

Nama: Syava Aprilia Puspitasari

NIM : 2241760129

Spark SQL, DataSources, DataFrame, dan Dataset APIs

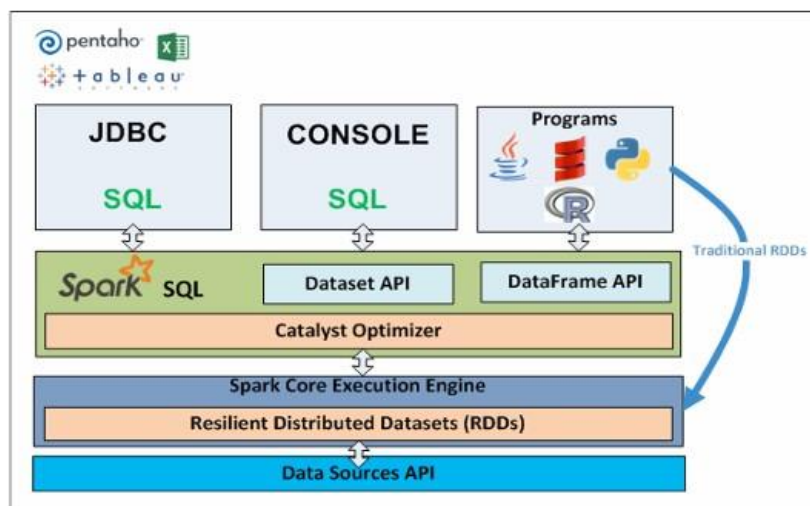
Tujuan Pembelajaran

1. Memahami konsep **Spark SQL** dan kegunaannya dalam pemrosesan data terstruktur.
2. Menguasai penggunaan **DataFrame** dan **Dataset APIs** untuk manipulasi data.
3. Mempelajari berbagai **DataSources** yang didukung Spark (JSON, CSV, Parquet, JDBC, dll.).
4. Membangun pipeline ETL sederhana menggunakan Spark SQL.

1. Pengenalan Spark SQL

Spark SQL adalah modul Apache Spark untuk pemrosesan data terstruktur. Fitur utamanya:

- **DataFrame & Dataset API:** Abstraksi data terstruktur dengan optimasi query.
- **SQL Support:** Eksekusi query SQL standar.
- **Data Source API:** Integrasi dengan berbagai format data (JSON, CSV, Parquet, Hive, JDBC, dll.).
- **Optimasi dengan Catalyst Optimizer:** Meningkatkan performa query secara otomatis.



Gambar. Arsitektur Spark SQL

2. DataFrame & Dataset APIs

Perbedaan DataFrame dan Dataset

DataFrame	Dataset
Koleksi data terdistribusi dalam bentuk <i>row</i> dengan skema	Koleksi data terdistribusi dengan tipe <i>stronglytyped</i> (Scala/Java)
Optimasi eksekusi (Tungsten, Catalyst)	Memiliki keunggulan type-safety (compile-time checking)
Dibangun di atas RDD	Hanya tersedia di Scala & Java

Contoh Pembuatan DataFrame

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

spark = SparkSession.builder.appName("DataFrameDemo").getOrCreate()

# Membuat DataFrame dari List
data = [("Alice", 25), ("Bob", 30), ("Charlie", 35)]
df = spark.createDataFrame(data, ["name", "age"])
df.show()
```

```
+-----+----+
|  name|age|
+-----+----+
|  Alice| 25|
|   Bob| 30|
|Charlie| 35|
+-----+----+
```

3. DataSources di Spark

Spark mendukung berbagai format data:

- **JSON**
- **CSV**
- **Parquet** (format kolumnar yang efisien)
- **JDBC** (koneksi database relasional)
- **Avro, ORC, Hive**, dll.

Contoh Membaca & Menulis Data

```
# Baca file CSV
```

```
df_csv = spark.read.csv("data.csv", header=True, inferSchema=True)

# Baca file JSON
df_json = spark.read.json("data.json")

# Baca dari Parquet
df_parquet = spark.read.parquet("data.parquet")

# Simpan DataFrame ke format berbeda
df.write.mode("overwrite").parquet("output.parquet")
```

4. Operasi DataFrame & SQL

Transformasi DataFrame

```
# Filter data
df_filtered = df.filter(df["age"] > 30)

# Select kolom
df_selected = df.select("name", "age")

# GroupBy & Agregasi
df_grouped = df.groupBy("name").agg({"age": "avg"})

# Join DataFrame df_join = df1.join(df2, df1["id"] == df2["id"],
"inner")
```

Menggunakan SQL Query

```
# Daftarkan DataFrame sebagai temporary view
df.createOrReplaceTempView("people")

# Eksekusi query SQL result = spark.sql("SELECT name, age FROM people
WHERE age > 30")
result.show()
```

Siapkan lingkungan Spark Cluster

Containers [Give feedback](#)

View all your running containers and applications. [Learn more](#)

<input type="checkbox"/>	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Ac
<input type="checkbox"/>	spark-master	36208e43c262	apache/spark-la	7077:7077 Show all ports (2)	0.13%	51 minutes ago	
<input type="checkbox"/>	spark-worker1	30f0a8fc4ab6	apache/spark-la		0.12%	50 minutes ago	
<input type="checkbox"/>	spark-worker2	2acca453017d	apache/spark-la		0.12%	49 minutes ago	



Spark Master at spark://172.18.0.2:7077

URL: spark://172.18.0.2:7077

Alive Workers: 2

Cores in use: 2 Total, 0 Used

Memory in use: 2.0 GiB Total, 0.0 B Used

Resources in use:

Applications: 0 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers (2)

Worker Id	Address
worker-20250422020128-172.18.0.3-32957	172.18.0.3:32957
worker-20250422020143-172.18.0.4-41673	172.18.0.4:41673

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per
----------------	------	-------	---------------------	---------------

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per
----------------	------	-------	---------------------	---------------

Jalankan Jupyter notebook dengan network yang sama dengan spark cluster, dan daftarkan envirotnment spark cluster

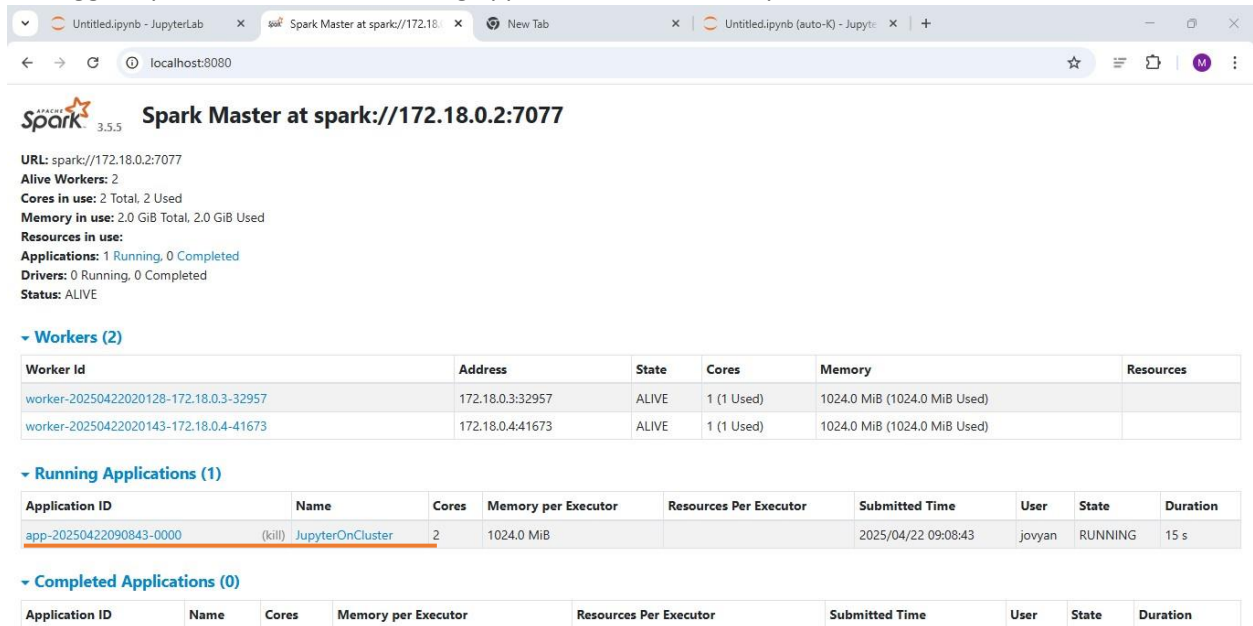
```
PS C:\Users\Acer> docker run -it -p 8888:8888 -p 4040:4040 --network spark-net -e SPARK_MASTER=spark://spark-master:7077 jupyter/all-spark-notebook
```

Buat spark session yang menggunakan spark cluster untuk lingkungan eksekusi

```
Untitled.ipynb
[6]: from pyspark.sql import SparkSession
import os

spark = SparkSession.builder \
    .master("spark://spark-master:7077") \
    .appName("JupyterOnCluster") \
    .config("spark.driver.host", os.environ.get('SPARK_DRIVER_HOST', 'localhost')) \
    .config("spark.driver.port", "8888") \
    .config("spark.ui.port", "4040") \
    .config("spark.executor.memory", "1g") \
    .config("spark.cores.max", "2") \
    .getOrCreate()
```

Sehingga dapat kita lihat ada 1 running application di halaman Spark UI



The screenshot shows the Spark Master UI at spark://172.18.0.2:7077. It displays system metrics such as URL, alive workers (2), cores in use (2 total, 2 used), and memory in use (2.0 GiB total, 2.0 GiB used). It also shows resources in use, applications (1 running, 0 completed), drivers (0 running, 0 completed), and status (ALIVE).

Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
worker-20250422020128-172.18.0.3-32957	172.18.0.3:32957	ALIVE	1 (1 Used)	1024.0 MiB (1024.0 MiB Used)	
worker-20250422020143-172.18.0.4-41673	172.18.0.4:41673	ALIVE	1 (1 Used)	1024.0 MiB (1024.0 MiB Used)	

Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20250422090843-0000	(kill) JupyterOnCluster	2	1024.0 MiB		2025/04/22 09:08:43	jovyan	RUNNING	15 s

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

5. Praktikum: Membangun ETL Pipeline

Tugas

1. **Extract:** Baca data dari file CSV (sales_data.csv).
2. **Transform:**
 - o Filter transaksi dengan Revenue > \$100.
 - o Hitung total penjualan per kategori.
3. **Load:** Simpan hasil ke Parquet.

Solusi

```
[*]: from pyspark.sql import SparkSession
import os
from pyspark.sql.functions import month, sum, count, col

spark = SparkSession.builder \
    .master("spark://spark-master:7077") \
    .appName("JupyterOnCluster") \
    .config("spark.driver.host", os.environ.get('SPARK_DRIVER_HOST', 'localhost')) \
    .config("spark.driver.port", "8888") \
    .config("spark.ui.port", "4040") \
    .config("spark.executor.memory", "1g") \
    .config("spark.cores.max", "2") \
    .getOrCreate()

# Extract
df = spark.read.csv("sales_data.csv", header=True, inferSchema=True)

# Transform
df_filtered = df.filter(col("Revenue") > 100)
df_result = df_filtered.groupBy("Product_Category").agg(sum("Revenue").alias("total_sales"))
df_result.show()

# Load
df_result.write.mode("overwrite").parquet("output_sales.parquet")
spark.stop()
```

Woul
Dlacc

```
+-----+-----+
|Product_Category|total_sales|
+-----+-----+
|      Clothing|      8198902|
|   Accessories|   13559164|
|         Bikes|   61782134|
+-----+-----+
```

Jawab:

The screenshot displays a JupyterLab environment. On the left, the file explorer shows a directory structure with files 'output_sale...', 'sales_data.csv', and 'Jobsheet9_...'. The code editor on the right contains a PySpark script that reads 'sales_data.csv', filters for revenue > 100, groups by product category, and calculates total sales. The output is a table showing total sales for Clothing, Accessories, and Bikes.

```
[2]: from pyspark.sql import SparkSession
from pyspark.sql.functions import sum, col

# Inisialisasi SparkSession
spark = SparkSession.builder \
    .appName("ETL_Pipeline") \
    .getOrCreate()

# Extract: Baca file CSV
df = spark.read.csv("sales_data.csv", header=True)

# Transform: Filter Revenue > 100 dan hitung total
df_filtered = df.filter(col("Revenue") > 100)
df_result = df_filtered.groupBy("Product_Category") \
    .agg(sum("Revenue").alias("total_sales"))

# Tampilkan hasil transformasi
df_result.show()

# Load: Simpan ke file Parquet
df_result.write.mode("overwrite").parquet("output")

# Tutup sesi Spark
spark.stop()
```

Product_Category	total_sales
Clothing	8198902
Accessories	13559164
Bikes	61782134

6. Analisis Data Retail

Dataset

- **Format:** CSV (sales_data.csv)

Tugas

1. Hitung total pendapatan per bulan.
2. Identifikasi 5 produk terlaris.
3. Simpan hasil dalam format Parquet.

Solusi

1. Pendapatan perbulan

```
df = spark.read.csv("sales_data.csv", header=True, inferSchema=True)

# Pendapatan per bulan
df_revenue = df.withColumn("month", month("Date")) \
    .groupBy("month") \
    .agg(sum(df["Unit_Price"] * df["Order_Quantity"]).alias("total_revenue"))

df_revenue.show()
```

month	total_revenue
12	10158080
1	7832338
6	10085537
3	8201790
5	9859851
9	6517880
4	8485163
8	6348349
7	6392045
10	6709394
11	6977157

Jawab:


```
[7]: from pyspark.sql import SparkSession
from pyspark.sql.functions import month, sum, coalesce

# 1. Start ulang SparkSession
spark = SparkSession.builder \
    .appName("Sales Analysis") \
    .getOrCreate()

# 2. Load file CSV
df = spark.read.csv("sales_data.csv", header=True, inferSchema=True)

# 3. Hitung pendapatan per bulan
df_revenue = df.withColumn("month", month("Date")) \
    .groupBy("month") \
    .agg(sum(df["Unit_Price"] * df["Order_Quantity"]) as "total_revenue")

df_revenue.show()

# 4. Produk terlaris
df_top_products = df.groupBy("Product") \
    .agg(count("*").alias("total_orders")) \
    .orderBy("total_orders", ascending=False) \
    .limit(5)

df_top_products.show()
```

month	total_revenue
12	10158080
1	7832338
6	10085537
3	8201790
5	9859851
9	6517880
4	8485163
8	6348349
7	6392045
10	6709394
11	6977157
2	7608734

2. Identifikasi 5 Produk terlaris

```
[10]: # 5 produk terlaris
df_top_products = df.groupBy("Product") \
    .agg(count("*").alias("total_orders")) \
    .orderBy("total_orders", ascending=False) \
    .limit(5)

df_top_products.show()
```

Product	total_orders
Water Bottle - 30...	10794
Patch Kit/8 Patches	10416
Mountain Tire Tube	6816
AWC Logo Cap	4358
Sport-100 Helmet,...	4220

Jawab:

The screenshot shows a Databricks workspace interface. On the left, a file explorer shows the directory structure: `/ Jobsheet9_SparkSQL /` with files `output_sale...` (13 minutes ago), `sales_data.c...` (19 minutes ago), and `Jobsheet9_...` (1 minute ago). The main area displays a notebook with the following code:

```
# 4. Produk terlaris
df_top_products = df.groupBy("Product") \
    .agg(count("*").alias("total_orders")) \
    .orderBy("total_orders", ascending=False) \
    .limit(5)

df_top_products.show()
```

The output of the code is displayed in two tables. The first table shows the top 5 products by total orders:

Product	total_orders
Water Bottle - 30...	10794
Patch Kit/8 Patches	10416
Mountain Tire Tube	6816
AWC Logo Cap	4358
Sport-100 Helmet,...	4220

The second table shows the top 12 months by total revenue:

month	total_revenue
12	10158080
1	7832338
6	10085537
3	8201790
5	9859851
9	6517880
4	8485163
8	6348349
7	6392045
10	6709394
11	6977157
2	7608734

3. simpan dalam format parquet

Filter files by name

Name	Last Modified
output_sale...	29 seconds ago
revenue_by...	46 seconds ago
top_produc...	46 seconds ago
work	last year
sales_data.c...	20 minutes ago
Untitled.ipyn...	now

```
[11]: # Simpan hasil
df_revenue.write.parquet("revenue_by_month.parquet")
df_top_products.write.parquet("top_products.parquet")
```

Jawab:

Filter files by name

Name	Last Modified
top_produc...	2 minutes ago
revenue_by...	2 minutes ago
output_sale...	21 minutes ago
sales_data.c...	27 minutes ago
Jobsheet9_...	1 minute ago

```
[10]: df_revenue.write.parquet("revenue_by_month.parquet")
df_top_products.write.parquet("top_products.parquet")
```

7. Evaluasi

Soal Latihan

1. Baca data dari table di database MySQL anda menggunakan Spark, dengan cara berikut

```
df = spark.read.format("jdbc") \
    .option("url", "jdbc:mysql://localhost:3306/db") \
    .option("dbtable", "table_name") \
    .option("user", "user") \
    .option("password", "password") \
```

```
.load()
```

Baca table apa saja.

2. Buat query Spark SQL untuk menghitung Jumlah row dalam table tersebut

Kesimpulan

- Spark SQL menyediakan antarmuka terstruktur untuk pemrosesan data besar.
- DataFrame & Dataset APIs memungkinkan manipulasi data dengan sintaks mirip SQL.
- DataSources API mendukung integrasi dengan berbagai format penyimpanan.