```
import matplotlib
import matplotlib.pyplot as plt
import pyspark
import pandas as pd
import numpy as np
from IPython import get_ipython


import urllib
ACCESS_KEY = "FOO"
SECRET_KEY = "FOO"
ENCODED_SECRET_KEY = urllib.parse.quote(SECRET_KEY)
AWS_BUCKET_NAME = "ntestbucket5"
MOUNT_NAME = "s3bucket"
dbutils.fs.mount("s3n://%s:%s@%s" % (ACCESS_KEY, ENCODED_SECRET_KEY, AWS_BUCKET_NAME), "/mnt/%s" %
MOUNT_NAME)

Out[32]: True

display(dbutils.fs.ls("/mnt/s3bucket/"))
```

path	name
dbfs:/mnt/s3bucket/AIRS.2011.06.01.L3.CO2Std001.v5.9.2.7.X11209112519.hdf.txt	AIRS.2011.06.01.
dbfs:/mnt/s3bucket/ISCCP_HGG/	ISCCP_HGG/
dbfs:/mnt/s3bucket/Land-Co2-DATA.csv	Land-Co2-DATA.c
dbfs:/mnt/s3bucket/co2_Antartica-Modified.txt	co2_Antartica-Mo
dbfs:/mnt/s3bucket/co2_barrow-Alaska-Modified.txt	co2_barrow-Alask
dbfs:/mnt/s3bucket/co2_hawaii-Modified.txt	co2_hawaii-Modif
dbfs:/mnt/s3bucket/co2_moleFraction_PPM.csv	co2_moleFraction
dbfs:/mnt/s3bucket/co2_samoa-Modified.txt	co2_samoa-Modif
dbfs:/mnt/s3bucket/nick_isccp_test.csv	nick_isccp_test.cs

```
df_vamsi = spark.read.csv('/mnt/s3bucket
/vamsi_GlobalLandTemperaturesByCity.csv',header='true',inferSchema='true')

df_scott_orig = spark.read.csv('/mnt/s3bucket
/co2_moleFraction_PPM.csv',header='true',inferSchema='true')
df_scott1 = spark.read.csv('/mnt/s3bucket/Land-Co2-DATA.csv',header='true',inferSchema='true')

df_tessa = spark.read.csv('/mnt/s3bucket
/tessa_1_1995_2015_WorldBank.csv',header='true',inferSchema='true')
df_tessa1 = spark.read.csv('/mnt/s3bucket
/tessa_2_annual_country_lat_long_pm2.5_co2_ozone.csv',header='true',inferSchema='true')

df_nick = spark.read.csv('/mnt/s3bucket/ISCCP_HGG/*.csv',header='true',inferSchema='true')
```

```
df_scott_orig.registerTempTable("table_scott")
df_scott = spark.sql('select * from table_scott ts where (ts.Date >= to_date("1991-06-01") and
ts.Date <= to_date("1991-06-01")) or (ts.Date >= to_date("2001-06-01") and ts.Date <=
to_date("2001-06-01")) or (ts.Date >= to_date("2011-06-06") and ts.Date <= to_date("2011-06-06"))')
df_scott.registerTempTable("table_scott_p")
df_scott1.registerTempTable("table_scott_1")

df_tessa.registerTempTable("table_tessa")
df_tessa1.registerTempTable("table_tessa1")

df_nick.registerTempTable("table_nick")
df_vamsi.registerTempTable("table_vamsi")
df_vamsi_p = spark.sql('select * from table_vamsi ts where (ts.dt >= to_date("1991-06-01") and
ts.dt <= to_date("1991-06-01")) or (ts.dt >= to_date("2001-06-01") and ts.dt <=
to_date("2001-06-01")) or (ts.dt >= to_date("2011-06-06") and ts.dt <= to_date("2011-06-06"))')
df_vamsi_p.registerTempTable("table_vamsi_p")


print(df_scott_orig.count())
print(df_scott.count())
print(df_scott1.count())
print(df_tessa.count())
print(df_nick.count())
print(df_vamsi_p.count())

3708432
13104
1525292
4431
1289081
7020

def lat_udf(latitude):
    ##assuming N latitude and W longitude
    latitude = sum(float(x) / 60 ** n for n, x in enumerate(latitude[:-1].split('-')))  * (1 if 'N'
in latitude[-1] else -1)
    return latitude

def lon_udf(longitude):
    ##assuming N latitude and W longitude
    longitude = sum(float(x) / 60 ** n for n, x in enumerate(longitude[:-1].split('-'))) * (1 if
'E' in longitude[-1] else -1)
    return longitude

from pyspark.sql.types import FloatType
from pyspark.sql.functions import udf
lat_udf_float2 = udf(lambda z: lat_udf(z), FloatType())
lon_udf_float2 = udf(lambda z: lon_udf(z), FloatType())

df_vamsi_p = df_vamsi_p.select("*", lat_udf_float2('Latitude').alias('lat2'),
lon_udf_float2('longitude').alias('lon2') )
df_vamsi_p.registerTempTable("table_vamsi_p")
```

```
all_join = spark.sql('select /*+ RANGE_JOIN(table_nick, 10) */ tt.date, ts.Lattitude, ts.Longitude,
ts.PPM, tt.mean_cloud_top_temp_ir, tt.snow_ice_pct, tt.mean_cloud_optic_depth_ir, tt.cloud_amount,
va.AverageTemperature, va.AverageTemperatureUncertainty, te.Methane_emissions_kt_of_CO2_equiv,
te.PM2_5_air_pollution_mean_annual_exposure, te.Total_greenhouse_gas_emi_kt_of_CO2_equiv from
table_scott_p as ts, table_nick as tt, table_vamsi_p as va, table_tessa as te where ts.Date=tt.Date
and ts.Longitude Between tt.lon-0.5 and tt.lon+0.5 and ts.Lattitude Between tt.lat-0.5 and
tt.lat+0.5 and ts.Longitude Between va.lon2-0.5 and va.lon2+0.5 and ts.Lattitude Between
va.lat2-0.5 and va.lat2+0.5 and ts.Longitude Between te.long_avg-5.0 and te.long_avg+5.0 and
ts.Lattitude Between te.lat_avg-5.0 and te.lat_avg+5.0')
```

```
display(all_join)
```

Lattitude	Longitude	PPM	mean_cloud_top_temp_ir	snow_ice_pct	mean_cloud_optic_depth_ir	clo
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.

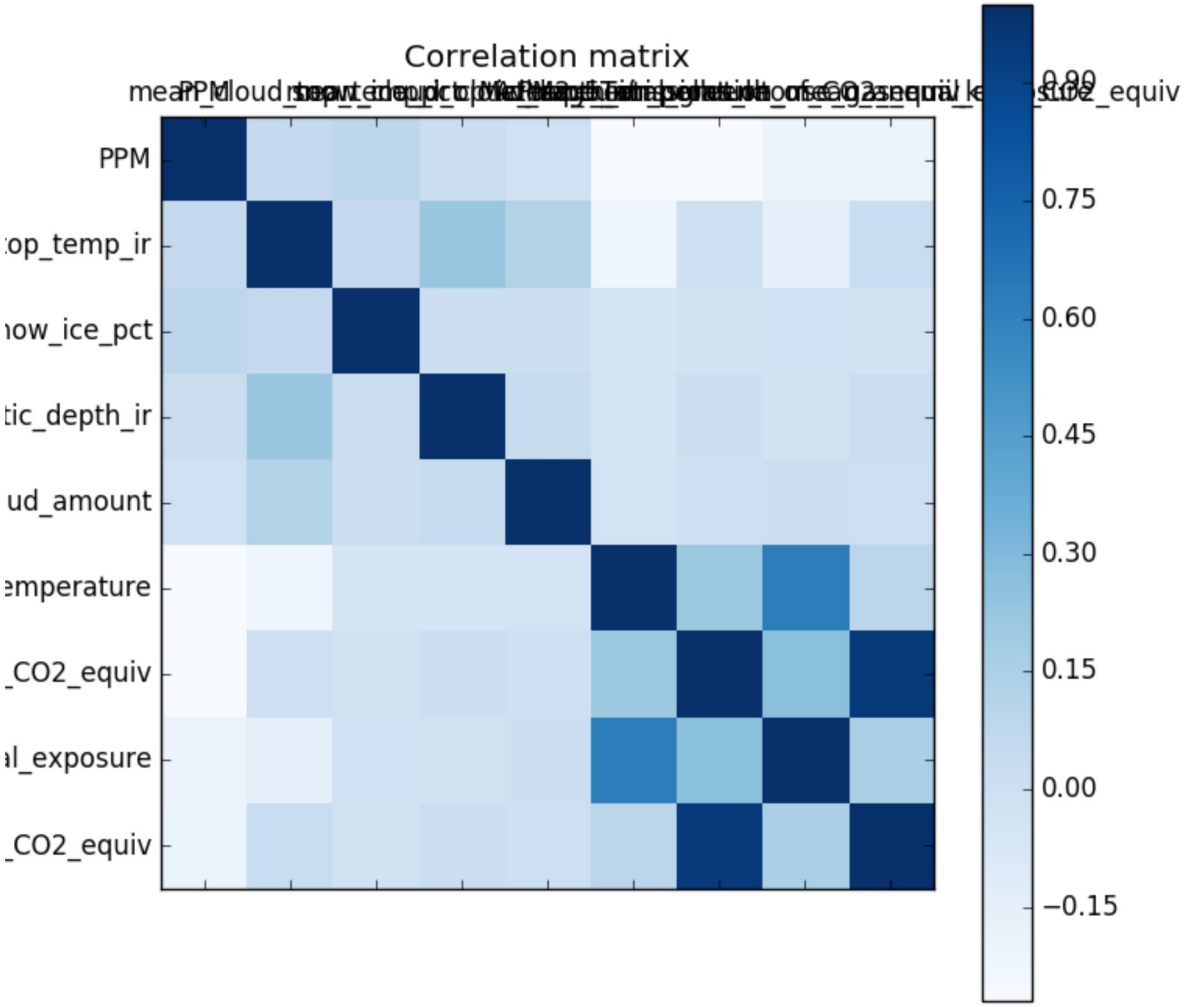
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.

38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.
38	35	394	279.5	0	-32767	37.

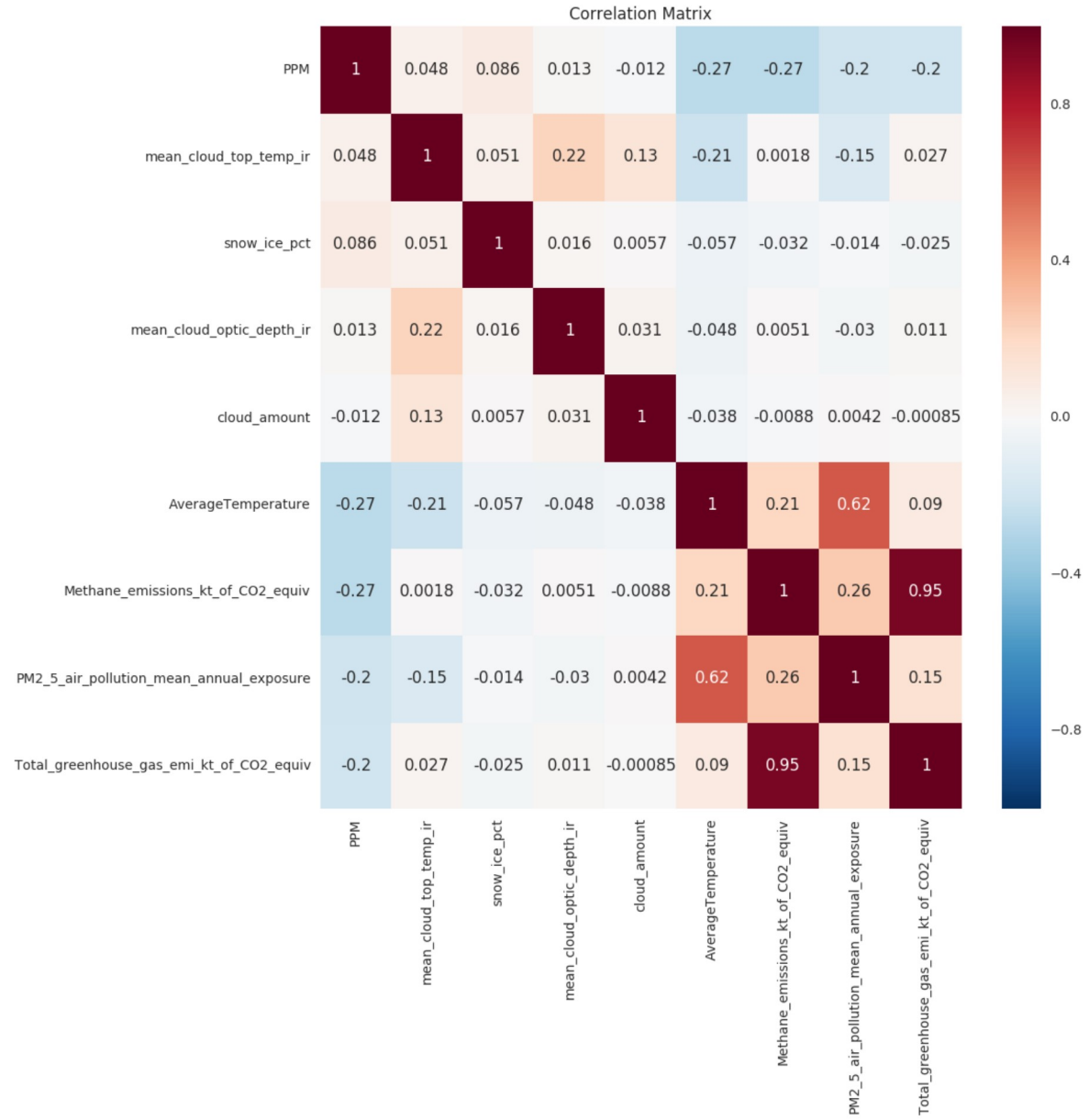
Showing the first 1000 rows.

```
#from pyspark.mllib.stat import Statistics
from pyspark.ml.linalg import Vectors
from pyspark.ml.stat import Correlation
from pyspark.ml.feature import VectorAssembler

col_list = list(["PPM","mean_cloud_top_temp_ir","snow_ice_pct", "mean_cloud_optic_depth_ir",
"cloud_amount", "AverageTemperature", "Methane_emissions_kt_of_CO2_equiv",
"PM2_5_air_pollution_mean_annual_exposure", "Total_greenhouse_gas_emi_kt_of_CO2_equiv"])
assembler = VectorAssembler(inputCols=col_list, outputCol="features", handleInvalid="skip")
#assembler = VectorAssembler(inputCols=all_join.columns, outputCol="features")
#assembler.transform(all_join).select("features").flatMap(lambda x: x)

corr1 = Correlation.corr(assembler.transform(all_join), "features").head()
# corr1 = Correlation.corr(all_join, ["ppm","mean_cloud_top_temp_ir","snow_ice_pct",
"mean_cloud_optic_depth_ir", "cloud_amount", "AverageTemperature", "AverageTemperatureUncertainty",
"Methane_emissions_kt_of_CO2_equiv", "PM2_5_air_pollution_mean_annual_exposure",
"Total_greenhouse_gas_emi_kt_of_CO2_equiv"])




"""
spark.sql.autoBroadcastJoinThreshold -1
spark.hadoop.fs.s3.impl com.databricks.s3a.S3AFileSystem
spark.hadoop.fs.s3a.acl.default BucketOwnerFullControl
spark.hadoop.fs.s3n.impl com.databricks.s3a.S3AFileSystem
spark.hadoop.fs.s3a.canned.acl BucketOwnerFullControl
spark.databricks.queryWatchdog.maxQueryTasks 200000
spark.speculation true
spark.hadoop.fs.s3a.impl com.databricks.s3a.S3AFileSystem
spark.task.reaper.enabled true
spark.task.reaper.killTimeout 3m
"""


corr1

Out[195]: Row(pearson(features)=DenseMatrix(9, 9, [1.0, 0.0476, 0.0857, 0.0127, -0.0122, -0.27, -0
.2681, -0.2044, ..., 0.0268, -0.0253, 0.0111, -0.0008, 0.0899, 0.9514, 0.1522, 1.0], False))

type(assembler.transform(all_join))
```

```
Out[143]: pyspark.sql.dataframe.DataFrame

dm = corr1.asDict()['pearson(features)']


#dm.toArray()


import matplotlib.pyplot as plt
from matplotlib import ticker

fig = plt.figure(figsize=(8,8))
ax = fig.add_subplot(111)

labels = ["PPM","mean_cloud_top_temp_ir","snow_ice_pct", "mean_cloud_optic_depth_ir",
"cloud_amount", "AverageTemperature", "Methane_emissions_kt_of_CO2_equiv",
"PM2_5_air_pollution_mean_annual_exposure", "Total_greenhouse_gas_emi_kt_of_CO2_equiv"]
# I also added cmap=cmap here, to make use of the
# colormap you specify in the function call
cax = ax.matshow(dm.toArray(),cmap=plt.cm.Blues)
plt.title('Correlation matrix')
fig.colorbar(cax)
if labels:
   ax.set_xticklabels(['']  labels)
   ax.set_yticklabels(['']  labels)

ax.xaxis.set_major_locator(ticker.MultipleLocator(1))
ax.yaxis.set_major_locator(ticker.MultipleLocator(1))
#plt.tight_layout()
#plt.gcf().subplots_adjust(bottom=0.15, left=.1, right=.11)
#plt.xlabel('Predicted')
#plt.setp('foo', rotation=45)
#plt.ylabel('True')
#plt.show()
# plt.savefig('confusionmatrix.png')
display(fig)
```




```
import seaborn as sns

fig = plt.figure(figsize=(12,12))
ax=plt.subplot()
sns.heatmap(dm.toArray(), annot=True, ax = ax)

# labels, title and ticks
#ax.set_xlabel('Predicted labels');
#ax.set_ylabel('True labels');
ax.set_title('Correlation Matrix')
ax.xaxis.set_ticklabels(list(reversed(labels)), rotation=90)
ax.yaxis.set_ticklabels(labels, rotation=0)
plt.tight_layout()
display(fig)
```




```
labels.reverse()
```

```
Out[194]:
['PPM',
 'mean_cloud_top_temp_ir',
 'snow_ice_pct',
 'mean_cloud_optic_depth_ir',
 'cloud_amount',
 'AverageTemperature',
 'Methane_emissions_kt_of_CO2_equiv',
 'PM2_5_air_pollution_mean_annual_exposure',
 'Total_greenhouse_gas_emi_kt_of_CO2_equiv']
```