Spark SQL

- You can run SQL queries against views or tables organized into databases
- You also can use system functions or define user functions and analyze query plans in order to optimize their workloads.
- This integrates directly into the DataFrame and Dataset API
- You can choose to express some of your data manipulations in SQL and others in DataFrames

In [1]:
```
# Import depedicies
from pyspark.sql import SparkSession
from pyspark.sql import *
from pyspark.sql.functions import *
from pyspark.sql.types import *
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
spark =SparkSession(sc)
```

SQL Syntax of SELECT statement

- SELECT select_list
- FROM table_source
- WHERE search_condition
- GROUP BY group_by_list
- HAVING search_condition
- ORDER BY order_bylist

In [2]:
```
# Create a new DataFrame and register as a temporary view to query it in SQL(SQL trasfo
# For Quering in SQL, name the SQL dataset table dfTable
df = spark.read.format("csv")\
.option("header", "true")\
.option("inferSchema", "true")\
.load("Resources/Vendors.csv")\
.coalesce(5) # reduce the number of partitions avoiding network shuffle
df.cache() # speed up performance
df.createOrReplaceTempView("Vendors")
```

In [4]:
```
def df_information(df):
    df.printSchema()
    df.show(5)
    df.describe().show()
    df.dtypes
    return(df)

print(df_information(df))
```

```
root
 |-- VendorID: integer (nullable = true)
 |-- VendorName: string (nullable = true)
 |-- VendorAddress1: string (nullable = true)
 |-- VendorAddress2: string (nullable = true)
 |-- VendorCity: string (nullable = true)
 |-- VendorState: string (nullable = true)
```

```
 |-- VendorZipCode: integer (nullable = true)
 |-- VendorPhone: string (nullable = true)
 |-- VendorContactLName: string (nullable = true)
 |-- VendorContactFName: string (nullable = true)
 |-- DefaultTermsID: integer (nullable = true)
 |-- DefaultAccountNo: integer (nullable = true)
```

```
+--------+--------------------+--------------------+--------------+-----------+---------
--+-------------+--------------+------------------+------------------+--------------+---
-------------+
|VendorID|          VendorName|      VendorAddress1|VendorAddress2| VendorCity|VendorSta
te|VendorZipCode|   VendorPhone|VendorContactLName|VendorContactFName|DefaultTermsID|Def
aultAccountNo|
+--------+--------------------+--------------------+--------------+-----------+---------
--+-------------+--------------+------------------+------------------+--------------+---
-------------+
|       1|   US Postal Service|Attn:  Supt. Wind...|   PO Box 7005|    Madison|
WI|        53707|(800) 555-1205|           Alberto|         Francesco|             1|
552|
|       2|National Informat...|        PO Box 96621|          null| Washington|
DC|        20090|(301) 555-8950|             Irvin|              Ania|             3|
540|
|       3|Register of Copyr...| Library Of Congress|          null| Washington|
DC|        20559|          null|             Liana|              Lukas|             3|
403|
|       4|             Jobtrak|1990 Westwood Blv...|          null|Los Angeles|
CA|        90025|(800) 555-8725|             Quinn|             Kenzie|             3|
572|
|       5| Newbrige Book Clubs|   3000 Cindel Drive|          null| Washington|
NJ|         7882|(800) 555-9980|             Marks|           Michelle|             4|
394|
+--------+--------------------+--------------------+--------------+-----------+---------
--+-------------+--------------+------------------+------------------+--------------+---
-------------+
only showing top 5 rows
```

```
+-------+------------------+------------+------------------+-----------------+---------
-+-----------+-----------------+--------------+------------------+------------------+---
---------------+------------------+
|summary|          VendorID|  VendorName|     VendorAddress1|   VendorAddress2|VendorCit
y|VendorState|    VendorZipCode|   VendorPhone|VendorContactLName|VendorContactFName|
DefaultTermsID|  DefaultAccountNo|
+-------+------------------+------------+------------------+-----------------+---------
-+-----------+-----------------+--------------+------------------+------------------+---
---------------+------------------+
|  count|               122|         122|               120|               19|       12
2|        122|              122|            97|              122|              122|
122|              122|
|   mean|62.278688524590166|        null|              null|             null|      nul
l|       null|72606.68032786885|          null|              null|             null|2.6
639344262295084| 511.4590163934426|
| stddev| 35.66331594300987|        null|              null|             null|      nul
l|       null|30709.57845736728|          null|              null|             null| 0.
819298790467335|108.96509013645611|
|    min|                 1|   ASC Signs|"1627 ""E"" Street"|1150 N Tustin Ave|   Anahei
m|         AZ|             2107|(201) 555-9742|          Aaronsen|             Aaron|
1|              150|
|    max|               123|Zylka Design| Secretary Of State|          Suite F|Washingto
n|         WI|            95887|(947) 555-3900|            Yobani|               Zev|
5|              631|
+-------+------------------+------------+------------------+-----------------+---------
-+-----------+-----------------+--------------+------------------+------------------+---
---------------+------------------+
```

```
DataFrame[VendorID: int, VendorName: string, VendorAddress1: string, VendorAddress2: str
```

```
ing, VendorCity: string, VendorState: string, VendorZipCode: int, VendorPhone: string, V
endorContactLName: string, VendorContactFName: string, DefaultTermsID: int, DefaultAccou
ntNo: int]
```

In [87]:
```
df.na.drop("any").show(5)
```

```
+---------+------------------+
|AccountNo| AccountDescription|
+---------+------------------+
|      100|              Cash|
|      110|Accounts Receivable|
|      120|     Book Inventory|
|      150|          Furniture|
|      160| Computer Equipment|
+---------+------------------+
only showing top 5 rows
```

In [4]:
```
# Create a new DataFrame and register as a temporary view to query it in SQL(SQL trasfo
# For Quering in SQL, name the SQL dataset table dfTable
df = spark.read.format("csv")\
.option("header", "true")\
.option("inferSchema", "true")\
.load("Resources/Invoices.csv")\
.coalesce(5)
df.cache()
df.createOrReplaceTempView("Invoices")
```

In [88]:
```
df.na.drop("any").show(5)
```

```
+---------+------------------+
|AccountNo| AccountDescription|
+---------+------------------+
|      100|              Cash|
|      110|Accounts Receivable|
|      120|     Book Inventory|
|      150|          Furniture|
|      160| Computer Equipment|
+---------+------------------+
only showing top 5 rows
```

In [89]:
```
spark.sql("""
SELECT *
FROM Invoices
LIMIT 5
""").show()
```

```
+---------+--------+-------------+------------------+-----------+-----------+--------
---+-------+------------------+------------------+
|InvoiceID|VendorID|InvoiceNumber|       InvoiceDate|InvoiceTotal|PaymentTotal|CreditTo
tal|TermsID|     InvoiceDueDate|       PaymentDate|
+---------+--------+-------------+------------------+-----------+-----------+--------
---+-------+------------------+------------------+
|        1|     122|   989319-457|2011-12-08 00:00:00|   $3,813.33|   $3,813.33|
$0.00|      3|2012-01-08 00:00:00|2012-01-07 00:00:00|
|        2|     123|    263253241|2011-12-10 00:00:00|      $40.20|      $40.20|
$0.00|      3|2012-01-10 00:00:00|2012-01-14 00:00:00|
|        3|     123|    963253234|2011-12-13 00:00:00|     $138.75|     $138.75|
$0.00|      3|2012-01-13 00:00:00|2012-01-09 00:00:00|
|        4|     123|   2-000-2993|2011-12-16 00:00:00|     $144.70|     $144.70|
$0.00|      3|2012-01-16 00:00:00|2012-01-12 00:00:00|
|        5|     123|    963253251|2011-12-16 00:00:00|      $15.50|      $15.50|
$0.00|      3|2012-01-16 00:00:00|2012-01-11 00:00:00|
```

```
+---------+--------+-------------+------------------+------------+------------+--------
---+-------+--------------------+------------------+
```

In [6]:
```
# Create a new DataFrame and register as a temporary view to query it in SQL(SQL trasfo
# For Quering in SQL, name the SQL dataset table dfTable
df = spark.read.format("csv")\
.option("header", "true")\
.option("inferSchema", "true")\
.load("Resources/InvoiceLineItems.csv")\
.coalesce(5)
df.cache()
df.createOrReplaceTempView("InvoiceLineItems")
```

In [90]:
```
df.na.drop("any").show(5)
```

```
+---------+-------------------+
|AccountNo| AccountDescription|
+---------+-------------------+
|      100|               Cash|
|      110|Accounts Receivable|
|      120|     Book Inventory|
|      150|          Furniture|
|      160| Computer Equipment|
+---------+-------------------+
only showing top 5 rows
```

In [8]:
```
# Create a new DataFrame and register as a temporary view to query it in SQL(SQL trasfo
# For Quering in SQL, name the SQL dataset table dfTable
df = spark.read.format("csv")\
.option("header", "true")\
.option("inferSchema", "true")\
.load("Resources/GLAccounts.csv")\
.coalesce(5)
df.cache()
df.createOrReplaceTempView("GLAccounts")
```

In [91]:
```
df.na.drop("any").show(5)
```

```
+---------+-------------------+
|AccountNo| AccountDescription|
+---------+-------------------+
|      100|               Cash|
|      110|Accounts Receivable|
|      120|     Book Inventory|
|      150|          Furniture|
|      160| Computer Equipment|
+---------+-------------------+
only showing top 5 rows
```

In [10]:
```
# Join the vendos and invoice tables at VendorID
spark.sql("""
SELECT InvoiceNumber, VendorName
FROM Vendors AS V
JOIN Invoices AS I
ON V.VendorID =I.VendorID
""").show(8)
```

```
+-------------+--------------------+
|InvoiceNumber|          VendorName|
```

```
+-------------+--------------------+
|      Q545443|                 IBM|
|      QP58872|                 IBM|
|    547480102|          Blue Cross|
|    547479217|          Blue Cross|
|    547481328|          Blue Cross|
|  P02-88D77S7|Fresno County Tax...|
|        39104|Data Reproduction...|
|        40318|Data Reproduction...|
+-------------+--------------------+
only showing top 8 rows
```

In [11]:
```
# join invoices and InvoiceLineItems table on 'InvoiceID'
spark.sql("""
SELECT InvoiceNumber, InvoiceDate, InvoiceTotal
FROM Invoices I
JOIN InvoiceLineItems AS LI
ON I.InvoiceID = LI.InvoiceID
""").show(5)
```

```
+-------------+-------------------+------------+
|InvoiceNumber|        InvoiceDate|InvoiceTotal|
+-------------+-------------------+------------+
|   989319-457|2011-12-08 00:00:00|   $3,813.33|
|    263253241|2011-12-10 00:00:00|      $40.20|
|    963253234|2011-12-13 00:00:00|     $138.75|
|   2-000-2993|2011-12-16 00:00:00|     $144.70|
|    963253251|2011-12-16 00:00:00|      $15.50|
+-------------+-------------------+------------+
only showing top 5 rows
```

In [12]:
```
# Return columns Invoices;InvoiceNumber, InvoiceDate, InvoiceTotal. InvoiceLineItems; I
# Join Invoices on InvoiceLineItems
spark.sql("""
SELECT InvoiceNumber, InvoiceDate, InvoiceTotal, LI.InvoiceLineItemAmount
FROM Invoices IN
JOIN InvoiceLineItems AS LI
ON IN.InvoiceID = LI.InvoiceID
WHERE IN.InvoiceTotal > LI.InvoiceLineItemAmount
ORDER BY InvoiceNumber

""").show(5)
```

```
+-------------+-------------------+------------+--------------------+
|InvoiceNumber|        InvoiceDate|InvoiceTotal|InvoiceLineItemAmount|
+-------------+-------------------+------------+--------------------+
|       97/522|2012-02-28 00:00:00|   $1,962.13|           $1,197.00|
|   I77271-001|2011-12-26 00:00:00|     $662.00|             $478.00|
|   I77271-001|2011-12-26 00:00:00|     $662.00|              $50.00|
|   I77271-001|2011-12-26 00:00:00|     $662.00|              $58.40|
+-------------+-------------------+------------+--------------------+
```

In [101…
```
# Return columns Vendors; VendorName, Invoices; InvoiceNumber, InvoiceTotal, PaymentTot

spark.sql("""
select VendorName, I.InvoiceNumber, I.InvoiceTotal,InvoiceTotal - PaymentTotal - Credit
from Vendors AS V
JOIN Invoices AS I
ON V.VendorID = I.VendorId
```

```
ORDER BY VendorName
""").show(5)
```

```
+--------------------+-------------+------------+-------+
|          VendorName|InvoiceNumber|InvoiceTotal|Balance|
+--------------------+-------------+------------+-------+
|Abbey Office Furn...|    203339-13|      $17.50|   null|
|Bertelsmann Indus...|       509786|   $6,940.25|   null|
|          Blue Cross|    547479217|     $116.00|   null|
|          Blue Cross|    547481328|     $224.00|   null|
|          Blue Cross|    547480102|     $224.00|   null|
+--------------------+-------------+------------+-------+
only showing top 5 rows
```

In [14]:
```
# Return three columns Vendors; VendorName, DefaultAccountNo. GLAccounts; AccountDescri
# the result set should have one row for each vendor, with th account number and accoun
# for that vendor's default account number
# sort by AccountDescription, VendorName
spark.sql("""
select V.VendorName, V.DefaultAccountNo, GL.AccountDescription
FROM Vendors AS V
JOIN GLAccounts AS GL
ON V.DefaultAccountNo = GL.AccountNo
ORDER BY AccountDescription, VendorName
""").show(5)
```

```
+--------------------+---------------+-------------------+
|          VendorName|DefaultAccountNo| AccountDescription|
+--------------------+---------------+-------------------+
|Dristas Groom & M...|            591|         Accounting|
|         DMV Renewal|            568|   Auto License Fee|
| Newbrige Book Clubs|            394|Book Club Royalties|
|Bertelsmann Indus...|            400|Book Printing Costs|
|Courier Companies...|            400|Book Printing Costs|
+--------------------+---------------+-------------------+
only showing top 5 rows
```

In [15]:
```
spark.sql("""
SELECT *
FROM InvoiceLineItems
""").show(5)
```

```
+---------+---------------+---------+--------------------+--------------------------+
|InvoiceID|InvoiceSequence|AccountNo|InvoiceLineItemAmount|InvoiceLineItemDescription|
+---------+---------------+---------+--------------------+--------------------------+
|        1|              1|      553|           $3,813.33|                   Freight|
|        2|              1|      553|              $40.20|                   Freight|
|        3|              1|      553|             $138.75|                   Freight|
|        4|              1|      553|             $144.70|           Int'l shipment|
|        5|              1|      553|              $15.50|                   Freight|
+---------+---------------+---------+--------------------+--------------------------+
only showing top 5 rows
```

In [16]:
```
# Write a select statements the retuns five columns from three tables using alias
# Vendor VendorName column, Date InvoiceDate column, Number invoicenumber column, LineI
# Tables Vendors, Invoices, InvoiceLineItems Table
# sort by Vendor, Date, Number
spark.sql("""
SELECT VendorName AS Vendor, InvoiceDate AS Date, InvoiceNumber AS Number, InvoiceLineI
FROM Vendors AS V
JOIN Invoices AS I
```

```
ON V.VendorID = I.VendorID
JOIN InvoiceLineItems AS LI
ON I.InvoiceID = LI.InvoiceID
""").show(5)
```

```
+--------------------+-------------------+----------+---------+
|              Vendor|               Date|    Number| LineItem|
+--------------------+-------------------+----------+---------+
|United Parcel Ser...|2011-12-08 00:00:00|989319-457|$3,813.33|
|Federal Express C...|2011-12-10 00:00:00| 263253241|   $40.20|
|Federal Express C...|2011-12-13 00:00:00| 963253234|  $138.75|
|Federal Express C...|2011-12-16 00:00:00|2-000-2993|  $144.70|
|Federal Express C...|2011-12-16 00:00:00| 963253251|   $15.50|
+--------------------+-------------------+----------+---------+
only showing top 5 rows
```

In [23]:
```
# Calculate the average invoice amount(InvoiceTotal) by vendorID, Invoices table
# With average invoice total > 2000 sort by Average Invoice Amount
spark.sql("""
SELECT VendorID, AVG(InvoiceTotal) AS Total
FROM Invoices
GROUP BY VendorID
HAVING Total < 20000
ORDER BY Total
""").show()
```

```
+--------+-----+
|VendorID|Total|
+--------+-----+
+--------+-----+
```

In [19]:
```
# Summary query that calculates the number of invoices and the average invoiceTotal amo
# each state and city, columns VendorState, VendorCity. tables Invoices, Vendors. Invoi
spark.sql("""
SELECT VendorState, VendorCity, COUNT(*) AS Quantity, AVG(InvoiceTotal) AS Average
FROM  Invoices AS I
JOIN Vendors AS V
ON I.VendorID = V.VendorID
GROUP BY VendorState, VendorCity
HAVING Quantity >= 12
ORDER BY VendorState, VendorCity
""").show()
```

```
+-----------+----------+--------+-------+
|VendorState|VendorCity|Quantity|Average|
+-----------+----------+--------+-------+
|         CA|    Fresno|      19|   null|
|         TN|   Memphis|      47|   null|
+-----------+----------+--------+-------+
```

In [60]:
```
# Return the number of Vendors and to Highest InvoiceNumbers for Invoices
# Select the date 2012, count > 1. Sort in Descending order
spark.sql("""
SELECT VendorID,  COUNT(*) As Qty, Max(InvoiceTotal) AS Total
FROM Invoices
WHERE InvoiceDate BETWEEN '2012-01-01' AND '2012-12-31'
GROUP BY VendorID
HAVING Qty > 2
ORDER BY VendorId DESC
""").show()
```

```
+--------+---+----------+
|VendorID|Qty|     Total|
+--------+---+----------+
|     123| 39|   $739.20|
|     122|  8| $3,689.99|
|     121|  7|   $953.10|
|     115|  4|     $6.00|
|     110|  5|$37,966.19|
|      95|  5|    $46.21|
|      37|  3|   $224.00|
+--------+---+----------+
```

Summary

- GROUP BY is used with Aggregrates(sum, count, max,min)
- SELECT InvoiceNumber, COUNT(*) you are counting the number of InvoiceNumbers
- SELECT VendorState, VendorCity COUNT(*) you are counting the number of VendorState, VendorCity
- SELECT VendorCity, AVG(InvoiceTotal) you are finding the average of the InvoiceTotal column
- SELECT VendorCity, SUM(InvoiceNumber) you are finding the the TOTAL amount in the InvoiceNumber column
- GROUP WITH ROLL UP : Adds a summary row(total) to each GROUP
- GROUP BY WITH CUBE: Add a summary row(total) at the end
- JOINS the are usually : ON

In [73]:
```
# Return the number of VendorState and VendorCityfrom Vendors table, VendorStates are =
# Add a summary row to each State & City
spark.sql("""
SELECT VendorState, VendorCity, COUNT(*) AS QtyVendors
FROM Vendors
WHERE VendorState IN('PA','NY','OH')
GROUP BY VendorState, VendorCity WITH ROLLUP
ORDER BY VendorState DESC, VendorCity DESC
""").show()
```

```
+-----------+---------------+----------+
|VendorState|     VendorCity|QtyVendors|
+-----------+---------------+----------+
|         PA|   Philadelphia|         2|
|         PA|Fort Washington|         1|
|         PA|           null|         3|
|         OH|        Oberlin|         1|
|         OH|         Marion|         1|
|         OH|       Columbus|         2|
|         OH|         Cleves|         1|
|         OH|     Cincinnati|         2|
|         OH|           null|         7|
|         NY|      Tarrytown|         1|
|         NY|       New York|         1|
|         NY|   New Rochelle|         1|
|         NY|           null|         3|
|       null|           null|        13|
+-----------+---------------+----------+
```

In [85]:
```
# Calculate the number of invoices, largest invoices, smallest invoices in the invoice
```

```
# Summarize data in a result set.(OVER Function) Return Individual Rows, columns Invoic
spark.sql("""
SELECT  InvoiceNumber, InvoiceDate, InvoiceTotal,
MAX(InvoiceTotal) OVER (PARTITION BY InvoiceDate) AS MaxTotal,
COUNT(InvoiceTotal) OVER (PARTITION BY InvoiceDate) AS DateCount,
MIN(InvoiceTotal) OVER (PARTITION BY InvoiceDate) AS MinTotal
FROM Invoices
LIMIT 6

""").show()
```

```
+---------------+-------------------+------------+---------+---------+---------+
|   InvoiceNumber|        InvoiceDate|InvoiceTotal| MaxTotal|DateCount| MinTotal|
+---------------+-------------------+------------+---------+---------+---------+
|CBM9920-M-T77109|2012-02-23 00:00:00|     $290.00|  $290.00|        1|  $290.00|
|      4-327-7357|2012-03-16 00:00:00|     $162.75|  $162.75|        1|  $162.75|
|        25022117|2012-01-01 00:00:00|       $6.00|    $6.00|        1|    $6.00|
|      989319-487|2012-02-20 00:00:00|   $1,927.54|$1,927.54|        1|$1,927.54|
|      21-4923721|2012-01-13 00:00:00|       $9.95|    $9.95|        3|$1,750.00|
|           77290|2012-01-13 00:00:00|   $1,750.00|    $9.95|        3|$1,750.00|
+---------------+-------------------+------------+---------+---------+---------+
```