WSL-Ubuntu

Terminal Sessions View X server Tools Games Settings Macros Help

Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit

Quick connect...

2. WSL-Ubuntu

/home/ekaterina/

Name

..
.cache
.config
.local
airflow-venv
.bash_history
.bash_logout
.bashrc
.motd_shown
.profile
.scala_history
.sudo_as_admin_successful
.wget-hsts
.Xauthority
All graphics.png
fifa_s_2.csv
get-pip.py
HW_2_fifa_s.scala
HW_4_task.py
s1_new.scala
s1_new.sh
s4_2_task_HW.xlsx
Sem1.xlsx
sem_6.py

Remote monitoring

Follow terminal folder

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/04/21 08:40:42 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
ERROR StatusLogger Log4j2 could not find a logging implementation. Please add log4j-core to the classpath. Using SimpleLogger to log to the console...
finished 00:00:15
(airflow-venv) ekaterina@MSI:~$ ^C
(airflow-venv) ekaterina@MSI:~$ python3 HW_4_task.py
24/04/21 08:44:37 WARN Utils: Your hostname, MSI resolves to a loopback address: 127.0.1.1; using 172.18.96.52 instead (on interface eth0)
24/04/21 08:44:37 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/04/21 08:44:39 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
ERROR StatusLogger Log4j2 could not find a logging implementation. Please add log4j-core to the classpath. Using SimpleLogger to log to the console...
(airflow-venv) ekaterina@MSI:~$ python3 HW_4_task.py
24/04/21 08:46:02 WARN Utils: Your hostname, MSI resolves to a loopback address: 127.0.1.1; using 172.18.96.52 instead (on interface eth0)
24/04/21 08:46:02 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/04/21 08:46:04 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
ERROR StatusLogger Log4j2 could not find a logging implementation. Please add log4j-core to the classpath. Using SimpleLogger to log to the console...
finished 00:00:17
(airflow-venv) ekaterina@MSI:~$ python3 HW_4_task.py
24/04/21 08:48:34 WARN Utils: Your hostname, MSI resolves to a loopback address: 127.0.1.1; using 172.18.96.52 instead (on interface eth0)
24/04/21 08:48:34 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/04/21 08:48:36 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
ERROR StatusLogger Log4j2 could not find a logging implementation. Please add log4j-core to the classpath. Using SimpleLogger to log to the console...
finished 00:00:15
(airflow-venv) ekaterina@MSI:~$ python3 HW_4_task.py
24/04/21 08:50:34 WARN Utils: Your hostname, MSI resolves to a loopback address: 127.0.1.1; using 172.18.96.52 instead (on interface eth0)
24/04/21 08:50:34 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/04/21 08:50:36 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
ERROR StatusLogger Log4j2 could not find a logging implementation. Please add log4j-core to the classpath. Using SimpleLogger to log to the console...
finished 00:00:15
(airflow-venv) ekaterina@MSI:~$
```

Unnamed-1\spark\HW_4_all\ - HeidiSQL 12.6.0.6765

Файл Редактировать Поиск Запрос Инструменты Переход Помощь

Donate

Фильтр баз данны Фильтр таблиц

Хост: 127.0.0.1 База данных: spark Таблица: HW_4_all Данные Запрос*

- Unnamed-1
  - Airflow
  - information_schema
  - mysql
  - performance_schema
  - **spark** 480,0 KiB
    - HW_2_task_3 384,0 KiB
    - **HW_4_all** 64,0 KiB
    - tasketl1a 16,0 KiB
    - tasketl1b 16,0 KiB
  - sys

spark.HW_4_all: 586 строк (exact)

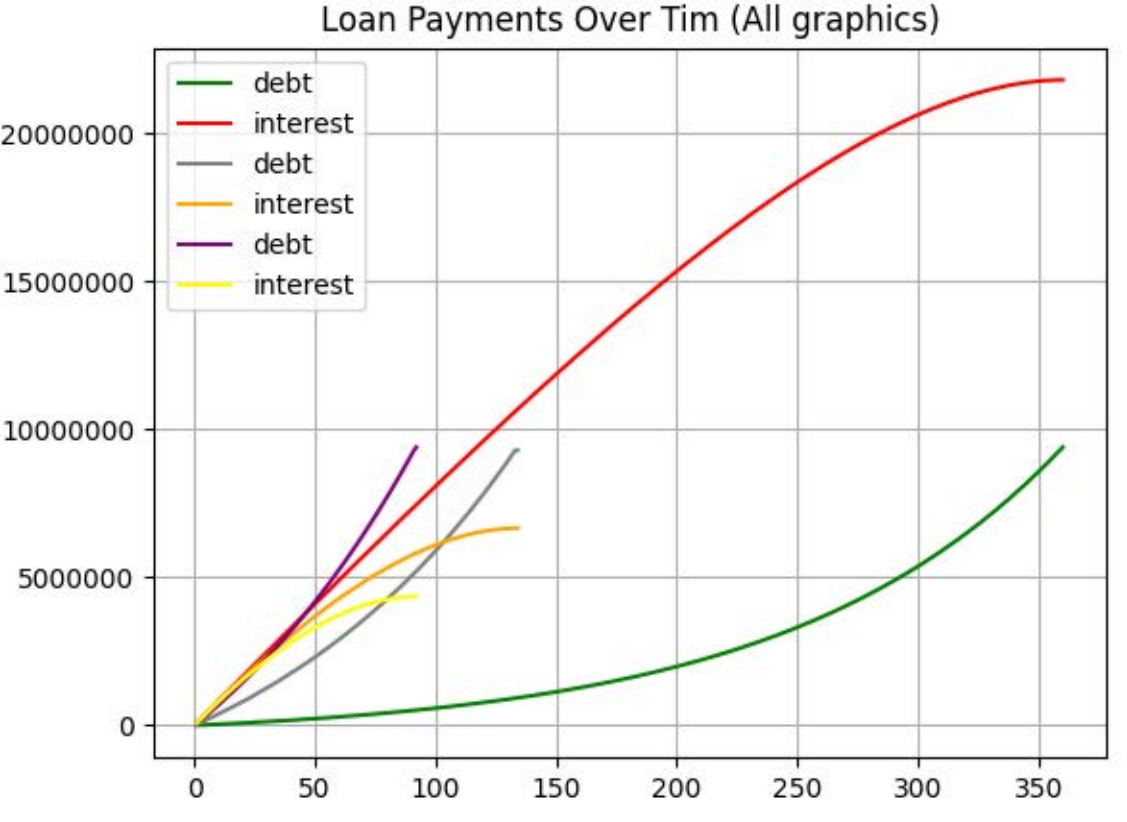
Далее Показать все Сортировка Столбцы (8/8) Фильтр

#	№	Month	Payment amount	Payment of the principal debt	Payment of interest	Balance of debt	interest	debt
1	1	2024-05-13	150 000	68 327,9	81 672,1	9 331 670	81 672,1	68 327,9
2	1	2024-05-13	120 000	38 327,9	81 672,1	9 361 670	81 672,1	38 327,9
3	2	2024-06-13	150 000	66 218,9	83 781,1	9 265 450	165 453	134 547
4	2	2024-06-13	120 000	35 949,6	84 050,4	9 325 720	165 723	74 277,5
5	3	2024-07-13	150 000	69 496,9	80 503,1	9 195 960	245 956	204 044
6	3	2024-07-13	120 000	38 973,2	81 026,8	9 286 750	246 749	113 251
7	4	2024-08-13	150 000	67 437,4	82 562,6	9 128 520	328 519	271 481
8	4	2024-08-13	120 000	36 622,2	83 377,8	9 250 130	330 127	149 873
9	5	2024-09-13	150 000	68 042,9	81 957,1	9 060 480	410 476	339 524
10	5	2024-09-13	120 000	36 951	83 049	9 213 180	413 176	186 824
11	6	2024-10-13	150 000	71 277,8	78 722,2	8 989 200	489 198	410 802
12	6	2024-10-13	120 000	39 951,1	80 048,9	9 173 220	493 225	226 775
13	7	2024-11-13	150 000	69 293,7	80 706,3	8 919 900	569 905	480 095
14	7	2024-11-13	120 000	37 641,5	82 358,5	9 135 580	575 583	264 417
15	8	2024-12-13	120 000	40 625,3	79 374,7	9 094 960	654 958	305 042
16	8	2024-12-13	150 000	72 499,2	77 500,8	8 847 400	647 405	552 595
17	9	2025-01-13	120 000	38 250,4	81 749,6	9 056 710	736 708	343 292
18	9	2025-01-13	150 000	70 475,5	79 524,5	8 776 930	726 930	623 070
19	10	2025-02-13	120 000	38 464,8	81 535,2	9 018 240	818 243	381 757
20	10	2025-02-13	150 000	70 983,6	79 016,4	8 705 950	805 946	694 054
21	1	2023-11-01	86 689	3 655,71	83 033,3	9 396 340	83 033,3	3 655,71
22	11	2025-03-13	120 000	46 668,1	73 331,9	8 971 580	891 575	428 425
23	2	2023-12-01	86 689	3 688	83 001	9 392 660	166 034	7 343,71
24	11	2025-03-13	150 000	79 207,5	70 792,5	8 626 740	876 739	773 261
25	12	2025-04-13	150 000	72 335,7	77 664,3	8 554 400	954 403	845 597
26	3	2024-01-01	86 689	3 720,58	82 968,5	9 388 940	249 003	11 064,3
27	12	2025-04-13	120 000	39 231,2	80 768,8	8 932 340	972 344	467 656
28	13	2025-05-13	150 000	75 471,2	74 528,8	8 478 930	1 028 930	921 068
29	13	2025-05-13	120 000	42 178,5	77 821,5	8 890 160	1 050 170	509 835

Фильт Регулярное выражение

```
253 SELECT *, EVENT_SCHEMA AS `Db`, EVENT_NAME AS `Name` FROM information_schema.`EVENTS` WHERE `EVENT_SCHEMA`='spark';
254 SELECT * FROM `information_schema`.`COLUMNS` WHERE TABLE_SCHEMA='spark' AND TABLE_NAME='HW_4_all' ORDER BY ORDINAL_POSITION;
255 SHOW INDEXES FROM `HW_4_all` FROM `spark`;
256 SELECT * FROM information_schema.REFERENTIAL_CONSTRAINTS WHERE   CONSTRAINT_SCHEMA='spark'   AND TABLE_NAME='HW_4_all'   AND REFERENCED_TABLE_NAME IS NOT NULL;
257 SELECT * FROM information_schema.KEY_COLUMN_USAGE WHERE   TABLE_SCHEMA='spark'   AND TABLE_NAME='HW_4_all'   AND REFERENCED_TABLE_NAME IS NOT NULL;
258 SHOW CREATE TABLE `spark`.`HW_4_all`;
259 SELECT tc.CONSTRAINT_NAME, cc.CHECK_CLAUSE FROM `information_schema`.`CHECK_CONSTRAINTS` AS cc, `information_schema`.`TABLE_CONSTRAINTS` AS tc WHERE tc.CONSTRAINT_SCHEMA='spark' AND tc.TABLE_NAME='HW_4_all' AND tc.CONSTRAINT_TYPE='CHECK' AND tc.CONSTRAINT_SCHEMA=
260 SELECT * FROM `spark`.`HW_4_all` LIMIT 1000;
```

r1 : c2 Подключено: 00:15 h MariaDB or MySQL 8.0.36 Время работы: 00:49 h Серверное время: Ожидание.


Loan Payments Over Tim (All graphics)
debt
interest
debt
interest
debt
interest
20000000
15000000
10000000
5000000
0
0
50
100
150
200
250
300
350

```
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://Airflow:1@localhost/spark")

os.environ['PYSPARK_PYTHON'] = sys.executable
os.environ['PYSPARK_DRIVER_PYTHON'] = sys.executable
spark=SparkSession.builder.appName("Hi").getOrCreate()

sql.execute("""drop table if exists spark.`HW_4_all`""",con)
sql.execute("""CREATE TABLE if not exists spark.`HW_4_all` (
        `№` 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("№").rowsBetween(Window.unboundedPreceding,
Window.currentRow)
df1 = spark.read.format("com.crealytics.spark.excel")\
        .option("dataAddress", "'sheet_sem'!A1")\
        .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/ekaterina/s4_2_task_HW.xlsx").limit(1000)\
        .withColumn("interest", sum1(col("Payment of interest")).over(w))\
        .withColumn("debt", sum1(col("Payment of the principal debt")).over(w))

df2 = spark.read.format("com.crealytics.spark.excel")\
        .option("dataAddress", "'sheet_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/ekaterina/s4_2_task_HW.xlsx").limit(1000)\
        .withColumn("interest", sum1(col("Payment of interest")).over(w))\
        .withColumn("debt", sum1(col("Payment of the principal debt")).over(w))


df3 = spark.read.format("com.crealytics.spark.excel")\
        .option("dataAddress", "'sheet_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/ekaterina/s4_2_task_HW.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 = df1.union(df2).union(df3)

df_combined.write.format("jdbc").option("url","jdbc:mysql://localhost:33061/spa
rk?user=root&password=1")\
        .option("driver", "com.mysql.cj.jdbc.Driver").option("dbtable",
"HW_4_all")\
        .mode("append").save()

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

df_pandas1 = df1.toPandas()
df_pandas2 = df2.toPandas()
df_pandas3 = df3.toPandas()

# Get current axis
ax = plt.gca()
ax.ticklabel_format(style='plain')

# bar plot
df_pandas1.plot(kind='line', x='№', y='debt', color='green', ax=ax)
df_pandas1.plot(kind='line', x='№', y='interest', color='red', ax=ax)
```

```
df_pandas2.plot(kind='line', x='№', y='debt', color='grey', ax=ax) #
ежемесячный платеж 120 000
df_pandas2.plot(kind='line', x='№', y='interest', color='orange', ax=ax)  #
ежемесячный платеж 120 000
df_pandas3.plot(kind='line', x='№', y='debt', color='purple', ax=ax)  #
ежемесячный платеж 150 000
df_pandas3.plot(kind='line', x='№', y='interest', color='yellow', ax=ax)  #
ежемесячный платеж 150 000

# set the title
plt.title('Loan Payments Over Tim (All graphics)')
plt.grid ( True )
ax.set(xlabel=None)


# save
plt.savefig("/home/ekaterina/All graphics.png")
spark.stop()
t1=time.time()
print('finished',time.strftime('%H:%M:%S',time.gmtime(round(t1-t0))))
```