Pyspark - Missing Values

-Dropping Columns -Dropping Rows -Various Parameter in Dropping Functionalities -Handling Missing Values by Mean, Median and Mode

In [2]:
```
pip --proxy http://[username]:[password]@noidaproxy.corp.exlservice.com:8000 i
nstall pyspark
```

```
Collecting pyspark
  Downloading https://files.pythonhosted.org/packages/b8/01/b2393cee7f6180d91
50274e92c8bdc1c81220e2ad7554ee5febca1866899/pyspark-3.3.0.tar.gz (281.3MB)
Collecting py4j==0.10.9.5 (from pyspark)
  Using cached https://files.pythonhosted.org/packages/86/ec/60880978512d5569
ca4bf32b3b4d7776a528ecf4bca4523936c98c92a3c8/py4j-0.10.9.5-py2.py3-none-any.w
hl
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py): started
  Building wheel for pyspark (setup.py): still running...
  Building wheel for pyspark (setup.py): finished with status 'done'
  Stored in directory: C:\Users\shrinath195156\AppData\Local\pip\Cache\wheels
\9e\c1\93\d40ec851fc2b278e1056c1353ff95a7a4ef1b219f74ca9c11f
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9.5 pyspark-3.3.0
Note: you may need to restart the kernel to use updated packages.
```

In [1]:
```
import pyspark
```

In [2]:
```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("Practice").getOrCreate()
```

In [3]:
```
spark
```

Out[3]:

SparkSession - in-memory

SparkContext

Spark UI (http://EXLAPLPNyCdxfzp.corp.exlservice.com:4041)

Version

v3.3.0

Master

local[*]

AppName

Practice

In [5]:
```
#reading dataset using spark session
df_pyspark=spark.read.csv("sparktest_1.csv", header=True, inferSchema=True)
df_pyspark.show()
```

```
+---------+----+----------+------+
|     name| age|experience|salary|
+---------+----+----------+------+
|    Krish|  31|        10| 30000|
|Sudhanshu|  30|         8| 25000|
|    Sunny|  29|         4| 20000|
|     Paul|  24|         3| 20000|
|   Harsha|  21|         1| 15000|
|  Shubham|  23|         2| 18000|
|   Mahesh|null|      null| 40000|
|     null|  34|        10| 38000|
|     null|  36|      null|  null|
+---------+----+----------+------+
```

In [7]:
```
#Dropping a column name
df_pyspark.drop("age").show()
```

```
+---------+----------+------+
|     name|experience|salary|
+---------+----------+------+
|    Krish|        10| 30000|
|Sudhanshu|         8| 25000|
|    Sunny|         4| 20000|
|     Paul|         3| 20000|
|   Harsha|         1| 15000|
|  Shubham|         2| 18000|
|   Mahesh|      null| 40000|
|     null|        10| 38000|
|     null|      null|  null|
+---------+----------+------+
```

In [11]:
```
#Dropping all the rows with Null values
df_pyspark.na.drop().show()
```

```
+---------+---+----------+------+
|     name|age|experience|salary|
+---------+---+----------+------+
|    Krish| 31|        10| 30000|
|Sudhanshu| 30|         8| 25000|
|    Sunny| 29|         4| 20000|
|     Paul| 24|         3| 20000|
|   Harsha| 21|         1| 15000|
|  Shubham| 23|         2| 18000|
+---------+---+----------+------+
```

In [12]:
```
#Dropping values using Drop fuctionalities Any (row with any cell value as null) = How (all the cell values in a row are null)
df_pyspark.na.drop(how="all").show()
```

```
+---------+----+----------+------+
|     name| age|experience|salary|
+---------+----+----------+------+
|    Krish|  31|        10| 30000|
|Sudhanshu|  30|         8| 25000|
|    Sunny|  29|         4| 20000|
|     Paul|  24|         3| 20000|
|   Harsha|  21|         1| 15000|
|  Shubham|  23|         2| 18000|
|   Mahesh|null|      null| 40000|
|     null|  34|        10| 38000|
|     null|  36|      null|  null|
+---------+----+----------+------+
```

In [13]:
```
#Dropping values using Threshold count for non-null values meaning atleast value input in thresh should be non-null in a row
#here threshold = 2 so cells with atleast 2 non-null values in a row
df_pyspark.na.drop(how="any",thresh=2).show()
```

```
+---------+----+----------+------+
|     name| age|experience|salary|
+---------+----+----------+------+
|    Krish|  31|        10| 30000|
|Sudhanshu|  30|         8| 25000|
|    Sunny|  29|         4| 20000|
|     Paul|  24|         3| 20000|
|   Harsha|  21|         1| 15000|
|  Shubham|  23|         2| 18000|
|   Mahesh|null|      null| 40000|
|     null|  34|        10| 38000|
+---------+----+----------+------+
```

In [16]:
```
#Dropping Null values using Subset - dropping null values only from a specific column
df_pyspark.na.drop(how="any",subset=["experience"]).show()
```

```
+---------+---+----------+------+
|     name|age|experience|salary|
+---------+---+----------+------+
|    Krish| 31|        10| 30000|
|Sudhanshu| 30|         8| 25000|
|    Sunny| 29|         4| 20000|
|     Paul| 24|         3| 20000|
|   Harsha| 21|         1| 15000|
|  Shubham| 23|         2| 18000|
|     null| 34|        10| 38000|
+---------+---+----------+------+
```

Filling the missing values not working for all the columns

df_pyspark.na.fill("Missing_Values").show()

In [34]:
```
#Filling the missing values
df_pyspark.na.fill("Missing Values").show()
```

```
+--------------+----+----------+------+
|          name| age|experience|salary|
+--------------+----+----------+------+
|         Krish|  31|        10| 30000|
|     Sudhanshu|  30|         8| 25000|
|         Sunny|  29|         4| 20000|
|          Paul|  24|         3| 20000|
|        Harsha|  21|         1| 15000|
|       Shubham|  23|         2| 18000|
|        Mahesh|null|      null| 40000|
|Missing Values|  34|        10| 38000|
|Missing Values|  36|      null|  null|
+--------------+----+----------+------+
```

In [35]:
```
#####Filling the missing values for subset/specified column ****ERROR IN THE R
ESULT TO DISCUSS****
df_pyspark.na.fill("Missing Values",["age","experience","salary"]).show()
```

```
+---------+----+----------+------+
|     name| age|experience|salary|
+---------+----+----------+------+
|    Krish|  31|        10| 30000|
|Sudhanshu|  30|         8| 25000|
|    Sunny|  29|         4| 20000|
|     Paul|  24|         3| 20000|
|   Harsha|  21|         1| 15000|
|  Shubham|  23|         2| 18000|
|   Mahesh|null|      null| 40000|
|     null|  34|        10| 38000|
|     null|  36|      null|  null|
+---------+----+----------+------+
```

In [37]:
```
#Using Imputer to fill MEAN for the null values
from pyspark.ml.feature import Imputer

imputer=Imputer(
    inputCols=["age","experience","salary"],
    outputCols=["{}_imputed".format(c) for c in ["age","experience","salary"]]
    ).setStrategy("mean")
```

In [38]:
```
#Added Imputation cols to df
imputer.fit(df_pyspark).transform(df_pyspark).show()
```

```
+---------+----+----------+------+-----------+-----------------+------------
--+
|     name| age|experience|salary|age_imputed|experience_imputed|salary_imput
ed|
+---------+----+----------+------+-----------+-----------------+------------
--+
|    Krish|  31|        10| 30000|         31|                10|         300
00|
|Sudhanshu|  30|         8| 25000|         30|                 8|         250
00|
|    Sunny|  29|         4| 20000|         29|                 4|         200
00|
|     Paul|  24|         3| 20000|         24|                 3|         200
00|
|   Harsha|  21|         1| 15000|         21|                 1|         150
00|
|  Shubham|  23|         2| 18000|         23|                 2|         180
00|
|   Mahesh|null|      null| 40000|         28|                 5|         400
00|
|     null|  34|        10| 38000|         34|                10|         380
00|
|     null|  36|      null|  null|         36|                 5|         257
50|
+---------+----+----------+------+-----------+-----------------+------------
--+
```

In [43]:
```
#Using Imputer to fill MEDIAN for the null values
from pyspark.ml.feature import Imputer

imputer=Imputer(
    inputCols=["age","experience","salary"],
    outputCols=["{}_imputed".format(c) for c in ["age","experience","salary"]]
    ).setStrategy("median")
```

In [44]:
```
#Added Imputation cols to df
imputer.fit(df_pyspark).transform(df_pyspark).show()
```

```
+---------+----+----------+------+-----------+-----------------+------------
--+
|     name| age|experience|salary|age_imputed|experience_imputed|salary_imput
ed|
+---------+----+----------+------+-----------+-----------------+------------
--+
|    Krish|  31|        10| 30000|         31|                10|         300
00|
|Sudhanshu|  30|         8| 25000|         30|                 8|         250
00|
|    Sunny|  29|         4| 20000|         29|                 4|         200
00|
|     Paul|  24|         3| 20000|         24|                 3|         200
00|
|   Harsha|  21|         1| 15000|         21|                 1|         150
00|
|  Shubham|  23|         2| 18000|         23|                 2|         180
00|
|   Mahesh|null|      null| 40000|         29|                 4|         400
00|
|     null|  34|        10| 38000|         34|                10|         380
00|
|     null|  36|      null|  null|         36|                 4|         200
00|
+---------+----+----------+------+-----------+-----------------+------------
--+
```

In [7]:
```
#Checking the schema
df_pyspark = spark.read.option("header","true").csv("sparktest.csv")
df_pyspark
```

Out[7]:
```
DataFrame[name: string, age: string, experience: string]
```

In [8]:
```
#Checking the schema using inferschema for the non string values e.g. age in a
bove result
df_pyspark = spark.read.option("header","true").csv("sparktest.csv",inferSchem
a=True)
df_pyspark
```

Out[8]:
```
DataFrame[name: string, age: int, experience: int]
```

In [9]:
```
df_pyspark.printSchema()
```

```
root
 |-- name: string (nullable = true)
 |-- age: integer (nullable = true)
 |-- experience: integer (nullable = true)
```

In [63]:
```
#Reading dataset using read
df_pyspark = spark.read.csv("sparktest.csv", header=True, inferSchema=True)
df_pyspark.show()
```

```
+---------+---+----------+
|     name|age|experience|
+---------+---+----------+
|    Krish| 31|         5|
|Sudhanshu| 30|         8|
|    Sunny| 29|         6|
+---------+---+----------+
```

In [64]:
```
df_pyspark.printSchema()
```

```
root
 |-- name: string (nullable = true)
 |-- age: integer (nullable = true)
 |-- experience: integer (nullable = true)
```

In [14]:
```
type(df_pyspark)
```

Out[14]:
```
pyspark.sql.dataframe.DataFrame
```

In [15]:
```
#getting columns
df_pyspark.columns
```

Out[15]:
```
['name', 'age', 'experience']
```

In [16]:
```
df_pyspark.head(3)
```

Out[16]:
```
[Row(name='Krish', age=31, experience=5),
 Row(name='Sudhanshu', age=30, experience=8),
 Row(name='Sunny', age=29, experience=6)]
```

In [17]:
```
#display dataframe
df_pyspark.show()
```

```
+---------+---+----------+
|     name|age|experience|
+---------+---+----------+
|    Krish| 31|         5|
|Sudhanshu| 30|         8|
|    Sunny| 29|         6|
+---------+---+----------+
```

In [20]:
```
#selecting data from only one column
df_pyspark.select("name").show()
```

```
+---------+
|     name|
+---------+
|    Krish|
|Sudhanshu|
|    Sunny|
+---------+
```

In [21]:
```
type(df_pyspark.select("name"))
```

Out[21]:
```
pyspark.sql.dataframe.DataFrame
```

In [22]:
```
#selecting data from multiple columns
df_pyspark.select(["name","experience"]).show()
```

```
+---------+----------+
|     name|experience|
+---------+----------+
|    Krish|         5|
|Sudhanshu|         8|
|    Sunny|         6|
+---------+----------+
```

In [24]:
```
#another way to select a column name
df_pyspark["name"]
```

Out[24]:
```
Column<'name'>
```

In [27]:
```
df_pyspark.dtypes
```

Out[27]:
```
[('name', 'string'), ('age', 'int'), ('experience', 'int')]
```

In [29]:
```
#Getting dataframe statistics using describe
df_pyspark.describe().show()
```

```
+-------+-----+----+------------------+
|summary| name| age|        experience|
+-------+-----+----+------------------+
|  count|    3|   3|                 3|
|   mean| null|30.0| 6.333333333333333|
| stddev| null| 1.0|1.5275252316519468|
|    min|Krish|  29|                 5|
|    max|Sunny|  31|                 8|
+-------+-----+----+------------------+
```

In [69]:
```
#addition of columns in dataframe using calculated column here
df_pyspark=df_pyspark.withColumn("experience after 2yrs",df_pyspark["experienc
e"]+2)
```

In [73]:
```
df_pyspark.show()
```

```
+---------+---+----------+--------------------+
|     name|age|experience|experience after 2yrs|
+---------+---+----------+--------------------+
|    Krish| 31|         5|                   7|
|Sudhanshu| 30|         8|                  10|
|    Sunny| 29|         6|                   8|
+---------+---+----------+--------------------+
```

In [75]:
```
#dropping column from the dataframe have to use variable to pass the change
df_pyspark=df_pyspark.drop("experience after 2yrs")
```

In [76]:
```
df_pyspark.show()
```

```
+---------+---+----------+
|     name|age|experience|
+---------+---+----------+
|    Krish| 31|         5|
|Sudhanshu| 30|         8|
|    Sunny| 29|         6|
+---------+---+----------+
```

In [77]:
```
#renaming one column using with function
df_pyspark.withColumnRenamed("name", "First_Name").show()
```

```
+----------+---+----------+
|First_Name|age|experience|
+----------+---+----------+
|     Krish| 31|         5|
| Sudhanshu| 30|         8|
|     Sunny| 29|         6|
+----------+---+----------+
```

In [97]:
```
#renaming multiple required columns using with function
(df_pyspark.withColumnRenamed("name", "first_name")
 .withColumnRenamed("experience","total_exp")).show()
```

```
+----------+---+---------+
|first_name|age|total_exp|
+----------+---+---------+
|     Krish| 31|        5|
| Sudhanshu| 30|        8|
|     Sunny| 29|        6|
+----------+---+---------+
```

In [96]:
```
#another way of renaming all column names
refined_column_name_list = ["first_name","age","total_exp"]
df_pyspark1=df_pyspark.toDF(*refined_column_name_list).show()
```

```
+----------+---+---------+
|first_name|age|total_exp|
+----------+---+---------+
|     Krish| 31|        5|
| Sudhanshu| 30|        8|
|     Sunny| 29|        6|
+----------+---+---------+
```