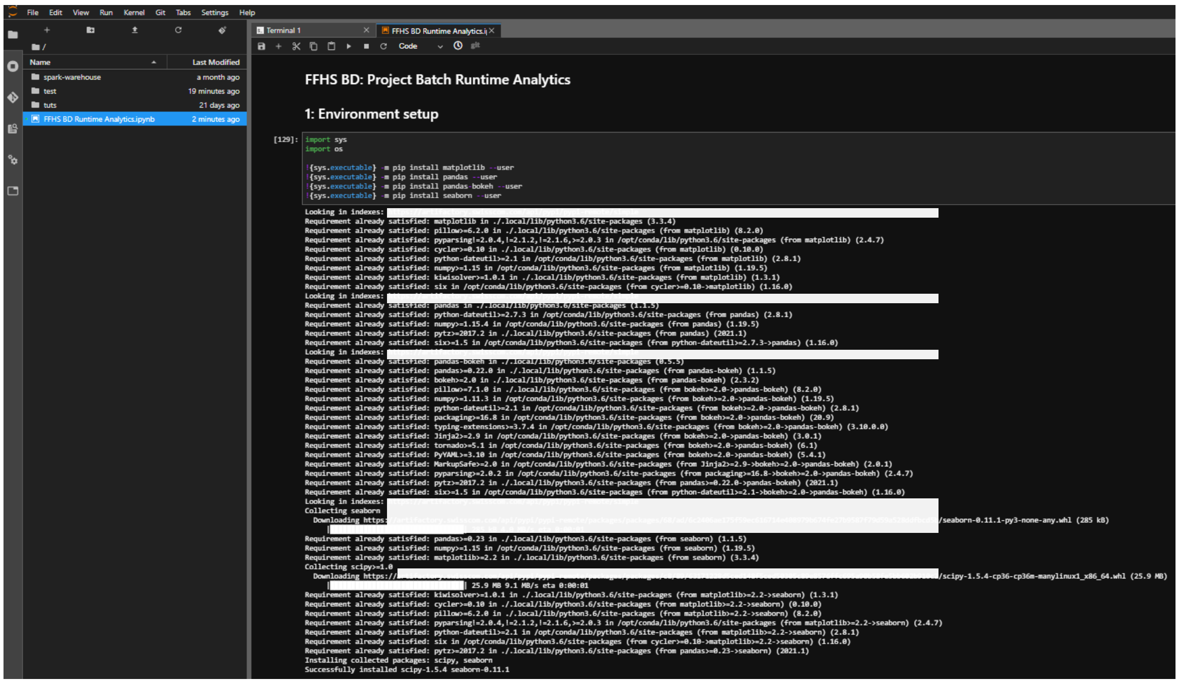
Code, partly anonymized



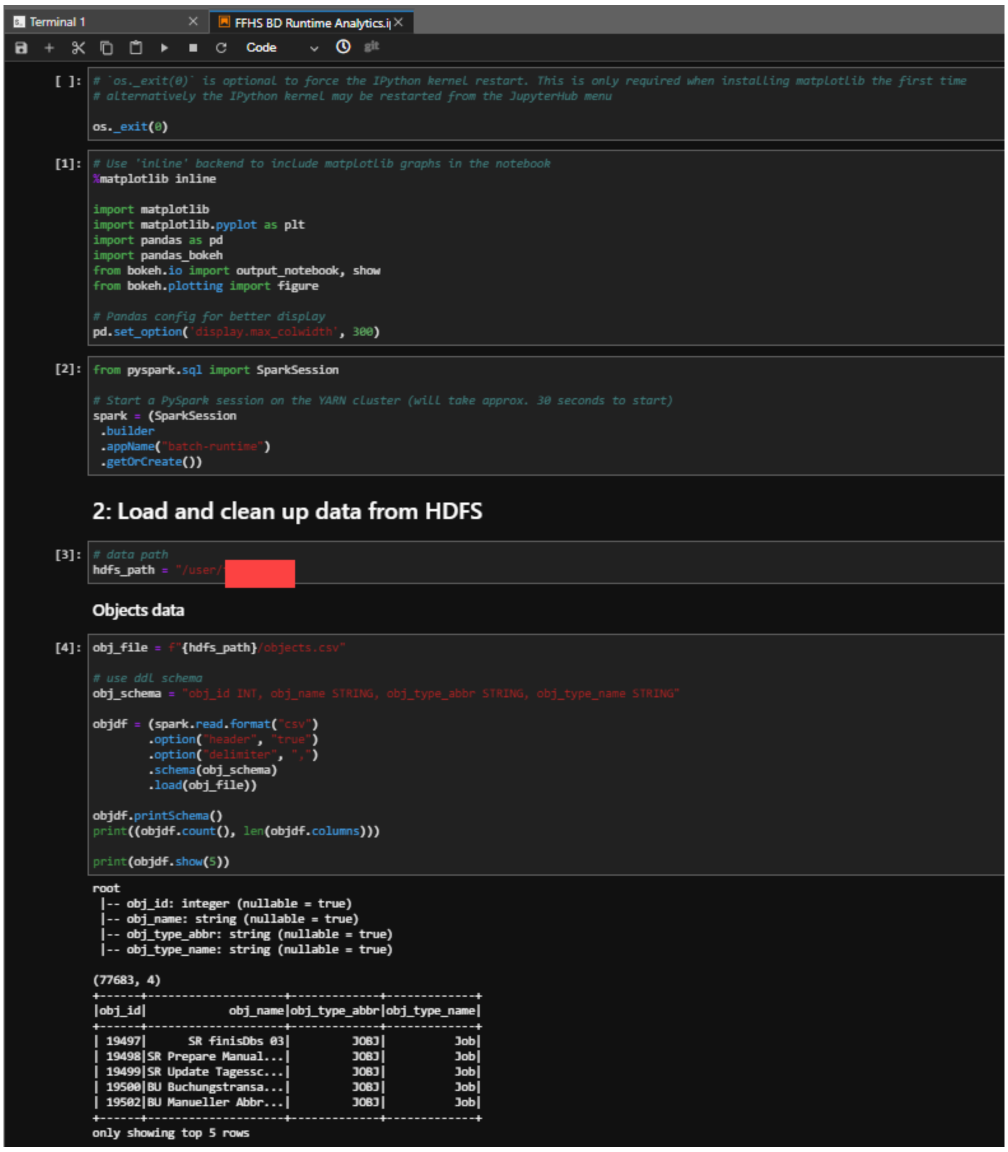
Name	Last Modified
spark-warehouse	a month ago
test	19 minutes ago
tuts	21 days ago
FFHS BD Runtime Analytics.ipynb	2 minutes ago

FFHS BD: Project Batch Runtime Analytics

1: Environment setup

```
[129]: import sys
       import os

       !{sys.executable} -m pip install matplotlib --user
       !{sys.executable} -m pip install pandas --user
       !{sys.executable} -m pip install pandas-bokeh --user
       !{sys.executable} -m pip install seaborn --user
```

```
Looking in indexes: [redacted]
Requirement already satisfied: matplotlib in ./.local/lib/python3.6/site-packages (3.3.4)
Requirement already satisfied: pillow>=6.2.0 in ./.local/lib/python3.6/site-packages (from matplotlib) (8.2.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in /opt/conda/lib/python3.6/site-packages (from matplotlib) (2.4.7)
Requirement already satisfied: cycler>=0.10 in ./.local/lib/python3.6/site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in /opt/conda/lib/python3.6/site-packages (from matplotlib) (2.8.1)
Requirement already satisfied: numpy>=1.15 in /opt/conda/lib/python3.6/site-packages (from matplotlib) (1.19.5)
Requirement already satisfied: kiwisolver>=1.0.1 in ./.local/lib/python3.6/site-packages (from matplotlib) (1.3.1)
Requirement already satisfied: six in /opt/conda/lib/python3.6/site-packages (from cycler>=0.10->matplotlib) (1.16.0)
Looking in indexes: [redacted]
Requirement already satisfied: pandas in ./.local/lib/python3.6/site-packages (1.1.5)
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/lib/python3.6/site-packages (from pandas) (2.8.1)
Requirement already satisfied: numpy>=1.15.4 in /opt/conda/lib/python3.6/site-packages (from pandas) (1.19.5)
Requirement already satisfied: pytz>=2017.2 in ./.local/lib/python3.6/site-packages (from pandas) (2021.1)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.6/site-packages (from python-dateutil>=2.7.3->pandas) (1.16.0)
Looking in indexes: [redacted]
Requirement already satisfied: pandas-bokeh in ./.local/lib/python3.6/site-packages (0.5.5)
Requirement already satisfied: pandas>=0.22.0 in ./.local/lib/python3.6/site-packages (from pandas-bokeh) (1.1.5)
Requirement already satisfied: bokeh>=2.0 in ./.local/lib/python3.6/site-packages (from pandas-bokeh) (2.3.2)
Requirement already satisfied: pillow>=7.1.0 in ./.local/lib/python3.6/site-packages (from bokeh>=2.0->pandas-bokeh) (8.2.0)
Requirement already satisfied: numpy>=1.11.3 in /opt/conda/lib/python3.6/site-packages (from bokeh>=2.0->pandas-bokeh) (1.19.5)
Requirement already satisfied: python-dateutil>=2.1 in /opt/conda/lib/python3.6/site-packages (from bokeh>=2.0->pandas-bokeh) (2.8.1)
Requirement already satisfied: packaging>=16.8 in /opt/conda/lib/python3.6/site-packages (from bokeh>=2.0->pandas-bokeh) (20.9)
Requirement already satisfied: typing-extensions>=3.7.4 in /opt/conda/lib/python3.6/site-packages (from bokeh>=2.0->pandas-bokeh) (3.10.0.0)
Requirement already satisfied: Jinja2>=2.9 in /opt/conda/lib/python3.6/site-packages (from bokeh>=2.0->pandas-bokeh) (3.0.1)
Requirement already satisfied: tornado>=5.1 in /opt/conda/lib/python3.6/site-packages (from bokeh>=2.0->pandas-bokeh) (6.1)
Requirement already satisfied: PyYAML>=3.10 in /opt/conda/lib/python3.6/site-packages (from bokeh>=2.0->pandas-bokeh) (5.4.1)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.6/site-packages (from Jinja2>=2.9->bokeh>=2.0->pandas-bokeh) (2.0.1)
Requirement already satisfied: pyparsing>=2.0.2 in /opt/conda/lib/python3.6/site-packages (from packaging>=16.8->bokeh>=2.0->pandas-bokeh) (2.4.7)
Requirement already satisfied: pytz>=2017.2 in ./.local/lib/python3.6/site-packages (from pandas>=0.22.0->pandas-bokeh) (2021.1)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.6/site-packages (from python-dateutil>=2.1->bokeh>=2.0->pandas-bokeh) (1.16.0)
Looking in indexes: [redacted]
Collecting seaborn
  Downloading https:[redacted]/seaborn-0.11.1-py3-none-any.whl (285 kB)
Requirement already satisfied: pandas>=0.23 in ./.local/lib/python3.6/site-packages (from seaborn) (1.1.5)
Requirement already satisfied: numpy>=1.15 in /opt/conda/lib/python3.6/site-packages (from seaborn) (1.19.5)
Requirement already satisfied: matplotlib>=2.2 in ./.local/lib/python3.6/site-packages (from seaborn) (3.3.4)
Collecting scipy>=1.0
  Downloading https://[redacted]/scipy-1.5.4-cp36-cp36m-manylinux1_x86_64.whl (25.9 MB)
     |████████████████████████████████| 25.9 MB 9.1 MB/s eta 0:00:01
Requirement already satisfied: kiwisolver>=1.0.1 in ./.local/lib/python3.6/site-packages (from matplotlib>=2.2->seaborn) (1.3.1)
Requirement already satisfied: cycler>=0.10 in ./.local/lib/python3.6/site-packages (from matplotlib>=2.2->seaborn) (0.10.0)
Requirement already satisfied: pillow>=6.2.0 in ./.local/lib/python3.6/site-packages (from matplotlib>=2.2->seaborn) (8.2.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in /opt/conda/lib/python3.6/site-packages (from matplotlib>=2.2->seaborn) (2.4.7)
Requirement already satisfied: python-dateutil>=2.1 in /opt/conda/lib/python3.6/site-packages (from matplotlib>=2.2->seaborn) (2.8.1)
Requirement already satisfied: six in /opt/conda/lib/python3.6/site-packages (from cycler>=0.10->matplotlib>=2.2->seaborn) (1.16.0)
Requirement already satisfied: pytz>=2017.2 in ./.local/lib/python3.6/site-packages (from pandas>=0.23->seaborn) (2021.1)
Installing collected packages: scipy, seaborn
Successfully installed scipy-1.5.4 seaborn-0.11.1
```



```
[ ]: # `os._exit(0)` is optional to force the IPython kernel restart. This is only required when installing matplotlib the first time
     # alternatively the IPython kernel may be restarted from the JupyterHub menu

     os._exit(0)
```

```
[1]: # Use 'inline' backend to include matplotlib graphs in the notebook
     %matplotlib inline

     import matplotlib
     import matplotlib.pyplot as plt
     import pandas as pd
     import pandas_bokeh
     from bokeh.io import output_notebook, show
     from bokeh.plotting import figure

     # Pandas config for better display
     pd.set_option('display.max_colwidth', 300)
```

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

     # Start a PySpark session on the YARN cluster (will take approx. 30 seconds to start)
     spark = (SparkSession
      .builder
      .appName("batch-runtime")
      .getOrCreate())
```

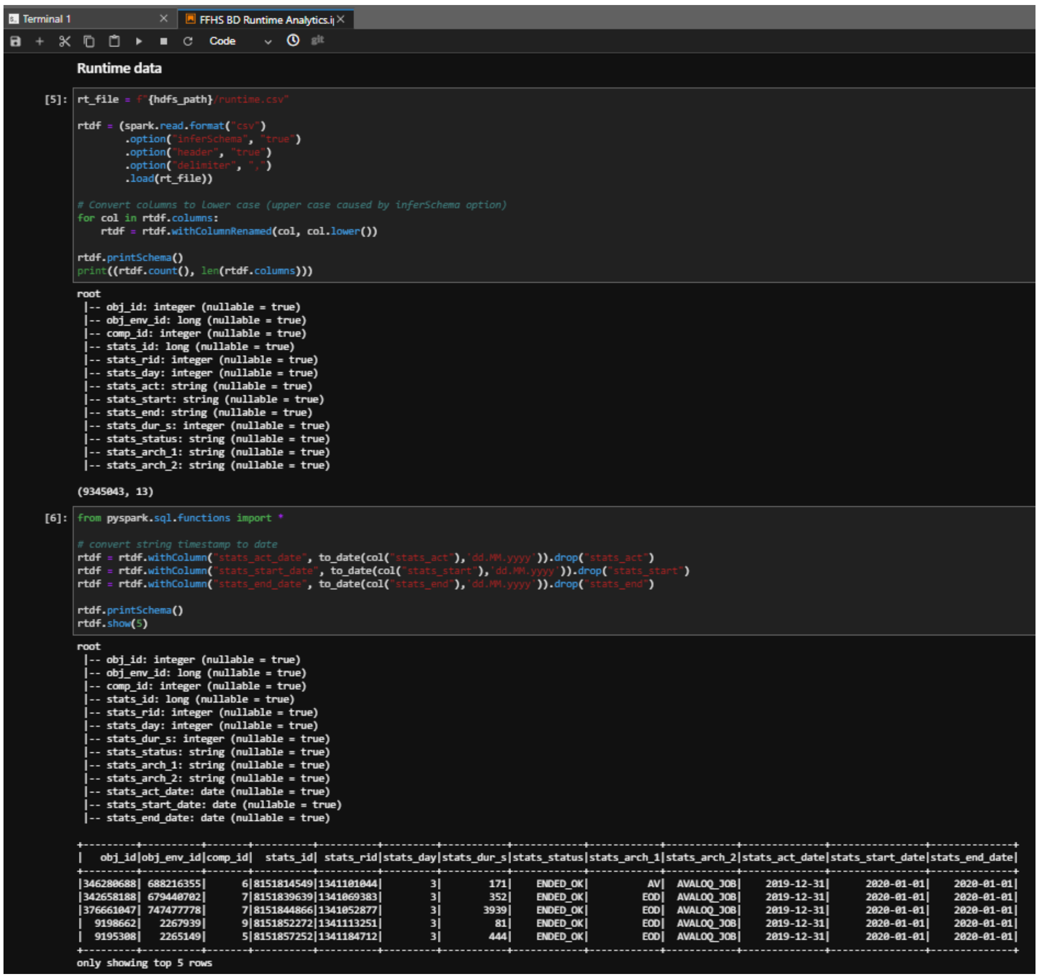
2: Load and clean up data from HDFS

```
[3]: # data path
     hdfs_path = "/user/[redacted]
```

Objects data

```
[4]: obj_file = f"{hdfs_path}/objects.csv"

     # use ddl schema
     obj_schema = "obj_id INT, obj_name STRING, obj_type_abbr STRING, obj_type_name STRING"

     objdf = (spark.read.format("csv")
             .option("header", "true")
             .option("delimiter", ",")
             .schema(obj_schema)
             .load(obj_file))

     objdf.printSchema()
     print((objdf.count(), len(objdf.columns)))

     print(objdf.show(5))
```

```
root
 |-- obj_id: integer (nullable = true)
 |-- obj_name: string (nullable = true)
 |-- obj_type_abbr: string (nullable = true)
 |-- obj_type_name: string (nullable = true)

(77683, 4)
+------+--------------------+-------------+-------------+
|obj_id|            obj_name|obj_type_abbr|obj_type_name|
+------+--------------------+-------------+-------------+
| 19497|       SR finisDbs 03|         JOBJ|          Job|
| 19498|SR Prepare Manual...|         JOBJ|          Job|
| 19499|SR Update Tagessc...|         JOBJ|          Job|
| 19500|BU Buchungstransa...|         JOBJ|          Job|
| 19502|BU Manueller Abbr...|         JOBJ|          Job|
+------+--------------------+-------------+-------------+
only showing top 5 rows
```



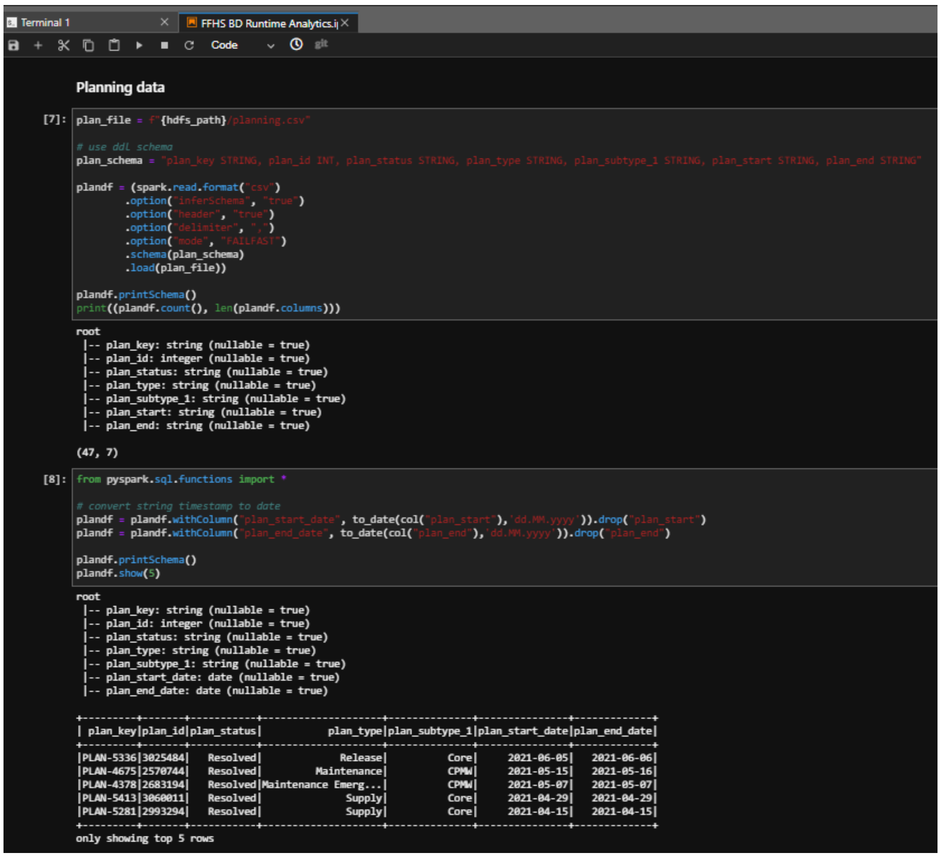
Runtime data

```
[5]: rt_file = f"{hdfs_path}/runtime.csv"

     rtdf = (spark.read.format("csv")
             .option("inferSchema", "true")
             .option("header", "true")
             .option("delimiter", ",")
             .load(rt_file))

     # Convert columns to lower case (upper case caused by inferSchema option)
     for col in rtdf.columns:
         rtdf = rtdf.withColumnRenamed(col, col.lower())

     rtdf.printSchema()
     print((rtdf.count(), len(rtdf.columns)))
```

```
root
 |-- obj_id: integer (nullable = true)
 |-- obj_env_id: long (nullable = true)
 |-- comp_id: integer (nullable = true)
 |-- stats_id: long (nullable = true)
 |-- stats_rid: integer (nullable = true)
 |-- stats_day: integer (nullable = true)
 |-- stats_act: string (nullable = true)
 |-- stats_start: string (nullable = true)
 |-- stats_end: string (nullable = true)
 |-- stats_dur_s: integer (nullable = true)
 |-- stats_status: string (nullable = true)
 |-- stats_arch_1: string (nullable = true)
 |-- stats_arch_2: string (nullable = true)

(9345043, 13)
```

```
[6]: from pyspark.sql.functions import *

     # convert string timestamp to date
     rtdf = rtdf.withColumn("stats_act_date", to_date(col("stats_act"),'dd.MM.yyyy')).drop("stats_act")
     rtdf = rtdf.withColumn("stats_start_date", to_date(col("stats_start"),'dd.MM.yyyy')).drop("stats_start")
     rtdf = rtdf.withColumn("stats_end_date", to_date(col("stats_end"),'dd.MM.yyyy')).drop("stats_end")

     rtdf.printSchema()
     rtdf.show(5)
```

```
root
 |-- obj_id: integer (nullable = true)
 |-- obj_env_id: long (nullable = true)
 |-- comp_id: integer (nullable = true)
 |-- stats_id: long (nullable = true)
 |-- stats_rid: integer (nullable = true)
 |-- stats_day: integer (nullable = true)
 |-- stats_dur_s: integer (nullable = true)
 |-- stats_status: string (nullable = true)
 |-- stats_arch_1: string (nullable = true)
 |-- stats_arch_2: string (nullable = true)
 |-- stats_act_date: date (nullable = true)
 |-- stats_start_date: date (nullable = true)
 |-- stats_end_date: date (nullable = true)

+---------+----------+-------+----------+----------+---------+-----------+------------+------------+------------+--------------+----------------+--------------+
|   obj_id|obj_env_id|comp_id|  stats_id| stats_rid|stats_day|stats_dur_s|stats_status|stats_arch_1|stats_arch_2|stats_act_date|stats_start_date|stats_end_date|
+---------+----------+-------+----------+----------+---------+-----------+------------+------------+------------+--------------+----------------+--------------+
|346280688| 688216355|      6|8151814549|1341101044|        3|        171|    ENDED_OK|          AV|  AVALOQ_JOB|    2019-12-31|      2020-01-01|    2020-01-01|
|342658188| 679440702|      7|8151839639|1341069383|        3|        352|    ENDED_OK|         EOD|  AVALOQ_JOB|    2019-12-31|      2020-01-01|    2020-01-01|
|376661047| 747477778|      7|8151844866|1341052877|        3|       3939|    ENDED_OK|         EOD|  AVALOQ_JOB|    2019-12-31|      2020-01-01|    2020-01-01|
|  9198662|   2267939|      9|8151852272|1341113251|        3|         81|    ENDED_OK|         EOD|  AVALOQ_JOB|    2019-12-31|      2020-01-01|    2020-01-01|
|  9195308|   2265149|      5|8151857252|1341184712|        3|        444|    ENDED_OK|         EOD|  AVALOQ_JOB|    2019-12-31|      2020-01-01|    2020-01-01|
+---------+----------+-------+----------+----------+---------+-----------+------------+------------+------------+--------------+----------------+--------------+
only showing top 5 rows
```



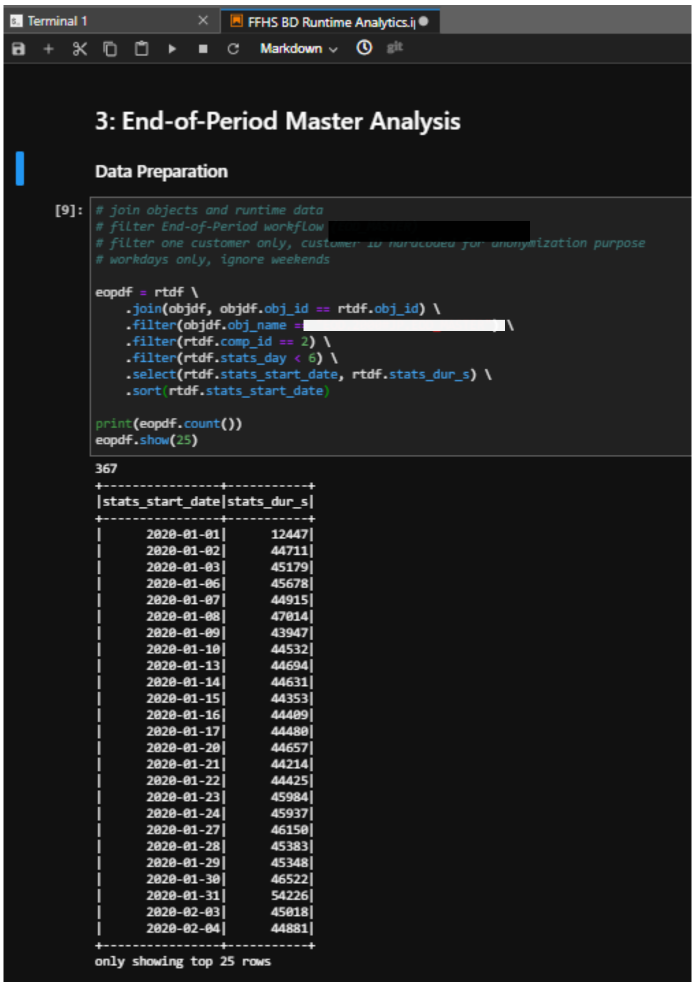
Planning data

```
[7]: plan_file = f"{hdfs_path}/planning.csv"

     # use ddl schema
     plan_schema = "plan_key STRING, plan_id INT, plan_status STRING, plan_type STRING, plan_subtype_1 STRING, plan_start STRING, plan_end STRING"

     plandf = (spark.read.format("csv")
             .option("inferSchema", "true")
             .option("header", "true")
             .option("delimiter", ",")
             .option("mode", "FAILFAST")
             .schema(plan_schema)
             .load(plan_file))

     plandf.printSchema()
     print((plandf.count(), len(plandf.columns)))
```

```
root
 |-- plan_key: string (nullable = true)
 |-- plan_id: integer (nullable = true)
 |-- plan_status: string (nullable = true)
 |-- plan_type: string (nullable = true)
 |-- plan_subtype_1: string (nullable = true)
 |-- plan_start: string (nullable = true)
 |-- plan_end: string (nullable = true)

(47, 7)
```

```
[8]: from pyspark.sql.functions import *

     # convert string timestamp to date
     plandf = plandf.withColumn("plan_start_date", to_date(col("plan_start"),'dd.MM.yyyy')).drop("plan_start")
     plandf = plandf.withColumn("plan_end_date", to_date(col("plan_end"),'dd.MM.yyyy')).drop("plan_end")

     plandf.printSchema()
     plandf.show(5)
```

```
root
 |-- plan_key: string (nullable = true)
 |-- plan_id: integer (nullable = true)
 |-- plan_status: string (nullable = true)
 |-- plan_type: string (nullable = true)
 |-- plan_subtype_1: string (nullable = true)
 |-- plan_start_date: date (nullable = true)
 |-- plan_end_date: date (nullable = true)

+---------+-------+-----------+--------------------+--------------+---------------+-------------+
| plan_key|plan_id|plan_status|           plan_type|plan_subtype_1|plan_start_date|plan_end_date|
+---------+-------+-----------+--------------------+--------------+---------------+-------------+
|PLAN-5336|3025484|   Resolved|             Release|          Core|     2021-06-05|   2021-06-06|
|PLAN-4675|2570744|   Resolved|         Maintenance|          CPMW|     2021-05-15|   2021-05-16|
|PLAN-4378|2683194|   Resolved|Maintenance Emerg...|          CPMW|     2021-05-07|   2021-05-07|
|PLAN-5413|3060011|   Resolved|              Supply|          Core|     2021-04-29|   2021-04-29|
|PLAN-5281|2993294|   Resolved|              Supply|          Core|     2021-04-15|   2021-04-15|
+---------+-------+-----------+--------------------+--------------+---------------+-------------+
only showing top 5 rows
```

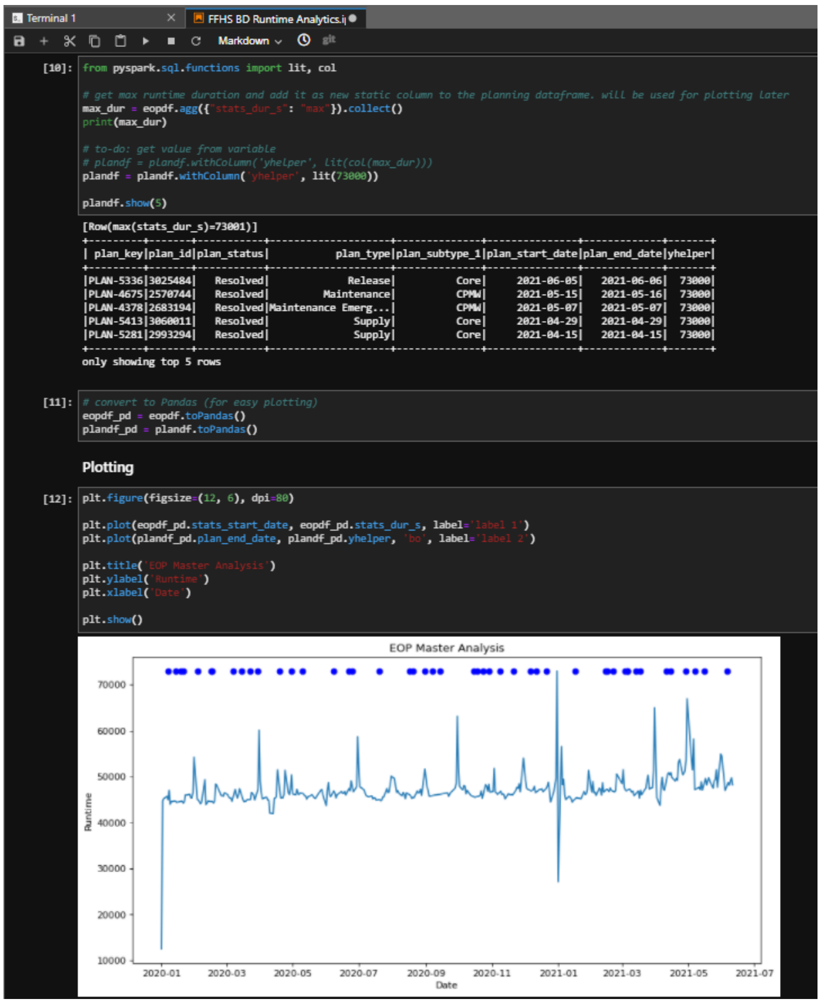


3: End-of-Period Master Analysis

Data Preparation

```
[9]: # join objects and runtime data
     # filter End-of-Period workflow [redacted]
     # filter one customer only, customer ID hardcoded for anonymization purpose
     # workdays only, ignore weekends

     eopdf = rtdf \
         .join(objdf, objdf.obj_id == rtdf.obj_id) \
         .filter(objdf.obj_name = [redacted] \
         .filter(rtdf.comp_id == 2) \
         .filter(rtdf.stats_day < 6) \
         .select(rtdf.stats_start_date, rtdf.stats_dur_s) \
         .sort(rtdf.stats_start_date)

     print(eopdf.count())
     eopdf.show(25)
```

```
367
+----------------+-----------+
|stats_start_date|stats_dur_s|
+----------------+-----------+
|      2020-01-01|      12447|
|      2020-01-02|      44711|
|      2020-01-03|      45179|
|      2020-01-06|      45678|
|      2020-01-07|      44915|
|      2020-01-08|      47014|
|      2020-01-09|      43947|
|      2020-01-10|      44532|
|      2020-01-13|      44694|
|      2020-01-14|      44631|
|      2020-01-15|      44353|
|      2020-01-16|      44409|
|      2020-01-17|      44480|
|      2020-01-20|      44657|
|      2020-01-21|      44214|
|      2020-01-22|      44425|
|      2020-01-23|      45984|
|      2020-01-24|      45937|
|      2020-01-27|      46150|
|      2020-01-28|      45383|
|      2020-01-29|      45348|
|      2020-01-30|      46522|
|      2020-01-31|      54226|
|      2020-02-03|      45018|
|      2020-02-04|      44881|
+----------------+-----------+
only showing top 25 rows
```



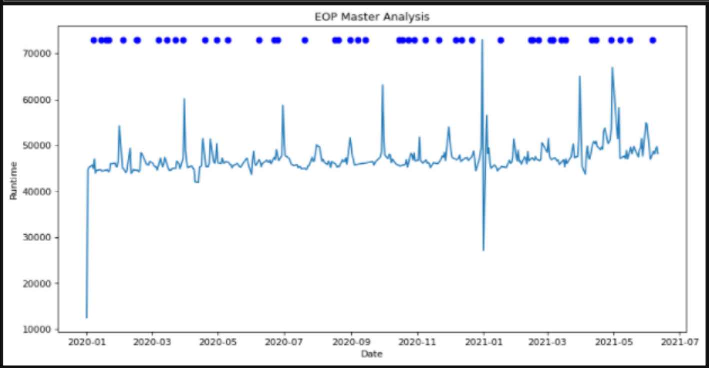
```
[10]: from pyspark.sql.functions import lit, col

      # get max runtime duration and add it as new static column to the planning dataframe. will be used for plotting later
      max_dur = eopdf.agg({"stats_dur_s": "max"}).collect()
      print(max_dur)

      # to-do: get value from variable
      # plandf = plandf.withColumn('yhelper', lit(col(max_dur)))
      plandf = plandf.withColumn('yhelper', lit(73000))

      plandf.show(5)
```

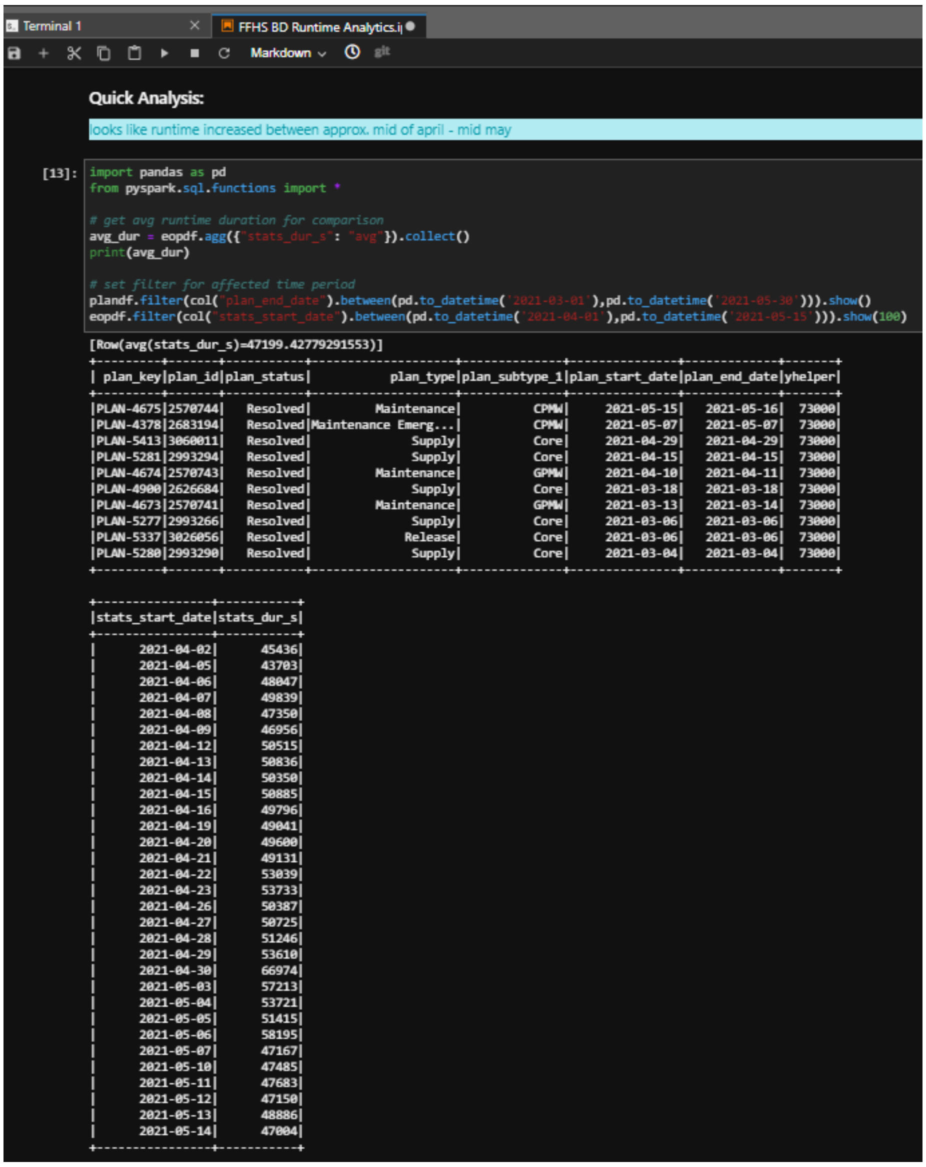
```
[Row(max(stats_dur_s)=73001)]
+---------+-------+-----------+--------------------+--------------+---------------+-------------+-------+
| plan_key|plan_id|plan_status|           plan_type|plan_subtype_1|plan_start_date|plan_end_date|yhelper|
+---------+-------+-----------+--------------------+--------------+---------------+-------------+-------+
|PLAN-5336|3025484|   Resolved|             Release|          Core|     2021-06-05|   2021-06-06|  73000|
|PLAN-4675|2570744|   Resolved|         Maintenance|          CPMW|     2021-05-15|   2021-05-16|  73000|
|PLAN-4378|2683194|   Resolved|Maintenance Emerg...|          CPMW|     2021-05-07|   2021-05-07|  73000|
|PLAN-5413|3060011|   Resolved|              Supply|          Core|     2021-04-29|   2021-04-29|  73000|
|PLAN-5281|2993294|   Resolved|              Supply|          Core|     2021-04-15|   2021-04-15|  73000|
+---------+-------+-----------+--------------------+--------------+---------------+-------------+-------+
only showing top 5 rows
```

```
[11]: # convert to Pandas (for easy plotting)
      eopdf_pd = eopdf.toPandas()
      plandf_pd = plandf.toPandas()
```

Plotting

```
[12]: plt.figure(figsize=(12, 6), dpi=80)

      plt.plot(eopdf_pd.stats_start_date, eopdf_pd.stats_dur_s, label='label 1')
      plt.plot(plandf_pd.plan_end_date, plandf_pd.yhelper, 'bo', label='label 2')

      plt.title('EOP Master Analysis')
      plt.ylabel('Runtime')
      plt.xlabel('Date')

      plt.show()
```



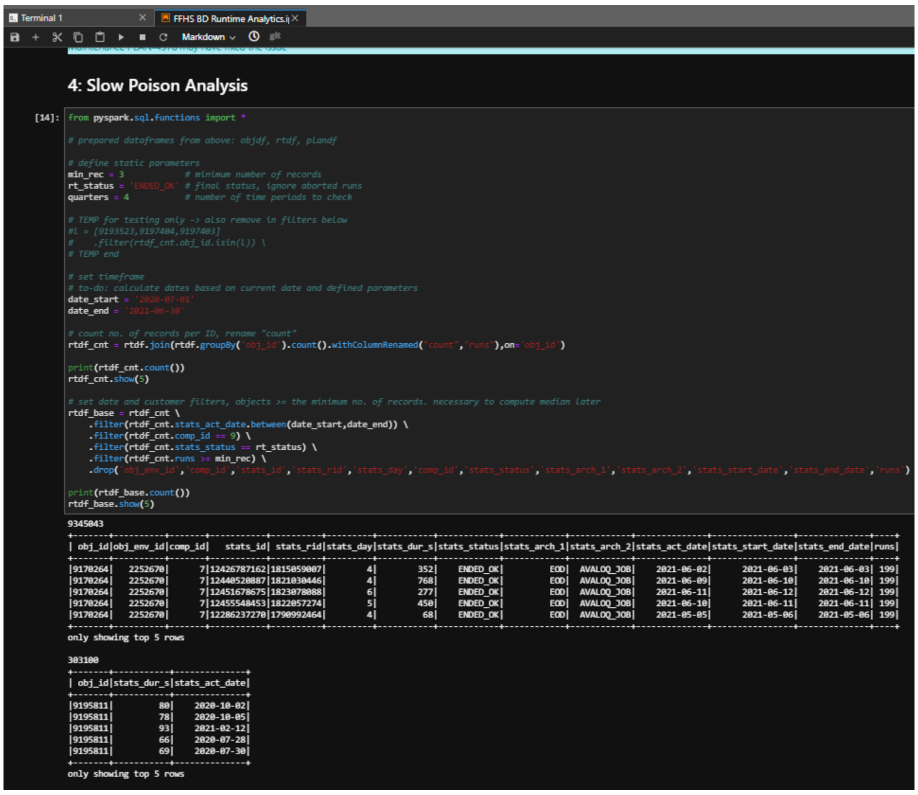


Quick Analysis:

looks like runtime increased between approx. mid of april - mid may

```
[13]: import pandas as pd
      from pyspark.sql.functions import *

      # get avg runtime duration for comparison
      avg_dur = eopdf.agg({"stats_dur_s": "avg"}).collect()
      print(avg_dur)

      # set filter for affected time period
      plandf.filter(col('plan_end_date').between(pd.to_datetime('2021-03-01'),pd.to_datetime('2021-05-30'))).show()
      eopdf.filter(col("stats_start_date").between(pd.to_datetime('2021-04-01'),pd.to_datetime('2021-05-15'))).show(100)
```

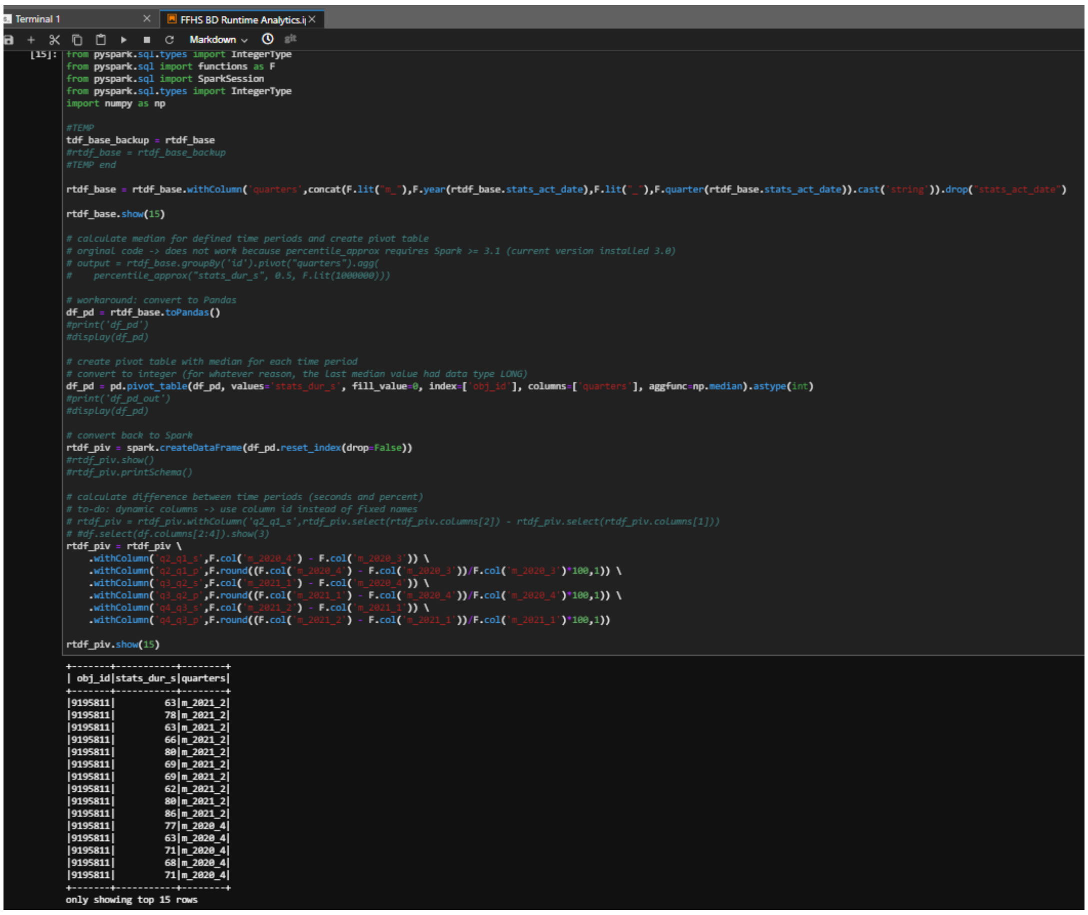
```
[Row(avg(stats_dur_s)=47199.42779291553)]
+---------+-------+-----------+-----------------+--------------+---------------+-------------+------+
| plan_key|plan_id|plan_status|        plan_type|plan_subtype_1|plan_start_date|plan_end_date|yhelper|
+---------+-------+-----------+-----------------+--------------+---------------+-------------+------+
|PLAN-4675|2570744|   Resolved|      Maintenance|          CPMW|     2021-05-15|   2021-05-16| 73000|
|PLAN-4378|2683194|   Resolved|Maintenance Emerg...|       CPMW|     2021-05-07|   2021-05-07| 73000|
|PLAN-5413|3060011|   Resolved|           Supply|          Core|     2021-04-29|   2021-04-29| 73000|
|PLAN-5281|2993294|   Resolved|           Supply|          Core|     2021-04-15|   2021-04-15| 73000|
|PLAN-4674|2570743|   Resolved|      Maintenance|          GPMW|     2021-04-10|   2021-04-11| 73000|
|PLAN-4900|2626684|   Resolved|           Supply|          Core|     2021-03-18|   2021-03-18| 73000|
|PLAN-4673|2570741|   Resolved|      Maintenance|          GPMW|     2021-03-13|   2021-03-14| 73000|
|PLAN-5277|2993266|   Resolved|           Supply|          Core|     2021-03-06|   2021-03-06| 73000|
|PLAN-5337|3026056|   Resolved|          Release|          Core|     2021-03-06|   2021-03-06| 73000|
|PLAN-5280|2993290|   Resolved|           Supply|          Core|     2021-03-04|   2021-03-04| 73000|
+---------+-------+-----------+-----------------+--------------+---------------+-------------+------+

+----------------+-----------+
|stats_start_date|stats_dur_s|
+----------------+-----------+
|      2021-04-02|      45436|
|      2021-04-05|      43703|
|      2021-04-06|      48047|
|      2021-04-07|      49839|
|      2021-04-08|      47350|
|      2021-04-09|      46956|
|      2021-04-12|      50515|
|      2021-04-13|      50836|
|      2021-04-14|      50350|
|      2021-04-15|      50885|
|      2021-04-16|      49796|
|      2021-04-19|      49041|
|      2021-04-20|      49600|
|      2021-04-21|      49131|
|      2021-04-22|      53039|
|      2021-04-23|      53733|
|      2021-04-26|      50387|
|      2021-04-27|      50725|
|      2021-04-28|      51246|
|      2021-04-29|      53610|
|      2021-04-30|      66974|
|      2021-05-03|      57213|
|      2021-05-04|      53721|
|      2021-05-05|      51415|
|      2021-05-06|      58195|
|      2021-05-07|      47167|
|      2021-05-10|      47485|
|      2021-05-11|      47683|
|      2021-05-12|      47150|
|      2021-05-13|      48886|
|      2021-05-14|      47004|
+----------------+-----------+
```



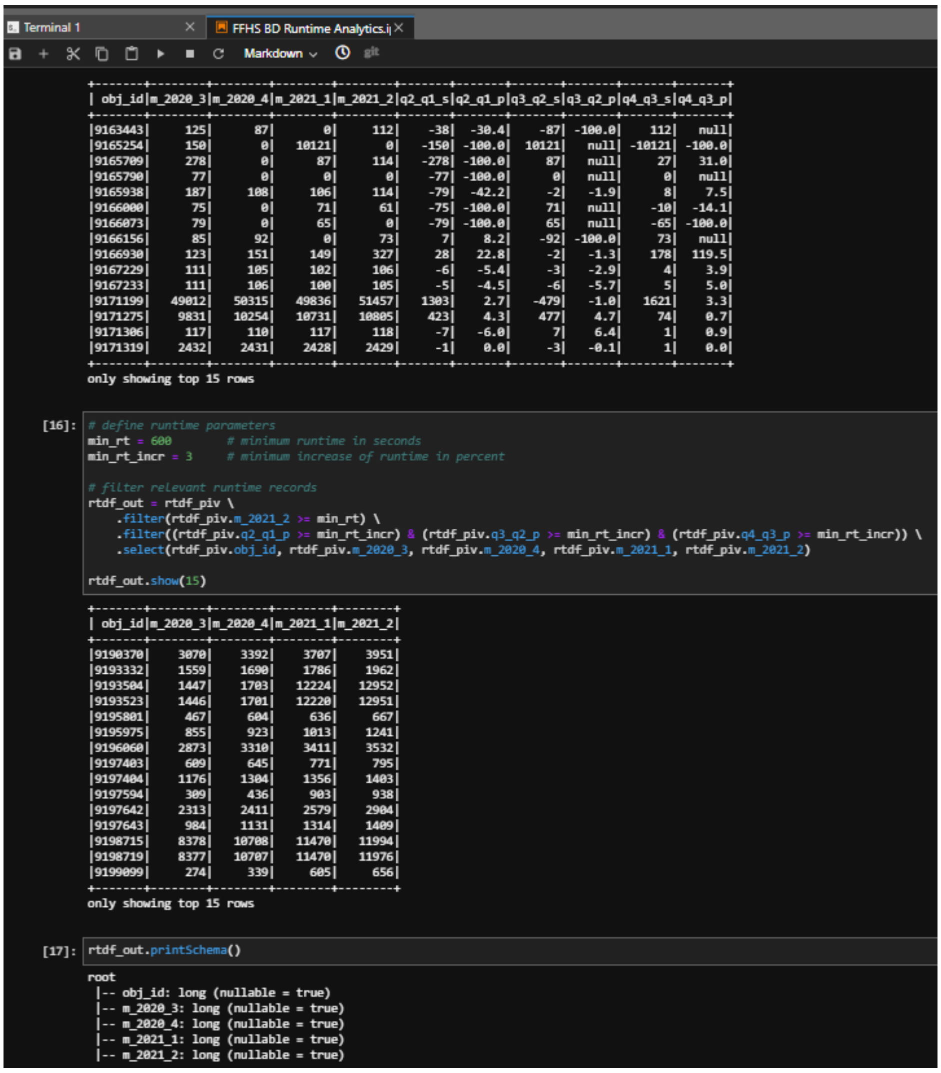
4: Slow Poison Analysis

```
[14]: from pyspark.sql.functions import *

      # prepared dataframes from above: objdf, rtdf, plandf

      # define static parameters
      min_rec = 3            # minimum number of records
      rt_status = 'ENDED_OK' # final status, ignore aborted runs
      quarters = 4           # number of time periods to check

      # TEMP for testing only -> also remove in filters below
      #L = [9193523,9197404,9197403]
      #    .filter(rtdf_cnt.obj_id.isin(L)) \
      # TEMP end

      # set timeframe
      # to-do: calculate dates based on current date and defined parameters
      date_start = '2020-07-01'
      date_end = '2021-06-30'

      # count no. of records per ID, rename "count"
      rtdf_cnt = rtdf.join(rtdf.groupBy('obj_id').count().withColumnRenamed("count","runs"),on='obj_id')

      print(rtdf_cnt.count())
      rtdf_cnt.show(5)

      # set date and customer filters, objects >= the minimum no. of records. necessary to compute median later
      rtdf_base = rtdf_cnt \
          .filter(rtdf_cnt.stats_act_date.between(date_start,date_end)) \
          .filter(rtdf_cnt.comp_id == 9) \
          .filter(rtdf_cnt.stats_status == rt_status) \
          .filter(rtdf_cnt.runs >= min_rec) \
          .drop('obj_env_id','comp_id','stats_id','stats_rid','stats_day','comp_id','stats_status','stats_arch_1','stats_arch_2','stats_start_date','stats_end_date','runs')

      print(rtdf_base.count())
      rtdf_base.show(5)
```

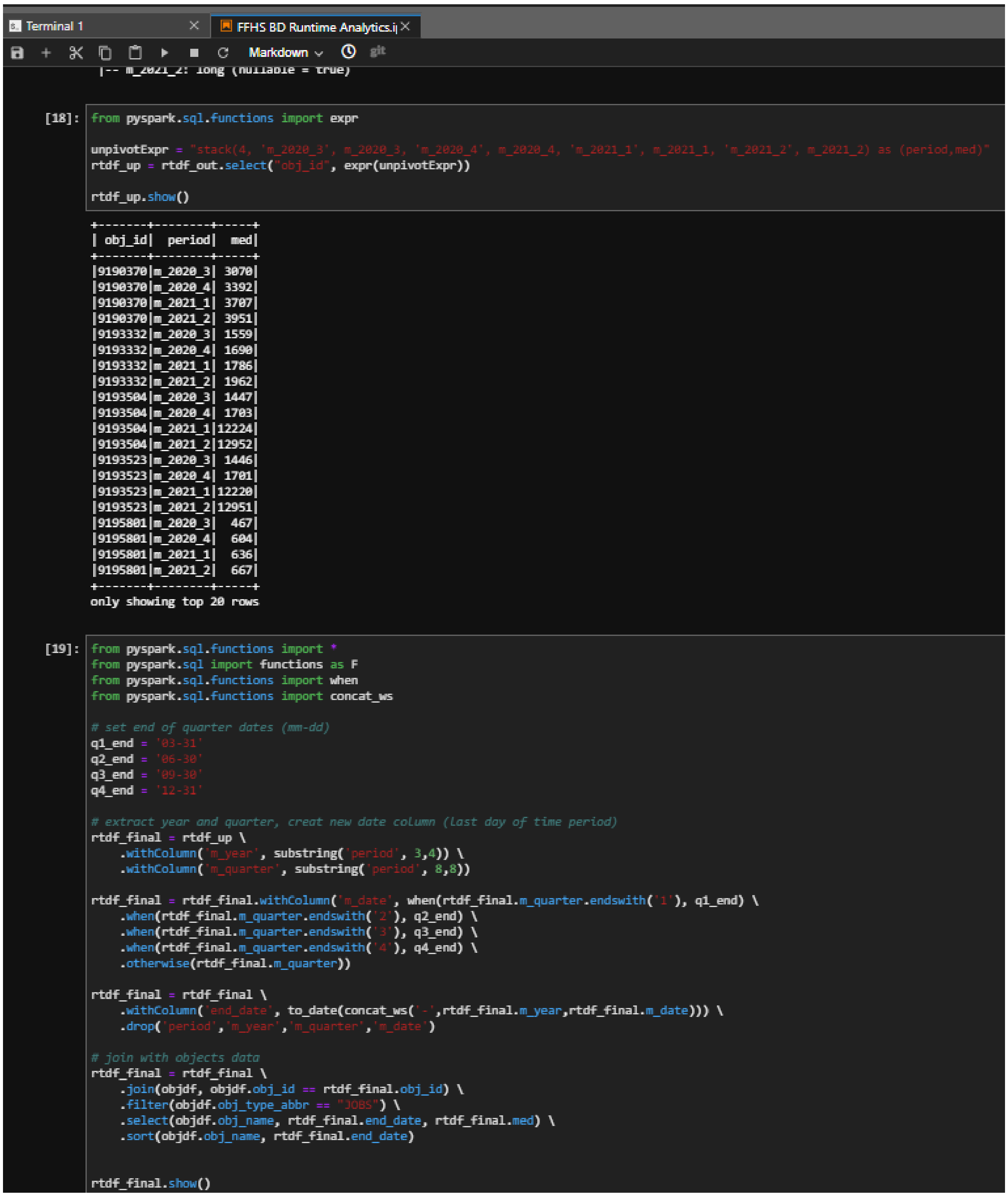
```
9345043
+-------+----------+-------+-----------+----------+---------+-----------+------------+------------+------------+--------------+----------------+--------------+----+
| obj_id|obj_env_id|comp_id|   stats_id| stats_rid|stats_day|stats_dur_s|stats_status|stats_arch_1|stats_arch_2|stats_act_date|stats_start_date|stats_end_date|runs|
+-------+----------+-------+-----------+----------+---------+-----------+------------+------------+------------+--------------+----------------+--------------+----+
|9170264|   2252670|      7|12426787162|1815059007|        4|        352|    ENDED_OK|         EOD| AVALOQ_JOB|    2021-06-02|      2021-06-03|    2021-06-03| 199|
|9170264|   2252670|      7|12440520887|1821030446|        4|        768|    ENDED_OK|         EOD| AVALOQ_JOB|    2021-06-09|      2021-06-10|    2021-06-10| 199|
|9170264|   2252670|      7|12451678675|1823078088|        6|        277|    ENDED_OK|         EOD| AVALOQ_JOB|    2021-06-11|      2021-06-12|    2021-06-12| 199|
|9170264|   2252670|      7|12455548453|1822057274|        5|        450|    ENDED_OK|         EOD| AVALOQ_JOB|    2021-06-10|      2021-06-11|    2021-06-11| 199|
|9170264|   2252670|      7|12286237270|1790992464|        4|         68|    ENDED_OK|         EOD| AVALOQ_JOB|    2021-05-05|      2021-05-06|    2021-05-06| 199|
+-------+----------+-------+-----------+----------+---------+-----------+------------+------------+------------+--------------+----------------+--------------+----+
only showing top 5 rows

303100
+-------+-----------+--------------+
| obj_id|stats_dur_s|stats_act_date|
+-------+-----------+--------------+
|9195811|         80|    2020-10-02|
|9195811|         78|    2020-10-05|
|9195811|         93|    2021-02-12|
|9195811|         66|    2020-07-28|
|9195811|         69|    2020-07-30|
+-------+-----------+--------------+
only showing top 5 rows
```



```
[15]: from pyspark.sql.types import IntegerType
      from pyspark.sql import functions as F
      from pyspark.sql import SparkSession
      from pyspark.sql.types import IntegerType
      import numpy as np

      #TEMP
      tdf_base_backup = rtdf_base
      #rtdf_base = rtdf_base_backup
      #TEMP end

      rtdf_base = rtdf_base.withColumn('quarters',concat(F.lit("m_"),F.year(rtdf_base.stats_act_date),F.lit("_"),F.quarter(rtdf_base.stats_act_date)).cast('string')).drop("stats_act_date")

      rtdf_base.show(15)

      # calculate median for defined time periods and create pivot table
      # orginal code -> does not work because percentile_approx requires Spark >= 3.1 (current version installed 3.0)
      # output = rtdf_base.groupBy('id').pivot("quarters").agg(
      #    percentile_approx("stats_dur_s", 0.5, F.lit(1000000)))

      # workaround: convert to Pandas
      df_pd = rtdf_base.toPandas()
      #print('df_pd')
      #display(df_pd)

      # create pivot table with median for each time period
      # convert to integer (for whatever reason, the last median value had data type LONG)
      df_pd = pd.pivot_table(df_pd, values='stats_dur_s', fill_value=0, index=['obj_id'], columns=['quarters'], aggfunc=np.median).astype(int)
      #print('df_pd_out')
      #display(df_pd)

      # convert back to Spark
      rtdf_piv = spark.createDataFrame(df_pd.reset_index(drop=False))
      #rtdf_piv.show()
      #rtdf_piv.printSchema()

      # calculate difference between time periods (seconds and percent)
      # to-do: dynamic columns -> use column id instead of fixed names
      # rtdf_piv = rtdf_piv.withColumn('q2_q1_s',rtdf_piv.select(rtdf_piv.columns[2]) - rtdf_piv.select(rtdf_piv.columns[1]))
      # #df.select(df.columns[2:4]).show(3)
      rtdf_piv = rtdf_piv \
          .withColumn('q2_q1_s',F.col('m_2020_4') - F.col('m_2020_3')) \
          .withColumn('q2_q1_p',F.round((F.col('m_2020_4') - F.col('m_2020_3'))/F.col('m_2020_3')*100,1)) \
          .withColumn('q3_q2_s',F.col('m_2021_1') - F.col('m_2020_4')) \
          .withColumn('q3_q2_p',F.round((F.col('m_2021_1') - F.col('m_2020_4'))/F.col('m_2020_4')*100,1)) \
          .withColumn('q4_q3_s',F.col('m_2021_2') - F.col('m_2021_1')) \
          .withColumn('q4_q3_p',F.round((F.col('m_2021_2') - F.col('m_2021_1'))/F.col('m_2021_1')*100,1))

      rtdf_piv.show(15)

      +-------+----------+--------+
      | obj_id|stats_dur_s|quarters|
      +-------+----------+--------+
      |9195811|        63|m_2021_2|
      |9195811|        78|m_2021_2|
      |9195811|        63|m_2021_2|
      |9195811|        66|m_2021_2|
      |9195811|        80|m_2021_2|
      |9195811|        69|m_2021_2|
      |9195811|        69|m_2021_2|
      |9195811|        62|m_2021_2|
      |9195811|        80|m_2021_2|
      |9195811|        86|m_2021_2|
      |9195811|        77|m_2020_4|
      |9195811|        63|m_2020_4|
      |9195811|        71|m_2020_4|
      |9195811|        68|m_2020_4|
      |9195811|        71|m_2020_4|
      +-------+----------+--------+
      only showing top 15 rows
```



```
      +-------+--------+--------+--------+--------+-------+-------+-------+-------+-------+-------+
      | obj_id|m_2020_3|m_2020_4|m_2021_1|m_2021_2|q2_q1_s|q2_q1_p|q3_q2_s|q3_q2_p|q4_q3_s|q4_q3_p|
      +-------+--------+--------+--------+--------+-------+-------+-------+-------+-------+-------+
      |9163443|     125|      87|       0|     112|    -38|  -30.4|    -87| -100.0|    112|   null|
      |9165254|     150|       0|   10121|       0|   -150| -100.0|  10121|   null| -10121| -100.0|
      |9165709|     278|       0|      87|     114|   -278| -100.0|     87|   null|     27|   31.0|
      |9165790|      77|       0|       0|       0|    -77| -100.0|      0|   null|      0|   null|
      |9165938|     187|     108|     106|     114|    -79|  -42.2|     -2|   -1.9|      8|    7.5|
      |9166000|      75|       0|      71|      61|    -75| -100.0|     71|   null|    -10|  -14.1|
      |9166073|      79|       0|      65|       0|    -79| -100.0|     65|   null|    -65| -100.0|
      |9166156|      85|      92|       0|      73|      7|    8.2|    -92| -100.0|     73|   null|
      |9166930|     123|     151|     149|     327|     28|   22.8|     -2|   -1.3|    178|  119.5|
      |9167229|     111|     105|     102|     106|     -6|   -5.4|     -3|   -2.9|      4|    3.9|
      |9167233|     111|     106|     100|     105|     -5|   -4.5|     -6|   -5.7|      5|    5.0|
      |9171199|   49012|   50315|   49836|   51457|   1303|    2.7|   -479|   -1.0|   1621|    3.3|
      |9171275|    9831|   10254|   10731|   10805|    423|    4.3|    477|    4.7|     74|    0.7|
      |9171306|     117|     110|     117|     118|     -7|   -6.0|      7|    6.4|      1|    0.9|
      |9171319|    2432|    2431|    2428|    2429|     -1|    0.0|     -3|   -0.1|      1|    0.0|
      +-------+--------+--------+--------+--------+-------+-------+-------+-------+-------+-------+
      only showing top 15 rows

[16]: # define runtime parameters
      min_rt = 600          # minimum runtime in seconds
      min_rt_incr = 3       # minimum increase of runtime in percent

      # filter relevant runtime records
      rtdf_out = rtdf_piv \
          .filter(rtdf_piv.m_2021_2 >= min_rt) \
          .filter((rtdf_piv.q2_q1_p >= min_rt_incr) & (rtdf_piv.q3_q2_p >= min_rt_incr) & (rtdf_piv.q4_q3_p >= min_rt_incr)) \
          .select(rtdf_piv.obj_id, rtdf_piv.m_2020_3, rtdf_piv.m_2020_4, rtdf_piv.m_2021_1, rtdf_piv.m_2021_2)

      rtdf_out.show(15)

      +-------+--------+--------+--------+--------+
      | obj_id|m_2020_3|m_2020_4|m_2021_1|m_2021_2|
      +-------+--------+--------+--------+--------+
      |9190370|    3070|    3392|    3707|    3951|
      |9193332|    1559|    1690|    1786|    1962|
      |9193504|    1447|    1703|   12224|   12952|
      |9193523|    1446|    1701|   12220|   12951|
      |9195801|     467|     604|     636|     667|
      |9195975|     855|     923|    1013|    1241|
      |9196060|    2873|    3310|    3411|    3532|
      |9197403|     609|     645|     771|     795|
      |9197404|    1176|    1304|    1356|    1403|
      |9197594|     309|     436|     903|     938|
      |9197642|    2313|    2411|    2579|    2904|
      |9197643|     984|    1131|    1314|    1409|
      |9198715|    8378|   10708|   11470|   11994|
      |9198719|    8377|   10707|   11470|   11976|
      |9199099|     274|     339|     605|     656|
      +-------+--------+--------+--------+--------+
      only showing top 15 rows

[17]: rtdf_out.printSchema()

      root
       |-- obj_id: long (nullable = true)
       |-- m_2020_3: long (nullable = true)
       |-- m_2020_4: long (nullable = true)
       |-- m_2021_1: long (nullable = true)
       |-- m_2021_2: long (nullable = true)
```

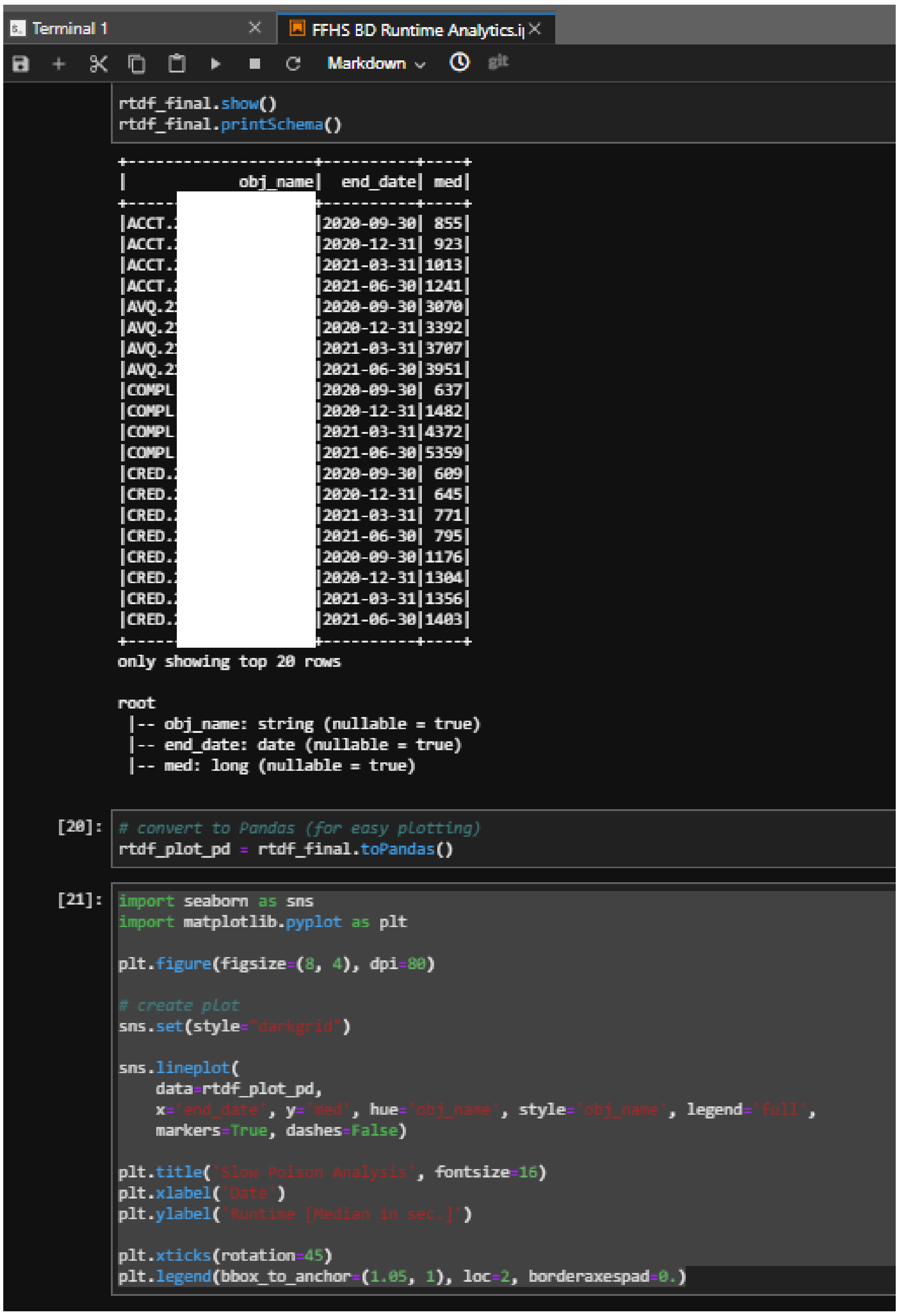


```
   |-- m_2021_2: long (nullable = true)
```

[18]:
```
from pyspark.sql.functions import expr

unpivotExpr = "stack(4, 'm_2020_3', m_2020_3, 'm_2020_4', m_2020_4, 'm_2021_1', m_2021_1, 'm_2021_2', m_2021_2) as (period,med)"
rtdf_up = rtdf_out.select("obj_id", expr(unpivotExpr))

rtdf_up.show()
```

```
+-------+--------+-----+
| obj_id|  period|  med|
+-------+--------+-----+
|9190370|m_2020_3| 3070|
|9190370|m_2020_4| 3392|
|9190370|m_2021_1| 3707|
|9190370|m_2021_2| 3951|
|9193332|m_2020_3| 1559|
|9193332|m_2020_4| 1690|
|9193332|m_2021_1| 1786|
|9193332|m_2021_2| 1962|
|9193504|m_2020_3| 1447|
|9193504|m_2020_4| 1703|
|9193504|m_2021_1|12224|
|9193504|m_2021_2|12952|
|9193523|m_2020_3| 1446|
|9193523|m_2020_4| 1701|
|9193523|m_2021_1|12220|
|9193523|m_2021_2|12951|
|9195801|m_2020_3|  467|
|9195801|m_2020_4|  604|
|9195801|m_2021_1|  636|
|9195801|m_2021_2|  667|
+-------+--------+-----+
only showing top 20 rows
```

[19]:
```
from pyspark.sql.functions import *
from pyspark.sql import functions as F
from pyspark.sql.functions import when
from pyspark.sql.functions import concat_ws

# set end of quarter dates (mm-dd)
q1_end = '03-31'
q2_end = '06-30'
q3_end = '09-30'
q4_end = '12-31'

# extract year and quarter, creat new date column (last day of time period)
rtdf_final = rtdf_up \
    .withColumn('m_year', substring('period', 3,4)) \
    .withColumn('m_quarter', substring('period', 8,8))

rtdf_final = rtdf_final.withColumn('m_date', when(rtdf_final.m_quarter.endswith('1'), q1_end) \
    .when(rtdf_final.m_quarter.endswith('2'), q2_end) \
    .when(rtdf_final.m_quarter.endswith('3'), q3_end) \
    .when(rtdf_final.m_quarter.endswith('4'), q4_end) \
    .otherwise(rtdf_final.m_quarter))

rtdf_final = rtdf_final \
    .withColumn('end_date', to_date(concat_ws('-',rtdf_final.m_year,rtdf_final.m_date))) \
    .drop('period','m_year','m_quarter','m_date')

# join with objects data
rtdf_final = rtdf_final \
    .join(objdf, objdf.obj_id == rtdf_final.obj_id) \
    .filter(objdf.obj_type_abbr == "JOBS") \
    .select(objdf.obj_name, rtdf_final.end_date, rtdf_final.med) \
    .sort(objdf.obj_name, rtdf_final.end_date)


rtdf_final.show()
```

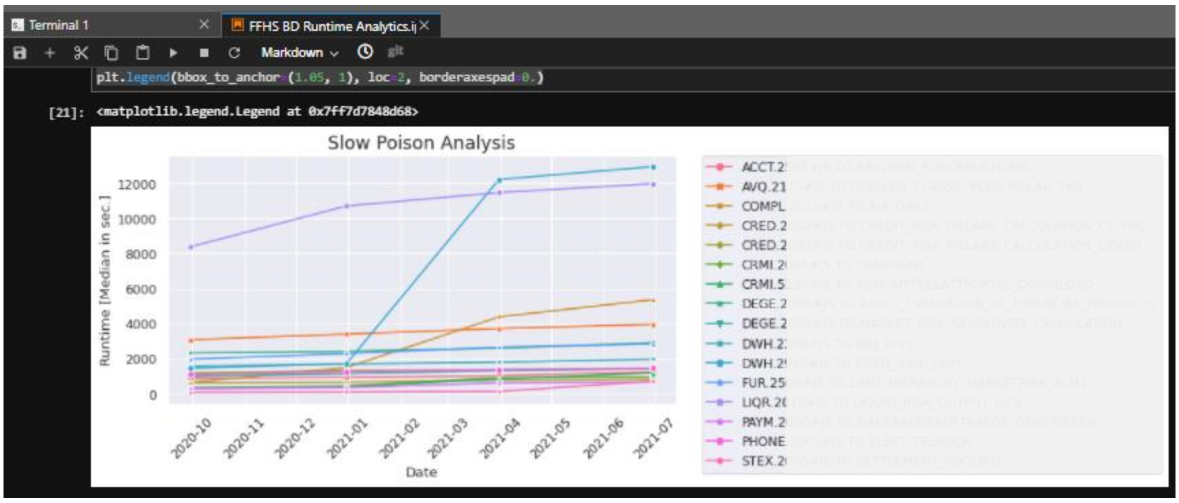


```
rtdf_final.show()
rtdf_final.printSchema()
```

```
+--------------------+----------+----+
|            obj_name|  end_date| med|
+--------------------+----------+----+
|ACCT.[redacted]|2020-09-30| 855|
|ACCT.[redacted]|2020-12-31| 923|
|ACCT.[redacted]|2021-03-31|1013|
|ACCT.[redacted]|2021-06-30|1241|
|AVQ.2[redacted]|2020-09-30|3070|
|AVQ.2[redacted]|2020-12-31|3392|
|AVQ.2[redacted]|2021-03-31|3707|
|AVQ.2[redacted]|2021-06-30|3951|
|COMPL[redacted]|2020-09-30| 637|
|COMPL[redacted]|2020-12-31|1482|
|COMPL[redacted]|2021-03-31|4372|
|COMPL[redacted]|2021-06-30|5359|
|CRED.[redacted]|2020-09-30| 609|
|CRED.[redacted]|2020-12-31| 645|
|CRED.[redacted]|2021-03-31| 771|
|CRED.[redacted]|2021-06-30| 795|
|CRED.[redacted]|2020-09-30|1176|
|CRED.[redacted]|2020-12-31|1304|
|CRED.[redacted]|2021-03-31|1356|
|CRED.[redacted]|2021-06-30|1403|
+-----[redacted]+----------+----+
only showing top 20 rows

root
 |-- obj_name: string (nullable = true)
 |-- end_date: date (nullable = true)
 |-- med: long (nullable = true)
```

```
[20]:  # convert to Pandas (for easy plotting)
       rtdf_plot_pd = rtdf_final.toPandas()
```

```
[21]:  import seaborn as sns
       import matplotlib.pyplot as plt

       plt.figure(figsize=(8, 4), dpi=80)

       # create plot
       sns.set(style="darkgrid")

       sns.lineplot(
           data=rtdf_plot_pd,
           x='end_date', y='med', hue='obj_name', style='obj_name', legend='full',
           markers=True, dashes=False)

       plt.title('Slow Poison Analysis', fontsize=16)
       plt.xlabel('Date')
       plt.ylabel('Runtime [Median in sec.]')

       plt.xticks(rotation=45)
       plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```



```
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

```
[21]: <matplotlib.legend.Legend at 0x7ff7d7848d68>
```