SQL_To_Pandas_Notebook

August 24, 2021

0.1 Overview

This notebook will guide you how you can re-write SQL code in pandas or vice-versa

The notebook is solved in Databricks. As it supports both SQL and Pandas

[0]:
```
# File location and type
file_location = "/FileStore/tables/Pandas_SQL-1.csv"
file_type = "csv"

# CSV options
infer_schema = "false"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be␣
↪ignored.
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

display(df)
```

[0]:
```
# Create a view or table

temp_table_name = "pandas_sql_csv"

df.createOrReplaceTempView(temp_table_name)
```

[0]:
```
%sql

/* Query the created temp table in a SQL cell */

select * from `pandas_sql_csv`
```

[0]:
```
```

```
# With this registered as a temp view, it will only be available to this␣
↪particular notebook. If you'd like other users to be able to query this␣
↪table, you can also create a table from the DataFrame.
# Once saved, this table will persist across cluster restarts as well as allow␣
↪various users across different notebooks to query this data.
# To do so, choose your table name and uncomment the bottom line.

permanent_table_name = "pandas_sql_1_csv"

# df.write.format("parquet").saveAsTable(permanent_table_name)
```

[0]:
```
import pandas as pd
df = (spark.read.option("header","true").csv(file_location))



df_pandas = df.select("*").toPandas()
```

[2]:
```
#If you want to load dataset in pandas using jupyter notebook
import pandas as pd
df_pandas =pd.read_csv("Pandas_SQL.csv")
```

Lets start by viewing the first 5 rows

[0]:
```
%sql

select * from `pandas_sql_csv` limit 5;
```

[0]:
```
df_pandas.head(5)
```

[0]:
```
%sql

select FirstName from `pandas_sql_csv`;
```

[3]:
```
df_pandas["FirstName"]
#or df_pandas.FirstName
```

[3]:
```
0     Chittaranjan
1             Mitu
2             Jeni
3        Adyashree
4              Ram
5         Jitendra
6            Dibas
7          Chandin
8     Chittaranjan
9        Adyashree
```

```
10    Chittaranjan
Name: FirstName, dtype: object
```

[0]:
```
%sql

select FirstName,LastName from `pandas_sql_csv`;
```

[0]:
```
df_pandas[["FirstName","LastName"]]
```

[0]:
```
%sql

select CONCAT(FirstName," ",LastName) as Full_Name from `pandas_sql_csv`
```

[4]:
```
df_pandas["FirstName"] + " " + df_pandas["LastName"]
```

[4]:
```
0     Chittaranjan Swain
1            Mitu Pradhan
2              Jeni Swain
3         Adyashree Swain
4               Ram Kumar
5          Jitendra Gouad
6            Dibas Hembram
7            Chandin Swain
8     Chittaranjan Swain
9         Adyashree Swain
10    Chittaranjan Swain
dtype: object
```

[0]:
```
%sql

select * from `pandas_sql_csv`
```

[0]:
```
%sql

select * from `pandas_sql_csv`
where Gender is null
```

[0]:
```
df_pandas[df_pandas["Gender"].isnull()]
```

[0]:
```
%sql

select COALESCE(Gender,"Unknown")  from `pandas_sql_csv`
```

[5]:
```
df_pandas["Gender"].fillna("Unknown", inplace = True)
df_pandas["Gender"]
```

[5]:
```
0        Male
1        Male
2      Female
3      Female
4     Unknown
5        Male
6     Unknown
7      Female
8        Male
9      Female
10       Male
Name: Gender, dtype: object
```

[0]:
```
%sql

select ifnull(Salary,0) as Salary from `pandas_sql_csv`
```

[6]:
```
df_pandas["Salary"].fillna(value = 0, inplace = True)
df_pandas["Salary"]
```

[6]:
```
0     80000.0
1     55000.0
2         0.0
3     49000.0
4     39000.0
5         0.0
6     55000.0
7     76000.0
8     80000.0
9     49000.0
10    80000.0
Name: Salary, dtype: float64
```

[0]:
```
%sql

select * from `pandas_sql_csv`
```

[0]:
```
%sql

select *,count(*) as No_of_duplicates_records from `pandas_sql_csv`
group by id,FirstName,LastName,Location,Gender,Salary
having count(*) > 1;
```

[0]:
```
df_pandas[df_pandas.duplicated()]
```

[0]:
```
%sql
```

```
select * from `pandas_sql_csv`
where FirstName like "A%"
```

```
[0]: df_pandas[df_pandas["FirstName"].str.startswith("A")]
```

```
[0]: %sql

     select * from `pandas_sql_csv`
     where FirstName like "%a"
```

```
[0]: df_pandas[df_pandas["FirstName"].str.endswith("a")]
```

```
[0]: %sql

     select * from `pandas_sql_csv`
     where FirstName like "%Ch%"
```

```
[0]: df_pandas[df_pandas["FirstName"].str.contains("Ch")]
```

```
[0]: %sql

     select upper(FirstName) as Name from `pandas_sql_csv`
```

```
[7]: df_pandas.FirstName.str.upper()
```

```
[7]: 0     CHITTARANJAN
     1             MITU
     2             JENI
     3        ADYASHREE
     4              RAM
     5         JITENDRA
     6            DIBAS
     7          CHANDIN
     8     CHITTARANJAN
     9        ADYASHREE
     10    CHITTARANJAN
     Name: FirstName, dtype: object
```

```
[0]: %sql

     select lower(FirstName) as Name from `pandas_sql_csv`
```

```
[8]: df_pandas.FirstName.str.lower()
```

```
[8]: 0     chittaranjan
     1             mitu
     2             jeni
```

```
3        adyashree
4              ram
5         jitendra
6            dibas
7          chandin
8     chittaranjan
9        adyashree
10    chittaranjan
Name: FirstName, dtype: object
```

Real World analyizng

```
[0]: # File location and type
file_location = "/FileStore/tables/FPL_DAY_DAY_DATASET.csv"
file_type = "csv"

# CSV options
infer_schema = "false"
first_row_is_header = "false"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be␣
 ↪ignored.
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location,header=True)

display(df)
```

```
[0]: temp_table_name = "FPL_DAY_DAY_DATASET_csv"

df.createOrReplaceTempView(temp_table_name)
```

```
[0]: # With this registered as a temp view, it will only be available to this␣
 ↪particular notebook. If you'd like other users to be able to query this␣
 ↪table, you can also create a table from the DataFrame.
# Once saved, this table will persist across cluster restarts as well as allow␣
 ↪various users across different notebooks to query this data.
# To do so, choose your table name and uncomment the bottom line.

permanent_table_name = "FPL_DAY_DAY_DATASET_csv"

# df.write.format("parquet").saveAsTable(permanent_table_name)
```

[0]:
```
%sql
select * from `FPL_DAY_DAY_DATASET_csv` limit 5
```

[0]:
```
df1 = (spark.read.option("header","true").csv("/FileStore/tables/
 ↪FPL_DAY_DAY_DATASET.csv"))



df_fpl = df1.select("*").toPandas()
df_fpl.head(5)
```

[9]:
```
df_fpl=pd.read_csv("FPL_DAY_DAY_DATASET.csv")
```

[0]:
```
%sql
select distinct(team) from `FPL_DAY_DAY_DATASET_csv`
```

[10]:
```
df_fpl.team.unique()
```

[10]:
```
array(['ARS', 'AVL', 'WBA', 'NEW', 'BHA', 'BUR', 'CHE', 'CRY', 'FUL',
       'SOU', 'EVE', 'LEE', 'LEI', 'LIV', 'SHU', 'MCI', 'MUN', 'WHU',
       'TOT', 'WOL'], dtype=object)
```

[0]:
```
%sql
select count(distinct(team)) as Number_of_teams from `FPL_DAY_DAY_DATASET_csv`
```

[0]:
```
df_fpl.team.nunique()
```

[0]:
```
%sql

SELECT pos, count(*)
FROM `FPL_DAY_DAY_DATASET_csv`
GROUP BY pos
```

[11]:
```
df_fpl.pos.value_counts()
```

[11]:
```
MID    9859
DEF    8626
FWD    3113
GKP    2767
Name: pos, dtype: int64
```

[0]:
```
%sql

SELECT max(team_a_score)
FROM `FPL_DAY_DAY_DATASET_csv`
```

[0]:
```
df_fpl.team_a_score.max()
```

```
[0]: %sql

     SELECT min(value)
     FROM `FPL_DAY_DAY_DATASET_csv`
```

```
[0]: df_fpl.value.min()
```

```
[0]: %sql

     SELECT sum(team_a_score) as Total_goals_scored_in_pl
     FROM `FPL_DAY_DAY_DATASET_csv`
```

```
[13]: df_fpl.team_a_score.sum()
```

```
[13]: 32440
```

```
[0]: %sql

     SELECT *
     FROM `FPL_DAY_DAY_DATASET_csv`
     order by fixture asc  limit 5;
```

```
[0]: df_fpl.sort_values(by="fixture",ascending=True).head(5)
```

```
[0]: %sql

     SELECT *
     FROM `FPL_DAY_DAY_DATASET_csv`
     order by
     transfers_in desc limit 5;
```

```
[0]: df_fpl.sort_values(by="transfers_in",ascending=False).head(5)
```

```
[0]: %sql

     SELECT *
     FROM `FPL_DAY_DAY_DATASET_csv`
     where name='Mohamed Salah' limit 5
```

```
[0]: df_fpl[df_fpl['name']=="Mohamed Salah"].head(5)
```

```
[0]: %sql

     SELECT *
     FROM `FPL_DAY_DAY_DATASET_csv`
     where name='Mohamed Salah' and opponent_team="ARS"
```

[0]:
```
df_fpl[(df_fpl['name']=="Mohamed Salah") & (df_fpl['opponent_team']=="ARS")]
```

[0]:
```
%sql

SELECT *
FROM `FPL_DAY_DAY_DATASET_csv`
where name='Mohamed Salah' and not opponent_team="ARS"
```

[0]:
```
df_fpl[(df_fpl['name']=="Mohamed Salah") & ~(df_fpl['opponent_team']=="ARS")].
 ↪head(5)
```

[0]:
```
%sql

SELECT *
FROM `FPL_DAY_DAY_DATASET_csv`
where name='Mohamed Salah' and total_points between 10 and 20
```

[16]:
```
df_fpl[(df_fpl['name']=="Mohamed Salah") & (df_fpl['total_points'].
 ↪between(10,20))]
```

[16]:
```
                name team  pos  game_week  fixture opponent_team  total_points  \
9617  Mohamed Salah  LIV  MID          1        3           LEE            20
9620  Mohamed Salah  LIV  MID          4       30           AVL            13
9627  Mohamed Salah  LIV  MID         11      103           WOL            14
9630  Mohamed Salah  LIV  MID         14      132           CRY            16
9637  Mohamed Salah  LIV  MID         21      208           WHU            15
9652  Mohamed Salah  LIV  MID         36      353           WBA            10

      was_home          kickoff_time  team_h_score  …  bps  influence  \
9617      True  2020-09-12T16:30:00Z             4  …   69      117.2
9620     False  2020-10-04T18:15:00Z             7  …   54       78.2
9627      True  2020-12-06T19:15:00Z             4  …   43       57.0
9630     False  2020-12-19T12:30:00Z             0  …   52       86.6
9637     False  2021-01-31T16:30:00Z             1  …   47       75.0
9652     False  2021-05-16T15:30:00Z             1  …   31       49.8

      creativity  threat  ict_index  value  transfers_balance  selected  \
9617        50.1   161.0       32.8    120                  0   1883241
9620        41.8    44.0       16.4    122             279768   2941545
9627        23.2    29.0       10.9    122              66389   2290651
9630        14.2    26.0       12.7    124             264071   2925498
9637        18.1    83.0       17.6    125              45974   3022589
9652        45.1    79.0       17.4    127             257608   3247999

      transfers_in  transfers_out
9617             0              0
9620        343946          64178
```

```
9627        152729        86340
9630        280489        16418
9637        146599       100625
9652        272307        14699

[6 rows x 33 columns]
```

[0]:
```
%sql

SELECT *
FROM `FPL_DAY_DAY_DATASET_csv`
where name='Mohamed Salah' or name='Jordan Henderson'
```

[0]:
```
df_fpl[(df_fpl['name']=="Mohamed Salah") | (df_fpl['name']=="Jordan Henderson")]
```

[0]:
```
%sql

SELECT *
FROM `FPL_DAY_DAY_DATASET_csv`
where name='Mohamed Salah' and opponent_team in('WBA','LEE','AVL')
```

[0]:
```
df_fpl[(df_fpl['name']=="Mohamed Salah") & (df_fpl['opponent_team'].
 ↪isin(['WBA','LEE','AVL']))]
```

[0]:
```
%sql

SELECT team,sum(goals_scored) as Number_of_goals_scored
FROM `FPL_DAY_DAY_DATASET_csv`
group by team
order by Number_of_goals_scored desc
```

[18]:
```
df_fpl.groupby("team")["goals_scored"].sum().sort_values(ascending=False)
```

[18]:
```
team
MCI    82
MUN    70
TOT    66
LEI    64
LIV    64
LEE    60
WHU    60
CHE    56
ARS    53
AVL    52
SOU    48
EVE    45
NEW    44
```

```
CRY    39
BHA    39
WOL    34
WBA    33
BUR    32
FUL    26
SHU    19
Name: goals_scored, dtype: int64
```

[0]:
```
%sql

SELECT team,avg(goals_conceded)
FROM `FPL_DAY_DAY_DATASET_csv`
group by team
```

[19]:
```
df_fpl.groupby("team")["goals_conceded"].mean()
```

[19]:
```
team
ARS    0.355178
AVL    0.407407
BHA    0.364248
BUR    0.522904
CHE    0.390702
CRY    0.602990
EVE    0.439533
FUL    0.450270
LEE    0.488889
LEI    0.472509
LIV    0.389456
MCI    0.290429
MUN    0.355360
NEW    0.538099
SHU    0.623762
SOU    0.610706
TOT    0.411814
WBA    0.673600
WHU    0.454705
WOL    0.424647
Name: goals_conceded, dtype: float64
```

[0]:
```
%sql

SELECT team,name,sum(goals_scored) as No_of_goals
FROM `FPL_DAY_DAY_DATASET_csv`
group by team,name
order by No_of_goals desc
```

[21]:
```
df_fpl.groupby(["team","name"])["goals_scored"].sum().
↪sort_values(ascending=False)
```

[21]:
```
team  name
TOT   Harry Kane                  23
LIV   Mohamed Salah               22
MUN   Bruno Miguel Fernandes      18
LEE   Patrick Bamford             17
TOT   Heung-Min Son               17
                                  ..
LEE   Sam Greenwood                0
SOU   Shane Long                   0
      Ryan Finnigan                0
LEI   Adrien Silva                 0
LIV   Billy Koumetio               0
Name: goals_scored, Length: 713, dtype: int64
```

Joins

[0]:
```
# File location and type
file_location = "/FileStore/tables/sales_data_set.csv"
file_type = "csv"

# CSV options
infer_schema = "false"
first_row_is_header = "false"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be␣
↪ignored.
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location,header=True)

display(df)
```

[0]:
```
temp_table_name = "sales_data_set_csv"

df.createOrReplaceTempView(temp_table_name)
```

[0]:
```
%sql

/* Query the created temp table in a SQL cell */

select * from `sales_data_set_csv`
```

[0]:
```
# With this registered as a temp view, it will only be available to this␣
↪particular notebook. If you'd like other users to be able to query this␣
↪table, you can also create a table from the DataFrame.
# Once saved, this table will persist across cluster restarts as well as allow␣
↪various users across different notebooks to query this data.
# To do so, choose your table name and uncomment the bottom line.

permanent_table_name = "sales_data_set_csv"
```

[0]:
```
import pandas as pd
df2 = (spark.read.option("header","true").csv("/FileStore/tables/sales_data_set.
↪csv"))




df_sales = df2.select("*").toPandas()
df_sales
```

[0]:
```
# File location and type
file_location = "/FileStore/tables/stores_data_set.csv"
file_type = "csv"

# CSV options
infer_schema = "false"
first_row_is_header = "false"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be␣
↪ignored.
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location,header=True)

display(df)
```

[0]:
```
# Create a view or table

temp_table_name = "stores_data_set_csv"

df.createOrReplaceTempView(temp_table_name)
```

[0]:
```
%sql

/* Query the created temp table in a SQL cell */
```

```
select * from `stores_data_set_csv`
```

[0]:
```
# With this registered as a temp view, it will only be available to this␣
↪particular notebook. If you'd like other users to be able to query this␣
↪table, you can also create a table from the DataFrame.
# Once saved, this table will persist across cluster restarts as well as allow␣
↪various users across different notebooks to query this data.
# To do so, choose your table name and uncomment the bottom line.

permanent_table_name = "stores_data_set_csv"

# df.write.format("parquet").saveAsTable(permanent_table_name)
```

[0]:
```
import pandas as pd
df3 = (spark.read.option("header","true").csv("/FileStore/tables/
↪stores_data_set.csv"))



df_store = df3.select("*").toPandas()
df_store.head()
```

[23]:
```
#loading the dataset in juptyer

import pandas as pd
df_sales=pd.read_csv("stores data-set.csv")
df_store=pd.read_csv("sales data-set.csv")
```

[0]:
```
%sql

select b.Type,sum(a.Weekly_Sales) as Total_sales from sales_data_set_csv as a
inner join stores_data_set_csv b on a.Store=b.Store
group by b.Type
order by b.type
```

[24]:
```
Join=df_sales.merge(df_store,on='Store',how='inner')
Join.head()
```

[24]:
```
   Store Type    Size  Dept        Date  Weekly_Sales  IsHoliday
0      1    A  151315     1  05/02/2010      24924.50      False
1      1    A  151315     1  12/02/2010      46039.49       True
2      1    A  151315     1  19/02/2010      41595.55      False
3      1    A  151315     1  26/02/2010      19403.54      False
4      1    A  151315     1  05/03/2010      21827.90      False
```

[25]:
```
Join.groupby('Type')["Weekly_Sales"].sum().sort_values(ascending=False)
```

```
[25]: Type
      A    4.331015e+09
      B    2.000701e+09
      C    4.055035e+08
      Name: Weekly_Sales, dtype: float64
```

```
[0]:
```