In [2]:
```python
from pyspark.sql import SparkSession
import pyspark.sql.functions as F
from pyspark.sql.types import *

spark = SparkSession\
    .builder\
    .appName("chapter-09-data-src")\
    .getOrCreate()

import os
SPARK_BOOK_DATA_PATH = os.environ['SPARK_BOOK_DATA_PATH']
```

In [3]:
```python
spark
```

Out[3]:

SparkSession - hive

SparkContext

Spark UI (http://172.17.0.1:4053)

Version

`v3.0.1`

Master

`local[*]`

AppName

`PySparkShell`

CSV

In [26]:
```python
file_path = SPARK_BOOK_DATA_PATH + "/data/flight-data/csv/2010-summa

csvFile = spark.read.format("csv")\
  .option("header", "true")\
  .option("mode", "FAILFAST")\
  .option("inferSchema", "true")\
  .load(file_path)
```

In [27]:
```python
csvFile.show(5)
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|    United States|            Romania|    1|
|    United States|            Ireland|  264|
|    United States|              India|   69|
|            Egypt|      United States|   24|
|Equatorial Guinea|      United States|    1|
+-----------------+-------------------+-----+
only showing top 5 rows
```

In [6]:
```
# COMMAND ----------

csvFile.write.format("csv").mode("overwrite").option("sep", "\t")\
  .save("/tmp/my-tsv-file.tsv")
```

Json

In [8]:
```
# COMMAND ----------

file_path = SPARK_BOOK_DATA_PATH + "/data/flight-data/json/2010-sumr
jsonFile = spark.read.option("mode","FAILFAST").option("inferSchema'
jsonFile.show(5)
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|    United States|            Romania|    1|
|    United States|            Ireland|  264|
|    United States|              India|   69|
|            Egypt|      United States|   24|
|Equatorial Guinea|      United States|    1|
+-----------------+-------------------+-----+
only showing top 5 rows
```

In [9]:
```
jsonFile.printSchema()
```

```
root
 |-- DEST_COUNTRY_NAME: string (nullable = true)
 |-- ORIGIN_COUNTRY_NAME: string (nullable = true)
 |-- count: long (nullable = true)
```

In [10]:
```
# COMMAND ----------

jsonFile.write.format("json").mode("overwrite").save("/tmp/my-json-
```

Parquet

In [11]:
```
# COMMAND ----------
file_path = SPARK_BOOK_DATA_PATH + "/data/flight-data/parquet/2010-s
df = spark.read.format("parquet").load(file_path)

df.show(5)
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|    United States|            Romania|    1|
|    United States|            Ireland|  264|
|    United States|              India|   69|
|            Egypt|      United States|   24|
|Equatorial Guinea|      United States|    1|
+-----------------+-------------------+-----+
only showing top 5 rows
```

In [12]:
```
# COMMAND ----------

df.write.format("parquet").mode("overwrite")\
  .save("/tmp/my-parquet-file.parquet")
```

Orc

In [13]:
```
# COMMAND ----------
file_path = SPARK_BOOK_DATA_PATH + "/data/flight-data/orc/2010-summa
df = spark.read.format("orc").load(file_path)

df.show(5)
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|    United States|            Romania|    1|
|    United States|            Ireland|  264|
|    United States|              India|   69|
|            Egypt|      United States|   24|
|Equatorial Guinea|      United States|    1|
+-----------------+-------------------+-----+
only showing top 5 rows
```

In [14]:
```
# COMMAND ----------

df.write.format("orc").mode("overwrite").save("/tmp/my-json-file.orc
```

Database - Sqlite

https://intellipaat.com/community/9608/how-to-load-table-from-sqllite-db-file-from-pyspark (https://intellipaat.com/community/9608/how-to-load-table-from-sqllite-db-file-from-pyspark)

https://repo1.maven.org/maven2/org/xerial/sqlite-jdbc/3.27.2.1/
(https://repo1.maven.org/maven2/org/xerial/sqlite-jdbc/3.27.2.1/)

/home/wengong/spark/spark-3.0.1-bin-hadoop2.7/jars/sqlite-jdbc-3.27.2.1.jar

In [7]:
```
# COMMAND ----------
file_path = SPARK_BOOK_DATA_PATH + "/data/flight-data/jdbc/my-sqlite
driver = "org.sqlite.JDBC"
path = file_path
url = "jdbc:sqlite:" + path
tablename = "flight_info"
```

In [8]:
```
file_path
```

Out[8]:
```
'/home/wengong/spark_data//data/flight-data/jdbc/my-sqlite.db'
```

In [9]:
```
# COMMAND ----------

dbDataFrame = spark.read.format("jdbc")\
    .option("url", url)\
    .option("dbtable", tablename)\
    .option("driver",  driver)\
    .load()
```

In [10]:
```
dbDataFrame.show(5)
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|    United States|            Romania|    1|
|    United States|            Ireland|  264|
|    United States|              India|   69|
|            Egypt|      United States|   24|
|Equatorial Guinea|      United States|    1|
+-----------------+-------------------+-----+
only showing top 5 rows
```

In [11]:
```
dbDataFrame.printSchema()
```

```
root
 |-- DEST_COUNTRY_NAME: string (nullable = true)
 |-- ORIGIN_COUNTRY_NAME: string (nullable = true)
 |-- count: decimal(20,0) (nullable = true)
```

```
# COMMAND ----------

pgDF = spark.read.format("jdbc")\
  .option("driver", "org.postgresql.Driver")\
  .option("url", "jdbc:postgresql://database_server")\
  .option("dbtable", "schema.tablename")\
  .option("user", "username").option("password", "my-secret-
password").load()
```

In [12]:
```
# COMMAND ----------

dbDataFrame.filter("DEST_COUNTRY_NAME in ('Anguilla', 'Sweden')").ex
```

```
== Physical Plan ==
*(1) Scan JDBCRelation(flight_info) [numPartitions=1] [DEST_COUNTRY_NA
ME#44,ORIGIN_COUNTRY_NAME#45,count#46] PushedFilters: [*In(DEST_COUNTR
Y_NAME, [Anguilla,Sweden])], ReadSchema: struct<DEST_COUNTRY_NAME:stri
ng,ORIGIN_COUNTRY_NAME:string,count:decimal(20,0)>
```

In [13]:
```
# COMMAND ----------

pushdownQuery = """(SELECT DISTINCT(DEST_COUNTRY_NAME) FROM flight_i
  AS flight_info"""
dbDataFrame = spark.read.format("jdbc")\
  .option("url", url).option("dbtable", pushdownQuery).option("drive
  .load()
```

In [16]:
```
dbDataFrame.show(3)
```

```
+-----------------+
|DEST_COUNTRY_NAME|
+-----------------+
|    United States|
|            Egypt|
|Equatorial Guinea|
+-----------------+
only showing top 3 rows
```

In [17]:
```
dbDataFrame.explain()
```

```
== Physical Plan ==
*(1) Scan JDBCRelation((SELECT DISTINCT(DEST_COUNTRY_NAME) FROM flight
_info)
  AS flight_info) [numPartitions=1] [DEST_COUNTRY_NAME#63] PushedFilte
rs: [], ReadSchema: struct<DEST_COUNTRY_NAME:string>
```

In [18]:
```
# COMMAND ----------

dbDataFrame = spark.read.format("jdbc")\
  .option("url", url).option("dbtable", tablename).option("driver",
  .option("numPartitions", 10).load()
```

In [19]:
```
dbDataFrame.show(3)
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|    United States|            Romania|    1|
|    United States|            Ireland|  264|
|    United States|              India|   69|
+-----------------+-------------------+-----+
only showing top 3 rows
```

In [20]:
```
dbDataFrame.explain()
```

```
== Physical Plan ==
*(1) Scan JDBCRelation(flight_info) [numPartitions=1] [DEST_COUNTRY_NA
ME#75,ORIGIN_COUNTRY_NAME#76,count#77] PushedFilters: [], ReadSchema:
struct<DEST_COUNTRY_NAME:string,ORIGIN_COUNTRY_NAME:string,count:decim
al(20,0)>
```

In [21]:
```
# COMMAND ----------

props = {"driver":"org.sqlite.JDBC"}
predicates = [
  "DEST_COUNTRY_NAME = 'Sweden' OR ORIGIN_COUNTRY_NAME = 'Sweden'",
  "DEST_COUNTRY_NAME = 'Anguilla' OR ORIGIN_COUNTRY_NAME = 'Anguilla
spark.read.jdbc(url, tablename, predicates=predicates, properties=pr
```

```
+-----------------+-------------------+-----+
|DEST_COUNTRY_NAME|ORIGIN_COUNTRY_NAME|count|
+-----------------+-------------------+-----+
|           Sweden|      United States|   65|
|    United States|             Sweden|   73|
|         Anguilla|      United States|   21|
|    United States|           Anguilla|   20|
+-----------------+-------------------+-----+
```

In [22]:
```
spark.read.jdbc(url,tablename,predicates=predicates,properties=props
  .rdd.getNumPartitions() # 2
```

Out[22]: 2

In [23]:
```
# COMMAND ----------

props = {"driver":"org.sqlite.JDBC"}
predicates = [
  "DEST_COUNTRY_NAME != 'Sweden' OR ORIGIN_COUNTRY_NAME != 'Sweden'
  "DEST_COUNTRY_NAME != 'Anguilla' OR ORIGIN_COUNTRY_NAME != 'Anguil
spark.read.jdbc(url, tablename, predicates=predicates, properties=pr
```

Out[23]: 510

In [25]:
```
# COMMAND ----------

colName = "count"
lowerBound = 0
upperBound = 348113 # this is the max count in our database
numPartitions = 10


# COMMAND ----------

spark.read.jdbc(url, tablename, column=colName, properties=props,
                lowerBound=lowerBound, upperBound=upperBound,
                numPartitions=numPartitions).count() # 255
```
Out[25]:
```
255
```

In [28]:
```
# COMMAND ----------

newPath = "jdbc:sqlite://tmp/my-sqlite.db"
csvFile.write.jdbc(newPath, tablename, mode="overwrite", properties=
```

In [29]:
```
# COMMAND ----------

spark.read.jdbc(newPath, tablename, properties=props).count() # 255
```
Out[29]:
```
255
```

In [30]:
```
# COMMAND ----------

csvFile.write.jdbc(newPath, tablename, mode="append", properties=pro
```

In [31]:
```
# COMMAND ----------

spark.read.jdbc(newPath, tablename, properties=props).count() # 510
```
Out[31]:
```
510
```

In [32]:
```
csvFile.limit(10).select("DEST_COUNTRY_NAME", "count").show()
```
```
+-----------------+-----+
|DEST_COUNTRY_NAME|count|
+-----------------+-----+
|    United States|    1|
|    United States|  264|
|    United States|   69|
|            Egypt|   24|
|Equatorial Guinea|    1|
|    United States|   25|
|    United States|   54|
|       Costa Rica|  477|
|          Senegal|   29|
|    United States|   44|
+-----------------+-----+
```

write out data by partition

In [33]:

```
# COMMAND ----------

csvFile.limit(10).select("DEST_COUNTRY_NAME", "count")\
  .write.partitionBy("count").text("/tmp/five-csv-files2py.csv")
```

In [34]:

```
# COMMAND ----------

csvFile.limit(10).write.mode("overwrite").partitionBy("DEST_COUNTRY_
  .save("/tmp/partitioned-files.parquet")


# COMMAND ----------
```

In [ ]:

```

```