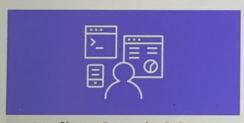
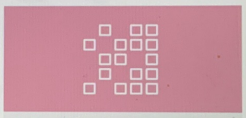
Spark SQL

What you will learn



Define Spark SQL



Identify the steps to use Spark SQL for DataFrame operations

What is Spark SQL?

for querying structured and semi structured data using SQL-like syntax

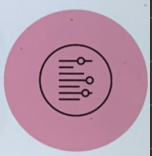
- Is a component of Apache Spark
- Provides a programming interface
- Enables seamless integration
- Helps perform relational and data manipulation operations

What is Spark SQL?

- Supports data sources like:
  - Hive tables
  - Parquet files
  - JSON
  - JDBC
- Provides optimizations for efficient execution

Spark SQL: Steps

1. Import the necessary libraries
2. Create a SparkSession
3. Load the data into a DataFrame
4. Register the DataFrame as a temporary view
5. Perform SQL queries
6. Show the results
7. Stop the Spark session



Spark SQL: Steps

Step 1 Import necessary libraries

- Serves as the entry point for Spark functionality

```
from pyspark.sql import SparkSession
```

Step 2 Create a SparkSession

- Use a specified application name

```
spark = SparkSession.builder.appName("SparkSQLExample").getOrCreate()
```

Spark SQL: Steps

Step 3 Load the data into a DataFrame

- Provides file path, header, and inferSchema parameters

```
data = spark.read.format("csv").option("header","true").load
("path_to_your_data.csv")
```

Step 4 Register DataFrame as a temporary view

- Register DataFrame using SQL syntax

```
data.createOrReplaceTempView("my_table")
```

Spark SQL: Steps

Step 5 Perform SQL queries

- Write SQL queries to analyze your data

```
result = spark.sql("SELECT column1, column2 FROM my_table")
```

Step 6 Show the result

- Print the first few result rows

```
Result.show()
```

Spark SQL: Steps

Step 7 Stop the SparkSession

- Releases the resources

```
spark.stop()
```

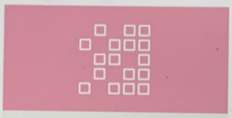
Use Spark SQL to load data, register it as a temporary view, perform SQL queries, and display the query results

ETL Workloads

Objectives

After watching this video, you will be able to:

Define ETL (Extract, Transform, Load)



Describe how to extract, transform and load data using Apache Spark

ETL – What is it?

ETL describes the process of moving data from a source to another destination that has a different data format or structure

- **Extract** obtains data from a source
- **Transform** the data in the needed output format
- **Load** the data into a database, data warehouse or other storage

ETL with Apache Spark

Apache Spark offers the following ETL advantages:

- Provides a well-supported big data ecosystem
- Can easily load and save popular big data sources
- Scales easily to handle large workloads

Extracting Data

Extracts data from one or more different sources Spark supports HDFS & the following data sources:

- Parquet
- Apache ORC
- Hive
- JDBC

```
# Load a sparkParquet file into a DataFrame
df =.read.parquet("people.parquet")
```

Transforming data

- Cleans the data
- Transforms data format to make the data more accessible for analysis
- Joins DataFrames
- Groups and aggregates data
- Uses Spark SQL operations to Select and others

```
# Create view of data for SQL queries
df.createOrReplaceTempView("people")
```

```
# Use Spark SQL to clean and transform data
names = spark.sql("SELECT name FROM
people WHERE age BETWEEN 13 AND 19")
```

Loading data

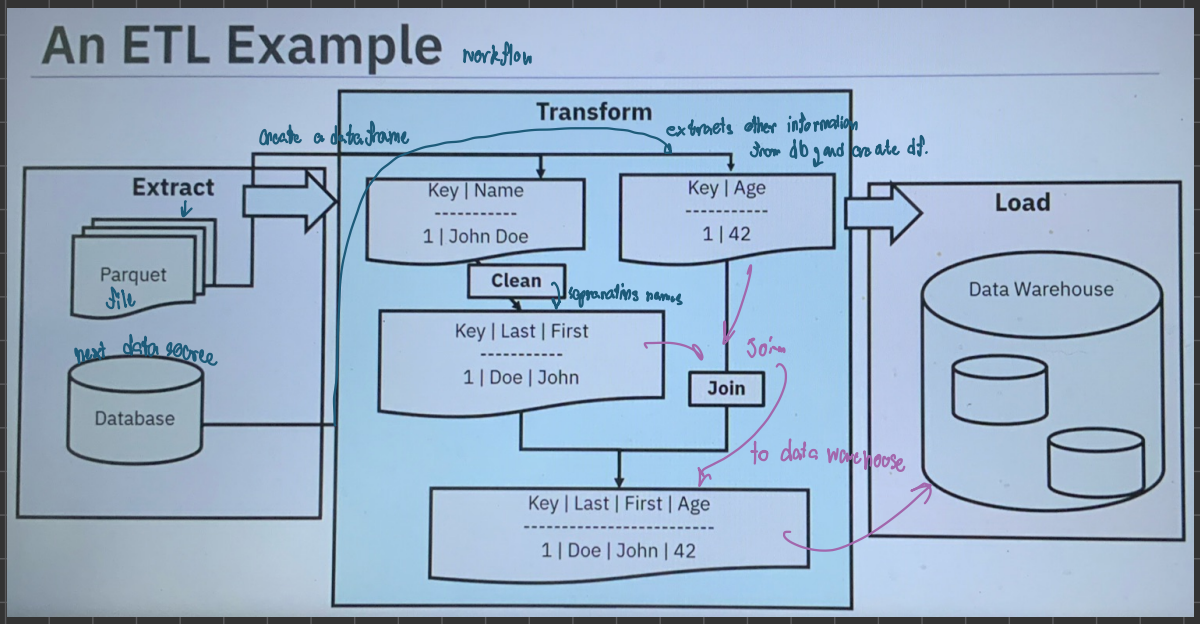
- Loads data into data warehouse, database or other data sink
- Uses available Spark data sources

```
# Ensure PostgreSQL driver is
# added to application classpath
./bin/spark-shell \
 --jars postgresql-9.4.1207.jar
```

```
# Load the DataFrame to PostgreSQL
jdbcDF.write
 .format("jdbc")
 .option("url", "jdbc:postgresql:dbserver")
 .option("dbtable", "schema.tablename")
 .option("user", "username")
 .option("password", "password")
 .save()
```

An ETL Example

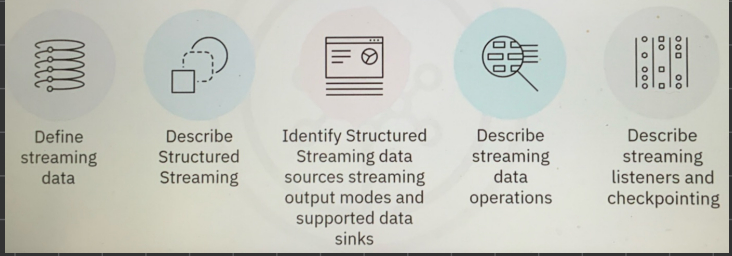
workflow



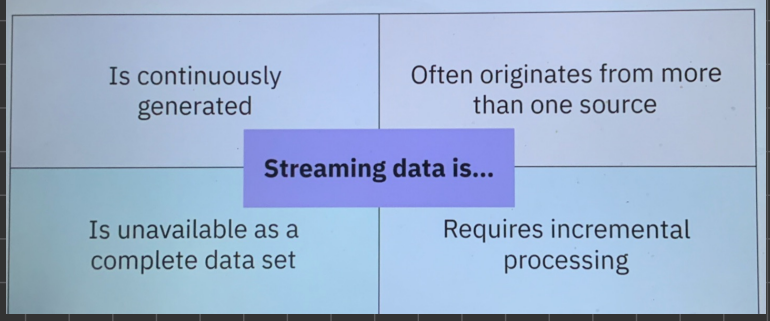
Spark Structured Streaming

Objectives

After watching this video, you will be able to:



What is streaming data



What is Structured Streaming

Apache Spark Structured Streaming:

- Uses Spark SQL
- Uses the same DataFrame and Dataset APIs
- Processes data in Micro-batches or continuously
- Optimizes queries using Spark SQL

Common Structured Streaming terms

Source	The data origination location
Sink	The location of the output
Event time	The record creation time the timestamp when a data point is created ↳ it's not the same time as arrival time or processing time
End-to-end latency	The measurement of the time needed for the data to go from source to sink
Watermarking	Manages late arriving data

Streaming Data Sources

- Files – streams files from a directory
- Kafka – streams from Apache Kafka
- IP Sockets and Rate sources – used for testing only
- ... and others

Streaming Data Operations

Apache Spark Structured Streaming: runs on top of the Spark SQL engine

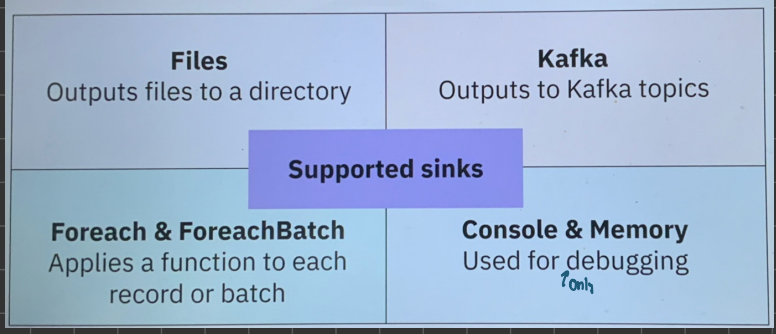
- Performs Standard SQL operations including select, projection, and aggregation
- Enables window operations over event time – sliding windows with aggregations
- Supports join operations – joins with static DataFrames or other streams

Output Modes

Output modes specify how data is written to the sink

- Append – Only new rows added
- Complete – Entire result table
- Update – Only updated rows

Streaming Data Sinks



Monitoring and Checkpointing

- Monitor your data via Spark external listeners, which work both through external frameworks or programmatically on data streams to trigger events.
- Checkpointing recovers query progress on failure (set checkpoint location on HDFS)

Feature Extration and Transformation

Data preprocessing

- Involves cleaning, transforming, and preparing raw data
- Used for analytical and machine learning tasks
- Involves feature extraction and transformation

Feature extraction

- Selecting or deriving relevant features from raw data
- Assists with analysis or modeling tasks
  selected feature should have a strong correlation with the target variable or capture important patterns and relationships in the data
- Involves techniques like:
  - PCA
  - Factor analysis
  - Feature engineering

Feature extraction using Spark

Apache Spark:

- An open-source data processing framework
- Supports distributed computing on large datasets
- Spark's MLlib offers:
  - Tokenzation
  - TF-IDF
  - Word2Vec
  - PCA

Feature extraction techniques

Tokenization

- Splits text data into individual words or tokens
- Uses Tokenizer class to tokenize text data

TF-IDF

- Identifies important words in a text document
- Uses TF-IDF class to calculate the TF-IDF score

Feature extraction techniques

Word2Vec

- Represents words in a text document as vectors
- Uses Word2Vec class to generate word vectors

PCA

- Identifies small set of features to explain variance
- Transforms high-dimensional data into lower-dimensional space
- Uses PCA class to perform PCA on a given dataset

Feature extraction steps using Spark

Step 1: Import Spark libraries

Step 2: Load data into Spark RDD or DataFrame

Step 3: Define the feature extraction technique

Step 4: Convert extracted features into a suitable format

Feature transformation

- Converts extracted features into a suitable format
- Involves modifying or manipulating original features
- Creates new representatives for analysis or modeling tasks
- Improves features' quality, distribution, or relationship

Feature transformation using Spark

Apache Spark:

- Provides many functions and libraries
- Ensures data is prepared appropriately
- Techniques used:
  - Scaling and normalization
  - One-hot encoding
  - StringIndexer
  - PCA

Feature transformation techniques

Scaling and normalization

- Transforms numerical features into a common scale
- Uses functions like StandardScaler, MinMaxScaler, and MaxAbsScaler

One-hot encoding

- Converts categorical features to numerical features
- Uses OneHotEncoder function
- Performs one-hot encoding on a given dataset

Feature transformation techniques

StringIndexer

- Converts categorical strings into numerical indices
- Assigns a unique numerical index to a distinct category
- Uses StringIndexer function to perform transformations

PCA

- Reduces the dimensionality of a dataset
- Uses PCA function to perform PCA on datasets
- Improves computational efficiency, removes noise, and visualizes data

Feature transformation steps using Spark

Step 1: Load data into a Spark RDD or DataFrame

Step 2: Define the feature transformation technique

Step 3: Apply the transformation to the data

Step 4: Convert the transformed data into a suitable format

Feature extraction and transformation

- Achieving reliable and meaningful results
- Improving model accuracy
- Removing irrelevant information
- Contributing to fast and efficient training

Machine Learning Pipelines using Spark

Machine learning pipelines

Machine learning pipelines:

- A systematic approach
- Builds and deploys machine learning models
- Provides a structured process
- Organizes and automates end-to-end machine learning process

Machine learning pipelines: Steps

Data Collection and Ingestion	Data Cleaning and Preprocessing	Feature Extraction
• Acquires data from various sources • Examples: Databases, files, APIs, and streams	• Cleans and processes the raw data • Examples: Dealing with outliers, scaling, and transformation	• Selects important and relevant features • Creates new features from existing ones

Machine learning pipelines: Steps

Model Selection and Training	Model Evaluation	Model Deployment
• Selects and trains machine learning model • Uses preprocessed data	• Determines the performance of the model • Uses: Accuracy, precision, recall, and F1-score metrics	• Deploys trained model to a production environment • Creates, integrates, and monitors

Advantages of ML pipelines

- Documents the ML process making it easy to reproduce and share
- Handles large volumes of data and is scalable
- Automates routine tasks
  - cleaning data
  - extracting features
  - selecting models,
  - tuning hyperparameter

Advantages of ML pipelines

- Are modular and flexible — allow DS to experiment with different algorithms and hyperparameters without rewriting the entire pipeline
- Ensures consistent data preparation and modeling practices by following the same steps each time the pipeline runs
- Is easy to deploy and maintain

Machine learning pipelines using Spark

ML pipelines using Spark:

- Provides powerful tools
- Organizes and automates the build and deploy process

Spark MLlib:

- Is a library for machine learning in Spark
- Provides a set of high-level APIs for scalable machine learning pipeline

Key components

Data Ingestion

- Provides various data sources and APIs
- Examples: Hadoop Distributed File System (HDFS), Amazon S3, and Apache Cassandra
- Allows seamless integration and parallel processing

Data Cleaning and Preprocessing

- Offers a variety of transformation functions
- Handles large-scale data-cleaning tasks
- Examples: Handling missing values, filtering outliers, and performing data transformations

Key components

Feature Extraction

- Provides feature extraction techniques and libraries
- Examples: Tokenization, TF-IDF, and word embedding

Model Selection and Training

- Provides various ML algorithms and functionalities
- Examples: Decision trees, random forests, and logistic regression
- Trains and selects appropriate models for data
- Helps to easily select, configure, and train models

Key components

Model Evaluation

- Provides evaluation metrics and tools
- Helps calculate various evaluation metrics, perform cross-validation, and generate performance reports
- Facilitates efficient evaluation, even on large-scale datasets

Model Deployment:

- Provides various options for deploying trained model
- Examples: Exporting the model as a JAR file or as a PFA file
- Enables seamless model deployment

Advantages of ML pipelines using Spark

- Scales to handle large volumes of data
- Quick data processing and machine learning algorithms
- Simplifies data ingestion and integration tasks
- Is easy to use
- Allows to work in seamless and integrated way

Model Persistence

What you will learn

Define model persistence

List the advantages of model persistence

What is model persistence?

Saves trained model to disk for future use

Allows for later loading and utilization

Importance of model persistence

Enables reuse and sharing of trained models

Facilitates deployment of models in production environments

Loads and employs models for real-time predictions

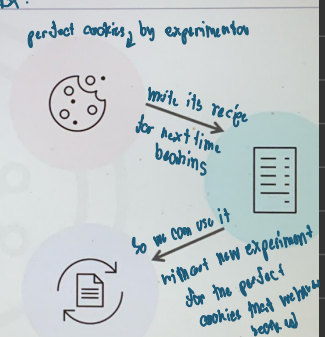
Advantages of model persistence

Some advantages in machine learning:

- Reusability
- Portability
- Efficiency
- Reproducibility
- Scalability
- Flexibility

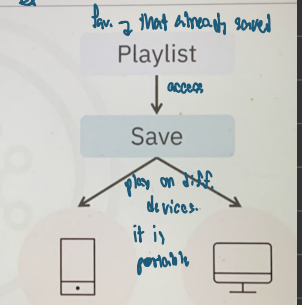
Reusability

- Allows reuse in different applications
- Eliminates the need for repetitive model training
- Reduces computational costs and time



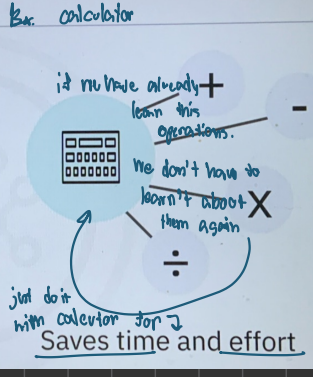
Portability

- Allows for effortless sharing
- Not concerned with the underlying computing environment or infrastructure
- Facilitates collaboration and knowledge exchange
- Promotes teamwork and enables integration



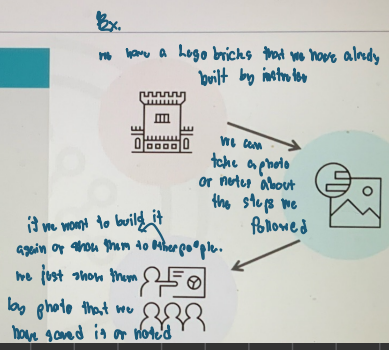
Efficiency

- Enables immediate utilization
- Saves time and computational resources
- Enables quick deployment



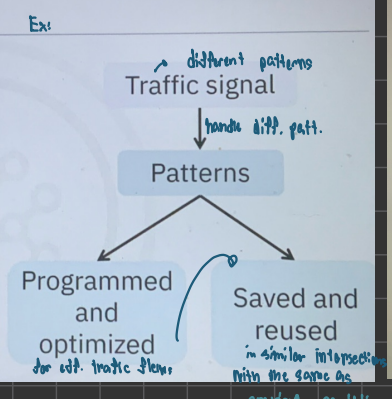
Reproducibility

- Allows replication and verification
- Ensures consistency in execution
- Promotes easy sharing of resources



Scalability

- Allows for easy deployment and scaling
- Enables efficient processing of data
- Supports real-time predictions
- Ideal for quick and accurate decision making



Flexibility

- Used in several applications
- Examples: Batch processing, real-time streaming, and web apps
- Integrates seamlessly and expands usability and applicability