1. reading the data - Reader API

2. crunching of data - transformations

3. write the data back - Writer API

Scala
======

```
orderDf.write.format("csv")
.mode(SaveMode.Overwrite)
.option("path","/Users/trendytech/Desktop/newfolder1")
.save()
```

pyspark
======

```
orderDf.write.format("csv")\
.mode("overwrite")\
.option("path","/Users/trendytech/Desktop/newfolder1")\
.save()
```

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

my_conf = SparkConf()
my_conf.set("spark.app.name", "my first application")
my_conf.set("spark.master","local[*]")

spark = SparkSession.builder.config(conf=my_conf).getOrCreate()

orderDf = spark.read.format("csv")\
    .option("header",True)\
    .option("inferSchema",True)\
    .option("path","/Users/trendytech/Desktop/data/orders.csv")\
    .load()

print("number of partitions are ", orderDf.rdd.getNumPartitions())

ordersRep = orderDf.repartition(4)
```

```
ordersRep.write.format("csv")\
.mode("overwrite")\
.option("path","/Users/trendytech/Desktop/newfolder1")\
.save()
```

====

overwrite
append
errorIfExists
ignore

=====

Parquet is the default file format in apache spark when we talk about structured api's

=====

Spark File Layout

Number of files is equal to number of partitions.

1. simple repartition - repartition

2. partitioning - partitionBy (allows partitioning pruning)

3. bucketBy

4. maxRecordsPerFile

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

my_conf = SparkConf()
my_conf.set("spark.app.name", "my first application")
my_conf.set("spark.master","local[*]")

spark = SparkSession.builder.config(conf=my_conf).getOrCreate()

orderDf = spark.read.format("csv")\
    .option("header",True)\
    .option("inferSchema",True)\
    .option("path","/Users/trendytech/Desktop/data/orders.csv")\
```

```
    .load()


orderDf.write.format("csv").partitionBy("order_status")\
.mode("overwrite")\
.option("path","/Users/trendytech/Desktop/newfolder4")\
.save()
```

=======

Avro

3.1.2 pyspark

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

my_conf = SparkConf()
my_conf.set("spark.app.name", "my first application")
my_conf.set("spark.master","local[*]")
my_conf.set("spark.jars","/Users/trendytech/Downloads/spark-avro_2.12-3.1.2.jar")

spark = SparkSession.builder.config(conf=my_conf).getOrCreate()

orderDf = spark.read.format("csv")\
    .option("header",True)\
    .option("inferSchema",True)\
    .option("path","/Users/trendytech/Desktop/data/orders.csv")\
    .load()


orderDf.write.format("avro")\
.mode("overwrite")\
.option("path","/Users/trendytech/Desktop/newfolder4")\
.save()
```

====

Spark SQL

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

```
my_conf = SparkConf()
my_conf.set("spark.app.name", "my first application")
my_conf.set("spark.master","local[*]")

spark = SparkSession.builder.config(conf=my_conf).getOrCreate()

orderDf = spark.read.format("csv")\
    .option("header",True)\
    .option("inferSchema",True)\
    .option("path","/Users/trendytech/Desktop/data/orders.csv")\
    .load()

orderDf.createOrReplaceTempView("orders")

resultDf = spark.sql("select order_status, count(*) as total_orders from orders group by
order_status")

resultDf.show()

====

from pyspark import SparkConf
from pyspark.sql import SparkSession

my_conf = SparkConf()
my_conf.set("spark.app.name", "my first application")
my_conf.set("spark.master","local[*]")

spark = SparkSession.builder.config(conf=my_conf).getOrCreate()

orderDf = spark.read.format("csv")\
    .option("header",True)\
    .option("inferSchema",True)\
    .option("path","/Users/trendytech/Desktop/data/orders.csv")\
    .load()

orderDf.createOrReplaceTempView("orders")

resultDf = spark.sql("select order_customer_id, count(*) as total_orders from orders where
order_status = 'CLOSED' group by order_customer_id order by total_orders desc")
resultDf.show()

====
```

Table has 2 parts

1. data - warehouse - spark.sql.warehouse.dir
2. metadata - catalog metastore - memory

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

my_conf = SparkConf()
my_conf.set("spark.app.name", "my first application")
my_conf.set("spark.master","local[*]")

spark = SparkSession.builder.config(conf=my_conf).getOrCreate()

orderDf = spark.read.format("csv")\
    .option("header",True)\
    .option("inferSchema",True)\
    .option("path","/Users/trendytech/Desktop/data/orders.csv")\
    .load()

orderDf.write.format("csv")\
    .mode("overwrite")\
    .saveAsTable("orders1")
```

========

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

my_conf = SparkConf()
my_conf.set("spark.app.name", "my first application")
my_conf.set("spark.master","local[*]")

spark = SparkSession.builder.config(conf=my_conf).getOrCreate()

orderDf = spark.read.format("csv")\
    .option("header",True)\
    .option("inferSchema",True)\
    .option("path","/Users/trendytech/Desktop/data/orders.csv")\
    .load()

spark.sql("create database if not exists retail")
```

```
orderDf.write.format("csv")\
    .mode("overwrite")\
    .saveAsTable("retail.orders2")


==========

from pyspark import SparkConf
from pyspark.sql import SparkSession

my_conf = SparkConf()
my_conf.set("spark.app.name", "my first application")
my_conf.set("spark.master","local[*]")

spark = SparkSession.builder.config(conf=my_conf).enableHiveSupport().getOrCreate()

orderDf = spark.read.format("csv")\
    .option("header",True)\
    .option("inferSchema",True)\
    .option("path","/Users/trendytech/Desktop/data/orders.csv")\
    .load()

spark.sql("create database if not exists retail")

orderDf.write.format("csv")\
    .mode("overwrite")\
    .saveAsTable("retail.orders3")

============

from pyspark import SparkConf
from pyspark.sql import SparkSession

my_conf = SparkConf()
my_conf.set("spark.app.name", "my first application")
my_conf.set("spark.master","local[*]")

spark = SparkSession.builder.config(conf=my_conf).enableHiveSupport().getOrCreate()

orderDf = spark.read.format("csv")\
    .option("header",True)\
    .option("inferSchema",True)\
    .option("path","/Users/trendytech/Desktop/data/orders.csv")\
```

```
    .load()

spark.sql("create database if not exists retail")

orderDf.write.format("csv")\
    .mode("overwrite")\
    .bucketBy(4,"order_customer_id")\
    .sortBy("order_customer_id")\
    .saveAsTable("retail.orders4")

============


Spark DF session 12
====================

from pyspark import SparkConf
from pyspark.sql import SparkSession
from pyspark.sql.functions import regexp_extract

my_conf = SparkConf()
my_conf.set("spark.app.name", "my first application")
my_conf.set("spark.master","local[*]")

spark = SparkSession.builder.config(conf=my_conf).getOrCreate()

myregex = r'^(\S+) (\S+)\t(\S+)\,(\S+)'

lines_df = spark.read.text("/Users/trendytech/Desktop/data/orders_new.csv")

#lines_df.printSchema()

#lines_df.show()

final_df =
lines_df.select(regexp_extract('value',myregex,1).alias("order_id"),regexp_extract('value',myreg
ex,2).alias("date"),regexp_extract('value',myregex,3).alias("customer_id"),regexp_extract('value'
,myregex,4).alias("status"))

final_df.printSchema()

final_df.show()

final_df.select("order_id").show()
```

```
final_df.groupby("status").count().show()
```

==========

spark df session 13

==========

Column String

Column object

```
from pyspark import SparkConf
from pyspark.sql import SparkSession
from pyspark.sql.functions import *

my_conf = SparkConf()
my_conf.set("spark.app.name", "my first application")
my_conf.set("spark.master","local[*]")

spark = SparkSession.builder.config(conf=my_conf).getOrCreate()

orderDf = spark.read.format("csv")\
    .option("header",True)\
    .option("inferSchema",True)\
    .option("path","/Users/trendytech/Desktop/data/orders.csv")\
    .load()

orderDf.select("order_id","order_date").show()

orderDf.select(col("order_id")).show()
```

==========

Spark DF Session 14
==========

Creating our own user defined function is spark.

1. Column object expression -- the function won't be registered in catalog

2. SQL expression -- the function will be registered in catalog.
So in this case we can even use it with spark SQL.

if the age is greater than 18 we have to populate the 4th column named Adult with "Y"

else we need to populated the column with "N"

```
from pyspark import SparkConf
from pyspark.sql import SparkSession
from pyspark.sql.functions import *

my_conf = SparkConf()
my_conf.set("spark.app.name", "my first application")
my_conf.set("spark.master","local[*]")

spark = SparkSession.builder.config(conf=my_conf).getOrCreate()

df = spark.read.format("csv")\
    .option("inferSchema",True)\
    .option("path","/Users/trendytech/Desktop/data/dataset1")\
    .load()

df1 = df.toDF("name","age","city")

def ageCheck(age):
    if(age > 18):
        return "Y"
    else:
        return "N"

parseAgeFunction = udf(ageCheck,StringType())

df2 = df1.withColumn("adult",parseAgeFunction("age"))

df2.printSchema()

df2.show()
```

==========

```
from pyspark import SparkConf
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
```

```
my_conf = SparkConf()
my_conf.set("spark.app.name", "my first application")
my_conf.set("spark.master","local[*]")

spark = SparkSession.builder.config(conf=my_conf).getOrCreate()

df = spark.read.format("csv")\
   .option("inferSchema",True)\
   .option("path","/Users/trendytech/Desktop/data/dataset1")\
   .load()

df1 = df.toDF("name","age","city")

def ageCheck(age):
   if(age > 18):
      return "Y"
   else:
      return "N"

spark.udf.register("parseAgeFunction",ageCheck,StringType())

for x in spark.catalog.listFunctions():
   print(x)

df2 = df1.withColumn("adult",expr("parseAgeFunction(age)"))

df2.show()
```

==========

Spark DF session 15
==========

create the spark session

create a local list

create a dataframe from this local list and give column names

add a new column date1 with unix timestamp

add one more column with monotonically increasing id

drop the duplicates based on combination of 2 columns

drop the orderid column

sort based on order date

```
from pyspark import SparkConf
from pyspark.sql import SparkSession
from pyspark.sql.functions import *

my_conf = SparkConf()
my_conf.set("spark.app.name", "my first application")
my_conf.set("spark.master","local[*]")

spark = SparkSession.builder.config(conf=my_conf).getOrCreate()

myList = [(1,"2013-07-25",11599,"CLOSED"),
(2,"2014-07-25",256,"PENDING_PAYMENT"),
(3,"2013-07-25",11599,"COMPLETE"),
(4,"2019-07-25",8827,"CLOSED")]

ordersDf = spark.createDataFrame(myList)\
    .toDF("orderid","orderdate","customerid","status")

newDf = ordersDf\
    .withColumn("date1",unix_timestamp(col("orderdate"))) \
    .withColumn("newid", monotonically_increasing_id()) \
    .dropDuplicates(["orderdate","customerid"])\
    .drop("orderid")\
    .sort("orderdate")

ordersDf.printSchema()
ordersDf.show()
newDf.show()
```

==========

Spark DF session 16
==========

Aggregate transformations

1. Simple aggregations

2. Grouping aggregations
3. window aggregates

//simple aggregates

```
from pyspark import SparkConf
from pyspark.sql import SparkSession
from pyspark.sql.functions import *

my_conf = SparkConf()
my_conf.set("spark.app.name", "my first application")
my_conf.set("spark.master","local[*]")

spark = SparkSession.builder.config(conf=my_conf).getOrCreate()

invoiceDF = spark.read
.format("csv")
.option("header",true)
.option("inferSchema",true)
.option("path","/Users/trendytech/Desktop/order_data.csv")
.load()

//column object expression
invoiceDF.select(
count("*").as("RowCount"),
sum("Quantity").as("TotalQuantity"),
avg("UnitPrice").as("AvgPrice"),
countDistinct("InvoiceNo").as("CountDistinct")).show()


//column string expression
invoiceDF.selectExpr(
"count(*) as RowCount",
"sum(Quantity) as TotalQuantity",
"avg(UnitPrice) as AvgPrice",
"count(Distinct(InvoiceNo)) as CountDistinct").show()


//spark SQL
invoiceDF.createOrReplaceTempView("sales")
spark.sql("select count(*),sum(Quantity),avg(UnitPrice),count(distinct(InvoiceNo)) from
sales").show
```

```
spark.stop()
}
```

```
from pyspark import SparkConf
from pyspark.sql import SparkSession
from pyspark.sql.functions import *

my_conf = SparkConf()
my_conf.set("spark.app.name", "my first application")
my_conf.set("spark.master","local[*]")

spark = SparkSession.builder.config(conf=my_conf).getOrCreate()

invoiceDF = spark.read\
.format("csv")\
.option("header",True)\
.option("inferSchema",True) \
.option("path","/Users/trendytech/Desktop/data/order_data.csv") \
.load()

invoiceDF.select(
    count("*").alias("RowCount"),
    sum("Quantity").alias("TotalQuantity"),
    avg("UnitPrice").alias("AvgPrice"),
    countDistinct("InvoiceNo").alias("CountDistinct")).show()

invoiceDF.selectExpr(
"count(*) as RowCount",
"sum(Quantity) as TotalQuantity",
"avg(UnitPrice) as AvgPrice",
"count(Distinct(InvoiceNo)) as CountDistinct").show()

invoiceDF.createOrReplaceTempView("sales")
spark.sql("select count(*),sum(Quantity),avg(UnitPrice),count(distinct(InvoiceNo)) from
sales").show()
```

==========

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

```
from pyspark.sql.functions import *

my_conf = SparkConf()
my_conf.set("spark.app.name", "my first application")
my_conf.set("spark.master","local[*]")

spark = SparkSession.builder.config(conf=my_conf).getOrCreate()

invoiceDF = spark.read\
.format("csv")\
.option("header",True)\
.option("inferSchema",True) \
.option("path","/Users/trendytech/Desktop/data/order_data.csv") \
.load()

#column object expression
val summaryDF = invoiceDF.groupBy("Country","InvoiceNo")
.agg(sum("Quantity").alias("TotalQuantity"),
sum(expr("Quantity * UnitPrice")).alias("InvoiceValue"))

summaryDF.show()


#string expression
val summaryDf1 = invoiceDF.groupBy("Country","InvoiceNo")
.agg(expr("sum(Quantity) as TotalQunatity"),
expr("sum(Quantity * UnitPrice) as InvoiceValue"))

summaryDf1.show()

#spark SQL
invoiceDF.createOrReplaceTempView("sales")

spark.sql("""select  country,InvoiceNo,sum(Quantity) as totQty,sum(Quantity * UnitPrice) as
InvoiceValue from sales group by country,InvoiceNo""").show()


from pyspark import SparkConf
from pyspark.sql import SparkSession
from pyspark.sql.functions import *

my_conf = SparkConf()
my_conf.set("spark.app.name", "my first application")
my_conf.set("spark.master","local[*]")
```

```
spark = SparkSession.builder.config(conf=my_conf).getOrCreate()

invoiceDF = spark.read\
.format("csv")\
.option("header",True)\
.option("inferSchema",True) \
.option("path","/Users/trendytech/Desktop/data/order_data.csv") \
.load()


#column object expression
summaryDF = invoiceDF.groupBy("Country","InvoiceNo")\
.agg(sum("Quantity").alias("TotalQuantity"),
sum(expr("Quantity * UnitPrice")).alias("InvoiceValue"))

summaryDF.show()


#string expression
summaryDf1 = invoiceDF.groupBy("Country","InvoiceNo")\
.agg(expr("sum(Quantity) as TotalQunatity"),
expr("sum(Quantity * UnitPrice) as InvoiceValue"))

summaryDf1.show()

#spark SQL
invoiceDF.createOrReplaceTempView("sales")

spark.sql("""select  country,InvoiceNo,sum(Quantity) as totQty,sum(Quantity * UnitPrice) as
InvoiceValue from sales group by country,InvoiceNo""").show()


==============

from pyspark import SparkConf
from pyspark.sql import SparkSession, Window
from pyspark.sql.functions import *

my_conf = SparkConf()
my_conf.set("spark.app.name", "my first application")
my_conf.set("spark.master", "local[*]")

spark = SparkSession.builder.config(conf=my_conf).getOrCreate()
```

```
invoiceDF = spark.read \
    .format("csv") \
    .option("header", True) \
    .option("inferSchema", True) \
    .option("path", "/Users/trendytech/Desktop/data/windowdata.csv") \
    .load()

myWindow = Window.partitionBy("country")\
    .orderBy("weeknum")\
    .rowsBetween(Window.unboundedPreceding, Window.currentRow)

mydf = invoiceDF.withColumn("RunningTotal",sum("invoicevalue").over(myWindow))

mydf.show()

==========
```