5_Dates_and_Timestamps

February 22, 2022

1 Dates and Timestamps

Very often, the real time data sets have to handle date and time..let us look into this..

[1]:
```
from pyspark.sql import SparkSession

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

```
22/02/22 11:47:21 WARN Utils: Your hostname, ThinkCentre resolves to a loopback
address: 127.0.1.1; using 10.180.5.223 instead (on interface eno1)
22/02/22 11:47:21 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
22/02/22 11:47:21 WARN NativeCodeLoader: Unable to load native-hadoop library
for your platform… using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
22/02/22 11:47:23 WARN Utils: Service 'SparkUI' could not bind on port 4040.
Attempting port 4041.
```

[2]:
```
df = spark.read.csv("appl_stock.csv",header=True,inferSchema=True)
```

[3]:
```
df.show(5)
```

```
+----------+----------+----------+------------------+------------------+--------
-+------------------+
|      Date|      Open|      High|               Low|             Close|
Volume|         Adj Close|
+----------+----------+----------+------------------+------------------+--------
-+------------------+
|2010-01-04|213.429998|214.499996|212.38000099999996|
214.009998|123432400|         27.727039|
|2010-01-05|214.599998|215.589994|        213.249994|
214.379993|150476200|27.774976000000002|
|2010-01-06|214.379993|    215.23|        210.750004|
210.969995|138040000|27.333178000000004|
|2010-01-07|    211.75|212.000006|        209.050005|
210.58|119282800|          27.28265|
|2010-01-08|210.299994|212.000006|209.06000500000002|211.98000499999998|11190270
0|         27.464034|
+----------+----------+----------+------------------+------------------+--------
-+------------------+
only showing top 5 rows
```

[4]:
```
df.printSchema()
```

```
root
 |-- Date: string (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)
```

[5]:
```
df.head(1)
```

[5]:
```
[Row(Date='2010-01-04', Open=213.429998, High=214.499996,
Low=212.38000099999996, Close=214.009998, Volume=123432400, Adj
Close=27.727039)]
```

Let's walk through how to grab parts of the timestamp data

[6]:
```
# Import the necessary functions related to date and time
from pyspark.sql.functions import format_number, dayofmonth, hour, dayofyear,␣
 ↪month, year, weekofyear, date_format
```

[7]:
```
# Show date only
df.select(dayofmonth(df['Date'])).show(5) # day of Month of first 5 rows
```

```
+----------------+
|dayofmonth(Date)|
+----------------+
|               4|
|               5|
|               6|
|               7|
|               8|
+----------------+
only showing top 5 rows
```

[8]:
```
# Print hour
df.select(hour(df['Date'])).show(10)
```

```
+----------+
|hour(Date)|
+----------+
|         0|
|         0|
|         0|
|         0|
|         0|
|         0|
|         0|
|         0|
|         0|
|         0|
+----------+
only showing top 10 rows
```

[10]:
```
# Print day of an year
df.select(dayofyear(df['Date'])).show(10)
```

```
+---------------+
|dayofyear(Date)|
+---------------+
|              4|
|              5|
|              6|
|              7|
|              8|
|             11|
|             12|
|             13|
|             14|
|             15|
+---------------+
only showing top 10 rows
```

[11]:
```
df.select(month(df['Date'])).show(40)
```

```
+-----------+
|month(Date)|
+-----------+
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          1|
|          2|
|          2|
|          2|
|          2|
|          2|
|          2|
|          2|
|          2|
|          2|
|          2|
|          2|
|          2|
|          2|
|          2|
|          2|
|          2|
|          2|
|          2|
|          2|
|          3|
|          3|
+-----------+
only showing top 40 rows
```

[12]:
```python
# How to select an year?
df.select(year(df['Date'])).show(5)
```

```
+----------+
|year(Date)|
+----------+
|      2010|
|      2010|
|      2010|
|      2010|
|      2010|
+----------+
only showing top 5 rows
```