Exercise on Big Data Fundamentals with PySpark

1. Understanding SparkContext

- Print the version of SparkContext in the PySpark shell.
- Print the Python version of SparkContext in the PySpark shell.
- What is the master of SparkContext in the PySpark shell?

2. Interactive Use of PySpark

- Create a python list named `numb` containing the numbers 1 to 100.
- Load the list into Spark using Spark Context's `parallelize` method and assign it to a variable `spark_data`.

3. Loading data in PySpark shell

- Load a local text file README.md in PySpark shell.

4. Use of lambda() with map()

- Print my_list which is available in your environment.
- Square each item in my_list using map() and lambda().
- Print the result of map function.

5. Use of lambda() with filter()

- Print `my_list2` which is available in your environment.
- Filter the numbers divisible by 10 from `my_list2` using `filter()` and `lambda()`.
- Print the numbers divisible by 10 from `my_list2`.

6. RDDs from Parallelized collections

- Create an RDD named RDD from a list of words.
- Confirm the object created is RDD.

7. RDDs from External Datasets

- Print the `file_path` in the PySpark shell.

- Create an RDD named `fileRDD` from a `file_path` with the file name `README.md`.
- Print the type of the `fileRDD` created.

8. Partitions in your data

- Find the number of partitions that support `fileRDD` RDD.
- Create an RDD named `fileRDD_part` from the file path but create 5 partitions.
- Confirm the number of partitions in the new `fileRDD_part` RDD.

9. Map and Collect

- Create `map()` transformation that cubes all of the numbers in `numbRDD`.
- Collect the results in a `numbers_all` variable.
- Print the output from `numbers_all` variable.

10. Filter and Count

- Create `filter()` transformation to select the lines containing the keyword `Spark`.
- How many lines in `fileRDD_filter` contains the keyword `Spark`?
- Print the first four lines of the resulting RDD.

11. ReduceBykey and Collect

- Create a pair RDD named `Rdd` with tuples `(1,2),(3,4),(3,6),(4,5)`.
- Transform the `Rdd` with `reduceByKey()` into a pair RDD `Rdd_Reduced` by adding the values with the same key.
- Collect the contents of pair RDD `Rdd_Reduced` and iterate to print the output.

12. SortByKey and Collect

- Sort the `Rdd_Reduced` RDD using the key in descending order.
- Collect the contents and iterate to print the output.

13. CountingBykeys

- Use the `countByKey()` action on the `Rdd` to count the unique keys and assign the result to a variable `total`.

- What is the type of `total`?
- Iterate over the `total` and print the keys and their counts.

14. Create a base RDD and transform it

- Create an RDD called `baseRDD` that reads lines from `file_path`.
- Transform the `baseRDD` into a long list of words and create a new `splitRDD`.
- Count the total words in `splitRDD`.

15. Remove stop words and reduce the dataset

- Convert the words in `splitRDD` in lower case and then remove stop words from `stop_words`.
- Create a pair RDD tuple containing the word and the number `1` from each word element in `splitRDD`.
- Get the count of the number of occurrences of each word (word frequency) in the pair RDD using `reduceByKey()`

16. Print word frequencies

- Print the first 10 words and their frequencies from the `resultRDD`.
- Swap the keys and values in the `resultRDD`.
- Sort the keys according to descending order.
- Print the top 10 most frequent words and their frequencies.

17. RDD to DataFrame

- Create a `sample_list` from tuples - `('Mona',20), ('Jennifer',34), ('John',20), ('Jim',26)`.
- Create an RDD from the `sample_list`.
- Create a PySpark DataFrame using the above RDD and schema.
- Confirm the output as PySpark DataFrame.

18. Loading CSV into DataFrame

- Create a DataFrame from `file_path` variable which is the path to the `people.csv` file.
- Confirm the output as PySpark DataFrame.

19. Inspecting data in PySpark DataFrame

- Print the first 10 observations in the `people_df` DataFrame.
- Count the number of rows in the `people_df` DataFrame.
- How many columns does `people_df` DataFrame have and what are their names?

20. PySpark DataFrame subsetting and cleaning

- Select 'name', 'sex' and 'date of birth' columns from `people_df` and create `people_df_sub` DataFrame.
- Print the first 10 observations in the `people_df` DataFrame.
- Remove duplicate entries from `people_df_sub` DataFrame and create `people_df_sub_nodup` DataFrame.
- How many rows are there before and after duplicates are removed?

21. Filtering your DataFrame

- Filter the `people_df` DataFrame to select all rows where sex is female into `people_df_female` DataFrame.
- Filter the `people_df` DataFrame to select all rows where sex is male into `people_df_male` DataFrame.
- Count the number of rows in `people_df_female` and `people_df_male` DataFrames.

22. Running SQL Queries Programmatically

- Create a temporary table `people` that's a pointer to the `people_df` DataFrame.
- Construct a `query` to select the names of the people from the temporary table `people`.
- Assign the result of Spark's `query` to a new DataFrame - `people_df_names`.
- Print the top 10 names of the people from `people_df_names` DataFrame.

23. SQL queries for filtering Table

- Filter the `people` table to select all rows where sex is female into `people_female_df` DataFrame.
- Filter the `people` table to select all rows where sex is male into `people_male_df` DataFrame.
- Count the number of rows in both `people_female` and `people_male` DataFrames.

24. PySpark DataFrame visualization

- Print the names of the columns in `names_df` DataFrame.
- Convert `names_df` DataFrame to `df_pandas` Pandas DataFrame.

- Use matplotlib's `plot()` method to create a horizontal bar plot with `'Name'` on x-axis and `'Age'` on y-axis.

25. Part 1: Create a DataFrame from CSV file

- Create a PySpark DataFrame from file_path which is the path to the Fifa2018_dataset.csv file.
- Print the schema of the DataFrame.
- Print the first 10 observations.
- How many rows are in there in the DataFrame?

26. Part 2: SQL Queries on DataFrame

- Create temporary table `fifa_df` from `fifa_df_table` DataFrame.
- Construct a "query" to extract the "Age" column from Germany players.
- Apply the SQL "query" to the temporary view table and create a new DataFrame.
- Computes basic statistics of the created DataFrame.

27. Part 3: Data visualization

- Convert `fifa_df_germany_age` to `fifa_df_germany_age_pandas` Pandas DataFrame.
- Generate a density plot of the 'Age' column from the `fifa_df_germany_age_pandas` Pandas DataFrame.

28. PySpark MLlib algorithms

- Import `pyspark.mllib` recommendation submodule and Alternating Least Squares class.
- Import `pyspark.mllib` classification submodule and Logistic Regression with LBFGS class.
- Import `pyspark.mllib` clustering submodule and kmeans class.

29. Loading Movie Lens dataset into RDDs

- Load the `ratings.csv` dataset into an RDD.
- Split the RDD using `,` as a delimiter.
- For each line of the RDD, using `Rating()` class create a tuple of `userID`, `productID`, `rating`.
- Randomly split the data into training data and test data (0.8 and 0.2).

30. Model training and predictions

- Train ALS algorithm with training data and configured parameters (`rank` = 10 and `iterations` = 10).
- Drop the `rating` column in the test data.
- Test the model by predicting the rating from the test data.
- Print the first two rows of the predicted ratings.

31. Model evaluation using MSE

- Organize `ratings` RDD to make `((user, product), rating)`.
- Organize `predictions` RDD to make `((user, product), rating)`.
- Join the prediction RDD with the ratings RDD.
- Evaluate the model using MSE between original rating and predicted rating and print it.

32. Loading spam and non-spam data

- Create two RDDS, one for 'spam' and one for 'non-spam (ham)'.
- Split each email in 'spam' and 'non-spam' RDDs into words.
- Print the first element in the split RDD of both 'spam' and 'non-spam'.

33. Feature hashing and LabelPoint

- Create a `HashingTF()` instance to map email text to vectors of 200 features.
- Each message in 'spam' and 'non-spam' datasets are split into words, and each word is mapped to one feature.
- Label the features: 1 for spam, 0 for non-spam.
- Combine both the spam and non-spam samples into a single dataset.

34. Logistic Regression model training

- Split the combined data into training and test sets (80/20).
- Train the Logistic Regression (`LBFGS` variant) model with the training dataset.
- Create a prediction label from the trained model on the test dataset.
- Combine the labels in the test dataset with the labels in the prediction dataset.
- Calculate the accuracy of the trained model using original and predicted labels on the `labels_and_preds`.

35. Loading and parsing the 5000 points data

- Load the `5000_points` dataset into a RDD named `clusterRDD`.
- Transform the `clusterRDD` by splitting the lines based on the tab ("\t").
- Transform the split RDD to create a list of integers for the two columns.
- Confirm that there are 5000 rows in the dataset.

36. K-means training

- Train the KMeans model with clusters from 13 to 16 and print the WSSSE for each cluster.
- Train the KMeans model again with the best cluster (k=15).
- Get the Cluster Centers (centroids) of KMeans model trained with k=15.

37. Visualizing clusters

- Convert `rdd_split_int` RDD into a Spark DataFrame.
- Convert Spark DataFrame into a Pandas DataFrame.
- Create a Pandas DataFrame from `cluster_centers` list.
- Create a scatter plot of the raw data and an overlaid scatter plot with centroids for k = 15.