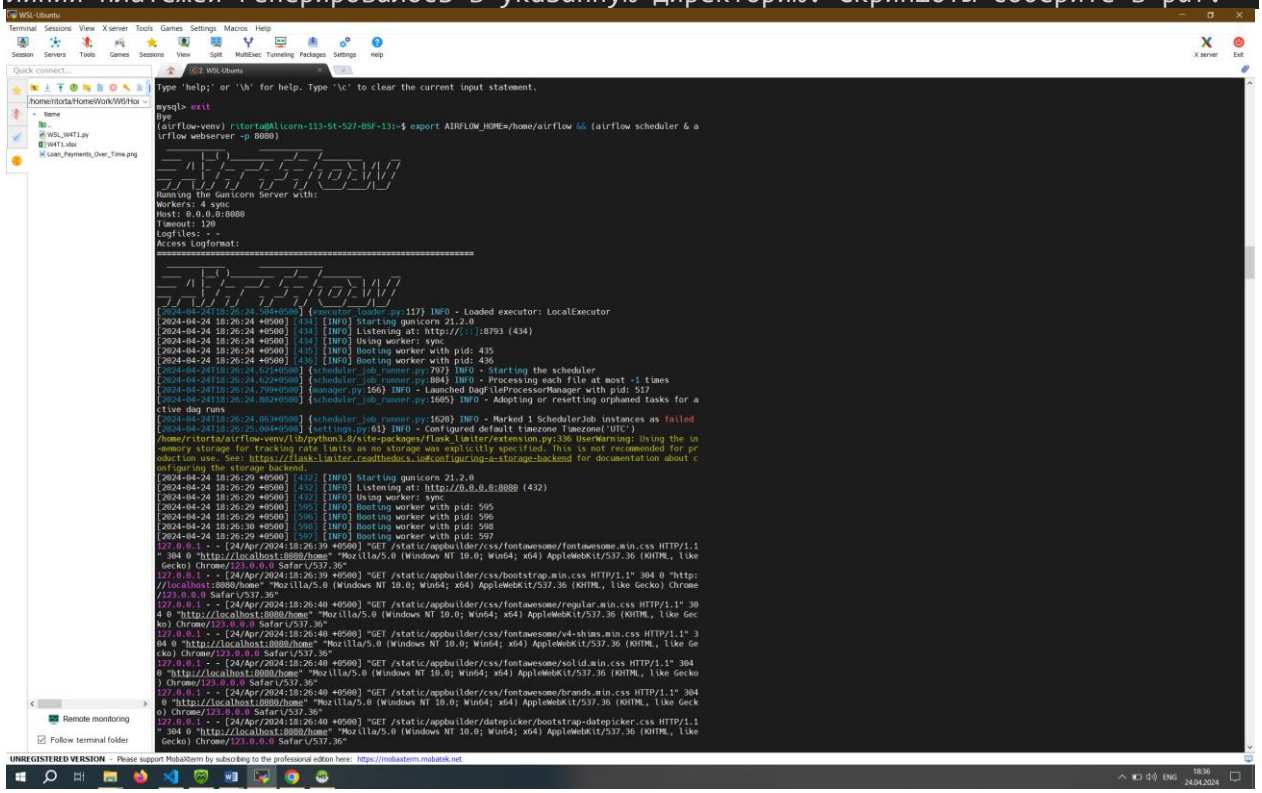
Урок 6. Операторы в Airflow и их применение для ETL

1. Установить спарк как показано на семинаре:
    - Для этого переместите папку spark в home.
    - Дайте права командой chmod -R 777 ./
    - nano ~/.bashrc
    - export SPARK_HOME=/home/spark && export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
    - source ~/.bashrc
    - sudo apt-get install openjdk-8-jdkpip
    - Указанные библиотеки нужно также установить и в виртуальную среду: python3 -m venv airflow venv && source airflow venv /bin/activate
    - pip install pyspark==3.2.4
    - pip install pandas==1.5.3
    - pip install SQLAlchemy==1.4.46

    Используйте ДЗ которые вы мне высылали для 3-4 семинара. Запустите данные задачи ПОСЛЕДОВАТЕЛЬНО, одну за другой в аирфлоу. Пришлите мне скриншоты выполненных задач в аирфлоу, логов аирфлоу, скриншоты что у вас записались таблицы в БД mysql на WSL. По возможности доработайте код чтобы изображение с линии платежей генерировалось в указанную директорию. Скриншоты соберите в pdf.



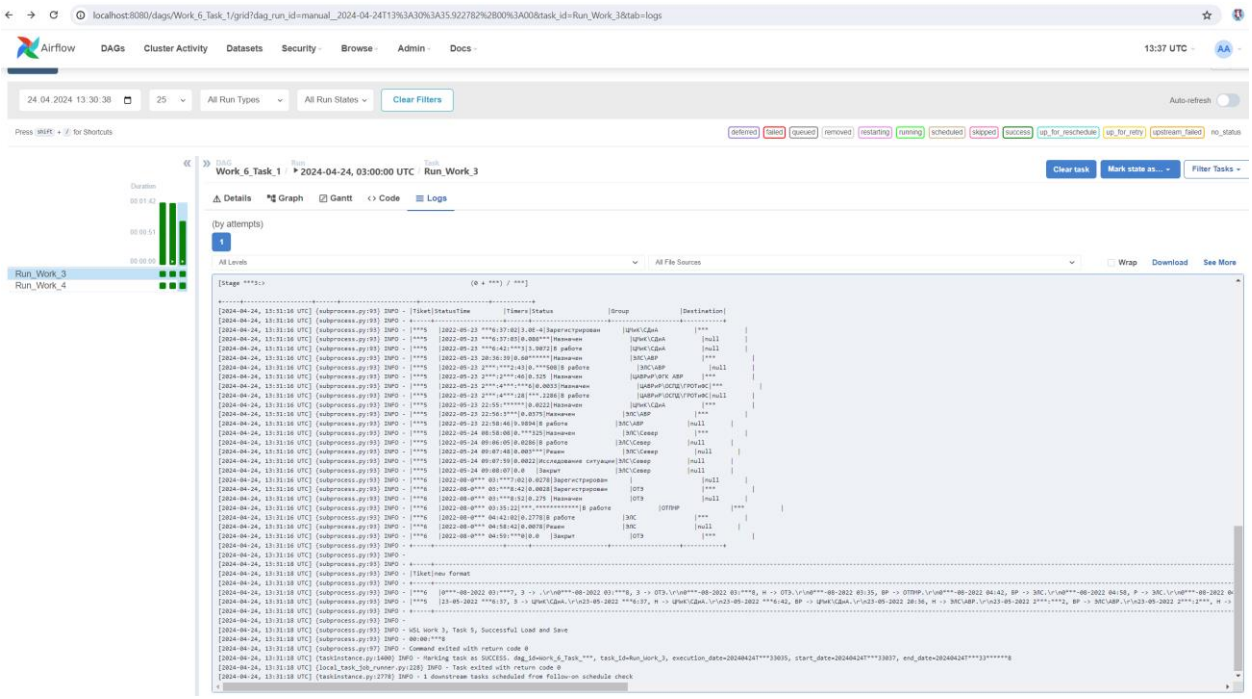

```
# Файл Task1 и файлы вместе с папками Home_3 и Home_4 надо скинуть в WSL для того, чтобы они работали.

from airflow import DAG
from airflow.operators.bash import BashOperator
from airflow.operators.python import PythonOperator, BranchPythonOperator
from datetime import datetime, timedelta
import pendulum
```

```
default_args = {
'owner': 'Ritorta',
'depends_on_past': False,
'start_date': pendulum.datetime(year=2024, month=4,
day=23).in_timezone('Europe/Moscow'),
'email': ['meddesu@yandex.ru'],
'email_on_failure': False,
'email_on_retry': False,
'retries': 0,
'retry_delay': timedelta(minutes=5)
}

dag1 = DAG('Work_6_Task_1',
default_args=default_args,
description="Home_Work_6",
catchup=False,
schedule_interval='0 6 * * *')

WSL_Home_3 = BashOperator(
task_id='Run_Work_3',
bash_command='export SPARK_HOME=/home/spark && export
PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin && spark-shell -i
/home/ritorta/HomeWork/W6/Home_3/WSL_W3Task_1_v2.scala',
dag=dag1)

WSL_Home_4  = BashOperator(
task_id='Run_Work_4',
bash_command='export SPARK_HOME=/home/spark && export
PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin && python3
/home/ritorta/HomeWork/W6/Home_4/WSL_W4T1.py',
dag=dag1)

WSL_Home_3 >> WSL_Home_4

/*
chcp 65001 && spark-shell -i /home/ritorta/HomeWork/W6/WSL_W6T1dag.py --conf
"spark.driver.extraJavaOptions=-Dfile.encoding=utf-8"
*/

import org.apache.spark.internal.Logging
import org.apache.spark.sql.functions.{col, collect_list, concat_ws}
import org.apache.spark.sql.{DataFrame, SparkSession}
import org.apache.spark.sql.expressions.Window
import scala.io.Source

sc.setLogLevel("ERROR")

var sqlCoun = s"jdbc:mysql://localhost:33061/spark?user=root&password=1"
var driver = "com.mysql.cj.jdbc.Driver"

val t1 = System.currentTimeMillis()
```

```
if(1==1){
    var df = spark.read.option("delimiter",",")
        .option("inferSchema", "true")
        .option("header", "true")
        .format("excel")
        .load("/home/ritorta/HomeWork/W6/Home_3/s3.xlsx")

    val df1 = df
        df1.write.format("jdbc").option("url", sqlCoun)
            .option("driver", driver).option("dbtable", "wsl_w3t5v2")
            .mode("overwrite").save()
    df1.show(400, truncate = false)

    val df2 = spark.read.format("jdbc").option("url", sqlCoun)
        .option("driver", driver)
        .option("dbtable", "wsl_w3t5v2")
        .load()

    val df_group = df2.distinct().where(col("fieldname") === "GNAME2")
        .select("objectid", "restime", "fieldvalue")
        .withColumnRenamed("fieldvalue", "Group")
        .withColumn("Destination", lit("1").cast("integer"))

    val df_status = df2.distinct().where(col("fieldname") === "Status")
        .select("objectid", "restime", "fieldvalue")
        .withColumnRenamed("fieldvalue", "Status")

    val df_sg = df2.filter((col("fieldname") isin ("status", "GNAME2")))
        .select("objectid", "restime").distinct()

    val df_inner = df_sg.as("a")
        .join(df_status.as("a1"),col("a.objectid") === col("a1.objectid") &&
col("a.restime") === col("a1.restime"),"left")
        .join(df_group.as("a2"),col("a.objectid") === col("a2.objectid") &&
col("a.restime") === col("a2.restime"),"left")
        .select(col("a.objectid"),col("a.restime"),col("a1.Status"),col("a2.Group
"),col("a2.Destination"))
        .withColumnRenamed("objectid", "Tiket")
        .withColumnRenamed("restime", "StatusTime")
        .distinct()

    val df_outer =
df_inner.select(col("Tiket"),col("StatusTime"),col("Status"),when(row_number().ov
er(Window.partitionBy(col("Tiket"))
        .orderBy(col("StatusTime"))) === 1 &&
col("Destination").isNull,"").otherwise(col("Group")).alias("Group"),col("Destina
tion"))
```

```
    val df_result =
df_outer.select(col("Tiket"),from_unixtime(col("StatusTime")).alias("StatusTime")
,((lead(col("StatusTime"), 1)
        .over(Window.partitionBy(col("Tiket")).orderBy(col("StatusTime"))) -
col("StatusTime")) / 3600).alias("Timers"),last(col("Status"), true)
        .over(Window.partitionBy(col("Tiket")).orderBy(col("StatusTime")))
        .alias("Status"),last(col("Group"),
true).over(Window.partitionBy(col("Tiket")).orderBy(col("StatusTime")))
        .alias("Group"),col("Destination"))
        .withColumn("Timers", coalesce(col("Timers"), lit(0)))
        .withColumn("Timers", round(col("Timers"), 4))

    df_result.write.format("jdbc").option("url", sqlCoun)
        .option("driver", driver).option("dbtable", "wsl_w3t5v2a")
        .mode("overwrite").save()
    df_result.show(400, truncate = false)

    val df3 = spark.read.format("jdbc").option("url", sqlCoun)
        .option("driver", driver)
        .option("dbtable", "wsl_w3t5v2a")
        .load()

val df3_concat = df3.groupBy("Tiket")
    .agg(concat_ws(".\r\n", collect_list(concat_ws(",
",when(date_format(col("StatusTime"), "yyyy-MM-dd") ===
current_date(),date_format(col("StatusTime"), "yyyy-MM-dd HH:mm:ss"))
    .otherwise(date_format(col("StatusTime"), "dd-MM-yyyy HH:mm")),
    concat_ws(" -> ",when(col("Status") === "Зарегистрирован", "З")
    .when(col("Status") === "Назначен", "Н")
    .when(col("Status") === "В работе", "ВР")
    .when(col("Status") === "Закрыт", "ЗТ")
    .when(col("Status") === "Исследование ситуации", "ИС")
    .when(col("Status") === "Решен", "Р"),col("Group")))))
    .alias("new format"))
    .withColumn("new format", concat(col("new format"),lit(".")))
    .withColumn("Tiket",col("Tiket"))

    df3_concat.write.format("jdbc").option("url", sqlCoun)
        .option("driver", driver).option("dbtable", "wsl_w3t5v2b")
        .mode("overwrite").save()
    df3_concat.show(400, truncate = false)

    println("WSL Work 3, Task 5, Successful Load and Save")

}

val s0 = (System.currentTimeMillis() - t1)/1000
val s = s0 % 60
val m = (s0/60) % 60
val h = (s0/60/60) % 24
println("%02d:%02d:%02d".format(h, m, s))
```

```
System.exit(0)
```

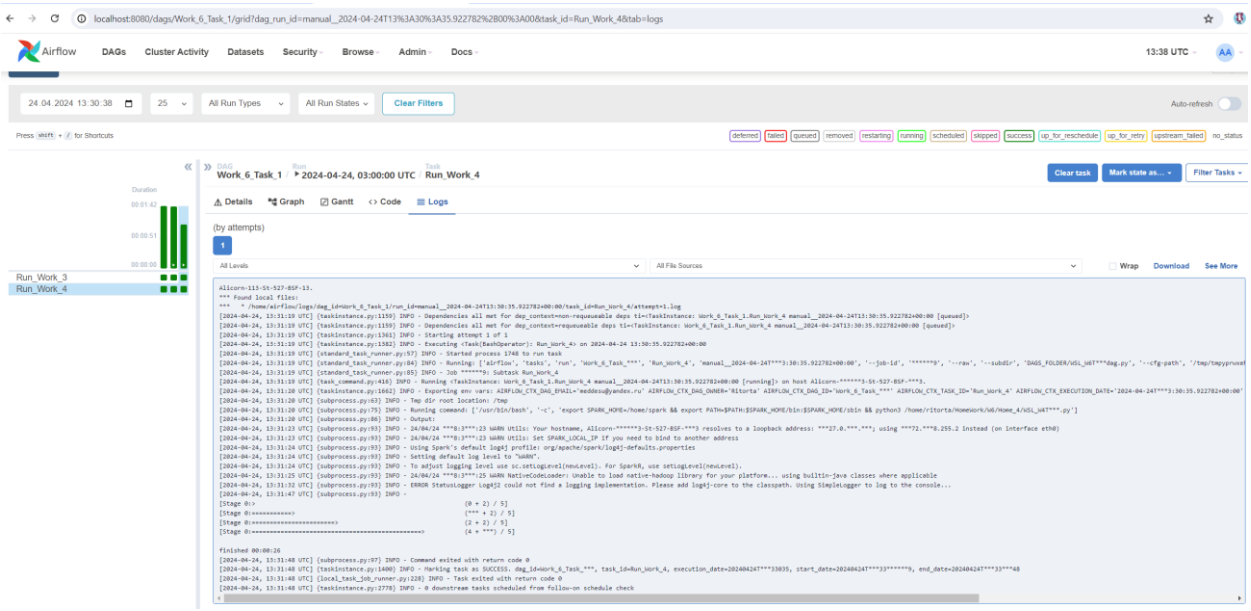


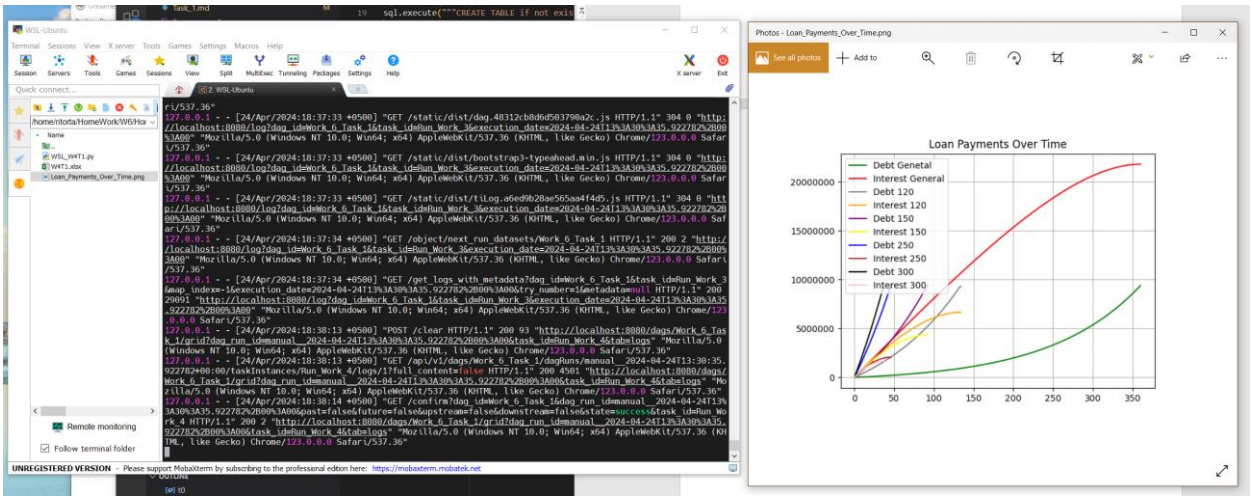

```
import pyspark,time,platform,sys,os
from datetime import datetime
from pyspark.sql.session import SparkSession
from pyspark.sql.functions import col,lit,current_timestamp
import pandas as pd
import matplotlib.pyplot as plt
from sqlalchemy import inspect,create_engine
from pandas.io import sql
import warnings,matplotlib

warnings.filterwarnings("ignore")
t0=time.time()
con=create_engine("mysql://root:1@localhost:33061/spark")
os.environ['PYSPARK_PYTHON'] = sys.executable
os.environ['PYSPARK_DRIVER_PYTHON'] = sys.executable
spark=SparkSession.builder.appName("WSL Home Work №4").getOrCreate()

sql.execute("""drop table if exists spark.`WSL_W4T1`""",con)
sql.execute("""CREATE TABLE if not exists spark.`WSL_W4T1` (
        `number` INT(10) NULL DEFAULT NULL,
        `Month` DATE NULL DEFAULT NULL,
        `Payment amount` FLOAT NULL DEFAULT NULL,
        `Payment of the principal debt` FLOAT NULL DEFAULT NULL,
        `Payment of interest` FLOAT NULL DEFAULT NULL,
        `Balance of debt` FLOAT NULL DEFAULT NULL,
        `interest` FLOAT NULL DEFAULT NULL,
        `debt` FLOAT NULL DEFAULT NULL
```

```
)
COLLATE='utf8mb4_general_ci'
ENGINE=InnoDB""",con)

from pyspark.sql.window import Window
from pyspark.sql.functions import sum as sum1
w =
Window.partitionBy(lit(1)).orderBy("number").rowsBetween(Window.unboundedPrecedin
g, Window.currentRow)
dfG = spark.read.format("com.crealytics.spark.excel")\
        .option("dataAddress", "'General'!A1:F361")\
        .option("useHeader", "false")\
        .option("treatEmptyValuesAsNulls", "false")\
        .option("inferSchema", "true").option("addColorColumns", "true")\
        .option("usePlainNumberFormat","true")\
        .option("startColumn", 0)\
        .option("endColumn", 99)\
        .option("timestampFormat", "MM-dd-yyyy HH:mm:ss")\
        .option("maxRowsInMemory", 20)\
        .option("excerptSize", 10)\
        .option("header", "true")\
        .format("excel")\
        .load("/home/ritorta/HomeWork/W6/Home_4/W4T1.xlsx").limit(1000)\
        .withColumn("interest", sum1(col("Payment of interest")).over(w))\
        .withColumn("debt", sum1(col("Payment of the principal debt")).over(w))

df120 = spark.read.format("com.crealytics.spark.excel")\
        .option("dataAddress", "'120'!A1:F135")\
        .option("useHeader", "false")\
        .option("treatEmptyValuesAsNulls", "false")\
        .option("inferSchema", "true").option("addColorColumns", "true")\
        .option("usePlainNumberFormat","true")\
        .option("startColumn", 0)\
        .option("endColumn", 99)\
        .option("timestampFormat", "MM-dd-yyyy HH:mm:ss")\
        .option("maxRowsInMemory", 20)\
        .option("excerptSize", 10)\
        .option("header", "true")\
        .format("excel")\
        .load("/home/ritorta/HomeWork/W6/Home_4/W4T1.xlsx").limit(1000)\
        .withColumn("interest", sum1(col("Payment of interest")).over(w))\
        .withColumn("debt", sum1(col("Payment of the principal debt")).over(w))

df150 = spark.read.format("com.crealytics.spark.excel")\
        .option("dataAddress", "'150'!A1:F93")\
        .option("useHeader", "false")\
        .option("treatEmptyValuesAsNulls", "false")\
        .option("inferSchema", "true").option("addColorColumns", "true")\
        .option("usePlainNumberFormat","true")\
        .option("startColumn", 0)\
        .option("endColumn", 99)\
```

```
        .option("timestampFormat", "MM-dd-yyyy HH:mm:ss")\
        .option("maxRowsInMemory", 20)\
        .option("excerptSize", 10)\
        .option("header", "true")\
        .format("excel")\
        .load("/home/ritorta/HomeWork/W6/Home_4/W4T1.xlsx").limit(1000)\
        .withColumn("interest", sum1(col("Payment of interest")).over(w))\
        .withColumn("debt", sum1(col("Payment of the principal debt")).over(w))

df250 = spark.read.format("com.crealytics.spark.excel")\
        .option("dataAddress", "'250'!A1:F47")\
        .option("useHeader", "false")\
        .option("treatEmptyValuesAsNulls", "false")\
        .option("inferSchema", "true").option("addColorColumns", "true")\
        .option("usePlainNumberFormat","true")\
        .option("startColumn", 0)\
        .option("endColumn", 99)\
        .option("timestampFormat", "MM-dd-yyyy HH:mm:ss")\
        .option("maxRowsInMemory", 20)\
        .option("excerptSize", 10)\
        .option("header", "true")\
        .format("excel")\
        .load("/home/ritorta/HomeWork/W6/Home_4/W4T1.xlsx").limit(1000)\
        .withColumn("interest", sum1(col("Payment of interest")).over(w))\
        .withColumn("debt", sum1(col("Payment of the principal debt")).over(w))

df300 = spark.read.format("com.crealytics.spark.excel")\
        .option("dataAddress", "'300'!A1:F38")\
        .option("useHeader", "false")\
        .option("treatEmptyValuesAsNulls", "false")\
        .option("inferSchema", "true").option("addColorColumns", "true")\
        .option("usePlainNumberFormat","true")\
        .option("startColumn", 0)\
        .option("endColumn", 99)\
        .option("timestampFormat", "MM-dd-yyyy HH:mm:ss")\
        .option("maxRowsInMemory", 20)\
        .option("excerptSize", 10)\
        .option("header", "true")\
        .format("excel")\
        .load("/home/ritorta/HomeWork/W6/Home_4/W4T1.xlsx").limit(1000)\
        .withColumn("interest", sum1(col("Payment of interest")).over(w))\
        .withColumn("debt", sum1(col("Payment of the principal debt")).over(w))

df_combined = dfG.union(df120).union(df150).union(df250).union(df300)

df_combined.write.format("jdbc").option("url","jdbc:mysql://localhost:33061/spark
?user=root&password=1")\
        .option("driver", "com.mysql.cj.jdbc.Driver").option("dbtable",
"WSL_W4T1")\
        .mode("append").save()
```

```
"""df_pandas = df_combined.toPandas()"""

df_pandas1 = dfG.toPandas()
df_pandas2 = df120.toPandas()
df_pandas3 = df150.toPandas()
df_pandas4 = df250.toPandas()
df_pandas5 = df300.toPandas()

ax = plt.gca()
ax.ticklabel_format(style='plain')

df_pandas1.plot(kind='line', x='number', y='debt', color='green', ax=ax,
label='Debt Genetal')
df_pandas1.plot(kind='line', x='number', y='interest', color='red', ax=ax,
label='Interest General')
df_pandas2.plot(kind='line', x='number', y='debt', color='grey', ax=ax,
label='Debt 120')
df_pandas2.plot(kind='line', x='number', y='interest', color='orange', ax=ax,
label='Interest 120')
df_pandas3.plot(kind='line', x='number', y='debt', color='purple', ax=ax,
label='Debt 150')
df_pandas3.plot(kind='line', x='number', y='interest', color='yellow', ax=ax,
label='Interest 150')
df_pandas4.plot(kind='line', x='number', y='debt', color='blue', ax=ax,
label='Debt 250')
df_pandas4.plot(kind='line', x='number', y='interest', color='brown', ax=ax,
label='Interest 250')
df_pandas5.plot(kind='line', x='number', y='debt', color='black', ax=ax,
label='Debt 300')
df_pandas5.plot(kind='line', x='number', y='interest', color='pink', ax=ax,
label='Interest 300')

plt.title('Loan Payments Over Time')
plt.grid ( True )
ax.set(xlabel=None)

plot_directory = "/home/ritorta/HomeWork/W6/Home_4/"
plot_filename = "Loan_Payments_Over_Time.png"
plt.savefig(plot_directory + plot_filename)

plt.show()
spark.stop()
t1=time.time()
print('finished',time.strftime('%H:%M:%S',time.gmtime(round(t1-t0))))
```


Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
13:38 UTC
Work_6_Task_1
2024-04-24, 03:00:00 UTC
Run_Work_4
Clear task
Mark state as...
Filter Tasks
Details
Graph
Gantt
Code
Logs
(by attempts)
Run_Work_3
Run_Work_4


WSL-Ubuntu
Photos - Loan_Payments_Over_Time.png
Loan Payments Over Time
Debt Genetal
Interest General
Debt 120
Interest 120
Debt 150
Interest 150
Debt 250
Interest 250
Debt 300
Interest 300
Remote monitoring
Follow terminal folder
UNREGISTERED VERSION