Spark DataFrame

In [1]:

```
import findspark
findspark.init('/home/jubinsoni/spark-2.1.0-bin-hadoop2.7')
from pyspark.sql import SparkSession
```

In [2]:

```
spark = SparkSession.builder.appName('Basics').getOrCreate()
```

In [3]:

```
df = spark.read.json('people.json')
```

In [4]:

```
df.show()
```

```
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

In [5]:

```
df.head()
```

Out[5]:

```
Row(age=None, name='Michael')
```

In [7]:

```
df.printSchema()
```

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

In [8]:

```
df.columns
```

Out[8]:

```
['age', 'name']
```

In [10]:

```
df.describe()
```

Out[10]:

```
DataFrame[summary: string, age: string, name: string]
```

In [15]:

```
df.describe().show()
```

```
+-------+------------------+-------+
|summary|               age|   name|
+-------+------------------+-------+
|  count|                 2|      3|
|   mean|              24.5|   null|
| stddev|7.778174593052023|   null|
|    min|                19|   Andy|
|    max|                30|Michael|
+-------+------------------+-------+
```

In [16]:

```
from pyspark.sql.types import (StructField, StringType,
                               IntegerType, StructType)
```

In [17]:

```
#age: name of variable, type of variable, True/False: if it can contain null
data_schema = [StructField('age', IntegerType(), True),
               StructField('name', StringType(), True)]
```

In [18]:

```
data_schema
```

Out[18]:

```
[StructField(age,IntegerType,true), StructField(name,StringType,true
)]
```

In [19]:

```
final_struc = StructType(fields=data_schema)
```

In [20]:

```
final_struc
```

Out[20]:

```
StructType(List(StructField(age,IntegerType,true),StructField(name,S
tringType,true)))
```

In [21]:

```
df = spark.read.json('people.json', schema=final_struc)
```

In [24]:

```
df.printSchema()
#Now it has corrected the age as integer instead of long
```

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

In [25]:

```
df.show()
```

```
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

In [26]:

```
df['age']
```

Out[26]:

```
Column<b'age'>
```

In [27]:

```
type(df['age'])
```

Out[27]:

```
pyspark.sql.column.Column
```

In [32]:

```
#If you want to select a dataframe column
df.select('age')
```

Out[32]:

```
DataFrame[age: int]
```

In [33]:

```
#If you want to view the results of dataframe column
df.select('age').show()
```

```
+----+
| age|
+----+
|null|
|  30|
|  19|
+----+
```

In [34]:

```
#If you want to view selected rows of dataframe
df.head(2)
```

Out[34]:

```
[Row(age=None, name='Michael'), Row(age=30, name='Andy')]
```

In [37]:

```
df.head(2)[0]
```

Out[37]:

```
Row(age=None, name='Michael')
```

In [38]:

```
df.head(2)[0]['name']
```

Out[38]:

```
'Michael'
```

In [39]:

```
type(df.head(2)[0])
```

Out[39]:

```
pyspark.sql.types.Row
```

In [40]:

```
type(df.head(2)[0]['name'])
```

Out[40]:

```
str
```

In [41]:

```
##The reason there are so many datatypes and objects in Spark because of its
#to read from a distributed source and map that out to compute
```

In [44]:

```
#selecting multiple columns
df.select(['age', 'name'])
```

Out[44]:

```
DataFrame[age: int, name: string]
```

In [43]:

```
df.select(['age', 'name']).show()
```

```
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

In [46]:

```
#Creating new columns
df.withColumn('newAge', df['age']).show()
```

```
+----+-------+------+
| age|   name|newAge|
+----+-------+------+
|null|Michael|  null|
|  30|   Andy|    30|
|  19| Justin|    19|
+----+-------+------+
```

In [47]:

```
#Creating new columns
df.withColumn('doubleAge', df['age']*2).show()
```

```
+----+-------+---------+
| age|   name|doubleAge|
+----+-------+---------+
|null|Michael|     null|
|  30|   Andy|       60|
|  19| Justin|       38|
+----+-------+---------+
```

In [48]:

```
#The above operations are not inplace so everytime you make
#any changes you have to save them to a new variable
```

In [49]:

```
#Renaming a column
df.withColumnRenamed('age', 'my_new_age').show()
```

```
+----------+-------+
|my_new_age|   name|
+----------+-------+
|      null|Michael|
|        30|   Andy|
|        19| Justin|
+----------+-------+
```

Working with Spark SQL queries

In [50]:

```
#In order to work with SQL queries we need to register spark
#As a temporary SQL view
#We do CreateOrReplace incase we have already existing view it replaces

df.createOrReplaceTempView('people')
```

In [51]:

```
#Now you can pass direct SQL queries
results = spark.sql("SELECT * FROM people")
```

In [52]:

```
results.show()
```

```
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

In [54]:

```
age_results = spark.sql("SELECT * FROM people WHERE age=30")
```

In [56]:

```
age_results.show()
```

```
+---+----+
|age|name|
+---+----+
| 30|Andy|
+---+----+
```

Working with JSON, CSV and other formats

In [57]:

```
from pyspark.sql import SparkSession
```

In [58]:

```
spark = SparkSession.builder.appName('ops').getOrCreate()
```

In [59]:

```
#infer schema option is available with CSV

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

In [60]:

```
df.printSchema()
```

```
root
 |-- Date: timestamp (nullable = true)
 |-- Open: double (nullable = true)
 |-- High: double (nullable = true)
 |-- Low: double (nullable = true)
 |-- Close: double (nullable = true)
 |-- Volume: integer (nullable = true)
 |-- Adj Close: double (nullable = true)
```

In [61]:

```
1  df.head(3)
```

Out[61]:

```
[Row(Date=datetime.datetime(2010, 1, 4, 0, 0), Open=213.429998, High
=214.499996, Low=212.38000099999996, Close=214.009998, Volume=123432
400, Adj Close=27.727039),
 Row(Date=datetime.datetime(2010, 1, 5, 0, 0), Open=214.599998, High
=215.589994, Low=213.249994, Close=214.379993, Volume=150476200, Adj
Close=27.774976000000002),
 Row(Date=datetime.datetime(2010, 1, 6, 0, 0), Open=214.379993, High
=215.23, Low=210.750004, Close=210.969995, Volume=138040000, Adj Clo
se=27.333178000000004)]
```

In [64]:

```
1  %%time
2  df.head(1)[0]
```

```
CPU times: user 0 ns, sys: 16 ms, total: 16 ms
Wall time: 180 ms
```

Out[64]:

```
Row(Date=datetime.datetime(2010, 1, 4, 0, 0), Open=213.429998, High=
214.499996, Low=212.38000099999996, Close=214.009998, Volume=1234324
00, Adj Close=27.727039)
```

In [65]:

```
1 df.show()
```

```
+-------------------+------------------+------------------+--------
---------+------------------+---------+------------------+
|               Date|              Open|              High|
Low|             Close|   Volume|         Adj Close|
+-------------------+------------------+------------------+--------
---------+------------------+---------+------------------+
|2010-01-04 00:00:...|        213.429998|        214.499996|212.3800
0099999996|        214.009998|123432400|         27.727039|
|2010-01-05 00:00:...|        214.599998|        215.589994|
213.249994|        214.379993|150476200|27.774976000000002|
|2010-01-06 00:00:...|        214.379993|            215.23|
210.750004|        210.969995|138040000|27.333178000000004|
|2010-01-07 00:00:...|            211.75|        212.000006|
209.050005|            210.58|119282800|          27.28265|
|2010-01-08 00:00:...|        210.299994|        212.000006|209.0600
0500000002|211.98000499999998|111902700|         27.464034|
|2010-01-11 00:00:...|212.79999700000002|        213.000002|
208.450005|210.11000299999998|115557400|         27.221758|
|2010-01-12 00:00:...|209.18999499999998|209.76999500000002|
206.419998|        207.720001|148614900|          26.91211|
|2010-01-13 00:00:...|        207.870005|210.92999500000002|
204.099998|        210.650002|151473000|          27.29172|
|2010-01-14 00:00:...|210.11000299999998|210.45999700000002|
209.020004|            209.43|108223500|         27.133657|
|2010-01-15 00:00:...|210.92999500000002|211.59999700000003|
205.869999|            205.93|148516900|26.680197999999997|
|2010-01-19 00:00:...|        208.330002|215.18999900000003|
207.240004|        215.039995|182501900|27.860484999999997|
|2010-01-20 00:00:...|        214.910006|        215.549994|
209.500002|            211.73|153038200|         27.431644|
|2010-01-21 00:00:...|        212.079994|213.30999599999998|
207.210003|        208.069996|152038600|         26.957455|
|2010-01-22 00:00:...|206.78000600000001|        207.499996|
197.16|            197.75|220441900|         25.620401|
|2010-01-25 00:00:...|202.51000200000001|        204.699999|
200.190002|        203.070002|266424900|26.309658000000002|
|2010-01-26 00:00:...|205.95000100000001|        213.710005|
202.580004|        205.940001|466777500|         26.681494|
|2010-01-27 00:00:...|        206.849995|            210.58|
199.530001|        207.880005|430642100|26.932840000000002|
|2010-01-28 00:00:...|        204.930004|        205.500004|
198.699995|        199.289995|293375600|25.819922000000002|
|2010-01-29 00:00:...|        201.079996|        202.199995|
190.250002|        192.060003|311488100|         24.883208|
|2010-02-01 00:00:...|192.36999699999998|             196.0|191.2999
9899999999|        194.729998|187469100|         25.229131|
+-------------------+------------------+------------------+--------
---------+------------------+---------+------------------+
only showing top 20 rows
```

In [69]:

```
#You can pass in conditions like
df.filter('Close < 500').show()
```

```
+------------------+
|              Open|
+------------------+
|        213.429998|
|        214.599998|
|        214.379993|
|            211.75|
|        210.299994|
|212.79999700000002|
|209.18999499999998|
|        207.870005|
|210.11000299999998|
|210.92999500000002|
|        208.330002|
|        214.910006|
|        212.079994|
|206.78000600000001|
|202.51000200000001|
|205.95000100000001|
|        206.849995|
|        204.930004|
|        201.079996|
|192.36999699999998|
+------------------+
only showing top 20 rows
```

In [70]:

```
#Opening price of every stock where Close price is less than 500
df.filter('Close < 500').select('Open').show()
```

```
+------------------+
|              Open|
+------------------+
|        213.429998|
|        214.599998|
|        214.379993|
|            211.75|
|        210.299994|
|212.79999700000002|
|209.18999499999998|
|        207.870005|
|210.11000299999998|
|210.92999500000002|
|        208.330002|
|        214.910006|
|        212.079994|
|206.78000600000001|
|202.51000200000001|
|205.95000100000001|
|        206.849995|
|        204.930004|
|        201.079996|
|192.36999699999998|
+------------------+
only showing top 20 rows
```

In [71]:

```
df.filter('Close < 500').select(['Open', 'Close']).show()
```

```
+------------------+------------------+
|              Open|             Close|
+------------------+------------------+
|         213.429998|         214.009998|
|         214.599998|         214.379993|
|         214.379993|         210.969995|
|             211.75|             210.58|
|         210.299994|211.98000499999998|
|212.79999700000002|210.11000299999998|
|209.18999499999998|         207.720001|
|         207.870005|         210.650002|
|210.11000299999998|             209.43|
|210.92999500000002|             205.93|
|         208.330002|         215.039995|
|         214.910006|             211.73|
|         212.079994|         208.069996|
|206.78000600000001|             197.75|
|202.51000200000001|         203.070002|
|205.95000100000001|         205.940001|
|         206.849995|         207.880005|
|         204.930004|         199.289995|
|         201.079996|         192.060003|
|192.36999699999998|         194.729998|
+------------------+------------------+
only showing top 20 rows
```

In [73]:

```
1 df.filter(df['Close'] < 500).show()
```

```
+--------------------+------------------+------------------+--------
---------+------------------+---------+------------------+
|                Date|              Open|              High|
Low|             Close|   Volume|         Adj Close|
+--------------------+------------------+------------------+--------
---------+------------------+---------+------------------+
|2010-01-04 00:00:...|        213.429998|        214.499996|212.3800
0099999996|        214.009998|123432400|         27.727039|
|2010-01-05 00:00:...|        214.599998|        215.589994|
213.249994|        214.379993|150476200|27.774976000000002|
|2010-01-06 00:00:...|        214.379993|            215.23|
210.750004|        210.969995|138040000|27.333178000000004|
|2010-01-07 00:00:...|            211.75|        212.000006|
209.050005|            210.58|119282800|          27.28265|
|2010-01-08 00:00:...|        210.299994|        212.000006|209.0600
0500000002|211.98000499999998|111902700|         27.464034|
|2010-01-11 00:00:...|212.79999700000002|        213.000002|
208.450005|210.11000299999998|115557400|         27.221758|
|2010-01-12 00:00:...|209.18999499999998|209.76999500000002|
206.419998|        207.720001|148614900|          26.91211|
|2010-01-13 00:00:...|        207.870005|210.92999500000002|
204.099998|        210.650002|151473000|          27.29172|
|2010-01-14 00:00:...|210.11000299999998|210.45999700000002|
209.020004|            209.43|108223500|         27.133657|
|2010-01-15 00:00:...|210.92999500000002|211.59999700000003|
205.869999|            205.93|148516900|26.680197999999997|
|2010-01-19 00:00:...|        208.330002|215.18999900000003|
207.240004|        215.039995|182501900|27.860484999999997|
|2010-01-20 00:00:...|        214.910006|        215.549994|
209.500002|            211.73|153038200|         27.431644|
|2010-01-21 00:00:...|        212.079994|213.30999599999998|
207.210003|        208.069996|152038600|         26.957455|
|2010-01-22 00:00:...|206.78000600000001|        207.499996|
197.16|            197.75|220441900|         25.620401|
|2010-01-25 00:00:...|202.51000200000001|        204.699999|
200.190002|        203.070002|266424900|26.309658000000002|
|2010-01-26 00:00:...|205.95000100000001|        213.710005|
202.580004|        205.940001|466777500|         26.681494|
|2010-01-27 00:00:...|        206.849995|            210.58|
199.530001|        207.880005|430642100|26.932840000000002|
|2010-01-28 00:00:...|        204.930004|        205.500004|
198.699995|        199.289995|293375600|25.819922000000002|
|2010-01-29 00:00:...|        201.079996|        202.199995|
190.250002|        192.060003|311488100|         24.883208|
|2010-02-01 00:00:...|192.36999699999998|             196.0|191.2999
9899999999|        194.729998|187469100|         25.229131|
+--------------------+------------------+------------------+--------
---------+------------------+---------+------------------+
only showing top 20 rows
```

In [83]:

```
df.filter(df['Close'] < 500).select('Volume').show()
```

```
+---------+
|   Volume|
+---------+
|123432400|
|150476200|
|138040000|
|119282800|
|111902700|
|115557400|
|148614900|
|151473000|
|108223500|
|148516900|
|182501900|
|153038200|
|152038600|
|220441900|
|266424900|
|466777500|
|430642100|
|293375600|
|311488100|
|187469100|
+---------+
only showing top 20 rows
```

In [84]:

```
#Multiple conditions in spark dataframe
df.filter((df['Close'] < 200) & (df['Open'] > 200)).show()
```

```
+-------------------+------------------+----------+----------+-----
-----+---------+------------------+
|               Date|              Open|      High|       Low|
Close|   Volume|         Adj Close|
+-------------------+------------------+----------+----------+-----
-----+---------+------------------+
|2010-01-22 00:00:...|206.78000600000001|207.499996|    197.16|    1
97.75|220441900|         25.620401|
|2010-01-28 00:00:...|        204.930004|205.500004|198.699995|199.2
89995|293375600|25.819922000000002|
|2010-01-29 00:00:...|        201.079996|202.199995|190.250002|192.0
60003|311488100|         24.883208|
+-------------------+------------------+----------+----------+-----
-----+---------+------------------+
```

In [85]:

```
#Not operator
df.filter((df['Close'] < 200) & ~(df['Open'] > 200)).show()
```

```
+-------------------+-----------------+-----------------+--------
---------+-----------------+---------+-----------------+
|               Date|             Open|             High|
Low|            Close|   Volume|        Adj Close|
+-------------------+-----------------+-----------------+--------
---------+-----------------+---------+-----------------+
|2010-02-01 00:00:...|192.36999699999998|            196.0|191.2999
9899999999|       194.729998|187469100|        25.229131|
|2010-02-02 00:00:...|       195.909998|       196.319994|193.3799
9299999998|       195.859997|174585600|25.375532999999997|
|2010-02-03 00:00:...|       195.169994|       200.200003|
194.420004|       199.229994|153832000|25.812148999999998|
|2010-02-04 00:00:...|       196.730003|       198.370001|
191.570005|       192.050003|189413000|        24.881912|
|2010-02-05 00:00:...|192.63000300000002|            196.0|
190.850002|       195.460001|212576700|25.323710000000002|
|2010-02-08 00:00:...|       195.690006|197.88000300000002|
193.999994|194.11999699999998|119567700|          25.1501|
|2010-02-09 00:00:...|       196.419996|       197.499994|
194.749998|196.19000400000002|158221700|        25.418289|
|2010-02-10 00:00:...|       195.889997|            196.6|
194.26|195.12000700000002| 92590400|         25.27966|
|2010-02-11 00:00:...|       194.880001|       199.750006|194.0599
9599999998|       198.669994|137586400|        25.739595|
|2010-02-23 00:00:...|       199.999998|       201.330002|
195.709993|       197.059998|143773700|        25.531005|
|2014-06-09 00:00:...|        92.699997|        93.879997|
91.75|         93.699997| 75415000|        88.906324|
|2014-06-10 00:00:...|        94.730003|        95.050003|
93.57|             94.25| 62777000|        89.428189|
|2014-06-11 00:00:...|        94.129997|        94.760002|
93.470001|        93.860001| 45681000|        89.058142|
|2014-06-12 00:00:...|        94.040001|        94.120003|
91.900002|        92.290001| 54749000|        87.568463|
|2014-06-13 00:00:...|        92.199997|        92.440002|
90.879997|        91.279999| 54525000|        86.610132|
|2014-06-16 00:00:...|        91.510002|             92.75|
91.449997|        92.199997| 35561000|        87.483064|
|2014-06-17 00:00:...|        92.309998|        92.699997|
91.800003| 92.08000200000001| 29726000| 87.36920699999999|
|2014-06-18 00:00:...|        92.269997|        92.290001|
91.349998|             92.18| 33514000|         87.46409|
|2014-06-19 00:00:...|        92.290001|        92.300003|
91.339996|        91.860001| 35528000|        87.160461|
|2014-06-20 00:00:...|        91.849998|        92.550003|
90.900002|        90.910004|100898000|        86.259066|
+-------------------+-----------------+-----------------+--------
---------+-----------------+---------+-----------------+
only showing top 20 rows
```

In [86]:

```
df.filter(df['Low'] == 197.16).show()
```

```
+-------------------+-----------------+----------+------+------+--
-------+---------+
|               Date|             Open|      High|   Low| Close|
Volume|Adj Close|
+-------------------+-----------------+----------+------+------+--
-------+---------+
|2010-01-22 00:00:...|206.78000600000001|207.499996|197.16|197.75|22
0441900|25.620401|
+-------------------+-----------------+----------+------+------+--
-------+---------+
```

In [88]:

```
#The above results look very messy and lot of times we want
#to collect the data to work with
df.filter(df['Low'] == 197.16).collect()
```

Out[88]:

```
[Row(Date=datetime.datetime(2010, 1, 22, 0, 0), Open=206.78000600000
001, High=207.499996, Low=197.16, Close=197.75, Volume=220441900, Ad
j Close=25.620401)]
```

In [89]:

```
result = df.filter(df['Low'] == 197.16).collect()
```

In [91]:

```
result[0]
```

Out[91]:

```
Row(Date=datetime.datetime(2010, 1, 22, 0, 0), Open=206.780006000000
01, High=207.499996, Low=197.16, Close=197.75, Volume=220441900, Adj
Close=25.620401)
```

In [92]:

```
row = result[0]

row.asDict()
```

Out[92]:

```
{'Adj Close': 25.620401,
 'Close': 197.75,
 'Date': datetime.datetime(2010, 1, 22, 0, 0),
 'High': 207.499996,
 'Low': 197.16,
 'Open': 206.78000600000001,
 'Volume': 220441900}
```

In [93]:

```
row.asDict()['Volume']
```

Out[93]:

```
220441900
```

In [94]:

```
result[0]['Volume']
```

Out[94]:

```
220441900
```

GroupBy, Aggregate and Sorting Functions

In [95]:

```
spark = SparkSession.builder.appName('aggs').getOrCreate()
```

In [96]:

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

In [98]:

```
df.count()
```

Out[98]:

```
12
```

In [99]:

```
df.printSchema()
```

```
root
 |-- Company: string (nullable = true)
 |-- Person: string (nullable = true)
 |-- Sales: double (nullable = true)
```

In [100]:

```
1 df.show()
```

```
+-------+-------+-----+
|Company| Person|Sales|
+-------+-------+-----+
|   GOOG|    Sam|200.0|
|   GOOG|Charlie|120.0|
|   GOOG|  Frank|340.0|
|   MSFT|   Tina|600.0|
|   MSFT|    Amy|124.0|
|   MSFT|Vanessa|243.0|
|     FB|   Carl|870.0|
|     FB|  Sarah|350.0|
|   APPL|   John|250.0|
|   APPL|  Linda|130.0|
|   APPL|   Mike|750.0|
|   APPL|  Chris|350.0|
+-------+-------+-----+
```

In [105]:

```
1 df.groupBy('Company').count().show()
```

```
+-------+-----+
|Company|count|
+-------+-----+
|   APPL|    4|
|   GOOG|    3|
|     FB|    2|
|   MSFT|    3|
+-------+-----+
```

In [107]:

```
1 df.groupBy('Company').mean().show()
```

```
+-------+-----------------+
|Company|       avg(Sales)|
+-------+-----------------+
|   APPL|            370.0|
|   GOOG|            220.0|
|     FB|            610.0|
|   MSFT|322.3333333333333|
+-------+-----------------+
```

In [108]:

```
df.groupBy('Company').max().show()
```

```
+-------+----------+
|Company|max(Sales)|
+-------+----------+
|   APPL|     750.0|
|   GOOG|     340.0|
|     FB|     870.0|
|   MSFT|     600.0|
+-------+----------+
```

In [109]:

```
df.groupBy('Company').min().show()
```

```
+-------+----------+
|Company|min(Sales)|
+-------+----------+
|   APPL|     130.0|
|   GOOG|     120.0|
|     FB|     350.0|
|   MSFT|     124.0|
+-------+----------+
```

In [110]:

```
## May be you dont need group by, you need something
#like average sales per company

df.agg({'Sales': 'sum'}).show()
```

```
+----------+
|sum(Sales)|
+----------+
|    4327.0|
+----------+
```

In [113]:

```
df.agg({'Sales': 'max', 'Sales': 'min'}).show()
```

```
+----------+
|min(Sales)|
+----------+
|     120.0|
+----------+
```

In [114]:

```
group_data = df.groupBy('Company')
```

In [116]:

```
group_data
```

Out[116]:

```
<pyspark.sql.group.GroupedData at 0x7fe465fefda0>
```

In [118]:

```
#The agg method done using groupby
group_data.agg({'Sales':'max'}).show()
```

```
+-------+----------+
|Company|max(Sales)|
+-------+----------+
|   APPL|     750.0|
|   GOOG|     340.0|
|     FB|     870.0|
|   MSFT|     600.0|
+-------+----------+
```

In [119]:

```
from pyspark.sql.functions import countDistinct, avg, stddev
```

In [121]:

```
#Counts distinct sales value
df.select(countDistinct('Sales')).show()
```

```
+---------------------+
|count(DISTINCT Sales)|
+---------------------+
|                   11|
+---------------------+
```

In [122]:

```
df.select(avg('Sales')).show()
```

```
+-----------------+
|       avg(Sales)|
+-----------------+
|360.5833333333333|
+-----------------+
```

In [124]:

```
#Alias - Give the column name appropriate name
df.select(avg('Sales').alias('Average Sales')).show()
```

```
+-----------------+
|    Average Sales|
+-----------------+
|360.5833333333333|
+-----------------+
```

In [125]:

```
df.select(stddev('Sales')).show()
```

```
+------------------+
|stddev_samp(Sales)|
+------------------+
|250.08742410799007|
+------------------+
```

In [126]:

```
#Formatting the number
from pyspark.sql.functions import format_number
```

In [127]:

```
sales_std = df.select(stddev('Sales').alias('std'))
```

In [128]:

```
sales_std.show()
```

```
+------------------+
|               std|
+------------------+
|250.08742410799007|
+------------------+
```

In [129]:

```
#number of decimal places I want to show
sales_std.select(format_number('std', 2)).show()
```

```
+---------------------+
|format_number(std, 2)|
+---------------------+
|               250.09|
+---------------------+
```

In [131]:

```
#Again chaining name here
sales_std.select(format_number('std', 2).alias('std')).show()
```

```
+------+
|   std|
+------+
|250.09|
+------+
```

In [134]:

```
#Ordering the columns, default is ascending order

df.orderBy('Sales').show()
```

```
+-------+-------+-----+
|Company| Person|Sales|
+-------+-------+-----+
|   GOOG|Charlie|120.0|
|   MSFT|    Amy|124.0|
|   APPL|  Linda|130.0|
|   GOOG|    Sam|200.0|
|   MSFT|Vanessa|243.0|
|   APPL|   John|250.0|
|   GOOG|  Frank|340.0|
|     FB|  Sarah|350.0|
|   APPL|  Chris|350.0|
|   MSFT|   Tina|600.0|
|   APPL|   Mike|750.0|
|     FB|   Carl|870.0|
+-------+-------+-----+
```

In [136]:

```
#Ordering in descending
df.orderBy(df['Sales'].desc()).show()
```

```
+-------+-------+-----+
|Company| Person|Sales|
+-------+-------+-----+
|     FB|   Carl|870.0|
|   APPL|   Mike|750.0|
|   MSFT|   Tina|600.0|
|     FB|  Sarah|350.0|
|   APPL|  Chris|350.0|
|   GOOG|  Frank|340.0|
|   APPL|   John|250.0|
|   MSFT|Vanessa|243.0|
|   GOOG|    Sam|200.0|
|   APPL|  Linda|130.0|
|   MSFT|    Amy|124.0|
|   GOOG|Charlie|120.0|
+-------+-------+-----+
```

Dealing with Missing Data in Spark

In [138]:

```
spark = SparkSession.builder.appName('miss').getOrCreate()
```

In [139]:

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

In [140]:

```
df.show()
```

```
+----+-----+-----+
|  Id| Name|Sales|
+----+-----+-----+
|emp1| John| null|
|emp2| null| null|
|emp3| null|345.0|
|emp4|Cindy|456.0|
+----+-----+-----+
```

In [142]:

```
#You can drop all the missing data
df.na.drop().show()
```

```
+----+-----+-----+
|  Id| Name|Sales|
+----+-----+-----+
|emp4|Cindy|456.0|
+----+-----+-----+
```

In [143]:

```
df.show()
```

```
+----+-----+-----+
|  Id| Name|Sales|
+----+-----+-----+
|emp1| John| null|
|emp2| null| null|
|emp3| null|345.0|
|emp4|Cindy|456.0|
+----+-----+-----+
```

In [144]:

```
#When we specify thresh=2, it looks at the row if it contains more than 2 nor
df.na.drop(thresh=2).show()
```

```
+----+-----+-----+
|  Id| Name|Sales|
+----+-----+-----+
|emp1| John| null|
|emp3| null|345.0|
|emp4|Cindy|456.0|
+----+-----+-----+
```

In [145]:

```
#By default how is set to 'aby', meaning drop if there are any null values

df.na.drop(how='any').show()
```

```
+----+-----+-----+
|  Id| Name|Sales|
+----+-----+-----+
|emp4|Cindy|456.0|
+----+-----+-----+
```

In [146]:

```
#We can change the how to 'all', if row has all null values only then drop

df.na.drop(how='all').show()
```

```
+----+-----+-----+
|  Id| Name|Sales|
+----+-----+-----+
|emp1| John| null|
|emp2| null| null|
|emp3| null|345.0|
|emp4|Cindy|456.0|
+----+-----+-----+
```

In [147]:

```
#The subset method will only drop if Sales colum has null values
df.na.drop(subset=['Sales']).show()
```

```
+----+-----+-----+
|  Id| Name|Sales|
+----+-----+-----+
|emp3| null|345.0|
|emp4|Cindy|456.0|
+----+-----+-----+
```

In [148]:

```
df.printSchema()
```

```
root
 |-- Id: string (nullable = true)
 |-- Name: string (nullable = true)
 |-- Sales: double (nullable = true)
```

In [149]:

```
#We can also fill NULL values
df.na.fill('FILL VALUE').show()
```

```
+----+----------+-----+
|  Id|      Name|Sales|
+----+----------+-----+
|emp1|      John| null|
|emp2|FILL VALUE| null|
|emp3|FILL VALUE|345.0|
|emp4|     Cindy|456.0|
+----+----------+-----+
```

In [151]:

```
#Spark is smart enough to recognize string datatype and only fill
#in string datatype column, similar thing it does for numeric values
df.na.fill(0).show()
```

```
+----+-----+-----+
|  Id| Name|Sales|
+----+-----+-----+
|emp1| John|  0.0|
|emp2| null|  0.0|
|emp3| null|345.0|
|emp4|Cindy|456.0|
+----+-----+-----+
```

In [153]:

```
#You can specify the column name in the subset parameter
#Its a best practice to declare subset always
df.na.fill('No Name', subset=['Name']).show()
```

```
+----+-------+-----+
|  Id|   Name|Sales|
+----+-------+-----+
|emp1|   John| null|
|emp2|No Name| null|
|emp3|No Name|345.0|
|emp4|  Cindy|456.0|
+----+-------+-----+
```

In [154]:

```
from pyspark.sql.functions import mean
```

In [155]:

```
mean_val = df.select(mean(df['Sales'])).collect()
```

In [157]:

```
mean_val[0]
```

Out[157]:

```
Row(avg(Sales)=400.5)
```

In [158]:

```
mean_val[0][0]
```

Out[158]:

```
400.5
```

In [159]:

```
mean_sales = mean_val[0][0]
```

In [161]:

```
df.na.fill(mean_sales, ['Sales']).show()
```

```
+----+-----+-----+
|  Id| Name|Sales|
+----+-----+-----+
|emp1| John|400.5|
|emp2| null|400.5|
|emp3| null|345.0|
|emp4|Cindy|456.0|
+----+-----+-----+
```

In [167]:

```
#Doing all this in one line
df.na.fill(df.select(mean(df['Sales'])).collect()[0][0], ['Sales']).show()
```

```
+----+-----+-----+
|  Id| Name|Sales|
+----+-----+-----+
|emp1| John|400.5|
|emp2| null|400.5|
|emp3| null|345.0|
|emp4|Cindy|456.0|
+----+-----+-----+
```

Dates and Timestamps

In [168]:

```
spark = SparkSession.builder.appName('dates').getOrCreate()
```

In [170]:

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

In [176]:

```
df.count()
```

Out[176]:

```
1762
```

In [178]:

```
df.head(1)
#datetime object in Date column, this looks messy here
```

Out[178]:

```
[Row(Date=datetime.datetime(2010, 1, 4, 0, 0), Open=213.429998, High
=214.499996, Low=212.38000099999996, Close=214.009998, Volume=123432
400, Adj Close=27.727039)]
```

In [179]:

```
df.select(['Date', 'Open']).show()
```

```
+--------------------+------------------+
|                Date|              Open|
+--------------------+------------------+
|2010-01-04 00:00:...|        213.429998|
|2010-01-05 00:00:...|        214.599998|
|2010-01-06 00:00:...|        214.379993|
|2010-01-07 00:00:...|            211.75|
|2010-01-08 00:00:...|        210.299994|
|2010-01-11 00:00:...|212.79999700000002|
|2010-01-12 00:00:...|209.18999499999998|
|2010-01-13 00:00:...|        207.870005|
|2010-01-14 00:00:...|210.11000299999998|
|2010-01-15 00:00:...|210.92999500000002|
|2010-01-19 00:00:...|        208.330002|
|2010-01-20 00:00:...|        214.910006|
|2010-01-21 00:00:...|        212.079994|
|2010-01-22 00:00:...|206.78000600000001|
|2010-01-25 00:00:...|202.51000200000001|
|2010-01-26 00:00:...|205.95000100000001|
|2010-01-27 00:00:...|        206.849995|
|2010-01-28 00:00:...|        204.930004|
|2010-01-29 00:00:...|        201.079996|
|2010-02-01 00:00:...|192.36999699999998|
+--------------------+------------------+
only showing top 20 rows
```

In [180]:

```
from pyspark.sql.functions import (dayofmonth, hour,
                                   dayofyear, month,
                                   year, weekofyear,
                                  format_number, date_format)
```

In [182]:

```
#Whenever you are using any pyspark imported function,
#you can directly use them in select on your column
df.select(dayofmonth(df['Date'])).show()
```

```
+----------------+
|dayofmonth(Date)|
+----------------+
|               4|
|               5|
|               6|
|               7|
|               8|
|              11|
|              12|
|              13|
|              14|
|              15|
|              19|
|              20|
|              21|
|              22|
|              25|
|              26|
|              27|
|              28|
|              29|
|               1|
+----------------+
only showing top 20 rows
```

In [183]:

```
1 df.select(hour(df['Date'])).show()
```

```
+----------+
|hour(Date)|
+----------+
|         0|
|         0|
|         0|
|         0|
|         0|
|         0|
|         0|
|         0|
|         0|
|         0|
|         0|
|         0|
|         0|
|         0|
|         0|
|         0|
|         0|
|         0|
|         0|
|         0|
+----------+
only showing top 20 rows
```

In [184]:

```
1 df.select(month(df['Date'])).show()
```

```
+-----------+
|month(Date)|
+-----------+
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          2|
+-----------+
only showing top 20 rows
```

In [188]:

```
## MAy be you want to know average closing price pwe year

df.select(year(df['Date'])).show()
```

```
+----------+
|year(Date)|
+----------+
|      2010|
|      2010|
|      2010|
|      2010|
|      2010|
|      2010|
|      2010|
|      2010|
|      2010|
|      2010|
|      2010|
|      2010|
|      2010|
|      2010|
|      2010|
|      2010|
|      2010|
|      2010|
|      2010|
|      2010|
+----------+
only showing top 20 rows
```

In [190]:

```
1 df.withColumn('Year', year(df['Date'])).show()
```

```
+--------------------+------------------+------------------+--------
---------+------------------+---------+------------------+----+
|                Date|              Open|              High|
Low|             Close|    Volume|         Adj Close|Year|
+--------------------+------------------+------------------+--------
---------+------------------+---------+------------------+----+
|2010-01-04 00:00:...|        213.429998|        214.499996|212.3800
0099999996|        214.009998|123432400|         27.727039|2010|
|2010-01-05 00:00:...|        214.599998|        215.589994|
213.249994|        214.379993|150476200|27.774976000000002|2010|
|2010-01-06 00:00:...|        214.379993|            215.23|
210.750004|        210.969995|138040000|27.333178000000004|2010|
|2010-01-07 00:00:...|            211.75|        212.000006|
209.050005|            210.58|119282800|          27.28265|2010|
|2010-01-08 00:00:...|        210.299994|        212.000006|209.0600
0500000002|211.98000499999998|111902700|         27.464034|2010|
|2010-01-11 00:00:...|212.79999700000002|        213.000002|
208.450005|210.11000299999998|115557400|         27.221758|2010|
|2010-01-12 00:00:...|209.18999499999998|209.76999500000002|
206.419998|        207.720001|148614900|          26.91211|2010|
|2010-01-13 00:00:...|        207.870005|210.92999500000002|
204.099998|        210.650002|151473000|          27.29172|2010|
|2010-01-14 00:00:...|210.11000299999998|210.45999700000002|
209.020004|            209.43|108223500|         27.133657|2010|
|2010-01-15 00:00:...|210.92999500000002|211.59999700000003|
205.869999|            205.93|148516900|26.680197999999997|2010|
|2010-01-19 00:00:...|        208.330002|215.18999900000003|
207.240004|        215.039995|182501900|27.860484999999997|2010|
|2010-01-20 00:00:...|        214.910006|        215.549994|
209.500002|            211.73|153038200|         27.431644|2010|
|2010-01-21 00:00:...|        212.079994|213.30999599999998|
207.210003|        208.069996|152038600|         26.957455|2010|
|2010-01-22 00:00:...|206.78000600000001|        207.499996|
197.16|            197.75|220441900|         25.620401|2010|
|2010-01-25 00:00:...|202.51000200000001|        204.699999|
200.190002|        203.070002|266424900|26.309658000000002|2010|
|2010-01-26 00:00:...|205.95000100000001|        213.710005|
202.580004|        205.940001|466777500|         26.681494|2010|
|2010-01-27 00:00:...|        206.849995|            210.58|
199.530001|        207.880005|430642100|26.932840000000002|2010|
|2010-01-28 00:00:...|        204.930004|        205.500004|
198.699995|        199.289995|293375600|25.819922000000002|2010|
|2010-01-29 00:00:...|        201.079996|        202.199995|
190.250002|        192.060003|311488100|         24.883208|2010|
|2010-02-01 00:00:...|192.36999699999998|             196.0|191.2999
9899999999|        194.729998|187469100|         25.229131|2010|
+--------------------+------------------+------------------+--------
---------+------------------+---------+------------------+----+
only showing top 20 rows
```

In [191]:

```
1 new_df = df.withColumn('Year', year(df['Date']))
```

In [192]:

```
1 new_df.groupBy('Year').mean().show()
```

```
+----+-----------------+-----------------+-----------------+-----
------------+------------------+-----------------+---------+
|Year|         avg(Open)|         avg(High)|          avg(Low)|
avg(Close)|        avg(Volume)|    avg(Adj Close)|avg(Year)|
+----+-----------------+-----------------+-----------------+-----
------------+------------------+-----------------+---------+
|2015|120.17575393253965|121.24452385714291| 118.8630954325397|120.0
3999980555547|  5.18378869047619E7|115.96740080555561|   2015.0|
|2013| 473.1281355634922| 477.6389272301587|468.24710264682557| 472.
6348802857143|          1.016087E8| 62.61798788492063|   2013.0|
|2014| 295.1426195357143|297.56103184523823| 292.9949599801587| 295.
4023416507935| 6.315273055555555E7| 87.63583323809523|   2014.0|
|2012|     576.652720788| 581.8254008040001| 569.9211606079999| 576.
0497195640002|       1.319642044E8| 74.81383696800002|   2012.0|
|2016|104.50777772619044| 105.4271825436508|103.69027771825397|104.6
0400786904763|  3.84153623015873E7|103.15032854761901|   2016.0|
|2010| 259.9576190992064|262.36880881349214|256.84761791269847| 259.
8424600000002|1.4982631666666666E8|33.665072424603196|   2010.0|
|2011|364.06142773412705| 367.4235704880951|360.29769878174613|364.0
0432532142867|1.2307474166666667E8| 47.16023692063492|   2011.0|
+----+-----------------+-----------------+-----------------+-----
------------+------------------+-----------------+---------+
```

In [195]:

```
1 new_df.groupBy('Year').mean().select(['avg(Close)', 'Year']).show()
```

```
+------------------+----+
|        avg(Close)|Year|
+------------------+----+
|120.03999980555547|2015|
| 472.6348802857143|2013|
| 295.4023416507935|2014|
| 576.0497195640002|2012|
|104.60400786904763|2016|
| 259.8424600000002|2010|
|364.00432532142867|2011|
+------------------+----+
```

In [198]:

```
1 result = new_df.groupBy('Year').mean().select(['avg(Close)', 'Year'])
```

In [199]:

```
1 result.show()
```

```
+------------------+----+
|        avg(Close)|Year|
+------------------+----+
|120.03999980555547|2015|
| 472.6348802857143|2013|
| 295.4023416507935|2014|
| 576.0497195640002|2012|
|104.60400786904763|2016|
| 259.8424600000002|2010|
|364.00432532142867|2011|
+------------------+----+
```

In [201]:

```
1 result = result.withColumnRenamed('avg(Close)', 'Average Closing Price')
```

In [205]:

```
1 result.select(['Year', format_number('Average Closing Price', 2)]).show()
```

```
+----+--------------------------------------+
|Year|format_number(Average Closing Price, 2)|
+----+--------------------------------------+
|2015|                                120.04|
|2013|                                472.63|
|2014|                                295.40|
|2012|                                576.05|
|2016|                                104.60|
|2010|                                259.84|
|2011|                                364.00|
+----+--------------------------------------+
```

In [206]:

```
1 result.select(['Year', format_number('Average Closing Price', 2).alias('Avera
```

```
+----+---------------------+
|Year|Average Closing Price|
+----+---------------------+
|2015|                120.04|
|2013|                472.63|
|2014|                295.40|
|2012|                576.05|
|2016|                104.60|
|2010|                259.84|
|2011|                364.00|
+----+---------------------+
```

Thank you!