**Name:** V Venkata Sri Prasad

**Batch:** Data Engineering

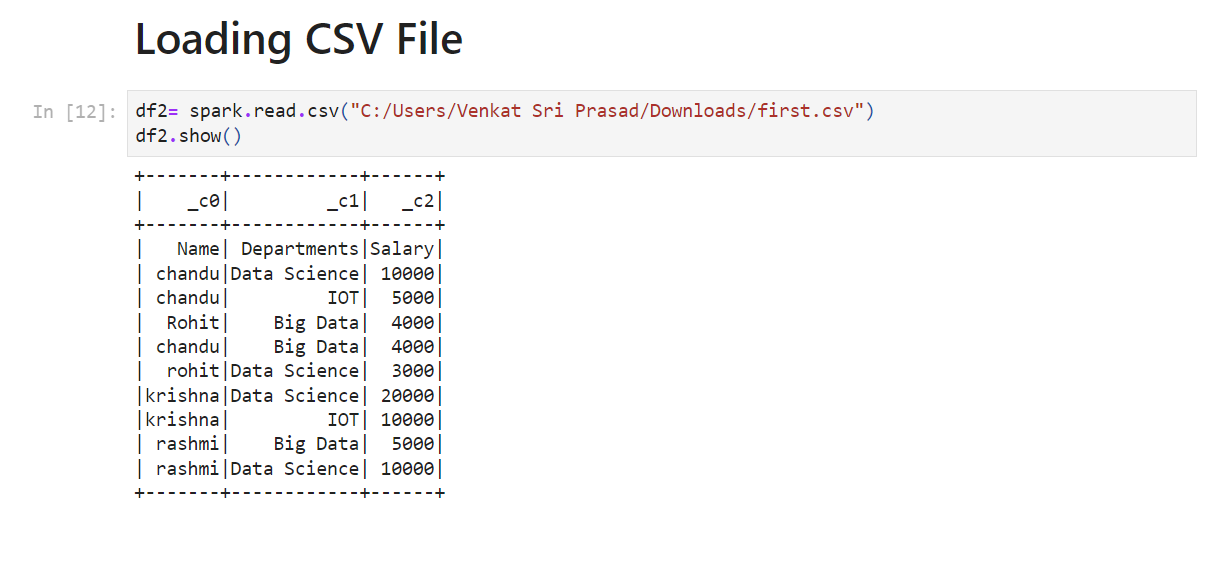
**Date:** 9/02/2024 – (Day 16)

Topic:

Manipulating, Droping, Sorting, Aggregations, Joining, GroupeBy DataFrames

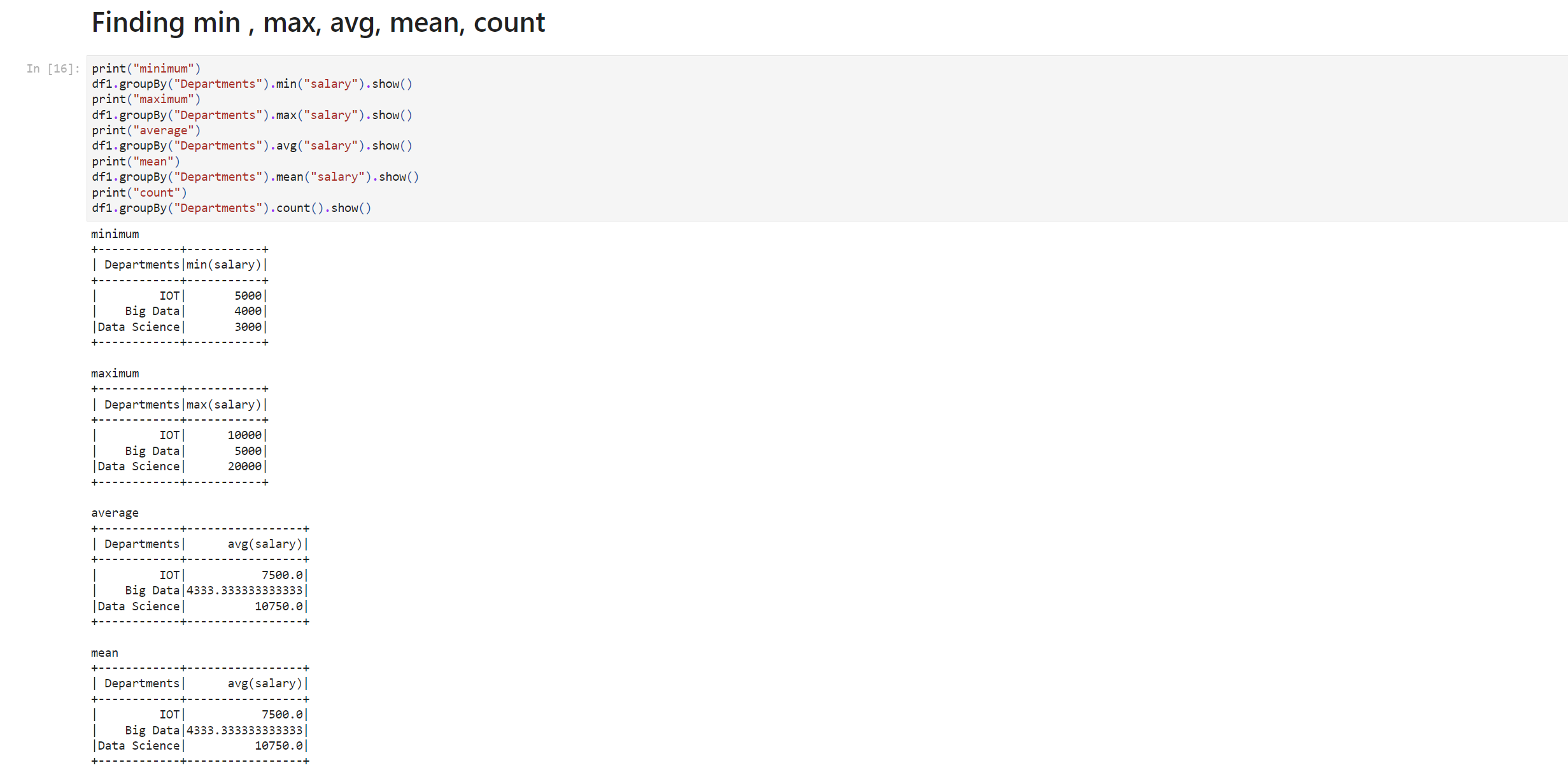
GroupBy and Aggregate function:

Similar to SQL GROUP BY clause, PySpark groupBy() function is used to collect the identical data into groups on DataFrame and perform count, sum, avg, min, and max functions on the grouped data.



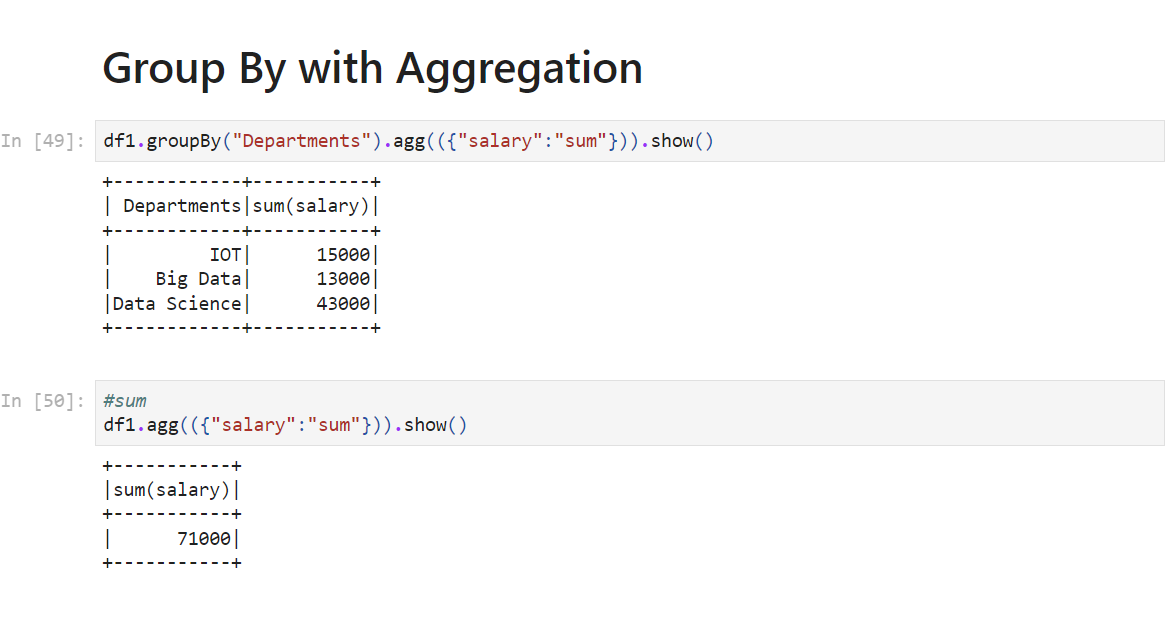


Perform min, max, mean, avg, and count using the groupBy function.

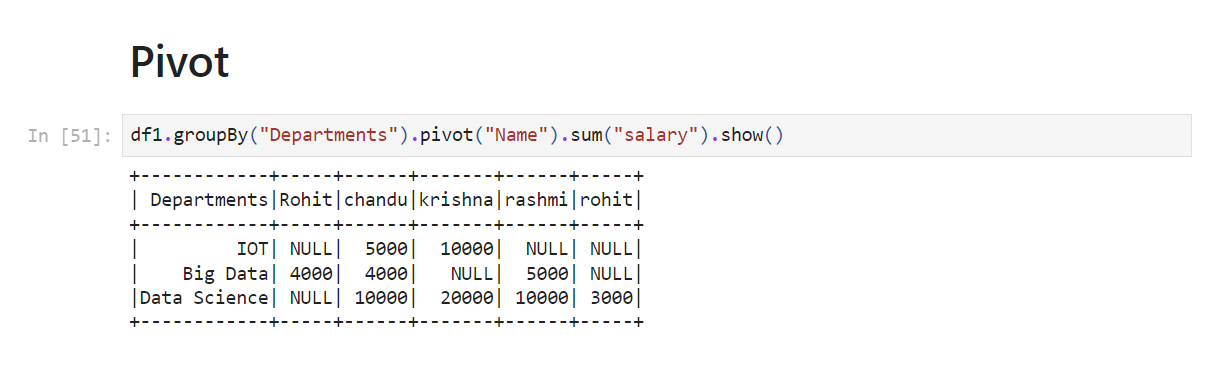




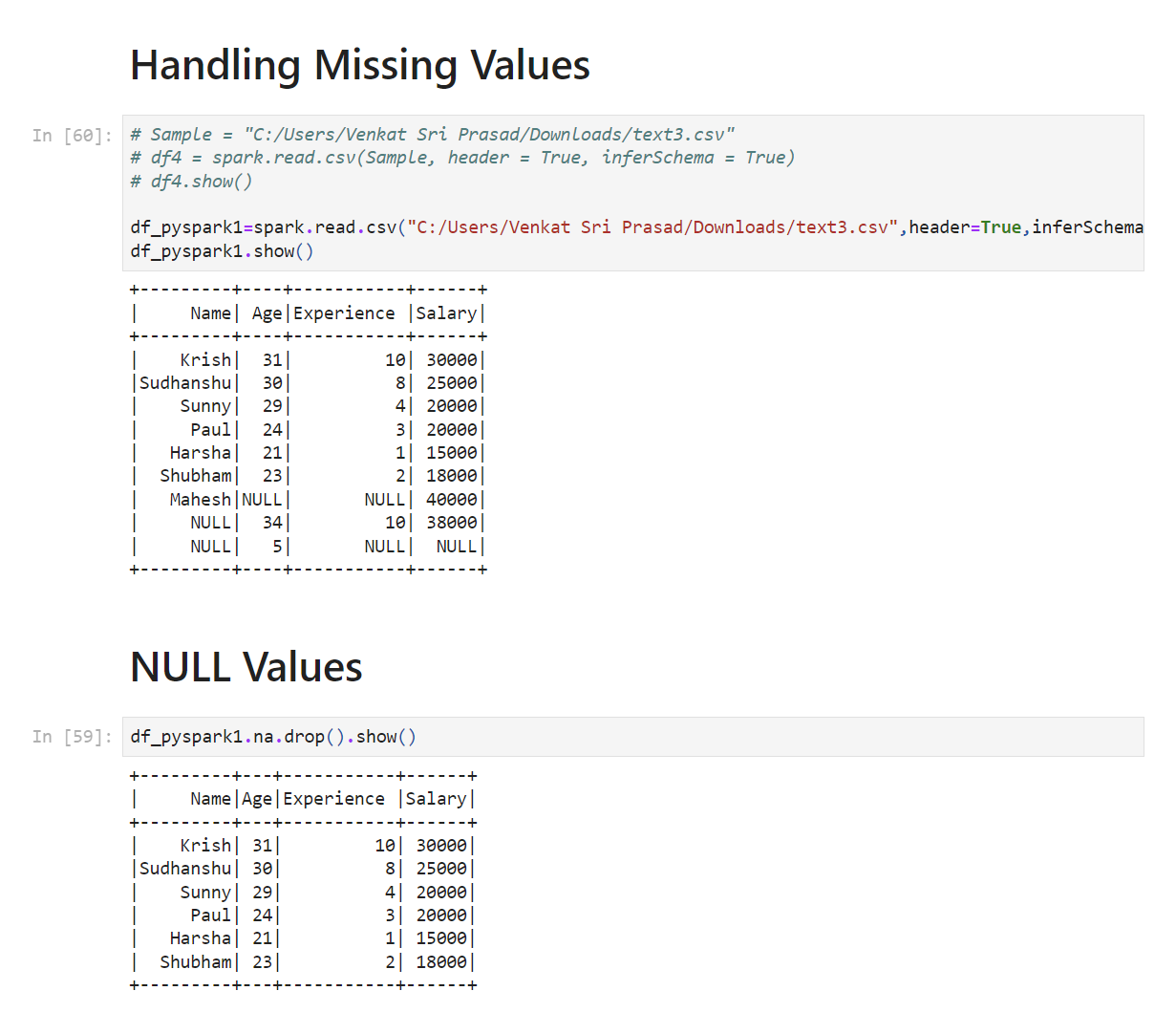
We can also perform agg() function on entire DataFrame without groupBy()



Using Pivot/ UnPivot — Spark SQL provides pivot() function to rotate the data from one column into multiple columns (transpose row to column). It is an aggregation where one of the grouping column values is transposed into individual columns with distinct data. Similary UnPivot can be used.



Handling Missing Values Pyspark



drop() has the following parameters — how, thresh, and subset

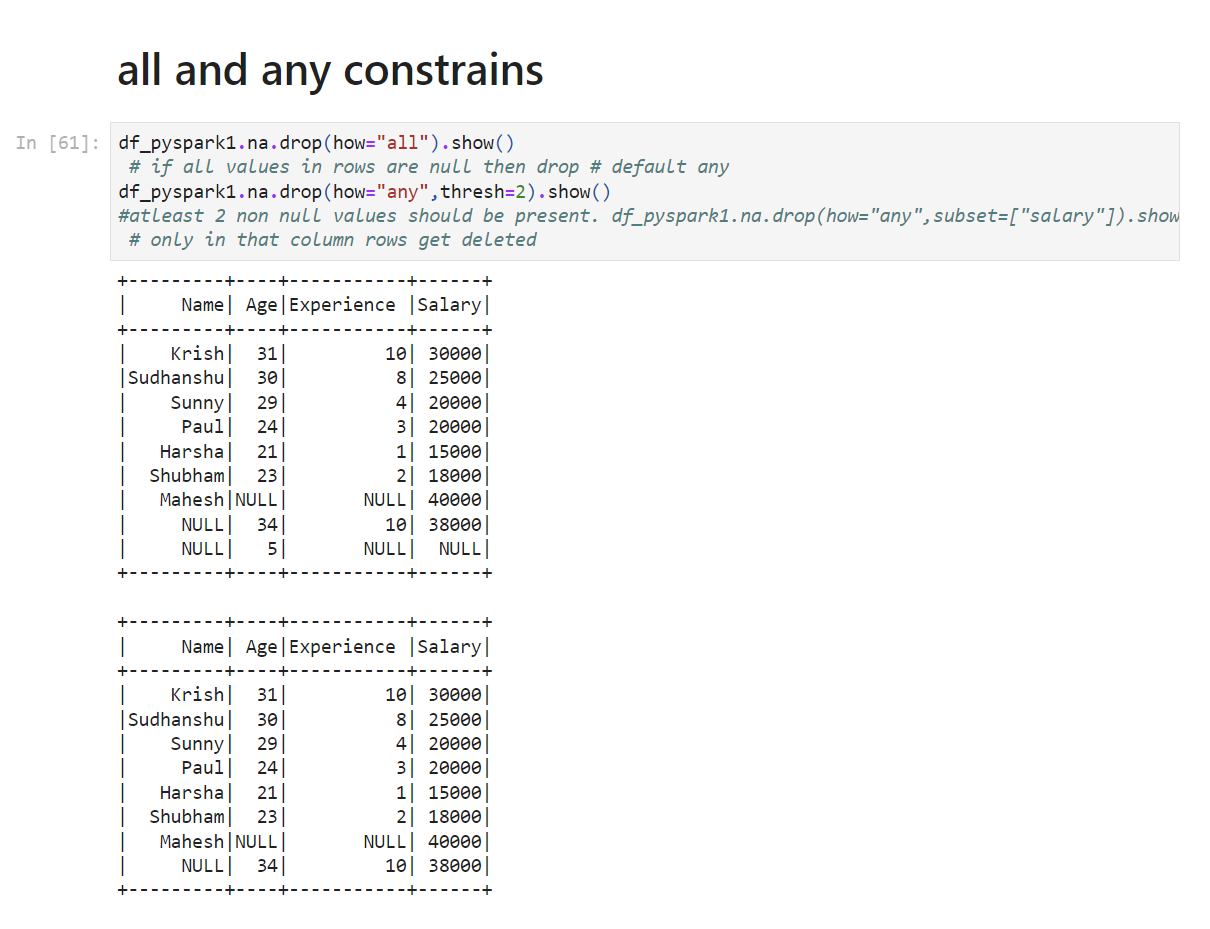
1. df\_pyspark1.na.drop(how="all").show()

 # if all values in rows are null then drop # default any

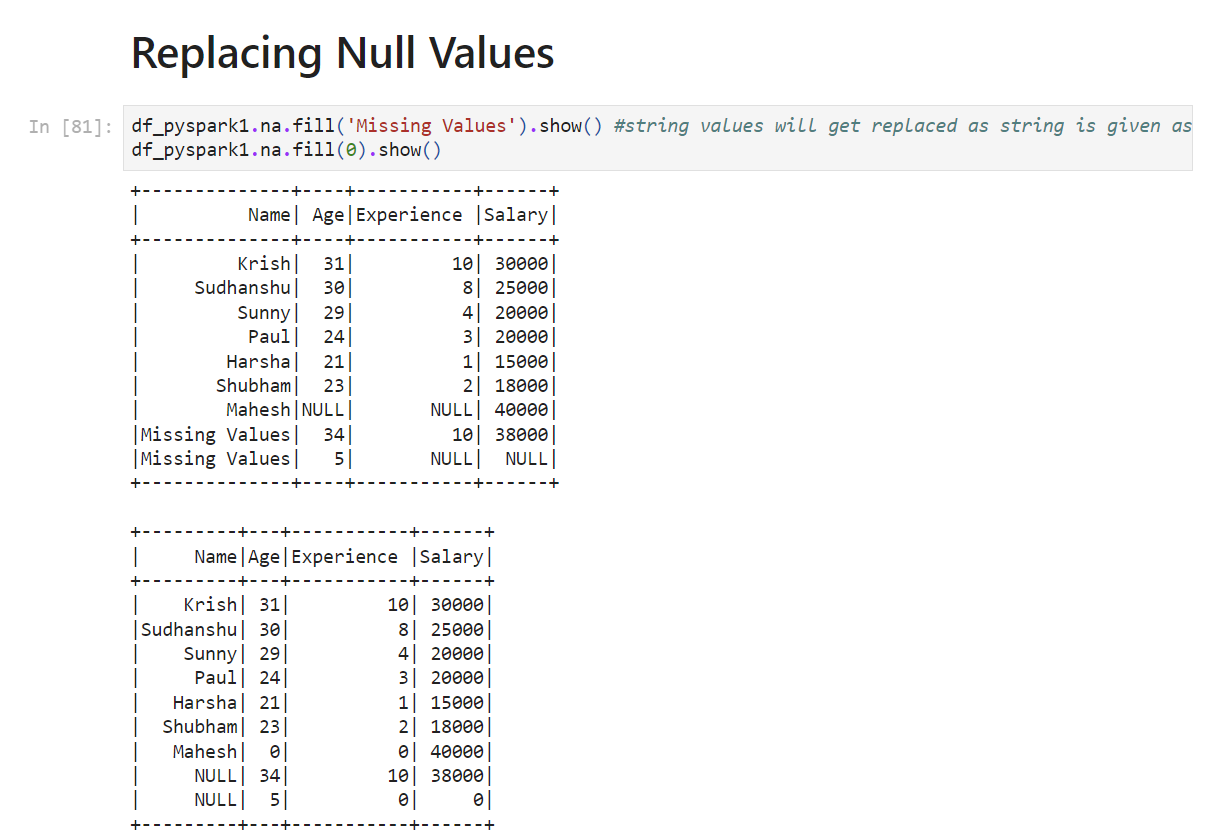
1. df\_pyspark1.na.drop(how="any",thresh=2).show()

#atleast 2 non null values should be present. df\_pyspark1.na.drop(how="any",subset=["salary"]).show()

 # only in that column rows get deleted

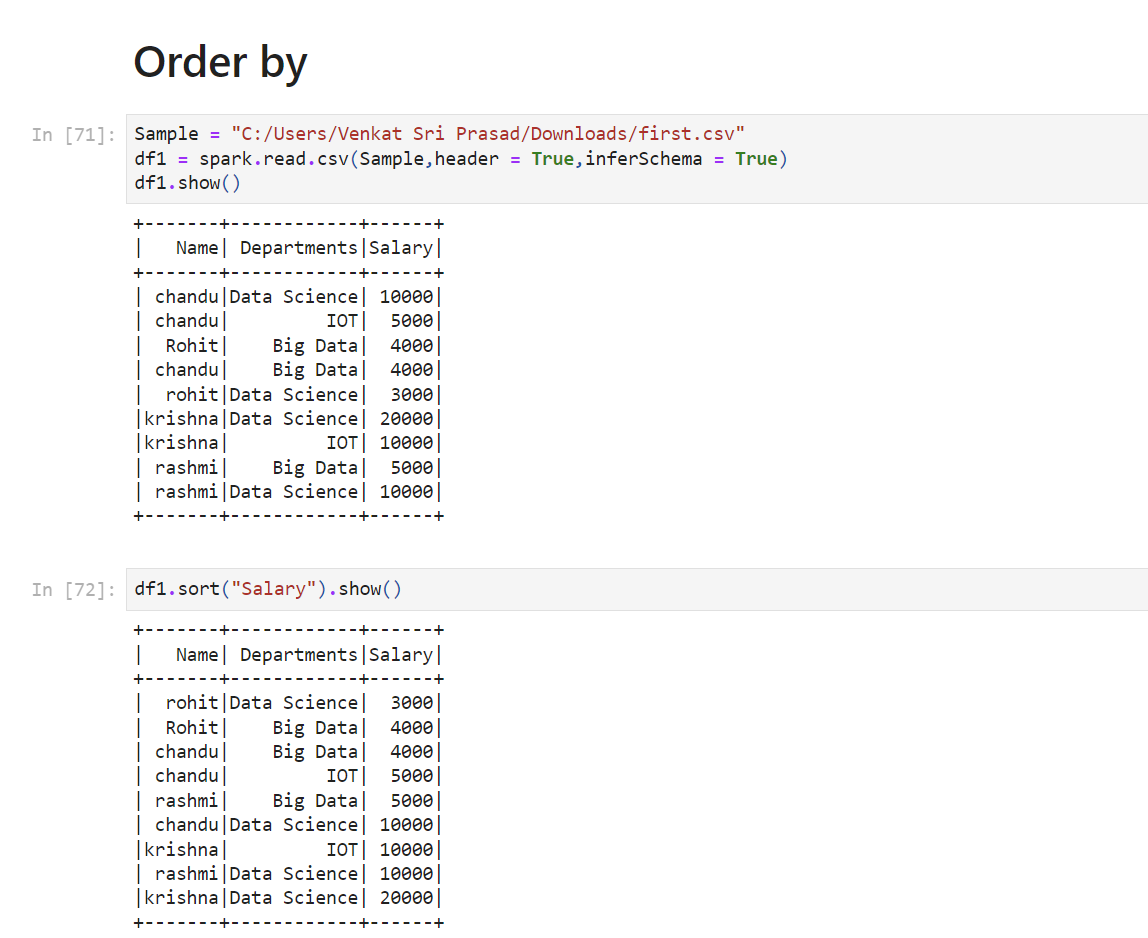


Filling missing values — Single Value



orderBy() and sort() in Pyspark DataFrame

We will be switching back to previous CSV file used



*sort()* — To sort a dataframe by using one or more columns, Default — ascending order

df\_pyspark.sort("salary").show() # Sort based on single column

df\_pyspark.sort(df\_pyspark["salary"].desc()).show() # sort based on descending order

df\_pyspark.sort("salary","Name").show() # Sort based on first column then second column

