FinalProject_WeatherData Untitled Untitled Untitled Untitled Untitled Untitled Untitled Untitle

Zeppelin

FinalProject_Weath...

default

FINISHED

```
import org.apache.spark.sql.functions._
import org.joda.time.format.DateTimeFormat
import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset
```

```
import org.apache.spark.sql.functions._
import org.joda.time.format.DateTimeFormat
import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset
```

Took 3 sec. Last updated by anonymous at January 29 2017, 4:16:28 PM.

FINISHED

```
// load aarhus city data - store different parameters in different variables

//define path for reading the json format file
val inputPath =  "D:/Aakash_Documents/MS_Collections/AcceptanceFromSaintPeters/ClassStuff/DS_670_Capst

//read dewpoint weather parameter
val dewpoint_feb_jun = sqlContext.read
        .format("com.databricks.spark.csv")
        .option("header", "true") // Use first line of all files as header
        .option("delimiter", ",")
        .option("inferSchema", "true") // Automatically infer data types
        .load(inputPath+"/dewpoint/dewptm_Feb_Jun.csv")

val dewpoint_aug_sep = sqlContext.read
        .format("com.databricks.spark.csv")
        .option("header", "true") // Use first line of all files as header
        .option("delimiter", ",")
        .option("inferSchema", "true") // Automatically infer data types
        .load(inputPath+"/dewpoint/dewptm_Aug_Sep.csv")

//read humidity weather parameter
val humidity_feb_jun = sqlContext.read
        .format("com.databricks.spark.csv")
        .option("header", "true") // Use first line of all files as header
        .option("delimiter", ",")
        .option("inferSchema", "true") // Automatically infer data types
        .load(inputPath+"/humidity/hum_feb_jun.csv")

val humidity_aug_sep = sqlContext.read
        .format("com.databricks.spark.csv")
        .option("header", "true") // Use first line of all files as header
        .option("delimiter", ",")
        .option("inferSchema", "true") // Automatically infer data types
        .load(inputPath+"/humidity/hum_aug_sep.csv")

//read pressure weather parameter
val pressure_feb_jun = sqlContext.read
        .format("com.databricks.spark.csv")
        .option("header", "true") // Use first line of all files as header
        .option("delimiter", ",")
        .option("inferSchema", "true") // Automatically infer data types
        .load(inputPath+"/pressure/pressurem_feb_jun.csv")
```

FinalProject_WeatherData

Zeppelin

FinalProject_Weath...

default

```
//read temperature weather parameter
val temp_feb_jun = sqlContext.read
        .format("com.databricks.spark.csv")
        .option("header", "true") // Use first line of all files as header
        .option("delimiter", ",")
        .option("inferSchema", "true") // Automatically infer data types
        .load(inputPath+"/temperature/tempm_feb_jun.csv")

val temp_aug_sep = sqlContext.read
        .format("com.databricks.spark.csv")
        .option("header", "true") // Use first line of all files as header
        .option("delimiter", ",")
        .option("inferSchema", "true") // Automatically infer data types
        .load(inputPath+"/temperature/tempm_aug_sept.csv")

//read wind direction weather parameter
val winddirection_feb_jun = sqlContext.read
        .format("com.databricks.spark.csv")
        .option("header", "true") // Use first line of all files as header
        .option("delimiter", ",")
        .option("inferSchema", "true") // Automatically infer data types
        .load(inputPath+"/winddirection/wdird_feb_jun.csv")

val winddirection_aug_sep = sqlContext.read
        .format("com.databricks.spark.csv")
        .option("header", "true") // Use first line of all files as header
        .option("delimiter", ",")
        .option("inferSchema", "true") // Automatically infer data types
        .load(inputPath+"/winddirection/wdird_aug_sept.csv")

//read wind speed weather parameter
val windspeed_feb_jun = sqlContext.read
        .format("com.databricks.spark.csv")
        .option("header", "true") // Use first line of all files as header
        .option("delimiter", ",")
        .option("inferSchema", "true") // Automatically infer data types
        .load(inputPath+"/windspeed/wspdm_feb_jun.csv")

val windspeed_aug_sep = sqlContext.read
        .format("com.databricks.spark.csv")
        .option("header", "true") // Use first line of all files as header
        .option("delimiter", ",")
        .option("inferSchema", "true") // Automatically infer data types
```

```
inputPath: String = D:/Aakash_Documents/MS_Collections/AcceptanceFromSaintPeters/ClassStuff/DS_670_Capstone/FinalProject_WeatherReport/dataset/raw_weather_data_aarhus
dewpoint_feb_jun: org.apache.spark.sql.DataFrame = [DateTime: timestamp, DewPoint: int]
dewpoint_aug_sep: org.apache.spark.sql.DataFrame = [DateTime: timestamp, DewPoint: int]
humidity_feb_jun: org.apache.spark.sql.DataFrame = [DateTime: timestamp, Humidity: int]
```

Took 7 sec. Last updated by anonymous at January 29 2017, 4:16:38 PM.

FINISHED

```
//let us verify the datatype of each variable
dewpoint_feb_jun.printSchema()
dewpoint_aug_sep.printSchema()

humidity_feb_jun.printSchema()
humidity_aug_sep.printSchema()

pressure_feb_jun.printSchema()
```

FinalProject_WeatherData

Zeppelin

FinalProject_Weath...

default

```
pressure_aug_sep.printSchema()

temp_feb_jun.printSchema()
temp_aug_sep.printSchema()

winddirection_feb_jun.printSchema()
winddirection_aug_sep.printSchema()

windspeed_feb_jun.printSchema()
windspeed_aug_sep.printSchema()
```

```
 |-- Temperature: integer (nullable = true)

root
 |-- DateTime: timestamp (nullable = true)
 |-- WindDirection: integer (nullable = true)

root
 |-- DateTime: timestamp (nullable = true)
 |-- WindDirection: integer (nullable = true)

root
 |-- DateTime: timestamp (nullable = true)
 |-- WindSpeed: double (nullable = true)

root
 |-- DateTime: timestamp (nullable = true)
 |-- WindSpeed: double (nullable = true)
```

Took 4 sec. Last updated by anonymous at January 29 2017, 4:16:50 PM.

FINISHED

```
//create temp table of each weather data, data frame
dewpoint_feb_jun.registerTempTable("dewpoint_feb_jun");
dewpoint_aug_sep.registerTempTable("dewpoint_feb_jun");

humidity_feb_jun.registerTempTable("humidity_feb_jun");
humidity_aug_sep.registerTempTable("humidity_aug_sep");

pressure_feb_jun.registerTempTable("pressure_feb_jun");
pressure_aug_sep.registerTempTable("pressure_aug_sep");

temp_feb_jun.registerTempTable("temp_feb_jun");
temp_aug_sep.registerTempTable("temp_aug_sep");

winddirection_feb_jun.registerTempTable("winddirection_feb_jun");
winddirection_aug_sep.registerTempTable("winddirection_aug_sep");

windspeed_feb_jun.registerTempTable("windspeed_feb_jun");
windspeed_aug_sep.registerTempTable("windspeed_aug_sep");
```

```
warning: there was one deprecation warning; re-run with -deprecation for details
warning: there was one deprecation warning; re-run with -deprecation for details
warning: there was one deprecation warning; re-run with -deprecation for details
warning: there was one deprecation warning; re-run with -deprecation for details
warning: there was one deprecation warning; re-run with -deprecation for details
warning: there was one deprecation warning; re-run with -deprecation for details
warning: there was one deprecation warning; re-run with -deprecation for details
warning: there was one deprecation warning; re-run with -deprecation for details
warning: there was one deprecation warning; re-run with -deprecation for details
warning: there was one deprecation warning; re-run with -deprecation for details
warning: there was one deprecation warning; re-run with -deprecation for details
```

FinalProject_WeatherData Untitled Untitled Untitled Untitled Untitled Untitled Untitled Untitle

warning: there was one deprecation warning; re-run wit deprecation for details

Took 12 sec. Last updated by anonymous at January 29 2017, 4:17:18 PM.

Zeppelin

FinalProject_Weath...

FINISHED default

```
%sql
/*find minimum,maximum and average values of dewpoint*/
select min(Dewpoint) as minimumValue,max(Dewpoint) as maximuValue, avg(Dewpoint) averagevalue from dew
```

minimumValue	maximuValue
0	19

Took 1 sec. Last updated by anonymous at January 29 2017, 4:22:10 PM.

FINISHED

```
%sql
/*find minimum,maximum and average values of humidity*/
select min(humidity) as minimumValue,max(humidity) as maximuValue, avg(humidity) averagevalue from hum
```

minimumValue	maximuValue
12	100

Took 0 sec. Last updated by anonymous at January 29 2017, 4:23:39 PM.

FinalProject_WeatherData

FinalProject_WeatherData Untitled Untitled Untitled Untitled Untitled Untitled Untitled Untitle

Zeppelin

FinalProject_Weath...

```
%sql
/*find minimum,maximum and average values of pressure*/
select min(pressure) as minimumValue,max(pressure) as maximuValue, avg(pressure) averagevalue from pre
```

FINISHED

default

minimumValue	maximuValue
986	1,038

Took 0 sec. Last updated by anonymous at January 29 2017, 4:24:55 PM.

```
%sql
/*find minimum,maximum and average values of temperature*/
select min(temperature) as minimumValue,max(temperature) as maximuValue, avg(temperature) averagevalue
```

FINISHED

minimumValue	maximuValue
-3	25

Took 0 sec. Last updated by anonymous at January 29 2017, 4:26:10 PM.

```
%sql
/*find minimum,maximum and average values of winddirection*/
select min(winddirection) as minimumValue,max(winddirection) as maximuValue, avg(winddirection) averag
```

FINISHED

Zeppelin

FinalProject_Weath...

minimumValue	maximuValue
0	360

Took 1 sec. Last updated by anonymous at January 29 2017, 4:29:52 PM.

FINISHED

```
%sql
/*find minimum,maximum and average values of windspeed*/
select min(windspeed) as minimumValue,max(windspeed) as maximuValue, avg(windspeed) averagevalue from
```

minimumValue	maximuValue
-9,999	64.8

Took 1 sec. Last updated by anonymous at January 29 2017, 4:31:05 PM. (outdated)

READY