**MNC HW 1 Command Shell**

**Question 1.**

pwd outputs /Users/marielle/Desktop/KaggleDatasets

* Mkdir zip\_files
* mv \*.zip /Users/marielle/Desktop/KaggleDatasets/zip\_files

**Question 2.**

* head -n 1 diabetes\_prediction\_dataset.csv
  + This shows the variables gender, age, hypertension, etc.
* head -n 1 diabetes\_prediction\_dataset.csv > diabetes\_prediction\_1.csv
* head -n 1 diabetes\_prediction\_dataset.csv > diabetes\_prediction\_2.csv
* head -n 1 diabetes\_prediction\_dataset.csv > diabetes\_prediction\_3.csv
* wc diabetes\_prediction\_dataset.csv outputs 100001 lines, 142264 words, 3810356 characters
  + The first line is the header so there are 10k entries, meaning each of the new .csv files should contain 33,333 entries
* tail -n +2 diabetes\_prediction\_dataset.csv | head -n 33333 >> diabetes\_prediction\_1.csv
  + Tail -n +2 skips the first line, head -n 33333 takes the next 33333 entries and the output of lines 2 to 333334 are directed to diabetes\_prediction\_1.csv
* tail -n +33335 diabetes\_prediction\_dataset.csv | head -n 33333 >> diabetes\_prediction\_2.csv
  + Tail -n +33335 skips the first 33334 rows, head -n 33333 grabs the next entries and the output of lines 33335 to 66667 are directed to diabetes\_prediction\_2.csv
* tail -n +66668 diabetes\_prediction\_dataset.csv >> diabetes\_prediction\_3.csv
  + This skips the first 66667 rows and takes the rest of the remaining data and outputs it into
* Nano each file to check if it worked

**Question 3.**

* head -n 1 Heart\_Disease\_Prediction.csv
  + Shows variables Age, Sex, Chest pain type, etc.
  + The last column “Heart Disease” shows us the presence and absence variables that we are looking for for this exercise
* head -n 1 Heart\_Disease\_Prediction.csv > Heart\_Disease\_Presence.csv
* head -n 1 Heart\_Disease\_Prediction.csv > Heart\_Disease\_Absence.csv
* grep "Presence" Heart\_Disease\_Prediction.csv >> Heart\_Disease\_Presence.csv
  + We do not need to specify which column to grep from because every other column is a numerical value and the “Heart Disease” column is the only text variable column
* grep "Absence" Heart\_Disease\_Prediction.csv >> Heart\_Disease\_Absence.csv
* Nano each file to check if it worked

**Question 4.**

* head -2 car\_web\_scraped\_dataset.csv
  + Take a look at what the data looks like and so we know which variables we need to use to find the fraction of cars that have no accidents
  + Note that in the condition column, some variables look like “No accidents reported, 1 Owner” and “No accidents reported, 2 Owners”
* grep "No accidents reported" car\_web\_scraped\_dataset.csv | wc -l
  + 2223
* tail -n +2 car\_web\_scraped\_dataset.csv | wc -l
  + 2840
* echo "scale=2; $(grep "No accidents reported" car\_web\_scraped\_dataset.csv | wc -l) / $(tail -n +2 car\_web\_scraped\_dataset.csv | wc -l)" | bc
  + Calculates the fraction of cars who have had no accidents by finding how many cars reported no accidents and dividing that by the number of entries in the dataset, excluding the first line
  + Scale is the number of decimal places and bc means basic calculator
  + 2223/2840

The answer is: 0.78

**Question 5.**

* head -2 Housing.csv
* sed -e 's/yes/1/g' -e 's/no/0/g' -e 's/unfurnished/0/g' -e 's/semi-furnished/2/g' -e 's/furnished/1/g' Housing.csv > Housing\_modified.csv
  + This looks complicated when put all together but it’s pretty easy to understand
  + -e tells sed to execute an expression
  + The expression goes “s for substitution / the word to be replaced / what to replace it with / global meaning for all instances” and then it pipes into the next expression to be executed
  + Replaced “furnished” last because sed is case-sensitive and will find the word “furnished” no matter where it is placed within the word
* nano Housing\_modified.csv to check if it worked

**Question 6.**

* head -2 Mall\_Customers.csv
* cut -d ',' -f 2- Mall\_Customers.csv > Mall\_Customers\_noCUSTOMERID.csv
  + Cuts by the comma delimitator and gives fields starting from column 2 and so on (because the CustomerID column was column 1 in the original .csv)
* nano Mall\_Customers\_noCUSTOMERID.csv to check if it worked

**Question 7.**

For this exercise, I had to do some data cleaning beforehand. This was because if I ran the pipeline on the original data, it output error “Parse error: bad character '�' <stdin>:16”. I went to that line and found that the dataset uses an unrecognized character so in order to do so, we had to change that character so the computer would not read non-numeric values.

* head -2 world\ all\ university\ rank\ and\ rank\ score.csv
* sed 's/–/-/g' "world all university rank and rank score.csv" > cleaned\_world\_university.csv
* tail -n +2 cleaned\_world\_university.csv | cut -d ',' -f 5-8 | tr ',' '+' | bc > university\_rank\_sums.csv
  + Skips the first row with the headers of the file, only keeps fields 5-8 while cutting out the rest by the comma delimitator, translates the commas to +’s, and then puts that into the basic calculator where the output goes to the university\_rank\_sums.csv
* nano university\_rank\_sums.csv to see if it worked

**Question 8.**

* head -2 cancer\ patient\ data\ sets.csv
* head -n 1 "cancer patient data sets.csv" > cancer\_data\_sorted.csv
* tail -n +2 "cancer patient data sets.csv" | sort -t ',' -k3,3n >> cancer\_data\_sorted.csv
  + Sorts by delimiter comma, and sorts by field/column 3 (which is Age), 3,3 sorts by Age only, and then n stands for numerical
* nano cancer\_data\_sorted.csv