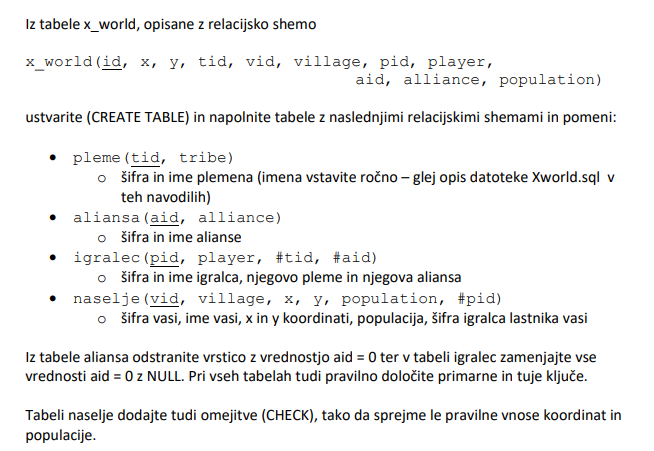
**PODATKOVNE BAZE**

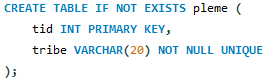
Seminarska naloga

Luka Šveigl, 63200301

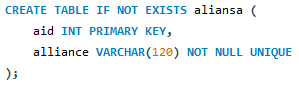
1. **NALOGA (DDL)**



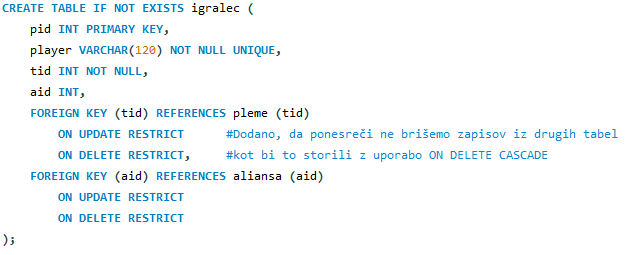
* 1. KREIRANJE TABELE PLEME



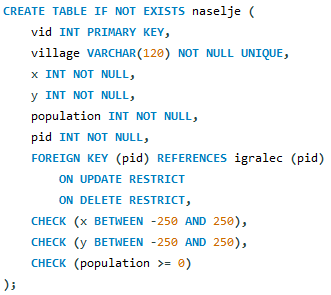
* 1. KREIRANJE TABELE ALIANSA



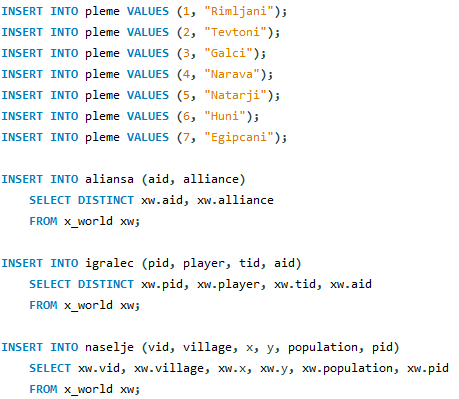
* 1. KREIRANJE TABELE IGRALEC



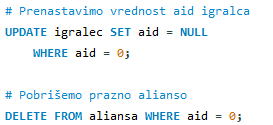
* 1. KREIRANJE TABELE NASELJE



* 1. POLNJENJE TABEL



* 1. ODSTRANITEV VREDNOSTI aid = 0

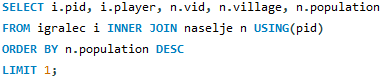


1. **NALOGA (DML)**

****

1. Izpišite šifro in ime igralca z največjim naseljem ter šifro, ime in velikost tega naselja.

Koda:

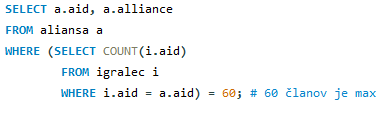


Rezultat:



1. Izpišite šifre in imena alians, ki imajo maksimalno število članov.

Koda:

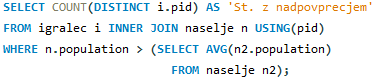


Rezultat:



1. Koliko igralcev ima nadpovprečno veliko naselje?

Koda:

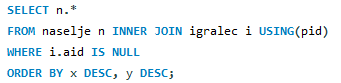


Rezultat:

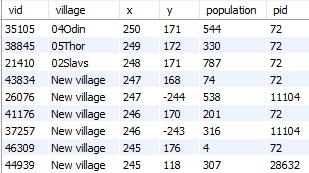


1. Izpiši podatke o vseh naseljih igralcev brez alianse, urejeno padajoče po x in nato y koordinati.

Koda:

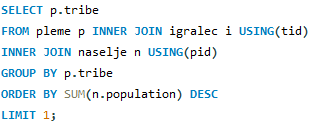


Rezultat:



1. Katero pleme je najštevilčnejše (glede na skupno populacijo)?

Koda:

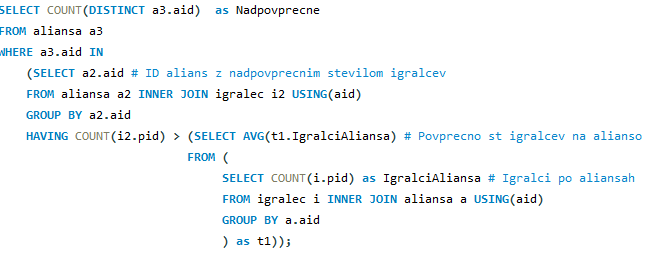


Rezultat:



1. Izpišite število alians z nadpovprečnim številom članov.

Koda:

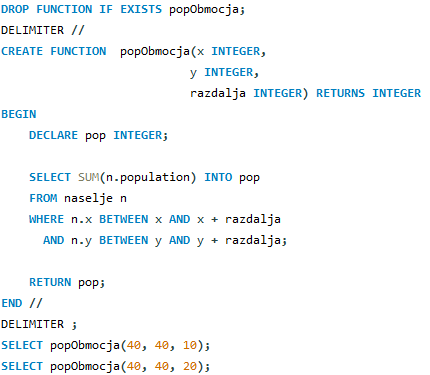


Rezultat:



1. Napišite shranjeno funkcijo popObmocja(x, y, razdalja), ki za poljubne koordinate vrne populacijo na območju od vključno [x,y] do [x+razdalja,y+razdalja]. Izpišite rezultat klica popObmocja(40, 40, 10) in popObmocja(40, 40, 20).

Koda:



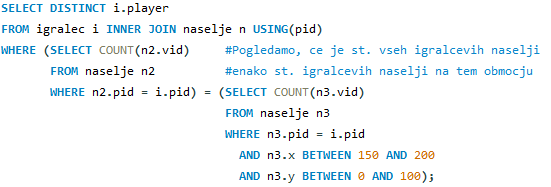
Rezultat:



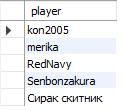


1. Izpišite imena igralcev, ki imajo vsa svoja naselja na območju x, ki je med 150 in 200 in y, ki je med 0 in 100.

Koda:

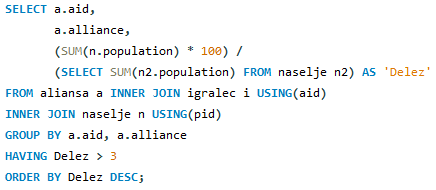


Rezultat:

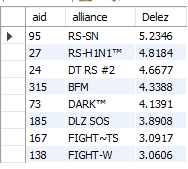


1. Izpišite šifre, imena in delež celotne populacije alians, ki imajo vsaj 3% vse populacije v igri. Rezultat uredite padajoče po deležu.

Koda:

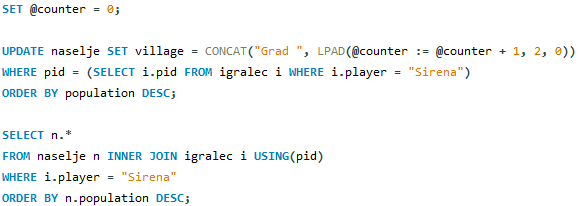


Rezultat:

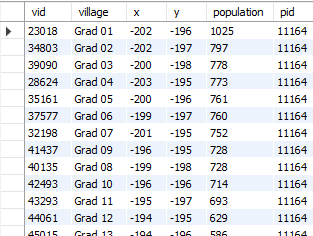


1. Igralec “Sirena” želi preimenovati vsa svoja naselja na naslednji način. Uredil jih bo po populaciji, najmočnejše bo “Grad 01”, naslednje “Grad 02” in tako dalje. Nalogo lahko rešite v več korakih (zaporedju poizvedb).

Koda:

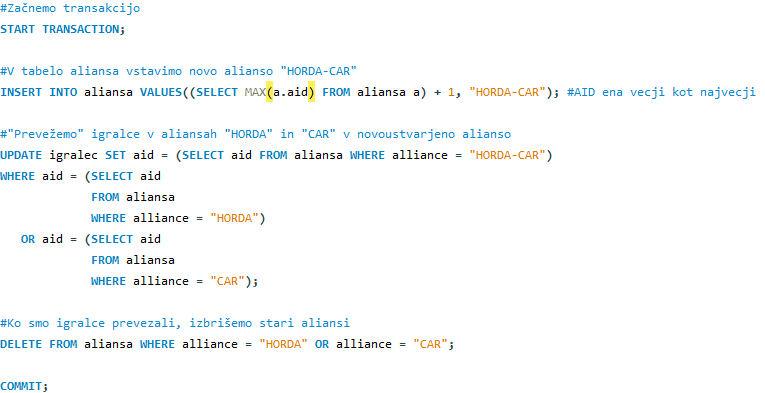


Rezultat:



1. **NALOGA (DDL)**
2. Napišite transakcijo (zaporedje ukazov), ki bo združila člane alians HORDA in CAR v novo imenovano alianso HORDA-CAR.

Koda:

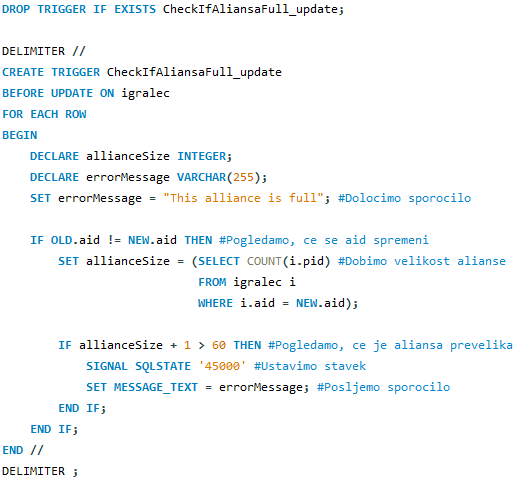


Rezultat:



1. Napišite bazni prožilec, ki bo ob spremembah vrednosti aid v tabeli igralec preveril, če aliansa še lahko sprejme novega člana.

Koda:



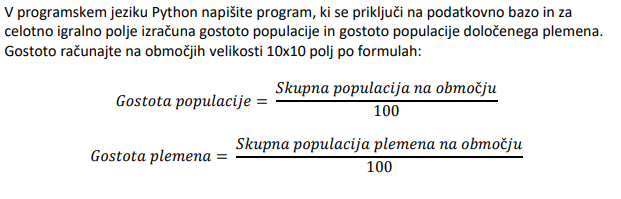


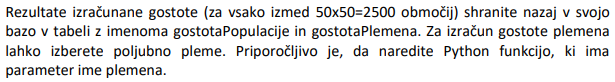
Rezultat:





1. **NALOGA (ODBC)**





* 1. OPIS REŠITVE

Pri reševanju te naloge sem se odločil za objektno orientiran pristop, saj mi razredi omogočajo enkapsulacijo kode in potrebnih atributov (spremenljivk), posledica tega pa je boljša berljivost in razdelanost kode.

V sklopu te naloge sem ustvaril razred DataBaseInterface, ki vsebuje atribute **conn** (tipa pyodbc.Connection, vsebuje povezavo na podatkovno bazo), **connection\_string** (tipa string, vsebuje podatke, potrebne za povezavo) in **pleme** (tipa string, vsebuje ime plemena, za katerega računamo populacijo). Ta razred vsebuje konstruktor, ki prejme ime podatkovne baze, ki jo želimo uporabiti, “javne” metode **prepare\_tabele()** (uporablja se za kreiranje potrebnih tabel), **calculate\_gostote(pleme)** (uporablja se za izracun gostot) in tudi **display\_gostote()** (požene se opcijsko, omogoča prikaz tabel). Ostale “privatne” metode le bolj podrobno razdelajo samo delovanje in se kličejo znotraj javnih metod avtomatsko.

Program vsebuje 2 skripti, **db\_interf.py**, ki vsebuje razred DataBaseInterface ter vso logiko programa in skripto **run\_db\_interf.py**, ki se uporablja le za zagon programa (kreiranje razreda, povpraševanje po imenu plemena in podatkovne baze, itd.). Sama koda v teh datotekah je tudi podrobno zakomentirana, tako da je delovanje hitro razumljivo.

Sam program izvajamo preko ukazne vrstice z ukazom “*python run\_db\_interf.py*”, skripta pa nas nato vpraša, katero shemo želimo uporabiti in za katero pleme želimo računati gostoto populacije. Da se program izvede v celoti, traja kar nekaj časa (lahko tudi do nekaj minut), saj Python ni najhitrejši jezik, poleg tega pa moramo izračunati gostote za ogromno podatkov (2x 2500).

* 1. PSEVDOKODA

1 vprašaj uporabnika po shemi -> shrani v spremenljivko database

2 vprašaj uporabnika po plemenu -> shrani v spremenljivko pleme

3 če pleme ni veljavno, ponovi korak 2

4 kreiraj objekt razreda DataBaseInterface in mu podaj spr. database kot argument

5 zbriši tabeli gostotaPopulacije in gostotaPlemena

6 ustvari tabeli gostotaPopulacije in gostotaPlemena

7 for x in range [-250, 250], x + 10

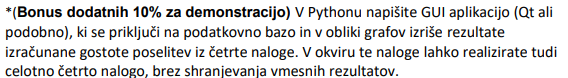
8 for y in range [-250, 250], y + 10

9 iz baze beremo podakte o gostoti na območju [x, x + 10][y, y + 10]

10 podatke shrani v spremenljivko val

11 v tabelo gostote zapiši val / 100

1. **NALOGA (ODBC)**

****

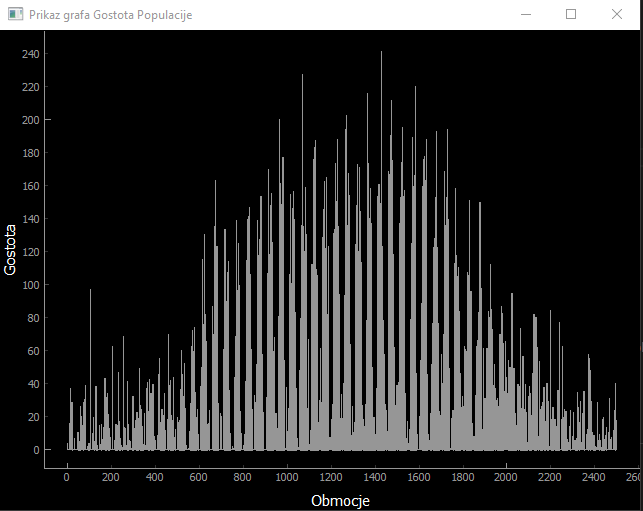
* 1. OPIS REŠITVE

Za reševanje te naloge sem uporabil Python knjižnico pyqtgraph, saj je preprosta za uporabo, zelo učinkovita, privzeto pa tudi omogoča premikanje po grafu in približevanje/oddaljevanje slike, kar nam omogoči makro in mikroskopski pogled na podatke. Knjižnico namestimo z ukazom “*pip install pyqtgraph*”.

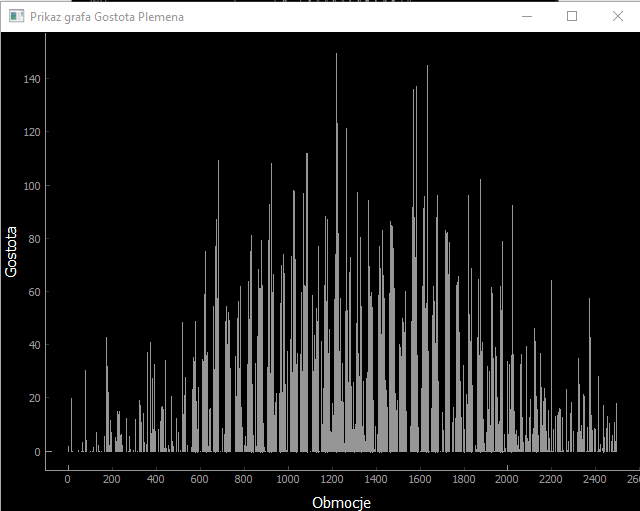
Tudi to nalogo sem reševal z objektno orientiranim pristopom, tako da sem ustvaril razred DataBaseGraphingInterface, ki vsebuje 2 atributa **window\_ple** in **window\_pop** (tipa PlotWidget, en za vsak graf), 2 atributa **xy\_pop** in **xy\_ple** (tipa dict, hranita vrednosti x in y za risanje grafov, x – zaporedna št. območja, y – gostota na tem območju), atribut **conn** (tipa pyodbc.Connection, hrani povezavo do podatkovne baze) in **connection\_string** (tipa string, vsebuje podatke, potrebne za povezavo). Ta razred vsebuje konstruktor, ki prejme ime podatkovne baze, ki jo želimo uporabiti, “javno” metodo **draw()** (uporablja se za pripravo (branje) podatkov in izris grafov na pravilno okno). Ostale “privatne” metode le bolj podrobno razdelajo samo delovanje in se kličejo znotraj javnih metod avtomatsko.

Cel razred je napisan v skripti **db\_graph\_interf.py**, prav tako kot skripta pri nalogi 4 pa se poganja preko skripte **run\_db\_interf.py**. To skripto sem modificiral tako, da na začetku uporabnika vpraša, katero nalogo želi pognati. Zaradi same strukture programa moramo pred zagonom naloge 5 **vsaj enkrat uspešno pognati nalogo 4**, da se kreirajo in napolnijo potrebne tabele.

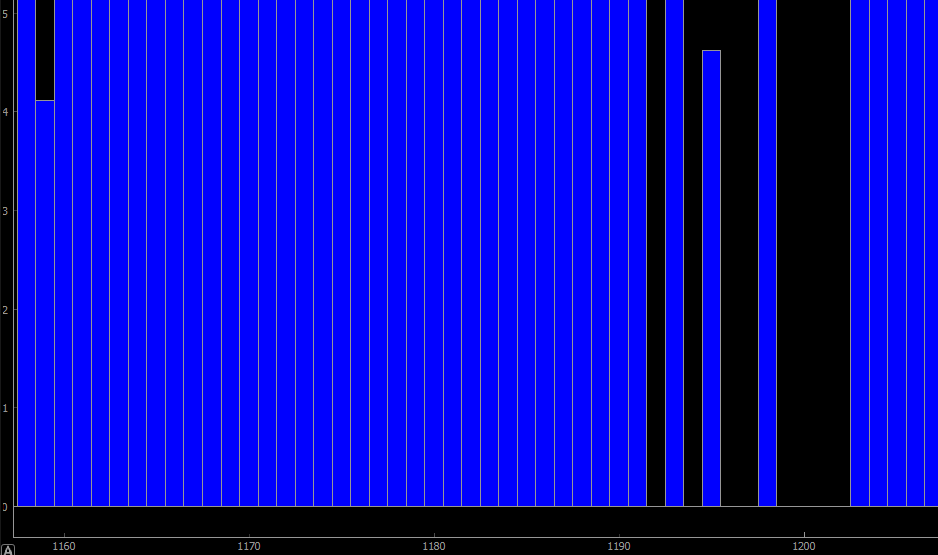
Kot lahko vidimo na naslednjih slikah, so posamezni stolpci dokaj tanki, saj jih je zelo veliko (2500). To niti ne predstavlja velikega problema, saj nam knjižnica pyqtgraph omogoča sprehanje po grafih z uporabo miške (kolešček, drag z levim gumbom, itd.), kar nam pomaga boljši pregled prikazanih podatkov.



Graf gostote populacije



Graf gostote plemena



Primer približanja