**Spark- Mini Project Case Study:**

As part of this module we will take a simple use case and try to scratch the surface of the Spark. We will be using this use case to demontrate end to end Data Engineering Pipeline.

* Understand Data Model
* Define Problem Statement
* Creating Spark Context
* Setting Run Time Job Properties
* Reading data from CSV Files
* Apply Filtering
* Row Level Transformations
* Perform Joins
* Aggregate Data
* Perform Sorting
* Write output to Files
* Complete Script
* Validating Output

**Understand Data Model**

**Let us understand the data model and also characteristics of the data.**

* **Base directory for retail\_db data sets is /public/retail\_db.**
* **It have six folders, each folder represents a separate table.**
  + **Product Catalog Tables**
    - **products**
    - **categories**
    - **departments**
  + **Customers Table**
    - **customers**
  + **Transactional Tables**
    - **orders**
    - **order\_items**
* **orders and order\_items are related. orders is parent table and order\_items is child table for orders.**
* **All folders have one ore more files under them.**
* **Each line represents a record and have values related to multiple columns. Each record is delimited or separated by comma (,).**
* **First field in each orders record is order\_id and it is a primary key (unique and not null)**
* **Second field in each order\_items record is order\_item\_order\_id which is a foreign key attribute to orders order\_id.**
* **There are other relationships as well, however they are not relevant to get started. We will primarily focus on orders and order\_items data.**

**Define Problem Statement**

Get monthly revenue considering complete or closed orders

* We will use orders and order\_items data.
* **orders** is available at **/public/retail\_db/orders**
* **order\_items** is available at **/public/retail\_db/order\_items**
* We need to consider orders with COMPLETE or CLOSED status.
* Revenue can be computed using **order\_item\_subtotal** which is 5th attribute in order\_items.

**Creating Spark Context**

**Let us understand how to create Spark Context using SparkSession from pyspark.sql.**

* **We need to have spark context to leverage both APIs as well as distributed computing framework.**
* **SparkSession is a wrapper class which will use existing Spark Context or create new one.**
* **We can customize the behavior of Spark Context created by passing properties using config or by using APIs such as appName, master etc.**
* **APIs are provided only for most commonly used properties.**

**Setting Run Time Job Properties**

**Let us understand how to customize run time behavior of submitted jobs.**

* **Once Spark Context is created, we can customize run time behavior by using spark.conf.set.**
* **In our case let us set a property called as spark.sql.shuffle.partitions to 2.**
* **If we do not set this property, by default it will use 200 threads.**

**Reading data from CSV Files**

**Let us quickly see how we can read data from CSV Files.**

* **Spark provide several APIs to read the files of different file formats.**
* **All the out of the box APIs are available under spark.read.**
* **In our case we have to read text files where each record is delimited or separated by comma (',').**
* **To create Data Frames for orders and order\_items we can pass the path to spark.read.csv.**
* **There are other options as well which can be passed using keyword arguments. You can run help on spark.read.csv.**