Όνομα: Στέφανος

Επώνυμο: Κάππας

Πατρώνυμο: Αλεξάνδρος

Αριθμός μητρώου: mai23020

Μάθημα: Ανάπτυξη Εφαρμογών Και Μεγάλα Δεδομένα

Εργασία: Ατομική

Περιεχόμενα

Μέθοδοι Καρακασίδη

-> Εισαγωγή ...................................................................................................................... σελ. 1

-> process\_corrupted\_nested\_hdfs.py ................................................................................... σελ. 2

-> Αρχικό σχέδιο επεξεργασίας .......................................................................................... σελ. 3

->-> Αρχικό στάδιο επεξεργασίας .......................................................................................... σελ. 3

->-> Επιλογή του κατάλληλου σχήματος ............................................................................ σελ. 4

->-> Αποθήκευση του αποτελέσματος στο νέο json αρχείο ................................................ σελ. 5

-> china\_gdp.py ...................................................................................................................... σελ. 6

-> Τελικό σχέδιο επεξεργασίας .......................................................................................... σελ. 8

Μέθοδοι Κασκάλη

-> Περιοχή html ...................................................................................................................... σελ. 9

-> Περιοχή Javascript ........................................................................................................ σελ. 10

-> Περιοχή CSS ............................................................................................................... σελ. 11

ΜΕΘΟΔΟΙ ΚΑΡΑΚΑΣΙΔΗ

ΕΙΣΑΓΩΓΗ

Η επεξεργασία του αρχείου JSON που έκανα δεν μοιάζει καθόλου με την προηγούμενη. Δηλαδή στην προηγούμενη εργασία η εκφώνηση μας έλεγε να πάμε στο Amadeus να επιλέξουμε διάφορα στοιχεία για τις πτήσεις και να κατεβάσουμε το JSON αρχείο από το Amadeus και με το PySpark να το τροποποιήσουμε σε CSV μορφή. Στην τωρινή εργασία απλά μας δίνονται τα JSON corrupted αρχεία από το drive και απλά τα κατεβάζουμε, τα τροποποιούμε με το PySpark στην σωστή τους JSON μορφή. Έτσι η διαδικασία αυτή θυμίζει την process\_corrupted\_nested\_hdfs.py, επειδή παίρνουμε ένα corrupted JSON αρχείο, το οποίο όλο είναι σε 1 γραμμή, το επεξεργαζόμαστε με το Spark και το αποτέλεσμα που βγαίνει είναι ένα κανονικό singleLine JSON.

Process\_corrupted\_nested\_hdfs.py

from pyspark.sql import SparkSession

if \_\_name\_\_ == "\_\_main\_\_":

  spark = SparkSession.builder.appName("Read corrupted json").getOrCreate()

  dfFromTxt=spark.read.text("./drive/MyDrive/corrupted\_nested.json")

  dfFromTxt.show(truncate=False)

  from pyspark.sql.functions import regexp\_replace

  dfFromTxt = dfFromTxt.withColumn('value', regexp\_replace('value', 'False', '"FALSE"'))

  dfFromTxt.count()

  dfFromTxt = dfFromTxt.withColumn('value', regexp\_replace('value', '\},\ \{', '\}\_\_SPLIT\_POINT\_\_\{'))

  dfFromTxt = dfFromTxt.withColumn('value', regexp\_replace('value','\},\ \{"Brand"', '\},\}\_\_SPLIT\_POINT\_\_\{"Brand"'))

  dfFromTxt.show(truncate=False)

  from pyspark.sql.types import StructType,StructField,IntegerType,StringType,DoubleType,BooleanType

  schema = StructType([

    StructField("Brand",StringType(),True),

    StructField("Days", IntegerType(), True),

    StructField("Model", StringType(), True),

    StructField("Rate", DoubleType(), True),

    StructField("Available", StringType(), True),

    StructField("Damages", StringType(), True)

  ])

  from pyspark.sql.functions import col,from\_json

  dfJSON = dfFromTxt.withColumn("jsonData",from\_json(col("value"),schema)).select("jsonData.\*").dropna()

  dfJSON.show()

  from pyspark.sql.functions import split, explode

  dfFromTxt = dfFromTxt.withColumn('value',explode(split('value','\_\_SPLIT\_POINT\_\_')))

  dfFromTxt.count()

  dfFromTxt.show(truncate=False)

  dfJSON = dfFromTxt.withColumn("jsonData",from\_json(col("value"),schema)).select("jsonData.\*").dropna()

  dfJSON.show()

  dfFromTxt = dfFromTxt.withColumn('value', regexp\_replace('value', '\[\{"Brand"', '\{"Brand"')).withColumn('value', regexp\_replace('value', '\}\]\}\]', '\}\]\}'))

  dfFromTxt.show(truncate=False)

  dfJSON = dfFromTxt.withColumn("jsonData",from\_json(col("value"),schema)).select("jsonData.\*").dropna()

  dfJSON.show(truncate=False)

  from pyspark.sql.functions import udf

  import json

  def parse\_json(array\_str):

    json\_obj = json.loads(array\_str)

    return len(json\_obj)

  parseUDF = udf(lambda x:parse\_json(x),IntegerType())

  dfJSON.withColumn("Num\_damages",parseUDF(col("Damages"))).show()

dfJSON.write.format('csv').mode('overwrite').save('./corrupted-json-example')

ΑΡΧΙΚΟ ΣΧΕΔΙΟ ΕΠΕΞΕΡΓΑΣΙΑΣ

Εδώ θέλω να επεξεργαστώ το corrupted JSON αρχείο για να το εξάγω σε ένα νέο επίσης JSON που μπορώ να το διαβάζων. Τα περιεχόμενα της επεξεργασίας είναι:

Αρχικό στάδιο επεξεργασίας

Περιοχή αρχικοποίησης

Άνοιγμα του αρχείου

Εισαγωγή του split point

Απαλοιφή της επικεφαλίδας του JSON

Απαλοιφή εμφωλευμένων περιοχών

Πρόβλημα που αντιμετώπισα με τα εμφωλευμένα πεδία

Τελικό στάδιο αρχικής επεξεργασίας

Επιλογή Του Κατάλληλου Σχήματος

Εισαγωγή στο σχήμα

Εμφάνιση με βάση του σχήμα

Προβλήματα που αντιμετώπισα με το σχήμα

Αποθήκευση Του Αποτελέσματος Στο Νέο Json Αρχείο

Αποθήκευση του αποτελέσματος σε νέο αρχείο

Πρόβλημα αποθήκευσης του αποτελέσματος

Αρχικό Στάδιο Επεξεργασίας

ΠΕΡΙΟΧΗ ΑΡΧΙΚΟΠΟΙΗΣΗΣ

Το σχέδιο για την επεξεργασία του JSON αρχείου στην σωστή του μορφή είναι με βάση το process\_corrupted\_nested\_hdfs.py = κώδικας δείγμα.

ΑΝΟΙΓΜΑ ΤΟΥ ΑΡΧΕΙΟΥ

Καταρχάς ως πλαίσιο εισόδου παίρνουμε το αρχείο (έστω για δοκιμή) το CN.NY.GDP.MKTP.CD.json με την ίδια λογική όπως στο κώδικα δείγμα.

ΕΙΣΑΓΩΓΗ ΤΟΥ SPLIT POIN

Στην συνέχεια, με την διαδικασία dfFromTxt.withColumn και τις παραμέτρους regexp\_replace που κατα γράμμα ήταν και στο κώδικα δείγμα. Αυτή αρχικά έκανε ότι ο κώδικας δείγμα, δηλαδή να διαχωρίζει τις γραμμές στο corrupted JSON αρχείο. Πώς γίνεται; Το πρόγραμμα διαβάζει χαρακτήρα-χαρακτήρα το JSON αρχείο ως txt μορφή και όταν συναντήσει την ακολουθία των χαρακτήρων },{ τότε αντικαθιστά το κόμμα με το χαρακτήρα διαχωρισμού των γραμμών, ή \_\_SPLIT\_POINT\_\_. Οι υπόλοιπες λειτουργίες του είναι διαφορετικές από τον κώδικα δείγμα αλλά είναι με βάση την ίδια λογική.

ΑΠΑΛΟΙΦΗ ΤΗΣ ΕΠΙΚΕΦΑΛΙΔΑΣ ΤΟΥ JSON

Δηλαδή στην συνέχεια θα αφαιρέσουμε το πρώτο πεδίο ή αλλιώς πεδίο επικεφαλίδας του JSON αρχείου επειδή, πρώτα είναι άχρηστο στην μετέπειτα επεξεργασία του αρχείου και επίσης στον χειρισμό με το HTML και δεύτερον μας προκαλεί προβλήματα με το σχήμα που θα δημιουργηθεί στην συνέχεια και κατόπιν στην αποθήκευση του αρχείου.

ΑΠΑΛΟΙΦΗ ΕΜΦΩΛΕΥΜΕΝΩΝ ΠΕΡΙΟΧΩΝ

Αργότερα γίνεται το σπάσιμο των εμφωλευμένων πεδίων στο JSON αρχείο, δηλαδή το πεδίο indicator, και το country έχουν εμφωλευμένα πεδία, το id και το value. Η διαδικασία γίνεται με το σπάσιμο των παρενθέσεων και την μετονομασία των εμφωλευμένων πεδίων σε indicatorId, indicatorValue, coutryId, countryValue.

ΠΡΟΒΛΗΜΑ ΠΟΥ ΑΝΤΙΜΕΤΩΠΙΣΑ ΜΕ ΤΑ ΕΜΦΩΛΕΥΜΕΝΑ ΠΕΔΙΑ

Εδώ αντιμετώπισα το πρόβλημα με αυτή την διαδικασία, δήλαδή στο JSON αρχείο υπάρχουν 2 id και 3 value. Έστω ότι τα id είναι αμέσως μετά από τα indicator, county και απλά απαλοίφεται το άγκιστρο και η άνω κάτω τελεία και συγχωνεύονται τα ονόματα στο εξωτερικόΕσωτερικό πεδίο (indicatorId) με αποτέλεσμα να μην υπάρχει πρόβλημα. Τα value όμως είναι πρώτων στο δεύτερο υποπεδίο των εξωτερικών πεδίων και δεύτερον υπάρχει ένα value εξωτερικό πεδίο που περιέχει π.χ. το gdp της Κίνας. Εάν το κάνω με την απλή αντικατάσταση τότε θα αλλάξουν όλα τα value στην νέα ονομασία. Η λύση είναι να διαβάσουμε και την τιμή των πεδίων value που θέλουμε να αλλάξουμε για να μην αλλάξουν όλα τα value. Ευτυχώς που στα indicator και country τα πεδία id, value έχουν την ίδια τιμή σε όλες τις γραμμές. Άρα παίρνουμε και την τιμή του πεδίου και μετονομάζουμε το όνομα του πεδίου. Αν δεν ήταν έτσι τότε θα έχουμε θέμα.

ΤΕΛΙΚΟ ΣΤΑΔΙΟ ΑΡΧΙΚΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ

Μετά αφαιρούμε το άγκιστρο που κλείνει το εμφωλευμένο πεδίο γιατί δεν το χρειαζόμαστε πλέον. Τέλος μετονομάζουμε το εξωτερικό value σε gdp επειδή περιέχει το ετήσιο gdp την κίνας.

Επιλογή Του Κατάλληλου Σχήματος

ΕΙΣΑΓΩΓΗ ΣΤΟ ΣΧΗΜΑ

Όπως και στο κώδικα δείγμα έτσι και εδώ δημιουργώ ένα σχήμα schema, το οποίο θα περιέχει τα πεδία του συγκεκριμένου JSON αρχείου μετά την τροποποίηση του. Τα πεδία έχουν ως εξής: indicatorId = String, indicatorValue = String, countryId = String, countryValue = String, countyisoiode3= String, date = Date, gdp = Double, unit = String, obs\_status = String, decimal = Integer.

ΕΜΦΑΝΙΣΗ ΜΕ ΒΑΣΗ ΤΟ ΣΧΗΜΑ

Στην συνέχεια ακριβώς όπως και στον κώδικα δείγμα διαβάζεται το JSON αρχείο με βάση το σχήμα που δώσαμε στο πρόγραμμα να διαβάσει. Μετά εμφανίζει στην οθόνη η σχηματοποιημένη του μορφή κενή επειδή δεν δώσαμε τα SPLIT POINTS στο πρόγραμμα να τα διαβάζει. Έτσι δίνουμε και τα SPLIT POINT να τα καταχωρήσει στο σχήμα με αποτέλεσμα όταν τα διαβάσει να εμφανίσει τις τιμές των πεδίων στις στήλες του παραχθέντος σχήματος για να είναι σε σωστή και αποθηκεύσιμη μορφή.

ΠΡΟΒΛΗΜΑΤΑ ΠΟΥ ΑΝΤΙΜΕΤΩΠΙΣΑ ΜΕ ΤΟ ΣΧΗΜΑ

Εδώ αντιμετώπισα το πρόβλημα με την δημιουργία εμφωλευμένου σχήματος. Για παράδειγμα θέλω να εμφανίζω το id του indicator και όχι όλο το indicator. Έτσι αρχικά στο schema έκανα την εντολή:

StructField(“indicator”, StructType([ StructField(“id”, StringType(), True), …])…

Εάν το κάνω έτσι τότε απλά όχι μόνο δεν θα διαβάζει την περιοχή id του indicator αλλά δεν θα διαβάσει καθόλου του αρχείο. Μία νέα εναλλακτική που έκανα ήταν πάλι στο ίδιο το schema να κάνω την εξής εντολή: StructField(“indicator.id”, StringType(), True), … και πάλι τα ίδια τα προβλήματα. Μία τελευταία εναλλακτική που έκανα πριν φτάσω στην ΑΠΑΛΟΙΦΗ ΕΜΦΩΛΕΥΜΕΝΩΝ ΠΕΡΙΟΧΩΝ είναι η χρήση 2 σχημάτων. Το ένα σχήμα περιέχει το id, value και το άλλο περιέχει τα εξωτερικά πεδία. Έτσι ο κώδικας είναι: Schema1 = StructField(id), StructField(value) και το Schema2 = StructField(indicator, Schema2) … Πάλι συντακτικά σφάλματα και σφάλματα εκτάλεσης έβγαλε με αποτέλεσμα να κάνω στην αρχικοποίηση της απαλοιφή των εμφωλευμένων πεδίων.

Αποθήκευση Του Αποτελέσματος Στο Νέο Json Αρχείο

ΑΠΟΘΗΚΕΥΣΗ ΤΟΥ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΣΕ ΝΕΟ ΑΡΧΕΙΟ

Το πρόγραμμα που δημιούργησα κάποια στιγμή θα αποθηκεύσει την επεξεργασία του JSON αρχείου σε ένα νέο JSON αρχείο. Όπως και στον κώδικα δείγμα, η διαδικασία είναι η ίδια. Η διαφορά είναι ότι στον κώδικα μου αφαίρεσα τις εντολές που αφορούν το parseUdf επειδή δεν θέλω να δημιουργήσω μία νέα στήλη που εμφανίζει το πλήθος των πεδίων που είναι εμφωλευμένα στο εξωτερικό πεδίο.

ΠΡΟΒΛΗΜΑ ΑΠΟΘΗΚΕΥΣΗΣ ΤΟΥ ΑΠΟΤΕΛΕΣΜΑΤΟΣ

Το πρόβλημα που αντιμετώπισα στην αρχή ήταν όταν χρησιμοποίησα το απλό σχήμα χωρίς την απαλοιφή των εμφωλευμένων πεδίων ήταν τα εμφωλευμένα πεδία να εμφανιστούν με την μορφή:

{"indicator":"{\"id\":\"NY.GDP.MKTP.CD\", … Παρατηρώ ότι εμφανίζονται τα \” που δυσκολεύουν την εμφάνιση των πεδίων στο αρχείο. Δεν ξέρω πως να τα απαλοίφσω αλλιώς ίσως με τις εντολές: schema = StructType().add("id", StrintType()) και

display(events.select(from\_json("indicator", schema).alias("c"))) αλλά πάλι θα μου βγάλει λάθος επειδή δεν έχω εξοικιωθεί με αυτές τις εντολές της ιστοσελίδας https://www.databricks.com/blog/2017/02/23/working-complex-data-formats-structured-streaming-apache-spark-2-1.html. Έτσι απλά έκανα την ΑΠΑΛΟΙΦΗ ΕΜΦΩΛΕΥΜΕΝΩΝ ΠΕΡΙΟΧΩΝ και μετά το αρχείο είναι αποθηκευμένο στην singleLine json μορφή. Επίσης δεν μπορώ να εμφανίστο στο HTML την singleLine μορφή του. Πρέπει να βάλω κόμματα σε κάθε κλείσιμο των άγκιστρων και 2 αγκύλες. Η μία στην αρχή και η άλλη στο τέλος. Ίσως με απλή Python να το καταφέρω εδώ.

china\_gdp.py

from pyspark.sql import SparkSession

if \_\_name\_\_ == "\_\_main\_\_":

  spark = SparkSession.builder.appName("Read corrupted json").getOrCreate()

  dfFromTxt=spark.read.text("./drive/MyDrive/CN.NY.GDP.MKTP.CD.json")

  dfFromTxt.show(truncate=False)

  from pyspark.sql.functions import regexp\_replace

  dfFromTxt = dfFromTxt.withColumn('value', regexp\_replace('value', '\},\{', '\}\_\_SPLIT\_POINT\_\_\{'))

  dfFromTxt.show(truncate=False)

  dfFromTxt = dfFromTxt.withColumn('value', regexp\_replace('value', '\[\{"page":1,"pages":1,"per\_page":5000,"total":63,"sourceid":"2","lastupdated":"2023-07-25"\}, \[','\['))

  dfFromTxt.show(truncate=False)

  dfFromTxt = dfFromTxt.withColumn('value', regexp\_replace('value', '"indicator":\{"id"','"indicatorId"'))

  dfFromTxt.show(truncate=False)

  dfFromTxt = dfFromTxt.withColumn('value', regexp\_replace('value', '"value":"GDP \(current US\$\)"','"indicatorValue":"GDP \(current US\$\)"'))

  dfFromTxt.show(truncate=False)

  dfFromTxt = dfFromTxt.withColumn('value', regexp\_replace('value', '\},"country":\{"id"',',"countryId"'))

  dfFromTxt.show(truncate=False)

  dfFromTxt = dfFromTxt.withColumn('value', regexp\_replace('value', '"value":"China"','"countryValue":"China"'))

  dfFromTxt.show(truncate=False)

  dfFromTxt = dfFromTxt.withColumn('value', regexp\_replace('value', '\},', ','))

  dfFromTxt.show(truncate=False)

  dfFromTxt = dfFromTxt.withColumn('value', regexp\_replace('value', 'value', 'gdp'))

  dfFromTxt.show(truncate=False)

  from pyspark.sql.types import StructType,StructField,IntegerType,StringType,DoubleType,BooleanType, DateType

  schema = StructType([

    StructField("indicatorId", StringType(), True),

    StructField("indicatorValue", StringType(), True),

    StructField("countryId", StringType(), True),

    StructField("countryValue", StringType(), True),

    StructField("countryiso3code", StringType(), True),

    StructField("date", DateType(), True),

    StructField("gdp", DoubleType(), True),

    StructField("unit", StringType(), True),

    StructField("obs\_status", StringType(), True),

    StructField("decimal", IntegerType(), True)

  ])

  from pyspark.sql.functions import col,from\_json

  dfJSON = dfFromTxt.withColumn("jsonData",from\_json(col("value"),schema)).select("jsonData.\*").dropna()

  dfJSON.show()

  from pyspark.sql.functions import split, explode

  dfFromTxt = dfFromTxt.withColumn('value',explode(split('value','\_\_SPLIT\_POINT\_\_')))

  dfFromTxt.count()

  dfFromTxt.show(truncate=False)

  dfJSON = dfFromTxt.withColumn("jsonData",from\_json(col("value"),schema)).select("jsonData.\*").dropna()

  dfJSON.show(truncate=False)

  dfJSON.write.format('json').mode('overwrite').save('./corrupted-json-example')

ΤΕΛΙΚΟ ΣΧΕΔΙΟ ΕΠΕΞΕΡΓΑΣΙΑΣ

Αρχικά ο κώδικας στην αρχικοποίηση λειτουργεί σχεδόν όπως και ο china\_gdp.py. Η διαφορά του είναι ότι δεν διαβάζει 1 αρχείο αλλά 2 και συγκεκριμένα στο dfFromTxt1 διαβάζει το αρχείο με επίθημα .NY.GDP.MKTP.CD.json και στο dfFromTxt2 διαβάζει το αρχείο με επίθημα .SP.POP.json. Στο παράδειγμα μας αναφέρω το CN.NY.GDP.MKTP.CD.json και το CN.SP.POP.json και εμφανίζει όπως είναι τα αρχεία στην οθόνη.

Στην αρχικό στάδιο επεξερχασίας επεξεργάζεται το dfFromTxt1 ακριβώς όπως και ο china\_gdp.py στα στάδια Εισαγωγή του split point, Απαλοιφή της επικεφαλίδας του JSON, Απαλοιφή εμφωλευμένων περιοχών, Πρόβλημα που αντιμετώπισα με τα εμφωλευμένα πεδία και τέλος το Τελικό στάδιο αρχικής επεξεργασίας. Η διαφορά του με το china\_gdp.py είναι ότι στο dfFromTxt2 στο Τελικό στάδιο αρχικής επεξεργασίας αντί για value να το κάνει gdp το κάνει population. Επίσης αλλάζει το όνομα ενός νέου πεδίου, το decimal που το κάνει gdpPerCapita. Τέλος εμφανίζει στην οθόνη τα 2 αρχεία μετά από την αρχική τους επεξεργασία.

Στην επιλογή του κατάλληλου σχήματος, χρησιμοποιεί 2 σχήματα schema1, schema2. Στο schema1 δεν κρατάμε όλα τα πεδία αλλά αυτά που ζητάει η άσκηση. Δηλαδή κρατάμε το countryValue, date, gdp. Στο schema2 κρατάμε το countryValue, date, population, gdpPerCapita. Φυσικά εδώ δεν χρειάζεται να βάλουμε όλα τα κοινά πεδία με το schema1. Αρκεί μόνο ένα κοινό (έστω το date) και γίνεται η δουλειά.

Στην εμφάνιση με βάση το σχήμα αφαιρώ τις εντολές για την εμφάνιση της κενής σχηματοποιημένη του μορφή στην οθόνη και επικεντρώνομαι στην καταχώρηση του \_\_SPLIT\_POINT\_\_ και στα 2 αρχεία, δηλαδή το dfFromTxt1 και dfFromTxt2. Στην συνέχεια την εξαγωγή την κάνω σε 2 dfJSON. To dfJSON1 από το dfFromTxt1 με το schema1 και το dfJSON με το schema2. Αφού το κάνω αυτό τότε εμφανίζω στην οθόνη τα 2 αρχεία με το σχήμα τους για επίδηξη. Έπειτα δημιουργώ μία νέα μεταβλητή, την dfJSON που σε αυτή θα γίνει η συνέννωση του dfJSON1 με το dfJSON2 με κοινό τους κλειδί το date. Το αποτέλεσμα αυτό, το εμφανίζω στην οθόνη. Πριν τελειώσω με αυτό το στάδιο, στο dfJSON τροποποιώ το πεδίο gdpPerCapita με την αποθήκευτση του αποτελέσματος της διαίρεση του πεδίου του gdp με το population για να βγάλω και το κατά κεφαλής ΑΕΠ. Στο τέλος εμφανίζω στην οθόνη το τελικό αποτέλεσμα.

Τέλος αποθηκεύω σε JSON μορφή το dfJSON ακριβώς όπως και στο china\_gdp.py. Το όνομα του κώδικα αυτού το ονόμασα pySparkChina.py.

Υπενθυμίζω ότι στους αλλους 2 κώδικες δηλαδή στο pySparkIndia.py και στο pySparkUS.py κάνω την ίδια δουλειά με την διαφορά ότι στην αρχικοποίηση επιλέγω τα κατάλληλα αρχεία και στο αρχικό στάδιο επεξεργασίας όπου China δώ το τροποποιώ στην κατάλληλη χώρα. Επίσης η προσθήκη των κομμάτων και των 2 αγκυλών στην αρχή και στο τέλος του json αρχείου γίνεται από τον κώδικα readJSON.py. Επιλέγεις το json αρχείο που παράχθηκε από το spark και γράφεις το αποτέλεσμα στο αρχείο προορισμού με όνομα που θέλεις.

ΤΕΛΟΣ ΜΕΘΟΔΟΙ ΚΑΡΑΚΑΣΙΔΗ

ΜΕΘΟΔΟΙ ΚΑΣΚΑΛΗ

ΠΕΡΙΟΧΗ HTML

Το περιεχόμενο που έφτιαξα περιλαμβάνει τα εξής πεδία: το Input type = file με id = fileInput και class = btn1 που είναι το κουμπί εισαγωγής ενός αρχείου .json για την εμφάνιση του στην οθόνη. Το table με class = array που περιέχει όλο το περιεχόμενο του json αρχείου που επιλέξαμε. Αυτό χωρίζεται σε επιμέρους τμήματα, το thead με class το hhead και το tbody με class το bbody.

Το hhead περιέχει 5 πεδία, που όλα έχουν class = c και id από sort 1 μέχρι sort 5. Εδώ θα εμφανίζονται τα ονόματα των στηλών όπως αυτά δίνονται στην εκφώνηση. Δηλαδή περιλαμβάνουν την χώρα, το έτος, τον πλυθυσμό, το ακαθάριστο εθνικό προιόν και το ακαθάριστο εθνικό προϊόν κατά κεφαλήν. Το bbody περιέχει επίσης 5 πεδία, έχει σε όλα του τα πεδία class = c και id από col1 μέχρι col5. Εδώ εμφανίζονται τα περιεχόμενα των στηλών από το json αρχείο.

Στο αρχείο υπάρχουν επίσης 2 κουμπιά με εικόνα ενός βέλους για να δείξει σε ποία σελίδα θα πάει το html εάν πατηθεί από τον χρήστη. Ποιο συγκεκριμένα δεν είναι κουμπιά αλλά είναι img με class btnl και btnr (l = left, r = right) και id = previous και next αντίστοιχα.

Στην συνέχεια υπάρχει ένα img σαν κουμπί το οποίο στην ουσία συμπεριφέρεται σαν menu, το οποίο εάν πατηθεί θα εμφανίσει στην οθόνη την λίστα ul με 5 κουμπιά button. Αυτό έχει class = hidden. Τα 5 κουμπιά έχουν class = u1 και id από row1 εώς row5. Αυτά έχουν την λειτουργία για εμφάνιση και απόκρυψη επιμέρους στηλών του πίνακα.

Τέλος υπάρχει και ένα label με id = page και class = lalabel, το οποίο εμφανίζει την τρέχουσα σελίδα που εμφανίζεται το περιεχόμενο του json, επειδή αυτό σελιδοποιήται όταν ο αριθμός των στηλών υπερβεί τις 20.

Υπενθυμίζω ότι όλο το περιεχόμενο βρίσκεται μέσα σε έναν μεγάλο πίνακα με 3 γραμμές και 3 στήλες. Ο εξωτερικός πίνακας έχει class = a, όλες οι γραμμές tr έχουν class = b, όλες οι στήλες td εξαιρουμένου της μεσαίας έχουν επίσης class = b και στην μεσαία γραμμή το μεσαίο td έχει class = b2 και τα υπόλοιπα td έχουν class = b1. Περισσότερες λεπτομέριες θα αναφέρω στην Περιοχή CSS.

ΠΕΡΙΟΧΗ JAVASCRIPT

Αυτή η περιοχή είναι βασισμένη στην εργασία του χειμερινού εξαμήνου που σας έδωσα, αλλά τροποποιήθηκε και προσαρμόστηκε για τις ανάγκες της σημερινής εργασίας.

Αρχικά όλο το περιεχόμενο είναι εμφωλευμένο στο use-script. Το script ξεκινάει με τις δηλώσεις όλων των καθολικών μεταβλητών (άσχετα εάν ο καθηγητής μας είπε να αποφύγουμε τις καθολικές μεταβλητές, εγώ ό,τι θέλω θα κάνω) που αφορούν τις 5 στήλες του πίνακα (στην πραγματικότητα ο πίνακας που εμφανίζεται στην οθόνη δεν είναι διδιάστατος, δηλαδή 63 επί 5 αλλά 5 μονοδιάστατοι πίνακες με μέγεθος = 63). Επίσης τοποθετήται το a\_row = συνολικός αριθμός γραμμών, start, finish, index για φορά ταξινόμησης, sizee για σελιδοποίηση στις 20 γραμμές και 5 μεταβλητές σημαίες flags με ονόματα από colunn1 εώς 5 για την εμφάνιση και απόκρυψη επιμέρους στηλών.

Η μέθοδος με το fileInput (όπως λέει και το όνομα της) φορτώνει το αρχείο στην οθόνη. Ποιο συγκεκριμένα η συνάρτηση αυτή εάν εκτελεστεί θα μηδενίζει τους 5 πίνακες για να μην τα περιεχόμενα από 2 χώρες στην οθόνη και επίσης να μην σφάλει η εφαρμογή. Στην συνέχεια προσαρτεί πεδίο-πεδίο του json σε κάθε πίνακα (ανάλογα με την στήλη) και επίσης στις στήλες gdp και gdpPerCapita αφαιρεί τα περίσσια δεκαδικά ψηφία για ωραία και συμπυκνωμένη εμφάνιση. Τέλος εμφανίζει στην οθόνη το περιεχόμενο του με την μέθοδο appear.

Η συνάρτηση sort ταξινομεί με τον σωστό τρόπο κάθε γραμμή των πινάκων με βάση τον επιλεγμένο πίνακα. Δηλαδή έστω οι πίνακες έτος, πληθυσμός και ο χρήστης επιλέξει ταξινόμηση κατά πληθυσμό τότε και το έτος θα ταξινομηθεί με βάση τον πληθυσμό, σαν δυδιαστατο πίνακα.

Η μέθοδος appear αρχικά σβήνει το tbody τελείως και μετά ελέγχει τις τιμές των μεταβλητών flag. Εάν όλες οι flag έχουν τιμή 0 τότε δεν επιρρεάζει κανένα πεδίο του thead και απλά προσαρτεί όλους του πίνακες σε όλα τα πεδία του tbody. Αν κάποιο flag έχει τιμή 1 τότε σβήνει το συγκεκριμένο πεδίο του thead και επίσης δεν προσαρτά τιμές στο συγκεκριμένο tbody. Βασικά εάν όλα τα flags έχουν τιμή 1 τότε ο πίνακας είναι κενός. Επίσης εμφανίζει και το κατάλληλο page για την τρέχουσα σελίδα.

Η μέθοδος previous ελέγχει το start και το finish της τρέχουσας σελίδας και εάν το start είναι > 0, δηλαδή 20, 40, ... τότε κάνει τον υπολογισμό start = start - sizee = start - 20 (και μειώνει το page κατά 1). Αλλιώς δεν κάνει τίποτα. Στο τέλος χρησιμοποιεί το appear για να εμφανίζει στην οθόνη την προηγούμενη σελίδα.

Η μέθοδος next ελέγχει το start και το finish της τρέχουσας σελίδας. Εάν το finish + sizee είναι μικρότερο από το a\_row τότε προχωράει στην επόμενη σελίδα (και αυξάνει το page κατά 1). Εάν το finish + sizee είναι > από το a\_row ή το πλήρες μέγεθος των πινάκων, δηλαδή 63 σε αυτή την περίπτωση τότε κάνει τον υπολογισμό finish = a\_row και start = start (π.χ. start = 60, finish = 63). Τέλος εάν το finish = a\_row τότε δεν αλλάζει σελίδα. Στο τέλος χρησιμοποιεί το appear για να εμφανίζει στην οθόνη την προηγούμενη σελίδα.

Οι μέθοδοι sort 1 εώς sort 5 στην αρχή εξέγχουν εάν το index έχει τιμή 0 ή 1. Εάν το index = 0 τότε κάνουν ταξινόμηση κατά αύξουσα σειρά. Αλλιώς κάνουν ταξινόμηση κατά φθίνουσα σειρά. Χρησιμοποιούν το sort για την ταξινόμηση και στο τέλος αλλάζουν το index από 0 σε 1 και το αντίστροφο και εμφανίζουν στην οθόνη τον ταξινομημένο πίνακα με την appear.

Οι μέθοδοι row1 εώς row5 αλλάζουν το flag τους (για παράδειγμα η row2 αλλάζει το column2) από 0 σε 1 και το αντίστροφο. Επίσης αλλάζουν το περιεχόμενο στα κουμπιά τους. Εάν το flag (σε ένα συγκεκριμένο κουμπί) από 0 γίνει 1 τότε το περιεχόμενο του κουμπιού γίνεται Εμφάνιση της συγκεκριμένης στήλης. Αλλιώς γίνεται Απόκρυψη της συγκεκριμένης στήλης (που είναι by default). Τέλος καλάνε την appear για την εμφάνιση του αποτελέσματος στην οθόνη.

Υπενθυμίζω όταν λέω μέθοδο εννοώ την συνάρτηση-λειτουργία που κάνει το πρόγραμμα όταν πατηθεί το στοιχείο με id το όνομα της μεθόδου. Επίσης παντού χρησιμοποιώ το getElementById και όχι άλλο επειδή έτσι μου αρέσει και είναι πιο βολικό.

ΠΕΡΙΟΧΗ CSS

Η εμφάνιση της ιστοσελίδας είναι σε 2 εκδοχές. Την έκδοση desktop για ανάλυση μεγαλύτερη από 500px και την έκδοση mobile για ανάλυση μικρότερη από 500px. Επίσης Αναφέρω ότι ο πίνακας με κλάση .a έχει 3 γραμμές και 3 στήλες. Η ονομασία των κελιών γίνεται με το όνομα γραμμής, όνομα στήλης. Έστω 1η γραμμή και δεύτερη στήλη = 1,2. Τα εύρη δηλαδή είναι από 1,1 εώς 3,3.

Στην έκδοση για desktop:

Το 1,1 περιέχει το fileInput με class btn1. Εδώ απλά εμφανίζεται το κουμπί μεγάλο με μεγάλους χαρακτήρες τύπου κούριες για καλύτερη εμφάνιση.

Το 2,1 περιέχει το αριστερό βέλος με class btnl.

Το 2,3 περιέχει το δεξί βέλος με class bntr.

Το 3,2 περιέχει την εικόνα μενού με class hidden και εάν πατηθεί τότε εξαφανίζεται (το πλάτος γίνεται 0) και εμφανίζεται το ul με τα 5 κουμπιά επιλογής εμφάνισης/απόκρυψης των στηλών συρόμενο από αριστερά στο κέντρο του κελιού 3,2 σε μορφή inline και εάν πατηθεί σε κάποιο μέρος της οθόνης κλικ, αυτά επιστρέφουν αριστερά (εκτός πεδίου), δηλαδή εξαφανίζονται και ξαναεμφανίζεται πάλι το htdden. Βεβαίως τα κουμπιά έχουν ροζ χρώμα και πιο μεγάλα είναι για ωραία εμφάνιση.

Το 3,3 περιέχει το label με class lalabel και id page για την εμφάνιση της τρέχουσας σελίδας με μεγάλα και έντονα μπλέ γράμματα.

Το .a έχει border 1 pixel σκούρου χρώματος για το καθορισμό των ορίων της ιστοσελίδας. Το .b και .b1 έχει border 1 pixel πράσινου χρώματος και φόντο κίτρινου χρώματος. Το .b2 έχει border 1 pixel πράσινου χρώματος και φόντο πράσινου χρώματος. Το .hhead έχει φόντο πράσινου χρώματος και η κλάση .c λειτουργεί με το hover, δηλαδή εάν το ποντίκι ακουμπήσει ένα πεδίο του .hhead τότε αλλάζει το πεδίο από κίτρινο σε άσπρο για ομορφιά. Τέλος το .bbody έχει φόντο τυρκουάζ χρώματος.

Στην έκδοση για mobile:

Ο πίνακας .a εμφανίζεται πλέον με την μορφή όλα (σχεδόν) τα κελιά εμφανίζονται με την μορφή block. Επίσης αλλάζει το φόντο του σε κίτρινο και border κόκκινο. Επίσης το .b πάλι εμφανίζει τα πεδία του με block μορφή και καταργεί τα border του για καθαρότερη εμφάνιση. Το .b1 εμφανίζεται ως table-footer επειδή θέλουμε το btnl και btnr να είναι κάτω και όχι το ένα κάτω και το άλλο πάνω. Επίσης δέν έχει και αυτό border. Το .b2 εμφανίζεται ως table-header επειδή ο πίνακας .array που εμφανίζει το περιεχόμενο από τα json αρχεία πρέπει να είναι πάντα υψηλότερος από τα βελάκια.

Τα κουμπιά αλλαγής σελίδας ή αλλιώς .btnl, .btnr επειδή ναι μεν εμφανίζονται κάτω από το .array αλλά πάλι το ένα πάνω από το άλλο, τροποποίησα το position σε relative και επίσης με τα top και left τα έβαλα στο ίδιο επίπεδο και κεντραρισμένα στο κέντρο της οθόνης για μικρότερο ύψος της οθόνης για μικρότερο σκρολάρισμα.

Το .lalabel το τροποποίησα με το position relative και τα left, bottom με σκοπό να εμφανίζεται κάτω δεξιά στην οθόνη εντός των ορίων σχεδόν σαν την desktop μορφή.

Το img.hidden δεν εμφανίζεται στην έκδοση mobile για απλότητα.

Τέλος το ul και το .u1 τα τροποποίησα με τα display block και position relative να εμφανίζονται στο ύψος των βελών (.btnl, .btnr) από αριστερά και μάλιστα το ένα κάτω από το άλλο για αποφυγή σκρολαρίσματος από αριστερά στα δεξιά.

Δεν κατάφερα να χρησιμοποιήσω flex, επίσης δεν κατάφερα να εμφανίζω αποκρύπτω στήλες με css. Η λύση ήταν η χρήση javascript.

ΤΕΛΟΣ ΜΕΘΟΔΟΙ ΚΑΣΚΑΛΗ