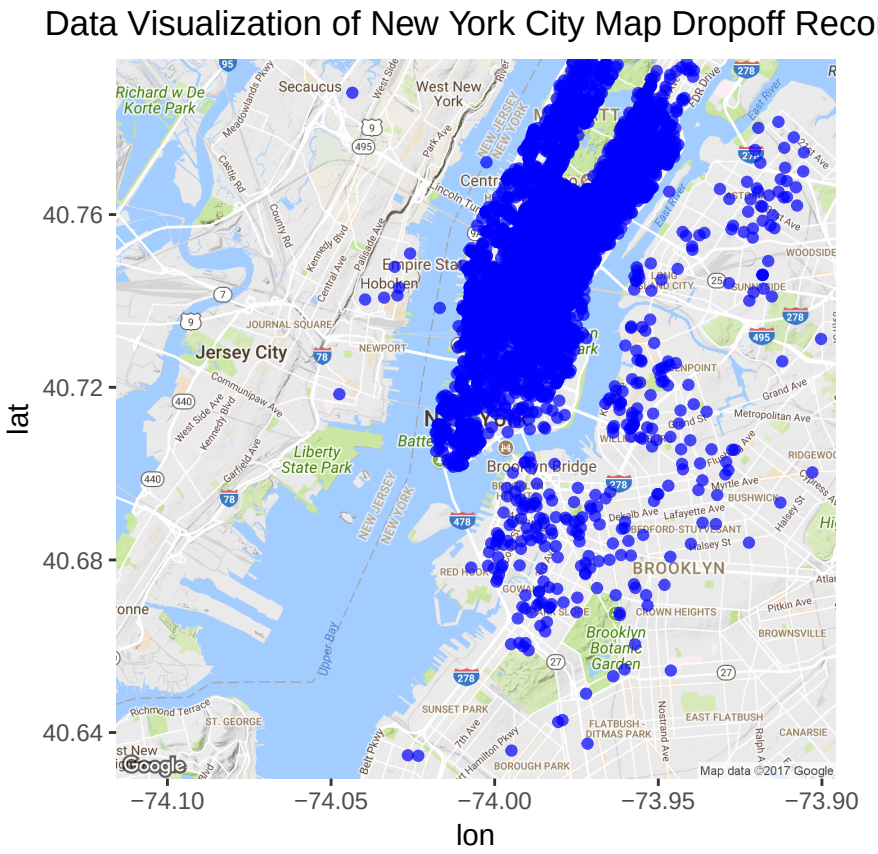
```
ggmap(Map) +
  geom_point(data = map_data_pickup, aes(x = pickup_longitude, y = pickup_latitude),color="red",alpha=0
  labs(title = "Data Visualization of New York City Map Pickup Record") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning: Removed 411 rows containing missing values (geom_point).
```




```
ggmap(Map) +
  geom_point(data = map_data_dropoff, aes(x = dropoff_longitude, y = dropoff_latitude),color="blue",alp
  labs(title = "Data Visualization of New York City Map Dropoff Record") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning: Removed 371 rows containing missing values (geom_point).
```




```
#We can see that from the side of location, no matter for pickup or dropoff, most data are collected wi
```

Congrats! :)

You just used SQL to access data in the cloud AND used R to generate summary stats and create visualizations. Hopefully you were able to translate the data into insights too.

These are highly valued skills in the data science realm, which is why I would add this assignment to your GitHub! To take this assignment to the next level for your GitHub, you can expand and organize the analysis to create a story, as well as draw conclusions / make suggestions based on the analysis.

We weren't able to use joins in this assignment, because it would take some time. If you'd like to expand your analysis and try to join the tables but aren't sure where to start (there are multiple common fields you'd have to join on), get in touch. I can send you some sample queries.