ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΜΑΤΙΚΗΣ

Εργασία Συστήματα Λήψης Αποφάσεων

Ομάδα 2:

Καζάκος Χρήστος, it22033

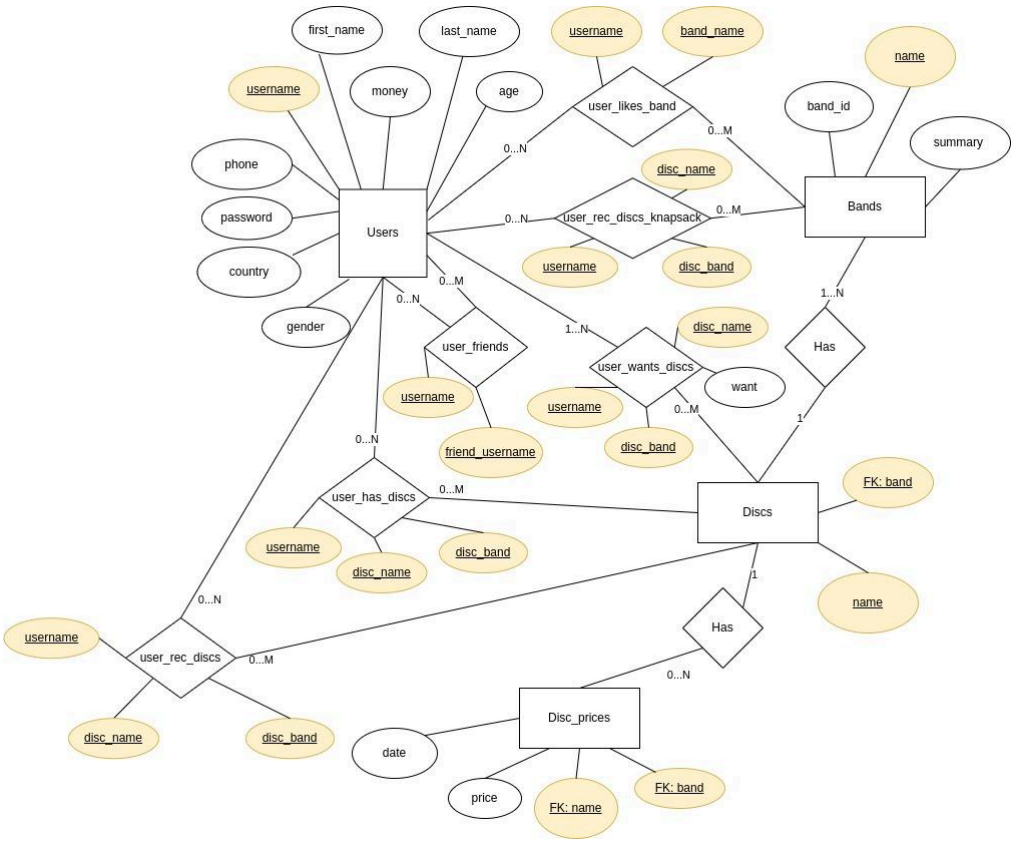
Κωνσταντίνος Κατσάρας, it22045

Μανούσος Λιναρδάκης, it22064

Contents

(α.) Σχεδιάστε το data model που θα αναπαριστά τους users του community, τα συγκροτήματα που τους αρέσουν καθώς και τους δίσκους που έχουν. Υλοποιήστε το scheme του data model σε μια σχεσιακή (SQL like) βάση δεδομένων (ΒΔ) και φτιάξτε συναρτήσεις που εισάγουν τυχαία δεδομένα της επιλογής σας για τους users.

ER Diagram:



Για καλύτερη προβολή του σχήματος, πατήστε εδώ.

Εδώ έχουμε τα Queries για τη δημιουργία των πινάκων της βάσης:

```
CREATE TABLE IF NOT EXISTS Users (
                username VARCHAR(50) PRIMARY KEY,
                password TEXT NOT NULL,
                first_name VARCHAR(50) NOT NULL,
                last_name VARCHAR(50) NOT NULL,
                gender CHAR,
                country VARCHAR(50),
                age INTEGER,
                phone VARCHAR(100) NOT NULL
            )

CREATE TABLE IF NOT EXISTS Bands (
                name VARCHAR(50) PRIMARY KEY,
                summary TEXT,
                band_id INTEGER
            )

CREATE TABLE IF NOT EXISTS Discs (
                name VARCHAR(250) NOT NULL,
                band VARCHAR(50) NOT NULL,
                PRIMARY KEY (name, band),
                FOREIGN KEY (band) REFERENCES Bands (name)
            )

CREATE TABLE IF NOT EXISTS user_has_discs (
                username VARCHAR(50),
                disc_name VARCHAR(250),
                disc_band VARCHAR(50),
                CONSTRAINT pk_user_has_discs PRIMARY KEY (username,
disc_name, disc_band),
                CONSTRAINT fk_user_has_discs_username FOREIGN KEY (username)
                    REFERENCES users (username),
                CONSTRAINT fk_user_has_discs_disc FOREIGN KEY (disc_name,
disc_band)
                    REFERENCES discs (name, band)
            )

CREATE TABLE IF NOT EXISTS user_likes_band (
                username VARCHAR(50),
                band_name VARCHAR(50),
                CONSTRAINT pk_user_likes_band PRIMARY KEY (username,
band_name),
                CONSTRAINT fk_user_likes_band_username FOREIGN KEY (username)
                    REFERENCES users (username),
                CONSTRAINT fk_user_likes_band_name FOREIGN KEY (band_name)
                    REFERENCES Bands (name)
            )
```

```
CREATE TABLE IF NOT EXISTS User_Friends (
               username VARCHAR(50) NOT NULL,
               friend_username VARCHAR(50) NOT NULL,
               PRIMARY KEY (username, friend_username),
               FOREIGN KEY (username) REFERENCES Users (username),
               FOREIGN KEY (friend_username) REFERENCES Users (username)
           )

CREATE TABLE IF NOT EXISTS disc_prices (
               date DATE NOT NULL,
               values FLOAT,
               name VARCHAR(250) NOT NULL,
               band VARCHAR(50) NOT NULL,
               FOREIGN KEY (name,band) REFERENCES Discs (name,band)
           )
CREATE TABLE IF NOT EXISTS user_rec_discs (
               username VARCHAR(50),
               disc_name VARCHAR(250),
               disc_band VARCHAR(50),
               CONSTRAINT pk_user_rec_discs PRIMARY KEY (username,
disc_name, disc_band),
               CONSTRAINT fk_user_rec_discs_username FOREIGN KEY (username)
                   REFERENCES users (username),
               CONSTRAINT fk_user_rec_discs_disc FOREIGN KEY (disc_name,
disc_band)
                   REFERENCES discs (name, band)
           )
```

```
CREATE TABLE IF NOT EXISTS user_wants_discs (
               username VARCHAR(50),
               disc_name VARCHAR(250),
               disc_band VARCHAR(50),
               want INTEGER,
               CONSTRAINT pk_user_wants_discs PRIMARY KEY (username,
disc_name, disc_band),
               CONSTRAINT fk_user_wants_discs_username FOREIGN KEY
(username)
                   REFERENCES users (username),
               CONSTRAINT fk_user_wants_discs_disc FOREIGN KEY (disc_name,
disc_band)
                   REFERENCES discs (name, band)
           )

CREATE TABLE IF NOT EXISTS user_rec_discs_knapsack (
       username VARCHAR(50),
       disc_name VARCHAR(250),
```

```
       disc_band VARCHAR(50),
       CONSTRAINT pk_user_rec_discs_knp PRIMARY KEY (username, disc_name,
disc_band),
       CONSTRAINT fk_user_rec_discs_username_knp FOREIGN KEY (username)
           REFERENCES users (username),
       CONSTRAINT fk_user_rec_discs_disc_knp FOREIGN KEY (disc_name,
disc_band)
           REFERENCES discs (name, band)
   )
```

(β.) Για τον κάθε user του community φτιαξτε συναρτήσεις που να ανακτούν δεδομένα από τα opendata σε σχέση με τα συγκροτήματα που ακούει και τους δίσκους που τον ενδιαφέρουν και να γεμίζουν τα αντίστοιχα tables στην ΒΔ. [Rest calls στα [1] [2] και Διάλεξη 3 MySQL]

Χρησιμοποιήσαμε τις μπάντες (ορισμένες στο .env file):

```
BAND_NAMES=coldplay scorpions the+beatles queen acdc u2
```

Επίσης, χρησιμοποιούμε και τα δύο προτεινόμενα api (discogs & lastfm). Από το discogs παίρνουμε τα ονόματα των δίσκων της κάθε μπάντας (έτσι ώστε να τα χρησιμοποιήσουμε ύστερα στο webscraping του discogs site), ενώ από το lastfm παίρνουμε επιπλέον πληροφορίες όπως το band summary και band name. Ύστερα στην βάση, συνδέουμε με foreign keys τις μπάντες με τους δίσκους που έχουν. Για τους χρήστες ύστερα διαλέγουμε τυχαία από τους διαθέσιμους δίσκους και μπάντες, έτσι ώστε να γεμίσουμε τα αντίστοιχα tables (user_likes_band, user_has_disc).

(γ.) Υλοποιήστε μια συνάρτηση που να κάνει έλεγχο και να αντιμετωπίζει missing [4], duplicate [5] ή outlier [6] values σε κάποιο από τα attributes της ΒΔ.

Για τον εντοπισμό των outliers, χρησιμοποιούμε την τεχνική winsorize, στην οποία τα outliers παίρνουν την κατάλληλη μέγιστη τιμή των boundaries. Συγκεκριμένα χρησιμοποιήσαμε το winsorize του scipy (tutorial). Όσο για τα missing values, παίρνουμε το μέσο όρο των προηγούμενων και το αντικαθιστούμε στη θέση του NaN. Τέλος, τα duplicates αντιμετωπίζονται με τα constraints που έχουμε ορίσει στη βάση (unique, primary keys). Σε περίπτωση που υπάρχει ήδη ένα ίδιο στοιχείο στη βάση, το αγνοούμε (η psql έχει την εντολή INSERT INTO ... ON CONFLICT DO NOTHING, η οποία απλώς αγνοεί τις εγγραφές που είναι duplicates).

(δ.) Βγάλτε κάποια στατιστικά συμπεράσματα για τα data features [7] [8]

Questions:

1. μέσος όρος ηλικίας χρηστών που ακούν τους Scorpions
2. χώρες που ακούνε περισσότερο μουσική
3. από πού κατάγονται οι περισσότεροι ακροατές των Scorpions
4. τι φύλου είναι οι περισσότεροι ακροατές των Scorpions
5. μπαντα με την μεγαλυτερη ακροαματικοτητα (μεσω των user favourites)
6. μέσος όρος δίσκων που έχουν οι χρήστες
7. από που είναι οι χρήστες που ακούν τους Scorpions
8. από που είναι οι χρήστες που έχουν τον δίσκο ”My Universe”
9. πλήθος χρηστών που έχουν τον δίσκο “My Universe”.
10. μεγαλύτερο πλήθος φύλων που έχουν τον δίσκο “My Universe”.
11. ποιο συγκρότημα προτιμούν οι χρήστες ανά χώρα
12. Top 5 Discs (σε ποσότητα)

Execution:

```
print("Q1 =============== ")
print(avg_user_band_age(conn, "scorpions"))
print("Q2 =============== ")
print(countries_most_music(conn))
print("Q3 =============== ")
print(band_most_listeners(conn, "scorpions"))
print("Q4 =============== ")
print(band_most_gender(conn, "scorpions"))
print("Q5 =============== ")
print(band_with_most_listeners(conn))
print("Q6 =============== ")
print(avg_disc_count(conn))
print("Q7 =============== ")
print(band_users_by_country(conn, "scorpions"))
print("Q8 =============== ")
print(disc_users_country(conn, "My Universe"))
print("Q9 =============== ")
print(num_users_with_disc(conn, "My Universe"))
print("Q10 =============== ")
print(disc_most_gender(conn, "My Universe"))
print("Q11 =============== ")
print(most_listened_bands_by_country(conn))
print("Q12 =============== ")
print(top_x_discs_by_quantity(conn))
```

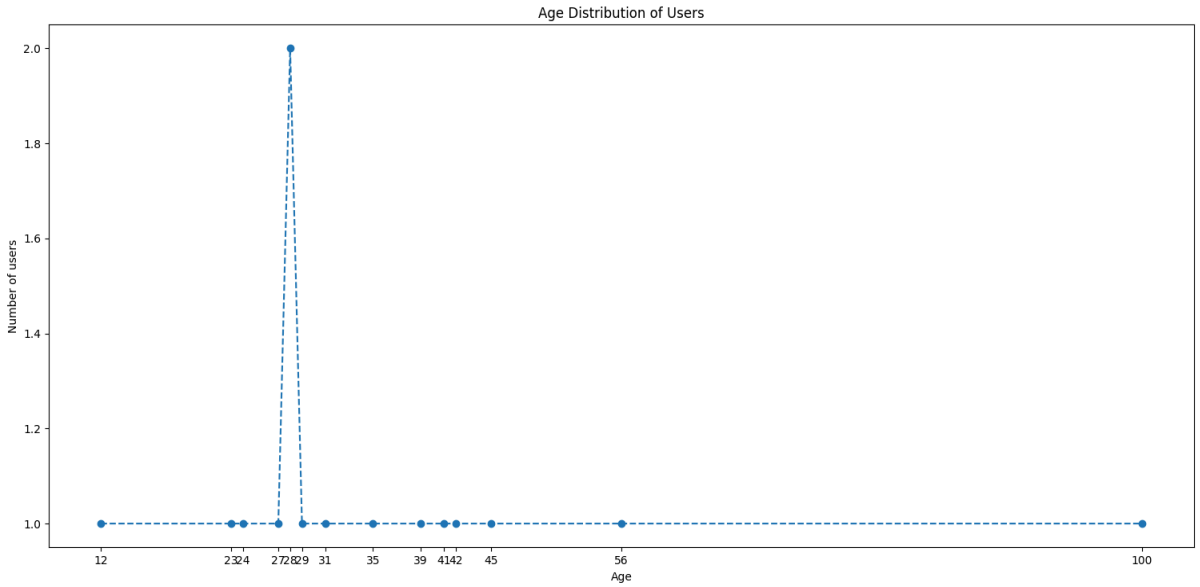
Results:

```
Q1 ===============
17.00
Q2 ===============
{'Italy': 1, 'United Kingdom': 1, 'Germany': 1, 'Canada': 2, 'Greece':
2, 'France': 1, 'South Africa': 1, 'Japan': 1, 'United States': 1,
'Brazil': 1, 'Australia': 1, 'Spain': 1, 'Mexico': 3, 'Poland': 1}
Q3 ===============
Canada
Q4 ===============
M
Q5 ===============
Queen
Q6 ===============
3
Q7 ===============
{'Canada': 1, 'Germany': 1, 'Greece': 1, 'Italy': 1, 'Mexico': 1,
'United States': 1}
Q8 ===============
['Greece', 'South Africa']
Q9 ===============
2
Q10 ===============
F
Q11 ===============
{'Australia': ('The Beatles', 1), 'Brazil': ('Coldplay', 1), 'Canada':
('Scorpions', 3), 'Germany': ('ACDC', 2), 'Greece': ('Scorpions', 3),
'Italy': ('Coldplay', 2), 'Japan': ('Scorpions', 2), 'Mexico':
('Scorpions', 1), 'South Africa': ('Coldplay', 3), 'Spain':
('Scorpions', 3), 'United Kingdom': ('Scorpions', 2), 'United States':
('Scorpions', 1)}
Q12 ===============
[('U218 Singles (deluxe version)', 'U2', 2), ('Moment of Glory',
'Scorpions', 2), ('Eye II Eye', 'Scorpions', 2), ('Rock Believer',
'Scorpions', 2), ('My Universe', 'Coldplay', 2)]
```

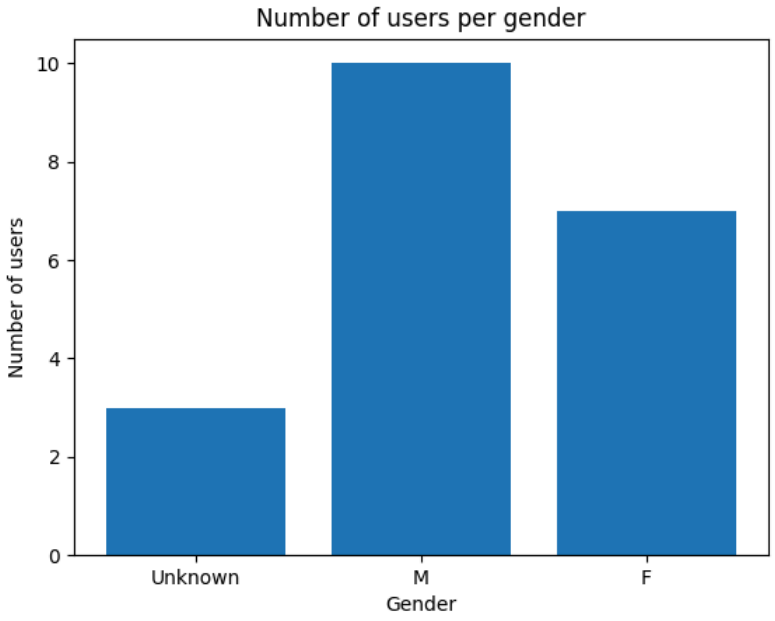
Τα παραπάνω αποτελέσματα είναι τα αποτελέσματα των ερωτημάτων που αναγράφονται πιο πάνω. Το Q1 αποτελεί την απάντηση της ερώτησης 1, το Q2 της ερώτησης 2 και ούτω καθεξής.

(ε.) Αναπαραστήστε γραφικά κάποια attributes των δεδομένων σας και εξηγήστε τα συμπεράσματα που εξάγετε [διάλεξη data visualization].

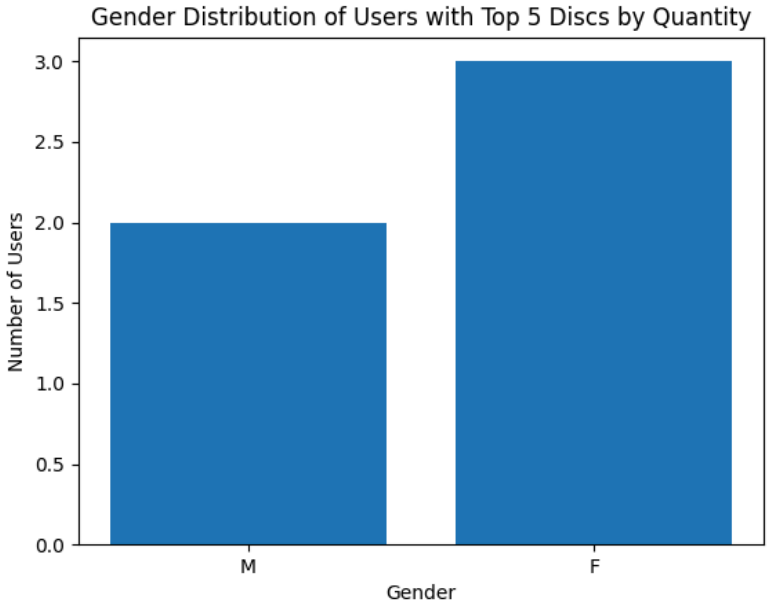
Τρέχοντας το stats.py, δημιουργήθηκαν τα παρακάτω γραφήματα:



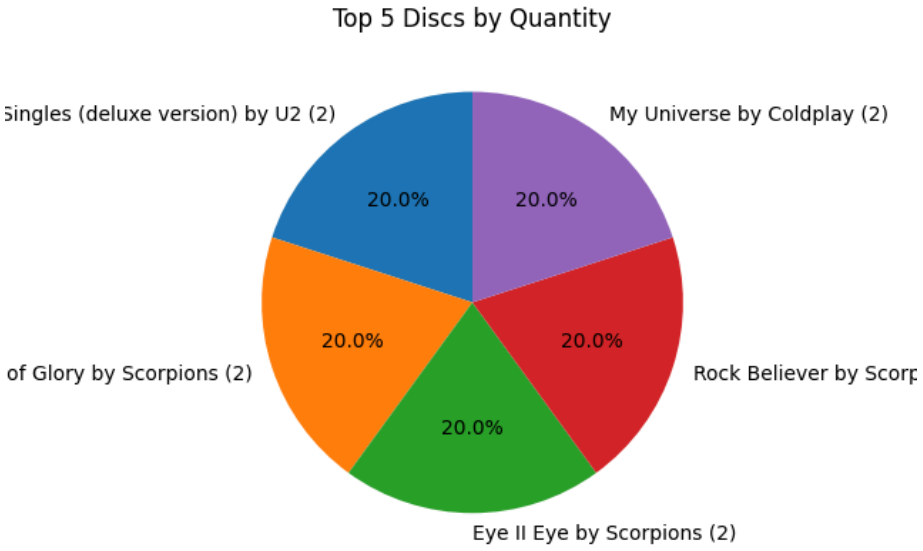
Στο παραπάνω διάγραμμα βλέπουμε το πώς οι χρήστες κατανέμονται με βάση την ηλικία τους.



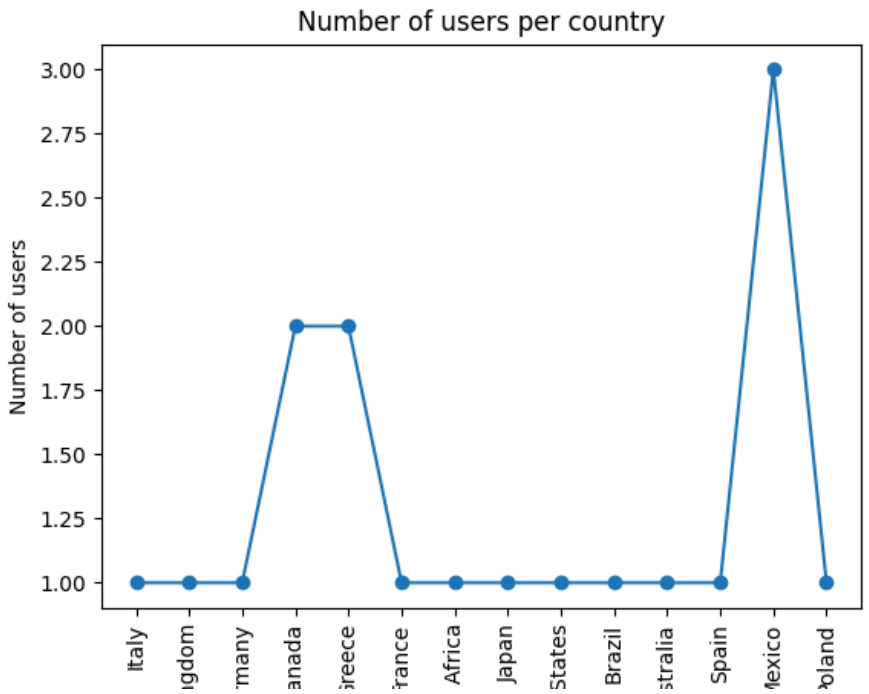
Στο παραπάνω διάγραμμα βλέπουμε τη κατανομή των χρηστών με βάση το φύλο τους. Συγκεκριμένα υπάρχουν 10 άντρες, 7 γυναίκες, ενώ 3 ακόμη άτομα δεν έχουν δηλώσει το φύλο τους.



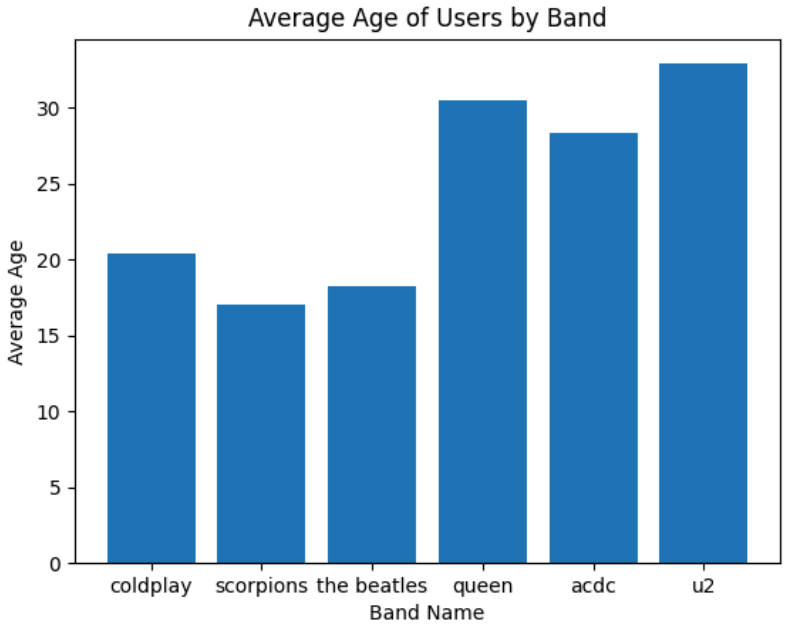
Στο παραπάνω διάγραμμα βλέπουμε πως διανέμονται οι top 5 δίσκοι στους χρήστες με βάση το φύλο τους. Δηλαδή οι γυναίκες κατέχουν 3 εξ αυτών ενώ οι άντρες 2.



Στο παραπάνω διάγραμμα βλέπουμε τους κορυφαίους 5 δίσκους που έχουν στη κατοχή τους οι χρήστες. Συγκεκριμένα το 20% των χρηστών κατέχουν κάθε έναν από τους δίσκους που φαίνονται στη “πίτα”.

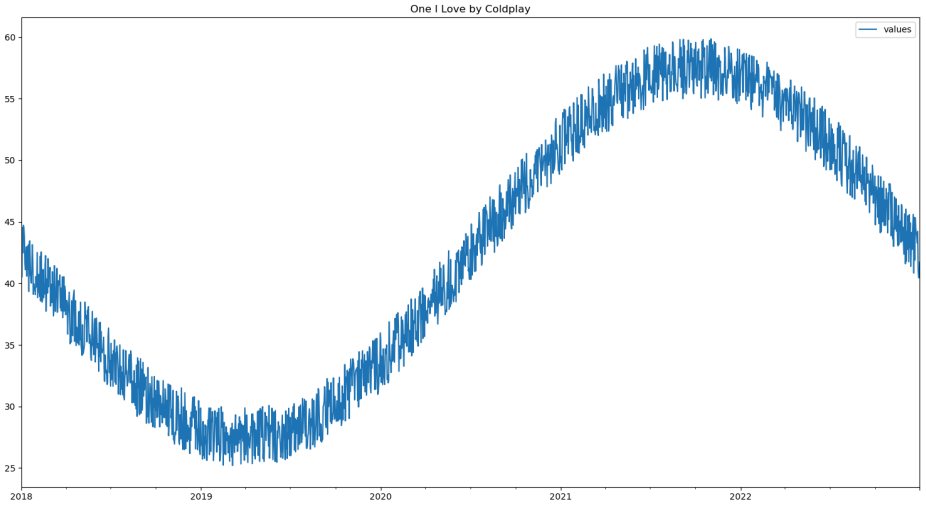


Στο παραπάνω διάγραμμα βλέπουμε τις χώρες καταγωγής των χρηστών. Συγκεκριμένα υπάρχει ένας χρήστης από την Ιταλία, το Ηνωμένο Βασίλειο, τη Γερμανία, τη Γαλλία, την Ιαπωνία, τις ΗΠΑ, τη Βραζιλία, την Αυστραλία, την Ισπανία, την Πολωνία και από την Αφρική. Παράλληλα, 2 χρήστες έχουν καταγωγή από την Ελλάδα, 2 από τον Καναδα και 3 από το Μεξικό.



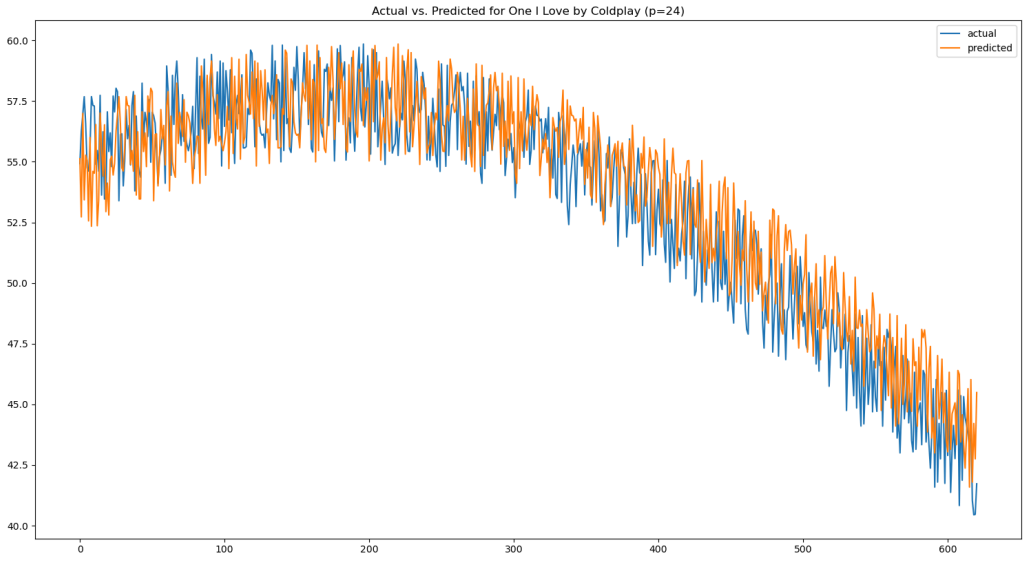
Στο παραπάνω διάγραμμα βλέπουμε τη μέση ηλικία των χρηστών που ακουν τη κάθε μπάντα. Δηλαδή ο μέσος όρος ηλικίας των φαν των Coldplay είναι 20, των Scorpions 17 και ούτω καθεξής.

(στ.) Έστω το synthetic time series dataset [9] που έχει την χαμηλότερη τιμή που πωλείται ένας δίσκος κάθε μέρα, αναπαραστήστε την χρονοσειρά και το decompose της σε trend, seasonality και residuals [διάλεξη και lab time series].



Στο διάγραμμα αυτό βλέπουμε τη διακύμανση των τιμών τα προηγούμενα χρόνια με βάση το δοσμένο csv αρχείο ή το web scraping από το discogs, ανάλογα με την επιλογή του χρήστη.

(ζ.) Χωρίστε τα data της χρονοσειράς σε 66% training και 34% testing, εκπαιδεύστε ένα ARIMA μοντέλο στο training split το οποίο να προβλέπει την χαμηλότερη τιμή που θα πωλείται ένας δίσκος την επόμενη ημέρα και αξιολογήστε την απόδοση του στο testing split [διάλεξη και lab time series].



Γραφική αναπαράσταση διαφοράς μεταξύ actual τιμής και της predicted. Παρατηρούμε ότι τα predictions είναι σχεδόν τα ίδια με την πραγματικότητα.
Η απόδοση του testing split έγινε χρησιμοποιώντας τη μετρική RMSE και έβγαλε ως αποτέλεσμα RMSE = 2.233.

(η.) Με χρήση κάποιας graph mining μετρικής πάνω στον γράφο του community να προτείνετε σε κάποιον user έναν δίσκο που δεν έχει επιλέξει [διάλεξη graph mining].

Για το recommendation χρησιμοποιήσαμε το networkx. Συγκεκριμένα, αφού φτιάξαμε το barabasi model, φτιάξαμε και έναν καινούργιο γράφο ο οποίος έχει ως nodes τα usernames και ως data features τους δίσκους των χρηστών αυτών. Είναι αξιοσημείωτο ότι τα edges του καινούργιου γράφου που δείχνουν τις φιλίες μεταξύ των χρηστών, φτιάχτηκαν με βάση τα edges του barabasi model. Αφού λοιπόν φτιάξαμε τον καινούργιο αυτόν γράφο, πήραμε για κάθε κόμβο (ή αλλιώς χρήστη) τους γείτονές του και μετρήσαμε τις συχνότητες του κάθε δίσκου (που έχουν οι γείτονες – φίλοι του) χωρίς να μετράμε τους δίσκους που έχει ήδη ο χρήστης.

(θ.) Έστω μια σειρά από users [10] που διαθέτουν κάποια χρηματικά ποσά και έχουν εκδηλώσει την βαθμονομημένη επιθυμία τους να αγοράσουν μια σειρά από δίσκους. Η επιθυμία εκδηλώνεται με έναν βαθμό απο 1 εως 5 για κάθε δίσκο και κάθε δίσκος έχει ένα χρηματικό κόστος [11]. Χρησιμοποιήστε έναν γενετικό αλγόριθμο για να επιλέξει τους δίσκους που θα προτείνει στους users να αγοράσουν με σκοπό να μεγιστοποιήσουν τον συνολικό βαθμό επιθυμίας των δίσκων, δεδομένου του διαθέσιμου χρηματικού ποσού που διαθέτει ο κάθε user. Συγκρίνετε τα αποτελέσματα με μια τυχαία επιλογή δίσκων. [διάλεξη genetic algorithms].

Για την υλοποίηση του παραπάνω ερωτήματος, υλοποιήσαμε τον δικό μας γενετικό αλγόριθμο, τον οποίο θα περιγράψουμε εδώ:

Στο αρχείο genetic.py έχουμε αρχικά την συνάρτηση generate genome, η οποία δημιουργεί ένα binary string από 0 και 1. Το 1 σημαίνει ύπαρξη του δίσκου, ενώ το 0 το αντίθετο. Ύστερα, η generate population δημιουργεί ένα πλήθος από genomes (εμείς επίσης δίνουμε ως παράμετρο τον αριθμό του πλήθους).

Ύστερα έχουμε το fitness function το οποίο ελέγχει για το κάθε genome αν έχει υπερβεί το όριο τιμής του χρήστη. Αν το έχει υπερβεί, επιστρέφουμε 0, αλλιώς επιστρέφουμε το άθροισμα των “επιθυμιών“ όλων των δίσκων (που έχει το genome), καθώς αυτό αποτελεί μία καλή αξιολόγηση του κάθε genome.

Έπειτα, η selection pair επιλέγει ένα τυχαίο συνδυασμό για τους γονείς της γενιάς. Είναι αξιοσημείωτο ότι έχουμε προσδιορίσει βάρη για να υπάρχει μεγαλύτερη πιθανότητα να επιλεχθούν οι “ισχυροί” γονείς. Τα βάρη αυτά τα υπολογίζουμε μέσω του fitness function και είναι ουσιαστικά η αξιολόγηση του κάθε genome της γενιάς. Αν αυτή η λίστα έχει μόνο 0, η συνάρτηση choices με προσδιορισμό των βάρων δεν δουλεύει, για αυτό επιλέγουμε δύο τυχαίους γονείς από το population.

Στην συνέχεια, η συνάρτηση crossover “παντρεύει” τους δύο γονείς και συγκεκριμένα βγάζει 2 παιδιά, τα οποία έχουν ένα (τυχαίο μέρος) του ενός γονιού και του άλλου. Αν χωρίσουμε τον πρώτο γονέα σε α και β και τον δεύτερο σε γ δ (τυχαία), τότε τα δύο παιδία θα είναι της μορφής: αδ το πρώτο και γβ το δεύτερο.

Επίσης, έχουμε και την συνάρτηση mutation, η οποία αλλάζει ένα bit του genome (by default) στο συμπληρωματικό του με πιθανότητα 0.5 (by default).

Επιπρόσθετα, η συνάρτηση run evolution είναι αυτή που παράγει το καλύτερο genome τρέχοντας για ένα προκαθορισμένο αριθμό γενεών. Είναι σημαντικό να σημειωθεί ότι στο “evolution loop” κρατάμε τους δύο ισχυρότερους γονείς και στην επόμενη γενιά (elitism). Ύστερα δοκιμάζουμε και για τον υπόλοιπο **αριθμό** των ζευγαριών και παράγουμε τα πιο ισχυρά παιδιά χρησιμοποιώντας τις συναρτήσεις που προαναφέρθηκαν (εξαγωγή γονεών από την γενιά – οι οποίοι είναι πολύ πιθανό να είναι οι καλύτεροι, ύστερα γίνεται το crossover των γονέων για την παραγωγή 2 παιδιών και τέλος το mutation του κάθε παιδιού). Τα παιδιά της διαδικασίας αυτής τα αποθηκεύουμε στη λίστα next generation, η οποία αποτελεί το population της επόμενης γενιάς. Η επανάληψη αυτή γίνεται όσες φορές έχουμε ορίσει πως θα είναι ο αριθμός των γενεών.
Μόλις φτάσουμε στην τελευταία γενιά, γίνεται ένα reverse sort της λίστας του population (της γενιάς) με βάση το fitness function, με αποτέλεσμα στην πρώτη θέση της λίστας να βρίσκεται το ισχυρότερο genome όλων των γενεών.

Τέλος με τα κατάλληλα queries στη βάση παίρνουμε τα διαθέσιμα χρήματα του κάθε χρήστη (για να ορίσουμε το price limit), την επιθυμία του χρήστη να αγοράσει το δίσκο (σύμφωνα με το attribute “want” του τραπεζιού “user_wants_discs”) για να ορίσουμε το “βάρος” του κάθε δίσκου (όσο μεγαλύτερο, τόσο το καλύτερο). Επίσης, παίρνουμε και το πιο **πρόσφατο** price του δίσκου (έτσι ώστε να γνωρίζουμε πότε έχουμε ξεπεράσει το price limit) και το όνομα και μπάντα του δίσκου. Τις τιμές αυτές τις έχουμε αρχικοποιήσει από πιο πριν κατά την δημιουργία της βάσης (επίσης, έχουμε φροντίσει ο χρήστης να μην έχει τους δίσκους που επιθυμεί). Έχοντας ύστερα λάβει υπόψη ότι το κάθε bit του genome δείχνει την απουσία (0) ή την παρουσία του δίσκου (1), ψάχνουμε για τον κάθε χρήστη το κατάλληλο συνδυασμό δίσκων έτσι ώστε για τα χρήματα που έχει να έχει τη μέγιστη “επιθυμία” από τους δίσκους που θέλει (τα αποτελέσματα με το ποιους δίσκους προτείνουμε τελικά από αυτό το knapsack πρόβλημα βρίσκονται στη τραπέζι “user_rec_discs_knapsack”).
Είναι επίσης σημαντικό να σημειωθεί ότι αν ο χρήστης έχει λιγότερα χρήματα από τον πιο φθηνό δίσκο (από αυτούς που θέλει) τότε δεν χρειάζεται να τρέξουμε το genetic algorithm (αφού δεν μπορεί να αγοράσει κανένα δίσκο) και έτσι δεν του προτείνουμε κανένα δίσκο. Το αντίθετο κάνουμε αν ο χρήστης έχει παραπάνω ή ίσα χρήματα από το άθροισμα των τιμών όλων των επιθυμητών δίσκων (τότε του επιστρέφουμε όλους τους δίσκους που θέλει). Με αυτόν τον τρόπο “γλιτώνουμε” πράξεις που θα μας έβγαζαν στο ίδιο αποτέλεσμα.

Ακολουθεί σύγκριση του αποτελέσματος με μία τυχαία επιλογή από τους επιθυμητούς δίσκους του χρήστη (απλώς δημιουργήσαμε ένα “μη εξελιγμένο” genome και αντιστοιχίσαμε όπου 0 δεν υπάρχει ο δίσκος και όπου 1 υπάρχει):

```
❯ python3 genetic.py
Genetic KnapSack Result:
User btonnesen6 with money 321.0 achieved want level 31 with cost 290.5559474213929.
=========
Random Result:
User btonnesen6 with money 321.0 achieved want level 41 with cost 456.58791737647454.
=========
```

Παρατηρούμε ότι το τυχαίο genome ξεπέρασε κατά πολύ το price limit του χρήστη (είναι invalid).

Ξανατρέχουμε τη σύγκριση:

```
❯ python3 genetic.py
Genetic KnapSack Result:
User btonnesen6 with money 321.0 achieved want level 31 with cost 290.5559474213929.
=========
Random Result:
User btonnesen6 with money 321.0 achieved want level 30 with cost 332.0639399101633.
=========
```

Αυτή τη φορά το rank των επιλογών είναι χειρότερο από αυτό που βρήκαμε με το genetic algorithm και το price πάλι πιο πάνω από αυτό που διαθέτει ο χρήστης.

Το συμπέρασμα είναι ότι ο γενετικός αλγόριθμος βγάζει πολύ καλύτερα αποτελέσματα από τον τυχαίο αλγόριθμο που χρησιμοποιήθηκε.

Flask Api

Για την εργασία ετοιμάσαμε και ένα api σε flask, στο οποίο ο χρήστης μπορεί να ανακτήσει δεδομένα της βάσης από το web scraping, τις ερωτήσεις με τα data features και άλλες χρήσιμες πληροφορίες. Σε κάποια endpoints απαιτείται και basic authorization με τα στοιχεία του χρήστη (username & password) για να έχει πρόσβαση στα δεδομένα (πχ για το endpoint /discs απαιτείται authorization και επιστρέφει τους δίσκους που έχει ο logged in χρήστης). Είναι αξιοσημείωτο ότι οι κωδικοί μένουν encrypted στη βάση και το encryption γίνεται με βάση το SECRET_KEY που έχει οριστεί στο .env αρχείο. Ακολουθούν οδηγίες χρήσεις με τα διαθέσιμα api endpoints με το τι επιστρέφουν και αν χρειάζονται authorization. Το flask τρέχει στο localhost:5000 και μπορείτε να το δοκιμάσετε στο postman βάζοντας και τα κατάλληλα authorization (αν απαιτείται).

Api Endpoint	Returns	Authorization
/discs	Επιστρέφει τους δίσκους του logged in χρήστη.	Ναι
/bands	Επιστρέφει τις μπάντες που ακούει ο logged in χρήστης.	Ναι
/recommend	Επιστρέφει τους recommended δίσκους στον logged in χρήστη.	Ναι
/friends	Επιστρέφεις όλους τους φίλους του logged in χρήστη (και τα στοιχεία τους, χωρίς το password).	Ναι
/friends/{username}	Επιστρέφεις τον ζητούμενο φίλο του logged in χρήστη (και τα στοιχεία τους, χωρίς το password).	Ναι
/discs/friends	Επιστρέφει όλους τους δίσκους των φίλων του logged in χρήστη.	Ναι

/discs/friends/{username}	Επιστρέφει όλους τους δίσκους του ζητούμενου φίλου του logged in χρήστη.	Ναι
/bands/friends	Επιστρέφει τις μπάντες που ακούνε όλοι οι φίλοι του logged in χρήστη.	Ναι
/bands/friends/{username}	Επιστρέφει τις μπάντες που ακούει ο ζητούμενος φίλος του logged in χρήστης.	Ναι
/price/{disc_name}/history	Επιστρέφει όλες τις διαθέσιμες τιμές από το web scraping του δίσκου.	Ναι
info/discs/{disc_name}	Επιστρέφει την πιο πρόσφατη τιμή του ζητούμενου δίσκου, μαζί με άλλες πληροφορίες όπως για την μπάντα του δίσκου.	Όχι
info/bands/{band_name}	Επιστρέφει πληροφορίες για την ζητούμενη μπάντα.	Όχι
/stats/average/age/{band_name}	Επιστρέφει τον μέσο όρο ηλικίας των ακροατών μιας μπάντας (Q1).	Ναι
/stats/average/discs	Επιστρέφει τον μέσο όρο δίσκων που έχουν οι χρήστες (Q6).	Ναι
/stats/countries/mostmusic	Επιστρέφει τις χώρες και πόσα άτομα ακούνε μουσική από αυτές (Q2).	Ναι
/stats/bandcountry/{band_name}	Επιστρέφει την καταγωγή των περισσότερων χρηστών που ακούνε τη ζητούμενη μπάντα (Q3).	Ναι
/stats/mostgender/{band_name}	Επιστρέφει το φύλο που έχουν οι περισσότεροι ακροατές της ζητούμενης μπάντας (Q4).	Ναι
/stats/mostband	Επιστρέφει τη μπάντα με την μεγαλύτερη ακροαματικότητα (Q5).	Ναι
/stats/bands/heritage/{band_name}	Επιστρέφει την καταγωγή των χρηστών που ακούνε μία μπάντα (και πόσοι είναι	Ναι

	– Q7).	
/stats/discs/heritage/{disc_name}	Επιστρέφει την καταγωγή των χρηστών που έχουν τον ζητούμενο δίσκο (Q8).	Ναι
/stats/discs/usercount/{disc_name}	Επιστρέφει το πλήθος των χρηστών που έχουν ένα δίσκο (Q9).	Ναι
/stats/discs/mostgender/{disc_name}	Επιστρέφει το φύλο που ακούει περισσότερο τον ζητούμενο δίσκο (Q10).	Ναι
/stats/bands/mostbands	Επιστρέφει τα συγκροτήματα που ακούνε οι χρήστες ανά χώρα (μαζί με το πλήθος των χρηστών αυτών – Q11).	Ναι
/stats/topdiscs/{x}	Επιστρέφει μία λίστα με τους top x δίσκους (σε ποσότητα – Q12).	Ναι

Σημαντικές παρατηρήσεις:

- Αν στο όνομα υπάρχουν κενά, αντικαταστήστε τα με ‘-’. Για παράδειγμα, ο δίσκος των coldplay “Ode To Deodorant”, θα γραφτεί “ode-to-deodorant”.

Web Scraping

Για το web scraping χρησιμοποιήσαμε ένα συνδυασμό των Frameworks Selenium και BeautifulSoup (τον οποίο εξηγούμε σε αυτή την παρουσίαση).

Github Repository:

https://github.com/manouslinard/music-recommender

Εκτέλεση Κώδικα:

Για να τρέξετε τον κώδικα (αφού έχει γίνει σωστά το configuration του .env file) μπορείτε να τρέξετε την παρακάτω εντολή (στο το φάκελο του repository):

```
python3 reset_db.py
```

Η εντολή αυτή γεμίζει τη βάση δεδομένων με όλα τις απαντήσεις στα προηγούμενα ερωτήματα.

Για να δείτε τις γραφικές παραστάσεις, τρέξτε την ακόλουθη εντολή:

```
python3 stats.py
```

By default το αρχείο αυτό “ζωγραφίζει” τα time series. Για να δείτε και άλλα γραφήματα ή απαντήσεις στα ερωτήματα των data features, μπορείτε να κάνετε uncomment τα επιθυμητά ερωτήματα στη main του stats.py.

Video Screencast:

Μπορείτε να δείτε ένα αναλυτικό screencast του κώδικα εδώ.