ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΡΟΧΩΡΗΜΕΝΑ ΘΕΜΑΤΑ ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ ΕΦΑΡΜΟΓΩΝ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΕΡΓΑΣΙΑ ΓΙΑ ΤΟ ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2022-2023

ΟΜΑΔΑ SOLO

ΝΙΚΟΣ ΤΑΦΛΑΜΠΑΣ, 4500

ΤΕΛΙΚΗ ΑΝΑΦΟΡΑ

ΜΑΪΟΣ 2023

ΙΣΤΟΡΙΚΟ ΠΡΟΗΓΟΥΜΕΝΩΝ ΕΚΔΟΣΕΩΝ

Ημερομηνία	Έκδοση	Περιγραφή	Συγγραφέας
2023/3/4	0.1	Database Schema, DDL, ETL Completion. Data loaded (minus the big file).	Nick
2023/4/16	0.2	Front-End finished.	Nick
2023/4/30	0.3	Disaster. Big file loaded. Major rework of Front End.	Nick
2023/5/3	0.4	Refinement. Simplified code. Added 5-year queries. Finished.	Nick
2023/5/19	-	Analysis, PDF	Nick

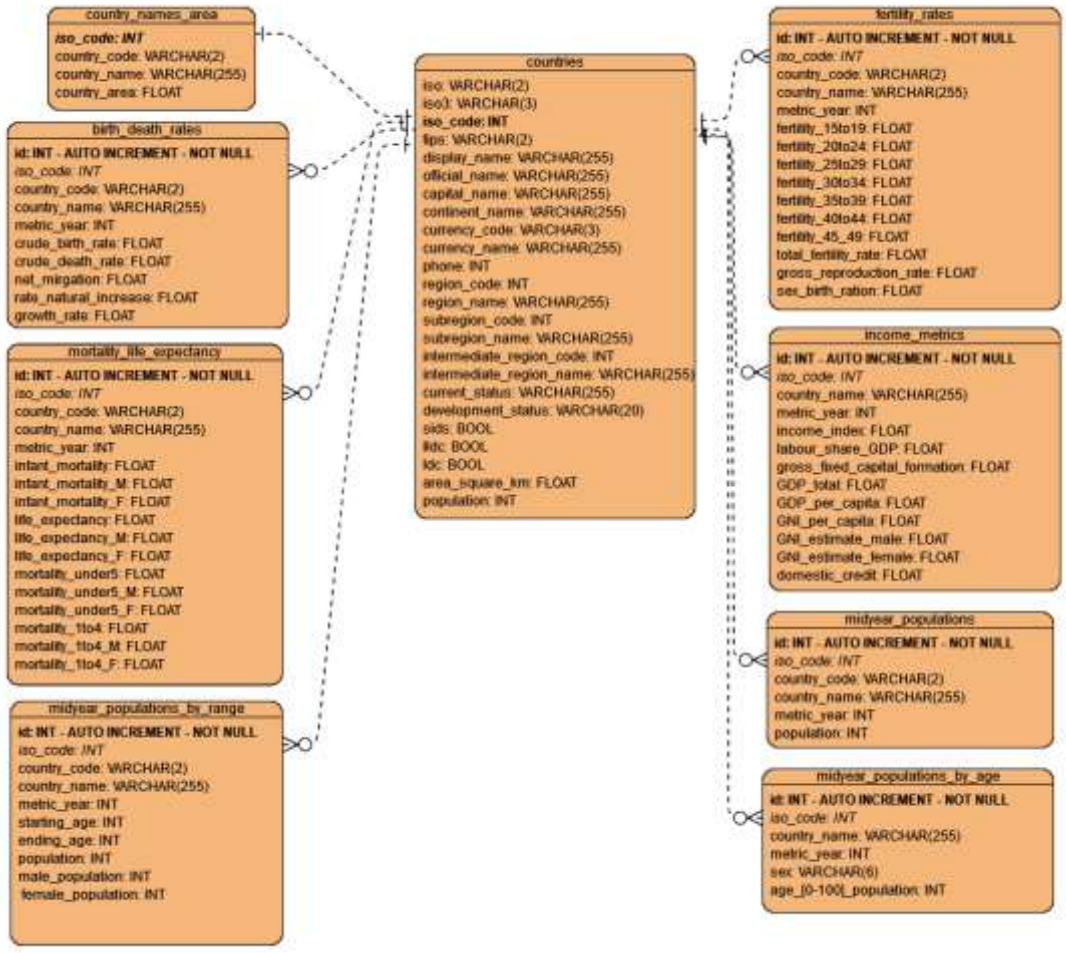
Το κείμενο συμπληρώνεται προοδευτικά, όπως προχωρείτε στις φάσεις του Project.

1 ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

Στην παρούσα ενότητα περιγράφονται τα σχήματα της βάσης δεδομένων που χρησιμοποιούνται στο project.

Ο σχετικος κωδικας για την δημιουργια των Tables θα βρήσκεται σε ξεχωριστό αρχειο του Github Repository

1.1 ΣΧΕΣΙΑΚΟ ΣΧΗΜΑ ΣΕ ΛΟΓΙΚΟ ΕΠΙΠΕΔΟ



Σχήμα 1.1 Σχεσιακό σχήμα της βάσης δεδομένων του συστήματο

1.2 ΣΧΕΣΙΑΚΟ ΣΧΗΜΑ ΣΕ ΦΥΣΙΚΟ ΕΠΙΠΕΔΟ

Όταν θα έχετε στήσει και ρυθμίσει τη βάση δεδομένων σας, εδώ καταγράφονται και οι ρυθμίσεις σε φυσικό επίπεδο.

1.2.1 ΡΥΘΜΙΣΗ ΤΩΝ ΠΑΡΑΜΕΤΡΩΝ ΤΟΥ DBMS

Τυπικές αλλαγές που γίναν στις μεταβλητές της βάσης μας, ήταν η μεταβλητή innodb_buffer_pool_size. Σε ιδανικό σενάριο, η αλλαγή του σε ένα 70-80% της μνήμης της συσκευής θα κάνει την φόρτωση των δεδομένων πιο γρήγορη. Στην περίπτωση μας, το έθεσα σε περίπου 2Gb.

1.2.2 ΡΥΘΜΙΣΗ ΤΟΥ ΦΥΣΙΚΟΥ ΣΧΗΜΑΤΟΣ ΤΗΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ

Αρκετές τιμές πεταχτήκαν από τα tables, καθώς μας ήταν άχρηστα, ή απλά δεν ξέραμε τι είναι. Κάποιες σημειώσεις για τόσο το ETL όσο και το σχήμα της βάσης μας είναι οι ακόλουθες:

-Υπάρχει συνεχής επανάληψη σε πολλά Tables των τιμών Country Code και Country Name. Αυτές υπάρχουν και στο Table Countries που είναι το κεντρικό ευρετήριο κάθε χώρας. Συνεπώς, θεωρητικά, μπορούν να αφαιρεθούν από κάθε table, αφού είναι περιττή πληροφορία.

-Ένα βασικό πρόβλημα που πέτυχα, είναι ο πίνακας midyear_populations_by_age. Ο πίνακας εξαρχής είχε 101 στήλες οπού κάθε στήλη έχει τον πληθυσμό για μία ηλικία. Αυτό, για να συμβαδίζει με το pattern που ακολούθησα με τα υπόλοιπα Tables, προσπάθησα να το μετατρέψω σε 101 ξεχωριστές εγγραφές του τύπου:

Id,*Iso_code*, metric_year, sex, age, population

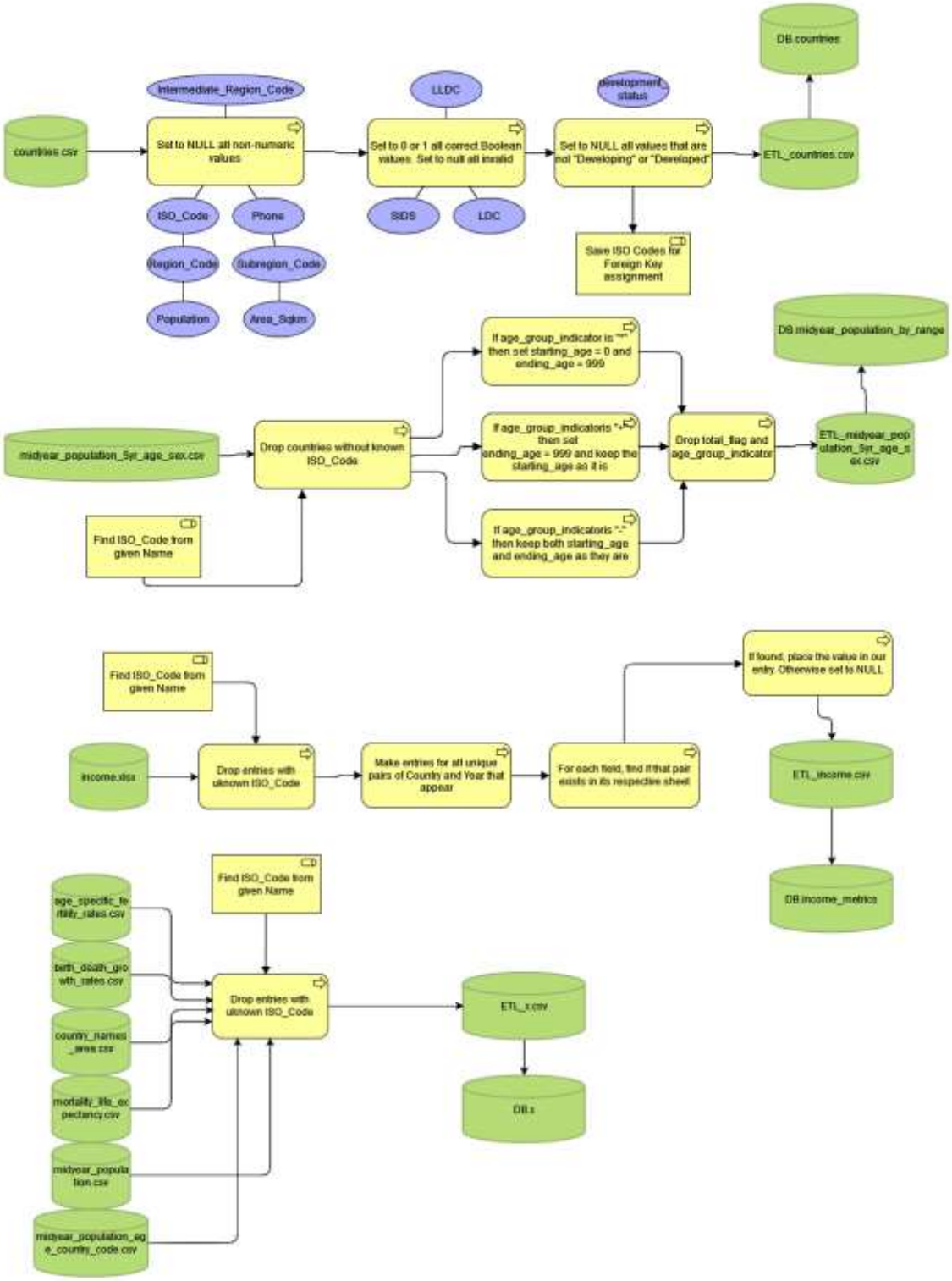
Αυτό δυστυχώς δεν έγινε. Ο λόγος είναι η έλλειψη φυσικού χώρου. Αν και το θετικό είναι ότι συμβαδίζει με τον τρόπου που υλοποιούμε τα queries μας, έχει το αρνητικό ότι για ΚΑΘΕ σειρά των αρχικών δεδομένων μας, δημιουργούμε 101 τιμές. Τα 1.7Gb πληροφορίας γίνονται μετά το ETL περίπου 17Gb, και με την φόρτωση στην βάση μας και την κατασκευή σε B+ Tree (διαδικασία που παίρνει περίπου 1 ώρα με της παραπάνω ρυθμίσεις), ο τελικός φόρτος είναι περίπου 25-30Gb. Αυτό, σε συνδυασμό μιας κατάρρευσης του DBMS, που ευτυχώς λύθηκε χάρης των Back-up, έκανε την υλοποίηση αδύνατη.

Σε σενάριο που έχουμε dedicated server, η παραπάνω υλοποίηση θεωρώ ότι είναι προτιμητέα, καθώς δίνει επεκτασιμότητα σε περισσότερες ηλικίες. Αλλά αφού δεν έχουμε αρκετό χώρο, αρκεστήκαμε στην τελική υλοποίση.

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 που γίνεται στο αρχείο Preprocessing.java του πακέτου Preprocessing.



Η δομή των περισσότερων πινάκων διατηρήθηκε, δεδομένου ότι τα στοιχεία τους ήταν σε αποδεκτή κατάσταση. Όσο για τους πίνακες income και midyear_population_by_range, τα δεδομένα πέρασαν μια διαδικασία απλοποίησης. Ο πίνακας countries από την άλλη, είχε αρκετά στοιχεία που δεν ήταν στην σωστή μορφή. Μετά το φιλτράρισμα όμως, μπορέσαμε εύκολα να τραβήξουμε τα ISO_CODES και να τα ενώσουμε με το αντίστοιχο όνομα της χώρας.

Iso Codes

Ένα βασικό πρόβλημα είναι ότι οι χώρες σε όλα τα αρχεία αναφέρονται μόνο από το όνομά τους, και συνεπώς δεν έχουμε κάποιο καλό Foreign/Primary Key για να τις ενώσουμε με άλλους πίνακες. Για αυτό τον λόγο, αποθηκεύουμε τα ακριβοί ονόματα από το αρχείο countries.csv μαζί με το iso_code. Έτσι, όταν βάζουμε ένα νέο entry στα ETL δεδομένα μας, βρίσκουμε το iso_code τους από το όνομα και το προσθέτουμε. Αν δεν υπάρχει, απλα το πετάμε.

Το πρόβλημα όμως είναι ότι μερικές χώρες δεν έχουν το ακριβώς ίδιο όνομα σε όλα τα αρχεία (πχ “Bahamas”, “The Bahamas” και “Bahamas, The”. Έτσι, αν το πρόγραμμά μας δεν βρει το iso_code στο λεξικό μας, ψάχνει μήπως η χώρα έχει διατυπωθεί αλλιώς, με την χρήση RegEx. Μαζί δίνεται ένα αρχείο με όνομα **country_map.txt** που δίνει έτοιμα όλα τα RegEx με το αντίστοιχο iso_code τους, για τα δεδομένα μας.

Income Metrics

Το αρχείο Income by Country έχει τα δεδομένα μας σε με ιδανική κατάσταση. Ο τρόπος που τα μετέτρεψα σε csv, ήταν να φτιάξω μοναδικά ζευγάρια για όλες τις τιμές που βρίσκονται της μορφής Country-Year, και να βάλω μια νέα σειρά στον πίνακα μου με ότι στοιχείο μπορώ να βρω. Πχ, το ζευγάρι Afghanista-1990 έχει την τιμή 0.466 στο πεδίο income_index, και την τιμή 2193 στο GNI_per_capita, αλλά δεν έχει τιμή στο Labour share of GDP επειδή δεν υπάρχει το 1990 σαν έτος. Συνεπώς γεμίζω την τιμή αυτή σαν NULL.

DDL and To-Do

Το DDL.java φτιάχνει τους πίνακες **σε ένα κενό database,** και φορτώνει τα ETL δεδομένα αυτόματα. Για να τρέξουν όμως όλα αυτά, μαζί και το ETL, πρέπει να δοθούν τα σωστά στοιχεία και στα 2 αρχεία. Στο preprocessing:

```
static final String IN_PATH = "C:\\Users\\nikos\\Desktop\\OldData\\";
static final String OUT_PATH = "C:\\Users\\nikos\\Desktop\\FixedData\\";
```

Το IN_PATH έχει το path για τα αρχεία μας πριν την επεξεργασία, και το OUT_PATH την τελική τοποθεσία των αρχείων ETL.

Στο DDL:

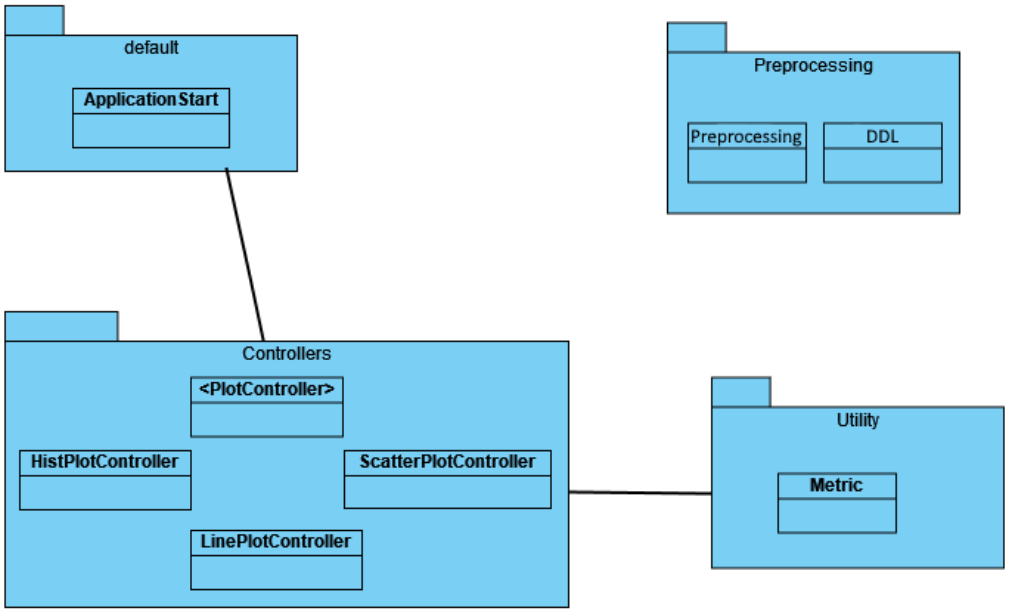
```
static final String ETL_PATH = "C://Users//nikos//Desktop//FixedData//";

static Connection setConnection() throws Exception
{
    String url = "jdbc:mysql://localhost:3306/countrydatabase";
    String name = "root";
    String password = "password";
```

Το ETL_PATH πρέπει να είναι το ίδιο με το OUT_PATH. Οσο για τα υπόλοιπα, είναι τα στοιχεία της βάσης μας.

Σαν υποσημείωση, θα αναφέρω ότι το DDL θα μπορούσε απλα να διαβάζει ένα SQL script αντι να έχει hard-coded τον κώδικα. Σαφώς αυτό δεν εγινε για διάφορους λόγους, αλλα κυρίως για το πρόβλημα με το **midyear_populations_by_age,** καθώς θα ήταν δύσκολο να γραφτούν 100 γραμμές για έναν πίνακα.

2.2 ΔΙΑΓΡΑΜΜΑΤΑ ΠΑΚΕΤΩΝ / ΥΠΟΣΥΣΤΗΜΑΤΩΝ ΚΕΝΤΡΙΚΗΣ ΕΦΑΡΜΟΓΗΣ

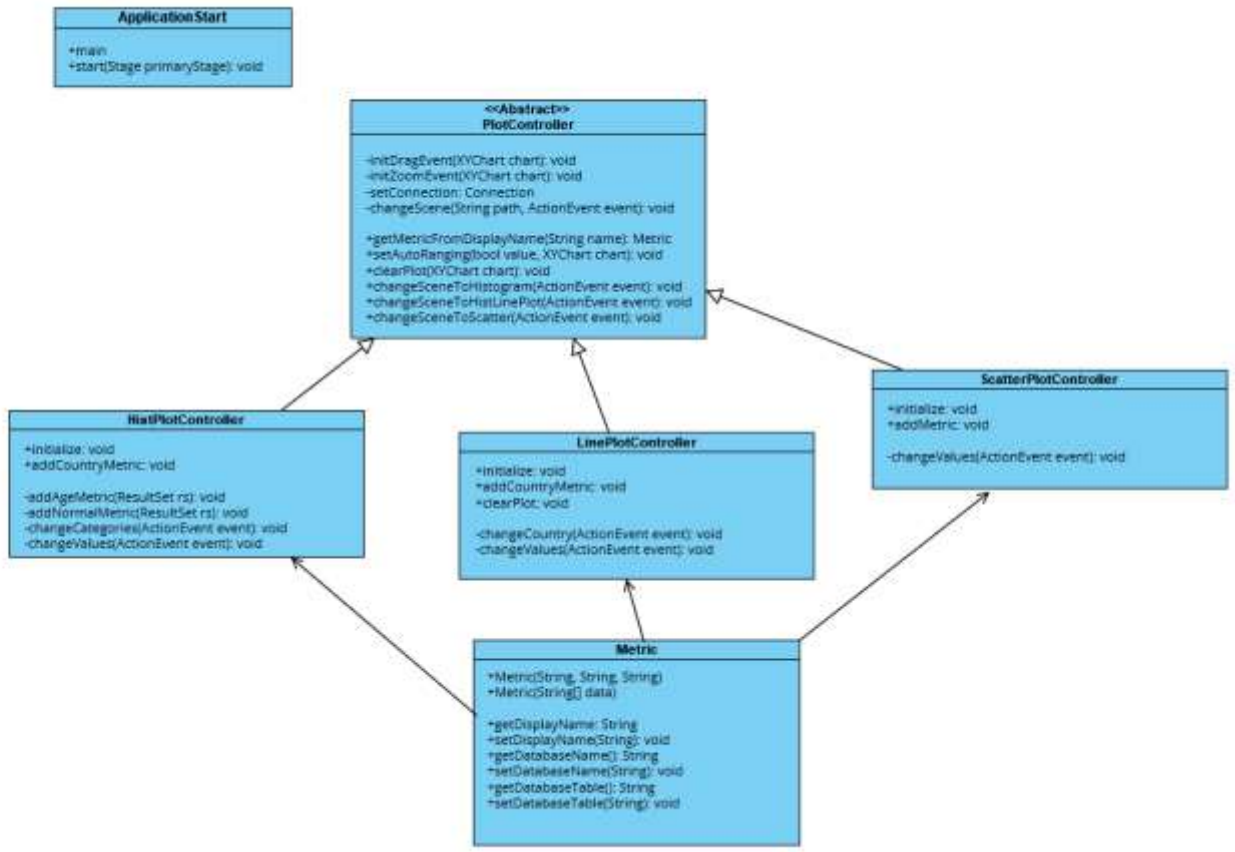
Το διάγραμμα για τα υποσυστήματα / πακέτα του λογισμικού που κατασκευάσατε ως κεντρική εφαρμογή επερώτησης..



Τα πακέτα μας είναι 4, εκ των οποίων το 4ο (Preprocessing) δεν παίζει κάποιον ρόλο στην εφαρμογή, και απλά υπάρχει για να διαχωρίζει τα ETL-DDL scripts από τα υπόλοιπα. Το default περιέχει την Main μας, η οποία τρέχει την εφαρμογή. Και τέλος τα Controllers ελέγχουν τις βασικές λειτουργίες για κάθε οθόνη. Το πακέτο Utillity έχει το αντικείμενο Metric που διευκολύνει την χρήση πεδίων, καθώς και την ανάκτηση πληροφορίας από την βάση μας.

2.3 ΔΙΑΓΡΑΜΜΑ(ΤΑ) ΚΛΑΣΕΩΝ ΚΕΝΤΡΙΚΗΣ ΕΦΑΡΜΟΓΗΣ

Αν η ανάπτυξη γίνει αντικειμενοστρεφώς, εδώ μπαίνουν τα διαγράμματα κλάσεων και ο σχολιασμός της κεντρικής εφαρμογής.



Οι κλάσεις Controller χειρίζονται μέσω της βιβλιοθήκες JavaFX όλες τις λειτουργίες του συστήματος. Είτε αυτό είναι να πρόσθεση μια μετρική σε κάποιο διάγραμμα είτε απλά το πάτημα ενός κουμπιού. Σαν «καινούριες» προσθήκες, υλοποίησα ένα Drag και ένα Zoom Feature για μερικά από τα Plots, για την πιθανή διευκόλυνση στην ανάλυση των δεδομένων. Τα features αυτά είναι ελαφρώς προβληματικά λόγο της φύσης της JavaFX, και η χρήση τους σε μερικές μετρικές μπορεί να προκαλέσει προβλήματα (αλλά όχι κάτι που θα κρασάρει την εφαρμογή, και συνήθως το κουμπί clear plot φτιάχνει τις αλλοιώσεις).

3 ΥΠΟΔΕΙΓΜΑΤΑ ΕΡΩΤΗΣΕΩΝ ΚΑΙ ΑΠΑΝΤΗΣΕΩΝ

Όταν μια οθόνη αρχικοποιείται, ζητείται κατευθείαν μια σύνδεση με την βάση μας. Ετσι, τα queries μπορούν να εκτελεστούν. Μπορεί κάνεις να βρει το αρχείο values_map.txt το οποίο κρατάει τα στοιχεία για τις περισσότερες μετρικές μας στην μορφή:

Display Name:Database Name:Table Name

Με αυτό το αρχείο, έχω όλα τα μετρικά μου στοιχεία, και μπορώ ευκολά να βρω σε πιο Table ανήκουν, καθώς και το όνομα του πεδίου τους στο table αυτό. Πχ η πρώτη γραμμή μας λέει πως το στοιχείο με όνομα Birth Rate(/1k) ανήκει στο Table birth_death_rates με όνομα crude_birth_rate.

Έπειτα, η τυπική διαδικασία είναι να γίνει Merge μεταξύ των 2 Tables από τα 2 στοιχεία που θέλουμε να παρουσιάσουμε. Το Merge τυπικά γίνεται πάνω στο iso_code και το έτος. Έπειτα, αφού πετάξουμε τις NULL τιμές, επιστρέφουμε ανά δυάδες τις τιμές, με αύξουσα σειρά ανά έτος.

```
query = "SELECT a." + valueMetric.getDatabaseName() + ", b." + coordinateMetric.getDatabaseName()
+ " FROM " + valueMetric.getDatabaseTable() + " a "
+ "JOIN " + coordinateMetric.getDatabaseTable() + " b ON a.metric_year = b.metric_year AND"
+ " a.iso_code = " + selectedCountryISOCode + " AND a.iso_code = b.iso_code AND a." + valueMetric.getDatabaseName()
+ " IS NOT NULL AND b." + coordinateMetric.getDatabaseName() + " IS NOT NULL ORDER BY b." + coordinateMetric.getDatabaseName();
```

Στα queries τυπικά το Value Metric αναπαριστά τον άξονα Y, ενώ το coordinate Metric τον άξονα X.

Μία εξαίρεση σε Query είναι όταν θέλουμε ο άξονας X να είναι το έτος. Αυτό φυσικά απλοποιεί τα πράγματα, καθώς δεν χρειαζόμαστε κάποιο Merge, αφού ΟΛΟΙ οι πίνακες έχουν το χρονικό έτος στα πεδία τους, και με ακριβώς το ίδιο όνομα.

```
query = "SELECT " + valueMetric.getDatabaseName() + ", metric_year "
        + "FROM " + valueMetric.getDatabaseTable() + " WHERE iso_code = " + selectedCountryISOCode
        + " AND metric_year IS NOT NULL AND " + valueMetric.getDatabaseName() + " IS NOT NULL "
        + " order by metric_year;";
```

5-Year-Interval

Το Histogram δίνει την δυνατότητα να παρουσιαστούν οι κάδοι ανά 5 έτη. Η διαδικασία είναι παρόμοια με αυτή του ανά έτος, αλλά αυτή την φορά τα κάνουμε order ανά το έτος τους δια του 5. Αφού το έτος είναι integer, όλες οι διαιρέσεις του θα κάνουν round down το αποτέλεσμα. Συνεπώς τα έτη 1900-1904 θα έχουν αποτέλεσμα το 380 όταν διαιρεθούν με το 5. Το 1905 όμως θα έχει 381, βάζοντάς το στην επόμενη 5αδα!

Το μόνο «πρόβλημα» με αυτή την υλοποίηση μας, είναι ότι το έτος 2050 θα είναι μόνο του στον τελικό κάδο, αφού τα δεδομένα μας τελειώνουν στο 2050. Για την ανάλυση των δεδομένων μας, ο τελευταίος κάδος παραλείπεται.

By Age

Το table midyear_population_by_range δεν έχει ιδιαίτερη χρήση στα δεδομένα μας, αφού μπορούμε να λάβουμε τα ίδια δεδομένα από τον midyear_population_by_age. Έτσι η επιλογή "By Age" στο Histogram μας δίνει την επιλογή να χρησιμοποιήσουμε αυτά τα δεδομένα όταν επιλέγουμε να τα δούμε ανά 5-ετη. **Η επιλογή αυτή βγαίνει μόνο όταν επιλέγουμε να δούμε το Population.**

4 ΛΟΙΠΑ ΣΧΟΛΙΑ

Λόγο φόρτου, δυστυχώς η τελική εφαρμογή είναι ελλιπής. Αν και περιέχει, θεωρώ, ότι έχει ζητηθεί, αυτό που είχα σχεδίαση δεν βγήκε πλήρως. Η επιλογή «Ανα 5 ετη» υπάρχει μόνο στο ιστόγραμμα. Επίσης το Line Plot είναι μη-ιδανικό καθώς τα δεδομένα σπάνια βγαίνουν σε μορφή που μπορούν να διαβαστούν. Αυτό φυσικά μπορεί να είναι λόγο των δεδομένων μας, ή λόγο της ίδιας JavaFX, που αν και εξαιρετικό εργαλείο, αποδείχθηκε προβληματική. Το ατυχές συμβάν με το τεράστιο Table επίσης είναι κάτι που σε πιο ιδανικές συνθήκες θεωρώ ότι θα μπορούσε να μου γλυτώσει πολλές γραμμές αχρείαστου κώδικα. Εν ολίγης, θα μπορούσε η εφαρμογή να γίνει πολύ καλύτερη, αλλά είναι σε ένα θεωρώ ικανοποιητικό στάδιο.

Οι βιβλιοθήκες που χρησιμοποιήθηκαν βρίσκονται στον φάκελο ETL-DDL-Libs καθώς και στον javafx-sdk-19.0.2.1.

-JavaFX (graphics, plotting, interface..)

-MySQL Connector (database communication)

-Apache POI (reading XML files)

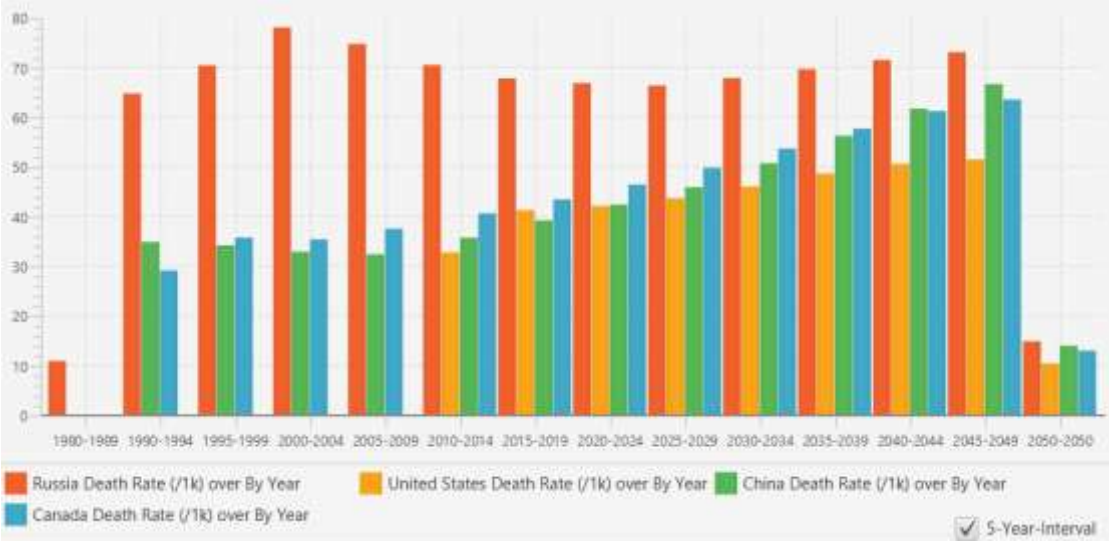
5 ΜΕΛΕΤΗ ΔΕΔΟΜΕΝΩΝ

Αφού λοιπόν έχουμε μια εφαρμογή ανάλυσης δεδομένων, πιο το νόημα να την φτιάχνουμε αν δεν την χρησιμοποιήσουμε.

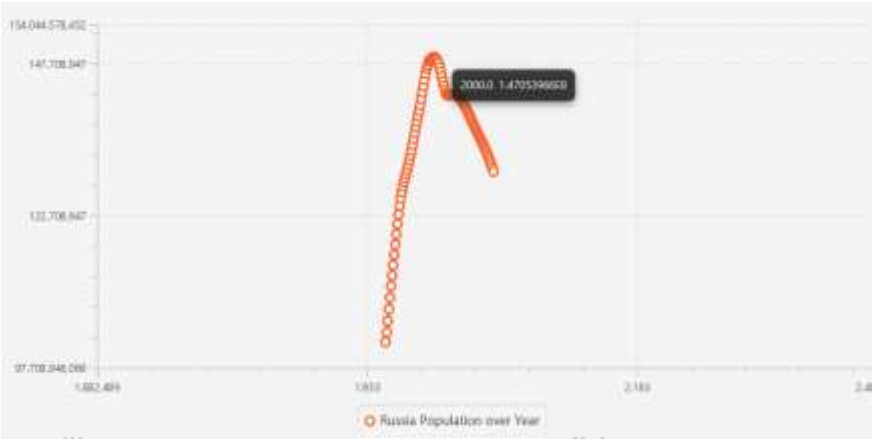
Ανάλυση στον Χρόνο

Έστω η εξής υπόθεση. *Με το πέρασμα του χρόνου, ο ανθρώπινος πληθυσμός αυξάνεται. Βάση λογικής, αν αυξάνεται το ποσοστό γεννήσεων, πρέπει να αυξάνεται και το ποσοστό θνησιμότητας.*

Ας δούμε λοιπόν αν αυτό ισχύει.



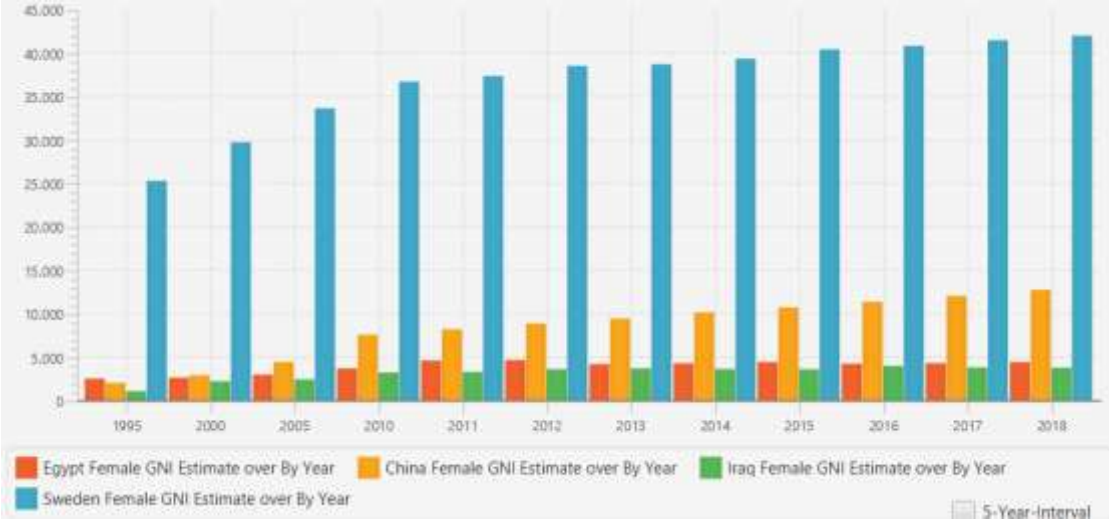
Διαλέγοντας μερικές από τις πιο μεγάλες χώρες, μπορούμε όντως να δούμε ότι τα ποσοστά θνησιμότητας από το 1990 έως το 2040 αυξάνονται με σταθερό ρυθμό. Άρα η υπόθεσή μας είναι πιθανό να είναι σωστή. Η Ρωσία όμως, έχει μια αυξομείωση στα ποσοστά της. Αρά λογικά, ο πληθυσμός της πρέπει κάπου γύρο στο 2000 να πέφτει σημαντικά



Πλοτάροντας τον πληθυσμό της Ρωσίας ανα έτος, όντως βλέπουμε μια πτώση μετά το 2000.

Σύγκριση Χωρών

Το να συγκρίνουμε μερικές χώρες σε μια άκυρη μετρική, δεν είναι ακριβώς επιστημονικό. Άρα ας κάνουμε και μία υπόθεση: *Με την ανάπτυξη των ηθικών αξίων, χώρες που είναι πιο αναπτυγμένες θα πρέπει να τηρούν πιο αυστηρά τους κοινωνικούς κανόνες περί σεξισμού. Συνεπώς, ο μέσος μισθός μιας γυναίκας πρέπει να δει σημαντική αύξηση σε χώρες πιο αναπτυγμένες στο πέρασμα του χρόνου.*

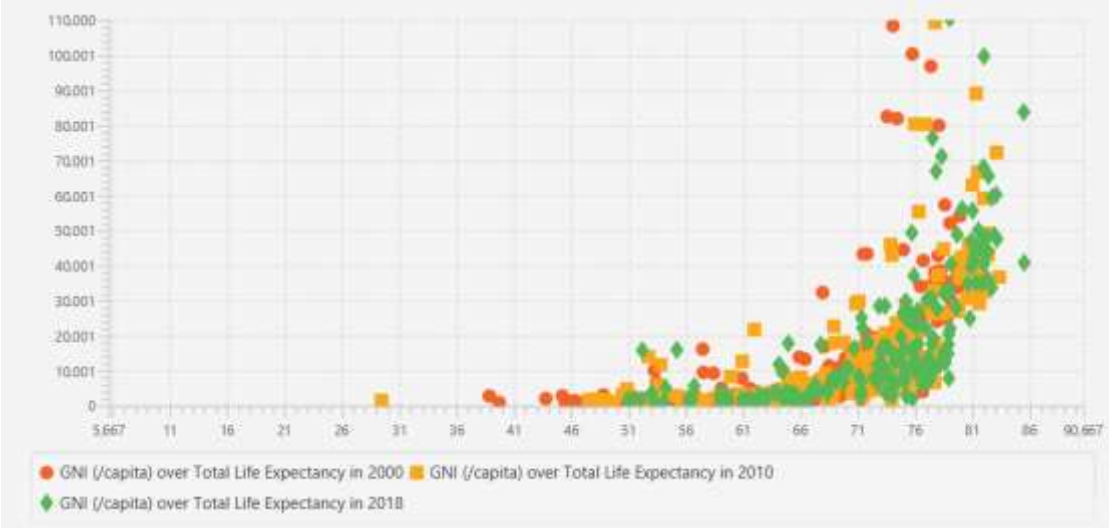


Αρχικά, παίρνουμε 4εις χώρες. Η Κίνα και η Σουηδία θεωρείτε αναπτυγμένη χώρα. Ενώ το Ιράκ και η Αίγυπτος είναι αναπτυσσόμενες χώρες. Η υπόθεση μας άλλη μια φορά φαίνεται να ισχύει, καθώς βλέπουμε ιδιαίτερα πιο σημαντική αύξηση στο μέσο γυναικείο μισθό μετά το 2005 στην Κίνα και την Σουηδία. Το Ιράκ και η Αίγυπτος από την άλλη, έχουν αυξομειώσεις λόγο πιθανώς της ασταθής οικονομίας.

Σχέση Οικομικών Στοιχείων – Δημογραφικών Στοιχείων

Τέλος, η υπόθεση μας θα βασιστεί στο να σύνδεση κάποιο οικονομικό στοιχείο με κάποιο δημογραφικό. Έστω λοιπόν η υπόθεση: *Η οικονομική κατάσταση μιας χώρας, καθώς και ενός πολίτη επηρεάζει την ποιότητα ζωής. Από γενική καθαριότητα έως διατροφή και φάρμακα, το να έχει κάποιος οικονομική άνεση του δίνει ένα πλεονέκτημα στην ζωή. Βάση αυτό, είναι λογικό να υποθέσουμε ότι χώρες με υψηλό εισόδημα, θα έχουν μεγαλύτερο μέσο όρο ζωής!*

Βάση αυτό, θα συγκρίνουμε γενικά όλες τις χώρες σε 3 έτη, βάση το γενικό GNI και τον μέσο όρο ζωής:



Κάθε σημείο στο Scatter Plot μας είναι και μία χώρα. Αρχικά, βλέπουμε κατευθείαν ότι η υπόθεσή μας είναι σωστή. Όσο αυξάνεται ο μέσος όρος ζωής, το GNI επίσης αυξάνεται. Αυτό φυσικά μπορεί να συνδεθεί και με το γεγονός ότι «πιο πλούσιες χώρες έχουν πιο καλή ποιότητα ζωής». Μια ακόμη παρατήρηση είναι ότι ο μέσος ορός ζωής από το 2000 έως το 2018 έχει σημαντικά αυξηθεί, πράγμα πιθανώς καλό για το μέλλον της ανθρωπότητας.