Q-2 [Spark DataFrame] Using the same country data set as in lab3 (country.json, city.json, and countrylanguage.json), write a Spark DataFrame script for each of the following questions. You use "import pyspark.sql.functions as fc". You need to display the full content of the columns (e.g., using show(truncate=False)).

```
>>> country = spark.read.json('country.json')
>>> city = spark.read.json('city.json')
>>> lang = spark.read.json('countrylanguage.json')
```

a. Find top-10 most popular official language, ranked by the number of countries where the language is official. Return the language and count in the descending order of the count.

Note: As nothing has been provided in the question, I am breaking ties by the name of language.

```
>>> lang.filter('IsOfficial =
"T"').groupby('Language').agg(fc.count('CountryCode').alias('count')).orderBy(['count','L
anguage'],ascending=[False,True]).show(10,truncate=False)

+----------+-----+
|Language  |count|
+----------+-----+
|English   |44   |
|Arabic    |22   |
|Spanish   |20   |
|French    |18   |
|German    |6    |
|Portuguese|6    |
|Dutch     |4    |
|Italian   |4    |
|Malay     |4    |
|Danish    |3    |
+----------+-----+
only showing top 10 rows
```

b. Find names of countries and their capital cities, for all countries in North America and having a GNP of at least 100,000. Output country and capital city names only.

```
>>> country_reduced = country.filter((country.Continent == "North America") &
(country.GNP >= 100000)).select(fc.col("Name").alias("Country"),"Capital")
>>> country_reduced.join(city,country_reduced.Capital == city.ID,
how="inner").select("Country",fc.col("Name").alias("Capital")).show(truncate=False)

+-------------+-----------------+
|      Country|          Capital|
+-------------+-----------------+
|       Canada|           Ottawa|
|       Mexico|Ciudad de MÃ©xico|
|United States|       Washington|
+-------------+-----------------+
```

c. Find names of countries in North America continent where English is an official language.

```
>>> countryNA = country.filter(country.Continent == "North
America").select("Name","Code")
>>> langofficial = lang.filter((lang.IsOfficial == "T") & (lang.Language ==
"English")).select("CountryCode")
>>> countryNA.join(langofficial,countryNA.Code ==
langofficial.CountryCode,how="inner").select("Name").show(truncate=False)
```

```
+------------------------------+
|Name                          |
+------------------------------+
|Anguilla                      |
|Antigua and Barbuda           |
|Belize                        |
|Bermuda                       |
|Barbados                      |
|Canada                        |
|Cayman Islands                |
|Saint Kitts and Nevis         |
|Saint Lucia                   |
|Montserrat                    |
|Turks and Caicos Islands      |
|United States                 |
|Saint Vincent and the Grenadines|
|Virgin Islands, British       |
|Virgin Islands, U.S.          |
+------------------------------+
```

d. Find the maximum population over all cities in USA.

```
>>>city.filter(city.CountryCode=="USA").select("Population").agg(fc.max("Population").ali
as("Max Population")).show()

+--------------+
|Max Population|
+--------------+
|       8008278|
+--------------+
```

e. Find country codes of the countries where both English and French are official languages.

```
>>>langOfficial = lang.filter((lang.IsOfficial == "T") & ((lang.Language=="English") |
(lang.Language=="French"))).select("CountryCode","Language")

>>>langOfficial.groupBy("CountryCode").agg(fc.size(fc.collect_list(fc.col("Language"))).a
lias("size")).filter("size == 2").select("CountryCode").show()


+-----------+
|CountryCode|
+-----------+
|        VUT|
|        SYC|
|        CAN|
+-----------+
```

Q-3. [Spark RDD] Using the country data set, write a Spark RDD script for each of the following questions. You use "import pyspark.sql.functions as fc".

```
>>> countryr = country.rdd
>>> cityr = city.rdd
>>> langr = lang.rdd
```

a. Find out how many countries have a GNP between 10,000 and 20,000 inclusive.

```
>>> countryr.filter(lambda r: r["GNP"]>=10000 and r["GNP"]<=20000).count()
```

With collect:

```
>>> countryr.filter(lambda r: r["GNP"]>=10000 and r["GNP"]<=20000).map(lambda
x:("count",1)).reduceByKey(lambda x,y:x+y).collect()
```

```
[('count', 20)]
```

b. For each continent, find the maximum GNP of countries in the continent.

```
>>> (countryr.map(lambda x:[x["Continent"],x["GNP"]]).map(lambda
x:(x[0],x)).reduceByKey(lambda x1,x2: max(x1,x2,key=lambda x:x[1])).values()).collect()
```

```
[['North America', 8510700.0],
 ['Asia', 3787042.0],
 ['Africa', 116729.0],
 ['Europe', 2133367.0],
 ['South America', 776739.0],
 ['Oceania', 351182.0],
 ['Antarctica', 0.0]]
```

c. Find the first 20 countries and names of their capital cities, ordered by the names of countries, descending.

```
>>> countryr.map(lambda x:[x[0],x[11]]).join(cityr.map(lambda x:[x[2],x[3]])).map(lambda
x:x[1]).sortBy(lambda x:x[0],ascending=False).take(20)
```

```
[('Zimbabwe', 'Harare'),
('Zambia', 'Lusaka'),
('Yugoslavia', 'Beograd'),
('Yemen', 'Sanaa'),
('Western Sahara', 'El-AaiÃºn'),
('Wallis and Futuna', 'Mata-Utu'),
('Virgin Islands, U.S.', 'Charlotte Amalie'),
('Virgin Islands, British', 'Road Town'),
('Vietnam', 'Hanoi'),
('Venezuela', 'Caracas'),
('Vanuatu', 'Port-Vila'),
('Uzbekistan', 'Toskent'),
('Uruguay', 'Montevideo'),
('United States', 'Washington'),
('United Kingdom', 'London'),
('United Arab Emirates', 'Abu Dhabi'),
('Ukraine', 'Kyiv'),
('Uganda', 'Kampala'),
('Tuvalu', 'Funafuti'),
('Turks and Caicos Islands', 'Cockburn Town')]
```

d. Find the maximum population of cities in USA.

```
>>> cityr.filter(lambda x:x["CountryCode"]=="USA").map(lambda x:x["Population"]).max()
```

```
8008278
```

Using collect:

```
>>> cityr.filter(lambda x:x["CountryCode"]=="USA").map(lambda
x:(x["CountryCode"],x["Population"])).reduceByKey(lambda x,y:max(x,y)).collect()
```

```
[('USA', 8008278)]
```

e. Find country codes of the countries where both English and French are official languages.

```
>>> langofficialr = langr.filter(lambda x:((x["IsOfficial"]=="T") and
((x["Language"]=="English") | (x["Language"]=="French")))).groupBy(lambda
x:x[0]).mapValues(lambda x:len(x)).filter(lambda x:x[1]==2)


>>> langofficialr.map(lambda x:x[0]).collect()

['CAN', 'SYC', 'VUT']
```