databricksgenerate dataframe

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

Attribution data

```
#Generate sample data
accounts = [
            (226,"aaa"),
            (971,"bbb"),
            (598,"ccc"),
            (999,"ddd")
           ]
#Create schema for accounts
acc_schema = StructType([
    StructField("account_id",IntegerType(),True),
    StructField("account_name",StringType(),True)
  ])

acc_df = spark.createDataFrame(data=accounts,schema=acc_schema)
acc_df.printSchema()
acc_df.show(truncate=False)
```

```
root
 |-- account_id: integer (nullable = true)
 |-- account_name: string (nullable = true)

+----------+------------+
|account_id|account_name|
+----------+------------+
|226       |aaa         |
|971       |bbb         |
|598       |ccc         |
|999       |ddd         |
+----------+------------+
```

```
#Generate sample data
source = [
         (226,"utmcsr","facebook"),
         (226,"utmcmd","cpc"),
         (971,"utmcsr","google"),
         (971,"utmcmd","cpc"),
         (598,"utmcmd","cpc"),
         (598,"utmcsr","google"),
         (999,"utmcsr","twitter")
         ]

#Create schema for source
source_schema = StructType([
    StructField("account_id",IntegerType(),True),
    StructField("utm_name",StringType(),True),
    StructField("utm_value",StringType(),True)
  ])

source_df = spark.createDataFrame(data=source,schema=source_schema)
source_df.printSchema()
source_df.show(truncate=False)
```

```
root
 |-- account_id: integer (nullable = true)
 |-- utm_name: string (nullable = true)
 |-- utm_value: string (nullable = true)

+----------+--------+---------+
|account_id|utm_name|utm_value|
+----------+--------+---------+
|226       |utmcsr  |facebook |
|226       |utmcmd  |cpc      |
|971       |utmcsr  |google   |
|971       |utmcmd  |cpc      |
|598       |utmcmd  |cpc      |
|598       |utmcsr  |google   |
|999       |utmcsr  |twitter  |
+----------+--------+---------+
```

```
#create temp view for SQL code
source_df.createOrReplaceTempView("source")
acc_df.createOrReplaceTempView("accounts")
```

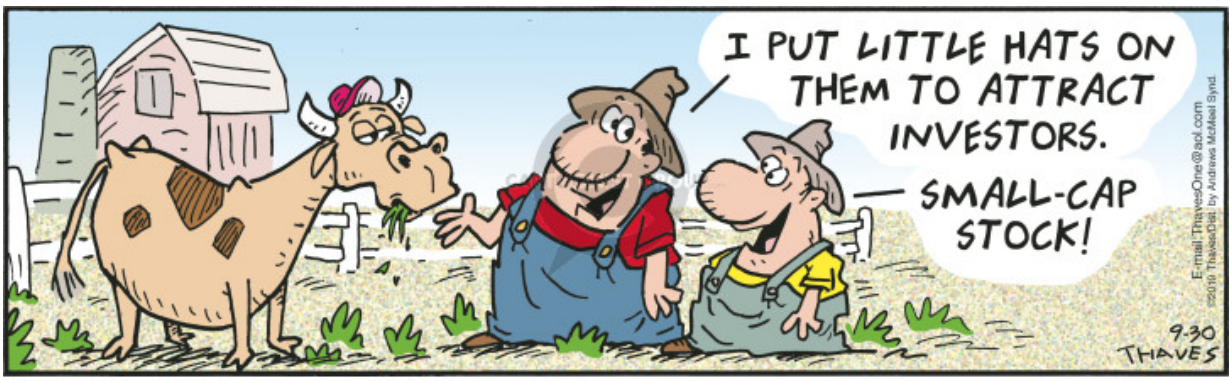
```
%sql

select
account_name,
max(utm_source) as utm_source,
max(utm_value) as utm_value
FROM (
      select
      account_name,
      CASE WHEN utm_name = 'utmcsr' THEN utm_value END as utm_source,
      CASE WHEN utm_name = 'utmcmd' THEN utm_value END as utm_value
      FROM accounts a
      INNER JOIN source s ON a.account_id = s.account_id
      ) as marketing
GROUP BY account_name
ORDER BY utm_source ASC
```

	account_name	utm_source	utm_value
1	aaa	facebook	cpc
2	bbb	google	cpc
3	ccc	google	cpc
4	ddd	twitter	null

Showing all 4 rows.

Argriculture company



For API call I might use `request` Python library.

1. Get credentials in secure way

```
import requests
grant_type="password"
username = dbutils.secrets.get(scope="<azure key vault or aws secret mar
password = dbutils.secrets.get(scope="<azure key vault or aws secret mar
client_id = dbutils.secrets.get(scope="<azure key vault or aws secret ma
client_secret = dbutils.secrets.get(scope="<azure key vault or aws secre
```

2. Prepare JSON for passing with API call for obtaining token

```
info = {
"username" : username,
"password" : password,
"client_id" : client_id,
"client_secret" : client_secret
}
```

3. Request token

```
request_token_url = "https://<my API url>/oauth/token"
response = requests.post(request_token_url, data=info)
respone_output = response.json()
```

4. Use `token` and get data using my query

```
headers = {'Authorization': token, 'Content-Type': 'application/json'}
#I can constract body and query using Python dict.
body = {}
url = "https://<API URL for quering data>"
#Run POST command
response = requests.post(url, json = body, headers=headers)
#Get API output in JSON
data = response.json()
```

I assume that `data` returns me:

```
[
{name: 'Pig', inventory: 3},
{name: 'Cow', inventory: 4},
{name: 'Chicken', inventory: -1},
{name: 'Dog',inventory: 1}
]
```

PS I would change this format to the something like `{"name": "Pig", "inventory": 3}`, i.e. with strings in quotes.

Farm - Solution 1

```
#Conver to the dict, adding
input = """[{name: 'Pig', inventory: 3}, {name: 'Cow', inventory: 4}, {name:
'Chicken', inventory: -1}, {name: 'Dog',inventory: 1}]"""

augm_input = input.replace("name","\'name\'")
augm_input = augm_input.replace("inventory","\'inventory\'")

#Replacesingle quote to the doble in order to use json.loads
augm_input = augm_input.replace("\'","\"")
augm_input


import json

#Read input string
api_list = json.loads(augm_input)


output_list = []

for dic in api_list:
    new_dict = {}
    for value in dic.items():
        new_dict['name'] = dic['name']
        if dic['inventory'] <= 0:
            new_dict['inventory_level'] = 'None'
        elif dic['inventory'] in [1,2]:
            new_dict['inventory_level'] = 'Low'
        elif dic['inventory'] >= 3:
            new_dict['inventory_level'] = 'Normal'
    #Append dict to the list
    output_list.append(new_dict)



#Return new list with new values
output_list



output =
str(output_list).replace("\'name\'","name").replace("\'inventory_level\'","inve
ntory_level")
```

```
if output == """[{name: 'Pig', inventory_level: 'Normal'}, {name: 'Cow',
inventory_level: 'Normal'}, {name: 'Chicken', inventory_level: 'None'}, {name:
'Dog', inventory_level: 'Low'}]""":
    print("Assertion passed.")
```

```
Assertion passed.
```

Farm - Solution 2

```
import json
import re

# function for remapping level
def get_level_descr(value):
    """
    Double quotas for wright replace
    """
    if value <= 0:
        return "'None'"
    if value in set([1, 2]):
        return "'Low'"
    else:
        return "'Normal'"

# raw json string from Farm API
input = """[{name: 'Pig', inventory: 3}, {name: 'Cow', inventory: 4}, {name:
'Chicken', inventory: -1}, {name: 'Dog', inventory: 1}]"""

pattern_key = 'inventory'
replace_pattern = 'inventory_level'

# replace 'invetory' -> 'inventory_level'
output = re.sub(pattern_key, replace_pattern, output)

# find digit_level and replace his with mapping function
for digit_level in re.findall(r'([-]?\d{1})', output):
    output = re.sub(digit_level, get_level_descr(int(digit_level)), output)

print(output)

# check result
if output == """[{name: 'Pig', inventory_level: 'Normal'}, {name: 'Cow',
inventory_level: 'Normal'}, {name: 'Chicken', inventory_level: 'None'}, {name:
'Dog', inventory_level: 'Low'}]""":
    print("Assertion passed.")
```

```
[{name: 'Pig', inventory_level_level_level_level: 'Normal'}, {name: 'Cow', inv
entory_level_level_level_level: 'Normal'}, {name: 'Chicken', inventory_level_l
evel_level_level: 'None'}, {name: 'Dog', inventory_level_level_level_level: 'L
ow'}]
```

Farm - Solution 3

```
[{name: 'Pig', inventory_level: 'Normal'}, {name: 'Cow', inventory_level: 'Nor
mal'}, {name: 'Chicken', inventory_level: 'None'}, {name: 'Dog', inventory_lev
el: 'Low'}]
Assertion passed.
```