In order to complete the coding assignment, you will need to register for an account on Databricks community edition, which you can do here:

[Community Edition Databricks](https://databricks.com/try-databricks)

Upon completion of the assignment, please share the end result notebooks with us by submitting the public links for notebook via the contact email address provided. If you have any questions when completing the assignment, please feel free to reach out to your point of contact(cc’ed in this email).

Most candidates spend between half-a-day (proficient Spark users) to 2 days on the assignment. We understand it can be difficult to find time during the work week. Please be sure to complete it no later than a week from today (9/3/2022). If that timing will be problematic, please let us know.

Finally, for the integrity of the process, please do not distribute or publicize this assignment.

**Coding Assignment**

**Baby Names Dataset**

This dataset comes from a website referenced by [Data.gov](http://catalog.data.gov/dataset/baby-names-beginning-2007). It lists baby names used in the state of NY from 2007 to 2012.

The following cells run commands that copy this file to the cluster.

**import** java.net.URL

**import** java.io.File

**import** org.apache.commons.io.FileUtils

**val** tmpFile = **new** File("/tmp/rows.json")

FileUtils.copyURLToFile(**new** URL("<https://health.data.ny.gov/api/views/jxy9-yhdk/rows.json?accessType=DOWNLOAD>"), tmpFile)

#### Question #1: Spark SQL's Native JSON Support

Use Spark SQL's native JSON support to create a temp table you can use to query the data (you'll use the registerTempTable operation). Show a simple sample query

#### Question #2: Working with Nested Data

What does the nested schema of this dataset look like? How can you bring these nested fields up to the top level in a DataFrame?

#### Question #3: Executing Full Data Pipelines

Create a second version of the answer to Question 2, and make sure one of your queries makes the original web call every time a query is run, while another version only executes the web call one time.

#### Question #4: Analyzing the Data

Using the tables you created, create a simple visualization that shows what is the most popular first letters baby names to start with in each year.

### Log Processing

Fetch some log files and load them to the notebook

#### Question #1: Parsing Logs

Parse the logs in to a DataFrame/Spark SQL table that can be queried. This should be done using the Dataset API.

#### Question #2: Analysis

Generate some insights from the log data.

### CSV Parsing

The following examples involve working with simple CSV data

#### Question #1: CSV Header Rows

Given the simple RDD full\_csv below, write the most efficient Spark job you can to remove the header row

**val** full\_csv = sc.parallelize(Array(

"col\_1, col\_2, col\_3",

"1, ABC, Foo1",

"2, ABCD, Foo2",

"3, ABCDE, Foo3",

"4, ABCDEF, Foo4",

"5, DEF, Foo5",

"6, DEFGHI, Foo6",

"7, GHI, Foo7",

"8, GHIJKL, Foo8",

"9, JKLMNO, Foo9",

"10, MNO, Foo10"))

#### Question #2: SparkSQL Dataframes

Using the full\_csv RDD above, write code that results in a DataFrame where the schema was created programmatically based on the heard row. Create a second RDD similair to full\_csv and uses the same function(s) you created in this step to make a Dataframe for it.

#### Question #3: Parsing Pairs

Write a Spark job that processes comma-separated lines that look like the below example to pull out Key Value pairs.

Given the following data:

Row-Key-001, K1, 10, A2, 20, K3, 30, B4, 42, K5, 19, C20, 20

Row-Key-002, X1, 20, Y6, 10, Z15, 35, X16, 42

Row-Key-003, L4, 30, M10, 5, N12, 38, O14, 41, P13, 8

You'll want to create an RDD that contains the following data:

Row-Key-001, K1

Row-Key-001, A2

Row-Key-001, K3

Row-Key-001, B4

Row-Key-001, K5

Row-Key-001, C20

Row-Key-002, X1

Row-Key-002, Y6

Row-Key-002, Z15

Row-Key-002, X16

Row-Key-003, L4

Row-Key-003, M10

Row-Key-003, N12

Row-Key-003, O14

Row-Key-003, P13

#### Question #4 Create Tables Programmatically And Cache The Table

Create a table using Scala or Python

* Use CREATE EXTERNAL TABLE in SQL, or DataFrame.saveAsTable() in Scala or Python, to register tables.
* Please refer to the Accessing Data guide for how to import specific data types.

### Temporary Tables

* Within each Spark cluster, temporary tables registered in the sqlContext with DataFrame.registerTempTable will also be shared across the notebooks attached to that Databricks cluster.
  + Run someDataFrame.registerTempTable(TEMP\_TABLE\_NAME) to give register a table.
* These tables will not be visible in the left-hand menu, but can be accessed by name in SQL and DataFrames.

### Others

1. Given a data set in raw text storage format. What other data storage format can you suggest optimizing the performance of a Spark workload if we were to frequently scan and read this dataset. Explain why you decide to go with this approach?
2. You are given to design a streaming data pipeline on AWS, consuming the data in the form of JSON events from an external producer. The customers consuming this data from the AWS endpoint you provide has a requirement that this data set be structured/normalised and easier to use. What are your tools of choice to design this streaming pipeline, how will you ensure the costs don’t surge, and how will you fulfil the customer’s request?

community edition, which you can do here:

[Community Edition Databricks](https://databricks.com/try-databricks)

Upon completion of the assignment, please share the end result notebooks with us by submitting the public links for notebook via the contact email address provided. If you have any questions when completing the assignment, please feel free to reach out to your point of contact(cc’ed in this email).

Most candidates spend between half-a-day (proficient Spark users) to 2 days on the assignment. We understand it can be difficult to find time during the work week. Please be sure to complete it no later than a week from today (9/3/2022). If that timing will be problematic, please let us know.

Finally, for the integrity of the process, please do not distribute or publicize this assignment.

**Coding Assignment**

**Baby Names Dataset**

This dataset comes from a website referenced by [Data.gov](http://catalog.data.gov/dataset/baby-names-beginning-2007). It lists baby names used in the state of NY from 2007 to 2012.

The following cells run commands that copy this file to the cluster.

**import** java.net.URL

**import** java.io.File

**import** org.apache.commons.io.FileUtils

**val** tmpFile = **new** File("/tmp/rows.json")

FileUtils.copyURLToFile(**new** URL("<https://health.data.ny.gov/api/views/jxy9-yhdk/rows.json?accessType=DOWNLOAD>"), tmpFile)

#### Question #1: Spark SQL's Native JSON Support

Use Spark SQL's native JSON support to create a temp table you can use to query the data (you'll use the registerTempTable operation). Show a simple sample query

#### Question #2: Working with Nested Data

What does the nested schema of this dataset look like? How can you bring these nested fields up to the top level in a DataFrame?

#### Question #3: Executing Full Data Pipelines

Create a second version of the answer to Question 2, and make sure one of your queries makes the original web call every time a query is run, while another version only executes the web call one time.

#### Question #4: Analyzing the Data

Using the tables you created, create a simple visualization that shows what is the most popular first letters baby names to start with in each year.

### Log Processing

Fetch some log files and load them to the notebook

#### Question #1: Parsing Logs

Parse the logs in to a DataFrame/Spark SQL table that can be queried. This should be done using the Dataset API.

#### Question #2: Analysis

Generate some insights from the log data.

### CSV Parsing

The following examples involve working with simple CSV data

#### Question #1: CSV Header Rows

Given the simple RDD full\_csv below, write the most efficient Spark job you can to remove the header row

**val** full\_csv = sc.parallelize(Array(

"col\_1, col\_2, col\_3",

"1, ABC, Foo1",

"2, ABCD, Foo2",

"3, ABCDE, Foo3",

"4, ABCDEF, Foo4",

"5, DEF, Foo5",

"6, DEFGHI, Foo6",

"7, GHI, Foo7",

"8, GHIJKL, Foo8",

"9, JKLMNO, Foo9",

"10, MNO, Foo10"))

#### Question #2: SparkSQL Dataframes

Using the full\_csv RDD above, write code that results in a DataFrame where the schema was created programmatically based on the heard row. Create a second RDD similair to full\_csv and uses the same function(s) you created in this step to make a Dataframe for it.

#### Question #3: Parsing Pairs

Write a Spark job that processes comma-separated lines that look like the below example to pull out Key Value pairs.

Given the following data:

Row-Key-001, K1, 10, A2, 20, K3, 30, B4, 42, K5, 19, C20, 20

Row-Key-002, X1, 20, Y6, 10, Z15, 35, X16, 42

Row-Key-003, L4, 30, M10, 5, N12, 38, O14, 41, P13, 8

You'll want to create an RDD that contains the following data:

Row-Key-001, K1

Row-Key-001, A2

Row-Key-001, K3

Row-Key-001, B4

Row-Key-001, K5

Row-Key-001, C20

Row-Key-002, X1

Row-Key-002, Y6

Row-Key-002, Z15

Row-Key-002, X16

Row-Key-003, L4

Row-Key-003, M10

Row-Key-003, N12

Row-Key-003, O14

Row-Key-003, P13

#### Question #4 Create Tables Programmatically And Cache The Table

Create a table using Scala or Python

* Use CREATE EXTERNAL TABLE in SQL, or DataFrame.saveAsTable() in Scala or Python, to register tables.
* Please refer to the Accessing Data guide for how to import specific data types.

### Temporary Tables

* Within each Spark cluster, temporary tables registered in the sqlContext with DataFrame.registerTempTable will also be shared across the notebooks attached to that Databricks cluster.
  + Run someDataFrame.registerTempTable(TEMP\_TABLE\_NAME) to give register a table.
* These tables will not be visible in the left-hand menu, but can be accessed by name in SQL and DataFrames.

### Others

1. Given a data set in raw text storage format. What other data storage format can you suggest optimizing the performance of a Spark workload if we were to frequently scan and read this dataset. Explain why you decide to go with this approach?
2. You are given to design a streaming data pipeline on AWS, consuming the data in the form of JSON events from an external producer. The customers consuming this data from the AWS endpoint you provide has a requirement that this data set be structured/normalised and easier to use. What are your tools of choice to design this streaming pipeline, how will you ensure the costs don’t surge, and how will you fulfil the customer’s request?