DS1EDP: Homework 05 – Solutions

1. Working with Text using functions

Question 1:

```
def word_count(word):
   return len(word.split())
```

Question 2:

```
chapter_lengths = table_chapters.apply(word_count, "Chapter text")
```

Question 3:

```
def character_count(word):
   word = word.replace(' ','')
   word = word.replace('.','')
   word = word.replace('!','')
   word = word.replace('?','')
   return len(word)
```

Note: You can also call `replace` chained. Example:
`word.replace(' ','').replace('.','').replace('!','').replace('?','')`

Question 4:

```
def chapter_number(chapter_text):
   return text_before(chapter_text, '.')
```

Note: Look at the contents of the chapters. The roman letter that indicates the chapter is always followed by a '.'. So, the '.' seems to be a good pattern to split the text by using `text_before`!

2. Uber

Question 1:

bins = np.arange(0, 120, 5)

boston.hist("ride time", bins=bins)

manila.hist("ride time", bins=bins)

city_with_long_ride_times = "Manila"

Question 2:

boston_under_10 = boston.where("ride time", are.below(10)).num_rows / boston.num_rows

manila_under_10 = manila.where("ride time", are.below(10)).num_rows / manila.num_rows

Question 3:

def average_ride_time_for_time(tbl, hod):

```
    return np.average(tbl.where("hod", are.equal_to(hod)).column("ride time"))

average_ride_time_for_time(boston, 12)
```

Question 4:

```
print(average_ride_time_for_time(manila, 10) - average_ride_time_for_time(manila, 22))

larger_diff = "Manila"
```

3. Faculty salaries

Question 1:

```
prof_names = profs.group("department", identity).select("department", "name identity")
```

Question 2:

```
def salary_range(salaries):
    return max(salaries)-min(salaries)

department_ranges = profs.group("department", salary_range)

biggest_range_dept = department_ranges.sort("gross_salary salary_range", descending=True).column("department").item(0)
```

4. Causes of Death by Year

Question 1:

```
cleaned_causes = causes.where("Cause of Death", are.not_equal_to("HOM")).where("Cause of Death", are.not_equal_to("HYP")).where("Cause of Death", are.not_equal_to("NEP")).join("Cause of Death", abbreviations, "Cause of Death").drop("Cause of Death").relabeled("Cause of Death (Full Description)", "Cause of Death")
```

Question 2:

```
cleaned_causes_by_year = cleaned_causes.pivot("Cause of Death", "Year", values='Count', collect=sum)
```

Question 3:

```
cleaned_causes_by_year.plot("Year")

most_happened = "Diseases of the Heart"
```

Question 4:

```
cleaned_causes_by_year.bar("Year", "Chronic Liver Disease and Cirrhosis")
cleaned_causes_by_year.bar("Year", "Cerebrovascular Disease (Stroke)")
# Don't change the code below this comment.
plots.title("% of total deaths / disease per year")
plots.xlabel("% of total deaths")
suspicion = True
```