```
Kmeans
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('cluster').getOrCreate()
print('Spark Version: {}'.format(spark.version))

#Loading the data
dataset = spark.read.csv("seeds_dataset.csv",header=True,inferSchema=True)
#show the data in the above file using the below command
dataset.show(5)
#Print schema
dataset.printSchema()

from pyspark.ml.feature import VectorAssembler
vec_assembler = VectorAssembler(inputCols = dataset.columns,outputCol='features')
final_data = vec_assembler.transform(dataset)
final_data.select('features').show(5)
fdata = final_data.select('features')


from pyspark.ml.feature import StandardScaler
scaler =
StandardScaler(inputCol="features",outputCol="scaledFeatures",withStd=True,withMe
an=False)
# Compute summary statistics by fitting the StandardScaler
scalerModel = scaler.fit(final_data)
# Normalize each feature to have unit standard deviation.
final_data = scalerModel.transform(final_data)
final_data.select('scaledFeatures').show(5)

#Importing the model
from pyspark.ml.clustering import KMeans
from pyspark.ml.evaluation import ClusteringEvaluator
silhouette_score=[]
evaluator = ClusteringEvaluator(predictionCol='prediction',
featuresCol='scaledFeatures', metricName='silhouette',
distanceMeasure='squaredEuclidean')
for i in range(2,10):
    kmeans=KMeans(featuresCol='scaledFeatures', k=i)
    model=kmeans.fit(final_data)
    predictions=model.transform(final_data)
    score=evaluator.evaluate(predictions)
    silhouette_score.append(score)
    print('Silhouette Score for k =',i,'is',score)

#Visualizing the silhouette scores in a plot
```

```
import matplotlib.pyplot as plt

plt.plot(range(2,10),silhouette_score)
plt.xlabel('k')
plt.ylabel('silhouette score')
plt.title('Silhouette Score')
plt.show()

# Trains a k-means model.
kmeans = KMeans(featuresCol='scaledFeatures',k=3)
model = kmeans.fit(final_data)
predictions = model.transform(final_data)

# Printing cluster centers
centers = model.clusterCenters()
print("Cluster Centers: ")
for center in centers:
    print(center)
predictions.select('prediction').show(5)
#End Session
spark.stop()
```

Linear regression

```
#linear regression
import pyspark
from pyspark.sql import SparkSession

spark=SparkSession.builder.appName('housing_price_model').getOrCreate()

df=spark.read.csv('./cruise_ship_info.csv',inferSchema=True,header=True)
df.show(10)
df.printSchema()
df.columns

from pyspark.ml.feature import StringIndexer
indexer=StringIndexer(inputCol='Cruise_line',outputCol='cruise_cat')
indexed=indexer.fit(df).transform(df)

for item in indexed.head(5):
    print(item)


from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
```

```
assembler=VectorAssembler(inputCols=['Age',
 'Tonnage',
 'passengers',
 'length',
 'cabins',
 'passenger_density',
 'cruise_cat'],outputCol='features')
output=assembler.transform(indexed)
output.select('features','crew').show(5)

final_data=output.select('features','crew')
train_data,test_data=final_data.randomSplit([0.7,0.3])
train_data.describe().show()

from pyspark.ml.regression import LinearRegression

ship_lr=LinearRegression(featuresCol='features',labelCol='crew')
trained_ship_model=ship_lr.fit(train_data)
ship_results=trained_ship_model.evaluate(train_data)

print("Rsquared Error: ",ship_results.r2)

unlabeled_data=test_data.select('features')
unlabeled_data.show(5)
predictions=trained_ship_model.transform(unlabeled_data)
predictions.show()
```

Mapreduce for count of words/ frequency

```
#mapper logic
import sys

for line in sys.stdin:
    line=line.strip()
    words=line.split()

    for word in words:
        print(f"{word}\t1")

#reducer logic
```

```
import sys

current_word=None
current_count=0
word=None

for line in sys.stdin:
    line=line.strip()
    word,count=line.split('\t',1)
    count=int(count)
    if current_word==word:
        current_count+=count
    else:
        if current_word:
            print(f"{current_word}\t{current_count}")
        current_count=count
        current_word=word
if current_word==word:
    print(f"{current_word}\t{current_count}")
```

Data Visualization

```
#Visuallization
import pandas
import matplotlib.pyplot as plt

df=pandas.read_csv('tips.csv')
df.head()

#scatter plot
plt.scatter(df['day'],df['tip'])
plt.title('scatter plot')
plt.show()

#linear plot
plt.plot(df['tip'])
plt.plot(df['size'])

# Adding Title to the Plot
plt.title("Scatter Plot")

# Setting the X and Y labels
```

```
plt.xlabel('Day')
plt.ylabel('Tip')
plt.show()

#bar plot
plt.bar(df['day'],df['tip'])
plt.title("Bar Chart")

# Setting the X and Y labels
plt.xlabel('Day')
plt.ylabel('Tip')

#histogram
plt.hist(df['total_bill'])
plt.title("Histogram")
plt.show()

import seaborn as sns
#linear plot
sns.lineplot(x="sex",y="total_bill",data=df)
#scatter plot
sns.scatterplot(x='day',y='tip',data=df,hue='sex')
#bar plot
sns.barplot(x='day',y='tip',data=df,hue='sex')
#histogram
sns.histplot(x='total_bill',data=df,kde=True,hue='sex')
plt.show()
```

Processing data using spark

```
# Import necessary PySpark libraries
from pyspark.sql import SparkSession

# Create a Spark session
spark = SparkSession.builder.appName("DataSetProcessing").getOrCreate()

# Load the CSV file into a DataFrame
data = spark.read.csv("dataprocess.csv", header=True, inferSchema=True)

# Show the first few rows of the dataset
data.show()
```

```
# Filter the dataset based on a condition
filtered_data = data.filter(data['age'] > 30)

# Show the filtered data
filtered_data.show()

# Perform some basic aggregation, e.g., calculating the average age
average_age = data.agg({'age': 'avg'})
average_age.show()


# Stop the Spark session when you're done
spark.stop()
```