Problem Statement

Here is the problem statement for which we will be exploring Data Frame APIs to come up with final solution.

* Get daily product revenue
* orders – order\_id, order\_date, order\_customer\_id, order\_status
* order\_items – order\_item\_id, order\_item\_order\_id, order\_item\_product\_id, order\_item\_quantity, order\_item\_subtotal, order\_item\_product\_price
* Data is comma separated
* We will fetch data using spark.read.csv
* Apply type cast functions to convert fields into their original type where ever is applicable.
* os.system('hadoop fs -ls /user/saurabhbhanwala/retail\_db')
* Found 7 items
* drwxr-xr-x - saurabhbhanwala hdfs 0 2017-02-20 01:35 /user/saurabhbhanwala/retail\_db/categories
* drwxr-xr-x - saurabhbhanwala hdfs 0 2017-02-20 01:35 /user/saurabhbhanwala/retail\_db/customers
* drwxr-xr-x - saurabhbhanwala hdfs 0 2017-02-20 01:35 /user/saurabhbhanwala/retail\_db/departments
* drwxr-xr-x - saurabhbhanwala hdfs 0 2017-02-20 01:35 /user/saurabhbhanwala/retail\_db/order\_items
* drwxr-xr-x - saurabhbhanwala hdfs 0 2017-02-20 01:35 /user/saurabhbhanwala/retail\_db/orders
* drwxr-xr-x - saurabhbhanwala hdfs 0 2017-02-20 01:35 /user/saurabhbhanwala/retail\_db/products
* drwxr-xr-x - saurabhbhanwala hdfs 0 2020-09-06 05:50 /user/saurabhbhanwala/retail\_db/retail\_db
* 0
* >>> ordersItems = spark.read.format('csv').schema('order\_item\_id int, order\_item\_order\_id int, order\_item\_product\_id int, order\_item\_quantity int, order\_item\_subtotal float, order\_item\_product\_price float').load('/user/saurabhbhanwala/retail\_db/order\_items')
* >>> orders = spark.read.format('csv').schema('order\_id int, order\_date string, order\_customer\_id int, order\_status string').load('/user/saurabhbhanwala/retail\_db/orders') >>>

>>> type(orders)

<class 'pyspark.sql.dataframe.DataFrame'>

>>> type(ordersItems)

<class 'pyspark.sql.dataframe.DataFrame'>

>>>

> from pyspark import SparkContext

>>> from pyspark.sql import SparkSession

>>> spark = SparkSession.builder.master('yarn').appName('CSV').getOrCreate()

>>> orders = spark.read.format('csv').schema('order\_id int, order\_date string, order\_customer\_id int, order\_status string').load('/user/saurabhbhanwala/retail\_db/orders')

>>> ordersItems = spark.read.format('csv').schema('order\_item\_id int, order\_item\_order\_id int, order\_item\_product\_id int, order\_item\_quantity int, order\_item\_subtotal float, order\_item\_product\_price float').load('/user/saurabhbhanwala/retail\_db/order\_items')

>>> orders.show()

+--------+--------------------+-----------------+---------------+

|order\_id| order\_date|order\_customer\_id| order\_status|

+--------+--------------------+-----------------+---------------+

| 1|2013-07-25 00:00:...| 11599| CLOSED|

| 2|2013-07-25 00:00:...| 256|PENDING\_PAYMENT|

| 3|2013-07-25 00:00:...| 12111| COMPLETE|

| 4|2013-07-25 00:00:...| 8827| CLOSED|

| 5|2013-07-25 00:00:...| 11318| COMPLETE|

| 6|2013-07-25 00:00:...| 7130| COMPLETE|

| 7|2013-07-25 00:00:...| 4530| COMPLETE|

| 8|2013-07-25 00:00:...| 2911| PROCESSING|

| 9|2013-07-25 00:00:...| 5657|PENDING\_PAYMENT|

| 10|2013-07-25 00:00:...| 5648|PENDING\_PAYMENT|

| 11|2013-07-25 00:00:...| 918| PAYMENT\_REVIEW|

| 12|2013-07-25 00:00:...| 1837| CLOSED|

| 13|2013-07-25 00:00:...| 9149|PENDING\_PAYMENT|

| 14|2013-07-25 00:00:...| 9842| PROCESSING|

| 15|2013-07-25 00:00:...| 2568| COMPLETE|

| 16|2013-07-25 00:00:...| 7276|PENDING\_PAYMENT|

| 17|2013-07-25 00:00:...| 2667| COMPLETE|

| 18|2013-07-25 00:00:...| 1205| CLOSED|

| 19|2013-07-25 00:00:...| 9488|PENDING\_PAYMENT|

| 20|2013-07-25 00:00:...| 9198| PROCESSING|

+--------+--------------------+-----------------+---------------+

only showing top 20 rows

>>> orderItems.show()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'orderItems' is not defined

>>> ordersItems.show()

+-------------+-------------------+---------------------+-------------------+-------------------+------------------------+

|order\_item\_id|order\_item\_order\_id|order\_item\_product\_id|order\_item\_quantity|order\_item\_subtotal|order\_item\_product\_price|

+-------------+-------------------+---------------------+-------------------+-------------------+------------------------+

| 1| 1| 957| 1| 299.98| 299.98|

| 2| 2| 1073| 1| 199.99| 199.99|

| 3| 2| 502| 5| 250.0| 50.0|

| 4| 2| 403| 1| 129.99| 129.99|

| 5| 4| 897| 2| 49.98| 24.99|

| 6| 4| 365| 5| 299.95| 59.99|

| 7| 4| 502| 3| 150.0| 50.0|

| 8| 4| 1014| 4| 199.92| 49.98|

| 9| 5| 957| 1| 299.98| 299.98|

| 10| 5| 365| 5| 299.95| 59.99|

| 11| 5| 1014| 2| 99.96| 49.98|

| 12| 5| 957| 1| 299.98| 299.98|

| 13| 5| 403| 1| 129.99| 129.99|

| 14| 7| 1073| 1| 199.99| 199.99|

| 15| 7| 957| 1| 299.98| 299.98|

| 16| 7| 926| 5| 79.95| 15.99|

| 17| 8| 365| 3| 179.97| 59.99|

| 18| 8| 365| 5| 299.95| 59.99|

| 19| 8| 1014| 4| 199.92| 49.98|

| 20| 8| 502| 1| 50.0| 50.0|

+-------------+-------------------+---------------------+-------------------+-------------------+------------------------+

only showing top 20 rows

ordersDF = spark.read.csv('/user/saurabhbhanwala/retail\_db/orders').toDF('order\_id', 'order\_date', 'order\_customer\_id', 'order\_status')

>>> orderItemsDF = spark.read.csv('/user/saurabhbhanwala/retail\_db/order\_items').toDF('order\_item\_id', 'order\_item\_order\_id', 'order\_item\_product\_id', 'order\_item\_quantity', 'order\_item\_subtotal', 'order\_item\_product\_price')

|  |
| --- |
| In case you are using pycharm, first you need to create object of type SparkSession |
|  | spark = SparkSession. \ |
|  | builder. \ |
|  | master('local'). \ |
|  | appName('CSV Example'). \ |
|  | getOrCreate() |
|  |  |
|  | ordersCSV = spark.read. \ |
|  | csv('/public/retail\_db/orders'). \ |
|  | toDF('order\_id', 'order\_date', 'order\_customer\_id', 'order\_status') |
|  |  |
|  | orderItemsCSV = spark.read. \ |
|  | csv('/public/retail\_db/order\_items'). \ |
|  | toDF('order\_item\_id', 'order\_item\_order\_id', 'order\_item\_product\_id', |
|  | 'order\_item\_quantity', 'order\_item\_subtotal', 'order\_item\_product\_price') |
|  |  |
|  | from pyspark.sql.types import IntegerType, FloatType |
|  |  |
|  | orders = ordersCSV. \ |
|  | withColumn('order\_id', ordersCSV.order\_id.cast(IntegerType())). \ |
|  | withColumn('order\_customer\_id', ordersCSV.order\_customer\_id.cast(IntegerType())) |
|  |  |
|  | orders.printSchema() |
|  | orders.show() |
|  |  |
|  | orderItems = orderItemsCSV.\ |
|  | withColumn('order\_item\_id', orderItemsCSV.order\_item\_id.cast(IntegerType())). \ |
|  | withColumn('order\_item\_order\_id', orderItemsCSV.order\_item\_order\_id.cast(IntegerType())). \ |
|  | withColumn('order\_item\_product\_id', orderItemsCSV.order\_item\_product\_id.cast(IntegerType())). \ |
|  | withColumn('order\_item\_quantity', orderItemsCSV.order\_item\_quantity.cast(IntegerType())). \ |
|  | withColumn('order\_item\_subtotal', orderItemsCSV.order\_item\_subtotal.cast(FloatType())). \ |
|  | withColumn('order\_item\_product\_price', orderItemsCSV.order\_item\_product\_price.cast(FloatType())) |
|  |  |
|  | orderItems.printSchema() |
|  | orderItems.show() |

 Selection or Projection of Data in Data  
Frames

from pyspark import SparkContext

>>> from pyspark.sql import SparkSession

>>> spark = SparkSession.builder.master('yarn').appName('select or present data').getOrCreate()

>>> orders = spark.read.format('csv').schema('order\_id int, order\_date string, order\_customer\_id int, order\_status string').load('/user/saurabhbhanwala/retail\_db/orders')

>>> ordersItems = spark.read.format('csv').schema('order\_item\_id int, order\_item\_order\_id int, order\_item\_product\_id int, order\_item\_quantity int, order\_item\_subtotal float, order\_item\_product\_price float').load('/user/saurabhbhanwala/retail\_db/order\_items')

>>> orders.select(orders.order\_id, orders.order\_date).show()

+--------+--------------------+

|order\_id| order\_date|

+--------+--------------------+

| 1|2013-07-25 00:00:...|

| 2|2013-07-25 00:00:...|

| 3|2013-07-25 00:00:...|

| 4|2013-07-25 00:00:...|

| 5|2013-07-25 00:00:...|

| 6|2013-07-25 00:00:...|

| 7|2013-07-25 00:00:...|

| 8|2013-07-25 00:00:...|

| 9|2013-07-25 00:00:...|

| 10|2013-07-25 00:00:...|

| 11|2013-07-25 00:00:...|

| 12|2013-07-25 00:00:...|

| 13|2013-07-25 00:00:...|

| 14|2013-07-25 00:00:...|

| 15|2013-07-25 00:00:...|

| 16|2013-07-25 00:00:...|

| 17|2013-07-25 00:00:...|

| 18|2013-07-25 00:00:...|

| 19|2013-07-25 00:00:...|

| 20|2013-07-25 00:00:...|

+--------+--------------------+

only showing top 20 rows

from pyspark.sql.functions import substring

orders.select(substring('order\_date', 1, 7)).show()

+---------------------------+

|substring(order\_date, 1, 7)|

+---------------------------+

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

+---------------------------+

>>> orders.select(substring('order\_date', 1, 7).alias('order\_date')).show()

+----------+

|order\_date|

+----------+

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

+----------+

>>> orders.select('order\_id', 'order\_date', 'order\_customer\_id', 'order\_status', substring('order\_date', 1, 7).alias('order\_month')).show()

+--------+--------------------+-----------------+---------------+-----------+

|order\_id| order\_date|order\_customer\_id| order\_status|order\_month|

+--------+--------------------+-----------------+---------------+-----------+

| 1|2013-07-25 00:00:...| 11599| CLOSED| 2013-07|

| 2|2013-07-25 00:00:...| 256|PENDING\_PAYMENT| 2013-07|

| 3|2013-07-25 00:00:...| 12111| COMPLETE| 2013-07|

| 4|2013-07-25 00:00:...| 8827| CLOSED| 2013-07|

| 5|2013-07-25 00:00:...| 11318| COMPLETE| 2013-07|

| 6|2013-07-25 00:00:...| 7130| COMPLETE| 2013-07|

| 7|2013-07-25 00:00:...| 4530| COMPLETE| 2013-07|

| 8|2013-07-25 00:00:...| 2911| PROCESSING| 2013-07|

| 9|2013-07-25 00:00:...| 5657|PENDING\_PAYMENT| 2013-07|

| 10|2013-07-25 00:00:...| 5648|PENDING\_PAYMENT| 2013-07|

| 11|2013-07-25 00:00:...| 918| PAYMENT\_REVIEW| 2013-07|

| 12|2013-07-25 00:00:...| 1837| CLOSED| 2013-07|

| 13|2013-07-25 00:00:...| 9149|PENDING\_PAYMENT| 2013-07|

| 14|2013-07-25 00:00:...| 9842| PROCESSING| 2013-07|

| 15|2013-07-25 00:00:...| 2568| COMPLETE| 2013-07|

| 16|2013-07-25 00:00:...| 7276|PENDING\_PAYMENT| 2013-07|

| 17|2013-07-25 00:00:...| 2667| COMPLETE| 2013-07|

| 18|2013-07-25 00:00:...| 1205| CLOSED| 2013-07|

| 19|2013-07-25 00:00:...| 9488|PENDING\_PAYMENT| 2013-07|

| 20|2013-07-25 00:00:...| 9198| PROCESSING| 2013-07|

+--------+--------------------+-----------------+---------------+-----------+

only showing top 20 rows

>>> orders.withColumn('order\_month', substring('order\_date',1,7)).show()

+--------+--------------------+-----------------+---------------+-----------+

|order\_id| order\_date|order\_customer\_id| order\_status|order\_month|

+--------+--------------------+-----------------+---------------+-----------+

| 1|2013-07-25 00:00:...| 11599| CLOSED| 2013-07|

| 2|2013-07-25 00:00:...| 256|PENDING\_PAYMENT| 2013-07|

| 3|2013-07-25 00:00:...| 12111| COMPLETE| 2013-07|

| 4|2013-07-25 00:00:...| 8827| CLOSED| 2013-07|

| 5|2013-07-25 00:00:...| 11318| COMPLETE| 2013-07|

| 6|2013-07-25 00:00:...| 7130| COMPLETE| 2013-07|

| 7|2013-07-25 00:00:...| 4530| COMPLETE| 2013-07|

| 8|2013-07-25 00:00:...| 2911| PROCESSING| 2013-07|

| 9|2013-07-25 00:00:...| 5657|PENDING\_PAYMENT| 2013-07|

| 10|2013-07-25 00:00:...| 5648|PENDING\_PAYMENT| 2013-07|

| 11|2013-07-25 00:00:...| 918| PAYMENT\_REVIEW| 2013-07|

| 12|2013-07-25 00:00:...| 1837| CLOSED| 2013-07|

| 13|2013-07-25 00:00:...| 9149|PENDING\_PAYMENT| 2013-07|

| 14|2013-07-25 00:00:...| 9842| PROCESSING| 2013-07|

| 15|2013-07-25 00:00:...| 2568| COMPLETE| 2013-07|

| 16|2013-07-25 00:00:...| 7276|PENDING\_PAYMENT| 2013-07|

| 17|2013-07-25 00:00:...| 2667| COMPLETE| 2013-07|

| 18|2013-07-25 00:00:...| 1205| CLOSED| 2013-07|

| 19|2013-07-25 00:00:...| 9488|PENDING\_PAYMENT| 2013-07|

| 20|2013-07-25 00:00:...| 9198| PROCESSING| 2013-07|

+--------+--------------------+-----------------+---------------+-----------+

only showing top 20 rows

Help on method drop in module pyspark.sql.dataframe:

**drop**(self, \*cols) method of pyspark.sql.dataframe.DataFrame instance

Returns a new :class:`DataFrame` that drops the specified column.

This is a no-op if schema doesn't contain the given column name(s).

:param cols: a string name of the column to drop, or a

:class:`Column` to drop, or a list of string name of the columns to drop.

>>> df.drop('age').collect()

[Row(name=u'Alice'), Row(name=u'Bob')]

>>> df.drop(df.age).collect()

[Row(name=u'Alice'), Row(name=u'Bob')]

>>> df.join(df2, df.name == df2.name, 'inner').drop(df.name).collect()

[Row(age=5, height=85, name=u'Bob')]

>>> df.join(df2, df.name == df2.name, 'inner').drop(df2.name).collect()

[Row(age=5, name=u'Bob', height=85)]

>>> df.join(df2, 'name', 'inner').drop('age', 'height').collect()

[Row(name=u'Bob')]

orders.drop('order\_id').show()

+--------------------+-----------------+---------------+

| order\_date|order\_customer\_id| order\_status|

+--------------------+-----------------+---------------+

|2013-07-25 00:00:...| 11599| CLOSED|

|2013-07-25 00:00:...| 256|PENDING\_PAYMENT|

|2013-07-25 00:00:...| 12111| COMPLETE|

|2013-07-25 00:00:...| 8827| CLOSED|

|2013-07-25 00:00:...| 11318| COMPLETE|

|2013-07-25 00:00:...| 7130| COMPLETE|

|2013-07-25 00:00:...| 4530| COMPLETE|

|2013-07-25 00:00:...| 2911| PROCESSING|

|2013-07-25 00:00:...| 5657|PENDING\_PAYMENT|

|2013-07-25 00:00:...| 5648|PENDING\_PAYMENT|

|2013-07-25 00:00:...| 918| PAYMENT\_REVIEW|

|2013-07-25 00:00:...| 1837| CLOSED|

|2013-07-25 00:00:...| 9149|PENDING\_PAYMENT|

|2013-07-25 00:00:...| 9842| PROCESSING|

|2013-07-25 00:00:...| 2568| COMPLETE|

|2013-07-25 00:00:...| 7276|PENDING\_PAYMENT|

|2013-07-25 00:00:...| 2667| COMPLETE|

|2013-07-25 00:00:...| 1205| CLOSED|

|2013-07-25 00:00:...| 9488|PENDING\_PAYMENT|

|2013-07-25 00:00:...| 9198| PROCESSING|

+--------------------+-----------------+---------------+

only showing top 20 rows

orders.selectExpr('substring(order\_date,1 ,7) as order\_month')

DataFrame[order\_month: string]

>>> orders.selectExpr('substring(order\_date,1 ,7) as order\_month').show()

+-----------+

|order\_month|

+-----------+

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

| 2013-07|

+-----------+

only showing top 20 rows

filtering of dataframe

using where or filtering clause

Help on method filter in module pyspark.sql.dataframe:

**filter**(self, condition) method of pyspark.sql.dataframe.DataFrame instance

:func:`where` is an alias for :func:`filter`.

.. versionadded:: 1.3

Help on method filter in module pyspark.sql.dataframe:

**filter**(self, condition) method of pyspark.sql.dataframe.DataFrame instance

Filters rows using the given condition.

:func:`where` is an alias for :func:`filter`.

:param condition: a :class:`Column` of :class:`types.BooleanType`

or a string of SQL expression.

>>> df.filter(df.age > 3).collect()

[Row(age=5, name=u'Bob')]

>>> df.where(df.age == 2).collect()

[Row(age=2, name=u'Alice')]

>>> df.filter("age > 3").collect()

[Row(age=5, name=u'Bob')]

>>> df.where("age = 2").collect()

[Row(age=2, name=u'Alice')]

.. versionadded:: 1.3

>>> orders.filter(orders.order\_status == 'COMPLETE').show()

+--------+--------------------+-----------------+------------+

|order\_id| order\_date|order\_customer\_id|order\_status|

+--------+--------------------+-----------------+------------+

| 3|2013-07-25 00:00:...| 12111| COMPLETE|

| 5|2013-07-25 00:00:...| 11318| COMPLETE|

| 6|2013-07-25 00:00:...| 7130| COMPLETE|

| 7|2013-07-25 00:00:...| 4530| COMPLETE|

| 15|2013-07-25 00:00:...| 2568| COMPLETE|

| 17|2013-07-25 00:00:...| 2667| COMPLETE|

| 22|2013-07-25 00:00:...| 333| COMPLETE|

| 26|2013-07-25 00:00:...| 7562| COMPLETE|

| 28|2013-07-25 00:00:...| 656| COMPLETE|

| 32|2013-07-25 00:00:...| 3960| COMPLETE|

| 35|2013-07-25 00:00:...| 4840| COMPLETE|

| 45|2013-07-25 00:00:...| 2636| COMPLETE|

| 56|2013-07-25 00:00:...| 10519| COMPLETE|

| 63|2013-07-25 00:00:...| 1148| COMPLETE|

| 65|2013-07-25 00:00:...| 5903| COMPLETE|

| 67|2013-07-25 00:00:...| 1406| COMPLETE|

| 71|2013-07-25 00:00:...| 8646| COMPLETE|

| 72|2013-07-25 00:00:...| 4349| COMPLETE|

| 76|2013-07-25 00:00:...| 6898| COMPLETE|

| 80|2013-07-25 00:00:...| 3007| COMPLETE|

+--------+--------------------+-----------------+------------+

only showing top 20 rows

orders.filter("order\_status = 'COMPLETE'").show()

+--------+--------------------+-----------------+------------+

|order\_id| order\_date|order\_customer\_id|order\_status|

+--------+--------------------+-----------------+------------+

| 3|2013-07-25 00:00:...| 12111| COMPLETE|

| 5|2013-07-25 00:00:...| 11318| COMPLETE|

| 6|2013-07-25 00:00:...| 7130| COMPLETE|

| 7|2013-07-25 00:00:...| 4530| COMPLETE|

| 15|2013-07-25 00:00:...| 2568| COMPLETE|

| 17|2013-07-25 00:00:...| 2667| COMPLETE|

| 22|2013-07-25 00:00:...| 333| COMPLETE|

| 26|2013-07-25 00:00:...| 7562| COMPLETE|

| 28|2013-07-25 00:00:...| 656| COMPLETE|

| 32|2013-07-25 00:00:...| 3960| COMPLETE|

| 35|2013-07-25 00:00:...| 4840| COMPLETE|

| 45|2013-07-25 00:00:...| 2636| COMPLETE|

| 56|2013-07-25 00:00:...| 10519| COMPLETE|

| 63|2013-07-25 00:00:...| 1148| COMPLETE|

| 65|2013-07-25 00:00:...| 5903| COMPLETE|

| 67|2013-07-25 00:00:...| 1406| COMPLETE|

| 71|2013-07-25 00:00:...| 8646| COMPLETE|

| 72|2013-07-25 00:00:...| 4349| COMPLETE|

| 76|2013-07-25 00:00:...| 6898| COMPLETE|

| 80|2013-07-25 00:00:...| 3007| COMPLETE|

+--------+--------------------+-----------------+------------+

only showing top 20 rows

orders which are either complete or closed

orders.filter((orders.order\_status =="COMPLETE") | (orders.order\_status =="CLOSED")).show()

+--------+--------------------+-----------------+------------+

|order\_id| order\_date|order\_customer\_id|order\_status|

+--------+--------------------+-----------------+------------+

| 1|2013-07-25 00:00:...| 11599| CLOSED|

| 3|2013-07-25 00:00:...| 12111| COMPLETE|

| 4|2013-07-25 00:00:...| 8827| CLOSED|

| 5|2013-07-25 00:00:...| 11318| COMPLETE|

| 6|2013-07-25 00:00:...| 7130| COMPLETE|

| 7|2013-07-25 00:00:...| 4530| COMPLETE|

| 12|2013-07-25 00:00:...| 1837| CLOSED|

| 15|2013-07-25 00:00:...| 2568| COMPLETE|

| 17|2013-07-25 00:00:...| 2667| COMPLETE|

| 18|2013-07-25 00:00:...| 1205| CLOSED|

| 22|2013-07-25 00:00:...| 333| COMPLETE|

| 24|2013-07-25 00:00:...| 11441| CLOSED|

| 25|2013-07-25 00:00:...| 9503| CLOSED|

| 26|2013-07-25 00:00:...| 7562| COMPLETE|

| 28|2013-07-25 00:00:...| 656| COMPLETE|

| 32|2013-07-25 00:00:...| 3960| COMPLETE|

| 35|2013-07-25 00:00:...| 4840| COMPLETE|

| 37|2013-07-25 00:00:...| 5863| CLOSED|

| 45|2013-07-25 00:00:...| 2636| COMPLETE|

| 51|2013-07-25 00:00:...| 12271| CLOSED|

+--------+--------------------+-----------------+------------+

only showing top 20 rows

>>> orders.filter("order\_status = 'COMPLETE' or order\_status =='CLOSED'").show()

+--------+--------------------+-----------------+------------+

|order\_id| order\_date|order\_customer\_id|order\_status|

+--------+--------------------+-----------------+------------+

| 1|2013-07-25 00:00:...| 11599| CLOSED|

| 3|2013-07-25 00:00:...| 12111| COMPLETE|

| 4|2013-07-25 00:00:...| 8827| CLOSED|

| 5|2013-07-25 00:00:...| 11318| COMPLETE|

| 6|2013-07-25 00:00:...| 7130| COMPLETE|

| 7|2013-07-25 00:00:...| 4530| COMPLETE|

| 12|2013-07-25 00:00:...| 1837| CLOSED|

| 15|2013-07-25 00:00:...| 2568| COMPLETE|

| 17|2013-07-25 00:00:...| 2667| COMPLETE|

| 18|2013-07-25 00:00:...| 1205| CLOSED|

| 22|2013-07-25 00:00:...| 333| COMPLETE|

| 24|2013-07-25 00:00:...| 11441| CLOSED|

| 25|2013-07-25 00:00:...| 9503| CLOSED|

| 26|2013-07-25 00:00:...| 7562| COMPLETE|

| 28|2013-07-25 00:00:...| 656| COMPLETE|

| 32|2013-07-25 00:00:...| 3960| COMPLETE|

| 35|2013-07-25 00:00:...| 4840| COMPLETE|

| 37|2013-07-25 00:00:...| 5863| CLOSED|

| 45|2013-07-25 00:00:...| 2636| COMPLETE|

| 51|2013-07-25 00:00:...| 12271| CLOSED|

+--------+--------------------+-----------------+------------+

>>> orders.filter(orders.order\_status.isin('COMPLETE', 'CLOSED')).show()

+--------+--------------------+-----------------+------------+

|order\_id| order\_date|order\_customer\_id|order\_status|

+--------+--------------------+-----------------+------------+

| 1|2013-07-25 00:00:...| 11599| CLOSED|

| 3|2013-07-25 00:00:...| 12111| COMPLETE|

| 4|2013-07-25 00:00:...| 8827| CLOSED|

| 5|2013-07-25 00:00:...| 11318| COMPLETE|

| 6|2013-07-25 00:00:...| 7130| COMPLETE|

| 7|2013-07-25 00:00:...| 4530| COMPLETE|

| 12|2013-07-25 00:00:...| 1837| CLOSED|

| 15|2013-07-25 00:00:...| 2568| COMPLETE|

| 17|2013-07-25 00:00:...| 2667| COMPLETE|

| 18|2013-07-25 00:00:...| 1205| CLOSED|

| 22|2013-07-25 00:00:...| 333| COMPLETE|

| 24|2013-07-25 00:00:...| 11441| CLOSED|

| 25|2013-07-25 00:00:...| 9503| CLOSED|

| 26|2013-07-25 00:00:...| 7562| COMPLETE|

| 28|2013-07-25 00:00:...| 656| COMPLETE|

| 32|2013-07-25 00:00:...| 3960| COMPLETE|

| 35|2013-07-25 00:00:...| 4840| COMPLETE|

| 37|2013-07-25 00:00:...| 5863| CLOSED|

| 45|2013-07-25 00:00:...| 2636| COMPLETE|

| 51|2013-07-25 00:00:...| 12271| CLOSED|

+--------+--------------------+-----------------+------------+

only showing top 20 rows

>>> orders.filter("order\_status in ('CLOSED', 'COMPLETE')").show()

+--------+--------------------+-----------------+------------+

|order\_id| order\_date|order\_customer\_id|order\_status|

+--------+--------------------+-----------------+------------+

| 1|2013-07-25 00:00:...| 11599| CLOSED|

| 3|2013-07-25 00:00:...| 12111| COMPLETE|

| 4|2013-07-25 00:00:...| 8827| CLOSED|

| 5|2013-07-25 00:00:...| 11318| COMPLETE|

| 6|2013-07-25 00:00:...| 7130| COMPLETE|

| 7|2013-07-25 00:00:...| 4530| COMPLETE|

| 12|2013-07-25 00:00:...| 1837| CLOSED|

| 15|2013-07-25 00:00:...| 2568| COMPLETE|

| 17|2013-07-25 00:00:...| 2667| COMPLETE|

| 18|2013-07-25 00:00:...| 1205| CLOSED|

| 22|2013-07-25 00:00:...| 333| COMPLETE|

| 24|2013-07-25 00:00:...| 11441| CLOSED|

| 25|2013-07-25 00:00:...| 9503| CLOSED|

| 26|2013-07-25 00:00:...| 7562| COMPLETE|

| 28|2013-07-25 00:00:...| 656| COMPLETE|

| 32|2013-07-25 00:00:...| 3960| COMPLETE|

| 35|2013-07-25 00:00:...| 4840| COMPLETE|

| 37|2013-07-25 00:00:...| 5863| CLOSED|

| 45|2013-07-25 00:00:...| 2636| COMPLETE|

| 51|2013-07-25 00:00:...| 12271| CLOSED|

+--------+--------------------+-----------------+------------+

only showing top 20 rows

orders which are either complete or closed and in month on aug 2013

orders.filter((orders.order\_status.isin('COMPLETE', 'CLOSED')) & (orders.order\_date.like('2013-08%'))).show()

+--------+--------------------+-----------------+------------+

|order\_id| order\_date|order\_customer\_id|order\_status|

+--------+--------------------+-----------------+------------+

| 1297|2013-08-01 00:00:...| 11607| COMPLETE|

| 1298|2013-08-01 00:00:...| 5105| CLOSED|

| 1299|2013-08-01 00:00:...| 7802| COMPLETE|

| 1302|2013-08-01 00:00:...| 1695| COMPLETE|

| 1304|2013-08-01 00:00:...| 2059| COMPLETE|

| 1305|2013-08-01 00:00:...| 3844| COMPLETE|

| 1307|2013-08-01 00:00:...| 4474| COMPLETE|

| 1309|2013-08-01 00:00:...| 2367| CLOSED|

| 1312|2013-08-01 00:00:...| 12291| COMPLETE|

| 1314|2013-08-01 00:00:...| 10993| COMPLETE|

| 1315|2013-08-01 00:00:...| 5660| COMPLETE|

| 1318|2013-08-01 00:00:...| 4212| COMPLETE|

| 1319|2013-08-01 00:00:...| 3966| COMPLETE|

| 1320|2013-08-01 00:00:...| 12270| COMPLETE|

| 1321|2013-08-01 00:00:...| 800| COMPLETE|

| 1322|2013-08-01 00:00:...| 9264| COMPLETE|

| 1323|2013-08-01 00:00:...| 7422| COMPLETE|

| 1324|2013-08-01 00:00:...| 4600| COMPLETE|

| 1327|2013-08-01 00:00:...| 9526| COMPLETE|

| 1329|2013-08-01 00:00:...| 6070| CLOSED|

+--------+--------------------+-----------------+------------+

only showing top 20 rows

orders.filter((orders.order\_status.isin('COMPLETE', 'CLOSED')).\_\_and\_\_(orders.order\_date.like('2013-08%'))).show()

+--------+--------------------+-----------------+------------+

|order\_id| order\_date|order\_customer\_id|order\_status|

+--------+--------------------+-----------------+------------+

| 1297|2013-08-01 00:00:...| 11607| COMPLETE|

| 1298|2013-08-01 00:00:...| 5105| CLOSED|

| 1299|2013-08-01 00:00:...| 7802| COMPLETE|

| 1302|2013-08-01 00:00:...| 1695| COMPLETE|

| 1304|2013-08-01 00:00:...| 2059| COMPLETE|

| 1305|2013-08-01 00:00:...| 3844| COMPLETE|

| 1307|2013-08-01 00:00:...| 4474| COMPLETE|

| 1309|2013-08-01 00:00:...| 2367| CLOSED|

| 1312|2013-08-01 00:00:...| 12291| COMPLETE|

| 1314|2013-08-01 00:00:...| 10993| COMPLETE|

| 1315|2013-08-01 00:00:...| 5660| COMPLETE|

| 1318|2013-08-01 00:00:...| 4212| COMPLETE|

| 1319|2013-08-01 00:00:...| 3966| COMPLETE|

| 1320|2013-08-01 00:00:...| 12270| COMPLETE|

| 1321|2013-08-01 00:00:...| 800| COMPLETE|

| 1322|2013-08-01 00:00:...| 9264| COMPLETE|

| 1323|2013-08-01 00:00:...| 7422| COMPLETE|

| 1324|2013-08-01 00:00:...| 4600| COMPLETE|

| 1327|2013-08-01 00:00:...| 9526| COMPLETE|

| 1329|2013-08-01 00:00:...| 6070| CLOSED|

+--------+--------------------+-----------------+------------+

only showing top 20 rows

>>> orders.filter("order\_status in ('CLOSED', 'COMPLETE') and order\_date like '2013-08%'").show()

+--------+--------------------+-----------------+------------+

|order\_id| order\_date|order\_customer\_id|order\_status|

+--------+--------------------+-----------------+------------+

| 1297|2013-08-01 00:00:...| 11607| COMPLETE|

| 1298|2013-08-01 00:00:...| 5105| CLOSED|

| 1299|2013-08-01 00:00:...| 7802| COMPLETE|

| 1302|2013-08-01 00:00:...| 1695| COMPLETE|

| 1304|2013-08-01 00:00:...| 2059| COMPLETE|

| 1305|2013-08-01 00:00:...| 3844| COMPLETE|

| 1307|2013-08-01 00:00:...| 4474| COMPLETE|

| 1309|2013-08-01 00:00:...| 2367| CLOSED|

| 1312|2013-08-01 00:00:...| 12291| COMPLETE|

| 1314|2013-08-01 00:00:...| 10993| COMPLETE|

| 1315|2013-08-01 00:00:...| 5660| COMPLETE|

| 1318|2013-08-01 00:00:...| 4212| COMPLETE|

| 1319|2013-08-01 00:00:...| 3966| COMPLETE|

| 1320|2013-08-01 00:00:...| 12270| COMPLETE|

| 1321|2013-08-01 00:00:...| 800| COMPLETE|

| 1322|2013-08-01 00:00:...| 9264| COMPLETE|

| 1323|2013-08-01 00:00:...| 7422| COMPLETE|

| 1324|2013-08-01 00:00:...| 4600| COMPLETE|

| 1327|2013-08-01 00:00:...| 9526| COMPLETE|

| 1329|2013-08-01 00:00:...| 6070| CLOSED|

+--------+--------------------+-----------------+------------+

only showing top 20 rows

>>> orders.filter("order\_status in ('CLOSED', 'COMPLETE') and order\_date like '2013-08%'").count()

2517

>>> ordersItems.select('order\_item\_subtotal', 'order\_item\_quantity', 'order\_item\_product\_price').show()

+-------------------+-------------------+------------------------+

|order\_item\_subtotal|order\_item\_quantity|order\_item\_product\_price|

+-------------------+-------------------+------------------------+

| 299.98| 1| 299.98|

| 199.99| 1| 199.99|

| 250.0| 5| 50.0|

| 129.99| 1| 129.99|

| 49.98| 2| 24.99|

| 299.95| 5| 59.99|

| 150.0| 3| 50.0|

| 199.92| 4| 49.98|

| 299.98| 1| 299.98|

| 299.95| 5| 59.99|

| 99.96| 2| 49.98|

| 299.98| 1| 299.98|

| 129.99| 1| 129.99|

| 199.99| 1| 199.99|

| 299.98| 1| 299.98|

| 79.95| 5| 15.99|

| 179.97| 3| 59.99|

| 299.95| 5| 59.99|

| 199.92| 4| 49.98|

| 50.0| 1| 50.0|

+-------------------+-------------------+---------------

ordersItems.select('order\_item\_subtotal', 'order\_item\_quantity', 'order\_item\_product\_price').where(ordersItems.order\_item\_subtotal != ordersItems.order\_item\_quantity \* ordersItems.order\_item\_product\_price).show()

ordersItems.select('order\_item\_subtotal', 'order\_item\_quantity', 'order\_item\_product\_price').where(ordersItems.order\_item\_subtotal != round(ordersItems.order\_item\_quantity \* ordersItems.order\_item\_product\_price)).show()

ordersItems.select('order\_item\_subtotal', 'order\_item\_quantity', 'order\_item\_product\_price').where(ordersItems.order\_item\_subtotal != round((ordersItems.order\_item\_quantity \* ordersItems.order\_item\_product\_price),2)).show()

+-------------------+-------------------+------------------------+

|order\_item\_subtotal|order\_item\_quantity|order\_item\_product\_price|

+-------------------+-------------------+------------------------+

+-------------------+-------------------+------------------------+

Get all orders which are placed on first of every month

>>> from pyspark.sql.functions import min

>>> orders.select(min(orders.order\_date)).show()

+--------------------+

| min(order\_date)|

+--------------------+

|2013-07-25 00:00:...|

+--------------------+

>>> from pyspark.sql.functions import max

>>> orders.select(max(orders.order\_date)).show()

+--------------------+

| max(order\_date)|

+--------------------+

|2014-07-24 00:00:...|

+--------------------+

>>> from pyspark.sql.functions import date\_format

>>> orders.select(date\_format(orders.order\_date, 'YYYY')).show()

+-----------------------------+

|date\_format(order\_date, YYYY)|

+-----------------------------+

| 2013|

| 2013|

| 2013|

| 2013|

| 2013|

| 2013|

| 2013|

| 2013|

| 2013|

| 2013|

| 2013|

| 2013|

| 2013|

| 2013|

| 2013|

| 2013|

| 2013|

| 2013|

| 2013|

| 2013|

+-----------------------------+

only showing top 20 rows

>>> orders.select(date\_format(orders.order\_date, 'MM')).show()

+---------------------------+

|date\_format(order\_date, MM)|

+---------------------------+

| 07|

| 07|

| 07|

| 07|

| 07|

| 07|

| 07|

| 07|

| 07|

| 07|

| 07|

| 07|

| 07|

| 07|

| 07|

| 07|

| 07|

| 07|

| 07|

| 07|

+---------------------------+

only showing top 20 rows

>>> orders.select(date\_format(orders.order\_date, 'DD')).show()

+---------------------------+

|date\_format(order\_date, DD)|

+---------------------------+

| 206|

| 206|

| 206|

| 206|

| 206|

| 206|

| 206|

| 206|

| 206|

| 206|

| 206|

| 206|

| 206|

| 206|

| 206|

| 206|

| 206|

| 206|

| 206|

| 206|

+---------------------------+

only showing top 20 rows

>>> orders.select(date\_format(orders.order\_date, 'dd')).show()

+---------------------------+

|date\_format(order\_date, dd)|

+---------------------------+

| 25|

| 25|

| 25|

| 25|

| 25|

| 25|

| 25|

| 25|

| 25|

| 25|

| 25|

| 25|

| 25|

| 25|

| 25|

| 25|

| 25|

| 25|

| 25|

| 25|

+---------------------------+

only showing top 20 rows

orders.filter(date\_format(orders.order\_date, 'dd') == '01').select(orders.order\_date).distinct().count()

12

>>> orders.filter(date\_format(orders.order\_date, 'dd') == '01').select(orders.order\_date).distinct().count()

12

>>> orders.filter(date\_format(orders.order\_date, 'dd') == '01').show()

+--------+--------------------+-----------------+---------------+

|order\_id| order\_date|order\_customer\_id| order\_status|

+--------+--------------------+-----------------+---------------+

| 1297|2013-08-01 00:00:...| 11607| COMPLETE|

| 1298|2013-08-01 00:00:...| 5105| CLOSED|

| 1299|2013-08-01 00:00:...| 7802| COMPLETE|

| 1300|2013-08-01 00:00:...| 553|PENDING\_PAYMENT|

| 1301|2013-08-01 00:00:...| 1604|PENDING\_PAYMENT|

| 1302|2013-08-01 00:00:...| 1695| COMPLETE|

| 1303|2013-08-01 00:00:...| 7018| PROCESSING|

| 1304|2013-08-01 00:00:...| 2059| COMPLETE|

| 1305|2013-08-01 00:00:...| 3844| COMPLETE|

| 1306|2013-08-01 00:00:...| 11672|PENDING\_PAYMENT|

| 1307|2013-08-01 00:00:...| 4474| COMPLETE|

| 1308|2013-08-01 00:00:...| 11645| PENDING|

| 1309|2013-08-01 00:00:...| 2367| CLOSED|

| 1310|2013-08-01 00:00:...| 5602| PENDING|

| 1311|2013-08-01 00:00:...| 5396|PENDING\_PAYMENT|

| 1312|2013-08-01 00:00:...| 12291| COMPLETE|

| 1313|2013-08-01 00:00:...| 3471| CANCELED|

| 1314|2013-08-01 00:00:...| 10993| COMPLETE|

| 1315|2013-08-01 00:00:...| 5660| COMPLETE|

| 1316|2013-08-01 00:00:...| 6376|PENDING\_PAYMENT|

+--------+--------------------+-----------------+---------------+

only showing top 20 rows

joining multiple datasets

orders and orderItems are transactional tables and customer products and departments are metadata tables

>>> from pyspark import SparkContext

>>> from pyspark.sql import SparkSession

>>> spark = SparkSession.builder.master('yarn').appName('select or present data').getOrCreate()

>>> orders = spark.read.format('csv').schema('order\_id int, order\_date string, order\_customer\_id int, order\_status string').load('/user/saurabhbhanwala/retail\_db/orders')

>>> ordersItems = spark.read.format('csv').schema('order\_item\_id int, order\_item\_order\_id int, order\_item\_product\_id int, order\_item\_quantity int, order\_item\_subtotal float, order\_item\_product\_price float').load('/user/saurabhbhanwala/retail\_db/order\_items')

Help on method join in module pyspark.sql.dataframe:

**join**(self, other, on=None, how=None) method of pyspark.sql.dataframe.DataFrame instance

Joins with another :class:`DataFrame`, using the given join expression.

:param other: Right side of the join

:param on: a string for the join column name, a list of column names,

a join expression (Column), or a list of Columns.

If `on` is a string or a list of strings indicating the name of the join column(s),

the column(s) must exist on both sides, and this performs an equi-join.

:param how: str, default ``inner``. Must be one of: ``inner``, ``cross``, ``outer``,

``full``, ``full\_outer``, ``left``, ``left\_outer``, ``right``, ``right\_outer``,

``left\_semi``, and ``left\_anti``.

The following performs a full outer join between ``df1`` and ``df2``.

>>> df.join(df2, df.name == df2.name, 'outer').select(df.name, df2.height).collect()

[Row(name=None, height=80), Row(name=u'Bob', height=85), Row(name=u'Alice', height=None)]

>>> df.join(df2, 'name', 'outer').select('name', 'height').collect()

[Row(name=u'Tom', height=80), Row(name=u'Bob', height=85), Row(name=u'Alice', height=None)]

>>> cond = [df.name == df3.name, df.age == df3.age]

>>> df.join(df3, cond, 'outer').select(df.name, df3.age).collect()

[Row(name=u'Alice', age=2), Row(name=u'Bob', age=5)]

>>> df.join(df2, 'name').select(df.name, df2.height).collect()

[Row(name=u'Bob', height=85)]

>>> df.join(df4, ['name', 'age']).select(df.name, df.age).collect()

[Row(name=u'Bob', age=5)]

>>> orders.where("order\_status in ('COMPLETE', 'CLOSED')").count()

30455

ordersFiltered = orders.where("order\_status in ('COMPLETE', 'CLOSED')")

>>> type(ordersFiltered)

<class 'pyspark.sql.dataframe.DataFrame'>

>>> ordersFiltered.show()

+--------+--------------------+-----------------+------------+

|order\_id| order\_date|order\_customer\_id|order\_status|

+--------+--------------------+-----------------+------------+

| 1|2013-07-25 00:00:...| 11599| CLOSED|

| 3|2013-07-25 00:00:...| 12111| COMPLETE|

| 4|2013-07-25 00:00:...| 8827| CLOSED|

| 5|2013-07-25 00:00:...| 11318| COMPLETE|

| 6|2013-07-25 00:00:...| 7130| COMPLETE|

| 7|2013-07-25 00:00:...| 4530| COMPLETE|

| 12|2013-07-25 00:00:...| 1837| CLOSED|

| 15|2013-07-25 00:00:...| 2568| COMPLETE|

| 17|2013-07-25 00:00:...| 2667| COMPLETE|

| 18|2013-07-25 00:00:...| 1205| CLOSED|

| 22|2013-07-25 00:00:...| 333| COMPLETE|

| 24|2013-07-25 00:00:...| 11441| CLOSED|

| 25|2013-07-25 00:00:...| 9503| CLOSED|

| 26|2013-07-25 00:00:...| 7562| COMPLETE|

| 28|2013-07-25 00:00:...| 656| COMPLETE|

| 32|2013-07-25 00:00:...| 3960| COMPLETE|

| 35|2013-07-25 00:00:...| 4840| COMPLETE|

| 37|2013-07-25 00:00:...| 5863| CLOSED|

| 45|2013-07-25 00:00:...| 2636| COMPLETE|

| 51|2013-07-25 00:00:...| 12271| CLOSED|

+--------+--------------------+-----------------+------------+

only showing top 20 rows

>>> ordersjoin = ordersFiltered.join(ordersItems, ordersFiltered.order\_id == ordersItems.order\_item\_order\_id, 'inner')

>>> ordersjoin.show()

+--------+--------------------+-----------------+------------+-------------+-------------------+---------------------+-------------------+-------------------+------------------------+

|order\_id| order\_date|order\_customer\_id|order\_status|order\_item\_id|order\_item\_order\_id|order\_item\_product\_id|order\_item\_quantity|order\_item\_subtotal|order\_item\_product\_price|

+--------+--------------------+-----------------+------------+-------------+-------------------+---------------------+-------------------+-------------------+------------------------+

| 1|2013-07-25 00:00:...| 11599| CLOSED| 1| 1| 957| 1| 299.98| 299.98|

| 4|2013-07-25 00:00:...| 8827| CLOSED| 5| 4| 897| 2| 49.98| 24.99|

| 4|2013-07-25 00:00:...| 8827| CLOSED| 6| 4| 365| 5| 299.95| 59.99|

| 4|2013-07-25 00:00:...| 8827| CLOSED| 7| 4| 502| 3| 150.0| 50.0|

| 4|2013-07-25 00:00:...| 8827| CLOSED| 8| 4| 1014| 4| 199.92| 49.98|

| 5|2013-07-25 00:00:...| 11318| COMPLETE| 9| 5| 957| 1| 299.98| 299.98|

| 5|2013-07-25 00:00:...| 11318| COMPLETE| 10| 5| 365| 5| 299.95| 59.99|

| 5|2013-07-25 00:00:...| 11318| COMPLETE| 11| 5| 1014| 2| 99.96| 49.98|

| 5|2013-07-25 00:00:...| 11318| COMPLETE| 12| 5| 957| 1| 299.98| 299.98|

| 5|2013-07-25 00:00:...| 11318| COMPLETE| 13| 5| 403| 1| 129.99| 129.99|

| 7|2013-07-25 00:00:...| 4530| COMPLETE| 14| 7| 1073| 1| 199.99| 199.99|

| 7|2013-07-25 00:00:...| 4530| COMPLETE| 15| 7| 957| 1| 299.98| 299.98|

| 7|2013-07-25 00:00:...| 4530| COMPLETE| 16| 7| 926| 5| 79.95| 15.99|

| 12|2013-07-25 00:00:...| 1837| CLOSED| 34| 12| 957| 1| 299.98| 299.98|

| 12|2013-07-25 00:00:...| 1837| CLOSED| 35| 12| 134| 4| 100.0| 25.0|

| 12|2013-07-25 00:00:...| 1837| CLOSED| 36| 12| 1014| 3| 149.94| 49.98|

| 12|2013-07-25 00:00:...| 1837| CLOSED| 37| 12| 191| 5| 499.95| 99.99|

| 12|2013-07-25 00:00:...| 1837| CLOSED| 38| 12| 502| 5| 250.0| 50.0|

| 15|2013-07-25 00:00:...| 2568| COMPLETE| 43| 15| 502| 1| 50.0| 50.0|

| 15|2013-07-25 00:00:...| 2568| COMPLETE| 44| 15| 1073| 1| 199.99| 199.99|

+--------+--------------------+-----------------+------------+-------------+-------------------+---------------------+-------------------+-------------------+------------------------+

only showing top 20 rows

>>> ordersjoin = ordersFiltered.join(ordersItems, ordersFiltered.order\_id == ordersItems.order\_item\_order\_id, 'inner').select(ordersFiltered.order\_id, ordersItems.order\_item\_subtotal)

>>> ordersjoin.show()

+--------+-------------------+

|order\_id|order\_item\_subtotal|

+--------+-------------------+

| 1| 299.98|

| 4| 49.98|

| 4| 299.95|

| 4| 150.0|

| 4| 199.92|

| 5| 299.98|

| 5| 299.95|

| 5| 99.96|

| 5| 299.98|

| 5| 129.99|

| 7| 199.99|

| 7| 299.98|

| 7| 79.95|

| 12| 299.98|

| 12| 100.0|

| 12| 149.94|

| 12| 499.95|

| 12| 250.0|

| 15| 50.0|

| 15| 199.99|

+--------+-------------------+

only showing top 20 rows

>>> orders.count()

68883

>>> ordersItems.count()

172198

>>> orders.select('order\_id').distinct().count()

68883

>>> orders.select('order\_id').distinct().count()

68883

>>> ordersleftOuterJoin = orders.join(ordersItems, orders.order\_id == ordersItems.order\_item\_order\_id, 'left')

>>> ordersleftOuterJoin.printSchema()

root

|-- order\_id: integer (nullable = true)

|-- order\_date: string (nullable = true)

|-- order\_customer\_id: integer (nullable = true)

|-- order\_status: string (nullable = true)

|-- order\_item\_id: integer (nullable = true)

|-- order\_item\_order\_id: integer (nullable = true)

|-- order\_item\_product\_id: integer (nullable = true)

|-- order\_item\_quantity: integer (nullable = true)

|-- order\_item\_subtotal: float (nullable = true)

|-- order\_item\_product\_price: float (nullable = true)

>>> ordersleftOuterJoin.show()

+--------+--------------------+-----------------+---------------+-------------+-------------------+---------------------+-------------------+-------------------+------------------------+

|order\_id| order\_date|order\_customer\_id| order\_status|order\_item\_id|order\_item\_order\_id|order\_item\_product\_id|order\_item\_quantity|order\_item\_subtotal|order\_item\_product\_price|

+--------+--------------------+-----------------+---------------+-------------+-------------------+---------------------+-------------------+-------------------+------------------------+

| 1|2013-07-25 00:00:...| 11599| CLOSED| 1| 1| 957| 1| 299.98| 299.98|

| 2|2013-07-25 00:00:...| 256|PENDING\_PAYMENT| 4| 2| 403| 1| 129.99| 129.99|

| 2|2013-07-25 00:00:...| 256|PENDING\_PAYMENT| 3| 2| 502| 5| 250.0| 50.0|

| 2|2013-07-25 00:00:...| 256|PENDING\_PAYMENT| 2| 2| 1073| 1| 199.99| 199.99|

| 3|2013-07-25 00:00:...| 12111| COMPLETE| null| null| null| null| null| null|

| 4|2013-07-25 00:00:...| 8827| CLOSED| 8| 4| 1014| 4| 199.92| 49.98|

| 4|2013-07-25 00:00:...| 8827| CLOSED| 7| 4| 502| 3| 150.0| 50.0|

| 4|2013-07-25 00:00:...| 8827| CLOSED| 6| 4| 365| 5| 299.95| 59.99|

| 4|2013-07-25 00:00:...| 8827| CLOSED| 5| 4| 897| 2| 49.98| 24.99|

| 5|2013-07-25 00:00:...| 11318| COMPLETE| 13| 5| 403| 1| 129.99| 129.99|

| 5|2013-07-25 00:00:...| 11318| COMPLETE| 12| 5| 957| 1| 299.98| 299.98|

| 5|2013-07-25 00:00:...| 11318| COMPLETE| 11| 5| 1014| 2| 99.96| 49.98|

| 5|2013-07-25 00:00:...| 11318| COMPLETE| 10| 5| 365| 5| 299.95| 59.99|

| 5|2013-07-25 00:00:...| 11318| COMPLETE| 9| 5| 957| 1| 299.98| 299.98|

| 6|2013-07-25 00:00:...| 7130| COMPLETE| null| null| null| null| null| null|

| 7|2013-07-25 00:00:...| 4530| COMPLETE| 16| 7| 926| 5| 79.95| 15.99|

| 7|2013-07-25 00:00:...| 4530| COMPLETE| 15| 7| 957| 1| 299.98| 299.98|

| 7|2013-07-25 00:00:...| 4530| COMPLETE| 14| 7| 1073| 1| 199.99| 199.99|

| 8|2013-07-25 00:00:...| 2911| PROCESSING| 20| 8| 502| 1| 50.0| 50.0|

| 8|2013-07-25 00:00:...| 2911| PROCESSING| 19| 8| 1014| 4| 199.92| 49.98|

+--------+--------------------+-----------------+---------------+-------------+-------------------+---------------------+-------------------+-------------------+------------------------+

only showing top 20 rows

>>> ordersleftOuterJoin.where('order\_item\_order\_id is null').count()

11452

. Perform Aggregations using Data Frames and grouping

> orders.select('order\_status').distinct().show()

+---------------+

| order\_status|

+---------------+

|PENDING\_PAYMENT|

| COMPLETE|

| ON\_HOLD|

| PAYMENT\_REVIEW|

| PROCESSING|

| CLOSED|

|SUSPECTED\_FRAUD|

| PENDING|

| CANCELED|

+---------------+

SyntaxError: invalid syntax

>>> orders.select(countDistinct('order\_status')).show()

+----------------------------+

|count(DISTINCT order\_status)|

+----------------------------+

| 9|

+----------------------------+

>>> orders.select(countDistinct('order\_status').alias('status\_count')).show()

+------------+

|status\_count|

+------------+

| 9|

+------------+

>>> ordersItems.where('order\_item\_order\_id =2').select(sum('order\_item\_subtotal')).show()

+------------------------+

|sum(order\_item\_subtotal)|

+------------------------+

| 579.9800109863281|

+------------------------+

>>> ordersItems.where('order\_item\_order\_id =2').select(round(sum('order\_item\_subtotal'),2)).show()

+----------------------------------+

|round(sum(order\_item\_subtotal), 2)|

+----------------------------------+

| 579.98|

+----------------------------------+

>>> ordersItems.groupBy('order\_item\_order\_id').sum('order\_item\_subtotal').show()

+-------------------+------------------------+

|order\_item\_order\_id|sum(order\_item\_subtotal)|

+-------------------+------------------------+

| 148| 479.99000549316406|

| 463| 829.9200096130371|

| 471| 169.98000717163086|

| 496| 441.950008392334|

| 1088| 249.97000885009766|

| 1580| 299.95001220703125|

| 1591| 439.8599967956543|

| 1645| 1509.790023803711|

| 2366| 299.9700012207031|

| 2659| 724.9100151062012|

| 2866| 569.960018157959|

| 3175| 209.97000122070312|

| 3749| 143.97000122070312|

| 3794| 299.95001220703125|

| 3918| 829.9300155639648|

| 3997| 579.9500122070312|

| 4101| 129.99000549316406|

| 4519| 79.9800033569336|

| 4818| 399.9800109863281|

| 4900| 179.9700050354004|

+-------------------+------------------------+

only showing top 20 rows

>>> ordersItems.groupBy('order\_item\_order\_id').agg(sum('order\_item\_subtotal').alias('revenue')).show()

+-------------------+------------------+

|order\_item\_order\_id| revenue|

+-------------------+------------------+

| 61793|1299.8700218200684|

| 62015|1139.8800354003906|

| 62680| 379.9600067138672|

| 62985|249.89999389648438|

| 63087|349.96001052856445|

| 63106| 1299.910026550293|

| 63155| 909.9200134277344|

| 63271| 749.9800109863281|

| 63574|199.99000549316406|

| 63645| 1269.900032043457|

| 63964|359.97000885009766|

| 64121|229.99000549316406|

| 64519|1179.9400253295898|

| 64628| 869.9100112915039|

| 64859| 399.9700012207031|

| 65220| 749.9800109863281|

| 65241| 1099.910026550293|

| 65251|1149.9300231933594|

| 65408| 598.9800109863281|

| 65867| 909.8900260925293|

+-------------------+------------------+

only showing top 20 rows

>>> orders.groupBy('order\_status').count().show()

+---------------+-----+

| order\_status|count|

+---------------+-----+

|PENDING\_PAYMENT|15030|

| COMPLETE|22899|

| ON\_HOLD| 3798|

| PAYMENT\_REVIEW| 729|

| PROCESSING| 8275|

| CLOSED| 7556|

|SUSPECTED\_FRAUD| 1558|

| PENDING| 7610|

| CANCELED| 1428|

+---------------+-----+

>>> orders.groupBy('order\_status').agg(count('order\_status')).show()

+---------------+-------------------+

| order\_status|count(order\_status)|

+---------------+-------------------+

|PENDING\_PAYMENT| 15030|

| COMPLETE| 22899|

| ON\_HOLD| 3798|

| PAYMENT\_REVIEW| 729|

| PROCESSING| 8275|

| CLOSED| 7556|

|SUSPECTED\_FRAUD| 1558|

| PENDING| 7610|

| CANCELED| 1428|

+---------------+-------------------+

>>> orders.groupBy('order\_status').agg(count('order\_status').alias('count\_status')).show()

+---------------+------------+

| order\_status|count\_status|

+---------------+------------+

|PENDING\_PAYMENT| 15030|

| COMPLETE| 22899|

| ON\_HOLD| 3798|

| PAYMENT\_REVIEW| 729|

| PROCESSING| 8275|

| CLOSED| 7556|

|SUSPECTED\_FRAUD| 1558|

| PENDING| 7610|

| CANCELED| 1428|

+---------------+------------+

>>> orderJoins.show()

+--------+--------------------+-----------------+---------------+-------------+-------------------+---------------------+-------------------+-------------------+------------------------+

|order\_id| order\_date|order\_customer\_id| order\_status|order\_item\_id|order\_item\_order\_id|order\_item\_product\_id|order\_item\_quantity|order\_item\_subtotal|order\_item\_product\_price|

+--------+--------------------+-----------------+---------------+-------------+-------------------+---------------------+-------------------+-------------------+------------------------+

| 1|2013-07-25 00:00:...| 11599| CLOSED| 1| 1| 957| 1| 299.98| 299.98|

| 2|2013-07-25 00:00:...| 256|PENDING\_PAYMENT| 2| 2| 1073| 1| 199.99| 199.99|

| 2|2013-07-25 00:00:...| 256|PENDING\_PAYMENT| 3| 2| 502| 5| 250.0| 50.0|

| 2|2013-07-25 00:00:...| 256|PENDING\_PAYMENT| 4| 2| 403| 1| 129.99| 129.99|

| 4|2013-07-25 00:00:...| 8827| CLOSED| 5| 4| 897| 2| 49.98| 24.99|

| 4|2013-07-25 00:00:...| 8827| CLOSED| 6| 4| 365| 5| 299.95| 59.99|

| 4|2013-07-25 00:00:...| 8827| CLOSED| 7| 4| 502| 3| 150.0| 50.0|

| 4|2013-07-25 00:00:...| 8827| CLOSED| 8| 4| 1014| 4| 199.92| 49.98|

| 5|2013-07-25 00:00:...| 11318| COMPLETE| 9| 5| 957| 1| 299.98| 299.98|

| 5|2013-07-25 00:00:...| 11318| COMPLETE| 10| 5| 365| 5| 299.95| 59.99|

| 5|2013-07-25 00:00:...| 11318| COMPLETE| 11| 5| 1014| 2| 99.96| 49.98|

| 5|2013-07-25 00:00:...| 11318| COMPLETE| 12| 5| 957| 1| 299.98| 299.98|

| 5|2013-07-25 00:00:...| 11318| COMPLETE| 13| 5| 403| 1| 129.99| 129.99|

| 7|2013-07-25 00:00:...| 4530| COMPLETE| 14| 7| 1073| 1| 199.99| 199.99|

| 7|2013-07-25 00:00:...| 4530| COMPLETE| 15| 7| 957| 1| 299.98| 299.98|

| 7|2013-07-25 00:00:...| 4530| COMPLETE| 16| 7| 926| 5| 79.95| 15.99|

| 8|2013-07-25 00:00:...| 2911| PROCESSING| 17| 8| 365| 3| 179.97| 59.99|

| 8|2013-07-25 00:00:...| 2911| PROCESSING| 18| 8| 365| 5| 299.95| 59.99|

| 8|2013-07-25 00:00:...| 2911| PROCESSING| 19| 8| 1014| 4| 199.92| 49.98|

| 8|2013-07-25 00:00:...| 2911| PROCESSING| 20| 8| 502| 1| 50.0| 50.0|

+--------+--------------------+-----------------+---------------+-------------+-------------------+---------------------+-------------------+-------------------+------------------------+

>>> from pyspark.sql.functions import \*

>>> orderJoins.groupBy('order\_date', 'order\_item\_product\_id').agg(round(sum('order\_item\_subtotal'),2).alias('product\_revenue')).show()

+--------------------+---------------------+---------------+

| order\_date|order\_item\_product\_id|product\_revenue|

+--------------------+---------------------+---------------+

|2013-07-27 00:00:...| 703| 99.95|

|2013-07-29 00:00:...| 793| 44.97|

|2013-08-04 00:00:...| 825| 95.97|

|2013-08-06 00:00:...| 93| 24.99|

|2013-08-12 00:00:...| 627| 5358.66|

|2013-08-15 00:00:...| 926| 15.99|

|2013-08-17 00:00:...| 116| 224.95|

|2013-09-04 00:00:...| 957| 7499.5|

|2013-09-07 00:00:...| 235| 104.97|

|2013-09-17 00:00:...| 792| 89.94|

|2013-09-25 00:00:...| 44| 359.94|

|2013-09-27 00:00:...| 276| 31.99|

|2013-09-28 00:00:...| 572| 119.97|

|2013-10-04 00:00:...| 792| 44.97|

|2013-10-05 00:00:...| 886| 24.99|

|2013-10-14 00:00:...| 835| 31.99|

|2013-10-16 00:00:...| 835| 31.99|

|2013-10-19 00:00:...| 1004| 18399.08|

|2013-11-06 00:00:...| 502| 9750.0|

|2013-11-12 00:00:...| 282| 63.98|

+--------------------+---------------------+---------------+

only showing top 20 rows

Sorting Data in Data Frames

Help on method sort in module pyspark.sql.dataframe:

**sort**(self, \*cols, \*\*kwargs) method of pyspark.sql.dataframe.DataFrame instance

Returns a new :class:`DataFrame` sorted by the specified column(s).

:param cols: list of :class:`Column` or column names to sort by.

:param ascending: boolean or list of boolean (default True).

Sort ascending vs. descending. Specify list for multiple sort orders.

If a list is specified, length of the list must equal length of the `cols`.

>>> df.sort(df.age.desc()).collect()

[Row(age=5, name=u'Bob'), Row(age=2, name=u'Alice')]

>>> df.sort("age", ascending=False).collect()

[Row(age=5, name=u'Bob'), Row(age=2, name=u'Alice')]

>>> df.orderBy(df.age.desc()).collect()

[Row(age=5, name=u'Bob'), Row(age=2, name=u'Alice')]

>>> from pyspark.sql.functions import \*

>>> df.sort(asc("age")).collect()

[Row(age=2, name=u'Alice'), Row(age=5, name=u'Bob')]

>>> df.orderBy(desc("age"), "name").collect()

[Row(age=5, name=u'Bob'), Row(age=2, name=u'Alice')]

>>> df.orderBy(["age", "name"], ascending=[0, 1]).collect()

[Row(age=5, name=u'Bob'), Row(age=2, name=u'Alice')]

>>> orders.sort('order\_date').show()

+--------+--------------------+-----------------+---------------+

|order\_id| order\_date|order\_customer\_id| order\_status|

+--------+--------------------+-----------------+---------------+

| 1|2013-07-25 00:00:...| 11599| CLOSED|

| 2|2013-07-25 00:00:...| 256|PENDING\_PAYMENT|

| 3|2013-07-25 00:00:...| 12111| COMPLETE|

| 4|2013-07-25 00:00:...| 8827| CLOSED|

| 5|2013-07-25 00:00:...| 11318| COMPLETE|

| 6|2013-07-25 00:00:...| 7130| COMPLETE|

| 7|2013-07-25 00:00:...| 4530| COMPLETE|

| 8|2013-07-25 00:00:...| 2911| PROCESSING|

| 9|2013-07-25 00:00:...| 5657|PENDING\_PAYMENT|

| 10|2013-07-25 00:00:...| 5648|PENDING\_PAYMENT|

| 11|2013-07-25 00:00:...| 918| PAYMENT\_REVIEW|

| 12|2013-07-25 00:00:...| 1837| CLOSED|

| 13|2013-07-25 00:00:...| 9149|PENDING\_PAYMENT|

| 14|2013-07-25 00:00:...| 9842| PROCESSING|

| 15|2013-07-25 00:00:...| 2568| COMPLETE|

| 16|2013-07-25 00:00:...| 7276|PENDING\_PAYMENT|

| 17|2013-07-25 00:00:...| 2667| COMPLETE|

| 18|2013-07-25 00:00:...| 1205| CLOSED|

| 19|2013-07-25 00:00:...| 9488|PENDING\_PAYMENT|

| 20|2013-07-25 00:00:...| 9198| PROCESSING|

+--------+--------------------+-----------------+---------------+

only showing top 20 rows

NameError: name 'order' is not defined

>>> orders.sort('order\_date','order\_customer\_id').show()

+--------+--------------------+-----------------+---------------+

|order\_id| order\_date|order\_customer\_id| order\_status|

+--------+--------------------+-----------------+---------------+

| 57762|2013-07-25 00:00:...| 192| COMPLETE|

| 29|2013-07-25 00:00:...| 196| PROCESSING|

| 57759|2013-07-25 00:00:...| 216| ON\_HOLD|

| 2|2013-07-25 00:00:...| 256|PENDING\_PAYMENT|

| 57782|2013-07-25 00:00:...| 284| CLOSED|

| 22|2013-07-25 00:00:...| 333| COMPLETE|

| 28|2013-07-25 00:00:...| 656| COMPLETE|

| 74|2013-07-25 00:00:...| 662|PENDING\_PAYMENT|

| 81|2013-07-25 00:00:...| 674| PROCESSING|

| 89|2013-07-25 00:00:...| 824| ON\_HOLD|

| 11|2013-07-25 00:00:...| 918| PAYMENT\_REVIEW|

| 57778|2013-07-25 00:00:...| 1143| PENDING|

| 63|2013-07-25 00:00:...| 1148| COMPLETE|

| 18|2013-07-25 00:00:...| 1205| CLOSED|

| 83|2013-07-25 00:00:...| 1265| COMPLETE|

| 57767|2013-07-25 00:00:...| 1338|PENDING\_PAYMENT|

| 67|2013-07-25 00:00:...| 1406| COMPLETE|

| 85|2013-07-25 00:00:...| 1485| PENDING|

| 46|2013-07-25 00:00:...| 1549| ON\_HOLD|

| 57764|2013-07-25 00:00:...| 1763| COMPLETE|

+--------+--------------------+-----------------+---------------+

only showing top 20 rows

>>> orders.sort(['order\_date', 'order\_customer\_id'],ascending=[1,0]).show()

+--------+--------------------+-----------------+---------------+

|order\_id| order\_date|order\_customer\_id| order\_status|

+--------+--------------------+-----------------+---------------+

| 57785|2013-07-25 00:00:...| 12347| PROCESSING|

| 57787|2013-07-25 00:00:...| 12294|PENDING\_PAYMENT|

| 51|2013-07-25 00:00:...| 12271| CLOSED|

| 103|2013-07-25 00:00:...| 12256| PROCESSING|

| 48|2013-07-25 00:00:...| 12186| PROCESSING|

| 100|2013-07-25 00:00:...| 12131| PROCESSING|

| 3|2013-07-25 00:00:...| 12111| COMPLETE|

| 40|2013-07-25 00:00:...| 12092|PENDING\_PAYMENT|

| 57779|2013-07-25 00:00:...| 11941| COMPLETE|

| 70|2013-07-25 00:00:...| 11809|PENDING\_PAYMENT|

| 59|2013-07-25 00:00:...| 11644|PENDING\_PAYMENT|

| 1|2013-07-25 00:00:...| 11599| CLOSED|

| 94|2013-07-25 00:00:...| 11589| PROCESSING|

| 38|2013-07-25 00:00:...| 11586| PROCESSING|

| 99|2013-07-25 00:00:...| 11542|PENDING\_PAYMENT|

| 24|2013-07-25 00:00:...| 11441| CLOSED|

| 5|2013-07-25 00:00:...| 11318| COMPLETE|

| 57774|2013-07-25 00:00:...| 11196|PENDING\_PAYMENT|

| 67416|2013-07-25 00:00:...| 10920| CLOSED|

| 97|2013-07-25 00:00:...| 10784| PENDING|

+--------+--------------------+-----------------+---------------+

only showing top 20 rows

>>> orders.sort(orders.order\_date, orders.order\_customer\_id.desc()).show()

+--------+--------------------+-----------------+---------------+

|order\_id| order\_date|order\_customer\_id| order\_status|

+--------+--------------------+-----------------+---------------+

| 57785|2013-07-25 00:00:...| 12347| PROCESSING|

| 57787|2013-07-25 00:00:...| 12294|PENDING\_PAYMENT|

| 51|2013-07-25 00:00:...| 12271| CLOSED|

| 103|2013-07-25 00:00:...| 12256| PROCESSING|

| 48|2013-07-25 00:00:...| 12186| PROCESSING|

| 100|2013-07-25 00:00:...| 12131| PROCESSING|

| 3|2013-07-25 00:00:...| 12111| COMPLETE|

| 40|2013-07-25 00:00:...| 12092|PENDING\_PAYMENT|

| 57779|2013-07-25 00:00:...| 11941| COMPLETE|

| 70|2013-07-25 00:00:...| 11809|PENDING\_PAYMENT|

| 59|2013-07-25 00:00:...| 11644|PENDING\_PAYMENT|

| 1|2013-07-25 00:00:...| 11599| CLOSED|

| 94|2013-07-25 00:00:...| 11589| PROCESSING|

| 38|2013-07-25 00:00:...| 11586| PROCESSING|

| 99|2013-07-25 00:00:...| 11542|PENDING\_PAYMENT|

| 24|2013-07-25 00:00:...| 11441| CLOSED|

| 5|2013-07-25 00:00:...| 11318| COMPLETE|

| 57774|2013-07-25 00:00:...| 11196|PENDING\_PAYMENT|

| 67416|2013-07-25 00:00:...| 10920| CLOSED|

| 97|2013-07-25 00:00:...| 10784| PENDING|

+--------+--------------------+-----------------+---------------+

only showing top 20 rows

Help on method sortWithinPartitions in module pyspark.sql.dataframe:

**sortWithinPartitions**(self, \*cols, \*\*kwargs) method of pyspark.sql.dataframe.DataFrame instance

Returns a new :class:`DataFrame` with each partition sorted by the specified column(s).

:param cols: list of :class:`Column` or column names to sort by.

:param ascending: boolean or list of boolean (default True).

Sort ascending vs. descending. Specify list for multiple sort orders.

If a list is specified, length of the list must equal length of the `cols`.

>>> df.sortWithinPartitions("age", ascending=False).show()

+---+-----+

|age| name|

+---+-----+

| 2|Alice|

| 5| Bob|

+---+-----+

.. versionadded:: 1.6

orders.sortWithinPartitions('order\_date', 'order\_customer\_id').show()

+--------+--------------------+-----------------+---------------+

|order\_id| order\_date|order\_customer\_id| order\_status|

+--------+--------------------+-----------------+---------------+

| 57762|2013-07-25 00:00:...| 192| COMPLETE|

| 29|2013-07-25 00:00:...| 196| PROCESSING|

| 57759|2013-07-25 00:00:...| 216| ON\_HOLD|

| 2|2013-07-25 00:00:...| 256|PENDING\_PAYMENT|

| 57782|2013-07-25 00:00:...| 284| CLOSED|

| 22|2013-07-25 00:00:...| 333| COMPLETE|

| 28|2013-07-25 00:00:...| 656| COMPLETE|

| 74|2013-07-25 00:00:...| 662|PENDING\_PAYMENT|

| 81|2013-07-25 00:00:...| 674| PROCESSING|

| 89|2013-07-25 00:00:...| 824| ON\_HOLD|

| 11|2013-07-25 00:00:...| 918| PAYMENT\_REVIEW|

| 57778|2013-07-25 00:00:...| 1143| PENDING|

| 63|2013-07-25 00:00:...| 1148| COMPLETE|

| 18|2013-07-25 00:00:...| 1205| CLOSED|

| 83|2013-07-25 00:00:...| 1265| COMPLETE|

| 57767|2013-07-25 00:00:...| 1338|PENDING\_PAYMENT|

| 67|2013-07-25 00:00:...| 1406| COMPLETE|

| 85|2013-07-25 00:00:...| 1485| PENDING|

| 46|2013-07-25 00:00:...| 1549| ON\_HOLD|

| 57764|2013-07-25 00:00:...| 1763| COMPLETE|

+--------+--------------------+-----------------+---------------+

only showing top 20 rows

>>> orders.sort('order\_status').show()

+--------+--------------------+-----------------+------------+

|order\_id| order\_date|order\_customer\_id|order\_status|

+--------+--------------------+-----------------+------------+

| 527|2013-07-28 00:00:...| 5426| CANCELED|

| 1435|2013-08-01 00:00:...| 1879| CANCELED|

| 552|2013-07-28 00:00:...| 1445| CANCELED|

| 112|2013-07-26 00:00:...| 5375| CANCELED|

| 564|2013-07-28 00:00:...| 2216| CANCELED|

| 955|2013-07-30 00:00:...| 8117| CANCELED|

| 1383|2013-08-01 00:00:...| 1753| CANCELED|

| 962|2013-07-30 00:00:...| 9492| CANCELED|

| 607|2013-07-28 00:00:...| 6376| CANCELED|

| 1013|2013-07-30 00:00:...| 1903| CANCELED|

| 667|2013-07-28 00:00:...| 4726| CANCELED|

| 1169|2013-07-31 00:00:...| 3971| CANCELED|

| 717|2013-07-29 00:00:...| 8208| CANCELED|

| 1186|2013-07-31 00:00:...| 11947| CANCELED|

| 753|2013-07-29 00:00:...| 5094| CANCELED|

| 1190|2013-07-31 00:00:...| 12360| CANCELED|

| 50|2013-07-25 00:00:...| 5225| CANCELED|

| 1313|2013-08-01 00:00:...| 3471| CANCELED|

| 716|2013-07-29 00:00:...| 2581| CANCELED|

| 1365|2013-08-01 00:00:...| 8567| CANCELED|

+--------+--------------------+-----------------+------------+

only showing top 20 rows

>>> orders.sort('order\_date', 'order\_status').show()

+--------+--------------------+-----------------+------------+

|order\_id| order\_date|order\_customer\_id|order\_status|

+--------+--------------------+-----------------+------------+

| 50|2013-07-25 00:00:...| 5225| CANCELED|

| 1|2013-07-25 00:00:...| 11599| CLOSED|

| 12|2013-07-25 00:00:...| 1837| CLOSED|

| 4|2013-07-25 00:00:...| 8827| CLOSED|

| 37|2013-07-25 00:00:...| 5863| CLOSED|

| 18|2013-07-25 00:00:...| 1205| CLOSED|

| 24|2013-07-25 00:00:...| 11441| CLOSED|

| 25|2013-07-25 00:00:...| 9503| CLOSED|

| 57754|2013-07-25 00:00:...| 4648| CLOSED|

| 90|2013-07-25 00:00:...| 9131| CLOSED|

| 51|2013-07-25 00:00:...| 12271| CLOSED|

| 57|2013-07-25 00:00:...| 7073| CLOSED|

| 61|2013-07-25 00:00:...| 4791| CLOSED|

| 62|2013-07-25 00:00:...| 9111| CLOSED|

| 87|2013-07-25 00:00:...| 3065| CLOSED|

| 101|2013-07-25 00:00:...| 5116| CLOSED|

| 57766|2013-07-25 00:00:...| 2376| CLOSED|

| 57781|2013-07-25 00:00:...| 6143| CLOSED|

| 57782|2013-07-25 00:00:...| 284| CLOSED|

| 67416|2013-07-25 00:00:...| 10920| CLOSED|

+--------+--------------------+-----------------+------------+

only showing top 20 rows

>>> ordersItems.sort(['order\_item\_order\_id', 'order\_item\_subtotal'], descending=[1,1]).show()

+-------------+-------------------+---------------------+-------------------+-------------------+------------------------+

|order\_item\_id|order\_item\_order\_id|order\_item\_product\_id|order\_item\_quantity|order\_item\_subtotal|order\_item\_product\_price|

+-------------+-------------------+---------------------+-------------------+-------------------+------------------------+

| 1| 1| 957| 1| 299.98| 299.98|

| 4| 2| 403| 1| 129.99| 129.99|

| 2| 2| 1073| 1| 199.99| 199.99|

| 3| 2| 502| 5| 250.0| 50.0|

| 5| 4| 897| 2| 49.98| 24.99|

| 7| 4| 502| 3| 150.0| 50.0|

| 8| 4| 1014| 4| 199.92| 49.98|

| 6| 4| 365| 5| 299.95| 59.99|

| 11| 5| 1014| 2| 99.96| 49.98|

| 13| 5| 403| 1| 129.99| 129.99|

| 10| 5| 365| 5| 299.95| 59.99|

| 9| 5| 957| 1| 299.98| 299.98|

| 12| 5| 957| 1| 299.98| 299.98|

| 16| 7| 926| 5| 79.95| 15.99|

| 14| 7| 1073| 1| 199.99| 199.99|

| 15| 7| 957| 1| 299.98| 299.98|

| 20| 8| 502| 1| 50.0| 50.0|

| 17| 8| 365| 3| 179.97| 59.99|

| 19| 8| 1014| 4| 199.92| 49.98|

| 18| 8| 365| 5| 299.95| 59.99|

+-------------+-------------------+---------------------+-------------------+-------------------+------------------------+

dailyRevenue = orders.where('order\_status in ("COMPLETE", "CLOSED")').join(ordersItems, orders.order\_id == ordersItems.order\_item\_order\_id, 'inner').groupBy(orders.order\_date, ordersItems.order\_item\_product\_id).agg(round(sum(ordersItems.order\_item\_subtotal), 2).alias('revenue'))

>>>

>>> dailyRevenue.show()

+--------------------+---------------------+-------+

| order\_date|order\_item\_product\_id|revenue|

+--------------------+---------------------+-------+

|2013-07-27 00:00:...| 703| 39.98|

|2013-07-29 00:00:...| 793| 44.97|

|2013-08-12 00:00:...| 627| 3199.2|

|2013-08-15 00:00:...| 926| 15.99|

|2013-09-04 00:00:...| 957|3599.76|

|2013-09-07 00:00:...| 235| 104.97|

|2013-09-17 00:00:...| 792| 14.99|

|2013-09-25 00:00:...| 44| 239.96|

|2013-09-27 00:00:...| 276| 31.99|

|2013-10-04 00:00:...| 792| 44.97|

|2013-10-14 00:00:...| 835| 31.99|

|2013-10-19 00:00:...| 1004|5599.72|

|2013-11-06 00:00:...| 502| 3800.0|

|2013-11-12 00:00:...| 282| 63.98|

|2013-11-19 00:00:...| 792| 59.96|

|2013-11-29 00:00:...| 235| 174.95|

|2013-11-30 00:00:...| 278| 44.99|

|2013-12-03 00:00:...| 778| 99.96|

|2013-12-09 00:00:...| 564| 180.0|

|2013-12-11 00:00:...| 775| 49.95|

+--------------------+---------------------+-------+

+--------------------+---------------------+-------+

| order\_date|order\_item\_product\_id|revenue|

+--------------------+---------------------+-------+

|2013-07-25 00:00:...| 1004|5599.72|

|2013-07-25 00:00:...| 191|5099.49|

|2013-07-25 00:00:...| 957| 4499.7|

|2013-07-25 00:00:...| 365|3359.44|

|2013-07-25 00:00:...| 1073|2999.85|

|2013-07-25 00:00:...| 1014|2798.88|

|2013-07-25 00:00:...| 403|1949.85|

|2013-07-25 00:00:...| 502| 1650.0|

|2013-07-25 00:00:...| 627|1079.73|

|2013-07-25 00:00:...| 226| 599.99|

|2013-07-25 00:00:...| 24| 319.96|

|2013-07-25 00:00:...| 821| 207.96|

|2013-07-25 00:00:...| 625| 199.99|

|2013-07-25 00:00:...| 705| 119.99|

|2013-07-25 00:00:...| 572| 119.97|

|2013-07-25 00:00:...| 666| 109.99|

|2013-07-25 00:00:...| 725| 108.0|

|2013-07-25 00:00:...| 134| 100.0|

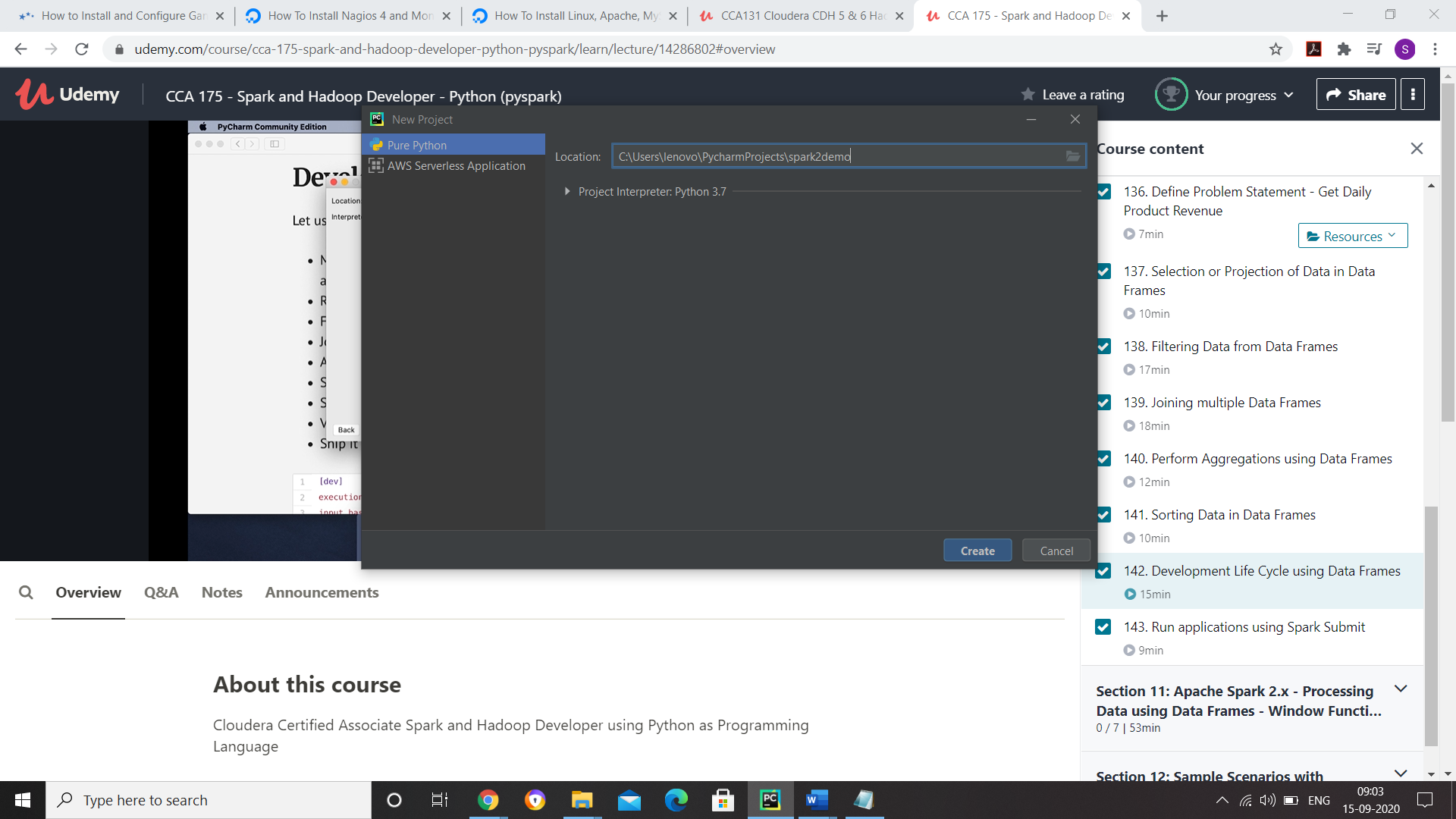
|2013-07-25 00:00:...| 906| 99.96|

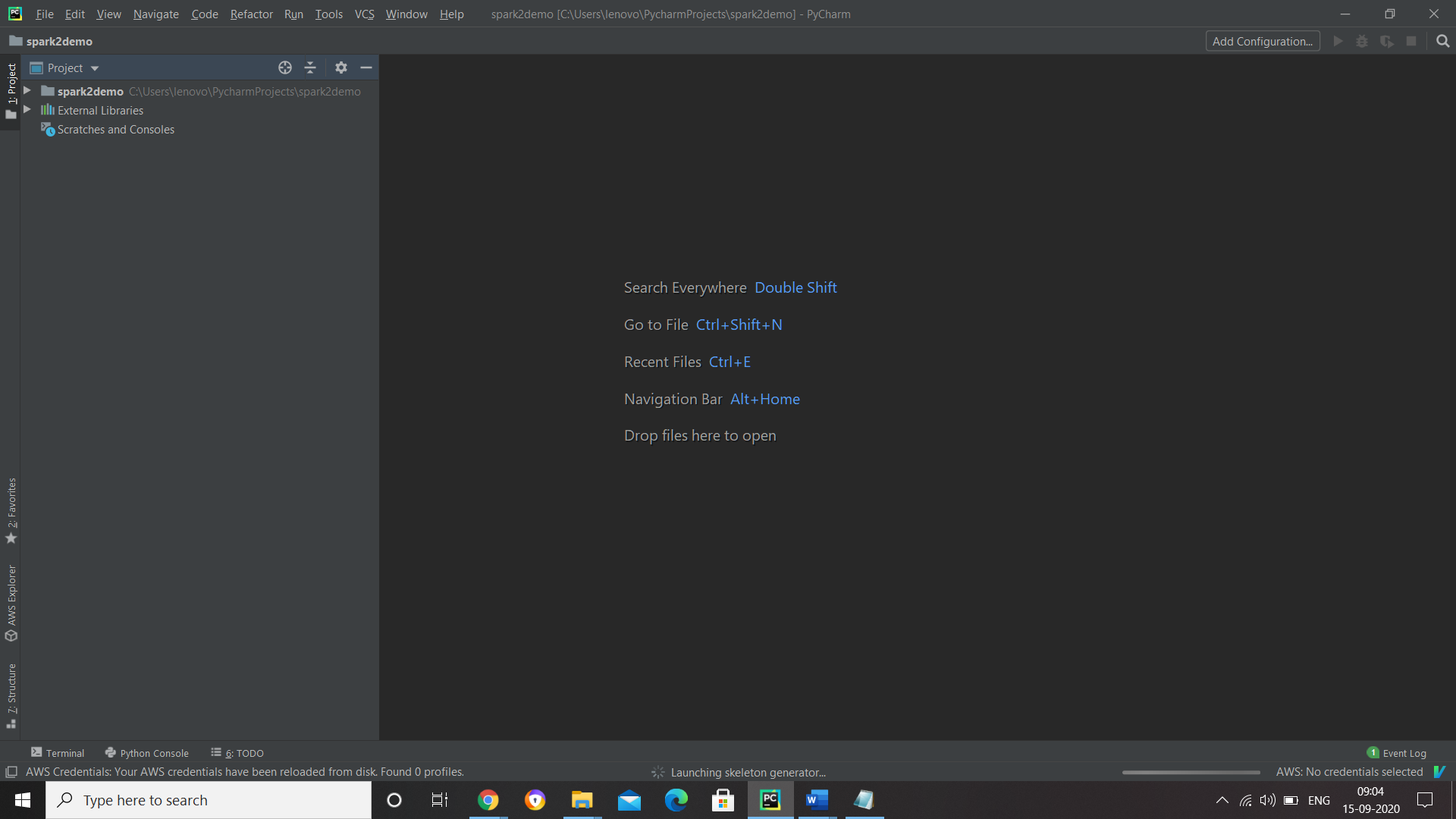
|2013-07-25 00:00:...| 828| 95.97|

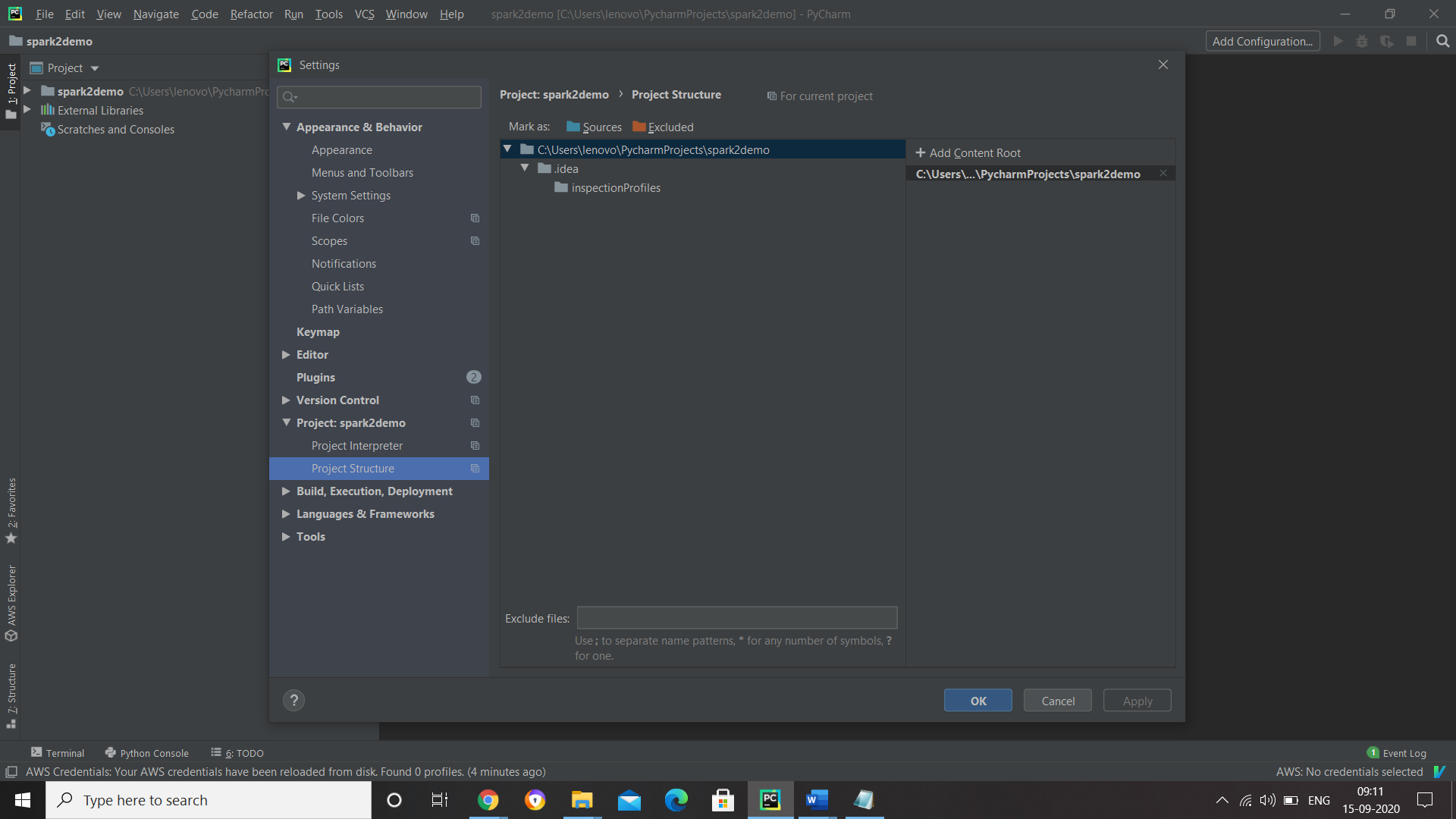
+--------------------+---------------------+-------+

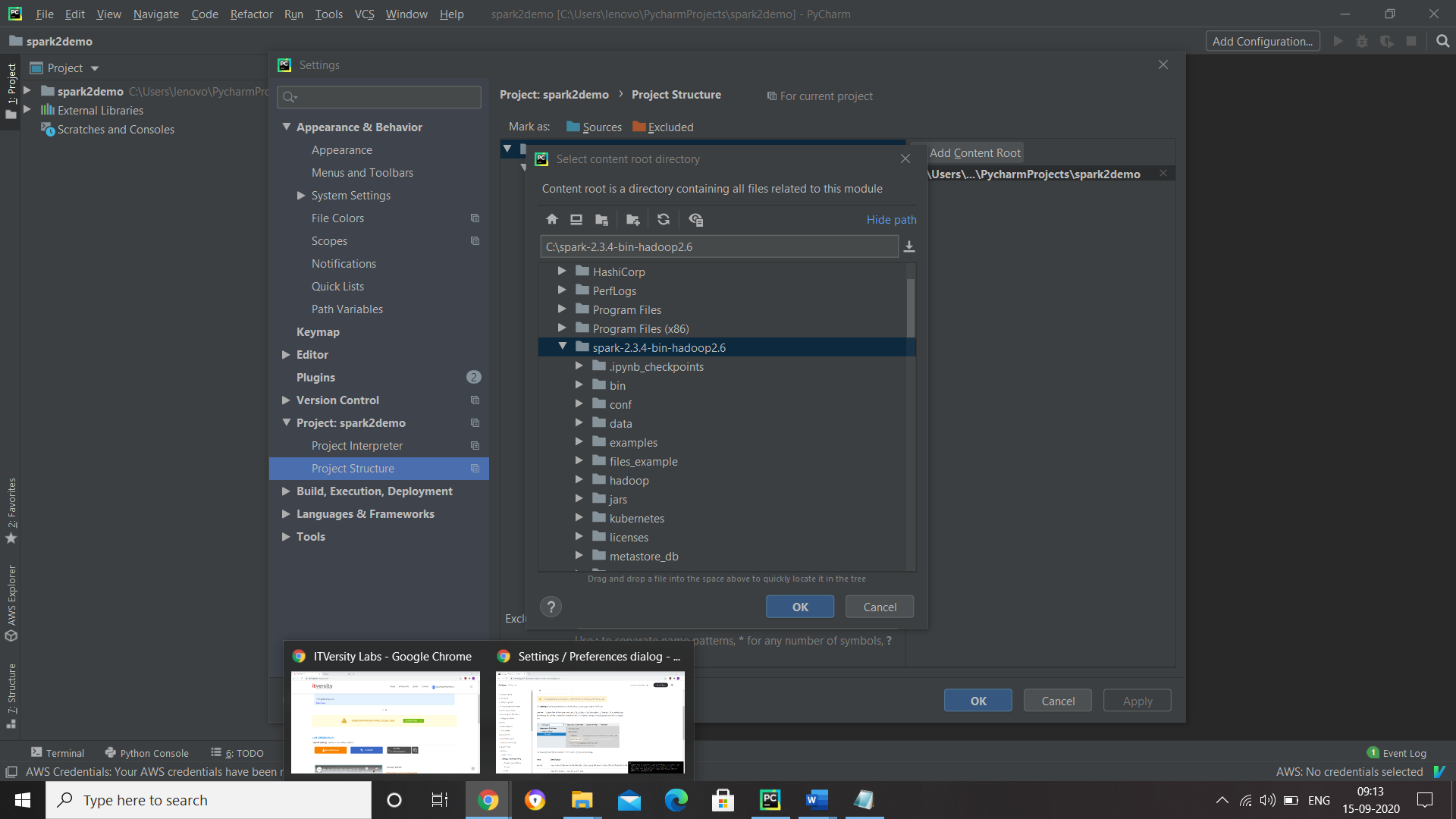
only showing top 20 rows

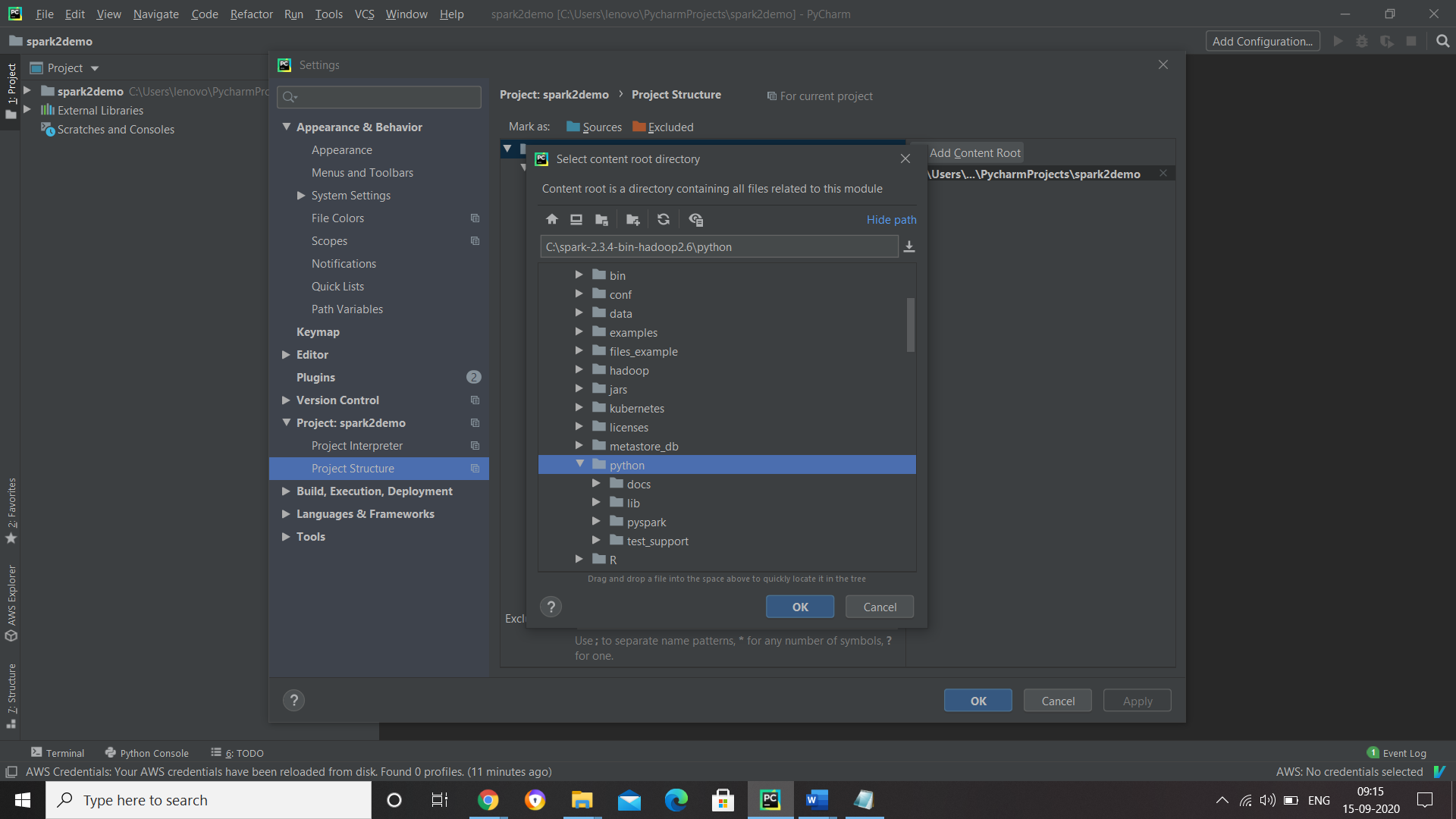
development life cycle IDE

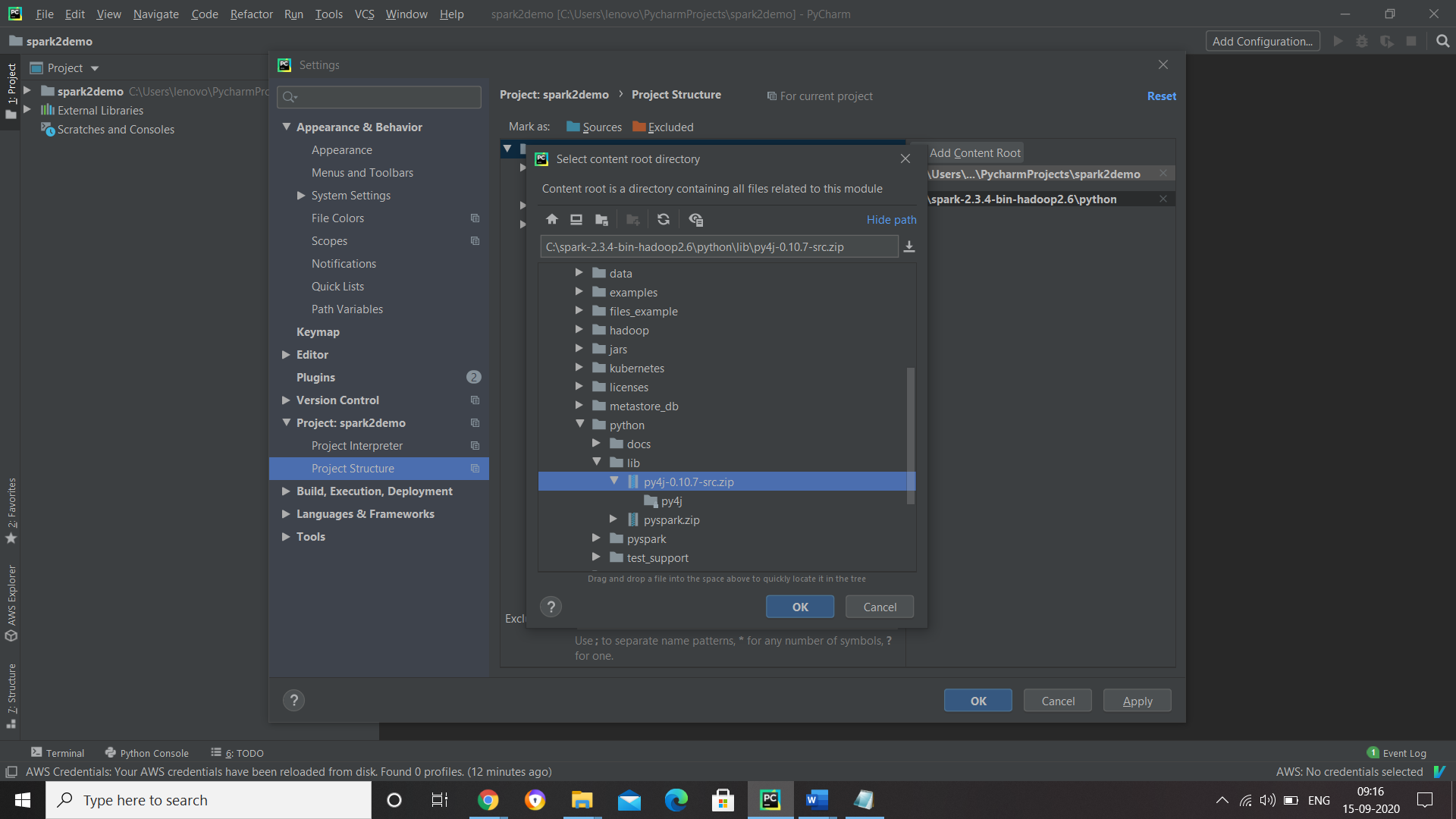


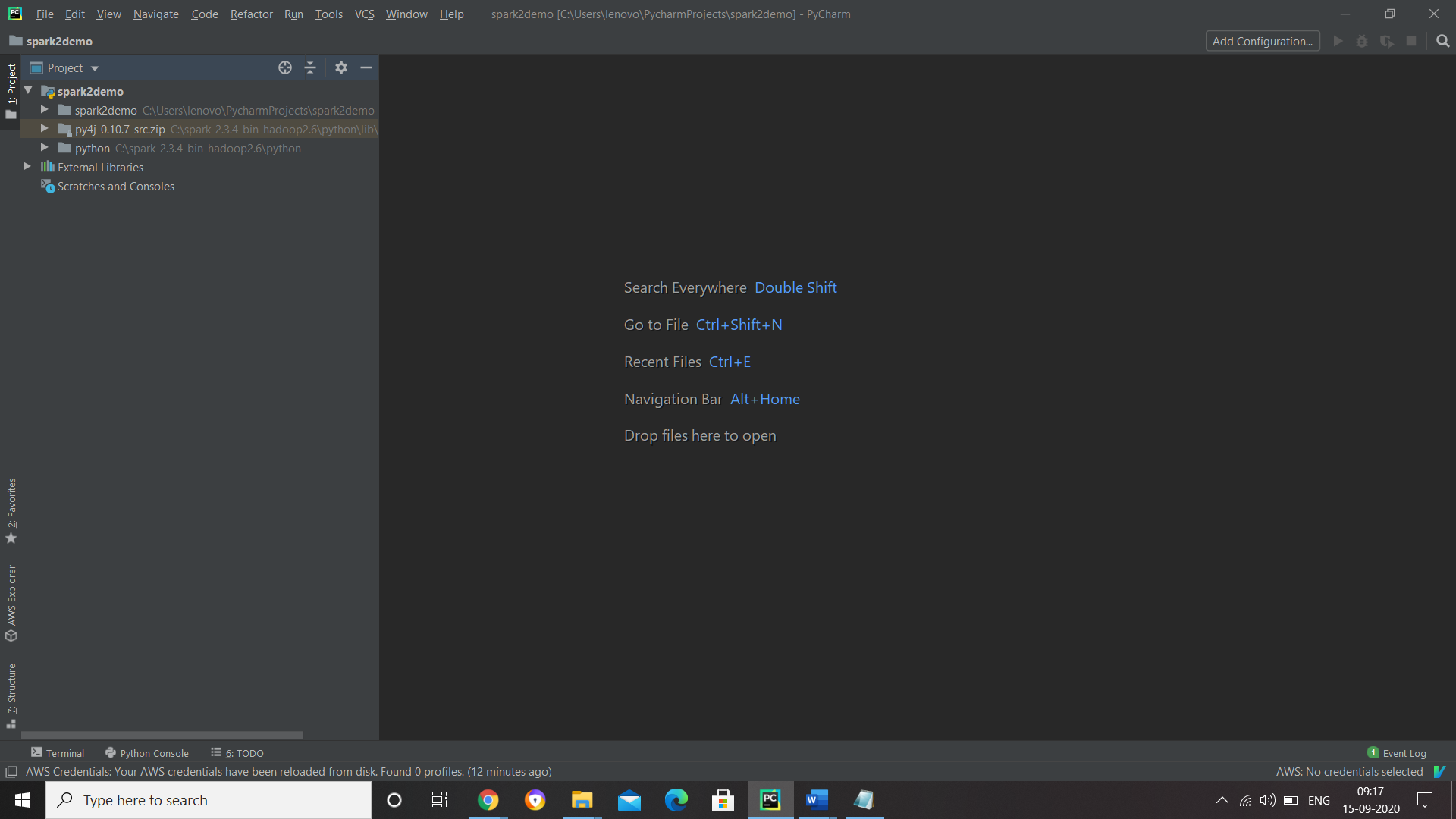


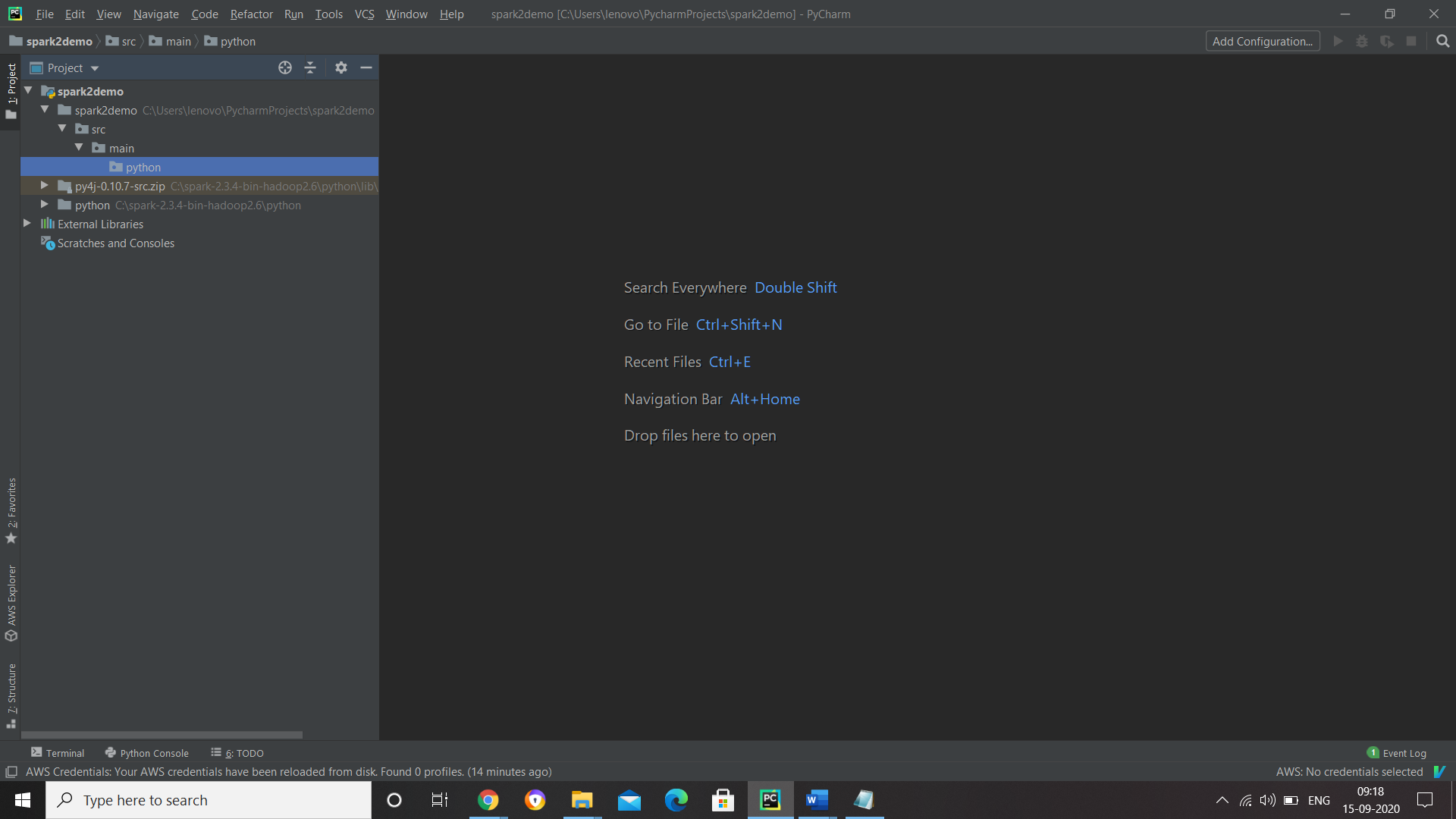


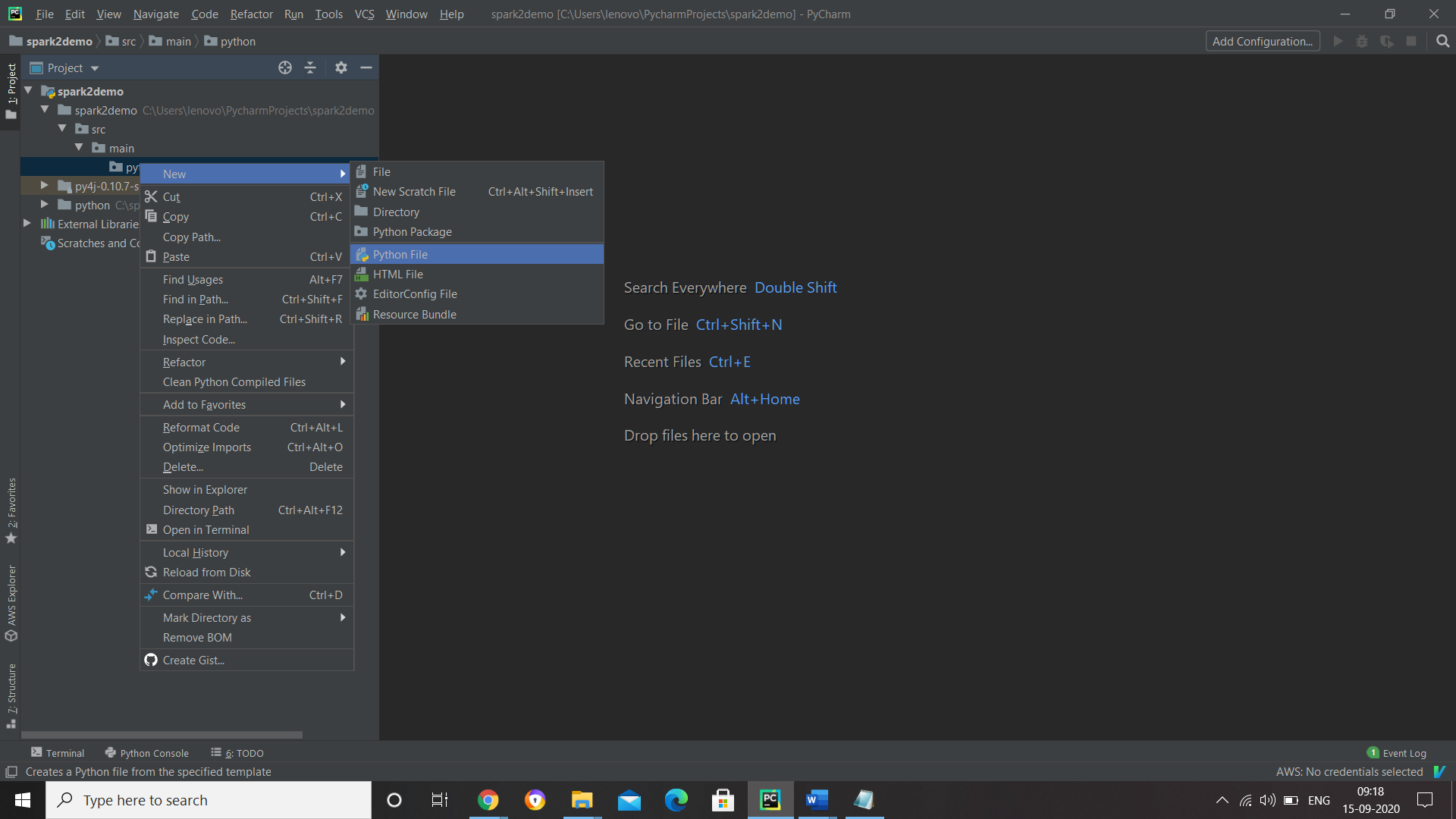


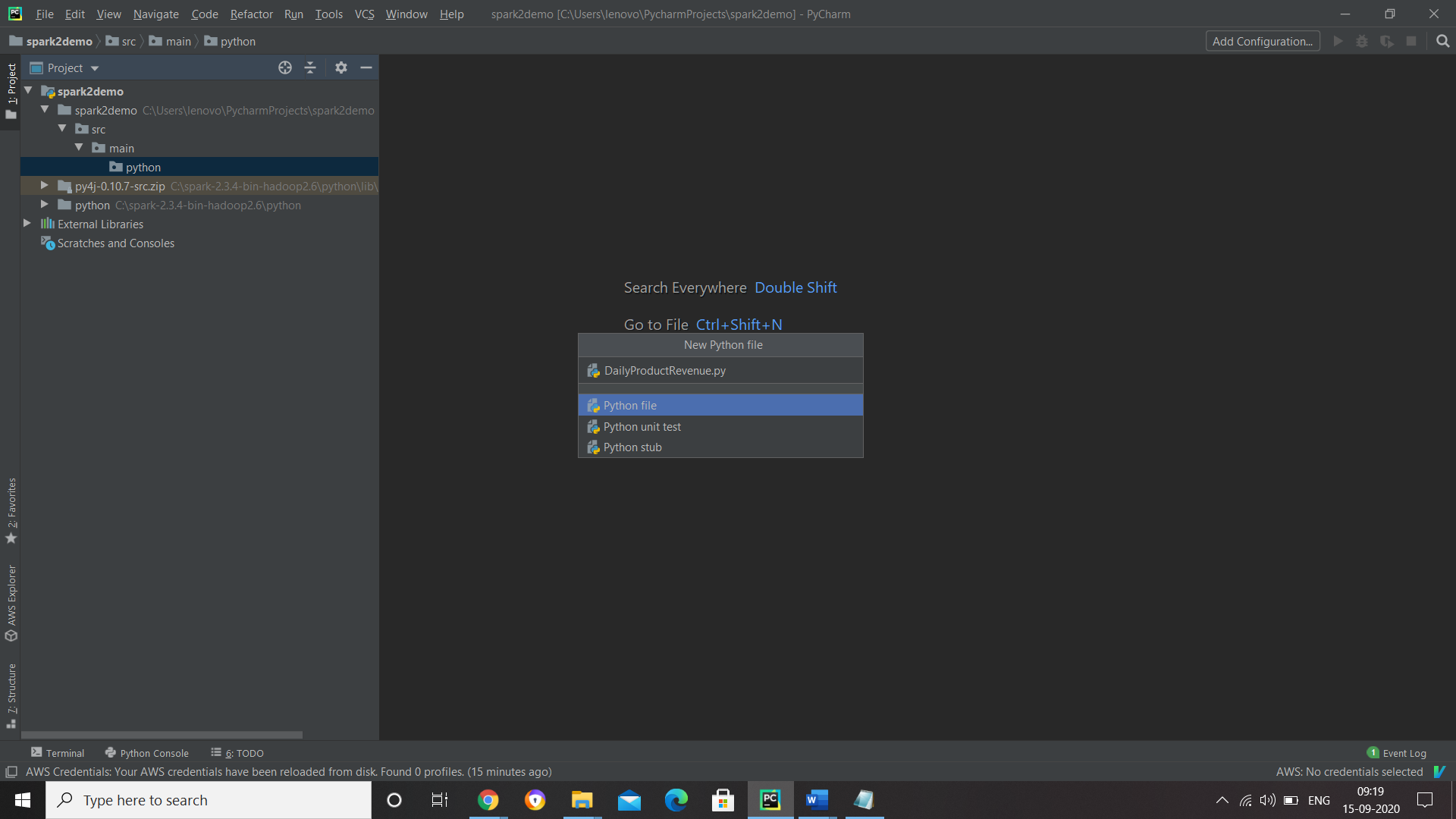


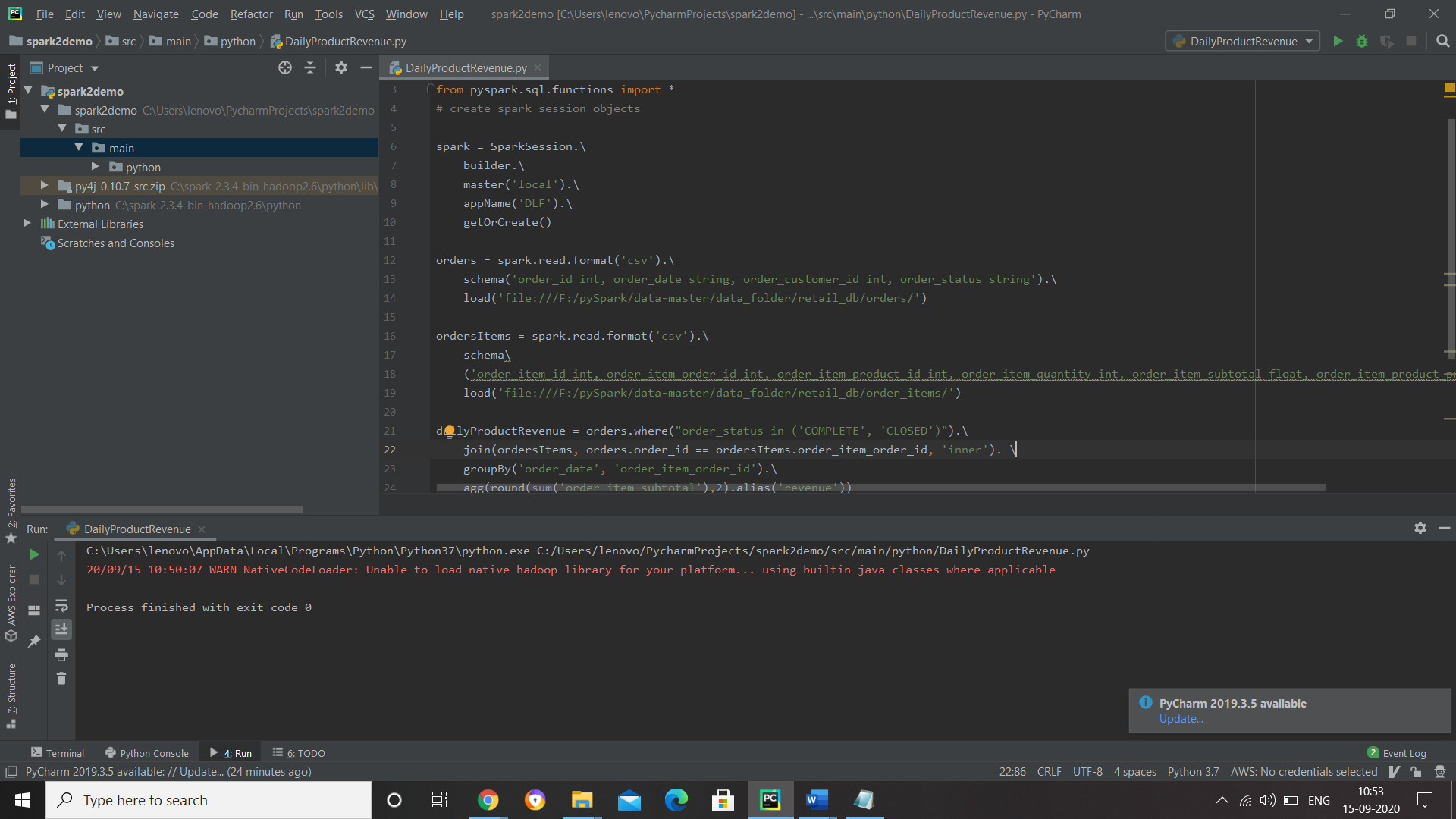




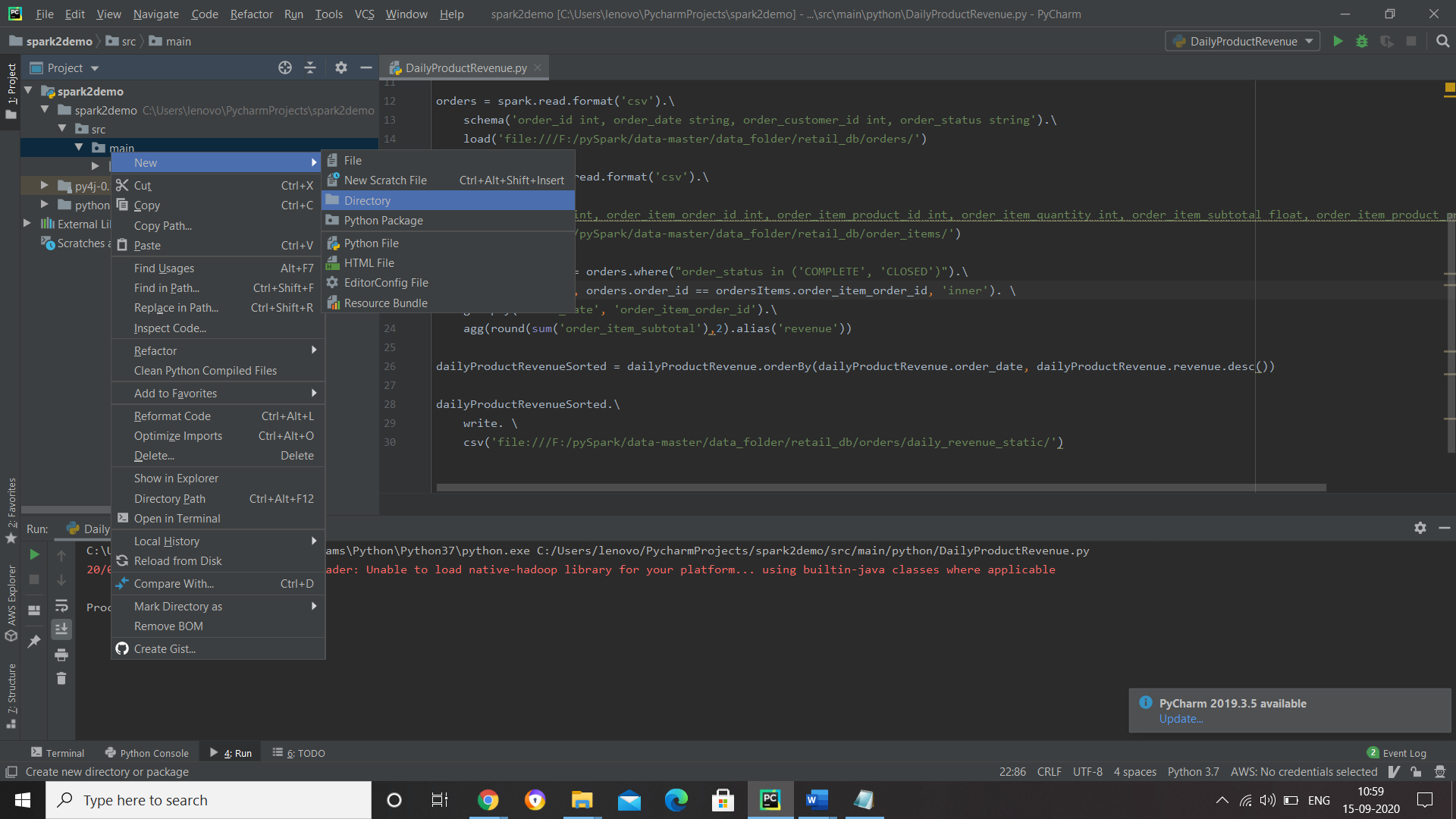


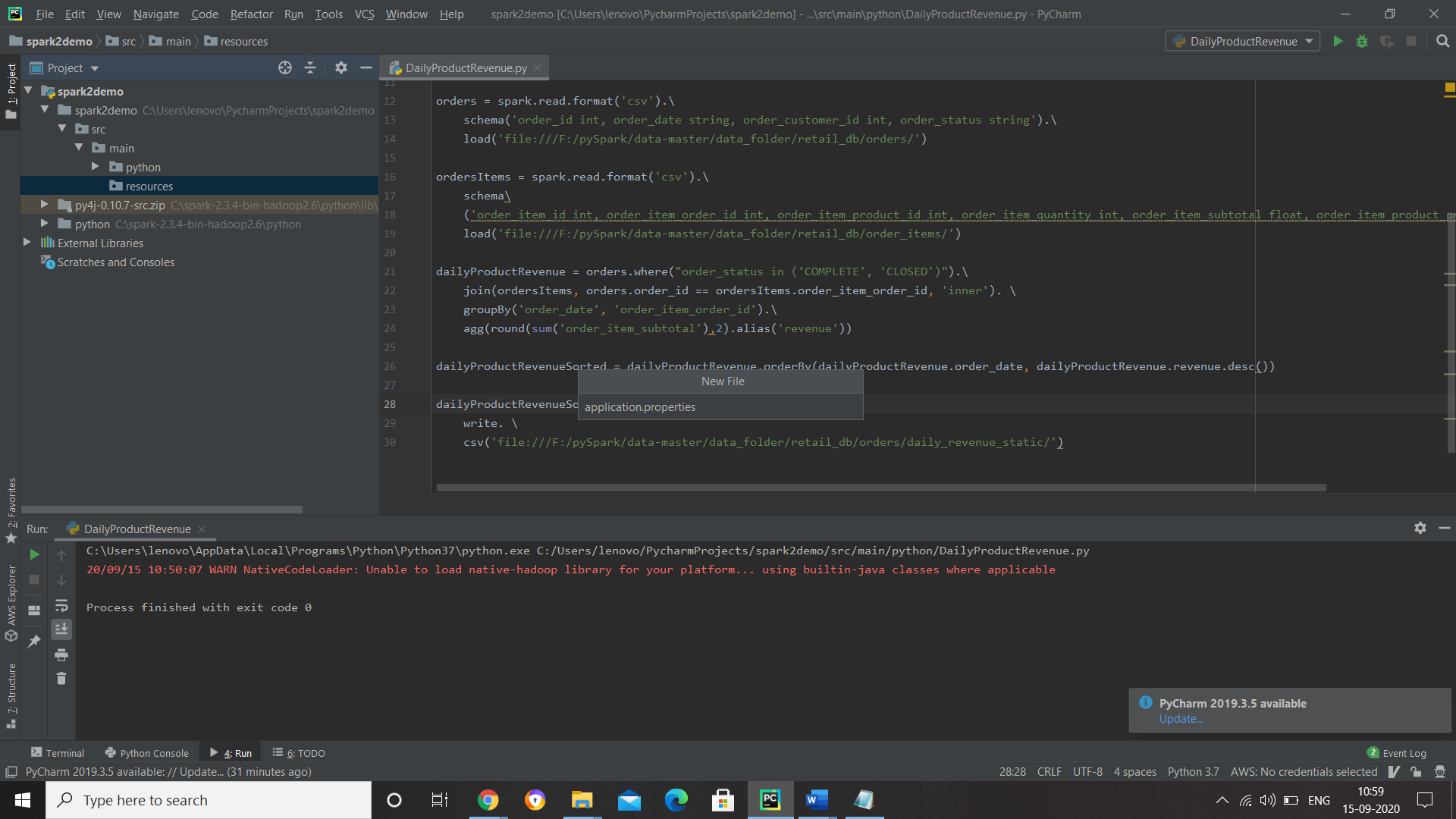


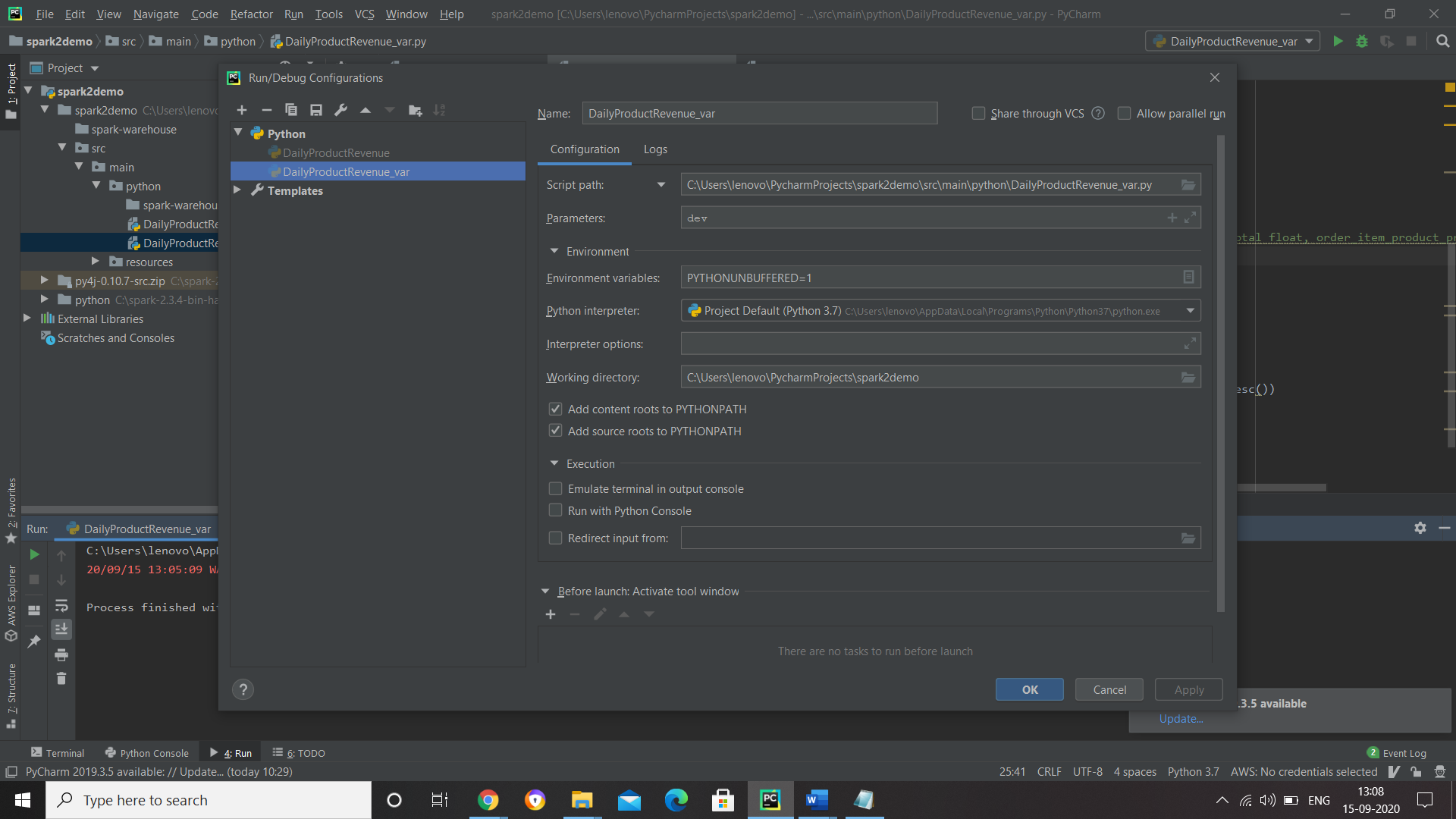




Externalise all properties







 Run applications using Spark Submit

Locally

/c/Users/lenovo/PycharmProjects/spark2demo

(base) C:\Users\lenovo\PycharmProjects>cd spark2demo

(base) C:\Users\lenovo\PycharmProjects\spark2demo>ls

'ls' is not recognized as an internal or external command,

operable program or batch file.

(base) C:\Users\lenovo\PycharmProjects\spark2demo>dir

Volume in drive C is Windows

Volume Serial Number is FCFC-EE17

Directory of C:\Users\lenovo\PycharmProjects\spark2demo

15-09-2020 13:05 <DIR> .

15-09-2020 13:05 <DIR> ..

15-09-2020 13:35 <DIR> .idea

15-09-2020 13:05 <DIR> spark-warehouse

15-09-2020 09:18 <DIR> src

0 File(s) 0 bytes

5 Dir(s) 817,188,970,496 bytes free

(base) C:\Users\lenovo\PycharmProjects\spark2demo>spark-submit src\main\python\DailyProductRevenue\_var.py dev

20/09/15 13:52:03 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

(base) C:\Users\lenovo\PycharmProjects\spark2demo>

lenovo@LAPTOP-LKH3JHUU MINGW64 ~/PycharmProjects

$ scp -r spark2demo saurabhbhanwala@gw03.itversity.com:~

saurabhbhanwala@gw03.itversity.com's password:

profiles\_settings.xml 100% 174 0.4KB/s 00:00

misc.xml 100% 188 0.5KB/s 00:00

modules.xml 100% 279 0.7KB/s 00:00

spark2demo.iml 100% 448 1.4KB/s 00:00

workspace.xml 100% 6923 21.7KB/s 00:00

DailyProductRevenue.py 100% 1301 4.1KB/s 00:00

DailyProductRevenue\_var.py 100% 1508 4.8KB/s 00:00

application.properties 100% 313 1.0KB/s 00:00

[saurabhbhanwala@gw03 ~]$ cd spark2demo/

[saurabhbhanwala@gw03 spark2demo]$ spark-submit \

> --master yarn-client \

> --conf spark.ui.port=12601 \

> src/main/python/DailyProductRevenue\_var.py prod

Multiple versions of Spark are installed but SPARK\_MAJOR\_VERSION is not set

Spark1 will be picked by default

Traceback (most recent call last):

File "/home/saurabhbhanwala/spark2demo/src/main/python/DailyProductRevenue\_var.py", line 5, in <module>

from pyspark.sql import SparkSession

ImportError: cannot import name SparkSession

[saurabhbhanwala@gw03 spark2demo]$

Export PYSPARK\_PYTHON=3

On cluster

export SPARK\_MAJOR\_VERSION=2

[saurabhbhanwala@gw03 spark2demo]$ spark-submit --master yarn-client --conf spark.ui.port=12601 src/main/python/DailyProductRevenue\_var.py prod

SPARK\_MAJOR\_VERSION is set to 2, using Spark2

Warning: Master yarn-client is deprecated since 2.0. Please use master "yarn" with specified deploy mode instead.

[saurabhbhanwala@gw03 spark2demo]$ hadoop fs -ls /user/saurabhbhanwala/retail\_db

Found 8 items

[saurabhbhanwala@gw03 spark2demo]$ spark2-submit --master yarn --deploy-mode client src/main/python/DailyProductRevenue\_var.py prod

SPARK\_MAJOR\_VERSION is set to 2, using Spark2

[saurabhbhanwala@gw03 spark2demo]$

Backing up from task

[saurabhbhanwala@gw03 spark2demo]$ hadoop fs -ls /user/saurabhbhanwala/.Trash/Current/user/saurabhbhanwala/retail\_db

Found 1 items

drwxr-xr-x - saurabhbhanwala hdfs 0 2020-09-15 04:41 /user/saurabhbhanwala/.Trash/Current/user/saurabhbhanwala/retail\_db/daily\_revenue\_static

[saurabhbhanwala@gw03 spark2demo]$ hadoop fs -ls /user/saurabhbhanwala/.Trash/Current/user/saurabhbhanwala/retail\_db1600159417993

Found 7 items

drwxr-xr-x - saurabhbhanwala hdfs 0 2017-02-20 01:35 /user/saurabhbhanwala/.Trash/Current/user/saurabhbhanwala/retail\_db1600159417993/categories

drwxr-xr-x - saurabhbhanwala hdfs 0 2017-02-20 01:35 /user/saurabhbhanwala/.Trash/Current/user/saurabhbhanwala/retail\_db1600159417993/customers

drwxr-xr-x - saurabhbhanwala hdfs 0 2017-02-20 01:35 /user/saurabhbhanwala/.Trash/Current/user/saurabhbhanwala/retail\_db1600159417993/departments

drwxr-xr-x - saurabhbhanwala hdfs 0 2017-02-20 01:35 /user/saurabhbhanwala/.Trash/Current/user/saurabhbhanwala/retail\_db1600159417993/order\_items

drwxr-xr-x - saurabhbhanwala hdfs 0 2017-02-20 01:35 /user/saurabhbhanwala/.Trash/Current/user/saurabhbhanwala/retail\_db1600159417993/orders

drwxr-xr-x - saurabhbhanwala hdfs 0 2017-02-20 01:35 /user/saurabhbhanwala/.Trash/Current/user/saurabhbhanwala/retail\_db1600159417993/products

drwxr-xr-x - saurabhbhanwala hdfs 0 2020-09-06 05:50 /user/saurabhbhanwala/.Trash/Current/user/saurabhbhanwala/retail\_db1600159417993/retail\_db

[saurabhbhanwala@gw03 spark2demo]$ hadoop fs -cp /user/saurabhbhanwala/.Trash/Current/user/saurabhbhanwala/retail\_db1600159417993 /user/saurabhbhanwala/

[saurabhbhanwala@gw03 spark2demo]$ mhadoop fs -ls /user/saurbahbhanwala

bash: mhadoop: command not found...

Similar command is: 'hadoop'

[saurabhbhanwala@gw03 spark2demo]$ hadoop fs -ls /user/saurabhbhanwala

Found 6 items

drwx------ - saurabhbhanwala hdfs 0 2020-09-15 04:43 /user/saurabhbhanwala/.Trash

drwxr-xr-x - saurabhbhanwala hdfs 0 2020-09-15 04:41 /user/saurabhbhanwala/.sparkStaging

drwx------ - saurabhbhanwala hdfs 0 2020-09-05 03:11 /user/saurabhbhanwala/.staging

drwxr-xr-x - saurabhbhanwala hdfs 0 2020-09-12 03:53 /user/saurabhbhanwala/orderItemscsvPartition

drwxr-xr-x - saurabhbhanwala hdfs 0 2020-09-15 04:50 /user/saurabhbhanwala/retail\_db1600159417993

drwxr-xr-x - saurabhbhanwala hdfs 0 2020-09-10 01:49 /user/saurabhbhanwala/retail\_db\_json

[saurabhbhanwala@gw03 spark2demo]$ hadoop fs -mkdir /user/saurabhbhanwala/retail\_db

[saurabhbhanwala@gw03 spark2demo]$ hadoop fs -mv /user/saurabhbhanwala/retail\_db1600159417993/\* /user/saurabhbhanwala/retail\_db/

[saurabhbhanwala@gw03 spark2demo]$ hadoop fs -ls /user/saurabhbhanwala/retail\_db/

Found 7 items

drwxr-xr-x - saurabhbhanwala hdfs 0 2020-09-15 04:50 /user/saurabhbhanwala/retail\_db/categories

drwxr-xr-x - saurabhbhanwala hdfs 0 2020-09-15 04:50 /user/saurabhbhanwala/retail\_db/customers

drwxr-xr-x - saurabhbhanwala hdfs 0 2020-09-15 04:50 /user/saurabhbhanwala/retail\_db/departments

drwxr-xr-x - saurabhbhanwala hdfs 0 2020-09-15 04:50 /user/saurabhbhanwala/retail\_db/order\_items

drwxr-xr-x - saurabhbhanwala hdfs 0 2020-09-15 04:50 /user/saurabhbhanwala/retail\_db/orders

drwxr-xr-x - saurabhbhanwala hdfs 0 2020-09-15 04:50 /user/saurabhbhanwala/retail\_db/products

drwxr-xr-x - saurabhbhanwala hdfs 0 2020-09-15 04:50 /user/saurabhbhanwala/retail\_db/retail\_db

[saurabhbhanwala@gw03 spark2demo]$ hadoop fs -rm -r /user/saurabhbhanwala/retail\_db1600159417993

20/09/15 04:53:44 INFO fs.TrashPolicyDefault: Moved: 'hdfs://nn01.itversity.com:8020/user/saurabhbhanwala/retail\_db1600159417993' to trash at: hdfs://nn01.itversity.com:8020/user/saurabhbhanwala/.Trash/Current/user/saurabhbhanwala/retail\_db16001594179931600160024798

[saurabhbhanwala@gw03 spark2demo]$

# Alternate conf dir. (Default: ${SPARK\_HOME}/conf)

export SPARK\_CONF\_DIR=${SPARK\_CONF\_DIR:-/usr/hdp/current/spark-client/conf}

# Where log files are stored.(Default:${SPARK\_HOME}/logs)

#export SPARK\_LOG\_DIR=${SPARK\_HOME:-/usr/hdp/current/spark-client}/logs

export SPARK\_LOG\_DIR=/var/log/spark

# Where the pid file is stored. (Default: /tmp)

export SPARK\_PID\_DIR=/var/run/spark

# A string representing this instance of spark.(Default: $USER)

SPARK\_IDENT\_STRING=$USER

# The scheduling priority for daemons. (Default: 0)

SPARK\_NICENESS=0

export HADOOP\_HOME=${HADOOP\_HOME:-/usr/hdp/2.6.5.0-292/hadoop}

export HADOOP\_CONF\_DIR=${HADOOP\_CONF\_DIR:-/usr/hdp/2.6.5.0-292/hadoop/conf}

# The java implementation to use.

export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.222.b10-0.el7\_6.x86\_64

#Memory for Master, Worker and history server (default: 1024MB)

export SPARK\_DAEMON\_MEMORY=1024m

if [ -d "/etc/tez/conf/" ]; then

export TEZ\_CONF\_DIR=/etc/tez/conf

else

export TEZ\_CONF\_DIR=

fi

<http://rm01.itversity.com:19088/cluster/app/application_1589064448439_45773>

