Knowledge Discovery - Sandy Schoettler

Motivating Question:

Let's consider item popularity over time, by looking at the number of monthly checkouts. If an item is very popular for a short amount of time, it may have many monthly checkouts for a period of time, but then those checkouts may decrease significantly over time. On the other hand, some items may have a more sustained level of popularity, that is more consistent over time. These items might be books that students must check out for school, or commonplace items which could have significant demand for a prolonged period of time (like the *Harry Potter* series).

My goal was to explore the question, "*Which items have a consistent, sustained level of popularity?*" In measurable terms, I decided to look for items with a low variance in the number of monthly checkouts. Noticing the possibility for untouched items to score highly here, I decided to also meaure the average number of monthly checkouts, restricting results to those with at least somewhat significant popularity.

SQL Query

```
SELECT
    itemNumber, AVG(`count`), VARIANCE(`count`)
FROM
    (SELECT
        itemNumber, MONTH(checkOut) AS month, COUNT(*) AS `count`
    FROM
        spl_2016.transactions
    WHERE
        '2016-01-01' <= checkOut
            AND checkOut <= '2016-12-30'
    GROUP BY itemNumber , month) AS checkout_counts
GROUP BY itemNumber
```

Difficulties

The SQL query I originally wanted to make, over the entire database, would've taken too long to compute. I tried downloading the `spl_2016.transactions` table, with the intention of launching a Docker container on my own machine to host a PostgreSQL server which could compute the query locally. I wasn't able to get this to work with the time I had, but it may be useful for my next project idea so I am hopeful and expectant that I can get it up and running soon.

For this project I was able to retrieve the data I was interested in for 50,000 items checked out in 2016.

Post-Processing

I parsed the CSV and sorted the rows by variance using a python script:

```python
import csv
from operator import itemgetter

with open("transaction-quantities-50k.csv", newline='\n') as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    A = []
    _ = next(reader) # throw away first row
    for row in reader:
        [item, avg, varc] = row
        if float(avg) > 0:
            A.append([int(item), float(avg), float(varc)])
```

Here is a sample of the data

```python
A.sort(key = itemgetter(2))
i = 0
for row in A:
    i += 1
    print(i, "\t", row)
```

```
index    [itemNumber, average, variance]

49909    [1312958, 2.5, 2.25]
49910    [1328389, 2.5, 2.25]
49911    [1338209, 2.5, 2.25]
49912    [1266100, 2.9, 2.29]
49913    [1150702, 3.2222, 2.3950617283950617]
49914    [585984, 2.0, 2.4]
49915    [1344950, 2.0, 2.4000000000000004]
49916    [910847, 3.5, 2.4166666666666665]
49917    [184256, 2.6667, 2.4444444444444446]
49918    [551617, 2.7143, 2.4897959183673466]
49919    [836315, 2.7143, 2.489795918367347]
49920    [1248586, 2.4286, 2.5306122448979593]
49921    [125452, 2.3333, 2.5555555555555554]
49922    [258380, 2.8, 2.56]
49923    [873039, 2.8, 2.56]
49924    [1299287, 3.2, 2.56]
49925    [1255224, 3.5, 2.583333333333333]
49926    [13380, 2.5, 2.5833333333333335]
49927    [437787, 2.5, 2.5833333333333335]
49928    [10485, 3.6, 2.6399999999999997]
```