Naya College

Cloud Data Engineering

Apache Spark

Lab5: Streaming Ingestion

Contents

Overview of the Three Exercises:

In essence, across these three exercises, you will be ingesting, refining, and then streaming data in real-time. This journey will give you a comprehensive experience - from handling raw data to pushing it into a real-time data streaming platform.

Pre-requisites:

1. **Input Data Source Files**:

- googleplaystore.csv: Contains details about applications in the Google Play Store.
- googleplaystore_user_reviews.csv: Contains user reviews for the applications in the Google Play Store.

2. **Data Load Directories**:

    - Applications data: /data/raw/google_apps/
    - User reviews data: /data/raw/google_reviews/

Lab5-Task1: Google Play Store

Ingestion of Google Play Store Data:

In the first exercise, you will be focusing on processing the data related to Google Play Store applications. Starting from reading the raw application details and user reviews, you will transform and restructure this data according to a specific format. This involves parsing details like the name of the application, category, rating, and more. Any irrelevant or faulty data will be filtered out. The culmination of this exercise will be saving the processed data in a Parquet file format, optimized for analytics and big data workloads.

Sample data looks like below:

App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
11st	SHOPPING	3.8	48732	20M	10,000,000+	Free	0	Everyone	Shopping	July 31, 2018	7.8.1	4.0 and up
4 Paws PH	MEDICAL	NaN	1	29M	5+	Free	0	Everyone	Medical	July 25, 2018	300000.1.11	4.0.3 and up
4 in a row	GAME	4.3	47698	4.1M	5,000,000+	Free	0	Everyone	Board	April 12, 2016	1.0.21	1.5 and up
¡Ay Caramba!	FAMILY	NaN	0	549k	1+	Paid	$1.99	Everyone	Education	June 13, 2014	1.2	3.0 and up
2048(AI)	FAMILY	4.3	20	3.5M	1,000+	Free	0	Everyone	Casual	June 3, 2015	0.33	2.3 and up
2RedBeans	DATING	4	337	32M	10,000+	Free	0	Mature 17+	Dating	July 31, 2018	2.8.0	4.3 and up

Instructions:

Data Reading:

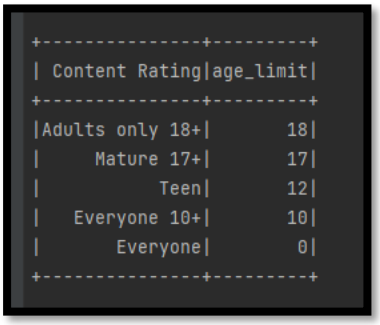
- Initiate by reading the data from the googleplaystore.csv file located at /data/raw/google_apps/.

Data Parsing:

Restructure the read data into a new format:

a. **App >> application_name** (String): Extract the application's name.
b. **Category >> category** (String): Identify the category of the app.
c. **Rating >> rating** (String): Extract the app rating. For any invalid or unknown values, replace them with **-1**.
d. **Reviews >> reviews** (Float).
e. **Size >> size** (String): Define the size of the app.
f. **installs >> num_of_installs** (double): Determine the number of installations. Make sure to remove any extra characters (For example, "10,000+" should become "10000").
g. **Price >> price** (double): Get the price of the application in the store.
h. **age_limit** (double): Convert "content rating" column into a minimum age requirement using the following mapping:
    - "Adults only 18+" → 18
    - "Mature 17+" → 17
    - "Teen" → 12
    - "Everyone 10+" → 10
    - "Everyone" → 0

Tip: create side dataframe as below and perform broadcast join on Content Rating Colum:



```
+---------------+---------+
| Content Rating|age_limit|
+---------------+---------+
|Adults only 18+|       18|
|     Mature 17+|       17|
|           Teen|       12|
|   Everyone 10+|       10|
|       Everyone|        0|
+---------------+---------+
```

i. **Genres >> genres** (String): Extract the genres associated with the app.
j. **Current ver >> version** (String): Identify the app's current version.

Data Cleaning:

- Filter out any rows that are either invalid or not relevant to the requirements.

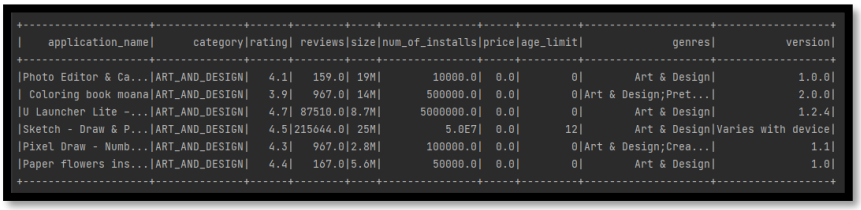
Data Saving:

- Store the final, transformed data in the **/data/source/google_apps** directory using the Parquet format.
- The script should be saved under the folder named **exercises_six** with the file name **google_apps.py**

Tip:

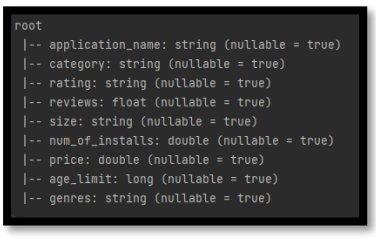
- As you go through the exercise, regularly validate your results at each step to ensure accuracy and proper transformation. This will help in mitigating any potential issues in the end result.
- use printing functions like:
  - selected_df.show()
  - selected_df.printSchema()

- Expected outpot:



```
+--------------------+-------------+------+-------+----+---------------+-----+---------+--------------------+-----------------+
|    application_name|     category|rating|reviews|size|num_of_installs|price|age_limit|              genres|          version|
+--------------------+-------------+------+-------+----+---------------+-----+---------+--------------------+-----------------+
|Photo Editor & Ca...|ART_AND_DESIGN|   4.1|  159.0| 19M|        10000.0|  0.0|        0|        Art & Design|            1.0.0|
| Coloring book moana|ART_AND_DESIGN|   3.9|  967.0| 14M|       500000.0|  0.0|        0|Art & Design;Pret...|            2.0.0|
|U Launcher Lite -...|ART_AND_DESIGN|   4.7|87510.0|8.7M|      5000000.0|  0.0|        0|        Art & Design|            1.2.4|
|Sketch - Draw & P...|ART_AND_DESIGN|   4.5|215644.0| 25M|         5.0E7|  0.0|       12|        Art & Design|Varies with device|
|Pixel Draw - Numb...|ART_AND_DESIGN|   4.3|  967.0|2.8M|       100000.0|  0.0|        0|Art & Design;Crea...|              1.1|
|Paper flowers ins...|ART_AND_DESIGN|   4.4|  167.0|5.6M|        50000.0|  0.0|        0|        Art & Design|              1.0|
+--------------------+-------------+------+-------+----+---------------+-----+---------+--------------------+-----------------+
```

- Expected schema:



```
root
 |-- application_name: string (nullable = true)
 |-- category: string (nullable = true)
 |-- rating: string (nullable = true)
 |-- reviews: float (nullable = true)
 |-- size: string (nullable = true)
 |-- num_of_installs: double (nullable = true)
 |-- price: double (nullable = true)
 |-- age_limit: long (nullable = true)
 |-- genres: string (nullable = true)
```

Lab5-Task2: Google Review Ingestion

Ingestion of Google Review Data:

The second exercise dives deeper into the realm of user reviews from the Google Play Store. From the previously ingested reviews data, the task will be to further refine it. You'll extract certain fields like the name of the application, the actual review text, and various sentiment metrics. Once you've restructured the data into the desired format, the end goal will be to save this information, again using the Parquet file format.

Sample data:

App	Translated_Review	Sentiment	Sentiment_Polarity	Sentiment_Subjectivity
11st	good enough	Positive	0.35	0.55
11st	best shop	Positive	1	0.3
11st	best	Positive	1	0.3
11st	good good	Positive	0.7	0.6
4 in a Row	Excellent	Positive	1	1
4 in a Row	Great	Positive	0.8	0.75

Instructions:

Data Reading:

- Start by loading the data from the googleplaystore_user_reviews.csv file situated at /data/raw/google_reviews/.

Data Structuring and Transformation:

Revise and arrange the imported data into a new layout:

- **App >> application_name (String):** Extract the application's name.
- **Translated_review >> translated_review (String):** Get the customer's review.
- **sentiment_rank (long):** Convert sentiment values to numeric scores:
  - If sentiment is "Positive", assign 1.
  - If sentiment is "Neutral", assign 0.
  - If sentiment is "Negative", assign -1.

Tip: create side dataframe as below and perform broadcast join on sentiment colum:

```
+---------+--------------+
|Sentiment|sentiment_rank|
+---------+--------------+
| Positive|             1|
|  Neutral|             0|
| Negative|            -1|
+---------+--------------+
```

- **Sentiment_polarity >> sentiment_polarity (Float):** Extract sentiment polarity directly from the input data.
- **Sentiment_subjectivity >> sentiment_subjectivity (Float):** Fetch sentiment subjectivity from the provided data.

Data Storing:

- Store the restructured data in the /data/source/google_reviews directory using the Parquet format.

- The dataset should be housed inside a folder named exercises_five and the file should be titled google_reviews.py

Tips:

- Frequently cross-check your intermediate results after each transformation step to guarantee precision and the desired format.
- Familiarize yourself with the source data before transformation to discern potential anomalies or issues.
- Expected outpot:

```
+--------------------+--------------------+-------------+-----------------+---------------------+
|    application_name|   translated_review|sentiment_rank|sentiment_polarity|sentiment_subjectivity|
+--------------------+--------------------+-------------+-----------------+---------------------+
|10 Best Foods for...|This help eating ...|            1|             0.25|           0.28846154|
|10 Best Foods for...|Works great espec...|            1|              0.4|                0.875|
|10 Best Foods for...|        Best idea us|            1|              1.0|                  0.3|
|10 Best Foods for...|            Best way|            1|              1.0|                  0.3|
|10 Best Foods for...|             Amazing|            1|              0.6|                  0.9|
|10 Best Foods for...|Looking forward app,|            0|              0.0|                  0.0|
|10 Best Foods for...|It helpful site !...|            0|              0.0|                  0.0|
```

- Expected schema:

```
root
 |-- application_name: string (nullable = true)
 |-- translated_review: string (nullable = true)
 |-- sentiment_rank: long (nullable = true)
 |-- sentiment_polarity: float (nullable = true)
 |-- sentiment_subjectivity: float (nullable = true)
```

Lab5-Task3: Producing to Kafka

Producing Data to Kafka:

In the third exercise, you'll work with one of the modern real-time data streaming platforms - Apache Kafka. The previously refined Google Play Store user reviews data will serve as your input. After performing any last-minute clean-ups or filtering on this data, you'll be setting up a Kafka producer. This producer will push the data to a specified Kafka topic. However, 😊, there's a catch. Instead of continuously streaming the data, you'll be controlling the rate, sending batches of 50 messages every 5 seconds, in the JSON format.

The task is to read the refined data and then produce the data to a Kafka topic at a controlled rate.

Instructions:

Data Reading:

- Begin by loading data from the Parquet file stored at **/data/source/google_reviews**.

Producing to Kafka:

- Set up a Kafka producer to send the data.
- Messages should be dispatched in batches of 50 every 5 seconds.
- Ensure that each message is in JSON format when sent to the Kafka topic.

Kafka Configuration:

- Produce data to the Kafka topic named **gps-user-review-source**.

Rate Limiting:

- Design a mechanism to control the rate of message production so that only 50 messages are sent every 5 seconds.

Steps:

1. Start by reading the Parquet file.
2. Configure your Kafka producer.
3. Implement a mechanism to produce messages to Kafka in batches and adhere to the rate requirement.
4. Validate the process by monitoring the messages on the **gps-user-review-source** Kafka topic.