# Spark SQL, DataSources, DataFrame, dan Dataset APIs

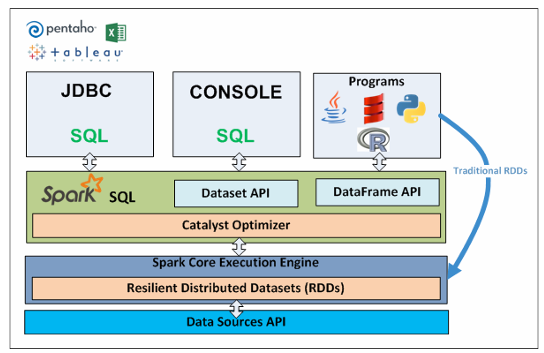
## Tujuan Pembelajaran

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 *row* dengan skema | Koleksi data terdistribusi dengan tipe *strongly-typed* (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**



### 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**

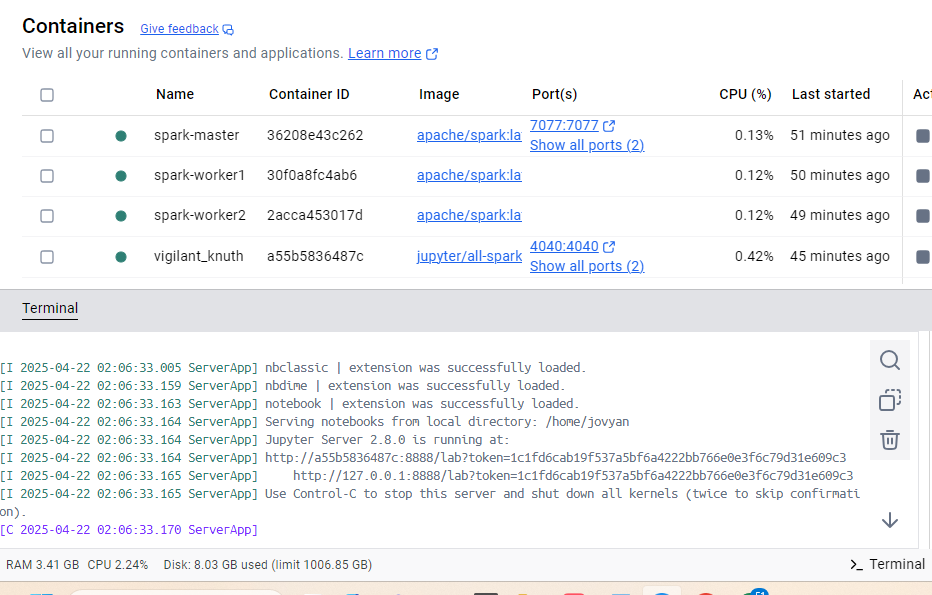
**Transformasi DataFrame**

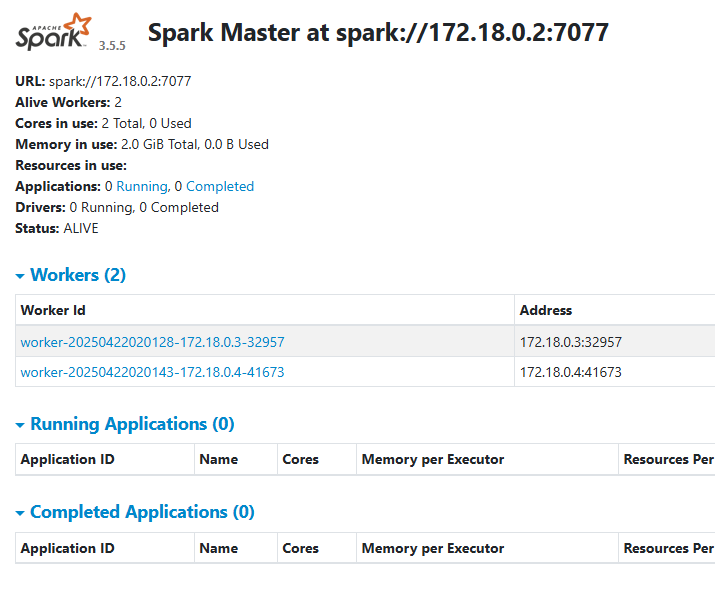
|  |
| --- |
| # Filter data  df\_filtered = df.filter(df["age"] > 30)  # Select kolom  df\_selected = df.select("name", "age")  # GroupBy & Aggregasi  df\_grouped = df.groupBy("name").agg({"age": "avg"})  # Join DataFrame  df\_join = df1.join(df2, df1["id"] == df2["id"], "inner") |

**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





## 5. Praktikum: Membangun ETL Pipeline

**Tugas**

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

**Solusi**

|  |
| --- |
| from pyspark.sql import SparkSession  from pyspark.sql.functions import col, sum  spark = SparkSession.builder.appName("ETLPipeline").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"))  *# Load*  df\_result.write.mode("overwrite").parquet("output\_sales.parquet")  spark.stop() |



### 6. Analisis Data Retail

**Dataset**

* **Format**: CSV (sales\_data.csv)

**Tugas**

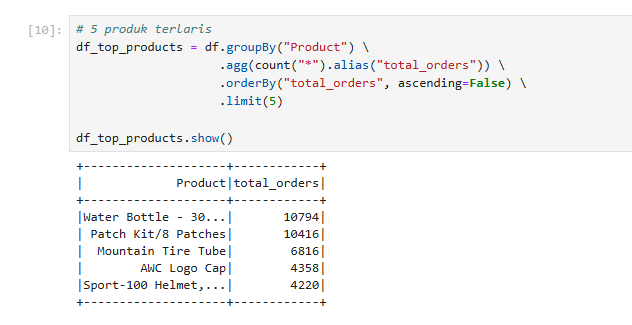
1. Hitung total pendapatan per bulan.
2. Identifikasi 5 produk terlaris.
3. Simpan hasil dalam format Parquet.

**Solusi**

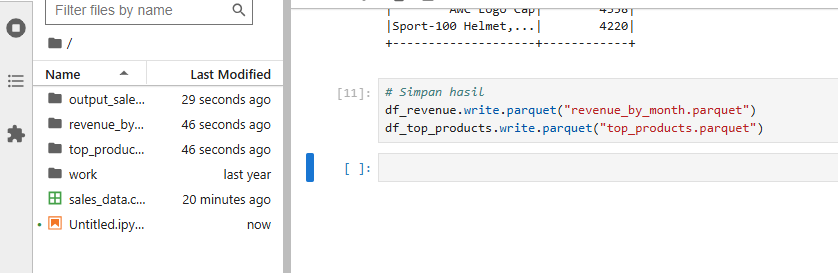
1. Pendapatan perbulan



2. Identifikasi 5 Produk terlaris



3. simpan dalam format 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.

1. 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.