SAKARYA ÜNİVERSİTESİ

BIG DATA

TOO BIG TO IGNORE

SÜMEYYE KAYNAK

OUTLINE

Spark SQL

Spark streaming

SPARK SQL

- Spark kütüphanelerinin bir parçasıdır.
- SQl benzeri script ve metotlar kullanarak kolayca analiz yapmamıza olanak sağlar.

SPARK-SQL

- Spark-sql dependence tanımlanmalıdır.
- Sparksession oluşturulmalıdır.

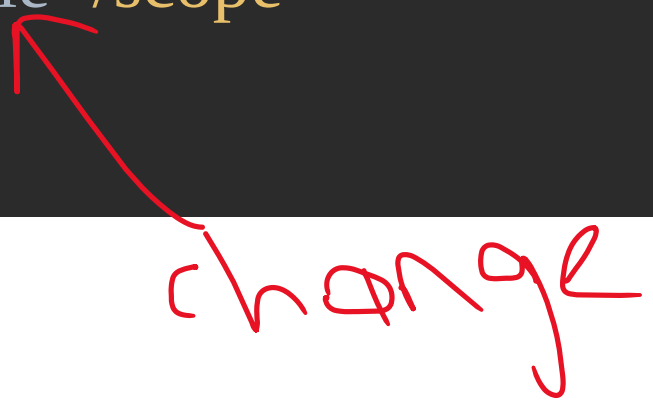
```
SparkSession spark =
SparkSession.Builder().AppName("word_count_sample").GetOrCreate();
```

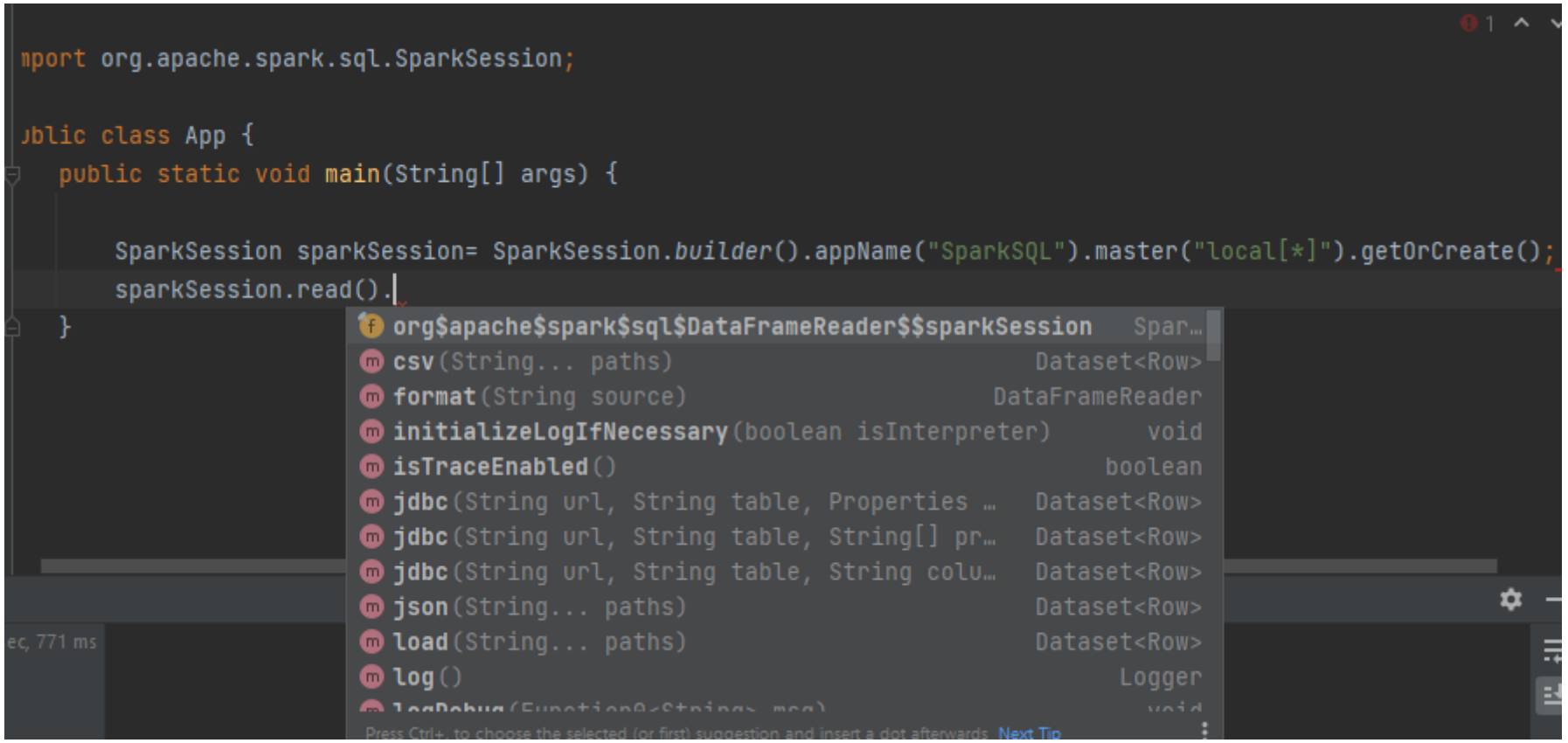
APPLICATION

- jdk version selection does not matter in spark sql.
- Select Spark project from Mavenrepository
- Create a java class
- Spark session is a part of spark-sql. Thus you need to change “provided” scope to “compile” for this library.

DEPENDENCE

```
<dependencies>
   <dependency>
      <groupId>org.apache.spark</groupId>
      <artifactId>spark-sql_2.11</artifactId>
      <version>2.0.0</version>
      <scope>compile</scope>
   </dependency>
</dependencies>
```




mport org.apache.spark.sql.SparkSession;
Jblic class App {
public static void main(String[] args) {
SparkSession sparkSession= SparkSession.builder().appName("SparkSQL").master("local[*]").getOrCreate();
sparkSession.read().
}
org$apache$sparksqlDataFrameReader$$sparkSession Spar…
csv(String... paths) Dataset<Row>
format(String source) DataFrameReader
initializeLogIfNecessary(boolean isInterpreter) void
isTraceEnabled() boolean
jdbc(String url, String table, Properties … Dataset<Row>
jdbc(String url, String table, String[] pr… Dataset<Row>
jdbc(String url, String table, String colu… Dataset<Row>
json(String... paths) Dataset<Row>
load(String... paths) Dataset<Row>
log() Logger
ec, 771 ms

```java
import org.apache.spark.sql.Row;
import org.apache.spark.sql.SparkSession;

public class App {
    public static void main(String[] args) {

        SparkSession sparkSession= SparkSession.builder().appName("SparkSQL").master("local[*]").getOrCreate();
        Dataset<Row> dataset = sparkSession.read().json( path: "C:\\bin\\FakeData.json");
        dataset.show();
    }
}
```

```java
public class App {
    public static void main(String[] args) {

        SparkSession sparksession= SparkSession.builder().appName("SQL").master("local[*]").getOrCreate();
        Dataset<Row> dataset = sparksession.read().json( path: "C:\\bin\\FakeData.json");
        /* dataset.show(); */
        dataset.printSchema();
    }
}
```

```
21/12/16 20:54:30 INFO TaskSchedulerImpl: Killin
21/12/16 20:54:30 INFO DAGScheduler: Job 0 finis
root
 |-- 123456: string (nullable = true)
 |-- Email: string (nullable = true)
 |-- FirstName: string (nullable = true)
 |-- LastName: string (nullable = true)
```

```java
public class App {
    public static void main(String[] args) {

        SparkSession sparksession= SparkSession.builder().appName("SQL").master("local[*]").getOrCreate();
        Dataset<Row> dataset = sparksession.read().option("multiline","true").json( path: "C://bin//MOCK_DATA.json");

        dataset.show();
    }
}
```

```
+--------------------+----------+-----------+---+---------------+------------+
|               email|first_name|     gender| id|     ip_address|   last_name|
+--------------------+----------+-----------+---+---------------+------------+
|ddockrey0@imagesh...|       Des| Polygender|  1|  162.85.92.210|     Dockrey|
|     nlethardy1@t.co| Nathanial|   Bigender|  2|  240.45.227.83|    Lethardy|
|bdellit2@answers.com|   Branden|Genderqueer|  3|170.126.106.178|      Dellit|
|    rmoxom3@yale.edu|   Rhianna|    Agender|  4|  176.110.4.201|       Moxom|
|jmagee4@bloglines...| Jacquelin|Genderfluid|  5| 227.71.244.133|       Magee|
|  wraun5@reuters.com|     Wendi| Polygender|  6|253.220.106.201|        Raun|
| wtilsley6@wufoo.com|     Wiley|Genderqueer|  7|   38.140.8.177|     Tilsley|
|qzorzetti7@artist...|   Quentin| Non-binary|  8|   239.7.88.129|    Zorzetti|
| kdurker8@netlog.com| Kristofor|Genderfluid|  9| 225.156.49.114|      Durker|
|bdunderdale9@netw...|     Bogey|     Female| 10|  240.77.215.31|  Dunderdale|
| dkincaida@google.nl|     Dorie|   Bigender| 11|  11.55.203.126|     Kincaid|
|kandreichikb@gmpg...|    Kailey|       Male| 12|179.199.229.197|  Andreichik|
|   kcroughanc@ed.gov|   Kendall|       Male| 13|  102.233.150.5|    Croughan|
|ebourtoumieuxd@ms...|     Elfie|   Bigender| 14| 182.196.100.70|Bourtoumieux|
|sizakse@xinhuanet...|      Suki|   Bigender| 15|   39.247.227.5|       Izaks|
|mshasnanf@mashabl...|    Michal|   Bigender| 16|  63.231.188.39|     Shasnan|
|       rdyosg@gov.uk|    Roland|       Male| 17|  22.139.120.94|        Dyos|
```

```java
public class App {
    public static void main(String[] args) {

        SparkSession sparksession= SparkSession.builder().appName("SQL").master("local[*]").getOrCreate();
        Dataset<Row> dataset = sparksession.read().option("multiline", true).json( path: "C://bin//MOCK_DATA1.json");
        Dataset<Row> filterdata = dataset.select( col: "first_name",  ...cols: "last_name");
        filterdata.show();
    }
}
```

```
+----------+------------+
|first_name|   last_name|
+----------+------------+
|       Des|     Dockrey|
| Nathanial|    Lethardy|
|   Branden|      Dellit|
|   Rhianna|       Moxom|
| Jacquelin|       Magee|
|     Wendi|        Raun|
|     Wiley|     Tilsley|
|   Quentin|    Zorzetti|
```

GROUPBY

```java
public class App {
    public static void main(String[] args) {

        SparkSession sparksession= SparkSession.builder().appName("SQL").master("local[*]").getOrCreate();
        Dataset<Row> dataset = sparksession.read().option("multiline", true).json( path: "C://bin//MOCK_DATA1.json");
        Dataset<Row> filterdata = dataset.select( col: "first_name",  ...cols: "last_name");
        Dataset<Row> groupdata= dataset.groupBy( col1: "gender").sum( ...colNames: "id");
        groupdata.show();
    }
}
```

```
+------+--------+
|gender|sum(id)|
+------+--------+
|   Man| 172220|
|Female|  71508|
| Woman| 126851|
|  Kids|  66026|
|  Male|  63895|
+------+--------+
```

CORRUPT_DATA ERROR

```
Dataset<Row> json = sparksession.read().json( path: "C://bin//generated.json");
json.show();
```

```
referenced columns only include the internal corrupt record column
(named _corrupt_record by default). For example:
spark.read.schema(schema).csv(file).filter($"_corrupt_record".isNotNull).count()
and spark.read.schema(schema).csv(file).select("_corrupt_record").show().
Instead, you can cache or save the parsed results and then send the same query.
For example, val df = spark.read.schema(schema).csv(file).cache() and then
df.filter($"_corrupt_record".isNotNull).count().
```

```
Dataset<Row> dataset2 = sparksession.read().json( path: "C://bin//MOCK_DATA1.json");
Dataset<Row> filterdata = dataset.select( col: "first_name", ...cols: "last_name");
Dataset<Row> groupdata= dataset.groupBy( col1: "gender").sum( ...colNames: "id");

/*Dataset<Row> json = sparksession.read().json("C://bin//generated.json");
json.show();*/

/* groupdata.show(); */

dataset2.show();
```

option ydk

```
21/12/16 23:03:12 INFO CodeGenerator: Code generated in 41.9293 ms
+--------------------+----------+------+---+---------------+------------+
|               email|first_name|gender| id|     ip_address|   last_name|
+--------------------+----------+------+---+---------------+------------+
|ddockrey0@imagesh...|       Des|   Man|  1|  162.85.92.210|     Dockrey|
|     nlethardy1@t.co| Nathanial|  Kids|  2|  240.45.227.83|    Lethardy|
|bdellit2@answers.com|   Branden| Woman|  3|170.126.106.178|      Dellit|
|    rmoxom3@yale.edu|   Rhianna| Woman|  4|  176.110.4.201|       Moxom|
|jmagee4@bloglines...| Jacquelin|   Man|  5| 227.71.244.133|       Magee|
|  wraun5@reuters.com|     Wendi|   Man|  6|253.220.106.201|        Raun|
```

FILTER

```
JavaRDD<CupModel> italy = map.filter(new Function<CupModel, Boolean>() {
        @Override
        public Boolean call(CupModel cupModel) throws Exception {
            boolean italy = cupModel.getFirst().equals("Italy");
            return italy;
        }
    });
```

RDD-Filter

```
public class Filter {
    public static void main(String[] args) {
        SparkSession sparksession= SparkSession.builder().appName("SQL").master("local[*]").getOrCreate();
        Dataset<Row> dataset = sparksession.read().option("multiline", true).json( path: "C://bin//MOCK_DATA1.json");

        dataset.show();

        Dataset<Row> DesData = dataset.filter(new Column( name: "first_name").equalTo("Des"));
        DesData.show();

    }
}
```

SparkSQL-Filter

FILTER

```java
import org.apache.spark.sql.Column;
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Row;
import org.apache.spark.sql.SparkSession;

public class Filter {
    public static void main(String[] args) {
        SparkSession sparksession = SparkSession.builder().appName("SQL").master("local[*]").getOrCreate();
        Dataset<Row> dataset = sparksession.read().option("multiline", true).json( path: "C://bin//generated.json");

        Dataset<Row> DesData = dataset.filter(new Column( name: "name").equalTo("Claria"));
        DesData.show();

        Dataset<Row> ManFilter = dataset.filter(new Column( name: "gender").contains("male"));
        ManFilter.show();
    }
}
```

FILTER

```
21/12/17 13:33:40 INFO DAGScheduler: Job 2 finished: show at Filter.java:15, took 0,088393 s
21/12/17 13:33:40 INFO CodeGenerator: Code generated in 41.0467 ms
+--------------------+--------------------+--------------------+---+---------+---------+--------------------+--------+-------------+----------
|                 _id|               about|             address|age|  balance|  company|               email|eyeColor|favoriteFruit|
+--------------------+--------------------+--------------------+---+---------+---------+--------------------+--------+-------------+----------
|61bb99518b92919b7...|Eiusmod reprehend...|632 Stockton Stre...| 31|$3,710.50|DATACATOR|mcmahonzimmerman@...|   green|        apple|[{0, Morris
|61bb995121db7946a...|Consequat aliqua ...|898 Hinckley Plac...| 20|$3,639.11|ISOSTREAM|concettarosales@i...|   brown|       banana|[{0, Swanso
|61bb9951c3e384f36...|Veniam deserunt a...|838 Hunterfly Pla...| 34|$2,170.20|     OULU|oliveschwartz@oul...|   green|        apple|[{0, Rachae
|61bb995156103e51f...|Est in in do cons...|117 Malta Street,...| 23|$2,460.73|      ZAJ| schultzduke@zaj.com|   brown|       banana|[{0, Kayla
|61bb9951bf0fc5bf4...|Sint aliquip amet...|838 Tech Place, L...| 29|$2,904.23|    ELITA|mooneycase@elita.com|   green|       banana|[{0, Lawrer
+--------------------+--------------------+--------------------+---+---------+---------+--------------------+--------+-------------+----------

21/12/17 13:33:41 INFO SparkContext: Invoking stop() from shutdown hook
21/12/17 13:33:41 INFO SparkUI: Stopped Spark web UI at http://DESKTOP-GK3VE45:4040
```

FILTER

```java
public class Filter {
    public static void main(String[] args) {
        SparkSession sparksession = SparkSession.builder().appName("SQL").master("local[*]").getOrCreate();
        Dataset<Row> dataset = sparksession.read().option("multiline", true).json( path: "C://bin//generated.json");

        Dataset<Row> DesData = dataset.filter(new Column( name: "name").equalTo("Claria"));
        DesData.show();

        Dataset<Row> ManFilter = dataset.filter(new Column( name: "gender").contains("male"));
        ManFilter.show();

        Dataset<Row> PhoneFilter = dataset.filter(new Column( name: "phone").contains("818"));
        PhoneFilter.show();
    }
}
```

```
+----------+------+--------------------+--------------------+-----+--------+---------+----------+----------------+----------------+--------------------
   friends|gender|            greeting|                guid|index|isActive| latitude|  longitude|            name|           phone|             pictur
+----------+------+--------------------+--------------------+-----+--------+---------+----------+----------------+----------------+--------------------
son Mad...|  male|Hello, Mcmahon Zi...|261e6fc1-7f8b-438...|    0|   false|47.832783|-122.356462|Mcmahon Zimmerman|+1 (818) 497-2647|http://placehold...
+----------+------+--------------------+--------------------+-----+--------+---------+----------+----------------+----------------+--------------------
```

GROUPBY

```java
public class Groupby {
    public static void main(String[] args) {
        SparkSession sparksession = SparkSession.builder().appName("SQL").master("local[*]").getOrCreate();
        Dataset<Row> dataset = sparksession.read().option("multiline", true).json( path: "C://bin//generated.json");

        Dataset<Row> gender = dataset.groupBy(new Column( name: "gender")).count();
        gender.show();
    }
}
```

```
21/12/17 14:20:58 INFO DAGScheduler:
21/12/17 14:20:58 INFO CodeGenerator:
+------+-----+
|gender|count|
+------+-----+
|female|    2|
|  male|    3|
+------+-----+
```

CORE AND SQL

```
JavaPairRDD<String, String> JavaPairRDD = map.mapToPair(new PairFunction<CupModel, String, String>() {
    @Override
    public Tuple2<String, String> call(CupModel cupModel) throws Exception {
        return new Tuple2<String, String>(cupModel.getFirst(), cupModel.getSecond());
    }
});
org.apache.spark.api.java.JavaPairRDD<String, Iterable<String>> result = JavaPairRDD.groupByKey();
```

```
Dataset<Row> gender = dataset.groupBy(new Column( name: "gender")).count();
```

GROUPBY-AVG

```java
public class Groupby {
    public static void main(String[] args) {
        SparkSession sparksession = SparkSession.builder().appName("SQL").master("local[*]").getOrCreate();
        Dataset<Row> dataset = sparksession.read().option("multiline", true).json( path: "C://bin//generated.json");

        Dataset<Row> gender = dataset.groupBy(new Column( name: "gender")).count();
        gender.show();

        Dataset<Row> avgAge = dataset.groupBy(new Column( name: "gender")).avg( ...colNames: "age");
        avgAge.show();
    }
}
```

```
21/12/17 14:51:22 INFO CodeGenerator:
+------+------------------+
|gender|          avg(age)|
+------+------------------+
|female|              27.0|
|  male|27.666666666666668|
+------+------------------+
```

GROUPBY

```java
public class Groupby {
    public static void main(String[] args) {
        SparkSession sparksession = SparkSession.builder().appName("SQL").master("local[*]").getOrCreate();
        Dataset<Row> dataset = sparksession.read().option("multiline", true).json( path: "C://bin//generated.json");

        Dataset<Row> gender = dataset.groupBy(new Column( name: "gender")).count();
        gender.show();

        Dataset<Row> avgAge = dataset.groupBy(new Column( name: "gender")).avg( ...colNames: "age");
        avgAge.show();

        Dataset<Row> max = dataset.groupBy(new Column( name: "gender")).max( ...colNames: "age");
        max.show();
    }
}
```

SQL

```java
public class Sql {
    public static void main(String[] args) {
        SparkSession sparksession = SparkSession.builder().appName("SQL").master("local[*]").getOrCreate();
        Dataset<Row> dataset = sparksession.read().option("multiline", true).json( path: "C://bin//generated.json");

        dataset.createOrReplaceTempView( viewName: "person");
        Dataset<Row> sql = sparksession.sql( sqlText: "select name,age from person");
        sql.show();
    }
}
```

```
+-----------------+---+
|             name|age|
+-----------------+---+
|Mcmahon Zimmerman| 31|
| Concetta Rosales| 20|
|   Olive Schwartz| 34|
|     Schultz Duke| 23|
|      Mooney Case| 29|
+-----------------+---+
```

SQL-GROUP

```java
public class Sql {
    public static void main(String[] args) {
        SparkSession sparksession = SparkSession.builder().appName("SQL").master("local[*]").getOrCreate();
        Dataset<Row> dataset = sparksession.read().option("multiline", true).json( path: "C://bin//generated.json");

        dataset.createOrReplaceTempView( viewName: "person");
        Dataset<Row> sql = sparksession.sql( sqlText: "select avg(age) from person group by gender");
        sql.show();
    }
}
```

```
21/12/17 16:52:52 INFO CodeGenerator:
+------------------+
|          avg(age)|
+------------------+
|              27.0|
|27.666666666666668|
+------------------+

21/12/17 16:52:52 INFO SparkContext:
```

SQL-GLOBAL TEMP VIEW

```java
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Row;
import org.apache.spark.sql.SparkSession;

public class GlobalView {
    public static void main(String[] args) {
        SparkSession sparksession = SparkSession.builder().appName("SQL").master("local[*]").getOrCreate();
        Dataset<Row> dataset = sparksession.read().option("multiline", true).json( path: "C://bin//generated.json");

        dataset.createOrReplaceGlobalTempView( viewName: "person");
        Dataset<Row> sql = sparksession.sql( sqlText: "SELECT * from person");
        sql.show();
    }
}
```

ENDOCER

```
public class App {
    public static void main(String[] args) {
        SparkConf conf= new SparkConf().setAppName("ABC").setMaster("local[*]").set("spark.ui.port", "8080");;
        JavaSparkContext cont = new JavaSparkContext (conf);
        String path = "C:\\bin\\WorldCup\\WorldCups.csv";
        // read text file to RDD
        JavaRDD<String> Raw_Data = cont.textFile(path);
        System.out.println(Raw_Data.count());

        JavaRDD<CupModel> map = Raw_Data.map(new Function<String, CupModel>() {
            @Override
            public CupModel call(String line) throws Exception {

                String[] split = line.split( regex: ",");
                var cupModel = new CupModel(
                        split[0],
                        split[1],
                        split[2],
                        split[3],
                        split[4],
                        split[5]);
                return cupModel;
            }
        });
```

ENDOCER-PERSON CLASS

```java
public class Person {

    private String name;
    private String email;
    private String gender;

    public Person(String name, String email, String gender) {
        this.name = name;
        this.email = email;
        this.gender = gender;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
```

ENCODER

```
public class Encoder {
    public static void main(String[] args) {
        SparkSession sparksession = SparkSession.builder().appName("SQL").master("local[*]").getOrCreate();

        org.apache.spark.sql.Encoder<Person> PersonEncoder= Encoders.bean(Person.class);
        Dataset<Person> dataset = sparksession.read().option("multiline", true).json( path: "C://bin//generated.json").as(PersonEncoder
        dataset.foreach(new ForeachFunction<Person>() {
            @Override
            public void call(Person person) throws Exception {
                System.out.println(person.getName()+"-"+person.getEmail());
            }
        });

    }
}
```

RDD

- RDD was the primary user-facing API in Spark since its inception.
- At the core, an RDD is an immutable distributed collection of elements of your data, partitioned across nodes in your cluster that can be operated in parallel with a low-level API that offers transformations and actions.

WHEN TO USE RDDS?

- You want low-level transformation and actions and control on your dataset.
- Your data is unstructured, such as media streams or streams of text;
- You don’t care about imposing a schema, such as columnar format, while processing or accessing data attributes by name or column.
- You can forgo some optimization and performance benefits available with DataFrames and Datasets for structured and semi-structured data.

DATAFRAME

- Data Frame is an immutable distributed collection of data.
- Unlike an RDD, data is organized into named columns, like a table in a relational database.
- Designed to make large data sets processing even easier, DataFrame allows developers to impose a structure onto a distributed collection of data, allowing higher-level abstraction;

it provides a domain specific language API to manipulate your distributed data;

and makes Spark accessible to a wider audience, beyond specialized data engineers.

DATASET

- Starting in Spark 2.0, Dataset takes on two distinct APIs characteristics:
  - strongly-typed API and an untyped API,
- Dataset, by contrast, is a collection of strongly-typed JVM objects, dictated by a case class you define in Scala or a class in Java.

WHEN SHOULD I USE DATAFRAMES OR DATASETS?

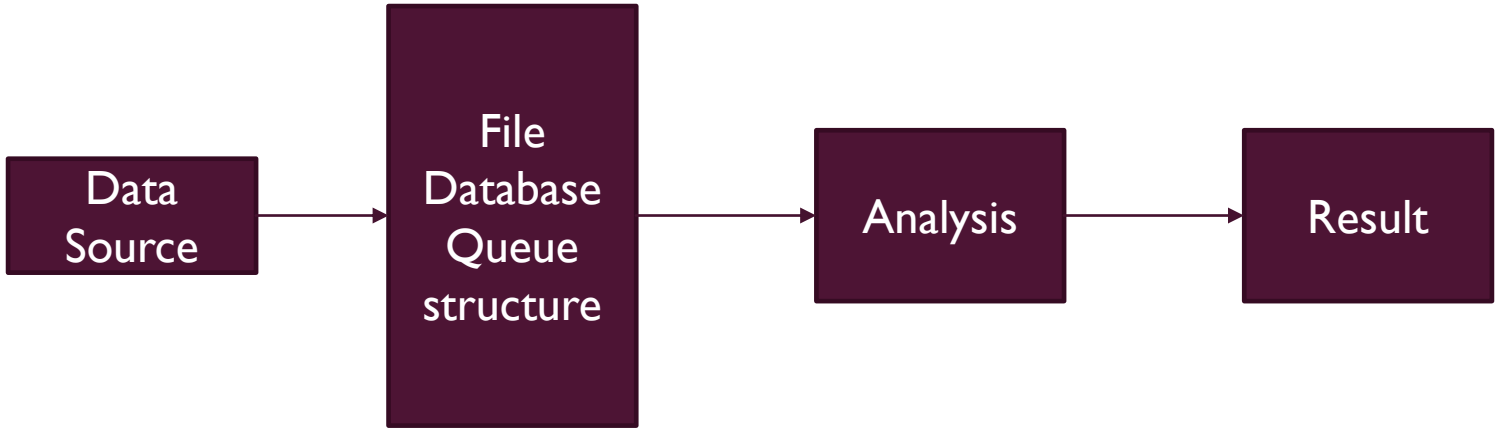
- If your processing demands high-level expressions, filters, maps, aggregation, averages, sum, SQL queries, columnar access and use of lambda functions on semi-structured data, use DataFrame or Dataset.
- If you are a R user, use DataFrames.
- If you are a Python user, use DataFrames and resort back to RDDs if you need more control.

STREAMING

1. Batch processing
2. Real-time processing

BATCH PROCESSING

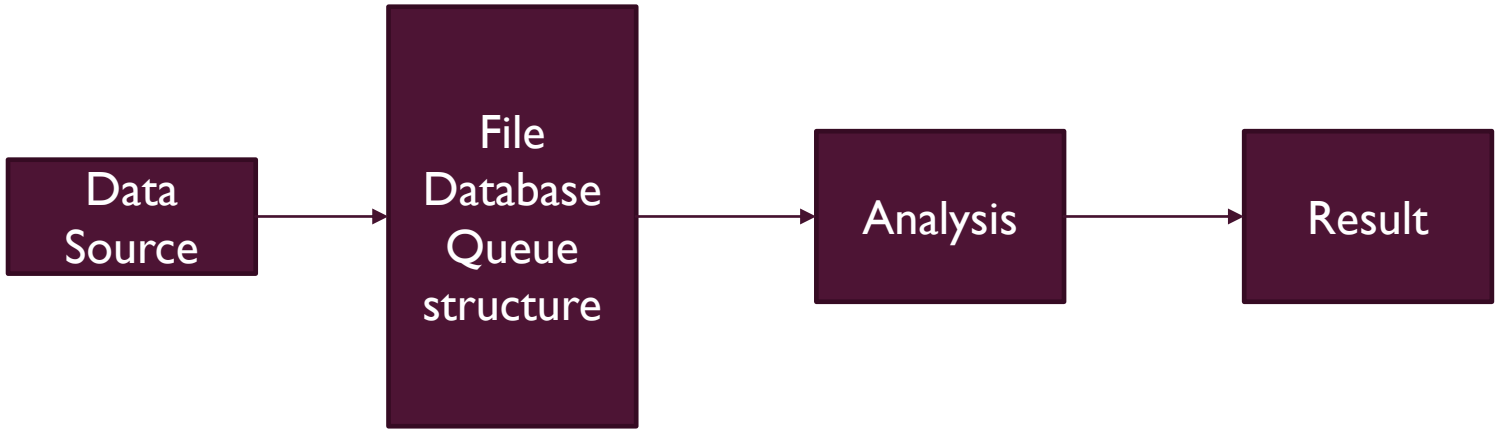
- A sufficient amount of data is collected and analyzes are performed on the collected data.



Usage areas: Basic queries on the database

REAL TIME PROCESSING

- Realization of the analysis instantly while the data is collected.



Usage areas: Analysis of data from the sensor

STREAMING TYPES

- Spark streaming
- Flink
- Apache storm
- Amazon kinesis

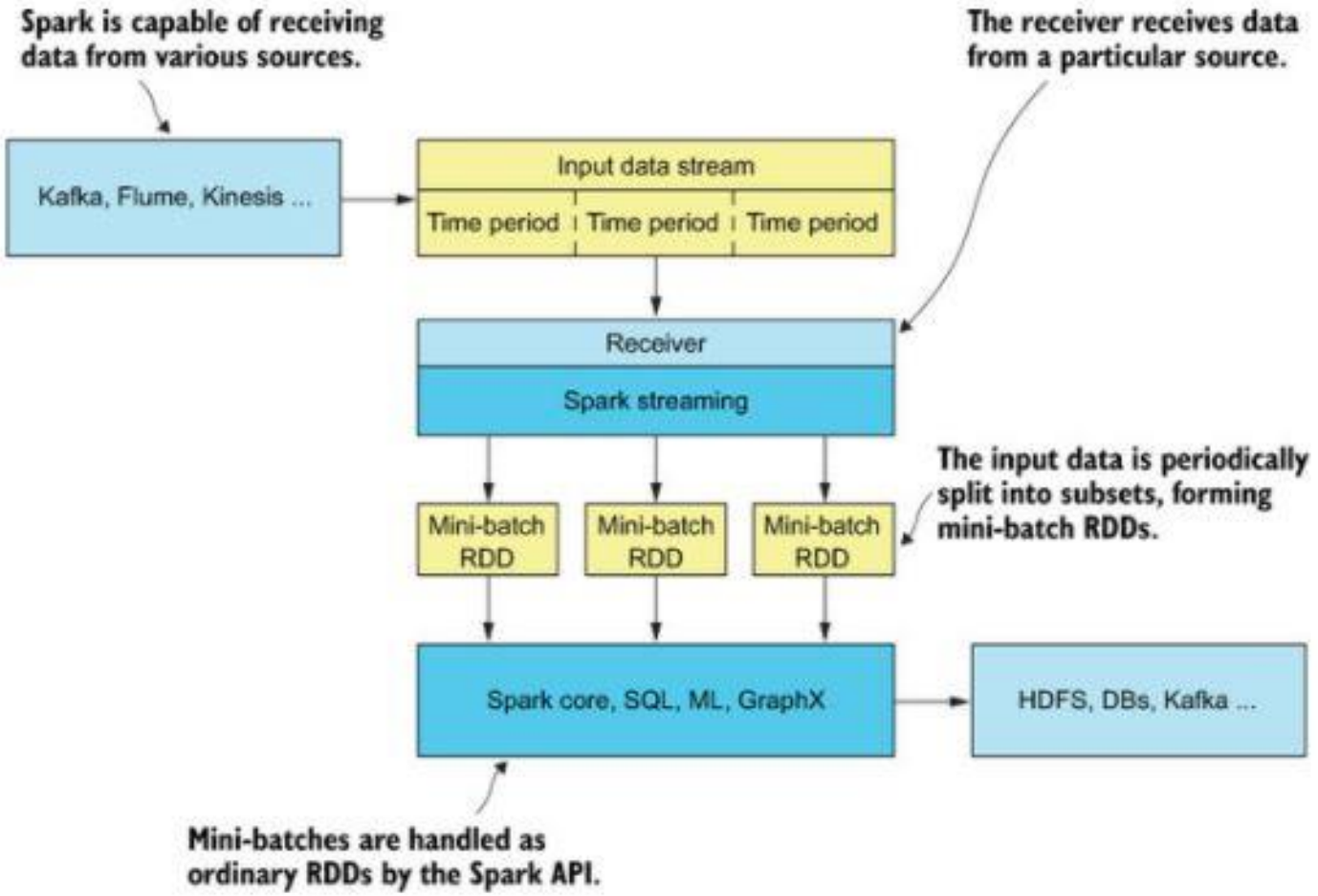
SPARK STREAMING

- It is a subproject of Apache spark project.
- It runs on the Apache spark engine and is a library that allows us to do real-time analysis.
- We can process the streamed data with high-level functions /map, reduce, join) and transfer the processed data to the database.

SPARK STREAMING

- It separates the streaming data into micro batches.
- This micro data batches are processed by Spark engine.
- Finally, the processed data is directed to the output as micro batches.

IOT


Spark is capable of receiving data from various sources.
The receiver receives data from a particular source.
Kafka, Flume, Kinesis ...
Input data stream
Time period
Time period
Time period
Receiver
Spark streaming
Mini-batch RDD
Mini-batch RDD
Mini-batch RDD
The input data is periodically split into subsets, forming mini-batch RDDs.
Spark core, SQL, ML, GraphX
HDFS, DBs, Kafka ...
Mini-batches are handled as ordinary RDDs by the Spark API.

APP

Nasılsın?
Canım sıkkın biraz
Alacağım telefonun fiyatı çok yükselmiş
XX markası
Umarın fiyatı düşer.

Port 8005

Spark Streaming

İyiyim sen?
Hayırdır ne oldu?
Derdin bu olsun?
Hangi marka düşünüyorsun?
Belki ilerde fiyatı düşer.

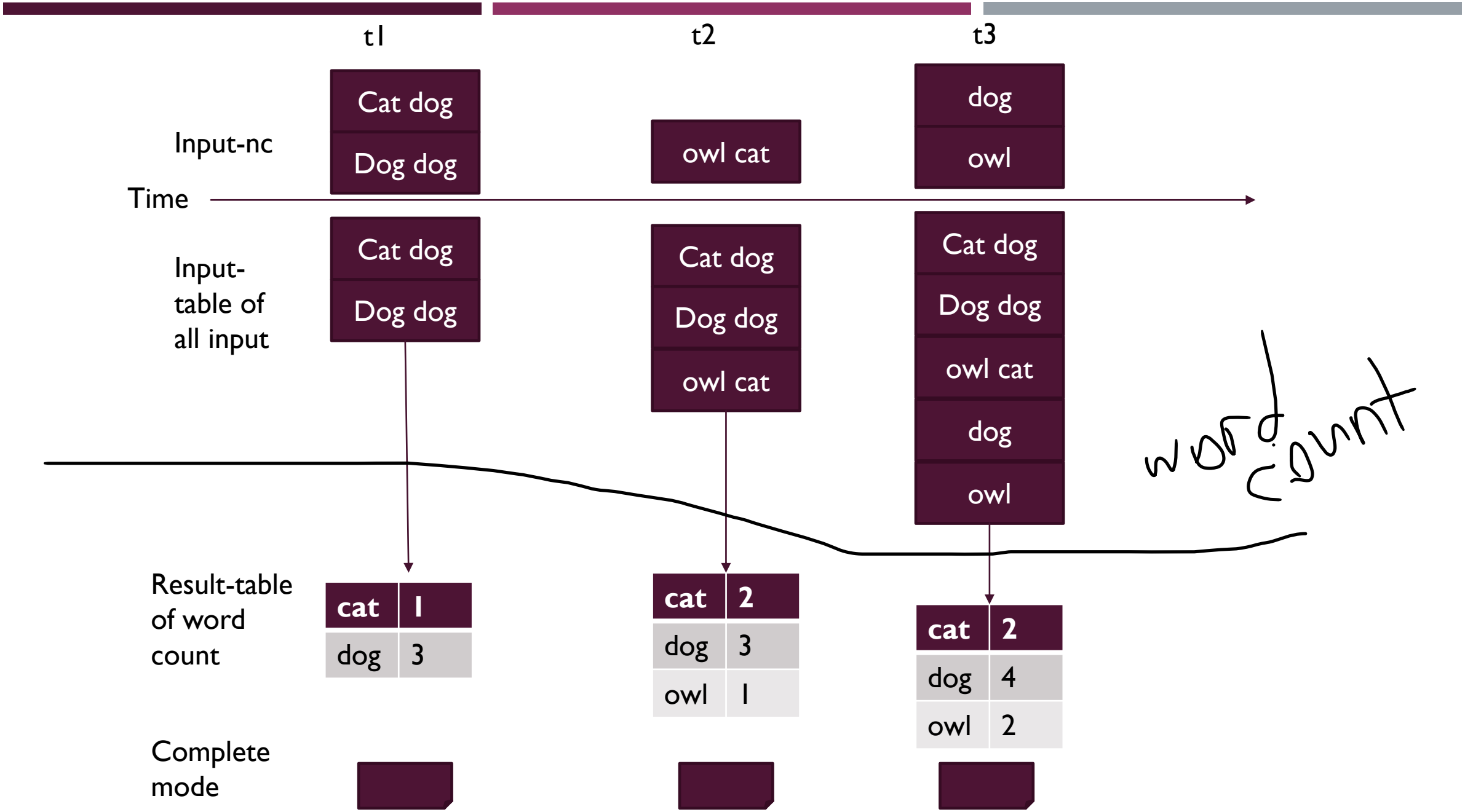
Nasılsın?	I	çok	I
Canım	I	yükselmiş	I
Sıkkın	I	XX	I
biraz	I	markası	I
Alacağım	I	Umarın	I
telefonun	I	fiyatı	I
fiyatı	I	düşer.	I

APP

```java
import org.apache.spark.sql.streaming.StreamingQuery;
import org.apache.spark.sql.streaming.StreamingQueryException;

import java.util.Arrays;
import java.util.Iterator;
import java.util.concurrent.TimeoutException;

public class App {
    public static void main(String[] args) throws StreamingQueryException, TimeoutException {
        SparkSession sparksession = SparkSession.builder().appName("StreamingMessageListener").master("local[*]").getOrCreate();
        Dataset<Row> loaddata = sparksession.readStream().format("socket").option("host", "localhost").option("port", "8005").load();

        Dataset<String> data = loaddata.as(Encoders.STRING());
        Dataset<String> stringDataset = data.flatMap(new FlatMapFunction<String, String>() {
            @Override
            public Iterator<String> call(String s) throws Exception {
                return Arrays.asList(s.split( regex: " ")).iterator();
            }
        }, Encoders.STRING());
        Dataset<Row> value = stringDataset.groupBy( col1: "value").count();
        StreamingQuery start = value.writeStream().outputMode("complete").format("console").start();
        start.awaitTermination();
    }
}
```

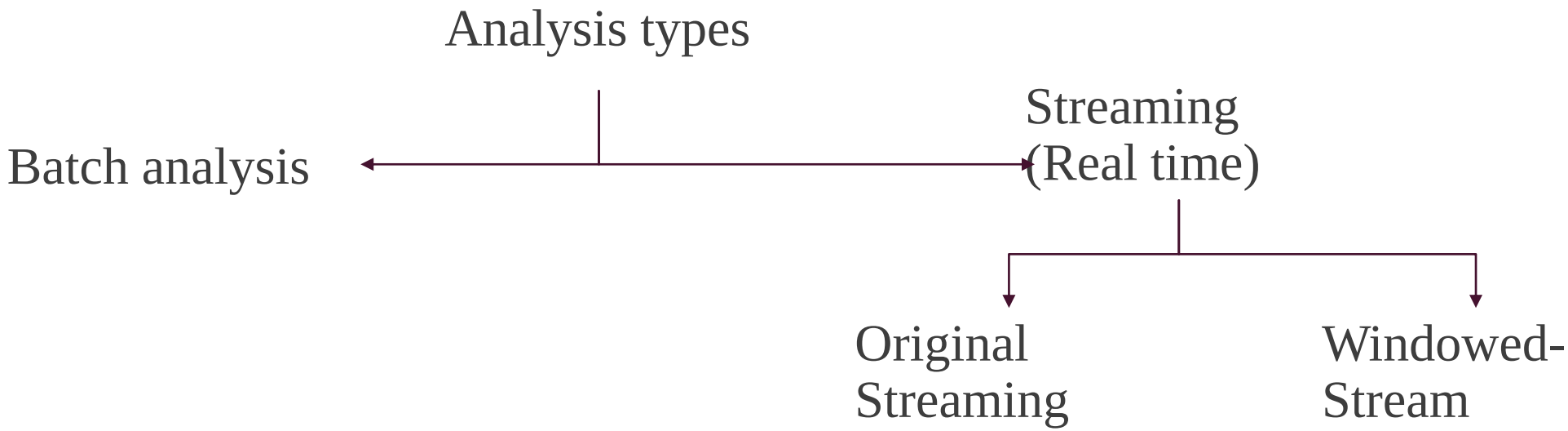
Recommended plugin available for dep
'java:com.google.inject:guice'.

COMPLETE MODE-UPDATE MODE

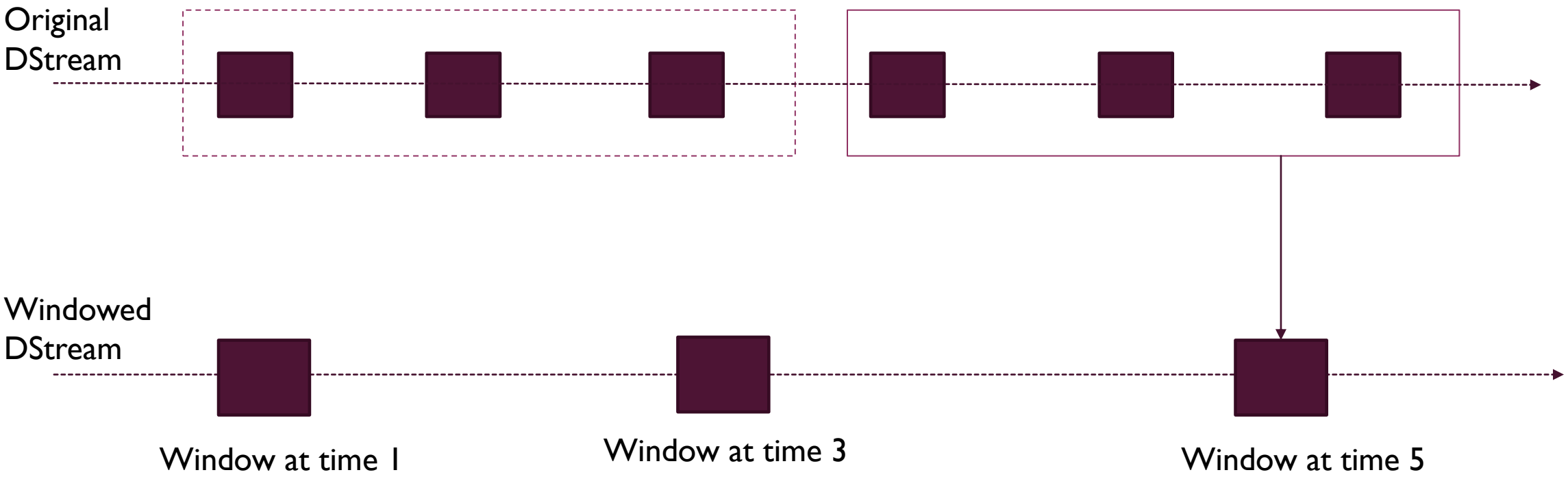

t1
t2
t3
Input-nc
Cat dog
Dog dog
owl cat
dog
owl
Time
Input-
table of
all input
Cat dog
Dog dog
Cat dog
Dog dog
owl cat
Cat dog
Dog dog
owl cat
dog
owl
word count
Result-table
of word
count
cat 1
dog 3
cat 2
dog 3
owl 1
cat 2
dog 4
owl 2
Complete
mode

TIME OPERATIONS

- It is used for data analysis in certain time groups.



TIME OPERATIONS



TIME OPERATIONS

- Number of tweets sent in 5 minutes on Twitter
- The most searched product in the last half hour on the e-commerce site
- How many money transfers and efts were made at which time?

TIME OPERATIONS