Syracuse University

School of Information Studies

M.S. Applied Data Science

Portfolio Milestone

Daniel Caley

SUID: 352784643

Github: [Masters Applied Data Science Portfolio]

Table of Contents

My Data Science Story

When Michael Burry, the hedge fund manager who combed through mountains and mountains of mortgage data and foresaw the worst economic crisis since the great depression, I knew then that I wanted to be a Data Scientist. At the time though I didn't know what the field was called, let alone if dedicated roles like that existed. So, I pursued a trading position because that what Michael was doing. I went to work for a company at ground zero of the financial crises, I decided I wanted to run towards the burning building, instead of away. I learned SQL, Advance Excel Techniques, and VBA to scrub, analyze, and bring value to the mortgage market. As I grew in the role, I realized that the Fantasy of becoming a trader did not meet up to the reality. I swallowed a tough pill and looked for companies more into Data. There at my lowest moment in my Career I found a Data Analytics role and my journey truly begun in the field.

I absolutely love every minute of obtaining, scrubbing, transforming, and visualizing data. I felt like the fantasy finally met up to the reality, but a sweeping fear came across me. I didn't know Machine Learning, I didn't understand Bayesian Hypothesis testing, I didn't appreciate base statistics as I should have. The Inferential and Predictive Statistics was lacking, and I feared that I would get stuck and forgotten in the world that I loved so much. There and then I understood that I had to lean into my passion, and I applied for a master's in applied data science at Syracuse University.

Self-Perceived Barriers

Coding didn't seem like something my future would hold. I looked to Software Engineers, not analysts, to discourage my ability to want to learn coding. Through my career and Syracuse, I was able to demystify this barrier and really unlock a lot of the powers of these tools. This led to a snowball effect where Libraries and Packages became these easy things to load and apply on Data. This too led to using advance statistics, Bayesian theory, Linear Regression, P-Values, and all of these terms that felt so scarry I now had confidence.

Applied Data Science Program

The Applied Data Science program at Syracuse University's School of Information Studies equips students with the opportunity to develop analytic, technical, and managerial skills in order to contribute measurable impacts in a highly competitive job market. The focus of the program is encapsulated in the word "Applied": students are taught to collect, manage, analyze and develop insights that can be communicated clearly to a broad audience of interested parties. Through courses such as Data Administration Concepts & Database Management (IST 659), Data Analytics (IST 707), Scripting for Data Analysis (IST 652) and Advanced Database Management (IST 769), I learned basic data structure (from byte to bit); how relational databases are built using SQL, Excel and Microsoft Access, meeting the business needs of organizations; how data mining techniques can be used to solve classification problems using R Studio; and,

scripting in Python to explore a dataset to gain insights; using visualization to explore data and to communicate findings to a broad spectrum of audiences.

The Applied Data Science Program has seven learning objectives which were exemplified by the applications in this portfolio:

1. Describe a broad overview of the major practice areas in data science.
2. Collect and organize data.
3. Identify patterns in data via visualization, statistical analysis, and data mining.
4. Develop alternative strategies based on the data.
5. Develop a plan of action to implement the business decisions derived from the analyses.
6. Demonstrate communication skills regarding data and its analysis for relevant professionals in their organization.
7. Synthesize the ethical dimensions of data science practice.

Project 1 – IST 659 Data Administration Concepts & Database Management

Course Description

IST 659 is an introductory course to database management systems. This course examines data structures, file organizations, concepts, and principles of database management systems (DBMS) as well as data analysis, database design, data modeling, database management, and database implementation. More specifically, it introduces hierarchical, network, and relational data models; entity-relationship modeling; basics of Structured Query Language (SQL); data normalization; and database design. Using Microsoft's Access and SQL Server DBMSs as implementation vehicles, this course provides hands-on experience in database design and implementation through assignments, lab exercises, and course projects. This course also introduces advanced database concepts such as transaction management and concurrency control, distributed databases, multitier client/server architectures, web-based database applications, data warehousing, and NoSQL.

Learning Objectives:

After taking this course, the students will be able to:

- Describe fundamental data and database concepts
- Explain and use the database development lifecycle
- Create databases and database objects using popular database management system products
- Solve problems by constructing database queries using Structured Query Language (SQL)
- Design databases using data modeling and data normalization techniques
- Develop insights into future data management tool and technique trends
- Recommend and justify strategies for managing data security, privacy, audit/control, fraud detection, backup and recovery
- Critique the effectiveness of DBMS in computer information systems

Deliverables(s) – SQL Database

Created a research focus database that looked at real data from the census bureau and CDC covid data. The goal of the project to dive into the socioeconomic impact of COVID on the American society.

I created a fictional non-profit that needed this data in a structured way in order to further this research. First a conceptual model of the database was created followed by a logical model with the objective of understanding how the data all ties together.

SQL Server Management Studio was used to create the tables and populate the data, all using SQL. Personalized views were constructed for the non-profit to easily access whats important to them, like average covid rates by state or ethnicity.

Project Process & Development

Project Summary:

A non for profit wants to understand the impact of COVID related cases and deaths across America. Currently they have daily csv files by county of the number of COVID cases and deaths starting from May 2020 until present. They are finding that they need all this data in one place and simply aggregating the daily csv files into a single excel file is not scalable to perform analysis. In addition to their COVID data, they want to bring in additional Census Bureau data to help make their analysis even richer. The data includes population, household median income, and race by ZCTA. They will also need the ability to map ZCTA to Counties, Cities, and State. The final outcome for this project will allow workers of the non-profit to have all of their data in one place in order to perform analysis.

Stakeholders:

The stakeholders in this project are the non-profit who is interested in the impact of COVID on race and socioeconomic status in America. Inside of the non-profit the analysts working there will use the data to create presentations and research material surrounding the effects of the Coronavirus on different segments of the US population. Ultimately to help inform businesses, local municipalities, and news organizations that are in need of additional funding and to help combat the coronavirus impact on America.

High Level Rules:

- COVID cases and deaths are broken out daily by County. They only report daily, not twice a day.

- Cases and deaths might not be reported if there was no cases and death on that day for that county. The CSV by the non- profit will fill in the missing days with 0 cases and deaths by county.
- Census Bureau data is created at the A. This is different from postal Zip Code. There can be many Zip Codes underlying a ZCTA. We are not working with Zip Code in this dataset so we will not have to worry. Just need to know that these are 2 different ways to break out County level data.
- A Geo County State map will be required to connect the County to ZCTA.
    - The Geo County State map will include a county id and State.
    - The hierarchy for the multiple geographical attributes in this table are the following, from lowest level to the top:
        1. ZCTA
        2. City
        3. County
        4. State
    - Said differently there can be many ZCTA underlying a City, there can be many Cities underlying a County, and there can be many Counties underlying a State.
- The Census Bureau provides population by age group and by ZCTA.
    - Each ZCTA will have multiple age groups and consist of 6 groups.
- The Census Bureau also provides Household Median Income by ZCTA.
- The Census Bureau also has race information by ZCTA. The race information provided in a race category. The Race categories consist of White, Asian, Black or African American, and Other.

Details of data needed and maintenance:

The following data will be needed:

1. COVID data will be needed and is tracked by the non-profit. They will be responsible for creating the CSV files. They will also be responsible for making sure that all counties exist in the CSV file with a 0 value for cases and deaths in the event there are none. The data engineer, myself, will be responsible for uploading the data into the database on a daily basis.
2. The Census Bureau data is ingested from an API. This data is updated yearly at the end of Quarter 1 or the beginning of Quarter 2. It's the responsibility of the Data Engineer to swap out the previous year's data with the current.
3. The non for profit does not want stale data from the Census Bureau and only wants the new data.

Glossary

A **ZCTA** stands for Zip Code Tabulation Areas and are generalized areal representations of United States Postal Service Zip Code service areas. ZCTA are trademarked by the U.S. Census Bureau. The hierarchy is that one or many Zip Codes can underly a ZCTA.

A **City** an inhabited place of greater size, population or importance. In the hierarchy many zip codes can exist inside of a city.

A **County** is political and administrative division of a state, providing certain local governmental services. There are many cities to a county.

A **State** is a territory considered as an organized political community under one government. There are many counties to a single State.

Household Median Income is the income level earned by a given household. The formal definition is the income cut-off where half of the households earn ore, and half earn less. In this case Household

Median Income is a widely used metric when defining socioeconomic areas in business and non for profits.

Population is all inhabitants of a particular Zip Code, ZCTA, city, county, or state.

Race is a grouping of people who identify with each other on the basis of shared attributes that distinguish them from other groups such as a common set of traditions, ancestry, language, history, society or skin color. For the census data Race consists of 6 groups. They are the following:

1. White alone
2. Black or African American alone
3. American Indian and Alaska Native alone
4. Asian alone
5. Native Hawaiian and Other Pacific Islander alone
6. Other Race alone

Age Group consist of 23 groups. They are the following:

1. Under 5 years
2. 5 to 9 years
3. 10 to 14 years
4. 15 to 17 years
5. 18 and 19 years
6. 20 years
7. 21 years
8. 22 to 24 years
9. 25 to 29 years
10. 30 to 34 years

11. 35 to 39 years
12. 40 to 44 years
13. 45 to 49 years
14. 50 to 54 years
15. 55 to 59 years
16. 60 and 61 years
17. 62 to 64 years
18. 65 and 66 years
19. 67 to 69 years
20. 70 to 74 years
21. 75 to 79 years
22. 80 to 84 years
23. 85 years and over

Cases refer to the number of COVID related cases.

Deaths refer to the number of COVID related deaths.

COVID a highly contagious respiratory disease caused by the SARS-CoV-2 virus. Also known for causing a pandemic across the world that has disrupt economies, impacted a way of life, and caused many deaths.

Data Questions

1. What is the trailing 7-day COVID cases and deaths from May 2020 until current?
    - Calculate the trailing 7-day cases divided by population.
    - Calculate the trailing 7-day deaths divided by population.
2. What is the Population and Household Median Income for the ZCTA 85203, Mesa AZ.

- This should be by ZCTA, County, & State
- What Percentage Lives in 85203, Maricopa vs the state?

3. What is the total population By County, Include State as a column?

    - What is the top 25 populated counties?
    - What is the trailing 7 days COVID cases divided by population for the top 25 counties?

4. What is the weighted average household median income by County?

    - What's the top 25 highest weighted average household median income by population?
    - What is the total population of these counties?
    - What's the bottom 25 weighted average household median income by population?
    - What is the trailing 7-day COVID cases divide by population?
    - What does the top 25 highest weighted average household median income look compare to the bottom 25?

5. For ZCTA 85203, Arizona, what is the population broken out by age group?
    - How does that compare to the County?
    - How does that compare to the State?
6. What where the top 25 Counties who had the highest deaths by percentage of population?
    - The population must be over 1000 people.
    - Include their household median income.
    - Not looking at New York
7. Group the age groups that are considered the high-risk age categories for COVID. That is age groups over 65 and under 5.

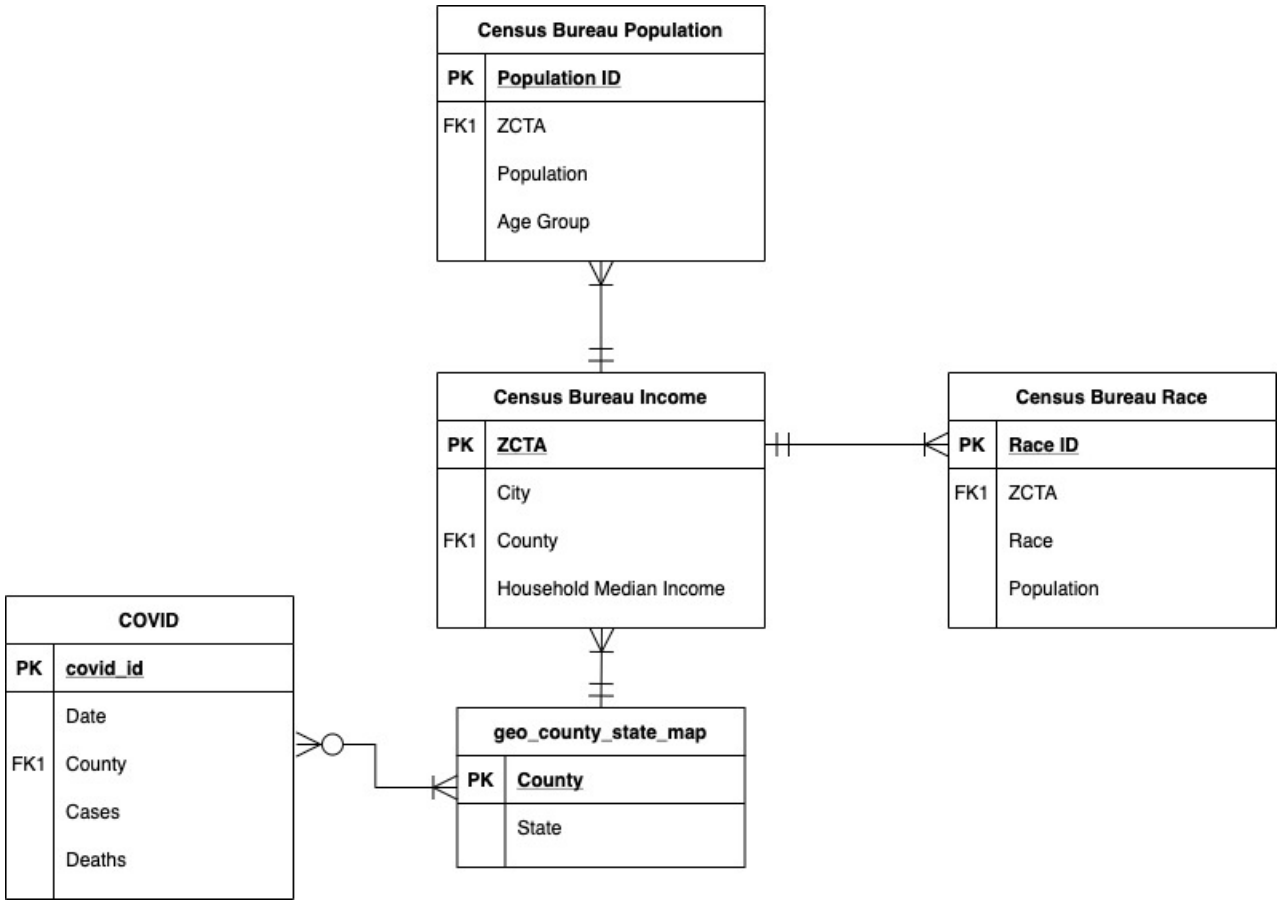
    - Make this a calculated field that high risk or low risk.

- Total the population by high risk and low risk by county.
- Any counties with a population that is greater than 30% of the total population what is the number of COVID cases and deaths by total population vs low risk counties.
- Group these counties into 2 rows, high risk and low risk. How do they total cases and deaths compare to each other?

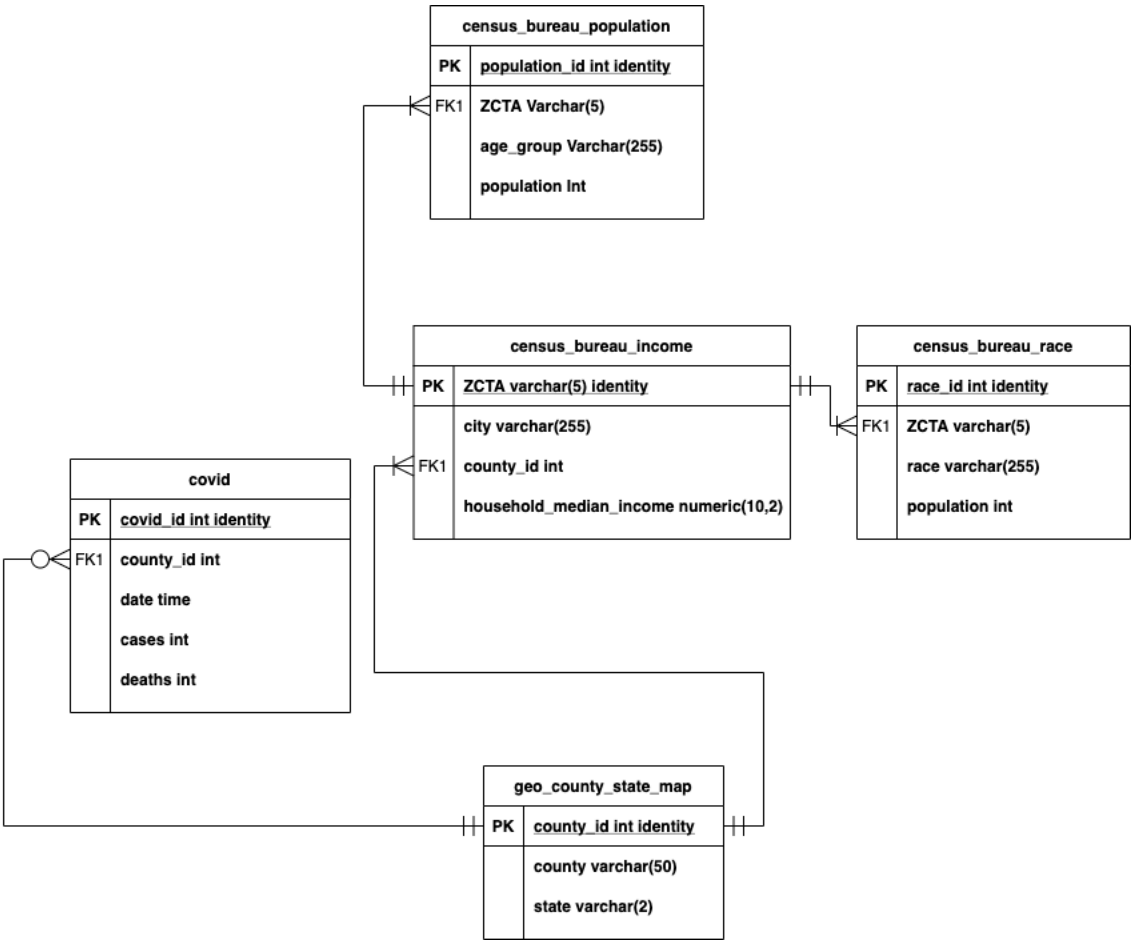
8. How many Deaths Vs. Total Population

- Compare 2 rows below 30% and above 30% and how do the total cases and deaths by population compare to each other?

Conceptual Model



Logical Model



Physical Database Design

```
-- Creating Tables, Views, procedure, and functions.
-- Creating Drops

--Drop Tables
Drop table if exists dc_census_bureau_population
go

drop table if exists dc_census_bureau_race
go

Drop table if exists dc_covid
go

Drop table if exists dc_census_bureau_income
go

Drop table if exists dc_geo_county_state_map
go



-- Creating database tables
-- Creating daily covid & death table
CREATE TABLE dc_geo_county_state_map(
	-- Creating Columns
	county_id varchar(5),
	county varchar(255) NOT NULL,
	state varchar(2) NOT NULL,
	-- Applying Constraints
	CONSTRAINT PK_dc_geo_county_state_map PRIMARY KEY (county_id)
	)
go


CREATE TABLE dc_covid (
	-- Creating Columns
	covid_id varchar(10),
	county_id varchar(5) NOT NULL,
	date date NOT NULL,
	cases int NOT NULL,
	deaths int NOT NULL,
	-- Applying Constraints
	CONSTRAINT PK_dc_covid PRIMARY KEY (covid_id),
	CONSTRAINT FK1_dc_covid FOREIGN KEY (county_id) REFERENCES
		dc_geo_county_state_map(county_id)
	)
go

CREATE TABLE dc_census_bureau_income (
	-- Creating Columns
	zcta varchar(5),
	county_id varchar(5) NOT NULL,
```

```
	household_median_income numeric(10,2),
	-- Applying Constraints
	CONSTRAINT PK_dc_census_bureau_income PRIMARY KEY (zcta),
	CONSTRAINT FK_dc_census_bureau_income FOREIGN KEY (county_id) REFERENCES
	dc_geo_county_state_map(county_id)
	)
go

CREATE TABLE dc_census_bureau_race (
	-- Creating Columns
	race_id varchar(8),
	zcta varchar(5),
	race varchar(255),
	population int,
	-- Applying Constraints
	CONSTRAINT PK_dc_census_bureau_race PRIMARY KEY (race_id),
	CONSTRAINT FK_dc_census_bureau_race FOREIGN KEY (zcta) REFERENCES
	dc_census_bureau_income(zcta)
	)
go

CREATE TABLE dc_census_bureau_population (
	-- Creating Columns
	population_id varchar(7),
	zcta varchar(5),
	age_group varchar(255),
	population int
	-- Applying Constraints
	CONSTRAINT PK_dc_bureau_population PRIMARY KEY (population_id),
	CONSTRAINT FK_dc_bureau_population FOREIGN KEY (zcta) REFERENCES
	dc_census_bureau_income(zcta)
	)
go


--- Inserting records into tables.  Total recoreds is over 1 million.
--- Therefore I will load the records through a bulk upload process.
--- The below will only show 5 records for each table.

--- Inserting Records for geo_county_state_map
INSERT INTO dc_geo_county_state_map(county_id, county, state)
VALUES  ('01001', 'Autauga County',  'AL'),
		('01003', 'Baldwin County',  'AL'),
		('01005', 'Barbour County',  'AL'),
		('01007', 'Bibb County',  'AL'),
		('01009', 'Blount County',  'AL')
go




-- Look to see if the insert worked
select * from dc_geo_county_state_map
```

Results | Messages

	county_id	county	state
1	01001	Autauga County	AL
2	01003	Baldwin County	AL
3	01005	Barbour County	AL
4	01007	Bibb County	AL
5	01009	Blount County	AL

```
--- Inserting Records for census_bureau_income
INSERT INTO dc_census_bureau_income(zcta, county_id, household_median_income)
VALUES  ('36003', '01001',  '37000'),
                        ('36480', '01003',  '27461'),
                        ('36005', '01005',  '49722'),
                        ('35034', '01007',  '39087'),
                        ('35013', '01009',  '0')
go

-- Look to see if the insert worked
select * from dc_census_bureau_income
```

	zcta	county_id	household_median_income
1	35013	01009	0.00
2	35034	01007	39087.00
3	36003	01001	37000.00
4	36005	01005	49722.00
5	36480	01003	27461.00

```
--- Inserting Records for covid
INSERT INTO dc_covid(covid_id, county_id, date, cases, deaths)
VALUES  ('0000009481', '01001',  '2020-03-24', '1', '0'),
                        ('0000001935', '01003',  '2020-03-14', '1', '0'),
                        ('0000028399', '01005',  '2020-04-03', '1', '0'),
                        ('0000019658', '01007',  '2020-03-30', '2', '0'),
                        ('0000010838', '01009',  '2020-03-25', '1', '0')
go

-- Look to see if the insert worked
select * from dc_covid
```

	covid_id	county_id	date	cases	deaths
1	0000001935	01003	2020-03-14	1	0
2	0000009481	01001	2020-03-24	1	0
3	0000010838	01009	2020-03-25	1	0
4	0000019658	01007	2020-03-30	2	0
5	0000028399	01005	2020-04-03	1	0

```
--- Inserting Records for population
INSERT INTO dc_census_bureau_population(population_id, zcta, age_group, population)
VALUES  ('360031', '36003', 'Under 5 years',  '111'),
```

```
                ('364801', '36480', 'Under 5 years',  '28'),
                ('360051', '36005', 'Under 5 years',  '57'),
                ('350341', '35034', 'Under 5 years',  '547'),
                ('350131', '35013', 'Under 5 years',  '0')
go


-- Look to see if the insert worked
select * from dc_census_bureau_population
```

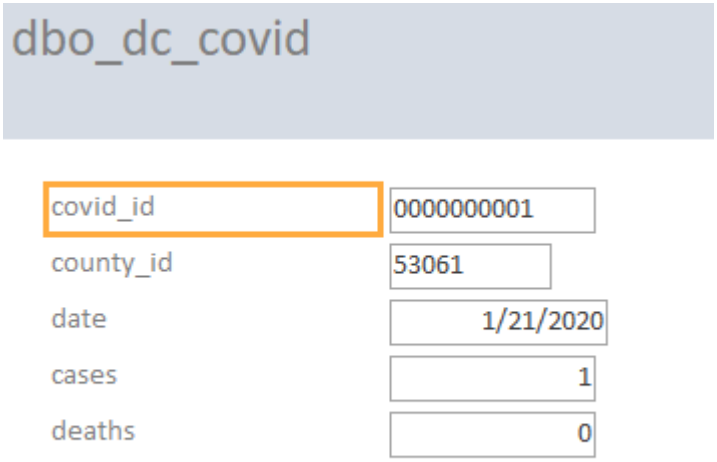
	population_id	zcta	age_group	population
1	350131	35013	Under 5 years	0
2	350341	35034	Under 5 years	547
3	360031	36003	Under 5 years	111
4	360051	36005	Under 5 years	57
5	364801	36480	Under 5 years	28

```
--- Inserting Records for race
INSERT INTO dc_census_bureau_race(race_id, zcta, race, population)
VALUES  ('36003101', '36003', 'Black or African American alone',  '1102'),
                ('36480100', '36480', 'White alone',  '1389'),
                ('36005102', '36005', 'American Indian and Alaska Native alone',  '0'),
                ('35034100', '35034', 'White alone',  '2942'),
                ('35013105', '35013', 'Other Race alone',  '0')
go

-- Look to see if the insert worked
select * from dc_census_bureau_race
```

	race_id	zcta	race	population
1	35013105	35013	Other Race alone	0
2	35034100	35034	White alone	2942
3	36003101	36003	Black or African American alone	1102
4	36005102	36005	American Indian and Alaska Native alone	0
5	36480100	36480	White alone	1389

Utilizing Access & Creating Procedures:

```
--- Form made from Access to help fix issues in the database
```




```
--- Creating a procedure to update a COVID cases based from the county id and date.
--- The first parameter is the user name for the user to change
--- The second is the new email address

CREATE PROCEDURE dc_ChangeCovidCases(@covid_cases int,@county_id varchar(5), @covid_date date)
AS
BEGIN
        UPDATE dc_covid SET cases = @covid_cases
        WHERE county_id = @county_id and date = @covid_date
END
GO
```

Results Messages

	covid_id	county_id	date	cases	deaths
1	0001141251	04013	2021-03-21	160	10

	covid_id	county_id	date	cases	deaths
1	0001141251	04013	2021-03-21	161	10

Creating Views and Functions:

```
-- Before answering the questions going to create a series of views and functions to help answer the questions.

--- Finding total covid cases and deaths by county.
drop view if exists dc_covid_county

go
CREATE VIEW dc_covid_county as
select dc_covid.county_id,
        dc_geo_county_state_map.county,
        dc_geo_county_state_map.state,
```

```
        sum(cases) total_cases,
        sum(deaths) total_deaths
from dc_covid
join dc_geo_county_state_map
on dc_covid.county_id = dc_geo_county_state_map.county_id
group by dc_covid.county_id,
                dc_geo_county_state_map.county,
                dc_geo_county_state_map.state
go
--- Checking to see if the view worked
select * from dc_covid_county
```

	county_id	county	state	total_cases	total_deaths
1	01009	Blount County	AL	6383	130
2	01011	Bullock County	AL	1194	39
3	01023	Choctaw County	AL	581	24
4	01029	Cleburne County	AL	1446	41
5	01049	DeKalb County	AL	8674	179
6	01053	Escambia County	AL	3839	74
7	01055	Etowah County	AL	13604	339
8	01063	Greene County	AL	897	33
9	01065	Hale County	AL	2147	72
10	01073	Jefferson County	AL	74079	1442
11	01087	Macon County	AL	1520	47
12	01099	Monroe County	AL	1692	39
13	01127	Walker County	AL	7015	269
14	01131	Wilcox County	AL	1247	26
15	02020	Anchorage Municipality	AK	27440	160
16	02050	Bethel Census Area	AK	3685	20
17	02068	Denali Borough	AK	78	0
18	02158	Kusilvak Census Area	AK	1217	3
19	02188	Northwest Arctic Borough	AK	595	2
20	02261	Valdez-Cordova Census Area	AK	521	2
21	04005	Coconino County	AZ	16952	323
22	04012	La Paz County	AZ	2430	74
23	05007	Benton County	AR	27946	399
24	05009	Boone County	AR	3713	78

```
--- Finding daily covid cases nation wide from May 2020 until current.
drop view if exists dc_daily_covid
go
CREATE VIEW dc_daily_covid as
        with daily_covid as (
                Select county_id,
                        date,
                        sum(cases) daily_cases,
                        sum(deaths) daily_deaths
                from dc_covid
                -- We have to go 7 days before May as we want a rolling 7 days.
                -- SQL will begin the sum but the first 6 records won't be a rolling 7 days
                where date between '20200424' and GETDATE()
                group by county_id, date
```

```
	),
	trailing_covid as (
		select *,
			sum(daily_cases) over (partition by county_id order by date rows between 6 preceding and
current row) t7d_cases,
			sum(daily_deaths) over (partition by county_id order by date rows between 6 preceding and
current row) t7d_deaths
		from daily_covid
	)
		select  * from  trailing_covid
		where date >= '20200501'
	go

--- Checking our view
select * from dc_daily_covid order by county_id, date
```

	county_id	date	daily_cases	daily_deaths	t7d_cases	t7d_deaths
1	01001	2020-05-01	0	-1	6	1
2	01001	2020-05-02	3	0	8	1
3	01001	2020-05-03	3	0	11	1
4	01001	2020-05-04	5	0	14	0
5	01001	2020-05-05	0	0	13	-1
6	01001	2020-05-06	5	0	15	-1
7	01001	2020-05-07	3	0	19	-1
8	01001	2020-05-08	6	1	25	1
9	01001	2020-05-09	1	0	23	1
10	01001	2020-05-10	6	0	26	1
11	01001	2020-05-11	10	0	31	1
12	01001	2020-05-12	7	0	38	1
13	01001	2020-05-13	2	0	35	1
14	01001	2020-05-14	11	0	43	1
15	01001	2020-05-15	-1	0	36	0
16	01001	2020-05-16	7	0	42	0
17	01001	2020-05-17	0	0	36	0
18	01001	2020-05-18	10	0	36	0
19	01001	2020-05-19	7	0	36	0
20	01001	2020-05-20	9	-1	43	-1
21	01001	2020-05-21	11	0	43	-1
22	01001	2020-05-22	2	0	46	-1
23	01001	2020-05-23	6	0	45	-1
24	01001	2020-05-24	4	0	49	-1

```
-- creating a zcta, county, state view with total population and median income
drop view if exists dc_geo_population
go
CREATE VIEW dc_geo_population as

	Select dc_census_bureau_income.zcta,
	    dc_census_bureau_income.county_id,
```

```
                dc_geo_county_state_map.county,
                dc_geo_county_state_map.state,
                sum(dc_census_bureau_population.population) population,
                dc_census_bureau_income.household_median_income
        from dc_census_bureau_income
                join dc_geo_county_state_map
                        on dc_census_bureau_income.county_id = dc_geo_county_state_map.county_id
                join dc_census_bureau_population
                        on dc_census_bureau_income.zcta = dc_census_bureau_population.zcta
        group by dc_census_bureau_income.zcta,
                        dc_census_bureau_income.county_id,
                        dc_geo_county_state_map.county,
                        dc_geo_county_state_map.state,
                        dc_census_bureau_income.household_median_income
go

-- Checking our view
select * from dc_geo_population
where state = 'NY' order by zcta
```

	zcta	county_id	county	state	population	household_median_income
1	06390	36103	Suffolk County	NY	125	61125.00
2	10001	36061	New York County	NY	24117	92840.00
3	10002	36061	New York County	NY	74479	36982.00
4	10003	36061	New York County	NY	53977	118161.00
5	10004	36061	New York County	NY	3335	190223.00
6	10005	36061	New York County	NY	8701	189702.00
7	10006	36061	New York County	NY	3092	179044.00
8	10007	36061	New York County	NY	7408	224063.00
9	10009	36061	New York County	NY	58293	63717.00
10	10010	36061	New York County	NY	35906	132988.00
11	10011	36061	New York County	NY	49949	138272.00
12	10012	36061	New York County	NY	23318	106467.00
13	10013	36061	New York County	NY	28799	113191.00
14	10014	36061	New York County	NY	30344	133501.00
15	10016	36061	New York County	NY	52886	126628.00
16	10017	36061	New York County	NY	15846	131045.00
17	10018	36061	New York County	NY	8806	122484.00
18	10019	36061	New York County	NY	45498	103792.00
19	10020	36061	New York County	NY	0	0.00
20	10021	36061	New York County	NY	44280	122169.00
21	10022	36061	New York County	NY	31130	150718.00
22	10023	36061	New York County	NY	62541	132605.00
23	10024	36061	New York County	NY	58102	143623.00
24	10025	36061	New York County	NY	92251	91624.00

```
-- Creating a view for age groups that includes a covid_age_type
-- and attributes like county and state
drop view if exists dc_age_group
```

```
go
create view dc_age_group as

select
	dc_census_bureau_population.zcta,
		case
			when age_group in ('Under 5 years','65 and 66 years',
								'67 to 69 years','70 to 74 years',
								'75 to 79 years','80 to 84 years',
								'85 years and over')
				then 'High Risk Age Group'
			else 'Low Risk Age Group'
				end as covid_age_group,
		dc_census_bureau_population.age_group,
		dc_geo_county_state_map.county_id,
		dc_geo_county_state_map.county,
		dc_geo_county_state_map.state,
		dc_census_bureau_population.population,
		case
			when age_group = '85 years and over'
				then 85
			else left(ltrim(right(age_group,8)),2)* 1
		end
			as sort_key
from dc_census_bureau_population
	left join dc_census_bureau_income
		on dc_census_bureau_population.zcta = dc_census_bureau_income.zcta
	left join dc_geo_county_state_map
		on dc_census_bureau_income.county_id = dc_geo_county_state_map.county_id
go

-- Checking our view

select * from dc_age_group
order by zcta, sort_key
```

	zcta	covid_age_group	age_group	county_id	county	state	population	sort_key
1	00601	High Risk Age Group	Under 5 years	72001	Adjuntas Municipio	PR	803	5
2	00601	Low Risk Age Group	5 to 9 years	72001	Adjuntas Municipio	PR	835	9
3	00601	Low Risk Age Group	10 to 14 years	72001	Adjuntas Municipio	PR	1270	14
4	00601	Low Risk Age Group	15 to 17 years	72001	Adjuntas Municipio	PR	652	17
5	00601	Low Risk Age Group	18 and 19 years	72001	Adjuntas Municipio	PR	479	19
6	00601	Low Risk Age Group	20 years	72001	Adjuntas Municipio	PR	216	20
7	00601	Low Risk Age Group	21 years	72001	Adjuntas Municipio	PR	253	21
8	00601	Low Risk Age Group	22 to 24 years	72001	Adjuntas Municipio	PR	671	24
9	00601	Low Risk Age Group	25 to 29 years	72001	Adjuntas Municipio	PR	1125	29
10	00601	Low Risk Age Group	30 to 34 years	72001	Adjuntas Municipio	PR	1012	34
11	00601	Low Risk Age Group	35 to 39 years	72001	Adjuntas Municipio	PR	825	39
12	00601	Low Risk Age Group	40 to 44 years	72001	Adjuntas Municipio	PR	1100	44
13	00601	Low Risk Age Group	45 to 49 years	72001	Adjuntas Municipio	PR	1138	49
14	00601	Low Risk Age Group	50 to 54 years	72001	Adjuntas Municipio	PR	1227	54
15	00601	Low Risk Age Group	55 to 59 years	72001	Adjuntas Municipio	PR	1110	59
16	00601	Low Risk Age Group	60 and 61 years	72001	Adjuntas Municipio	PR	555	61
17	00601	Low Risk Age Group	62 to 64 years	72001	Adjuntas Municipio	PR	731	64
18	00601	High Risk Age Group	65 and 66 years	72001	Adjuntas Municipio	PR	498	66
19	00601	High Risk Age Group	67 to 69 years	72001	Adjuntas Municipio	PR	640	69
20	00601	High Risk Age Group	70 to 74 years	72001	Adjuntas Municipio	PR	654	74
21	00601	High Risk Age Group	75 to 79 years	72001	Adjuntas Municipio	PR	582	79
22	00601	High Risk Age Group	80 to 84 years	72001	Adjuntas Municipio	PR	393	84
23	00601	High Risk Age Group	85 years and over	72001	Adjuntas Municipio	PR	344	85
24	00602	High Risk Age Group	Under 5 years	72003	Aguada Municipio	PR	1492	5

```
Drop function if exists dbo.dc_GetPopulation

go
CREATE FUNCTION dbo.dc_GetPopulation(@countyid varchar(5))
RETURNS INT AS
BEGIN
        DECLARE @Popcounty int

        SELECT @Popcounty = SUM(population) from dc_geo_population where county_id = @countyid

        RETURN @Popcounty
END
Go
--- Checking to see if the function works. Maricopa County is 04013
select top 1 county_id, dbo.dc_GetPopulation(county_id) county_population
from dc_covid
where county_id = '04013'
```

	county_id	county_population
1	04013	4401060

```
--- Create a function that, when given the zcta, returns the total population by County
Drop function if exists dbo.dc_ZCTA_CountyPop

go
CREATE FUNCTION dbo.dc_ZCTA_CountyPop(@zcta varchar(5))
RETURNS INT AS
BEGIN
	DECLARE @Countypop int

	SELECT @Countypop = SUM(population) from dc_geo_population where county_id = (select county_id from
dc_geo_population where zcta =  @zcta)

	RETURN @Countypop
END
Go

-- Checking to see if the function works. Using ZCTA 85203
select zcta, dbo.dc_ZCTA_CountyPop(zcta) county_population
from dc_geo_population
where zcta = '85203'
```

	zcta	county_population
1	85203	4401060

```
--- Create a function that, when given the zcta, returns the total population by State
Drop function if exists dbo.dc_ZCTA_StatePop

go
CREATE FUNCTION dbo.dc_ZCTA_StatePop(@zcta varchar(5))
RETURNS INT AS
BEGIN
	DECLARE @Statepop int

	SELECT @Statepop = SUM(population) from dc_geo_population where state = (select state from
dc_geo_population where zcta =  @zcta)

	RETURN @Statepop
END
Go

-- Checking to if the function works
Select zcta, dbo.dc_ZCTA_StatePop(zcta) state_population
from dc_geo_population
where zcta = '85203'
```

	zcta	state_population
1	85203	7057786

```
--- Create a function that, when given the zcta, returns the total Household Median Income by County
Drop function if exists dbo.dc_County_MedianIncome

go
CREATE FUNCTION dbo.dc_County_MedianIncome(@zcta varchar(5))
RETURNS INT AS
BEGIN
	DECLARE @CountyIncome numeric(10,2)
```

```
	SELECT @CountyIncome = SUM(household_median_income*population) / sum(population) from
dc_geo_population where county_id = (select county_id from dc_geo_population where zcta =  @zcta)

	RETURN @CountyIncome
END
Go

-- Checking to if the function works
Select zcta, household_median_income, dbo.dc_County_MedianIncome(zcta) County_MedianIncome
from dc_geo_population
where zcta = '85203'
```

	zcta	household_median_income	County_MedianIncome
1	85203	54919.00	68403

```
--- Create a function that, when given the zcta, returns the Household Median Income by State
Drop function if exists dbo.dc_State_MedianIncome

go
CREATE FUNCTION dbo.dc_State_MedianIncome(@zcta varchar(5))
RETURNS INT AS
BEGIN
	DECLARE @StateIncome numeric(10,2)

	SELECT @StateIncome = SUM(household_median_income*population) / sum(population) from
dc_geo_population where state = (select state from dc_geo_population where zcta =  @zcta)

	RETURN @StateIncome
END
Go

-- Checking to if all these median income function works
Select zcta, household_median_income, dbo.dc_State_MedianIncome(zcta) State_MedianIncome
from dc_geo_population
where zcta = '85203'
```

	zcta	household_median_income	State_MedianIncome
1	85203	54919.00	62285

```
Drop function if exists dbo.dc_County_AgeGroup

go
CREATE FUNCTION dbo.dc_County_AgeGroup(@zcta varchar(5), @agegroup varchar(255))
RETURNS INT AS
BEGIN
	DECLARE @Countypop int

	SELECT @Countypop = sum(population) from dc_age_group where county =
                                                                    (select county
                                                                        from
dc_age_group
                                                                        where
zcta =  @zcta and age_group = @agegroup)
                                                                    and age_group =
@agegroup
```

```
        RETURN @Countypop
END
Go

-- Checking to if all the age group county function works
Select zcta, population, age_group, dbo.dc_County_AgeGroup(zcta, age_group) County_Population
from dc_age_group
where zcta = '85203'
order by sort_key
```

	zcta	population	age_group	County_Population
1	85203	3241	Under 5 years	282759
2	85203	3264	5 to 9 years	291566
3	85203	2903	10 to 14 years	307546
4	85203	1760	15 to 17 years	180807
5	85203	1225	18 and 19 years	115921
6	85203	548	20 years	59131
7	85203	258	21 years	59573
8	85203	1762	22 to 24 years	174934
9	85203	3654	25 to 29 years	330686
10	85203	2885	30 to 34 years	308067
11	85203	2455	35 to 39 years	291103
12	85203	2401	40 to 44 years	282633
13	85203	2504	45 to 49 years	282690
14	85203	2370	50 to 54 years	272665
15	85203	2423	55 to 59 years	262802
16	85203	769	60 and 61 years	100101
17	85203	1196	62 to 64 years	141200
18	85203	606	65 and 66 years	87136
19	85203	953	67 to 69 years	124696
20	85203	986	70 to 74 years	170966
21	85203	721	75 to 79 years	121053
22	85203	512	80 to 84 years	77887
23	85203	401	85 years and over	75138

```
--- Create a function that, when given the zcta and age group, returns the States total population
Drop function if exists dbo.dc_State_AgeGroup

go
CREATE FUNCTION dbo.dc_State_AgeGroup(@zcta varchar(5), @agegroup varchar(255))
RETURNS INT AS
BEGIN
        DECLARE @Statepop int

        SELECT @statepop = sum(population) from dc_age_group where state =
                                                                (select state
                                                                        from
dc_age_group
```

```
                                                                    where
zcta =  @zcta and age_group = @agegroup)
                                                                    and age_group =
@agegroup

        RETURN @statepop
END
Go

-- Checking to if all the age group state function works
Select zcta, population, age_group, dbo.dc_State_AgeGroup(zcta, age_group) State_Population
from dc_age_group
where zcta = '85203'
order by sort_key
```

	zcta	population	age_group	State_Population
1	85203	3241	Under 5 years	434649
2	85203	3264	5 to 9 years	451200
3	85203	2903	10 to 14 years	472429
4	85203	1760	15 to 17 years	279414
5	85203	1225	18 and 19 years	194500
6	85203	548	20 years	102784
7	85203	258	21 years	100655
8	85203	1762	22 to 24 years	285223
9	85203	3654	25 to 29 years	501843
10	85203	2885	30 to 34 years	464121
11	85203	2455	35 to 39 years	444010
12	85203	2401	40 to 44 years	423540
13	85203	2504	45 to 49 years	425403
14	85203	2370	50 to 54 years	422843
15	85203	2423	55 to 59 years	427914
16	85203	769	60 and 61 years	171524
17	85203	1196	62 to 64 years	248113
18	85203	606	65 and 66 years	159150
19	85203	953	67 to 69 years	227293
20	85203	986	70 to 74 years	314562
21	85203	721	75 to 79 years	229963
22	85203	512	80 to 84 years	145588
23	85203	401	85 years and over	131065

Answering Data Questions:

```
-- 1. What is the daily and trailing 7-day COVID cases and deaths from May 2020 until current?
--                    Calculate the trailing 7-day cases divided by population.
--                    Calculate the trailing 7-day deaths divided by population.


-- 1. What is the daily and trailing 7-day COVID cases and deaths May 2020 until current?
--                    Going to create 2 temporay views, commit the tables to daily cases and deaths view, and then drop the
first 2 views


-- Droping views to not muddy up the database.  Was only using them as suedo temp table.

--                    Calculate the trailing 7-day cases divided by population.
--                    Calculate the trailing 7-day deaths divided by population.



-- 2.       What is the Population and Household Median Income for the ZCTA 85203.
--                                 This should be by ZCTA, County, & State
--                                 What Percentage Lives in 85203, Maricopa vs the state
Select zcta,
                    county,
                    state,
                    population ZCTA_Population,
                    dbo.dc_ZCTA_CountyPop(zcta) County_Population,
                    dbo.dc_ZCTA_StatePop(zcta) State_Population,
                    household_median_income ZCTA_MedianIncome,
                    dbo.dc_County_MedianIncome(zcta) County_MedianIncome,
                    dbo.dc_State_MedianIncome(zcta) State_MedianIncome
from dc_geo_population
where zcta = '85203'


Select zcta,
                    county,
                    state,
                    population * 1.0 / dbo.dc_ZCTA_StatePop(zcta) ZCTA_perc_vs_state,
                    dbo.dc_ZCTA_CountyPop(zcta) * 1.0 / dbo.dc_ZCTA_StatePop(zcta) County_perc_vs_state
from dc_geo_population

where zcta = '85203'
```

	zcta	county	state	ZCTA_Population	County_Population	State_Population	ZCTA_MedianIncome	County_MedianIncome	State_MedianIncome
1	85203	Maricopa County	AZ	39797	4401060	7057786	54919.00	68403	62285

	zcta	county	state	ZCTA_perc_vs_state	County_perc_vs_state
1	85203	Maricopa County	AZ	0.005638737133	0.623575155154

```
--- 3.      What is the total population By County?
---                         What is the top 25 populated counties?
---                         What's their Weighted Average Household Median Income by population?
---                         What is the trailing 7 days COVID cases divided by population?

select state,
                    county,
                    SUM(population) total_population
from dc_geo_population
group by state, county

select top 25 state,
                    county,
                    SUM(population) total_population
from dc_geo_population
group by state, county
order by total_population desc

select top 25 state,
            county,
            SUM(population) total_population
from dc_geo_population
group by state, county
order by total_population desc

go
with top_25_county as (
                                                select top 25 state,
                                                            county,
                                                            county_id,
                                                            SUM(population) total_population
                                                from dc_geo_population
                                                group by state, county, county_id
                                                order by total_population desc)

select date,
            state,
            county,
            top_25_county.county_id,
            daily_cases,
            t7d_cases,
            t7d_deaths,
      t7d_deaths * 1.0 / t7d_cases deaths_cases_perc,
            t7d_cases * 1.0 / total_population cases_by_population
from top_25_county
          left join dc_daily_covid
          on top_25_county.county_id = dc_daily_covid.county_id
order by total_population desc, date
```

	state	county	total_population
1	CA	Los Angeles County	10294506
2	IL	Cook County	5508302
3	TX	Harris County	4405610
4	AZ	Maricopa County	4401060
5	CA	San Diego County	3260294
6	CA	Orange County	3075155
7	TX	Dallas County	2741458
8	FL	Miami-Dade County	2699439
9	NY	Kings County	2589974
10	CA	Riverside County	2536415
11	NY	Queens County	2255738
12	WA	King County	2215875
13	NV	Clark County	2179948
14	TX	Bexar County	2030193
15	CA	San Bernardino County	1969461
16	FL	Broward County	1926205
17	CA	Santa Clara County	1913644
18	CA	Alameda County	1732726
19	TX	Tarrant County	1670120
20	NY	New York County	1621398
21	MA	Middlesex County	1615535
22	MI	Wayne County	1588978
23	PA	Philadelphia County	1546122
24	CA	Sacramento County	1530796
25	FL	Hillsborough County	1508389

	state	county	total_population
1	CA	Los Angeles County	10294506
2	IL	Cook County	5508302
3	TX	Harris County	4405610
4	AZ	Maricopa County	4401060
5	CA	San Diego County	3260294
6	CA	Orange County	3075155
7	TX	Dallas County	2741458
8	FL	Miami-Dade County	2699439
9	NY	Kings County	2589974
10	CA	Riverside County	2536415
11	NY	Queens County	2255738
12	WA	King County	2215875
13	NV	Clark County	2179948
14	TX	Bexar County	2030193
15	CA	San Bernardino County	1969461
16	FL	Broward County	1926205
17	CA	Santa Clara County	1913644
18	CA	Alameda County	1732726
19	TX	Tarrant County	1670120
20	NY	New York County	1621398
21	MA	Middlesex County	1615535
22	MI	Wayne County	1588978
23	PA	Philadelphia County	1546122
24	CA	Sacramento County	1530796
25	FL	Hillsborough County	1508389

	date	state	county	county_id	daily_cases	t7d_cases	cases_by_population
1	2020-05-01	CA	Los Angeles County	06037	1033	5670	0.000550779221
2	2020-05-02	CA	Los Angeles County	06037	679	5787	0.000562144506
3	2020-05-03	CA	Los Angeles County	06037	768	6134	0.000595851806
4	2020-05-04	CA	Los Angeles County	06037	555	5800	0.000563407316
5	2020-05-05	CA	Los Angeles County	06037	1598	6839	0.000664334937
6	2020-05-06	CA	Los Angeles County	06037	829	6159	0.000598280286
7	2020-05-07	CA	Los Angeles County	06037	783	6245	0.000606634257
8	2020-05-08	CA	Los Angeles County	06037	869	6081	0.000590703429
9	2020-05-09	CA	Los Angeles County	06037	901	6303	0.000612268330
10	2020-05-10	CA	Los Angeles County	06037	480	6015	0.000584292242
11	2020-05-11	CA	Los Angeles County	06037	581	6041	0.000586817861
12	2020-05-12	CA	Los Angeles County	06037	922	5365	0.000521151767
13	2020-05-13	CA	Los Angeles County	06037	1248	5784	0.000561853089
14	2020-05-14	CA	Los Angeles County	06037	901	5902	0.000573315514
15	2020-05-15	CA	Los Angeles County	06037	930	5963	0.000579241004
16	2020-05-16	CA	Los Angeles County	06037	1044	6106	0.000593131909
17	2020-05-17	CA	Los Angeles County	06037	671	6297	0.000611685495
18	2020-05-18	CA	Los Angeles County	06037	477	6193	0.000601583019
19	2020-05-19	CA	Los Angeles County	06037	1122	6393	0.000621010857
20	2020-05-20	CA	Los Angeles County	06037	1284	6429	0.000624507868
21	2020-05-21	CA	Los Angeles County	06037	1180	6708	0.000651609703
22	2020-05-22	CA	Los Angeles County	06037	1015	6793	0.000659866534
23	2020-05-23	CA	Los Angeles County	06037	1003	6752	0.000655883827
24	2020-05-24	CA	Los Angeles County	06037	933	7014	0.000681334296
25	2020-05-25	CA	Los Angeles County	06037	1030	7567	0.000735052269
26	2020-05-26	CA	Los Angeles County	06037	1804	8249	0.000801201198

```
--4.	What is the weighted average household median income by County?
--		What's the top 25 highest weighted average household median income by population?
--		What is the total population of these counties?
--		What's the bottom 25 weighted average household median income by population?
--		What is the trailing 7-day COVID cases divide by population?
--		What does the top 25 highest weighted average household median income look compare to the
bottom 25?


select top 25 state,
		county,
		SUM(population) total_population,
		coalesce(sum(population * household_median_income) / nullif(sum(population),0),0) weighted_avg_income
from dc_geo_population
group by state, county
order by weighted_avg_income desc

select top 25 state,
		county,
		SUM(population) total_population,
		coalesce(sum(population * household_median_income) / nullif(sum(population),0),0) weighted_avg_income
```

```
from dc_geo_population
where state != 'PR'
group by state, county
having coalesce(sum(population * household_median_income) / nullif(sum(population),0),0) > 0
order by weighted_avg_income


go
with top_25_county as (
                                                    select top 25 state,
                                                                county,
                                                                county_id,
                                                                SUM(population) total_population,
                                                                coalesce(sum(population *
household_median_income) / nullif(sum(population),0),0) weighted_avg_income
                                                    from dc_geo_population
                                                    where state != 'PR'
                                                    group by state, county, county_id
                                                    order by weighted_avg_income desc),

            top_25_cases as (
                                                    select date,
                                                                state,
                                                                county,
                                                                top_25_county.county_id,
                                                                total_population,
                                                                weighted_avg_income,
                                                                daily_cases,
                                                                t7d_cases,
                                                                t7d_cases * 1.0 / total_population
cases_by_population
                                                    from top_25_county
                                                                left join dc_daily_covid
                                                                on top_25_county.county_id =
dc_daily_covid.county_id
                                                    ),
            top_25_totaled as (
                                                    select 'Top 25 Counties' as covid_impact,
                                                                date,
                                                                sum(total_population) total_population,
                                                                sum(total_population * weighted_avg_income) /
sum(total_population) weighted_avg_income,
                                                                sum(t7d_cases) t7d_cases,
                                                                sum(t7d_cases) * 1.0 / sum(total_population)
cases_by_population
                                                    from top_25_cases
                                                    group by date

            ),
            bottom_25_county as (
                                                    select top 25 state,
                                                                county,
                                                                county_id,
                                                                SUM(population) total_population,
                                                                coalesce(sum(population *
household_median_income) / nullif(sum(population),0),0) weighted_avg_income
                                                    from dc_geo_population
                                                    where state != 'PR'
                                                    group by state, county, county_id
                                                    order by weighted_avg_income),
```

```
        bottom_25_cases as (
                                                select date,
                                                        state,
                                                        county,
                                                        bottom_25_county.county_id,
                                                        total_population,
                                                        weighted_avg_income,
                                                        daily_cases,
                                                        t7d_cases,
                                                        t7d_cases * 1.0 / total_population
cases_by_population
                                                from bottom_25_county
                                                        left join dc_daily_covid
                                                        on bottom_25_county.county_id =
dc_daily_covid.county_id
                                                ),
        bottom_25_totaled as (
                                                select 'Bottom 25 Counties' as covid_impact,
                                                        date,
                                                        sum(total_population) total_population,
                                                        sum(total_population * weighted_avg_income) /
sum(total_population) weighted_avg_income,
                                                        sum(t7d_cases) t7d_cases,
                                                        sum(t7d_cases) * 1.0 / sum(total_population)
cases_by_population
                                                from bottom_25_cases
                                                group by date
        )
        select top_25_totaled.date,
                top_25_totaled.cases_by_population cases_by_population_top_25_income,
                bottom_25_totaled.cases_by_population cases_by_population_bottom_25_income,
                top_25_totaled.cases_by_population - bottom_25_totaled.cases_by_population perc_delta

        from top_25_totaled
        join bottom_25_totaled

        on top_25_totaled.date = bottom_25_totaled.date
```

	state	county	total_population	weighted_avg_income
1	VA	Loudoun County	390359	143618.815556
2	VA	Arlington County	263721	135089.083656
3	TX	Hays County	27052	129645.794728
4	MD	Howard County	250942	128820.279686
5	VA	Alexandria city	66334	128628.040522
6	CA	San Mateo County	786170	127817.613326
7	CA	Santa Clara County	1913644	125876.116410
8	VA	Fairfax County	1245820	123120.462702
9	NJ	Morris County	550333	121818.730881
10	NM	Los Alamos County	18625	121324.000000
11	CO	Douglas County	313457	120753.046197
12	NY	Nassau County	1369611	119087.739613

	state	county	total_population	weighted_avg_income
1	SC	McCormick County	1437	6367.867084
2	VA	Sussex County	2666	8064.580645
3	KY	Wolfe County	327	14977.000000
4	KY	Monroe County	1722	16956.422764
5	SC	Barnwell County	147	17009.000000
6	TN	Putnam County	2948	17611.311397
7	KY	Owsley County	520	18917.000000
8	TN	Hancock County	1125	20065.706666
9	MS	Jefferson County	5290	20886.000000
10	GA	Clay County	5785	21748.852722
11	LA	East Carroll Parish	8687	22733.230804
12	AR	Phillips County	14553	23812.452209

	date	cases_by_population_top_25_income	cases_by_population_bottom_25_income	perc_delta
1	2020-05-05	0.001046083190	0.012347796261	-0.011301713071
2	2020-05-07	0.000971479211	0.014105642256	-0.013134163045
3	2020-05-14	0.000758538884	0.025595952666	-0.024837413782
4	2020-05-16	0.000713294761	0.026367689933	-0.025654395172
5	2020-05-23	0.000671500909	0.018646146040	-0.017974645131
6	2020-05-25	0.000668245936	0.022536044987	-0.021867799051
7	2020-05-30	0.000629772156	0.024675378898	-0.024045606742
8	2020-06-01	0.000654900547	0.022629535301	-0.021974634754
9	2020-06-03	0.000626647382	0.037660222955	-0.037033575573
10	2020-06-08	0.000466958412	0.030061375307	-0.029594416895
11	2020-06-10	0.000430307417	0.011273015740	-0.010842708323
12	2020-06-17	0.000337019893	0.013986889900	-0.013649870007
13	2020-06-19	0.000327971069	0.012400317314	-0.012072346245
14	2020-06-24	0.000376535264	0.014237401361	-0.013860866097
15	2020-06-26	0.000433236892	0.017243538891	-0.016810301999
16	2020-06-28	0.000463182643	0.019832157321	-0.019368974678
17	2020-07-03	0.000445801088	0.022295520020	-0.021849718932
18	2020-07-05	0.000426531648	0.022379023840	-0.021952492192
19	2020-07-12	0.000456672697	0.023339317773	-0.022882645076
20	2020-07-14	0.000489808321	0.027973779800	-0.027483971479
21	2020-07-19	0.000541171794	0.033109264748	-0.032568092954
22	2020-07-21	0.000541432191	0.032775249467	-0.032233817276
23	2020-07-23	0.000553150094	0.032107218905	-0.031554068811
24	2020-07-28	0.000526394217	0.046511627906	-0.045985233689
25	2020-07-30	0.000547616640	0.047012650828	-0.046465034188
26	2020-08-06	0.000559920438	0.039956578013	-0.039396657575

--5. For ZCTA 85203, Arizona, what is the population broken out by age group?
-- How does that compare to the State?

--- Heres the answere by population but this doesn't really tell us how 85203 compares. We will have to normalize.
--- We can normalize this by taking the population of the repective location and divide by the total population.
--- ie. zcta age group population divide by zcta total population.

```
select zcta,
        age_group,
        population zcta_popualtion,
        dbo.dc_County_AgeGroup(zcta,age_group) county_population,
        dbo.dc_State_AgeGroup(zcta, age_group) state_population
from dc_age_group
where zcta = '85203'
order by sort_key
```

```
-- Normalizing the value per the above.
--- ie. zcta age group population divide by zcta total population.
                select zcta,
                                age_group,
                                population * 1.0/ sum(population) over () zcta_perc,
                                dbo.dc_County_AgeGroup(zcta,age_group) * 1.0/
sum(dbo.dc_County_AgeGroup(zcta,age_group)) over () county_perc,
                                dbo.dc_State_AgeGroup(zcta, age_group) * 1.0/ sum(dbo.dc_State_AgeGroup(zcta,
age_group)) over () state_perc
                from dc_age_group
                where zcta = '85203'
                order by sort_key

go
```

	zcta	age_group	zcta_popualtion	county_population	state_population
1	85203	Under 5 years	3241	282759	434649
2	85203	5 to 9 years	3264	291566	451200
3	85203	10 to 14 years	2903	307546	472429
4	85203	15 to 17 years	1760	180807	279414
5	85203	18 and 19 years	1225	115921	194500
6	85203	20 years	548	59131	102784
7	85203	21 years	258	59573	100655
8	85203	22 to 24 years	1762	174934	285223
9	85203	25 to 29 years	3654	330686	501843
10	85203	30 to 34 years	2885	308067	464121
11	85203	35 to 39 years	2455	291103	444010
12	85203	40 to 44 years	2401	282633	423540
13	85203	45 to 49 years	2504	282690	425403
14	85203	50 to 54 years	2370	272665	422843
15	85203	55 to 59 years	2423	262802	427914
16	85203	60 and 61 years	769	100101	171524
17	85203	62 to 64 years	1196	141200	248113
18	85203	65 and 66 years	606	87136	159150
19	85203	67 to 69 years	953	124696	227293
20	85203	70 to 74 years	986	170966	314562
21	85203	75 to 79 years	721	121053	229963
22	85203	80 to 84 years	512	77887	145588
23	85203	85 years and over	401	75138	131065

	zcta	age_group	zcta_perc	county_perc	state_perc
1	85203	Under 5 years	0.081438299369	0.064247931180	0.061584326869
2	85203	5 to 9 years	0.082016232379	0.066249040003	0.063929396555
3	85203	10 to 14 years	0.072945196874	0.069879983458	0.066937280331
4	85203	15 to 17 years	0.044224439028	0.041082602827	0.039589468992
5	85203	18 and 19 years	0.030781214664	0.026339336432	0.027558217265
6	85203	20 years	0.013769882151	0.013435626871	0.014563207215
7	85203	21 years	0.006482900721	0.013536057222	0.014261554544
8	85203	22 to 24 years	0.044274694072	0.039748151581	0.040412531635
9	85203	25 to 29 years	0.091815966027	0.075137807709	0.071104876231
10	85203	30 to 34 years	0.072492901474	0.069998364030	0.065760140644
11	85203	35 to 39 years	0.061688066939	0.066143838075	0.062910663485
12	85203	40 to 44 years	0.060331180742	0.064219301713	0.060010320516
13	85203	45 to 49 years	0.062919315526	0.064232253139	0.060274284315
14	85203	50 to 54 years	0.059552227554	0.061954392805	0.059911564334
15	85203	55 to 59 years	0.060883986230	0.059713341785	0.060630061608
16	85203	60 and 61 years	0.019323064552	0.022744747856	0.024302805440
17	85203	62 to 64 years	0.030052516521	0.032083179961	0.035154508793
18	85203	65 and 66 years	0.015227278438	0.019798866636	0.022549564410
19	85203	67 to 69 years	0.023946528632	0.028333174280	0.032204575202
20	85203	70 to 74 years	0.024775736864	0.038846550603	0.044569500973
21	85203	75 to 79 years	0.018116943488	0.027505419149	0.032582880807
22	85203	80 to 84 years	0.012865291353	0.017697327462	0.020627998638
23	85203	85 years and ...	0.010076136392	0.017072705211	0.018570271187

```
--          6.          What Counties had the highest deaths by percentage of population?
--                              The population must be over 1000 people.
--                              Include their household median income.

with population_counties as (
                                                                        select county_id,
                                                                                county,
                                                                                state,
                                                                                sum(population) population,
                                                                                sum(population *
household_median_income) / nullif(sum(household_median_income),0) household_median_income
                                                                        from dc_geo_population
                                                                        group by county_id, county, state
                                                                        )
        select top 25
                population_counties.county_id,
                population_counties.county,
                population_counties.state,
                population_counties.population,
                population_counties.household_median_income,
                total_deaths,
                total_deaths * 1.0 / population_counties.population death_perc
        from population_counties
        join dc_covid_county
        on population_counties.county_id = dc_covid_county.county_id
        where population >= 10000
        and population_counties.county_id != 36061
        order by death_perc desc
```

	county_id	county	state	population	household_median_income	total_deaths	death_perc
1	18177	Wayne County	IN	10383	1801.440481	200	0.019262255610
2	01069	Houston County	AL	16062	2721.598410	269	0.016747603038
3	55135	Waupaca County	WI	10791	2277.563581	160	0.014827170790
4	01121	Talladega County	AL	12667	3387.807250	167	0.013183863582
5	01123	Tallapoosa County	AL	10956	5597.036792	144	0.013143483023
6	39141	Ross County	OH	10888	2465.719700	140	0.012858192505
7	45035	Dorchester County	SC	16814	3589.557216	201	0.011954323777
8	13277	Tift County	GA	11027	6392.198741	123	0.011154439104
9	13275	Thomas County	GA	10905	10905.000000	116	0.010637322329
10	18109	Morgan County	IN	13205	2966.830320	138	0.010450586898
11	04017	Navajo County	AZ	49546	3048.477072	517	0.010434747507
12	13299	Ware County	GA	16237	15441.000000	157	0.009669273880
13	13129	Gordon County	GA	11352	3775.749680	106	0.009337561663
14	19139	Muscatine County	IA	10073	2137.119821	94	0.009331877295
15	46103	Pennington County	SD	20799	3760.130833	190	0.009135054569
16	13297	Walton County	GA	26516	26516.000000	240	0.009051138934
17	08087	Morgan County	CO	10345	1510.783390	92	0.008893185113
18	48209	Hays County	TX	27052	9704.878311	235	0.008686973236
19	47179	Washington County	TN	27866	13522.094551	235	0.008433216105
20	13313	Whitfield County	GA	27557	20235.023611	231	0.008382625104
21	01055	Etowah County	AL	42335	7891.882736	339	0.008007558757
22	54029	Hancock County	WV	12278	2748.813009	95	0.007737416517
23	35031	McKinley County	NM	58951	3662.833996	454	0.007701311258
24	16083	Twin Falls County	ID	16476	4404.069994	126	0.007647487254
25	39077	Huron County	OH	14399	3676.020854	110	0.007639419404

```
---7.	Group the age groups that are considered the high-risk age categories for COVID.  That is age groups over 65 and under 5.
---			Make this a calculated field that is a Boolean field being either 1 or 0; high risk or low risk.
---			Total the population by high risk and low risk by county.
---			Any counties that are greater than 30% being high risk what is the number of COVID cases and deaths by total population vs low risk counties.
---			Group these counties into 2 rows, high risk and low risk.  How do they total cases and deaths compare to each other?
go
with risk_counties as (
                                                select dc_age_group.county,
                                                                covid_age_group,
                                                                SUM(dc_age_group.population)
population_age_group,
                                                                coalesce(SUM(dc_age_group.population) * 1.0 /

        nullif(SUM(SUM(dc_age_group.population)) over (Partition by dc_age_group.county),0),0) perc_population
                                                from dc_age_group
                                                group by dc_age_group.county,
                                                                covid_age_group),
        county_population as (select county_id,
                                                                county,
```

```
                                                    sum(population) total_population
                                                    from dc_geo_population
                                                    group by county_id,
                                                                    county
                                                    )

        select 'High Risk' as risk_type,
                    sum(total_population) total_population,
                    sum(total_cases) * 1.0 / sum(total_population) perc_cases,
                    sum(total_deaths) * 1.0 / sum(total_population) perc_deaths
        from county_population
        left join dc_covid_county
        on county_population.county_id = dc_covid_county.county_id
        where county_population.county in (select distinct county from risk_counties where perc_population > 0.30 and
covid_age_group = 'High Risk Age Group')

        union

        select 'Low Risk' as risk_type,
                    sum(total_population) total_population,
                    sum(total_cases) * 1.0 / sum(total_population) perc_cases,
                    sum(total_deaths) * 1.0 / sum(total_population) perc_deaths
        from county_population
        left join dc_covid_county
        on county_population.county_id = dc_covid_county.county_id
        where county_population.county in (select distinct county from risk_counties where perc_population > 0.90 and
covid_age_group = 'Low Risk Age Group')
```

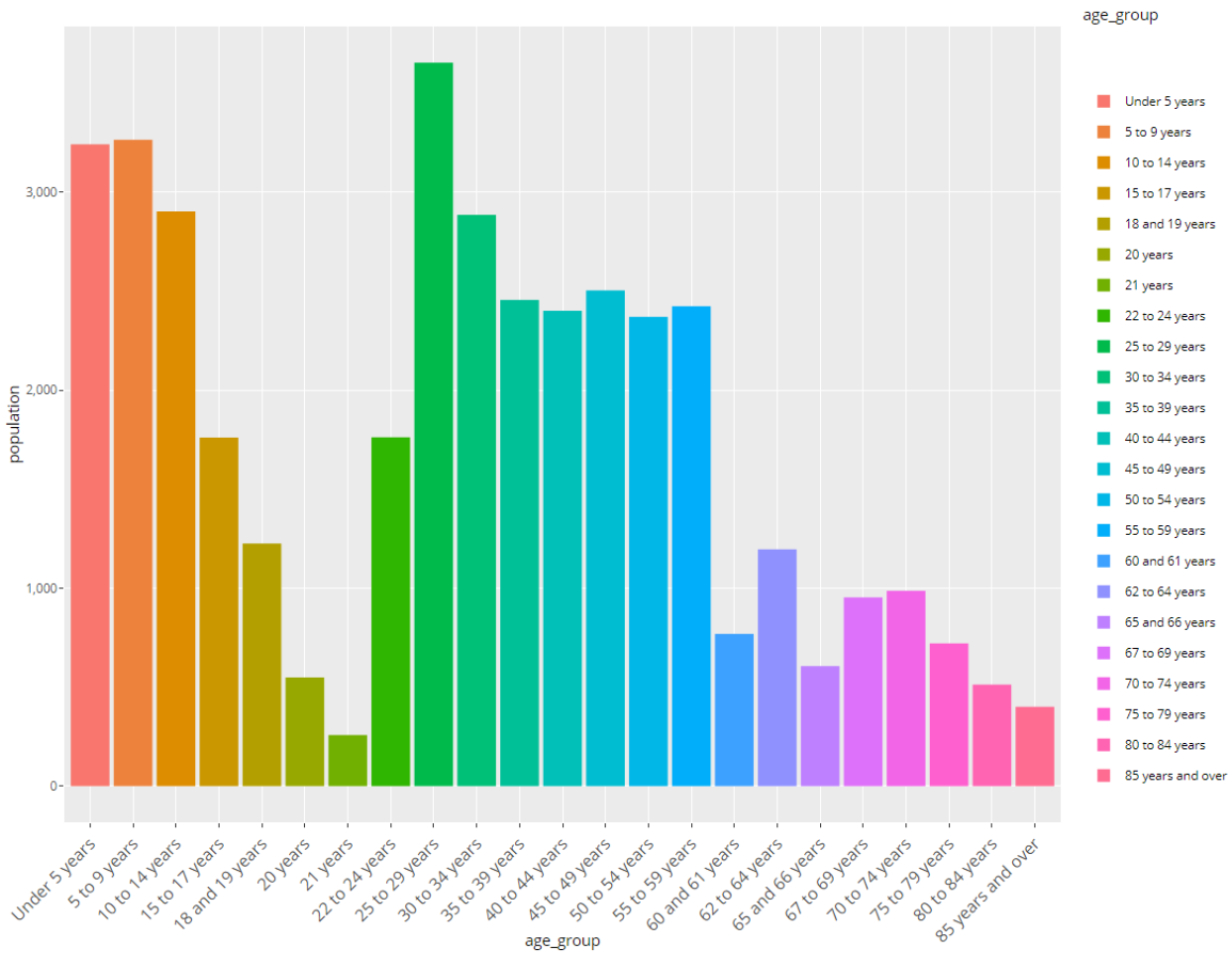
	risk_type	total_population	perc_cases	perc_deaths
1	High Risk	5205626	0.077999456741	0.001918693352
2	Low Risk	15538	1.066289097695	0.018856995752

```
-- 8.      How many Deaths Vs. Total Population
go
with county_population as (
                                                    select county_id,
                                                                sum(population) population
                                                    from dc_geo_population
                                                    group by county_id
                                                    )

select SUM(total_deaths) total_deaths,
            SUM(population) total_population,
            SUM(total_deaths) * 1.0 / SUM(population) death_perc
from dc_covid_county
join county_population

on dc_covid_county.county_id = county_population.county_id
```

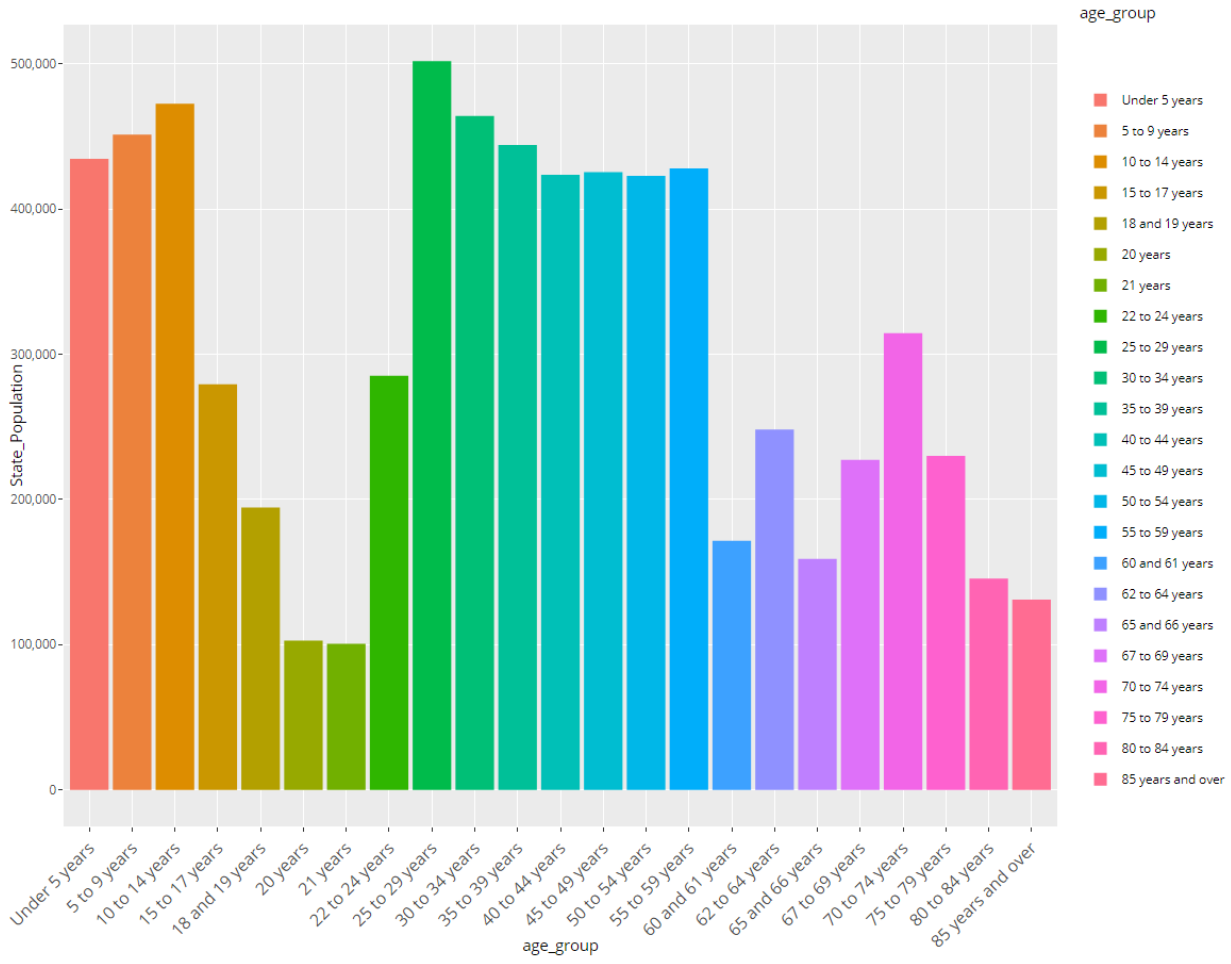
	total_deaths	total_population	death_perc
1	534385	321182638	0.001663804131

Data Visualizations:

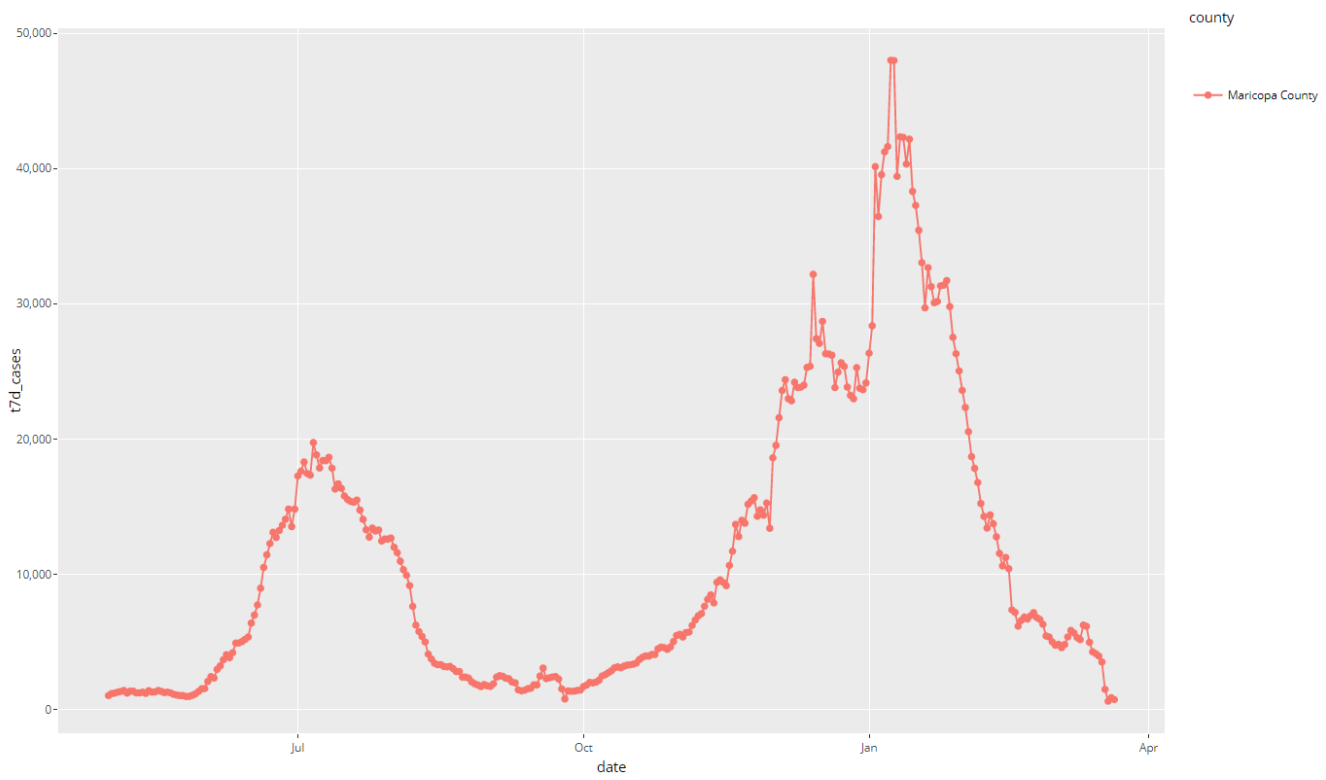
Population by ZCTA and Age Group



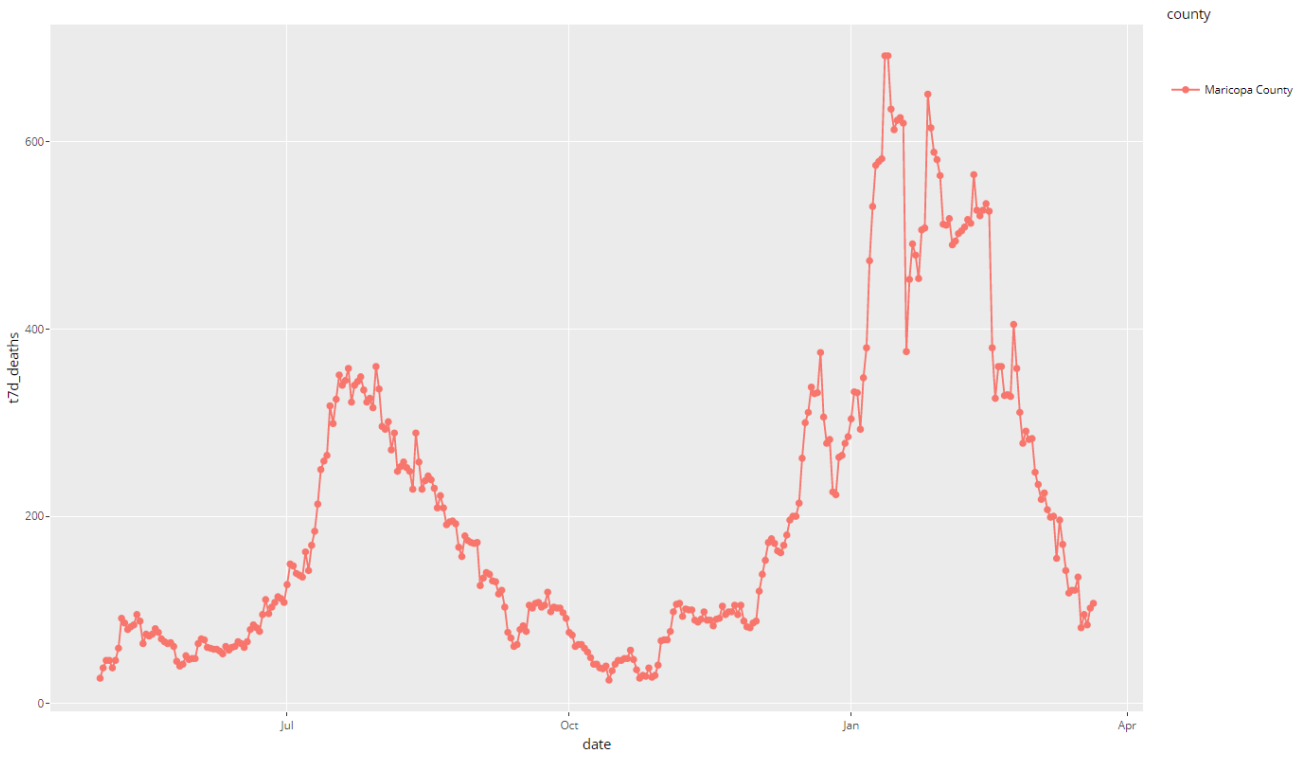
Population by State and Age Group



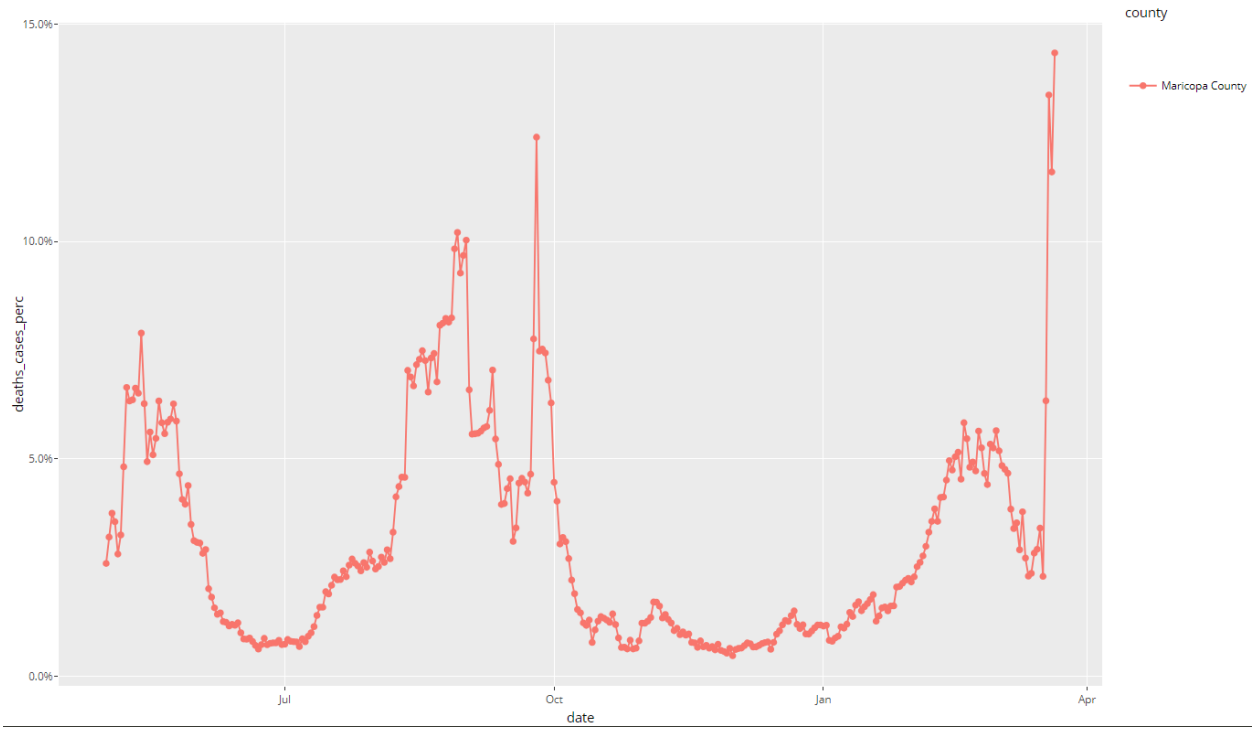
Daily COVID Cases for Maricopa County



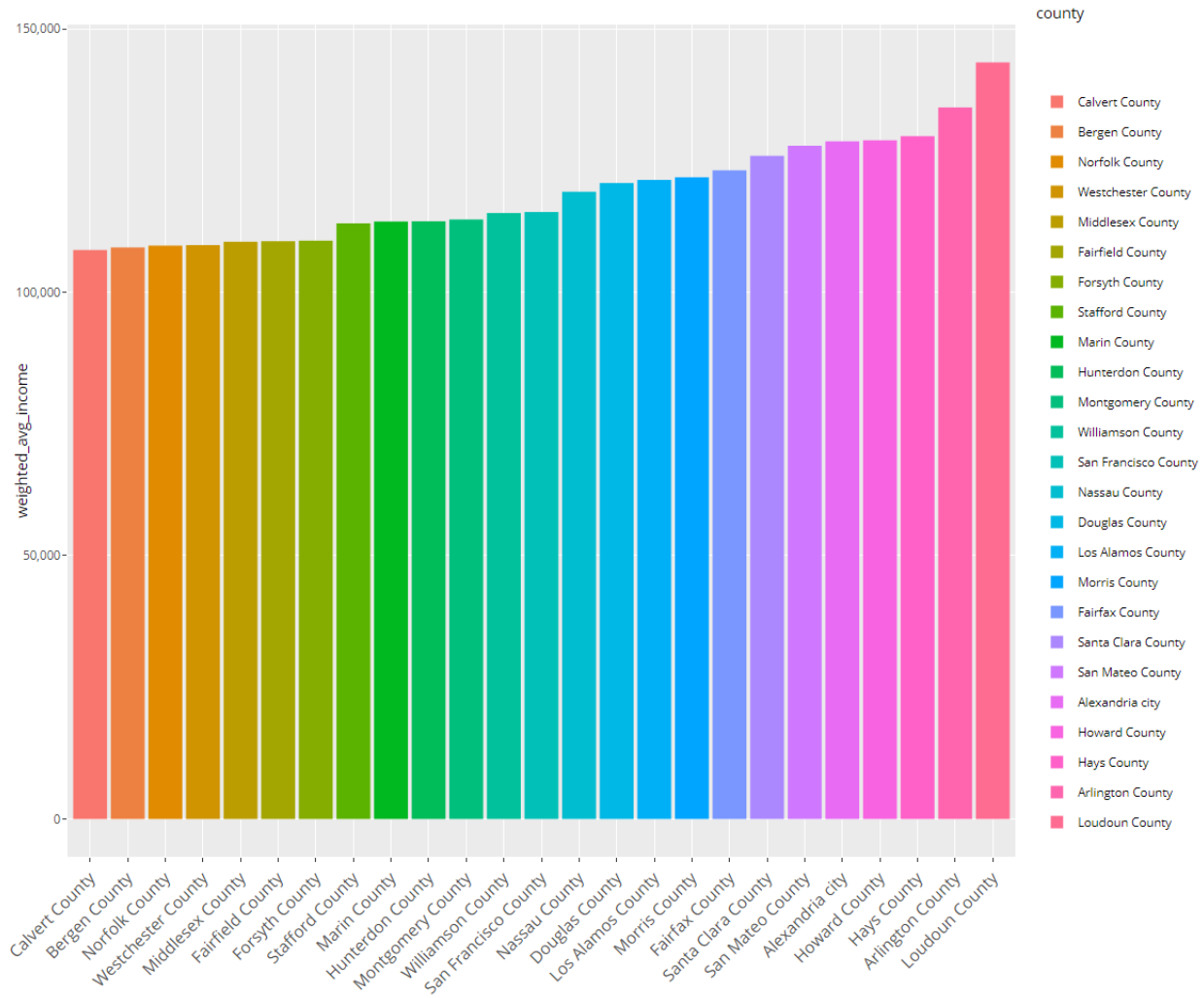
Daily COVID Deaths for Maricopa County



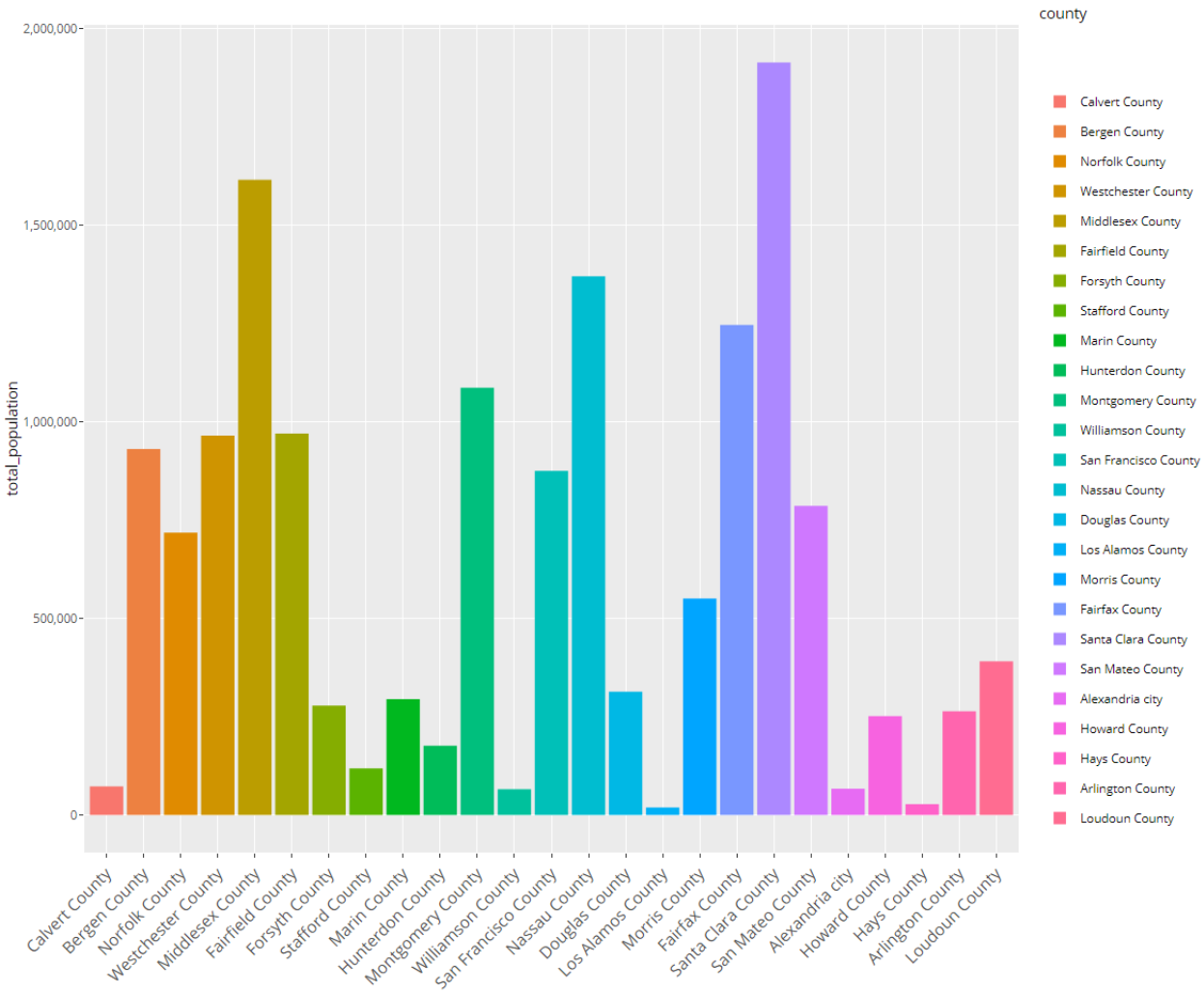
Daily COVID Deaths per cases for Maricopa County



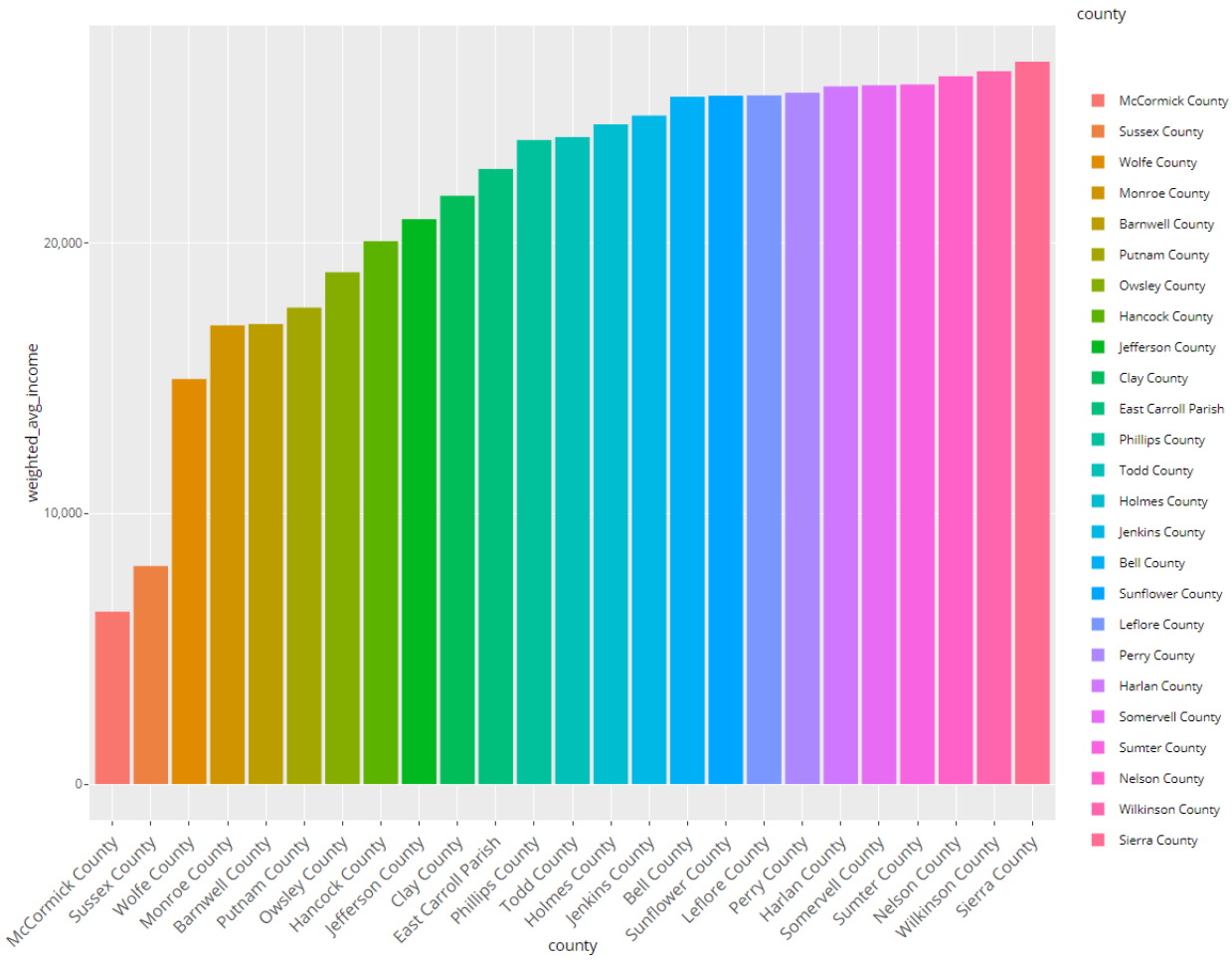
Top 25 Household Median Income by County (Image 1) & Population (Image 2)



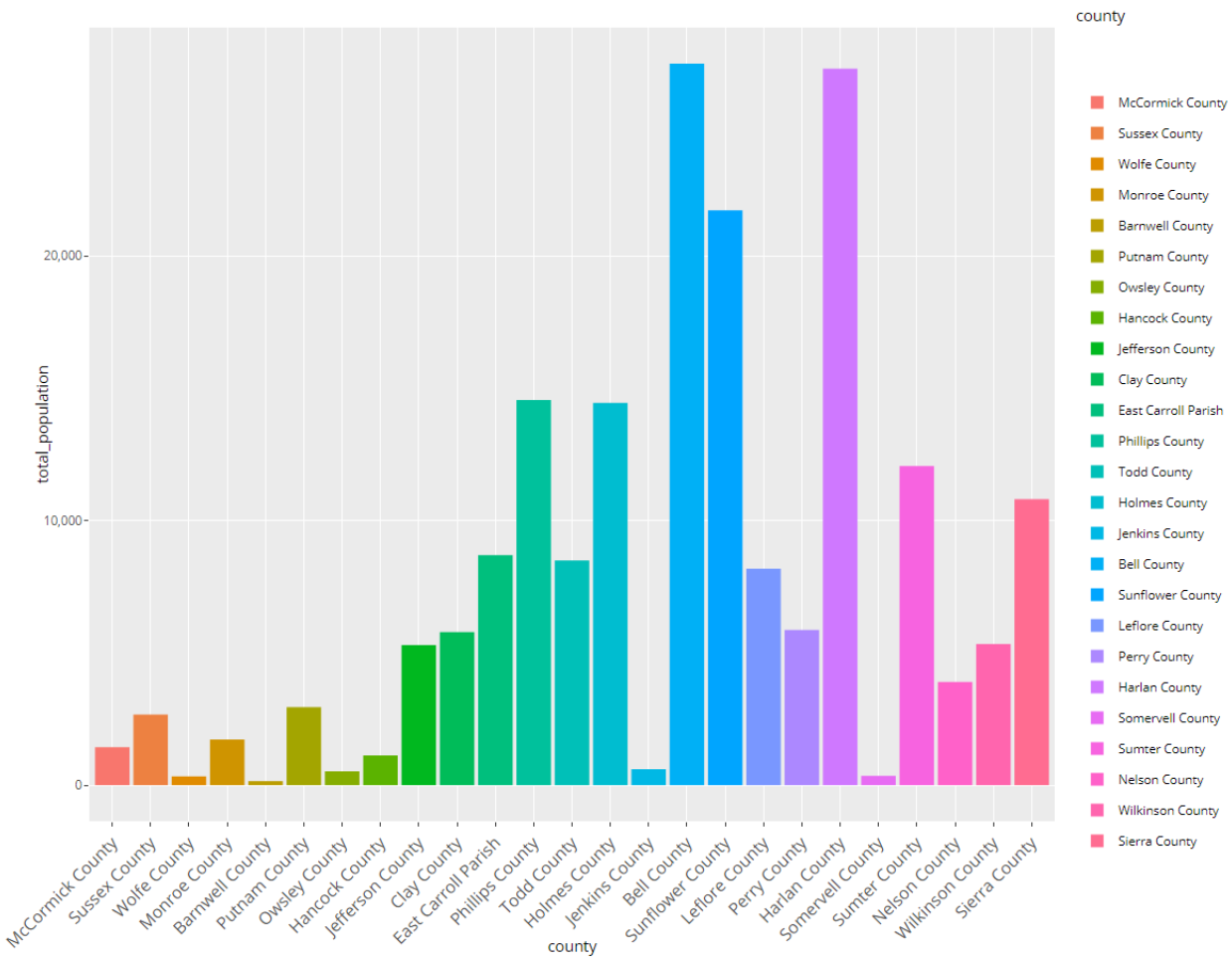
The Population of Top 25 Household Median Income sorted by income.



Bottom 25 Household Median Income by County (Image 1) & Population (Image 2)


county
McCormick County
Sussex County
Wolfe County
Monroe County
Barnwell County
Putnam County
Owsley County
Hancock County
Jefferson County
Clay County
East Carroll Parish
Phillips County
Todd County
Holmes County
Jenkins County
Bell County
Sunflower County
Leflore County
Perry County
Harlan County
Somervell County
Sumter County
Nelson County
Wilkinson County
Sierra County
weighted_avg_income
20,000
10,000
0
county

The Population of Top 25 Household Median Income sorted by income.



Reflection:

I have worked with databases now for 9 years and at first, I was under the impression that I would only get a little out of this class. In short, I was wrong. From Logical and Conceptual Models to Normal Form and then creating functions and procedures, I'm able to really approach database tasks and problems more efficiently and intelligently. For example, I would have skipped the modeling phase, dodged the normal form exercise and begin grabbing data to put into tables prior to this class. I would have ended up spending an exorbitant amount of time fixing my table, making mistakes when doing analytics, and had to update multiple tables with the same data. Something I could have done better is understanding the COVID data. What I discovered later on is that the cases and deaths where a cumulative number. I had to transform that data, drop the old and insert the new records after discovering this. How I built my database made it easy to perform this task. The COVID data was by county but broke it's own rule by combining New York City Counties into one County called New York City. I could have created an additional table that stored the county codes associate to New York County, then joined up on this table to bring back to the census data. The last thing I could have done better, which I started to do, was create more views. For example, I created a zcta that had county, state as the dimensions and household median income as the metrics. I also did this for age group. I should have created a few more views. One for Race and then an aggregated view on county and state. This view would have made it easy to perform analytics on. Rather than building long queries users can just query these tables.

Summary:

In building this database I had an objective to use census data to help answer questions surrounding the impact of COVID on our society. The lenses I was exploring was from a

socioeconomics, age, and race perspective. I thought by weighting cases and deaths by population I could discern what the impact looked like for each of these groups. I will say there's probably a more intelligent way to discern this information and I'm excited to continue my data discovery exercise outside of this class. I'll probably implement the views I discussed in my reflection to make the analysis easier to perform. Deploy functions to aggregate metrics in a more straightforward method. Grow my understanding of procedures and implement them to help manage my data. I cannot help to be thankful of everything I learned in this class and ready to apply my new skills.

Project 2 – IST 718 Big Data Analytics

Course Description

A broad introduction to analytical processing tools and techniques for information professionals. Students will develop a portfolio of resources, demonstrations, recipes, and examples of various analytical techniques. You will find if much easier to succeed if you have completed IST687, IST777 or both. Familiarity with command-line interfaces, basic quantitative skills, including statistics, as well as programming skills with languages such as R or Python. Most of the course work will be using Python, Spark, and Tensorflow.

Learning Objectives:

1. **O**btain data and explain data structures and data elements.
2. **S**crub data by applying scripting methods, to include debugging, for data manipulation in Python, R or other languages.
3. **E**xplore data by analyzing using qualitative techniques including descriptive statistics, summarization, and visualizations.
4. **M**odel relationships between data using the appropriate analytical methodologies matched to the information and the needs of clients and users.
5. Interpret the data, model, analysis, and findings. Communicate the results in a meaningful way.
6. Select an applicable analytical methodology for real problems in areas such as business, science, and engineering.

Deliverables(s) – Predicting Zillow House Prices

Looking to new markets to open a business or buy a home can be challenging. A way to solve this problem is to look to the housing market. This can really help in understanding the economic health of an area. In this exercise understanding historical returns, the risk of an area, and by using forecasting techniques a Data Scientist needs to distil where a business should be opened in United States.

Project Process & Development

Obtain

There were 4 datasets used in performing this analysis:

1. Zillow Static Data set found at https://files.zillowstatic.com/research/public/Zip/Zip_Zhvi_SingleFamilyResidence.csv
2. Zip Code Tabulation Area (ZCTA) Household Median Income and Population level data from the Census Bureau.
3. County Household Median Income and Population level data from the Census Bureau.
4. State Household Median Income and Population level data from the Census Bureau.

5. State code data mapping to plot on a geographic map.

Zillow Data

To obtain the Zillow data set the Pandas read csv function was used by inserting the above url.

```
In [3]: zillow = pd.read_csv("https://files.zillowstatic.com/research/public/Zip/Zip_Zhvi_SingleFamilyResidence.csv")
        zillow.head()
```

Out[3]:

	RegionID	SizeRank	RegionName	RegionType	StateName	State	City	Metro	CountyName	1996-01-31	...	2019-06-30	2019-07-31	2019-08-31	201
0	61639	0	10025	Zip	NY	NY	New York	New York-Newark-Jersey City	New York County	NaN	...	1413747.0	1405862.0	1402547.0	1390
1	84654	1	60657	Zip	IL	IL	Chicago	Chicago-Naperville-Elgin	Cook County	364892.0	...	974693.0	975616.0	975734.0	975
2	61637	2	10023	Zip	NY	NY	New York	New York-Newark-Jersey City	New York County	NaN	...	1528603.0	1514894.0	1502233.0	1492

Census Data

To obtain the census data, the following code lines were used:

1. Ping the census bureau api for Household Median Income and Population by year
2. Append the data to a dataframe
3. Loop through 2011 – 2020, this is everything that the census bureau has.
4. Rename the headers to Median Income and Population
5. Create a Pandas Dataframe

```
In [8]: acs.printtable(acs.censustable('acs5', 2009, 'B19013'))
        acs.printtable(acs.censustable('acs5', 2009, 'B01003'))
```

```
Variable     | Table                          | Label                                                                                          | Type
-------------------------------------------------------------------------------------------------------------------------------------------------
B19013_001E  | MEDIAN HOUSEHOLD INCOME IN THE | !! Estimate Median household income in the past 12 month                                       | int
-------------------------------------------------------------------------------------------------------------------------------------------------
Variable     | Table                          | Label                                                                                          | Type
-------------------------------------------------------------------------------------------------------------------------------------------------
B01003_001E  | TOTAL POPULATION               | !! Estimate Total                                                                              | int
-------------------------------------------------------------------------------------------------------------------------------------------------
```

Downloading American Census Data by Zipcode

```
In [9]: acs_data = acs.download('acs5', 2011, acs.censusgeo([('zip code tabulation area', '*')]),
                                 ['B19013_001E', 'B01003_001E'])
```

```
In [10]: census_year = list(range(2011,2020))
         census_pull = pd.DataFrame()

         for y in census_year:
           acs_data = acs.download('acs5', y, acs.censusgeo([('zip code tabulation area', '*')]),
                                   ['B19013_001E', 'B01003_001E'])
           cbd = pd.DataFrame(acs_data)
           cbd['year'] = y
           census_pull = census_pull.append(cbd)
```

In [14]:
```
acs.exportcsv('census_data.csv', census_data)
census_data = pd.read_csv('census_data.csv')
census_data.head()
```

Out[14]:

	state	zip code tabulation area	NAME	median_income	population	year
0	72	601	ZCTA5 00601	13318.0	18533	2011
1	72	602	ZCTA5 00602	14947.0	41930	2011
2	72	603	ZCTA5 00603	14437.0	54475	2011
3	72	606	ZCTA5 00606	11155.0	6386	2011
4	72	610	ZCTA5 00610	16367.0	29111	2011

This data frame shows the House Hold Median Income by year. We can clearly see that there is some data to be cleaned with every min is -66666, representing 0. Due to this the average gets thrown off. The median Household income can be seen at the 50%. By 2019 this increased by $8k. If accurate this track with inflation. Meaning inflation on average is 2%, over the course of 10 years that comes out to roughly a little above 20%

	median_income							
	count	mean	std	min	25%	50%	75%	max
year								
2011	33120.0	-1.852902e+07	1.097405e+08	-666666666.0	36595.00	46354.0	59471.00	250001.0
2012	33120.0	-1.744154e+07	1.065713e+08	-666666666.0	36875.00	46775.0	59821.00	250001.0
2013	33120.0	-2.279601e+07	1.212914e+08	-666666666.0	36944.00	46926.5	59938.50	250001.0
2014	33120.0	-2.233244e+07	1.200994e+08	-666666666.0	37285.50	47529.0	60625.00	250001.0
2015	32157.0	-2.254584e+07	1.206529e+08	-666666666.0	38095.00	48333.0	61339.00	250001.0
2016	33120.0	-4.036731e+07	1.591133e+08	-666666666.0	37857.00	48929.0	62188.00	250001.0
2017	33120.0	-4.350572e+07	1.647643e+08	-666666666.0	39005.00	50635.5	64560.75	250001.0
2018	33085.0	-4.415424e+07	1.659040e+08	-666666666.0	40595.00	52500.0	66910.00	250001.0
2019	33120.0	-4.621912e+07	1.694563e+08	-666666666.0	41899.25	54250.0	69583.00	250001.0

	population							
	count	mean	std	min	25%	50%	75%	max
year								
2011	33120.0	9369.842512	13669.672456	0.0	716.00	2792.0	12838.00	114941.0
2012	33120.0	9445.567693	13807.690410	0.0	720.75	2786.0	12952.00	115538.0
2013	33120.0	9516.959994	13939.177211	0.0	721.00	2801.5	13000.00	114734.0
2014	33120.0	9593.274607	14090.093299	0.0	717.00	2805.5	13066.00	115013.0
2015	33120.0	9664.375151	14237.949376	0.0	718.75	2808.0	13139.25	114982.0
2016	33120.0	9724.409300	14358.657599	0.0	718.00	2807.5	13177.75	115104.0
2017	33120.0	9796.435085	14510.547644	0.0	707.00	2804.0	13290.25	119204.0
2018	33120.0	9851.278865	14614.856872	0.0	705.00	2803.5	13378.50	122814.0
2019	33120.0	9903.343961	14714.043400	0.0	705.75	2801.0	13475.25	128294.0

This process was repeated for both the County and State level.

The mapping for State Name and State abbreviation for to map the geographic visualizations were manually inserted.

```
In [24]: code = {'Alabama': 'AL',
                 'Alaska': 'AK',
                 'Arizona': 'AZ',
                 'Arkansas': 'AR',
                 'California': 'CA',
                 'Colorado': 'CO',
                 'Connecticut': 'CT',
```

Scrub

Zillow Data

For the Zillow Data the following was needed to be scrubbed:

1. Zip code needed to have leading 0's. Meaning a Zip Code is 5 digits with 0 at the front in some instances
2. The dataframe was melted to have the dates as rows instead of columns
3. The date name and the values were then called date_zestimate and zestimate.
4. NaN were dropped from the dataset completely. Due to having so much data losing about 25% wasn't a huge hit like normal datasets.

Before

In [26]:
```
zillow.head()
```

Out[26]:

	RegionID	SizeRank	RegionName	RegionType	StateName	State	City	Metro	CountyName	1996-01-31	...	2019-06-30	2019-07-31	2019-08-31	201
0	61639	0	10025	Zip	NY	NY	New York	New York-Newark-Jersey City	New York County	NaN	...	1413747.0	1405862.0	1402547.0	1390
1	84654	1	60657	Zip	IL	IL	Chicago	Chicago-Naperville-Elgin	Cook County	364892.0	...	974693.0	975616.0	975734.0	975
2	61637	2	10023	Zip	NY	NY	New York	New York-Newark-Jersey City	New York County	NaN	...	1528603.0	1514894.0	1502233.0	1492
3	91982	3	77494	Zip	TX	TX	Katy	Houston-The Woodlands-Sugar Land	Harris County	200475.0	...	335536.0	335878.0	335940.0	336
4	84616	4	60614	Zip	IL	IL	Chicago	Chicago-Naperville-Elgin	Cook County	546663.0	...	1207765.0	1208853.0	1208481.0	1206

5 rows × 300 columns

After

```
In [27]:
idvars = ['RegionID', 'SizeRank', 'RegionName', 'RegionType', 'StateName','State', 'City', 'Metro', 'CountyName']
zillow = zillow.melt(id_vars= idvars, var_name = 'date_zestimate', value_name='zestimate')

zillow = zillow.rename(columns = {'RegionName': 'Zipcode'})
zillow = zillow.drop(columns = ['RegionType','StateName'])
```

```
In [28]:
zillow
```

Out[28]:

	RegionID	SizeRank	Zipcode	State	City	Metro	CountyName	date_zestimate	zestimate
0	61639	0	10025	NY	New York	New York-Newark-Jersey City	New York County	1996-01-31	NaN
1	84654	1	60657	IL	Chicago	Chicago-Naperville-Elgin	Cook County	1996-01-31	364892.0
2	61637	2	10023	NY	New York	New York-Newark-Jersey City	New York County	1996-01-31	NaN
3	91982	3	77494	TX	Katy	Houston-The Woodlands-Sugar Land	Harris County	1996-01-31	200475.0
4	84616	4	60614	IL	Chicago	Chicago-Naperville-Elgin	Cook County	1996-01-31	546663.0
...	...	...	...	...	...	...	...	...	...
8865019	58111	35187	802	UT	Charlotte Amalie	NaN	Kane County	2020-03-31	132127.0
8865020	58115	35187	820	LA	Choudrant	Ruston	Lincoln Parish	2020-03-31	100708.0
8865021	58117	35187	822	LA	Choudrant	Ruston	Lincoln Parish	2020-03-31	181195.0
8865022	58121	35187	831	AL	Logan	Cullman	Cullman County	2020-03-31	75464.0
8865023	58125	35187	851	CO	Granby	NaN	Grand County	2020-03-31	456250.0

8865024 rows × 9 columns

Dropped Data Results:

```
In [47]:
zillow_clean = zillow_census.dropna(subset=['zestimate'])
print(1- len(zillow_clean) / len(zillow_census))
len(zillow_clean)
```

```
0.24057676455999066
```

```
Out[47]: 6662900
```

Census Data

For the census data the following was needed to scrubbed:

1. The ZCTA data was pretty clean after obtaining the data.
2. The county data needed to be reformatted to create FIPS codes which is just the state code concatenated with the county code.

In [22]:
```
census_county['state'] = census_county['state'].apply(lambda x: '{0:0>2}'.format(x))
census_county['county'] = census_county['county'].apply(lambda x: '{0:0>3}'.format(x))
```

In [23]:
```
census_county["FIPS"] = census_county["state"].astype(str) + census_county["county"].astype(st
census_county.head()
```

Out[23]:

	state	county	median_income	population	year	county_name	state_name	FIPS
0	37	043	36711.0	10506	2011	Clay County	North Carolina	37043
1	37	051	44861.0	316478	2011	Cumberland County	North Carolina	37051
2	37	081	46288.0	483081	2011	Guilford County	North Carolina	37081
3	37	099	36826.0	39574	2011	Jackson County	North Carolina	37099
4	37	139	45298.0	40511	2011	Pasquotank County	North Carolina	37139

Merging Zillow and Census Data

Merged the Zillow data with the census data. This is joining Census onto Zillow. The Zip Code in Zillow is a copy right of the postal service and the ZCTA is a copyright of the census bureau. These two Zip Codes are different, so this is not a perfect match. There is a mapping file but for this analysis combining on the Zillow Zip Code will be sufficient due to only 2% of the Zip Codes were unable to be mapped. This is opportunity to improve the precision of the analysis.

In [40]:
```
zillow_census = pd.merge(zillow, census_data,
                         how="left",
                         left_on = ["Zipcode","year"],
                         right_on = ["Zipcode","year"])

zillow = None
census_data = None
```

In [41]:
```
zillow_census.head()
```

Out[41]:

	RegionID	SizeRank	Zipcode	State	City	Metro	CountyName	date_zestimate	zestimate	year	month	state	median_income	population
0	61639	0	10025	NY	New York	New York-Newark-Jersey City	New York County	1996-01-31	NaN	1996	1	NaN	NaN	NaN
1	84654	1	60657	IL	Chicago	Chicago-Naperville-Elgin	Cook County	1996-01-31	364892.0	1996	1	NaN	NaN	NaN
2	61637	2	10023	NY	New York	New York-Newark-Jersey City	New York County	1996-01-31	NaN	1996	1	NaN	NaN	NaN
3	91982	3	77494	TX	Katy	Houston-The Woodlands-Sugar Land	Harris County	1996-01-31	200475.0	1996	1	NaN	NaN	NaN
4	84616	4	60614	IL	Chicago	Chicago-Naperville-Elgin	Cook County	1996-01-31	546663.0	1996	1	NaN	NaN	NaN

This is telling me that 2% of the zipcodes can't be mapped to the zillow dataset.

In addition I only have 2011 - 2020 worth of estimates

We are going to drop the 2% later in the model building.

An example of a Zip Code 85203

In [48]:
```
zillow_clean['zestimate_growth'] = (zillow_clean["zestimate"] / zillow_clean.groupby(['Zipcode'])['zestimate'].shift
zillow_clean['zestimate_annualized'] = (1 + zillow_clean.zestimate_growth)**12 - 1
zillow_clean[zillow_clean["Zipcode"]=="85203"].tail()
```

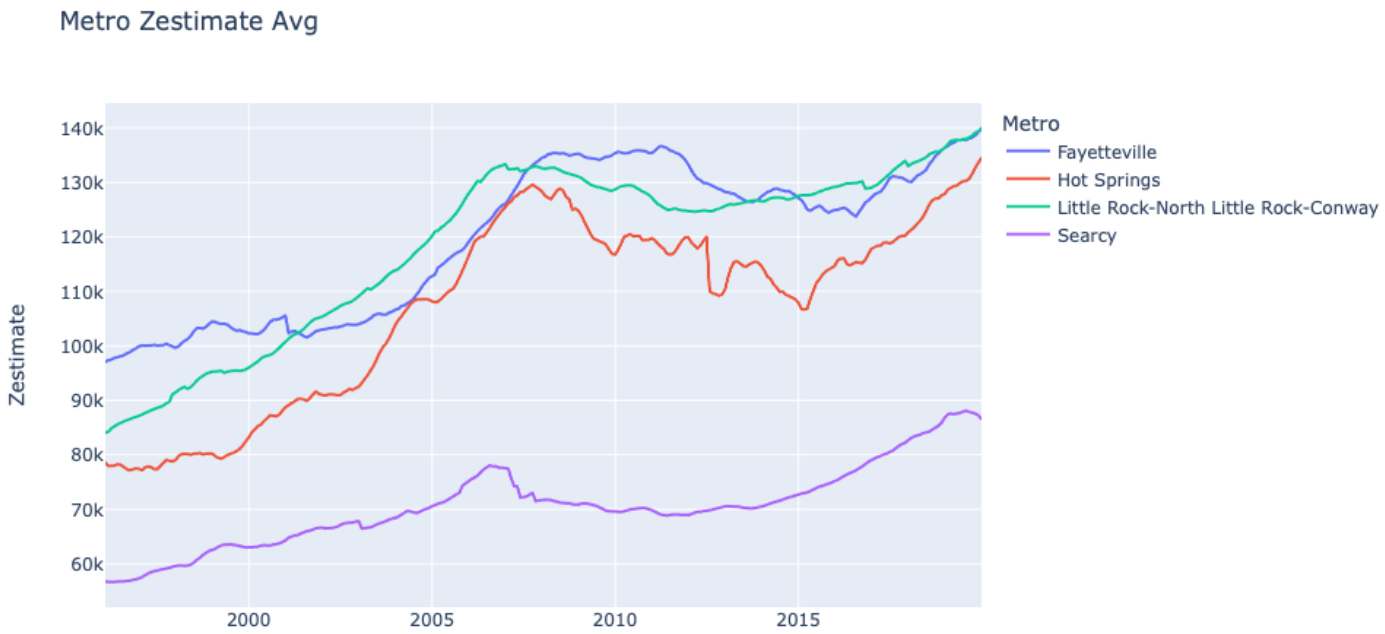
Out[48]:

	RegionID	SizeRank	Zipcode	State	City	Metro	CountyName	date_zestimate	zestimate	year	month	state	median_income	population	ze
8624435	94798	3150	85203	AZ	Mesa	Phoenix-Mesa-Scottsdale	Maricopa County	2019-08-31	274241.0	2019	8	4.0	54919.0	39797.0	
8654899	94798	3150	85203	AZ	Mesa	Phoenix-Mesa-Scottsdale	Maricopa County	2019-09-30	276154.0	2019	9	4.0	54919.0	39797.0	
8685363	94798	3150	85203	AZ	Mesa	Phoenix-Mesa-Scottsdale	Maricopa County	2019-10-31	278431.0	2019	10	4.0	54919.0	39797.0	
8715827	94798	3150	85203	AZ	Mesa	Phoenix-Mesa-Scottsdale	Maricopa County	2019-11-30	280874.0	2019	11	4.0	54919.0	39797.0	
8746291	94798	3150	85203	AZ	Mesa	Phoenix-Mesa-Scottsdale	Maricopa County	2019-12-31	283464.0	2019	12	4.0	54919.0	39797.0	

In [49]:
```
zillow_census = None
```

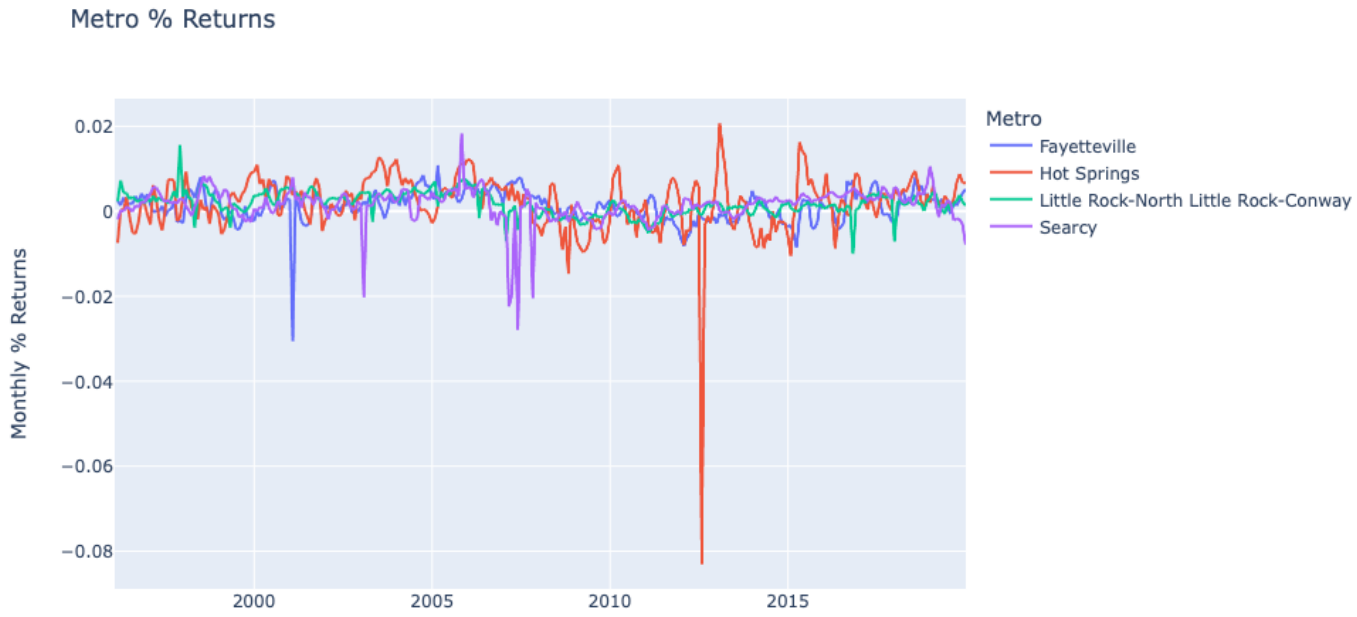
Explore

Arkansas Metro Time Series Plot



The graph above shows the 4 Metro areas Average Zestimate. In terms of growth Hot Springs appears to be the best with Little Rock being second best. This is hard to tell looking pearly out this graph. Also there is a lot of risk or volatility in all the Metro's besides Searcy. Using some finance techniques lets look at the overall Return, Risk, and Return over Risk also known as Sharpe Ratio.

Arkansas Metro Percentage Return (1997 – 2019)



This graph is a good visualization in volatility. Hot springs from 1996 until current has more volatility than the group with Searcy having the second most. Confirming our suspicions. Although Searcy appears to have done better after 2008.

Overall, from 1996 until the end of 2019 Overall Returns were as followed:

```
0.44 Return - Fayettevilee
0.71 Return - Hot Springs
0.66 Return - Little Rock
0.53 Return - Searcy
118.75 Sharpe Ratio - Fayettevilee
96.64 Sharpe Ratio - Hot Springs
238.46 Sharpe Ratio - Little Rock
125.32 Sharpe Ratio - Searcy
```

The Higher the Sharpe Ratio is better. Meaning that for every Return an investor received they took on smaller risk compared to other investments. Hot Springs return overall was 71% but an investor had to take on seen in the below chart compared to Little Rock where very risk was needed.

```
0.0037 Fayettevilee
0.0074 Hot Springs
0.0028 Little Rock
0.0042 Searcy
```

Arkansas Metro Results (2010 – 2019)

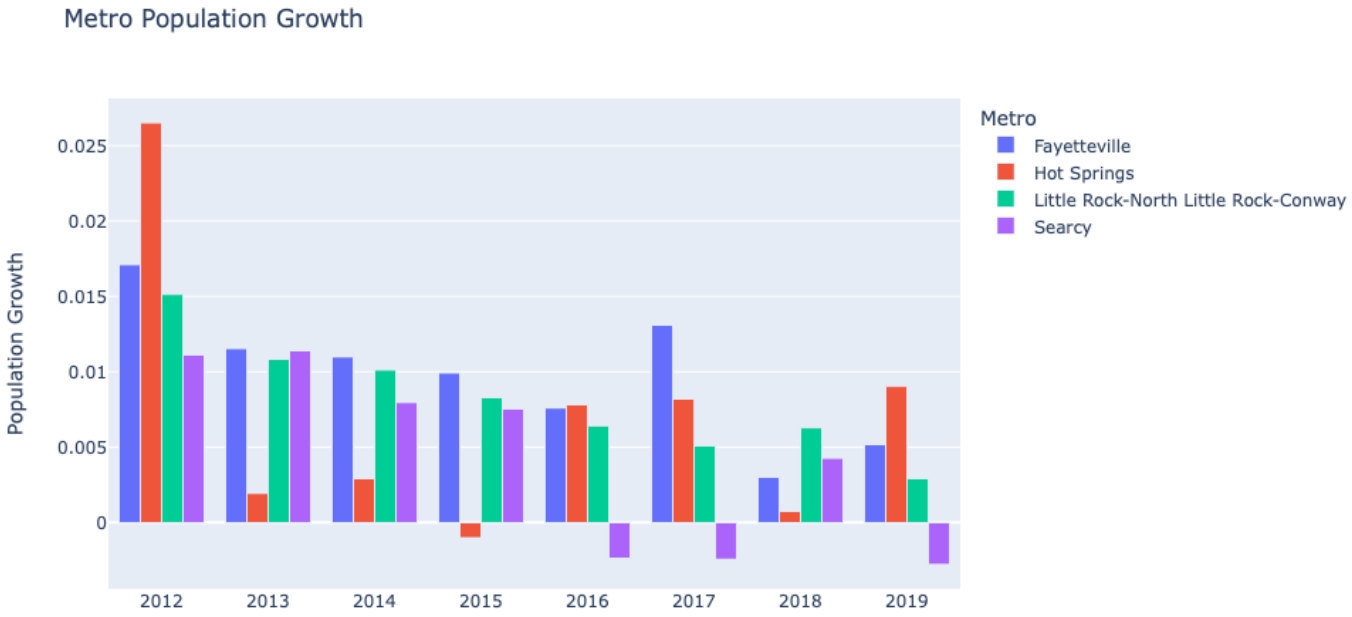
```
0.0033 std - Fayettevilee
0.0095 std - Hot Springs
0.0022 std - Little Rock
0.0025 std - Searcy
```

```
0.03 Return - Fayettevilee
0.14 Return - Hot Springs
0.08 Return - Little Rock
0.25 Return - Searcy
9.53 Sharpe Ratio - Fayettevilee
15.03 Sharpe Ratio - Hot Springs
37.04 Sharpe Ratio - Little Rock
97.68 Sharpe Ratio - Searcy
```

Before Hot Spring overall was the better choice from a historical perspective with the Return and Risk balanced very well. Looking to just this past decade Searcy has better return and a much higher Sharpe Ratio. The Riskiest area is actually Hot Springs now with Little over much smaller risk. Looking at more recent data will be important in the modeling stage of the analysis.

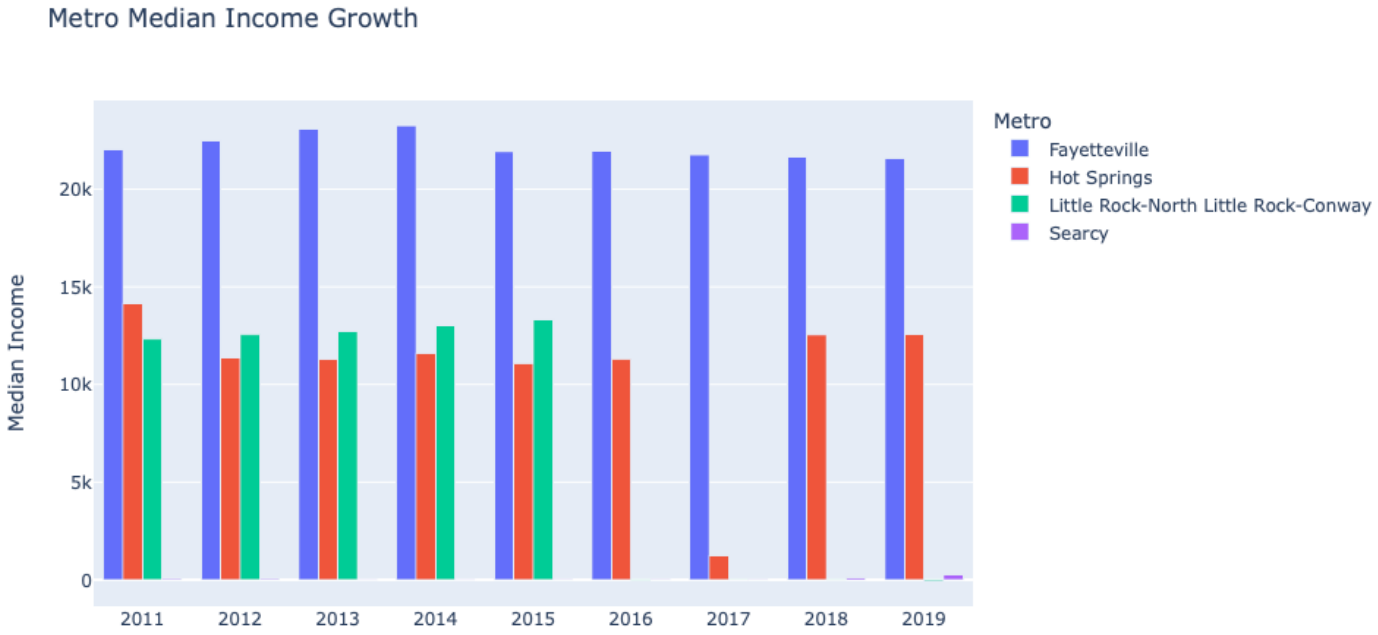
Looking at 2010 until 2019 the results are different. For the remaining of the analysis.

Arkansas Metro Population and Household Median Income (2010 – 2019)



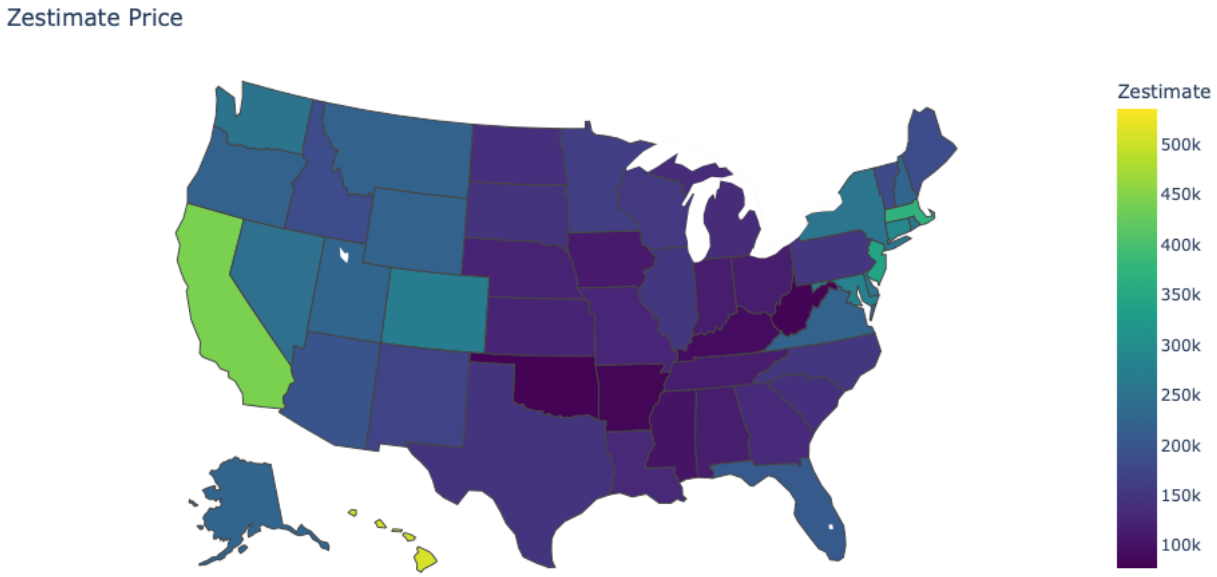
The population represented hear is just the zip codes that Zillow provided. This might not be a full representation of the population growth. There appears to be a disconnect with Searcy and Zillow data. Meaning that housing prices are going up, yet population increased the first 4 years and then is on the decline. Either this population is now becoming homeowners or houses are

disappearing from the market. Census data is not always accurate and with the 2020 census data coming next year Searcy might see an adjusted population growth.

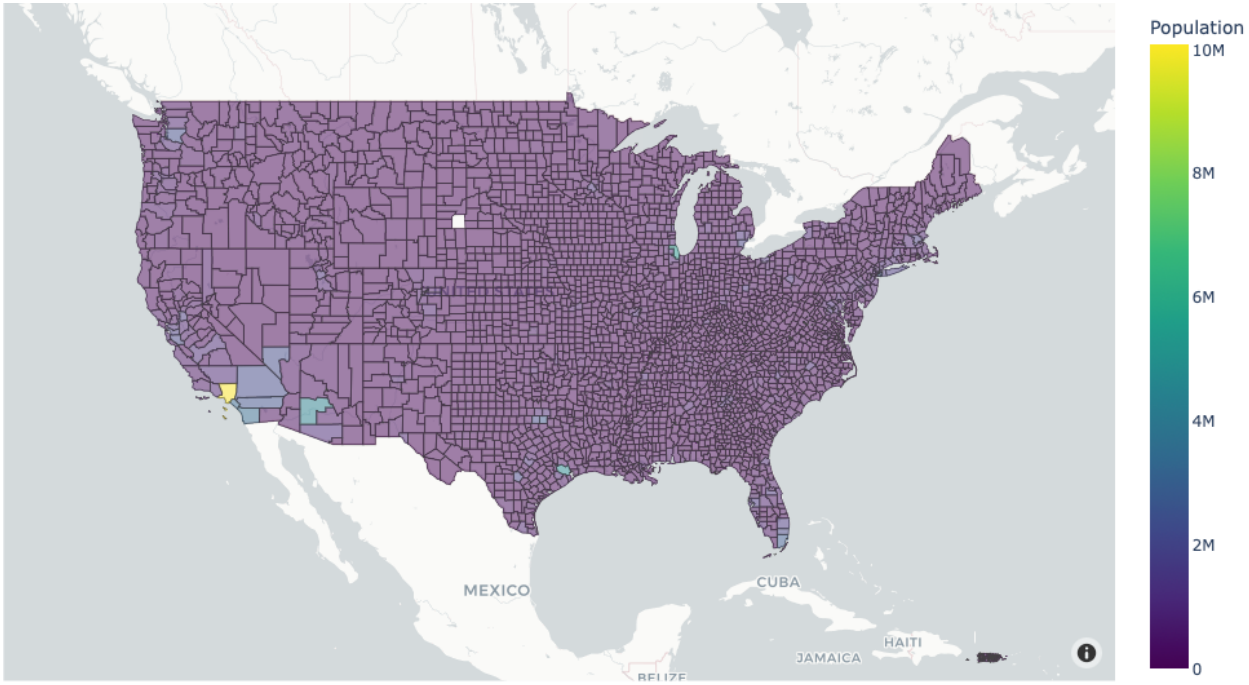


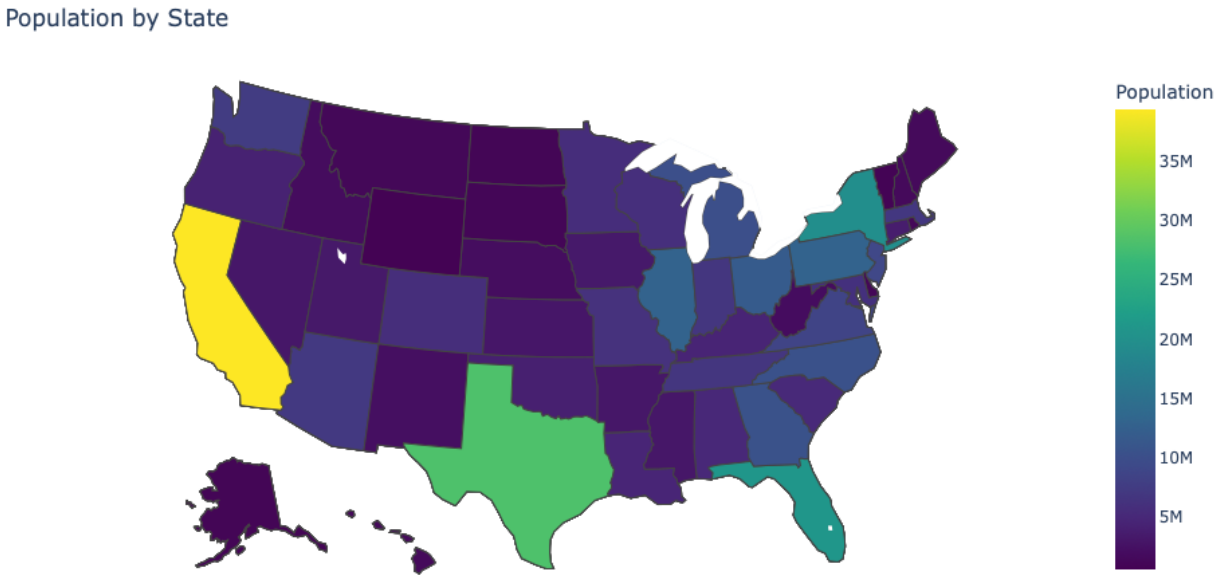
Confirming the same data issue with Searcy shows that household median income isn't increases although housing prices and population are. A big disconnect. The other graphs track with steady growth.

Bonus Geographic Visualization –Median Housing Price

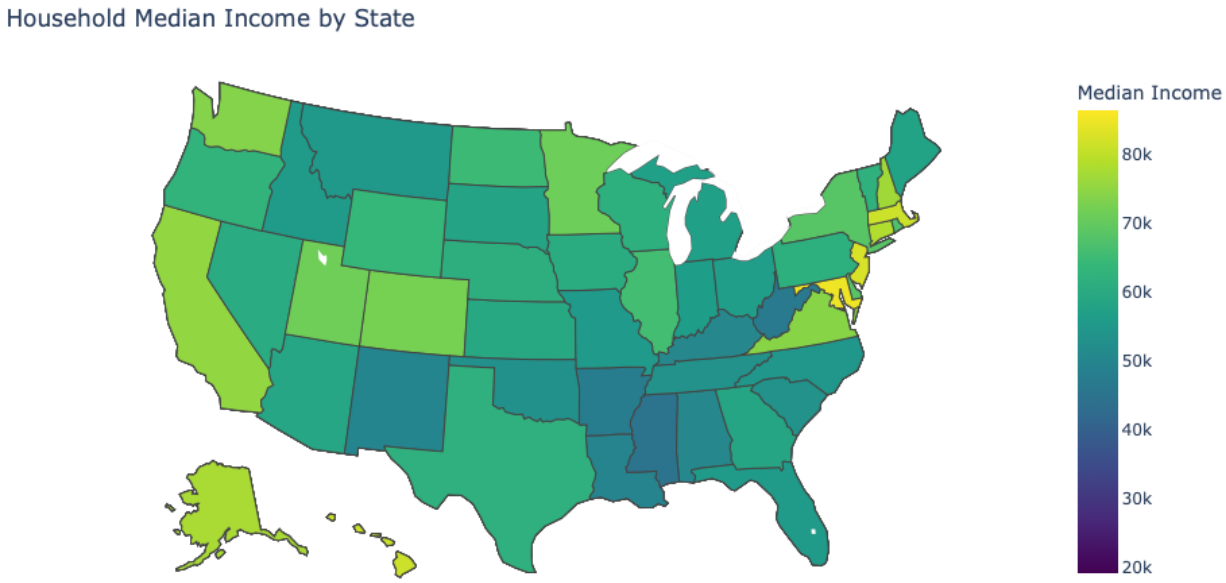


Bonus Geographic Visualization – Population





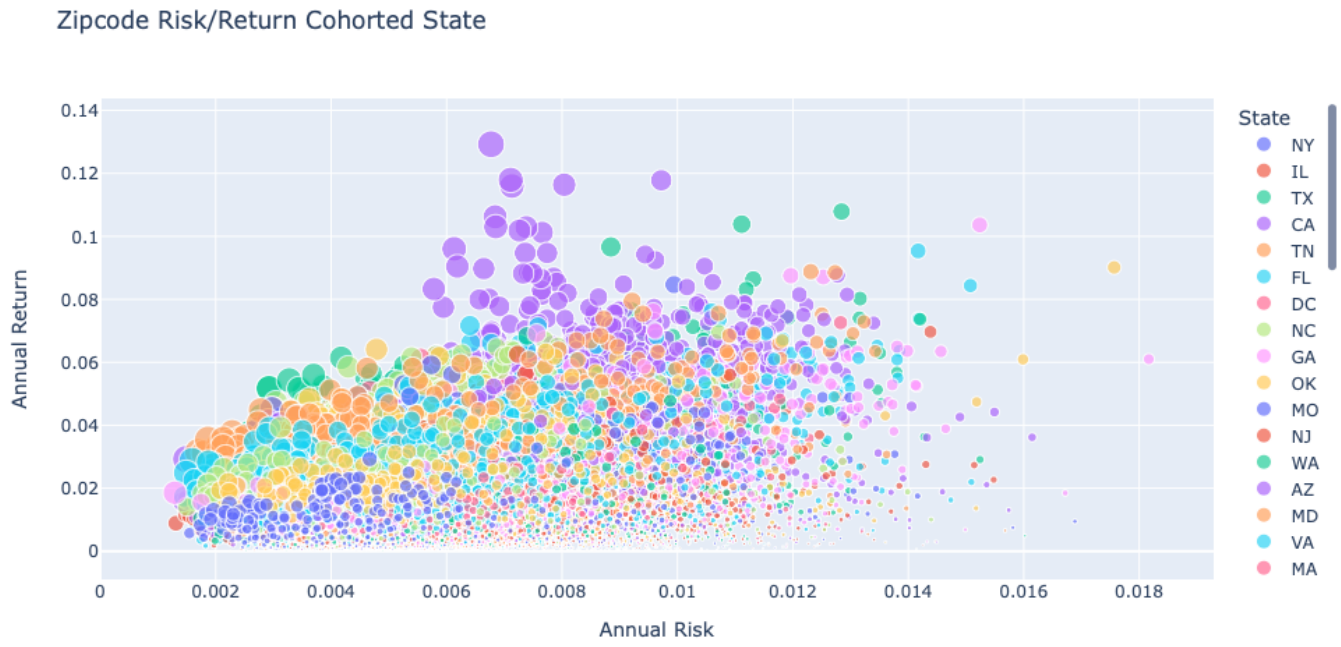
Bonus Geographic Visualization - Household Median Income



Modeling

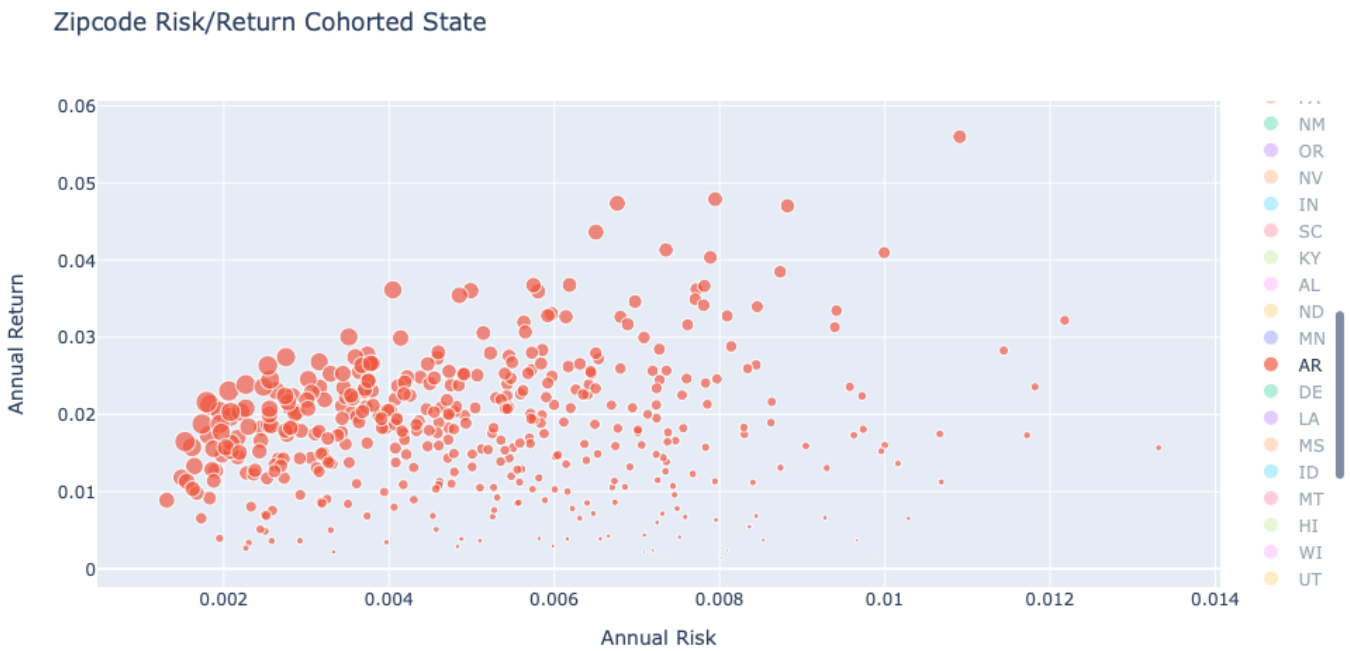
Historic Risk and Return - USA

When looking to perform the modeling historic returns should be considered by Zip code. The chart below shows all Zip Codes color coded by State. The objective in the analysis is too have the most return with lower risk. Meaning If a zip code achieves 12% return and a Standard Deviation (Risk) of 2% and another zip code achieves 12% return with a Risk percentage of 1% then taking the ladder zip code is the most optimal solution. This can also be described as a Sharpe Ratio where return is divided by risk. The higher the Sharpe Ratio means a more balance Return over Risk solution.



This graph shows every state from 2010 – 2019. The size is represented the Sharpe Ratio. As can be seen that California has a higher return with lower risk than many of the zip codes across the nation. Let's focus though on Arkansas.

Historic Risk and Return - Arkansas

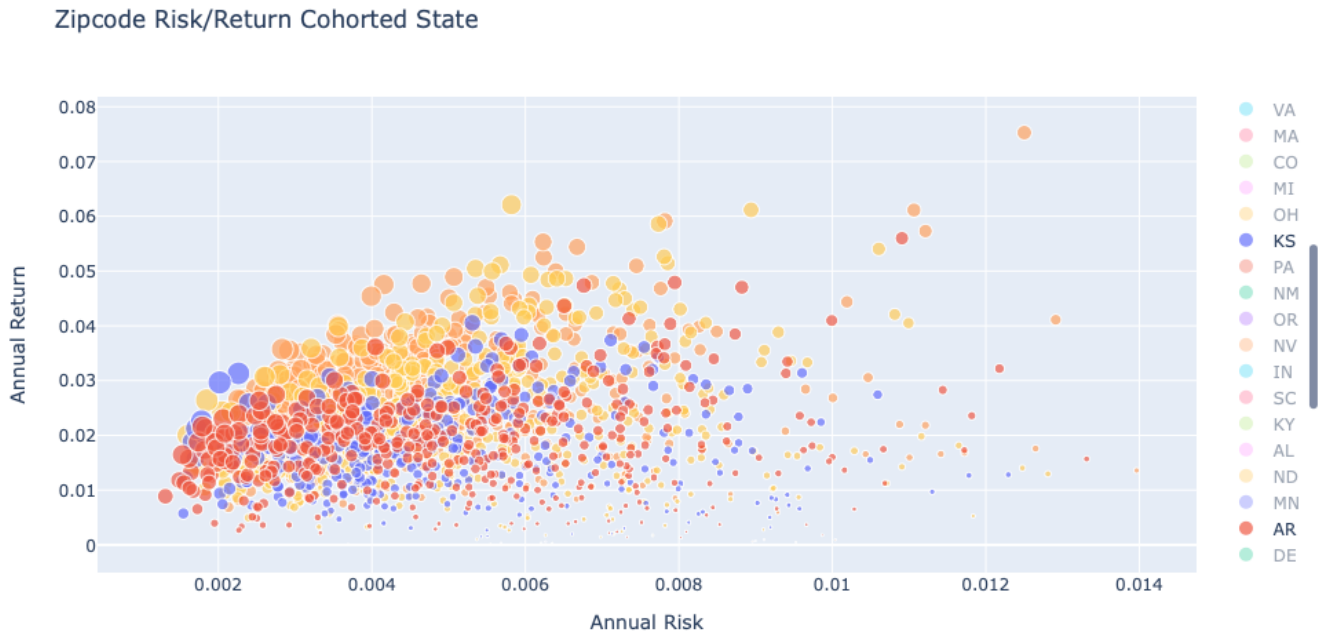


Historically Speaking zip code 72447 has the highest return at 5.5% with some of the highest risk. Depending on the risk tolerance of an investor will depend if investing in this zip makes sense.

The next 3 highest zip codes have similar returns at 5% but have different risk levels:

- 71740
- 72675
- 72645

The better of the 3 and of even the highest would be 72645 achieving 4.7% return with much lower risk than the highest.



KS = Blue
AR = Red
OK = Yellow
TN = Orange

When looking at Neighboring states, excluding Texas There is better returns over risk then Arkansas. Meaning any different color bubble than red and is above red means that there is more return for the same amount risk.

Forecasting Arkansas Return - Scrubing

When forecasting for Arkansas in 2020 and finding the 3 highest Zip codes Facebook Prophet was used in modeling these results. Before running the prophet additional scrubbing had to be done:

1. Filtered for just Zestimates from 2010 – 2019.
2. Only looked at Zipcodes with historic returns greater than 3% over the past 4 years.
3. Why looking over 3% is because inflation on average is 2% and investors minimum need to be compensated for taking on housing risk by a factor of 1%.
4. Then looked at just Arkansas which comes out to be 420 data points.
5. Changed names so that the Prophet would ingest the data.
6. Date_zestimate changed to ds and zestimate to y
7. Looped the data through prophet by zipcode and appened to an empty dataframe including the zip code as a column.
8. Found the annualized returns predicted, annual risk predicted, and the sharpe ratio predicted.
9. Please see the below code for these steps.

Looking at just a state

In [90]:
```
prophet_input =  zillow_clean[["Zipcode","date_zestimate","zestimate"]]
prophet_input = prophet_input[(prophet_input["date_zestimate"]>="2010-01-31") &
                              prophet_input["Zipcode"].isin(fronteir_filter) &
                              prophet_input["Zipcode"].isin(state_filter)
                              ]

prophet_input = prophet_input.rename(columns={'date_zestimate': 'ds',
                                              'zestimate' : 'y'})

len(prophet_input["Zipcode"].unique())
```

Out[90]: 420

In [91]:
```
prophet_input.head()
```

Out[91]:

	Zipcode	ds	y
5118283	71913	2010-01-31	137404.0
5118615	72034	2010-01-31	156265.0
5118820	72701	2010-01-31	163882.0
5118865	72764	2010-01-31	126111.0
5119182	72401	2010-01-31	104320.0

In [98]:
```
zipcode_list = prophet_input["Zipcode"].unique()
prophet_state_staging = pd.DataFrame()

for z in zipcode_list:
#for z in zipcode_list[:5]:

  prophet_zip = prophet_input[prophet_input["Zipcode"]==z]
  prophet_zip = prophet_zip[["ds","y"]].reset_index()
  forecast = run_prophet(prophet_zip)
  forecast["Zipcode"] = z
  prophet_state_staging = prophet_state_staging.append(forecast)
```

```
INFO:fbprophet:n_changepoints greater than number of observations. Using 24.
INFO:fbprophet:n_changepoints greater than number of observations. Using 23.
INFO:fbprophet:n_changepoints greater than number of observations. Using 23.
INFO:fbprophet:n_changepoints greater than number of observations. Using 20.
INFO:fbprophet:n_changepoints greater than number of observations. Using 20.
INFO:fbprophet:n_changepoints greater than number of observations. Using 18.
INFO:fbprophet:n_changepoints greater than number of observations. Using 17.
```

In [ ]:
```
prophet_state = prophet_state_staging
```

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	additive_terms_upper	multiplicat
0	2020-01-01	281164.549597	280052.789384	282239.433515	281164.549597	281164.549597	0.0	0.0	0.0	
1	2020-02-01	282541.223462	281315.715841	283606.205996	282520.580022	282541.223462	0.0	0.0	0.0	
2	2020-03-01	283829.079659	282497.956149	285154.693978	283460.668011	284050.937447	0.0	0.0	0.0	
3	2020-04-01	285205.753525	283548.256462	286654.651844	284301.996779	285861.273787	0.0	0.0	0.0	
4	2020-05-01	286538.018556	284488.383967	288235.842715	284887.231278	287765.569688	0.0	0.0	0.0	
5	2020-06-01	287914.692421	285240.945100	289953.373526	285407.169436	289754.472816	0.0	0.0	0.0	
6	2020-07-01	289246.957452	285882.474423	291757.021184	285978.341023	291879.720117	0.0	0.0	0.0	
7	2020-08-01	290623.631318	286317.166392	294263.472236	286546.221033	294111.783835	0.0	0.0	0.0	
8	2020-09-01	292000.305183	286760.059617	296426.644971	286969.188651	296321.188914	0.0	0.0	0.0	
9	2020-10-01	293332.570214	287071.378488	298707.073353	287293.743728	298770.585618	0.0	0.0	0.0	
10	2020-11-01	294709.244080	287527.296198	301228.205888	287516.505038	301213.037647	0.0	0.0	0.0	
11	2020-12-01	296041.509111	287745.459285	303752.382758	287864.496141	303595.335319	0.0	0.0	0.0	

In [108]:
```
prophet_state.head()
```

Out[108]:

	Zipcode	annualized_predicted	annual_risk_predicted	sharpe_ratio_predicted	State	City	Metro	CountyName
0	00727	0.045572	0.001158	39.353874	AR	Walnut Ridge	NaN	Lawrence County
1	00907	0.100686	0.003528	28.536166	AR	Widener	Forrest City	Saint Francis County
2	05030	0.005875	0.000133	44.296648	AR	Hoxie	NaN	Lawrence County
3	71601	0.039274	0.000969	40.547210	AR	Pine Bluff	Pine Bluff	Jefferson County
4	71602	0.036127	0.000879	41.107490	AR	White Hall	Pine Bluff	Jefferson County

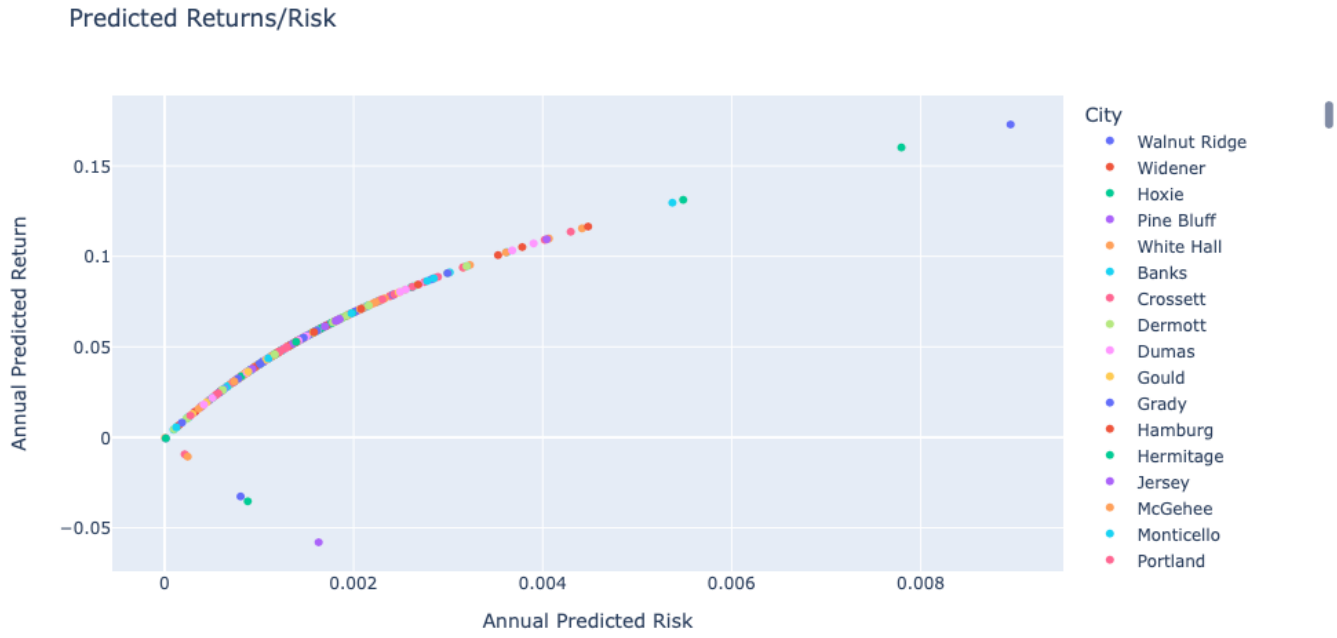
Forecasting Arkansas Return - Results

The below graphs shows that the 3 highest return for 2020:

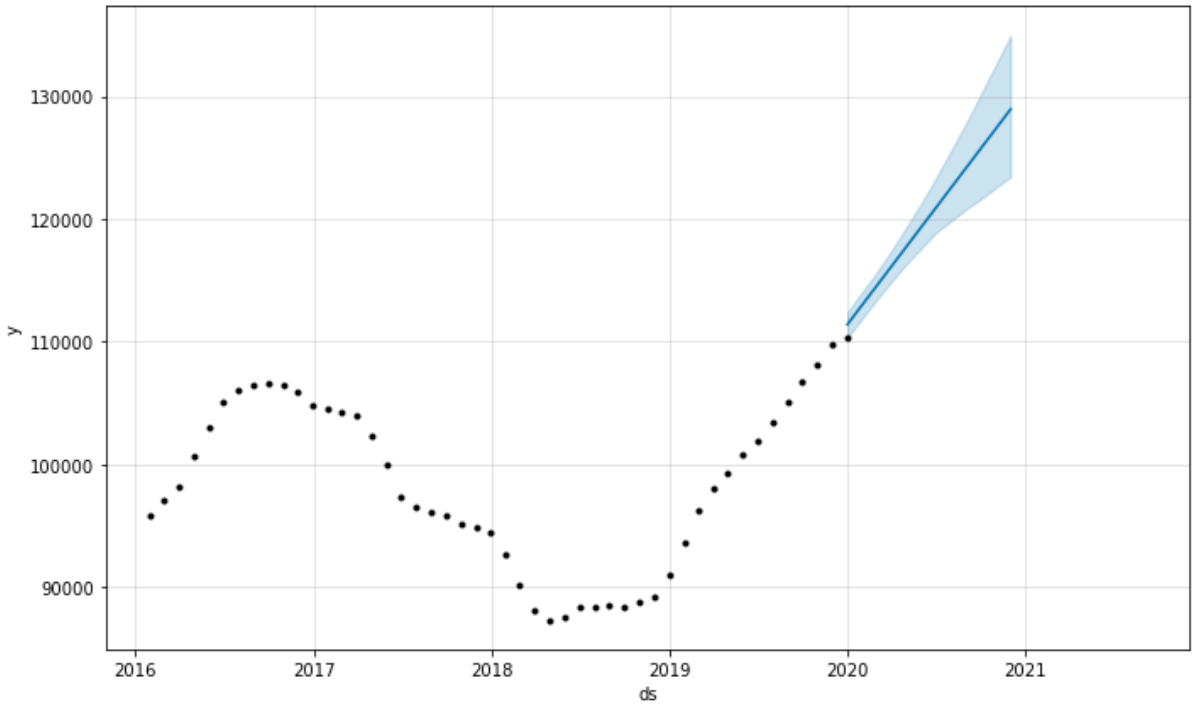
1. 72630 – Diamond City with a return of 17% and risk of 0.9%
2. 72431 – Grubbs with a return of 16% and risk of 0.8%
3. 71935 – Caddo Gap with a return of 13% and risk of 0.5%

Of the 3 that are the highest in graph 3 – 5 the predictive power illustrates the error bands in this forecast. Zip code 71935 appears to have much better predictive power of the 3 where the other's do not.

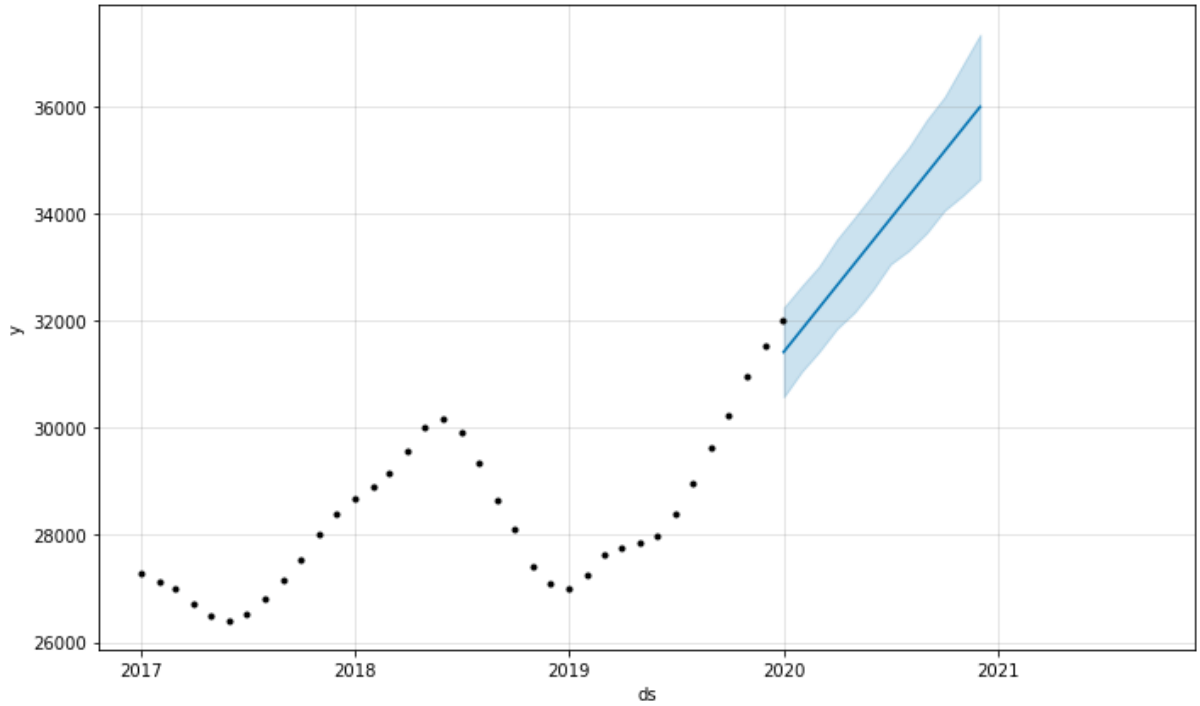
	Zipcode	annualized_predicted	annual_risk_predicted	sharpe_ratio_predicted	State	City	Metro	CountyName
0	71935	0.131250	0.005487	23.922168	AR	Caddo Gap	NaN	Montgomery County
1	72431	0.160193	0.007795	20.550340	AR	Grubbs	NaN	Jackson County
2	72630	0.172900	0.008950	19.317935	AR	Diamond City	Harrison	Boone County



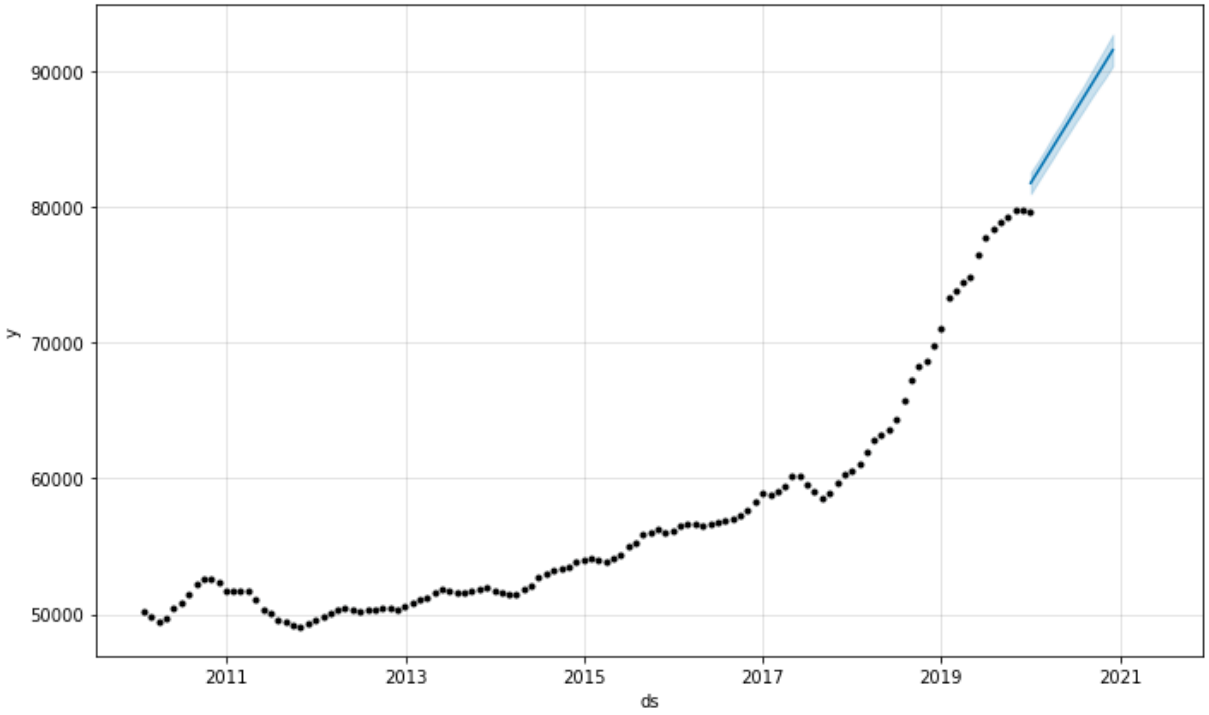
Zip code: 72630



Zip code: 72431



Zip code: 71935



	Zipcode	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	yhat
11	71935	2020-12-01	91567.496325	90426.706941	92689.232897	90690.328156	92372.641497	91567.496325
11	72630	2020-12-01	128944.672435	123551.605313	134661.304506	123813.694430	134569.823105	128944.672435
11	72431	2020-12-01	36003.709169	34743.758524	37330.122973	35024.207577	37010.344439	36003.709169

Project 3 – IST 664 Natural Language Processing

Course Description

Linguistic and computational aspects of natural language processing technologies. Lectures, readings, and projects in the computational techniques required to perform all levels of linguistic processing of text.

This course is designed to develop an understanding of how natural language processing (NLP) can process written text and produce a linguistic analysis that can be used in other applications. This goal will be achieved by:

- Readings, lectures, and class discussions of the multiple levels of linguistic analysis required for a computer to accept natural language input, interpret it, and carry out a particular application.
- Lab exercises and assignments in using some of the computational techniques required to perform these levels of natural language processing of text.
- Studies of real world applications that incorporate substantive NLP modules.

  The course primarily covers the techniques of NLP in the levels of linguistic analysis, going through tokenization, word level semantics, part of speech tagging, syntax, semantics, and on up to the discourse level. It also includes the use of the NLP techniques, such as information retrieval, question answering, sentiment analysis, summarization, and dialogue systems, in applications.

Learning Objectives:

At the end of the course the student will be able to:

- Demonstrate the levels of linguistic analysis, the computational techniques used to understand text at each level, and what the challenges are for those techniques.
- Process text through the language levels using the resources of the Natural
- Language Toolkit (NLTK) and some rudimentary use of the programming language Python.
- Describe how NLP is used in many types of real world applications.

Deliverables(s) – Classifying IMDB Movie Reviews

Can reviews be used to predict rating, or better yet can reviews be deconstructed using multiple Natural Language Processing techniques and then apply machine learning to predict a rating. For this project IMDB reviews were used across many different movies to see the predictive power in what people say.

Project Process & Development

Overall

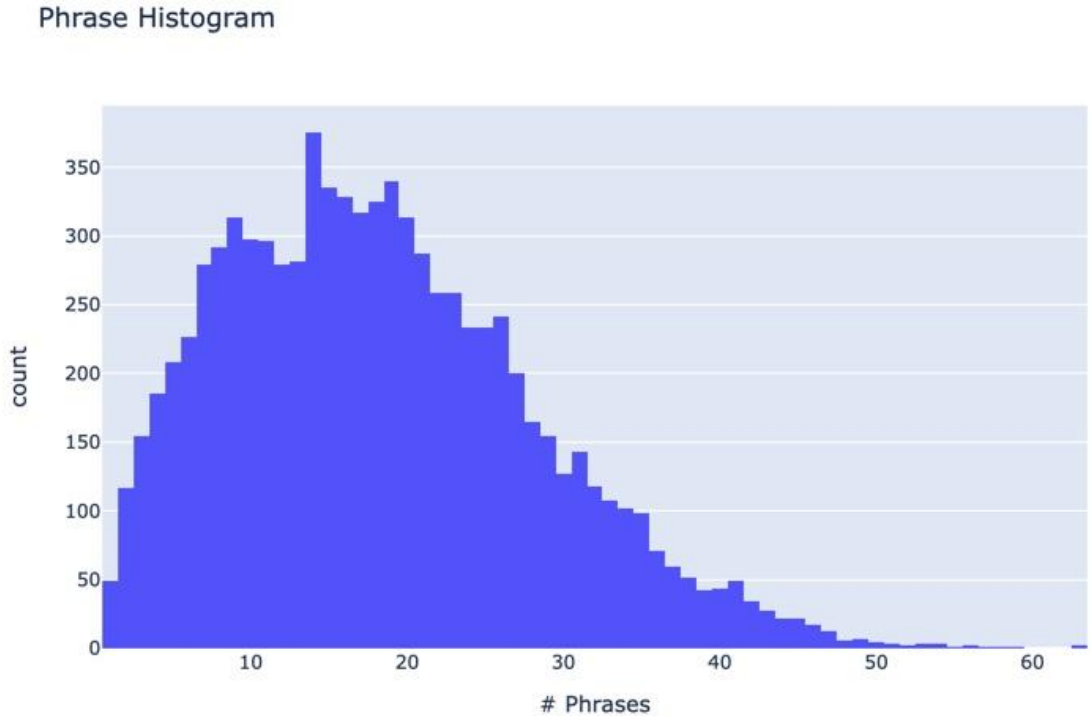
The goal we set out today is to predict the rating of a list of comments using classification algorithms and different cleaning techniques. The exhaustive list of items includes tokenizing, filtering, pre-processing the word list. Then apply feature engineering techniques that span Unigrams, Bigrams, and all the way to subjectivity. Then use multiple models like Bayes Classifier, Random Forest, and Support Vector Machine Classification. To round off the analysis, we used cross-validation to help prove the results will help in the prediction task.

No stone was left un-turn, with 30 different permutations of accuracy scores created based on this analysis's filtering, feature engineering, and models. Let’s jump in and see how well we can apply a rating to a comment?

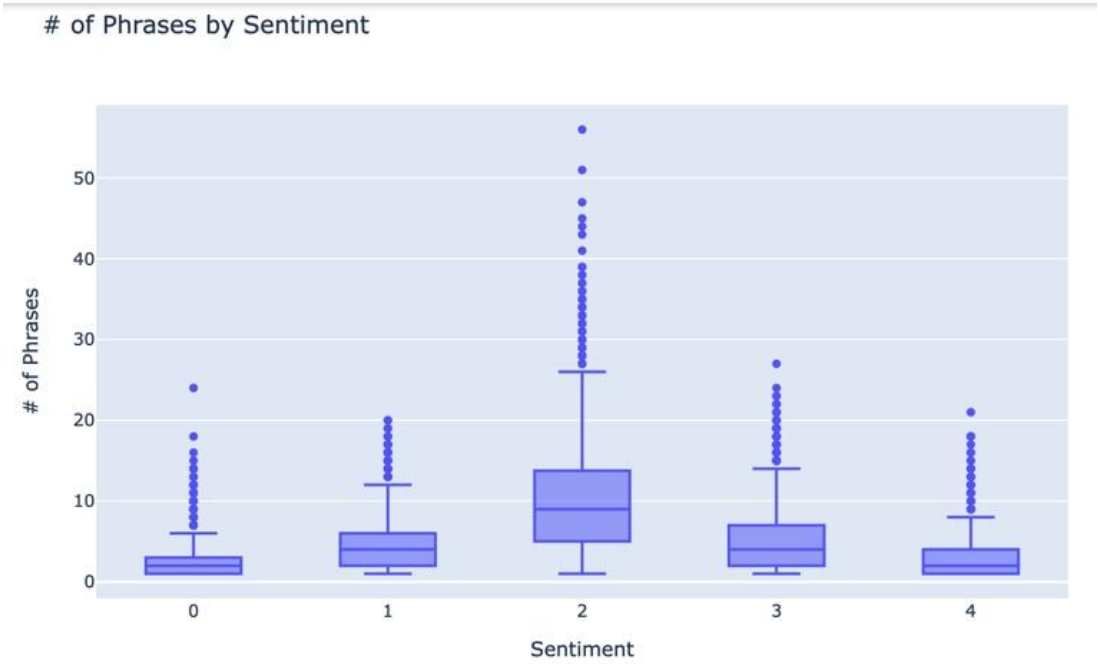
Reviewing the Data

The dataset we chose to review and analyze was the Kaggle movie review list. This dataset was produced for the Kaggle competition, which uses data from the sentiment analysis by Socher et al. The data was originally taken from the Pang and Lee movie review corpus based on reviews from Rotten Tomatoes website. Socher's group used crowd-sourcing to manually annotate all the sub-phrases of sentences with sentiment label scoring. The ranges included the following: "negative", "somewhat negative", "neutral", "somewhat positive", "positive".

Before reviewing the original dataset we needed to import all the necessary packages needed for preprocessing and filtering. First we created training and test data frames using the given train.tsv and test.tsv files that were given to us. We wanted to review exactly how many unique sentences and compare them to the total number of sentences supplied in the training dataset. The total number of full sentences (also considered "unique_sentences") were 8520, out of a total of 156,060 listed phrases (also considered "total_sentence"). That is approximately 18 times the amount of unique sentences. Next, we assign a "FullSentenceId" to each phrase, while grouping them by the SentenceId, this way we can see which smaller phrases are assigned to which completed full sentence. After putting the training data into a table with the new columns we could see that this dataframe was similar to a break down tree, similar to the grammar context previously learned in the course. To better review this discovery, we created a phrase histogram that compared the count along the y-axis and number of phrases along x-axis. The number exceeded above 350 count as its peak early on at approximately 15 phrases as you can see below.



To continue our visualization analysis of the training data, we then compared the number of phrases to the sentiment scores that were assigned to them. As scores 1 and 4 were the least, the score of 2 had the largest upper and lower bounds with exceeding whiskers past the 50 count of assigned phrases as you can see below. This essentially confirmed that there was a lot of cleaning and filtering that needed to be done prior to creating a new model to run our experiments on.

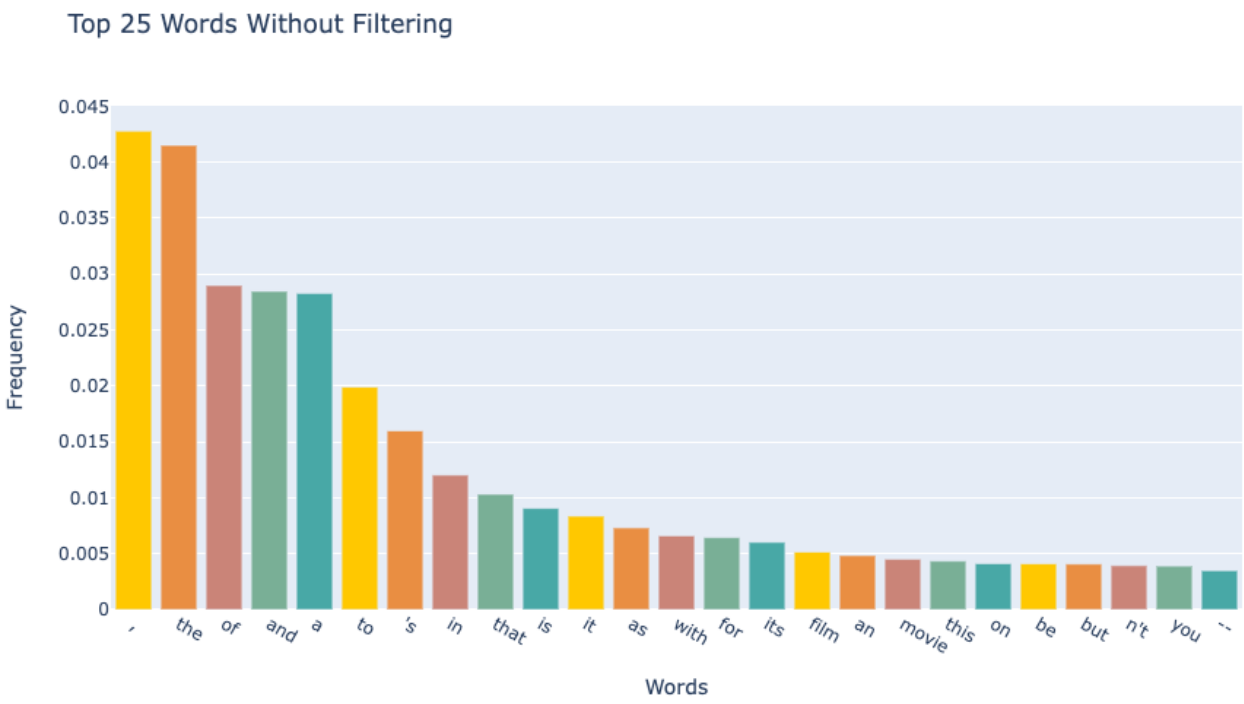

# of Phrases by Sentiment
# of Phrases
50
40
30
20
10
0
0
1
2
3
4
Sentiment

Tokenized data

In order to tokenize our data we first join the training data frame creating the "Phrase" column into 'characters.' Using a nltk.word function that tokenizes the testtext vector to make our tokens. And to make sure all of our characters are seen the same, we made sure they were all lowercase into a new vector named testwords. The length of testtokens outputted 981,478 total words without filtering. We then wanted to dive deeper reviewing our bag ("test words") of words by using the FreqDist function. 25 of the most common words based on the frequency were displayed to see the depth of filtering that needs to be done.

```
,       0.04279464236590122
the     0.04151901519952561
of      0.028987914145808667
and     0.02846625191802567
a       0.028286930527225265
to      0.019897542278074495
's      0.01599220767047249
in      0.012034910614399916
that    0.010320149814871041
is      0.009058786850036374
it      0.008362897589146165
as      0.007317535390502895
with    0.006610438542687661
for     0.006450475711121391
its     0.00602356853643179
film    0.005148357884741176
an      0.004837602065456383
movie   0.004518695273862481
this    0.004342430497677992
on      0.00412235424533204
be      0.004100957943020628
but     0.004079561640709216
n't     0.003939976239915719
you     0.003904315736063366
```

As you can see below, one of the top results was a comma and other "words" such as "'s" or "n't" were considered in that top 25 of most common. Therefore, filtering needs to be done in order to get a better accuracy of our dataframe. Below is a better visualization of the word frequency without filtering:

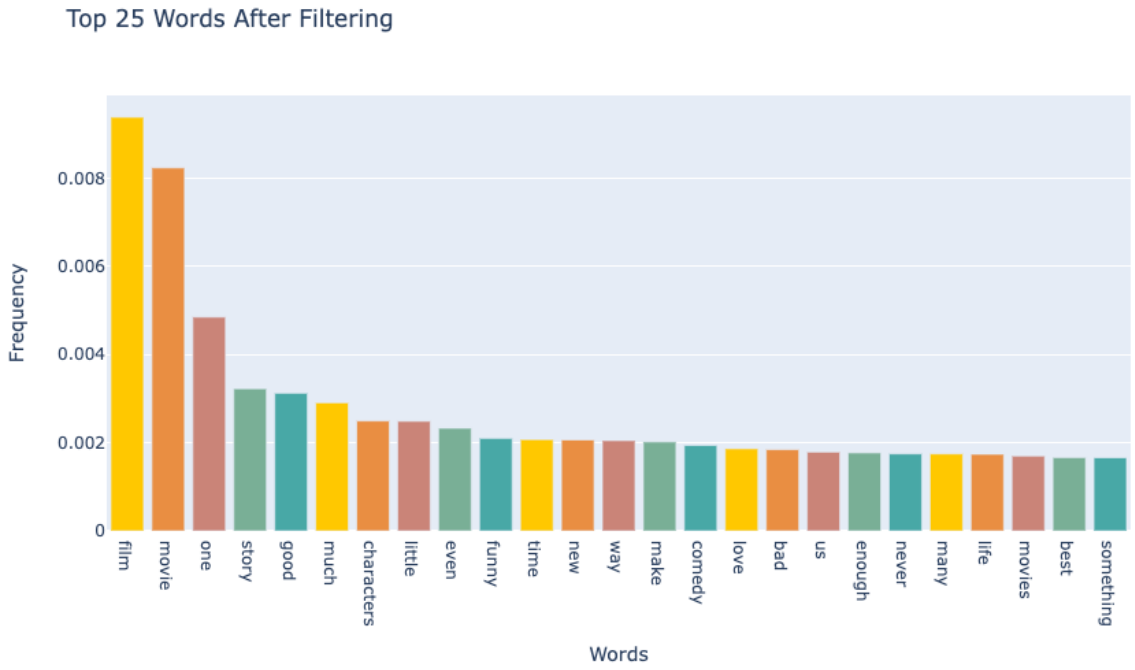

Top 25 Words Without Filtering
Frequency
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
, the of and a to 's in that is it as with for its film an movie this on be but n't you --
Words

Filtered and tokenized data

First step we decided to do was remove punctuations from our dataframe using the list(puncuation) function and apply that to the testwords vector we already had stored. The length of words decreased to 9,30730. The frequency distribution of our testwords using the 25 most common were compared cause a slight change. Next, we remove any non-alphabetical characters in our test words with Regular Expression practices by using the re.compile('^[^a-z]+$') function within defining an alpha_filter. Doing this significantly causes 10% of our test words to decrease. However, we still need to remove periods as well. We repeat this re.compile('\ |\?|\.|\!|\/|\;|\:') function toward the newly created alphatestwords vector. This also decreases our number of testwords again.

Pre-processed and tokenized data

Now that unnecessary characters have been removed from the training dataset, we have to focus on the words tokens themselves. A lot of the time, English stop words decrease accuracy or natural language processing so it is best to remove them to better our models. We obtained the list of English stop words from the NLTK corpus (total of 179). Then, we created a new vector of stoppedtestwords that removes the nltkstopwords from the alphatestwords. A drastic decrease in total words was applied here making a total of 568,580 words in the training dataset. Another frequency distribution was applied to the new total words vector. Some characters that are not considered valid words were displayed in the most common list. Taking another list of morestopwords taken from a previous lab was used in addition to other characters from the frequency distribution and then added to the nltkstopwords. Ultimately, this help increase the frequency distribution for a majority of the most common characters as you can see below:



Another form of tokenizing data and pre-processing is stemming and lemmatizing words. This essentially is mering the words together in order to increase the frequency overall. For stemming we apply the porter.stem function toward the stoppedtestwords and join that with the total words vector. For lemmatizing we apply the word_lemma.lemmatize function as well to the previously

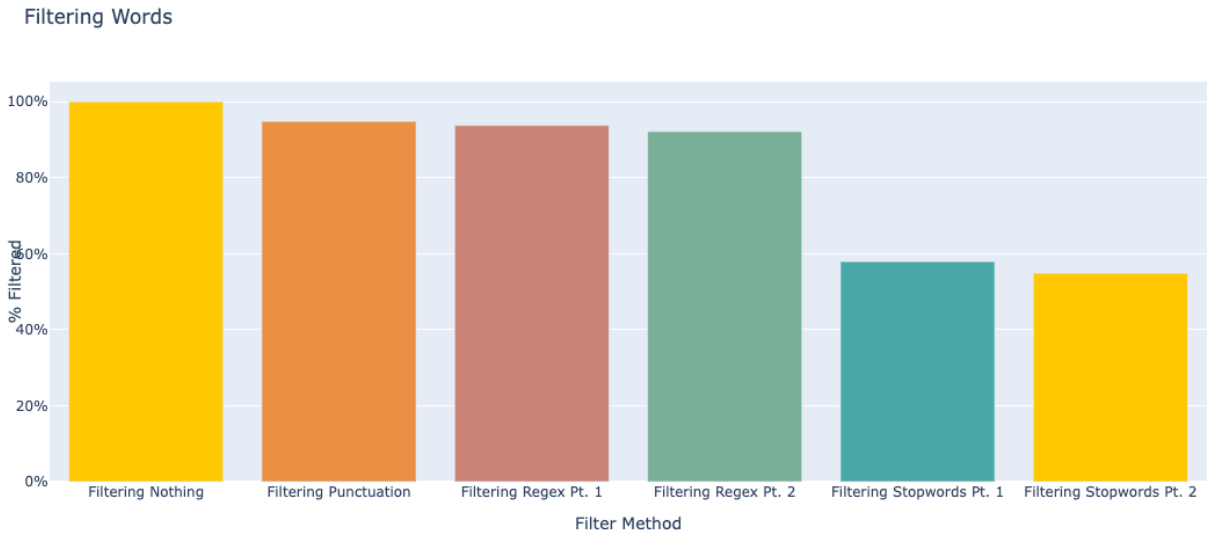
created stemmedtestwords. This exercise did not decrease the number of words but did help improve the frequency. That is due to words being combined together and therefore increased the occurrence of the words.

The picture below illustrates how stemming and lemmatizing moved the word film from 0.008 to 0.01.

```
film      0.010609335769785028
movi      0.009937083091916626
one       0.005067893806361516
make      0.004133796853411776
charact          0.003938806435798012
stori     0.0035766813745153093
good      0.003181129384498819
much      0.0029081427998395508
time      0.002898857541857943
littl     0.00248844913907088
work      0.0024253093847959473
even      0.0024197382300069825
way       0.0023825971980805513
love      0.0022618888443196507
comedi    0.002156036903329322
feel      0.00215232280013679
funni     0.0021188958714028912
look      0.002115181768210248
new       0.0020706125298985308
get       0.0019276195569817712
bad       0.0018681939058994815
perform          0.0018626227511105169
u         0.0018069112032208703
see       0.0018031971000282271
enough    0.0017716272228907608
```

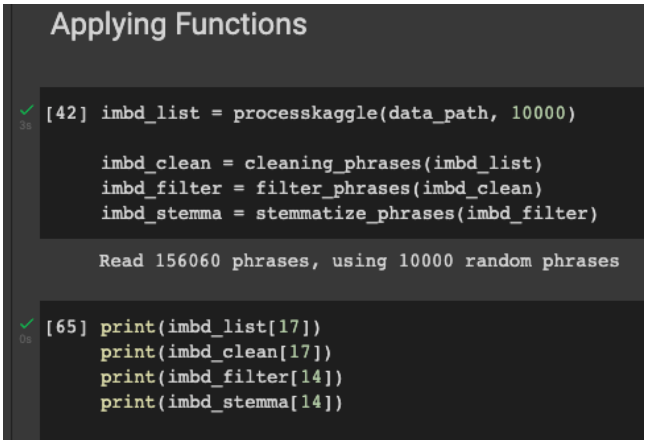
In summary the chart and graph below shows the effect of each filtering technique on that data. What can be seen is that filtering stop words nearly cut the total words in half with the other technique cutting about 8% from the data.

filter_type	num_words	perc_filter
Filtering Nothing	981478	1
Filtering Punctuation	930730	0.948294
Filtering Regex Pt. 1	920174	0.937539
Filtering Regex Pt. 2	904479	0.921548
Filtering Stopwords Pt. 1	568580	0.57931
Filtering Stopwords Pt. 2	538488	0.54865


Filtering Words
100%
80%
60%
40%
20%
0%
% Filtered
Filtering Nothing
Filtering Punctuation
Filtering Regex Pt. 1
Filtering Regex Pt. 2
Filtering Stopwords Pt. 1
Filtering Stopwords Pt. 2
Filter Method

Pre-processed, filtered, and tokenized data

The Final Step was to build a function that leverages the pre-processing and filtering techniques discussed above to then apply to each individual sentence. First, we defined a processkaggle function to read the kaggle training file, train and test a classifier. This incorporated a limiting argument, looping over lines, picking a random sample of length, and then printing the phrase data. A cleaning_phrases function was then defined by creating a list of phrase documents that first tokenizes the list of words, then makes those tokens lowercase, removes punctuations, and removes any tokens that are less than 1 at length. The next function created was filter_phrases which incorporated tokenizing again, removing non-alphabetical characters, removing words under three characters, removing all stopwords, and again removing any tokens that are less than 1 at length. Lastly, the stemming and lemmatize techniques were combined into one stemmatize_phrase function which tokens the phrases again, stemming, lemmatizing, and returning a final token list. Below are screenshots of applying these condensed functions to our imdb dataset and printing out the final breakdown of these words.



Output:

['see a study in contrasts ; the wide range of one actor , and the limited range of a comedian', '2']
(['see', 'a', 'study', 'in', 'contrasts', 'the', 'wide', 'range', 'of', 'one', 'actor', 'and', 'the', 'limited', 'range', 'of', 'a', 'comedian'], 2)
(['see', 'study', 'contrasts', 'wide', 'range', 'one', 'actor', 'limited', 'range', 'comedian'], 2)
(['see', 'studi', 'contrast', 'wide', 'rang', 'one', 'actor', 'limit', 'rang', 'comedian'], 2)

What can be seen here is that the first sentence is not tokenized. Each of these sentences have the rating at the far right. The next 3 in the output above shows the sentence tokenized with everything lowercase and punctuations removed. The third sentence filters any stop words from NLTK and the custom list created. The last sentence shows the Stemmatize and Lemmatize technique. This can be seen with the word “study” changing to “studi”.

Feature Engineering

The objective for feature engineering is to create the featuresets by passing the sentences into a function that will add a true or false value for every unigram, bigram, Part of Speech etc. A lexicon of different features like part of speech, sentiment, or subjectivity can be applied. More of this will be discussed further in the document.

Bag of Words & Unigrams Features

This type of feature tokenizing every word in the sentence applies a True or False rating. At the end of Unigram dictionary contains the sentiment. This can be seen in the chart below.

```
({'V_film': False,
  'V_movi': False,
  'V_one': False,
  'V_stori': False,
  'V_make': False,
  'V_charact': False,
  'V_time': False,
  'V_good': False,
  'V_even': False,
  'V_work': False},
 1)
```

To begin our feature engineering we first get all the words from our dataset imbd_stemma and put it into a frequency distribution. The output of the printed length returns 7056. Out of that total, we take 1000 of the most common word items and create a word features vector. Next, we define a document_features function that takes the keywords of a document for bag of words and unigram baseline. Imbd_clean is stored back into a document vector to store so we can replace this instead of all the individual features. After creating a new dataframe called feturesets that includes the previously stored documents, keyword features and category features. Doing this increases the length up to 9998.

Bigrams Features

A bigram group two words together in a phrase and then assigns a rating of True or False. For example “Dan is the man” would translate to “Dan is”, “is the”, “the man”. Then a chi squared measure is used to find the top 500 bigrams. The feature set will then include just these word phrases.

We used the nltk.collaction.BigramAssocMeasures function on all the words in our sequence and stored that into a finder vector. Next, we define the top 500 bigrams using the chi squared measure and store that into a bigram_features vector. We then define a bigram_document_feature using both document, word_features, and bigram_features described

beforehand. Using this function will create feature sets for all the sentences. To test these bigrams we use the train_set again to test a classifier and report the accuracy at 45.6%

({'B_10-year_delay': False,
'B_18-year-old_mistress': False,
'B_22-year-old_girlfriend': False},1)

Part-of-speech Features

A lexicon was used here to understand how the part of speech might play a role in a sentence and in a model. In the second picture below we can see the number of nouns, verbs, adjectives, and adverbs for the first sentence.

The POS feature takes a document list of words and returns a feature dictionary in order to run the default POS tagger (Standord tagger) on the document. This way the feature can count 4 types of POS tags to use as features, such as, nouns, verbs, adjectives, and adverbs. The length of POS_features outputted 1004. Below is an example of the first sentence.

```
[ ] # the first sentence
    print(documents[0])
    # the pos tag features for this sentence
    print('num nouns', POS_featuresets[0][0]['nouns'])
    print('num verbs', POS_featuresets[0][0]['verbs'])
    print('num adjectives', POS_featuresets[0][0]['adjectives'])
    print('num adverbs', POS_featuresets[0][0]['adverbs'])
```

```
(['do', 'well', 'to', 'cram', 'earplugs'], 1)
num nouns 1
num verbs 1
num adjectives 0
num adverbs 2
```

Sentiment Lexicon (LIWC)

This feature is also considered linguistic inquiry and word count. This form of text analysis essentially calculates the degree to which the various categories in our kaggle training dataset of words are used in text. This lexicon assigned a positive or negative rating then adds to either a pos or negative list in the dictionary. A sample of data can be seen in the jupyter notebook and is very similar to the other examples above.

Using the sentiment_read_LIWC_pos_neg_words.read_words function was stored into a path vector, which was then separated into a positive and negative path. We created a LIWC_feature that includes word counts of the subjectivity words. The negative features will have a number of weakly negative words and 2 times the number of strongly negative words. The final length of LIWC_featuresets outputs 9998.

Subjectivity

To begin engineering subjectivity, we imported in the necessary readSubjectivity libraries from the Kaggle movie reviews. Once the SLpath was stored into a vector, we defined the features that included word counts of the subjectivity words plus 2 times the number of strongly positive and negative words. Keeping in mind that positive features have similar definitions, not counting the neutral words. If the word was tagged as “weaksubj” or “strongsubj” it would now be considered positive or negative. The final length of SL_features outputted 1002.

Experiments

Bayes Classifier

Once our condensed features were created, we decided to test out our training_set and test_set using naive Bayesian Classifier. After taking 500 random samples of featuresets within the train_set and test_set, we used the NLTK Naive Bayes Classifier function and evaluated the accuracy and it came out to be 53.8%. Using the bigram_featuresets the accuracy resulted in 54.4%. Using the POS_featuresets to train and test the classifier our accuracy increased slightly to 54.6%. Lastly, the subjectivity lexicon resulted in a 56.4% accuracy and sentiment LWIC featuresets resulted in 53.8%.

Cross Validation

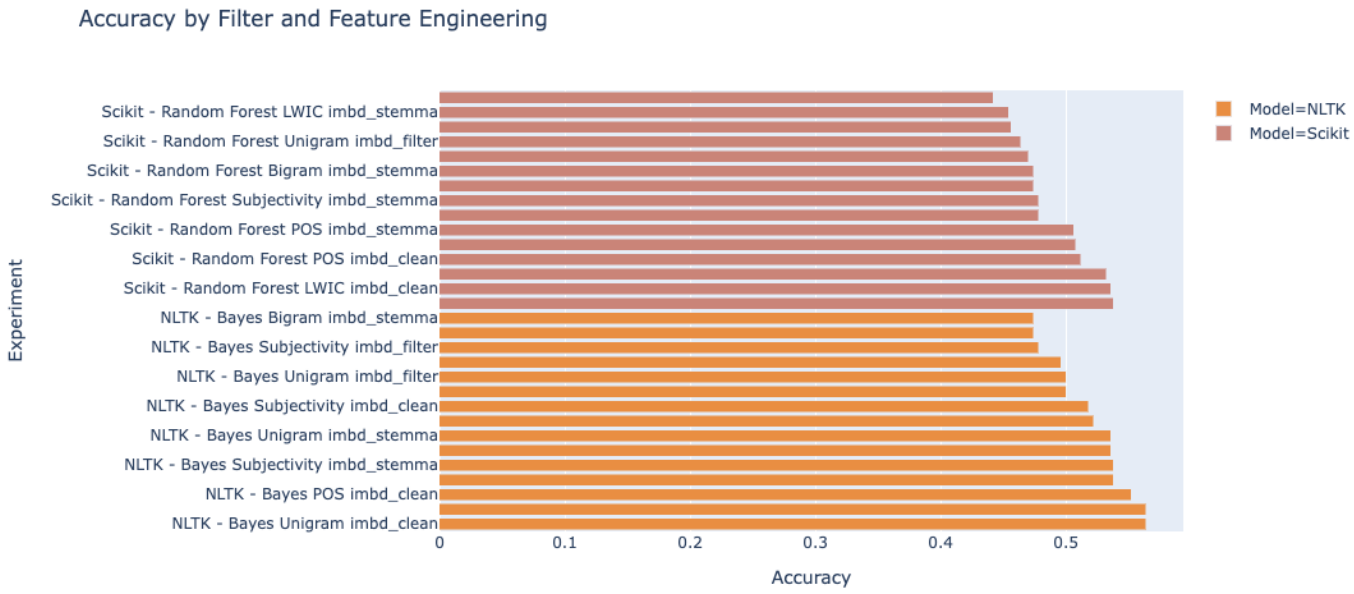
In the cross validation experiment we created a cross_validation_accuracy function that the number of folds and feature sets, then iterates over the folds. By using different sections for training and testing, this function will print the accuracy for each fold and the average accuracy at the end of the output. When using the unigram word features and using a number fold of 3 the mean accuracy resulted in 48.0% with each fold size being 3332. When using the bigram_featuresets and the same number of folds, the mean accuracy also resulted in 52.1%. Using the POS_featuresets within the cross_validation_accuracy function, the mean accuracy resulted in 53.9%. Lastly, using the subjectivity lexicon, the mean accuracy resulted in 53.1%.

Sci-kit Learn - Advance Task

Sciki-Learn is another machine learning program used for word efficiency models like NLTK. However, it was used within our program to see if the applications and accuracy of our models were in comparison to our other classifier and feature combinations. The first classifier we used was a random forest using the POS_featuresets and that accuracy result was higher than the NLTK with 57.8%. The next one we used was the Support Vector Machine classifier using the same POS_featuresets and that resulted in 54.6%, similarly to the NLTK.

Results

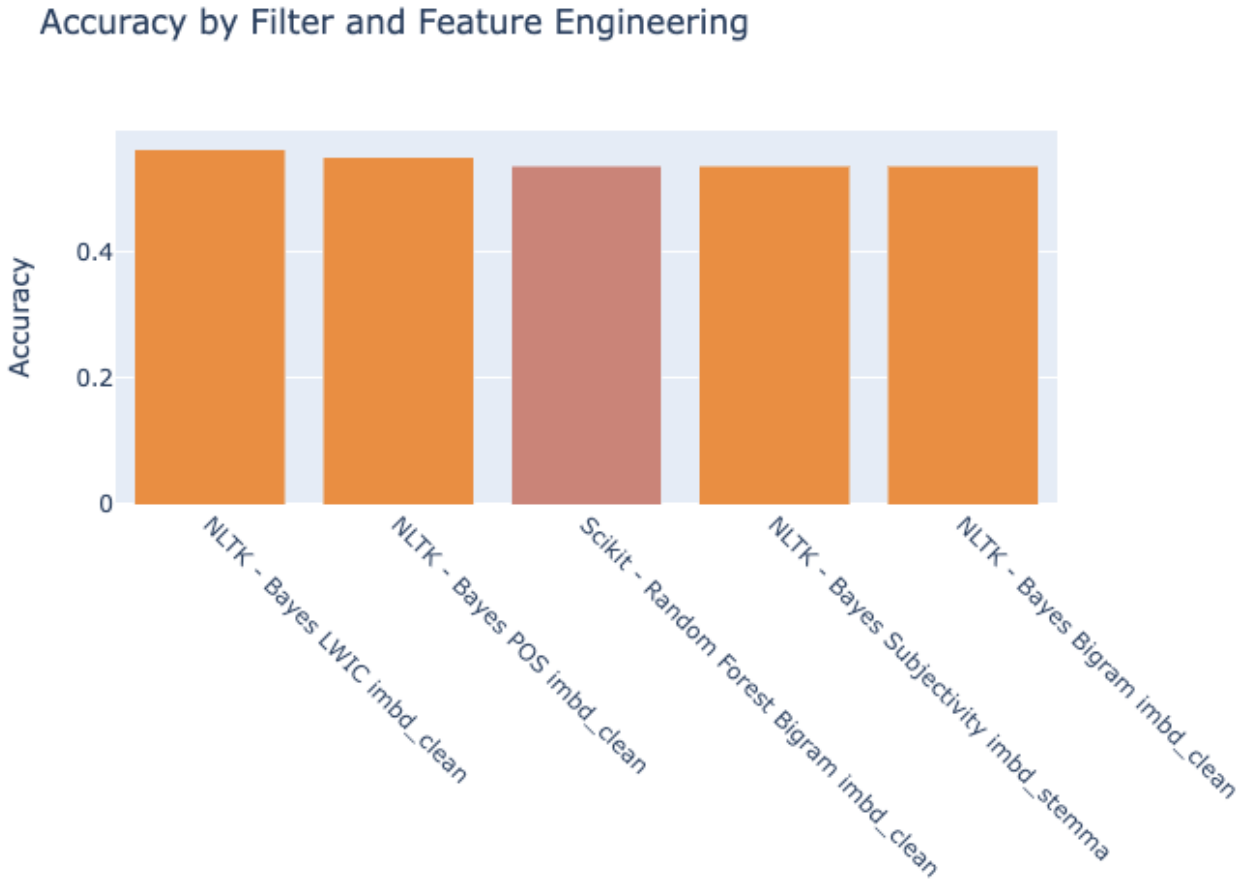
After building 3 filters, 5 feature engineering, and 2 models ran every permutation of possible experiments. This included 30 different experiments between Bayes NLTK and Random Forest Scikit learn. The results ranged from 56% to 44% with each new experiment incrementally improving from the other. The overall results are still poor with accuracy scores equating to nearly 50/50 shot in predicting the right rating based on comments.



Regardless, drilling down into the top 5 experiments which includes the following:

1. Bayes LWIC using the clean feature
2. Bayes Unigram Clean feature
3. Bayes Part of Speech clean feature
4. Scikit Random Forest Bigram Clean feature
5. Bayes Subjectivity clean data feature

The top 5 ranged from 56% to 52% from an accuracy score. With Scikit Random forest as the only non NLTK model.

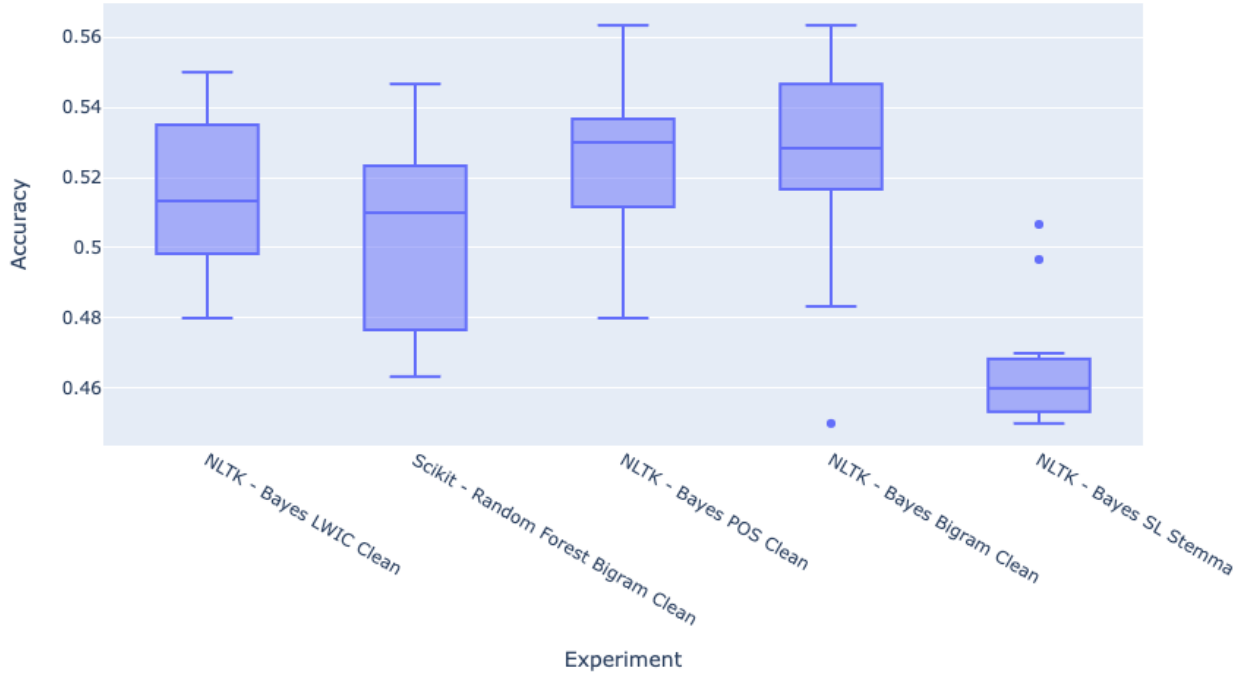


Model	Feature	Filter	Experiment	Accuracy
NLTK - Bayes	LWIC	imbd_clean	NLTK - Bayes LWIC imbd_clean	0.564
NLTK - Bayes	POS	imbd_clean	NLTK - Bayes POS imbd_clean	0.552
NLTK - Bayes	Bigram	imbd_clean	NLTK - Bayes Bigram imbd_clean	0.538
Scikit - Random Forest	Bigram	imbd_clean	Scikit - Random Forest Bigram imbd_clean	0.538
NLTK - Bayes	Subjectivity	imbd_stemma	NLTK - Bayes Subjectivity imbd_stemma	0.538

Taking the top 5 results and running them each through cross validation but with 3,600 sentences and 12 folds instead of the original 1000 and 3 folds shown above.

Experiment	Avg Accuracy	Stdv Accuracy
NLTK - Bayes Bigram Clean	0.523611	0.031509
NLTK - Bayes LWIC Clean	0.515000	0.023463
NLTK - Bayes POS Clean	0.524167	0.024500
NLTK - Bayes SL Stemma	0.465833	0.017929
Scikit - Random Forest Bigram Clean	0.504167	0.026973

The results from the cross validation shows that the average Accuracy is different then what we saw with the test and train data set. Overall the results are about 4% less than before. The top Accuracy is now the NLTK Bayes Part of Speech with 52%. When looking at the standard deviation, 2.4% is much better than the second best accuracy score, Bayes Bigram Clean. Meaning that POS with 52.4% and a standard deviation of 2.5% is the most optimal from the list above.



Looking at the 12 folds of the cross-validation results above Bayes POS Clean has a much higher average and the best median value than the rest. What is interesting is that the Bayes Subjectivity Lexicon using Stigmatizing and Lematizing has the tightest box plot of the bunch but the worst in Accuracy scores.

Final Thoughts

The path to predicting a rating based on comments was long and arduous. Through filtering, transforming the data, applying multiple feature engineering techniques, and then modeling the featuresets, a conclusion can be seen. The power to predict comes at a great price. Throughout the process we only achieved an accuracy score of around 55%. Though the journey was hard, the exercises learned were invaluable. From the numerous cleaning and transforming techniques to ingest into a model. To the excitement of seeing results. Then to use methods to help prove the results, by cross validating the predictions. Although only 3 models were explored, additional models can be applied, and more data could be used. At what cost depends on the machine and time necessary to continue striving for a higher accuracy score. In summary the techniques in this paper illustrate what is needed to succeed in modeling text data. In addition, it serves as a terrific blueprint to begin applying Natural Language Processing in the real world.