

ФІНАЛЬНИЙ ПРОЕКТ ДЕ.

1. КОД для частини 1: (середовище виконання – Google Colab)

```
# Встановлення PySpark
!pip install pyspark findspark

# Завантаження mysql-connector
!wget -q https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-j-8.0.32.tar.gz
!tar -xzf mysql-connector-j-8.0.32.tar.gz
!cp mysql-connector-j-8.0.32/mysql-connector-j-8.0.32.jar /content/
# Spark сесія

# Встановлення Kafka-конектора для Spark
!wget https://repol.maven.org/maven2/org/apache/spark/spark-sql-kafka-0-10_2.12/3.5.1/spark-sql-kafka-0-10_2.12-3.5.1.jar -P /content/
!wget https://repol.maven.org/maven2/org/apache/kafka/kafka-clients/3.7.0/kafka-clients-3.7.0.jar -P /content/
!wget https://repol.maven.org/maven2/org/apache/commons/commons-pool2/2.12.0/commons-pool2-2.12.0.jar -P /content/

import findspark
findspark.init()

from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *
import random, time
spark = SparkSession.builder \
    .appName("KafkaSparkStreaming") \
    .config("spark.jars", "/content/spark-sql-kafka-0-10_2.12-3.5.1.jar,"
            "/content/kafka-clients-3.7.0.jar,"
            "/content/commons-pool2-2.12.0.jar,"
            "/content/mysql-connector-j-8.0.32.jar") \
    .getOrCreate()

# Під'єднання до MySQL та завантаження даних
jdbc_url = "jdbc:mysql://217.61.57.46:3306/olympic_dataset"
jdbc_table = "athlete bio"
jdbc_user = "neo_data_admin"
jdbc_password = "Proyahaxuqithab9opl"

athlete_bio_df = spark.read.format('jdbc').options(
    url=jdbc_url,
    driver='com.mysql.cj.jdbc.Driver',
    dbtable=jdbc_table,
    user=jdbc_user,
    password=jdbc_password
).load()

athlete_bio_df = athlete_bio_df \
    .filter(athlete_bio_df['height'].cast("float").isNotNull() &
athlete_bio_df['weight'].cast("float").isNotNull()) \
    .withColumn('height', col('height').cast('float')) \
    .withColumn('weight', col('weight').cast('float'))

athlete_bio_df.show(5)
# Створення стрімінг-джерела даних з Kafka

from pyspark.sql.functions import from_json, col, current_timestamp
from pyspark.sql.types import StructType, StructField, IntegerType, StringType

# Вказуємо схему для JSON-повідомлень Kafka
schema = StructType([
    StructField("athlete_id", IntegerType(), True),
    StructField("medal", StringType(), True),
    StructField("sport", StringType(), True),
    StructField("sex", StringType(), True),
    StructField("country_noc", StringType(), True),
])

# Створення Streaming DataFrame з Kafka
kafka_stream_df = spark \
    .readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "athlete_event_results") \
    .option("startingOffsets", "earliest") \
    .load()
```

```

# Декодування значень JSON з Kafka
streaming_df = kafka_stream_df \
    .selectExpr("CAST(value AS STRING)") \
    .select(from_json(col("value"), schema).alias("data")) \
    .select("data.*") \
    .withColumn("timestamp", current_timestamp())

# Запуск стрімінгу (імітуючи дані, що надходять в режимі реального часу)
streaming_query = streaming_df.writeStream \
    .format("memory") \
    .queryName("competition_stream") \
    .outputMode("append") \
    .start()

# Перетворення статичних даних в Streaming DataFrame

streaming_df = kafka_stream_df \
    .withColumn("timestamp", current_timestamp())

# Імітація стрімінгу (дані надходять раз в 5 секунд)
streaming_query = streaming_df.writeStream \
    .format("memory") \
    .queryName("competition_stream") \
    .outputMode("append") \
    .start()

# Функція для обробки кожного мікробатчу (foreachBatch в завданні)

from pyspark.sql.functions import col, avg, current_timestamp

def process_batch(batch_df, batch_id):
    joined_df = batch_df.join(athlete_bio_df, 'athlete_id', 'inner')

    agg_df = joined_df.groupBy('sport', 'medal', 'sex', 'country_noc') \
        .agg(avg('height').alias('avg height'), avg('weight').alias('avg weight')) \
        .withColumn('calculated_at', current_timestamp())

    agg_df.show(truncate=False)

# Використовуємо існуючий streaming_df напряму
stream_query = streaming_df.writeStream \
    .foreachBatch(process_batch) \
    .outputMode("update") \
    .start()

# Зчитування результатів з бази даних назад у DataFrame PySpark

jdbc_url = "jdbc:mysql://217.61.57.46:3306/olympic_dataset"
jdbc_table = "athlete_enriched_agg"
jdbc_user = "neo_data_admin"
jdbc_password = "Proyahaxuqithab9oplp"

result_df = spark.read.format('jdbc').options(
    url=jdbc_url,
    driver='com.mysql.cj.jdbc.Driver',
    dbtable=jdbc_table,
    user=jdbc_user,
    password=jdbc_password).load()

result_df.show()
# або перетворення на Pandas DataFrame (більш зручно)
pdf = result_df.toPandas()
pdf.head(20)

# Зупинка стрімінгу після виконання роботи

streaming_query.stop()
stream_query.stop()
spark.stop()

```

РЕЗУЛЬТАТИ ВИКОНАННЯ для частини 1:

result_df.show()

	sport	medal	sex	country_noc	avg_height	avg_weight	timestamp
	Swimming	nan	Male	ANZ	170.0	65.0	2024-12-18 01:39:48
	Rowing	nan	Male	ANZ	189.0	85.0	2024-12-18 01:39:48
	Athletics	Bronze	Male	ANZ	184.0	76.0	2024-12-18 01:39:48
	Swimming	Silver	Male	ANZ	170.0	65.0	2024-12-18 01:39:48
	Swimming	Bronze	Male	ANZ	170.0	65.0	2024-12-18 01:39:48
	Rugby	Gold	Male	ANZ	175.0	75.0	2024-12-18 01:39:48
	Athletics	nan	Male	ANZ	170.35294117647058	65.05882352941177	2024-12-18 01:39:48
	Tennis	Bronze	Male	ANZ	188.0	84.0	2024-12-18 01:39:48
	Boxing	nan	Male	ANZ	167.0	62.0	2024-12-18 01:39:48
	Athletics	nan	Male	BAN	169.7826086956522	63.391304347826086	2024-12-18 01:40:10
	Archery	nan	Male	BHU	168.64705882352942	63.76470588235294	2024-12-18 01:40:10
	Swimming	nan	Male	ANZ	170.0	65.0	2024-12-18 01:40:10
	Sailing	nan	Male	ARU	184.0	88.0	2024-12-18 01:40:10
	Wrestling	nan	Male	ASA	188.0	127.0	2024-12-18 01:40:10
	CyclingRoad	nan	Male	ARU	172.66666666666666	73.66666666666667	2024-12-18 01:40:10
	Athletics	nan	Male	ASA	188.0	74.0	2024-12-18 01:40:10
	Weightlifting	nan	Male	ARU	173.33333333333334	5179.333333333333	2024-12-18 01:40:10
	Swimming	Silver	Male	ANZ	170.0	65.0	2024-12-18 01:40:10
	Shooting	nan	Male	ARU	172.0	65.0	2024-12-18 01:40:10
	Golf	nan	Male	BAN	165.0	84.0	2024-12-18 01:40:10

only showing top 20 rows

та сама таблиця в іншому вигляді:

[9] pdt = result_df.toPandas()
pdf.head(20)

	sport	medal	sex	country_noc	avg_height	avg_weight	timestamp
0	Swimming	nan	Male	ANZ	170.000000	65.000000	2024-12-18 01:39:48
1	Rowing	nan	Male	ANZ	189.000000	85.000000	2024-12-18 01:39:48
2	Athletics	Bronze	Male	ANZ	184.000000	76.000000	2024-12-18 01:39:48
3	Swimming	Silver	Male	ANZ	170.000000	65.000000	2024-12-18 01:39:48
4	Swimming	Bronze	Male	ANZ	170.000000	65.000000	2024-12-18 01:39:48
5	Rugby	Gold	Male	ANZ	175.000000	75.000000	2024-12-18 01:39:48
6	Athletics	nan	Male	ANZ	170.352941	65.058824	2024-12-18 01:39:48
7	Tennis	Bronze	Male	ANZ	188.000000	84.000000	2024-12-18 01:39:48
8	Boxing	nan	Male	ANZ	167.000000	62.000000	2024-12-18 01:39:48
9	Athletics	nan	Male	BAN	169.782609	63.391304	2024-12-18 01:40:10
10	Archery	nan	Male	BHU	168.647059	63.764706	2024-12-18 01:40:10
11	Swimming	nan	Male	ANZ	170.000000	65.000000	2024-12-18 01:40:10
12	Sailing	nan	Male	ARU	184.000000	88.000000	2024-12-18 01:40:10
13	Wrestling	nan	Male	ASA	188.000000	127.000000	2024-12-18 01:40:10
14	CyclingRoad	nan	Male	ARU	172.666667	73.666667	2024-12-18 01:40:10
15	Athletics	nan	Male	ASA	188.000000	74.000000	2024-12-18 01:40:10
16	Weightlifting	nan	Male	ARU	173.333333	5179.333333	2024-12-18 01:40:10
17	Swimming	Silver	Male	ANZ	170.000000	65.000000	2024-12-18 01:40:10
18	Shooting	nan	Male	ARU	172.000000	65.000000	2024-12-18 01:40:10
19	Golf	nan	Male	BAN	165.000000	84.000000	2024-12-18 01:40:10

2. КОД для частини 2: (середовище виконання – Google Colab)

```
# Встановлення PySpark
!pip install pyspark requests

# Імпорт необхідних бібліотек і створення функції завантаження даних із FTP-сервера

import requests

# Функція для завантаження даних з FTP сервера
def download_data(local_file_name):
    url = "https://ftp.goit.study/neoversity/"
    full_url = url + local_file_name + ".csv"
    response = requests.get(full_url)

    if response.status_code == 200:
        with open(local_file_name + ".csv", 'wb') as file:
            file.write(response.content)
        print(f"📄 File '{local_file_name}.csv' downloaded successfully.")
    else:
        raise Exception(f"❌ Error {response.status_code}: File could not be downloaded.")

# Завантаження CSV файлів (athlete_bio.csv та athlete_event_results.csv)

# Завантажуємо athlete_bio.csv
download_data("athlete_bio")
# Завантажуємо athlete event results.csv
download_data("athlete_event_results")
```

ЕТАП BRONZE

```
# Запуск SparkSession та завантаження CSV, збереження у форматі Parquet (bronze шар)

from pyspark.sql import SparkSession
# Створюємо SparkSession
spark = SparkSession.builder \
    .appName("LandingToBronze") \
    .getOrCreate()

# Шлях до завантажених CSV файлів
athlete_bio_csv = "athlete_bio.csv"
athlete_event_csv = "athlete_event_results.csv"

# Зчитуємо athlete_bio.csv та зберігаємо у Parquet
athlete_bio_df = spark.read.option("header", True).option("inferSchema", True).csv(athlete_bio_csv)
athlete_bio_df.write.mode("overwrite").parquet("bronze/athlete_bio")

# Виведення результату
print("📄 athlete_bio таблиця успішно збережена в bronze шарі")
athlete_bio_df.show(5)

# Зчитуємо athlete_event_results.csv та зберігаємо у Parquet
athlete_event_df = spark.read.option("header", True).option("inferSchema", True).csv(athlete_event_csv)
athlete_event_df.write.mode("overwrite").parquet("bronze/athlete_event_results")

# Виведення результату
print("📄 athlete_event_results таблиця успішно збережена в bronze шарі")
athlete_event_df.show(5)

# Зупинка SparkSession
spark.stop()
```

ЕТАП BRONZE TO SILVER

```
# Запуск Spark-сесії і завантаження bronze-шару

from pyspark.sql import SparkSession
spark = SparkSession.builder \
    .appName("BronzeToSilver") \
    .getOrCreate()

# Завантаження даних з bronze
athlete_bio_df = spark.read.parquet("bronze/athlete_bio")
athlete_event_df = spark.read.parquet("bronze/athlete_event_results")

# Створюємо функцію очищення тексту та реєструємо як UDF (user defined function)

import re
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType

# Функція очищення тексту
```

```

def clean_text(text):
    return re.sub(r'^a-zA-Z0-9,.\\"\'', '', str(text))

# Реєструємо UDF
clean_text_udf = udf(clean_text, StringType())

# Застосовуємо очищення тексту до всіх текстових колонок

from pyspark.sql.types import StringType

# Функція для очищення текстових колонок
def clean_dataframe_text_columns(df):
    for column_name, column_type in df.dtypes:
        if column_type == 'string':
            df = df.withColumn(column_name, clean_text_udf(df[column_name]))
    return df

# Очищення текстових колонок в обох DataFrame
athlete_bio_cleaned_df = clean_dataframe_text_columns(athlete_bio_df)
athlete_event_cleaned_df = clean_dataframe_text_columns(athlete_event_df)
# Видаляємо дублікати

athlete_bio_cleaned_df = athlete_bio_cleaned_df.dropDuplicates()
athlete_event_cleaned_df = athlete_event_cleaned_df.dropDuplicates()

# Запис очищених даних у silver шар (формат parquet)

# Збереження у silver шар
athlete_bio_cleaned_df.write.mode("overwrite").parquet("silver/athlete_bio")
athlete_event_cleaned_df.write.mode("overwrite").parquet("silver/athlete_event_results")

# Виведення результатів
print("□ athlete_bio таблиця успішно збережена в silver шарі")
athlete_bio_cleaned_df.show(5, truncate=False)

print("□ athlete_event_results таблиця успішно збережена в silver шарі")
athlete_event_cleaned_df.show(5, truncate=False)

# Зупиняємо Spark сесію
spark.stop()

```

ЕТАП SILVER TO GOLD

```

from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, current_timestamp, col
from pyspark.sql.types import FloatType

# Створення сесії Spark
spark = SparkSession.builder.appName("SilverToGoldJob").getOrCreate()

# Зчитування silver даних
bio_df = spark.read.parquet("silver/athlete_bio")
results_df = spark.read.parquet("silver/athlete_event_results")

# Перетворення height і weight у числовий тип
bio_df = bio_df.withColumn("height", col("height").cast(FloatType()))
bio_df = bio_df.withColumn("weight", col("weight").cast(FloatType()))

# Видаляємо country_noc з одного з датафреймів, щоб уникнути конфлікту
bio_df = bio_df.drop("country_noc") # або results_df = results_df.drop("country_noc")

# Join по athlete_id
joined_df = results_df.join(bio_df, on="athlete_id", how="inner")

# Групування та обчислення середніх значень
agg_df = joined_df.groupBy("sport", "medal", "sex", "country_noc") \
    .agg(
        avg("height").alias("avg_height"),
        avg("weight").alias("avg_weight")
    ) \
    .withColumn("timestamp", current_timestamp())

# Вивід результату в лог
agg_df.show(truncate=False)

# Запис до gold шару
agg_df.write.mode("overwrite").parquet("gold/avg_stats")

```

ЕТАП СТВОРЕННЯ DAG-фалу

```
# Встановлення Apache Airflow та супутніх пакетів для належної роботи в Google Colab
!pip install apache-airflow[mysql]==2.8.4 --constraint
"https://raw.githubusercontent.com/apache/airflow/constraints-2.8.4/constraints-3.10.txt"
!pip uninstall -y pluggy
!pip install pluggy==1.3.0

!pip uninstall -y pytest
!pip install pytest==7.4.4
!pip show pluggy pytest

# Перевірка встановлення Airflow
import airflow
print(airflow. version )
# Клонування репозиторію GitHub
!git clone https://github.com/goitacademy/airflow_sandbox.git
%cd airflow sandbox
# налаштування URL з токеном goit
!git remote set-url origin
https://github.com/pat11AFNXSNA0A7kpaEdqhyIuED065C8vIQ40ow94CbXmq7mfFUI1YcsLW1K4yAfAegNMx4V5ZDDiIo6rWpY@github.com/goitacademy/airflow_sandbox.git

# створюємо папку dags/
import os
os.makedirs("airflow_sandbox/dags", exist_ok=True)

# Проектний DAG-файл з основним кодом

%%writefile chuboo_fin_p_dag.py
# %%writefile airflow_sandbox/dags/chuboo_fin_p_dag.py
from airflow import DAG
from airflow.providers.apache.spark.operators.spark_submit import SparkSubmitOperator
from datetime import datetime

# DAG definition
default_args = {
    'owner': 'airflow',
    'start_date': datetime(2024, 1, 1),
    'retries': 1
}

dag = DAG(
    dag_id='chuboo_fin_p',
    default_args=default_args,
    description='DAG to run Spark ETL pipeline for Data Lake',
    schedule_interval=None, # On-demand
    catchup=False
)

# Task 1: Landing to Bronze
landing_to_bronze = SparkSubmitOperator(
    task_id='landing_to_bronze',
    application='dags/landing_to_bronze.py',
    conn_id='spark-default',
    verbose=True,
    dag=dag
)

# Task 2: Bronze to Silver
bronze_to_silver = SparkSubmitOperator(
    task_id='bronze_to_silver',
    application='dags/bronze_to_silver.py',
    conn_id='spark-default',
    verbose=True,
    dag=dag
)

# Task 3: Silver to Gold
silver_to_gold = SparkSubmitOperator(
    task_id='silver_to_gold',
    application='dags/silver_to_gold.py',
    conn_id='spark-default',
    verbose=True,
    dag=dag
)

# Define DAG dependencies
landing_to_bronze >> bronze_to_silver >> silver_to_gold
```

```

# перевірка чи створено файл
!ls chuboo_fin_p_dag.py
# додавання усіх змін
!git add .
# створення коміту
!git config --global user.email "chubaroll@gmail.com"
!git config --global user.name "Chubar_OO"
!git commit -m "Додав DAG chuboo_fin_p_dag.py"

# Відправка зміни на GitHub
!git push origin main
# шифрування
!chmod +x encrypt_and_archive.sh
!./encrypt_and_archive.sh chuboo_fin_p_dag.py public_key.pem
# Переміщення зашифрованого архіву до папки encrypted file
!mv chuboo_fin_p_dag.py_encrypted_parts.tar.gz encrypted_file/
!ls encrypted_file
# додавання усіх змін
!git add .
# створення коміту
!git config --global user.email "chubaroll@gmail.com"
!git config --global user.name "Chubar_OO"
!git commit -m "Шифрування файлів та додавання до репозиторію."

# Відправка зміни на GitHub
!git push origin main
# Клонування репозиторію
!git clone https://github.com/goitacademy/airflow_sandbox.git


# перехід у вищу директорію
%cd /content

# Переміщення DAG-файл у папку dags всередині клону репозиторію
!mv airflow_sandbox/chuboo_fin_p_dag.py airflow_sandbox/dags/
# перевірка
!ls /content/airflow_sandbox/dags
# перехід до репозиторію
%cd /content/airflow_sandbox
# фіксація змін та push команда
!git add .
!git commit -m "Додавання DAG до папки dags"
!git pull origin main
!git push origin main

```

РЕЗУЛЬТАТИ ВИКОНАННЯ та ПОЯСНЕННЯ для частини 2:

Таблиці в бронзовому шапі:

 ☒ athlete_bio таблиця успішно збережена в bronze шапі

athlete_id	name	sex	born	height	weight	country	country_noc	description	special_notes
65649	Ivanka Bonova	Female	4 April 1949	166	55	Bulgaria	BUL	Personal Best: 40...	NULL
112510	Nataliya Uryadova	Female	15 March 1977	184	70	Russian Federation	RUS	NULL	Listed in Olympia...
114973	Essa Ismail Rashed	Male	14 December 1986	165	55	Qatar	QAT	Personal Best: 10...	Listed in Olympia...
30359	Péter Boros	Male	12 January 1908	NULL	NULL	Hungary	HUN	Between 1927 and ...	NULL
50557	Rudolf Piowatý	Male	28 April 1900	NULL	NULL	Czechoslovakia	TCH	Rudolf Piowaty jo...	NULL

only showing top 5 rows

☒ athlete_event_results таблиця успішно збережена в bronze шапі

edition	edition_id	country_noc	sport	event	result_id	athlete	athlete_id	pos	medal	isTeamSport
1908 Summer Olympics	5	ANZ	Athletics	100 metres, Men	56265	Ernest Hutcheon	64710	DNS	NULL	false
1908 Summer Olympics	5	ANZ	Athletics	400 metres, Men	56313	Henry Murray	64756	DNS	NULL	false
1908 Summer Olympics	5	ANZ	Athletics	800 metres, Men	56338	Harvey Sutton	64808	3 h8 r1/2	NULL	false
1908 Summer Olympics	5	ANZ	Athletics	800 metres, Men	56338	Guy Haskins	922519	DNS	NULL	false
1908 Summer Olympics	5	ANZ	Athletics	800 metres, Men	56338	Joseph Lynch	64735	DNS	NULL	false

only showing top 5 rows

Таблиці в срібному шарі:

athlete_bio таблиця усього збережена в silver шарі

athlete_id	name	sex	born	height	weight	country	country_noc	description	special_notes
52558	JuulichKurizawa	Male	25February1965	198	75	Japan	JPN	None	None
36298	PatKalter	Male	10February1959	198	87	Canada	CAN	PatKalterwongoldindoublesculwithBruceFordatthe1979PanAmericanGames.(ListedinOlympiansWhonotMedalistsattheSummerPanAmericanGames19815795SanJuanOlympicindoublesculisListedinOlympiansWhonotMedalistsattheBritisEmpireCommonwealthGames19815802EdinburghOlympicindoublesculis)	
1151985	MichaelJozif	Male	24January1981	177	88	Germany	GER	None	None
32638	KonradAffolter	Male	17May1954	187	87	Switzerland	SUI	None	None
88995	LottaAlmlberg	Female	30October1962	177	67	Sweden	SWE	None	None

only showing top 5 rows

athlete_event_results таблиця усього збережена в silver шарі

edition	edition_id	country_noc	sport	event	result_id	athlete	athlete_id	pos	medal	isTeamSport
1988SummerOlympics	5	AMZ	Athletics	800metres, Men	56338	GuyHaskins	932519	DNS	None	False
1988SummerOlympics	22	ANT	Athletics	440metresRelay, Men	63424	OralSelridge	64355	6th/13	None	True
2012SummerOlympics	54	ANT	Athletics	200metres, Men	382343	BrendanChristian	189523	5th/13	None	False
2016SummerOlympics	59	BRN	Athletics	5, 800metres, Men	358633	BirhanuBalaw	132668	9	None	False
2016SummerOlympics	59	BRN	Shooting	SmallBore Rifle, Prone, 50metres, Men	395728	MahmoodHaji	132678	46	None	False

only showing top 5 rows

Таблиці в золотому шарі:

sport	medal	sex	country_noc	avg_height	avg_weight	timestamp
CrossCountrySkiing	None	Male	ARM	170.45454545454547	67.13636363636364	2025-04-08 15:03:25.573693
Swimming	None	Male	DEN	189.67708333333334	82.20833333333333	2025-04-08 15:03:25.573693
Shooting	None	Male	GDR	177.59183673469389	76.63265306122449	2025-04-08 15:03:25.573693
IceHockey	None	Male	SVK	185.5939393939394	90.61818181818182	2025-04-08 15:03:25.573693
FreestyleSkiing	None	Male	GBR	172.38888888888889	70.05555555555556	2025-04-08 15:03:25.573693
Sailing	None	Male	BER	179.04761904761904	81.47619047619048	2025-04-08 15:03:25.573693
Football	None	Male	SWE	182.7741935483871	78.64516129032258	2025-04-08 15:03:25.573693
Swimming	None	Female	JOR	168.2	58.0	2025-04-08 15:03:25.573693
SkiJumping	None	Male	SLO	176.5897435897436	60.833333333333336	2025-04-08 15:03:25.573693
Handball	None	Male	SWE	189.359375	89.0	2025-04-08 15:03:25.573693
CrossCountrySkiing	None	Female	CZE	166.648	57.12	2025-04-08 15:03:25.573693
CyclingTrack	Gold	Male	FRA	180.1578947368421	77.42105263157895	2025-04-08 15:03:25.573693
Swimming	None	Male	NCA	175.5	68.66666666666667	2025-04-08 15:03:25.573693
Judo	Silver	Female	JPN	160.6	12569.0	2025-04-08 15:03:25.573693
Volleyball	None	Female	PER	173.25	65.4375	2025-04-08 15:03:25.573693
CrossCountrySkiing	None	Female	NOR	170.18243243243242	59.21621621621622	2025-04-08 15:03:25.573693
CyclingTrack	None	Male	BEL	178.86153846153846	73.16923076923077	2025-04-08 15:03:25.573693
Weightlifting	None	Male	CAN	172.78947368421052	9640.342105263158	2025-04-08 15:03:25.573693
Swimming	None	Female	BAH	173.15384615384616	63.23076923076923	2025-04-08 15:03:25.573693
Athletics	Bronze	Male	SUI	200.0	128.0	2025-04-08 15:03:25.573693

only showing top 20 rows

DAG-файл з повним кодом “chuboo_fin_p_dag.py” на репозиторії GoIT на GitHub:

goitacademy / airflow_sandbox

CodeIssuesPull requests4ActionsProjectsSecurityInsights

Files

main

Go to file

.github

idea

SergiiR

airflow_sandbox

dags

encrypted_file

DZ7_DAG.py_encrypted_parts.tar.gz

Sergii_S.tar.gz

Sergii_S.tar.gz_encrypted_parts.tar...

airflow_practice.py_encrypted_part...

bronze_to_silver.py_encrypted_par...

chuboo_fin_p_dag.py_encrypted_p...

airflow_sandbox / encrypted_file

SanSan987 Додавання DAG до папки dags 940ff1 · 1 minute ago History

Name	Last commit message	Last commit date
..		
DZ7_DAG.py_encrypted_parts.tar.gz	Шифрування файлів та додавання до репозиторію_Sergii	2 weeks ago
Sergii_S.tar.gz	Шифрування	15 hours ago
Sergii_S.tar.gz_encrypted_parts.tar.gz	Шифрування	15 hours ago
airflow_practice.py_encrypted_parts.tar.gz	Шифрування файлів та додавання до репозиторію.	4 days ago
bronze_to_silver.py_encrypted_parts.tar.gz	Шифрування	15 hours ago
chuboo_fin_p_dag.py_encrypted_parts.tar.gz	Додавання DAG до папки dags	1 minute ago
chuboo_fin_pro_dag.py_encrypted_parts.tar.gz	Додавання DAG до папки dags	1 minute ago
fp_dag_ok_encrypted_parts.tar.gz	ok, fp dag	15 hours ago

DAG-файл з кодом “chuboo_fin_p_dag.py” на репозиторії GoIT переміщений в папку «days», з якої файл повинен синхронізуватися та автоматично запускатися на Airflow (GoIT):

The image consists of two screenshots of a GitHub repository interface. The top screenshot shows the 'airflow_sandbox / days' directory. A commit message from SanSan987 indicates the removal of the 'chuboo_fin_pro_dag.py' file. A table lists files in the 'days' directory: '..' (parent directory), 'chuboo_fin_p_dag.py' (added 22 minutes ago), and 'viach_hw_de_07_v2.py' (added 2 weeks ago). The bottom screenshot shows the content of 'chuboo_fin_p_dag.py'. The code defines a DAG with two tasks: 'landing_to_bronze' and 'bronze_to_silver', both using SparkSubmitOperator. The DAG is named 'chuboo_fin_p' and has a start date of 2024-01-01.

airflow_sandbox / days /

SanSan987 Видалення застарілого DAG-файлу chuboo_fin_pro_dag.py

Name	Last commit message	Last commit date
..		
chuboo_fin_p_dag.py	Додавання DAG до папки days	22 minutes ago
viach_hw_de_07_v2.py	Зменшив затримку до 10 секунд для перевірки успішного сенсора	2 weeks ago

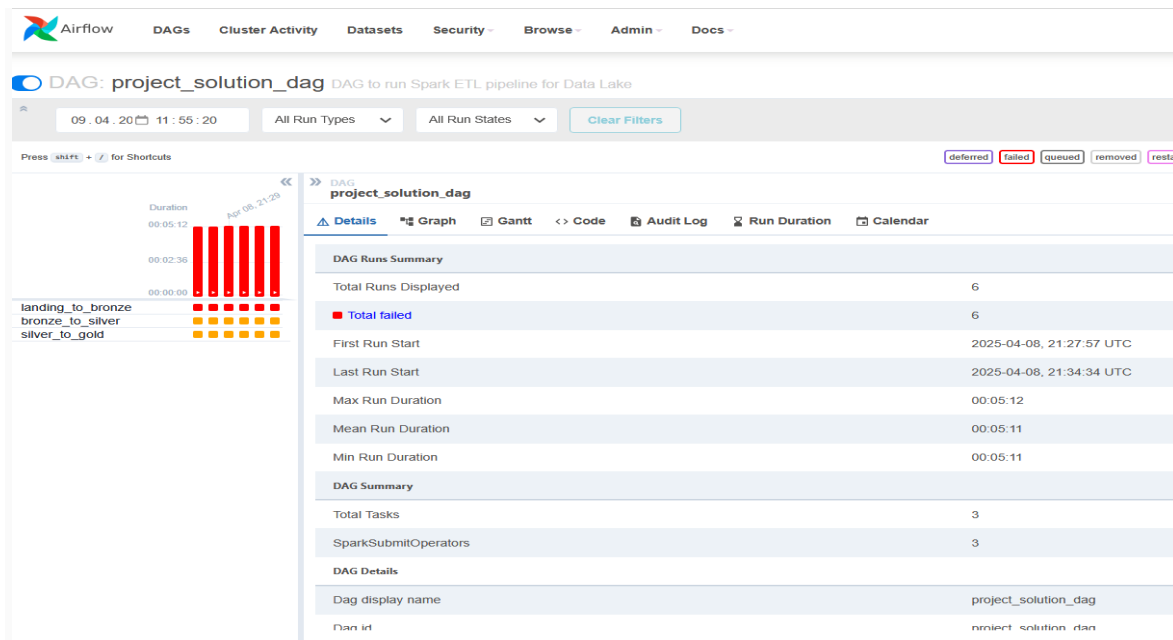
airflow_sandbox / days / chuboo_fin_p_dag.py

SanSan987 Додавання DAG до папки days

```
1  #%%writefile airflow_sandbox/days/chuboo_fin_p_dag.py
2  from airflow import DAG
3  from airflow.providers.apache.spark.operators.spark_submit import SparkSubmitOperator
4  from datetime import datetime
5
6  # DAG definition
7  default_args = {
8      'owner': 'airflow',
9      'start_date': datetime(2024, 1, 1),
10     'retries': 1
11 }
12
13 dag = DAG(
14     dag_id='chuboo_fin_p',
15     default_args=default_args,
16     description='DAG to run Spark ETL pipeline for Data Lake',
17     schedule_interval=None, # On-demand
18     catchup=False
19 )
20
21 # Task 1: Landing to Bronze
22 landing_to_bronze = SparkSubmitOperator(
23     task_id='landing_to_bronze',
24     application='days/landing_to_bronze.py',
25     conn_id='spark-default',
26     verbose=True,
27     dag=dag
28 )
29
30 # Task 2: Bronze to Silver
31 bronze_to_silver = SparkSubmitOperator(
32     task_id='bronze_to_silver',
33     application='days/bronze_to_silver.py',
34     conn_id='spark-default',
35     verbose=True,
36     dag=dag
37 )
```

Отже, шляхом виконання належних команд через Google Colab мій проектний DAG-файл з кодом “chuboo_fin_p_dag.py” надійшов до репозиторію GoIT і був переміщений там в папку «days», з якої він повинен був запускатися на сервері Airflow (GoIT). Однак станом після сплину 2 годин з моменту вчинення вказаних дій, запуск на сервері Airflow (GoIT) не з'явився.

Найбільш ймовірно така ситуація з не запуском DAG-файлу трапилася з незалежних від мене причин. Додатковим підтвердженням цього є те, що попередній мій DAG-файл “chuboo_fin_pro_dag.py” з dag_id='project_solution_dag' був запуснений на сервері Airflow (GoIT):



Однак в тому DAG-фалі була виявлена помилка:

в рядках коду: "application='dags/oleksiy/...'", який був взятий з конспекту, потрібно було видалити 'oleksiy'.

В новому DAG-файлі "chuboo_fin_p_dag.py" цю помилку було виправлено:

"application='dags/...'"

Відповідно DAG-файл "chuboo_fin_p_dag.py" повинен належним чином виконуватися на сервері Airflow (GoIT).