**Quiz 3 Structure (Lectures 12, 14, 15, & 17)**

1. Replace the values of a column **(12)**
   1. Obtain unique string values of a column -
   2. Use the “.replace()” command
2. Recode a numeric column **(end of 12)**
   1. Use the “pd.cut()” command to create a new column based on an interval.
3. Aggregate and query **(14)**
   1. Use a combination of the following commands to produce a new dataset
      1. .query()
         * Filters a subset of rows
      2. .groupby() .agg()
4. Aggregate and sort **(14)**
   1. Use a combination of the following commands to produce a new dataset
      1. .groupby() .agg()
      2. .sort\_values() - sorts the rows of the DataFrame based on one or more columns' values.
5. Rename column **(15)**
   1. Create a dictionary
   2. Rename one or more columns in a dataset using a dictionary
6. Merge dataset **(15)**
   1. Use “pd.merge” to combine two datasets: a primary and a secondary
   2. Only merge a subset of the columns of the secondary dataset
   3. Use “display” to show the merged dataset, extracting a subset of the columns

**Lecture 12 – Replacing and recoding variables**

**NaNs**

* np.nan
  + Means “Not a number”
  + Denotes missing values
* The “np.mean()” doesn’t work if the mean includes “NaNs”: does not work
  + **Print(np.mean(vec\_with\_nans)) – result: nan**
* The “np.nanmean()” computes the mean over the numeric observations (may not be accurate): ignores the nans
  + **Print(np.nanmean(vec\_with\_nans)) – result: 4.5**
* **PANDAS SUMMARY STATISTICS IGNORE NANS**

**Data Cleaning**

* Get data types - ex. circuits.dtypes
* Check rows with numeric values – ex. circuits[“alt”].str.isnumeric()
* Extract list of non-numeric values
  + ex. subset = circuits.query(“alt.str.isnumeric() == False”)
  + list\_unique = pd.unique(subset[“alt”])
  + print(list\_unique)
* Replace certain values – ex. circuits[“alt”] = circuits[“alt”].replace(list\_old, list\_new)
* Store a “cleaned” dataset

**Recording Numeric Variables**

* **Convert column to numeric** 
  + Circuits[“alt\_numeric”] = pd.to\_numeric(circuits[“alt”])
  + Print(circuits[“alt\_numeric”].mean()
* Before you convert, make sure to “clean” the variable as we did before by checking what the non-numeric values are
* **Recode values based on an interval**

**Lecture 14**

**Transform + groupby**

**Aggregate + groupby**

* **Summary statistics**

**Lecture 15**

Key terms:

* [] – extracts columns
* .query() – filters a subset of rows
* .recode() – replaces values
* .groupby() .agg() – aggregate statistics by subgroup
* .rename() – change name of columns

Lecture 17

Examples of chaining

* Subsetting before extracting columns
* Subsetting before aggregating
* Substting after aggregating
* **Subset before Aggregating:**
  + When you need to perform different aggregations on different subsets of your data.
  + When you have a large dataset and you only need to aggregate specific portions of it.
  + When you want to filter out irrelevant data before performing aggregations to improve computational efficiency.

For example, suppose you have a dataset of sales transactions and you want to calculate the total revenue for each product category. You might subset your data by product category first and then aggregate the sales amounts within each subset.

* **Subset after Aggregating:**
  + When you need to perform aggregations on the entire dataset first and then apply further filtering or analysis.
  + When the aggregation operation itself helps in determining which subsets are relevant.
  + When the subsets you need are determined by the aggregated results.

For instance, if you want to find the average age of customers who purchased a certain product, you might first aggregate the data to calculate the average age of all customers who made purchases, and then subset this result to focus on customers who purchased the specific product.

* Chaining inside queries + NaNs
  + Is.na() – true/false vector, checks for missing values
  + Is.notna() – true/false vector, checks for non-missing values
  + .str.isnumeric() – checks whether individual rows of a string column are numeric