In [6]:
```
from pyspark.sql import SparkSession
spark=SparkSession.builder.appName("DfApp").getOrCreate()
```

In [2]:
```
pip install mysql-connector-python
```

```
Requirement already satisfied: mysql-connector-python in c:\users\user\anaconda3\lib\site-packages (8.0.28)
Requirement already satisfied: protobuf>=3.0.0 in c:\users\user\anaconda3\lib\site-packages (from mysql-connector-python) (3.19.3)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\user\anaconda3\lib\site-packages (1.3.4)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\user\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in c:\users\user\anaconda3\lib\site-packages (from pandas) (2021.3)
Requirement already satisfied: numpy>=1.17.3 in c:\users\user\anaconda3\lib\site-packages (from pandas) (1.20.3)
Requirement already satisfied: six>=1.5 in c:\users\user\anaconda3\lib\site-packages (from python-dateutil>=2.7.3->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [7]:
```
import mysql.connector as c
import pandas as pd
```

In [3]:
```
conn=c.Connect(host='localhost',port=3306,user='root',password='root',database='customer')
cursor=conn.cursor()
```

In [4]:
```
pdf=pd.read_sql("SELECT Item_id,Item_name, Item_cost, Supplier_id FROM items",con=conn)
print(pdf)
```

```
   Item_id Item_name  Item_cost  Supplier_id
0      101     Pizza       2000          110
1      103     Maggi       1600          130
2      104    Panner       2500          140
3      105    Burger       2300          150
```

In [5]:
```
type(pdf)
```

Out[5]:
```
pandas.core.frame.DataFrame
```

In [41]:
```
pip install SQLAlchemy
```

```
Requirement already satisfied: SQLAlchemy in c:\users\user\anaconda3\lib\site-packages (1.4.22)
Requirement already satisfied: greenlet!=0.4.17 in c:\users\user\anaconda3\lib\site-packages (from SQLAlchemy) (1.1.1)
Note: you may need to restart the kernel to use updated packages.
```

In [10]:
```
#spark.createDataFrame(pd.read_sql("select Item_id,Item_name, Item_cost, Supplier_id from items",con=conn)).show()
```

In [6]:
```
from pyspark.sql.types import *
pdf.to_csv("items.csv")
sch=StructType([
    StructField("ItemId",IntegerType(),True),
    StructField("ItemName",StringType(),True),
    StructField("ItemCost",IntegerType(),True),
    StructField("ItemQty",IntegerType(),True),
    StructField("SupplierId",StringType(),True)])
df=spark.read.option("header","true").csv('D:\items.csv',inferSchema=True)
df.show()
df.printSchema()
```

```
+------+--------+--------+-------+----------+
|ItemId|ItemName|ItemCost|ItemQty|SupplierId|
+------+--------+--------+-------+----------+
|     4|   Chock|    65.76|     23|         X|
|     5|  Pencil|   45.65|     34|         Y|
|     6|     Pen|   76.87|     32|         X|
|     7|  Duster|    54.0|     10|         Z|
|     8|    null|    23.0|     45|         Y|
|     9|    book|    53.0|     25|      null|
+------+--------+--------+-------+----------+

root
 |-- ItemId: integer (nullable = true)
 |-- ItemName: string (nullable = true)
 |-- ItemCost: double (nullable = true)
 |-- ItemQty: integer (nullable = true)
 |-- SupplierId: string (nullable = true)
```

In [8]:
```
#pdf=spark.read.option("header","true").csv('D:\Books.csv',inferSchema=True)
#pdf.show()
pdf=spark.read.csv("D:\Books.csv",header=True, inferSchema=True)
pdf.show()
```

```
+------+-------------+--------+---+
|BookId|     BookName|BookCost|Aid|
+------+-------------+--------+---+
|  1001|      I Dare|     300|120|
|  1002|  The Shivaji|    2000|122|
|  1003|      Panipat|    1000|122|
|  1004|  Asami Asami|    2000|123|
|  1005|   Sweet Home|     500|121|
|  1006|  India Today|    2250|122|
|  1007|Know Yourself|    1334|124|
+------+-------------+--------+---+
```

In [78]:
```
type(pdf.collect())
```

Out[78]:
```
list
```

In [79]:
```
for row in pdf.collect():
    print(row["BookId"],",",row["BookName"])
```

```
1001 , I Dare
1002 , The Shivaji
1003 , Panipat
1004 , Asami Asami
1005 , Sweet Home
1006 , India Today
1007 , Know Yourself
```

In [80]:
```
for itr in pdf.rdd.toLocalIterator():
    print(itr['BookId'],",",itr['BookName'])
```

```
1001 , I Dare
1002 , The Shivaji
1003 , Panipat
1004 , Asami Asami
1005 , Sweet Home
1006 , India Today
1007 , Know Yourself
```

In [86]:
```
df=pdf.toPandas()
df
```

Out[86]:

	BookId	BookName	BookCost	Aid
0	1001	I Dare	300	120
1	1002	The Shivaji	2000	122
2	1003	Panipat	1000	122
3	1004	Asami Asami	2000	123
4	1005	Sweet Home	500	121
5	1006	India Today	2250	122
6	1007	Know Yourself	1334	124

In [88]:
```
df.iterrows
for rowindex,row in df.iterrows():
    print(rowindex,"   -   ",row[0],",",row[1]," ,",row[3])
```

```
0    -    1001 , I Dare  , 120
1    -    1002 , The Shivaji  , 122
2    -    1003 , Panipat  , 122
3    -    1004 , Asami Asami  , 123
4    -    1005 , Sweet Home  , 121
5    -    1006 , India Today  , 122
6    -    1007 , Know Yourself  , 124
```

In [90]:
```
df.iterrows
for rowindex,row in df.iterrows():
    if(rowindex%2==0):
        print(rowindex,"   -   ",row[0],",",row[1]," ,",row[3])
```

```
0    -    1001 , I Dare  , 120
2    -    1003 , Panipat  , 122
4    -    1005 , Sweet Home  , 121
6    -    1007 , Know Yourself  , 124
```

In [91]:
```
pdf.show()
```

```
+------+-------------+--------+---+
|BookId|     BookName|BookCost|Aid|
+------+-------------+--------+---+
|  1001|      I Dare|     300|120|
|  1002|  The Shivaji|    2000|122|
|  1003|      Panipat|    1000|122|
|  1004|  Asami Asami|    2000|123|
|  1005|   Sweet Home|     500|121|
|  1006|  India Today|    2250|122|
|  1007|Know Yourself|    1334|124|
+------+-------------+--------+---+
```

In [81]:
```
for row in pdf.select("BookId","BookName","BookCost").collect():
    if(row['BookCost']>1000):
        print(row['BookName'])
```

```
The Shivaji
Asami Asami
India Today
Know Yourself
```

In [82]:
```
pdf.filter(pdf["BookCost"]>1000)
```

Out[82]:
```
DataFrame[BookId: int, BookName: string, BookCost: int, Aid: int]
```

In [85]:
```
pdf.select("BookId","BookName","BookCost").filter(pdf["BookCost"]>1000).collect()
```

Out[85]:
```
[Row(BookId=1002, BookName='The Shivaji', BookCost=2000),
 Row(BookId=1004, BookName='Asami Asami', BookCost=2000),
 Row(BookId=1006, BookName='India Today', BookCost=2250),
 Row(BookId=1007, BookName='Know Yourself', BookCost=1334)]
```

In [98]:
```
l=[row["BookName"] for row in pdf.rdd.collect()]
l
```

Out[98]:
```
['I Dare',
 'The Shivaji',
 'Panipat',
 'Asami Asami',
 'Sweet Home',
 'India Today',
 'Know Yourself']
```

In [95]:
```
for value in [row["BookName"] for row in pdf.rdd.collect()]:
    print(value)
```

```
I Dare
The Shivaji
Panipat
Asami Asami
Sweet Home
India Today
Know Yourself
```

In [108]:
```
l=[row["BookCost"] for row in pdf.rdd.collect()]
list(map(lambda a : a+500,l))
```

Out[108]:
```
[800, 2500, 1500, 2500, 1000, 2750, 1834]
```

In [109]:
```
pdf
```

Out[109]:
```
DataFrame[BookId: int, BookName: string, BookCost: int, Aid: int]
```

In [2]:
```
from pyspark.sql.types import *
bk=StructType([
    StructField("BookId",IntegerType(),True),
    StructField("BookName",StringType(),True)])
df=spark.read.option("header","true").csv('D:\Books.csv',inferSchema=True)
df.show()
df.printSchema()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_12676/3900125577.py in <module>
      3     StructField("BookId",IntegerType(),True),
      4     StructField("BookName",StringType(),True)])
----> 5 df=spark.read.option("header","true").csv('D:\Books.csv',inferSchema=True)
      6 df.show()
      7 df.printSchema()

NameError: name 'spark' is not defined
```

In [9]:
```
rdd=pdf.rdd.map(lambda row: (row["BookId"],row["BookName"]))
#rdd.toDF('BookId','BookName').collect()
rdd.toDF(bk)
```

Out[9]:
```
DataFrame[BookId: int, BookName: string]
```

In [10]:
```
rdd=pdf.rdd.map(lambda row: (row["BookId"],row["BookName"]))
for rdd in rdd.toDF(bk):
    print(rdd)
```

```
Column<'BookId'>
Column<'BookName'>
```

In [ ]:
```

```