Pandas for Data Wrangling Quiz Solutions

1. The “Unnamed: 0” column is most likely an index column that was written to the dataframe when the to_csv function was called without setting the index parameter to False. Pandas will automatically add an index column by default when writing to a csv.

2. We can use the interquartile range since it’s a measure of statistical dispersion. We calculate the difference Q3-Q1, which is the range between the 25^{th} and 75^{th} percentiles. Anything outside of this range can be considered an outlier. A similar answer could be to use a box plot to identify these points and remove them accordingly.
We can also use the skew() function which will determine the extent to which the data follows a normal distribution. A value of between [-1,+1] is preferred but if a feature is identified as heavily skewed, it’s possible to transform it via log transformation or something similar to maintain a normal distribution.

3. pd.Dataframe(np.array(counts).reshape(10,10))
pd.DataFrame(counts.values.reshape(10,10))

4. function: time.strptime(date, format)
Format: “%m/%d/%Y

5. When it comes to Boolean indexing, the nuance is that loc accepts Boolean arrays and series while iloc only accepts Boolean arrays. For example, suppose we were using the listings dataframe from the assignment:

df.loc[df['price'] < 90,:]

df.iloc[df['price'] < 90, :] #Error

The second line would yield an error. To remedy this, we must first convert into a numpy array.

df.iloc[(df['price'] < 90).values, :] #Correct

6. Applying to a groupby object will apply a function to the whole dataframe along an axis. When aggregating, we can use one or multiple operations per column over an axis. Transform will return the same length dataframe, with an operation (function, a string function, a list of functions, or a dict) applied to each of the values. Transform will not produce aggregated results.

7. When looking at time sensitive data, it would make the most sense to convert the data into key/value pairs since this will translate the best into a machine learning context. Having these many features would affect the performance of a model and lengthen the work required to modify the data. The most logical transformation would be to use melt() in order to property keys and date/price values.

8. pd.get_dummies(); It doesn't make sense to one hot encode a feature that may have a very large number of unique values since this wouldn't provide use for the model to derive relationships in the data and adds unnecessary length (100 unique values = 100 new columns).

9. Date times are specific date and times, with time zone support. Time deltas are an absolute time duration. Time spans cover a length in time defined by a point in time and its frequency. Date offsets are relative time durations that respect calendar arithmetic (see documentation).

10. The most obvious way would be to use vectorization of Pandas series since this improves computational speed. Replacing series with NumPy arrays would improve performance when Pandas isn't necessary since NumPy operations on arrays are much faster. Lastly, one could also loop with apply() which would cut down on the repetitive computations.