

Τμήμα Μηχανικών Η/Υ και Πληροφορικής

ΔΠΜΣ ΥΔΑ

Αποκεντρωμένα Συστήματα Διαχείρισης Μεγάλου Όγκου
Δεδομένων

Άσκηση Ακαδημαϊκού Έτους 2020-2021

Πετράκης Κωνσταντίνος

ΑΜ: 1041589

Για την υλοποίηση του Project εγκατέστησα το spark τοπικά στον υπολογιστή μου. Όλα τα ζητούμενα υλοποιήθηκαν με χρήση του pyspark. Επίσης για το κάθε ένα από τα 2 dataset έχω συμπεριλάβει ένα αρχείο preprocessing.py το οποίο αφορά την προ-επεξεργασία κάθε φορά του αντίστοιχου συνόλου δεδομένων, όπως εξηγώ παρακάτω. Για να φορτώσω και τα 2 σύνολα δεδομένων στο pyspark μετέτρεψα πρώτα τα .xlsx αρχεία σε .csv με τη χρήση της βιβλιοθήκης pandas.

Σημείωση: Γνωρίζω ότι στο databricks μπορούσα να τα φορτώσω κατευθείαν χρησιμοποιώντας κάτι σαν αυτό

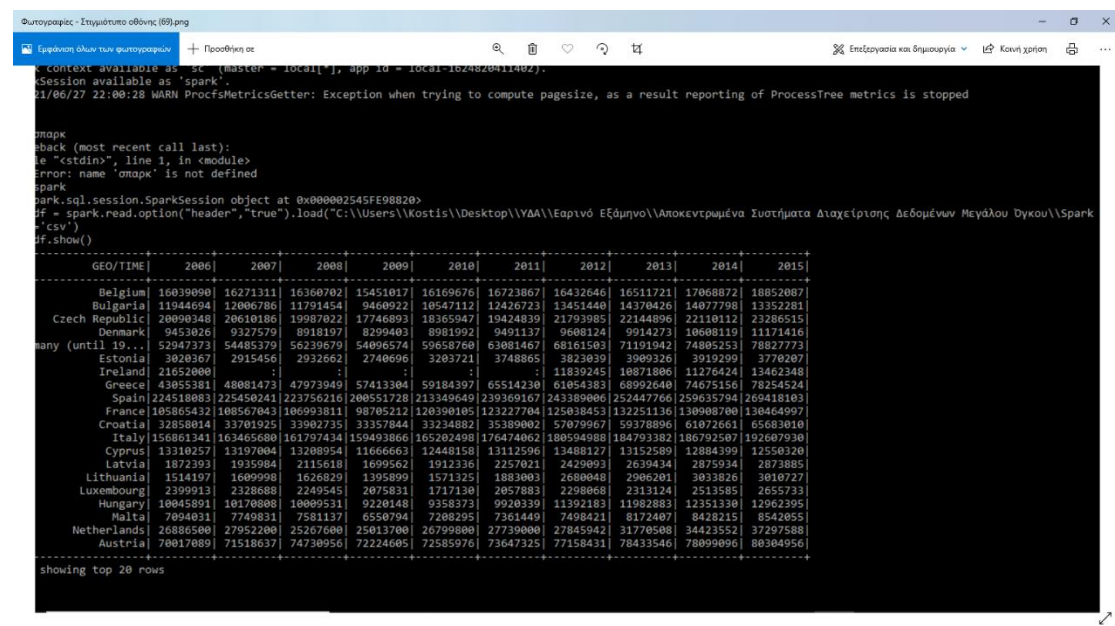
```
df = spark.read.format("com.crealytics.spark.excel") \
.option("useHeader", "true") \
.option("inferSchema", "true") \
.load("tour_occ_ninat.xlsx"))
```

Αλλά δυσκολεύτηκα πολύ να εγκαταστήσω τα κατάλληλα πακέτα τοπικά στον υπολογιστή μου γι' αυτό αρκέστηκα στην παραπάνω λύση.

Μέρος Α

Προ-επεξεργασία

Αρχικά μετέτρεψα το .xlsx αρχείο σε .csv αρχείο με τη χρήση της βιβλιοθήκης pandas στην Python και στη συνέχεια κατασκεύασα το αντίστοιχο dataframe στο pyspark. Τα περιεχόμενα αυτού του dataframe φαίνονται στο ακόλουθο στιγμιότυπο.



```
context available as 'sc' (master = local[*], app id = local-1624820411402).
Session available as 'spark'.
21/06/27 22:00:28 WARN ProfScMetricsGetter: Exception when trying to compute pagesize, as a result reporting of ProcessTree metrics is stopped

spark
In [ ]: spark
Out[ ]: spark
spark.sql.session.SparkSession object at 0x000002545FE98820>
df = spark.read.option("header", "true").load("C:\\Users\\Kostis\\Desktop\\ΥΔΑ\\Εαρινό Εξάμηνο\\Αποκεντρωμένα Συστήματα Διαχείρισης Δεδομένων Μεγάλου Όγκου\\Spark\\tour_occ_ninat.csv")
df.show()
```

GEO/TIME	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
Belgium	16039090	16271311	16360702	15451017	16109670	16723867	16432646	16511721	17068872	18852087
Bulgaria	11944694	12006786	11791454	9460922	10547112	12426723	13451440	14370426	14077798	13352281
Czech Republic	20090348	20610186	19987022	17746893	18365947	19424839	21793985	22144896	22110112	23286515
Denmark	9453026	9327579	8918197	8299403	8981992	9491137	9608124	9914273	10608119	11171416
Germany (until 19...	52947373	54485379	56239679	54096574	59658760	63081467	68161503	71191942	74805253	78827773
Estonia	3020367	2915456	2932662	2740696	3203721	3748865	3823039	3909326	3919299	3770207
Ireland	21652000	21652000	21652000	21652000	21652000	21652000	21652000	21652000	21652000	21652000
Greece	43055381	48081473	47973949	57413304	59184397	65514230	61854383	68992640	74675156	78254524
Spain	224518083	225450241	223756216	200551728	213349649	239369167	243389006	252447766	259635794	269418103
France	105865432	108567043	106993811	98705212	120390105	123227704	125038453	132251136	130908700	130464997
Croatia	32858014	33701925	33902735	33357844	33234882	35389002	57079967	59378896	61072661	65683010
Italy	156861341	163465680	161797434	159493866	165202498	176474062	180594988	184793382	186792507	192607930
Cyprus	13310257	13197004	13208954	11666663	12448150	13112596	13488127	13152589	12804399	12550320
Latvia	1872393	1935984	2115618	1699562	1912336	2257021	2429093	2639434	2875934	2873885
Lithuania	1514197	1609998	1626829	1395899	1571325	1883003	2680048	2906201	3033826	3010727
Luxembourg	2399913	2328688	2249545	2075831	1717130	2057883	2298068	2313124	2513585	2655733
Hungary	10045891	10170808	10009531	9220148	9358373	9920339	11392183	11982883	12351330	12962395
Malta	7094031	7749831	7501137	6550794	7208295	7361440	7490421	8172407	8428215	8542855
Netherlands	26806500	27952200	25267600	25013700	26799000	27739000	27045942	31770500	34423521	37297588
Austria	70017089	71518637	74730956	72224605	72585076	73647325	77158431	78433546	78099006	80304956

showing top 20 rows

Το πρώτο που παρατηρώ εξετάζοντας το σχήμα (printSchema) του dataframe είναι πως ο τύπος δεδομένων όλων των στηλών είναι string. Επειδή έχουμε να κάνουμε με πλήθος διανυκτερεύσεων μετέτρεψα τον τύπο των στηλών που αντιστοιχούν στις χρονιές σε ακέραιο (int). Στο ακόλουθο στιγμιότυπο φαίνεται το νέο σχήμα του dataframe.

Ερώτημα 1

Για το παρόν ερώτημα πρόσθεσα απλά στο dataframe μια νέα στήλη με το όνομα 'mean_07-14' παίρνοντας το άθροισμα των στηλών 2007 έως 2014 (με χρήση της sql function col) και διαιρώντας με το πλήθος τους (δηλαδή το 8). Το αποτέλεσμα φαίνεται στο ακόλουθο στιγμιότυπο.

```
Φωτογραφίες - Στιγμιότυπο οθόνης (7/3.png)
Εμφάνιση όλων των φωτογραφιών + Προσθήκη σε
>>> col()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'col' is not defined
>>> from pyspark.sql.functions import *
>>>
>>> df.withColumn("mean_07-14", (col('2007')+col('2008')+col('2009')+col('2010')+col('2011')+col('2012')+col('2013')+col('2014'))/8).show()
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|GEO/TIME|2006|2007|2008|2009|2010|2011|2012|2013|2014|2015|mean_07-14|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|Belgium|16039090|16271311|16360702|15451017|16169676|16723867|16432646|16511721|17068872|18852087|1.63737265E7|
|Bulgaria|11944694|12006786|11791454|9460922|10547112|12426723|13451448|14370426|14077798|13352281|1.2266582625E7|
|Czech Republic|20090348|20610186|19987022|17746893|18365947|19424839|21793985|22144896|22110112|23286515|2.0272985E7|
|Denmark|9453026|9327579|8918197|8299403|8981992|9491137|9608124|9914273|10608119|11171416|9393603.0|
|Germany (until 19...|52947373|54485379|56239679|54096574|59658760|63081467|68161503|71191942|74805253|78827773|6.2715069625E7|
|Estonia|3020367|2915456|2932662|2740696|3203721|3748865|3823039|3909326|3919299|3770207|3399133.0|
|Ireland|21652000|0|0|0|0|0|11839245|10871806|11276424|13462348|4248434.375|
|Greece|43055381|48081473|47973949|57413304|59184397|65514230|61054383|68992640|74675156|78254524|6.03611915E7|
|Spain|224518083|225450241|223756216|200551728|213349649|239369167|243389006|252447766|259635794|269418103|2.32243695875E8|
|France|105805432|108567043|106993811|98705212|120390105|123227704|125038453|132251136|130908700|130464997|1.182602705E8|
|Croatia|32858014|33701925|33902735|33357844|33234882|35389002|57079967|59378896|61072661|65683010|4.3389739E7|
|Italy|156801341|163465680|161797434|159493866|165202498|176474062|180594988|184793382|186792507|192607930|1.72326802125E8|
|Cyprus|13310257|13197004|13208054|11666663|12448158|13112596|13488127|13152589|12884399|12550320|1.289481125E7|
|Latvia|1872393|1935984|2115618|1699562|1912336|2257021|2420093|2639434|2875934|2873885|2233122.75|
|Lithuania|1514197|1609998|1626829|1395899|1571325|1883003|2680048|2966201|3033826|3010727|2088391.125|
|Luxembourg|2399913|2328688|2249545|2075831|1717130|2057883|2298008|2313124|2513585|2655733|2194231.75|
|Hungary|10045891|10170808|10009531|9220148|9358373|9920339|11302183|11982883|12351330|12962395|1.0550699375E7|
|Malta|7094031|7749831|7581137|6550794|7208295|7361449|7498421|8172407|8428215|8542055|7568818.625|
|Netherlands|26886500|27952200|25267600|25013700|26799800|27739000|27845942|31770508|34423552|37297588|2.835153775E7|
|Austria|70017089|71518637|74730956|72224605|72585976|73647325|77158431|78433546|78099006|80304956|7.47998215E7|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 20 rows
>>>
```

Ερώτημα 2

Για αυτό το ερώτημα αρχικά κατασκευάζω ένα νέο dataframe με τα στοιχεία 6 χωρών, της Ελλάδας και 5 ακόμη της επιλογής μου. Όσον αφορά τις 5 χώρες επιλογής μου χρησιμοποίησα 2 διαφορετικές λίστες χωρών απλά για να φανούν τα 2 ξεχωριστά στιγμιότυπα λειτουργίας (να είναι δηλαδή η Ελλάδα η χώρα με τις περισσότερες διανυκτερεύσεις για κάποιες χρονιές, στη μία περίπτωση, και στην άλλη να μην είναι).

Αρχικά επιλέγω το Βέλγιο, τη Δανία, τη Γερμανία, τη Κροατία και τη Κύπρο. Κατασκευάζω το νέο dataframe df_2 φιλτράροντας (με χρήση της συνάρτησης filter) το αρχικό dataframe για εκείνες τις χώρες της στήλης 'GEO/TIME' που περιέχονται στη λίστα με τις χώρες που έχω επιλέξει.

Αφού κατασκευάσω το df_2 ορίζω μια νέα άδεια λίστα years_greece_was_greatest στην οποία θα προστίθεται οι χρονιές για τις οποίες η Ελλάδα είχε το μέγιστο αριθμό διανυκτερεύσεων. Διατρέχω τις στήλες του dataframe που αντιστοιχούν στις χρονιές και ταξινομώ ως προς την κάθε χρονιά. Με την εντολή df.sort(df[year].desc()).collect()[0][0] λαμβάνω το όνομα της χώρας με τις περισσότερες διανυκτερεύσεις για την χρονιά (year) στην οποία βρίσκεται ο βρόγχος. Ελέγχω αν αυτή η χώρα είναι η Ελλάδα. Αν είναι τότε προσθέτω την αντίστοιχη χρονιά year στη λίστα years_greece_was_greatest. Αφού εκτελέσω αυτή τη διαδικασία για όλες τις χρονιές (τελειώσει ο βρόγχος) η λίστα years_greece_was_greatest περιέχει τις χρονιές που η Ελλάδα είχε τις περισσότερες διανυκτερεύσεις σε σχέση με τις 5 επιλεγμένες χώρες. Το μέγεθος της λίστας αντιστοιχεί στο πόσες χρονιές ήταν οι διανυκτερεύσεις στην Ελλάδα περισσότερες.

Στα παρακάτω στιγμιότυπα φαίνεται το ενδιάμεσο dataframe df_2 με τις επιλεγμένες χώρες και το τελικό αποτέλεσμα αφού εκτυπώσω τη λίστα years_greece_was_greatest και το μήκος της.

```
Φωτογραφίες - Στιγμιότυπο οθόνης (77).png
Εργασία όλων των φωτογραφιών Προσθήκη σε
only showing top 20 rows
>>> selected_countries = ['Greece', 'Belgium', 'Denmark', 'Croatia', 'Germany (until 1990 former territory of the FRG)', 'Cyprus']
>>>
>>> df_2 = df.filter(col('GEO/TIME').isin(selected_countries))
>>> df_2.show()
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| GEO/TIME | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 | 2013 | 2014 | 2015 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Belgium | 16039090 | 16271311 | 16360702 | 15451017 | 16169676 | 16723867 | 16432646 | 16511721 | 17068872 | 18852087 |
| Denmark | 9453026 | 9327579 | 8918197 | 8299403 | 8981992 | 9491137 | 9608124 | 9914273 | 10608119 | 11171416 |
| Germany (until 19... | 52947373 | 54485379 | 56239679 | 54096574 | 59658760 | 63081467 | 68161503 | 71191942 | 74805253 | 78827773 |
| Greece | 43055381 | 48081473 | 47973949 | 57413304 | 59184397 | 65514230 | 61054383 | 68992640 | 74675156 | 78254524 |
| Croatia | 32858014 | 33701925 | 33902735 | 33357844 | 33234882 | 35389002 | 57079967 | 59378896 | 61072661 | 65683010 |
| Cyprus | 13310257 | 13197004 | 13208954 | 11666663 | 12448158 | 13112596 | 13488127 | 13152589 | 12884399 | 12550320 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
>>> -
```

```
Φωτογραφίες - Στιγμιότυπο οθόνης (78).png
Εργασία όλων των φωτογραφιών Προσθήκη σε
File "<stdin>", line 3
^
IndentationError: expected an indented block
>>>
>>> for year in df_2.columns[1:]:
...     if(df_2.sort(df[year].desc()).collect()[0][0]=='Greece'):
...         years_greece_was_greatest.append(year)
...         count+=1
>>>
>>>
>>>
>>> print(f"H Ellada eixe perissoteres episkpseis tis xronies: {years_greece_was_greatest}")
H Ellada eixe perissoteres episkpseis tis xronies: ['2009', '2011']
>>> print(f"Sinolika h Ellada eixe perissotera arrivals gia {count} xronies")
Sinolika h Ellada eixe perissotera arrivals gia 2 xronies
>>>
```


Στα παρακάτω 2 στιγμιότυπα φαίνονται το ενδιάμεσο dataframe `df_2` και το τελικό αποτέλεσμα εκτυπώνοντας τη λίστα `years_greece_was_greatest` και το μήκος της, έχοντας όμως τώρα επιλέξει σαν χώρες σύγκρισης το Βέλγιο, τη Γαλλία, τη Κροατία, τη Κύπρο και την Ουγγαρία.

```

>>> from pyspark.sql.functions import *
>>>
>>> selected_countries = ['Greece', 'Belgium', 'Croatia', 'Cyprus', 'France', 'Hungary']
>>> df_2 = df.filter(col('GEO/TIME').isin(selected_countries))
>>> df_2.show()
only showing top 20 rows
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|GEO/TIME| 2006| 2007| 2008| 2009| 2010| 2011| 2012| 2013| 2014| 2015|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|Belgium|16039090|16271311|16360702|15451017|16169676|16723867|16432646|16511721|17068872|18852087|
|Greece|43055381|48081473|47973949|57413304|59184397|65514230|61054383|68992640|74675156|78254524|
|France|105865432|108567043|106993811|98705212|120390105|123227704|125038453|132251136|130908700|130464997|
|Croatia|32858014|33701925|33902735|33357844|33234882|35389002|57079967|59378896|61072661|65683010|
|Cyprus|13310257|13197004|13208954|11666663|12448158|13112596|13488127|13152589|12884399|12550320|
|Hungary|10045891|10170808|10009531|9220148|9358373|9920339|11392183|11982883|12351330|12962395|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

```

>>> years_greece_was_greatest = []
>>> count = 0
>>> for year in df_2.columns[1:]:
...     if (df_2.sort(df_2[year].desc()).collect()[0][0]=='Greece'):
...         years_greece_was_greatest.append(year)
...         count+=1
>>>
>>> print(f"H Ellada eixe perissoteres episkepseis tis xronies: {years_greece_was_greatest}")
H Ellada eixe perissoteres episkepseis tis xronies: []
>>> print(f"Sinolika h Ellada eixe perissotera arrivals gia {count} xronies")
Sinolika h Ellada eixe perissotera arrivals gia 0 xronies
>>>

```

Όπως βλέπουμε πλέον η Ελλάδα δεν έχει περισσότερες διανυκτερεύσεις σε σχέση με όλες τις επιλεγμένες χώρες καμία χρονιά. (η λίστα `years_greece_was_greatest` είναι κενή!)

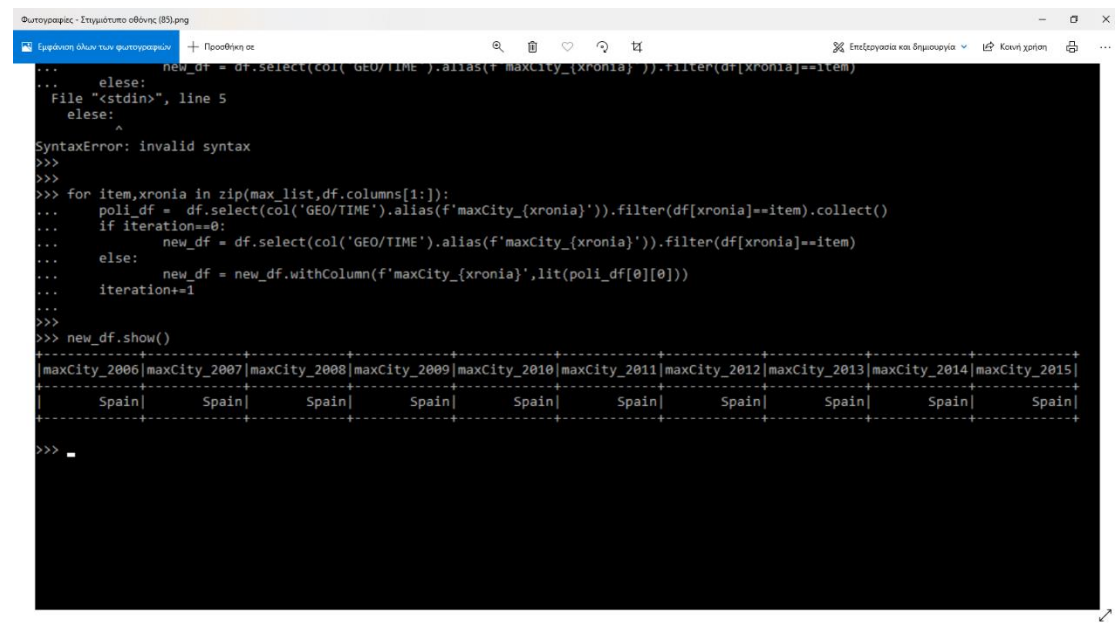
Ερώτημα 3

Για το παρόν ερώτημα θα παράξω ένα dataframe το οποίο για κάθε χρονιά θα περιέχει το όνομα της χώρας που είχε το μέγιστο αριθμό διανυκτερεύσεων εκείνη την χρονιά. Αρχικά ορίζω τη `max_list` η οποία περιέχει το μέγιστο αριθμό διανυκτερεύσεων για κάθε χρονιά. Η `max_list` είναι ένα Row με τη μέγιστη τιμή κάθε στήλης που αντιστοιχεί σε χρονιά. Ορίζω έναν βρόγχο που τρέχει για κάθε ζευγάρι χρονιάς-μέγιστου αριθμού διανυκτερεύσεων εκείνης

της χρονιάς. Αρχικά εξάγω το όνομα της πόλης που είχε το μέγιστο αριθμό διανυκτερεύσεων την εκάστοτε χρονιά στην οποία βρίσκεται ο βρόγχος.

Εν συνεχεία αν είμαι στο πρώτο πέρασμα δημιουργώ το νέο dataframe με μια στήλη που περιέχει το όνομα της πόλης που είχε το μέγιστο εκείνη τη χρονιά. Πιο αναλυτικά χρησιμοποιώ τη συνάρτηση filter για να εντοπίσω ποια είναι η χώρα στη στήλη 'GEO/TIME' για την οποία η τιμή της επιλεγμένης χρονιάς είναι ίση με τη μέγιστη τιμή.

Αν δεν είμαι στο πρώτο πέρασμα του βρόγχου απλά προσθέτω, στο υπάρχον πλέον new_df, τη νέα στήλη με τη πόλη που είχε το μέγιστο αριθμό διανυκτερεύσεων την αντίστοιχη χρονιά, ακριβώς με το ίδιο σκεπτικό με πριν. Στο παρακάτω στιγμιότυπο φαίνεται το τελικό dataframe, το οποίο περιέχει για κάθε χρονιά τη χώρα με το μέγιστο αριθμό διανυκτερεύσεων.



```
...     else:
...         new_df = df.select(col('GEO/TIME').alias(f'maxCity_{xronia}')).filter(df[xronia]==item)
...     File "<stdin>", line 5
...     else:
...         ^
SyntaxError: invalid syntax
>>>
>>>
>>> for item,xronia in zip(max_list,df.columns[1:]):
...     poli_df = df.select(col('GEO/TIME').alias(f'maxCity_{xronia}')).filter(df[xronia]==item).collect()
...     if iteration==0:
...         new_df = df.select(col('GEO/TIME').alias(f'maxCity_{xronia}')).filter(df[xronia]==item)
...     else:
...         new_df = new_df.withColumn(f'maxCity_{xronia}',lit(poli_df[0][0]))
...     iteration+=1
...
>>> new_df.show()
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|maxCity_2006|maxCity_2007|maxCity_2008|maxCity_2009|maxCity_2010|maxCity_2011|maxCity_2012|maxCity_2013|maxCity_2014|maxCity_2015|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|    Spain|    Spain|    Spain|    Spain|    Spain|    Spain|    Spain|    Spain|    Spain|    Spain|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
>>> .
```

Ερώτημα 4

Για το παρόν ερώτημα προσθέτω μια νέα στήλη year_min_was_found στο dataframe, η οποία θα περιέχει για κάθε χώρα τη χρονιά στην οποία είχε τον ελάχιστο αριθμό διανυκτερεύσεων. Για το σκοπό αυτό κάνω όλους του πιθανούς ελέγχους χρησιμοποιώντας τη συνάρτηση when. Όταν η στήλη 2006 έχει μικρότερη τιμή από όλες τις υπόλοιπες στήλες, τότε στη στήλη year_min_was_found αποθηκεύουμε τη τιμή 2006, όταν η 2007 είχε την ελάχιστη τιμή σε σχέση με όλες τις υπόλοιπες στήλες στη στήλη year_min_was_found αποθηκεύουμε τη τιμή 2007 και ούτω καθ' εξής. Στο παρακάτω στιγμιότυπο φαίνεται το τελικό αποτέλεσμα με τη χρονιά που η κάθε χώρα είχε τον ελάχιστο αριθμό διανυκτερεύσεων στη στήλη year_min_was_found.

```
Φωτογραφίες - Στιγμιότυπο οθόνης (89).png
Εργασία όλων των φωτογραφιών + Προβολή σε
... when((col('2012') < col('2006') & (col('2012')<col('2007')) & (col('2012')<col('2008')) & (col('2012')<col('2009')) & (col('2012')<col('2013')) & (col('2012')<col('2014')) & (col('2012')<col('2015'))), '2012'),
1('2012')) & (col('2013') < col('2006') & (col('2013')<col('2007')) & (col('2013')<col('2008')) & (col('2013')<col('2009')) & (col('2013')<col('2014')) & (col('2013')<col('2015'))), '2013'),
1('2012')) & (col('2014') < col('2006') & (col('2014')<col('2007')) & (col('2014')<col('2008')) & (col('2014')<col('2009')) & (col('2014')<col('2015'))), '2014'),
1('2012')) & (col('2015') < col('2006') & (col('2015')<col('2007')) & (col('2015')<col('2008')) & (col('2015')<col('2009')) & (col('2015')<col('2014')) & (col('2015')<col('2015'))), '2015').
... otherwise('more than one min')).show()
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| GEO/TIME | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 | 2013 | 2014 | 2015 | year_min_was_found |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Belgium | 16039890 | 16271311 | 16360702 | 15451017 | 16169676 | 16723867 | 16432646 | 16511721 | 17068872 | 18852087 | 2009 |
| Bulgaria | 11944694 | 12005786 | 11791454 | 9460922 | 10547112 | 12426723 | 13451440 | 14370426 | 14877798 | 13352201 | 2009 |
| Czech Republic | 20090348 | 20610186 | 19087022 | 17746893 | 18365947 | 19424839 | 21703085 | 22144806 | 22110112 | 23286515 | 2009 |
| Denmark | 9453026 | 9327579 | 8918197 | 8299403 | 8981992 | 9491137 | 9688124 | 9914273 | 10608119 | 11171416 | 2009 |
| Germany (until 19... | 52947373 | 54485379 | 56239679 | 54096574 | 59658760 | 63081467 | 68161503 | 71191942 | 74805253 | 78827773 | 2006 |
| Estonia | 3020367 | 2915456 | 2932662 | 2740696 | 3203721 | 3748865 | 3823039 | 3909326 | 3919299 | 3770207 | 2009 |
| Ireland | 21652000 | 0 | 0 | 0 | 0 | 0 | 11839245 | 10871806 | 11276424 | 13462348 | more than one min |
| Greece | 43055381 | 48081473 | 47973949 | 57413304 | 59184397 | 65514230 | 61054383 | 68992640 | 74675156 | 78254524 | 2006 |
| Spain | 224518083 | 225450241 | 223756216 | 200551728 | 213349649 | 239369167 | 243389006 | 252447766 | 259635794 | 269418103 | 2009 |
| France | 105865432 | 108567043 | 106993811 | 98705212 | 120398105 | 123227704 | 125038453 | 132251136 | 130908700 | 130464997 | 2009 |
| Croatia | 32858014 | 33701925 | 33902735 | 33357844 | 33234882 | 35389002 | 57079967 | 59378896 | 61072661 | 65683010 | 2006 |
| Italy | 156861341 | 163465680 | 161797434 | 159493866 | 165202498 | 176474062 | 180594988 | 184793382 | 186792507 | 192607930 | 2006 |
| Cyprus | 13310257 | 13197004 | 13208954 | 11666663 | 12448158 | 13112596 | 13488127 | 13152589 | 12884399 | 12550320 | 2009 |
| Latvia | 1872393 | 1935984 | 2115618 | 1699562 | 1912336 | 2257021 | 2429093 | 2639434 | 2875934 | 2873885 | 2009 |
| Lithuania | 1514197 | 1609998 | 1626829 | 1395899 | 1571325 | 1883003 | 2680048 | 2906201 | 3033826 | 3010727 | 2009 |
| Luxembourg | 2399913 | 2328688 | 2249565 | 2075931 | 1717130 | 2057883 | 2200068 | 2313124 | 2513585 | 2659733 | 2018 |
| Hungary | 10045891 | 10170808 | 10005531 | 9220148 | 9358373 | 9920339 | 11302183 | 11082083 | 12351330 | 12962395 | 2009 |
| Malta | 7094031 | 7749831 | 7581137 | 6550794 | 7208295 | 7361449 | 7498421 | 8172407 | 8420215 | 8542055 | 2009 |
| Netherlands | 26806500 | 27952200 | 25267600 | 25013700 | 26799800 | 27739000 | 27845942 | 31770508 | 34423552 | 37207588 | 2009 |
| Austria | 70017889 | 71518637 | 74730956 | 72224605 | 72585976 | 73647325 | 77158431 | 78433546 | 78099096 | 80804956 | 2006 |
only showing top 20 rows
>>>
```

Μέρος Β

Προ-επεξεργασία

Και εδώ μετέτρεψα το .xlsx αρχείο σε .csv αρχείο με τη χρήση της βιβλιοθήκης pandas στην Python και στη συνέχεια κατασκεύασα το αντίστοιχο dataframe στο pyspark. Επειδή η πρώτη στήλη με τα ονόματα των περιοχών δεν υπάρχει στο σύνολο που δίνεται της έδωσα το όνομα 'Region'. Τα περιεχόμενα αυτού του dataframe για το αρχείο 'international_tourist_arrivals.csv' φαίνονται στο ακόλουθο στιγμιότυπο.

```
Φωτογραφίες - Στιγμιότυπο οθόνης (53).png
Εργασία όλων των φωτογραφιών + Προβολή σε
pyspark.sql.utils.AnalysisException: Path does not exist: file:/C:/Users/Kostis/international_tourist_arrivals.csv
>>> pwd
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'pwd' is not defined
>>>
>>>
>>> df = spark.read.option("header","true").load("Desktop\\ΥΔΑ\\Εαρινό Εξάμηνο\\Αποκεντρωμένα Συστήματα Διαχείρισης Δεδομένων Μεγάλου Όγκου\\ISp
t='csv')
>>> df = df.withColumnRenamed('Unnamed: 0','Region')
>>> df.show()
+-----+-----+-----+-----+-----+-----+-----+-----+
| Region | 1990 | 1995 | 2000 | 2005 | 2010 | 2014 | 2015 |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Europe | null | null | null | null | null | null | null |
| Northern Europe | 28.7 | 36.4 | 44.8 | 59.9 | 62.8 | 70.8 | 75.9 |
| Western Europe | 108.6 | 112.2 | 139.7 | 141.7 | 154.4 | 174.4 | 180.0 |
| Central/Eastern E... | 32.9 | 57.9 | 69.6 | 95.3 | 90.9 | 120.2 | 126.6 |
| Southern/Medit. E... | 90.3 | 98.0 | 132.6 | 156.4 | 173.3 | 214.8 | 225.2 |
| Asia and the Paci... | null | null | null | null | null | null | null |
| North-East Asia | 26.4 | 41.3 | 58.3 | 85.9 | 111.5 | 136.3 | 142.1 |
| South-East Asia | 21.2 | 28.5 | 36.3 | 49.0 | 70.5 | 97.3 | 104.6 |
| Oceania | 5.2 | 8.1 | 9.6 | 10.9 | 11.4 | 13.3 | 14.2 |
| South Asia | 3.2 | 4.2 | 6.1 | 8.2 | 12.1 | 17.5 | 18.3 |
| Americas | null | null | null | null | null | null | null |
| North America | 71.8 | 80.5 | 91.5 | 89.9 | 99.5 | 120.9 | 127.6 |
| Caribbean | 11.4 | 14.0 | 17.1 | 18.8 | 19.5 | 22.3 | 23.9 |
| Central America | 1.9 | 2.6 | 4.3 | 6.3 | 7.9 | 9.6 | 10.3 |
| South America | 7.7 | 11.7 | 15.3 | 18.3 | 23.2 | 29.1 | 30.8 |
| Africa | null | null | null | null | null | null | null |
| North Africa | 8.4 | 7.3 | 10.2 | 13.9 | 19.7 | 20.4 | 18.0 |
| Sub-Saharan Africa | 6.4 | 11.5 | 16.0 | 20.9 | 30.7 | 34.9 | 35.4 |
| Middle East | 9.6 | 12.7 | 22.4 | 33.7 | 54.7 | 52.4 | 53.3 |
>>>
```

Εξετάζοντας το σχήμα (printSchema) του dataframe παρατηρώ πως ο τύπος δεδομένων όλων των στηλών είναι string. Επειδή βλέπω πως το dataframe περιέχει πραγματικές τιμές μετατρέπω τις στήλες που αντιστοιχούν στις χρονιές σε τύπο float.

Στο ακόλουθο στιγμιότυπο φαίνεται το νέο σχήμα του dataframe μετά το casting σε float.


```
Φωτογραφίες - Στιγμιότυπο οθόνης (54).png
Εμφάνιση όλων των φωτογραφιών Προσθήκη σε
North Africa | 8.4 | 7.3 | 10.2 | 13.9 | 19.7 | 20.4 | 18.0 |
Subsaharan Africa | 6.4 | 11.5 | 16.0 | 20.9 | 30.7 | 34.9 | 35.4 |
Middle East | 9.6 | 12.7 | 22.4 | 33.7 | 54.7 | 52.4 | 53.3 |
-----+-----+-----+-----+-----+-----+-----+
>>> for column in df.columns[1:]:
...     df = df.withColumn(column,df[column].cast('float'))
...
>>> df.printSchema()
root
|-- Region: string (nullable = true)
|-- 1990: float (nullable = true)
|-- 1995: float (nullable = true)
|-- 2000: float (nullable = true)
|-- 2005: float (nullable = true)
|-- 2010: float (nullable = true)
|-- 2014: float (nullable = true)
|-- 2015: float (nullable = true)
>>> _
```

Τέλος, όσον αφορά την προ-επεξεργασία, επειδή και αυτό το σύνολο δεδομένων περιέχει ελλιπείς τιμές, αντικαθιστώ και εδώ όλες τις null τιμές με το 0, αφού έχουμε να κάνουμε με αφίξεις τουριστών. Το dataframe μετά την προ-επεξεργασία φαίνεται στο ακόλουθο στιγμιότυπο.

```
Φωτογραφίες - Στιγμιότυπο οθόνης (54).png
Εμφάνιση όλων των φωτογραφιών Προσθήκη σε
pySpark.sql.utils.AnalysisException: Path does not exist: file:/C:/Users/kostis/international_tourist_arrivals.csv
>>> pwd
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'pwd' is not defined
>>>
>>> df = spark.read.option("header","true").load("Desktop\\YBA\\Εαρινό Εξάμηνο\\Αποκεντρωμένα Συστήματα Διαχείρισης Δεδομένων Μεγάλου Όγκου\\Sp
t='csv')
>>> df = df.withColumnRenamed('Unnamed: 0','Region')
>>> df.show()
-----+-----+-----+-----+-----+-----+-----+
Region | 1990 | 1995 | 2000 | 2005 | 2010 | 2014 | 2015 |
-----+-----+-----+-----+-----+-----+-----+
Europe | null | null | null | null | null | null | null |
Northern Europe | 28.7 | 36.4 | 44.8 | 59.9 | 62.8 | 70.8 | 75.9 |
Western Europe | 108.6 | 112.2 | 139.7 | 141.7 | 154.4 | 174.4 | 180.0 |
Central/Eastern E... | 33.9 | 57.9 | 69.6 | 95.3 | 98.9 | 120.2 | 126.6 |
Southern/Medit. E... | 90.3 | 98.0 | 132.6 | 156.4 | 173.3 | 214.8 | 225.2 |
Asia and the Paci... | null | null | null | null | null | null | null |
North-East Asia | 26.4 | 41.3 | 58.3 | 85.9 | 111.5 | 136.3 | 142.1 |
South-East Asia | 21.2 | 28.5 | 36.3 | 49.0 | 70.5 | 97.3 | 104.6 |
Oceania | 5.2 | 8.1 | 9.6 | 10.9 | 11.4 | 13.3 | 14.2 |
South Asia | 3.2 | 4.2 | 6.1 | 8.2 | 12.1 | 17.5 | 18.3 |
Americas | null | null | null | null | null | null | null |
North America | 71.8 | 80.5 | 91.5 | 89.9 | 99.5 | 120.9 | 127.6 |
Caribbean | 11.4 | 14.0 | 17.1 | 18.8 | 19.5 | 22.3 | 23.9 |
Central America | 1.9 | 2.6 | 4.3 | 6.3 | 7.9 | 9.6 | 10.3 |
South America | 7.7 | 11.7 | 15.3 | 18.3 | 23.2 | 29.1 | 30.8 |
Africa | null | null | null | null | null | null | null |
North Africa | 8.4 | 7.3 | 10.2 | 13.9 | 19.7 | 20.4 | 18.0 |
Subsaharan Africa | 6.4 | 11.5 | 16.0 | 20.9 | 30.7 | 34.9 | 35.4 |
Middle East | 9.6 | 12.7 | 22.4 | 33.7 | 54.7 | 52.4 | 53.3 |
>>> _
```

Η συμπλήρωση των ελλιπών τιμών με το 0 θα μου χρησιμεύσει και για την υλοποίηση του ερωτήματος 1, όπως θα δούμε παρακάτω.

Ερώτημα 1

Η λογική μου σε αυτό το ερώτημα είναι η εξής. Αρχικά, για κάθε ομάδα περιοχών που ζητείται, εξάγω ένα Row με τα αθροίσματα για κάθε στήλη. Για παράδειγμα για την Ευρώπη αυτό το κάνω με την εντολή

`df.filter(col('Region').contains('Europe')).groupby().sum().collect()[0].asDict()`. Ελέγγω ποια στοιχεία της στήλης Region περιέχουν στο όνομα το 'Europe' και παίρνω το άθροισμα κάθε

Επειδή έχω χρησιμοποιήσει την `asDict()` λαμβάνω ένα dictionary κάθε φορά το οποίο έχει τη μορφή

Το παραπάνω Dictionary είναι για την Ευρώπη. Τα αντίστοιχα Dictionaries με τα αθροίσματα ανά στήλη για όλες τις ομάδες περιοχών φαίνονται στο παρακάτω στιγμιότυπο.

Φωτογραφία - Στοιχείο οδών (57) ping

Εργαστήριο όλων των φωτογραφιών + Προσθήκη σε

Επεξεργασία και δημοσίευση

Κάνε χρήση

Γράψτε ερώτηση - gyspsek

Caribbean	11.399999618530273	14.0	17.100000181469727	18.799999237060547	19.5	22.99999237060547	23.899999618530273
Central America	1.89999976158142	2.599999804632684	4.300000190734863	6.300000190734863	7.900000095367432	9.400000181469727	10.300000190734863
South America	7.699999809265137	11.699999809265137	15.300000190734863	18.299999237060547	23.200000762939453	29.100000181469727	30.799999237060547
Africa	0.0	0.0	0.0	0.0	0.0	0.0	0.0
North Africa	8.399999618530273	7.300000190734863	10.199999809265137	13.899999618530273	10.700000762939453	20.399999618530273	18.0
Subsaharan Africa	6.400000095367432	11.5	16.0	20.899999618530273	30.700000762939453	34.900001525878906	35.400001525878906
Middle East	0.600000181469727	12.699999809265137	22.399999618530273	33.700000762939453	54.700000762939453	52.400001525878906	53.29999923706055

```
>>>
>>> row_of_america_sum = df.filter((col("Region").contains("America"))) | col("Region").contains("Caribbean")).groupBy().sum().collect()[0].asDict()
>>> row_of_africa_sum = df.filter((col("Region").contains("Africa"))).groupBy().sum().collect()[0].asDict()
>>> row_of_asia_sum = df.filter((col("Region").contains("Asia")) | (col("Region").contains("Oceania"))).groupBy().sum().collect()[0].asDict()
>>> row_of_europe_sum = df.filter((col("Region").contains("Europe"))).groupBy().sum().collect()[0].asDict()
{'sum(1990)': 261.500000181469727, 'sum(1995)': 384.5, 'sum(2000)': 386.70000076293945, 'sum(2005)': 453.2999954223633, 'sum(2010)': 489.3999977118104, 'sum(2014)': 580.1999969482422, 'sum(2015)': 261.500000181469727}
>>> row_of_africa_sum = df.filter((col("Region").contains("Africa"))).groupBy().sum().collect()[0].asDict()
{'sum(1990)': 14.799999713897705, 'sum(1995)': 18.800000190734863, 'sum(2000)': 26.199999809265137, 'sum(2005)': 34.79999923706055, 'sum(2010)': 50.400001525878906, 'sum(2014)': 55.30000114440918, 'sum(2015)': 53.400001525878906}
>>> row_of_asia_sum = df.filter((col("Region").contains("Asia")) | (col("Region").contains("Oceania"))).groupBy().sum().collect()[0].asDict()
{'sum(1990)': 56.000000023841858, 'sum(1995)': 82.09999942779541, 'sum(2000)': 110.29999876022330, 'sum(2005)': 154.00000095367432, 'sum(2010)': 205.5, 'sum(2014)': 264.40000062042505, 'sum(2015)': 279.20000016120934}
>>> row_of_europe_sum = df.filter((col("Region").contains("Europe"))).groupBy().sum().collect()[0].asDict()
{'sum(1990)': 82.80000245571136, 'sum(1995)': 108.799997138977, 'sum(2000)': 128.20000076293945, 'sum(2005)': 133.30000019073486, 'sum(2010)': 150.10000085830688, 'sum(2014)': 181.9000015258789, 'sum(2015)': 192.59999752046078}
>>>
```

Το τελικό αποτέλεσμα με όλες τις τιμές του dataframe κατάλληλα συμπληρωμένες φαίνεται στο παρακάτω στιγμιότυπο.

