1. Download and Install Anaconda

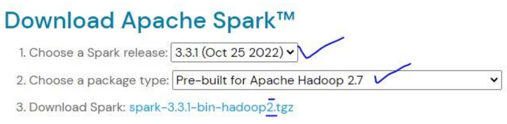
[Anaconda | The World's Most Popular Data Science Platform]

While installing, click on: just me(recommended)
Add Anaconda3 to my PATH environment variables

2. Download and Install latest Java software
   https://www.oracle.com/java/technologies/downloads/#jdk19-windows

3. Download and Extract spark-3.3.1-bin-hadoop2 cut past the extracted file n C-drive

For Hadoop to work in cmd : Type in Google Download Apache Spark



Download Apache Spark™

1. Choose a Spark release: 3.3.1 (Oct 25 2022) ✓
2. Choose a package type: Pre-built for Apache Hadoop 2.7 ✓
3. Download Spark: spark-3.3.1-bin-hadoop2.tgz

Downloads | Apache Spark click step 3it'll direct to below link

https://dlcdn.apache.org/spark/spark-3.3.1/spark-3.3.1-bin-hadoop2.tgz

Download winutils.exe from this link

winutils/winutils.exe at master · steveloughran/winutils (github.com) **select** *(Hadoop 2.7.1)* Copy past it in spark-3.3.1-bin-hadoop2/bin

cut past the extracted file(spark-3.3.1-bin-hadoop2) in C-drive

NOTE: Just copy past "spark-3.3.1-bin-hadoop2(with_winutils)" it contains all

4. Press windows type '**Edit the system environment**'
   Open Environmental Variable

   - **System variables** click on new
     Variable name: SPARK_HOME
     Variable values: Browse C:\spark-3.3.1-bin-hadoop2
     Variable name: HADOOP_HOME
     Variable values: Browse C:\spark-3.3.1-bin-hadoop2
   - **User Variable:**
     path: *edit: New*: %**SPARK_HOME**%\bin press Enter
     %**JAVA_HOME**%\bin press Enter
     New: *Variable name*: PYTHONPATH
     Variable values: %**SPARK_HOME**%\hadoop3\python\lib\py4j-0.10.9.5-src.zip (Also: %SPARK_HOME%\hadoop3\python\lib;%SPARK_HOME%\hadoop3\python;%SPARK_HOME%\hadoop3\python\lib\py4j-0.10.9.5-src.zip , click on- Browse file: C:\spark-3.3.1-bin-hadoop2\python\lib\py4j-0.10.9.5-src.zip) Ok, ok, ok

5. Open cmd, type java then javac
   C:\Users\syeds>conda create -n spark
   conda activate spark conda install
   openjdk pip install findspark
   Pyspark
   ## **Quit()** to Quit**, cntl+c**: to terminate, **cls** to clear, Conda create -n spark --clone base
   Jupyter notebook

To Start Again: open cmd, **conda activate spark,** jupyter notebook

```
import findspark
findspark.init()
import pyspark
```

1. from pyspark import SparkContext, SparkConf conf= SparkConf().setAppName("app").setMaster("local")

   sc= SparkContext(conf=conf) **sc**

   o/p: **SparkContext**

   Spark UI *(Note: you can open Spark UI and check the reports, Analysis. Etc)*
   Version `v3.3.0`
   Master `local`
   AppName `app`

2. from pyspark.sql import SparkSession

   ```
   spark = SparkSession .builder \
   .appName("Python Spark SQL basic example") \
   .config("spark.some.config.option", "some-value") \  # .config is Optional
   .getOrCreate() spark
   ```

   o/p: SparkSession - in-memory
   SparkContext
   Spark UI
   Version `v3.3.0`
   Master `local`
   AppName `app`

To get o/p passing code.py and i/p using cmd:

Type in cmd: conda activate spark

If necessary libraries not install, then you can install it in cmd only **Ex.** pip install pandas

conda install pyarrow

python code.py input-path output-path (if in code arguments passed is 3) python code.py input-path1 input-path2 input-path3 …. n output-path (if in code arguments passed is n) Ex. are shown below.

```
import sys import pandas as pd
import pyspark from pyspark.sql
import SparkSession from
pyspark.sql.functions import *
 from datetime import
datetime
 import uuid from
pyspark.sql.types import *

if __name__ == "__main__":  # make sure to follow the indentation I,e space between if statement
if (len(sys.argv) != 3):
        print("USAGE: file.py [input-folder] [output-folder]")
sys.exit(0)
                             OR
if __name__ == "__main__":
    if (len(sys.argv) != n): # if arguments are 7 i,e 0 for code.py (1 to 5th i/p) (6th o/p)
print("USAGE: file.py [input-folder1] [input-folder2] [input-folder3]…(n-2) [output-folder]")
sys.exit(0)
     spark = SparkSession
\
        .builder \
        .appName("NYTrip") \
        .getOrCreate()

    df = spark.read.parquet(sys.argv[1])
                      OR
    Df1 = spark.read .option("header",
"true").csv(sys.argv[1])      Df2 = spark.read
.option("header", "true").csv(sys.argv[2])      .      .
    Dfn = spark.read .option("header", "true").csv(sys.argv[n-2])


    Df = df.filter(df.c1 > 2.0) # perform the transactions here
     Df \
        .write \
        .partitionBy("c1", "c2", ..”nth") \ # if needed
        .format("parquet") \ # format may be any
        .save(sys.argv[3]) or .save(sys.argv[6]) or .save(sys.argv[n-1])
```

We can pass input and out path instead of passing arguments inside if statement

```
import sys

import pyspark
import pandas as pd
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from typing import Iterator

from datetime import datetime

import uuid
from pyspark.sql.types import *

spark = SparkSession.builder.appName("sports").getOrCreate()
spark.conf.set("spark.sql.execution.arrow.pyspark.enabled", "true")

@pandas_udf("long")  # type: ignore[call-overload]
def plus_ten(iterator: Iterator[pd.Series]) -> Iterator[pd.Series]:
    for x in iterator:
        yield x + 10

finance = spark.read.option("header", "true").csv("s3://dec-syed/rawdata/finance.csv")

finance.createOrReplaceTempView('finance_new')
finance_new = spark.sql(''' select product_id, float(listing_price),
                            float(sale_price), float(discount), float(revenue)
                            from  finance_new''')

fin_tran = finance_new.select(plus_ten("sale_price"))

fin_tran.write.format("parquet").save("s3://dec-syed/output/pyArrow/")
```

executing the files inside cmd using spark-submit file.py or python file.py
just pass the path of o/p and give path in i/p spark documentation

```
import pyspark
from pyspark.sql import SparkSession

spark = SparkSession.builder \
                    .appName('DataSource') \
                    .getOrCreate()

# load Parquet save parquet
df =
spark.read.load("/home/promantus/Documents/spark/examples/src/main/resources/users.parqu
et")
df.select("name",
"favorite_color").write.mode('overwrite').save("namesAndFavColors.parquet")

# load json save parquet
df2 =
spark.read.load("/home/promantus/Documents/spark/examples/src/main/resources/people.json
", format="json")
df2.select("name", "age").write.mode('overwrite').save("namej.parquet",
format="parquet")
```

```
# load csv

df3 =
spark.read.load("/home/promantus/Documents/spark/examples/src/main/resources/people.csv"
,
                       format="csv", sep=";", inferSchema="true", header="true")
print(df3.show())

# load ORC
df4
=  spark.read.orc("/home/promantus/Documents/spark/examples/src/main/resources/users.orc
")
print(df4.show())

Run SQL on files directly
df5 = spark.sql(''' SELECT * FROM
                   parquet.`/home/promantus/Documents/spark/examples/src/main/resources/
users.parquet`
                            ''')
print(df5.show())

df5.write.option('path', 'hive').saveAsTable('t')

df6 = spark.read.parquet('hive')
print(df6.show())


# Bucket
df3.write.bucketBy(42, "name").sortBy("age").saveAsTable("people_bucketed")

df7 = spark.read.parquet('spark-warehouse/people_bucketed')
print(df7.show())

df.write.partitionBy("favorite_color") \
    .format("parquet").save("namePartByColor")
```

Struct data frame

```
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType,StructField, StringType, IntegerType

spark = SparkSession.builder \
                    .appName('SparkByExamples.com') \
                    .getOrCreate()

structureData = [
    (("James","","Smith"),"36636","M",3100),
    (("Michael","Rose",""),"40288","M",4300),
    (("Robert","","Williams"),"42114","M",1400),
    (("Maria","Anne","Jones"),"39192","F",5500),
    (("Jen","Mary","Brown"),"","F",-1)
  ]
structureSchema = StructType([
        StructField('name', StructType([
             StructField('firstname', StringType(), True),
             StructField('middlename', StringType(), True),
             StructField('lastname', StringType(), True)
             ])),
         StructField('id', StringType(), True),
         StructField('gender', StringType(), True),
         StructField('salary', IntegerType(), True)
         ])

df2 = spark.createDataFrame(data=structureData,schema=structureSchema)
df2.printSchema()
df2.show(truncate=False)

# 5. Adding & Changing struct of the DataFrame
from pyspark.sql.functions import col,struct,when

updatedDF = df2.withColumn("OtherInfo",
    struct(col("id").alias("identifier"),
    col("gender").alias("gender"),
    col("salary").alias("salary"),
    when(col("salary").cast(IntegerType()) < 2000,"Low")
      .when(col("salary").cast(IntegerType()) < 4000,"Medium")
      .otherwise("High").alias("Salary_Grade")
  )).drop("id","gender","salary")

updatedDF.printSchema()
updatedDF.show(truncate=False)


# Q7  Creating StructType object struct from JSON file

print(df2.schema.json())

schema = df2.schema.json()

import json
schemaFromJson = StructType.fromJson(json.loads(schema))
df3 = spark.createDataFrame(
        spark.sparkContext.parallelize(structureData),schemaFromJson)
df3.printSchema()


# 9. Checking if a Column Exists in a DataFrame

print(df2.schema.fieldNames.contains("firstname"))
```