

ΠΡΟΧΩΡΗΜΕΝΑ ΘΕΜΑΤΑ
ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ ΕΦΑΡΜΟΓΩΝ
ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΕΡΓΑΣΙΑ ΓΙΑ
ΤΟ ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2019-2020

ΟΜΑΔΑ BEAST

ΧΡΗΣΤΟΣ ΓΡΗΓΟΡΙΑΔΗΣ,
2860

ΔΗΜΗΤΡΗΣ ΤΣΑΚΤΣΙΡΑΣ, 2838

ΧΡΗΣΤΟΣ ΣΟΥΡΗΣ, 2819

ΤΕΛΙΚΗ ΑΝΑΦΟΡΑ

ΙΣΤΟΡΙΚΟ ΠΡΟΗΓΟΥΜΕΝΩΝ ΕΚΔΟΣΕΩΝ

Ημερομηνία	Έκδοση	Περιγραφή	Συγγραφέας
yyyy/mm/dd	x.x		

Το κείμενο συμπληρώνεται προοδευτικά, όπως προχωρείτε στις φάσεις του Project.

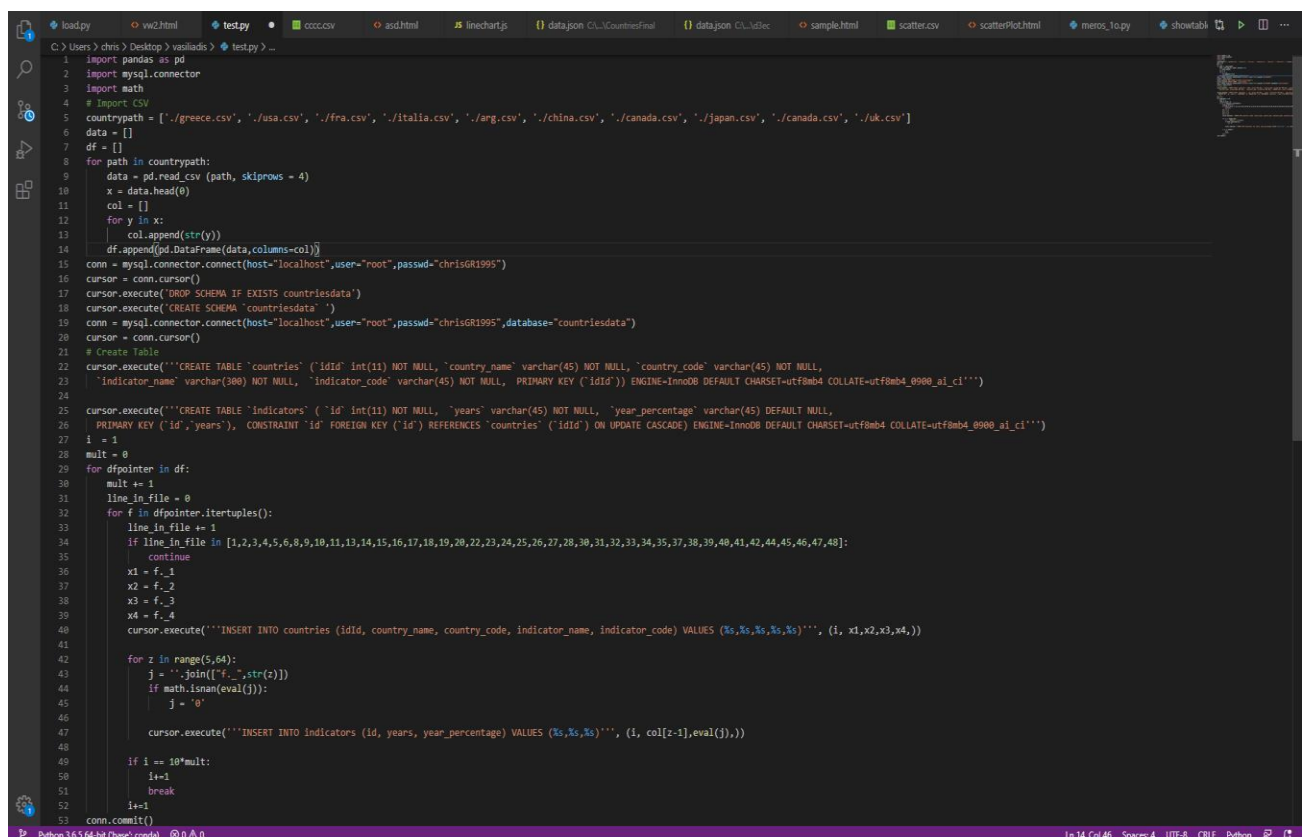
1 ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

1.1 ΣΧΕΣΙΑΚΟ ΣΧΗΜΑ ΣΕ ΛΟΓΙΚΟ ΕΠΙΠΕΔΟ

Παρακάτω παραθέτουμε ένα python script το οποίο δημιουργεί ένα νέο σχήμα (countriesdata.db) , δημιουργεί τα απαραίτητα tables και τα γεμίζει με τα δεδομένα που φορτώνουμε από τα csv αρχεία.

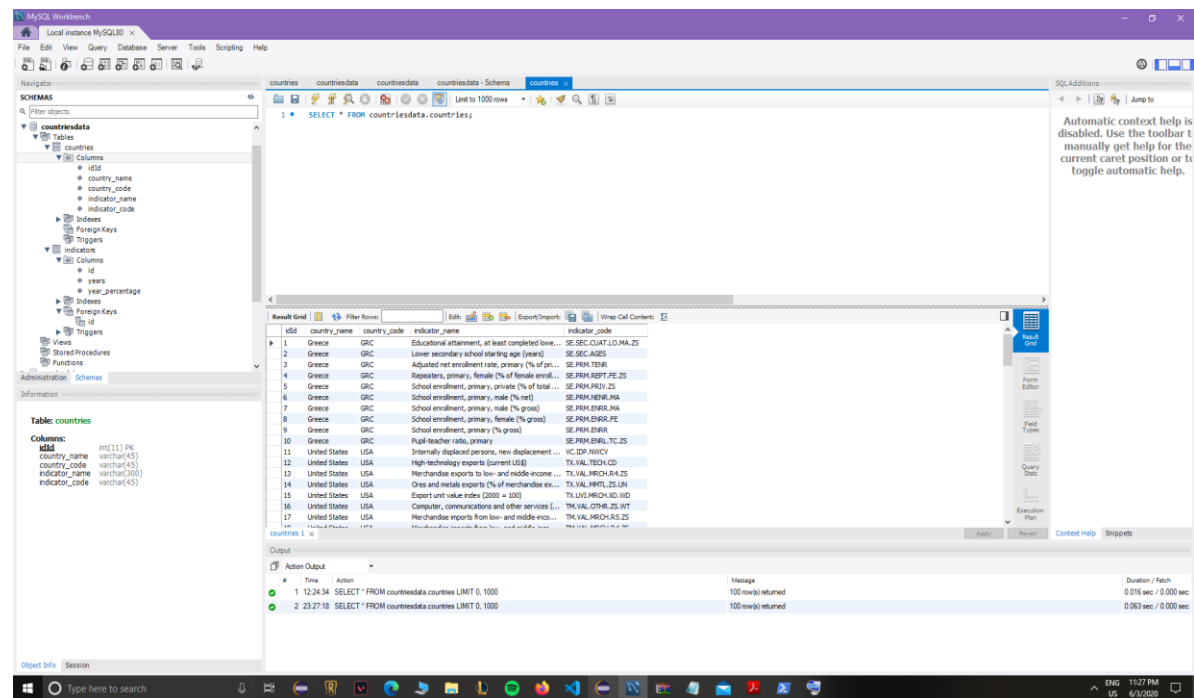
Απαραίτητες προϋποθέσεις:

- Να τρέχει ο sql server
- Να υπάρχει το mysql module της python (pip install mysql-connector-python)
- Σωστά στοιχεία για την σύνδεση με το server (γραμμή 15 και 19)

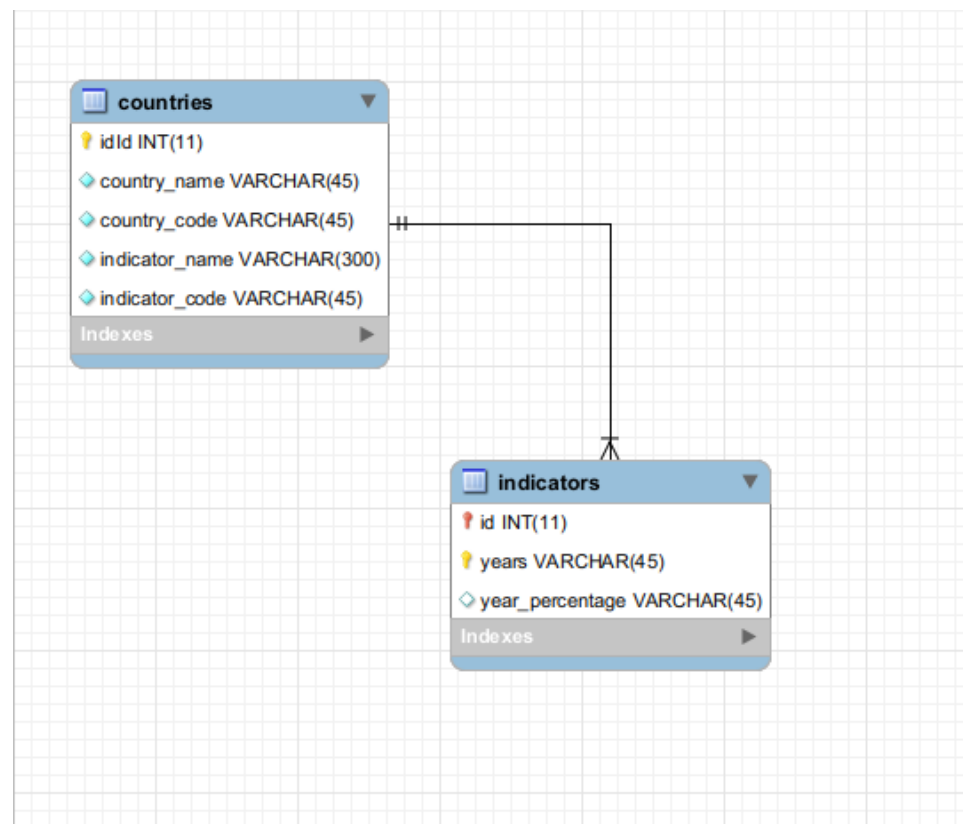


```
1 import pandas as pd
2 import mysql.connector
3 import math
4 # Import CSV
5 countrypath = ['./greece.csv', './usa.csv', './fra.csv', './italia.csv', './arg.csv', './china.csv', './canada.csv', './japan.csv', './canada.csv', './uk.csv']
6 data = []
7 df = []
8 for path in countrypath:
9     data = pd.read_csv(path, skiprows = 4)
10    x = data.head(0)
11    col = []
12    for y in x:
13        col.append(str(y))
14    df.append(pd.DataFrame(data, columns=col))
15 conn = mysql.connector.connect(host="localhost", user="root", passwd="chr1sGRI995")
16 cursor = conn.cursor()
17 cursor.execute("DROP SCHEMA IF EXISTS countriesdata")
18 cursor.execute("CREATE SCHEMA 'countriesdata' ")
19 conn = mysql.connector.connect(host="localhost", user="root", passwd="chr1sGRI995", database="countriesdata")
20 cursor = conn.cursor()
21 # Create table
22 cursor.execute("CREATE TABLE 'countries' ('id' int(11) NOT NULL, 'country_name' varchar(45) NOT NULL, 'country_code' varchar(45) NOT NULL,
23 'indicator_name' varchar(300) NOT NULL, 'indicator_code' varchar(45) NOT NULL, PRIMARY KEY ('id')) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci")
24
25 cursor.execute("CREATE TABLE 'indicators' ('id' int(11) NOT NULL, 'years' varchar(45) NOT NULL, 'year_percentage' varchar(45) DEFAULT NULL,
26 PRIMARY KEY ('id','years'), CONSTRAINT 'id' FOREIGN KEY ('id') REFERENCES 'countries' ('id') ON UPDATE CASCADE) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci")
27 i = 1
28 mult = 0
29 for dfpointer in df:
30     mult += 1
31     line_in_file = 0
32     for f in dfpointer.iteruples():
33         line_in_file += 1
34         if line_in_file in [1,2,3,4,5,6,8,9,10,11,13,14,15,16,17,18,19,20,22,23,24,25,26,27,28,30,31,32,33,34,35,37,38,39,40,41,42,44,45,46,47,48]:
35             continue
36         x1 = f._1
37         x2 = f._2
38         x3 = f._3
39         x4 = f._4
40         cursor.execute("INSERT INTO countries (id, country_name, country_code, indicator_name, indicator_code) VALUES (%s,%s,%s,%s,%s)", (i, x1,x2,x3,x4))
41
42         for z in range(5,64):
43             j = ''.join(["f_",str(z)])
44             if math.isnan(eval(j)):
45                 j = '0'
46
47             cursor.execute("INSERT INTO indicators (id, years, year_percentage) VALUES (%s,%s,%s)", (i, col[z-1],eval(j)))
48
49         if i == 10*mult:
50             i+=1
51             break
52         i+=1
53     conn.commit()
```

Όταν τελειώσει το script το workbench θα είναι ως εξής:



1.2 ΣΧΕΣΙΑΚΟ ΣΧΗΜΑ ΣΕ ΦΥΣΙΚΟ ΕΠΙΠΕΔΟ



1.2.1 ΡΥΘΜΙΣΗ ΤΩΝ ΠΑΡΑΜΕΤΡΩΝ ΤΟΥ DBMS

Στο python script όταν δημιουργούμε τα tables στο τέλος της ερώτησης προσθέτουμε
`ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci`
(γραμμή 25)

storage engine, memory allocation (of various kinds), ...

1.2.2 ΡΥΘΜΙΣΗ ΤΟΥ ΦΥΣΙΚΟΥ ΣΧΗΜΑΤΟΣ ΤΗΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ

ορισμός πιθανών ευρετηρίων (indexes), όψεων (views) που είναι υλοποιημένες ή μη, αλλαγές στο σχήμα των πινάκων για λόγους απόδοσης, κλπ. Τεκμηριώστε τα παραπάνω με βάση τα πλάνα από τα ερωτήματα που καθυστερούν ή με βάση την εσωτερική δομή του κώδικα και της δυσκολίας συγγραφής του.

1.2.3 ΡΥΘΜΙΣΗ ΑΣΦΑΛΕΙΑΣ

ορισμός δικαιωμάτων καταχώρησης ή ανάκτησης δεδομένων σε διαφορετικούς ρόλους και χρήστες του συστήματος.

2 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΛΟΓΙΣΜΙΚΟΥ

2.1 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΚΑΙ ΔΟΜΗ ETL

Αρχικά πρέπει να φέρουμε τα δεδομένα μέσα στη βάση μας για περαιτέρω επεξεργασία. Εδώ καταγράφεται η αρχιτεκτονική της ETL διαδικασίας (είτε μέσω εργαλείου, είτε μέσω των όποιων scripts προεπεξεργασίας και φόρτωσης δεδομένων φτιάξετε).

Είναι σημαντικό η διαδικασία να καταγραφεί με ακρίβεια στις λεπτομέρειες. Μπορείτε να χρησιμοποιήσετε UML-based / BPMN /ETL-specific formalisms για τη διαγραμματική απεικόνιση. Δείτε τις σχετικές οδηγίες στο συνοδευτικό κείμενο στο web site του μαθήματος.

Λογικά, για ότι είναι αυτοματοποιημένο, αρκεί να πείτε τι ρυθμίσεις χρειάζονται.

Αν έχετε όμως παρεμβάσεις που γίνονται manually, πρέπει να καταγραφούν επίσης οι λεπτομέρειες.

2.2 ΔΙΑΓΡΑΜΜΑΤΑ ΠΑΚΕΤΩΝ / ΥΠΟΣΥΣΤΗΜΑΤΩΝ ΚΕΝΤΡΙΚΗΣ ΕΦΑΡΜΟΓΗΣ

Το διάγραμμα για τα υποσυστήματα / πακέτα του λογισμικού που κατασκευάσατε ως κεντρική εφαρμογή επερώτησης. Ο στόχος είναι να φανεί η high-level αρχιτεκτονική του συστήματος, χωρίς λεπτομέρειες των επί μέρους κλάσεων.

2.3 ΔΙΑΓΡΑΜΜΑ(ΤΑ) ΚΛΑΣΕΩΝ ΚΕΝΤΡΙΚΗΣ ΕΦΑΡΜΟΓΗΣ

Αν η ανάπτυξη γίνει αντικειμενοστρεφώς, εδώ μπαίνουν τα διαγράμματα κλάσεων + ο σχολιασμός της κεντρικής εφαρμογής. Αλλιώς μπαίνουν διαγράμματα που διευκολύνουν την κατανόηση της εσωτερικής αρχιτεκτονικής του λογισμικού (π.χ., component/ deployment diagrams / ...)

3 ΥΠΟΔΕΙΓΜΑΤΑ ΕΡΩΤΗΣΕΩΝ ΚΑΙ ΑΠΑΝΤΗΣΕΩΝ

Select Chart:

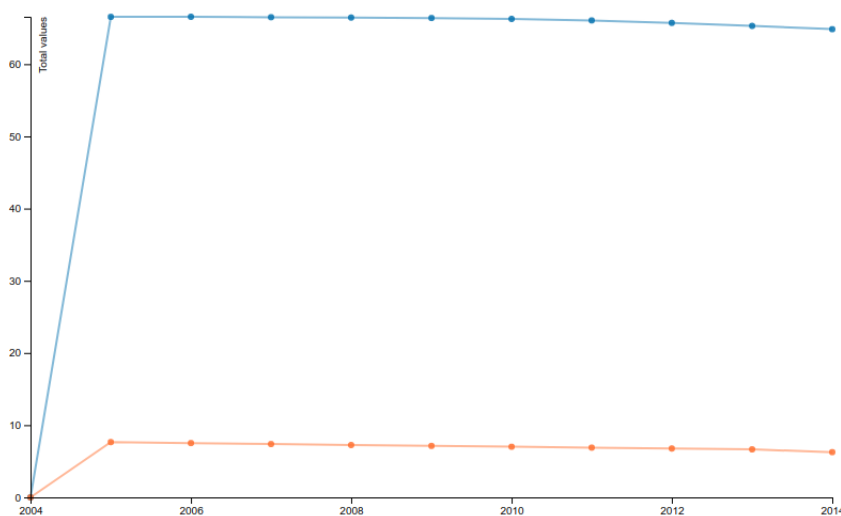
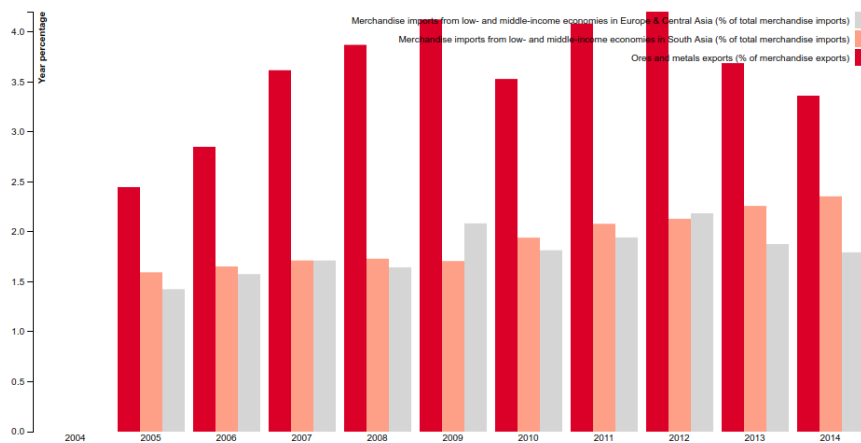
Select Countries:

Select Indicators:

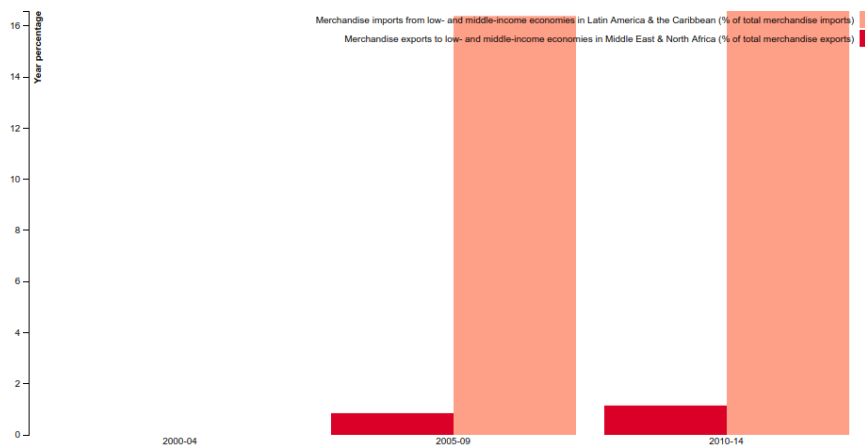
Select Presentation:

Select Year From:

Select Year To:



Ανά 5 χρόνια:



4 ΤΕΚΜΗΡΙΩΣΗ ΚΑΙ ΛΟΙΠΑ ΣΧΟΛΙΑ

ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

- Στη βάση μας έχουμε δεδομένα για 10 χώρες.
- Κάθε χώρα έχει δεδομένα για 10 διαφορετικούς indicators.
- Οι Χώρες, μεταξύ τους, δεν έχουν τους ίδιους indicators.

ΔΙΕΠΑΦΗ ΧΡΗΣΤΗ

Το πρόγραμμα αρχίζει ανοίγοντας το παράθυρο με τις επιλογές που μπορεί να επιλέξει ο χρήστης. Οι επιλογές που μπορεί να παραμετροποιήσει ο χρήστης είναι οι εξής:

- Επιλογή γραφικής παράστασης
- Επιλογή Χώρας
- Επιλογή Indicator
- Επιλογή παρουσίασης ανά χρόνια
- Επιλογή χρονιάς από
- Επιλογή χρονιάς μέχρι

Οι επιλογές που εμφανίζονται στις Χώρες, Indicators , Year From, Year To, φορτώνονται από τη βάση. Οπότε όταν ο χρήστης όταν επιλέξει μια Χώρα στο drop down μενού των indicators θα εμφανίζεται κάθε φορά τα indicators αυτής της Χώρας.

Αν ο χρήστης δεν θέλει κάποια επιλογή για τον τρόπο παρουσίασης, δηλαδή να εμφανιστούν όλες οι χρονιές, θα πρέπει να αφήσει κενό το πεδίο επιλογής για την παρουσίαση.

BAR CHART - TIMELINE PLOT

- Ο χρήστης μπορεί να επιλέξει όσες Χώρες, Indicators και παρουσίαση θέλει.

SCATTER PLOT

- Ο χρήστης πρέπει να επιλέξει μόνο μια χώρα με δύο indicators και όποια παρουσίαση θέλει.

Για να πάρει το πρόγραμμα τους indicators που επέλεξε ο χρήστης πρέπει να πατήσει το κουμπί "Add To Chart". Κάθε φορά που θέλει να δημιουργήσει νέο διάγραμμα πρέπει να κάνει την ίδια δουλειά. Επίσης, υπάρχει και ένα παραθυράκι που ο χρήστης βλέπει τις επιλογές που έχει βάλει για να δημιουργηθεί το διάγραμμα.

Πατώντας το κουμπί "Run" το πρόγραμμα θα δρομολογήσει τις επιλογές του χρήστη για την δημιουργία του διαγράμματος και θα ανοίξει το διάγραμμα σε έναν περιηγητή.

ΔΗΜΙΟΥΡΓΙΑ ΔΙΑΓΡΑΜΜΑΤΩΝ

Ο κώδικας για την δημιουργία του Bar Chart και του Timeline Plot είναι παρόμοιος . Έχουμε φτιάξει την κλάση Parameter όπου περνάμε τις επιλογές του χρήστη και καλούμε την barChart.class ή την Timeline.class. Μέσα σε αυτές τις κλάσεις φτιάχνουμε το query το οποίο είναι το ίδιο για οποιαδήποτε παρουσίαση επιλέξει ο χρήστης. Η μετατροπή του σε 5ετία - 20ετία γίνεται μέσα από τον κώδικα που έχουμε υλοποιήσει. Ουσιαστικά ο χρήστης θα μπορούσε να επιλέξει οποιαδήποτε μορφή παρουσίασης και όχι μόνο 5/10/20ετία. Μετά από αυτή την επεξεργασία του query δημιουργούμε τα αρχεία .js με τα δεδομένα που προέκυψαν. Τέλος, καλούμε την openHtml για να ανοίξουμε το αρχείο που δημιουργεί τα διαγράμματα. Για να ανοίξουν τα αρχεία πρέπει να ανοίξουμε ένα τοπικό server και αυτό το κάνουμε μέσα τρέχοντας ένας server με τη βοήθεια της python.

Το query για το Bar Chart/ Timeline Plot:

```
public String getQuery(String country, String indicator, int yearFrom, int yearTo) {
    String query = "SELECT years,year_percentage" +
        " FROM countries,indicators " +
        "WHERE countries.idId = indicators.id " +
        "AND country_name = '" + country + "' AND indicator_name = '" + indicator + "' " +
        "AND years>= " + yearFrom + " AND years<= " + yearTo + " ";
    return query;
}
```

Το Scatter Plot είναι λίγο διαφορετικό γιατί δεν φτιάχνουμε .js αρχεία, αλλά csv. Επίσης, επειδή υποτίθεται ότι στο scatter plot θα δώσουμε δύο δεδομένα μιας χώρας τρέχουμε διαφορετικό query. Λόγω της μορφής που θέλει να περάσουμε τα δεδομένα στο csv αρχείο φτιάχνουμε και τρέχουμε ταυτόχρονα τα δύο queries για τα δύο indicators. Τέλος, φτιάχνουμε το .html αρχείο και καλούμε την openHtml για να μας δείξει το διάγραμμα.

Το query που τρέχουμε για το Scatter plot είναι παραμετροποιήσιμο για να ζητάει από τη βάση αν ο χρήστης έχει δώσει τρόπο παρουσίασης(5/10/20ετία), ώστε να επιστρέφει το AVERAGE των ποσοστών από τις χρονολογίες.

To query για το Scatter Plot:

```
private String getQuery1(String str1,String str2, String indicator, int yearFrom, int yearTo ) {  
    String query = "SELECT "+ str1 + "year_percentage " + str2 + "AS year_percentage FROM countries,indicators " +  
        "WHERE countries.idId = indicators.id " +  
        "AND country_name = '" + parameter.getCountry() + "' " + "AND indicator_name = '" + indicator + "' " +  
        "AND years>= " + yearFrom + " AND years<= " + yearTo + " ";  
    return query;  
}
```