SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Filip Novački

Analiza vulgarnosti u Tarantinovim filmovima

PROJEKT

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Filip Novački

Matični broj: 44531/15-R

Studij: Informatika u obrazovanju

Analiza vulgarnosti u Tarantinovim filmovima

Projekt

Mentor:

Prof. dr. sc. Kornelije Rabuzin

Varaždin, lipanj 2020.

Filip Novački

Izjava o izvornosti

Izjavljujem da je moj projekt izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Rad opisuje nastajanje skladišta podataka na primjeru vulgarnosti u Tarantinovim filmovima. Obrada podataka i prikaz bazira se na Pythonu, a baza i skladište napravljeni su u SQLiteu. SQLite baza popunjava se preko API-ja koji je napravljen za Python. Osim upravljanja podatcima opisuje se i način transformacije iz modela baze podataka pomoću ERA modela u model skladišta podataka (zvijezda dijagram) također pomoću ERA modela. Na kraju je napravljena analiza podataka po raznim parametrima koji nam pomažu donijeti zaključke o filmovima s obzirom na vulgarnosti u njima. Primjer podataka je malen u odnosu na baze i skladišta koji se mogu naći u *stvarnom* svijetu, no pokazani primjeri se jednako mogu primijeniti na velikim bazama i skladištima kao i na ovom malom primjeru. Stoga je važno pridržavati se takvih preporuka unatoč tome što na sami primjer te paradigme nemaju tolikog utjecaja.

Ključne riječi: skladište podataka, baza podataka, sqlite, data warehouse, model zvijezde, csv, python, matplotlib

Sadržaj

1. Uvod

Ovo je projektni rad za kolegij Skladišta podataka i poslovna inteligencija. Cilj projekta je pokazati manipulaciju podatcima u duhu skladišta podataka. Dakle, obuhvaća korake:

- Modeliranje baze podataka
- Stvaranje baze podataka
- Uvoz podataka iz CSV datoteke
- Modeliranje skladišta podataka
- Stvaranje skladišta podataka
- Stvaranje zaključaka o podatcima iz skladišta

Prve tri točke su ovdje tehničke naravi te su potrebne kako bi se stvorila početna točka za rad sa skladištem. Točka modeliranje skladišta podataka odnosi se na stvaranje modela zvijezde, točka stvaranje skladišta popunjavanje je tog istog modela podatcima i točka stvaranje zaključaka odnosi se na crtanje grafova temeljeno na podatcima iz skladišta.

Za obradu podataka korišten je programski jezik Python, a za pohranu u bazu `SQLite`. Obrada je napravljena u okruženju `Jupyter-notebook`. Baza je povezana pomoću biblioteke `sqlite3`. U obradi podataka korištena je biblioteka `pandas`, a za crtanje grafikona bibioteka `matplotlib`.

S obzirom na to da je domena podataka vrlo eksplicitna, čitatelja se moli za diskreciju kod čitanja i da ima na umu da se sve riječi iz domene koriste samo u svrhu obrade podataka i nikako ne u svrhu omalovažavanja bilo kojih skupina ili vrijeđanja.

Dio I

Izgradnja skladišta

2. Podatci i cilj

Domena ovog projekta su vulgarnosti u sedam Tarantinovih filmova. Podatci su izvučeni s https://github.com/fivethirtyeight/data/tree/master/tarantino, a njihova je struktura prikazana tablicom 1.

Tablica 1: Opis strukture podataka dobivene u CSV datoteci

Indeks	Naziv filma	Tip vulgarnosti	Riječ	Vrijeme
1	Reservoir Dogs	word	dick	0.40
2	Reservoir Dogs	word	dicks	0.43
3	Reservoir Dogs	word	fucked	0.55
...	...	...	...	...
50	Reservoir Dogs	word	bullshit	6.81
...	...	...	...	...
952	Kill Bill: Vol. 1	death	-	37.65
953	Kill Bill: Vol. 1	death	-	39.57

Dakle, prvi stupac je indeks koji nije dostupan u CSV datoteci, nego je naknadno dodan u strukturi pri obradi jer ne postoji pravi primarni ključ u već zadanim podatcima. Bilo je razmatranja da se za PK uzme dvokomponentni primarni ključ između atributa vrijeme i naziv filma, no to nam ne garantira jedinstvenost jer je moguće da se više psovki dogodi u isto vrijeme.

Atribut `Naziv filma` nam govori kako se zove film u kojem se dogodila neka vulgarnost. `Tip vulgarnosti` je atribut koji označava na koji je način neka vulgarnost iskazana. Pojavljuju se samo dvije opcije: *word* i *death*. U atributu `riječ` zapisana je informacija o izrečenoj psovki. Kad je tip vulgarnosti *death*, atribut `riječ` iznosi `NULL`. Atribut `Vrijeme` označava minutu u kojem se neka vulgarnost dogodila. Minute su zapisane u decimalnom zapisu (6.81 je šesta minuta i četrdeset i deveta sekunda)

U obradi podataka se u rubriku riječ upisivalo *death* ukoliko se radi o smrti iz jednostavnosti čitanja podataka, a riječ *death* u engleskom jeziku nije vulgarna.

Osim odabranog CSV-a uneseni su još i dodatni podatci o filmovima: godina, trajanje i ocjena. Podatci su preuzeti s IMDb-a.

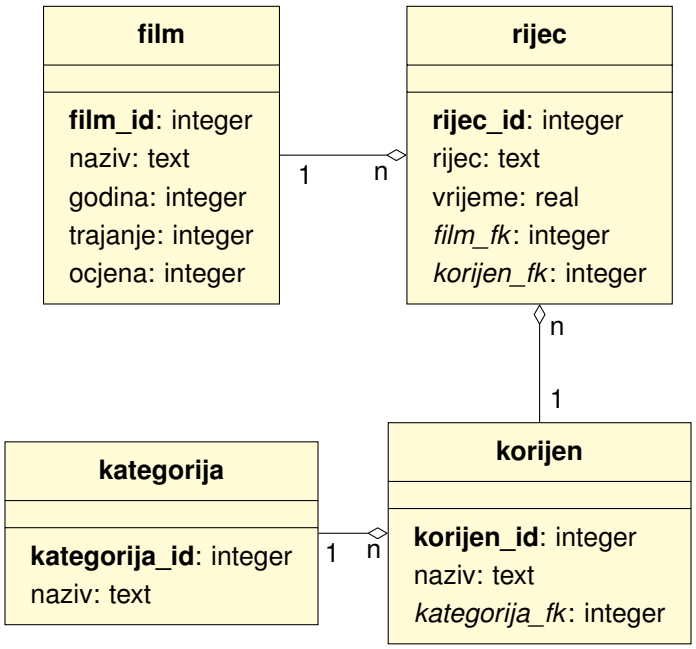
Cilj projekta je iz ovih podataka zaključiti kako psovke utječu na uspjeh filma, postoji li korelacija u godinama i količini psovki, kako su vulgarnosti raspoređene po filmu, zastupljenost određenih psovki itd.

3. Baza podataka

U ovom poglavlju opisat će se kako se modelirala i stvarala baza podataka. Unatoč tome što je vrlo jednostavna, postojat će razlika između baze i skladišta, a takve promjene na puno većim primjerima značile bi i veću razliku u performansama.

3.1. ERA model

ERA model koji se koristio za bazu podataka prikazan je na slici 1.



Slika 1: ERA model baze podataka

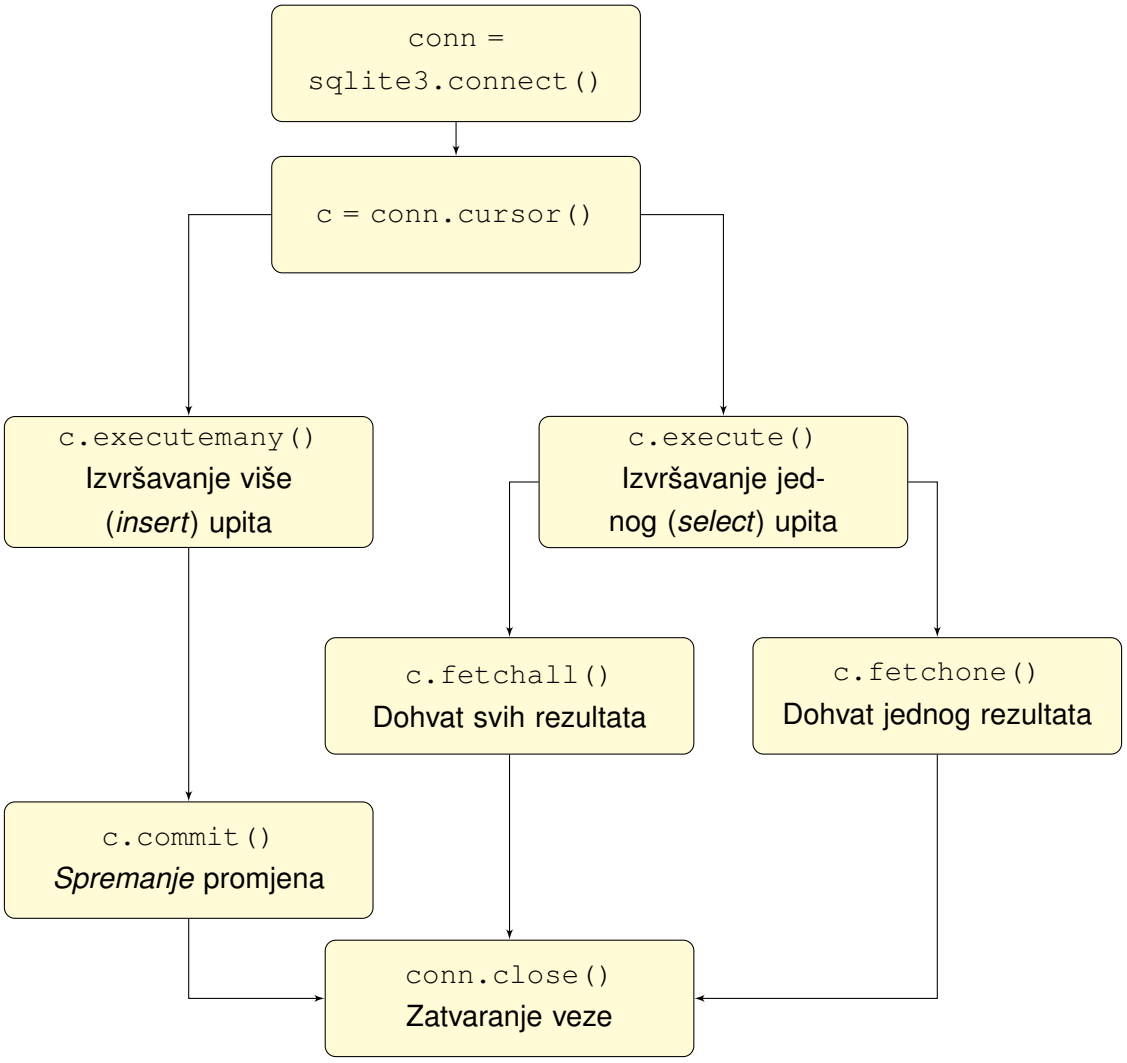
3.2. Kreiranje baze

Baza je dalje kreirana ručno upitima koristeći `sqlite3` biblioteku (API) u Pythonu. Brisanje tablice uvijek se nalazi na početku ćelija jer kod izvršavanja ćelije bilo bi puno grešaka kad bi se pisalo po već postojećoj tablici. U realnosti nam `DROP TABLE` naredba ne treba na tom mjestu.

Cijela se baza kreira *ručno*, odnosno bez drugog softvera osim biblioteke `sqlite3`. Sažetak rada te biblioteke prikazan je na slici 2. Rad s bazom ne razlikuje se od onoga što je za očekivati od ostalih alata za rukovanje bazom. Prvo je potrebno otvoriti bazu i napraviti kursor.

```
import sqlite3
```

```
conn = sqlite3.connect('baza.db')
c = conn.cursor()
```



Slika 2: Kratki sažetak funkcija biblioteke `sqlite3` za komunikaciju između baze

Ono što se razlikuje od izvršavanja upita direktno u bazi je povezanost upita sa strukturama u Pythonu. `sqlite3` biblioteka omogućuje unos velike količine podataka funkcijom `executemany()`. To se radi tako da se prvo pripremi lista n-torki s podatcima koji se žele unijeti. Nakon toga se na željenim mjestima u upitu umjesto vrijednosti napišu upitnici imajući na umu da će te upitnike zamijeniti vrijednosti iz n-torki onim redoslijedom kojim su navedene. Takav se upit skupa s podatcima proslijedi funkciji `executemany()` te će ti podatci biti uneseni u bazu. Kako bi se podatci spremili potrebno je na kraju pozvati i `commit()` funkciju.

S druge strane, za izvršavanje pojedinačnih upita (najčešće `SELECT`, katkada i `INSERT`) koristi se funkcija `execute()`. Ta funkcija vraća strukturu svih odgovora baze kroz koje se može iterirati. U ovu svrhu to je osobito praktično jer kad ćemo dohvaćati podatke iz baze morat ćemo ih *cratati* pomoću `matplotlib` biblioteke kojoj se upravo te liste ili drugi *iterables* daju na crtanje.

3.2.1. Tablica `korijen`

Tablica korijen posprema korijene riječi koje su popisane u bazi podataka. To nam omogućava grupiranje riječi bez obzira na to u kojem su ubliku, npr. sa sufiksom *-er*, *-ed* i sl. Tablica korijen stvorena je kodom:

```
drop_table_korijen = '''
DROP TABLE korijen;
'''
c.execute(drop_table_korijen)

create_table_korijen='''
CREATE TABLE korijen(
    korijen_id INTEGER PRIMARY KEY AUTOINCREMENT,
    naziv TEXT
);
'''

c.execute(create_table_korijen)

insert_into_korijen = '''
INSERT INTO korijen(naziv) VALUES (?);
'''
c.executemany(insert_into_korijen, korijen_db)
conn.commit()
```

Po stvaranju tablice u bazu se ubacuju korijeni riječi unaprijed pripremljeni u strukturi iza varijable `korijen_db`.

3.2.2. Tablica `kategorija`

Tablica `kategorija` u sebi sadrži kategorije riječi, odnosno na kojoj osnovi vulgarnosti vrijeđaju. Neke od kategorija su seks, rasa, nacija itd. Kod traženja značenja koristio se rječnik sleng riječi www.urbandictionary.com

```
drop_table_kat = '''
DROP TABLE kategorija;
'''
c.execute(drop_table_kat)

create_table_kat = '''
CREATE TABLE kategorija(
    kategorija_id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
    naziv TEXT
);'''

c.execute(create_table_kat)
insert_into_kat = '''
INSERT INTO kategorija(naziv) VALUES (?);
'''
c.executemany(insert_into_kat, kategorije_db)
conn.commit()
```

3.2.3. Tablica `film`

Tablica film sadrži podatke o filmovima. Osim imena filmova koji su zadani u CSV-u, podatci su nadopunjeni godinama izdavanja, trajanjem i ocjenom.

```
drop_table_film = '''DROP TABLE film;'''
c.execute(drop_table_film)

create_table_film = '''
CREATE TABLE film(
    film_id INTEGER PRIMARY KEY AUTOINCREMENT,
    naziv TEXT NOT NULL,
    godina INTEGER,
    trajanje INTEGER,
    ocjena INTEGER
    CHECK (ocjena <= 100 AND
           ocjena >= 0)
);
'''
c.execute(create_table_film)

insert_into_film = '''
INSERT INTO film (naziv, ocjena, godina, trajanje)
VALUES (?, ?, ?, ?);
'''
c.executemany(insert_into_film, film)
conn.commit()
```

Ovdje možemo primijetiti kako se u upitu nalaze po četiri upitnika što znači da se u bazu ubacuje lista četvorki onim redom kako su posložene četvorke.

3.2.4. Tablica `riječ`

Tablica riječ je u ovom radu zapravo najvažnija tablica jer nosi suštinu podataka koji se mjere. Dakle, sadrži samo svoj sadržaj, `id` i vanjske ključeve na druge tablice.

```
drop_table_rijec ='''
DROP TABLE rijec;
'''
c.execute(drop_table_rijec)

create_table_rijec = '''
CREATE TABLE rijec (
    rijec_id INTEGER PRIMARY KEY,
    rijec TEXT NOT NULL,
    vrijeme REAL NOT NULL,
    film_fk INTEGER,
    korijen_fk INTEGER,
    kategorija_fk INTEGER
);
'''
c.execute(create_table_rijec)

insert_into_rijec = '''
INSERT INTO rijec VALUES (?,?,?,NULL,NULL,NULL)
'''
c.executemany(insert_into_rijec, [(a, data[a][2], data[a][3]) for␣
 ↪a in data])
conn.commit()
```

Osim same tablice potrebno je popuniti i vanjske ključeve.

```
update_korijen_fk = '''
UPDATE rijec SET korijen_fk = (
    SELECT korijen_id FROM korijen
    WHERE naziv = ?
) WHERE rijec = ?
'''
c.executemany(update_korijen_fk, sorted_vulgs_db)
conn.commit()
```

```
update_film_fk = '''
UPDATE rijec SET film_fk = (
    SELECT film_id FROM film
    WHERE naziv = ?
```

```
) WHERE rijec_id = ?
'''
film_id = [(data[a][0], a) for a in data]
c.executemany(update_film_fk, film_id)
conn.commit()
```

```
update_kategorija_fk = '''
UPDATE rijec SET kategorija_fk = (
    SELECT kategorija_id FROM kategorija
    WHERE naziv = ?
) WHERE korijen_fk = (
    SELECT korijen_id FROM korijen
    WHERE naziv = ?
);
'''
c.executemany(update_kategorija_fk, [(a[1], a[0]) for a in␣
 ↪korijen_kat])
conn.commit()
```

4. Skladište podataka

Skladište podataka koje će se raditi nad ovim podatcima bit će poprilično slično bazi podataka, ali će biti jedna vrlo značajna razlika, i to na jednoj vezi te će se tako oformiti i izgled zvijezda modela.

Kako bismo dobili izmodelirano skladište, potrebno je napraviti nekoliko preinaka u bazi. Te se preinake oslikavaju na performanse kod uzimanja podataka iz ogromnih skladišta jer su dizajnirani tako da brže rade u svrhu za koju su namijenjeni, a ne za umetanje podataka u bazu.

4.1. ETL proces

ETL proces obuhvaća obradu podataka i unos podataka u skladište kako bi se mogla raditi obrada podataka na učinkovit način. S obzirom na jenostavan skup podataka, ETL samo popravlja neke nedostatke koji su napravljeni u izvorima podataka.

Ovaj projekt napravljen je tako da mu kreirana baza bude funkcionalna i spremna za korištenje, a da se nad njom rade modifikacije te da ta nova baza (odnosno skladište) služe za stvaranje izvještaja. Stoga je prvi korak uvesti podatke u bazu podataka koju modeliramo prema ERA dijagramu na slici 1.

Iz podataka iz CSV datoteke izvlače se (odnosno odabiru se pojave) entiteta po atributima odnosno grupiraju se filmovi. Oni se filtriraju korištenjem Pythonove strukture `set` koja se ponaša kao skup u matematici - dopušta samo jedinstvene elemente, a dupliće izbacuje. Tako zapamćeni filmovi ubacuju se u bazu podataka uz ostale podatke koji su preuzeti s IMDb-a. S obzirom na to da su se podatci uveli u strukturu `pandas DataFrame`, bilo je jednostavno od toga napraviti skup:

```
filmovi = set(df.movie)
```

Iduće što nije korektno obavljeno u CSV datoteci je riječ koja predstavlja vulgarnost, odnosno u slučaju da vulgarnost nije verbalna, nego je riječ o smrti, onda riječ za taj unos ostaje prazan unos. Stoga je ovdje praktično namjerno ubaciti riječ *death* kako upiti za vulgarnostima bili što jednostavniji.

```
rijec = set(df.word)
rijec.add('death')
```

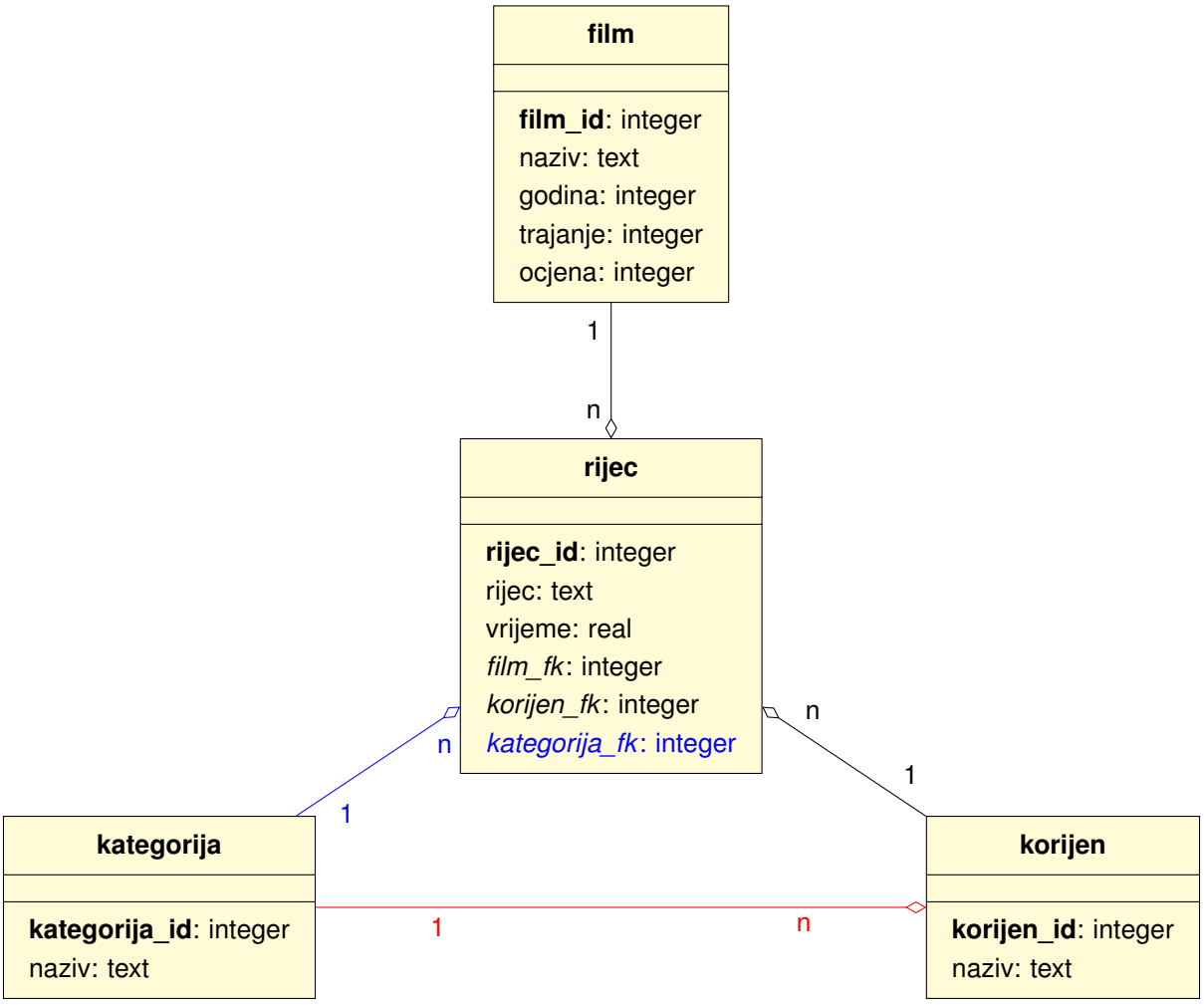
Osim podataka koji su nepraktično definirani u CSV-u potrebno je napraviti i preinake kod transformacije iz baze u skladište. Prvo je potrebno dodati vanjski ključ u relaciju `rijec`, a *izbaciti* ga iz relacije `korijen`. Nakon toga uz jedan snažan upit potrebno je pronaći kojoj kategoriji pripadaju koje riječi prema onome što smo imali zapisano u relaciji `korijen`. Nakon toga potrebno je samo obrisati atribut u relaciji `korijen` kako ne bismo imali iste vanjske ključeve u dvije tablice. ERA model će detaljnije biti opisanu idućem poglavlju.

Tablica 2: Kratki prikaz promjene podataka u slučaju neverbalne vulgarnosti

Indeks	Naziv filma	Tip vulgarnosti	Riječ	Vrijeme
952	Kill Bill: Vol. 1	death	-	37.65
953	Kill Bill: Vol. 1	death	-	39.57
952	Kill Bill: Vol. 1	death	death	37.65
953	Kill Bill: Vol. 1	death	death	39.57

4.2. ERA model skladišta

Na slici 3 prikazan je ERA model skladišta. Kod stvaranja skladišta temeljenog na bazi potrebno je samo vanjske ključeve iz relacije `korijen` prebaciti na pripadajuća mjesta u relaciji `rijec`.



Slika 3: Zvijezda model skladišta podataka - plavom bojom označena veza dodana je u odnosu na ERA model baze, a crvenom bojom označena veza izbačena je u odnosu na ERA model baze

U kontekstu zvijezda dijagrama, relacija `rijec` je činjenica, a ostale relacije su dimenzije. Tome u prilog idu i svojstva tih relacija - relacija `rijec` nema mnogo atributa, ali zato ima

mnogo vanjskih ključeva. Ovdje se napravila iznimka jer entitet relacije `rijec` nema brojevnu vrijednost, ali to je zato što svaka riječ sebe broji jednom što i nema nekog smisla zapisivati u relaciju. Nadalje, relacija `film` ima mnogo atributa što je svojstveno za dimenzije u skladištima. Relacije `kategorija` i `korijen` nisu posve tipične za zvijezda dijagrame jer nemaju mnogo atributa, no to je u ovom slučaju više iznimka jer su te relacije takvog sadržaja da je teško dodatno opisati njihove entitete.

S druge strane, atribut `vrijeme` je onaj pomoću kojeg možemo agregirati u skupine unatoč tome što nema smisla zbrajati vrijednosti.

4.3. Metapodatci skladišta

Metapodatci su podatci koji opisuju podatke u skladištu, odnosno olakšavaju korištenje skladišta svim njegovim korisnicima.

Element	Sintaksa
naziv relacije	malim slovima, jedna riječ
primarni ključ	`{ime relacije}_id`
vanjski ključ	`{ime relacije}_fk`

Tablica 3: Opis sintaktičkih pravila elemenata u skladištu podataka

Rbr.	Relacija	Atribut	Format	Izvor
1	`rijec`	`rijec`	*string*	CSV
2	`rijec`	`vrijeme`	*real* (`{minute}.{decimala}`)[1]	CSV
3	`film`	`naziv`	*string*	CSV
4	`film`	`godina`	*integer* (godina izdavanja)	IMDb
5	`film`	`trajanje`	*integer* (u minutama)	IMDb
6	`film`	`ocjena`	*integer* ($0 <$ ocjena < 100)	IMDb
7	`korijen`	`naziv`	*string*	rječnik
8	`kategorija`	`naziv`	*string*	Urban Dictionary

Tablica 4: Opis sadržaja u relacijama

[1] Zapis minuta kakav je u podatcima nije uobičajen za čitanje ljudima, već je napravljen za jednostavan rad s računalom. Stoga minuta nije podijeljena na sekunde, nego su izražene samo minutama, a preciznosti radi su decimale napisane u dekadskom sustavu. Dakle, trenutak u trećoj minuti u tridesetoj sekundi piše se 2.50 jer je taj trenutak na pola minute, a ne 2.30 kako su ljudi navikli u životu.

Dio II

Stvaranje izvještaja

5. Izvještaji

Iz skladišta koje je definirano potrebno je na učinkovit način izvući podatke i manipulirati njima. Ovdje je bit osloniti se na funkciju `COUNT` iz SQL-a koja će brojati koliko se u nekom slučaju puta pojavljuje neka psovka.

Veza ocjene i broja psovki Na slici 4 cilj je zaključiti ima li količina psovki veze s uspješnosti filma (prema ocjeni). Iz podataka se može zaključiti da postoji blago pozitivan trend, no ne smije se smetnuti s uma da najbolji Tarantinov film, Pulp Fiction i najgori, Jackie Brown, imaju sličan broj psovki u prosjeku.

Upit za ovaj graf glasi:

```
avg_per_film = '''
SELECT  CAST (COUNT() as float)/film.trajanje, film.ocjena
FROM rijec, film
WHERE film_id = film_fk
GROUP BY film_fk
'''
rez = c.execute(avg_per_film).fetchall()
```

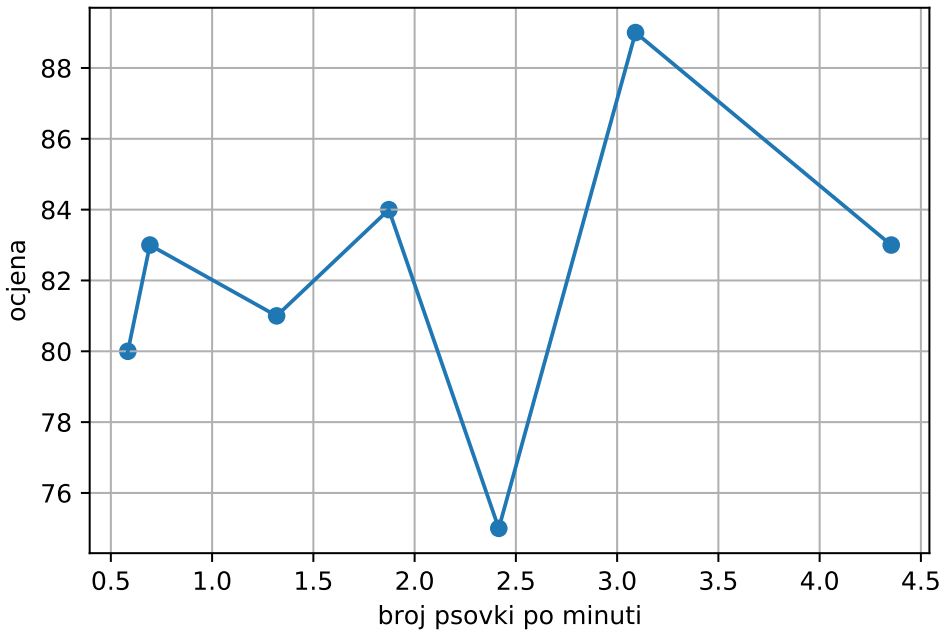
Nakon uspješnog upita podatci se sortiraju, prepakiravaju u listu te crtaju u graf.

```
podatci_avg = sorted(rez)
list(zip(*podatci_avg))
fig = plt.figure()
ax1 = fig.add_subplot(111)
ax1.set_ylabel('ocjena')
ax1.set_xlabel('broj psovki po minuti')

plt.grid(True)
plt.scatter(*zip(*podatci_avg))
plt.plot(*zip(*podatci_avg))
plt.savefig('rad/slike/psovke_po_minuti.pdf')
plt.show()
```

Distribucija vulgarnosti za vrijeme trajanja filma Idući zaključak kojeg bi bilo dobro donijeti je koja je distribucija vulgarnosti za vrijeme trajanja filma. Ovi izvještaji su napravljeni tako da je film vremenski podijeljen na trideset jednakih dijelova (neovisno o tome koliko traje, dakle dužem filmu će jedna tridesetima duže trajati nego kratkom) te su u tom razdoblju prebrojane psovke.

Za ovu svrhu se definira funkcija koja će davati graf s obzirom na to koje argumente damo. Funkcija se definira ovako:

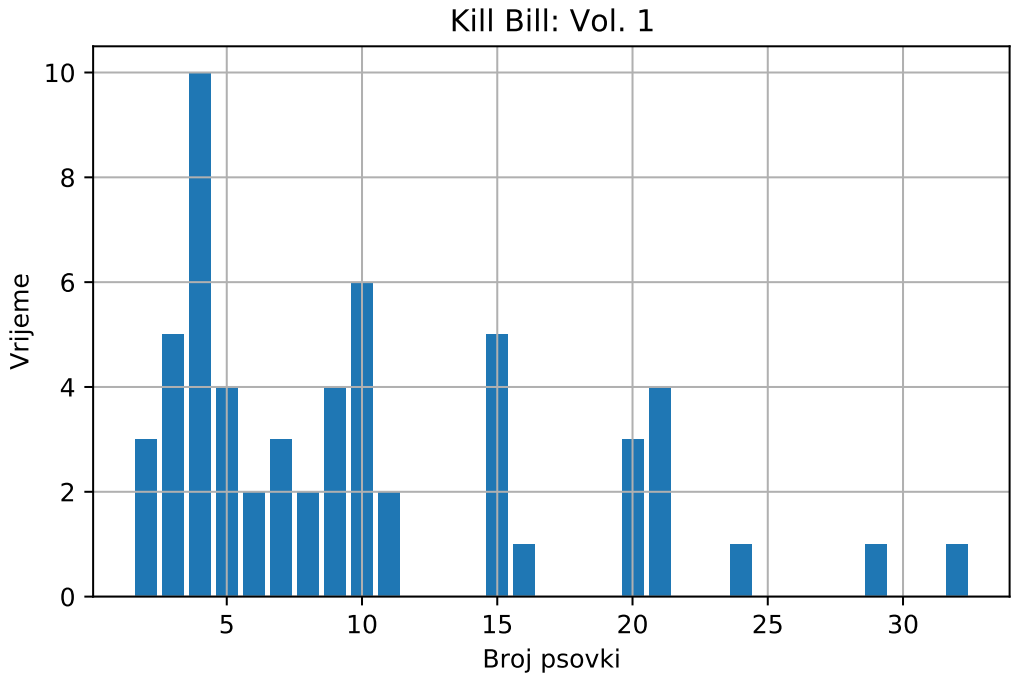


Slika 4: Usporedba ocjene s brojem psovki po minuti

```
def psovka_dio_filma(db_cursor, ime_filma, detail = 10, vulg =␣
↪'word'):
    vulg = '!' if vulg == 'word' else '='
    upit = '''
    SELECT CAST (rijec.vrijeme/film.trajanje*{0} as INTEGER) as␣
↪psovka_pc
    FROM film, rijec
    WHERE rijec.film_fk = film_id
        AND rijec.rijec {2}= ''
        AND film.naziv = '{1}'
    '''.format(detail, ime_filma, vulg)
    data = db_cursor.execute(upit).fetchall()
    fig.tight_layout()
    ax.grid(True)
    ax.bar(data.keys(), data.values())
    plt.savefig("rad/slike/" + ime_filma + "_" + vulg + ".pdf")
```

Jedan poziv ove funkcije izgleda:

```
psovka_dio_filma(c, 'Kill Bill: Vol. 2', detail = 20, vulg='word')
```



Slika 5: Prikaz distribucije psovki u filmu *Kill Bill: Vol. 1*

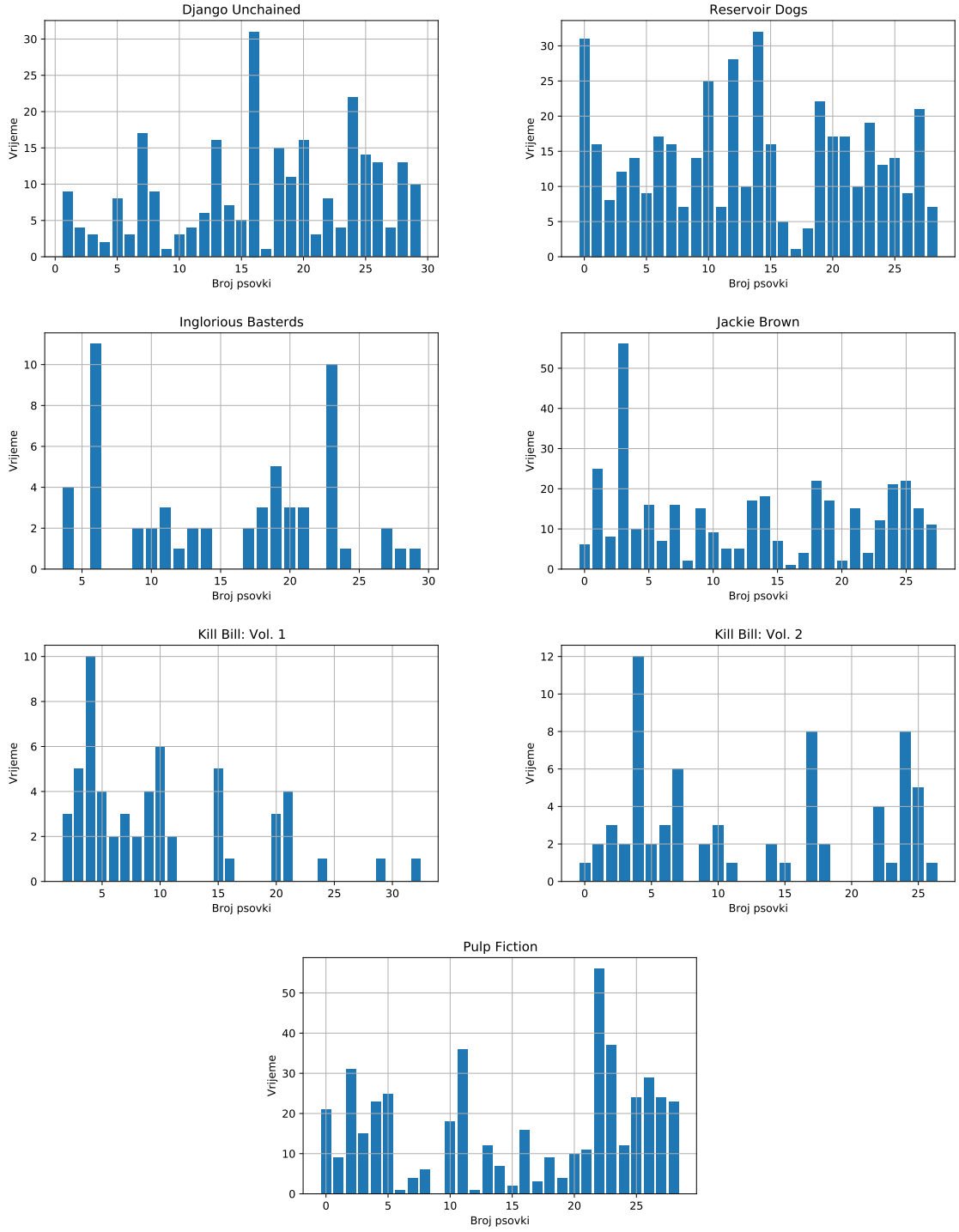
A istu funkciju možemo pustiti i kroz petlju i dobiti sliku za svaki film u stupčastom dijagramu (slika 6).

Iz dijagrama (slika 7) može se vidjeti koji filmovi imaju veću koncentraciju u kojem dijelu, odnosno kako su psovke distribuirane kroz film.

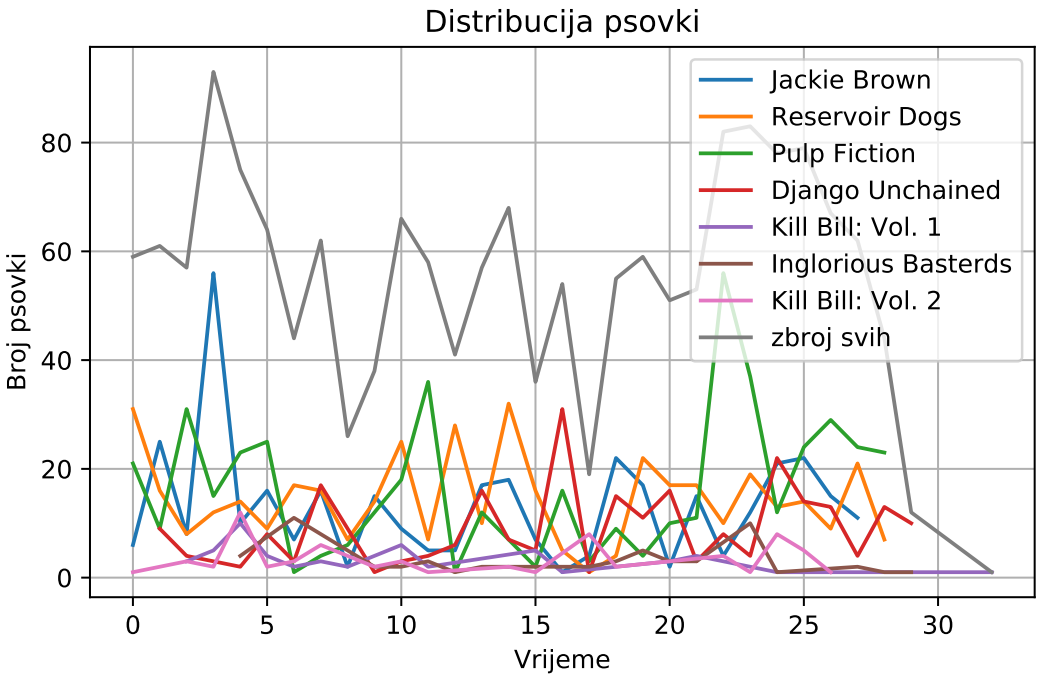
Količina psovki kroz godine Iduće što je korisno za analizirati je kako se broj psovki kretao kroz godine (slika 8. Za ovaj primjer također će se gledati prosjek po minuti, a ne apsolutan broj.

Kategorizacija psovki Iz odabranih filmova može se i pogledati koje su se kategorije psovki najčešće izgovarale. Psovke su kategorizirane prema temi na koju se odnose. Ovisno o tome koliko je ova podjela dobro napravljena, toliko će se lako moći isčitati rezultati o zastupljenosti pojedinih psovki stoga je dobra ideja detaljno proučiti značenje svakih od izraza kako bi se dobilo što jasnije rješenje.

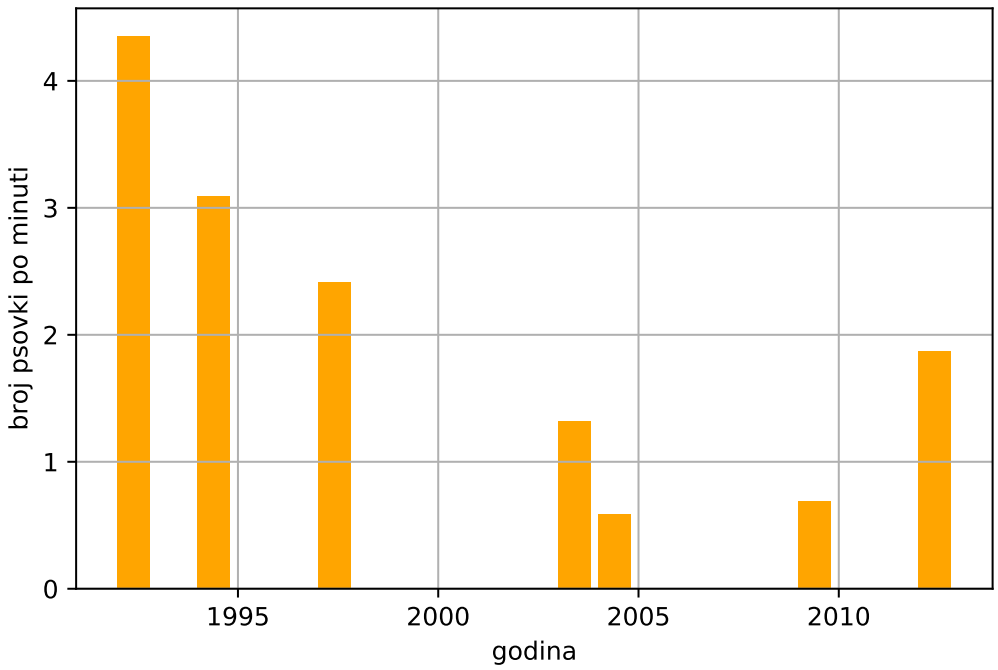
Ono što je također značajno za vidjeti je kako su koje psovke zastupljene po filmovima. Prema grafu na slici 10 vidi se da su u Pulp Fictionu najzastupljenije psovke vezane za seksualne teme, a rasistički se izrazi najviše koriste u Django Unchainedu.



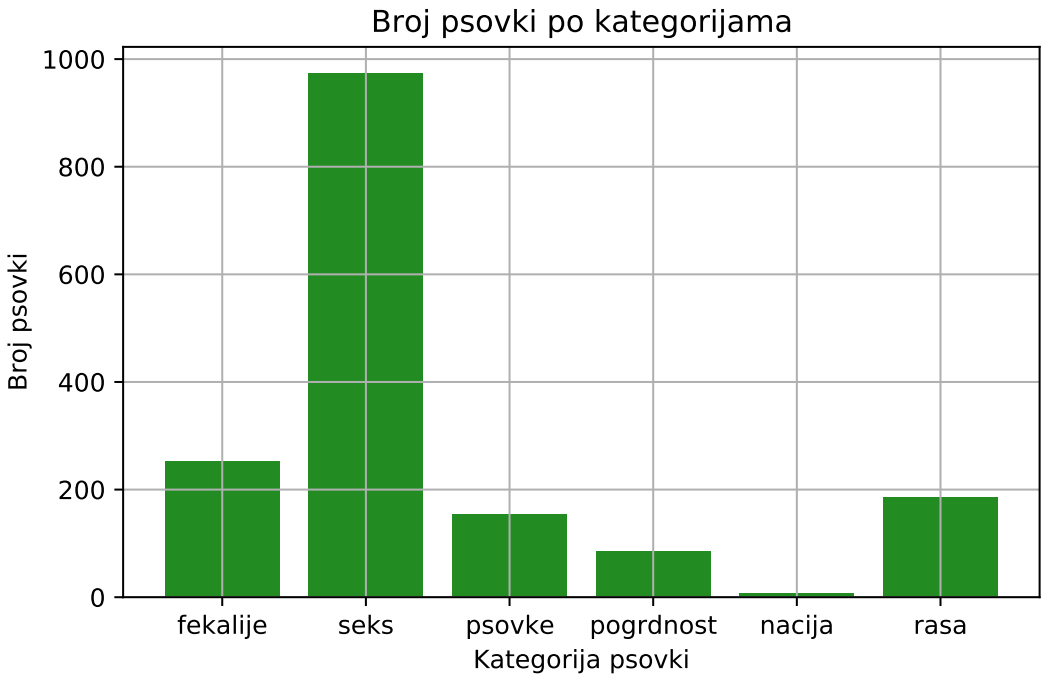
Slika 6: Prikaz distribucije psovki po filmovima



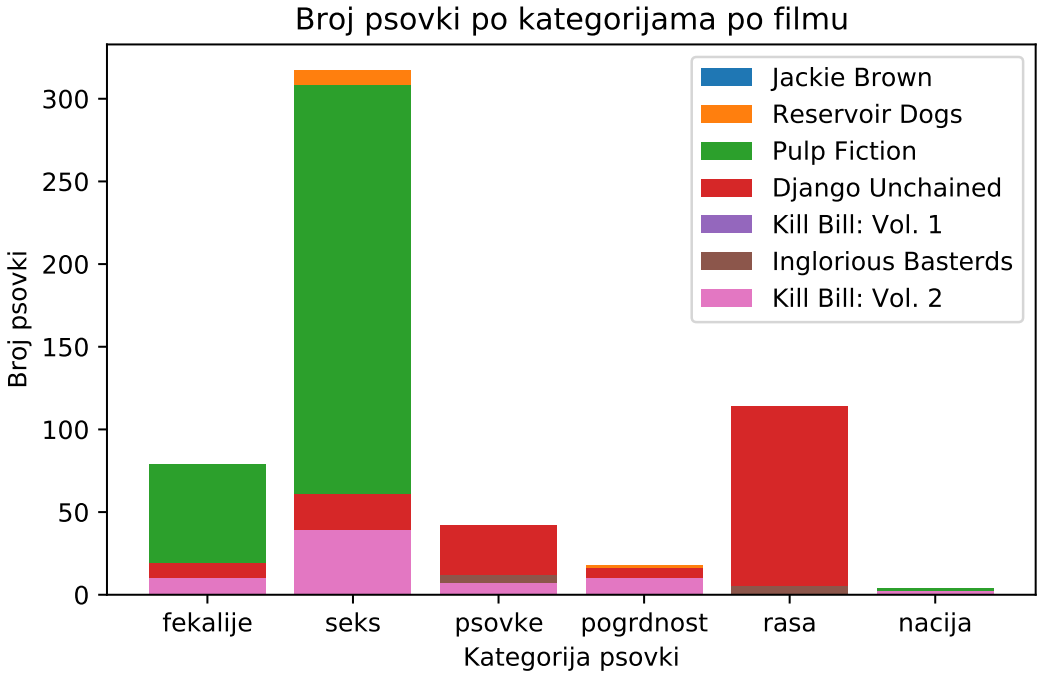
Slika 7: Prikaz distribucije psovki u svim filmovima



Slika 8: Trendovi psovki po godinama



Slika 9: Broj psovki po kategorijama



Slika 10: Broj psovki po kategorijama

6. Zaključak

Kroz ovaj projekt pokazale su se osnovne značajke skladišta podataka. Zbog vrlo malenog *dataseta* nije moguće u potpunosti demonstrirati sve funkcionalnost i prednosti skladišta nad bazama u svrhu izvještavanja, no vodeći se ovim primjerima nastala skladišta u puno većim razmjerima bi zasigurno pokazala bolje performanse i lakoću izrade izvještaja.

Ovaj projekt je također i pokazao kako se na vrlo jednostavan način mogu spojiti tehnologije koje djeluju na puno nižoj razini apstrakcije od, npr. *SQL Management Studija*, a bez mnogo kompliciranja u obradi podataka. To nam je omogućio Python kojemu je specijalnost jednostavno rukovanje podatcima na jednostavan, a opet učinkovit način.

Također, `SQLite` je jedinstven sustav za upravljanje bazom jer nema potrebu za serverom, odnosno sve je spremljeno u jednoj datoteci i tako se dobro nadovezuje na skup tehnologija koje su se koristile do tada. Jupyter Notebook također je odličan za ovaku primjenu jer omogućuje jednostavnu vizualizaciju svega što se crta te ima jednostavan pregled koda.

Sve u svemu, tehnologije koje su demonstrirane dobro služe ovoj svrsi i kad se koriste na kvalitetan način čine ekosustav za upravljanje podatcima na elegantan i efikasan način. Skladište je usredotočeno na brojenje riječi po određenim kriterijima te izvještaji omogućuju mjerenje količine i distribucije psovki po filmovima na puno načina.

A. Jupyter Notebook bilježnica s cijelim procesom

Tarantino profanity notebook

June 6, 2020

Dodatak - Jupyter Notebook iz kojeg su vađeni ekstrakti koda spominjani u radu.

1 Analiza vulgarnosti u Tarantinovim filmovima

Alati: Jupyter Notebook, Python (pandas, matplotlib i sl.), SQLite

[1]:
```
import pandas as pd
df = pd.read_csv('profanity.csv')
df
```

[1]:

	movie	type	word	minutes_in
0	Reservoir Dogs	word	dick	0.40
1	Reservoir Dogs	word	dicks	0.43
2	Reservoir Dogs	word	fucked	0.55
3	Reservoir Dogs	word	fucking	0.61
4	Reservoir Dogs	word	bullshit	0.61
...	...	...	...	...
1889	Jackie Brown	word	motherfucker	141.93
1890	Jackie Brown	word	ass	142.43
1891	Jackie Brown	word	fucking	142.47
1892	Jackie Brown	word	goddamn	142.97
1893	Jackie Brown	death	NaN	143.13

[1894 rows x 4 columns]

[2]:
```
data = {}
with open('profanity.csv') as file:
    for f in list(enumerate(file.readlines()[1:])):
        data[f[0]] = f[1:][0].rstrip().split(',')
```

[3]:
```
rijeci = set(df.word)
```

[5]:
```
filmovi = set(df.movie)
```

[6]:
```
film = [('Django Unchained', 84, 2012, 165),
 ('Pulp Fiction', 89, 1994, 154),
 ('Inglorious Basterds', 83, 2009, 153),
```

```
 ('Kill Bill: Vol. 1', 81, 2003, 91),
 ('Jackie Brown', 75, 1997, 154),
 ('Reservoir Dogs', 83, 1992, 99),
 ('Kill Bill: Vol. 2', 80, 2004, 137)]
```

[7]:
```
tip_vulgarnosti = set(df.type)
```

[8]:
```
rijec = set(df.word)
rijec.add('death')
```

[9]:
```
vulg = {}.fromkeys(rijec, [])
for entry in data:
    time = [[ entry, data[entry][3] ]]
    if data[entry][1] == 'word':
        vulg[data[entry][2]] = vulg[data[entry][2]] + time
    elif data[entry][1] == 'death':
        vulg['death'] = vulg['death'] + time
```

[10]:
```
sorted_vulgs = []
only_sorted = []
base_words = ['shit', 'fuck', 'ass', 'damn', 'dick', 'cock', 'bitch', 'cunt', ]
for word in vulg.keys():
    for base in base_words:
        if type(word) == type('') and base in word:
            sorted_vulgs.append((base, word))
            only_sorted.append(word)
```

[11]:
```
sorted_vulgs_db = sorted_vulgs + [(r,r) for r in rijec if r not in only_sorted]
sorted_vulgs_db[:5]
```

[11]:
```
[('fuck', 'motherfucker'),
 ('shit', 'horeshit'),
 ('damn', 'damned'),
 ('fuck', 'fuck'),
 ('fuck', 'fuckhead')]
```

[12]:
```
korijen = base_words + [r for r in rijec if r not in only_sorted]
```

[13]:
```
korijen_db = []
for item in korijen:
    korijen_db.append((item,))
```

[14]:
```
kategorije = ['fekalije', 'seks', 'psovke', 'pogrdnost', 'nacija', 'rasa',␣
 ↪'smrt']
kategorije_db = []
for item in kategorije:
    kategorije_db.append((item,))
```

[15]:
```
korijen_kat = [('shit', 'fekalije'),
('fuck', 'seks'),
('ass', 'seks'),
('damn', 'psovke'),
('dick', 'seks'),
('cock', 'seks'),
('bitch', 'pogrdnost'),
('cunt', 'seks'),
('slut', 'seks'),
('slope', 'rasa' ),
('jap', 'nacija'),
('pussy', 'seks'),
('death', 'smrt'),
('n-word ', 'rasa'),
('bastards', 'pogrdnost'),
('gooks', 'nacija'),
('hell', 'psovka'),
('jew (verb)', 'pogrdnost'),
('faggot', 'seks'),
('squaw', 'seks'),
('wetback', 'nacija'),
('merde', 'fekalije'),
('bastard', 'pogrdnost'),
('gook', 'nacija'),
('japs', 'nacija'),
('negro ', 'rasa')]
```

1.1 Upravljanje bazom podataka

[16]:
```
import sqlite3
```

[17]:
```
conn = sqlite3.connect('baza.db')
c = conn.cursor()
```

1.1.1 Tablica korijen

```
CREATE TABLE korijen(
    korijen_id INTEGER PRIMARY KEY AUTOINCREMENT,
    naziv TEXT
    kategorija_fk INTEGER   --- vraceno u korijen
    FOREIGN KEY (kategorija_fk)
        REFERENCES kategorija (kategorija_id)

);
```

```
INSERT INTO korijen(naziv) VALUES (?);
```

[18]:
```
drop_table_korijen = '''
DROP TABLE korijen;
'''
c.execute(drop_table_korijen)

create_table_korijen='''
CREATE TABLE korijen(
    korijen_id INTEGER PRIMARY KEY AUTOINCREMENT,
    naziv TEXT,
    kategorija_fk INTEGER,
    FOREIGN KEY (kategorija_fk)
        REFERENCES kategorija (kategorija_id)

);
'''

c.execute(create_table_korijen)

insert_into_korijen = '''
INSERT INTO korijen(naziv) VALUES (?);
'''
c.executemany(insert_into_korijen, korijen_db)
conn.commit()
```

1.1.2 Tablica kategorija

```
CREATE TABLE kategorija(
    kategorija_id INTEGER PRIMARY KEY AUTOINCREMENT,
    naziv TEXT
);

INSERT INTO kategorija(naziv) VALUES (?);
```

[19]:
```
drop_table_kat = '''
DROP TABLE kategorija;
'''
c.execute(drop_table_kat)

create_table_kat = '''
CREATE TABLE kategorija(
    kategorija_id INTEGER PRIMARY KEY AUTOINCREMENT,
    naziv TEXT
);'''

c.execute(create_table_kat)
insert_into_kat = '''
INSERT INTO kategorija(naziv) VALUES (?);
'''
```

```
c.executemany(insert_into_kat, kategorije_db)
conn.commit()
```

1.1.3 Tablica film

```
CREATE TABLE film(
    film_id INTEGER PRIMARY KEY AUTOINCREMENT,
    naziv TEXT NOT NULL,
    godina INTEGER,
    trajanje INTEGER, --- minute
    ocjena INTEGER, --- 0 <= ocjena <= 100
    CHECK (ocjena <= 100 AND
           ocjena >= 0)
);

INSERT INTO film (naziv, ocjena, godina) VALUES (?, ?, ?);
```

[20]:
```
drop_table_film = '''DROP TABLE film;'''
c.execute(drop_table_film)

create_table_film = '''
CREATE TABLE film(
    film_id INTEGER PRIMARY KEY AUTOINCREMENT,
    naziv TEXT NOT NULL,
    godina INTEGER,
    trajanje INTEGER,
    ocjena INTEGER
    CHECK (ocjena <= 100 AND
           ocjena >= 0)
);
'''
c.execute(create_table_film)

insert_into_film = '''
INSERT INTO film (naziv, ocjena, godina, trajanje) VALUES (?, ?, ?, ?);
'''
c.executemany(insert_into_film, film)
conn.commit()
```

1.1.4 Tablica rijec

```
CREATE TABLE rijec (
    rijec_id INTEGER PRIMARY KEY, ---non-autoincrement!
    rijec TEXT NOT NULL,
    vrijeme REAL NOT NULL,
    film_fk INTEGER,
    FOREIGN KEY (film_dk)
        REFERENCES film (film_id)
```

```
    korijen_fk INTEGER,
    FOREIGN KEY (korijen_fk)
        REFENCES korijen (korijen_id)
    kategorija_fk INTEGER   --- vraceno u korijen
    FOREIGN KEY (kategorija_fk)
        REFERENCES kategorija (kategorija_id)
);```


```sqlite
INSERT INTO rijec VALUES (?,?,?,NULL,NULL,NULL)
```

[21]:
```
drop_table_rijec ='''
DROP TABLE rijec;
'''
c.execute(drop_table_rijec)

create_table_rijec = '''
CREATE TABLE rijec (
    rijec_id INTEGER PRIMARY KEY,
    rijec TEXT NOT NULL,
    vrijeme REAL NOT NULL,
    film_fk INTEGER,
    korijen_fk INTEGER,
    kategorija_fk INTEGER,   --- vraceno u korijen
    FOREIGN KEY (kategorija_fk)
        REFERENCES kategorija (kategorija_id)

);
'''
c.execute(create_table_rijec)

insert_into_rijec = '''
INSERT INTO rijec VALUES (?,?,?,NULL,NULL,NULL)
'''
c.executemany(insert_into_rijec, [(a, data[a][2], data[a][3]) for a in data])
conn.commit()
```

```
UPDATE rijec SET korijen_fk = (
    SELECT korijen_id FROM korijen
    WHERE naziv = ?
) WHERE rijec = ?
```

[22]:
```
update_korijen_fk = '''
UPDATE rijec SET korijen_fk = (
    SELECT korijen_id FROM korijen
    WHERE naziv = ?
) WHERE rijec = ?
```
```

```
'''
c.executemany(update_korijen_fk, sorted_vulgs_db)
conn.commit()
```

```
UPDATE rijec SET film_fk = (
    SELECT film_id FROM film
    WHERE naziv = ?
) WHERE rijec_id = ?
```

[23]:
```
update_film_fk = '''
UPDATE rijec SET film_fk = (
    SELECT film_id FROM film
    WHERE naziv = ?
) WHERE rijec_id = ?
'''
film_id = [(data[a][0], a) for a in data]
c.executemany(update_film_fk, film_id)
conn.commit()
```

```
UPDATE rijec SET kategorija_fk = (
    SELECT kategorija_id FROM kategorija
    WHERE naziv = ?
) WHERE korijen = (
    SELECT korijen_id FROM korijen
    WHERE naziv = ?
);
```

[24]:
```
update_kategorija_fk = '''
UPDATE rijec SET kategorija_fk = (
    SELECT kategorija_id FROM kategorija
    WHERE naziv = ?
) WHERE korijen_fk = (
    SELECT korijen.korijen_id
    FROM korijen
    WHERE naziv = ?
);
'''
c.executemany(update_kategorija_fk, [(a[1], a[0]) for a in korijen_kat])
conn.commit()
```

1.1.5 Upiti

Ispitivanje postoji li vezanost između količine psovki i ocjene.

[25]:
```
import matplotlib as mpl
import matplotlib.pyplot as plt
```

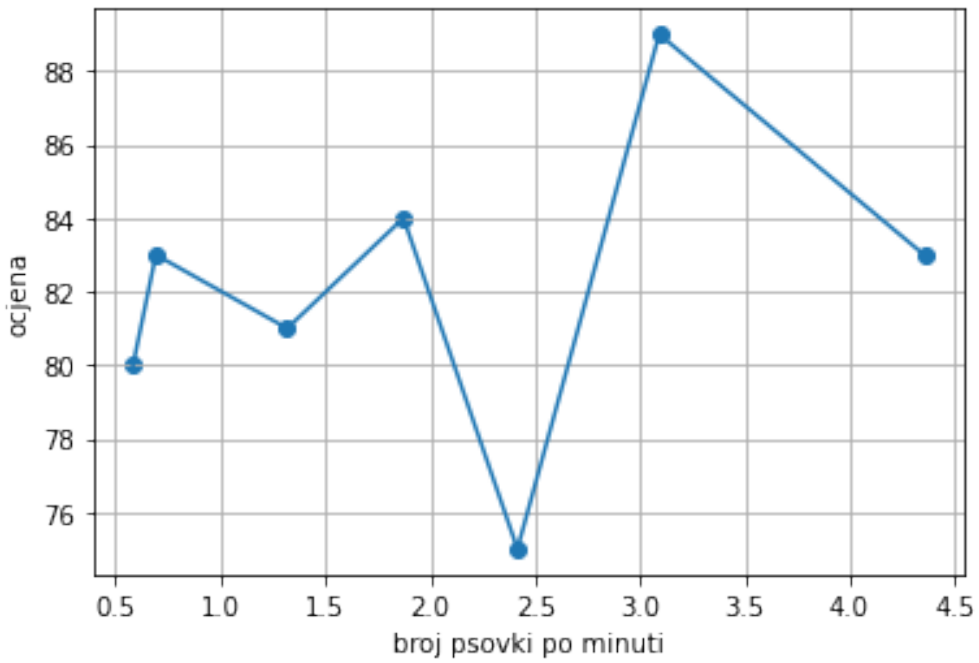
[27]:
```
#Ispitivanje postoji li veza izmedu gustoce psovki i ocjene.

avg_per_film = '''
SELECT  CAST (COUNT() as float)/film.trajanje, film.ocjena
FROM rijec, film
WHERE film_id = film_fk
GROUP BY film_fk
'''
rez = c.execute(avg_per_film).fetchall()

podatci_avg = sorted(rez)
list(zip(*podatci_avg))

fig = plt.figure()
ax1 = fig.add_subplot(111)
ax1.set_ylabel('ocjena')
ax1.set_xlabel('broj psovki po minuti')

plt.grid(True)
plt.scatter(*zip(*podatci_avg))
plt.plot(*zip(*podatci_avg))
plt.savefig('rad/slike/psovke_po_minuti.pdf')
plt.show()
```



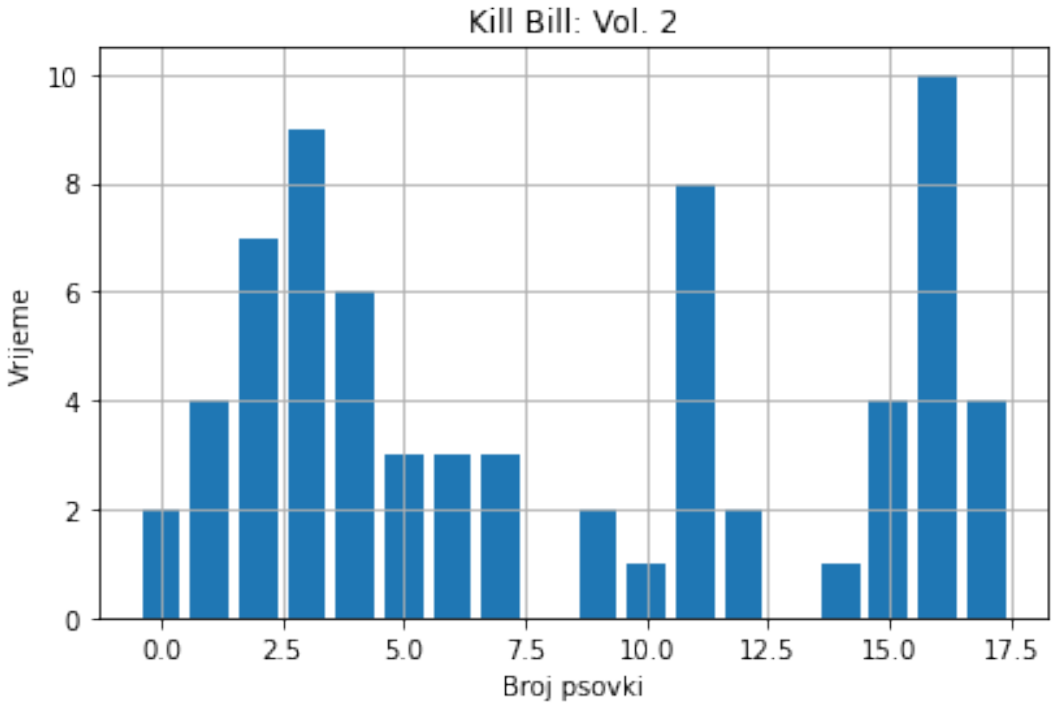
[28]:
```
from collections import Counter
```

[29]:
```
def psovka_dio_filma(db_cursor, ime_filma, detail = 10, vulg = 'word'):
    vulg = '!' if vulg == 'word' else '='
    upit = '''
    SELECT CAST (rijec.vrijeme/film.trajanje*{0} as INTEGER) as psovka_pc
    FROM film, rijec
    WHERE rijec.film_fk = film_id
        AND rijec.rijec {2}= ''
        AND film.naziv = '{1}'
    '''.format(detail, ime_filma, vulg)
    data = db_cursor.execute(upit).fetchall()

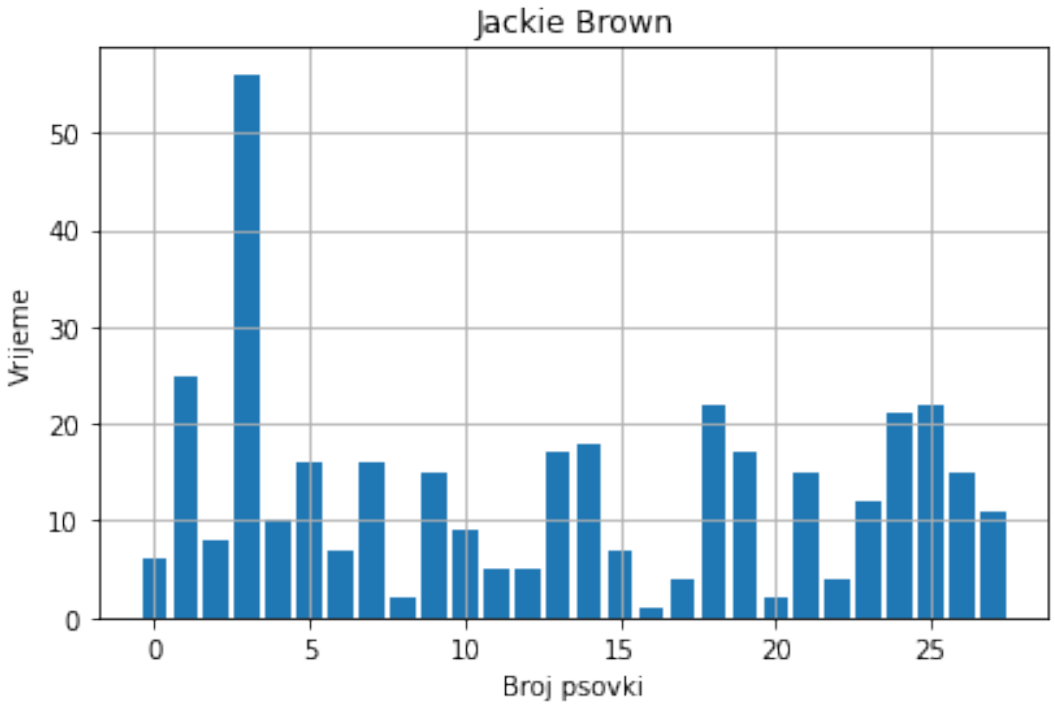
    data = list(x[0] for x in data)
    data = dict(Counter(data))
    fig, ax = plt.subplots()
    ax.set_title(ime_filma)
    ax.set_ylabel('Vrijeme')
    ax.set_xlabel('Broj psovki')

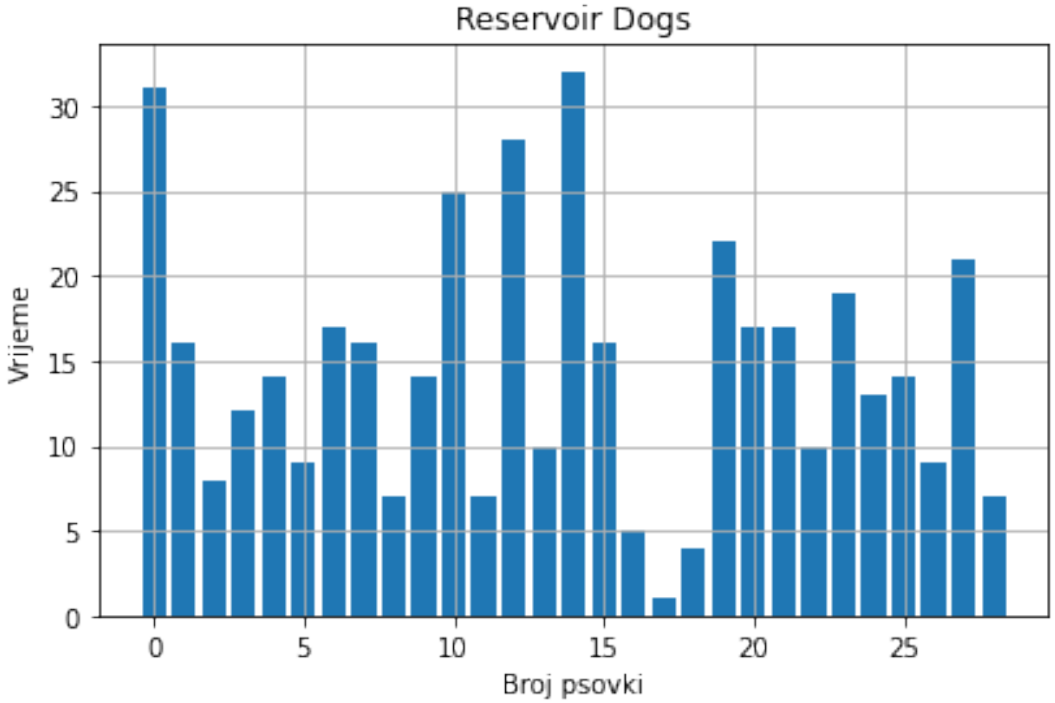

    fig.tight_layout()
    ax.grid(True)
    ax.bar(data.keys(), data.values())
    plt.savefig("rad/slike/" + ime_filma + "_" + vulg + ".pdf")
    plt.show()
```

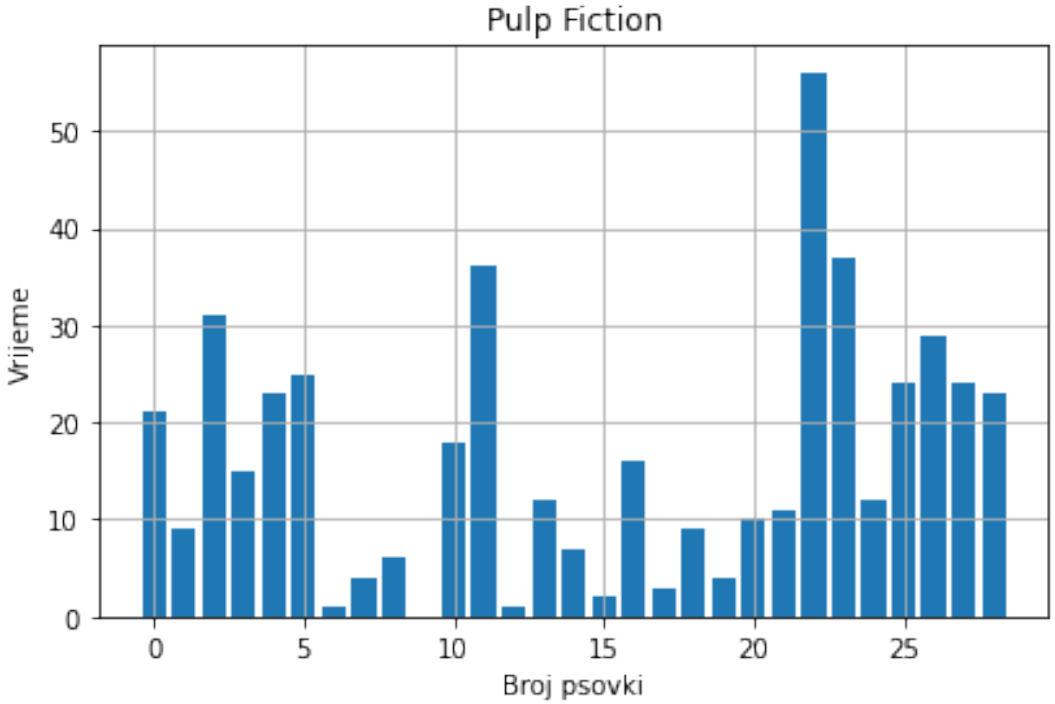
[30]:
```
psovka_dio_filma(c, 'Kill Bill: Vol. 2', detail = 20, vulg='word')
```

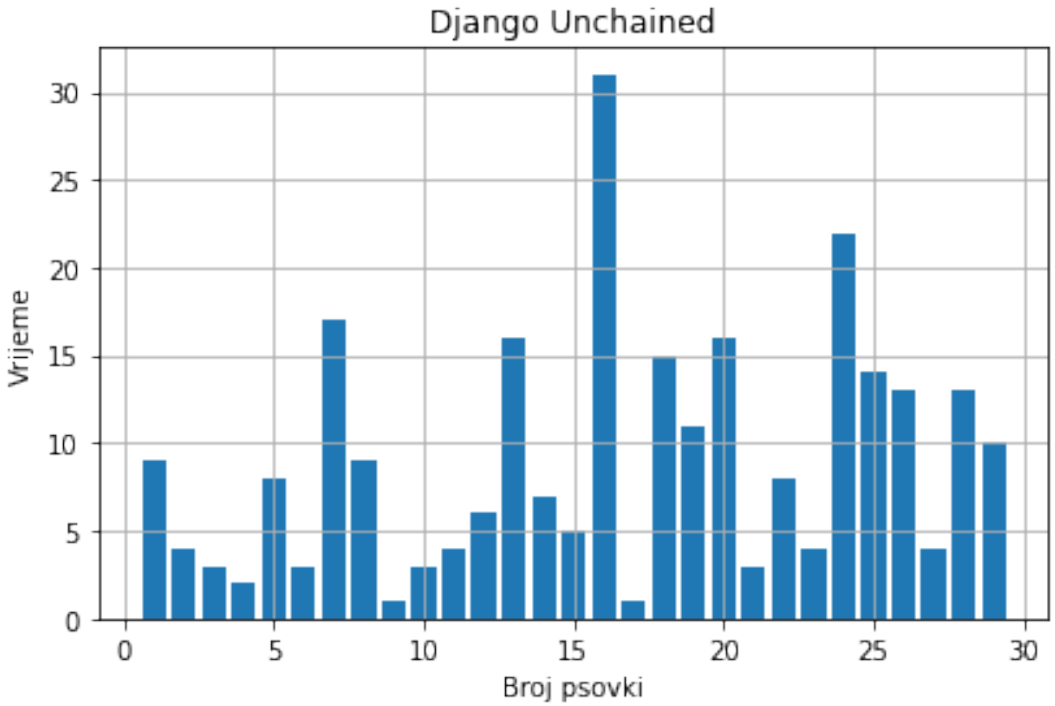


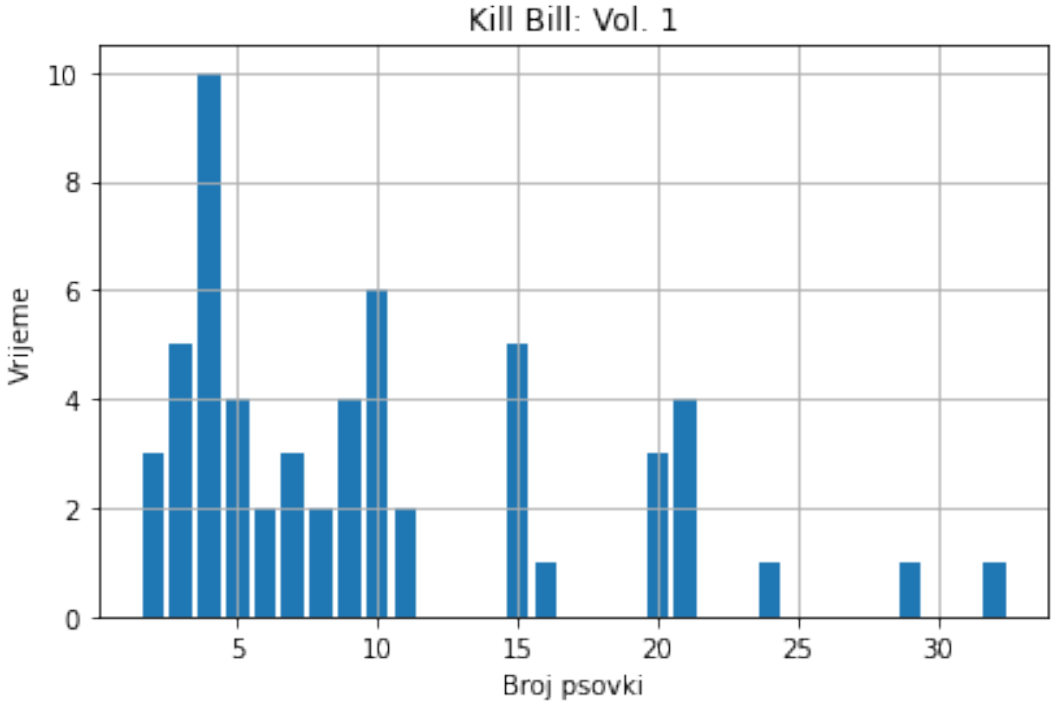
[31]:
```
for film in filmovi:
    psovka_dio_filma(c, film, detail = 30)
```

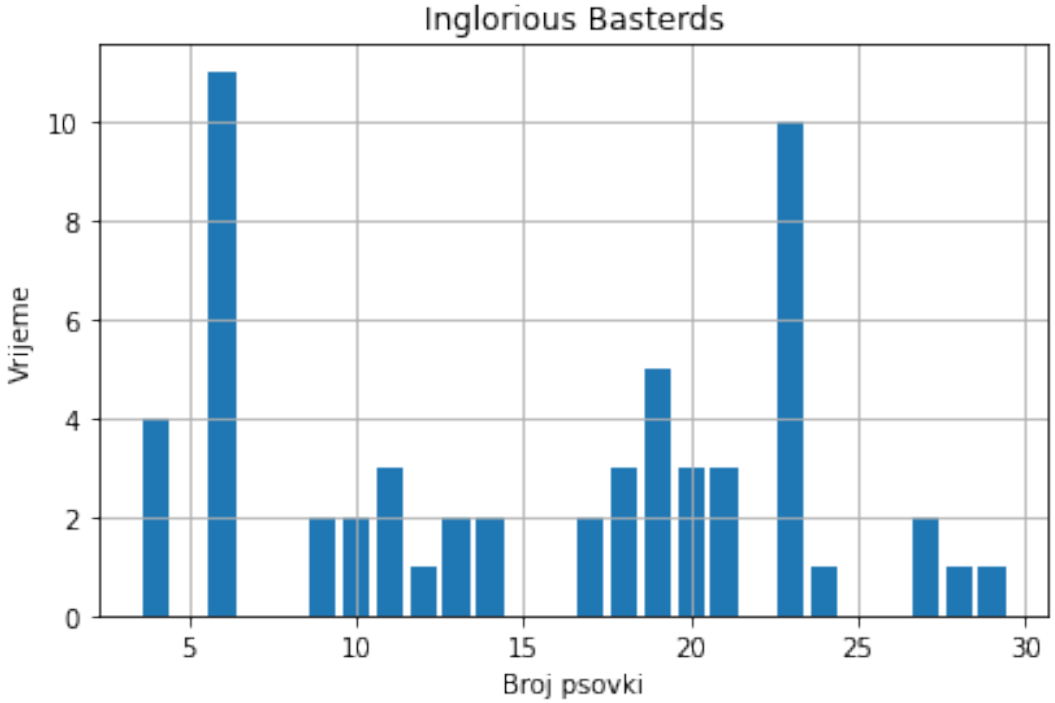


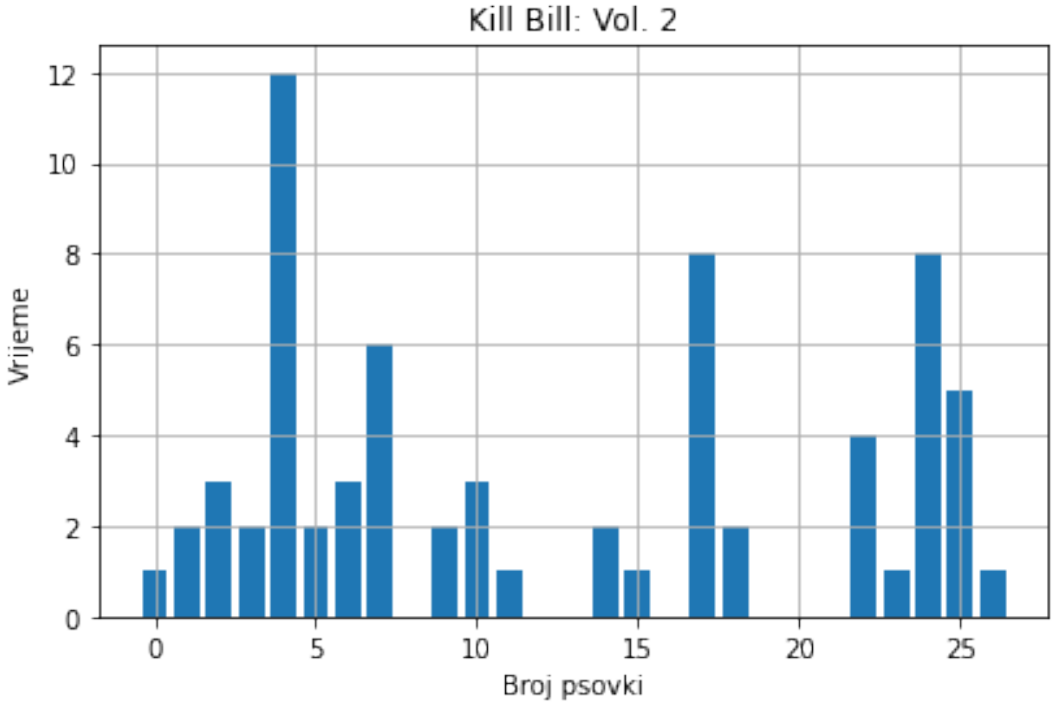

Reservoir Dogs
Vrijeme
Broj psovki
0
5
10
15
20
25
30


Pulp Fiction
Vrijeme
Broj psovki
0
5
10
15
20
25
30
40
50


Django Unchained
Vrijeme
Broj psovki
0
5
10
15
20
25
30


Kill Bill: Vol. 1
Vrijeme
Broj psovki
0
2
4
6
8
10
5
10
15
20
25
30

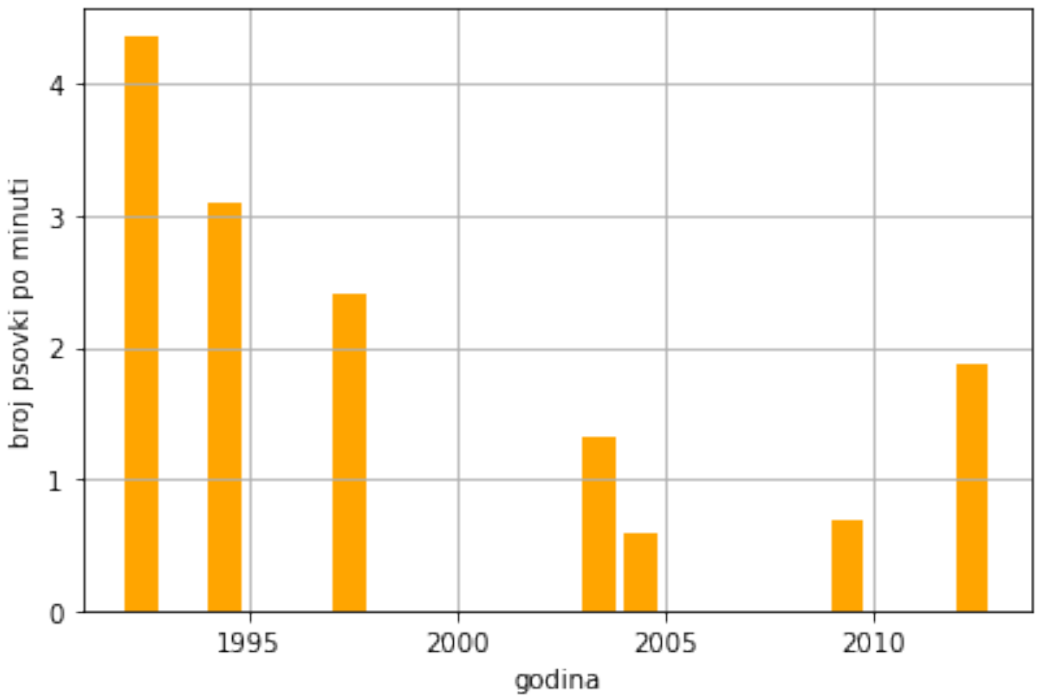

Inglorious Basterds
Vrijeme
Broj psovki
0
2
4
6
8
10
5
10
15
20
25
30


Kill Bill: Vol. 2
Vrijeme
Broj psovki
0
2
4
6
8
10
12
0
5
10
15
20
25

[32]:
```
psovki_po_godini = '''
SELECT  CAST (COUNT() as float)/film.trajanje, film.godina
FROM rijec, film
WHERE film_id = film_fk
GROUP BY film_fk
'''
rez = c.execute(psovki_po_godini).fetchall()
print(rez)
```

```
[(1.8727272727272728, 2012), (3.090909090909091, 1994), (0.6928104575163399,
2009), (1.3186813186813187, 2003), (2.4155844155844157, 1997),
(4.353535353535354, 1992), (0.583941605839416, 2004)]
```

[33]:
```
fig = plt.figure()
ax1 = fig.add_subplot(111)
ax1.set_ylabel('broj psovki po minuti')
ax1.set_xlabel('godina')
fig.tight_layout()
plt.grid(True)
plt.xticks([1990, 1995,2000,2005,2010,2015])
plt.bar([int(a[1]) for a in rez], [a[0] for a in rez], align='edge', width=0.8,␣
 ↪color='orange')
#plt.plot(*zip(*rez))
plt.savefig('rad/slike/psovke_godine.pdf')
plt.show()
```



[34]:
```
detail = 30
fig, ax = plt.subplots()
ax.set_title('Distribucija psovki')
vulg = '!'
ax.set_xlabel('Vrijeme')
ax.set_ylabel('Broj psovki')
suma = []
for film in filmovi:
    upit = '''
    SELECT CAST (rijec.vrijeme/film.trajanje*{0} as INTEGER) as psovka_pc
    FROM film, rijec
    WHERE rijec.film_fk = film_id
        AND rijec.rijec {2}= ''
        AND film.naziv = '{1}'
    '''.format(detail, film, vulg)

    data = c.execute(upit).fetchall()
    data = list(x[0] for x in data)

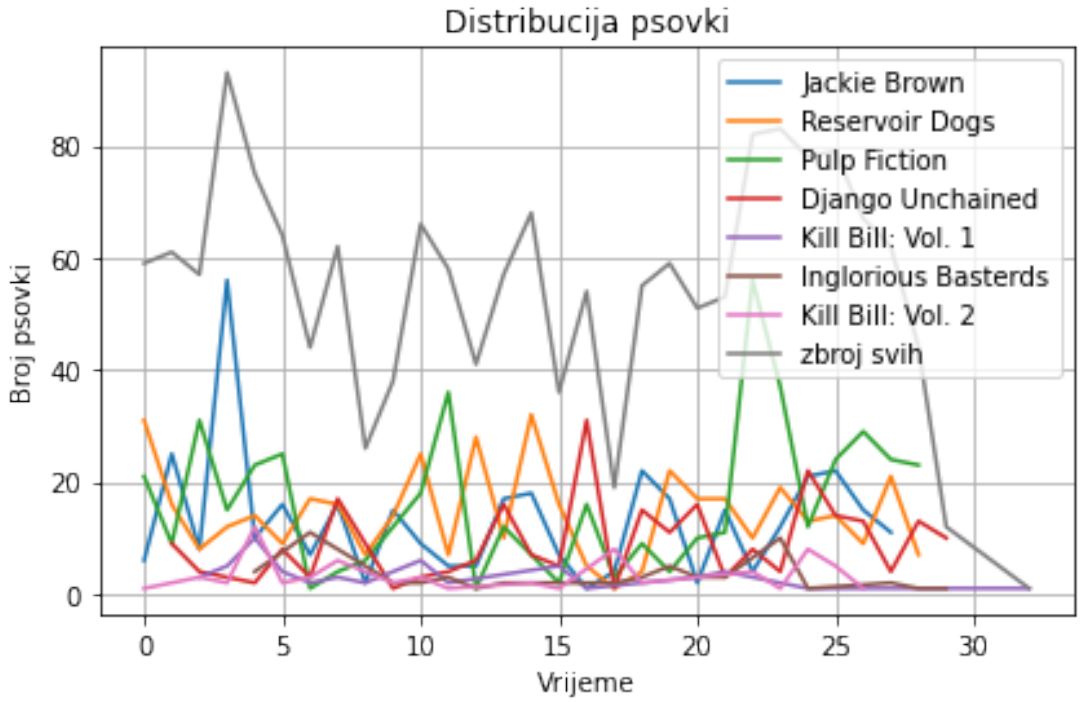
    data = dict(Counter(data))
    suma.append(data)
    #print(suma)



    ax.plot(list(data.keys()), list(data.values()), label=film)
import functools, operator
result = dict(functools.reduce(operator.add, map(Counter, suma)))
ax.plot(list(result.keys()), list(result.values()), label="zbroj svih")
fig.tight_layout()
ax.grid(True)

plt.legend(loc="upper right")

plt.savefig("rad/slike/filmovi_distribucija_psovki.pdf")
plt.show()
```



Najcesce psovke:

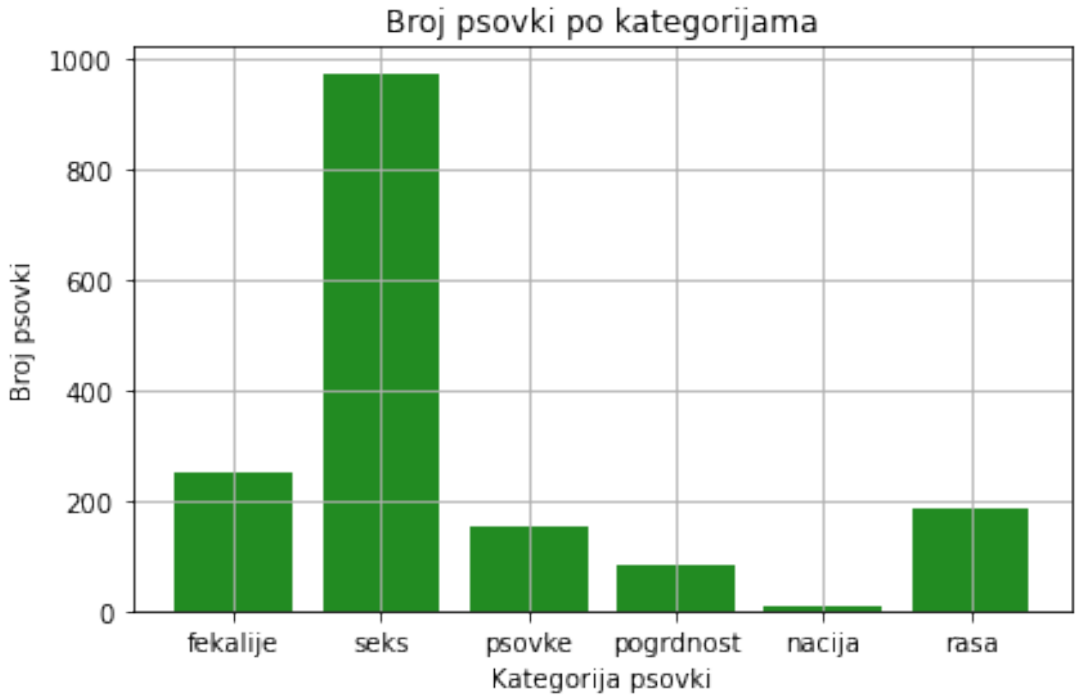
[35]:
```
najcesce_psovke = '''
SELECT kategorija.naziv, COUNT(rijec.kategorija_fk) ---, korijen.naziv
FROM kategorija, rijec
WHERE rijec.kategorija_fk = kategorija_id
GROUP BY rijec.kategorija_fk;
'''
rez = c.execute(najcesce_psovke).fetchall()
```

[36]:
```
fig, ax = plt.subplots()
ax.set_title('Broj psovki po kategorijama')
ax.set_xlabel('Kategorija psovki')
ax.set_ylabel('Broj psovki')

print(rez)

fig.tight_layout()
ax.grid(False)
ax.bar(*zip(*rez), color='ForestGreen')
ax.grid(True)
plt.savefig("rad/slike/kategorije_total.pdf")
plt.show()
```

```
[('fekalije', 253), ('seks', 974), ('psovke', 154), ('pogrdnost', 85),
('nacija', 8), ('rasa', 185)]
```



[37]:
```
fig, ax = plt.subplots()
ax.set_title('Broj psovki po kategorijama po filmu')
ax.set_ylabel('Broj psovki')
ax.set_xlabel('Kategorija psovki')
for film in filmovi:
    najcesce_psovke = '''
    SELECT kategorija.naziv, COUNT(rijec.kategorija_fk), film.naziv ---, korijen.
 ↪naziv
    FROM kategorija, rijec, film
    WHERE rijec.kategorija_fk = kategorija_id
        AND film.film_id = rijec.film_fk
        AND rijec.film_fk = (SELECT film.film_id FROM film WHERE film.naziv =␣
 ↪"{}" )
    GROUP BY rijec.kategorija_fk;
    '''.format(film)
    rez = c.execute(najcesce_psovke).fetchall()

    kategorije = list(list(a) for a in zip(*rez))
    fig.tight_layout()
    ax.grid(False)
    ax.bar(kategorije[0], kategorije[1], label=kategorije[2][1])
```

```
plt.legend(loc = "best")
plt.savefig("rad/slike/kategorije_total_po_filmovima.pdf")
plt.show()
```