1. Spam Detection

In today's world there are many cases where the data is dirty or corrupted because of spam data or because someone deliberately corrupted our data with malicious records. Luckily our data is not like... wait... oh no! we just received information that "Evil Media Association", the largest media association, has inserted malicious and false information to our files in fear that we will surpass it.

"Distributed Media Association" is in dire need of your help to find the malicious information. Don't worry - our spy team has managed to find the algorithm that decides which record is malicious and which is not. However, their training is far inferior to the data science & engineering students at the Technion, which is why we need your immediate assistance.

The malicious records are from the "Daily Program Data". Use spark to compute the 6 following conditions for each record. If 4 or more of the conditions apply to a record then we say it is a *malicious record*:

- The prog_code was viewed by a device with less than 5 average daily events.
- The prog_code was watched by a device from a DMA name that contains the letter ['z'] (case insensitive).
- The program was watched by a family with less than 3 adults and their net worth is higher than 8 (both exclusive).
- The program code was aired (at least once) between Friday at 6PM and Saturday at 7PM (both inclusive) and there was at-least one household who watched the program with size higher than or equal to 8 (inclusive).
- The prog_code was watched (at least once) by a device from an household with more than 3 devices (exclusive) and the income of that household is lower than the average household income in the data.
- The program contains at least one of the genres ['Talk', 'Politics', 'News', 'Community', 'Crime'] and has a duration of more than 35 minutes (exclusive).

1.1 Extract and Transform (

1. Load data, preprocess it, apply transformations and combine it into one dataframe. Use Spark.
You should filter columns that are not used and you can also filter records by value. Add new columns that will save time on calculations. Try to add at least 3 new columns.

Consider principals discussed in class: data formats, hierarchical data, (un)structured data, etc.

1.2 Computation and Detection (

1. Notice that the data is divided among many schemata. One way to approach it is to first join all of them into a single schema and work with it exclusively. There are other ways to tackle this problem. Suggest another solution and write at least 2 pros and 2 cons for the solution we've suggested. Write them in the PDF of this part.

2. Find the malicious records in the data. You may use your results from 1.1 or a different method.

Tip: Remember Caching in Spark

Find the records that at least 4 out of the 6 conditions presented above apply to them.

- Your code should save the records to a CSV file.
- Print/Show the top 150 malicious program entries, ordered by ascending lexicographic order of the prog_code.

2. Design and Deployment

Now the association needs your help with something else. They need your help with organizing their data.

In order to give quick access to all their clients, the association has put servers in every DMA, however due to financial hardship, they can't replicate all the data in every DMA. Therefore, they need to choose where to put each fragment of the data. This is where you come in, as the proud students of the course.

2.1 DMA Popularity (

For each DMA return a list of all the genres ranked by their popularity in that DMA.

In other words - for each DMA and genre, calculate the amount of viewing entries (by program code, using ProgramViewingData relation) for any program that included that genre, watched from that DMA.

Order the genres (for each DMA separately) in descending order according to the amount of entries that watched that genre in that DMA.

- Save the output in a CSV file.
- Print/Show the top 5 genres (by descending order of views in that DMA) for each DMA in the following list:

['Waco-Temple-Bryan', 'New York', 'Washington, DC (Hagrstwn)']

2.2 DMA Popularity according to Wealth (

Unfortunately, like most big companies the Distributed Media Association was also corrupted by money. Instead of treating every DMA equally they consider some DMAs more than others according to their wealth score, determined by their net worth and income.

Where applicable onvert letters to appropriate numbers - A-D = 10-13. Empty/Null values are considered as 0.

Calculate for each DMA their wealth score, which is the normalized average of the net worth and the income:

$$\frac{\#avg\_net\_worth\_in\_dma}{\#max\_net\_worth\_all\_data} + \frac{\#avg\_income\_in\_dma}{\#max\_income\_all\_data}$$

- For each DMA return its 8 most popular genres - with a limitation - a genre can't be duplicated.

The wealthiest DMA chooses the top 8 genres according to its popularity rating. After that, the 2^{nd} most wealthy DMA chooses its top 8 genres, from a pool that does not contain the previously taken genres. We continue like this until there are no genres left - and place empty lists at the remaining DMAs.

- Save your entire ouput to a CSV file.
- Print/Show the top 25 DMAs, their wealth score and ordered list of genres in this order:

"DMA NAME", "WELATH SCORE", "ORDERED LIST OF GENRES"

(Exact formatting, apart from being clearly and easily understandable isn't important)

3. Deploy via Spark (

Implement your design from previous section via spark in Python (PySpark). Your output should be divided into as much directories as the number of DMAs which aren't empty. Directories - one for each DMA - one for each site. Each directory will contain the fragments (partitions by genre) of each site according to your design. Each partition is a table for a given genre containing all the programs' details of the programs that have that genre (at least).