In [1]:

```
import pyspark
```

In [2]:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import split, explode, col
```

In [3]:

```
spark = SparkSession.builder \
    .appName("WordCountExample") \
    .getOrCreate()
spark
```

```
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.p
roperties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
23/08/05 14:09:03 WARN NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicab
le
```

Out[3]:

SparkSession - in-memory

SparkContext

Spark UI (http://192.168.0.11:4040)

Version

`v3.2.1`

Master

`local[*]`

AppName

`WordCountExample`

In [119]:

```
text_file_path = "/Users/myyntiimac/Desktop/Salary_Data.csv"

# Read the text file into a DataFrame
df = spark.read.csv(text_file_path)
```

In [120]:

```
df.show()
```

```
+---------------+------+
|            _c0|   _c1|
+---------------+------+
|YearsExperience|Salary|
|            1.1| 39343|
|              0| 21111|
|            1.5| 37731|
|              2| 43525|
|            2.2| 39891|
|            2.9| 56642|
|              3| 60105|
|            3.2|  null|
|            3.2| 64445|
|            3.7| 57189|
|            3.9| 63218|
|              4| 55794|
|              4| 56957|
|            4.1| 57081|
|            4.5| 61111|
|              0| 21111|
|            5.1|  null|
|            5.3| 83088|
|            5.9| 81363|
+---------------+------+
only showing top 20 rows
```

In [121]:

```
df=spark.read.csv(text_file_path,header=True,inferSchema=True)
```

In [122]:

```
df.show()
```

```
+---------------+------+
|YearsExperience|Salary|
+---------------+------+
|            1.1| 39343|
|            0.0| 21111|
|            1.5| 37731|
|            2.0| 43525|
|            2.2| 39891|
|            2.9| 56642|
|            3.0| 60105|
|            3.2|  null|
|            3.2| 64445|
|            3.7| 57189|
|            3.9| 63218|
|            4.0| 55794|
|            4.0| 56957|
|            4.1| 57081|
|            4.5| 61111|
|            0.0| 21111|
|            5.1|  null|
|            5.3| 83088|
|            5.9| 81363|
|            6.0|  9394|
+---------------+------+
only showing top 20 rows
```

In [123]:

```
df.printSchema()
```

```
root
 |-- YearsExperience: double (nullable = true)
 |-- Salary: integer (nullable = true)
```

In [124]:

```
#Handling missing value
df.na.drop().show()
```

```
+---------------+------+
|YearsExperience|Salary|
+---------------+------+
|            1.1| 39343|
|            0.0| 21111|
|            1.5| 37731|
|            2.0| 43525|
|            2.2| 39891|
|            2.9| 56642|
|            3.0| 60105|
|            3.2| 64445|
|            3.7| 57189|
|            3.9| 63218|
|            4.0| 55794|
|            4.0| 56957|
|            4.1| 57081|
|            4.5| 61111|
|            0.0| 21111|
|            5.3| 83088|
|            5.9| 81363|
|            6.0|  9394|
|            0.0| 21111|
|            7.1| 98273|
+---------------+------+
only showing top 20 rows
```

no row contain all null value Check with other method

In [125]:

```python
df.na.drop(how ="all").show()
```

```
+---------------+------+
|YearsExperience|Salary|
+---------------+------+
|            1.1| 39343|
|            0.0| 21111|
|            1.5| 37731|
|            2.0| 43525|
|            2.2| 39891|
|            2.9| 56642|
|            3.0| 60105|
|            3.2|  null|
|            3.2| 64445|
|            3.7| 57189|
|            3.9| 63218|
|            4.0| 55794|
|            4.0| 56957|
|            4.1| 57081|
|            4.5| 61111|
|            0.0| 21111|
|            5.1|  null|
|            5.3| 83088|
|            5.9| 81363|
|            6.0|  9394|
+---------------+------+
only showing top 20 rows
```

In [126]:

```python
#define imputer
from pyspark.ml.feature import Imputer
imputer_mean = Imputer(
    inputCols=["YearsExperience", "Salary"],
    outputCols=["{}_imputed".format(c) for c in ["YearsExperience", "Salary"]]
).setStrategy("mean")

```

In [127]:

```python
# Fit and transform the DataFrame using the imputer
imputed_df = imputer_mean.fit(df).transform(df)
```

In [128]:

```
imputed_df.show()
```

```
+---------------+------+-----------------------+--------------+
|YearsExperience|Salary|YearsExperience_imputed|Salary_imputed|
+---------------+------+-----------------------+--------------+
|            1.1| 39343|                    1.1|         39343|
|            0.0| 21111|                    0.0|         21111|
|            1.5| 37731|                    1.5|         37731|
|            2.0| 43525|                    2.0|         43525|
|            2.2| 39891|                    2.2|         39891|
|            2.9| 56642|                    2.9|         56642|
|            3.0| 60105|                    3.0|         60105|
|            3.2|  null|                    3.2|         62890|
|            3.2| 64445|                    3.2|         64445|
|            3.7| 57189|                    3.7|         57189|
|            3.9| 63218|                    3.9|         63218|
|            4.0| 55794|                    4.0|         55794|
|            4.0| 56957|                    4.0|         56957|
|            4.1| 57081|                    4.1|         57081|
|            4.5| 61111|                    4.5|         61111|
|            0.0| 21111|                    0.0|         21111|
|            5.1|  null|                    5.1|         62890|
|            5.3| 83088|                    5.3|         83088|
|            5.9| 81363|                    5.9|         81363|
|            6.0|  9394|                    6.0|          9394|
+---------------+------+-----------------------+--------------+
only showing top 20 rows
```

In [129]:

```
imputed_df.printSchema()
```

```
root
 |-- YearsExperience: double (nullable = true)
 |-- Salary: integer (nullable = true)
 |-- YearsExperience_imputed: double (nullable = true)
 |-- Salary_imputed: integer (nullable = true)
```

In [130]:

```
imputed_df.columns
```

Out[130]:

```
['YearsExperience', 'Salary', 'YearsExperience_imputed', 'Salary_imput
ed']
```

In [131]:

```
df1=imputed_df['YearsExperience_imputed', 'Salary_imputed']
```

In [132]:

```
df1.show()
```

```
+-----------------------+--------------+
|YearsExperience_imputed|Salary_imputed|
+-----------------------+--------------+
|                    1.1|         39343|
|                    0.0|         21111|
|                    1.5|         37731|
|                    2.0|         43525|
|                    2.2|         39891|
|                    2.9|         56642|
|                    3.0|         60105|
|                    3.2|         62890|
|                    3.2|         64445|
|                    3.7|         57189|
|                    3.9|         63218|
|                    4.0|         55794|
|                    4.0|         56957|
|                    4.1|         57081|
|                    4.5|         61111|
|                    0.0|         21111|
|                    5.1|         62890|
|                    5.3|         83088|
|                    5.9|         81363|
|                    6.0|          9394|
+-----------------------+--------------+
only showing top 20 rows
```

In [133]:

```
# Rename the columns in the DataFrame
renamed_df = df1.withColumnRenamed("YearsExperience_imputed", "Experience") \
                         .withColumnRenamed("Salary_imputed", "Salary")

# Show the renamed DataFrame
renamed_df.show()
```

```
+----------+------+
|Experience|Salary|
+----------+------+
|       1.1| 39343|
|       0.0| 21111|
|       1.5| 37731|
|       2.0| 43525|
|       2.2| 39891|
|       2.9| 56642|
|       3.0| 60105|
|       3.2| 62890|
|       3.2| 64445|
|       3.7| 57189|
|       3.9| 63218|
|       4.0| 55794|
|       4.0| 56957|
|       4.1| 57081|
|       4.5| 61111|
|       0.0| 21111|
|       5.1| 62890|
|       5.3| 83088|
|       5.9| 81363|
|       6.0|  9394|
+----------+------+
only showing top 20 rows
```

In [171]:

```
from pyspark.sql.functions import col, sum, when
# Count null values in each column using 'agg()'
null_counts = renamed_df.agg(*[sum(when(col(c).isNull(), 1).otherwise(0)).alias(

# Display the DataFrame
null_counts.show()
```

```
+----------+------+
|Experience|Salary|
+----------+------+
|         0|     0|
+----------+------+
```

In [189]:

```
from pyspark.ml.feature import VectorAssembler

assembler = VectorAssembler(inputCols=["Experience"], outputCol="features")
df3 = assembler.transform(renamed_df)
```

In [192]:

```python
from pyspark.ml.regression import LinearRegression

# Create a LinearRegression model
# Create a Linear Regression model with non-zero regParam
lr = LinearRegression(featuresCol="features", labelCol="Salary", regParam=0.01)


# Fit the model to the data
lr_model = lr.fit(df3)
```

In [193]:

```python
# Make predictions using the model
predictions = lr_model.transform(df3)

# Show the predictions
predictions.select("Experience", "Salary", "prediction").show()
```

```
+----------+------+------------------+
|Experience|Salary|        prediction|
+----------+------+------------------+
|       1.1| 39343| 32846.95255634046|
|       0.0| 21111| 23675.52271692542|
|       1.5| 37731| 36182.01795249138|
|       2.0| 43525| 40350.84969768003|
|       2.2| 39891| 42018.38239575549|
|       2.9| 56642|  47854.7468390196|
|       3.0| 60105| 48688.51318805733|
|       3.2| 62890| 50356.04588613279|
|       3.2| 64445| 50356.04588613279|
|       3.7| 57189| 54524.87763132145|
|       3.9| 63218| 56192.41032939691|
|       4.0| 55794| 57026.17667843464|
|       4.0| 56957| 57026.17667843464|
|       4.1| 57081| 57859.94302747237|
|       4.5| 61111|61195.008423623294|
|       0.0| 21111| 23675.52271692542|
|       5.1| 62890| 66197.60651784966|
|       5.3| 83088| 67865.13921592513|
|       5.9| 81363| 72867.73731015151|
|       6.0|  9394| 73701.50365918924|
+----------+------+------------------+
only showing top 20 rows
```

In [ ]:

```python

```