```
# US Household Income Exploratory Data Analysis

# Look at the whole US household income table.
SELECT *
FROM us_project.us_household_income
;

# Look at the whole US household income statistics table.
SELECT *
FROM us_project.us_household_income_statistics
;

# Alex points out that there are no dates in the data,
# so there's no way look at trends over time with this dataset.

# Alex looks through the data and points out categorical data
# in the US household income table along with the land and water values.
# In the US household income statistics table, Alex notices the means,
# medians, and standard deviations for the incomes which are the main points
# of this dataset.

# Alex starts out with exploring the land and water area data.
SELECT State_Name, ALand, AWater
FROM us_project.us_household_income
;

# Repeat the above query but include counties as well.
SELECT State_Name, County, ALand, AWater
FROM us_project.us_household_income
;

# Repeat the above query but include the cities as well.
SELECT State_Name, County, City, ALand, AWater
FROM us_project.us_household_income
;

# Alex points out that the data on land and water go down to the city level,
# so any analyses at higher levels, like the state level, will likely
# involve aggregations.

# Alex investigates the total area of land and water for each state
# sorted by the second column in descending order
# Reminder: MySQL indexes starting from the number 1.
SELECT State_Name, SUM(ALand), SUM(AWater)
FROM us_project.us_household_income
GROUP BY State_Name
ORDER BY 2 DESC
;
```

```
# Alex points out that the resulting table makes sense since the states at the top
# of the table are fairly large states. He then points out that it's worthwhile to
# look at values and think about whether or not those values make sense.

# I'd like to point out that the top states
# can have a lot of towns and cities that show up in the table which in turn contributes
# to the states' total land area. Moreover, I'm under the impression that Alaska is the largest state in the US
# in terms of square miles, but because that state is so sparsely populated, fewer towns or cities for Alaska
# will show up in the table than some of the other states
# which in turn results in a smaller total area of land that can count towards the total area of inhabited land.

# Alex then points out that states at the bottom of the resulting table make sense because they are small
# in terms of overall square miles.
SELECT State_Name, SUM(ALand), SUM(AWater)
FROM us_project.us_household_income
GROUP BY State_Name
ORDER BY 3 DESC
;

# Alex comments that states at the top of the resulting table make sense
# as well as the states at the bottom. He does admit, though, that Texas is
# a bit of a surprise, especially since he's lived in Dallas, Texax, for a while.

# Alex decides to look at the top ten states by area of land.
SELECT State_Name, SUM(ALand), SUM(AWater)
FROM us_project.us_household_income
GROUP BY State_Name
ORDER BY 2 DESC
LIMIT 10
;

# Once again, Alaska is the largest state in the US based off of total square miles
# (source = https://www.usatoday.com/story/news/2022/12/09/what-largest-state-us-size-states-land-area/8083288001/);
# however, because it's so sparsely populated, it doesn't have a lot of towns or cities
# to contribute to this table
SELECT State_Name, COUNT(City)
FROM us_household_income
WHERE State_Name IN ('Texas', 'California', 'Alaska')
GROUP BY State_Name
;

# Alex then decides to look at the top ten states by area of water.
SELECT State_Name, SUM(ALand), SUM(AWater)
```

```
FROM us_project.us_household_income
GROUP BY State_Name
ORDER BY 3 DESC
LIMIT 10
;

# Review the household income table and the household income statistics table.
SELECT *
FROM us_project.us_household_income
;

SELECT *
FROM us_project.us_household_income_statistics
;

# Both tables have ID columns, so they can be joined on those values.
SELECT *
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
;

# There was a bit of a pause when running the above query because
# MySQL has to work on so much data.

# Alex points out that more rows of the US household income statistics table
# were imported into MySQL than row of the US household income table, so he
# opts to perform a right join and focus on the rows where values under the id
# column of the US household income table are null (because those would be rows
# that were likley missed during the import step)

# I'm under the impression that right joins are rare in practice because the
# table on the right could be placed on the left and a left join can be performed instead.
# However, I can understand how Alex could opt for a right join because the US household
# income statistics table is on the right for the inner join above, and performing a right join
# can help with mentally organizing data.
SELECT *
FROM us_project.us_household_income AS u
RIGHT JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE u.id IS NULL
;

# Based off of the result of the above query, Alex notes that missing records occur for
# a bunch of different states. He points out that the data could be revisited in Excel and fixed,
# or the rows in the US household income statistics table without any corresponding rows
# in the US household income table could be deleted. However, Alex decides to just ignore the
# rows with null values after the right join, so he moves forward with an inner join.
```

```
# Moreover, he emphasizes the need to keep these issues in mind when exploring data - fix, delete, or
ignore.

SELECT *
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
;

# Alex says that he wants to use the mean, median, and standard deviation data. He accidentally clicked
on the
# column name for the mean values which organized the rows by mean values in ascneding order and in
the process
# realized that some means, medians, and standard deviations are set to zero. Alex points out that some
people or
# organizations don't like to report to other people or organizations, so it's very possible that the
presence of zeros
# is a result of that non-reporting.

# Run the above query again but focus on the rows that don't have a mean value of zero.
SELECT *
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
;

# Alex then decides to focus on some columns that he thinks are interesting to work with
# Reminder: I'm inclined to use the grave accent mark (`) around column names if MySQL also uses those
# names as keywords in order to make it clear that I want to work with column names.
# Because State_Name shows up in the two tables being joined together, the column in the US
household
# income table is specified.
SELECT u.State_Name, County, `Type`, `Primary`, Mean, Median
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
;

# Alex decides to look at data at the state level.
SELECT u.State_Name, Mean, Median
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
;
```

```
# Run the above query again but focus on the averages for the means and medians.
# Round the averages for better readability. Alex opts to round the numbers to one
# decimal place. I'll do the same for the sake of following along.
SELECT u.State_Name, ROUND(AVG(Mean), 1) , ROUND(AVG(Median), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
GROUP BY u.State_Name
;

# Repeat the above query but order the rows by the average
# of the mean incomes for each city listed.
# Reminder: MySQL appears to be one-indexed.
SELECT u.State_Name, ROUND(AVG(Mean), 1) , ROUND(AVG(Median), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
GROUP BY u.State_Name
ORDER BY 2 ASC
;

# Puerto Rico has the lowest of the average mean incomes
# followed by Mississippi and Arkansas. I tihnk it's worth noting that
# Puerto Rico isn't a state and neither is the District of Columbia, so it might be
# worthwhile to rename State_Name to something more accurate like State_or_Territory_or_District.
# The column would be wider, but at least the new name could better describe the contents of the
# column - unless the data dictionary explains these nuances, then I suppose the column can stay as
# State_Name.

# Alex investigates the bottom five states by mean average income.
SELECT u.State_Name, ROUND(AVG(Mean), 1) , ROUND(AVG(Median), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
GROUP BY u.State_Name
ORDER BY 2 ASC
LIMIT 5
;

# Alex next investigates the top five states by mean average income.
SELECT u.State_Name, ROUND(AVG(Mean), 1) , ROUND(AVG(Median), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
```

```
GROUP BY u.State_Name
ORDER BY 2 DESC
LIMIT 5
;

# Alex investigates the top ten states by mean average income
# to see where California ranks. He originally guessed California would be on top,
# but it turned out that that state wasn't in the top five.
SELECT u.State_Name, ROUND(AVG(Mean), 1) , ROUND(AVG(Median), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
GROUP BY u.State_Name
ORDER BY 2 DESC
LIMIT 10
;

# Alex points out that these states have a high cost of living, especially the states
# in the northeast of the US, such as Connecticut, New Jersey, and New York. Alex is
# surprised that Alaska has such high income values because it seems so remote;
# however, I did a quick search on Google, and it looks like the cost of living is high
# because there are few highways connecting Alaska to the rest of the US, so goods need to be
# shipped or flown to that state (source = https://www.worldpackers.com/articles/is-alaska-expensive),
# and it makes sense for incomes to align with costs.

# Repeat the above query but sort by average median income in descending order.
SELECT u.State_Name, ROUND(AVG(Mean), 1) , ROUND(AVG(Median), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
GROUP BY u.State_Name
ORDER BY 3 DESC
LIMIT 10
;

# Repeat the above query but sort by average median income in ascending order.
SELECT u.State_Name, ROUND(AVG(Mean), 1) , ROUND(AVG(Median), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
GROUP BY u.State_Name
ORDER BY 3 ASC
LIMIT 10
;
```

```
# Alex points out that when looking at the bottom few states by average median income,
# the differences between the average median values and the corresponding average mean values are
less than
# the differences between the average median values and the corresponding average mean values of the
top
# few states by average median income

# Alex reruns a previous query.
SELECT u.State_Name, County, `Type`, `Primary`, Mean, Median
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
;

# Alex decides to focus on type and track based off of the result of the above query.
# He also says that it's a good thing to reuse code.
SELECT `Type`, `Primary`, ROUND(AVG(Mean), 1) , ROUND(AVG(Median), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
GROUP BY `Type`, `Primary`
ORDER BY 3 DESC
LIMIT 10
;

# Based off of the result of the above query, Alex doesn't feel that
# the Primary column is very useful, so he runs the above query once again
# without that column. He also admits that he doesn't know exactly what that column means,
# and I in turn am inclined to be believe that any data dictionary that originally came with the dataset
# didn't give a proper explanation.
SELECT `Type`, ROUND(AVG(Mean), 1) , ROUND(AVG(Median), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
GROUP BY `Type`
ORDER BY 3 DESC
LIMIT 10
;

# Rerun the above query with the rows ordered by column 2 (the rounded average
# of the mean incomes).
SELECT `Type`, ROUND(AVG(Mean), 1) , ROUND(AVG(Median), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
```

```
WHERE Mean != 0
GROUP BY `Type`
ORDER BY 2 DESC
LIMIT 10
;

# Alex notes that the Municipality type has a noteable average mean income,
# so he decides to run the above query again but with a count of each type included.
SELECT `Type`, COUNT(`Type`), ROUND(AVG(Mean), 1) , ROUND(AVG(Median), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
GROUP BY `Type`
ORDER BY 2 DESC
LIMIT 10
;

# Interestingly, the Municipality type disappeared, so he repeats the query above minus
# the COUNT() function.
SELECT `Type`, ROUND(AVG(Mean), 1) , ROUND(AVG(Median), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
GROUP BY `Type`
ORDER BY 2 DESC
LIMIT 10
;

# He's not sure why the Municipality type disappears when querying for the count of the different types,
# so he tries repeating the query two above but sorted by the column 1.
SELECT `Type`, COUNT(`Type`), ROUND(AVG(Mean), 1) , ROUND(AVG(Median), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
GROUP BY `Type`
ORDER BY 1 DESC
LIMIT 10
;

# Alex loses the Municipality type in his output, but I actually see that type in my output.

# After replaying the video for a bit, I think his mistake was that he typed out GROUP BY 1 ORDER BY 2
DESC
# while I have GROUP BY `Type` ORDER BY 1 DESC. As for the case of the missing Municipality type, it
looks like
```

the Municipality type in one query result gets replaced with a Community type in another.

At this point, I'm not sure why values under the Type header are changing between queries. I might have to just wait
until Alex sorts out his query in the video.

Alex then does his own investigation. Moreover, he says that he wants to keep this confusing detail in the video
because he wants the viewers to see how he works through this issue.

After doing his own investigation, it turns out that the values under the Type header change between each query
because he still has the LIMIT keyword included. When setting the LIMIT number to 20, more values show up.

I'll rerun my query with the LIMIT set to 20 in order to match Alex's output in the video.
```
SELECT `Type`, COUNT(`Type`), ROUND(AVG(Mean), 1) , ROUND(AVG(Median), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
GROUP BY `Type`
ORDER BY 1 DESC
LIMIT 20
;
```

Alex would like to order by column 3.
```
SELECT `Type`, COUNT(`Type`), ROUND(AVG(Mean), 1) , ROUND(AVG(Median), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
GROUP BY `Type`
ORDER BY 3 DESC
LIMIT 20
;
```

Alex points out that a contributing factor to the Municipality type having such a high
average mean income is the fact that there's only one instance of Municiplaity,
so there are no other values to bring it down. However, I'm not sure if that average mean
is necessarily because of the fact that there's only one instance of Municipality.
Other factors could impact the average mean income of a municipality, such as state or nearby cities. Moreover, for
all I know, the value for that one instance of Municipality could be too low for what typically is found for
municipalities across the US, and there just aren't enough data points to raise the average mean up.

At any rate, I'll continue with Alex's progress in the video.

```
# Alex continues with his observations based off of the above query. For example, he suspects that CPD and City
# are the same thing becaue the average means are so similiar, but he's not inclined to change one to the other and merge
# the data. He's concerned that the average mean incomes for Urban and Community are so low because basic necessities
# don't appear to be viable with their values.

# Alex then decides to run the above query once again, but this time he
# opts to sort by average median income in descending order.
SELECT `Type`, COUNT(`Type`), ROUND(AVG(Mean), 1) , ROUND(AVG(Median), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
GROUP BY `Type`
ORDER BY 4 DESC
LIMIT 20
;

# Alex points out that the CDP type has a high average median income
# along with the Track type. He also points out that the count of the Track type is the highest
# among all of the distinct values in the Type column. Moreover, he notes that the
# County, Urban, and Community types all have low average median incomes.

# Out of curiosity, Alex decides to investigate which states have the Community type.
# Moreover, he points out that exploratory data analysis typically involves
# curiosity-motivated side-explorations.
SELECT *
FROM us_household_income
WHERE `Type` = 'Community'
;

# Based off of the result of the above query, the Community types show up in Puerto Rico,
# and this makes sense according to Alex because the average mean salary and the average median
# salary for Puerto Rico was found to be low when compared to the states in the US or the District of Columbia.
# Reminder: Puerto Rico is a territory and not a state.

# Alex then repeats the following query:
SELECT `Type`, COUNT(`Type`), ROUND(AVG(Mean), 1) , ROUND(AVG(Median), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
GROUP BY `Type`
ORDER BY 4 DESC
```

```
LIMIT 20
;

# Alex points out that sometimes it makes sense to exclude
# certain data points because they occur so infrequently, so he
# decides to run the above query again but this time retaining only
# types with counts greater than one hundred. Alex refers to these infrequently occurring
# types as outliers, and I suppose that makese sense when judging by frequency of type.
SELECT `Type`, COUNT(`Type`), ROUND(AVG(Mean), 1) , ROUND(AVG(Median), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
WHERE Mean != 0
GROUP BY `Type`
HAVING 100 < COUNT(`Type`)
ORDER BY 4 DESC
LIMIT 20
;

# Alex mentions that the decision to exclude rows can depend on one's familiarity with the dataset.
# He admits that he doesn't know the data like the back of his hand, but he does say that
# outliers do need to be taken into consideration. Alex then focuses on the types that occur more
frequently.

# Alex reviews the inner join from earlier.
# Again, both tables have ID columns, so they can be joined on those values
SELECT *
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
;

# Alex points out there are states with really big cities, and he would like to see
# the salaray data for those places.
SELECT u.State_Name, City, ROUND(AVG(Mean), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
GROUP BY u.State_Name, City
;

# Alex repeats the above query but this time he sorts the rows
# by average mean income in descending order.
SELECT u.State_Name, City, ROUND(AVG(Mean), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
	ON u.id = us.id
GROUP BY u.State_Name, City
```

```
ORDER BY ROUND(AVG(Mean), 1) DESC
;

# Some of the cities in the above query result have average mean incomes
# above 200,000. I'm familiar with Short Hills, NJ; that place
# is a high-income area. Other places on the list that I recognize as high-income places,
# even though the average mean incomes there are between 100,000 and 200,000, are Upper Saddle
River, NJ;
# North Caldwell, NJ; and Ridgewood, NJ.

# Alex notices that a lot of the cities in the same query result are located in
# the northeast of the US.

# Out of curiosity, Alex decides to investigate the average median income for the cities by
# repeating the above query and adding a column for average median incomes.
# Note: If each city has only one mean income and one median income, then taking averages
# of those values might not be necessary because the averages would just be the same as the single data
points.
# My guess is that Alex took averages for different groups earlier in the EDA process, so it likely
# makes sense to do that again here with the cities.
SELECT u.State_Name, City, ROUND(AVG(Mean), 1), ROUND(AVG(Median), 1)
FROM us_project.us_household_income AS u
INNER JOIN us_project.us_household_income_statistics AS us
        ON u.id = us.id
GROUP BY u.State_Name, City
ORDER BY ROUND(AVG(Mean), 1) DESC
;

# Alex notes that some average medians are set to 300000.0, and he suspects that
# there might be an intentional cap on median household income. This cap could be
# something that can be investigated in the future.

# Alex then closes out the EDA process here with a review of what had been investigated
# over the course of the video. He also mentions that in the real world, some direction is
# provided as to what should be investigated in a dataset, or some expertise in a dataset
# can offer guidance in terms what to investigate. However, in the case of this dataset,
# Alex dove right in, saw what he could find, and found some interesting aggregations.

# I'm inclined to agree with the idea that some direction can provided at a job, and that
# expertise on a dataset can guide analysis. However, barring either of these, I'm inclined
# to take a more systematic approach to the EDA process - go column by column, perform some
descriptive
# statistics, and take note of any multivariate analyses that I could perform later on. If I find
# something interesting while performing univariate anlysis for a particular column, I might dive deeper
# right then and there, but I'll try not to dive so deep that I end up neglecting the other columns, ya
know?
```