Peter Antonaros
Math 290 Homeowork 4

NOTE: Most of the queries took way too long to execute. Some are tested and working others are my best approach to the problem.
Exercise 1

select count(*) as distinct_observation_count from (select distinct "vendorID", ... from yellowtaxi_data) as column_counter;

The … represent all the names of the other columns.

Exercise 2

We know there is no singular column that would result in a unique primary key. A single column would have to return a value of length(column) to imply that all values are unique from each other. Seeing the totals for each column though, we can say that there is a composite key which can be made from some number of columns. Personally I would find 3 of the column totals who's sum is greater than the length of the table and use that. It will be guaranteed that they are unique.

Exercise 3

1) How many rows have a "passenger_count" equal to 5.

> **select count(*) from (select * from yellowtaxi_data where passenger_count = 5) as pCount**
>
> **There are 5,040,905 rows where there were 5 passengers.**

2) How many distinct trips have a "passenger_count" greater than 3?

> An assumption I made is that all the rows are distinct because they each represent a single taxi ride on record. To me that is distinct by default
>
> **select count(*) from (select * from yellowtaxi_data where passenger_count > 3) as pCount**

3) How many rows have a tpep_pickup_datetime between '2018-04-01 00:00:00' and '2018-05-01 00:00:00'?

4) How many distinct trips occurred in June where the tip_amount was greater than equal to $5.00?

5) How many distinct trips occurred in May where the passenger_count was greater than three and tip_amount was between $2.00 and $5.00?

Peter Antonaros
Math 290 Homeowork 4

6) What is the sum of tip_amount in the 2018_Yellow_Taxi_Trip_Data dataset? (Hint: use the SUM() function to find the answer)

> **select sum(tip_amount) from yellowtaxi_data**
>
> **The sum was $210,156,392.48**

Exercise 4

base = ~15gb
databse = ~ 13gb

After deletion there was no change in the size of either. This makes sense since with most things computers, simply deleting something does not remove the object from memory but deferences it. Leads into the next question

Exercise 5

The size of the database folders has decreased. My initial thoughts are vaccum is some sort of cleanup procedure that postgres will run, after all the name is “vaccum”.

At this point I went to postgres documentation and searched for “vaccum” yielding this as the description for what it does; **VACUUM reclaims storage occupied by dead tuples.**

To me this is similar to garbage collection in a language like java. Objects which no longer have pointer references are discarded as junk to free up memory space. I assume postgres is doing something similar to tuple object who do not have pointers referencing them.

Exercise 6

> **Table truncated and re-imported**