



INTERIM PROJECT

STREAMING PROJECT



A Project Report Submitted in the fulfilment of the requirements for
[Interim-Project Evaluation](#)



Submitted by

NEELAM SAI KOUSHIK

EMP ID: 2320381



INTRODUCTION TO SPARK:

Apache Spark, an open-source distributed computing system, stands as a cornerstone in the realm of big data processing and analytics. Its versatility, speed, and scalability have propelled it to the forefront of modern data processing frameworks. Spark's architecture is designed to handle a myriad of data processing tasks, including batch processing, real-time streaming, machine learning, and interactive querying, all within a unified and cohesive framework.

At its core, Spark boasts impressive speed, owing to its innovative approach to in-memory computing and optimization techniques. By leveraging lazy evaluation and pipelining, Spark outperforms traditional MapReduce-based systems, enabling lightning-fast processing of large-scale datasets.

Ease of use is another hallmark of Spark, thanks to its user-friendly APIs available in multiple languages like Scala, Java, Python, and R. These high-level APIs abstract away the complexities of distributed computing, allowing developers to write concise and expressive code for a variety of data processing tasks.

Spark's versatility knows no bounds, with its support for an array of data processing operations. Whether it's batch processing using Spark Core, structured data manipulation with Spark SQL, real-time streaming analytics through Spark Streaming, or machine learning with MLlib, Spark offers a comprehensive solution for diverse data processing needs.

Underpinning Spark's reliability is its fault tolerance mechanism, powered by the resilient distributed dataset (RDD) abstraction. RDDs ensure data integrity and resilience by automatically recovering from node failures and recomputing lost partitions as needed.

Scalability is yet another forte of Spark, as it seamlessly scales from single-node clusters to large-scale distributed environments. Its dynamic task allocation and efficient resource utilization facilitate linear scalability, making it a preferred choice for processing massive datasets.

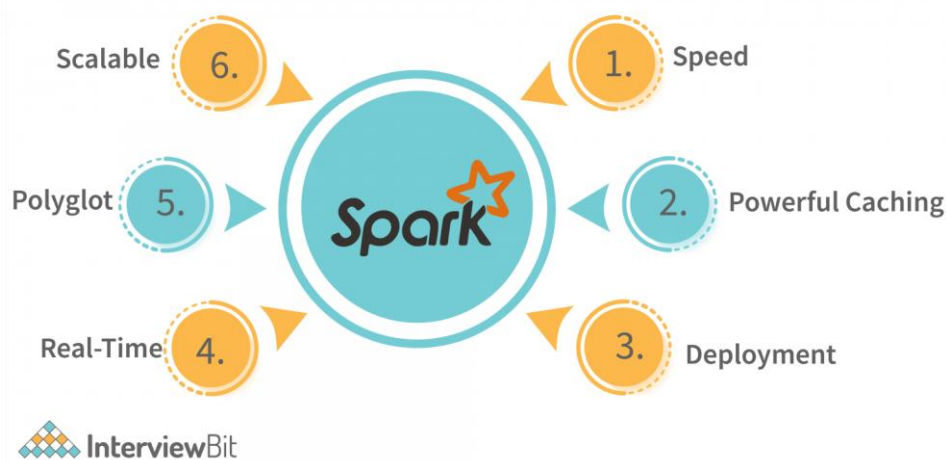
Moreover, Spark integrates seamlessly with other big data technologies such as Hadoop, Apache Hive, Kafka, and more. This interoperability allows users to leverage existing infrastructure and data sources, making Spark a versatile and adaptable solution for various data processing workflows.

In conclusion, Apache Spark's combination of speed, ease of use, versatility, fault tolerance, scalability, and integration capabilities make it a powerhouse in the world of big data processing and analytics. With its rich ecosystem of components and APIs, Spark continues to drive innovation and empower organizations to extract valuable insights from their data at scale.

Apache Spark supports two main types of processing:

- **Batch Processing:** In batch processing, data is collected over a period, processed in fixed-size chunks or batches, and then output is generated after processing the entire batch. This mode of processing is suitable for scenarios where data can be processed offline, and near-real-time processing is not required.
- **Streaming Processing:** Streaming processing, also known as real-time processing or stream processing, deals with processing data in real-time as it arrives. Spark Structured Streaming is a scalable and fault-tolerant stream processing engine built on Spark SQL. It enables continuous processing of data streams with support for windowed computations, event-time processing, and exactly once semantics.

Features of Spark: -



DATA FRAMES:

DataFrames in Apache Spark serve as a fundamental abstraction for processing structured data efficiently and scalably. They represent data as distributed collections of rows organized into named columns, akin to tables in relational databases or spreadsheets.

1. One notable characteristic of DataFrames is their immutability; once created, DataFrames cannot be modified directly, but transformations can be applied to produce new DataFrames.
2. These transformations are lazily evaluated, meaning they are not executed until an action is invoked.
3. DataFrames enforce a schema, specifying the structure of the data including column names and data types.
4. DataFrames support a wide array of high-level operations, including filtering, selection, aggregation, joining, and sorting, which can be expressed using either DataFrame APIs or SQL-like syntax.
5. This flexibility DataFrames in Spark provide a powerful, unified, and flexible framework for working with structured data, facilitating efficient data processing, analysis, and manipulation at scale. Their rich set of features, optimization capabilities, and interoperability make them a cornerstone for developing data-driven applications and solutions in the Spark ecosystem.

INTERM PROJECT DOCUMENTATION

Streaming Data Enrichment: Enrich streaming data with additional information from external sources in real-time, enhancing its value for analysis.

What is Streaming Data Enrichment?

Streaming data enrichment is a process that combines real-time data with external sources to gain a more complete and accurate understanding of operations. By augmenting raw data from one source with related data from a second source, businesses can make more informed decisions and improve overall efficiency. Let's explore this concept further.

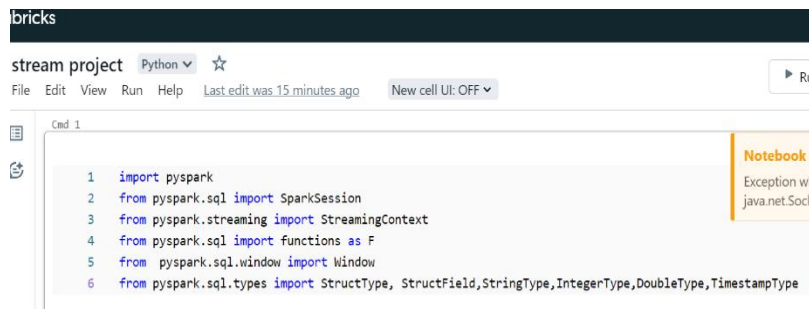
What is data cleansing?

Data cleansing is the process of finding and removing errors, inconsistencies, duplications, and missing entries from data to increase data consistency and quality also known as data scrubbing or cleaning. While organizations can be proactive about data quality in the collection stage, it can still be noisy or dirty.

What operations we will perform for data cleansing:-

Input data:-

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	EMPLOYEE	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	MANAGER_ID	DEPARTMENT	AGE	SPORTS	
2	1	ANJI	LAKKA	ANJANEYA	512.124.51	07-Jun-02	AC_ACCOUNT	8300	205	110	32	CRICKET	
3	2	Steven	King	SKING	515.123.41	17-Jun-03	AD_PRES	24000		90	34	HOCKEY	
4	3	Donald	OConnell	DOCONNEL	650.507.91	21-Jun-07	SH_CLERK	2600	124	50	43	CRICKET	
5	4	Donald	OConnell	DGRANT	650.507.91	13-Jan-08	SH_CLERK	2600	130	50	56	HOCKEY	
6	5	Jennifer	Whalen	JWHALEN	515.123.41	#####	AD_ASST	4400	101	10	33	VOLLEY BALL	
7	4	Jennifer	Whalen	MHARTST	515.123.51	#####	MK_MAN	13000	100	20	28	CHESS	
8	5	Pat	Fay	PFAY	603.123.61	#####	MK_REP	6000	201	20	56	CRICKET	
9	5	Pat	Fay	PFAY	603.123.61	#####	MK_REP	6000	201	20	56	CRICKET	
10	7	Shelley	Higgins	SHIGGINS	515.123.81	07-Jun-02	AC_MGR	12008	101	110	43	CHESS	
11	7	Shelley	Higgins	SHIGGINS	515.123.81	07-Jun-02	AC_MGR	12008	101	110	43	CHESS	
12	20	John	Chen	JCHEN	515.124.41	#####	FI_ACCOUNT	8200	108	100	35	CHESS	
13	21	John	Chen	JCHEN	515.124.41	#####	FI_ACCOUNT	8200	108	100	35	CRICKET	
14	22	Jose Manuel	Urman	JMURMAN	515.124.41	#####	FI_ACCOUNT	7800		100	24	HOCKEY	
15	23	Luis	Popp	LPOPP	515.124.41	#####	FI_ACCOUNT	6900	108	100	567	VOLLEY BALL	
16	24	Den	Raphaely	DRAPHAEL	515.127.41	#####	PU_MAN	11000	100	30	35	CHESS	
17	11	Neena	Kochhar	NKOCCHA	515.123.41	#####	AD_VP	17000	100	90	53	VOLLEY BALL	
18	12	Lex	De Haan	LDEHAAN	515.123.41	13-Jan-01	AD_VP	17000	100	90	42	CHESS	
19	13	Alexander	Hunold	AHUNOLD	590.423.41	03-Jan-06	IT_PROG	9000	102	60	42	CRICKET	
20	14	Bruce	Ernst	BERNST	590.423.41	#####	IT_PROG	6000	103	60	53	HOCKEY	



The screenshot shows a Jupyter Notebook titled 'stream project' in a Python environment. The first code cell, labeled 'Cmd 1', contains the following imports:

```
1 import pyspark
2 from pyspark.sql import SparkSession
3 from pyspark.streaming import StreamingContext
4 from pyspark.sql import functions as F
5 from pyspark.sql.window import Window
6 from pyspark.sql.types import StructType, StructField, StringType, IntegerType, DoubleType, TimestampType
```

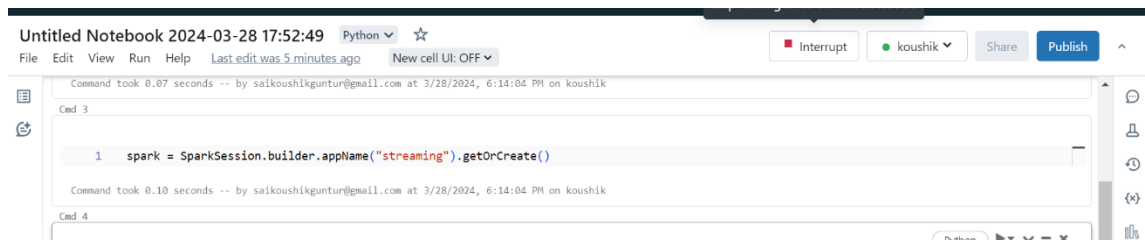
- This line imports the **pyspark** module, which is necessary to work with PySpark functionalities.
- This line imports the **SparkSession** class from the **pyspark.sql** module. SparkSession is the entry point to programming Spark with the DataFrame and SQL API.
- This line imports the **StreamingContext** class from the **pyspark.streaming** module. StreamingContext is the main entry point for creating DStream (discretized stream) objects, which represent continuous streams of data in Spark Streaming.
- This line imports the **functions** module from the **pyspark.sql** package and aliases it as **F**. This module provides functions for working with DataFrames and performing various operations like aggregations, transformations, and more.
- This line imports the **Window** class from the **pyspark.sql.window** module. Window is used for defining window specifications for window functions in PySpark SQL.
- This line imports specific data types (**StructType**, **StructField**, **StringType**, **IntegerType**, **DoubleType**, **TimestampType**) from the **pyspark.sql.types** module. These data types are used for defining the schema of DataFrames in PySpark.



The screenshot shows the second code cell, labeled 'Cmd 2', defining a schema for a DataFrame:

```
1 schema = StructType([
2     StructField("EMPLOYEE_ID", IntegerType(), True),
3     StructField("FIRST_NAME", StringType(), True),
4     StructField("LAST_NAME", StringType(), True),
5     StructField("EMAIL", StringType(), True),
6     StructField("PHONE_NUMBER", StringType(), True),
7     StructField("HIRE_DATE", StringType(), True),
8     StructField("JOB_ID", StringType(), True),
9     StructField("SALARY", IntegerType(), True),
10    StructField("MANAGER_ID", IntegerType(), True),
11    StructField("DEPARTMENT_ID", IntegerType(), True),
12    StructField("AGE", IntegerType(), True),
13    StructField("SPORTS", StringType(), True)
14 ])
```

- StructType is used to define the schema for a DataFrame in PySpark.
- StructField is used to define each column in the DataFrame schema.



- **SparkSession.builder:** SparkSession is the entry point to programming Spark with the DataFrame and SQL API in PySpark.
- **.builder** is a method used to create a Builder instance for configuring and creating a **SparkSession.getOrCreate()** is a method that tries to reuse an existing SparkSession if it already exists or creates a new one if none exists.
- This method is typically used to ensure that there is only one active SparkSession per JVM (Java Virtual Machine).



Create New Table

Data source ⓘ

Upload File S3 DBFS Other Data Sources

DBFS Target Directory ⓘ

/FileStore/tables/ (optional)

Files uploaded to DBFS are accessible by everyone who has access to this workspace. [Learn more](#)

Files ⓘ

employees.csv

2 KB

[Remove file](#)

✓ File uploaded to /FileStore/tables/employees-5.csv

ⓘ

- **spark.readStream:** spark is the SparkSession object you previously created, which serves as the entry point to programming Spark with the DataFrame and SQL API.
- **.readStream** indicates that you are reading streaming data rather than static data. This is crucial for structured streaming in PySpark.

Untitled Notebook 2024-03-28 17:52:49 Python

```
1 df_filtered = streaming_df.filter(streaming_df.MANAGER_ID.isNotNull()).display()
```

Cancel

▶ (1) Spark Jobs

▶ display_query_12 (id: ca5f4702-e156-45fa-8279-941821b5d48d) Last updated: 10 seconds ago

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	MANAGER_ID
1	100	William	Gietz	WGIEZT	515.123.8181	07-Jun-02	AC_ACCOUNT	8300	205
2	112	Donald	OConnell	DOCONNEL	650.507.9833	21-Jun-07	SH_CLERK	2600	124
3	113	Donald	OConnell	DGRANT	650.507.9844	13-Jan-08	SH_CLERK	2600	130
4	114	Jennifer	Whalen	JWHALEN	515.123.4444	17-Sep-03	AD_ASST	4400	101
5	4	Jennifer	Whalen	MHARTSTE	515.123.5555	17-Feb-04	MK_MAN	13000	100
6	5	Pat	Fay	PFAY	603.123.6666	17-Aug-05	MK_REP	6000	201
7	5	Pat	Fav	PFAY	603.123.6666	17-Aug-05	MK_REP	6000	201

105 rows

- `streaming_df`: This is the original DataFrame on which filtering is applied..`filter(...)`: This method is used to filter rows based on a condition specified inside the parentheses.
- `streaming_df.MANAGER_ID.isNotNull()`: This condition checks if the `MANAGER_ID` column in `streaming_df` is not null for each row.
- `.display()`: This method displays the resulting DataFrame after applying the filter.

```
1 df_distinct=df_filtered.select("EMPLOYEE_ID","FIRST_NAME","LAST_NAME","EMAIL","SPORTS","AGE").distinct()
```

df_distinct: pyspark.sql.dataframe.DataFrame = [EMPLOYEE_ID: integer, FIRST_NAME: string ... 4 more fields]

Command took 0.19 seconds -- by saikoushikguntur@gmail.com at 3/28/2024, 6:14:04 PM on koushik

- `.select(...)`: This method selects specific columns from the DataFrame to be included in the resulting DataFrame.
- `"EMPLOYEE_ID", "FIRST_NAME", "LAST_NAME", "EMAIL", "SPORTS", "AGE"`: These are the column names selected from `df_filtered` to be included in the `df_distinct` DataFrame.
- `.distinct()`: This method removes duplicate rows from the DataFrame, so `df_distinct` will contain only unique combinations of the selected columns.

Untitled Notebook 2024-03-28 17:52:49 Python

```
1 df_age=df_distinct.filter("int(Age)<60")
```

▶ (1) Spark Jobs

▶ display_query_13 (id: ecb630d5-0a7e-4aee-ba33-3598e0e26a48) Last updated: 1 minute ago

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	SPORTS	AGE
1	100	William	Gietz	WGIEZT	CRICKET	32
2	13	Alexander	Hunold	AHUNOLD	CRICKET	42
3	20	John	Chen	JCHEN	CHESS	35
4	5	Jennifer	Whalen	JWHALEN	VOLLEY BALL	33
5	5	Pat	Fay	PFAY	CRICKET	56
6	114	Jennifer	Whalen	JWHALEN	VOLLEY BALL	33
7	7	Shellev	Hiaains	SHIGGINS	CHESS	43

25 rows

- `df_distinct`: This is the DataFrame containing distinct rows based on the selected columns from the previous step.
- `.filter(...)`: This method is used to filter rows based on a specified condition.
- `'int(Age)<60'`: This is the condition used for filtering. It checks if the integer value of the Age column is less than 60 for each row. The `'int(...)'` function converts the value in the Age column to an integer before applying the comparison.

```
1 df_no_duplicates = df_distinct.dropDuplicates(['FIRST_NAME'])
```

- `df_distinct`: This is presumably a DataFrame object containing data, possibly with duplicate entries.
- `dropDuplicates(['FIRST_NAME'])`: This method is applied to the DataFrame `df_distinct`. It removes duplicate rows based on the specified column(s). In this case, it removes rows where the values in the column 'FIRST_NAME' are duplicates, keeping only the first occurrence of each unique 'FIRST_NAME' value.
- `df_no_duplicates`: This is the new DataFrame created after removing the duplicate rows based on the 'FIRST_NAME' column. It contains the same columns as `df_distinct`, but with duplicate rows removed based on the specified column criteria.

```
1 window_spec = Window.orderBy("EMPLOYEE_ID").rowsBetween(Window.unboundedPreceding, Window.unboundedFollowing)
2 display(window_spec)
```

<pyspark.sql.window.WindowSpec at 0x7fc1c464be58>

1. **`Window.orderBy("EMPLOYEE_ID")`**: This part specifies the ordering of rows within the window based on the 'EMPLOYEE_ID' column. The rows will be sorted in ascending order of 'EMPLOYEE_ID'.
2. **`.rowsBetween(Window.unboundedPreceding, Window.unboundedFollowing)`**: This part defines the frame specification for the window.
3. **`Window.unboundedPreceding`**: Refers to the starting point of the window frame, which is set to unbounded preceding, meaning it includes all rows from the beginning of the window partition up to the current row.
4. **`Window.unboundedFollowing`**: Refers to the ending point of the window frame, which is set to unbounded following, meaning it includes all rows from the current row up to the end of the window partition.


```

1 query = df_no_duplicates.writeStream \
2   .format("console").outputMode("update").trigger(processingTime="10 seconds").start()
3 display(df_no_duplicates)

```

▶ (2) Spark Jobs

- 79d67804-71e6-4c30-9714-3f3fd11e3348 Last updated: 15 seconds ago
- display_query_14 (id: e9d30a96-729f-4161-b39f-3e5d5f265fb6) Last updated: 0 seconds ago

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	SPORTS	AGE
1	11	Neena	Kochhar	NKOCHHAR	VOLLEY BALL	53
2	7	Shelley	Higgins	SHIGGINS	CHESS	43
3	5	Jennifer	Whalen	JWHALEN	VOLLEY BALL	33
4	5	Pat	Fay	PFAY	CRICKET	56
5	4	Donald	OConnell	DGRANT	HOCKEY	56
6	20	John	Chen	JCHEN	CHESS	35
7	12	Lex	De Haan	LDEHAAN	CHESS	42

- `query = df_no_duplicates.writeStream \`: This line begins the definition of a streaming query based on the DataFrame `df_no_duplicates` using the `writeStream` method.
- `.format("console")`: This specifies the output sink format for the streaming query. In this case, it's set to output data to the console for debugging or monitoring purposes.
- `.outputMode("update")`: This defines the output mode for the streaming query. Here, it's set to "update" mode, which means only the updated rows in the streaming DataFrame will be written to the output sink.
- `.trigger(processingTime="10 seconds")`: This sets the trigger for the streaming query. It specifies that the query should process the data and write to the output sink every 10 seconds.
- `.start()`: This method starts the streaming query execution.
- `display(df_no_duplicates)`: This line is not directly related to the streaming query setup but is used to display the contents of the DataFrame `df_no_duplicates` in a notebook or interactive environment. Note that `display()` is typically used in environments like Databricks for visualization purposes.

Databricks

Untitled Notebook 2024-03-28 17:52:49 Python

File Edit View Run Help Last edit was 11 minutes ago New cell UI: OFF

Interrupt koushik Share

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	SPORTS	AGE
1	11	Neena	Kochhar	NKOCHHAR	VOLLEY BALL	53
2	7	Shelley	Higgins	SHIGGINS	CHESS	43
3	5	Jennifer	Whalen	JWHALEN	VOLLEY BALL	33
4	5	Pat	Fay	PFAY	CRICKET	56
5	4	Donald	OConnell	DGRANT	HOCKEY	56
6	20	John	Chen	JCHEN	CHESS	35
7	12	Lex	De Haan	LDEHAAN	CHESS	42

16 rows

```

1 query.awaitTermination(10000)
2 query.stop()

```

Cmd 11

Python

Running command...

`query.awaitTermination(timeout=10000)`: This line waits for the streaming query represented by the query object to terminate, with a timeout of 100,00 milliseconds (100 seconds). During this waiting period, the program will be blocked until either the query terminates, or the timeout occurs.