Spark core
===========

rdd's

Structured API's
==================

Dataframes and datasets

A dataframe is a distributed collection of data organized into named columns. It is conceptually equivalent to a table in a relation database.

employee - 100000

dataframe

4 partitions

each parition holding 25k records..

richer optimizations are possible.

rdd till now.. spark 1 style..

spark 2 style of writing code.

are dataframes and datasets launched in spark 2?

df/ds were available in spark 1 as well.

from spark 2 onwards we got a better support for these 2 and both of these are merged into a single API datasets api..

sc (spark context)

separate separate context for each and every thing. spark context ,hive context, sql context spark session - is a unified entry point of a spark application.

it provides a way to interact with various spark functionilty with a lesser number of constructs.

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

spark session is a singleton object

.builder() it returns a builder object

this will help us to configure the spark session

.appName
.master

treat your spark session like your driver.

spark.stop()

Structured API's session - 2

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

spark session

Structured API's session - 3
================================

number of jobs is ideally equal to the number of actions.

each job will have stages - number of shuffle boundaries

each stage will have tasks - number of partitions

Structured API's session - 4
================================

whenever we are working with dataframes or datasets
we are dealing with higher level programming constructs..

when we were working with raw rdd that was a low level code.

your spark compiler will convert your high level code (dataframe code) to low level rdd code.

driver   executors

driver will convert your high level code into low level rdd code and then it will send the low level code to the executors..

```
rdd.map( x => {
      //anonymous function (low level code)
})
```

Structured API's session - 5
=============================

rdd vs dataframe vs datasets

rdd
====

when we deal with raw rdd. we deal with low level code

map
filter
flatMap
reduceByKey

this low level code is not developer friendly.
rdd lacks some of the basic optimizations.

dataframes
==========

spark 1.3 version

higher level constructs which makes the developer life easy.

challenges with dataframes
==========================

1. dataframes do not offer strongly typed code..
   type errors wont be caught at compile time rather we get surprise at runtime.

2. developers felt that there flexibility has become limited..

that the dataframes can be converved to rdd

df.rdd (whenever we want more flexibility and type safety)

1. this conversion from dataframes to rdd is not seamless.

2. if we work with raw rdd by converting dataframe to rdd. we will miss out on some of the major optimizations.

catalyst optimizer / tungsten engine

Dataset API
============

spark 1.6

1. compile time safety

2. we get more flexibility in terms of using lower level code.

conversion from dataframes to datasets is seamless.

we wont lose on any of the optimizations.

before spark 2 both dataframes and datasets are 2 different things..

Spark 2

in spark 2 they merged these 2 into a single unified spark dataset API (Structured API)

Dataframe is nothing but a Dataset[Row]

Row is nothing but a generic type which will be bound at runtime.

in case of dataframes the datatypes are bound at runtime however Dataset[Employee]

the type will be bound at compile time.

Dataset[Row] -> dataframe (type errors are caught at runtime)

Dataset[Employee] -> dataset (compile time type safely)

how to convert dataframe to a dataset

if we replace generic Row with specific object then it becomes a Dataset.

Structured API's session - 6
==============================

with dataframes you will get runtime errors.

with datasets you will be able to catch errors at compile time.

we saw that types are resolved at runtime in case of data frames

now let us convert our dataframe to a dataset

how to convert a dataframe to a dataset?

this is where you require a case class.

create case class

and create dataset[OrdersData]

Dataframes are more preferred over Datasets..

to convert from dataframes to datasets there is an overhead involved and this is for casting it to a particular type

Serialization - converting data into a binary form..

when we are dealing with dataframes then the serialization is managed by tungsten binary format.. (encoders) when we are dealing with datasets then the serialization is managed by java serialization (slow)

using datasets will help us to cut down on developer mistakes..
but it comes with an extra cost of casting and expensive serialization.

Structured API's session - 7
==============================

1. Read the data from a Data Source and create a dataframe/datasets.

- external data source (mysql database, redshift, mongodb)
- internal data source (hdfs, s3, azure blob, google storage)

so we have the flexibility in spark to create a dataframe directly from an external data source.

spark is very good at processing but is not that efficient at ingesting data.

spark gives you a jdbc connector to ingest the data from mysql database directly.

sqoop to get the data to my hdfs and then I will load the data from hdfs to spark dataframe.

use tools like sqoop which are specially made for data ingestion to get your data from external data source to internal data source.

2. Performing a bunch of transformations and actions.

transformations/actions using higher level constructs.

3. writing the data to the target (Sink)

internal/external

Read the data from source (internal data source)

do bunch of transformations/actions

dump the output to the sink

3 read modes
================

1. PERMISSIVE (it sets all the fields to null when it encouters a corrupted record) - default

 _corrupt_record

2. DROPMALFORMED (will ignore the malformed record)

3. FAILFAST (whenever a malformed record is encountered a exception is raised)

file formats
===============
csv

json

parquet

read modes
===========
PERMISSIVE
DROPMALFORMED
FAILFAST

SCHEMA
========
1. INFER (INFER SCHEMA OPTION)
2. IMPLICIT (PARQUET, AVRO ETC..)
3. EXPLICIT

Structured API's session - 8
===============================

3 options to have the schema for a dataframe.

1. infer schema (inferSchema as true)
2. implicit schema (reading parquet, avro etc...)
3. explicit schema (manually defining the schema)

explicit schema
==================

1. Programatically using StructType
2. DDL String

Programatic approach
========================

```
val ordersSchema = StructType(List(
StructField("orderid", IntegerType),
StructField("orderdate", TimestampType),
StructField("customerid", IntegerType),
StructField("status", StringType)
))
```

Int - scala
IntegerType - spark

Long - scala

LongType - spark

Float	FloatType
Double	DoubleType
String	StringType
Date	DateType
Timestamp	TimestampType

DDL String
===========

val ordersSchemaDDL = "orderid Int, orderdate String, custid Int, ordstatus String"

How to convert dataframe to dataset.. it is by using a case class..

dataframe is a dataset[Row]

dataset is a dataset[Orders]