Task 1 - SQL

Build SparkSession:

In [2]:
```
import findspark
findspark.init()
import pyspark
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName('SparkSQL').enableHiveSupport().getOrCr
```

```
21/10/26 16:06:38 WARN Utils: Your hostname, yousri-Lenovo-Legion-5-15IMH05H re
solves to a loopback address: 127.0.1.1; using 192.168.1.105 instead (on interf
ace wlp0s20f3)
21/10/26 16:06:38 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/o
pt/spark/jars/spark-unsafe_2.12-3.0.1.jar) to constructor java.nio.DirectByteBu
ffer(long,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.
unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflec
tive access operations
WARNING: All illegal access operations will be denied in a future release
21/10/26 16:06:50 WARN NativeCodeLoader: Unable to load native-hadoop library f
or your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLev
el(newLevel).
```

Read the json file:

In [24]:
```
json_file = 'DataFrames_sample.json'
df_schema = "Id INT, Model STRING, Year INT, ScreenSize STRING, RAM STRING,
df = spark.read.format('json').schema(df_schema).option('header','True').loa
```

In [25]:
```
# OR simply
df = spark.read.json(json_file, schema=df_schema)
```

Display the schema:

In [26]:
```
1 df.printSchema()
```

```
root
 |-- Id: integer (nullable = true)
 |-- Model: string (nullable = true)
 |-- Year: integer (nullable = true)
 |-- ScreenSize: string (nullable = true)
 |-- RAM: string (nullable = true)
 |-- HDD: string (nullable = true)
 |-- W: double (nullable = true)
 |-- D: double (nullable = true)
 |-- H: double (nullable = true)
 |-- Weight: double (nullable = true)
```

In [27]:
```
1 df.show()
```

```
+---+-----------+----+----------+----+---------+-----+----+----+------+
| Id|      Model|Year|ScreenSize| RAM|      HDD|    W|   D|   H|Weight|
+---+-----------+----+----------+----+---------+-----+----+----+------+
|  1|MacBook Pro|2015|       15"|16GB|512GB SSD|13.75|9.48|0.61|  4.02|
|  2|    MacBook|2016|       12"| 8GB|256GB SSD|11.04|7.74|0.52|  2.03|
|  3|MacBook Air|2016|     13.3"| 8GB|128GB SSD| 12.8|8.94|0.68|  2.96|
|  4|       iMac|2017|       27"|64GB|  1TB SSD| 25.6| 8.0|20.3|  20.8|
+---+-----------+----+----------+----+---------+-----+----+----+------+
```

Get all the data when "Model" equal "MacBook Pro":

In [30]:
```
1 from pyspark.sql.functions import *
```

In [32]:
```
1 df.where(col('Model') == 'MacBook Pro').show()
```

```
+---+-----------+----+----------+----+---------+-----+----+----+------+
| Id|      Model|Year|ScreenSize| RAM|      HDD|    W|   D|   H|Weight|
+---+-----------+----+----------+----+---------+-----+----+----+------+
|  1|MacBook Pro|2015|       15"|16GB|512GB SSD|13.75|9.48|0.61|  4.02|
+---+-----------+----+----------+----+---------+-----+----+----+------+
```

In [35]:
```
1 spark.sql(""" SELECT *
2               FROM APPLE
3               WHERE MODEL == 'MacBook Pro'
4             """).show()
```

```
+---+-----------+----+----------+----+---------+-----+----+----+------+
| Id|      Model|Year|ScreenSize| RAM|      HDD|    W|   D|   H|Weight|
+---+-----------+----+----------+----+---------+-----+----+----+------+
|  1|MacBook Pro|2015|       15"|16GB|512GB SSD|13.75|9.48|0.61|  4.02|
+---+-----------+----+----------+----+---------+-----+----+----+------+
```

Create TempView:

In [63]:
```
# register the DataFrame as a temporary view
df.createOrReplaceTempView("APPLE")
```

Display "RAM"column and count "RAM" column:

In [43]:
```
spark.sql(""" SELECT RAM
              FROM APPLE
              """).show()
spark.sql(""" SELECT COUNT(RAM)
              FROM APPLE
              """).show()
```

```
+----+
| RAM|
+----+
|16GB|
| 8GB|
| 8GB|
|64GB|
+----+

+----------+
|count(RAM)|
+----------+
|         4|
+----------+
```

Get all columns when "Year" column equal "2015"

In [44]:
```
spark.sql(""" SELECT *
              FROM APPLE
              WHERE Year = 2015
          """).show()
```

```
+---+-----------+----+----------+----+---------+-----+----+----+------+
| Id|      Model|Year|ScreenSize| RAM|      HDD|    W|   D|   H|Weight|
+---+-----------+----+----------+----+---------+-----+----+----+------+
|  1|MacBook Pro|2015|       15"|16GB|512GB SSD|13.75|9.48|0.61|  4.02|
+---+-----------+----+----------+----+---------+-----+----+----+------+
```

In [45]:
```
1 df.where(col('Year') == 2015).show()
```

```
+---+----------+----+----------+----+---------+-----+----+----+------+
| Id|     Model|Year|ScreenSize| RAM|      HDD|    W|   D|   H|Weight|
+---+----------+----+----------+----+---------+-----+----+----+------+
|  1|MacBook Pro|2015|       15"|16GB|512GB SSD|13.75|9.48|0.61|  4.02|
+---+----------+----+----------+----+---------+-----+----+----+------+
```

Get all when "Model" start with "M":

In [46]:
```
1 df.where(col('Model').startswith('M')).show()
```

```
+---+----------+----+----------+----+---------+-----+----+----+------+
| Id|     Model|Year|ScreenSize| RAM|      HDD|    W|   D|   H|Weight|
+---+----------+----+----------+----+---------+-----+----+----+------+
|  1|MacBook Pro|2015|       15"|16GB|512GB SSD|13.75|9.48|0.61|  4.02|
|  2|    MacBook|2016|       12"| 8GB|256GB SSD|11.04|7.74|0.52|  2.03|
|  3|MacBook Air|2016|     13.3"| 8GB|128GB SSD| 12.8|8.94|0.68|  2.96|
+---+----------+----+----------+----+---------+-----+----+----+------+
```

In [47]:
```
1 spark.sql(""" SELECT *
2               FROM APPLE
3               WHERE Model LIKE 'M%'
4             """).show()
```

```
+---+----------+----+----------+----+---------+-----+----+----+------+
| Id|     Model|Year|ScreenSize| RAM|      HDD|    W|   D|   H|Weight|
+---+----------+----+----------+----+---------+-----+----+----+------+
|  1|MacBook Pro|2015|       15"|16GB|512GB SSD|13.75|9.48|0.61|  4.02|
|  2|    MacBook|2016|       12"| 8GB|256GB SSD|11.04|7.74|0.52|  2.03|
|  3|MacBook Air|2016|     13.3"| 8GB|128GB SSD| 12.8|8.94|0.68|  2.96|
+---+----------+----+----------+----+---------+-----+----+----+------+
```

Get all data when "Model" column equal "MacBook Pro"

In [50]:
```
1 spark.sql(""" SELECT *
2               FROM APPLE
3               WHERE MODEL == 'MacBook Pro'
4             """).show()
```

```
+---+----------+----+----------+----+---------+-----+----+----+------+
| Id|     Model|Year|ScreenSize| RAM|      HDD|    W|   D|   H|Weight|
+---+----------+----+----------+----+---------+-----+----+----+------+
|  1|MacBook Pro|2015|       15"|16GB|512GB SSD|13.75|9.48|0.61|  4.02|
+---+----------+----+----------+----+---------+-----+----+----+------+
```

Get all data with Multiple Conditions when "RAM" column equal "8GB"

and "Model" column is "Macbook".

In [51]:
```
spark.sql(""" SELECT *
              FROM APPLE
              WHERE RAM == '8GB' and Model == 'MacBook'
          """).show()
```

```
+---+-------+----+----------+---+---------+-----+----+----+------+
| Id|  Model|Year|ScreenSize|RAM|      HDD|    W|   D|   H|Weight|
+---+-------+----+----------+---+---------+-----+----+----+------+
|  2|MacBook|2016|       12"|8GB|256GB SSD|11.04|7.74|0.52|  2.03|
+---+-------+----+----------+---+---------+-----+----+----+------+
```

Get all data with Multiple Conditions when "D" greater than or equal "8" and "Model" column is "iMac".

In [52]:
```
spark.sql(""" SELECT *
              FROM APPLE
              WHERE D = 8 and Model = 'iMac'
            """).show()
```

```
+---+-----+----+----------+----+-------+----+---+----+------+
| Id|Model|Year|ScreenSize| RAM|    HDD|   W|  D|   H|Weight|
+---+-----+----+----------+----+-------+----+---+----+------+
|  4| iMac|2017|       27"|64GB|1TB SSD|25.6|8.0|20.3|  20.8|
+---+-----+----+----------+----+-------+----+---+----+------+
```

Task 2

Read "test1" dataset:

In [56]:
```
test1_schema = 'Name STRING, age INT, Experience INT, Salary DOUBLE'
df_test1 = spark.read.format('csv').schema(test1_schema).option('header','Tr
df_test1.printSchema()
```

```
root
 |-- Name: string (nullable = true)
 |-- age: integer (nullable = true)
 |-- Experience: integer (nullable = true)
 |-- Salary: double (nullable = true)
```

In [57]:
```
1 df_test1.show()
```

```
+---------+---+----------+-------+
|     Name|age|Experience| Salary|
+---------+---+----------+-------+
|    Krish| 31|        10|30000.0|
|Sudhanshu| 30|         8|25000.0|
|    Sunny| 29|         4|20000.0|
|     Paul| 24|         3|20000.0|
|   Harsha| 21|         1|15000.0|
|  Shubham| 23|         2|18000.0|
+---------+---+----------+-------+
```

In [65]:
```
1 # register the DataFrame as a temporary view
2 df_test1.createOrReplaceTempView("test1")
```

Display Salary of the people less than or equal to 20000

In [70]:
```
1 df_test1.select('Name','Salary').where(col('Salary') <= 20000).show()
```

```
+-------+-------+
|   Name| Salary|
+-------+-------+
|  Sunny|20000.0|
|   Paul|20000.0|
| Harsha|15000.0|
|Shubham|18000.0|
+-------+-------+
```

In [66]:
```
1 spark.sql(""" SELECT Name , Salary
2               FROM test1
3               WHERE Salary <= 20000""").show()
```

```
+-------+-------+
|   Name| Salary|
+-------+-------+
|  Sunny|20000.0|
|   Paul|20000.0|
| Harsha|15000.0|
|Shubham|18000.0|
+-------+-------+
```

Display Salary of the people less than or equal to 20000 and greater than or equal 15000

In [67]:
```
spark.sql(""" SELECT Name , Salary
              FROM test1
              WHERE Salary <= 20000 and Salary >=15000""").show()
```

```
+-------+-------+
|   Name| Salary|
+-------+-------+
|  Sunny|20000.0|
|   Paul|20000.0|
| Harsha|15000.0|
|Shubham|18000.0|
+-------+-------+
```

In [72]:
```
df_test1.select('Name','Salary').filter(col('Salary').between(15000,20000)).
```

```
+-------+-------+
|   Name| Salary|
+-------+-------+
|  Sunny|20000.0|
|   Paul|20000.0|
| Harsha|15000.0|
|Shubham|18000.0|
+-------+-------+
```

Task 3

Read "test3" dataset:

In [92]:
```
test3_schema = 'Name STRING,Departments STRING,salary DOUBLE'
df_test3 = spark.read.format('csv').schema(test3_schema).option('header','Tr
```

Display dataset

In [93]:
```
1 df_test3.show()
```

```
+---------+------------+-------+
|     Name| Departments| salary|
+---------+------------+-------+
|    Krish|Data Science|10000.0|
|    Krish|         IOT| 5000.0|
|   Mahesh|    Big Data| 4000.0|
|    Krish|    Big Data| 4000.0|
|   Mahesh|Data Science| 3000.0|
|Sudhanshu|Data Science|20000.0|
|Sudhanshu|         IOT|10000.0|
|Sudhanshu|    Big Data| 5000.0|
|    Sunny|Data Science|10000.0|
|    Sunny|    Big Data| 2000.0|
+---------+------------+-------+
```

Display schema

In [94]:
```
1 df_test3.printSchema()
```

```
root
 |-- Name: string (nullable = true)
 |-- Departments: string (nullable = true)
 |-- salary: double (nullable = true)
```

In [95]:
```
1 # register the DataFrame as a temporary view
2 df_test3.createOrReplaceTempView("test3")
```

Group by "Name" column and using sum function on "Name" column

In [96]:
```
1 spark.sql(""" SELECT Name,COUNT(Name)
2               FROM test3
3               GROUP BY Name
4     """).show()
```

```
+---------+-----------+
|     Name|count(Name)|
+---------+-----------+
|Sudhanshu|          3|
|    Sunny|          2|
|    Krish|          3|
|   Mahesh|          2|
+---------+-----------+
```

In [103]:
```
spark.sql(""" SELECT Name,SUM(Salary)
              FROM test3
              GROUP BY Name
    """).show()
```

```
+---------+-----------+
|     Name|sum(Salary)|
+---------+-----------+
|Sudhanshu|    35000.0|
|    Sunny|    12000.0|
|    Krish|    19000.0|
|   Mahesh|     7000.0|
+---------+-----------+
```

Group by "Name" column and using avg function on "Name" column

In [97]:
```
spark.sql(""" SELECT Name,AVG(Name)
              FROM test3
              GROUP BY Name
    """).show()
```

```
+---------+------------------------+
|     Name|avg(CAST(Name AS DOUBLE))|
+---------+------------------------+
|Sudhanshu|                    null|
|    Sunny|                    null|
|    Krish|                    null|
|   Mahesh|                    null|
+---------+------------------------+
```

In [102]:
```
spark.sql(""" SELECT Name,AVG(Salary)
              FROM test3
              GROUP BY Name
    """).show()
```

```
+---------+------------------+
|     Name|       avg(Salary)|
+---------+------------------+
|Sudhanshu|11666.666666666666|
|    Sunny|            6000.0|
|    Krish| 6333.333333333333|
|   Mahesh|            3500.0|
+---------+------------------+
```

Group by "Departments" column and using sum function on "Departments" column

In [99]:
```
spark.sql(""" SELECT Departments,SUM(Departments)
              FROM test3
              GROUP BY Departments
    """).show()
```

```
+------------+-------------------------------+
| Departments|sum(CAST(Departments AS DOUBLE))|
+------------+-------------------------------+
|         IOT|                           null|
|    Big Data|                           null|
|Data Science|                           null|
+------------+-------------------------------+
```

In [104]:
```
spark.sql(""" SELECT Departments,SUM(Salary)
              FROM test3
              GROUP BY Departments
    """).show()
```

```
+------------+-----------+
| Departments|sum(Salary)|
+------------+-----------+
|         IOT|    15000.0|
|    Big Data|    15000.0|
|Data Science|    43000.0|
+------------+-----------+
```

Group by "Departments" column and using mean function on "Departments" column:

In [100]:
```
spark.sql(""" SELECT Departments,AVG(Departments)
              FROM test3
              GROUP BY Departments
    """).show()
```

```
+------------+-------------------------------+
| Departments|avg(CAST(Departments AS DOUBLE))|
+------------+-------------------------------+
|         IOT|                           null|
|    Big Data|                           null|
|Data Science|                           null|
+------------+-------------------------------+
```

In [105]:
```
spark.sql(""" SELECT Departments,AVG(Salary)
              FROM test3
              GROUP BY Departments
    """).show()
```

```
+------------+-----------+
| Departments|avg(Salary)|
+------------+-----------+
|         IOT|     7500.0|
|    Big Data|     3750.0|
|Data Science|    10750.0|
+------------+-----------+
```

Group by "Departments" column and using count function on "Departments" column:

In [101]:
```
spark.sql(""" SELECT Departments,COUNT(Departments)
              FROM test3
              GROUP BY Departments
    """).show()
```

```
+------------+------------------+
| Departments|count(Departments)|
+------------+------------------+
|         IOT|                 2|
|    Big Data|                 4|
|Data Science|                 4|
+------------+------------------+
```

In [106]:
```
spark.sql(""" SELECT Departments,COUNT(Salary)
              FROM test3
              GROUP BY Departments
    """).show()
```

```
+------------+-------------+
| Departments|count(Salary)|
+------------+-------------+
|         IOT|            2|
|    Big Data|            4|
|Data Science|            4|
+------------+-------------+
```

Apply agg to using sum function get the total of salaries

In [109]:
```
spark.sql(""" SELECT Name,SUM(Salary)
              FROM test3
              GROUP BY Name
    """).show()
```

```
+---------+-----------+
|     Name|sum(Salary)|
+---------+-----------+
|Sudhanshu|    35000.0|
|    Sunny|    12000.0|
|    Krish|    19000.0|
|   Mahesh|     7000.0|
+---------+-----------+
```

In [111]:
```
spark.sql(""" SELECT Departments,SUM(Salary)
              FROM test3
              GROUP BY Departments
    """).show()
```

```
+------------+-----------+
| Departments|sum(Salary)|
+------------+-----------+
|         IOT|    15000.0|
|    Big Data|    15000.0|
|Data Science|    43000.0|
+------------+-----------+
```

Task 4

You've been flown to their headquarters in Ulsan, South Korea, to assist them in accurately estimating the number of crew members a ship will need.

They're currently building new ships for certain customers, and they'd like you to create a model and utilize it to estimate how many crew members the ships will require.

Metadata:

1. Measurements of ship size
2. capacity
3. crew
4. age for 158 cruise ships.

It is saved in a csv file for you called "ITI_data.csv". our task is to develop a regression model that will assist in predicting the number of crew members required for future ships. The client also indicated that they have found that particular cruise lines will differ in acceptable crew counts, thus this is most likely an important factor to consider when conducting your investigation.

In [134]:
```
iti_schema = 'Ship_name STRING,Cruise_line STRING,Age DOUBLE,Tonnage DOUBLE,
df_iti = spark.read.format('csv').schema(iti_schema).option('header','True')
```

In [135]:
```
df_iti.printSchema()
```

```
root
 |-- Ship_name: string (nullable = true)
 |-- Cruise_line: string (nullable = true)
 |-- Age: double (nullable = true)
 |-- Tonnage: double (nullable = true)
 |-- passengers: double (nullable = true)
 |-- length: double (nullable = true)
 |-- cabins: double (nullable = true)
 |-- passenger_density: double (nullable = true)
 |-- crew: double (nullable = true)
```

In [136]:
```
1 df_iti.show()
```

```
+----------+----------+----+------------------+----------+------+------+-----
-----------+----+
|  Ship_name|Cruise_line| Age|           Tonnage|passengers|length|cabins|passe
nger_density|crew|
+----------+----------+----+------------------+----------+------+------+-----
-----------+----+
|    Journey|    Azamara| 6.0|30.276999999999997|      6.94|  5.94|  3.55|
42.64|3.55|
|      Quest|    Azamara| 6.0|30.276999999999997|      6.94|  5.94|  3.55|
42.64|3.55|
|Celebration|   Carnival|26.0|            47.262|     14.86|  7.22|  7.43|
31.8| 6.7|
|   Conquest|   Carnival|11.0|             110.0|     29.74|  9.53| 14.88|
36.99|19.1|
|    Destiny|   Carnival|17.0|           101.353|     26.42|  8.92| 13.21|
38.36|10.0|
|    Ecstasy|   Carnival|22.0|            70.367|     20.52|  8.55|  10.2|
34.29| 9.2|
|    Elation|   Carnival|15.0|            70.367|     20.52|  8.55|  10.2|
34.29| 9.2|
|    Fantasy|   Carnival|23.0|            70.367|     20.56|  8.55| 10.22|
34.23| 9.2|
|Fascination|   Carnival|19.0|            70.367|     20.52|  8.55|  10.2|
34.29| 9.2|
|    Freedom|   Carnival| 6.0|110.23899999999999|      37.0|  9.51| 14.87|
29.79|11.5|
|      Glory|   Carnival|10.0|             110.0|     29.74|  9.51| 14.87|
36.99|11.6|
|    Holiday|   Carnival|28.0|            46.052|     14.52|  7.27|  7.26|
31.72| 6.6|
|Imagination|   Carnival|18.0|            70.367|     20.52|  8.55|  10.2|
34.29| 9.2|
|Inspiration|   Carnival|17.0|            70.367|     20.52|  8.55|  10.2|
34.29| 9.2|
|     Legend|   Carnival|11.0|              86.0|     21.24|  9.63| 10.62|
40.49| 9.3|
|   Liberty*|   Carnival| 8.0|             110.0|     29.74|  9.51| 14.87|
36.99|11.6|
|    Miracle|   Carnival| 9.0|              88.5|     21.24|  9.63| 10.62|
41.67|10.3|
|   Paradise|   Carnival|15.0|            70.367|     20.52|  8.55|  10.2|
34.29| 9.2|
|      Pride|   Carnival|12.0|              88.5|     21.24|  9.63| 11.62|
41.67| 9.3|
|  Sensation|   Carnival|20.0|            70.367|     20.52|  8.55|  10.2|
34.29| 9.2|
+----------+----------+----+------------------+----------+------+------+-----
-----------+----+
only showing top 20 rows
```

In [150]:
```
# Splitting Data
train_df , test_df = df_iti.randomSplit([0.8, 0.2] , seed = 42)
print(f"There are {train_df.count()} rows in the training set, and {test_df.
```

```
There are 133 rows in the training set, and 25 in the test set
```

OneHotEncoder

In [137]:
```
from pyspark.ml.feature import StringIndexer, OneHotEncoder
```

In [138]:
```
df_iti.dtypes
```

Out[138]:
```
[('Ship_name', 'string'),
 ('Cruise_line', 'string'),
 ('Age', 'double'),
 ('Tonnage', 'double'),
 ('passengers', 'double'),
 ('length', 'double'),
 ('cabins', 'double'),
 ('passenger_density', 'double'),
 ('crew', 'double')]
```

In [139]:
```
categoricalCols = [col for (col, colType) in df_iti.dtypes
                   if colType == 'string']
print("Categorical Cols:", categoricalCols)

indexOutputCols = [col +'_index' for col in categoricalCols]
print("Index Cols:", indexOutputCols)

oheOutputCols = [col +'_OHE' for col in categoricalCols]
print("OHE Cols:", oheOutputCols)
```

```
Categorical Cols: ['Ship_name', 'Cruise_line']
Index Cols: ['Ship_name_index', 'Cruise_line_index']
OHE Cols: ['Ship_name_OHE', 'Cruise_line_OHE']
```

In [160]:
```
stringIndexer = StringIndexer(inputCols = categoricalCols,
                              outputCols= indexOutputCols,
                             handleInvalid='skip')

oheEncoder = OneHotEncoder(inputCols=indexOutputCols,
                           outputCols=oheOutputCols)
```

In [161]:
```
numricCol = [col for (col, colType) in df_iti.dtypes
             if ((colType=='double')& (col!='crew'))]
print('Numeric Col:',numricCol)
```

```
Numeric Col: ['Age', 'Tonnage', 'passengers', 'length', 'cabins', 'passenger_de
nsity']
```

###Use VectorAssembler to merge all columns into one column:

In [162]:
```
assemeberlInputs = oheOutputCols + numricCol
assemeberlInputs
```

Out[162]:
```
['Ship_name_OHE',
 'Cruise_line_OHE',
 'Age',
 'Tonnage',
 'passengers',
 'length',
 'cabins',
 'passenger_density']
```

In [145]:
```
from pyspark.ml.feature import VectorAssembler
```

In [148]:
```
vecAssembler = VectorAssembler(inputCols=assemeberlInputs, outputCol='featur
```

Create a Linear Regression Model

In [151]:
```
from pyspark.ml.regression import LinearRegression
```

In [152]:
```
lr = LinearRegression(labelCol='crew',featuresCol='features')
```

Creating a Pipeline

In [153]:
```
# Building the pipeline
from pyspark.ml import Pipeline
```

In [164]:
```
pipeline =Pipeline(stages = [stringIndexer,oheEncoder,vecAssembler,lr])
```

Model Evaluation

In [165]:
```
pipelineModel = pipeline.fit(train_df)
predictions = pipelineModel.transform(test_df)
```

```
21/10/26 18:13:00 WARN Instrumentation: [7baa24f4] regParam is zero, which migh
t cause numerical instability and overfitting.
21/10/26 18:13:00 WARN BLAS: Failed to load implementation from: com.github.fom
mil.netlib.NativeSystemBLAS
21/10/26 18:13:00 WARN BLAS: Failed to load implementation from: com.github.fom
mil.netlib.NativeRefBLAS
21/10/26 18:13:00 WARN LAPACK: Failed to load implementation from: com.github.f
ommil.netlib.NativeSystemLAPACK
21/10/26 18:13:00 WARN LAPACK: Failed to load implementation from: com.github.f
ommil.netlib.NativeRefLAPACK
21/10/26 18:13:00 WARN Instrumentation: [7baa24f4] Cholesky solver failed due t
o singular covariance matrix. Retrying with Quasi-Newton solver.
```

In [166]:
```
predictions.select('features','crew','prediction').show(5)
```

```
+--------------------+----+------------------+
|            features|crew|        prediction|
+--------------------+----+------------------+
|(141,[26,119,135,...|12.0| 14.832213385454127|
|(141,[49,118,135,...|8.69| 7.5615145918695985|
|(141,[89,121,135,...| 6.3| 6.2578311275208724|
|(141,[9,134,135,1...|0.88| -2.862148098151032|
|(141,[112,133,135...|1.97|-2.2664609079940385|
+--------------------+----+------------------+
```

In [167]:
```
# Using RMSE
from pyspark.ml.evaluation import RegressionEvaluator
```

In [168]:
```
regressionEvaluator = RegressionEvaluator(predictionCol='prediction',
                                          labelCol='crew',
                                          metricName='rmse')
rmse = regressionEvaluator.evaluate(predictions)
print(f"RMSE is {rmse:.1f}")
```

```
RMSE is 2.9
```

In [170]:
```
# Using R^2
r2 = RegressionEvaluator(predictionCol='prediction',
                                          labelCol='crew',
                                          metricName='r2').evaluate(predictio
print(f"R2 is {r2}")
```

```
R2 is 0.519121561294428
```

By Eng. Mostafa Nabieh If you have questions, please feel free to ask.

My Email : nabieh.mostafa@yahoo.com (mailto:nabieh.mostafa@yahoo.com)

My Whatsapp : +201015197566