Data Analysis of Netflix Movies and TV Shows using PySpark

Introduction

Big data refers to the constantly increasing data that an organization can use for analysis purpose. For a data-driven company like Netflix data is of key importance. With its humble beginning as a DVD rental platform, one of the principal factors that play a significant role in the success of this Over-The-Top media provider giant is its underlying use of Big Data. Netflix's ability to collect and use the data is the reason behind their success. Over the past couple of years, Netflix has become the de-facto destination for viewers looking to binge on movies and TV shows. Last year Netflix announced that it signed on 135 million Paid customers worldwide. In 2019 alone, Netflix made a tad more than $20 billion, by offering its viewers some of the top-tier content at their convenience.[1]

Big data analytics and visualization help Netflix to make better business decisions. Based on visual representation of key insights of data analysis important questions can be addressed which will further help to find efficient solution. Netflix generates stream of big data which needs to be processed in real-time for faster decision-making process. [2] Spark is an analytics engine that is used by data scientists all over the world for big data processing. It is built on top of Hadoop and can process batch as well as streaming data. Hadoop is a framework for distributed computing that splits the data across multiple nodes in a cluster and then uses of-the-self computing resources for computing the data in parallel. As this is open-source software and it works lightning fast, it is broadly used for big data processing.

This project focuses on understanding and implementing the data pipeline of Netflix dataset. This pipeline consists of steps ranging from importing, cleaning, transforming data to visualizing it. Since the size of data is considerably large PySpark is used for distributed computing on Jetstream VM platform. PySpark is a tool that was developed to support Python and enable working with Resilient Distributed Datasets (RDDs) in Python. At the core, an RDD is an immutable distributed collection of elements of the data that can be partitioned across nodes in the cluster to perform parallel operations (transformations and actions).
Inferring conclusion from data and its visualization is the ultimate motive of this project.

Background

For any company or organization, data collection is essential. Imagine Netflix with its 203 million subscribers. Studying the traits of the data of these many customers would be a tremendous task. Netflix uses the collected information, converts them into insights, results, or visualizations, and recommends TV shows and movies as per customers' preferences and interests. It almost feels like a supernatural talent or power.[3]

Netflix's recommendation system is extremely efficient compared to others like Hulu, Disney+, etc. Exploratory Data Analysis is the foundation of building recommendation system. Knowing what data variables are of importance and prioritizing them is what makes any business's recommendation engine different. To know the generic pattern and trends of a data may seem trivial but it is what makes the solid ground for building prediction algorithm.

Netflix is world's top streaming platform for movies and other shows. To understand what makes Netflix different from other streaming services and how exactly Netflix preprocess and analyze stream of data from multiple sources in real-time is intriguing. To understand what big data technology it uses to find key insights is interesting as a data science student.

Considering the popularity of Netflix, the data that it generates, and collects is going to increase enormously. Parallel/batch processing of such data is the effective way for getting results quickly. This is where big data comes into action and plays key role in creating a dynamic pipeline containing streamline process of data transformation based on business needs.

Apache Spark is a framework for fast real-time processing. It does in-memory computations to analyze data in real-time. It can also do batch processing (similar to Apache Hadoop MapReduce). Since majority of data scientists use Python because of its rich library set, integrating Python with Spark is a great advantage. Embedding spark environment inside python makes many things easier. To get hands on practice of it I decided to use PySpark for creating data pipeline. [4]

Methodology

Steps for implementing the problem are divided into following categories:

1. Setting up Environment
2. Loading Data and required Packages
3. Data Cleaning
4. Data Transformation and removing Outliers
5. Storing Output
6. Data Visualization

Setting Up Environment

- Logged in to Jetstream with valid and created a new project named as “Final - Project” (https://use.jetstreamcloud.org/application/projects/10305/instances/47912)
- Navigated to newly created project and launched the instance of image “Pyspark-Ubuntu18.04-instance” within the course allocation and created a quad VM.
- After instance became “Active” launched the VM in a Web Desktop mode.

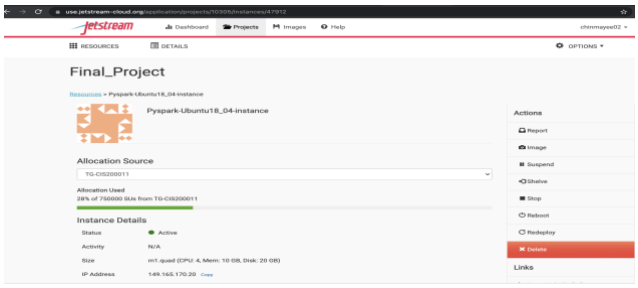


Fig 1: Setting up Environment

- Navigate to /opt/spark directory and make sure the Spark installation is there.
- Copy .bashrc from /opt/spark to /home/chinmayee02/ directory using the *cp* command “cp /opt/spark/.bashrc /home/chinmayee02/”
- After verifying that the file has been copied and "Load" the .bashrc file by typing “source .bashrc”.
- Verified that Java, Scala and Python3 are installed.

Fig 2: Setting up Environment

- Confirmed that Spark will be using Python 3 by entering “echo $PYSPARK_PYTHON”.
- Exported two additional environment variables for Spark: local IP (to bind VM to this IP address) and master (to define the master host, which, in this case, is the same machine)
- Verified this has taken place by entering echo command.
- Created a folder in home directory “program_file” using the *mkdir* command.
- Installed Jupyter notebook and py4J using pip3.
- Launched Jupyter notebook application.

Fig 3: Setting up Environment

Loading Data and required Packages

- Import all necessary packages required for execution of problem.
- Initialize Spark Session.
- Load data file (csv format) in spark environment.
- Check for data type of loaded file and perform preliminary operations to understand structure of data.

Data Cleaning

- For this project I've focused upon 'show id (Primary Key)', 'country', 'type', 'rating', 'release year' data columns as they are important to know the generic trend based on different demographic and categorical behaviors.
- Create new data frame with selected columns.
- Count null values for each column using "isNull().count()" command.

```
In [7]: df_filter = df.select('show_id','country','type','rating','release_year')
        df_filter.show()
        df_filter.count()
```

```
+-------+--------------------+-------+------+------------+
|show_id|             country|   type|rating|release_year|
+-------+--------------------+-------+------+------------+
|     s1|       United States|  Movie| PG-13|        2020|
|     s2|        South Africa|TV Show| TV-MA|        2021|
|     s3|                null|TV Show| TV-MA|        2021|
|     s4|                null|TV Show| TV-MA|        2021|
|     s5|               India|TV Show| TV-MA|        2021|
|     s6|                null|TV Show| TV-MA|        2021|
|     s7|                null|  Movie|    PG|        2021|
|     s8|United States, Gh...|  Movie| TV-MA|        1993|
|     s9|      United Kingdom|TV Show| TV-14|        2021|
|    s10|       United States|  Movie| PG-13|        2021|
|    s11|                null|TV Show| TV-MA|        2021|
|    s12|                null|TV Show| TV-MA|        2021|
|    s13|Germany, Czech Re...|  Movie| TV-MA|        2021|
|    s14|                null|  Movie| TV-PG|        2021|
|    s15|                null|TV Show| TV-MA|        2021|
|    s16|       United States|TV Show| TV-MA|        2021|
|    s17|                null|  Movie| TV-MA|        2020|
|    s18|              Mexico|TV Show| TV-MA|        2020|
|    s19|                null|  Movie| TV-14|        2021|
|    s20|                null|TV Show| TV-MA|        2021|
+-------+--------------------+-------+------+------------+
only showing top 20 rows
```

```
Out[7]: 8809
```

```
In [8]: from pyspark.sql.functions import col, when
        nulls1 = df_filter.filter(col('country').isNull()).count()
        nulls2 = df_filter.filter(col('type').isNull()).count()
        nulls3 = df_filter.filter(col('rating').isNull()).count()
        nulls4 = df_filter.filter(col('release_year').isNull()).count()
        print(nulls1,nulls2,nulls3,nulls4)
```

```
832 1 6 2
```

Fig 4: Data Cleaning

- Based on the count column "country" has large number of NA values. Keeping them might affect the result of analysis. Remove them using na.drop() is better idea.
- Convert PySpark data frame to RDD data frame and Execute general RDD Actions.

Data Transformation and removing Outliers

- Perform map – reduce on column to find count of each unique data values.

i) Content type: Movie, TV Show

To answer question, what different types of show or movie uploaded on Netflix we need to count total number of movies and TV Shows that are present in Netflix over the period.

```
In [13]: type_map = df_rdd.map(lambda x: (x[2],1)).reduceByKey(lambda a,b : a+b)
         type_map.collect()
```

```
Out[13]: [('Movie', 5691), ('TV Show', 2285), ('William Wyler', 1)]
```

Remove Outlier

```
In [14]: type_map = type_map.filter(lambda x: x[1] > 1)
         #print(rating_map)
         type_map.collect()
```

```
Out[14]: [('Movie', 5691), ('TV Show', 2285)]
```

Fig 5: Data Transformation and Removing Outliers

ii) Rating:

To answer question, what are the different types of rating that were given on the Netflix we need to find count of total number of ratings for content that is present in Netflix over the period. (Similar to Fig 5)

iii) Release Year:

To compare, how many movies and shows were released on Netflix every year in order to understand growth of the business. (Similar to Fig 5)

iv) Country:

To answer question, in which country movies and shows were released the most.

Country column contains multiple commas separated values in single row. To count occurrence of each country it is required to split these values into different rows. For this purpose, I created a new data frame with column country only. Transformed this new column in such a way that it becomes easy for performing map-reduce. (Similar to Fig 5)

```
In [19]: from pyspark.sql.functions import split, explode

split_country_df = (df_filter
                    .select(split(df_filter.country, ', ').alias('split')))
Split_country_single_df = (split_country_df
                    .select(explode(split_country_df.split).alias('word')))
country_df = Split_country_single_df.where(Split_country_single_df.word != '')
country_df.show()
country_df_count = country_df.count()
print (country_df_count)
```

```
+--------------+
|          word|
+--------------+
| United States|
|  South Africa|
|         India|
| United States|
|         Ghana|
|  Burkina Faso|
|United Kingdom|
|       Germany|
|      Ethiopia|
|United Kingdom|
| United States|
|       Germany|
|Czech Republic|
| United States|
|        Mexico|
|        Turkey|
|         India|
|     Australia|
| United States|
| United States|
+--------------+
only showing top 20 rows

10010
```

Fig 6: Data manipulation for column “Country”

v) Release Year and Content Type:

To compare what type of content i.e., movie or TV Show were made available on Netflix the most each year. (Similar to Fig 5)

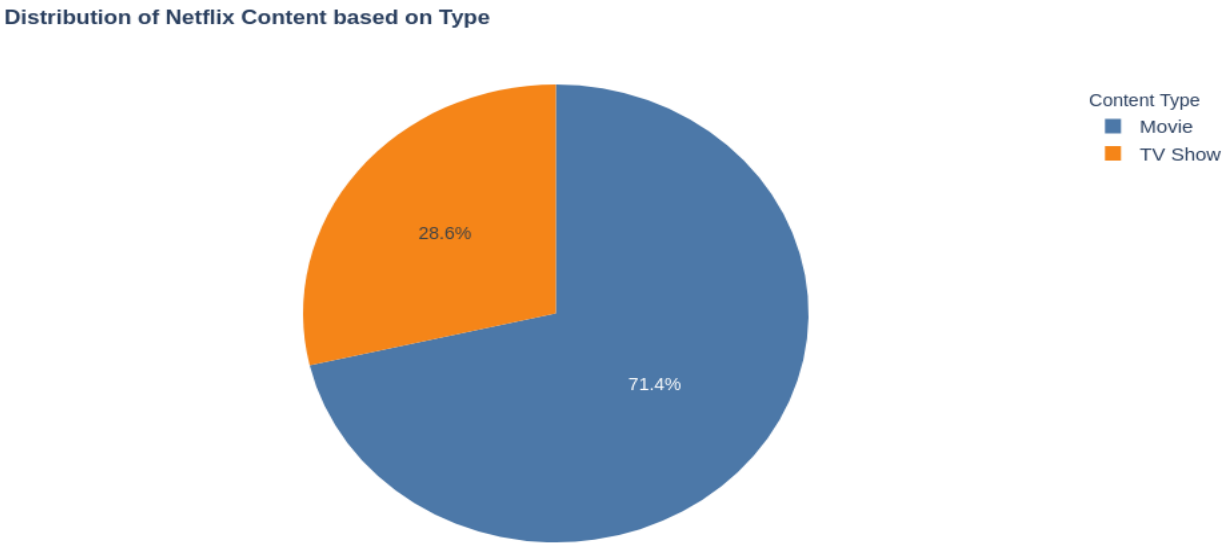
Storing Output

- The result generated from the above step is stored as a csv file on VM in row-column format.

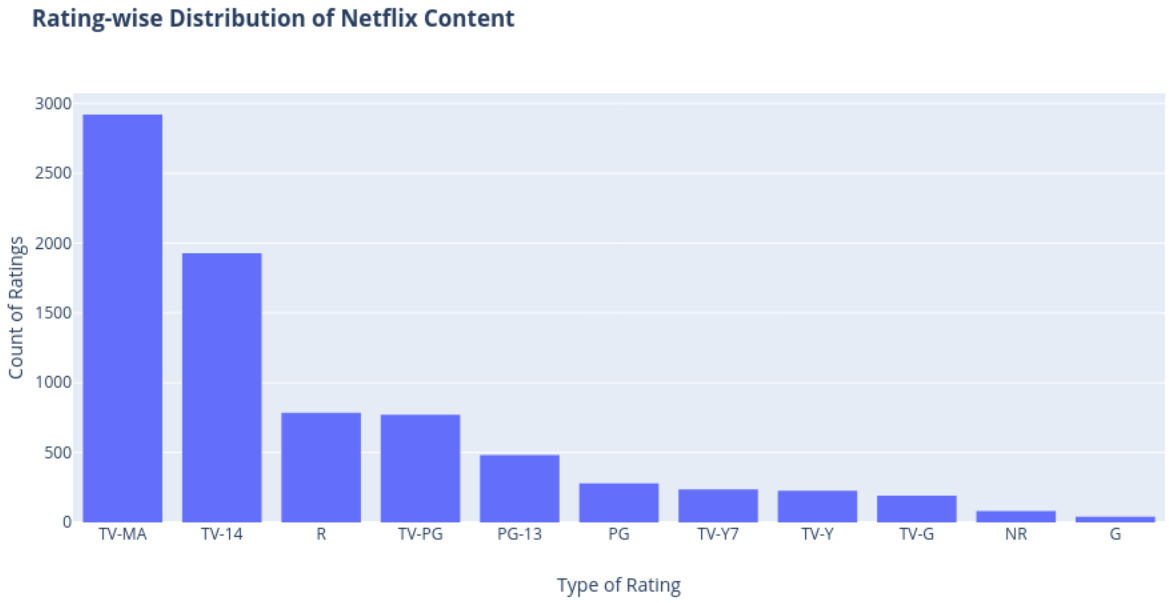
Results

Data Visualization

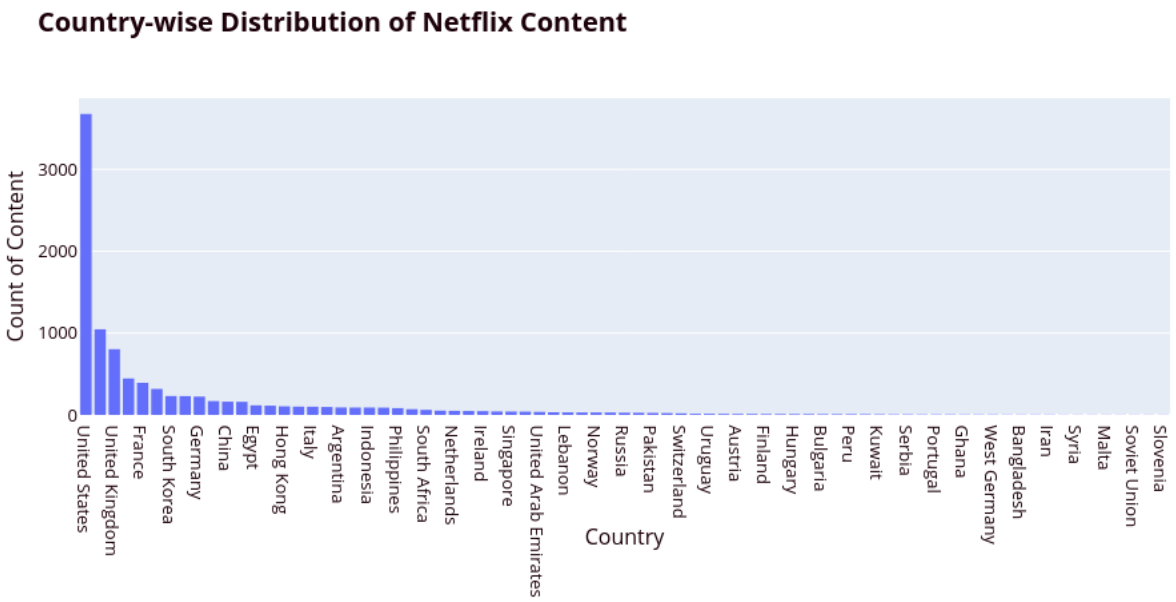
- The csv files of output are imported into plotly environment to create various visualization plots to present the key findings in more appealing manner. The plots are shown below.



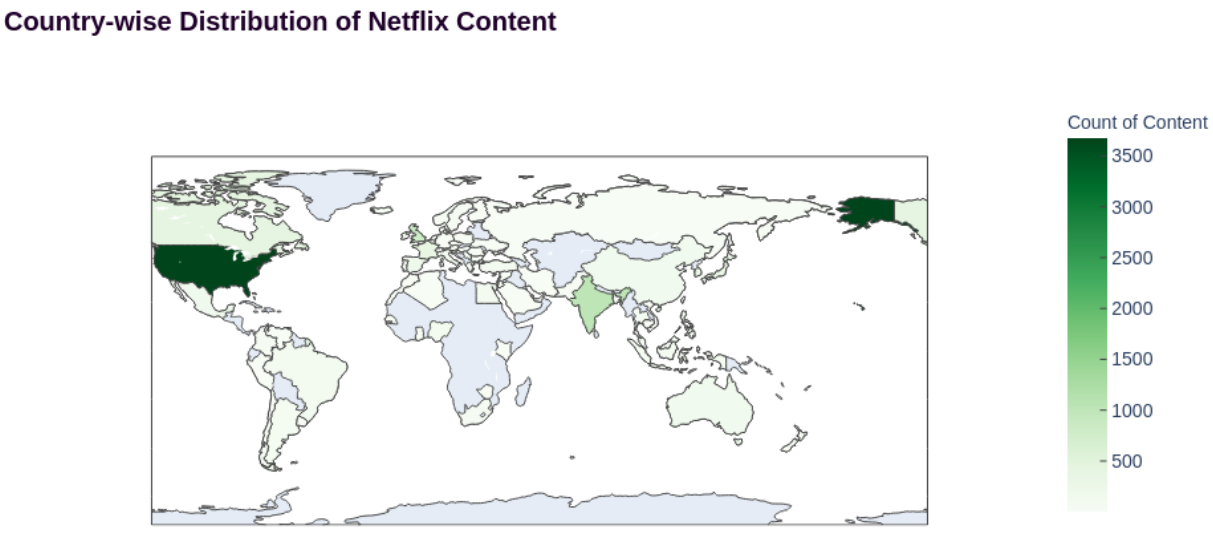
Plot 1: Pie Chart for Distribution of Netflix Content based on Type



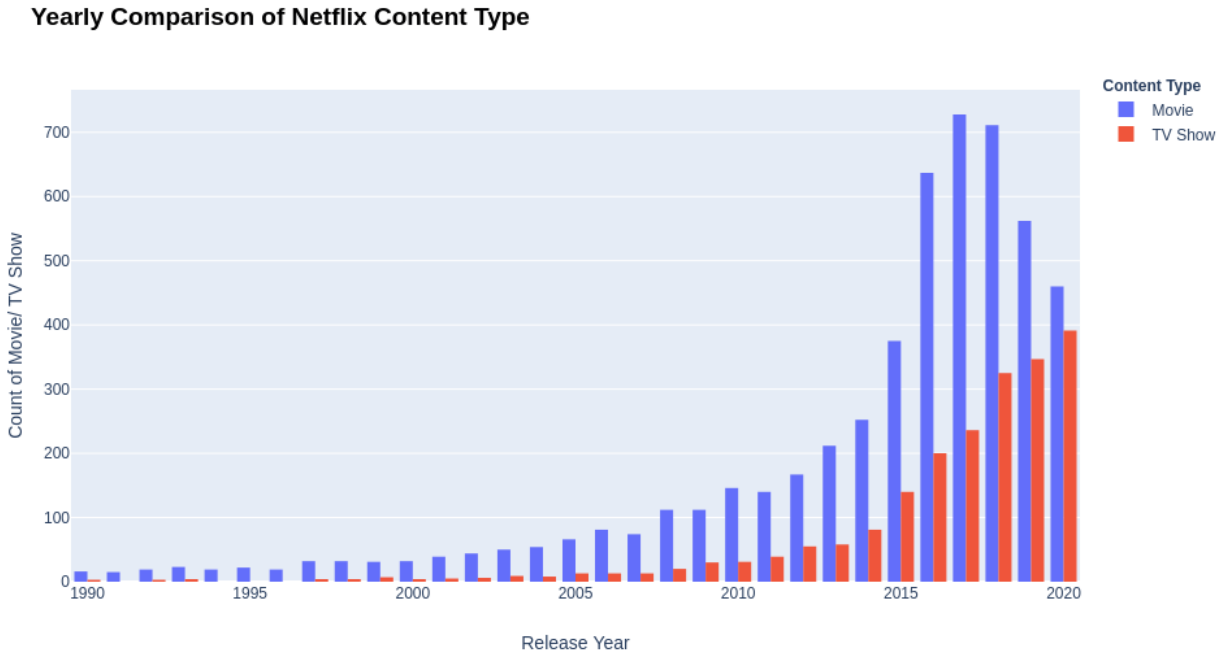
Plot 2: Bar Chart for Distribution of Netflix Content based on Rating



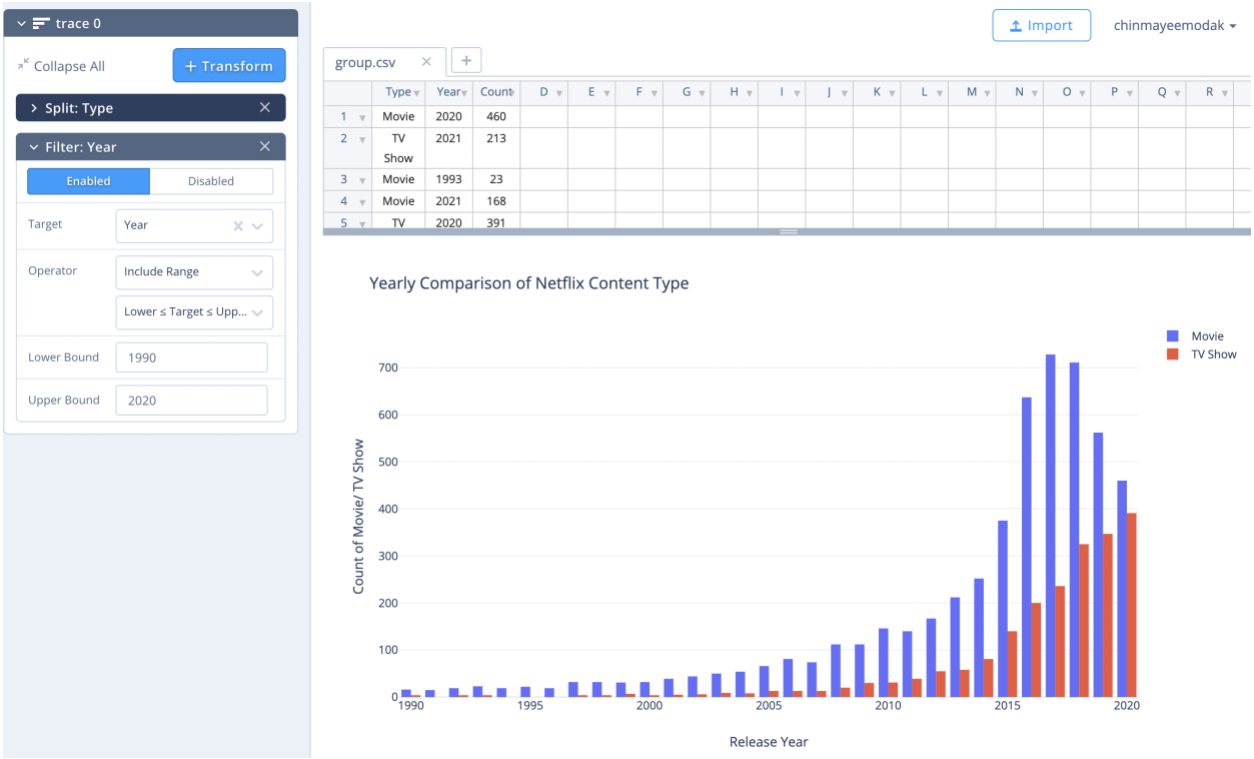
Plot 3: Bar Chart for Distribution of Netflix Content based on Country



Plot 4: Map for Distribution of Netflix Content based on Country



Plot 5: Bar Chart for Distribution of Netflix Content Yearly



Plot 6: Bar Chart for Distribution of Netflix Content Yearly with Filter

Discussion

- **Interpretation of the results**

The main objective of this project is to find trend in different variables combined or separate. The interpretation of results is looking at interesting aspects of data.

Based on the pie chart from plot 1 we can say that Netflix produces more movies than the TV Shows. Almost 71% of Netflix content are movies. This states that the profit generated by movies is much more than TV Shows. The major reason behind this might be the fact that Netflix doesn't self-produce most of its content.

Plot 2 of ratings of content across Netflix suggests that content with TV-MA rating is highest followed by TV-14. This also means that targeted audience for the platform is mainly young adults. This makes sense because people who mostly use technology are also young adults.

To answer the question which country produce more content there are two different plots for easier interpretation. In both the plots we can see that United States is the main market for Netflix content with more that 3500 available contents as it is also the origin country for Netflix and home to Hollywood. India based content follows United States due to the popularity of Bollywood.

Plot 5 shows bar chart of frequency of release over the years based on type of content. Year 2017 saw the highest rate of movie releases then it is decreasing gradually. Opposite of that can be observed for TV Shows. Number of TV Shows release is increasing periodically. In 2020 there is not much difference between the count of movies and TV shows released. 2019 and 2020 saw a sudden decrease in movie content release this could be due to COVID-19 pandemic and lockdown as many businesses suffered losses during that time frame.

- **Technologies/Skill employed**

The entire project and its data reside in the Jetstream VM instance which I learned in module “Virtualization”. Starting from how to create VM to the use and versatility of it was implemented for creating suitable environment. The data pipeline implemented here (Plan - Data load/acquire – Data Clean – Data Transformation – Data Analysis) resembles closely to USGS data Lifecycle model (Plan – Acquire – Process - Analyze) which I learned in “Lifecycle and Pipelines” module.

Use of PySpark RDD transformations and actions to manipulate data and know about the data type and its structure was done. Transformation functions such as filter, map, reduceByKey, sortByKey and actions like count, take, first, collect which we learned in “Processing and Analytics” module are used for this project. Further map – reduce evaluation for word count was also executed which basically does the parallel/distributed processing. This was part of “Distributed Computing and File Systems” module.

- **Barriers/Failures Encountered**

PySpark RDD have its own way of storing data. Superficially it might appear as a normal python data type like list but the commands to manipulate the pipelineRDD list is different. While splitting country column into different rows such that each row contains only one value, I struggled a lot. I was not familiar with underlying logic of how RDD in PySpark works. After trying numerous different things, I finally found a way around by using PySpark data frame.

The second point where I got stuck was the visualization part. I’ve used all the visualization libraries of python before, but I just was not able to figure out how RDD objects are to be plotted. I tried installing different supporting libraries and converting the RDD object, but it got much complicated. Finally, I decided to generate csv file of those results as I did for the “Complete: Analyzing with PySpark” exercise.

Overall, learning about PySpark methods was a bit challenging since I’m not very familiar with it. But after going through various blogs and videos I understood the gist of it.

Conclusion

Created data pipeline, gathers data from all sources into one common destination, also enables faster data analysis for business insights. Consistent data quality is also established, which is crucial for reliable business insights. The data transformation follows distributed processing approach to find and show pattern of different data variables can be implemented for any large dataset. Using PySpark it becomes convenient to combine to do parallel processing in virtual environment. Doing any huge task in virtual machine is quicker because of reduced workload and better uptime. Based on the result and its interpretation we can say that popularity of Netflix is not going down anytime soon as we saw the target customers of Netflix are young adults. This in turn implies that size and amount of data will increase exponentially. The more data we have more complexity is there which means there is scope for Big Data Analytics to grow even further.

References

1. https://towardsdatascience.com/how-data-science-is-boosting-netflix-785a1cba7e45
2. https://www.analyticsvidhya.com/blog/2021/08/an-introduction-to-data-analysis-using-spark-sql/
3. https://python.plainenglish.io/how-netflix-uses-data-analytics-data-science-general-research-case-study-4d525b881038
4. https://iu.instructure.com/courses/1999601/assignments/12257378
5. Dataset Source: https://www.kaggle.com/shivamb/netflix-shows
6. Code Source: https://databricksprodcloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/3328674740105987/4033840715400609/6441317451288404/latest.html
7. https://sparkbyexamples.com/pyspark/pyspark-map-transformation/