what all we have learnt?

what is big data

Introduction to hadoop

Big data the big picture

Advantages of cloud

Distributed Storage

HDFS Architecture & it's commands

Linux commands

Distributed Processing

Mapreduce & how to execute it

challenges with Mapreduce

Apache Spark

RDD - why RDD's are resilient, transformations vs actions, parallelize, reduce vs reduceByKey, Join, broadcast join, repartition vs coalesce, cache, Spark UI

Dataframes & Spark SQL -
Datasets - Scala & java

Standard dataframe reader API, shortcut methods

how to create a spark table

managed table vs external table

we saw some transformation and actions

Utility functions

inside each worker node we can have multiple executors

thin vs fat executors

we were inferring the schema

is inferring the schema the right choice??

=> we can end up inferring wrong schema

for example a date might be inferred as string
spark has to scan the data to infer the datatypes - takes time

so we should be enforcing the schema & should not infer it.

=========

```
df = spark.read \
.format("csv") \
.option("header","true") \
.option("inferSchema","true") \
.option("samplingRatio",.01) \
.load("/public/yelp-dataset/yelp_user.csv")
```

/public/trendytech/datasets/orders_sample1.csv

2 ways to enforce the schema
==========

1. orders_schema = 'order_id long, order_date date, cust_id long, order_status string'

schema ddl

```
df = spark.read \
.format("csv") \
.schema(orders_schema) \
.load("/public/trendytech/datasets/orders_sample1.csv")
```

if there is a datatype issue we will get the column values as null

2. StructType

```
orders_schema_struct = StructType([
StructField("orderid",LongType()),
StructField("orderdate",DateType()),
StructField("customerid",IntegerType()),
StructField("orderstatus",StringType()),
])
```

```
df = spark.read \
.format("csv") \
.schema(orders_schema_struct) \
.load("/public/trendytech/datasets/orders_sample1.csv")
```

yyyy-mm-dd 2013-07-25

mm-dd-yyyy 07-25-2013

the dates are hard to deal sometimes, and different versions might have different behaviour

to load a date as string and then later convert it to a relevant form.

```
df = spark.read \
.format("csv") \
.schema(orders_schema) \
.option("dateFormat","mm-dd-yyyy") \
.load("/public/trendytech/datasets/orders_sample2.csv")
```

[Correction: the date format should be MM-dd-yyyy (MM should be in uppercase) in the above options field, else it shows incorrect results]

initially load the date column as string

and then later apply a transformation and convert the datatype of this column from string to date

```
new_df = df.withColumn("order_date",to_date("order_date","mm-dd-yyyy"))
```

3 types of modes
==========

=> failfast (as soon as any malformed record is seen it will error out)

=> permissive (default)
if spark is not able to parse it due to datatype mismatch then make it as null without impacting the other results

=> dropmalformed (if any record has any issues, please get rid of those records and continue with the records which are totally okay)

how to infer the schema

we should enforce the schema

=> schema ddl
=> struct type/struct fields

how to deal with dates

yyyy-mm-dd

if your date is in different format either you specify that upfront.

or you can load it as string and later convert it to date as and when required.

spark2

spark3

withColumn transformation can either create a new column or it can modify the existing column.

if there are parsing issues, generally the data shows as null..

Modes
=====

permissive

failfast

dropmalformed

=========

ways of creating a Dataframe
============================

spark.read()

spark.sql()

spark.table()

spark.range(5)

spark.range(0,8)
spark.range(0,8,2)
column name is id

spark.createDataFrame

to create a rdd from a local list
spark.sparkContext.parallelize(list)

to create a dataframe from a local list
spark.createDataFrame(list)

orders_list = [(1,'2013-07-25 00:00:00.0',11599,'CLOSED'),
(2,'2013-07-25 00:00:00.0',256,'PENDING_PAYMENT'),
(3,'2013-07-25 00:00:00.0',12111,'COMPLETE')]

orders_raw_df = spark.createDataFrame(orders_list)

how to fix the column names

and what if I want to enforce the datatypes..

I do not want to go with the inferred datatypes

lets see how to fix the column names

orders_schema = ["order_id","order_date","cust_id","order_status"]

orders_schema = 'order_id long, order_date string, cust_id int, order_status string'

single step process
df = spark.createDataFrame(orders_list,orders_schema)

2 step process
spark.createDataFrame(orders_list).toDF('order_id','order_date','customer_id','order_status')

earlier you learnt how to convert local python list to a dataframe

rdd to a dataframe

df = spark.createDataFrame(new_orders_rdd,orders_schema)

new_df =
spark.createDataFrame(new_orders_rdd).toDF('order_id','order_date','custom
er_id','order_status')

to convert a rdd to a dataframe

rdd.toDF(schema)

to convert a rdd to a dataframe there are 2 approches:

1. spark.createDataFrame(rdd,schema)
   spark.createDataFrame(rdd).toDF(list of column names)

2. rdd.toDF(schema)

spark.read

spark.sql

spark.table

spark.range

spark.createDataFrame
	=> local python list
	=> Rdd to a dataframe - seen 2 approaches.

=======

how to deal with nested Schema
==========

{"customer_id":1,"fullname":{"firstname":"sumit","lastname":"mittal"},"city":"ban
galore"}
{"customer_id":2,"fullname":{"firstname":"ram","lastname":"kumar"},"city":"hyde
rabad"}
{"customer_id":3,"fullname":{"firstname":"vijay","lastname":"shankar"},"city":"pu
ne"}

schema ddl
ddlSchema = "customer_id long, fullname struct<firstname:string,
lastname:string>,city string"

struct type
customer_schema = StructType([
StructField("customer_id",LongType()),
StructField("fullname",StructType([StructField("firstname",StringType()),Struct
Field("lastname",StringType())])),
StructField("city",StringType())
])

==========

to add a new column - withColumn

to rename an existing column - withColumnRenamed

to drop a column - drop

select vs selectExpr

order items

order_item_id, order_id,product_id,quantity,subtotal,product_price

1,1,957,1,299.98,299.98
2,2,1073,1,199.99,199.99
3,2,502,5,250.0,50.0
4,2,403,1,129.99,129.99
5,4,897,2,49.98,24.99

Nike 20%

Armour 10%

other 0%

new_df = df1.withColumn("product_price",expr("CASE WHEN product_name
like '%Nike%' THEN product_price * 1.2 WHEN product_name like
'%Armour%' THEN product_price * 1.1 ELSE product_price END"))

drop a column name - drop

rename a column name - withColumnRenamed

add a new column or modify an existing one - withColumn

expr and its role..

select vs selectExpr

how to remove duplicate records from our dataframe
==========

```
mylist = [
(1,"Kapil",34),
(1,"Kapil",34),
(1,"Satish",26),
(2,"Satish",26),
]
```

distinct() - when you are talking about all the columns and not a subset of columns

```
df.dropDuplicates(["name","age"]).show()
```

========

Spark Session
==========

```
from pyspark.sql import SparkSession
import getpass
username = getpass.getuser()
spark = SparkSession. \
    builder. \
    config('spark.ui.port', '0'). \
    config("spark.sql.warehouse.dir", f"/user/{username}/warehouse"). \
    enableHiveSupport(). \
    master('yarn'). \
    getOrCreate()


orders_df = spark.read \
.format("csv") \
.option("header","true") \
.option("inferSchema","true") \
.load("/public/trendytech/orders_wh/*")
```

Its an entry point to the spark cluster

Dataframes & Spark SQL

Instead of having a spark context, hive context, sql context .. now all of it is encapsulated in a single spark session

we need spark context when dealing at the RDD level

appName is to set the application name

config("spark.sql.warehouse.dir","/user/itv005857/warehouse")

for any of your managed spark tables the data will be stored at above directory.

enableHiveSupport()
I should be able to access the hive metastore

master('yarn')
master('local[*]')

I am mentioning that my spark job should be running on yarn managed hadoop cluster

if I have a 5 node cluster..

what is the need of spark session when we were having spark context

spark session unifies all the different contexts.

you want two isolated environments in the application

when we used to create 2 spark contexts within the same application then it used to give issues..

with spark session this issue is solved..

and we can have more than 1 spark session in a single application..

both the spark sessions will share the same spark context

within one spark application we have only one spark context

this same spark context will be shared across multiple spark sessions.

deploy mode is client

per spark application we have
a driver - master
and multiple executors - workers

1 driver - 3 executors

client mode - in which driver runs at the client node..

cluster mode - in which your driver runs in the cluster itself

production ready scenario then we should go for cluster mode..

dev

stage

prod

==========