SPARK&SCALA

NAME : P. SANJANA REDDY

EMP ID:MT4020

Table of Content

USER STORY 1

Title - JSON PARSING using Circe (https://circe.github.io/circe/) and case classes

Description

Parse the below given jsons using scala using circe library and case classes

a.


```
{
  "profilingJson": [
    {
      "table_config_map": {
        "process_id": 16303275621155,
        "dataplace_id": 258,
        "schema_id": 54,
        "table_id": 250471,
        "datastore_id": 115,
        "datastore_table_id": 250471,
        "table_name": "myFile_pipe.csv",
        "estimated_count": 0,
        "source_type": "gcs",
        "profling_table_name": "myFile_pipe.csv",
        "source_credential": {
```

"credentialId": 152

"credentialTypeId": 3

},

"columns_config": {

"id": {

"column_comment": "",

"column_id": 554,

"dataType": "INT"

},

"firstname": {

"column_comment": "",

"column_id": 553,

"dataType": "STRING"

},

"lastname": {

"column_comment": "",

"column_id": 555,

"dataType": } }

}],

"authInfo": {

"authorizationToken":
"eyJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJGSVJFU0hPVFNfQVBQTElDQVRJT04iLCJzZXNzaW9uX2lkIjoy

NTcwODEwNjYsImV4cCI6MTYzMDM0OTMyNCwidXNlcklkIjoiYm90c19zdmNfYWNjIiwiaWF0Ijox
NjMwMzI3NzI0fQ.CRtShmE9-
jFvsySHOx92rSR7Y2i2a1MDm35URGeEGokNsM5gcNHVugt1ywDKHdiRKmvUzsx24DgE3pF-
B8gLV5GOzjdG0wTLvLd-mhIT75FK4qirflXbE-
xXNz3H2XyleM1EPxXSOPeupISpRqlV_4BVH_gmNVL6maYJLnTuwWjvSjYQoQxXYGEehimNGaXCA
_sAOqkoTZy2nL8DcUUeZXsPUwMFf8ypRj3OpUcmu4Xxg9XufftY1MQQqpE0IrD3mnVfgiIJMyRjC
9wESi2hZkqTkmlRG57-O9wEaGG86TEgtZQSf7xXUz9r4aKYvtDf_sWNWJzRQUnja4XalJHd9A",

"endPoint": "https://localhost:8061/fireshots/credentialsFetchWebService"

}

}

*** source_credential can be null i.e check the below json

b.

{

"profilingJson": [

{

"table_config_map": {

"process_id": 16303275621155,

"dataplace_id": 258,

"schema_id": 54,

"table_id": 250471,

"datastore_id": 115,

"datastore_table_id": 250471,

"table_name": "myFile_pipe.csv",

```
  "estimated_count": 0,
  "source_type": "gcs",
  "profling_table_name": "myFile_pipe.csv",
  "source_credential": null,
  "columns_config": {
    "id": {
      "column_comment": "",
      "column_id": 554,
      "dataType": "INT"
    },
    "firstname": {
      "column_comment": "",
      "column_id": 553,
      "dataType": "STRING"
    },
    "lastname": {
      "column_comment": "",
      "column_id": 555,
      "dataType": "STRING"
    }
  }
}
```

}

],

"authInfo": {

"authorizationToken":
"eyJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJGSVJFU0hPVFNfQVBQTElDQVRJT04iLCJzZXNzaW9uX2lkIjoyNTcwODEwNjYsImV4cCI6MTYzMDM0OTMyNCwidXNlcklkIjoiYm90c19zdmNfYWNjIiwiaWF0IjoxNjMwMzI3NzI0fQ.CRtShmE9-jFvsySHOx92rSR7Y2i2a1MDm35URGeEGokNsM5gcNHVugt1ywDKHdiRKmvUzsx24DgE3pF-B8gLV5GOzjdG0wTLvLd-mhIT75FK4qirflXbE-xXNz3H2XyleM1EPxXSOPeupISpRqlV_4BVH_gmNVL6maYJLnTuwWjvSjYQoQxXYGEehimNGaXCA_sAOqkoTZy2nL8DcUUeZXsPUwMFf8ypRj3OpUcmu4Xxg9XufftY1MQQqpE0IrD3mnVfgiIJMyRjC9wESi2hZkqTkmlRG57-O9wEaGG86TEgtZQSf7xXUz9r4aKYvtDf_sWNWJzRQUnja4XalJHd9A",

"endPoint": "https://localhost:8061/fireshots/credentialsFetchWebService"

}

}

Acceptance Criteria

*Need to parse the above 2 jsons using case classes in scala.

*case classes will be same for the 2 jsons provided above.

*Need to use the appropriate datatypes for each field in the json.

After json parsing , print the following

1.print authInfo

2.print datastore_id and profling_table_name

3.Print the columns_config in the following order

column_name column_comment column_id dataType

Example :


```
  "columns_config": {
        "id": {
          "column_comment": "",
          "column_id": 554,
          "dataType": "INT"
        },
        "firstname": {
          "column_comment": "",
          "column_id": 553,
          "dataType": "STRING"
        },
        "lastname": {
          "column_comment": "",
          "column_id": 555,
          "dataType": "STRING"
        }
      }
```


for the above columns_config , output will be like below :

id "" 554 INT

firstname "" 553 STRING

lastname "" 555 STRING

Column config can have more columns as well.

Code Explanation:

build.sbt :

```
ThisBuild / name := "test"
ThisBuild / organization := "com.modak"
lazy val scala211 = "2.11.12"
lazy val scala212 = "2.12.15"
crossScalaVersions := Seq(scala211, scala212)
ThisBuild / scalaVersion := scala212
val sparkVersion = "2.4.7"
val circeVersion = "0.12.0-M3"
libraryDependencies ++= Seq(
  "org.apache.spark" %% "spark-core" % sparkVersion % "provided",
  "org.apache.spark" %% "spark-sql" % sparkVersion % "provided",
  "org.apache.spark" %% "spark-hive" % sparkVersion % "provided",
  "io.circe" %% "circe-core" % circeVersion,
  "io.circe" %% "circe-generic" % circeVersion,
  "io.circe" %% "circe-parser" % circeVersion,
  "com.typesafe.scala-logging" %% "scala-logging" % "3.9.2",
  "org.scalatest" %% "scalatest" % "3.0.5" % "test",
  "com.github.music-of-the-ainur" %% "almaren-framework" % "0.9.3-2.4" % "provided",
  "org.slf4j" % "slf4j-api" % "1.7.36",
  "org.slf4j" % "slf4j-simple" % "1.7.36"
)
assemblyMergeStrategy in assembly := {
  case PathList("META-INF", xs@_*) => MergeStrategy.discard
  case x => MergeStrategy.first
```

Explanation:

- **Syntax to add dependencies:** libraryDependencies ++= Seq(groupID%%artifactID%version)
- **"provided"** indicates that "it expects respective dependency to be mentioned at the runtime".
- Circe is a Scala library that simplifies working with JSON, allowing us to easily decode a JSON string into a Scala object or convert a Scala object to JSON.
- Three dependencies were added to install this Circe library.
- Logging is used to maintain the logs of an application. It is very much required to monitor our application.
- Dependencies were added for logging the output.
- Simple Logging Facade for Java (abbreviated SLF4J) acts as a facade for different logging frameworks (e.g., java.util.logging, logback, Log4j). It offers a generic API, making the logging independent of the actual implementation
- "org.scalatest" %% "scalatest" % "3.0.5" % "test"
    - **Scalatest** is a testing library to test Scala code by running the test cases.
    - **"test"** indicates that the dependency is limited to the test class itself.

Explanation :

- Here Scala versions can be 2.11.12 and 2.12.15 , by default the Scala version is 2.12.15.
- ++2.11.12 changes the Scala version from 2.12 to 2.11.12

```
sbt:jsonparsing> ++2.11.12
[info] Setting Scala version to 2.11.12 on 1 projects.
[info] Reapplying settings...
[info] set current project to jsonparsing (in build file:/mnt/c/WorkArea/Spark/JsonParsing/)
sbt:jsonparsing>
```

- Spark version 2.4.7 and Circe version 0.12.0-M3 was declared and this variable used in adding dependencies.

Plugins.sbt :

addSbtPlugin("com.eed3si9n" % "sbt-assembly" % "0.14.10")

Explanantion :

- The sbt-assembly plugin is an SBT plugin for building a single independent fat JAR file with all dependencies included.

Test class :

```
package modak

import com.modak.Main
import org.scalatest._

class Test extends FunSuite with BeforeAndAfter {

  //Json String to parse
  val plainString = """ {
                |  "profilingJson": [
                |    {
                |      "table_config_map": {
                |        "process_id": 16303275621155,
                |        "dataplace_id": 258,
                |        "schema_id": 54,
                |        "table_id": 250471,
                |        "datastore_id": 115,
                |        "datastore_table_id": 250471,
                |        "table_name": "myFile_pipe.csv",
                |        "estimated_count": 0,
                |        "source_type": "gcs",
                |        "profling_table_name": "myFile_pipe.csv",
                |        "source_credential": {
```

| "credentialId": 152,
| "credentialTypeId": 3
| },
| "columns_config": {
| "id": {
| "column_comment": "",
| "column_id": 554,
| "dataType": "INT"
| },
| "firstname": {
| "column_comment": "",
| "column_id": 553,
| "dataType": "STRING"
| },
| "lastname": {
| "column_comment": "",
| "column_id": 555,
| "dataType": "STRING"
| }
| }
| }
| }
|],
| "authInfo": {
| "authorizationToken":
"eyJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJGSVJFU0hPVFNfQVBQTElDQVRJT04iLCJzZXNzaW9uX2lkIjoy
NTcwODEwNjYsImV4cCI6MTYzMDM0OTMyNCwidXNlcklkIjoiYm90c19zdmNfYWNjIiwiaWF0Ijox
NjMwMzI3NzI0fQ.CRtShmE9-
jFvsySHOx92rSR7Y2i2a1MDm35URGeEGokNsM5gcNHVugt1ywDKHdiRKmvUzsx24DgE3pF-

B8gLV5GOzjdG0wTLvLd-mhIT75FK4qirflXbE-
xXNz3H2XyleM1EPxXSOPeupISpRqlV_4BVH_gmNVL6maYJLnTuwWjvSjYQoQxXYGEehimNGaXCA
_sAOqkoTZy2nL8DcUUeZXsPUwMFf8ypRj3OpUcmu4Xxg9XufftY1MQQqpE0IrD3mnVfgiIJMyRjC
9wESi2hZkqTkmlRG57-O9wEaGG86TEgtZQSf7xXUz9r4aKYvtDf_sWNWJzRQUnja4XalJHd9A",
| "endPoint": "https://localhost:8061/fireshots/credentialsFetchWebService"
| }
|}
| """.stripMargin

```
  //encoding json String
  val encodedString = java.util.Base64.getEncoder.encodeToString(plainString.getBytes())
  //calling main method  which is in  Main class
  Main.main(Array[String](encodedString))

}
}
```

Explanation :

- Base64 is a binary-to-text encoding scheme that represents binary data in a printable ASCII string format.
- getEncoder : It returns a Base64.Encoder that encodes using the Basic type base64 encoding scheme.
- Basic: This is the standard Base64 encoding defined in RFC 4648. The output contains characters from the set A-Z, a-z, 0-9, + and /. The decoder rejects data that contains characters outside this set.
- The encoded String passed to the main method in Main class.

Types.scala :

```
package com.modak

import scala.collection.immutable.ListMap
```

```
//case classes for parsing the Json

case class columnsconfig(
        column_comment : String ,
        column_id : Long ,
        dataType : String
        )
case class sourcecredential(
        credentialId : Long ,
        credentialTypeId : Int
        )
case class tableconfigmap(
         process_id : Long ,
         dataplace_id : Long ,
         schema_id : Int ,
         table_id : Long ,
         datastore_id : Long ,
         datastore_table_id : Long ,
         table_name : String ,
         estimated_count : Int ,
         source_type : String ,
         profling_table_name : String ,
         source_credential : Option[sourcecredential] ,
         columns_config : ListMap[String,columnsconfig ]
        )
 case class auth_Info(authorizationToken : String , endPoint : String)
 case class profiling_Json(table_config_map : tableconfigmap)
 case class JSON(profilingJson : List[profiling_Json] , authInfo : auth_Info)
```

Explanation :

- Case classes were designed that matches the fields of the parsed JSON string.
- A case class named JSON is created with parameters profilingJson whose datatype is given as list and authInfo whose datatype is assumed to be as Nested2.
- For Nested1, table_config_map is given as parameter whose datatype is assumed as Nested3.
- For Nested2 , authorizationToken as String and endPoint as String.
- Case class Nested3 is defined with parameters and datatypes respectively assigned as:
- process_id Long dataplace_id Long, schema_id Int, table_id Long, datastore_id-Int, datastore_table_id Long, table_name String, estimated_count Int,source_type String, profiling_table_name String, source_credential Option[Nested4], columns_config Nested5.
- Option is given so as to fetch the data even if the data is NULL.
- A case class Nested4 is created with parameters credentialId Long,credentialTypeId Int.
- Case class Nested5 is created with parameters and datatypes assigned as column_comment String, column_id Int, dataType String.

Main class :

```
package com.modak
import com.typesafe.scalalogging.LazyLogging
import scala.util.{Failure, Success, Try}
import io.circe.parser.decode
import io.circe.generic.auto._

object Main extends LazyLogging {
  def main(args: Array[String]): Unit = {
    Try {
      //Getting the encoded String from the command line arguments
      val encodedString = args.head
```

```
    logger.info(s"Encoded Json string :  $encodedString")
    //Decoded the String to json
    val decodedString = decodeBase64String(encodedString)
    logger.info(s"Decoded Json string :  $decodedString")
    //parsing the json by calling JsonParser method
    val parsedJson = jsonParser(decodedString)
    //Logging the required fields
    logger.info(s"authorizationToken is : ${parsedJson.authInfo.authorizationToken}")
    logger.info(s"endpoint is : ${parsedJson.authInfo.endPoint}")
    logger.info(s"datastore_id : ${parsedJson.profilingJson(0).table_config_map.datastore_id}")
    logger.info(s"profling_table_name :
${parsedJson.profilingJson(0).table_config_map.profling_table_name}")
    for (a <- parsedJson.profilingJson(0).table_config_map.columns_config) {
      logger.info(a._1 + " " + "\"\"" + " " + a._2.column_id + " " + a._2.dataType)
    }
  }
  match {
    case Success(j) =>
      logger.info("Json parsing completed successfully")
    case Failure(f) =>
      logger.error(s"Json Parsing failed with the exception : ${f.getLocalizedMessage}")
      throw f
  }
  def decodeBase64String(str: String): String = {
    new String(java.util.Base64.getDecoder.decode(str))
  }
  def jsonParser(str: String): JSON = {
    logger.info("Started Json parsing")
    decode[JSON](str) match {
```

```
      case Right(json) => json
      case Left(exception) =>
        logger.error(s"Json Parsing Failed with Exception ${exception.getLocalizedMessage}")
        logger.error("Invalid Input JSON Provided")
        throw exception
    }
  }
 }
}
```

Explanantion :

- Imported the required libraries.
- Encoded String was decoded using decode method of Base64 schema.
- As a result, we get either a ParsingError or a Json object. We'll then use the match statement to distinguish between the returned values .
- If it is right json, it returns json object or it gives exception .
- Then printed the required fields.
- Here columns_config datatype is List. So using 'for each' to iterate over each fields in it

Execution :

Compiled the code :

- The Scala version was changed to 2.11.12.

```
sbt:jsonparsing> ++2.11.12
[info] Setting Scala version to 2.11.12 on 1 projects.
[info] Reapplying settings...
[info] set current project to jsonparsing (in build file:/mnt/c/WorkArea/Spark/JsonParsing/)
```

- Compile the code using compile command.

```
sbt:jsonparsing> compile
[info] compiling 2 Scala sources to /mnt/c/WorkArea/Spark/JsonParsing/target/scala-2.11/classes ...
[success] Total time: 47 s, completed Jun 9, 2022 12:31:05 PM
sbt:jsonparsing>
```

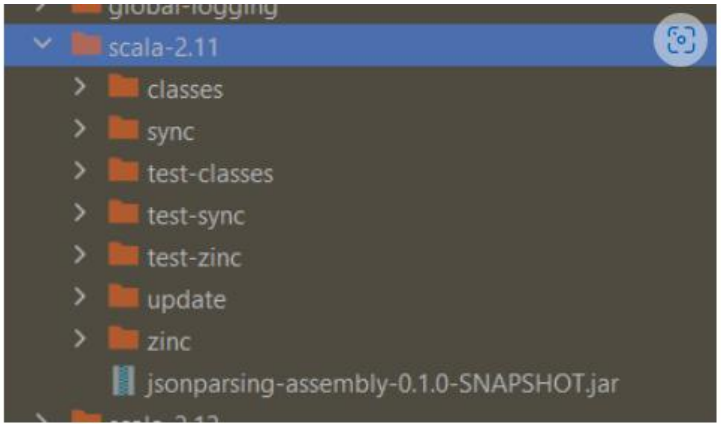
- Run the code by using test command.

```
[pool-49-thread-5] INFO com.modak.Main$ - Started Json parsing
[pool-49-thread-5] INFO com.modak.Main$ - authorizationToken is : eyJhbGciOiJSUzI1N
IiwiaWF0IjoxNjMwMzI3NzI0fQ.CRtShmE9-jFvsySHOx92rSR7Y2i2a1MDm35URGeEGokNsM5gcNHVugt1
himNGaXCA_sAOqkoTZy2nL8DcUUeZXsPUwMFf8ypRj3OpUcmu4Xxg9XufftY1MQQqpE0IrD3mnVfgiIJMyR
[pool-49-thread-5] INFO com.modak.Main$ - endpoint is : https://localhost:8061/fire
[pool-49-thread-5] INFO com.modak.Main$ - datastore_id : 115
[pool-49-thread-5] INFO com.modak.Main$ - profling_table_name : myFile_pipe.csv
[pool-49-thread-5] INFO com.modak.Main$ - id "" 554 INT
[pool-49-thread-5] INFO com.modak.Main$ - firstname "" 553 STRING
[pool-49-thread-5] INFO com.modak.Main$ - lastname "" 555 STRING
[pool-49-thread-5] INFO com.modak.Main$ - Json parsing completed successfully
[info] Test:
[info] Run completed in 1 second, 417 milliseconds.
[info] Total number of tests run: 0
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 0, failed 0, canceled 0, ignored 0, pending 0
[info] No tests were executed.
[success] Total time: 6 s, completed Jun 9, 2022 12:34:24 PM
```

Built jars :

- Used assembly command to built jars.

```
[info] Run completed in 1 second, 363 milliseconds.
[info] Total number of tests run: 0
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 0, failed 0, canceled 0, ignored 0, pending 0
[info] No tests were executed.
[info] Strategy 'discard' was applied to a file (Run the task at debug level to see details)
[warn] Ignored unknown package option FixedTimestamp(Some(1262304000000))
[success] Total time: 192 s (03:12), completed Jun 9, 2022 12:39:49 PM
sbt:jsonparsing>
```



- Clean command is used to remove the built jars.

Executed jar using spark submit :

- The spark-submit command is a utility to run or submit a Spark or PySpark application program (or job) to the cluster by specifying options and configurations.
- spark-submit --packages "com.github.music-of-the-ainur:almaren-framework_2.11:0.9.2-2.4,org.apache.spark:spark-hive_2.11:2.4.7" --class com.modak.Main target/scala-2.11/jsonparsing-assembly-0.1.0-SNAPSHOT.jar

eyAKCiAgInByb2ZpbGluZ0pzb24iOiBbIAoKICAgIHsgCgogICAgICAidGFibGVfY29uZmlnX21
hcCI6IHsgCgogICAgICAgICJwcm9jZXNzX2lkIjogMTYzMDMyNzU2MjExNTUsIAoKICAgICAgI
CAiZGF0YXBsYWNlX2lkIjogMjU4LCAKCiAgICAgICAgInNjaGVtYV9pZCI6IDU0LCAKCiAgICAg
ICAgInRhYmxlX2lkIjogMjUwNDcxLCAKCiAgICAgICAgImRhdGFzdG9yZV9pZCI6IDExNSwgC
gogICAgICAgICJkYXRhc3RvcmVfdGFibGVfaWQiOiAyNTA0NzEsIAoKICAgICAgICAidGFibGV
fbmFtZSI6ICJteUZpbGVfcGlwZS5jc3YiLCAKCiAgICAgICAgImVzdGltYXRlZF9jb3VudCI6IDAsI
AoKICAgICAgICAic291cmNlX3R5cGUiOiAiZ2NzIiwgCgogICAgICAgICJwcm9mbGluZ190YWJ
sZV9uYW1lIjogIm15RmlsZV9waXBlLmNzdiIsIAoKICAgICAgICAic291cmNlX2NyZWRlbnRpY
WwiOiBudWxsLCAKCiAgICAgICAgImNvbHVtbnNfY29uZmlnIjogeyAKCiAgICAgICAgICAiaW
QiOiB7IAoKICAgICAgICAgICAgImNvbHVtbl9jb21tZW50IjogIiIsIAoKICAgICAgICAgICAgImNv
bHVtbl9pZCI6IDU1NCwgCgogICAgICAgICAgICAiZGF0YVR5cGUiOiAiSU5UIiAKCiAgICAgICA
gICB9LCAKCiAgICAgICAgICAiZmlyc3RuYW1lIjogeyAKCiAgICAgICAgICAgICJjb2x1bW5fY29t
bWVudCI6ICIiLCAKCiAgICAgICAgICAgICJjb2x1bW5faWQiOiA1NTMsIAoKICAgICAgICAgICA
gImRhdGFUeXBlIjogIlNUUklORyIgCgogICAgICAgICAgfSwgCgogICAgICAgICAgImxhc3RuYW
1lIjogeyAKCiAgICAgICAgICAgICJjb2x1bW5fY29tbWVudCI6ICIiLCAKCiAgICAgICAgICAgICJjb
2x1bW5faWQiOiA1NTUsIAoKICAgICAgICAgICAgImRhdGFUeXBlIjogIlNUUklORyIgCgogICA
gICAgICAgfSAKCiAgICAgICAgfSAKCiAgICAgIH0gCgogICAgfSAKCiAgXSwgCgogICJhdXRoSW
5mbyI6IHsgCgogICAgImF1dGhvcml6YXRpb25Ub2tlbiI6ICJleUpoYkdjaU9pSlNVekkxTmlKO
S5leUpwYzNNaU9pSkdTVkpGVTBoUFZGTmZRVkJRVEVsRFFWUkpUMDRpTENKelpYTnph
Vzl1WDJsa0lqb3lOVGN3T0RFd05qWXNJbVY0Y0NJNk1UWXpNRE0wT1RNeU5Dd2lkWE5s
Y2tsa0lqb2lZbTkwYzE5emRtTmZZV05qSWl3aWFXRjBJam94TmpNd016STNOekkwZlEuQ1
J0U2htRTktakZ2c3lTSE94OTJyU1I3WTJpMmExTURtMzVVUkdlRUdva05zTTVnY05IVnVnd
DF5d0RLSGRpUkttdlV6c3gyNERnRTNwRi1COGdMVjVHT3pqZEcwd1RMdkxkLW1oSVQ3N
UZLNHFpcmZsWGJFLXhYTnozSDJYeWxlTTFFUHhYU09QZXVwSVNwUnFsVl80QlZIX2dtTlZ

MNm1hWUpMblR1d1dqdlNqWVFvUXhYWUdFZWhpbU5HYVhDQV9zQU9xa29UWnkybk
w4RGNVVWVaWHNQVXdNRmY4eXBSajNPcFVjbXU0WHhnOVh1ZmZ0WTFNUVFxcEUwS
XJEM21uVmZnaUlKTXlSakM5d0VTaTJoWmtxVGttbFJHNTctTzl3RWFHRzg2VEVndFpRU2Y
3eFhVejlyNGFLWXZ0RGZfc1dOV0p6UlFVbmphNFhhbEpIZDlBIiwgCgogICAgImVuZFBvaW
50IjogImh0dHBzOi8vbG9jYWxob3N0OjgwNjEvZmlyZXNob3RzL2NyZWRlbnRpYWxzRmV0
Y2hXZWJTZXJ2aWNlIiAKCiAgfSAKCn0g

- Output of first Json String.

```
[pool-48-thread-3] INFO com.modak.Main$ - Started Json parsing
[pool-48-thread-3] INFO com.modak.Main$ - authorizationToken is : eyJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJGSVJFU0hPVFNfQV
IiwiaWF0IjoxNjMwMzI3NzI0fQ.CRtShmE9-jFvsySHOx92rSR7Y2i2a1MDm35URGeEGokNsM5gcNHVugt1ywDKHdiRKmvUzsx24DgE3pF-B8gLV5
himNGaXCA_sAOqkoTZy2nL8DcUUeZXsPUwMFf8ypRj3OpUcmu4Xxg9XufftY1MQQqpE0IrD3mnVfgiIJMyRjC9wESi2hZkqTkmlRG57-O9wEaGG86
[pool-48-thread-3] INFO com.modak.Main$ - endpoint is : https://localhost:8061/fireshots/credentialsFetchWebServi
[pool-48-thread-3] INFO com.modak.Main$ - datastore_id : 115
[pool-48-thread-3] INFO com.modak.Main$ - profling_table_name : myFile_pipe.csv
[pool-48-thread-3] INFO com.modak.Main$ - id "" 554 INT
[pool-48-thread-3] INFO com.modak.Main$ - firstname "" 553 STRING
[pool-48-thread-3] INFO com.modak.Main$ - lastname "" 555 STRING
[pool-48-thread-3] INFO com.modak.Main$ - Json parsing completed successfully
[info] Test:
[info] Run completed in 1 second, 90 milliseconds.
[info] Total number of tests run: 0
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 0, failed 0, canceled 0, ignored 0, pending 0
[info] No tests were executed.
[success] Total time: 6 s, completed Jun 11, 2022 11:15:00 AM
```

- Output of second Json String.

```
2/06/09 12:45:14 INFO Main$: Started Json parsing
2/06/09 12:45:15 INFO Main$: authorizationToken is : eyJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJGSVJFU0hPVFNfQVBQTElDQVRJT04iLCJzZXNzaW9uX2
jMwMzI3NzI0fQ.CRtShmE9-jFvsySHOx92rSR7Y2i2a1MDm35URGeEGokNsM5gcNHVugt1ywDKHdiRKmvUzsx24DgE3pF-B8gLV5GOzjdG0wTLvLd-mhIT75FK4qirfl
kkoTZy2nL8DcUUeZXsPUwMFf8ypRj3OpUcmu4Xxg9XufftY1MQQqpE0IrD3mnVfgiIJMyRjC9wESi2hZkqTkmlRG57-O9wEaGG86TEgtZQSf7xXUz9r4aKYvtDf_sWNw
2/06/09 12:45:15 INFO Main$: endpoint is : https://localhost:8061/fireshots/credentialsFetchWebService
2/06/09 12:45:15 INFO Main$: datastore_id : 115
2/06/09 12:45:15 INFO Main$: profling_table_name : myFile_pipe.csv
2/06/09 12:45:15 INFO Main$: id "" 554 INT
2/06/09 12:45:15 INFO Main$: firstname "" 553 STRING
2/06/09 12:45:15 INFO Main$: lastname "" 555 STRING
2/06/09 12:45:15 INFO Main$: Json parsing completed successfully
2/06/09 12:45:15 INFO ShutdownHookManager: Shutdown hook called
2/06/09 12:45:15 INFO ShutdownHookManager: Deleting directory /tmp/spark-423cc1a6-a936-4631-8aef-4c2aff9dfbd3
anjana@mt4020-sanjana:/mnt/c/WorkArea/Spark/JsonParsing$
```

USER STORY 2

Title - Ingestion from source JDBC to target File (csv,parquet,avro)

Description

a. Design input json of your choice for reading from source JDBC and writing to the target File

Design the json in generic way so that all we need to change is the input json for different permutations .

Example :

{

source : {JDBC} - "have all required fields necessary connect to JDBC using spark (jdbc url, driver , table name etc)"

target : {FILE} - "have all the fields necessary to write to the target file (format,path etc)"

}

b.Take the input json as base64 encoded string argument to the code.

c. Decode the base64 string

d.Parse the input json using circe library

c.Read source Details and write to target with the help of input json parsed .

Acceptance Criteria

*USE Almren sourceJdbc componnent to read the data from source JDBC

*After geting the dataframe , write to the target file using spark.

*Source JDBC can be postgres,orcale,mysql (ANY JDBC SOURCE)

*All the options should be taken as part of the input json (for reading csv,paruqet files and writing to JDBC)

*Target FILE can be any format (avro,parquet,csv)

*CODE should be packaged as jar and run using spark-submit

Code Explanantion :

Build.sbt :

```
ThisBuild / name := "test"
ThisBuild / organization := "com.modak"
lazy val scala211 = "2.11.12"
lazy val scala212 = "2.12.15"
crossScalaVersions := Seq(scala211, scala212)
ThisBuild / scalaVersion := scala212
val sparkVersion = "2.4.7"
val circeVersion = "0.12.0-M3"
libraryDependencies ++= Seq(
  "org.apache.spark" %% "spark-core" % sparkVersion % "provided",
  "org.apache.spark" %% "spark-sql" % sparkVersion % "provided",
  "org.apache.spark" %% "spark-hive" % sparkVersion % "provided",
  "io.circe" %% "circe-core" % circeVersion,
  "io.circe" %% "circe-generic" % circeVersion,
  "io.circe" %% "circe-parser" % circeVersion,
  "org.postgresql" % "postgresql" % "42.3.3",
  "com.typesafe.scala-logging" %% "scala-logging" % "3.9.2",
  "com.github.music-of-the-ainur" %% "almaren-framework" % "0.9.3-2.4" % "provided",
  "org.scalatest" %% "scalatest" % "3.0.5" % "test",
  "org.slf4j" % "slf4j-api" % "1.7.36",
  "org.slf4j" % "slf4j-simple" % "1.7.36"
)
```

```
assemblyMergeStrategy in assembly := {
  case PathList("META-INF", xs@_*) => MergeStrategy.discard
  case x => MergeStrategy.first
}
```

Explanation :

- **Syntax to add dependencies:** libraryDependencies ++= Seq(groupID%%artifactID%version)
- **"provided"** indicates that "it expects respective dependency to be mentioned at the runtime".
- Circe is a Scala library that simplifies working with JSON, allowing us to easily decode a JSON string into a Scala object or convert a Scala object to JSON.
- Three dependencies were added to install this Circe library.
- Logging is used to maintain the logs of an application. It is very much required to monitor our application.
- Dependencies were added for logging the output.
- Simple Logging Facade for Java (abbreviated SLF4J) acts as a facade for different logging frameworks (e.g., java.util.logging, logback, Log4j). It offers a generic API, making the logging independent of the actual implementation
- "org.scalatest" %% "scalatest" % "3.0.5" % "test"
    - **Scalatest** is a testing library to test Scala code by running the test cases.
    - **"test"** indicates that the dependency is limited to the test class itself.

plugins.sbt :

```
addSbtPlugin("com.eed3si9n" % "sbt-assembly" % "0.14.10")
```

Explanation :

- The sbt-assembly plugin is an SBT plugin for building a single independent fat JAR file with all dependencies included.

Test class :

```
package modak
import com.modak.Main
import org.scalatest._

class Test extends FunSuite with BeforeAndAfter {
  //Json String to parse
          val plainString = """ {
          "source":{
          "url": "jdbc:postgresql://w3.training5.modak.com:5432/training",
          "username": "mt4020",
          "password": "mt4020@m02y22",
          "driver": "org.postgresql.Driver",
          "tableName": "empp"},
           "target":{
            "format" : "avro",
            "path" : "/mnt/c/WorkArea/Spark/TableToFiles/Files/file.avro"
         }
         }""".stripMargin
  //encoding json String
  val encodedString = java.util.Base64.getEncoder.encodeToString(plainString.getBytes())
  println(encodedString)
  //calling main method  which is in  Main class
  Main.main(Array[String](encodedString))

}
```

Explanation :

- Base64 is a binary-to-text encoding scheme that represents binary data in a printable ASCII string format.
- getEncoder : It returns a Base64.Encoder that encodes using the Basic type base64 encoding scheme.
- Basic: This is the standard Base64 encoding defined in RFC 4648. The output contains characters from the set A-Z, a-z, 0-9, + and /. The decoder rejects data that contains characters outside this set.
- The encoded String passed to the main method in Main class.

Types.scala :

```
package com.modak

//case classes for Json parsing
case class Source_1(url : String,username : String,password : String,driver : String, tableName : String)
case class Target_1(format : String, path : String )
case class JSON( source : Source_1 , target : Target_1)
```

Explanation :

- Case classes were designed that matches the fields of the parsed JSON string.
- Case class JSON was created with parameters target and source.
- target with datatype assumed as Target_1 and source datatype assumed as Source_1.

- Target_1 has parameters format datatype String and path datatype String.
- Source_1 has parameters url, username, password, driver and tableName datatype as String.

Main Class :

```
package com.modak

import com.github.music.of.the.ainur.almaren.builder.Core.Implicit
import com.github.music.of.the.ainur.almaren.Almaren
import com.typesafe.scalalogging.LazyLogging
import scala.util.{Failure, Success, Try}
import io.circe.parser.decode
import io.circe.generic.auto._
import org.apache.spark.sql.SaveMode

object Main extends LazyLogging {
  def main(args: Array[String]): Unit = {
    val almaren = Almaren("Table to Files")
    val spark = almaren.spark.master("local[*]").config("spark.sql.shuffle.partitions", "1")
    Try {
      //Getting the encoded String from the command line arguments
      val encodedString = args.head
      logger.info(s"Encoded Json string :  $encodedString")
      //Decoded the String to json
      val decodedString = decodeBase64String(encodedString)
      logger.info(s"Decoded Json string :  $decodedString")
      //parsing the json by calling JsonParser method
      val parsedJson = jsonParser(decodedString)
      //getting the files  information from the parsed Json
      val format=parsedJson.target.format
```

```
    val path=parsedJson.target.path
    //getting database information from the parsed Json
    val url=parsedJson.source.url
    val username=parsedJson.source.username
    val password=parsedJson.source.password
    val driver=parsedJson.source.driver
    val table=parsedJson.source.tableName
    //creating a dataframe from the table
    val df=almaren.builder.sourceJdbc(s"${url}", s"${driver}","select * from "+
s"${table}",Some(s"${username}"),Some(s"${password}"),Map()).batch
    //writing to a file in csv format
    if(path.contains("csv")){

df.write.format(s"${format}").mode(SaveMode.Overwrite).option("header","true").save(s"${pa
th}")
    }
    //writing to a file in avro or parquet format
    else{
      df.write.format(s"${format}").mode(SaveMode.Overwrite).save(s"${path}")
    }
  }
  match {
    case Success(j) =>
      logger.info("Json parsing completed successfully")
    case Failure(f) =>
      logger.error(s"Json Parsing failed with the exception : ${f.getLocalizedMessage}")
      throw f
  }
  def decodeBase64String(str: String): String = {
```

```
    new String(java.util.Base64.getDecoder.decode(str))
  }
  def jsonParser(str: String): JSON = {
    logger.info("Started Json parsing")
    decode[JSON](str) match {
      case Right(json) => json
      case Left(exception) =>
        logger.error(s"Json Parsing Failed with Exception ${exception.getLocalizedMessage}")
        logger.error("Invalid Input JSON Provided")
        throw exception
    }
  }
 }
}
```

Explanation of Main class :

- Imported the required libraries.
- Encoded String was decoded using decode method of Base64 schema.
- As a result, we get either a ParsingError or a Json object. We'll then use the match statement to distinguish between the returned values .
- If it is right json, it returns json object or it gives exception .
- Almaren object and Spark session was created.
- SourceJdbc component of Almaren framework was used to get table information form the database.
- Created the dataframe by using batch.
- **Batch** executes the almaren tree returning a Dataframe.
- Then created a files with the same information present in dataframe.

Execution of Code :

Compiled the code :

- The Scala version was changed to 2.11.12.

```
sbt:tabletofiles> ++2.11.12
[info] Setting Scala version to 2.11.12 on 1 projects.
[info] Reapplying settings...
[info] set current project to tabletofiles (in build file:/mnt/c/workarea/spark/TableToFiles/)
sbt:tabletofiles>
```

- Compile the code using compile command.

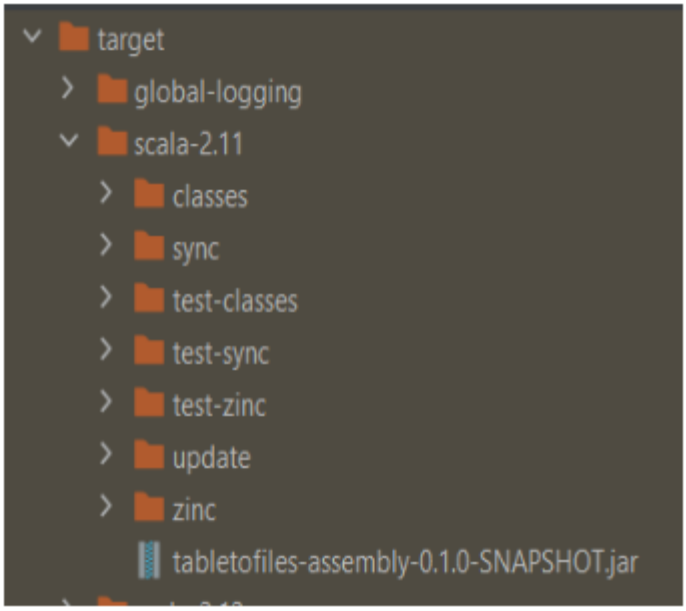
```
sbt:tabletofiles> compile
[info] compiling 2 Scala sources to /mnt/c/workarea/spark/TableToFiles/target/scala-2.11/classes ...
[success] Total time: 24 s, completed Jun 11, 2022 8:52:00 PM
sbt:tabletofiles>
```

- Run the code by using test command.

```
[info] Test:
[info] Run completed in 20 seconds, 773 milliseconds.
[info] Total number of tests run: 0
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 0, failed 0, canceled 0, ignored 0, pending 0
[info] No tests were executed.
[success] Total time: 24 s, completed Jun 11, 2022 8:58:02 PM
sbt:tabletofiles>
```

Built jars :

- Used assembly command to built jars.



- Clean command is used to remove the built jars.

Executed jar using spark submit :

- The spark-submit command is a utility to run or submit a Spark or PySpark application program (or job) to the cluster by specifying options and configurations.
- spark-submit --packages "com.github.music-of-the-ainur:almaren-framework_2.11:0.9.2-2.4,org.apache.spark:spark-hive_2.11:2.4.7,org.postgresql:postgresql:42.3.3" --class com.modak.Main target/scala-2.11/tabletofiles-assembly-0.1.0-SNAPSHOT.jar
IHsKICAgICAgICAgICAgInNvdXJjZSI6ewogICAgICAgICAgICAidXJsIjogImpkYmM6cG9zdGdyZXNxbDovL3czLnRyYWluaW5nNS5tb2Rhay5jb206NTQzMi90cmFpbmluZyIsCiAgICAgICAgICAgICJ1c2VybmFtZSI6ICJtdDQwMjAiLAogICAgICAgICAgICAicGFzc3dvcmQiOiAibXQ0MDIwQG0wMnkyMiIsCiAgICAgICAgICAgICJkcml2ZXIiOiAib3JnLnBvc3RncmVzcWwuRHJpdmVyIiwKICAgICAgICAgICAgInRhYmxlTmFtZSI6ICJlbXBwIn0sCiAgICAgICAgICAgICAidGFyZ2V0Ijp7CiAgICAgICAgICAgICAgICJmb3JtYXQiIDogImNzdiIsCiAgICAgICAgICAgICAgICJwYXRoIiA6ICIvbW50L2MvV29ya0FyZWEvU3BhcmsvVGFibGVUb0ZpbGVzL0ZpbGVzL2ZpbGUuY3N2IgogICAgICAgICAgICAgfQogICAgICAgICAgIH0=

- Data in empp table in the database.

select * from empp Enter a SQL expression to filter results (us

	eid	ename	sal	doj
1	1	sanju	1,000	2022-11-01
2	2	sanju	2,000	2022-11-01
3	3	vishnu	3,000	2022-08-01
4	4	madhav	4,000	2022-07-01
5	5	keshav	5,000	2022-06-01
6	9	keshav	90,000	2022-02-01
7	10	kesha	10,000	2022-02-01
8	11	keshav	90,000	2022-02-01

- Data in csv file

```
1,sanju,1000.00,2022-11-01
2,sanju,2000.00,2022-11-01
3,vishnu,3000.00,2022-08-01
4,madhav,4000.00,2022-07-01
5,keshav,5000.00,2022-06-01
9,keshav,90000.00,2022-02-01
10,kesha,10000.00,2022-02-01
11,keshav,90000.00,2022-02-01
```

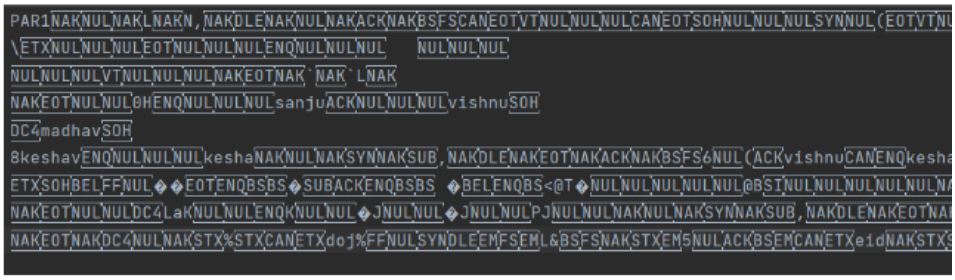
- Data in avro file.

Objavro.schema{"type":"record","name":"topLevelRecord","fields":[{"name":"eid","type":["int","null"]},{"name":"ename","type":["string","null"]},{"name":"sal","type":[{"type":"fixed","name":"fixed","namespace":"topLevelRecord.sal","size":5,"logicalType":"decimal","precision":10,"scale":2},"null"]},{"name":"doj","type":[{"type":"int","logicalType":"date"},"null"]}]}0org.apach

e.spark.version

2.4.7avro.codecsnappy

- Data in parquet file.



USER STORY 3 :

Title - Ingestion form source File (csv,parquet,avro)) to target JDBC

Description

a. Design input json of your choice for reading from source FILE and writing to the target File

Design the json in generic way so that all we need to change is the input json for different permutations

Example :

{

source : {FILE} - "have all the fields necessary to read FILE (format,path etc)"

target : {JDBC} - "have all required fields necessary connect to JDBC using spark and write to it (jdbc url, driver , table name etc)"

}

b.Take the input json as base64 encoded string argument to the code.

c. Decode the base64 string

d.Parse the input json using circe library

c.Read source Details and write to target with the help of input json parsed .

Acceptance Criteria

*USE Almaren sourceFile component for reading the file and targetJdbc Component for writing to target

*source FILE can be any format (avro,parquet,csv)

*target JDBC can be postgres,orcale,mysql (ANY JDBC SOURCE)

*All the options should be taken as part of the input json (for reading csv,paruqet files and writing to JDBC)

*CODE should be packaged as jar and run using spark-submit

built.sbt :

```
ThisBuild / name := "test"
ThisBuild / organization := "com.modak"

lazy val scala211 = "2.11.12"
lazy val scala212 = "2.12.15"

crossScalaVersions := Seq(scala211, scala212)
ThisBuild / scalaVersion := scala212

val sparkVersion = "2.4.7"
val circeVersion = "0.12.0-M3"
libraryDependencies ++= Seq(
  "org.apache.spark" %% "spark-core" % sparkVersion % "provided",
  "org.apache.spark" %% "spark-sql" % sparkVersion % "provided",
  "org.apache.spark" %% "spark-hive" % sparkVersion % "provided",
```

```
  "io.circe" %% "circe-core" % circeVersion,
  "io.circe" %% "circe-generic" % circeVersion,
  "io.circe" %% "circe-parser" % circeVersion,
  "org.postgresql" % "postgresql" % "42.3.3",
  "com.typesafe.scala-logging" %% "scala-logging" % "3.9.2",
  "com.github.music-of-the-ainur" %% "almaren-framework" % "0.9.3-2.4" % "provided",
  "org.scalatest" %% "scalatest" % "3.0.5" % "test",
  "org.slf4j" % "slf4j-api" % "1.7.36",
  "org.slf4j" % "slf4j-simple" % "1.7.36"
)

assemblyMergeStrategy in assembly := {
  case PathList("META-INF", xs@_*) => MergeStrategy.discard
  case x => MergeStrategy.first
}
```

Explanation :

- **Syntax to add dependencies:** libraryDependencies ++= Seq(groupID%%artifactID%version)
- **"provided"** indicates that "it expects respective dependency to be mentioned at the runtime".
- Circe is a Scala library that simplifies working with JSON, allowing us to easily decode a JSON string into a Scala object or convert a Scala object to JSON.
- Three dependencies were added to install this Circe library.
- Logging is used to maintain the logs of an application. It is very much required to monitor our application.
- Dependencies were added for logging the output.

- Simple Logging Facade for Java (abbreviated SLF4J) acts as a facade for different logging frameworks (e.g., java.util.logging, logback, Log4j). It offers a generic API, making the logging independent of the actual implementation
- "org.scalatest" %% "scalatest" % "3.0.5" % "test"
    - **Scalatest** is a testing library to test Scala code by running the test cases.
    - **"test"** indicates that the dependency is limited to the test class itsel

plugins.sbt :

```
addSbtPlugin("com.eed3si9n" % "sbt-assembly" % "0.14.10")
```

Test class :

```
package modak

import com.modak.Main
import org.scalatest._

class Test extends FunSuite with BeforeAndAfter {
  //Json String to parse
  val plainString = """ {
                |  "source":{
                |    "format" : "avro",
                |    "path" : "/mnt/c/WorkArea/Spark/FilesToTables/Files/file3.avro",
                |    "header" : "true"
                |},
                |  "target1":{
                |  "url": "jdbc:postgresql://w3.training5.modak.com:5432/training",
                |  "username": "mt4020",
```

```
                | "password": "mt4020@m02y22",
                | "driver": "org.postgresql.Driver",
                | "tableName": "Us3_table2"}
                | }""".stripMargin
  //encoding the Json String
  val encodedString = java.util.Base64.getEncoder.encodeToString(plainString.getBytes())
  println(encodedString)
  //calling main method  which is in  Main class
  Main.main(Array[String](encodedString))
}
```

Explanation :

- Base64 is a binary-to-text encoding scheme that represents binary data in a printable ASCII string format.
- getEncoder : It returns a Base64.Encoder that encodes using the Basic type base64 encoding scheme.
- Basic: This is the standard Base64 encoding defined in RFC 4648. The output contains characters from the set A-Z, a-z, 0-9, + and /. The decoder rejects data that contains characters outside this set.
- The encoded String passed to the main method in Main class.

Types.scala :

```
package com.modak
//case classes for Json parsing
case class Target_1(url : String,username : String,password : String,driver : String, tableName :
String)
case class Source_1 (format : String, path : String , header : String)
case class JSON(source : Source_1  , target : Target_1)
```

Explanation :

- Case classes were designed that matches the fields of the parsed JSON string.
- Case class JSON was created with parameters target and source.
- target with datatype assumed as Target_1 and source datatype assumed as Source_1.
- Source_1 has parameters format datatype String and path datatype String and header.
- Target_1 has parameters url, username, password, driver and tableName datatype as String.

Main class :

```
package com.modak

import com.github.music.of.the.ainur.almaren.builder.Core.Implicit
import com.github.music.of.the.ainur.almaren.Almaren
import com.typesafe.scalalogging.LazyLogging
import scala.util.{Failure, Success, Try}
import io.circe.parser.decode
import io.circe.generic.auto._
import org.apache.spark.sql.SaveMode
object Main extends LazyLogging {
  def main(args: Array[String]): Unit = {
    val almaren = Almaren("App Name")
    val spark = almaren.spark.master("local[*]").config("spark.sql.shuffle.partitions", "1")
    Try {
      //Getting the encoded String from the command line arguments
      val encodedString = args.head
      logger.info(s"Encoded Json string :  $encodedString")
      //Decoded the String to json
      val decodedString = decodeBase64String(encodedString)
```

```
    logger.info(s"Decoded Json string :  $decodedString")
    //parsing the json by calling JsonParser method
    val parsedJson = jsonParser(decodedString)
    //getting the files information from the parsed Json
    val format=parsedJson.source.format
    val path=parsedJson.source.path
    val header=parsedJson.source.header
    //getting database information from the parsed Json
    val url=parsedJson.target.url
    val username=parsedJson.target.username
    val password=parsedJson.target.password
    val driver=parsedJson.target.driver
    val table=parsedJson.target.tableName
    //reading files and stored into table in database
    val df = almaren.builder.sourceFile(s"${format}", s"${path}", Map("header" -> s"${header}",
"inferSchema" -> "true"))
      .targetJdbc(s"${url}", s"${driver}", s"${table}", SaveMode.Overwrite,
Some(s"${username}"), Some(s"${password}")).batch
  }
  match {
    case Success(j) =>
      logger.info("Json parsing completed successfully")
    case Failure(f) =>
      logger.error(s"Json Parsing failed with the exception : ${f.getLocalizedMessage}")
      throw f
  }
  def decodeBase64String(str: String): String = {
    new String(java.util.Base64.getDecoder.decode(str))
  }
```

```
  def jsonParser(str: String): JSON = {
    logger.info("Started Json parsing")
    decode[JSON](str) match {
      case Right(json) => json
      case Left(exception) =>
        logger.error(s"Json Parsing Failed with Exception ${exception.getLocalizedMessage}")
        logger.error("Invalid Input JSON Provided")
        throw exception
    }
  }
 }
}
```

Explanation of Main class :

- Imported the required libraries.
- Encoded String was decoded using decode method of Base64 schema.
- As a result, we get either a ParsingError or a Json object. We'll then use the match statement to distinguish between the returned values .
- If it is right json, it returns json object or it gives exception .
- Almaren object and Spark session was created.
- SourceFile component of Almaren framework was used to read data from a files of different formats.
- TargetJdbc component of Almaren framework was used to store information to the table in database

- Created the dataframe by using batch.
- **Batch** executes the almaren tree returning a Dataframe.
- Then created a files with the same information present in dataframe.

Compilation of code :

- The Scala version was changed to 2.11.12.

```
bt:filestotables> ++ 2.11.12
info] Setting Scala version to 2.11.12 on 1 projects.
info] Reapplying settings...
info] set current project to filestotables (in build file:/mnt/c/workArea/spark/FilesToTables/)
bt:filestotables>
```

- Compile the code using compile command.

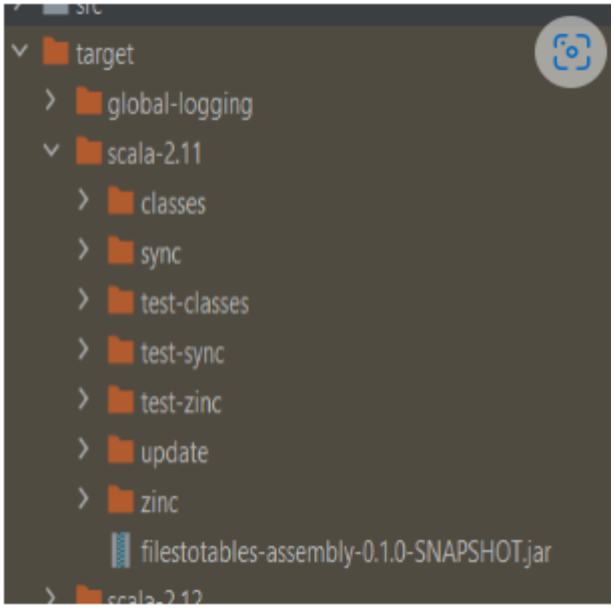
```
sbt:filestotables> compile
[info] compiling 2 Scala sources to /mnt/c/workArea/spark/FilesToTables/target/scala-2.11/classes ...
[success] Total time: 31 s, completed Jun 12, 2022 11:13:09 AM
sbt:filestotables>
```

- Run the code by using test command.

```
[info] Test:
[info] Run completed in 23 seconds, 789 milliseconds.
[info] Total number of tests run: 0
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 0, failed 0, canceled 0, ignored 0, pending 0
[info] No tests were executed.
[success] Total time: 31 s, completed Jun 12, 2022 11:13:54 AM
sbt:filestotables>
```

Built jars :

- Used assembly command to built jars.



- Clean command is used to remove the built jars.

Executed jar using spark submit :

- The spark-submit command is a utility to run or submit a Spark or PySpark application program (or job) to the cluster by specifying options and configurations.
- spark-submit --packages "com.github.music-of-the-ainur:almaren-framework_2.11:0.9.2-2.4,org.apache.spark:spark-hive_2.11:2.4.7,org.postgresql:postgresql:42.3.3" --class com.modak.Main target/scala-2.11/filestotables-assembly-0.1.0-SNAPSHOT.jar
IHsKICAic291cmNlIjp7CiAgICAgImZvcm1hdCIgOiAicGFycXVldCIsCiAgICAgInBhdGgiIDogIi9tbnQvYy9Xb3JrQXJlYS9TcGFyay9GaWxlc1RvVGFibGVzL0ZpbGVzL2ZpbGUyLnBhcnF1ZXQiLAogICAgICJoZWFkZXIiIDogInRydWUiCn0sCiAgInRhcmdldCI6ewogICJ1cmwiOiAiamRiYzpwb3N0Z3Jlc3FsOi8vdzMudHJhaW5pbmc1Lm1vZGFrLmNvbTo1NDMyL3RyYWluaW5nIiwKICAidXNlcm5hbWUiOiAibXQ0MDIwIiwKICAicGFzc3dvcmQiOiAibXQ0MDIwQG0wMnkyMiIsCiAgImRyaXZlciI6ICJvcmcucG9zdGdyZXNxbC5Ecml2ZXIiLAogICJ0YWJsZU5hbWUiOiAiVXMzX3RhYmxlMSJ9CiB9
- Data in the table after executing it spark submit command.

select * from Us3_table1 Enter a SQL expression to filter

	eid	ename	sal	doj
1	1	sanju	1,000	2022-11-01
2	2	sanju	2,000	2022-11-01
3	3	vishnu	3,000	2022-08-01
4	4	madhav	4,000	2022-07-01
5	5	keshav	5,000	2022-06-01
6	9	keshav	90,000	2022-02-01
7	10	kesha	10,000	2022-02-01
8	11	keshav	90,000	2022-02-01

USER STORY 4 :

Title - Ingestion from HTTP API convert to structure Dataframe to target JDBC

Description

a. Read any open-source API which gives a JSON as response from response deserialize and convert to a structure dataframe using quenya-dsl and write to any target JDBC

Almaren Links :

https://github.com/modakanalytics/quenya-dsl

https://github.com/modakanalytics/http.almaren

Example:

Open-Source API get request:

https://jsonplaceholder.typicode.com/users/

Search for different open-Source API to get different responses.

a.Read data from API using http.almaren connector

b.Deserialize the data and apply quenya-dsl to get required fileds from json object

c. Write the response dataframe to any target JDBC/ target File system (parquet preferable)

Acceptance Criteria

*USE Almaren sourceFile component for reading the API and targetJdbc Component for writing to target

*Any API response we can take and convert them

*Target JDBC can be postgres,orcale,mysql (ANY JDBC SOURCE)

*All the options should be taken as part of the input json (for reading csv,paruqet files and writing to JDBC)

*CODE should be packaged as jar and run using spark-submit or it can be as snippet code in Spark Shell Ex: spark-shell -I file_path_of_scala_file -packages "..."

Explanation of code :

build.sbt :

```
ThisBuild / name := "test"
ThisBuild / organization := "com.modak"
lazy val scala211 = "2.11.12"
lazy val scala212 = "2.12.15"
crossScalaVersions := Seq(scala211, scala212)
ThisBuild / scalaVersion := scala212

val sparkVersion = "2.4.7"
val circeVersion = "0.12.0-M3"
```

```
libraryDependencies ++= Seq(
  "org.apache.spark" %% "spark-core" % sparkVersion % "provided",
  "org.apache.spark" %% "spark-sql" % sparkVersion % "provided",
  "org.apache.spark" %% "spark-hive" % sparkVersion % "provided",
  "io.circe" %% "circe-core" % circeVersion,
  "io.circe" %% "circe-generic" % circeVersion,
  "io.circe" %% "circe-parser" % circeVersion,
  "com.typesafe.scala-logging" %% "scala-logging" % "3.9.2",
  "org.scalatest" %% "scalatest" % "3.0.5" % "test",
  "org.slf4j" % "slf4j-api" % "1.7.36",
  "org.slf4j" % "slf4j-simple" % "1.7.36",
  "com.github.music-of-the-ainur" %% "almaren-framework" % "0.9.3-2.4",
  "org.postgresql" % "postgresql" % "42.3.3",
  "com.github.music-of-the-ainur" %% "http-almaren" % "1.2.4-2.4"
)

assemblyMergeStrategy in assembly := {
  case PathList("META-INF", xs@_*) => MergeStrategy.discard
  case x => MergeStrategy.first
}
```

Explanation :

- **Syntax to add dependencies:** libraryDependencies ++= Seq(groupID%%artifactID%version)
- **"provided"** indicates that "it expects respective dependency to be mentioned at the runtime".
- Circe is a Scala library that simplifies working with JSON, allowing us to easily decode a JSON string into a Scala object or convert a Scala object to JSON.
- Three dependencies were added to install this Circe library.

- Logging is used to maintain the logs of an application. It is very much required to monitor our application.
- Dependencies were added for logging the output.
- Simple Logging Facade for Java (abbreviated SLF4J) acts as a facade for different logging frameworks (e.g., java.util.logging, logback, Log4j). It offers a generic API, making the logging independent of the actual implementation
- "org.scalatest" %% "scalatest" % "3.0.5" % "test"
    - **Scalatest** is a testing library to test Scala code by running the test cases.
    - **"test"** indicates that the dependency is limited to the test class itself.

Plugins.sbt :

```
addSbtPlugin("com.eed3si9n" % "sbt-assembly" % "0.14.10")
```

Test class :

```
package modak
import com.modak.Main
import org.scalatest._

class Test extends FunSuite with BeforeAndAfter {
  //case classes for Json parsing
  val plainString = """  {
                |  "target":{
                |  "url": "jdbc:postgresql://w3.training5.modak.com:5432/training",
                |  "username": "mt4020",
                |  "password": "mt4020@m02y22",
                |  "driver": "org.postgresql.Driver",
                |  "tableName": "Us4_table1"}
                | }""".stripMargin
  //encoding the Json String
  val encodedString = java.util.Base64.getEncoder.encodeToString(plainString.getBytes())
```

```
  println(encodedString)
  //calling main method  which is in  Main class
  Main.main(Array[String](encodedString))

}
```

Explanation :

- Base64 is a binary-to-text encoding scheme that represents binary data in a printable ASCII string format.
- getEncoder : It returns a Base64.Encoder that encodes using the Basic type base64 encoding scheme.
- Basic: This is the standard Base64 encoding defined in RFC 4648. The output contains characters from the set A-Z, a-z, 0-9, + and /. The decoder rejects data that contains characters outside this set.
- The encoded String passed to the main method in Main class.

Types.scala :

```
package com.modak

//Case classes for Json parsing

case class Target_1(url : String,username : String,password : String,
                    driver : String, tableName : String)
case class JSON(target : Target_1)
```

Explanation :

- Case classes were designed that matches the fields of the parsed JSON string.
- Case class JSON has parameter target and datatype assumed as Target_1.
- Case class Target_1 has url, username, password, driver and tableName as parameters with datatype String.

Main class :

```
package com.modak

import com.github.music.of.the.ainur.almaren.Almaren
import com.github.music.of.the.ainur.almaren.builder.Core.Implicit
import com.github.music.of.the.ainur.almaren.http.HTTPConn.HTTPImplicit
import com.github.music.of.the.ainur.quenya.QuenyaDSL
import com.typesafe.scalalogging.LazyLogging
import io.circe.generic.auto._
import io.circe.parser.decode
import org.apache.spark.sql.SaveMode

import scala.util.{Failure, Success, Try}

object Main extends LazyLogging {

  def main(args: Array[String]): Unit = {
    val almaren = Almaren("Http to postgres")
    val spark = almaren.spark.master("local[*]").config("spark.sql.shuffle.partitions", "1")
    Try {
      //Getting the encoded String from the command line arguments
      val encodedString = args.head
      //Decoded the String to json
      logger.info(s"Encoded Json string :  $encodedString")
      //parsing the json by calling JsonParser method
      val decodedString = decodeBase64String(encodedString)
      logger.info(s"Decoded Json string :  $decodedString")
      //getting the files  information from the parsed Json
```

```
    val parsedJson = jsonParser(decodedString)
    //schema for deserializer json String
    val httpOutpustSchema = Some("`Count` BIGINT,`Message` STRING,`SearchCriteria` STRING
,`Results` ARRAY<STRUCT<`Make_ID`: BIGINT, `Make_Name`: STRING>>")
    //creating a frame from the given API
    val df = almaren.builder.sourceSql("SELECT monotonically_increasing_id() as __ID__ ,
'https://vpic.nhtsa.dot.gov/api/vehicles/getallmakes?format=json' as __URL__").http(method =
"GET").deserializer("JSON", "__BODY__", httpOutpustSchema).batch
    df.show(false)
    val quenyaDsl = QuenyaDSL
    quenyaDsl.printDsl(df.select("Results"))
    //wrting quenyaDsl for getting desired fields from json
    val dsl="""Results@Results
            Results.Make_ID$Make_ID:LongType
            Results.Make_Name$Make_Name:StringType"""
    val df2 = almaren.builder.sourceSql("SELECT monotonically_increasing_id() as __ID__ ,
'https://vpic.nhtsa.dot.gov/api/vehicles/getallmakes?format=json' as __URL__").http(method =
"GET").deserializer("JSON", "__BODY__", httpOutpustSchema).dsl(dsl)
    //getting information to connect to database from parsed Json
    val url=parsedJson.target.url
    val username=parsedJson.target.username
    val password=parsedJson.target.password
    val driver=parsedJson.target.driver
    val table=parsedJson.target.tableName
    //Storing the fields ino the database
      df2.targetJdbc(s"${url}", s"${driver}", s"${table}", SaveMode.Overwrite,
Some(s"${username}"), Some(s"${password}")).batch
    //df2.show();
   } match {
```

```
    case Success(j) =>
      logger.info("Json parsing completed successfully")
    case Failure(f) =>
      logger.error(s"Json Parsing failed with the exception : ${f.getLocalizedMessage}")
      throw f
  }
  def decodeBase64String(str: String): String = {
    new String(java.util.Base64.getDecoder.decode(str))
  }

  def jsonParser(str: String): JSON = {
    logger.info("Started Json parsing")
    decode[JSON](str) match {
      case Right(json) => json
      case Left(exception) =>
        logger.error(s"Json Parsing Failed with Exception ${exception.getLocalizedMessage}")
        logger.error("Invalid Input JSON Provided")
        throw exception
    }
  }

 }
}
```

Explanation of code in Main class :

- Imported the required libraries.
- Encoded String was decoded using decode method of Base64 schema.
- As a result, we get either a ParsingError or a Json object. We'll then use the match statement to distinguish between the returned values .
- If it is right json, it returns json object or it gives exception .

- Almaren object and Spark session was created.
- Deserializer : Deserialize the following types XML, JSON and Avro to Spark DataFrame.
- Quenya DSL(Domain Specific Language) is a language that simplifies the task to parser complex semi-structured data.
- **printDsl :** can generate and print a DSL based on a DataFrame
- TargetJdbc component of Almaren framework was used to store information into the tables in the database.
- **Batch** executes the almaren tree returning a Dataframe.

Execution of code :

Compiled the code :

- The Scala version was changed to 2.11.12.

```
sbt:httptojdbc> ++2.11.12
[info] Setting Scala version to 2.11.12 on 1 projects.
[info] Reapplying settings...
[info] set current project to httptojdbc (in build file:/mnt/c/workArea/spark/HttpToJdbc/)
sbt:httptojdbc>
```

- Compile the code using compile command.

```
https://repo1.maven.org/maven2/com/lihaoyi/requests_2.11/0.7.0/requests_2.11-0.7.0.jar
  100.0% [##########] 148.3 KiB (73.2 KiB / s)
[info] Fetched artifacts of
[info] compiling 2 Scala sources to /mnt/c/workArea/spark/HttpToJdbc/target/scala-2.11/classes ...
[success] Total time: 53 s, completed Jun 12, 2022 3:06:29 PM
sbt:httptojdbc>
```

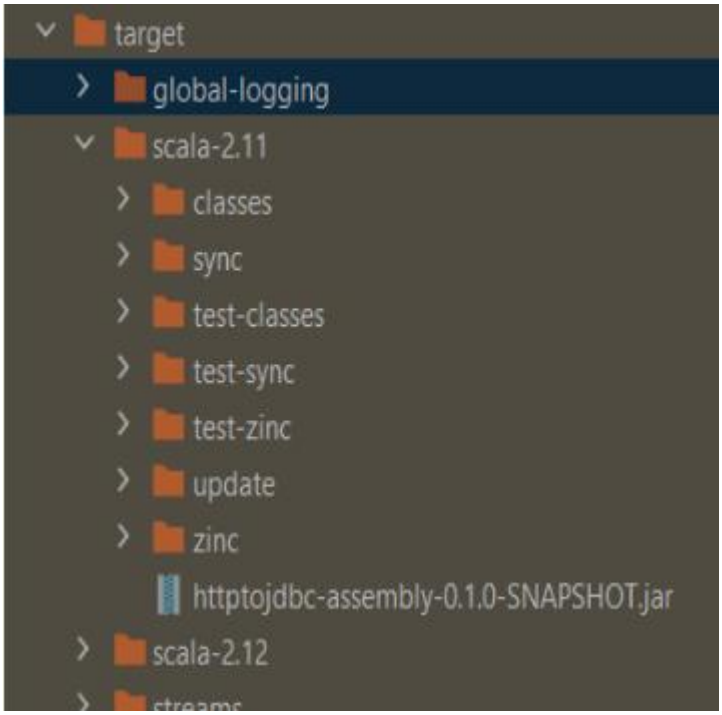
- Run the code by using test command.

```
[pool-67-thread-3] INFO org.apache.spark.scheduler.DAGScheduler - Job 1 finished:
[pool-67-thread-3] INFO com.modak.Main$ - Json parsing completed successfully
[info] Test:
[info] Run completed in 28 seconds, 462 milliseconds.
[info] Total number of tests run: 0
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 0, failed 0, canceled 0, ignored 0, pending 0
[info] No tests were executed.
[success] Total time: 31 s, completed Jun 12, 2022 3:10:30 PM
sbt:httptojdbc>
```

Built jars :

- Used assembly command to built jars.

```
[Spark Context Cleaner] INFO org.apache.spark.ContextCleaner - Cleaned accumulator 36
[Spark Context Cleaner] INFO org.apache.spark.ContextCleaner - Cleaned accumulator 46
[Spark Context Cleaner] INFO org.apache.spark.ContextCleaner - Cleaned accumulator 50
[Spark Context Cleaner] INFO org.apache.spark.ContextCleaner - Cleaned accumulator 41
[Spark Context Cleaner] INFO org.apache.spark.ContextCleaner - Cleaned accumulator 48
[Spark Context Cleaner] INFO org.apache.spark.ContextCleaner - Cleaned accumulator 34
[info] Strategy 'discard' was applied to 26 files (Run the task at debug level to see details)
[info] Strategy 'first' was applied to a file (Run the task at debug level to see details)
[warn] Ignored unknown package option FixedTimestamp(Some(1262304000000))
[success] Total time: 1036 s (17:16), completed Jun 12, 2022 3:34:11 PM
```



- Clean command is used to remove the built jars.

Executed jar using spark submit :

- The spark-submit command is a utility to run or submit a Spark or PySpark application program (or job) to the cluster by specifying options and configurations.

- spark-submit --packages "com.github.music-of-the-ainur:almaren-framework_2.11:0.9.2-2.4,org.apache.spark:spark-hive_2.11:2.4.7,org.postgresql:postgresql:42.3.3" --class com.modak.Main target/scala-2.11/httptojdbc-assembly-0.1.0-SNAPSHOT.jar ICB7CiAgInRhcmdldCI6ewogICJ1cmwiOiAiamRiYzpwb3N0Z3Jlc3FsOi8vdzMudHJhaW5pbmc1Lm1vZGFrLmNvbTo1NDMyL3RyYWluaW5nIiwKICAidXNlcm5hbWUiOiAibXQ0MDIwIiwKICAicGFzc3dvcmQiOiAibXQ0MDIwQG0wMnkyMiIsCiAgImRyaXZlciI6ICJvcmcucG9zdGdyZXNxbC5Ecml2ZXIiLAogICJ0YWJsZU5hbWUiOiAiVXM2X3RhYmxlMSJ9CiB9
- Jar was executed using spark submit command.
- Table was created in the database.

select * from Us6_table1 | Enter a SQL expression to filter results (u

	Make_ID	Make_Name
1	440	ASTON MARTIN
2	441	TESLA
3	442	JAGUAR
4	443	MASERATI
5	444	LAND ROVER
6	445	ROLLS ROYCE
7	446	EBR
8	447	JIALING
9	448	TOYOTA
10	449	MERCEDES-BENZ
11	450	FREIGHTLINER
12	451	FULMER FABRICATIONS
13	452	BMW

User story: 5

Title: Parse the JSON using deserializer, Quenya DSL and generate custom DSL's to save the data to the target

JSON:


```
{
  "ingestionConfig": [
  {
      "config": {
        "processId": 5,
        "dataplaceId": 13,
        "schemaId": 5,
        "tableId": 100001,
        "sourceType": "postgres",
        "datamovementId": 67,
        "estimatedCount": 1000000,
        "sourceCredential": {
          "credentialId": 181,
          "credentialTypeId": 1
        },
        "targetCredential": null,
        "columnsType": {
          "serial_test": "integer" ,
          "small_test" :"smallint" ,
```

"bigint_test" :"bigint" ,

"bool_test" :"boolean" ,

"bytea_test": "bytea" ,

"char_test" :"character" ,

"varchar_test": "character varying" ,

"text_test" :"text" ,

"floatn_test" :"double precision" ,

"real_test" :"real" ,

"numeric_test" :"numeric",

"date_type" :"date" ,

"time_type" :"time without time zone" ,

"timestamp_type" :" timestamp without time zone" ,

"timestampz_type" :"time with time zone" ,

"interval_type" :"interval" ,

"array_test" :"ARRAY" ,

"json_type" :"json" ,

"jsonb_type" :"jsonb" ,

"uuid_test" :"uuid" ,

"cidr_test" :"cidr" ,

"tsquery_test" :"tsquery" ,

"xml_test" :"xml" ,

"mood_plane" :"point" ,

```
    "infinite_line" :"line" ,
    "finite_line" :"lseg" ,
    "rectangular_type" : "box" ,
    "open_closed_type" : "path" ,
    "data_size" : "polygon" ,
    "rotation" : "circle"
  }
},
"source": {
  "Jdbc": {
    "url": "jdbc:postgresql://192.168.1.17:5432/nabu",
    "table": "chembl26.postgres_datatype_v2",
    "driver": "org.postgresql.Driver"
  }
},
"target": {
  "Hive": {
    "info": {
      "partition": null,
      "format": "parquet",
      "saveMode": "overwrite"
    },
```

"table": "postgres_alldatatypes_hive",

"path":
"s3a://cdpmodakbucket/cdpdevenv/data/warehouse/tablespace/external/hive/test"

}

}

}

],

"authInfo": {

"authorizationToken": "eyJhbGciYmBXNJGTgwzPNmm0WetR2g",

"endPoint": "http://192.168.1.27/fireshots/credentialsFetchWebService"

},

"verificationThreshold":0.1

}

Description:

Take the input JSON as Base64 encoded string argument to the code.

Form a dataframe out of the Base64 decoded JSON string and generate the schema.

Print the DSL using the QuenyaDSL.

Design individual(custom) DSLs from the parent DSL for “source” and “target” and change the key names by providing alias of it as per your choice respectively.

Get the dataframe for each of the custom DSL and save it to the target using targetJdbc component of Almaren framework by iterating over each of the dataframe (source and target).

Implement Scala logging and functional error handling in Scala.

Acceptance Criteria:

Scala code should be functional with optimized approach.

Table name in the target should be unique

Write to the target using Almaren component.

Target JDBC can be PostgreSQL, Oracle, MySQL etc.

Code should be packaged as JAR and run using spark-submit.

Example:

Input JSON:

```
{
“source”:
{
“url”: “http://xyz:123”,
“table”: “table_1”,
“driver”: “driver_name”
},
“target”:
{
“partition”: “123”,
“format”: “any format”,
“savemode”: “overwrite”
}
}
```

Output:

table_df1

URLGeneratedTargetTable DriverName

http://xyz:123 table_1 driver_name

key -> column name (alias)

“url” -> “URLGenerated”

“table” -> “TargetTable”

“driver” -> “DriverName”

table_df2

NumOfPartitions FormatSaveModeOption

123 any format overwrite

“partition” -> NumOfPartitions

“format” -> Format

“savemode” -> SaveModeOption

build.sbt :

```
ThisBuild / name := "test"
ThisBuild / organization := "com.modak"

lazy val scala211 = "2.11.12"
lazy val scala212 = "2.12.15"

crossScalaVersions := Seq(scala211, scala212)
ThisBuild / scalaVersion := scala212
```

```
val sparkVersion = "2.4.7"
val circeVersion = "0.12.0-M3"

libraryDependencies ++= Seq(
  "org.apache.spark" %% "spark-core" % sparkVersion % "provided",
  "org.apache.spark" %% "spark-sql" % sparkVersion % "provided",
  "org.apache.spark" %% "spark-hive" % sparkVersion % "provided",
  "io.circe" %% "circe-core" % circeVersion,
  "io.circe" %% "circe-generic" % circeVersion,
  "io.circe" %% "circe-parser" % circeVersion,
  "com.typesafe.scala-logging" %% "scala-logging" % "3.9.2",
  "org.scalatest" %% "scalatest" % "3.0.5" % "test",
  "com.github.music-of-the-ainur" %% "almaren-framework" % "0.9.3-2.4" % "provided",
  "org.postgresql" % "postgresql" % "42.3.3",
  "org.slf4j" % "slf4j-api" % "1.7.36",
  "org.slf4j" % "slf4j-simple" % "1.7.36"
)

assemblyMergeStrategy in assembly := {
  case PathList("META-INF", xs@_*) => MergeStrategy.discard
  case x => MergeStrategy.first
}
```

Explanation :

- **Syntax to add dependencies:** libraryDependencies ++= Seq(groupID%%artifactID%version)
- **"provided"** indicates that "it expects respective dependency to be mentioned at the runtime".
- Circe is a Scala library that simplifies working with JSON, allowing us to easily decode a JSON string into a Scala object or convert a Scala object to JSON.
- Three dependencies were added to install this Circe library.
- Logging is used to maintain the logs of an application. It is very much required to monitor our application.
- Dependencies were added for logging the output.
- Simple Logging Facade for Java (abbreviated SLF4J) acts as a facade for different logging frameworks (e.g., java.util.logging, logback, Log4j). It offers a generic API, making the logging independent of the actual implementation
- "org.scalatest" %% "scalatest" % "3.0.5" % "test"
    - **Scalatest** is a testing library to test Scala code by running the test cases.
    - **"test"** indicates that the dependency is limited to the test class itself.

Plugins.sbt :

```
addSbtPlugin("com.eed3si9n" % "sbt-assembly" % "0.14.10")
```

Test class :

```
package com.modak
import org.scalatest._

class Test extends FunSuite with BeforeAndAfter {
  //Json String to parse
  val plainString = """ {
                |  "ingestionConfig": [
```

```
| {
|   "config": {
|     "processId": 5,
|     "dataplaceId": 13,
|     "schemaId": 5,
|     "tableId": 100001,
|     "sourceType": "postgres",
|     "datamovementId": 67,
|     "estimatedCount": 1000000,
|     "sourceCredential": {
|       "credentialId": 181,
|       "credentialTypeId": 1
|     },
|     "targetCredential": null,
|     "columnsType": {
|       "serial_test": "integer" ,
|       "small_test" :"smallint" ,
|       "bigint_test" :"bigint" ,
|       "bool_test" :"boolean" ,
|       "bytea_test": "bytea" ,
|       "char_test" :"character" ,
|       "varchar_test": "character varying" ,
|       "text_test" :"text" ,
|       "floatn_test" :"double precision" ,
|       "real_test" :"real" ,
|       "numeric_test" :"numeric",
|       "date_type" :"date" ,
|       "time_type" :"time without time zone" ,
|       "timestamp_type" :" timestamp without time zone" ,
```

```
|         "timestampz_type" :"time with time zone" ,
|         "interval_type" :"interval" ,
|         "array_test" :"ARRAY" ,
|         "json_type" :"json" ,
|         "jsonb_type" :"jsonb" ,
|         "uuid_test" :"uuid" ,
|         "cidr_test" :"cidr" ,
|         "tsquery_test" :"tsquery" ,
|         "xml_test" :"xml" ,
|         "mood_plane" :"point" ,
|         "infinite_line" :"line" ,
|         "finite_line" :"lseg" ,
|         "rectangular_type" : "box" ,
|         "open_closed_type" : "path" ,
|         "data_size" : "polygon" ,
|         "rotation" : "circle"
|       }
|     },
|     "source": {
|       "Jdbc": {
|         "url": "jdbc:postgresql://192.168.1.17:5432/nabu",
|         "table": "chembl26.postgres_datatype_v2",
|         "driver": "org.postgresql.Driver"
|       }
|     },
|     "target": {
|       "Hive": {
|         "info": {
|           "partition": null,
```

```
|          "format": "parquet",
|          "saveMode": "overwrite"
|        },
|        "table": "postgres_alldatatypes_hive",
|        "path":
"s3a://cdpmodakbucket/cdpdevenv/data/warehouse/tablespace/external/hive/test"
|       }
|     }
|    }
| ],
| "authInfo": {
|   "authorizationToken": "eyJhbGciYmBXNJGTgwzPNmm0WetR2g",
|   "endPoint": "http://192.168.1.27/fireshots/credentialsFetchWebService"
| },
|"verificationThreshold":0.1
|}
| """.stripMargin
val plainString2="""" {
|  "target":{
|  "url": "jdbc:postgresql://w3.training5.modak.com:5432/training",
|  "username": "mt4020",
|  "password": "mt4020@m02y22",
|  "driver": "org.postgresql.Driver",
|  "tableName1": "Us5_table1",
|  "tableName2": "Us5_table2"}
| } """.stripMargin
//encoding the plainString1
val encodedString1 = java.util.Base64.getEncoder.encodeToString(plainString.getBytes())
//encoding the plainString2
```

```
  val encodedString2= java.util.Base64.getEncoder.encodeToString(plainString2.getBytes())
  println(encodedString1)
  println(encodedString2)
  //calling main method  which is in  Main class
  Main.main(Array[String](encodedString1,encodedString2))

}
```

Explanation :

- Base64 is a binary-to-text encoding scheme that represents binary data in a printable ASCII string format.
- getEncoder : It returns a Base64.Encoder that encodes using the Basic type base64 encoding scheme.
- Basic: This is the standard Base64 encoding defined in RFC 4648. The output contains characters from the set A-Z, a-z, 0-9, + and /. The decoder rejects data that contains characters outside this set.
- The encoded String passed to the main method in Main class.

Types.scala :

```
package com.modak

case class Target_1(url : String,username : String,password : String,driver : String, tableName1 : String, tableName2 : String)
case class JSON(target : Target_1)
```

Explanation :

- Case classes were designed that matches the fields of the parsed JSON string.
- Case class JSON was created with parameters target ,target with datatype assumed as Target_1.
- Target_1 has parameters url, username, password, driver and tableName1 and tableName2 datatype as String.

Main class :

```
package com.modak

import com.typesafe.scalalogging.LazyLogging
import io.circe.generic.auto._
import io.circe.parser.decode
import com.github.music.of.the.ainur.almaren.builder.Core.Implicit
import com.github.music.of.the.ainur.almaren.Almaren
import com.github.music.of.the.ainur.quenya.QuenyaDSL
import scala.util.{Failure, Success, Try}
import org.apache.spark.sql.{DataFrame, SaveMode, SparkSession}
object Main extends LazyLogging {
  val almaren = Almaren("App Name")
  val spark = almaren.spark.master("local[*]").config("spark.sql.shuffle.partitions",
"1").getOrCreate()

    import spark.implicits._

  def main(args: Array[String]): Unit = {

    Try {
      //Getting the encoded String from the command line arguments
```

```
    val encodedString1 = args(0)
    val encodedString2=args(1);
    logger.info(s"Encoded Json string :  $encodedString1")
    val decodedString1 = decodeBase64String(encodedString1)
    logger.info(s"Encoded Json string :  $encodedString2")
    val decodedString2 = decodeBase64String(encodedString2)
    logger.info(s"Decoded Json string :  $decodedString2")
    val df=Seq(s"""${decodedString1}""").toDF("Body")
      val dfForDeserialization = almaren.builder.sourceDataFrame(df).deserializer("JSON",
"Body", None).batch
      dfForDeserialization.show(false)
      val quenyaDsl = QuenyaDSL
      quenyaDsl.printDsl(dfForDeserialization)
    //parsing the json by calling JsonParser method
    val parsedJson2 = jsonParser(decodedString2)
    //getting database information from the parsedJson
    val url=parsedJson2.target.url
    val username=parsedJson2.target.username
    val password=parsedJson2.target.password
    val driver=parsedJson2.target.driver
    val table1=parsedJson2.target.tableName1
    val table2=parsedJson2.target.tableName2
    //dsl for getting source fields
    val dsl1="""ingestionConfig@ingestionConfig
        ingestionConfig.source.Jdbc.url$url:StringType
        ingestionConfig.source.Jdbc.table$Table:StringType
          ingestionConfig.source.Jdbc.driver$Driver:StringType"""
      //dsl for getting target fields
    val dsl2="""ingestionConfig@ingestionConfig
```

```
            ingestionConfig.target.Hive.info.partition$partition:StringType
            ingestionConfig.target.Hive.info.format$format:StringType
            ingestionConfig.target.Hive.info.saveMode$saveMode:StringType
            ingestionConfig.target.Hive.table$table:StringType
            ingestionConfig.target.Hive.path$path:StringType"""
    //storing the required fields in the database of source fields
    val
df1=almaren.builder.sourceDataFrame(df).deserializer("JSON","Body",None).dsl(dsl1).targetJdb
c(s"${url}", s"${driver}", s"${table1}", SaveMode.Append, Some(s"${username}"),
Some(s"${password}")).batch
    //storing the required fields in the database of target fields
    val
df2=almaren.builder.sourceDataFrame(df).deserializer("JSON","Body",None).dsl(dsl2).targetJdb
c(s"${url}", s"${driver}", s"${table2}", SaveMode.Append, Some(s"${username}"),
Some(s"${password}")).batch
  }
  match {
    case Success(j) =>
      logger.info("Json parsing c" +
        "ompleted successfully")
    case Failure(f) =>
      logger.error(s"Json Parsing failed with the exception : ${f.getLocalizedMessage}")
      throw f
  }

  def decodeBase64String(str: String): String = {
    new String(java.util.Base64.getDecoder.decode(str))
  }
  def jsonParser(str: String): JSON = {
```

```
    logger.info("Started Json parsing")
    decode[JSON](str) match {
      case Right(json) => json
      case Left(exception) =>
        logger.error(s"Json Parsing Failed with Exception ${exception.getLocalizedMessage}")
        logger.error("Invalid Input JSON Provided")
        throw exception
    }
  }

 }
}
```

Explanation of Main class :

- Imported the required libraries.
- Encoded String was decoded using decode method of Base64 schema.
- As a result, we get either a ParsingError or a Json object. We'll then use the match statement to distinguish between the returned values .
- If it is right json, it returns json object or it gives exception .
- Almaren object and Spark session was created.
- Deserializer : Deserialize the following types XML, JSON and Avro to Spark DataFrame.
- Quenya DSL(Domain Specific Language) is a language that simplifies the task to parser complex semi-structured data.
- **printDsl :** can generate and print a DSL based on a DataFrame
- TargetJdbc component of Almaren framework was used to store information into the tables in the database.
- **Batch** executes the almaren tree returning a Dataframe.

Compiled the code :

- The Scala version was changed to 2.11.12.

```
sbt:quenyadsl> ++2.11.12
[info] Setting Scala version to 2.11.12 on 1 projects.
[info] Reapplying settings...
[info] set current project to quenyadsl (in build file:/mnt/c/workArea/spark/QuenyaDSL/)
sbt:quenyadsl>
```

- Compile the code using compile command.

```
sbt:quenyadsl> compile
[success] Total time: 0 s, completed Jun 12, 2022 5:27:46 PM
sbt:quenyadsl>
```

- Run the code by using test command.

```
[task-result-getter-0] INFO org.apache.spark.scheduler.TaskSchedulerImpl - Removed Task
[dag-scheduler-event-loop] INFO org.apache.spark.scheduler.DAGScheduler - ResultStage 4
[pool-87-thread-4] INFO org.apache.spark.scheduler.DAGScheduler - Job 4 finished: save
[pool-87-thread-4] INFO com.modak.Main$ - Json parsing completed successfully
[info] Test:
[info] Run completed in 33 seconds, 432 milliseconds.
[info] Total number of tests run: 0
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 0, failed 0, canceled 0, ignored 0, pending 0
[info] No tests were executed.
```

- Quenyadsl printed in the following format.

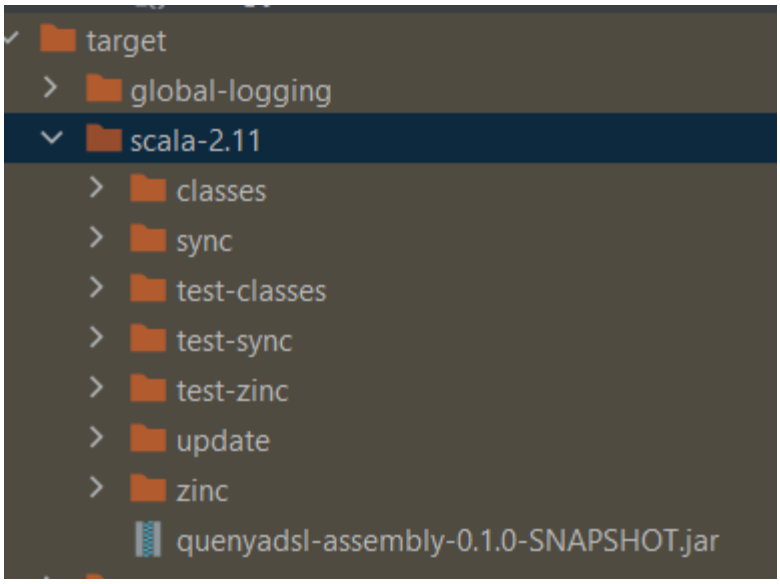
```
ingestionConfig@ingestionConfig
        ingestionConfig.config.columnsType.array_test$array_test:StringType
        ingestionConfig.config.columnsType.bigint_test$bigint_test:StringType
        ingestionConfig.config.columnsType.bool_test$bool_test:StringType
        ingestionConfig.config.columnsType.bytea_test$bytea_test:StringType
        ingestionConfig.config.columnsType.char_test$char_test:StringType
        ingestionConfig.config.columnsType.cidr_test$cidr_test:StringType
        ingestionConfig.config.columnsType.data_size$data_size:StringType
        ingestionConfig.config.columnsType.date_type$date_type:StringType
        ingestionConfig.config.columnsType.finite_line$finite_line:StringType
        ingestionConfig.config.columnsType.floatn_test$floatn_test:StringType
        ingestionConfig.config.columnsType.infinite_line$infinite_line:StringType
        ingestionConfig.config.columnsType.interval_type$interval_type:StringType
        ingestionConfig.config.columnsType.json_type$json_type:StringType
        ingestionConfig.config.columnsType.jsonb_type$jsonb_type:StringType
        ingestionConfig.config.columnsType.mood_plane$mood_plane:StringType
        ingestionConfig.config.columnsType.numeric_test$numeric_test:StringType
        ingestionConfig.config.columnsType.open_closed_type$open_closed_type:StringType
        ingestionConfig.config.columnsType.real_test$real_test:StringType
        ingestionConfig.config.columnsType.rectangular_type$rectangular_type:StringType
        ingestionConfig.config.columnsType.rotation$rotation:StringType
        ingestionConfig.config.columnsType.serial_test$serial_test:StringType
        ingestionConfig.config.columnsType.small_test$small_test:StringType
        ingestionConfig.config.columnsType.text_test$text_test:StringType
```

Built jars :

- Used assembly command to built jars.

```
[dag-scheduler-event-loop] INFO org.apache.spark.scheduler.DAGScheduler - ResultStage 4 (save
[pool-107-thread-3] INFO org.apache.spark.scheduler.DAGScheduler - Job 4 finished: save at Tar
[pool-107-thread-3] INFO com.modak.Main$ - Json parsing completed successfully
[info] Test:
[info] Run completed in 17 seconds, 867 milliseconds.
[info] Total number of tests run: 0
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 0, failed 0, canceled 0, ignored 0, pending 0
[info] No tests were executed.


  | => quenyadsl / assembly 1s
```



- Clean command is used to remove the built jars.

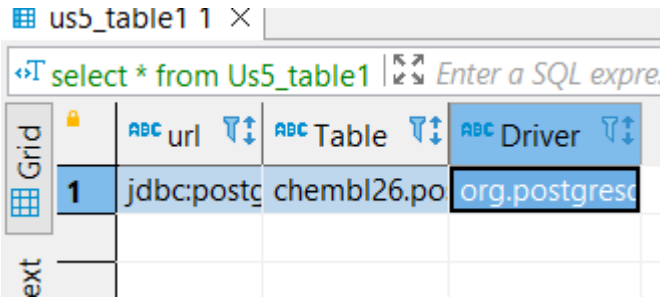
Executed jar using spark submit :

- The spark-submit command is a utility to run or submit a Spark or PySpark application program (or job) to the cluster by specifying options and configurations.
- spark-submit --packages "com.github.music-of-the-ainur:almaren-framework_2.11:0.9.2-2.4,org.apache.spark:spark-hive_2.11:2.4.7" --class com.modak.Main target/scala-2.11/quenyadsl-assembly-0.1.0-SNAPSHOT.jar
IHsKICAiaW5nZXN0aW9uQ29uZmlnIjogWwogIHsKICAgICAgImNvbmZpZyI6IHsKICAgICAgI
CAicHJvY2Vzc0lkIjogNSwKICAgICAgICAiZGF0YXBsYWNlSWQiOiAxMywKICAgICAgICAic2N
oZW1hSWQiOiA1LAogICAgICAgICJ0YWJsZUlkIjogMTAwMDAxLAogICAgICAgICJzb3VyY2V
UeXBlIjogInBvc3RncmVzIiwKICAgICAgICAiZGF0YW1vdmVtZW50SWQiOiA2NywKICAgICA
gICAiZXN0aW1hdGVkQ291bnQiOiAxMDAwMDAwLAogICAgICAgICJzb3VyY2VDcmVkZW5
0aWFsIjogewogICAgICAgICAgImNyZWRlbnRpYWxJZCI6IDE4MSwKICAgICAgICAgICJjcmVk
ZW50aWFsVHlwZUlkIjogMQogICAgICAgIH0sCiAgICAgICAgInRhcmdldENyZWRlbnRpYWwi
OiBudWxsLAogICAgICAgICJjb2x1bW5zVHlwZSI6IHsKICAgICAgICAgICJzZXJpYWxfdGVzdCI6
ICJpbnRlZ2VyIiAsCiAgICAgICAgICAic21hbGxfdGVzdCIgOiJzbWFsbGludCIgLAogICAgICAgIC

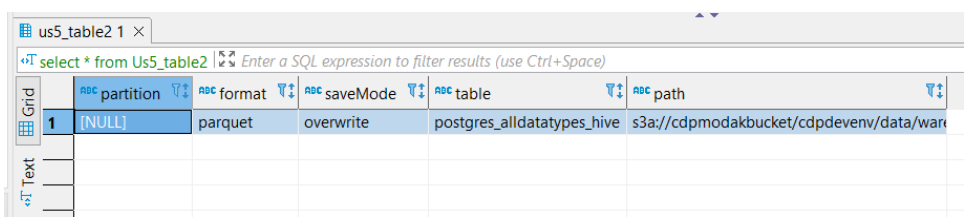
AgImJpZ2ludF90ZXN0IiA6ImJpZ2ludCIgLAogICAgICAgICAgImJvb2xfdGVzdCIgOiJib29sZWF
uIiAsCiAgICAgICAgICAiYnl0ZWFfdGVzdCI6ICJieXRlYSIgLAogICAgICAgICAgImNoYXJfdGVzd
CIgOiJjaGFyYWN0ZXIiICwKICAgICAgICAgICJ2YXJjaGFyX3Rlc3QiOiAiY2hhcmFjdGVyIHZhcnl
pbmciICwKICAgICAgICAgICJ0ZXh0X3Rlc3QiIDoidGV4dCIgLAogICAgICAgICAgImZsb2F0bl9
0ZXN0IiA6ImRvdWJsZSBwcmVjaXNpb24iICwKICAgICAgICAgICJyZWFsX3Rlc3QiIDoicmVhb
CIgLAogICAgICAgICAgIm51bWVyaWNfdGVzdCIgOiJudW1lcmljIiwKICAgICAgICAgICJkYXRl
X3R5cGUiIDoiZGF0ZSIgLAogICAgICAgICAgInRpbWVfdHlwZSIgOiJ0aW1lIHdpdGhvdXQgdG
ltZSB6b25lIiAsCiAgICAgICAgICAidGltZXN0YW1wX3R5cGUiIDoiIHRpbWVzdGFtcCB3aXRob
3V0IHRpbWUgem9uZSIgLAogICAgICAgICAgInRpbWVzdGFtcHfdHlwZSIgOiJ0aW1lIHdpd
GggdGltZSB6b25lIiAsCiAgICAgICAgICAiaW50ZXJ2YWxfdHlwZSIgOiJpbnRlcnZhbCIgLAogIC
AgICAgICAgImFycmF5X3Rlc3QiIDoiQVJSQVkiICwKICAgICAgICAgICJqc29uX3R5cGUiIDoian
NvbiIgLAogICAgICAgICAgImpzb25iX3R5cGUiIDoianNvbmIiICwKICAgICAgICAgICJ1dWlkX3
Rlc3QiIDoidXVpZCIgLAogICAgICAgICAgImNpZHJfdGVzdCIgOiJjaWRyIiAsCiAgICAgICAgICAi
dHNxdWVyeV90ZXN0IiA6InRzcXVlcnkiICwKICAgICAgICAgICJ4bWxfdGVzdCIgOiJ4bWwiIC
wKICAgICAgICAgICJtb29kX3BsYW5lIiA6InBvaW50IiAsCiAgICAgICAgICAiaW5maW5pdGVf
bGluZSIgOiJsaW5lIiAsCiAgICAgICAgICAiZmluaXRlX2xpbmUiIDoibHNlZyIgLAogICAgICAgICAgIC
AgInJlY3Rhbmd1bGFyX3R5cGUiIDogImJveCIgLAogICAgICAgICAgIm9wZW5fY2xvc2VkX3R
5cGUiIDogInBhdGgiICwKICAgICAgICAgICJkYXRhX3NpemUiIDogInBvbHlnb24iICwKICAgICAgIC
AgICAgICJyb3RhdGlvbiIgOiAiY2lyY2xlIgogICAgICAgIH0KICAgICAgfSwKICAgICAgInNvdXJjZS
I6IHsKICAgICAgICAiSmRiYyI6IHsKICAgICAgICAgICJ1cmwiOiAiamRiYzpwb3N0Z3Jlc3FsOi8v
MTkyLjE2OC4xLjE3OjU0MzIvbmFidSIsCiAgICAgICAgICAidGFibGUiOiAiY2hlbWJsMjYucG9z
dGdyZXNfZGF0YXR5cGVfdjIiLAogICAgICAgICAgImRyaXZlciI6ICJvcmcucG9zdGdyZXNxbC5E
cml2ZXIiCiAgICAgICAgfQogICAgICB9LAogICAgICAidGFyZ2V0IjogewogICAgICAgICJlaXZlIjo
gewogICAgICAgICAgImluZm8iOiB7CiAgICAgICAgICAgICJwYXJ0aXRpb24iOiBudWxsLAogIC
AgICAgICAgICAiZm9ybWF0IjogInBhcnF1ZXQiLAogICAgICAgICAgICAic2F2ZU1vZGUiOiAib3
ZlcndyaXRlIgogICAgICAgICAgfSwKICAgICAgICAgICJ0YWJsZSI6ICJwb3N0Z3Jlc19hbGxkYXR
hdHlwZXNfaGl2ZSIsCiAgICAgICAgICAicGF0aCI6ICJzM2E6Ly9jZHBtb2Rha2J1Y2tldC9jZHBk
ZXZlbnYvZGF0YS93YXJlaG91c2UvdGFibGVzcGFjZS9leHRlcm5hbC9oaXZlL3Rlc3QiCiAgICAg
ICAgfQogICAgICB9CiAgICAgfQogIF0sCiAgImF1dGhJbmZvIjogewogICAgImF1dGhvcml6YXR

pb25Ub2tlbiI6ICJleUpoYkdjaVltQlhOSkdUZ3d6UE5tbTBXZXRSMmciLAogICAgImVuZFBva
W50IjogImh0dHA6Ly8xOTIuMTY4LjEuMjcvZmlyZXNob3RzL2NyZWRlbnRpYWxzRmV0Y2h
XZWJTZXJ2aWNlIgogIH0sCiJ2ZXJpZmljYXRpb25UaHJlc2hvbGQiOjAuMQp9CiA=
IHsKICAidGFyZ2V0Ijp7CiAgInVybCI6ICJqZGJjOnBvc3RncmVzcWw6Ly93My50cmFpbmluZz
UubW9kYWsuY29tOjU0MzIvdHJhaW5pbmciLAogICJ1c2VybmFtZSI6ICJtdDQwMjAiLAogI
==

- Jar was executed using spark submit command
- Data in the table Us5_table1



- Data in the table Us5_table1



User story - 6

Title: Fetch two different files with something in common, either from same location or different location using source components of the Almaren. Perform a join on it and do group by on the output of the join and cache it into the JDBC. (https://github.com/music-of-the-ainur/almaren-framework#example-3)

Description:

Take any file format as an input dataframe and convert it into a table or you can also perform further steps on a dataframe.

Perform join over the columns which can be used as a primary key.

Perform group-by over the joined table.

Cache the result of the group-by at any target component location available on the Almaren framework.

Implement the Scala-logging and Functional-Error-Handling in Scala.

Acceptance Criteria:

Scala code should be functional with optimized approach.

Read and write from source to target using the Almaren component.

Cached table must have a descriptive name.

Code should be packaged as JAR and run it using proper spark-submit command.

Code explanantion :

build.sbt :

```
ThisBuild / name := "test"
ThisBuild / organization := "com.modak"

lazy val scala211 = "2.11.12"
lazy val scala212 = "2.12.15"
```

```
crossScalaVersions := Seq(scala211, scala212)
ThisBuild / scalaVersion := scala212

val sparkVersion = "2.4.7"
val circeVersion = "0.12.0-M3"

libraryDependencies ++= Seq(
  "org.apache.spark" %% "spark-core" % sparkVersion % "provided",
  "org.apache.spark" %% "spark-sql" % sparkVersion % "provided",
  "org.apache.spark" %% "spark-hive" % sparkVersion % "provided",
  "io.circe" %% "circe-core" % circeVersion,
  "io.circe" %% "circe-generic" % circeVersion,
  "io.circe" %% "circe-parser" % circeVersion,
  "com.typesafe.scala-logging" %% "scala-logging" % "3.9.2",
  "org.scalatest" %% "scalatest" % "3.0.5" % "test",
  "com.github.music-of-the-ainur" %% "almaren-framework" % "0.9.3-2.4" % "provided",
  "org.postgresql" % "postgresql" % "42.3.3",
  "org.slf4j" % "slf4j-api" % "1.7.36",
  "org.slf4j" % "slf4j-simple" % "1.7.36"
)


assemblyMergeStrategy in assembly := {
  case PathList("META-INF", xs@_*) => MergeStrategy.discard
  case x => MergeStrategy.first
}
```

Explanation :

- Syntax to add dependencies: libraryDependencies ++= Seq(groupID%%artifactID%version)

- "provided" indicates that "it expects respective dependency to be mentioned at the runtime".
- Circe is a Scala library that simplifies working with JSON, allowing us to easily decode a JSON string into a Scala object or convert a Scala object to JSON.
- Three dependencies were added to install this Circe library.
- Logging is used to maintain the logs of an application. It is very much required to monitor our application.
- Dependencies were added for logging the output.
- Simple Logging Facade for Java (abbreviated SLF4J) acts as a facade for different logging frameworks (e.g., java.util.logging, logback, Log4j). It offers a generic API, making the logging independent of the actual implementation
- "org.scalatest" %% "scalatest" % "3.0.5" % "test"
    - Scalatest is a testing library to test Scala code by running the test cases.
    - "test" indicates that the dependency is limited to the test class itself.

Plugins.sbt :

```
addSbtPlugin("com.eed3si9n" % "sbt-assembly" % "0.14.10")
```

Test class :

```
package com.modak
import org.scalatest._

class Test extends FunSuite with BeforeAndAfter {
 //Json String to parse
```

```
 val plainString = """  {
                |  "source":{
                |     "format" : "csv",
                |     "path1" : "/mnt/c/WorkArea/Spark/JoinTables/Files/Employee1.csv",
                |     "path2" : "/mnt/c/WorkArea/Spark/JoinTables/Files/salesman1.csv",
                |     "header" : "true"
                |},
                |  "target":{
                |  "url": "jdbc:postgresql://w3.training5.modak.com:5432/training",
                |  "username": "mt4020",
                |  "password": "mt4020@m02y22",
                |  "driver": "org.postgresql.Driver",
                |  "tableName": "Us6_table1"}
                | }""".stripMargin
//encoding the Json String
 val encodedString = java.util.Base64.getEncoder.encodeToString(plainString.getBytes())
 println(encodedString)
//calling main method  which is in  Main class
 Main.main(Array[String](encodedString))

}
```

Explanation :

- Base64 is a binary-to-text encoding scheme that represents binary data in a printable ASCII string format.

- getEncoder : It returns a Base64.Encoder that encodes using the Basic type base64 encoding scheme.
- Basic: This is the standard Base64 encoding defined in RFC 4648. The output contains characters from the set A-Z, a-z, 0-9, + and /. The decoder rejects data that contains characters outside this set.
- The encoded String passed to the main method in Main class.

Types.scala :

```
package com.modak
case class Target_1(url : String,username : String,password : String,driver : String, tableName : String)
case class Source_1(format : String, path1 : String , path2 : String, header : String)
case class JSON(source : Source_1 , target : Target_1)
```

Explanation :

- Case classes were designed that matches the fields of the parsed JSON string.
- Case class JSON was created with parameters target and source.
- target with datatype assumed as Target_1 and source datatype assumed as Source_1.
- Source_1 has parameters format datatype String and path datatype String and header.
- Target_1 has parameters url, username, password, driver and tableName datatype as String.

Main class :

```
package com.modak

import com.github.music.of.the.ainur.almaren.builder.Core.Implicit
import com.github.music.of.the.ainur.almaren.Almaren
import com.typesafe.scalalogging.LazyLogging

import scala.util.{Failure, Success, Try}
import io.circe.parser.decode
import io.circe.generic.auto._
import org.apache.spark.sql.SaveMode

object Main extends LazyLogging {

  def main(args: Array[String]): Unit = {
    val almaren = Almaren("App Name")
    val spark = almaren.spark.master("local[*]").config("spark.sql.shuffle.partitions", "1")
    Try {
      //Getting the encoded String from the command line arguments
      val encodedString = args.head
      logger.info(s"Encoded Json string :  $encodedString")
      //Decoded the String to json
      val decodedString = decodeBase64String(encodedString)
      logger.info(s"Decoded Json string :  $decodedString")
      //parsing the json by calling JsonParser method
      val parsedJson = jsonParser(decodedString)
      //Getting the required source fields from parsedJson
      val format=parsedJson.source.format
      val path1=parsedJson.source.path1
      val path2=parsedJson.source.path2
      val header=parsedJson.source.header
```

```
    //Getting the required target fields from parsedJson
    val url=parsedJson.target.url
    val username=parsedJson.target.username
    val password=parsedJson.target.password
    val driver=parsedJson.target.driver
    val table=parsedJson.target.tableName
    //creating a dataframe from data of file1
    val df1= almaren.builder.sourceFile(s"${format}", s"${path1}", Map("header" ->
s"${header}", "inferSchema" -> "true")).batch
    //creating a  daatframe from data of file2
    val df2= almaren.builder.sourceFile(s"${format}", s"${path2}", Map("header" ->
s"${header}", "inferSchema" -> "true")).batch
    //from dataframe creating a table
    df1.createOrReplaceTempView("Table1")
    df2.createOrReplaceTempView("Table2")
    //joining two tables and storing results in target table in database.
    almaren.builder.sourceSql("select Table1.state,count(*) as no_of_state  from Table1  join
Table2 on Table1.id=Table2.id group by Table1.state order by no_of_state")
      .targetJdbc(s"${url}", s"${driver}", s"${table}", SaveMode.Append, Some(s"${username}"),
Some(s"${password}")).batch

  } match {
    case Success(j) =>
      logger.info("Json parsing completed successfully")
    case Failure(f) =>
      logger.error(s"Json Parsing failed with the exception : ${f.getLocalizedMessage}")
      throw f
  }
```

```
  def decodeBase64String(str: String): String = {
    new String(java.util.Base64.getDecoder.decode(str))
  }

  def jsonParser(str: String): JSON = {
    logger.info("Started Json parsing")
    decode[JSON](str) match {
      case Right(json) => json
      case Left(exception) =>
        logger.error(s"Json Parsing Failed with Exception ${exception.getLocalizedMessage}")
        logger.error("Invalid Input JSON Provided")
        throw exception
    }
  }
 }
}
```

Explanation of Main class :

- Imported the required libraries.
- Encoded String was decoded using decode method of Base64 schema.
- As a result, we get either a ParsingError or a Json object. We'll then use the match statement to distinguish between the returned values .
- If it is right json, it returns json object or it gives exception .
- Almaren object and Spark session was created.
- Read data from file1 and file2 using sourceFile component of almaren framework.
- **createOrReplaceTempView** : Creates or replaces a local temporary view with the DataFrame.

- Used sourceSql component used to execute the sql query.
- TargetJdbc component of Almaren framework was used to store information to the table in database.

Compilation of code :

- The Scala version was changed to 2.11.12.

```
sbt:jointables> ++2.11.12
[info] Setting Scala version to 2.11.12 on 1 projects.
[info] Reapplying settings...
[info] set current project to jointables (in build file:/mnt/c/WorkArea/spark/JoinTables/)
sbt:jointables>
```

- Compile the code using compile command.

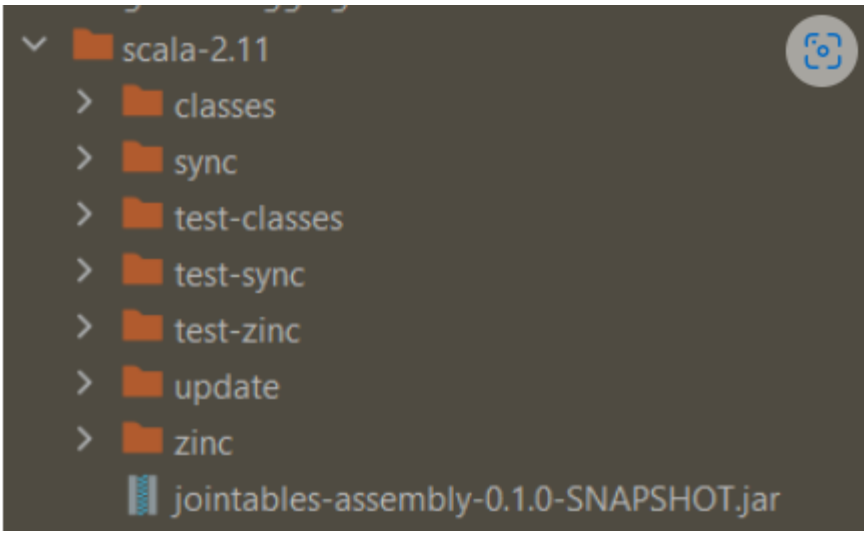
```
sbt:jointables> compile
[success] Total time: 0 s, completed Jun 12, 2022 10:36:34 PM
sbt:jointables>
```

- Run the code by using test command.

```
[dag-scheduler-event-loop] INFO org.apache.spark.scheduler.DAGScheduler - ResultStage 7 (sa
[pool-69-thread-1] INFO org.apache.spark.scheduler.DAGScheduler - Job 5 finished: save at T
[pool-69-thread-1] INFO com.modak.Main$ - Json parsing completed successfully
[info] Test:
[info] Run completed in 25 seconds, 537 milliseconds.
[info] Total number of tests run: 0
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 0, failed 0, canceled 0, ignored 0, pending 0
[info] No tests were executed.
[success] Total time: 27 s, completed Jun 12, 2022 10:37:39 PM
sbt:jointables>
```

Built jars :

- Used assembly command to built jars.



- Clean command is used to remove the built jars.

Executed jar using spark submit :

- The spark-submit command is a utility to run or submit a Spark or PySpark application program (or job) to the cluster by specifying options and configurations.
- spark-submit --packages "com.github.music-of-the-ainur:almaren-framework_2.11:0.9.2-2.4,org.apache.spark:spark-hive_2.11:2.4.7,org.postgresql:postgresql:42.3.3" --class com.modak.Main target/scala-2.11/jointables-assembly-0.1.0-SNAPSHOT.jar
ICB7CiAgInNvdXJjZSI6ewogICAgICJmb3JtYXQiIDogImNzdiIsCiAgICAgInBhdGgxIiA6ICIvbW50L2MvV29ya0FyZWEvU3BhcmsvSm9pblRhYmxlcy9GaWxlcy9FbXBsb3llZTEuY3N2IiwKICAgICAicGF0aDIiIDogIi9tbnQvYy9Xb3JrQXJlYS9TcGFyay9Kb2luVGFibGVzL0ZpbGVzL3NhbGVzbWFuMS5jc3YiLAogICAgICJoZWFkZXIiIDogInRydWUiCn0sCiAgInRhcmdldCI6ewogICJ1cmwiOiAiamRiYzpwb3N0Z3Jlc3FsOi8vdzMudHJhaW5pbmc1Lm1vZGFrLmNvbTo1NDMyL3RyYWluaW5nIiwKICAidXNlcm5hbWUiOiAibXQ0MDIwIiwKICAicGFzc3dvcmQiOiAibXQ0MDIwQG0wMnkyMiIsCiAgImRyaXZlciI6ICJvcmcucG9zdGdyZXNxbC5Ecml2ZXIiLAogICJ0YWJsZU5hbWUiOiAiVXM2X3RhYmxlMSJ9CiB9
- The content in the files having id as common column.

```
id,ename,state
5001,Sanjana,Telangana
5002,Venkata,Telangana
5005,Manal,Tamil Nadu
5003,Anaida,Kerala
5004,Nikita,Telangana
5009,Deepam,Uttar pradesh
```

```
id,name,city,commission
5001,James Hogg,New York,0.14
5002,Rakesh,Paris,0.13
5005,Pit Alex,London,0.49
5006,Mc Lyon,Rome,0.15
5007,Paul Adam,Paris,0.12
5003,Laudon Hen,San Jose,0.34
```

- "select Table1.state,count(*) as no_of_state from Table1 join Table2 on Table1.id=Table2.id group by Table1.state order by no_of_state".
- After joining the two tables based on id and applied group by on state field.

select * from Us6_table1 Enter a SQL

	state	no_of_state
1	Tamil Nadu	1
2	Kerala	1
3	Telangana	2

Userstory-7

Title: Read an excel file and store into the target file and JDBC (https://github.com/crealytics/spark-excel).

Description:

Read an excel file which contains n no. of sheets and parse all the excel sheet.

Store each sheet into the target as an individual file in the target folder in any file format and simultaneously store it into the target JDBC with different table names specifying the main excel final name with the sheet name.

Input file name: abc.xls

Sheet names: a,b,c

Target file name or table name:

abc_a.fileformat, abc_b. fileformat abc_c. fileformat

Target JDBC table names:

abc_a, abc_b, abc_c

Acceptance Criteria:

Scala code should be functional with optimized approach.

Write to the target must be done using the Almaren component wherever required.

Excel file must have minimum of 4-5 sheets.

Code should be packaged as JAR and run it using proper spark-submit command

build.sbt :

```
ThisBuild / name := "test"
ThisBuild / organization := "com.modak"

lazy val scala211 = "2.11.12"
lazy val scala212 = "2.12.15"

crossScalaVersions := Seq(scala211, scala212)
ThisBuild / scalaVersion := scala212

val sparkVersion = "2.4.7"
val circeVersion = "0.12.0-M3"
libraryDependencies ++= Seq(
  "org.apache.spark" %% "spark-core" % sparkVersion % "provided",
  "org.apache.spark" %% "spark-sql" % sparkVersion % "provided",
  "org.apache.spark" %% "spark-hive" % sparkVersion % "provided",
  "io.circe" %% "circe-core" % circeVersion,
  "io.circe" %% "circe-generic" % circeVersion,
  "io.circe" %% "circe-parser" % circeVersion,
  "org.postgresql" % "postgresql" % "42.3.3",
  "com.typesafe.scala-logging" %% "scala-logging" % "3.9.2",
  "com.github.music-of-the-ainur" %% "almaren-framework" % "0.9.3-2.4" % "provided",
  "org.scalatest" %% "scalatest" % "3.0.5" % "test",
  "org.slf4j" % "slf4j-api" % "1.7.36",
  "org.slf4j" % "slf4j-simple" % "1.7.36",
  "com.crealytics" %% "spark-excel" % "0.13.7"
)

assemblyMergeStrategy in assembly := {
  case PathList("META-INF", xs@_*) => MergeStrategy.discard
  case x => MergeStrategy.first
```

}

Explanation :

- **Syntax to add dependencies:** libraryDependencies ++= Seq(groupID%%artifactID%version)
- **"provided"** indicates that "it expects respective dependency to be mentioned at the runtime".
- Circe is a Scala library that simplifies working with JSON, allowing us to easily decode a JSON string into a Scala object or convert a Scala object to JSON.
- Three dependencies were added to install this Circe library.
- Logging is used to maintain the logs of an application. It is very much required to monitor our application.
- Dependencies were added for logging the output.
- Simple Logging Facade for Java (abbreviated SLF4J) acts as a facade for different logging frameworks (e.g., java.util.logging, logback, Log4j). It offers a generic API, making the logging independent of the actual implementation
- "org.scalatest" %% "scalatest" % "3.0.5" % "test"
    - **Scalatest** is a testing library to test Scala code by running the test cases.
    - **"test"** indicates that the dependency is limited to the test class itself.

Plugins.sbt :

addSbtPlugin("com.eed3si9n" % "sbt-assembly" % "0.14.10")

Test class :

package modak

import com.modak.Main
import org.scalatest._

```
class Test extends FunSuite with BeforeAndAfter {
  //Json String to parse
  val plainString = """ {
                |  "source":{
                |     "path" : "/mnt/c/Users/mt4020/Downloads/Spark_us7.xlsx"
                |},
                |  "target1":{
                |  "url": "jdbc:postgresql://w3.training5.modak.com:5432/training",
                |  "username": "mt4020",
                |  "password": "mt4020@m02y22",
                |  "driver": "org.postgresql.Driver"},
                |  "target2":{
                |   "path" : "/mnt/c/WorkArea/Spark/ExcelToJdbc/Files/"}
                | } """.stripMargin
  //encoding the planString.
  val encodedString = java.util.Base64.getEncoder.encodeToString(plainString.getBytes())
  println(encodedString)
  //calling main method  which is in  Main class
  Main.main(Array[String](encodedString))

}
```

Explanation :

- Base64 is a binary-to-text encoding scheme that represents binary data in a printable ASCII string format.
- getEncoder : It returns a Base64.Encoder that encodes using the Basic type base64 encoding scheme.
- Basic: This is the standard Base64 encoding defined in RFC 4648. The output contains characters from the set A-Z, a-z, 0-9, + and /. The decoder rejects data that contains characters outside this set.

- The encoded String passed to the main method in Main class.

Types.scala :

```
package com.modak

//case classes for json parsing
case class Target_2(path : String)
case class Target_1(url : String,username : String,password : String,driver : String)
case class Source_1 (path : String)
case class JSON(source : Source_1  , target1 : Target_1 , target2 : Target_2)
```

Explanation :

- Case classes were designed that matches the fields of the parsed JSON string.
- Case class JSON has parameter source ,target1 and target2.
- Source_1 has parameter path with datatype String.
- Target_1 has parameters url, username, password and driver with datatype string.
- Target_2 has parameter path of type String.

Main class :

```
package com.modak
import com.github.music.of.the.ainur.almaren.builder.Core.Implicit
import com.github.music.of.the.ainur.almaren.Almaren
import com.typesafe.scalalogging.LazyLogging
import scala.util.{Failure, Success, Try}
import io.circe.generic.auto._
import io.circe.parser.decode
import org.apache.spark.sql.SaveMode
import com.crealytics.spark.excel._
import com.crealytics.spark.excel.WorkbookReader
```

```
import com.crealytics.spark.excel._

object Main extends LazyLogging {

  def main(args: Array[String]): Unit = {
    val almaren = Almaren("App Name")
    val spark = almaren.spark.master("local[*]").config("spark.sql.shuffle.partitions",
"1").enableHiveSupport().getOrCreate()
    Try {
      //Getting the encoded String from the command line arguments
      val encodedString = args.head
      logger.info(s"Encoded Json string :  $encodedString")
      //Decoded the String to json
      val decodedString = decodeBase64String(encodedString)
      logger.info(s"Decoded Json string :  $decodedString")
      //parsing the json by calling JsonParser method
      val parsedJson = jsonParser(decodedString)
      //getting the files information from the parsed Json
      val path = parsedJson.source.path
      //getting database information from the parsed Json
      val url=parsedJson.target1.url
      val username=parsedJson.target1.username
      val password=parsedJson.target1.password
      val driver=parsedJson.target1.driver
      //getting folder path to store files.
      val patht=parsedJson.target2.path
      //getting the sheetNames from the WorkBook
      val sheetNames = WorkbookReader( Map("path" -> s"${path}"),
spark.sparkContext.hadoopConfiguration).sheetNames
```

```
    //reading the each sheets and storing it in the csv files and tables
    for(sheet <- sheetNames )
    {
      val df = spark.read.excel(header = true,inferSchema = false,  dataAddress = s"'$sheet'!A1"
,treatEmptyValuesAsNulls = true).load(s"${path}")
      val pathfile=s"${patht}"+sheet+".csv"

df.repartition(1).write.format("csv").mode(SaveMode.Overwrite).option("header","true").save(
s"${pathfile}")
      val table="Us7_"+sheet
        var df2=almaren.builder.sourceDataFrame(df).targetJdbc(s"${url}", s"${driver}",
s"${table}", SaveMode.Overwrite, Some(s"${username}"), Some(s"${password}")).batch
    }
  } match {
    case Success(j) =>
      logger.info("Json parsing completed successfully")
    case Failure(f) =>
      logger.error(s"Json Parsing failed with the exception : ${f.getLocalizedMessage}")
      throw f
  }

  def decodeBase64String(str: String): String = {
    new String(java.util.Base64.getDecoder.decode(str))
  }

  def jsonParser(str: String): JSON = {
    logger.info("Started Json parsing")
    decode[JSON](str) match {
      case Right(json) => json
```

```
      case Left(exception) =>
        logger.error(s"Json Parsing Failed with Exception ${exception.getLocalizedMessage}")
        logger.error("Invalid Input JSON Provided")
        throw exception
    }
  }
 }
}
```

Explanation of code in Main class:

- Imported the required libraries.
- Encoded String was decoded using decode method of Base64 schema.
- As a result, we get either a ParsingError or a Json object. We'll then use the match statement to distinguish between the returned values .
- If it is right json, it returns json object or it gives exception .
- Almaren object and Spark session was created.
- WorkBookReader() method gives the sheet names in the excel book.xlsx.
- Reading each excel sheets and stroing data in csv files and tables.
- TargetJdbc component of Almaren framework was used to store information into the tables in the database.

  Compiled the code :

- The Scala version was changed to 2.11.12.

```
sbt:exceltojdbc> ++2.11.12
[info] Setting Scala version to 2.11.12 on 1 projects.
[info] Reapplying settings...
[info] set current project to exceltojdbc (in build file:/mnt/c/workarea/spark/ExcelToJdbc/)
sbt:exceltojdbc>
```

- Compile the code using compile command.

```
https://repo1.maven.org/maven2/com/lihaoyi/requests_2.11/0.7.0/requests_2.11-0.7.0.jar
  100.0% [##########] 148.3 KiB (73.2 KiB / s)
[info] Fetched artifacts of
[info] compiling 2 Scala sources to /mnt/c/workArea/spark/HttpToJdbc/target/scala-2.11/classes ..
[success] Total time: 53 s, completed Jun 12, 2022 3:06:29 PM
sbt:httptojdbc>
```

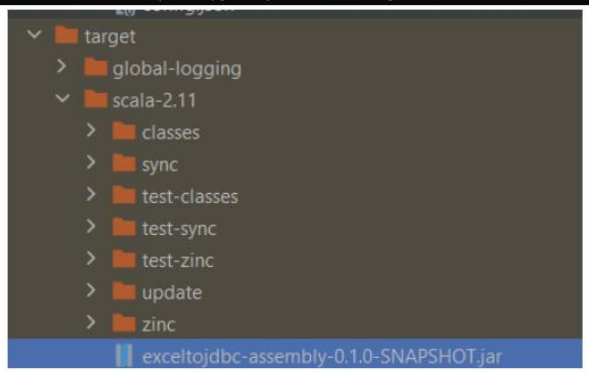
- Run the code by using test command.

```
https://repo1.maven.org/maven2/org/scala-lang/modules/scala-xml_2.11/1.3.0/scala-xml_2.11-1.3.0.jar
  100.0% [##########] 664.2 KiB (291.3 KiB / s)
https://repo1.maven.org/maven2/com/norbitltd/spoiwo_2.11/1.8.0/spoiwo_2.11-1.8.0.jar
  100.0% [##########] 845.8 KiB (368.2 KiB / s)
https://repo1.maven.org/maven2/com/crealytics/spark-excel_2.11/0.13.7/spark-excel_2.11-0.13.7.jar
  100.0% [##########] 6.3 MiB (1.7 MiB / s)
[info] Fetched artifacts of
[info] compiling 2 Scala sources to /mnt/c/workarea/spark/ExcelToJdbc/target/scala-2.11/classes ...
[success] Total time: 39 s, completed Jun 13, 2022 6:05:00 PM
```

Built jars :

- Used assembly command to built jars.

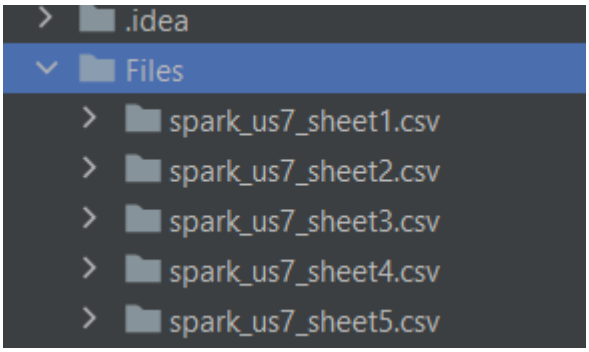
```
[Spark Context Cleaner] INFO org.apache.spark.ContextCleaner - Cleaned accumulator 36
[Spark Context Cleaner] INFO org.apache.spark.ContextCleaner - Cleaned accumulator 46
[Spark Context Cleaner] INFO org.apache.spark.ContextCleaner - Cleaned accumulator 50
[Spark Context Cleaner] INFO org.apache.spark.ContextCleaner - Cleaned accumulator 41
[Spark Context Cleaner] INFO org.apache.spark.ContextCleaner - Cleaned accumulator 48
[Spark Context Cleaner] INFO org.apache.spark.ContextCleaner - Cleaned accumulator 34
[info] Strategy 'discard' was applied to 26 files (Run the task at debug level to see details)
[info] Strategy 'first' was applied to a file (Run the task at debug level to see details)
[warn] Ignored unknown package option FixedTimestamp(Some(1262304000000))
[success] Total time: 1036 s (17:16), completed Jun 12, 2022 3:34:11 PM
```



- Clean command is used to remove the built jars.

Executed jar using spark submit :

- The spark-submit command is a utility to run or submit a Spark or PySpark application program (or job) to the cluster by specifying options and configurations.
- spark-submit --packages "com.github.music-of-the-ainur:almaren-framework_2.11:0.9.2-2.4,org.apache.spark:spark-hive_2.11:2.4.7,org.postgresql:postgresql:42.3.3" --class com.modak.Main target/scala-2.11/exceltojdbc-assembly-0.1.0-SNAPSHOT.jar
IHsKICAic291cmNlIjp7CiAgICAgInBhdGgiIDogIi9tbnQvYy9Vc2Vycy9tdDQwMjAvRG93bmxvYWRzL1NwYXJrX3VzNy54bHN4Igp9LAogICJ0YXJnZXQxIjp7CiAgInVybCI6ICJqZGJjOnBvc3RncmVzcWw6Ly93My50cmFpbmluZzUubW9kYWsuY29tOjU0MzIvdHJhaW5pbmciLAogICJ1c2VybmFtZSI6ICJtdDQwMjAiLAogICJwYXNzd29yZCI6ICJtdDQwMjBAbTAyeTIyIiwKICAiZHJpdmVyIjogIm9yZy5wb3N0Z3Jlc3FsLkRyaXZlciJ9LAogICJ0YXJnZXQyIjp7CiAgICJwYXRoIiA6ICIvbW50L2MvV29ya0FyZWEvU3BhcmsvRXhjZWxUb0pkYmMvRmlsZXMvIn0KIH0g
- Jar was executed using spark submit command.
- Five files got created.



- Five tables got created.

select * from us7_spark_us7_sheet1 Enter a SQL expression to filter results (use Ctrl+Space)

	constraint_catalog	constraint_schema	constraint_name	check_clause
1	training	public	2200_268157_1_not_null	voter_id IS NOT NULL
2	training	public	2200_266514_2_not_null	ename IS NOT NULL
3	training	public	2200_268157_1_not_null	voter_id IS NOT NULL
4	training	public	2200_266514_1_not_null	eid IS NOT NULL
5	training	public	2200_266514_2_not_null	ename IS NOT NULL
6	training	public	2200_266514_1_not_null	eid IS NOT NULL
7	training	public	2200_276077_2_not_null	ename1 IS NOT NULL
8	training	public	2200_276048_2_not_null	ename IS NOT NULL
9	training	public	2200_276048_2_not_null	ename IS NOT NULL
10	training	public	2200_276048_1_not_null	eid IS NOT NULL
11	training	public	2200_276048_1_not_null	eid IS NOT NULL
12	training	public	2200_276077_2_not_null	ename1 IS NOT NULL
13	training	public	2200_276077_1_not_null	eid IS NOT NULL

	id	firstname	dept_name
1	678	pavan	reasearch
2	789	sanjana	analyst
3	234	cherishma	operations
4	891	harini	operations
5	125	sateesh	accountant
6	246	santhoshi	research
7	345	lavanya	research
8	456	poojitha	research
9	123	vijaya	accounting
10	567	sivaram	analyst

us7_spark_us7_sheet3 1

select * from us7_spark_us7_sheet3

	emp_name	age
1	dean	22

us7_spark_us7_sheet4 1

select * from us7_spark_us7_sheet4

	city	commission
1	Rome	0.15
2	New York	0.14

select * from us7_spark_us7_shee Enter a SQL expressic

	id	course_name	dept_name
1	125	science	researh
2	123	computer	accounting
3	567	electrical	analyst
4	456	mba	accounting