Alexandra Norman, Lauren Darr, Ramesh Simhambhatla , John Heinen

**I.** **INTRODUCTION**

Tableau Performance with Big Data Sets(Summer 2017)

**T**ableau is a data visualization software developed by U.S. company Tableau Software. It is popularly used as a business intelligence (BI) tool. In the words of the company, this software meets the need of “making data understandable to ordinary people” (Tableau, 2017). Tableau is a big player in the world of data analytics with 26,000 customers as of 2014 (Trefis Team, 2015). One reason Tableau is so useful is its ability to query relational databases to build visualizations. Group member, Alexandra Norman’s personal experience with Tableau at work inspired this exploration into the functional limitations of Tableau with big data. Norman reported that Tableau slowed down considerably when reading ‘large’ tables. But, what is too big for Tableau to handle? The primary goal of this project is to test the performance of Tableau when querying large databases.

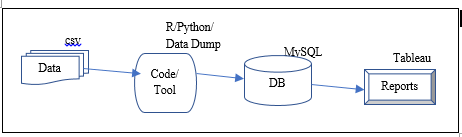
# **Project Plan**

**‘B**ig data’ is a popular term casually used in industry and academia alike. Unfortunately, there is no singular definition for how big ‘big data’ is. Forbes contributing author Lisa Arthur suggested that ‘big data’ doesn’t need to be limited to unstructured or exponentially growing data, but rather it is more important that it is well defined within every enterprise (Arthur, 2013). The aim of this project isn’t to cater to any prevailing definition of ‘big data’ but to focus on functionality. Therefore, the project’s initial plan is defined around a hypothesis from practical experience. It is surmised that a suitable starting database size to capture any decline in performance is approximately 1GB.

The general idea is to query a singular database at a series of ascending sizes to build visualizations and record the processing metrics. There are three major steps leading up to this final testing (Figure 1). Publicly available U.S. census data was acquired from the U.S. Census Bureau. More specifically, the American Community Survey Public Use Microdata Sample (PUMS) from 2011-2015. This data set is a sample of raw data intended to be used for custom analyses. The chosen records contain individual person responses as opposed to household responses. This data set consists of 298 variables.

An ideal platform for creating and using a large collaborative database is a dedicated server or cloud computing service. In theory, this would provide more efficiency than using personal desktops to upload large files to distinct, but identical databases in MySQL.

**Figure 1. Data extraction and test work flow**



# **Limitations**

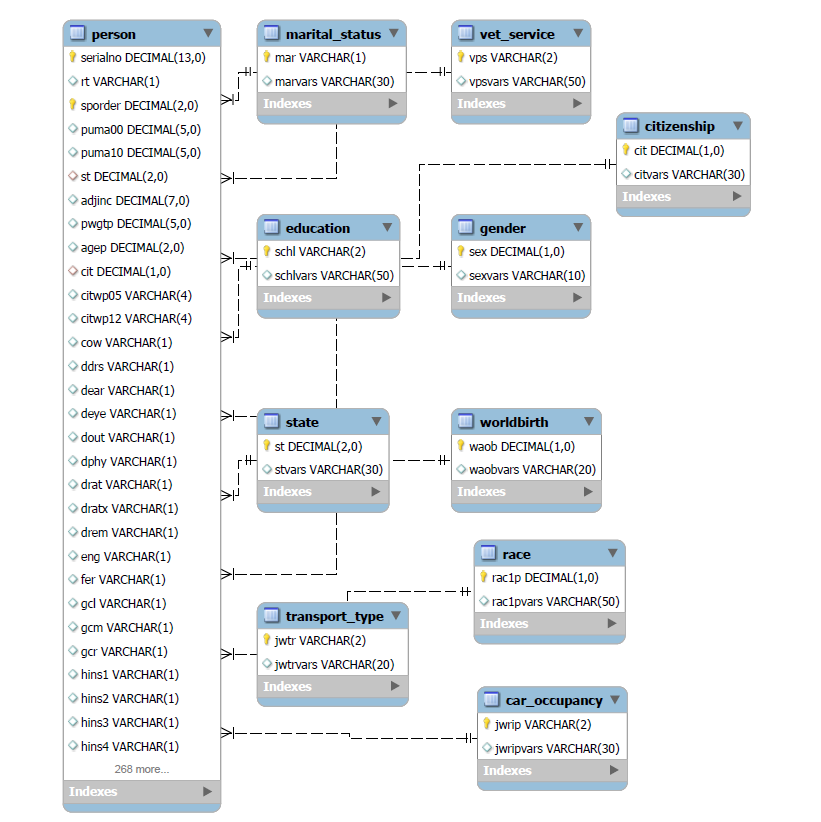
**T**he need for a hosting service proved to be a major limitation in the project plan. It was determined that a hosting service of adequate size to test a large database would not be available for free. This hindrance means it may not be possible to test Tableau with database sizes large enough to detect performance break-down.

# **Revised Methods**

**W**ith limited free hosting services available the project plan was revised to include a proof of concept and a new focus on capabilities for data scientists working remotely. A relational database was created using SQL (See appendix). This database consists of a ‘person’ table including all 298 variables available. Ten smaller tables with primary keys linked to foreign keys in the ‘person’ table were also created (Figure 2.).

Person table files were uploaded and tested at an increment of 500,000 records, with an initial table that started with 500,000 records. Tableau visuals were created on a personal computer: system model-HP Spectre x360 Convertible 15-bl0XX, Processor: Intel(R) Core™ i7-7500CPU@ 2.70GHz, 2904 Mhz, 2 Core(s), 4 Logical Processor(s). Tableau connected to MySQL server.

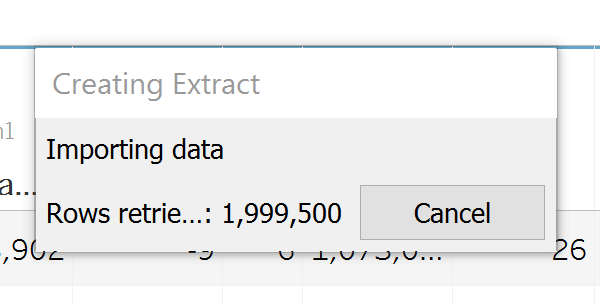
**Figure 2. Database Schema**

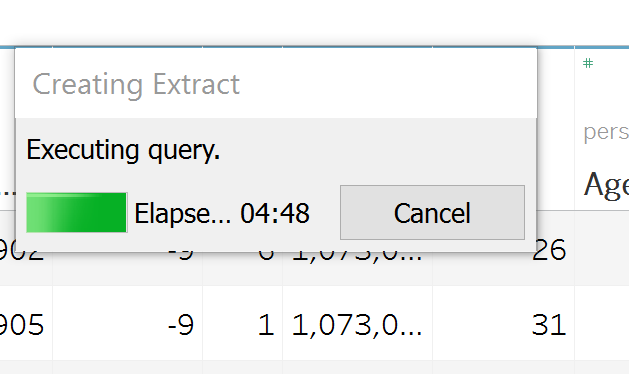


# **EXECUTION**

Tableau provides a number of metrics to test the performance of a workbook. A brief description of each metric included in this project is included below. Further information about how to ‘record and analyze workbook performance’ can be found here: <http://onlinehelp.tableau.com/current/pro/desktop/en-us/perf_record_create_desktop.html>

**Generating Extracts:** This measures the amount of time to import the extract data at the time of refresh or in this case, the first time the data is loaded. This metric is only used when doing extracts.





**Executing Query**: Executing a query includes any instance when data is pulled. This includes pulling data from multiple tables into the same visual and filtering on dashboards or individual workbooks. In general, executing a query will take longer during a live connection. A live connection assumes that data is constantly being updated in the database of interest and therefore Tableau must reconnect directly to the database for every query. Extract mode stores the database information into memory through a Tableau data engine. In this project, a live connection was not primarily used because the database of interest held static data.

**Connecting to the data source**: This is the metric that captures how long it takes to connect to MySQL server every time you want a new data source.

**Geocoding**: This metric captures how long it takes to compute the longitude and latitudes for the values already in the data.

**Computing Layout**: This metric is specific to creating a map visualization. It computes the time is takes to create the layout.

# **Testing**

To do the performance analysis, a new workbook was created, and the performance recording setting was initialized ( HELP > Settings and Performance > Start Performance Recording). After turning the recording on, the workbook was connected to the data sources and rendered in the data. While a live connection was not actually needed to capture the census database accurately, the differences between live and extract connections were recorded for the smallest and largest data inputs.

Then, the performance was recorded on extract data on four ascending increments of 500,000 records from ~500,000-2,000,000. The data was retrieved in the same order as the order of extracted data files. Then a series of visuals was created and two dashboards that compiled a few visuals. Filters were added to the dashboards to measure the run time on changing of the visuals as well, which is counted as executing query time. The same variables for each test were used in these filters. For example, for the State Demographics, Alabama is selected for all tests and for the Transportation table, Bicycle was selected for all tests.

The following tableau workbooks were created to test the performance and visuals:

* Live Person1 500,000
* Extract person1 500000
* Performance Recording extract person1 500000
* Performance Recording live person1 500000
* Person1 with 1000000
* Performance Recording person1 1000000
* Person1 with 1500000
* Performance Recording person1 1500000
* Live Person1 2000000
* Performance Recording live person1 2000000
* Extract person1 2000000 Final v2
* Performance Recording extract person1 2000000 final v2

# **Results**

The following table compiles the performance results for each metric tested under each size and connection condition:

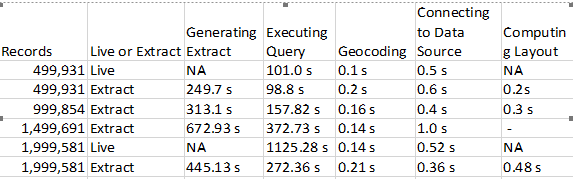
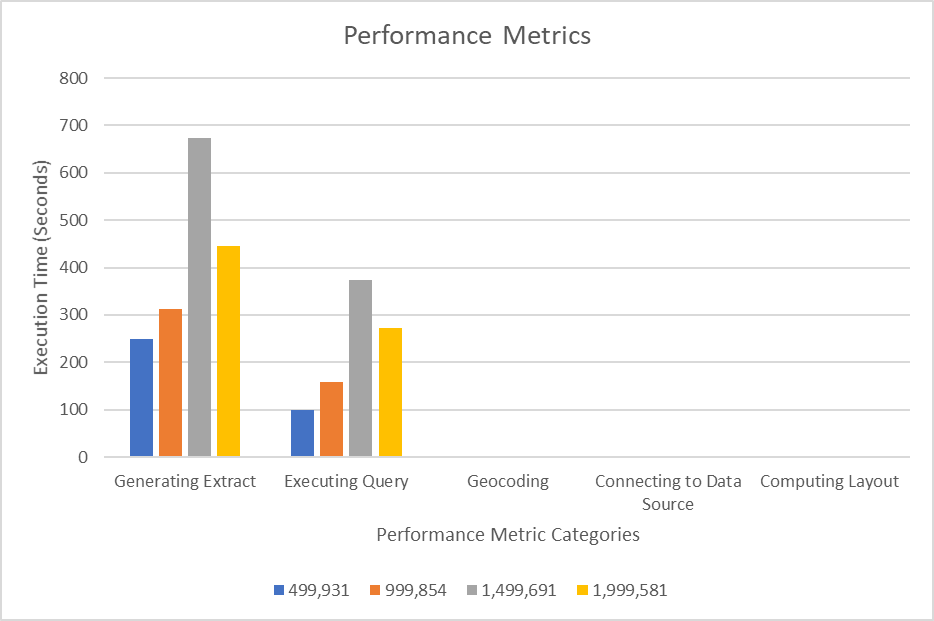


Figure 3. A visual representation of the performance metrics for the database at each of the four database sizes.



The graph of performance metrics in Figure 3 emphasizes which actions have the greatest cost in terms of performance. The execution time for the metrics geocoding, connecting to data source, and computing layout were all less than or equal to 1 second. These times are so small compared to the generating extract and executing query metrics that they are not visible on a scale of 100 seconds. This is evidence that generating an extract and executing a query have the greatest potential to slow down performance.

# **conclusion**

Based on the generating extract and executing query metrics, in general the performance decreases with the increasing number of records. There was an unexpected speed up in performance from 1.5 to 2 million records. However, this did not disrupt the natural trend upward in seconds/execution. While the performance of Tableau appeared sensitive to increasing number of records, 2 million records did not conclusively represent a ‘breaking point’ for Tableau performance, but rather hints at the potential for a large slow down. At most this project supports a need for further testing of Tableau performance. If this project were optimized, more increments of record increases would be included as well as multiple replications for each tested unit.

# **lessons learned**

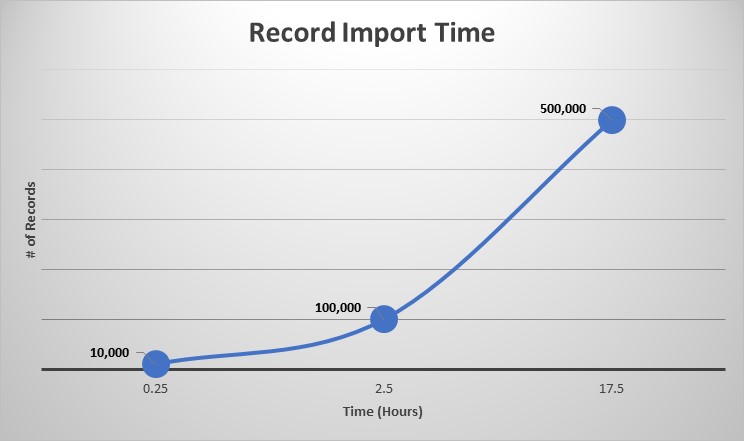
**T**he project scope at the start was to execute and identify Tableau performance bottlenecks with big data files (10M+ records). However, we have learned that, this effort will require infrastructure, knowledge and experience to complete (preferably on cloud S3 or NoSQL on AWS or Azure) that extend beyond this class.

Working with “Big Data” is difficult. With 298 variables and an infinite amount of records at our disposal, it took some time to get our hands around the data and how we were going to use it. This data could not be opened in MS Excel or Access since it had more than 255 columns. We found a csv splitter tool (<https://sourceforge.net/projects/splitcsv/>) that allowed us to split these huge files with more than a million records each into smaller, easier to manage files to help us get started.

Not being able to work in a shared server environment forced us to be creative about how we shared information. We quickly found that loading data into our database became a time-consuming process. As shown in Figure 3., you can see that the larger the amount of records we loaded, the longer it took to load data at an exponential rate. We were forced to share the database by taking a “data dump” using the tool native to MySQL, share the “dump” file folder on Google Drive, then the other person would have to use the tool to import the “dump” into their database. Once the data was loaded into the MySQL database, this process did not take much time and allowed us to share in the responsibilities for completing our tests.

We found that when loading the data through the import tool there were some records that would be left out due to a constraint in our database. If we used a data loading script, we could have developed a code in that script to capture the records that were rejected. Without this, it would be very difficult to identify why these records were not loaded and take steps to clean-up any data issues. This was another unexpected issue we found working with a “Big Data” set.

Several attempts to insert data with more efficiency were made with observed issues. *Using INSERT INTO <table name><fields> VALUES <data>.* This process is very cumbersome due to the large file size and number of fields. Insertion of records will require looping the data as many times as number of records, and was abandoned after observing very poor insertion performance. Using Panda package *(df.to\_sql(name='mytest3', con=engine, if\_exists='append', index=False):* This method worked well for smaller data sets (this method can create a new table on the fly, if doesn’t exist or can append the data for the existing table). However, the laptop with 8MB RAM consumed 100% memory when 10K records file attempted. So, the conclusion was to not to use this method, though the method could logically work.



***Figure 3:*** *Load times of People Census Data by number of records to MySQL tables. The graph shows that the larger the number of records loaded at a time, the longer the load took to run. For 10k to 100k, there was a 1:1 ratio, for 100k to 500k, the ratio rose to 1.4:1.*

**Appendix**

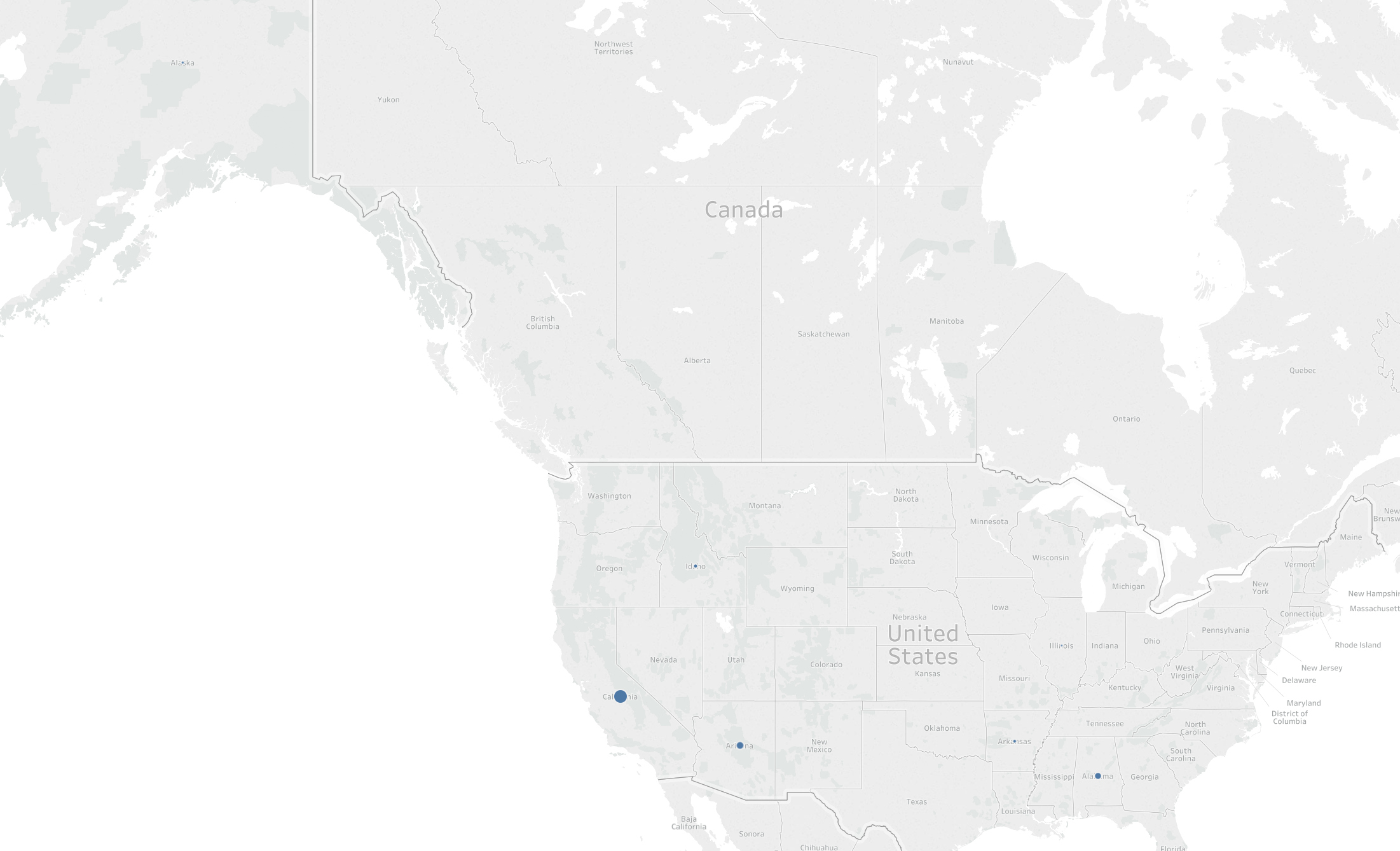
**SQL Script Files**

****

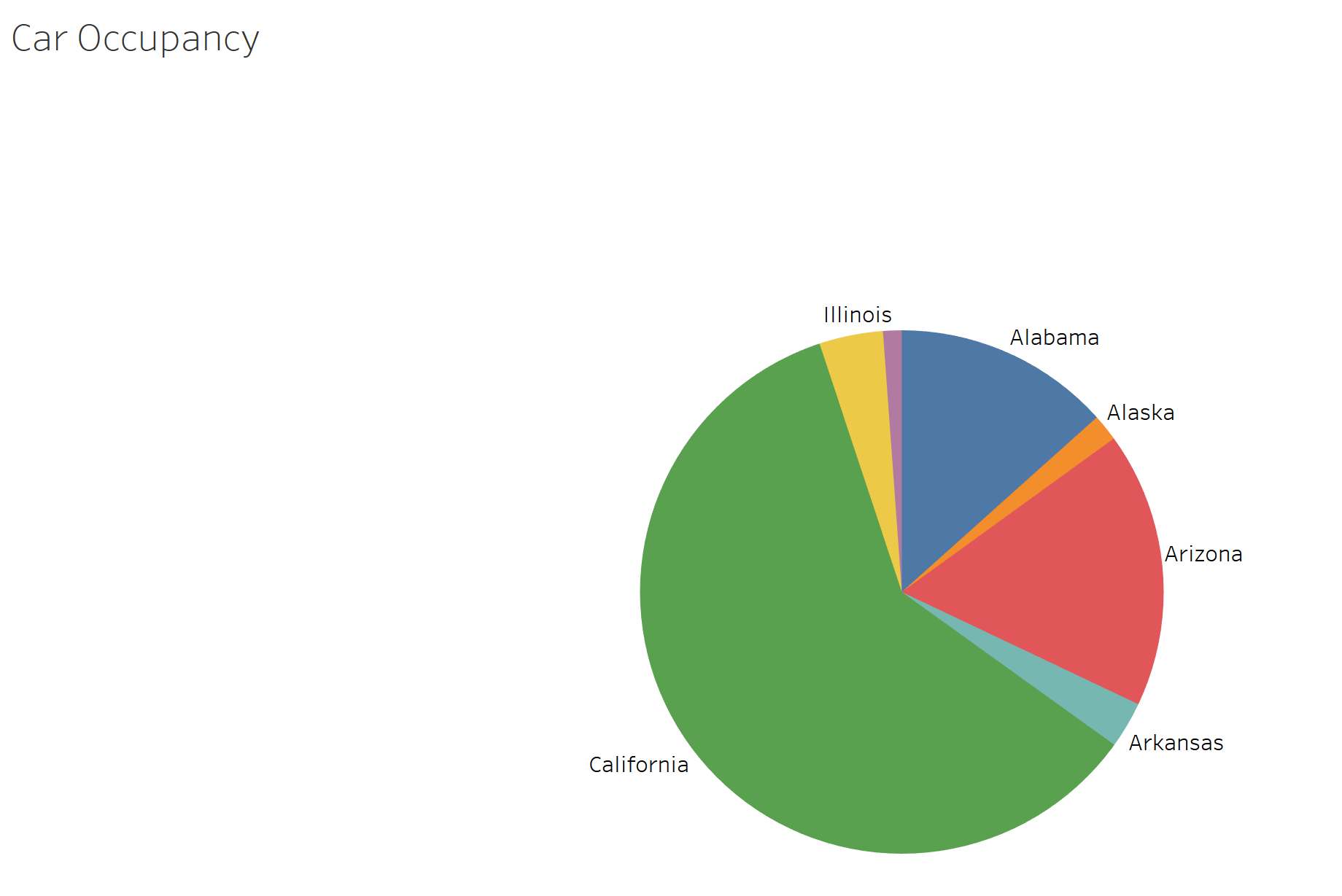
* **01\_Create\_Census\_Schema.sql – This script creates the Census database schema and the tables that will be used in that schema, establishing primary and foreign keys and indexing.**
* **02\_alex\_data\_dump\_script.sql – This script was used in creating a temporary schema and tables that would be transferred to our main database to be used for Tableau reporting.**
* **03\_lauren\_data\_dump\_script.sql – This script was used in creating a temporary schema and tables that would be transferred to our main database to be used for Tableau reporting.**
* **04\_Append\_100k\_records\_to\_person\_at\_a\_time.sql – This script was used to append 100,000 records to our main table to test Tableau reporting performance.**
* **05\_duplicate\_fix.sql – This script was used to fix an issue we found at the last minute when data with duplicate primary keys were being loaded to our main person table.**

**Visuals created during tableau performance testing:**

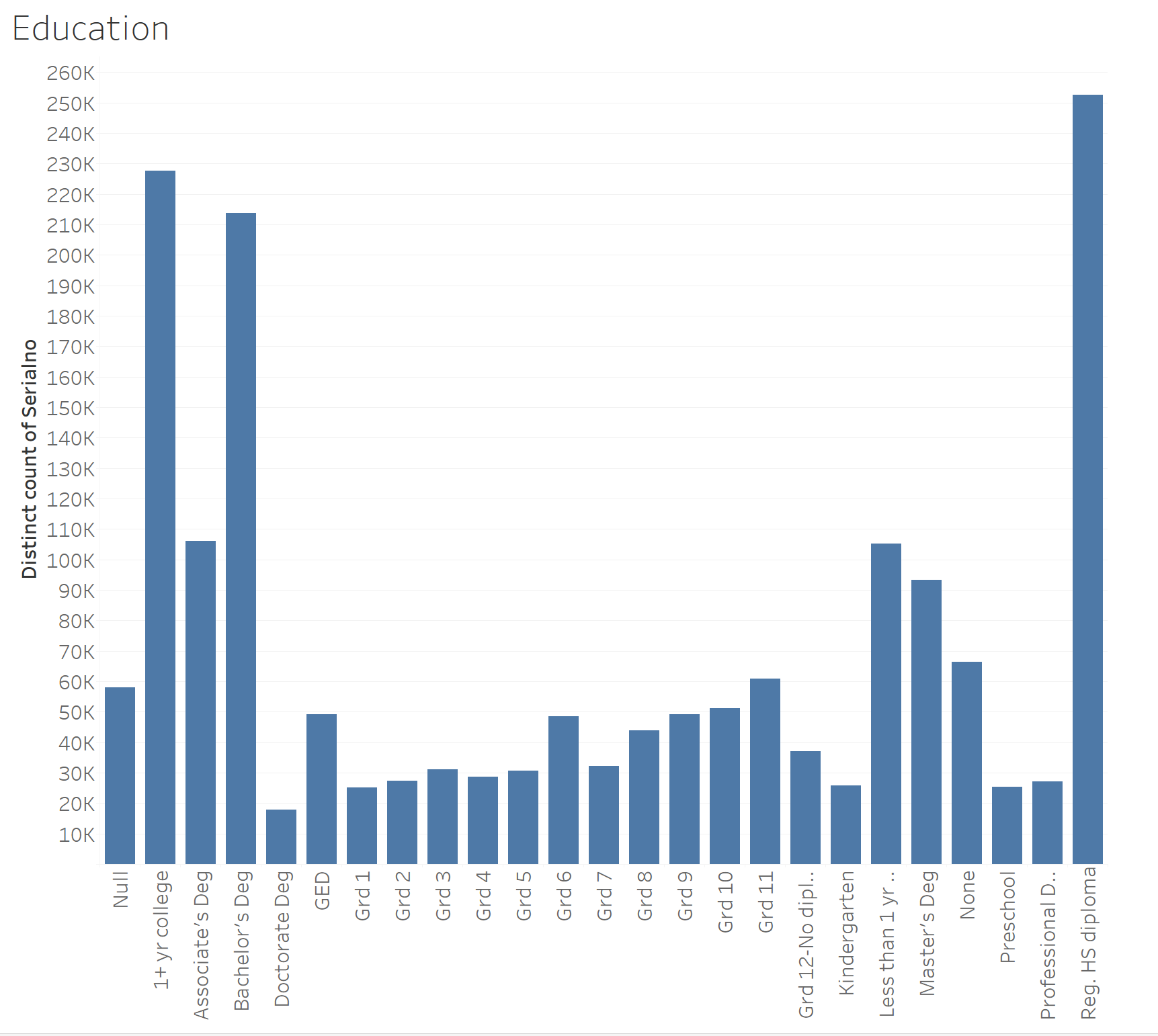
**Resident per State:** Looking at the distinct count of people that live in each state.



**Car Occupancy:** looking at what state has the most of each car occupancy. With the ability to filter to look at each type of occupancy



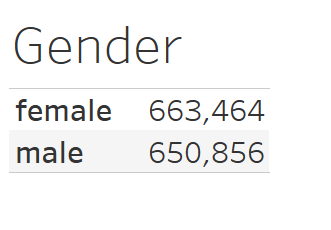
**Education:** Looking at the amount of residents that have a certain education by each state. (filter on the state)



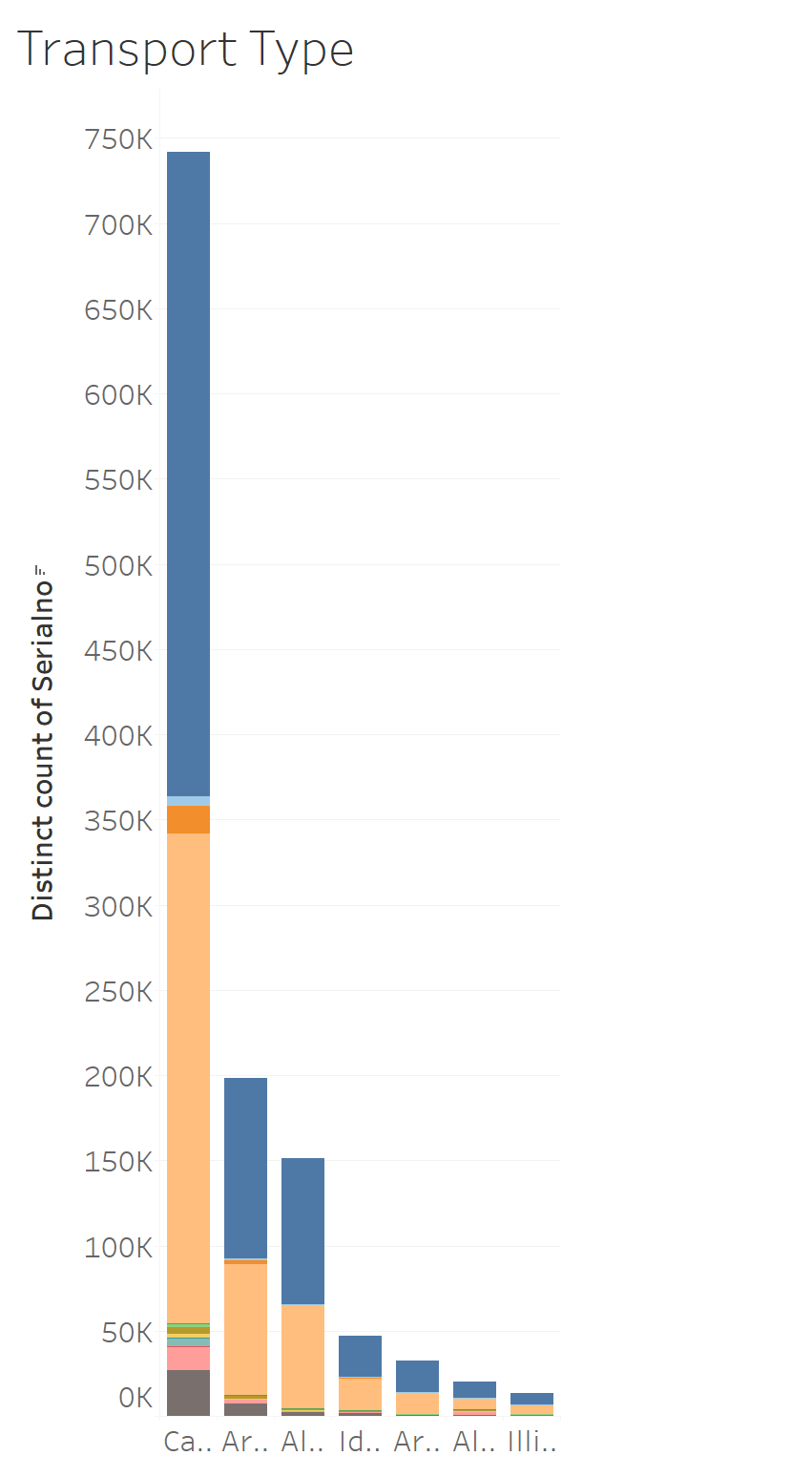
**Marital Status:** Looking at the spread of marital status for each state.



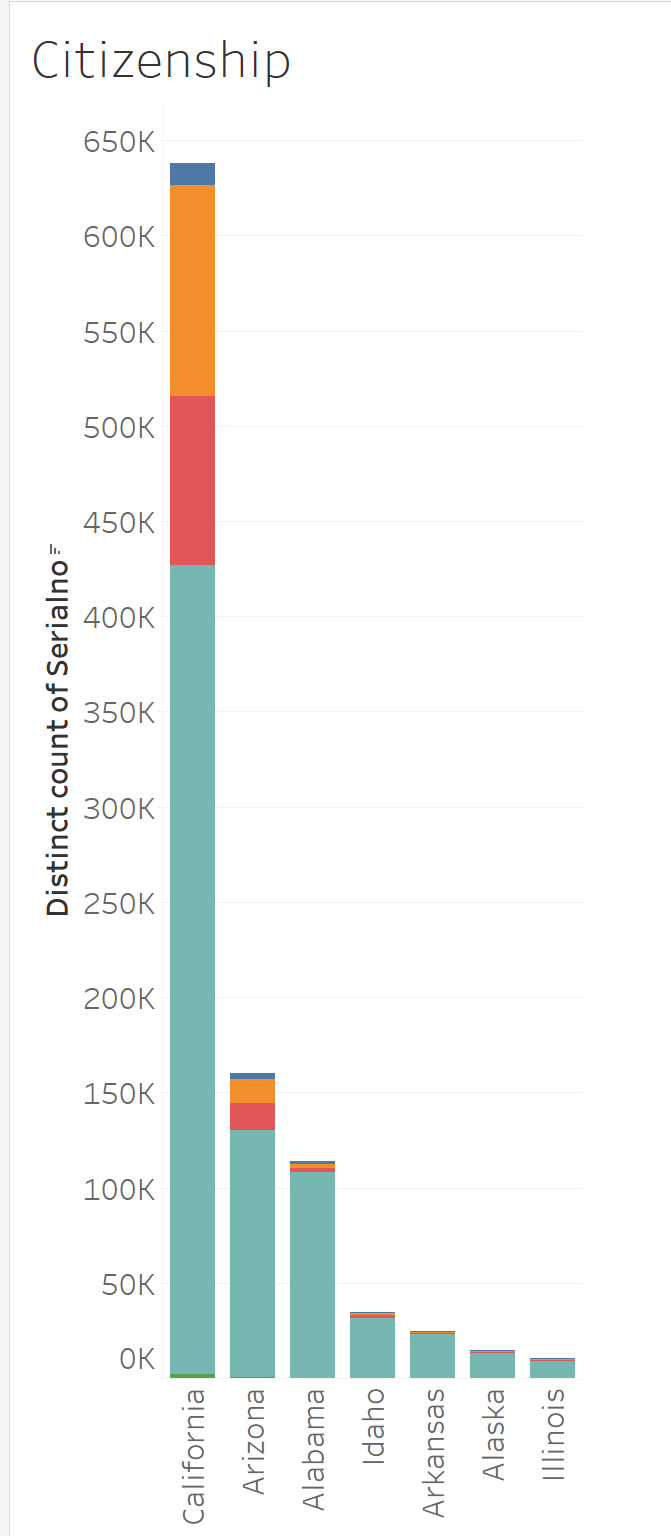
**Gender:** The amount of females or males in a given State.



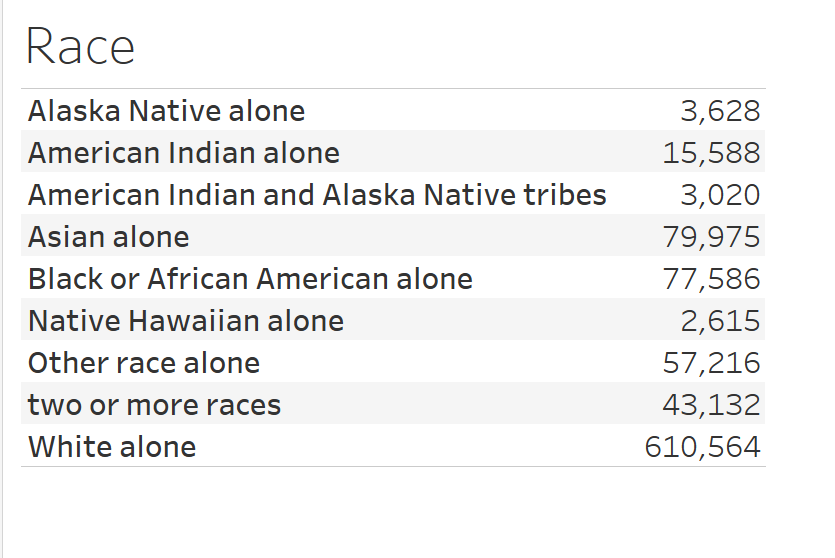
**Transport Type:** Looking at the different modes of transportation for the States.



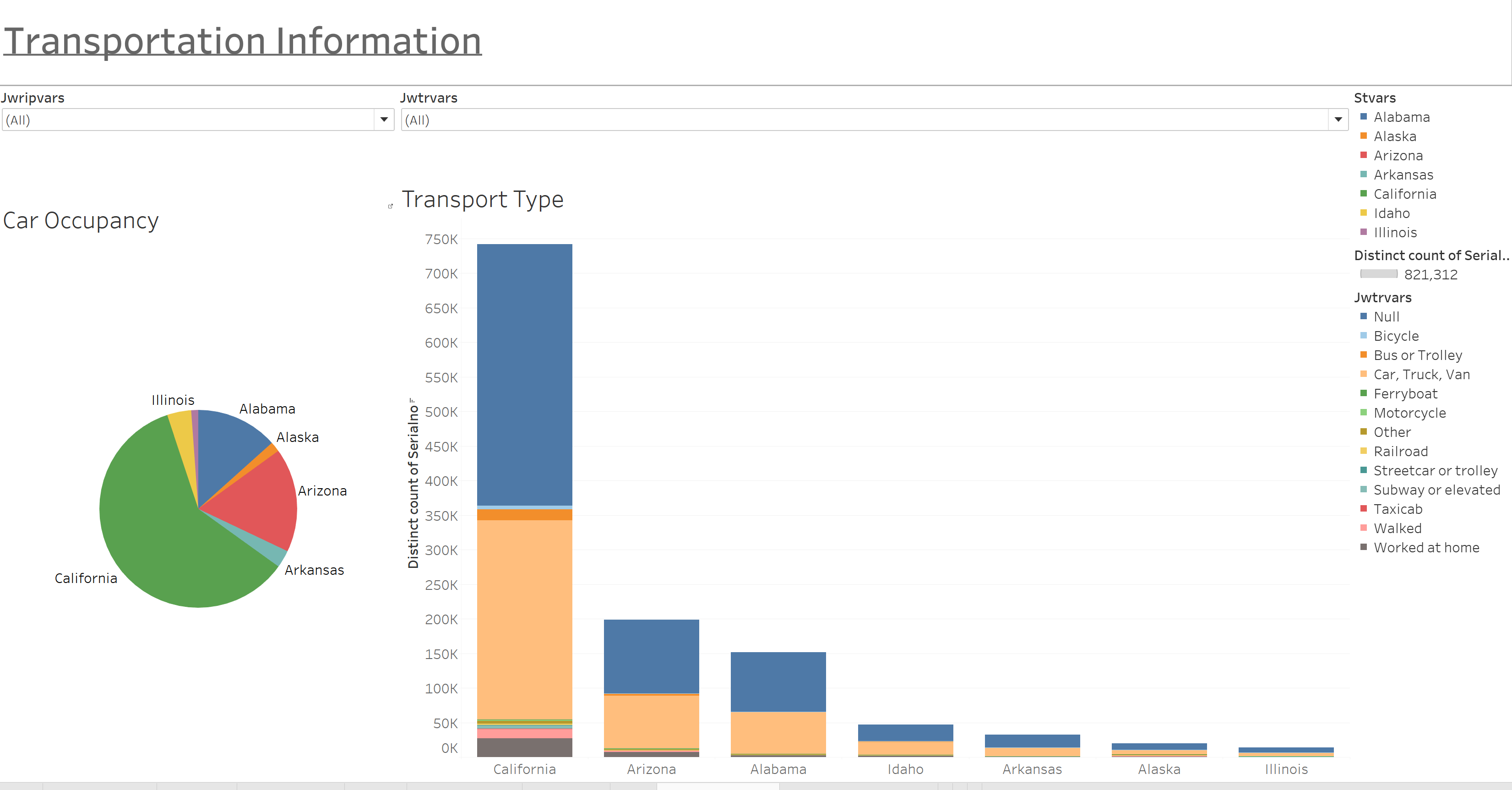
**Citizenship:** Look at what are the different citizenships that the resident of each State has.



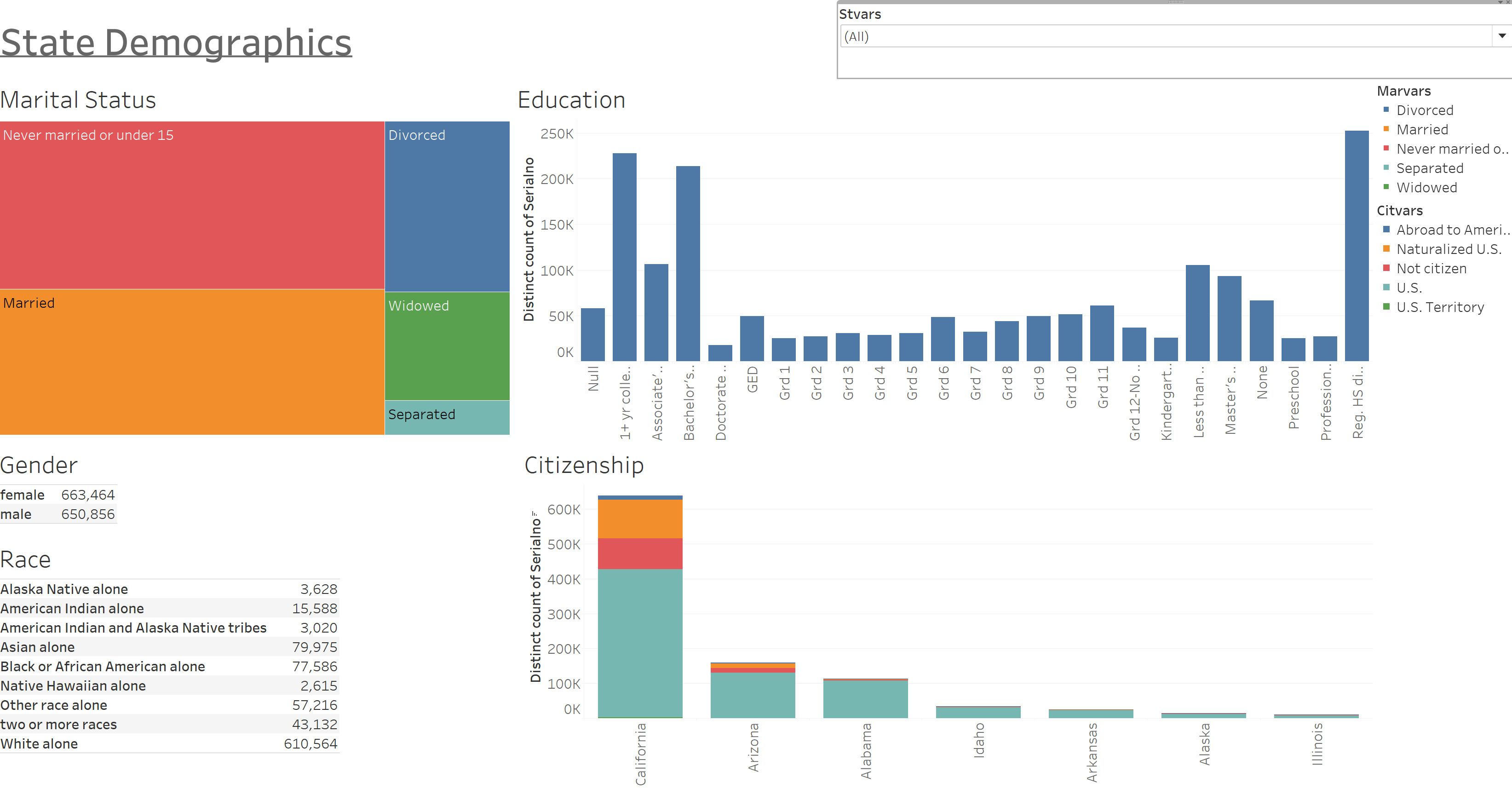
**Race:** The number of residents for each race within a given State.



**Transportation dashboard:** Contains the car occupancy worksheet and the transport type worksheet. There is a filter on car occupancy to see the amount of residents in each state has the selected car occupancy. The filter on Transport type lets you see the amount of residents in each state for the given transportation mode. When using these filters and changing the visuals we accrue executing query time.



**State Demographics dashboard:** This dashboard contains marital status, education, gender, race and citizenship information for the States. There is a State filter that will filter the data in the marital status, gender, education and race visuals, this time is included in the executing query time as well.



Note: On all of the visuals minus the resident per state visual we had to blend the data within Tableau using the primary and foreign keys that we established in our database schema.

**References**

1. Tableau, 2017. <https://www.tableau.com/about>

2. Trefis Team, 2015. *A Closer Look at Tableau’s Customer Base Growth.*

<https://www.forbes.com/sites/greatspeculations/2015/04/15/a-closer-look-at-tableaus-customer-base-growth/#4720765a7be2>

3. Arthur, Lisa. 2013. *What Is Big Data? https://www.forbes.com/sites/lisaarthur/2013/08/15/what-is-big-data/#16cb80f45c85*