

Αλέξιος Φιλιππακόπουλος

p3190212

Βασίλειος Δημόπουλος

t8190038

Data Warehousing and Data Mining

Για τις ανάγκες της συγκεκριμένης εργασίας αποφασίσαμε να χρησιμοποιήσουμε τα παρακάτω δύο datasets:

1) University Enrollment Dataset (Πηγή: [kaggle](#))

Το dataset αυτό συγκεντρώνει δεδομένα εγγραφών σε επίπεδο πανεπιστημιακών ιδρυμάτων από το 1950 έως και το 2020. Επιπλέον, περιλαμβάνει πάνω από 17.000 ινστιτούτα σε πάνω από 180 χώρες. Τέλος, εκτός από τις εγγραφές, περιλαμβάνεται και ένα πλήθος άλλων χρήσιμων μεταβλητών που σχετίζονται με τα χαρακτηριστικά του κάθε ιδρύματος όπως περιγράφονται παρακάτω:

- 1) country: Name of country.
- 2) countrycode: Three-letter code outlined by ISO 3166.
- 3) region: Region in which the institution is currently situated.
- 4) incomegroup: Income group, merged from the World Bank.
- 5) iau_id: Unique identifier for each institution from IAU's World Higher Education.
- 6) iau_id1: Unique identifier for each institution that was once separated but later merged.
- 7) eng_name: Current name of university in English.

- 8) orig_name: Current name of university in Latin characters, as reported to IAU-WHED.
- 9) foundedyr: Year in which each institution was founded/established.
- 10) yrclosed: Year university closed, if closed and year known.
- 11) private01: Binary variable for private = 1, or public = 0 institution.
- 12) coordinates: GPS coordinates for each institution.
- 13) latitude: GPS coordinate representing the institution's latitude.
- 14) longitude: GPS coordinate representing the institution's longitude.
- 15) phd_granting: Binary variable for the offering of a PhD degree.
- 16) m_granting: Binary variable for the offering of a Masters degree.
- 17) b_granting: Binary variable for the offering of a Bachelor's degree.
- 18) divisions: Number of different faculties offered by each institution.
- 19) total_fields: Number of degree programs offered by each institution.
- 20) unique_fields: Number of unique degree programs offered by each institution.
- 21) specialized: Binary variable if the institution has a list of words in its name: Arts, Theater, Music, Drama, Institute, or if it has less than three unique programs.
- 22) merger: Binary variable for if the university today the result of a merger in the past between multiple/other institutions is.
- 23) noiau: Binary variable (0/1) to identify if the university does not have an IAU ID on WHED.
- 24) year: Calendar year.

- 25) students5_interpolated : Raw or interpolated enrollment data.
- 26) students5_extrapolated: Extrapolated enrollment data.
- 27) students5_estimated: Raw, interpolated, or extrapolated values, fills in an estimated value for institutions that never reported any enrollments

2) Population by Country – 2020: (Πηγή: [kaggle](#))

Το dataset αυτό περιλαμβάνει πάνω από 200 χώρες καθώς και διάφορα στατιστικά στοιχεία που αφορούν τον πληθυσμό της κάθε μίας. Απαρτίζεται από 11 στήλες όπως περιγράφονται παρακάτω:

- 1) Country (or dependency): Name of country.
- 2) Population (2020): Population per country as of 2020.
- 3) Yearly Change: Population changes on a yearly level.
- 4) Net Change: Net change of the population.
- 5) Density: Density of the population.
- 6) Land Area: Land area in terms of square kilometers.
- 7) Migrants: Number of migrants.
- 8) Fert. Rate: Fertility/Growth rate of population.
- 9) Med. Age: Median/Average lifespan in the country.
- 10) Urban Pop %: Percentage of population in urban cities.
- 11) World Share: Population's contribution to the world's share

Data Manipulation

Παρατηρώντας τα datasets παρατηρήσαμε κάποιες ανωμαλίες στα δεδομένα τις οποίες και εξαλείψαμε χρησιμοποιώντας την Python ως ένα data manipulation step πριν την εισαγωγή των δεδομένων στο Data Warehouse.

Ξεκινάμε με το data manipulation του university dataset.

Αρχικά διαβάζουμε τα δεδομένα μας.

```
In [9]: df = pd.read_excel('enroll.xlsx')
df
```

Out[9]:

	Unnamed: 0	country	countrycode	region	incomegroup	iau_id	iau_id1	eng_name	orig_name	foundedyr	...	divisions	total_fields	unique_
0	0	afghanistan	AFG	South Asia	Low income	IAU-000810	IAU-000810-1	Alberoni University	Alberoni University	1998	...	NaN	NaN	
1	1	afghanistan	AFG	South Asia	Low income	IAU-000810	IAU-000810-1	Alberoni University	Alberoni University	1998	...	NaN	NaN	
2	2	afghanistan	AFG	South Asia	Low income	IAU-000810	IAU-000810-1	Alberoni University	Alberoni University	1998	...	NaN	NaN	
3	3	afghanistan	AFG	South Asia	Low income	IAU-000810	IAU-000810-1	Alberoni University	Alberoni University	1998	...	NaN	NaN	
4	4	afghanistan	AFG	South Asia	Low income	IAU-000810	IAU-000810-1	Alberoni University	Alberoni University	1998	...	9.0	22.0	
...
138347	161555	zimbabwe	ZWE	Sub-Saharan Africa	Lower middle income	IAU-021853	IAU-021853-1	Zimbabwe Open University	(ZOU)	1993	...	NaN	NaN	
138348	161556	zimbabwe	ZWE	Sub-Saharan Africa	Lower middle income	IAU-021853	IAU-021853-1	Zimbabwe Open University	(ZOU)	1993	...	NaN	NaN	
138349	161557	zimbabwe	ZWE	Sub-Saharan Africa	Lower middle income	IAU-021853	IAU-021853-1	Zimbabwe Open University	(ZOU)	1993	...	6.0	40.0	
138350	161558	zimbabwe	ZWE	Sub-Saharan Africa	Lower middle income	IAU-024536	IAU-024536-1	Zimbabwe Ezekiel Gut University	(ZEGU)	2012	...	NaN	NaN	
138351	161559	zimbabwe	ZWE	Sub-Saharan Africa	Lower middle income	IAU-024536	IAU-024536-1	Zimbabwe Ezekiel Gut University	(ZEGU)	2012	...	4.0	25.0	

Εφόσον οι στήλες `eng_name`, `year`, `coordinates` και `country` θα αποτελέσουν `dimensions` πρέπει να ελέγξουμε για τυχόν `NaN` τιμές.

```
In [26]: df['eng_name'].isna().sum(), df['year'].isna().sum(), df['coordinates'].isna().sum(), df['country'].isna().sum()
Out[26]: (0, 0, 0, 0)
```

Βλέπουμε πως τέτοιες τιμές δεν υπάρχουν.

Ωστόσο παρατηρούμε πως όσον αφορά τις στήλες `phd_granting`, `m_granting` και `b_granting`, τα δεδομένα είναι καταχωρημένα μόνο στην τελευταία εγγραφή κάθε πανεπιστημίου.

```
In [28]: df[df['eng_name'] == 'American University Of Afghanistan']['phd_granting', 'm_granting', 'b_granting']
Out[28]:
```

	phd_granting	m_granting	b_granting
5	0	NaN	NaN
6	0	NaN	NaN
7	0	1.0	1.0

Με σκοπό να γεμίσουμε και τις υπόλοιπες εγγραφές δημιουργούμε μια λίστα `unis` που περιέχει κάθε πανεπιστήμιο που εμπεριέχεται στο `dataset`.

```
In [27]: unis = list(df['eng_name'].unique())
unis
['Herat University',
'Jawzjan University',
'Shaheed Rabbani Education University',
'Kabiver University Of Medical Sciences - Abu Ali Ibn Sina',
'Kabul Polytechnic University',
'Kandahar University',
'Carney University',
'Kateb University',
'Khatam Al-Nabieen University',
'Khurasan University',
'Kunduz University',
'Maiwand Institute Of Higher Education',
'Maryam University',
'The Other University',
'Nangarhar University',
'Paktia University',
'Parwan University',
'Kabul University',
'Samangan Institute Of Higher Education',
'Shaikh Zayed University - Khost']
```

Έπειτα αποθηκεύουμε τα indexes που αντιστοιχούν σε κάθε πανεπιστήμιο σε μια λίστα indexes και για κάθε πανεπιστήμιο γεμίζουμε τις εγγραφές των τριών αυτών στηλών, phd_granting, m_granting και b_granting σύμφωνα με την εγγραφή του τελευταίου index.

```
In [29]: for uni in unis:
          indexes = df[df['eng_name'] == uni].index
          df.loc[indexes[:], 'phd_granting'] = df.loc[indexes[-1], 'phd_granting']
          df.loc[indexes[:], 'b_granting'] = df.loc[indexes[-1], 'b_granting']
          df.loc[indexes[:], 'm_granting'] = df.loc[indexes[-1], 'm_granting']
```

Έτσι πετυχαίνουμε τον στόχο μας.

```
In [30]: df[df['eng_name'] == 'American University Of Afghanistan'][['phd_granting', 'm_granting', 'b_granting']]
Out[30]:
```

	phd_granting	m_granting	b_granting
5	0	1.0	1.0
6	0	1.0	1.0
7	0	1.0	1.0

Επόμενο βήμα είναι να ελέγξουμε για τυχόν NaN τιμές στις στήλες αυτές.

```
In [31]: df['phd_granting'].isna().sum(), df['b_granting'].isna().sum(), df['m_granting'].isna().sum()
Out[31]: (0, 3264, 3264)
```

Παρατηρούμε ότι οι στήλες b_granting και m_granting έχουν τον ίδιο αριθμό NaN τιμών, κάτι που ίσως υποδηλώνει ότι αναφερόμαστε στα ίδια indexes. Για να το ελέγξουμε αυτό αποθηκεύουμε τα indexes των NaN τιμών κάθε στήλης σε μία λίστα (bachelor_nans και master_nans).

```
In [32]: bachelor_nans = list(df[df['b_granting'].isna()].index)
bachelor_nans
```

```
7893,
7894,
7895,
7921,
7922,
7923,
7924,
7986,
7987,
7988,
7989,
7990,
7991,
7992,
7993,
7994,
7995,
7996,
7997,
8002.
```

```
In [33]: master_nans = list(df[df['m_granting'].isna()].index)
master_nans
```

```
Out[33]: [2799,
2800,
2801,
2802,
6157,
6158,
6159,
6160,
6161,
6162,
6163,
6164,
6165,
6166,
6167,
6168,
6169,
6170,
6171,
....]
```

Κάνουμε έλεγχο των στοιχείων της κάθε λίστας και πράγματι αναφερόμαστε στα ίδια indexes.

```
In [34]: if all(item in bachelor_nans for item in master_nans):
print("The lists have the same elements")
else:
print("The lists do not have the same elements")
```

```
The lists have the same elements
```

Επομένως αποφασίζουμε να γεμίσουμε τις εγγραφές αυτές με 0.

```
In [35]: df.loc[bachelor_nans[:], 'b_granting'] = 0
df.loc[bachelor_nans[:], 'm_granting'] = 0
```

Τέλος, κάνουμε export το αρχείο σε μορφή xlsx.

```
In [36]: df['phd_granting'].isna().sum(), df['b_granting'].isna().sum(), df['m_granting'].isna().sum()
Out[36]: (0, 0, 0)
```

Υστερα έχουμε το data manipulation του population dataset.

Ομοίως ξεκινάμε με το να διαβάσουμε τα δεδομένα.

```
In [71]: import pandas as pd

In [72]: pop = pd.read_csv('population.csv')
pop
```

Out[72]:

	Country (or dependency)	Population (2020)	Yearly Change	Net Change	Density (P/Km²)	Land Area (Km²)	Migrants (net)	Fert. Rate	Med. Age	Urban Pop %	World Share
0	China	1440297825	0.39 %	5540090	153	9388211	-348399.0	1.7	38	61 %	18.47 %
1	India	1382345085	0.99 %	13586631	464	2973190	-532687.0	2.2	28	35 %	17.70 %
2	United States	331341050	0.59 %	1937734	36	9147420	954806.0	1.8	38	83 %	4.25 %
3	Indonesia	274021604	1.07 %	2898047	151	1811570	-98955.0	2.3	30	56 %	3.51 %
4	Pakistan	221612785	2.00 %	4327022	287	770880	-233379.0	3.6	23	35 %	2.83 %
...
230	Montserrat	4993	0.06 %	3	50	100	NaN	N.A.	N.A.	10 %	0.00 %
231	Falkland Islands	3497	3.05 %	103	0	12170	NaN	N.A.	N.A.	66 %	0.00 %
232	Niue	1628	0.68 %	11	6	260	NaN	N.A.	N.A.	46 %	0.00 %
233	Tokelau	1360	1.27 %	17	136	10	NaN	N.A.	N.A.	0 %	0.00 %
234	Holy See	801	0.25 %	2	2003	0	NaN	N.A.	N.A.	N.A.	0.00 %

235 rows × 11 columns

```
In [73]: df = pd.read_excel('enroll.xlsx')
df
```

Out[73]:

	Unnamed: 0	country	countrycode	region	incomegroup	iau_id	iau_id1	eng_name	orig_name	foundedyr	...	divisions	total_fields	unique_
0	0	afghanistan	AFG	South Asia	Low income	IAU-000810	IAU-000810-1	Alberoni University	Alberoni University	1998	...	NaN	NaN	
1	1	afghanistan	AFG	South Asia	Low income	IAU-000810	IAU-000810-1	Alberoni University	Alberoni University	1998	...	NaN	NaN	
2	2	afghanistan	AFG	South Asia	Low income	IAU-000810	IAU-000810-1	Alberoni University	Alberoni University	1998	...	NaN	NaN	
3	3	afghanistan	AFG	South Asia	Low income	IAU-000810	IAU-000810-1	Alberoni University	Alberoni University	1998	...	NaN	NaN	
4	4	afghanistan	AFG	South Asia	Low income	IAU-000810	IAU-000810-1	Alberoni University	Alberoni University	1998	...	9.0	22.0	
...
138347	161555	zimbabwe	ZWE	Sub-Saharan Africa	Lower middle income	IAU-021853	IAU-021853-1	Zimbabwe Open University	(ZOU)	1993	...	NaN	NaN	
138348	161556	zimbabwe	ZWE	Sub-Saharan Africa	Lower middle income	IAU-021853	IAU-021853-1	Zimbabwe Open University	(ZOU)	1993	...	NaN	NaN	
138349	161557	zimbabwe	ZWE	Sub-Saharan Africa	Lower middle income	IAU-021853	IAU-021853-1	Zimbabwe Open University	(ZOU)	1993	...	6.0	40.0	
138350	161558	zimbabwe	ZWE	Sub-Saharan Africa	Lower middle income	IAU-024536	IAU-024536-1	Zimbabwe Ezekiel Gut University	(ZEGU)	2012	...	NaN	NaN	

Παρατηρούμε ότι οι χώρες στο population dataset ξεκινάνε με κεφαλαίο γράμμα ενώ στο university dataset όχι. Αυτό θα είναι ένα πρόβλημα που πρέπει να λάβουμε υπ' όψη.

Επόμενο βήμα είναι να δούμε αν όλες οι χώρες του university dataset εμπεριέχονται στο population dataset.

Για να το επιτύχουμε αυτό αποθηκεύουμε σε δύο λίστες country_pop και country_enroll τις διαφορετικές χώρες που περιέχει κάθε dataset. Βλέπουμε ότι το population dataset έχει 235 χώρες ενώ το university dataset 194.

Στην συνέχεια κάνουμε την σύγκριση μετατρέποντας τα γράμματα σε πεζά και βλέπουμε ότι υπάρχουν 167 κοινές χώρες και 68 μη κοινές.

```
In [74]: countries_pop = list(pop['country (or dependency)'].unique())
          len(countries_pop)

Out[74]: 235

In [75]: countries_enroll = list(df['country'].unique())
          len(countries_enroll)

Out[75]: 194

In [76]: common_elements = set([c.lower() for c in countries_pop]).intersection([c.lower() for c in countries_enroll])
          len(common_elements)

Out[76]: 167

In [77]: non_common_elements = set([c.lower() for c in countries_pop]).difference([c.lower() for c in countries_enroll])
          len(non_common_elements)

Out[77]: 68
```

Έπειτα από διεξοδική ανάλυση των δεδομένων βρήκαμε ότι οι παρακάτω χώρες εμπεριέχονται και στα δύο datasets αλλά με διαφορετικά ονόματα. Συνεπώς τις μετατρέψαμε στην μορφή που απαντώνται στο dataset των universities.

```
In [78]: for ind in pop.index:
    if pop.loc[ind, 'Country (or dependency)'] == 'Bahamas':
        pop.loc[ind, 'Country (or dependency)'] = 'bahamas, the'

    elif pop.loc[ind, 'Country (or dependency)'] == 'Brunei':
        pop.loc[ind, 'Country (or dependency)'] = 'brunei darussalam'

    elif pop.loc[ind, 'Country (or dependency)'] == 'Hong Kong':
        pop.loc[ind, 'Country (or dependency)'] = 'china - hong kong'

    elif pop.loc[ind, 'Country (or dependency)'] == "Côte d'Ivoire":
        pop.loc[ind, 'Country (or dependency)'] = "cote d'ivoire"

    elif pop.loc[ind, 'Country (or dependency)'] == 'Curaçao':
        pop.loc[ind, 'Country (or dependency)'] = 'curaçao'

    elif pop.loc[ind, 'Country (or dependency)'] == 'Czech Republic (Czechia)':
        pop.loc[ind, 'Country (or dependency)'] = 'czech republic'

    elif pop.loc[ind, 'Country (or dependency)'] == 'Faeroe Islands':
        pop.loc[ind, 'Country (or dependency)'] = 'faroe islands'

    elif pop.loc[ind, 'Country (or dependency)'] == 'Kyrgyzstan':
        pop.loc[ind, 'Country (or dependency)'] = 'kyrgyz republic'

    elif pop.loc[ind, 'Country (or dependency)'] == 'Guinea-Bissau':
        pop.loc[ind, 'Country (or dependency)'] = 'guinea bissau'

    elif pop.loc[ind, 'Country (or dependency)'] == 'Laos':
        pop.loc[ind, 'Country (or dependency)'] = "lao pdr"

    elif pop.loc[ind, 'Country (or dependency)'] == 'Russia':
        pop.loc[ind, 'Country (or dependency)'] = 'russian federation'

    elif pop.loc[ind, 'Country (or dependency)'] == 'Saint Kitts & Nevis':
        pop.loc[ind, 'Country (or dependency)'] = 'saint kitts and nevis'
```

```
elif pop.loc[ind, 'Country (or dependency)'] == 'Sao Tome & Principe':
    pop.loc[ind, 'Country (or dependency)'] = 'sao tome and principe'

elif pop.loc[ind, 'Country (or dependency)'] == 'Slovakia':
    pop.loc[ind, 'Country (or dependency)'] = 'slovak republic'

elif pop.loc[ind, 'Country (or dependency)'] == 'Syria':
    pop.loc[ind, 'Country (or dependency)'] = "syrian arab republic"

elif pop.loc[ind, 'Country (or dependency)'] == 'Gambia':
    pop.loc[ind, 'Country (or dependency)'] = 'the gambia'

elif pop.loc[ind, 'Country (or dependency)'] == 'Timor-Leste':
    pop.loc[ind, 'Country (or dependency)'] = 'timor leste'
```

Στην συνέχεια μετατρέψαμε όλες τις χώρες του population dataset ώστε να είναι γραμμένες με πεζά γράμματα.

```
In [79]: pop['Country (or dependency)'] = pop['Country (or dependency)'].apply(lambda x: x.lower())
```

Επόμενο βήμα ήταν να εντοπίσουμε τις υπόλοιπες χώρες που δεν βρισκόντουσαν στο population dataset και αναζητώντας τον πληθυσμό τους το 2020 στο διαδίκτυο να δημιουργήσουμε εγγραφές στο population dataset.

```
In [80]: newrows = [
    {'Country (or dependency)': 'congo, dem rep', 'Population (2020)': 89560000, 'Yearly Change': None,
     'Net Change': None, 'Density (P/Km²)': None, 'Land Area (Km²)': None, 'Migrants (net)': None,
     'Fert. Rate': None, 'Med. Age': None, 'Urban Pop %': None, 'World Share': None},

    {'Country (or dependency)': 'congo, rep', 'Population (2020)': 5518000, 'Yearly Change': None,
     'Net Change': None, 'Density (P/Km²)': None, 'Land Area (Km²)': None, 'Migrants (net)': None,
     'Fert. Rate': None, 'Med. Age': None, 'Urban Pop %': None, 'World Share': None},

    {'Country (or dependency)': 'france - french guyana', 'Population (2020)': 298682, 'Yearly Change': None,
     'Net Change': None, 'Density (P/Km²)': None, 'Land Area (Km²)': None, 'Migrants (net)': None,
     'Fert. Rate': None, 'Med. Age': None, 'Urban Pop %': None, 'World Share': None},

    {'Country (or dependency)': 'france - french polynesia', 'Population (2020)': 280904, 'Yearly Change': None,
     'Net Change': None, 'Density (P/Km²)': None, 'Land Area (Km²)': None, 'Migrants (net)': None,
     'Fert. Rate': None, 'Med. Age': None, 'Urban Pop %': None, 'World Share': None},

    {'Country (or dependency)': 'france - martinique', 'Population (2020)': 376403, 'Yearly Change': None,
     'Net Change': None, 'Density (P/Km²)': None, 'Land Area (Km²)': None, 'Migrants (net)': None,
     'Fert. Rate': None, 'Med. Age': None, 'Urban Pop %': None, 'World Share': None},

    {'Country (or dependency)': 'france - new caledonia', 'Population (2020)': 271960, 'Yearly Change': None,
     'Net Change': None, 'Density (P/Km²)': None, 'Land Area (Km²)': None, 'Migrants (net)': None,
     'Fert. Rate': None, 'Med. Age': None, 'Urban Pop %': None, 'World Share': None},

    {'Country (or dependency)': 'france - reunion', 'Population (2020)': 895312, 'Yearly Change': None,
     'Net Change': None, 'Density (P/Km²)': None, 'Land Area (Km²)': None, 'Migrants (net)': None,
     'Fert. Rate': None, 'Med. Age': None, 'Urban Pop %': None, 'World Share': None},

    {'Country (or dependency)': 'korea, dem people ♦♦♦s rep', 'Population (2020)': 25780000, 'Yearly Change': None,
     'Net Change': None, 'Density (P/Km²)': None, 'Land Area (Km²)': None, 'Migrants (net)': None,
     'Fert. Rate': None, 'Med. Age': None, 'Urban Pop %': None, 'World Share': None},
]
```

```
    {'Country (or dependency)': 'korea, rep', 'Population (2020)': 51840000, 'Yearly Change': None,
     'Net Change': None, 'Density (P/Km²)': None, 'Land Area (Km²)': None, 'Migrants (net)': None,
     'Fert. Rate': None, 'Med. Age': None, 'Urban Pop %': None, 'World Share': None},

    {'Country (or dependency)': 'palestine', 'Population (2020)': 4800000, 'Yearly Change': None,
     'Net Change': None, 'Density (P/Km²)': None, 'Land Area (Km²)': None, 'Migrants (net)': None,
     'Fert. Rate': None, 'Med. Age': None, 'Urban Pop %': None, 'World Share': None},
]

pop = pop.append(newrows, ignore_index=True)
```

Τέλος ξανακάνοντας τον έλεγχο παρατηρούμε ότι πλέον τα δύο datasets εμπεριέχουν τις ίδιες χώρες.

```
In [81]: countries_pop = list(pop['Country (or dependency)'].unique())
len(countries_pop)

Out[81]: 245

In [82]: countries_enroll = list(df['country'].unique())
len(countries_enroll)

Out[82]: 194

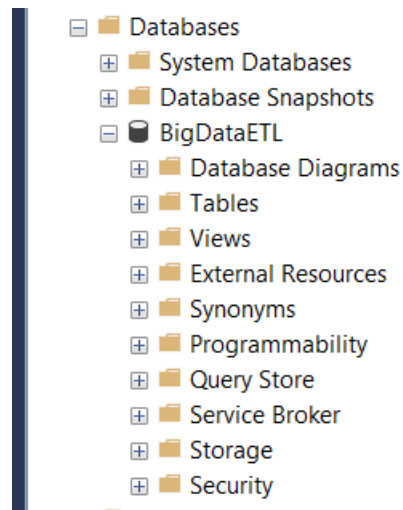
In [83]: common_elements = set(countries_enroll).intersection(countries_pop)
len(common_elements)

Out[83]: 194

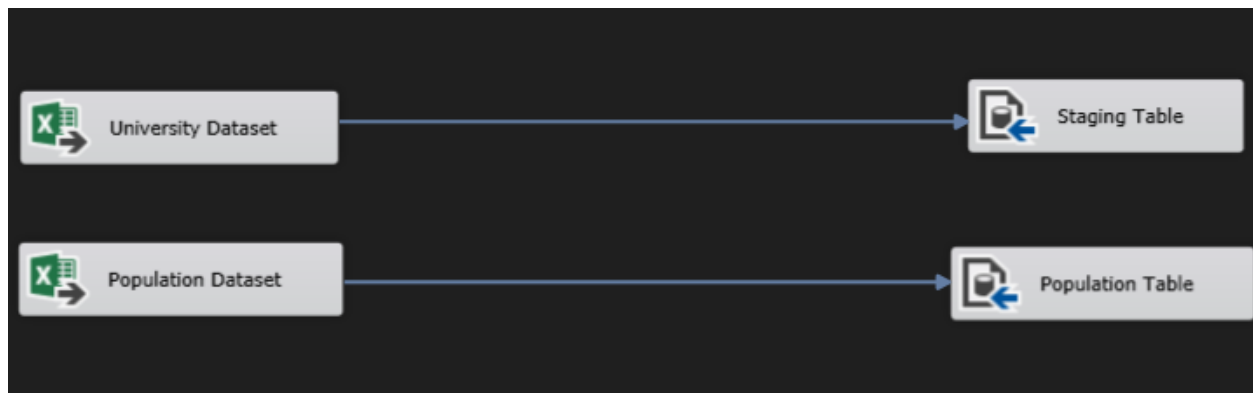
In [84]: pop.to_excel('country_populations.xlsx', index=False)
```

Data Warehousing

Ξεκινήσαμε με την δημιουργία της βάσης δεδομένων στο Microsoft SQL Server Management Studio.



Έπειτα δημιουργήσαμε ένα Data Flow Task όπου έχουμε δύο Excel Sources που εμπεριέχουν τα δύο datasets και δύο SQL Server Destinations με προορισμό την βάση μας. Έτσι δημιουργούμε τον πίνακα staging1 με τα στοιχεία του university dataset και ένα πίνακα country_populations με τα στοιχεία του population dataset.



Specify a connection manager, data source, or data source view, and select the table or the view into which the data is copied. Click New to create a new table or view.

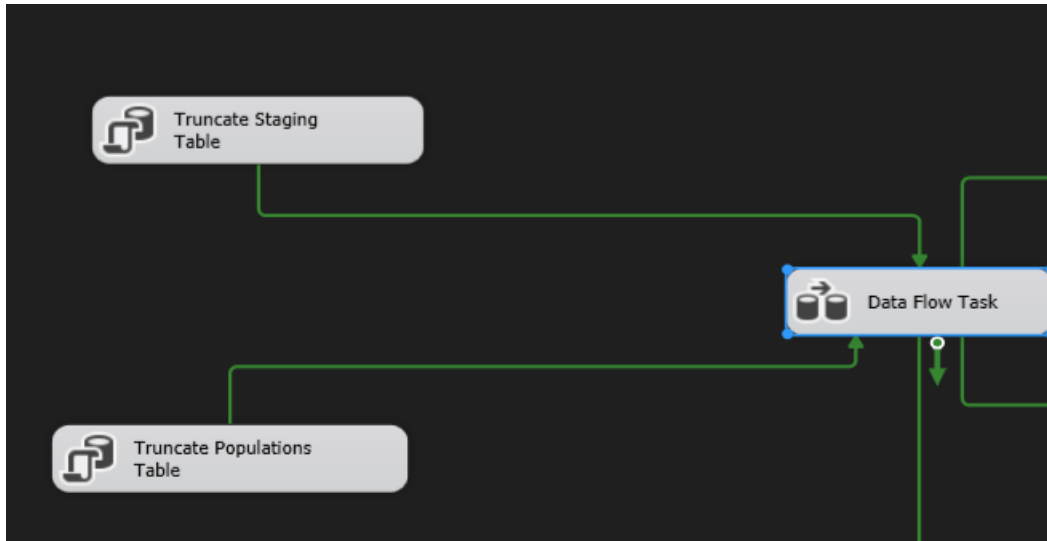
Connection manager:

Use a table or view:

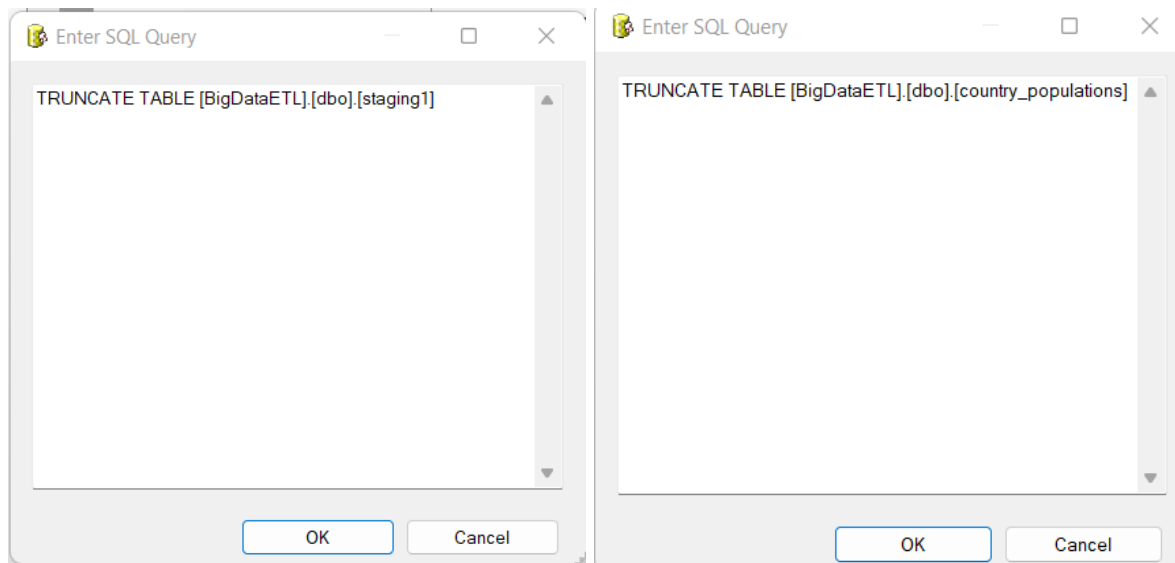
Connection manager:

Use a table or view:

Επόμενο βήμα ήταν η δημιουργία δύο Execute SQL Task τα οποία συνδέσαμε με το παραπάνω Data Flow Task. Σκοπός τους είναι να κάνουν truncate τους δύο πίνακες που δημιουργήσαμε παραπάνω για να αποφεύγεται η επανειλημμένη εισαγωγή στοιχείων σε αυτούς χωρίς την διαγραφή των προηγούμενων εγγραφών.



Τα SQL Queries που περιέχονται είναι τα παρακάτω.



Αμέσως μετά ασχοληθήκαμε με τα dimensions. Δημιουργήσαμε τέσσερα dimensions με βάση τις στήλες year, coordinates, eng_name και country. Αρχικά δημιουργήσαμε έναν πίνακα για κάθε dimension:

1) Year Dimension

	Column Name	Data Type	Allow Nulls
▶	id_year	int	<input type="checkbox"/>
	year	int	<input type="checkbox"/>

Ορίσαμε ως primary key την στήλη id_year η οποία δεν πρέπει να δέχεται null τιμές ενώ είναι integer και αυξάνεται κατά 1 κάθε φορά.

▼ Identity Specification	Yes
(Is Identity)	Yes
Identity Increment	1
Identity Seed	1

Τέλος, ορίσαμε ένα index για την στήλη year δηλώνοντας ότι δέχεται μοναδικές τιμές και πρέπει να κάνει ignore διπλότυπες τιμές. Εφόσον με βάση την στήλη αυτή θα παίρνει τιμές η στήλη id_year.

Indexes/Keys

Selected Primary/Unique Key or Index:


IX_year_dim
PK_year_dim

Editing properties for existing primary/unique key or index.

▼ (General)	
Columns	year (ASC)
Is Unique	Yes
Type	Index
▼ Identity	
(Name)	IX_year_dim
Description	
▼ Table Designer	
Create As Clustered	No
> Data Space Specification	PRIMARY
> Fill Specification	
Ignore Duplicate Keys	Yes

Add Delete Close

2) Geolocation Dim:

	Column Name	Data Type	Allow Nulls
	geolocation_id	int	<input type="checkbox"/>
	coordinates	nvarchar(255)	<input checked="" type="checkbox"/>
	latitude	float	<input checked="" type="checkbox"/>
	longitude	float	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Ορίσαμε ως primary key την στήλη geolocation_id η οποία δεν πρέπει να δέχεται null τιμές ενώ είναι integer και αυξάνεται κατά 1 κάθε φορά.

▼ Identity Specification	Yes
(Is Identity)	Yes
Identity Increment	1
Identity Seed	1

Τέλος, ορίσαμε τρία indexes, ένα για κάθε στήλη, παρόλο που επιτρέπουμε null τιμές ξέρουμε ότι δεν θα δεχτούμε τέτοιες λόγω του data manipulation. Επιπλέον μπορούμε να δηλώσουμε ότι όλες δέχονται μοναδικές τιμές (διότι είναι αδύνατον δύο πανεπιστήμια να μοιράζονται τις ίδιες γεωγραφικές συντεταγμένες). Συνεπώς με βάση τις στήλες αυτές θα παίρνει τιμές η στήλη geolocation_id.

Indexes/Keys

Selected Primary/Unique Key or Index:

- IX_geolocation_dim
- IX_geolocation_dim_1
- IX_geolocation_dim_2
- PK_geolocation_dim

Editing properties for existing primary/unique key or index.

(General)	
Columns	coordinates (ASC)
Is Unique	Yes
Type	Index
Identity	
(Name)	IX_geolocation_dim
Description	
Table Designer	
Create As Clustered	No
Data Space Specification	PRIMARY
Fill Specification	
Ignore Duplicate Keys	Yes

Add Delete Close

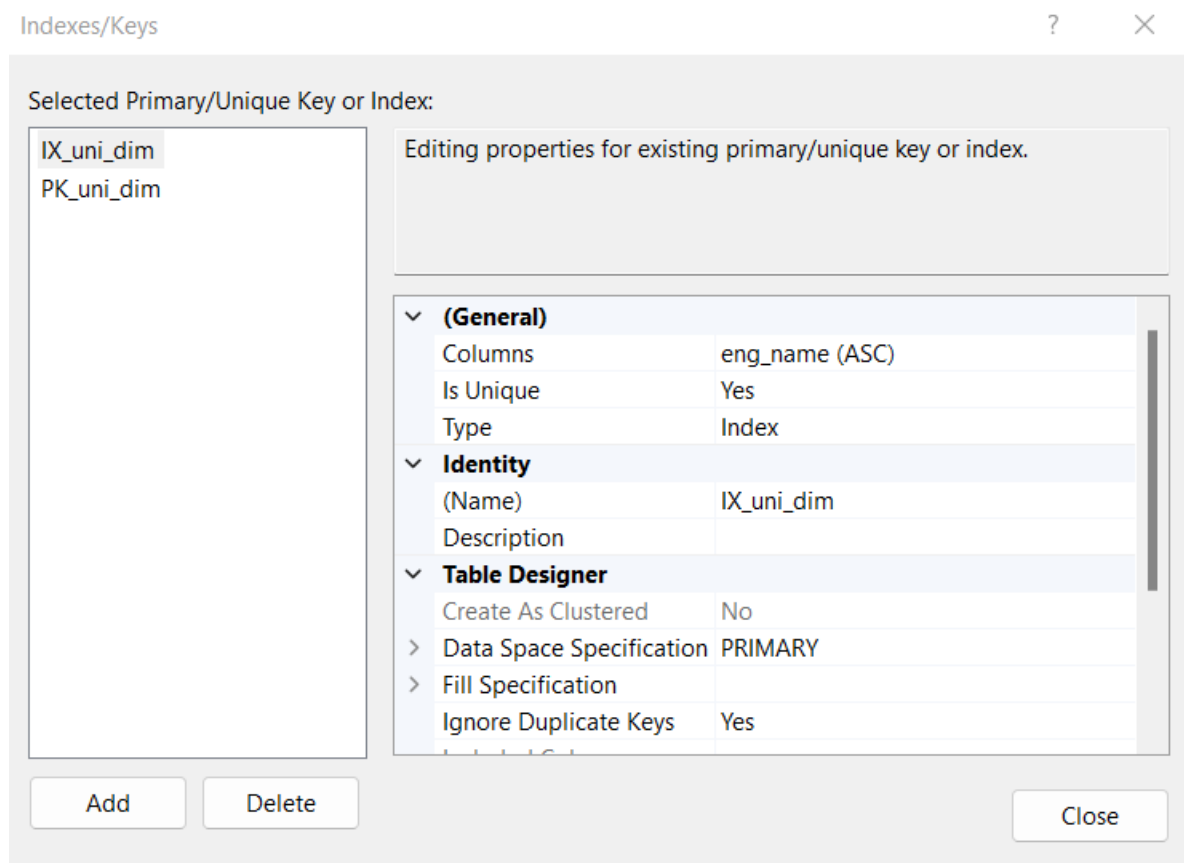
3) University Dimension:

	Column Name	Data Type	Allow Nulls
PK	uni_id	int	<input type="checkbox"/>
	iau_id	nvarchar(255)	<input checked="" type="checkbox"/>
	iau_id1	nvarchar(255)	<input checked="" type="checkbox"/>
	noiau	float	<input checked="" type="checkbox"/>
	eng_name	nvarchar(255)	<input type="checkbox"/>
	orig_name	nvarchar(255)	<input checked="" type="checkbox"/>
	foundedyr	float	<input checked="" type="checkbox"/>
	yrclosed	nvarchar(255)	<input checked="" type="checkbox"/>
	merger	float	<input checked="" type="checkbox"/>
	private01	float	<input checked="" type="checkbox"/>
	phd_granting	float	<input checked="" type="checkbox"/>
	m_granting	float	<input checked="" type="checkbox"/>
	b_granting	float	<input checked="" type="checkbox"/>
	total_fields	float	<input checked="" type="checkbox"/>
	divisions	float	<input checked="" type="checkbox"/>
	unique_fields	float	<input checked="" type="checkbox"/>
	specialized	float	<input checked="" type="checkbox"/>

Ορίσαμε ως primary key την στήλη uni_id η οποία δεν πρέπει να δέχεται null τιμές ενώ είναι integer και αυξάνεται κατά 1 κάθε φορά.

▼ Identity Specification	Yes
(Is Identity)	Yes
Identity Increment	1
Identity Seed	1

Τέλος, ορίσαμε ένα index για την στήλη eng_name δηλώνοντας ότι δέχεται μοναδικές τιμές και πρέπει να κάνει ignore διπλότυπες τιμές. Εφόσον με βάση την στήλη αυτή θα παίρνει τιμές η στήλη uni_id.



4) Country Dimension:

	Column Name	Data Type	Allow Nulls
🔑	country_id	int	<input type="checkbox"/>
	country	nvarchar(255)	<input checked="" type="checkbox"/>
	countrycode	nvarchar(255)	<input checked="" type="checkbox"/>
	region	nvarchar(255)	<input checked="" type="checkbox"/>
	incomegroup	nvarchar(255)	<input checked="" type="checkbox"/>
	population	float	<input checked="" type="checkbox"/>
	median_age	nvarchar(255)	<input checked="" type="checkbox"/>
	urban_pop_percentage	nvarchar(255)	<input checked="" type="checkbox"/>
	world_share	nvarchar(255)	<input checked="" type="checkbox"/>

Ορίσαμε ως primary key την στήλη country_id η οποία δεν πρέπει να δέχεται null τιμές ενώ είναι integer και αυξάνεται κατά 1 κάθε φορά.

▼ Identity Specification	Yes
(Is Identity)	Yes
Identity Increment	1
Identity Seed	1

Τέλος, ορίσαμε ένα index για την στήλη country δηλώνοντας ότι δέχεται μοναδικές τιμές και πρέπει να κάνει ignore διπλότυπες τιμές. Επιπλέον παρόλο που επιτρέπουμε null τιμές ξέρουμε ότι δεν θα δεχτούμε τέτοιες λόγω του data manipulation. Συνεπώς με βάση την στήλη αυτή θα παίρνει τιμές η στήλη uni_id.

Indexes/Keys

Selected Primary/Unique Key or Index:

IX_country_dim
PK_country_dim

Editing properties for existing primary/unique key or index.

▼ (General)

Columns country (ASC)

Is Unique Yes

Type Index

▼ Identity

(Name) IX_country_dim

Description

▼ Table Designer

Create As Clustered No

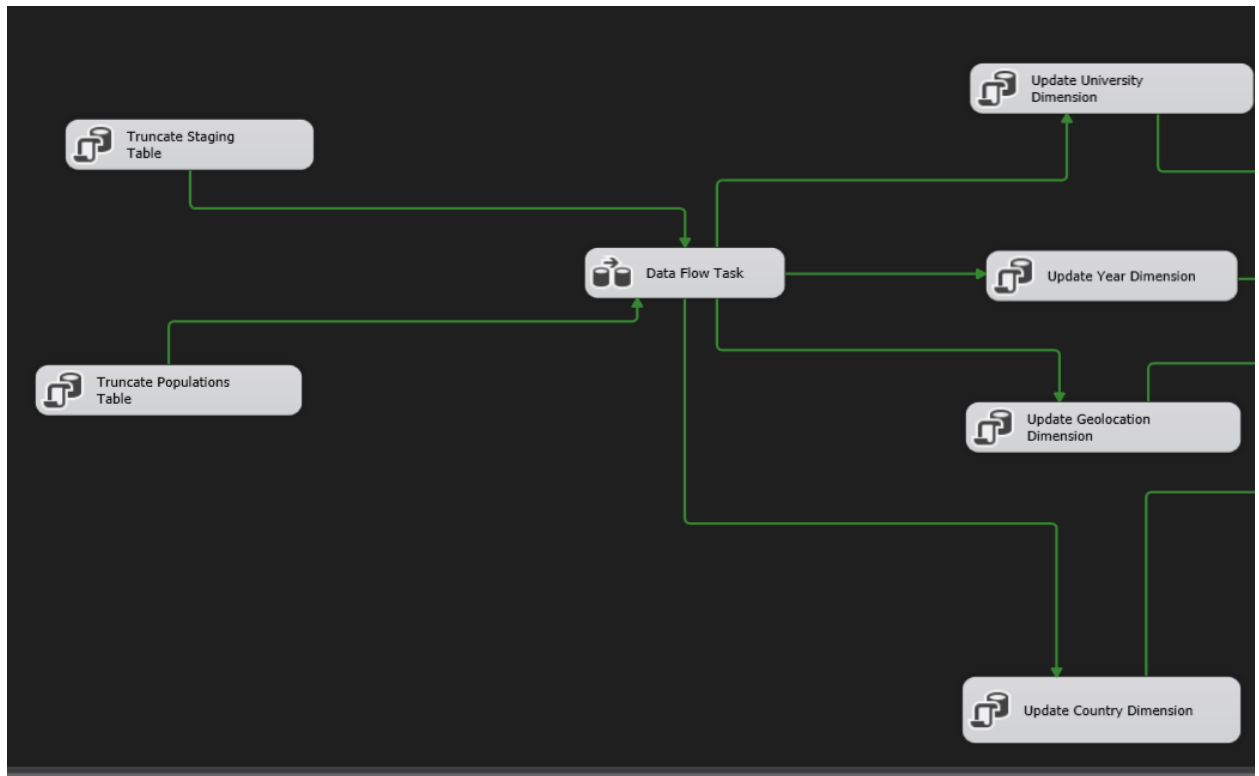
> Data Space Specification PRIMARY

> Fill Specification

Ignore Duplicate Keys Yes

Add Delete Close

Εν συνεχεία δημιουργήσαμε τέσσερα Execute SQL Task, ένα για κάθε dimension, τα οποία και συνδέσαμε με το Data Flow Task που δημιουργήσαμε παραπάνω.



Σκοπός αυτών των tasks είναι η ενημέρωση / γέμισμα των πινάκων που αφορούν τα dimensions.

1) Update University Dimension

Ο παρακάτω κώδικας εισάγει τις ανάλογες τιμές στον πίνακα του University Dimension.

```

INSERT INTO university_dim (iau_id, iau_id1, eng_name, orig_name, foundedyr, yrclosed,
private01, merger, noiau, phd_granting, m_granting, b_granting, divisions, total_fields,
unique_fields, specialized)
SELECT iau_id, iau_id1, eng_name, orig_name, foundedyr, yrclosed, private01, merger, noiau,
phd_granting, m_granting, b_granting, divisions, total_fields, unique_fields, specialized
FROM staging1
  
```

2) Update Year Dimension

Ο παρακάτω κώδικας εισάγει τις ανάλογες τιμές στον πίνακα του Year Dimension.

```
INSERT INTO year_dim (year) SELECT DISTINCT year  
FROM staging1 WHERE year IS NOT NULL
```

3) Update Geolocation Dimension

Ο παρακάτω κώδικας εισάγει τις ανάλογες τιμές στον πίνακα του Geolocation Dimension.

```
INSERT INTO geolocation_dim (coordinates, latitude, longitude)  
SELECT coordinates, latitude, longitude FROM staging1
```

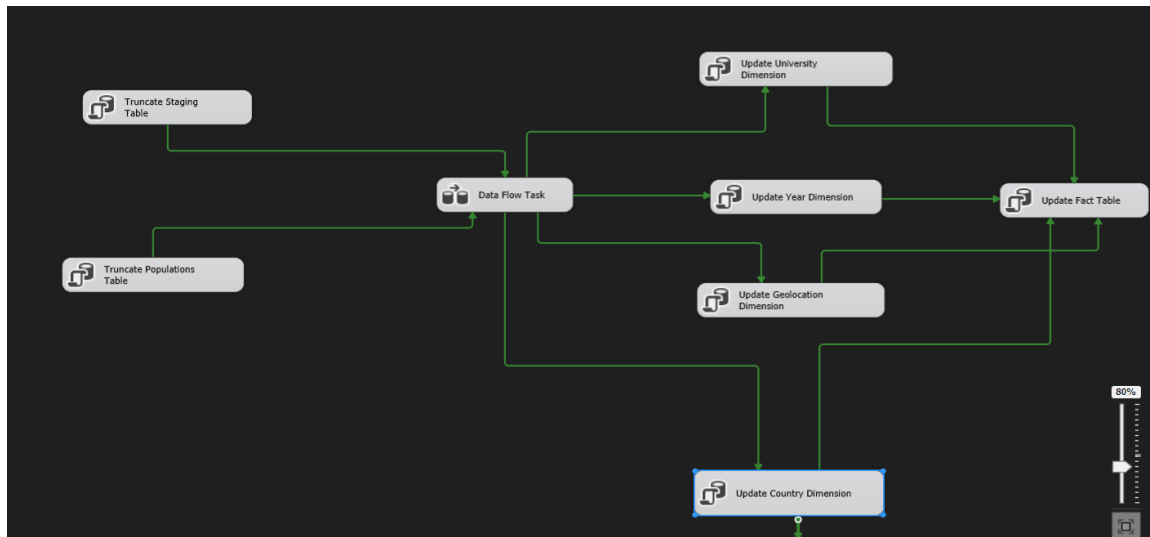
4) Update Country Dimension

Ο παρακάτω κώδικας εισάγει τις ανάλογες τιμές στον πίνακα του Country Dimension.

```
INSERT INTO country_dim(country, countrycode, region,  
incomegroup, population, median_age,  
urban_pop_percentage, world_share)  
SELECT country, countrycode, region, incomegroup,  
population, median_age, urban_pop_percentage, world_share  
FROM staging1  
INNER JOIN country_populations  
ON staging1.country = country_populations.country_name
```

Αφότου γεμίσουν οι πίνακες των dimensions, μένει η δημιουργία του fact table.

Έτσι δημιουργούμε ένα ακόμα Execute SQL Task το οποίο συνδέεται με καθένα από τα Execute SQL Task που αφορούν τα dimensions.

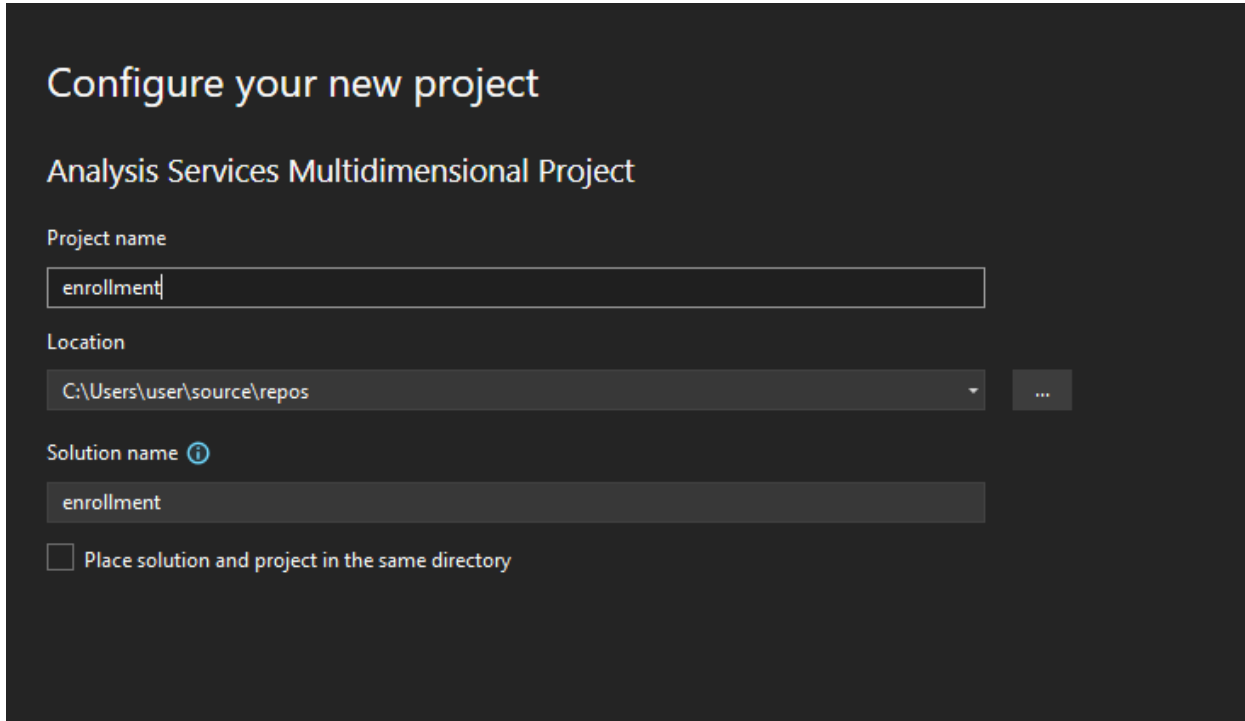


Με το task αυτό δημιουργούμε το fact table χρησιμοποιώντας τον παρακάτω κώδικα και ολοκληρώνοντας το Warehousing Phase της εργασίας.

```
INSERT INTO [BigDataETL].[dbo].[fact](geolocation, country,
year, university, enrollments)
SELECT
[geolocation_dim].[geolocation_id] AS [geolocation],
[country_dim].[country_id] AS [country],
[year_dim].[id_year] AS [year],
[uni_dim].[uni_id] AS [university],
[BigDataETL].[dbo].[staging1].[students5_estimated]
FROM [BigDataETL].[dbo].[staging1]
INNER JOIN [BigDataETL].[dbo].[geolocation_dim] ON
[staging1].[coordinates] = [geolocation_dim].[coordinates]
INNER JOIN [BigDataETL].[dbo].[country_dim] ON [staging1].
[country] = [BigDataETL].[dbo].[country_dim].[country]
INNER JOIN [BigDataETL].[dbo].[year_dim] ON [staging1].
[year] = [year_dim].[year]
INNER JOIN [BigDataETL].[dbo].[uni_dim] ON [staging1].
[eng_name] = [uni_dim].[eng_name];
```

Cubes

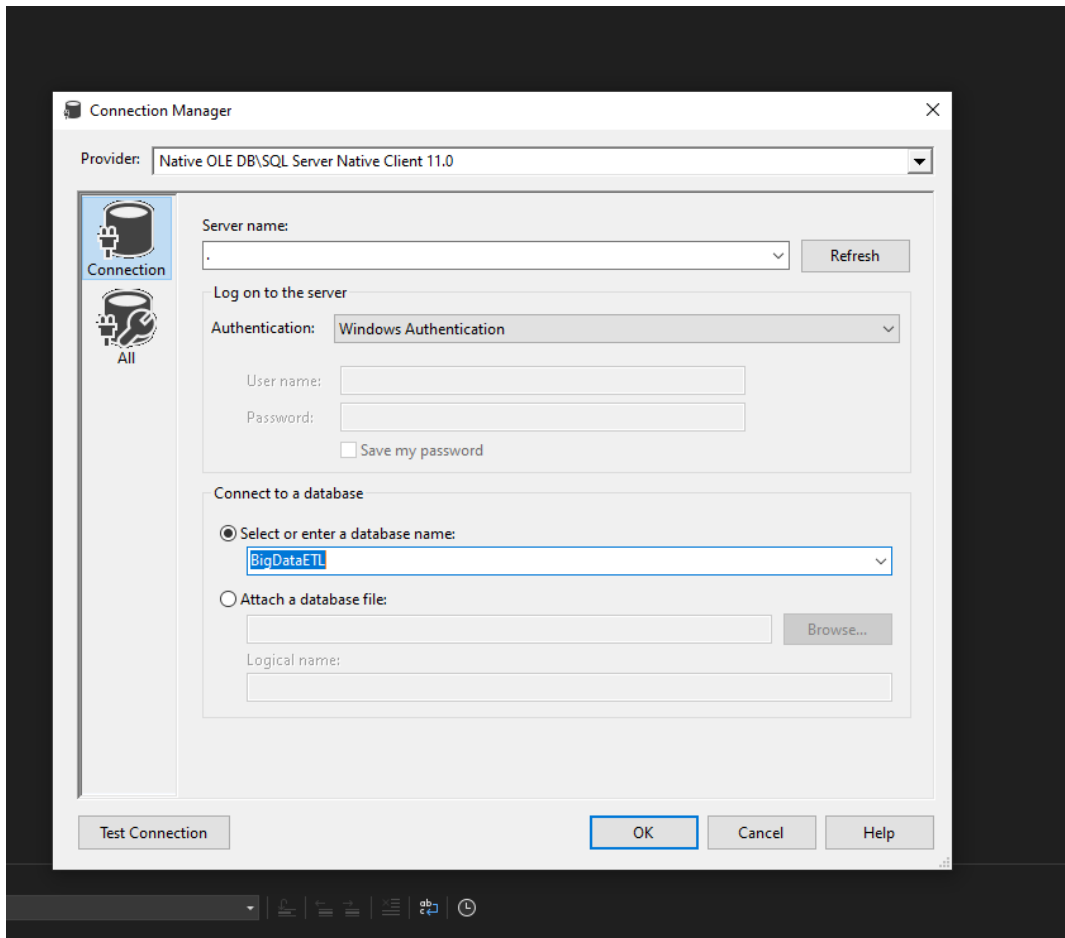
Για να δημιουργήσουμε τον κύβο των δεδομένων μας αρχικά στο Visual Studio θα δημιουργήσουμε ένα Analysis Services Multidimensional Project.



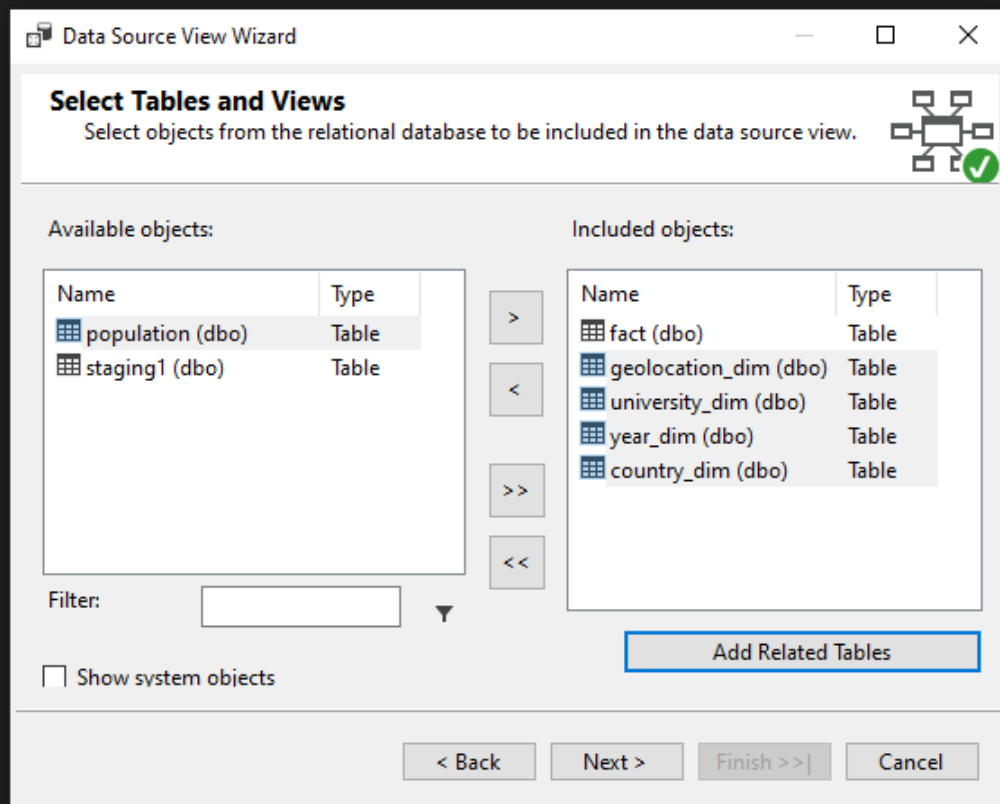
The screenshot shows the 'Configure your new project' dialog box in Visual Studio. The title is 'Configure your new project'. Below it, the project type is 'Analysis Services Multidimensional Project'. There are three input fields: 'Project name' with the value 'enrollment', 'Location' with the value 'C:\Users\user\source\repos', and 'Solution name' with the value 'enrollment'. There is an information icon (i) next to the 'Solution name' label. At the bottom, there is a checkbox labeled 'Place solution and project in the same directory' which is currently unchecked.

Όταν το ανοίξουμε μπορούμε να δούμε στο δεξιό πλαίσιο στο Solution Explorer, τα Data Sources, τα Data Source Views και τα Cubes που θέλουμε να ορίσουμε.

Ξεκινώντας με τα Data Sources πατάμε δεξί κλικ και new Data Source και βάζουμε ως νέα σύνδεση τον local server και ως database το BigDataETL και το αποθηκεύουμε.

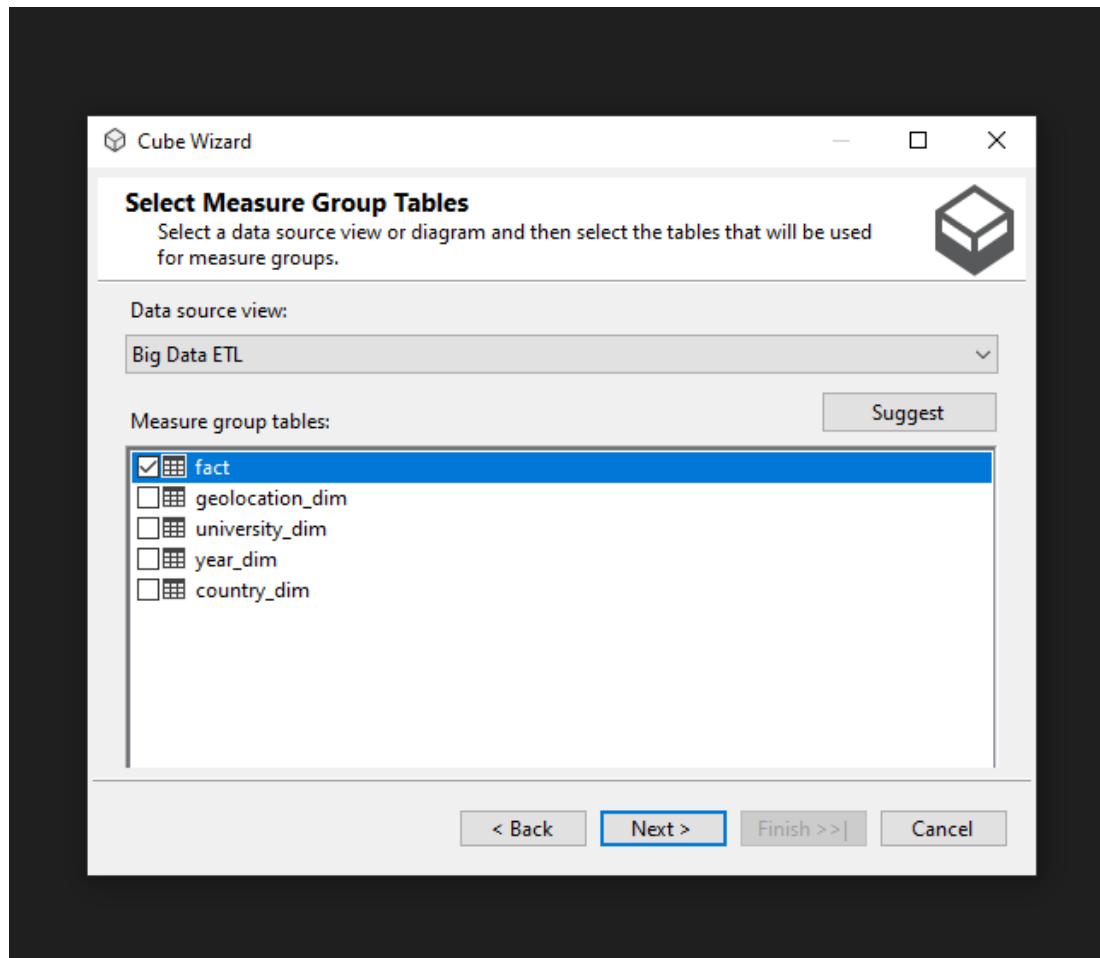


Στην συνέχεια ορίζουμε τα Data Source Views. Πατάμε new Data Source Views, επιλέγουμε στον wizard το data source που ορίσαμε πάνω και μεταφέρουμε το fact table και τα dimension tables και αποθηκεύουμε.

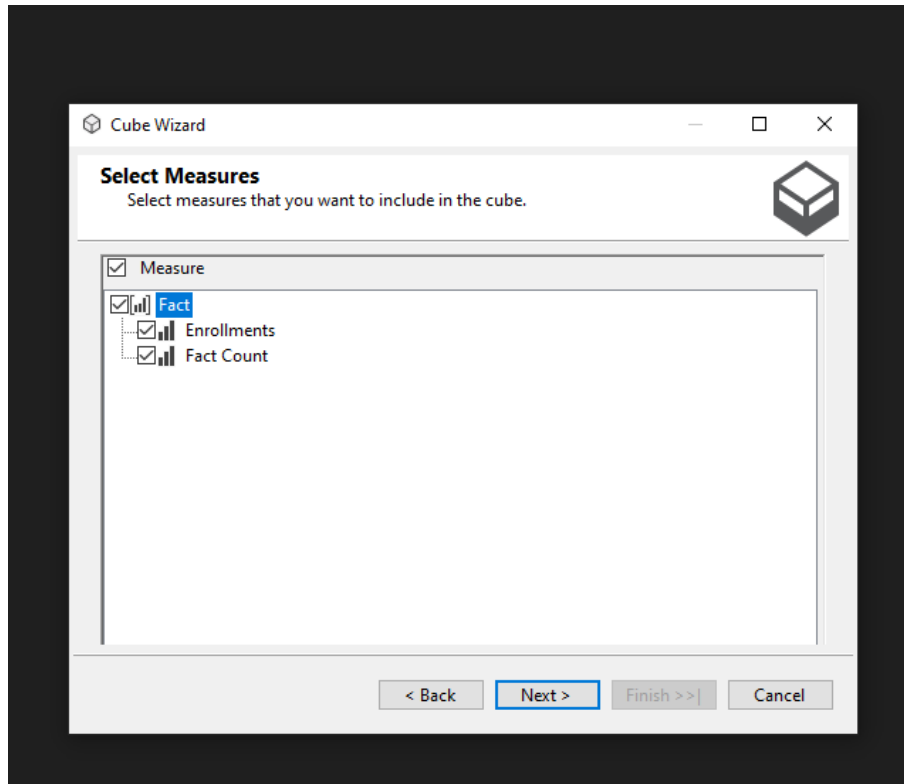


Το επόμενο βήμα είναι η δημιουργία του κύβου. Πατάμε create new cube και μπαίνουμε στον wizard.

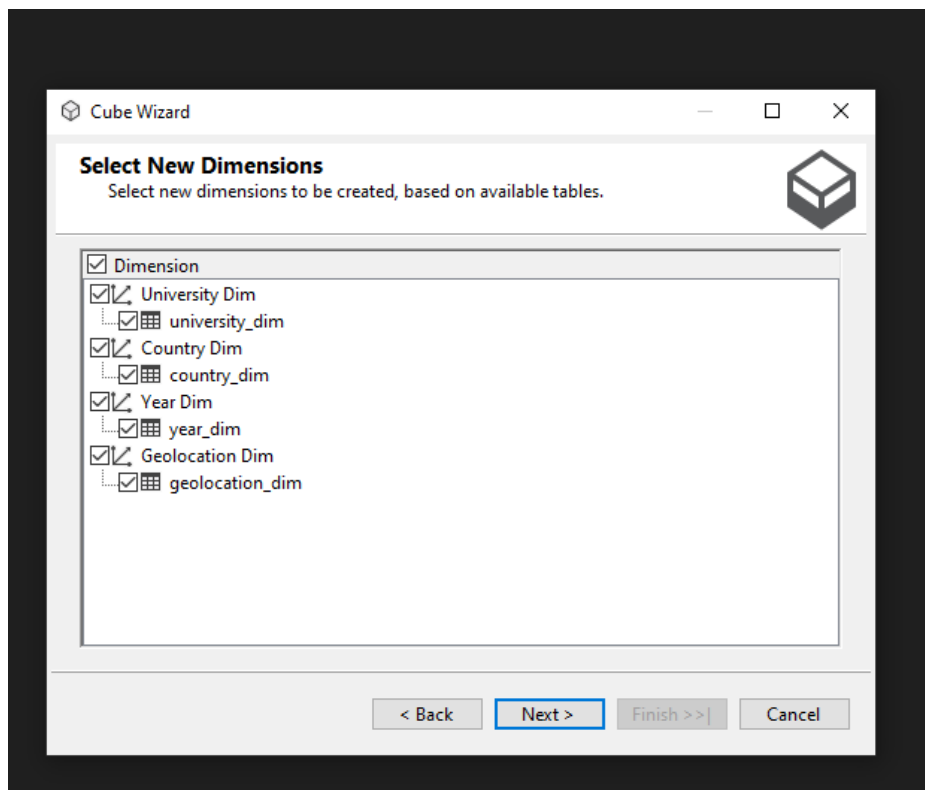
Στην πρώτη σελίδα επιλέγουμε το table που περιέχει τα measures, δηλαδή το fact table.



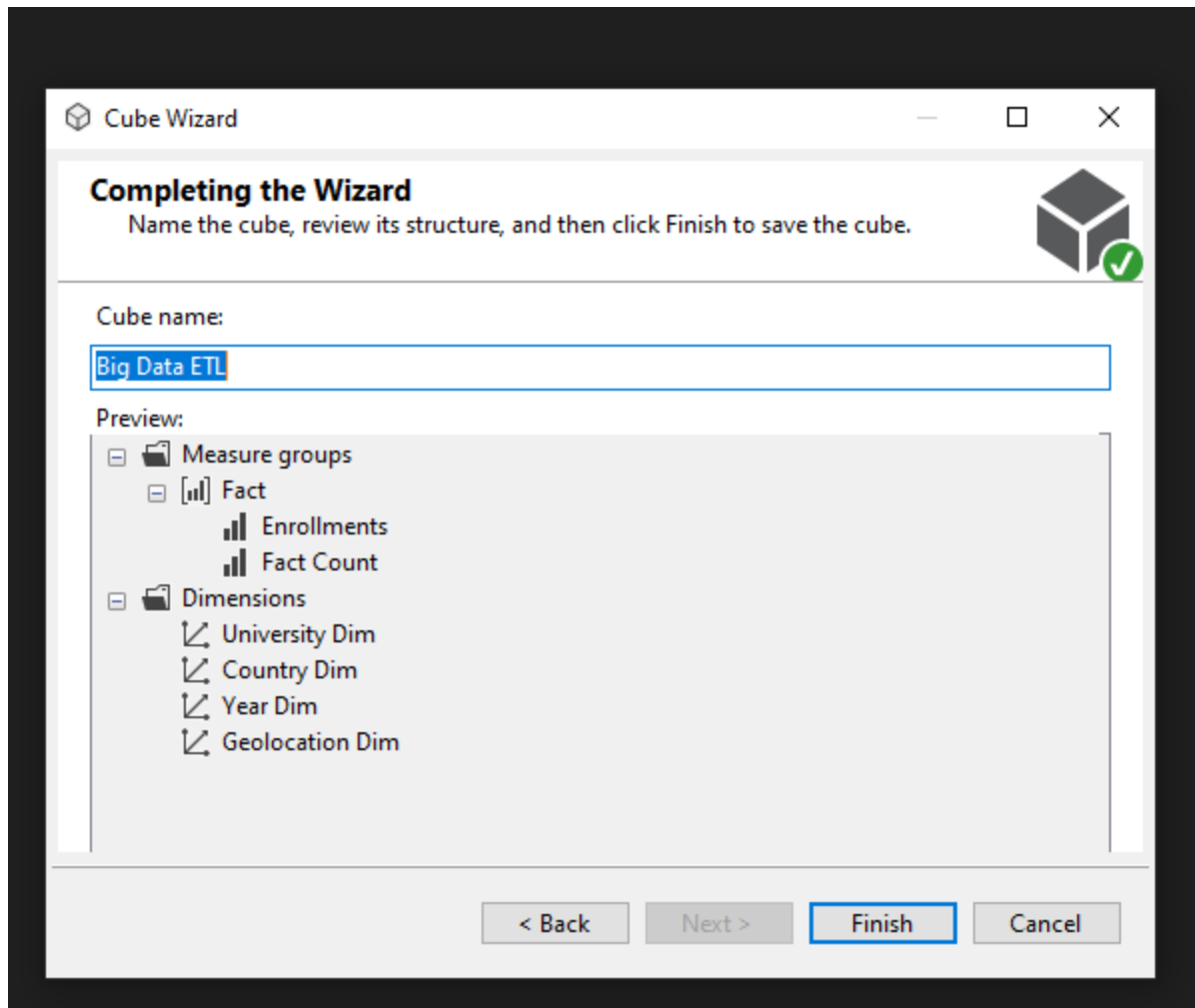
Στην συνέχεια επιλέγουμε τα measures που θέλουμε να μετράει ο κύβος μας, δηλαδή τα enrollments και τα counts.



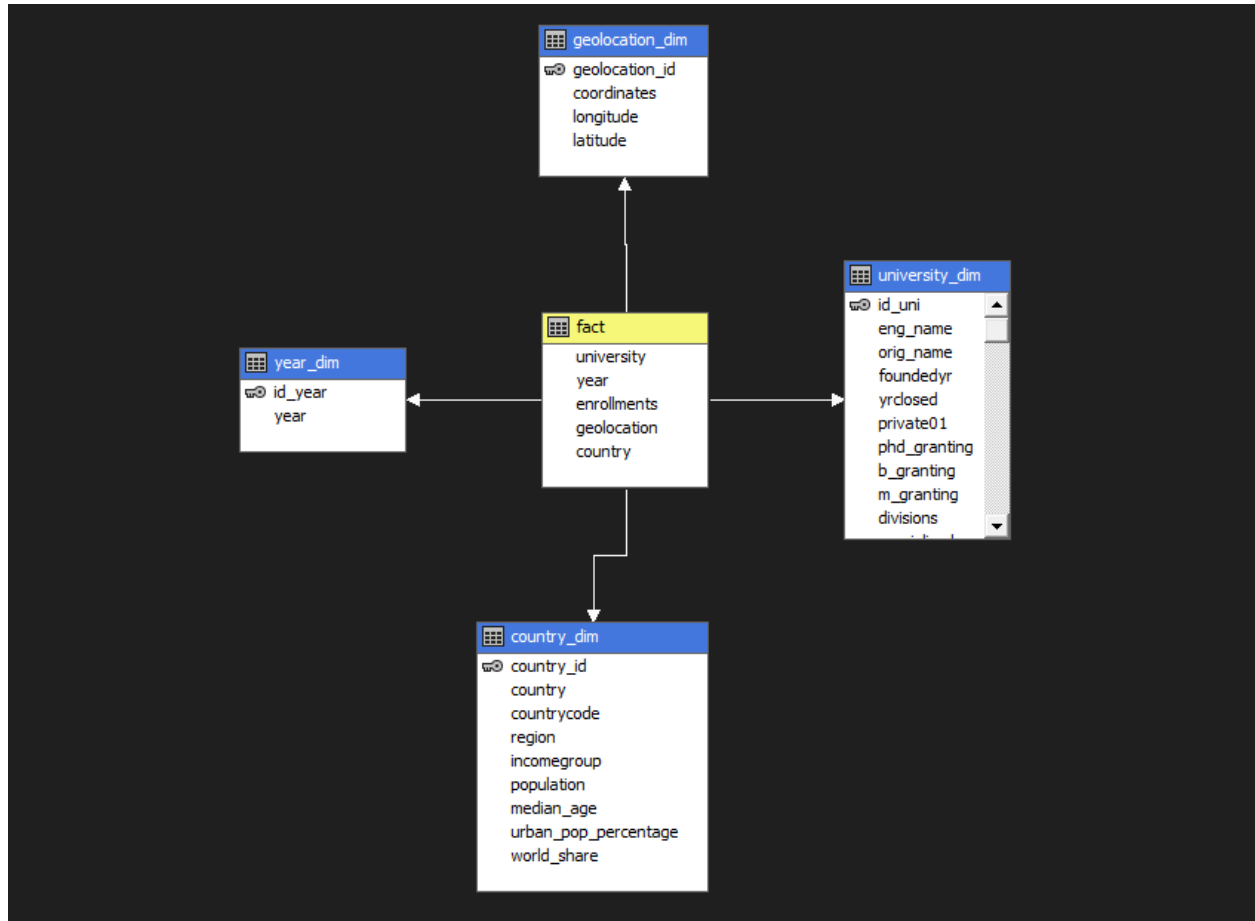
Μετά επιλέγουμε τα dimension table μας.



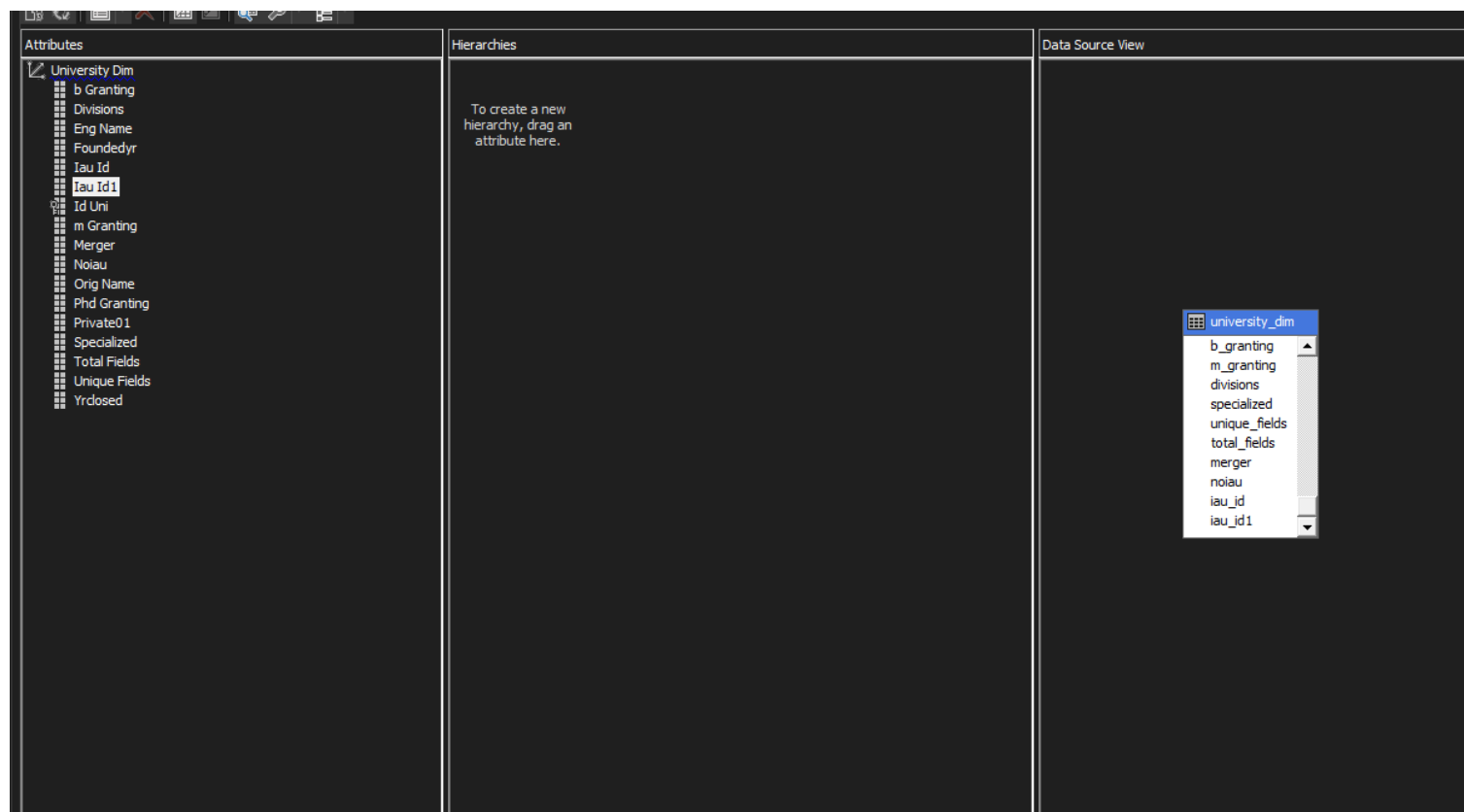
Τέλος αποθηκεύουμε τον κύβο και κλείνουμε τον wizard.



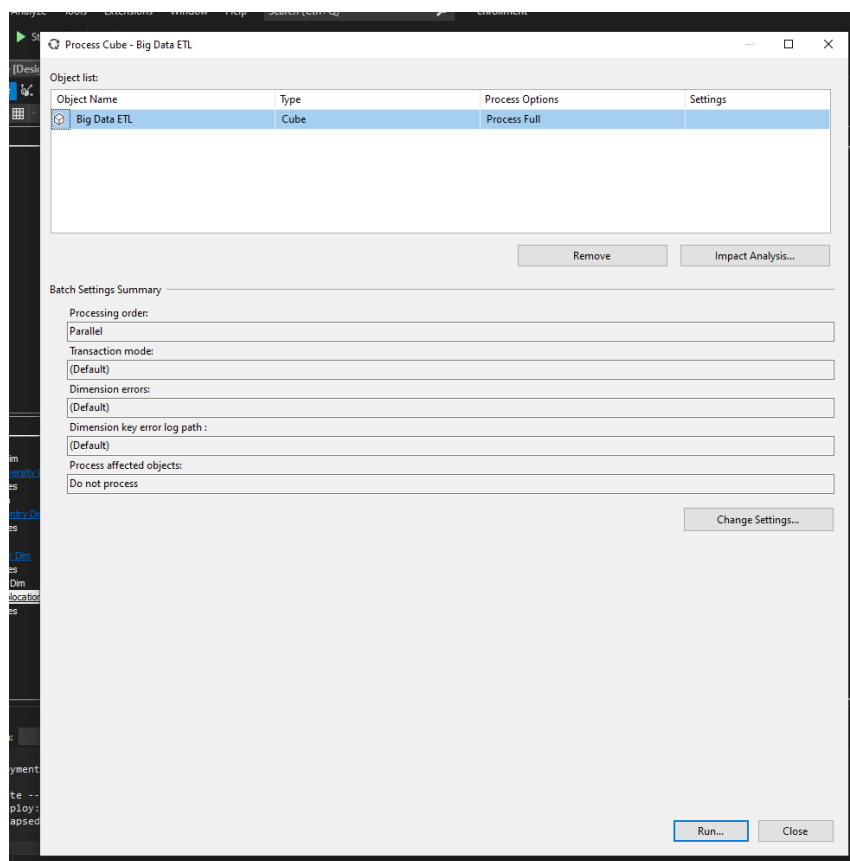
Επιστρέφοντας στο cube structure παρατηρούμε ότι το Visual Studio δημιούργησε το star schema μας.



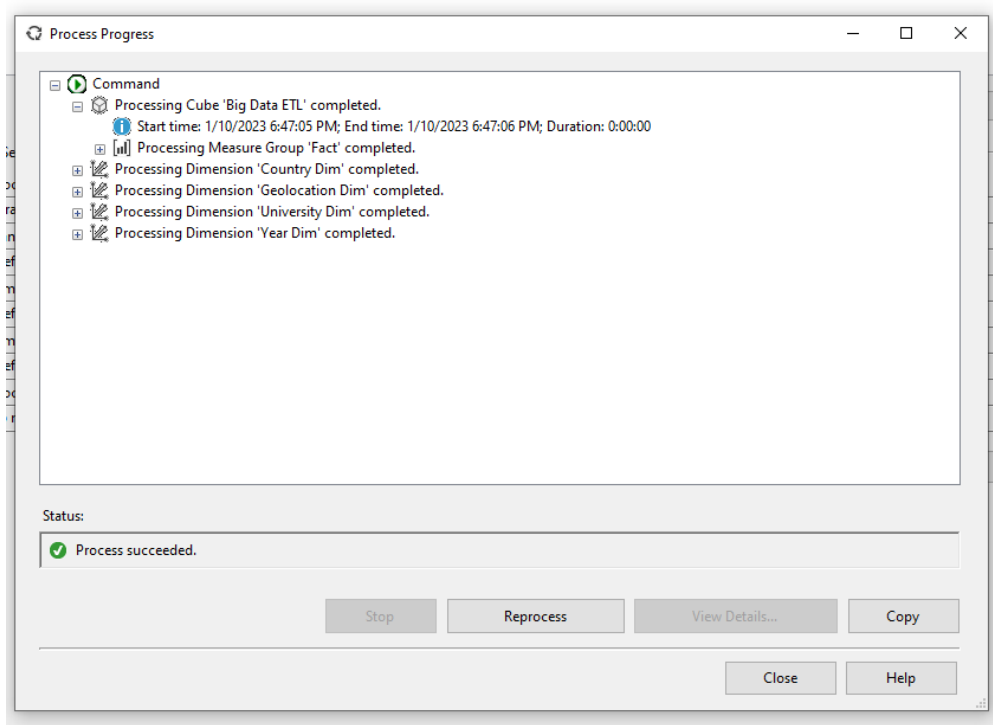
Πριν κάνουμε process τον κύβο πρέπει να προσθέσουμε στα dimensions μας τα υπόλοιπα attributes εκτός από το id του dimension, για να έχουμε πρόσβαση σε αυτά στη συνέχεια στον browser. Αυτό κάνουμε παρακάτω για το university dimension και αντίστοιχα για τα υπόλοιπα.



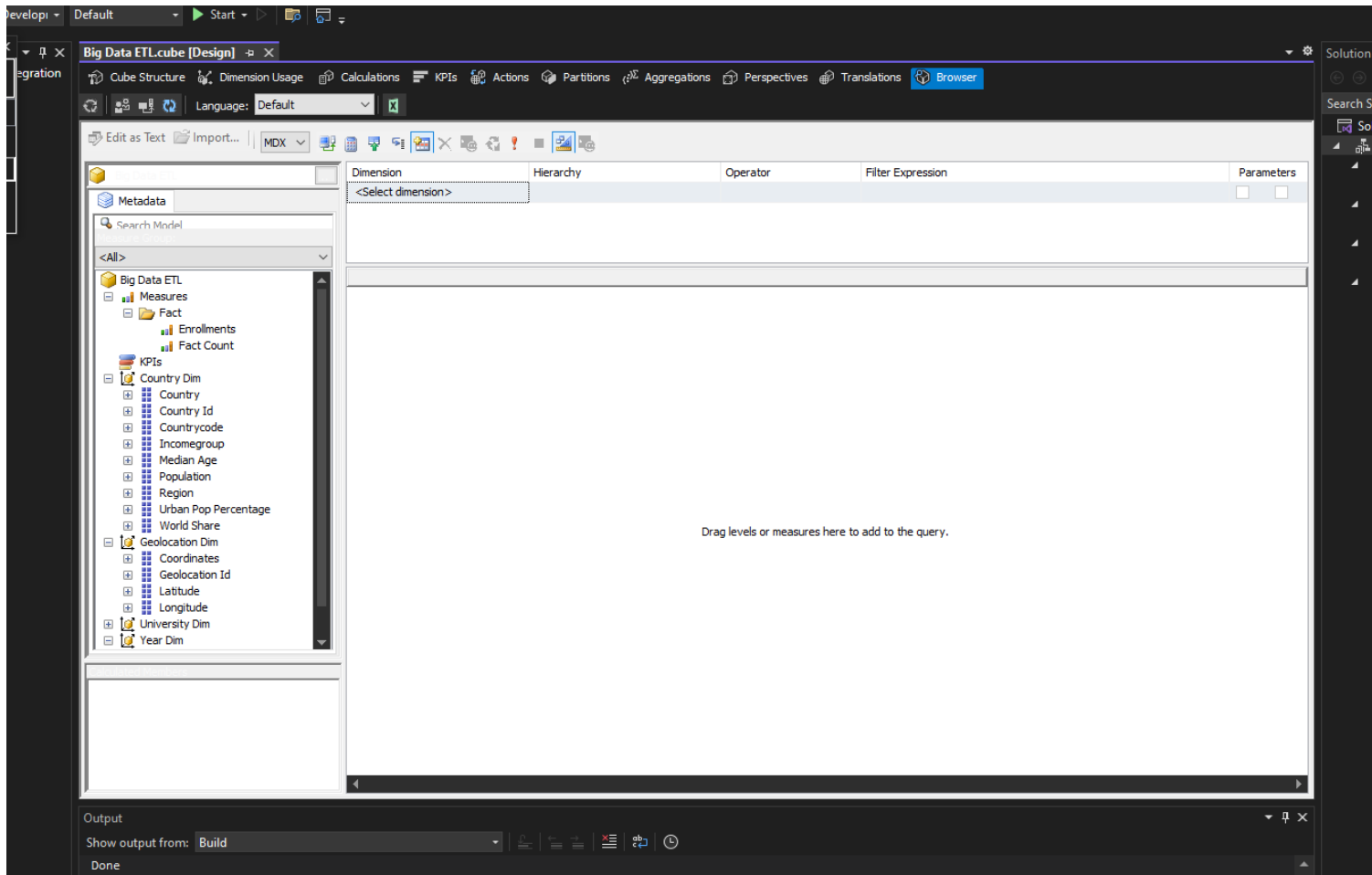
Και τώρα είμαστε έτοιμοι να κάνουμε process τον κύβο πατώντας το κουμπί “process” ανοίγει το αντίστοιχο dialog μετά από επιτυχές deployment.



Πατάμε run για να τρέξει ο κύβος μας.



Ο κύβος δημιουργήθηκε με επιτυχία. Πηγαίνοντας στο Browser μπορούμε να δούμε όλες τις επιλογές που υπάρχουν ώστε να κάνουμε queries και aggregations.



Τοποθετώντας τα attributes year και country και το measurement enrollments παίρνουμε τα συνολικά enrollments ανα χρονιά και χώρα και ξέρουμε ότι ο κύβος μας λειτουργεί επιτυχώς.

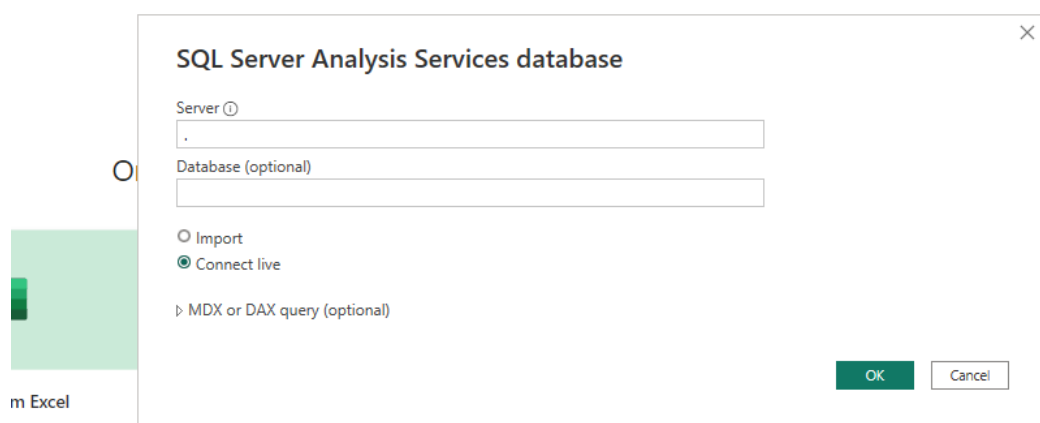
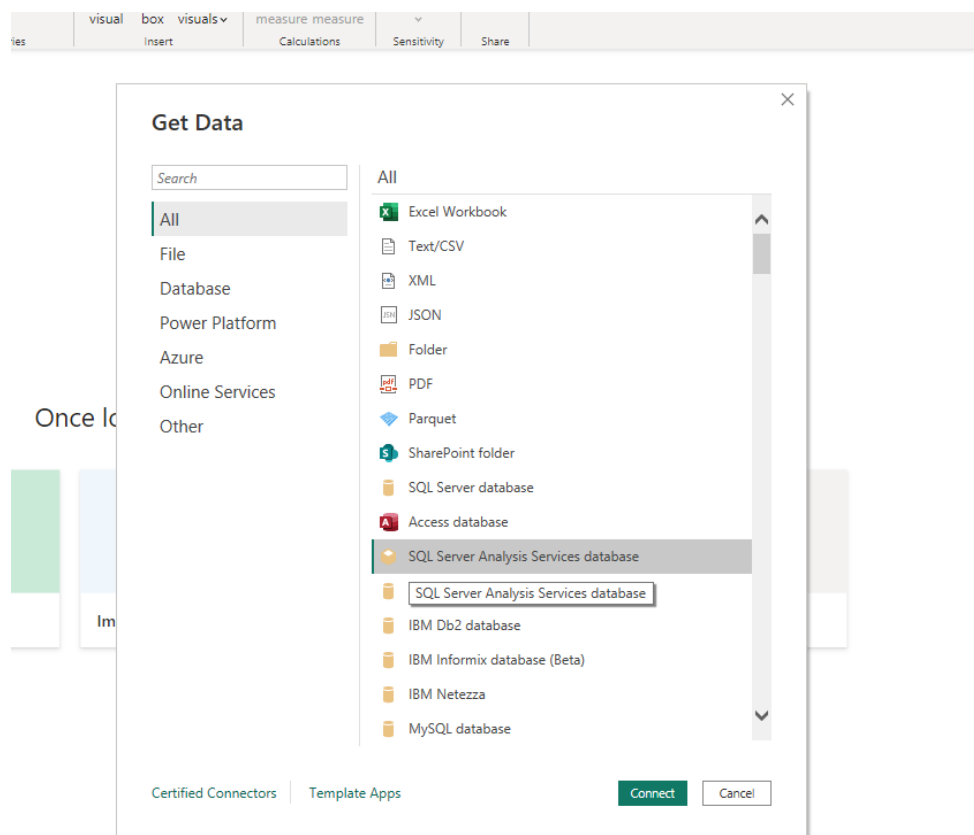
The screenshot shows the Big Data ETLcube [Design] interface. The left pane displays the metadata tree with dimensions: Country Dim, Geolocation Dim, University Dim, and Year Dim. The main area shows a table with columns: Country, Year, and Enrollments. The table contains data for Afghanistan and Albania from 1950 to 1995.

Country	Year	Enrollments
afghani...	1950	1122
afghani...	1955	1255
afghani...	1960	1670
afghani...	1965	3145
afghani...	1970	5158
afghani...	1975	11208
afghani...	1980	14506
afghani...	1985	12643
afghani...	1990	14050
afghani...	1995	20070
afghani...	2000	31588
afghani...	2005	47353
afghani...	2010	80078
afghani...	2015	125744
afghani...	2020	185436
albania	1950	0
albania	1955	473
albania	1960	5729
albania	1965	9376
albania	1970	15671
albania	1975	19905
albania	1980	21802
albania	1985	18064
albania	1990	20492
albania	1995	18286

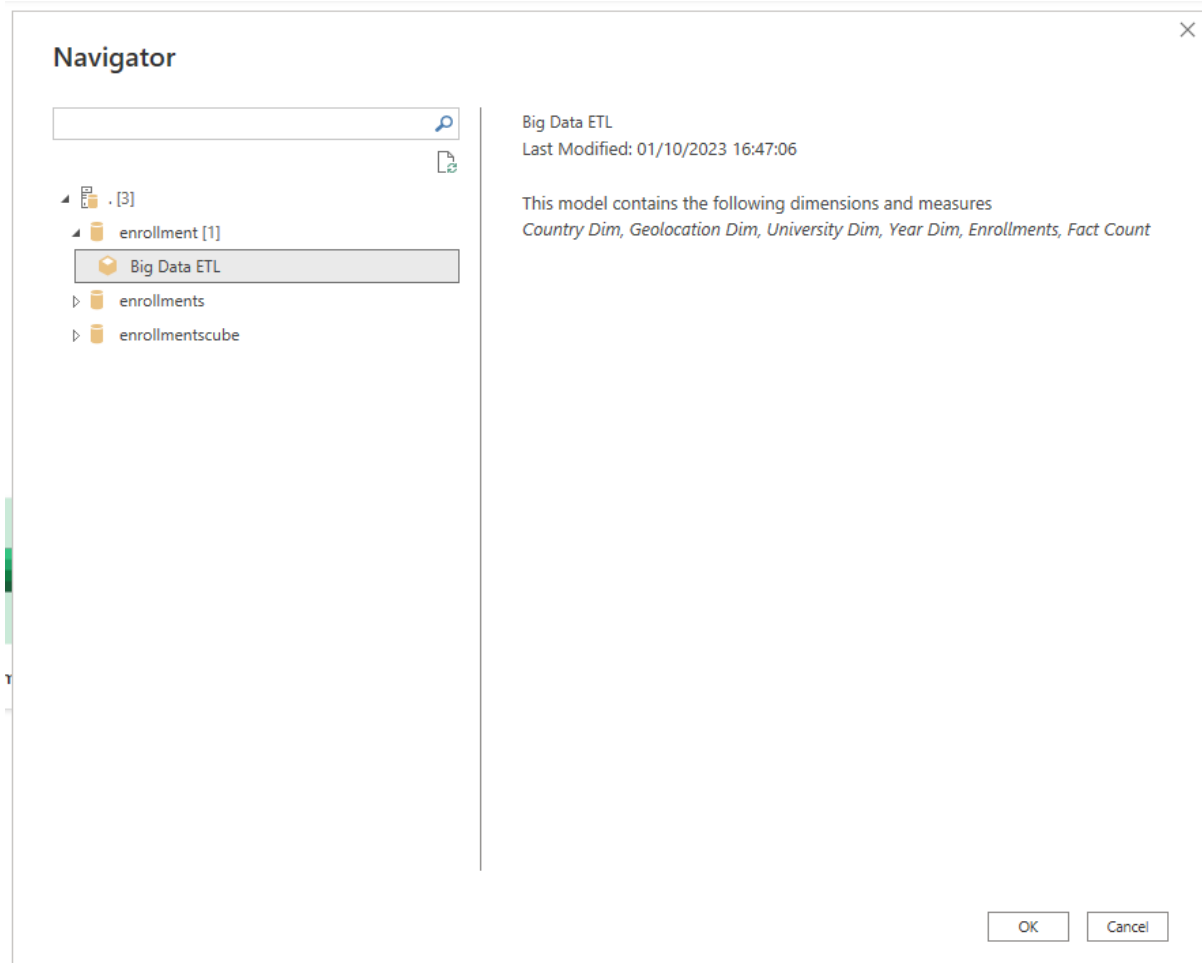
Data Visualization

Μετά την δημιουργία του κύβου χρησιμοποιήθηκε το PowerBI για την δημιουργία dashboard και visualizations.

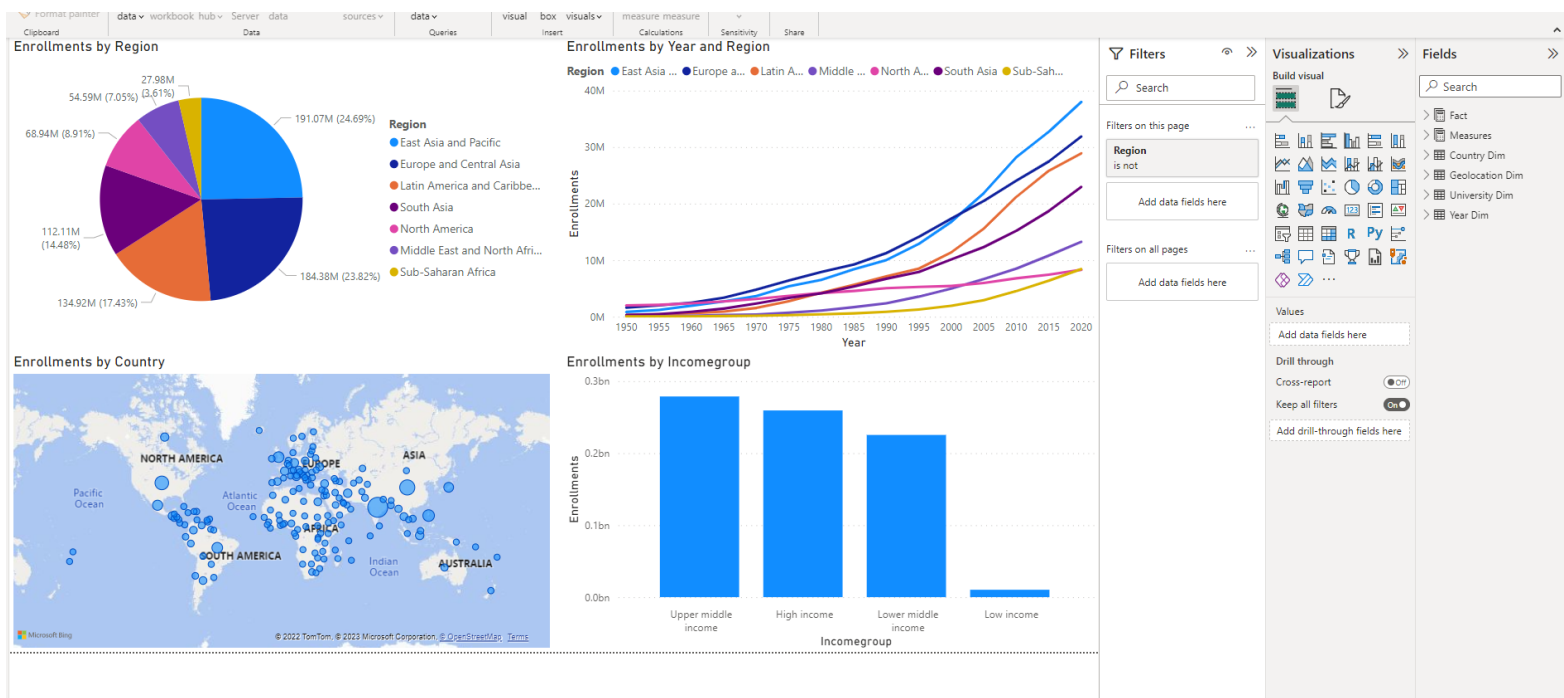
Πρώτα έγινε η σύνδεση του PowerBI Desktop με το Analysis Services Project μας και συγκεκριμένα τον κύβο μας.



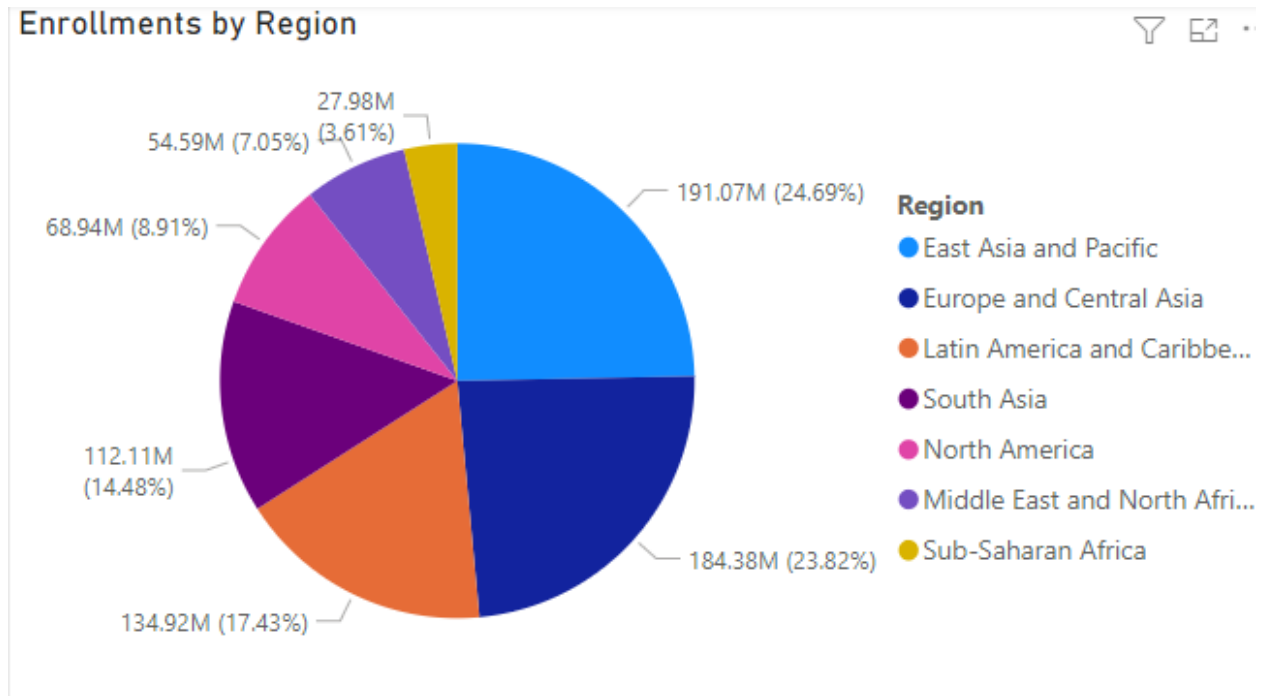
Get data from another source →



Στην συνέχεια δημιουργήσαμε το παρακάτω dashboard με τα visualization μας.

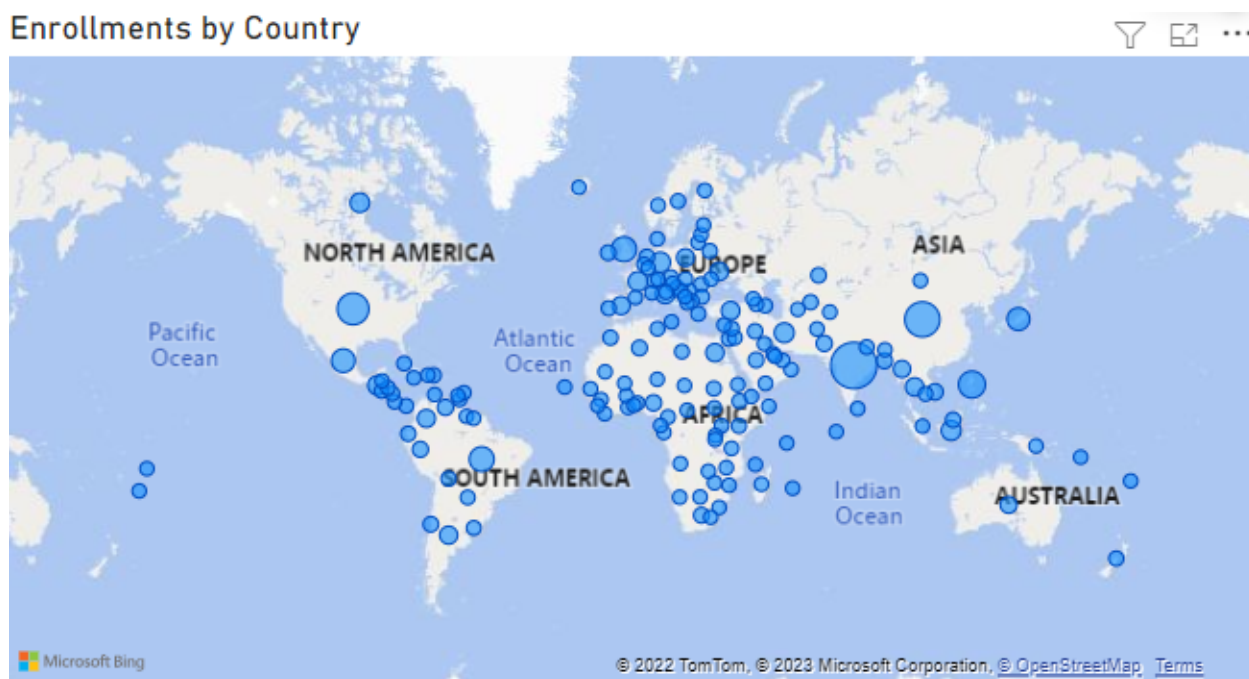


Enrollments by Region



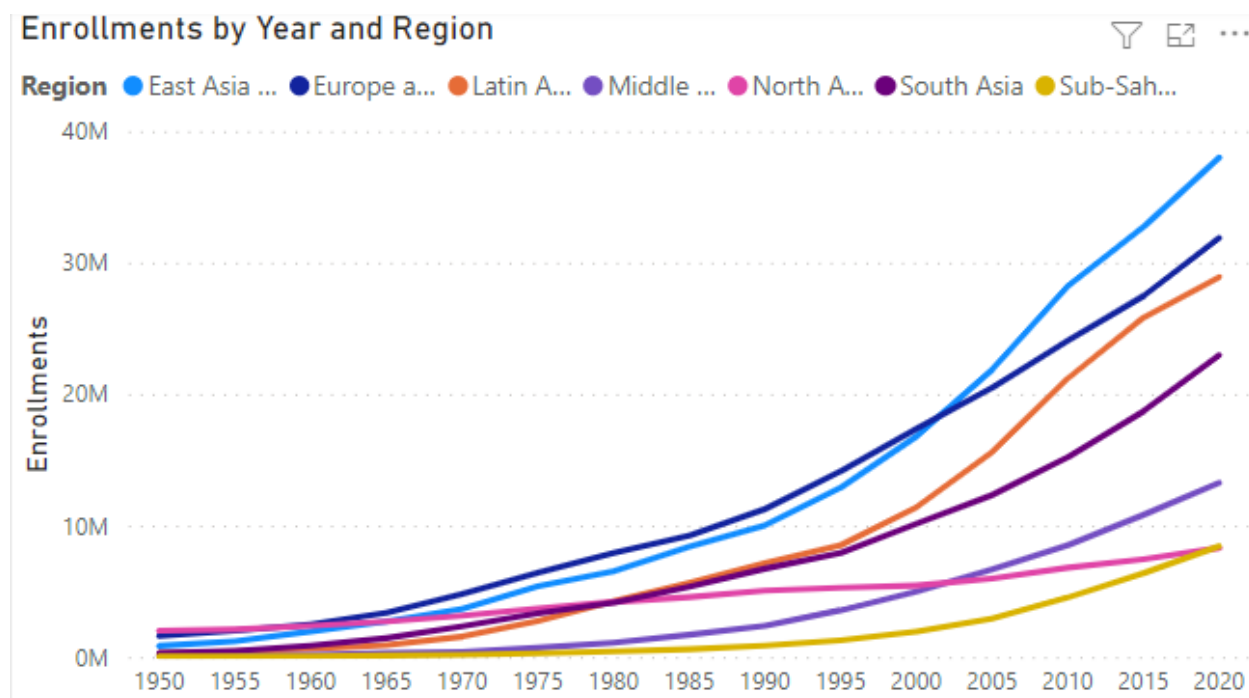
Στον παραπάνω pie chart μπορούμε να δούμε το ποσοστό των συνολικών enrollments για κάθε region (Ανατολική Ασία και Ειρηνικός, Ευρώπη και Κεντρική Ασία κλπ).

Enrollments by country



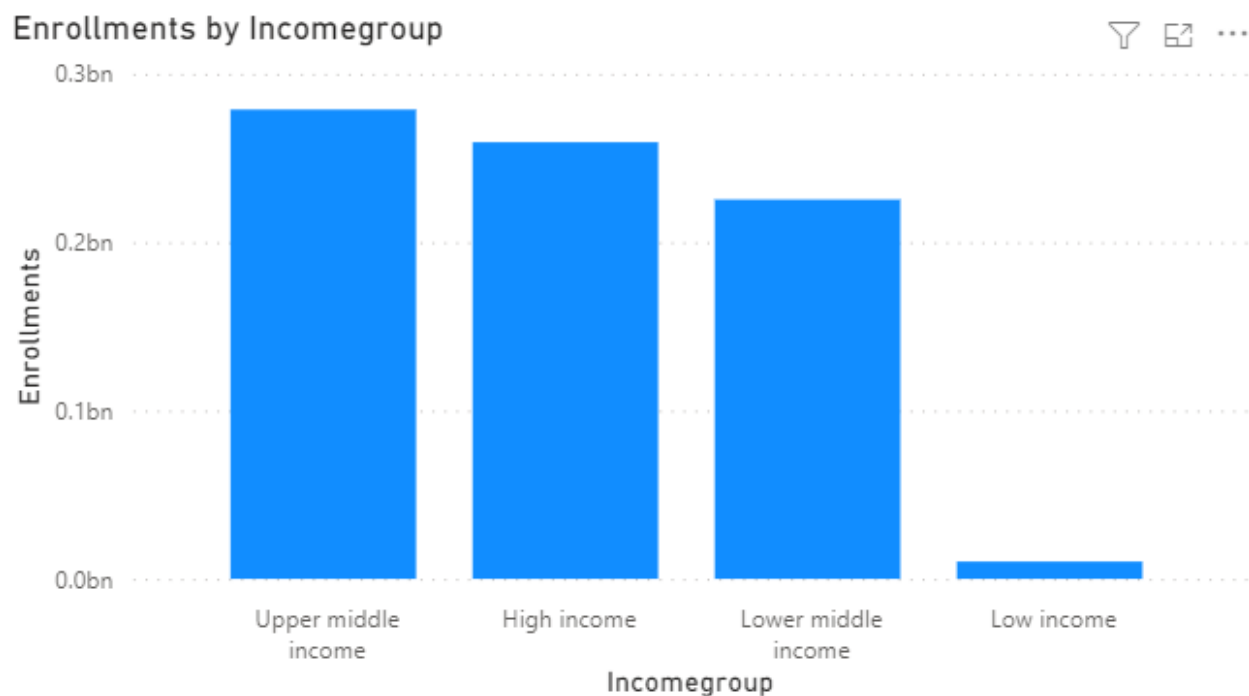
Στον παραπάνω χάρτη παρατηρούμε την τοποθεσία κάθε χώρας, σε κάθε κουκίδα και το σύνολο των enrollments της χώρας αυτής, με το μέγεθος της κουκίδας.

Enrollments by Year and Region



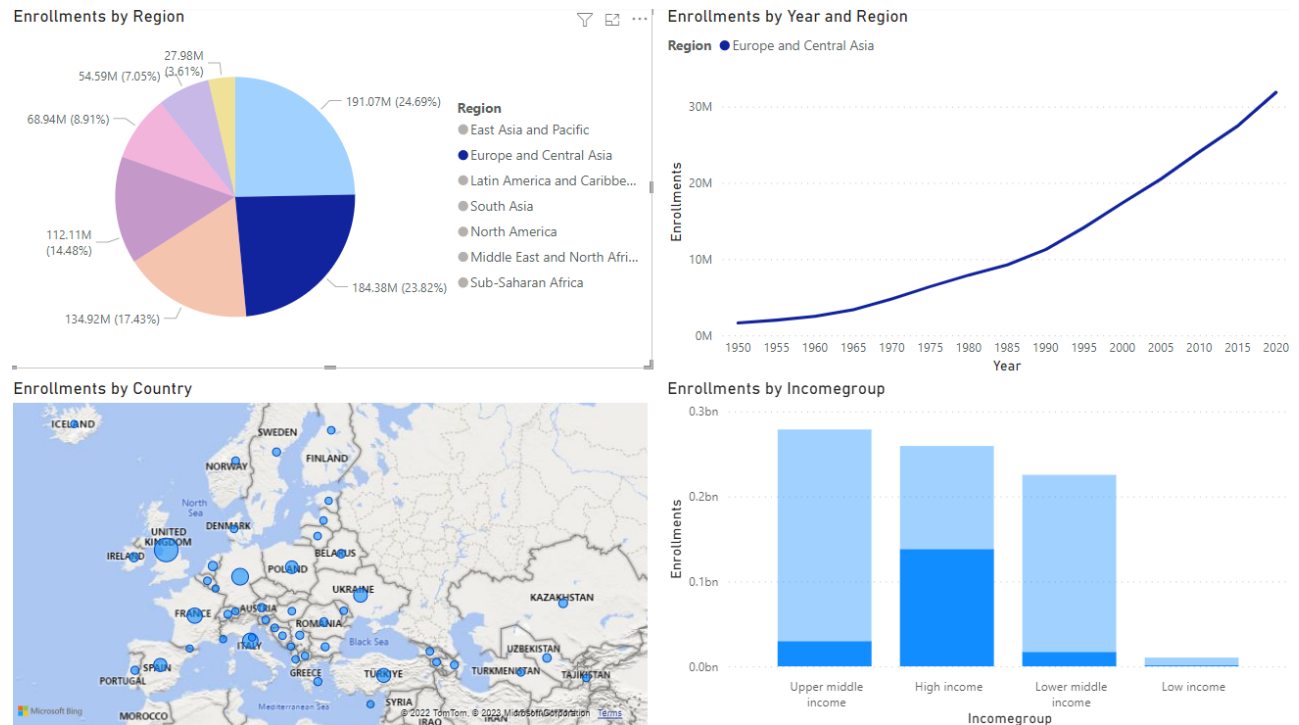
Στο παραπάνω line chart παρατηρούμε την αύξηση των enrollments ανά 5ετία για κάθε region.

Enrollments by income group

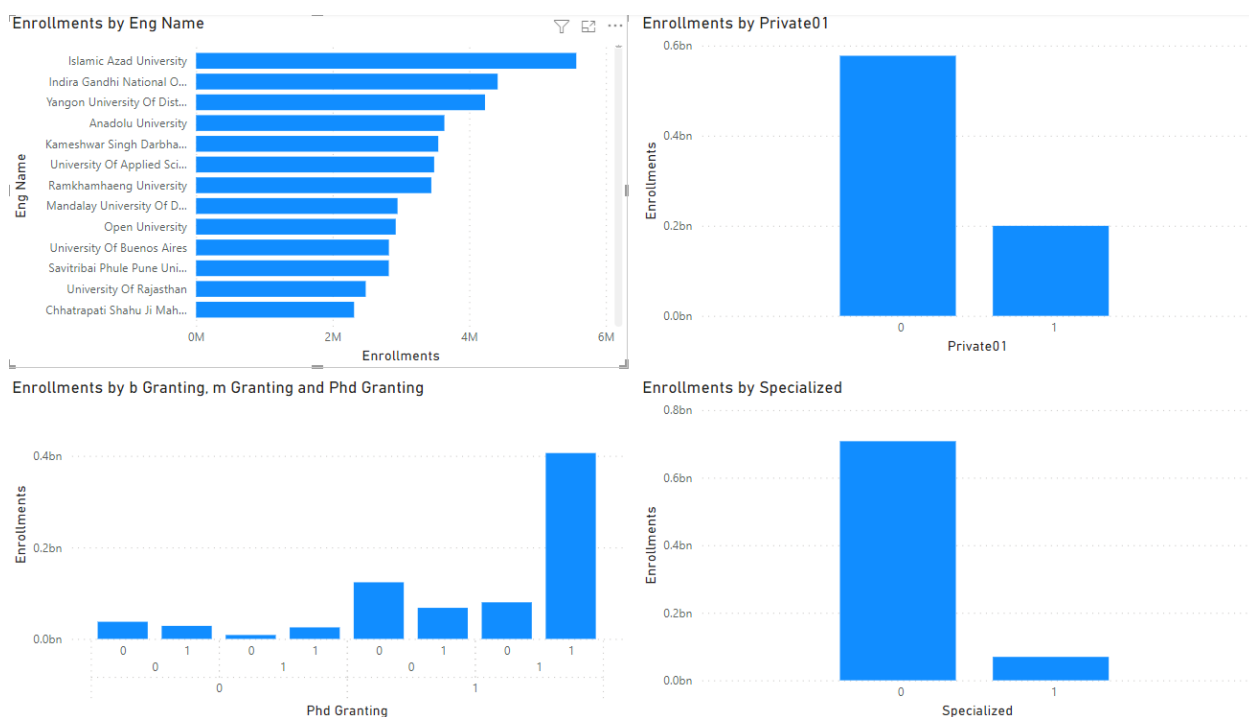


Στο παραπάνω column chart παρατηρούμε το σύνολο των enrollments παγκοσμίως ανά income group.

Κάνοντας κλικ σε ένα region ή μια χώρα, μπορούμε να δούμε τα δεδομένα αυτά μόνο για αυτή.



Στο δεύτερο dashboard φαίνονται visualization με πληροφορίες για τα πανεπιστήμια.

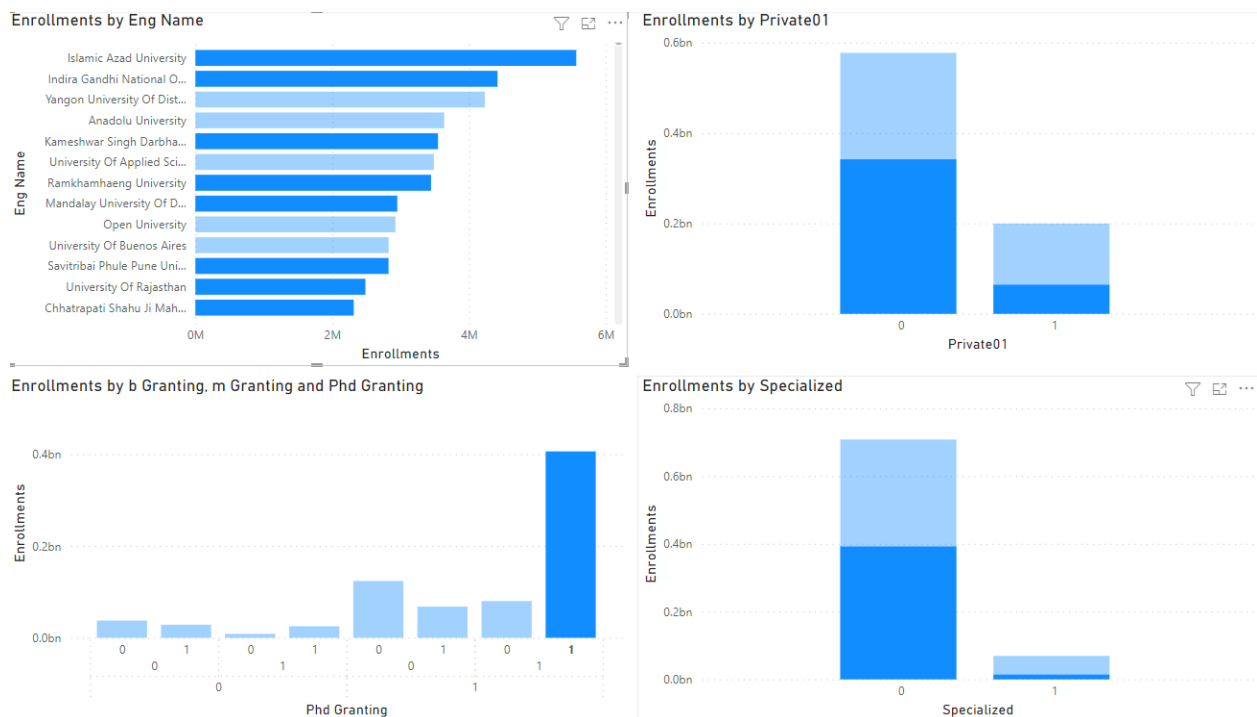


Στο πάνω αριστερά σχήμα βλέπουμε σε bar chart τα top πανεπιστήμια παγκοσμίως σε enrollments.

Στα άλλα 3 column charts βλέπουμε τα enrollments ανάλογα με το αν είναι ιδιωτικά ή όχι, τι παροχές έχουν σε πτυχία (phd, bachelors, masters) και άμα είναι ειδικευμένα ή όχι.

Επιλέγοντας ένα από αυτά τα χαρακτηριστικά, μπορούμε επίσης να δούμε ποια από τα πανεπιστήμια βρίσκονται σε αυτές τις κατηγορίες.

Για παράδειγμα παρακάτω φαίνονται τα πανεπιστήμια που παρέχουν και τους 3 τύπους grandings.



Data Mining Tasks

1. Decision Tree

Αρχικά θέλαμε να δούμε με ποιον τρόπο επηρεάζουν τα grantings, τα departments και τα fields τον αριθμό των enrollments για ένα πανεπιστήμιο.

Αρχικά έγινε ένα group by για κάθε πανεπιστήμιο, κρατώντας τον μέσο όρο των enrollments για όλες τις χρονιές.

Out[14]:

	eng_name	region	incomegroup	private01	phd_granting	b_granting	divisions	total_fields	unique_fields	specialized	merger	students5_estimated
0	17 August 1945 University Cirebon	East Asia and Pacific	Upper middle income	1	0	1	6	15	15	0	0	833.750000
1	17 August 1945 University Jakarta	East Asia and Pacific	Upper middle income	1	1	1	5	14	14	0	0	1765.846154
2	17 August 1945 University Samarinda	East Asia and Pacific	Upper middle income	1	0	1	6	17	16	0	0	830.166667
3	17 August 1945 University Semarang	East Asia and Pacific	Upper middle income	1	1	1	7	37	33	0	0	8214.555556
4	17 August 1945 University Surabaya	East Asia and Pacific	Upper middle income	1	1	1	6	23	23	0	0	4378.846154
...
15421	🔹🔹🔹Buraydah College For Applied Medical Sciences	Middle East and North Africa	High income	1	0	1	4	5	5	0	0	4163.750000
15422	🔹🔹🔹Buraydah College Of Administrative Sciences...	Middle East and North Africa	High income	1	0	1	4	5	5	0	0	4499.333333
15423	🔹🔹🔹Buraydah College Of Dentistry And Pharmacy	Middle East and North Africa	High income	1	0	1	2	8	8	0	0	4163.000000
15424	🔹🔹🔹Buraydah College Of Engineering And Informa...	Middle East and North Africa	High income	1	0	1	4	4	3	0	0	4163.750000
15425	🔹🔹🔹Dir University	Europe and Central Asia	Upper middle income	0	0	1	5	28	21	0	0	12729.000000

15426 rows × 12 columns

Υπολογίσαμε το βέλτιστο depth για το decision tree μας με GridSearch

```
In [48]: from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold

parameters = [
    {'max_depth': list(range(1, 11))},
]

cv = KFold(n_splits=5, shuffle=True, random_state=13)
enrollments_tree = GridSearchCV(DecisionTreeRegressor(), parameters, cv=cv)
enrollments_tree = enrollments_tree.fit(X_train, y_train)
print(enrollments_tree.best_estimator_)
print(enrollments_tree.best_params_)

DecisionTreeRegressor(max_depth=4)
{'max_depth': 4}
```

Και στο τέλος φτιάξαμε το decision tree

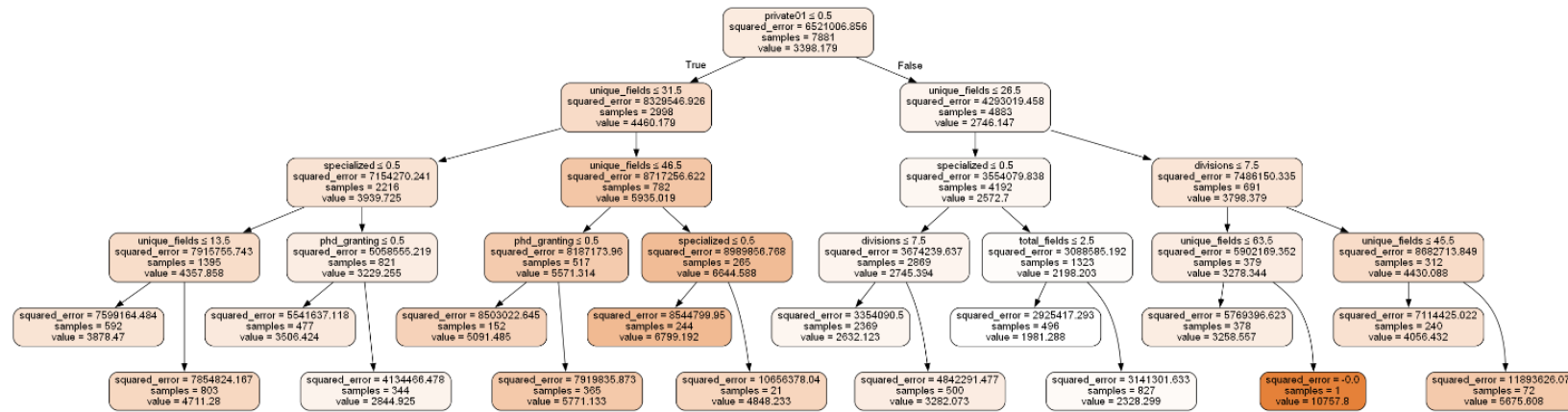
- -

```
In [54]: from sklearn.tree import DecisionTreeRegressor, export_graphviz

enrollments_tree = DecisionTreeRegressor(max_depth=4)

enrollments_tree.fit(X_train, y_train)
tree.export_graphviz(enrollments_tree, out_file= 'dec_tree')
```

Το τελικό αποτέλεσμα ήταν το εξής:



2. Clustering

Στην συνέχεια θέλαμε να δούμε ποια clusters incomegroup και region θα αποδώσουν τα περισσότερα enrollments

Αρχικά ξεκινήσαμε με το καθαρισμένο dataset από decision tree.

Κρατήσαμε μόνο τις στήλες του incomegroup, του region και των enrollments και

δημιουργήσαμε dummy variables για τις πρώτες 2.

```
In [23]: staging1 = staging.loc[:, ['region', 'incomegroup', 'students5_estimated']]
staging2 = pd.get_dummies(staging1, columns=['region', 'incomegroup'])
staging2
```

```
Out[23]:
```

	students5_estimated	region_East Asia and Pacific	region_Europe and Central Asia	region_Latin America and Caribbean	region_Middle East and North Africa	region_North America	region_South Asia	region_Sub- Saharan Africa	incomegroup_High income	incomegroup_in
0	833.750000	1	0	0	0	0	0	0	0	
1	1765.846154	1	0	0	0	0	0	0	0	
2	830.166667	1	0	0	0	0	0	0	0	
3	8214.555556	1	0	0	0	0	0	0	0	
4	4378.846154	1	0	0	0	0	0	0	0	
...
10452	4503.666667	0	0	0	1	0	0	0	1	
10453	4163.750000	0	0	0	1	0	0	0	1	
10454	4499.333333	0	0	0	1	0	0	0	1	
10455	4163.000000	0	0	0	1	0	0	0	1	
10456	4163.750000	0	0	0	1	0	0	0	1	

10457 rows x 12 columns

Στην συνέχεια κάναμε standard scaling στις μεταβλητές μας ώστε να επηρεάζουν με τον ίδιο τρόπο το clustering.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X = scaler.fit_transform(staging2)
scaled_staging = pd.DataFrame(X, index=staging2.index,
                              columns=staging2.columns)
scaled_staging
```

	students5_estimated	region_East Asia and Pacific	region_Europe and Central Asia	region_Latin America and Caribbean	region_Middle East and North Africa	region_North America	region_South Asia	region_Sub- Saharan Africa	incomegroup_High income	incomegroup_in
0	-0.994987	1.450479	-0.522561	-0.441376	-0.272940	-0.261645	-0.289183	-0.31391	-0.596017	-0.2
1	-0.629909	1.450479	-0.522561	-0.441376	-0.272940	-0.261645	-0.289183	-0.31391	-0.596017	-0.2
2	-0.996391	1.450479	-0.522561	-0.441376	-0.272940	-0.261645	-0.289183	-0.31391	-0.596017	-0.2
3	1.895885	1.450479	-0.522561	-0.441376	-0.272940	-0.261645	-0.289183	-0.31391	-0.596017	-0.2
4	0.393536	1.450479	-0.522561	-0.441376	-0.272940	-0.261645	-0.289183	-0.31391	-0.596017	-0.2
...
10452	0.442425	-0.689428	-0.522561	-0.441376	3.663802	-0.261645	-0.289183	-0.31391	1.677805	-0.2
10453	0.309288	-0.689428	-0.522561	-0.441376	3.663802	-0.261645	-0.289183	-0.31391	1.677805	-0.2
10454	0.440728	-0.689428	-0.522561	-0.441376	3.663802	-0.261645	-0.289183	-0.31391	1.677805	-0.2
10455	0.308995	-0.689428	-0.522561	-0.441376	3.663802	-0.261645	-0.289183	-0.31391	1.677805	-0.2
10456	0.309288	-0.689428	-0.522561	-0.441376	3.663802	-0.261645	-0.289183	-0.31391	1.677805	-0.2

10457 rows x 12 columns

Έπειτα με την χρήση σιλουέτας υπολογίσαμε τον βέλτιστο αριθμό clusters για το k-means που ήταν 19.

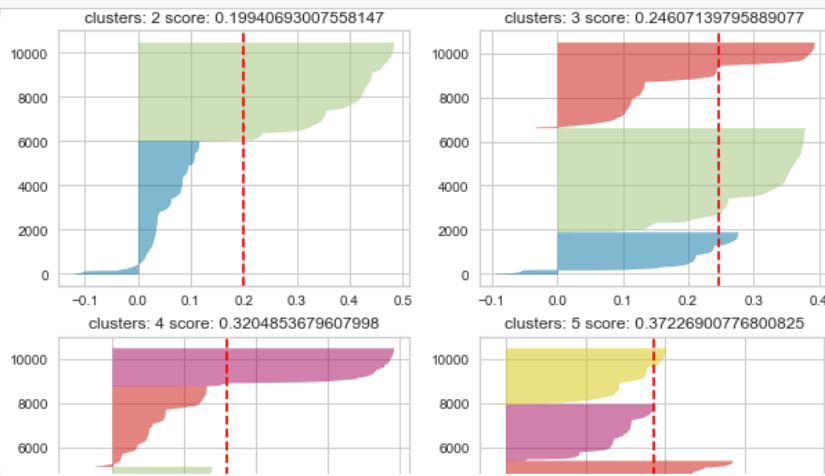
```

In [27]: from sklearn.cluster import KMeans
from matplotlib import pyplot as plt
from yellowbrick.cluster import SilhouetteVisualizer

plt.figure(figsize=(2 * 5, 10 * 4))

scores = {}
for n_clusters in range(2, 40):
    plt.subplot(10, 2, n_clusters - 1)
    kmeans = KMeans(n_clusters, random_state=42)
    visualizer = SilhouetteVisualizer(kmeans, colors='yellowbrick')
    visualizer.fit(scaled_staging)
    scores[n_clusters] = visualizer.silhouette_score_
    plt.title(f'clusters: {n_clusters} score: {visualizer.silhouette_score_}')

```



```

In [28]: sorted(scores.items(), key=lambda kv: kv[1], reverse=True)

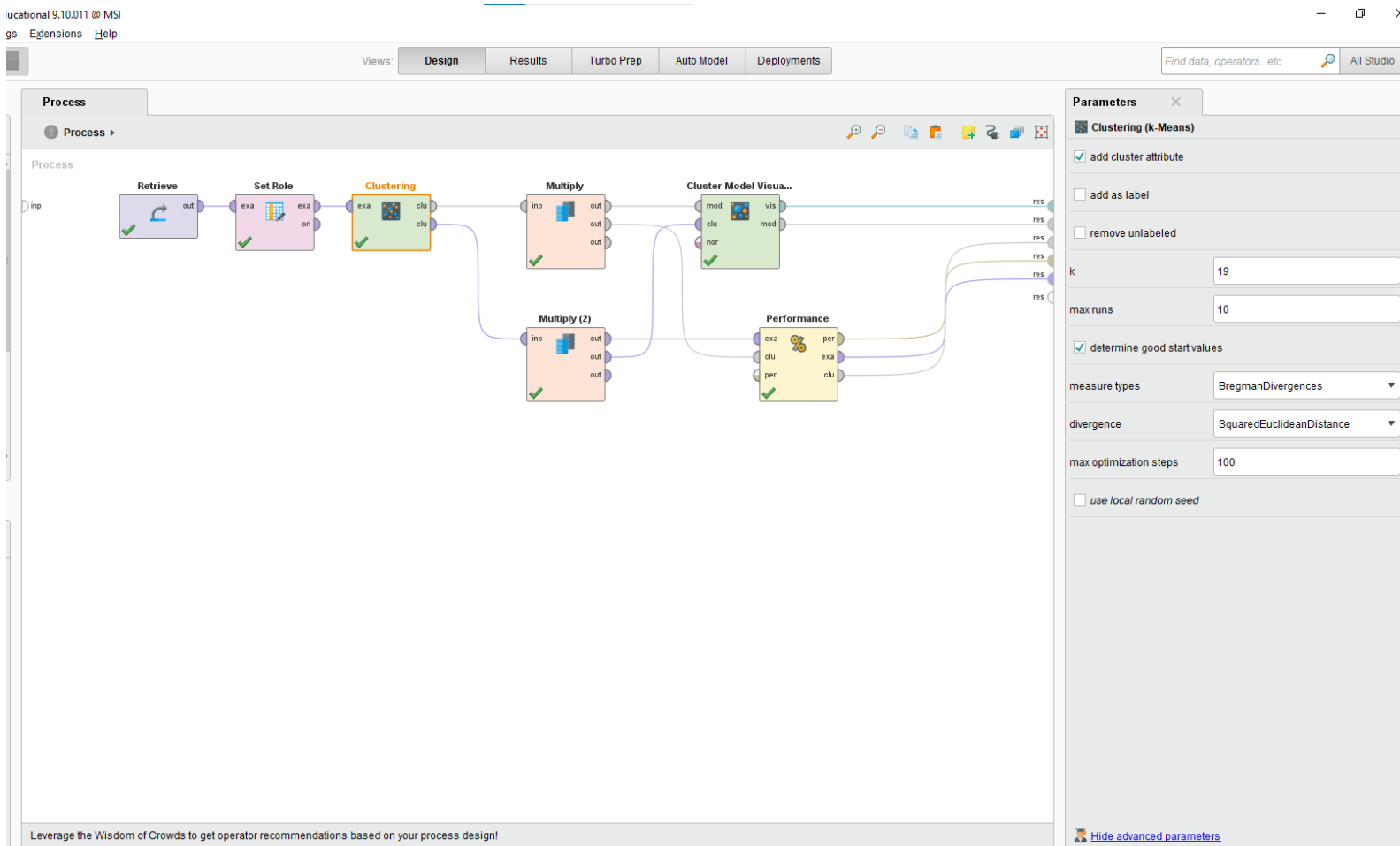
```

```

Out[28]: [(19, 0.711622161866037),
(20, 0.7084429841241247),
(18, 0.7078224445717771),
(17, 0.7012507313931889),
(21, 0.687923312217199),
(16, 0.6862705757568016),
(15, 0.6841712008037962),
(12, 0.6746370513328211),
(14, 0.6715534846994692),
(13, 0.6668373481943545),
(11, 0.6471567254736891),

```

Για το clustering χρησιμοποιήσαμε RapidMiner και το k-means process του



Και ως αποτέλεσμα λάβαμε το εξής:

