

Skladišta i rudarenje podataka

Antonio Labinjan

Fakultet Informatike u Puli / Skladišta i rudarenje podataka

31. svibnja 2025.

Sadržaj

1	Uvod	2
2	Poslovna inteligencija	3
3	Skladišta podataka	4
4	Pronalazak i analiza dataseta	5
5	Inicijalna analiza podataka u Pythonu	5
6	Relacijski model podataka	9
7	Dimenzijski model podataka	14
8	Apache Spark, stvaranje i punjenje dimenzijskog modela	16
8.1	Extract na primjeru izbora	17
8.2	Transform na primjeru izbora	19
8.3	Transformacije dimenzijskih tablica	23
8.4	Load na primjeru izbora	34
9	Interaktivni web-based dashboard i vizualizacija podataka	37

1. Uvod

Politički sustavi diljem svijeta generiraju ogromne količine podataka tijekom izbora – od registracije birača, izlaznosti, rezultata po strankama i kandidatima, do povijesnih trendova i promjena u izbornim jedinicama. Takvi podaci predstavljaju vrijedan resurs za analizu ponašanja birača, evaluaciju učinkovitosti izbornih procesa te donošenje strateških odluka političkih aktera i institucija koje nadziru izborni proces. Međutim, sama količina i kompleksnost podataka predstavljaju izazov u pogledu njihove obrade, skladištenja i analitičke interpretacije.

Cilj ovog seminara je prikazati cjelokupni proces izgradnje sustava za analizu izbornih rezultata temeljenog na skladištu podataka. U okviru rada razvijen je dimenzijski model koji omogućuje učinkovito pohranjivanje i dohvat podataka relevantnih za parlamentarne izbore, uključujući stranke, biračke jedinice, vremenske dimenzije, povijesne promjene i druge povezane entitete. Poseban naglasak stavljen je na korištenje naprednih koncepata poput sporo mijenjajućih dimenzija (SCD) i degradiranih dimenzija, kako bi se omogućila dugoročna analitika i točno praćenje promjena kroz vrijeme.

Nakon oblikovanja podatkovnog modela, pristupilo se implementaciji interaktivnog web-based dashboarda koji korisnicima omogućuje vizualno istraživanje podataka i donošenje zaključaka temeljenih na realnim brojkama i trendovima. Umjesto korištenja gotovih alata za dashboard analizu poput Power BI-ja ili Tableaui, odlučeno je da se razvije vlastito rješenje u tehnologijama otvorenog koda — Vue.js, Matplotlib i Metabase — kako bi se postigla maksimalna fleksibilnost i prilagodba specifičnim zahtjevima domene izbornih podataka. Gotova rješenja, iako moćna, često imaju ograničenja kada je riječ o povezivanju s kompleksnim podatkovnim modelima i implementaciji interaktivnosti prilagođene korisnicima.

Kroz ovaj seminar objašnjeni su teorijski temelji i praktična realizacija sustava, počevši od modeliranja i implementacije baze podataka, preko ETL procesa, pa sve do krajnje vizualizacije podataka putem modernog web sučelja. Time se ne prikazuje samo tehnička strana izgradnje skladišta podataka, već i važnost kvalitetnog dizajna, razumijevanja domene i potrebe za dostupnom i razumljivom prezentacijom informacija krajnjim korisnicima.

2. Poslovna inteligencija

Poslovna inteligencija (engl. *Business Intelligence*, BI) obuhvaća skup tehnologija, procesa i metodologija kojima se organizacijama omogućuje prikupljanje, integracija, analiza i prezentacija poslovnih podataka u svrhu donošenja informiranih odluka. Ključna svrha BI sustava je pretvaranje sirovih podataka u korisne informacije koje pomažu menadžmentu u prepoznavanju obrazaca, trendova i prilika na tržištu. U modernim poslovnim okruženjima, BI ne podrazumijeva samo izvještavanje i analizu povijesnih podataka, već uključuje i prediktivne modele, vizualizaciju podataka te integraciju s operativnim sustavima u stvarnom vremenu. Korištenjem naprednih analitičkih alata i vizualizacijskih tehnika, poslovna inteligencija postaje neizostavan dio strateškog planiranja i operativnog upravljanja u brojnim industrijama.

3. Skladišta podataka

Skladište podataka (engl. *Data Warehouse*) predstavlja centralizirani repozitorij strukturiranih podataka namijenjen analitičkoj obradi, donošenju odluka i podršci poslovnoj inteligenciji. Za razliku od operativnih baza podataka koje su optimizirane za česte transakcije i ažuriranja, skladišta podataka su dizajnirana za čitanje velikih količina povijesnih podataka, agregaciju, kompleksne upite i izvještavanje. Podaci u skladište dolaze iz različitih izvora putem procesa ekstrakcije, transformacije i učitavanja (ETL), pri čemu se čiste, usklađuju i konsolidiraju kako bi omogućili jedinstvenu i pouzdanu analizu. Arhitektura skladišta podataka obično se temelji na višeslojnom modelu, uključujući sloj izvora podataka, ETL proces, centralno skladište i sloj prezentacije kroz kojeg krajnji korisnici pristupaju podacima. Unutar samog skladišta podataka često se koristi dimenzijski model koji omogućuje učinkovito izvođenje analitičkih upita kroz tablice činjenica i dimenzija. Korištenjem skladišta podataka organizacije mogu prepoznati trendove, identificirati poslovne prilike te podržati strateško planiranje temeljem pouzdanih i sveobuhvatnih podataka. Suvremena skladišta podataka sve češće uključuju i elemente distribuiranih sustava, pohranu u oblaku te podršku za polustrukturirane podatke, čime proširuju mogućnosti analize u raznim kontekstima i skalabilnim okruženjima.

4. Pronalazak i analiza dataseta

U procesu pronalaska dataseta isprobano je nekoliko standardnih izvora poput Kagglea, no većina tema nije bila dovoljno interesantna ili pak nije bilo dovoljno redaka u datasetu što bi onemogućilo ozbiljnije daljnje obrade i ne bi pružilo adekvatne analitičke uvide. Naposljetku, odabran je dataset o portugalskim parlamentarnim izborima naen na UC Irvine Machine Learning Repositoryju: <https://archive.ics.uci.edu/dataset/513/>

Ovaj dataset zadovoljavao je više ključnih uvjeta: dovoljan broj redaka i atributa, dostupnost podataka za različite teritorijalne jedinice, mogućnost vremenske analize te potencijal za dimenzijsko modeliranje po strankama, teritorijima i vremenu. Osim toga, tema izbora se pokazala zanimljivom za dublju obradu, vizualizaciju i izradu interaktivnog dashboarda.

Nakon odabira i preuzimanja dataseta, pristupilo se inicijalnoj obradi i upoznavanju s podacima koristeći Python u Google Colab notebook okruženju. U početku je skripta ispisala poruku "Hello World" radi provjere ispravnosti Python okruženja. Sljedeći korak bio je učitavanje biblioteke Pandas (Python and Data Science), koja se koristi za obradu i analizu strukturiranih podataka u obliku dataframeova.

Dataset je učitao iz CSV datoteke pomoću `pd.read_csv()`, pri čemu je prvotno analizirano samo prvih 2000 redaka kako bi se omogućio brzi pregled i izbjeglo opterećenje memorije. Ispisani su prvi redovi dataseta kako bi se dobio uvid u strukturu tablice i moguće varijable.

Daljnjom analizom ispisane su dimenzije cjelokupnog skupa podataka (`data.shape`), broj null vrijednosti po stupcima (`data.isna().sum()`), broj jedinstvenih vrijednosti po atributima (`data.nunique()`), kao i tipovi podataka po svakom stupcu (`data.dtypes`). Na kraju, korištenjem `value_counts()` za svaki stupac, dobiven je detaljan prikaz distribucije vrijednosti, što je pomoglo u identifikaciji potencijalnih dimenzija i metrika za kasnije faze modeliranja.

Za ostvaranje opisanih rezultata korišten je sljedeći kod:

5. Inicijalna analiza podataka u Pythonu

```
1 # Provjera da li je python instaliran na sustavu
2 print("Hello World")
3
4 # U itavanje potrebnih biblioteka
5 import pandas as pd
6
7 # U itavanje podataka iz CSV datoteke
8 PATH = "/content/ElectionData.csv"
9
10 data = pd.read_csv(PATH, delimiter=',')
11
12 # U itavanje prvih 2000 redova
13 data_first_2000 = pd.read_csv(PATH, delimiter=',', nrows=2000)
14
15 # Ispis prvih 5 redova
16 print(data_first_2000.head())
```

```
17
18 # Ispis dimenzija skupa podataka (potrebno je učitati sve podatke
    radi analize, ne samo prvih 2000 redova)
19 print(data.shape)
20
21 # Ispis imena stupaca i nedostaju ih vrijednosti
22 print(data.isna().sum())
23
24 # Ispis broja jedinstvenih vrijednosti po stupcima
25 print(data.nunique())
26
27 # Ispis tipova podataka po stupcima (analiza kvantitativnih i
    kvalitativnih varijabli)
28 print(data.dtypes)
29
30 # Ispis broja jedinstvenih vrijednosti po stupcima
31 for column in data:
32     print(data[column].value_counts())
33     input("...")
34
35 # Ispis imena stupaca
36 print(data.columns.values)
```

Listing 1: Učitavanje i inicijalna analiza izbornog dataseta u Pythonu

Za dodatno utvrđivanje kvalitete dataseta bilo je potrebno ispuniti određene uvjete.



(21643, 28)

Slika 1: Dimenzije dataseta

Vidljivo je kako dataset ima 21643 retka i čak 28 stupaca što zadovoljava inicijalne uvjete veličine dataseta od bar 15000 redaka i 10 stupaca. Bilo je potrebno postaviti donju granicu veličine iz razloga što premali dataset ne bi pružao dobre mogućnosti analize jer se iz njega ne bi mogle izvući određene pravilnosti, korelacije i sl.

Nakon što je utvrđeno da je dataset dovoljno velik, ispisani su svi stupci i broj nedostajućih vrijednosti u njima. Poželjno je da dataset ima čim je manje mogući broj takvih vrijednosti zato što one smanjuju kvalitetu analize i ne pružaju potpuni uvid u promatrane entitete. Nedostajuće vrijednosti mogu dovesti do iskrivljenih rezultata u statističkim analizama, jer se algoritmi mogu oslanjati na nepotpune ili neprecizne podatke. Osim toga, prisutnost null vrijednosti često zahtijeva dodatne korake obrade podataka poput popunjavanja nedostajućih vrijednosti, konverzije null vrijednosti u druge tipove (npr. pretvorba NULL u broj 0 ili u prazan string (" ")) ili filtriranja redaka, što može dodatno povećati kompleksnost analize i smanjiti interpretabilnost rezultata. U kontekstu poslovne inteligencije, takvi podaci mogu dovesti do donošenja pogrešnih poslovnih odluka ako se na njih ne obrati pažnja. Zbog svega navedenog, kvaliteta dataseta i njegova čistoća ključni su preduvjeti za izvođenje pouzdane i korisne analize. U ovom slučaju, dataset nije imao niti jednu nedostajuću vrijednost što je indikator njegove kvalitete i potpunosti i dodatna potvrda kako može biti korišten za implementaciju skladišta podataka.

```

TimeElapsed      0
time             0
territoryName     0
totalMandates     0
availableMandates 0
numParishes      0
numParishesApproved 0
blankVotes        0
blankVotesPercentage 0
nullVotes         0
nullVotesPercentage 0
votersPercentage  0
subscribedVoters  0
totalVoters       0
pre_blankVotes    0
pre_blankVotesPercentage 0
pre_nullVotes     0
pre_nullVotesPercentage 0
pre_votersPercentage 0
pre_subscribedVoters 0
pre_totalVoters   0
Party            0
Mandates         0
Percentage       0
validVotesPercentage 0
Votes           0
Hondt            0
FinalMandates    0
dtype: int64

```

Slika 2: Provjera broja nedostajućih vrijednosti

Nadalje, provjeren je i broj unique (jedinstvenih) vrijednosti u stupcima kako bi se utvrdila raznolikost podataka što je vrlo bitno u analizi zato jer raznolikost podataka izravno utječe na kvalitetu i dubinu analize koju možemo provesti. Stupci s velikim brojem jedinstvenih vrijednosti često sadrže ključne informacije koje omogućuju dublje razumijevanje podataka koji se analiziraju. Na primjer, atribut s mnogo različitih vrijednosti može predstavljati identifikatore poput naziva teritorija, političkih stranaka ili vremenskih oznaka, što nam omogućuje detaljnu analizu po tim dimenzijama.

S druge strane, stupci s vrlo malo jedinstvenih vrijednosti često predstavljaju kategorijske varijable koje se mogu koristiti za grupiranje podataka, što je korisno pri stvaranju dimenzijskih modela ili pri agregiranju podataka za vizualizaciju. U poslovnoj inteligenciji, razumijevanje strukture i raznolikosti podataka ključno je za izgradnju korisnih izvještaja, preciznih modela i donošenje odluka temeljenih na podacima. Analiza jedinstvenih vrijednosti pomaže i pri otkrivanju potencijalnih problema poput kodiranih vrijednosti koje predstavljaju iste entitete (npr. "HR" i "Hrvatska"), što može dovesti do neujednačenosti i netočnosti u interpretaciji zato što se ponekad isti real-life entitet može u datasetu evidentirati na više različitih načina.

```

TimeElapsed      54
time             54
territoryName     21
totalMandates     62
availableMandates 69
numParishes      98
numParishesApproved 219
blankVotes        329
blankVotesPercentage 146
nullVotes         331
nullVotesPercentage 187
votersPercentage  282
subscribedVoters  335
totalVoters       336
pre_blankVotes    323
pre_blankVotesPercentage 138
pre_nullVotes     329
pre_nullVotesPercentage 98
pre_votersPercentage 278
pre_subscribedVoters 331
pre_totalVoters   331
Party            21
Mandates         67
Percentage       1363
validVotesPercentage 1387
Votes           4829
Hondt            41
FinalMandates    17
dtype: int64

```

Slika 3: Provjera broja jedinstvenih vrijednosti

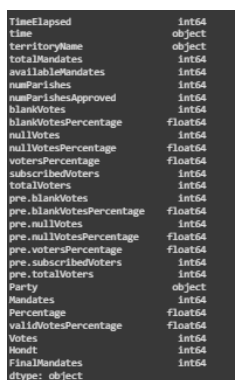
Vidljivo je kako svaki stupac ima zadovoljavajuć broj jedinstvenih vrijednosti što upućuje na to da dataset pruža dovoljno raznolikih informacija za provođenje kvalitetne analize. To znači da podaci nisu redundantni niti previše uniformni, što je čest

problem u manje kvalitetnim datasetima. Zadovoljavajući broj jedinstvenih vrijednosti omogućuje nam da uspješno razlikujemo, klasificiramo i grupiramo entitete po relevantnim kriterijima te izgradnju smislenih hijerarhija i veza unutar dimenzijskog modela.

Na primjer, ako stupac koji označava političke stranke ima dovoljno različitih vrijednosti, možemo analizirati ponašanje birača prema različitim strankama, promatrati distribuciju glasova kroz teritorije i vrijeme, te povezati rezultate s demografskim ili geografskim faktorima. Slično tome, stupac s nazivima teritorija omogućava prostornu analizu rezultata izbora, što je korisno za lokalizirano donošenje zaključaka i izradu ciljanih vizualizacija. Svaki će stupac biti detaljnije objašnjen u daljnjoj analizi.

U idućem koraku bilo je potrebno provjeriti tipove podataka u svakom stupcu kako bi se utvrdilo sadrži li dataset i kvalitativne i kvantitativne podatke. Kvantitativni podaci odnose se na brojeve dok kvalitativni podaci mogu biti u raznim formatima, no ovdje se specifično radi o stringovima. Kvantitativni podaci, poput broja glasova, omogućuju izvođenje statističkih analiza, izračun prosjeka, suma, postotaka te primjenu naprednijih metoda poput regresije ili klasifikacije. S druge strane, kvalitativni podaci – poput imena političkih stranaka ili teritorija – služe za kategorizaciju, grupiranje i filtriranje, što je osnova za stvaranje dimenzija u dimenzijskom modelu.

Identifikacija tipova podataka pomaže i u pripremi za obradu i vizualizaciju. Na primjer, numeričke vrijednosti mogu se prikazati u obliku grafikona, dok se kategorijske vrijednosti češće koriste za filtriranje rezultata u dashboardima. Također, razumijevanje strukture podataka omogućuje rano otkrivanje potencijalnih pogrešaka – primjerice, ako bi se brojevi podaci greškom učitali kao tekstualni, bilo bi onemogućeno izvođenje matematičkih operacija nad njima.



TimeElapsed	int64
time	object
territoryName	object
totalMandates	int64
availableMandates	int64
numParishes	int64
numParishesApproved	int64
blankVotes	int64
blankVotesPercentage	float64
nullVotes	int64
nullVotesPercentage	float64
votersPercentage	float64
subscribedVoters	int64
totalVoters	int64
pre_blankVotes	int64
pre_blankVotesPercentage	float64
pre_nullVotes	int64
pre_nullVotesPercentage	float64
pre_votersPercentage	float64
pre_subscribedVoters	int64
pre_totalVoters	int64
Party	object
Mandates	int64
Percentage	float64
validVotesPercentage	float64
Votes	int64
Hondt	int64
FinalMandates	int64
dtype:	object

Slika 4: Provjera tipova podataka

Iz analize je vidljivo kako prevladavaju numeričke vrijednosti (cjelobrojne i decimalne), ali i da postoje pojedini "object" stupci. U ovom slučaju object predstavlja stringove i timestampove.

Na samom kraju ispisana su sva imena stupaca i primjer prvih nekoliko redaka data-seta. Također, za shvaćanje konteksta samih podataka, potrebno je razjasniti što svaki pojedini stupac predstavlja, pa tako, redom:

Stupac	Opis
TimeElapsed	Vrijeme proteklo od početka brojanja glasova.
time	Točno vrijeme kada su podaci zabilježeni (timestamp).
territoryName	Naziv izborne jedinice (u konkretnom slučaju, radi se o saveznom državama unutar Portugala).
totalMandates	Ukupan broj mandata koji se dodjeljuju u teritoriju.
availableMandates	Broj još neraspodijeljenih mandata.
numParishes	Ukupan broj biračkih mjesta.
numParishesApproved	Broj biračkih mjesta čiji su rezultati obrađeni.
blankVotes	Broj praznih (nepopunjenih) listića.
blankVotesPercentage	Postotak praznih listića.
nullVotes	Broj nevažećih (nepravilno ispunjenih) listića.
nullVotesPercentage	Postotak nevažećih listića.
votersPercentage	Postotak birača koji su glasali.
subscribedVoters	Ukupan broj registriranih birača.
totalVoters	Ukupan broj birača koji su izašli na izbore.
pre.blankVotes	Broj praznih listića u prethodnom izvještaju.
pre.blankVotesPercentage	Postotak praznih listića iz prethodnog izvještaja.
pre.nullVotes	Broj nevažećih listića iz prethodnog izvještaja.
pre.nullVotesPercentage	Postotak nevažećih listića iz prethodnog izvještaja.
pre.votersPercentage	Odaziv birača u prethodnom izvještaju.
pre.subscribedVoters	Broj registriranih birača u prethodnom izvještaju.
pre.totalVoters	Broj birača koji su glasali u prethodnom izvještaju.
Party	Naziv političke stranke na koju se podaci odnose.

Tablica 1: Opis svakog stupca u skupu podataka

Isto tako, važno je istaknuti da su finalnom dimenzijskom modelu dodani neki novi atributi kojih nije bilo u ovom csv-u kako bi se omogućila bolja analiza s dubljim uvidom u rezultate izbora. Po završetku analiziranja dataseta i dodatne potvrde da je zaista dobar, nastavljena je daljna izrada projekta. Idući korak bio je izrada relacijskog modela.

6. Relacijski model podataka

Zatim je bilo potrebno stvoriti smisleni relacijski model za prikupljene podatke. Relacijski model podataka predstavlja način strukturiranja informacija pomoću međusobno povezanih tablica koje odražavaju stvarne entitete i odnose među njima. U kontekstu analize izbornih rezultata, to označava modeliranje tablica za stranke, teritorije, rezultate po biračkom mjestu, vremenske oznake i slično. Poanta relacijskog modela je omogućiti organiziranu, normaliziranu i nedvosmislenu pohranu podataka tako da se smanji redundancija i poveća dosljednost. Ključne prednosti relacijskog modela uključuju jasno definirane odnose putem primarnih i stranih ključeva, fleksibilnost u pisanju SQL upita, integritet podataka te mogućnost jednostavne nadogradnje sustava bez narušavanja postojećih struktura. Također, relacijski modeli su široko prihvaćeni

u poslovnom okruženju te podržani od strane gotovo svih standardnih baza podataka, što ih čini vrlo praktičnim za implementaciju.

S druge strane, relacijski model nije savršen. U analitičkim i BI sustavima koji se bave velikim količinama podataka i zahtijevaju brze agregacije, relacijski model može biti sporiji u usporedbi s denormaliziranim strukturama poput dimenzijskih modela. Također, složenost modela može otežati razumijevanje krajnjim korisnicima koji nisu tehnički potkovani i na barataju znanjem o ključevima, relacijama, kardinalnostima i sl. Unatoč tim izazovima, izgradnja dobrog relacijskog modela ostaje temelj za kvalitetnu obradu i pripremu podataka za napredne analitičke i poslovne potrebe.

Izrađena su 2 dijagrama: ER i EER. ER model koristi osnovne elemente poput entiteta, atributa i veza za opis strukture podataka, dok EER model dodaje dodatne mogućnosti poput nasljeđivanja između entiteta kako bi se preciznije prikazale složenije situacije iz stvarnog svijeta. Također, ER dijagram izrađen je "ručno" koristeći Lucidchart dok je EER dijagram automatski generiran obrnutim inženjerstvom u MySQL workbenchu.

Kao alat za izradu relacijskog (i kasnije dimenzijskog) modela izabran je MySQL. No, prije samog rada s bazom, bilo je potrebno konceptualno izmodelirati entitete iz izbornih podataka, pa je stoga najprije izrađen ER dijagram.

Za potrebe obrade i analize izbornih podataka, kreiran je sveobuhvatan relacijski model u programskom jeziku Python koristeći SQLAlchemy – objektno-relacijski mapper (ORM) koji omogućava definiranje strukture baze podataka koristeći klase i objekte. Kroz ovaj model uspostavljene su četiri glavne tablice koje reprezentiraju ključne entitete iz domene parlamentarnih izbora u Portugalu: države ("country"), izbori ("election"), političke stranke ("party") i izborni rezultati ("result"), uz dodatnu pomoćnu tablicu za povijesne podatke o izborima ("election_history").

Prvo je definirana tablica **Country**, koja predstavlja teritorijalnu jedinicu – u ovom slučaju izborne jedinice koje su u modelu interpretirane kao države (ili "districts"). Svaka država ima svoj jedinstveni naziv i ID koji se koristi kao primarni ključ te se na njega referenciraaju izbori koji se u toj državi održavaju.

Nakon toga, definirana je tablica **Election**, koja sadrži sve relevantne informacije o pojedinim izborima, uključujući godinu održavanja, broj dostupnih i ukupnih mandata, broj biračkih mjesta (parishes), broj odobrenih biračkih mjesta, broj i postotak praznih i nevažećih listića, postotak izlaznosti birača, broj upisanih birača te ukupan broj glasača. Tablica je povezana s tablicom "country" pomoću stranog ključa kako bi se znalo kojoj državi pojedini izbori pripadaju.

Tablica **Party** predstavlja političke stranke koje su sudjelovale na izborima. Svaka stranka ima jedinstveni naziv i identifikator.

Tablica **Result** povezuje izbore i političke stranke, te bilježi broj mandata koje je pojedina stranka osvojila, postotak glasova, postotak važećih glasova, ukupan broj osvojenih glasova i konačan broj osvojenih mandata. Time se omogućuje detaljna analiza rezultata izbora po strankama i povezivanje s konkretnim izbornim ciklusima i teritorijima.

Pomoćna tablica **ElectionHistory** dodaje dodatne informacije koje se odnose na povijesne podatke pojedinih izbora, poput broja praznih i nevažećih listića prije izbora,

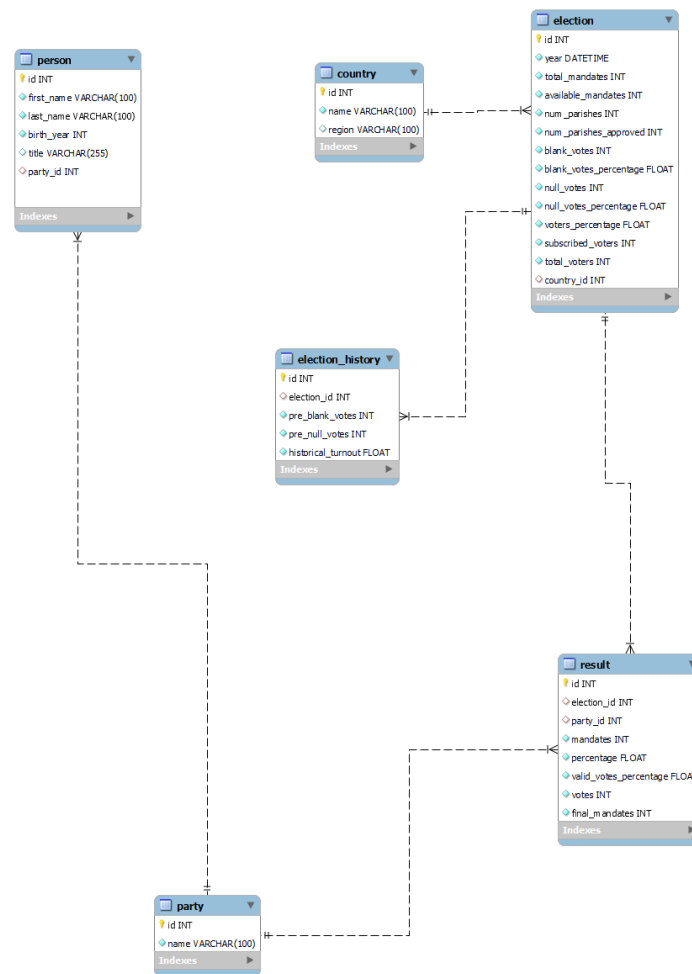
te povijesni postotak izlaznosti. Ova tablica je također povezana s tablicom "election" putem stranog ključa.

Podaci su u sustav učitani iz CSV datoteke koja sadrži prethodno obrađene izborne podatke. Svaki entitet u modelu najprije je inicijaliziran (npr. sve države i stranke su inicijalno stvorene i spremljene u bazu), a zatim su redovi CSV-a iterirani kako bi se za svaki unos stvorili novi izbori, rezultati i povijesni zapisi, pri čemu su uspostavljene sve potrebne veze između entiteta.

Ovakav relacijski model omogućuje:

preciznu i konzistentnu pohranu podataka – svaka informacija se nalazi na samo jednom mjestu (npr. ime stranke), normalizaciju podataka – smanjuje se redundancija, jednostavno postavljanje upita – zahvaljujući primarnim i stranim ključevima, mogućnost proširenja – ako bi se u budućnosti htjeli dodati novi entiteti poput kandidata, regija unutar država ili koalicija, model se može lako proširiti bez potrebe za rekonstrukcijom cijele baze, analitičku dubinu – moguće je pisati složene SQL upite za analizu povijesnih trendova, ponašanja birača ili uspjeha stranaka po teritorijima i godinama.

Ovakva struktura predstavlja dobar temelj za naprednu analitiku, vizualizaciju izbornih podataka, ali i buduću izgradnju aplikacije za prikaz rezultata ili predviđanje izbornih ishoda temeljem povijesnih uzoraka. Na ovaj način postignuta je maksimalna iskoristivost dostupnih podataka kroz sustavno i logično modeliranje relacija u domeni političkih izbora. Također, iako je ovaj relacijski model već sam po sebi poprilično zadovoljavajući, u idućoj iteraciji projekta (pretvorba u dimenzijski model) dodani su još neki atributi koji će predstavljati nove dimenzije za promatranje podataka



Slika 5: EER dijagram relacijskog modela

Analiza EER Dijagrama Parlamentarnih Izbora

U nastavku se nalazi detaljna analiza entitetsko-relacijskog (EER) modela koji prikazuje strukturu baze podataka namijenjene vođenju evidencije parlamentarnih izbora. Model uključuje entitete poput izbora, rezultata, političkih stranaka i osoba, te opisuje njihove međusobne veze i kardinalnosti.

Entiteti i njihova svojstva

1. person Entitet **person** predstavlja osobu povezanu s nekom političkom strankom. Svojstva entiteta su:

- **id** – jedinstveni identifikator osobe (INT)
- **first_name** – ime (VARCHAR)
- **last_name** – prezime (VARCHAR)
- **birth_year** – godina rođenja (INT)
- **title** – titula ili funkcija osobe (VARCHAR)

- `party_id` – strani ključ prema entitetu `party`
2. `party` Entitet `party` predstavlja političku stranku. Svojstva:
- `id` – jedinstveni identifikator stranke (INT)
 - `name` – naziv stranke (VARCHAR)
3. `country` Entitet `country` predstavlja izbornu jedinicu. Svojstva:
- `id` – identifikator izborne jedinice (INT)
 - `name` – ime izborne jedinice (VARCHAR)
 - `region` – regija kojoj jedinica pripada (VARCHAR)
4. `election` Entitet `election` sadrži podatke o određenom izbornom ciklusu. Svojstva uključuju:
- `id`, `year`, `total_mandates`, `available_mandates`, `num_parishes`, `num_parishes_approved`
 - `blank_votes`, `null_votes`, `blank_votes_percentage`, `null_votes_percentage`
 - `voters_percentage`, `subscribed_voters`, `total_voters`
 - `country_id` – strani ključ prema entitetu `country`
5. `result` Entitet `result` prikazuje rezultate izbora za pojedine stranke:
- `id`, `election_id`, `party_id` – strani ključevi
 - `mandates`, `final_mandates`, `percentage`, `valid_votes_percentage`, `votes`
6. `election_history` Entitet `election_history` sadrži povijesne informacije povezane s određenim izborima:
- `id`, `election_id` – strani ključ
 - `pre_blank_votes`, `pre_null_votes`, `historical_turnout`

Veze i kardinalnosti

- **person – party:** Veza je **N:1**, jer više osoba može pripadati jednoj stranci.
- **election – country:** Veza je **N:1**, jer više izbora se može održati u jednoj izbornoj jedinici.
- **election_history – election:** **1:1** veza, jer je povijest direktno vezana uz jedan izbor.
- **result – election:** **N:1**, jer se svaki rezultat odnosi na jedan izbor.
- **result – party:** **N:1**, jer se više rezultata može odnositi na istu stranku (kroz vrijeme).

EER model omogućava detaljno praćenje izbora, njihovih rezultata, pripadnosti osoba političkim strankama, te povijesnih podataka. Jasna hijerarhija veza i kardinalnosti omogućuje izvođenje složenih upita i potpunu analizu izbornog procesa. U daljnjem razvoju dimenzijskog modela neki su entiteti modificirani te su im dodani stupci koji omogućavaju razvoj hijerarhija i razvijena je logika za punjenje tih stupaca.

7. Dimenzijski model podataka

Dimenzijski model predstavlja jednu od temeljnih metoda modeliranja podataka u skladištima podataka, s glavnim ciljem optimizacije upita i omogućavanja brze i fleksibilne analize. Za razliku od relacijskog modela koji je usmjeren na normizaciju i konzistenciju podataka, dimenzijski model je denormaliziran kako bi omogućio lakše čitanje velikih količina podataka. Osnovni elementi dimenzijskog modela su tablice činjenica (*fact tables*) i tablice dimenzija (*dimension tables*).

Tablica činjenica sadrži mjerne podatke ili kvantitativne metrike koje korisnici žele analizirati. U kontekstu ovog modela, glavna tablica činjenica je `fact_election_result`, koja sadrži metrike kao što su broj mandata, broj glasova, postotak glasova, broj nevažećih i praznih listića, ukupno birača, izlaznost i druge agregatne vrijednosti povezane s izbornim rezultatima. Ključ svakog zapisa u tablici činjenica definiran je kombinacijom stranih ključeva koji vode prema odgovarajućim dimenzijama: izbori, stranke, vrijeme i povijest izbora.

Tablice dimenzija sadrže opisne podatke koji se koriste za filtriranje, grupiranje i označavanje činjenica. U promatranom modelu možemo uočiti više dimenzijskih tablica:

- `dim_election`: opisuje pojedine izbore, uključujući njihov identifikator te poveznicu na datum (`date_tk`) i zemlju (`country_id`).
- `dim_date`: omogućuje vremensku analizu podataka, uključujući detalje poput dana, mjeseca, godine i dana u tjednu.
- `dim_country`: predstavlja političku geografiju, uključujući regije i nazive zemalja (odnosno izbornih jedinica).
- `dim_party`: opisuje političke stranke koje sudjeluju u izborima, uključujući njihov naziv i političku orijentaciju.
- `dim_person`: dodatna dimenzija koja povezuje osobu sa strankom i omogućuje analiziranje političara, uključujući ime, prezime, godinu rođenja i titulu.
- `dim_election_history`: omogućuje povijesni kontekst pojedinih izbora, npr. koliko je bilo praznih i nevažećih listića u prethodnim verzijama, datume početka i završetka povijesnih razdoblja, kao i izlaznost.

Važan koncept u ovakvim modelima su sporo mijenjajuće dimenzije (*slowly changing dimensions*, SCD). Ove dimenzije se mijenjaju rijetko, ali je važno sačuvati staru vrijednost za potrebe analize povijesnih podataka. U ovom modelu `dim_election_history` predstavlja sporo mijenjajuću dimenziju, jer se rezultati i statistike prethodnih verzija

izbora zadržavaju u vremenskim intervalima `date_from` i `date_to`. Time se omogućuje točna vremenska analiza promjena u povijesti izbora.

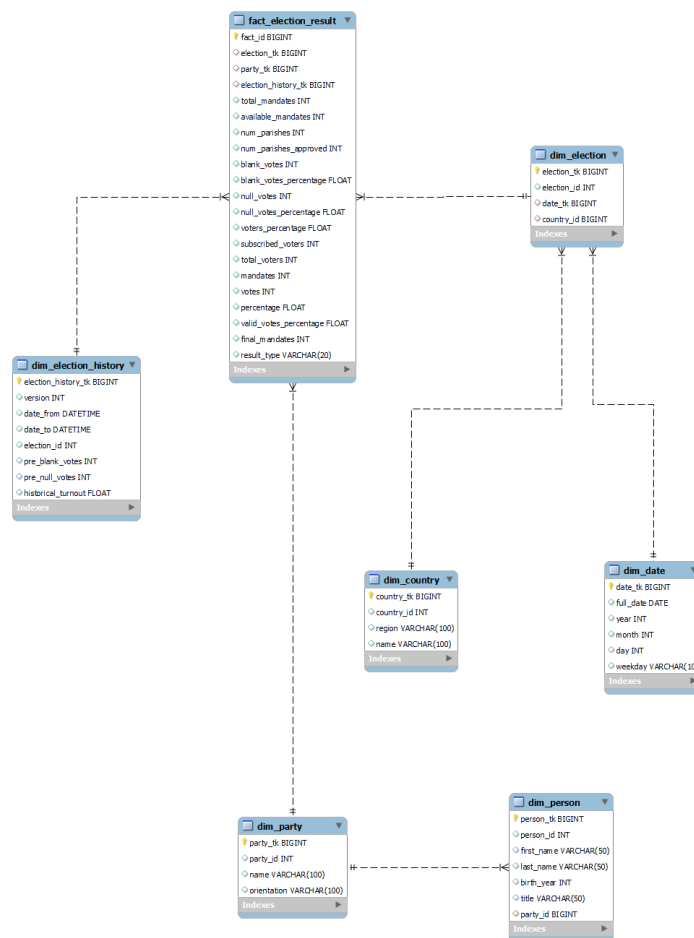
Dodatno, `dim_person` također može predstavljati sporo mijenjajuću dimenziju ako se, primjerice, političar prebaci iz jedne stranke u drugu – za takav slučaj moguće je pratiti njegovu pripadnost kroz različite vremenske točke zadržavajući povijest.

Također, treba napomenuti kako su neki od podataka izgenerirani te se nisu nalazili u originalnom datasetu. To su redom: degenerirana dimenzija `result_type` koja će biti detaljnije objašnjena u idućem paragrafu, `version`, `date_from`, `date_to` koji služe za praćenje promjena u glasovima na izborima kroz vrijeme (sa svakim novim dodavanjem obrađenih glasova, kreira se nova verzija `election_history`), `region` unutar `dim_country` koji služi za stvaranje dodatne hijerarhije unutar dimenzije i predstavlja geografsku regiju unutar Portugala. `Year`, `month`, `day` i `weekday` unutar `dim_date` koji omogućuju detaljniju vremensku analizu podataka te su kreirani ekstrakcijom dijelova `timestampa` iz dataseta. `Orientation` unutar `dim_party` koji predstavlja političku orijentaciju pojedine stranke (lijevo, desno ili centar) te `title` unutar dimenzije `person` koja označava titulu kandidata na izborima (doktor, magistar i sl.)

U modelu je prisutna i degradirana dimenzija (*degenerate dimension*), koja se odnosi na atribut `result_type` unutar tablice činjenica. Iako se radi o opisnom atributu (npr. *konačni*, *privremeni*), za njega nije definirana posebna tablica dimenzija. Umjesto toga, `result_type` je direktno pohranjen u tablici činjenica. Ovakav pristup je čest kada atribut nema dodatne opisne podatke i koristi se isključivo za filtriranje ili identifikaciju rezultata.

Što se tiče strukture modela, ovdje se ne koristi čista zvjezdasta shema (*star schema*), već tzv. snježna pahulja (*snowflake schema*). U snježnoj shemi dimenzije su dodatno normirane, čime se smanjuje redundantnost podataka, ali i povećava broj pridruživanja (*joinova*) prilikom izvršavanja upita. To se jasno vidi u relaciji između `dim_election` i `dim_country` te `dim_election` i `dim_date`, gdje su dimenzije izvedene putem dodatnih vanjskih ključeva, a ne direktno unutar tablice činjenica. Time je postignuta veća fleksibilnost i konzistencija, ali na račun nešto slabijih performansi u analitičkim sustavima.

Zaključno, ovaj model predstavlja dobro strukturiran dimenzijski model temeljen na "pahuljici", koji omogućuje složenu analizu izbornih rezultata kroz različite dimenzije, uključujući vrijeme, lokaciju, političke stranke, kandidate i povijesni kontekst. Korištenje sporo mijenjajućih dimenzija i višerazinskih veza omogućuje točnu i povijesno konzistentnu analitiku u političkom okruženju, dok upotreba degradirane dimenzije `result_type` dodatno pojednostavljuje klasifikaciju rezultata bez potrebe za dodatnim relacijama.



Slika 6: Snowflake shema dimezijskog modela

8. Apache Spark, stvaranje i punjenje dimenzijskog modela

ETL (Extract, Transform, Load) predstavlja ključni proces u sustavima za skladištenje podataka, čija je primarna svrha integracija podataka iz različitih izvora, struktura i formata u jedinstveno, konzistentno i strukturirano okruženje pogodno za analitičku obradu. Faza Extract obuhvaća dohvat podataka iz raznih izvorišta, koja mogu uključivati relacijske baze podataka, nestrukturirane datoteke, web servise ili aplikacijske sustave. U fazi Transform podaci se podvrgavaju operacijama čišćenja, normalizacije, obogaćivanja, konsolidacije i mapiranja, pri čemu se sirovi podaci pretvaraju u semantički ujednačen oblik. Posljednja faza, Load, uključuje učitavanje pripremljenih podataka u skladište podataka (Data Warehouse), pri čemu se poštuju pravila integriteta, optimizira performans i osigurava konsistentnost. ETL proces je od presudne važnosti jer omogućuje pouzdanost i točnost analitičkih sustava, smanjuje redundantnost i neusklađenost podataka, te postavlja temelj za učinkovito izvršavanje OLAP operacija, izvještavanje i donošenje poslovnih odluka temeljenih na podacima.

Za potrebe izvođenja ETL procesa korišten je Apache Spark, open-source distribuirani framework za obradu podataka otvorenog koda, poznat po svojoj visokoj skalabilnosti

i brzini. Iako Spark primarno nije specijaliziran alat za klasične ETL zadatke, već je zapravo generalni sustav za obradu velikih količina podataka u memoriji, njegova sposobnost paralelizacije i podrška za rad s različitim izvorima podataka čine ga izuzetno pogodnim za takve procese. U konkretnom slučaju, Spark je uspješno korišten za dohvat podataka, njihovu transformaciju (uključujući filtriranje, čišćenje i strukturiranje) te učitavanje u konačni analitički model. Time je potvrđena njegova primjenjivost u ETL kontekstu, posebno u okruženjima gdje je važna učinkovitost u radu s većim količinama podataka ili potreba za distribucijom obrade. U konkretnom slučaju, distribuiranost obrade se očituje upravo u učitavanju iz više izvora (koji su kreirani u drugom koraku pri splittanju dataseta).

Također, jedna od prednosti Sparka u odnosu na Pentaho (koji je bio alternativna opcija) je svakako mogućnost rada kroz kod bez ikakve potreba za uporabom grafičkog korisničkog sučelja i snalaženja u brojnim tipkama/raznim tabovima sučelja i sl. No, prije nego što će biti objašnjen sam kod i konretan ETL proces za obradu podataka izbora, potrebno je malo detaljnije objasniti od čega se sastoji svaki korak ETL procesa.

Extract (dohvat podataka)

Faza *Extract* označava početak ETL procesa i odnosi se na dohvat podataka iz različitih izvora — relacijskih baza podataka, CSV i JSON datoteka, web servisa ili aplikacijskih logova. Cilj ove faze je prikupljanje sirovih podataka u što potpunijem i vjerodostojnijem obliku, neovisno o njihovoj izvornoj strukturi i formatu. Kvalitetan dohvat osigurava temelj za pouzdanu obradu u narednim fazama.

Transform (obrada i priprema podataka)

U fazi *Transform* dohvaćeni podaci podvrgavaju se obradi kako bi se pripremili za skladištenje. Transformacije uključuju operacije poput čišćenja (uklanjanje praznih ili nepravilnih vrijednosti), normalizacije formata (npr. datuma i decimalnih točaka), grupiranja, izračuna dodatnih atributa i spajanja iz više izvora. Ova faza je ključna za osiguravanje konzistentnosti i točnosti podataka u konačnom modelu.

Load (učitavanje u skladište podataka)

Završna faza, *Load*, odnosi se na pohranu transformiranih podataka u ciljni sustav — najčešće skladište podataka (data warehouse). Učitavanje se može vršiti inicijalno (puni unos) ili inkrementalno (samo nove ili promijenjene vrijednosti). Kroz ovu fazu osigurava se integritet podataka, usklađenost s postojećim modelom te dostupnost informacija za daljnju analitičku obradu i izvještavanje.

SAD ĆE TU TRIBAT PASAT KROZ SAV KOD; HITAT PRIMJERE I OBJAŠNJAVAT,
ALI TO ĆU SUTRA

8.1 Extract na primjeru izbora

U sklopu ETL procesa za konkretni primjer izbora, faza *Extract* odgovorna je za dohvat sirovih podataka iz izvornih sustava. U ovom slučaju, dohvat je implementiran korištenjem PySparka i podijeljen u dva modula: jedan za učitavanje CSV datoteka,

a drugi za povezivanje s MySQL bazom podataka. Datoteka `extract_csv.py` sadrži funkciju `extract_from_csv`, koja uz pomoć Spark sessiona učitava CSV datoteku s uključenim zaglavljima i automatski detektira tipove podataka (`inferSchema=True`).

```
1 # extract/extract_csv.py
2 from spark_session import get_spark_session
3
4 def extract_from_csv(file_path):
5     spark = get_spark_session("ETL Extract - CSV")
6     df = spark.read.option("header", True).option("inferSchema", True)
7         .csv(file_path)
8     return df
```

Listing 2: Dohvat podataka iz CSV datoteke pomoću PySparka

Drugi dio, `extract_mysql.py`, fokusira se na dohvat relacijskih tablica iz MySQL baze podataka korištenjem JDBC konekcije. Funkcija `extract_table` definira sve potrebne parametre za povezivanje — URL baze, korisničke podatke te naziv JDBC drivera — i dohvaća tablicu u obliku Spark DataFrame objekta. Osim pojedinačnog dohvaćanja, funkcija `extract_all_tables` grupira učitavanje svih relevantnih tablica (`country`, `election`, `election_history`, `party`, `person`, `result`) koje čine temelj za daljnju transformaciju i analizu.

```
1 # extract/extract_mysql.py
2 from spark_session import get_spark_session
3
4 def extract_table(table_name):
5     spark = get_spark_session("ETL_App")
6
7     jdbc_url = "jdbc:mysql://127.0.0.1:3306/elections_brazil?useSSL=
8         false"
9     connection_properties = {
10         "user": "root",
11         "password": "root",
12         "driver": "com.mysql.cj.jdbc.Driver"
13     }
14
15     df = spark.read.jdbc(url=jdbc_url, table=table_name, properties=
16         connection_properties)
17     return df
18
19 def extract_all_tables():
20     return {
21         "country": extract_table("country"),
22         "election": extract_table("election"),
23         "election_history": extract_table("election_history"),
24         "party": extract_table("party"),
25         "person": extract_table("person"),
26         "result": extract_table("result"),
27     }
```

Listing 3: Dohvat podataka iz MySQL baze pomoću PySparka

8.2 Transform na primjeru izbora

U okviru ovog sustava, faza *Transform* imala je ključnu ulogu u obradi sirovih podataka te njihovom pretvaranju u strukturirani oblik prikladan za učitavanje u skladište podataka. Cilj ove faze bio je osigurati konzistentnost, kvalitetu i semantičku jasnoću podataka prije pohrane. U kontekstu implementacije, transformacijski proces je modularno organiziran: za svaku dimenzijsku tablicu razvijena je zasebna transformacijska funkcija, dok je nad svim funkcijama integracijski nadređena tzv. *pipeline* skripta, koja orkestrira izvođenje svih transformacija nad dohvaćenim sirovim podacima.

Kod prikazan u nastavku predstavlja sadržaj skripte `pipeline.py`, koja izvršava transformaciju nad svim ključnim tablicama — `dim_country`, `dim_date`, `dim_party`, `dim_election`, `dim_election_history`, `dim_person`, te nad činjenicama u tablici `fact_election_data`. Svaka transformacija poziva odgovarajuću funkciju iz zasebnog modula, pri čemu se podaci kombiniraju iz relacijske baze i dodatnih CSV datoteka. Ova podjela omogućuje bolju preglednost, višekratnu iskoristivost i lakšu izolaciju pogrešaka. Osim što omogućuje filtriranje i čišćenje podataka, transformacijski koraci uključuju i povezivanje dimenzijskih podataka u konačnu činjeničnu tablicu, čime se definira analitički model koji čini temelj za OLAP obradu.

Transform korak ETL procesa bio je najsloženiji zbog potrebe za usklađivanjem više odvojenih skripti gdje je svaka od njih mogla biti uzrok potencijalne greške koja onemogućava izvršavanje transformacija i ponajviše zbog brojnih joinova koje je bilo potrebno uskladiti.

```
1 from transform.dimensions.country_dim import transform_country_dim
2 from transform.dimensions.date_dim import transform_date_dim
3 from transform.dimensions.party_dim import transform_party_dim
4 from transform.dimensions.election_dim import transform_election_dim
5 from transform.dimensions.election_history_dim import
   transform_election_history_dim
6 from transform.dimensions.person_dim import transform_person_dim
7 from transform.facts.elections_fact import transform_elections_fact
8
9 print(" Pokrecemo pipeline.py")
10
11 def run_transformations(raw_data):
12
13     first_row = next(iter(raw_data.items()))
14     print(first_row)
15
16     print("\n Starting all transformations...\n")
17
18     print(" [1] Transforming Country dimension...")
19     try:
20         country_dim = transform_country_dim(
21             raw_data["country"],
22             csv_country_df=raw_data.get("ElectionData")
23         )
24         print("[1] Country dimension complete\n")
25     except Exception as e:
26         print(f" [1] Country dimension failed: {e}")
27         raise
28
```

```
29 print(" [4] Transforming Election dimension...")
30 try:
31     election_dim = transform_election_dim(
32         raw_data["election"],
33         csv_election_df=raw_data.get("ElectionData")
34     )
35     print(" [4] Election dimension complete\n")
36 except Exception as e:
37     print(f" [4] Election dimension failed: {e}")
38     raise
39
40 print(" [5] Transforming Election History dimension...")
41 try:
42     election_history_dim = transform_election_history_dim(
43         raw_data["election_history"],
44         csv_election_history_df=raw_data.get("ElectionData")
45     )
46     print(" [5] Election History dimension complete\n")
47 except Exception as e:
48     print(f" [5] Election History dimension failed: {e}")
49     raise
50
51 print(" [6] Transforming Person dimension...")
52 try:
53     person_dim = transform_person_dim(
54         raw_data["person"],
55         csv_person_df=raw_data.get("ElectionData")
56     )
57     print(" [6] Person dimension complete\n")
58 except Exception as e:
59     print(f" [6] Person dimension failed: {e}")
60     raise
61
62 print(" [3] Transforming Party dimension...")
63 try:
64     party_dim = transform_party_dim(
65         raw_data["party"],
66         csv_party_df=raw_data.get("ElectionData")
67     )
68     print(" [3] Party dimension complete\n")
69 except Exception as e:
70     print(f" [3] Party dimension failed: {e}")
71     raise
72
73 print(" [2] Transforming Date dimension...")
74 try:
75     date_dim = transform_date_dim(
76         raw_data["election"],
77         csv_date_df=raw_data.get("ElectionData")
78     )
79     print(" [2] Date dimension complete\n")
80 except Exception as e:
81     print(f" [2] Date dimension failed: {e}")
82     raise
83
84 print(" [7] Transforming Election Data fact table...")
85 try:
```

```

86     election_data_fact = transform_elections_fact(
87         country_dim,
88         date_dim,
89         party_dim,
90         election_dim,
91         election_history_dim,
92         person_dim
93     )
94     print(" [7] Election Data fact table complete\n")
95 except Exception as e:
96     print(f" [7] Election Data fact table failed: {e}")
97     raise
98
99     print(" All transformations completed successfully!")
100
101     return {
102         "dim_country": country_dim,
103         "dim_date": date_dim,
104         "dim_party": party_dim,
105         "dim_election": election_dim,
106         "dim_election_history": election_history_dim,
107         "dim_person": person_dim,
108         "fact_election_data": election_data_fact
109     }

```

Listing 4: Transformacijski pipeline koji orkestrira obradu svih dimenzija i činjenica

Kod predstavlja transformacijski *pipeline* koji orkestrira izvođenje svih transformacija nad sirovim podacima. Funkcija `run_transformations` prima dictionary ulaznih podataka (`raw_data`) i redom poziva funkcije za obradu pojedinih dimenzijskih tablica — država, izbora, povijesti izbora, stranaka, osoba i datuma. Svaka funkcija koristi podatke iz baze i/ili CSV datoteka, a rezultat se sprema u odgovarajući objekt. Nakon uspješnih transformacija dimenzija, poziva se funkcija za izgradnju činjenične tablice `fact_election_data`, koja povezuje sve prethodno obrađene dimenzije u cjeloviti analitički model. Završni rezultat funkcije je dictionary svih obrađenih tablica spremnih za učitavanje u skladište podataka.

Nadalje, vidljivo je kako se unutar pipelinea pozivaju određene funkcije od kojih je, naravno, svaku bilo potrebno zasebno definirati. Radi preglednosti, definirane su u zasebnim fajlovima (transformacija za svaku pojedinu tablicu je posebna datoteka koja sadrži transformacijsku funkciju).

Transformacija fact tablice `fact_election_data`

Transformacija fact tablice `fact_election_data` odrađena je korištenjem funkcije `transform_elections_fact` 5. Funkcija `transform_elections_fact` koristi podatke iz CSV datoteke (koja sadrži sve sirove attribute vezane za rezultate izbora po teritorijima) te ih spaja s relevantnim dimenzijama: `dim_country`, `dim_election`, `dim_election_history` i `dim_date`. U uvodnom dijelu funkcije provodi se osnovno čišćenje podataka — primjerice, standardizacija imena teritorija. Nakon toga slijedi niz `join` operacija s prethodno transformiranim dimenzijama, koje omogućuju dohvaćanje pripadajućih tehničkih ključeva. U završnoj fazi, funkcija izdvaja relevantne attribute i oblikuje ih

u konačnu fact tablicu, uz generiranje jedinstvenog `fact_id` identifikatora korištenjem funkcije `row_number()`. Logično, fact tablica se transformirala zadnja, pošto se unutar nje referenciraju sve dimenzijske tablice i ukoliko one nisu kreirane, narušit će se referencijalni integritet. Još jednom vrijedi istaknuti, kako su joinovi bili prilično izazovni jer je trebalo uskladiti sve nazive stupaca, tablica i sl. Koristio se *LEFT JOIN* kako bi se zadržali svi zapisi iz sirovih (ulaznih) podataka, čak i ako neki od njih nemaju pripadajuće vrijednosti u dimenzijskim tablicama. Time se osigurava potpunost podataka i sprječava njihov gubitak tijekom spajanja, što je standardna praksa u ETL procesu.

```
1 from pyspark.sql.functions import col, trim, initcap, row_number
2 from pyspark.sql.window import Window
3
4 print("Pokrećemo elections_fact")
5
6 def transform_elections_fact(
7     raw_data,
8     dim_country_df,
9     dim_election_df,
10    dim_election_history_df,
11    dim_date_df
12 ):
13     csv_df = raw_data
14
15     print("Dohvaćen election data iz CSV")
16
17     if csv_df is None:
18         raise ValueError("CSV podaci nisu pronađeni u raw_data!")
19
20     cleaned_csv = (
21         csv_df
22         .withColumn("territoryName", initcap(trim(col("country_name"))))
23     )
24     print("CSV očiscen")
25
26     enriched_df = (
27         cleaned_csv.alias("c")
28         .join(dim_country_df.alias("co"), col("c.territoryName") ==
29             col("co.country_name"), "left")
30         .join(dim_election_df.alias("e"), col("co.country_tk") == col(
31             "e.country_tk"), "left")
32         .join(dim_election_history_df.alias("eh"), col("e.election_id"
33             ) == col("eh.election_id"), "left")
34         .join(dim_date_df.alias("d"), col("c.electionDate") == col("d.
35             date"), "left")
36     )
37     print("Prva join-sesija OK")
38
39     fact_df = (
40         enriched_df
41         .select(
42             col("e.election_tk").alias("election_tk"),
43             col("eh.election_history_tk").alias("election_history_tk"),
44             col("c.availableMandates").alias("available_mandates"),
```

```

41         col("c.numParishes").alias("num_parishes"),
42         col("c.numParishesApproved").alias("num_parishes_approved"
43         ),
44         col("c.blankVotes").alias("blank_votes"),
45         col("c.blankVotesPercentage").alias("
46         blank_votes_percentage"),
47         col("c.nullVotes").alias("null_votes"),
48         col("c.nullVotesPercentage").alias("null_votes_percentage"
49         ),
50         col("c.votersPercentage").alias("voters_percentage"),
51         col("c.subscribedVoters").alias("subscribed_voters"),
52         col("c.totalVoters").alias("total_voters"),
53         col("c.Mandates").alias("mandates"),
54         col("c.Percentage").alias("percentage"),
55         col("c.validVotesPercentage").alias("
56         valid_votes_percentage"),
57         col("c.FinalMandates").alias("final_mandates"),
58         col("d.date").alias("election_date")
59     )
60     .withColumn("fact_id", row_number().over(Window.orderBy("
61         election_tk")))
62 )
63 print("Fact je gotov")
64 return fact_df

```

Listing 5: Transformacija činjenične tablice fact_election_data

8.3 Transformacije dimenzijskih tablica

Transformacija dimenzije person

Transformacijska funkcija `transform_person_dim`, prikazana u listingu 6, odgovorna je za izgradnju tablice `dim_person` koja sadrži podatke o kandidatima. Prvo se normalizira ulazni DataFrame iz MySQL baze — uklanjaju se prazna polja i duplikati te se podaci tipiziraju i čiste (npr. `trim` nad tekstualnim kolonama). Ako je dostupna dodatna CSV datoteka s istim podacima, ona se spaja s MySQL izvorom korištenjem operacije `unionByName`. Nakon toga, ako je dostupan DataFrame tablice `dim_party`, provodi se `join` kako bi se izvorni `party_id` zamijenio pripadajućim tehničkim ključem (`party_tk`) zato što su se u nekim djelovima ETL procesa slučajno pomješali TK-jevi i PK-jevi te je za svaku tablicu bilo potrebno provjeriti spajaju li se preko TK ili PK. Na kraju, koristi se funkcija `row_number()` za generiranje jedinstvenog surrogate ključa `person_tk`, kojim se završna tablica jednoznačno indeksira.

```

1 from pyspark.sql.functions import col, trim, row_number
2 from pyspark.sql.window import Window
3 from spark_session import get_spark_session
4
5 spark = get_spark_session()
6
7 print("Pokrecemo person_dim")
8
9 def transform_person_dim(mysql_person_df, csv_person_df=None,
10    party_dim_df=None):

```



```
10
11     mysql_df = (
12         mysql_person_df
13         .select(
14             col("id").cast("int"),
15             trim(col("first_name")).alias("first_name"),
16             trim(col("last_name")).alias("last_name"),
17             col("birth_year").cast("int"),
18             trim(col("title")).alias("title"),
19             col("party_id").cast("bigint")
20         )
21         .dropna(subset=["id", "first_name", "last_name", "birth_year",
22             "title", "party_id"])
23         .dropDuplicates(["id", "first_name", "last_name", "birth_year",
24             "title", "party_id"])
25     )
26
27     print(f"Person dim: MySQL podaci obradeni, broj redaka: {mysql_df.
28         count()}")
29
30     if csv_person_df:
31         csv_df = (
32             csv_person_df
33             .select(
34                 col("id").cast("int"),
35                 trim(col("first_name")).alias("first_name"),
36                 trim(col("last_name")).alias("last_name"),
37                 col("birth_year").cast("int"),
38                 trim(col("title")).alias("title"),
39                 col("party_id").cast("bigint")
40             )
41             .dropna(subset=["id", "first_name", "last_name", "
42                 birth_year", "title", "party_id"])
43             .dropDuplicates(["id", "first_name", "last_name", "
44                 birth_year", "title", "party_id"])
45         )
46
47         combined_df = mysql_df.unionByName(csv_df).dropDuplicates(["id
48             ", "first_name", "last_name", "birth_year", "title", "
49             party_id"])
50     else:
51         combined_df = mysql_df
52
53     print(f"Person dim: Kombinacija MySQL + CSV podataka završena,
54         broj redaka: {combined_df.count()}")
55
56     if party_dim_df is not None:
57         combined_df = combined_df.join(party_dim_df, combined_df.
58             party_id == party_dim_df.party_id, "left") \
59             .withColumn("party_id", party_dim_df.
60                 party_id) \
61             .drop("party_id")
62
63     print(f"Person dim: Join sa party_dim_df završen, broj redaka: {
64         combined_df.count()}")
65
66     window = Window.orderBy("id", "first_name", "last_name")
```

```

56     final_df = combined_df.withColumn("person_tk", row_number().over(
57         window)) \
58         .select("person_tk", "id", "first_name", "
59             last_name", "birth_year", "title", "
60             party_id")
61
62     print(f"Person dim: Dodan surrogate key, broj redaka: {final_df.
63         count()}")
64
65     return final_df.orderBy("person_tk")

```

Listing 6: Transformacija dimenzije dim_person

Transformacija dimenzije party

Transformacijska funkcija `transform_party_dim`, prikazana u listingu 7, obrađuje podatke o političkim strankama. Ulazni podaci dolaze primarno iz MySQL baze, dok se dodatni (opcionalni) podaci mogu dopuniti iz CSV izvora. U prvoj fazi funkcija čisti i tipizira podatke te ih povezuje s unaprijed definiranim rječnikom koji sadrži političke orijentacije po ID-u stranke. Ako je prisutan CSV izvor, njegovi se zapisi dodatno očiste, generira im se vlastiti ID pomoću `row_number()` i pridružuje im se orijentacija "unknown", nakon čega se podaci spajaju s MySQL izvorom. Završna faza uključuje generiranje tehničkog ključa `party_tk`, koji omogućuje jednoznačnu identifikaciju zapisa unutar skladišta podataka. Ova dimenzija ima posebnu važnost jer omogućuje analizu rezultata izbora kroz prizmu političke orijentacije i puni se povlačenjem podataka iz MySQL baze. Zatim se pridružuju orijentacije svakoj stranci koristeći `party_orientations` dictionary u kojem se za ključ uzima id stranke, a za vrijednost orijentacija stranke.

```

1  from pyspark.sql.functions import col, trim, row_number, lit
2  from pyspark.sql.window import Window
3  from spark_session import get_spark_session
4
5  spark = get_spark_session()
6  print("Pokrecemo party_dim...")
7
8  party_orientations = {
9      1: "left", 2: "right", 3: "center", 4: "right", 5: "center",
10     6: "left", 7: "center", 8: "left", 9: "center", 10: "right",
11     11: "right", 12: "left", 13: "center", 14: "center", 15: "left",
12     16: "left", 17: "center", 18: "right", 19: "center", 20: "center",
13     21: "left"
14 }
15
16 orientation_df = spark.createDataFrame(
17     [(k, v) for k, v in party_orientations.items()],
18     ["party_id", "orientation"]
19 )
20
21 def transform_party_dim(mysql_party_df, csv_party_df=None):
22     if mysql_party_df is None:
23         raise ValueError("MySQL DataFrame ne smije biti None!")
24
25     mysql_df = (
26         mysql_party_df

```

```

27         .select(
28             col("id").cast("int").alias("party_id"),
29             trim(col("name")).alias("name")
30         )
31         .dropna(subset=["party_id", "name"])
32         .dropDuplicates(["party_id", "name"])
33     )
34
35     mysql_df = mysql_df.join(orientation_df, on="party_id", how="left"
36     )
37     combined_df = mysql_df
38
39     if csv_party_df is not None:
40         csv_clean = (
41             csv_party_df
42             .select(trim(col("Party")).alias("name"))
43             .dropna(subset=["name"])
44             .dropDuplicates(["name"])
45         )
46
47         window_spec = Window.orderBy("name")
48         csv_clean = csv_clean.withColumn("party_id", row_number().over(
49             window_spec) + 1000)
50         csv_with_orientation = csv_clean.withColumn("orientation", lit(
51             "unknown"))
52
53         combined_df = (
54             mysql_df.unionByName(csv_with_orientation)
55             .dropDuplicates(["party_id", "name", "orientation"])
56         )
57
58         window = Window.orderBy("party_id", "name")
59         final_df = (
60             combined_df
61             .withColumn("party_id", row_number().over(window))
62             .select("party_id", "party_id", "name", "orientation")
63             .orderBy("party_id")
64         )
65
66     return final_df

```

Listing 7: Transformacija dimenzije dim_party

Također, kreirane su i određene helper funkcije za obogaćivanje podataka kroz neke dodatne attribute (taj se proces mogao odraditi ručnim insertovima, no ovako je bilo brže)

Transformacija dimenzije date

Dimenzija datuma omogućuje analizu vremenskih aspekata podataka, poput trendova po godinama, mjesecima ili danima u tjednu. Funkcija `transform_date_dim`, prikazana u listingu 8, koristi podatke o vremenu iz baze podataka i/ili CSV datoteka, koje se obrađuju pomoću pomoćne funkcije `normalize_time_df`. Ta funkcija pokušava automatski identificirati kolonu s datumom (npr. `time` ili `year`), standardizira format, pretvara vrijednosti u `TimestampType` i uklanja nedosljednosti poput duplikata i `NULL`

vrijednosti. Nakon što se svi izvori spoje u jednu koherentnu tablicu, izvode se dodatni atributi dimenzije — godina, mjesec, dan i dan u tjednu — te se svakom zapisu dodjeljuje tehnički ključ `date_tk`. Ova dimenzija omogućuje jednostavno vremensko filtriranje i grupiranje unutar OLAP sustava.

```
1 from pyspark.sql.functions import (
2     col, trim, to_timestamp, year, month, dayofmonth, date_format,
3     row_number
4 )
5 from pyspark.sql.types import TimestampType
6 from pyspark.sql.window import Window
7 from spark_session import get_spark_session
8 from pyspark.sql import DataFrame
9
10 def normalize_time_df(df: DataFrame, source_name="unknown") ->
11     DataFrame:
12     print(f"[normalize_time_df] Obrada izvora: {source_name}")
13
14     possible_date_columns = ["time", "year"]
15     date_column = next((col_name for col_name in possible_date_columns
16         if col_name in df.columns), None)
17
18     if not date_column:
19         raise ValueError(f"[{source_name}] Nema prepoznatljive kolone
20             s datumom. Dostupno: {df.columns}")
21
22     df = (
23         df
24         .withColumn("time_str", trim(col(date_column)))
25         .withColumn("time", to_timestamp("time_str", "yyyy-MM-dd HH:mm:ss"))
26         .dropna(subset=["time"])
27         .dropDuplicates(["time"])
28     )
29
30     null_count = df.filter(col("time").isNull()).count()
31     if null_count > 0:
32         print(f"[{source_name}] {null_count} neuspjesno parsiranih
33             redova (NULL)")
34
35     print(f"[{source_name}] Validnih redova: {df.count()}")
36     return df.select("time")
37
38 def transform_date_dim(mysql_date_df: DataFrame, csv_date_df:
39     DataFrame = None) -> DataFrame:
40     spark = get_spark_session()
41     print("Pokrecemo transformaciju date_dim")
42
43     sources = []
44
45     if mysql_date_df is not None:
46         sources.append(normalize_time_df(mysql_date_df, "MySQL"))
47     if csv_date_df is not None:
48         sources.append(normalize_time_df(csv_date_df, "CSV"))
49
50     if not sources:
51         raise ValueError("Nema dostupnih izvora (MySQL ni CSV)")
```

```

46 combined_df = sources[0]
47 for other_df in sources[1:]:
48     combined_df = combined_df.unionByName(other_df).dropDuplicates
49         ([ "time" ])
50
51 enriched_df = (
52     combined_df
53     .withColumn("year", year("time"))
54     .withColumn("month", month("time"))
55     .withColumn("day", dayofmonth("time"))
56     .withColumn("weekday", date_format("time", "EEEE"))
57 )
58
59 window = Window.orderBy("time")
60 final_df = (
61     enriched_df
62     .withColumn("date_tk", row_number().over(window))
63     .select("date_tk", "time", "year", "month", "day", "weekday")
64     .orderBy("time")
65 )
66
67 print(f"Transformacija završena. Ukupan broj redova: {final_df.
68     count()}")
69 return final_df

```

Listing 8: Transformacija vremenske dimenzije dim_date

Transformacija dimenzije country

Dimenzija `country` omogućuje analizu izbornih rezultata po teritorijalnim jedinicama. Funkcija `transform_country_dim`, prikazana u listingu 9, integrira podatke o državama iz različitih izvora — primarno iz MySQL baze, uz mogućnost dopune iz CSV datoteka. Na početku se čiste i formatiraju nazivi država i pripadajućih regija, nakon čega se uklanjaju duplikati i podaci sa već postojećim `country_id` vrijednostima, ako je omogućena provjera putem JDBC konekcije prema postojećem skladištu. Nedostajuće vrijednosti za regiju se popunjavaju nasumičnim izborom iz unaprijed definiranog skupa. Nakon toga se generira tehnički ključ `country_tk` korištenjem `row_number()` nad kombinacijom regije i imena. Završni rezultat uključuje jedinstvene i konzistentne zapise, spremne za učitavanje u skladište podataka.

```

1 from pyspark.sql.functions import col, trim, initcap, lit, row_number,
   when, expr
2 from pyspark.sql.window import Window
3 from spark_session import get_spark_session
4
5 spark = get_spark_session()
6 print("pokrećemo country_dim")
7
8 def transform_country_dim(mysql_country_df, csv_country_df=None,
   jdbc_url=None, connection_properties=None):
9     mysql_df = (
10         mysql_country_df
11         .select(
12             col("id").cast("bigint").alias("country_id"),

```

```

13         initcap(trim(col("name"))).alias("country_name"),
14         initcap(trim(col("region"))).alias("region")
15     )
16     .dropDuplicates(["country_name"])
17 )
18
19 if csv_country_df:
20     csv_df = (
21         csv_country_df
22         .select(initcap(trim(col("name"))).alias("name"))
23         .withColumn("country_id", lit(None).cast("long"))
24         .withColumn("region", lit(None).cast("string"))
25         .dropDuplicates(["name"])
26     )
27     combined_df = mysql_df.unionByName(csv_df)
28 else:
29     combined_df = mysql_df
30
31 if jdbc_url and connection_properties:
32     try:
33         existing_ids_df = spark.read.jdbc(url=jdbc_url, table="
34             dim_country", properties=connection_properties)
35         existing_ids = [row["country_id"] for row in
36             existing_ids_df.select("country_id").dropna().collect
37             ()]
38         if existing_ids:
39             combined_df = combined_df.filter(~col("country_id").
40                 isin(existing_ids))
41             print(f"Filtrirano {len(existing_ids)} ve
42                 postoje ih country_id-ova.")
43         else:
44             print("Nema postoje ih country_id-ova za filtrirati.")
45     except Exception as e:
46         print(f"Gre ka pri dohva anju postoje ih ID-eva: {e}")
47
48 combined_df = combined_df.dropDuplicates(["country_name"])
49
50 combined_df = (
51     combined_df
52     .withColumn("dup_flag", when(col("country_id").isNull(), lit
53         (-1)).otherwise(col("country_id")))
54     .dropDuplicates(["dup_flag"])
55     .drop("dup_flag")
56 )
57
58 regions = ["Center", "East", "South", "North", "West"]
59 combined_df = (
60     combined_df
61     .withColumn(
62         "region",
63         when(col("region").isNotNull(), col("region"))
64         .otherwise(expr(f"element_at(array({'','.join([f'\"{r}\"'
65             for r in regions]))), cast(rand() * {len(regions)} + 1
66             as int)))"))
67 )

```

```

61 window = Window.orderBy("region", "country_name")
62 final_df = (
63     combined_df
64     .withColumn("country_tk", row_number().over(window))
65     .select("country_tk", "country_id", "country_name", "region")
66 )
67
68 final_df = final_df.withColumn("rownum", row_number().over(Window.
69     partitionBy("country_id").orderBy("country_tk")))
70 final_df = final_df.filter((col("country_id").isNull()) | (col("
71     rownum") == 1)).drop("rownum")
72
73 try:
74     print("Broj redaka u dimenziji drzava:", final_df.count())
75 except Exception as e:
76     print("Greska pri brojanju redova:", str(e))
77     final_df.show(5)
78     final_df.printSchema()
79
80 print("Final country_dim rows:", final_df.count())
81
82 return final_df

```

Listing 9: Transformacija dimenzije dim_country

Transformacija dimenzije election

Funkcija `transform_election_dim` služi za transformaciju i integraciju podataka o izborima prikupljenih iz dva izvora: MySQL baze podataka i eventualno CSV datoteke. U prvom koraku, iz MySQL DataFrame-a se biraju ključni stupci `id`, `year` i `country_id`, gdje se `id` i `country_id` konvertiraju u odgovarajuće numeričke tipove, a `year` se obrađuje kao string koji se zatim pretvara u datum formata `yyyy-MM-dd`. Tako očišćeni podaci prolaze kroz proces filtriranja gdje se uklanjaju redci s nedostajućim vrijednostima i duplicirani zapisi. Ukoliko je dostupan CSV DataFrame, isti proces normalizacije se provodi i na njemu, a zatim se oba skupa podataka spajaju u jedan jedinstveni DataFrame uz dodatno uklanjanje duplikata.

Nakon toga, integrirani DataFrame se povezuje s dimenzijskim tablicama `date_dim` i `country_dim` kako bi se zamijenile originalne vrijednosti datuma i države njihovim surrogatnim ključevima, čime se omogućuje konzistentnost i optimizacija za skladištenje podataka.

U završnoj fazi, funkcija dodjeljuje jedinstveni surrogatni ključ `election_tk` svakom retku koristeći `row_number()` funkciju preko definirane particije i redoslijeda, što rezultira numeričkim indeksom koji služi kao primarni ključ u dimenzijskoj tablici izbora. Konačni rezultat je sortirani i očišćeni DataFrame spreman za daljnju analizu ili učitavanje u skladište podataka. Time se osigurava da podaci budu dosljedni, bez nepotrebnih dupliciranja te da su pravilno povezani s ostalim dimenzijama u sustavu.

```

1 from pyspark.sql.functions import col, trim, row_number, to_date
2 from pyspark.sql.window import Window
3 from spark_session import get_spark_session

```

```
4
5 spark = get_spark_session()
6
7 print("pokrecemo election_dim")
8 def transform_election_dim(mysql_election_df, csv_election_df=None,
9                             date_dim_df=None, country_dim_df=None):
10     print("usli smo u funkciju transform election dim")
11
12     # --- Step 1: Normalize MySQL data ---
13     mysql_df = (
14         mysql_election_df
15         .select(
16             col("id").cast("int"),
17             trim(col("year")).alias("election_date_str"),
18             col("country_id").cast("bigint")
19         )
20         .withColumn("election_date", to_date("election_date_str", "
21             yyyy-MM-dd"))
22         .dropna(subset=["id", "election_date", "country_id"])
23         .dropDuplicates(["id", "election_date", "country_id"])
24
25     # --- Step 2: Normalize CSV data (if provided) ---
26     if csv_election_df:
27         csv_df = (
28             csv_election_df
29             .select(
30                 col("id").cast("int"),
31                 trim(col("election_date")).alias("election_date_str"),
32                 col("country_id").cast("bigint")
33             )
34             .withColumn("election_date", to_date("election_date_str", "
35                 yyyy-MM-dd"))
36             .dropna(subset=["id", "election_date", "country_id"])
37             .dropDuplicates(["id", "election_date", "country_id"])
38
39         combined_df = mysql_df.unionByName(csv_df).dropDuplicates(["id
40             ", "election_date", "country_id"])
41     else:
42         combined_df = mysql_df
43
44     # --- Step 3: Join with dim_date and dim_country to replace
45     # date_id and country_id with surrogate keys ---
46     if date_dim_df is not None:
47         combined_df = combined_df.join(date_dim_df, combined_df.
48             election_date == date_dim_df.time, "left") \
49             .withColumn("election_date",
50                 date_dim_df.time) \
51             .drop("time") # Drop the original
52                             date after join
53
54     if country_dim_df is not None:
55         combined_df = combined_df.join(country_dim_df, combined_df.
56             country_id == country_dim_df.country_id, "left") \
57             .withColumn("country_id",
58                 country_dim_df.country_id)
```



```

51
52 # --- Step 4: Add surrogate key ---
53 window = Window.orderBy("id", "election_date")
54 final_df = combined_df.withColumn("election_tk", row_number().over
    (window)) \
55     .select("election_tk", "id", "
        election_date_str", "country_id")
56
57 # Provjera uspješnosti (ispis broja redaka)
58 row_count = final_df.count()
59 print(f"WOOHOO! Sve je OK! Broj redaka u finalnom DataFrame-u: {
    row_count}")
60
61 return final_df.orderBy("election_tk")

```

Listing 10: Transformacija dimenzijske tablice *dim_election*

Transformacija dimenzije election history

Listing 11 prikazuje Python skriptu koja puni dimenzijsku tablicu *dim_election_history* iz CSV datoteke koristeći Pandas i MySQL konekciju. Skripta se sastoji od nekoliko ključnih koraka: učitavanje CSV datoteke, konverzija vremena, izračunavanje izlaznosti birača u povijesnom kontekstu (atribut *historical_turnout*), te umetanje ili ažuriranje podataka u tablici.

Prvo se provjerava postoji li već zapis za određeni teritorij koji ima oznaku *is_current* = *TRUE*. Ako postoji, taj se zapis ažurira tako da mu se postavlja atribut *date_to* na trenutni datum i označava kao neaktivan (*is_current* = *FALSE*). Nakon toga se unosi nova verzija zapisa s povećanom verzijom i *is_current* = *TRUE*.

Ova dimenzija je izrazito složena zbog višestrukih referenciranja i povezivanja s drugim tablicama u skladištu podataka. Svaka promjena podataka zahtijeva čuvanje povijesnog zapisa, što je ostvareno konceptom *Slowly Changing Dimension Type 2* (SCD2), gdje se čuva puna povijest promjena svakog teritorija s vremenskim oznakama *date_from* i *date_to*. Time se omogućuje praćenje kako su se atributi teritorija mijenjali tijekom vremena, što je ključno za analitičke potrebe.

```

1 import mysql.connector
2 import pandas as pd
3 from datetime import datetime
4
5 db_connection = mysql.connector.connect(
6     host="localhost",
7     user="root",
8     password="root",
9     database="dimensional_database"
10 )
11
12 cursor = db_connection.cursor()
13 df = pd.read_csv('ElectionData.csv')
14
15 def convert_time_format(time_str):
16     try:
17         return datetime.strptime(time_str, "%H:%M:%S").strftime("%H:%M:%S")

```

```
18     except Exception as e:
19         return None
20
21 def calculate_historical_turnout(total_voters, subscribed_voters):
22     if subscribed_voters != 0:
23         return (total_voters / subscribed_voters) * 100
24     else:
25         return None
26
27 row_count = 0
28
29 for index, row in df.iterrows():
30     time_elapsed = convert_time_format(row['TimeElapsed'])
31     timestamp = datetime.strptime(row['time'], "%Y-%m-%d %H:%M:%S")
32     territory_name = row['territoryName']
33     total_mandates = row['totalMandates']
34     available_mandates = row['availableMandates']
35     num_parishes = row['numParishes']
36     num_parishes_approved = row['numParishesApproved']
37     blank_votes = row['blankVotes']
38     blank_votes_percentage = row['blankVotesPercentage']
39     null_votes = row['nullVotes']
40     null_votes_percentage = row['nullVotesPercentage']
41     total_voters = row['totalVoters']
42     subscribed_voters = row['subscribedVoters']
43     voters_percentage = row['votersPercentage']
44     pre_blank_votes = row['pre.blankVotes']
45     pre_null_votes = row['pre.nullVotes']
46
47     historical_turnout = calculate_historical_turnout(total_voters,
48                                                       subscribed_voters)
49
50     select_query = """
51     SELECT version, date_to FROM dim_election_history
52     WHERE territory_name = %s AND is_current = TRUE
53     ORDER BY version DESC LIMIT 1
54     """
55     cursor.execute(select_query, (territory_name,))
56     result = cursor.fetchone()
57
58     if result:
59         version, date_to = result
60         update_query = """
61         UPDATE dim_election_history SET date_to = NOW(), is_current =
62             FALSE
63         WHERE territory_name = %s AND version = %s
64         """
65         cursor.execute(update_query, (territory_name, version))
66         row_count += 1
67
68     insert_query = """
69     INSERT INTO dim_election_history (
70         version, date_from, date_to, territory_name, total_mandates,
71         available_mandates,
```

```

71         subscribed_voters ,  
            voters_percentage , pre_blank_votes , pre_null_votes ,  
                historical_turnout , is_current  
72     ) VALUES (  
73         %s, NOW() , NULL , %s , %s , %s , %s , %s , %s , %s , %s , %s , %s , %  
            s , %s , %s , %s , TRUE  
74     )  
75     "" "  
  
76  
77     version = 1  
78     data = (  
79         version , territory_name , total_mandates , available_mandates ,  
            num_parishes ,  
80         num_parishes_approved , blank_votes , blank_votes_percentage ,  
            null_votes ,  
81         null_votes_percentage , total_voters , subscribed_voters ,  
            voters_percentage ,  
82         pre_blank_votes , pre_null_votes , historical_turnout  
83     )  
84  
85     try:  
86         cursor.execute(insert_query , data)  
87         db_connection.commit()  
88     except mysql.connector.Error as err:  
89         print(f"Greska: {err}")  
90         db_connection.rollback()  
91  
92     print(f"Ukupno je a uneseno {row_count} zapisa.")  
93     cursor.close()  
94     db_connection.close()
```

Listing 11: Transformacija dimenzijske tablice `dim_election_history` iz CSV-a

TU IH POHITAT :)

8.4 Load na primjeru izbora

U završnoj fazi ETL procesa, implementirana je funkcija `write_spark_df_to_mysql`, koja omogućuje učitavanje transformiranih podataka u ciljno skladište — u ovom slučaju MySQL bazu podataka naziva `zadnji`. Funkcija prima tri parametra: Spark DataFrame (`spark_df`), naziv tablice u koju se podaci učitavaju (`table_name`) i način učitavanja (`mode`), koji je prema zadanim postavkama postavljen na `append`, što znači da se novi podaci dodaju postojećima. Korištenjem metode `write.jdbc()`, podaci se trajno zapisuju u bazu uz poštivanje definiranih konekcijskih svojstava (korisničko ime, lozinka i JDBC driver). Ovakav pristup omogućuje automatizirano, fleksibilno i višekratno učitavanje podataka bez potrebe za ručnom intervencijom, što je jedna od ključnih prednosti faze *Load* u ETL arhitekturi.

```
1 from pyspark.sql import DataFrame
2
3 print("pokrecemo run_loading.py")
4
5 def write_spark_df_to_mysql(spark_df: DataFrame, table_name: str, mode
  : str = "append"):
6     jdbc_url = "jdbc:mysql://127.0.0.1:3306/zadnji?useSSL=false"
```

```

7  print(jdbc_url)
8  connection_properties = {
9      "user": "root",
10     "password": "root",
11     "driver": "com.mysql.cj.jdbc.Driver"
12 }
13
14 print(f"Writing to table '{table_name}' with mode '{mode}'...")
15 spark_df.write.jdbc(
16     url=jdbc_url,
17     table=table_name,
18     mode=mode,
19     properties=connection_properties
20 )
21 print(f" Done writing to '{table_name}'.")

```

Listing 12: Učitavanje Spark DataFrame-a u MySQL bazu

Po uspješnom završetku ETL procesa, u dimenzijski su model pohranjeni podaci, što se može provjeriti jednostavnim selectovima iz svake tablice.

1) *dim_country*

country_tk	country_id	name	region
1	4	Braga	Center
2	11	Leiria	Center
3	12	Lisboa	Center
4	6	Castelo Branco	East
5	9	Faro	East
6	13	Madeira	East
7	10	Guarda	North

Slika 7: dim_country

Dimenzija country predstavlja izborne jedinice i sastoji se od tehničkog ključa, ključa izborne jedinice, naziva i regije. Svaki je naziv unikatan. Regija označava geografsku pripadnost pojedinom djelu Portugala. Omogućava pregled rezultata sa teritorijalnog aspekta.

2) *dim_date*

45	2019-10-06	2019	10	6	Sunday	2019-10-07 05:00:00
46	2019-10-06	2019	10	6	Sunday	2019-10-07 05:15:00
47	2019-10-07	2019	10	7	Monday	2019-10-07 05:30:00
48	2019-10-07	2019	10	7	Monday	2019-10-07 05:45:00

Slika 8: dim_date

Dimenzija date predstavlja vremensku dimenziju. Konkretno, to su dani kada su se izbori odvijali. Sadrži timestempove na 2 datuma (6.10 i 7.10) u intervalima od 15

minuta. Dodane su i vrijednosti koje označavaju redni broj dana u tjednu i naziv tog dana. Omogućava pregled rezultata kroz vrijeme.

3) *dim_person*

	person_tk	person_id	first_name	last_name	birth_year	title	party_id	gender
▶	1	1	Manuel	Martins	1962	Prof.	215	M
	2	2	Maria	Ferreira	1952	Prof. ^a	216	F
	3	3	Miguel	Lopes	1991	Dr.	217	M
	4	4	Tiago	Ferreira	1950	Dr.	218	M
	5	5	Joana	Silva	1995	Sra.	219	F

Slika 9: *dim_person*

Dimenzija *person* predstavlja kandidate na izborima. Za svakog se kandidata bilježe ime i prezime, godina rođenja, spol, titula, stranka i spol. Omogućava pregled rezultata kroz prizmu izbornih pobjednika/gubitnika.

4) *dim_election*

election_tk	election_id	date_tk	country_id
265	265	13	250
266	266	13	251
267	267	13	252
268	268	13	253
269	269	13	254

Slika 10: *dim_election*

Dimenzija *election* je izrazito jednostavna, ali bitna. Naime, ona služi za praćenje samih izbora i povezuje više podataka iz ostalih dimenzija. Konkretno, povezuje svaki izborni zapis s pripadnim datumom i izbornom jedinicom.

5) *dim_election_history*

Dimenzija *election_history* predstavlja povijesne zapise tijeka izbora. Strukturirana je tako da omogućuje praćenje promjena kroz vrijeme. Svaki redak pohranjuje verziju (*version*) podataka o pojedinoj izornoj jedinici (*territory_name*), s početnim i završnim datumom važenja (*date_from*, *date_to*) koji se ažuriraju pri svakom novom dodavanju zapisa (u pravilu intervali od 15 minuta, no može se prilagoditi) te ključnim statističkim pokazateljima: broj i raspoloživost mandata (*total_mandates*, *available_mandates*), broj izbornih jedinica i njihov status odobrenja (*num_parishes*, *num_parishes_approved*), udjelu praznih i nevažećih listića (*blank_votes*, *blank_votes_percentage*, *null_votes*, *null_votes_percentage*), kao i ukupnim podacima o biračima i izlaznosti (*total_voters*, *subscribed_voters*, *voters_percentage*, *historical_turnout*). Polje *election_id* povezuje zapis s konkretnim zapisom o izborima, a flag *is_current* označava aktivnu (trenutnu) verziju. Naravno, samo je finalni zapis označen kao *is_current* pošto su u skladište već učitani svi podaci od početka do kraja izbora. No, pri "live" dodavanju, posljednji dodani redak uvijek bi se označavao kao *current* sve dok ne pristigne noviji.

6) *dim_party*

	party_tk	party_id	name	orientation
►	215	1	PS	Center
	216	2	PPD/PSD	Left
	217	3	B.E.	Center
	218	4	CDS-PP	Center
	219	5	PCP-PEV	Center
	220	6	PAN	Center
	221	7	CH	Right

Slika 11: *dim_party*

Dimenzija *party* predstavlja sve stranke koje imaju svoje kandidate na izborima te se njen strani ključ nalazi u dimenziji *person* što označava da svaka osoba pripada nekoj stranci, dok 1 stranci može pripadati više osoba. Za svaku stranku bilježi se naziv (zapravo, skraćenica naziva) i orijentacija koja predstavlja dodatnu dimenziju koja omogućava pregled stranaka po političkoj orijentaciji (lijevo/centar/desno) i dodatan, dublji uvid u podatke i rezultate što će biti vidljivo tijekom vizualizacije.

9. Interaktivni web-based dashboard i vizualizacija podataka

Za potrebe vizualizacije podataka iz dimenzijskog modela te omogućavanja korisnicima intuitivne i interaktivne analize izbornih rezultata, izrađen je prilagođeni web-based dashboard. Umjesto korištenja gotovih alata kao što su Tableau ili Power BI, koji su često ograničeni u fleksibilnosti i prilagodbi specifičnim potrebama domene, odlučili smo se za izradu vlastitog rješenja. Bilo bi šteta osloniti se na generičke alate kad je moguće dizajnirati vizualizacijsko sučelje koje precizno odgovara strukturi i logici našeg dimenzijskog modela.

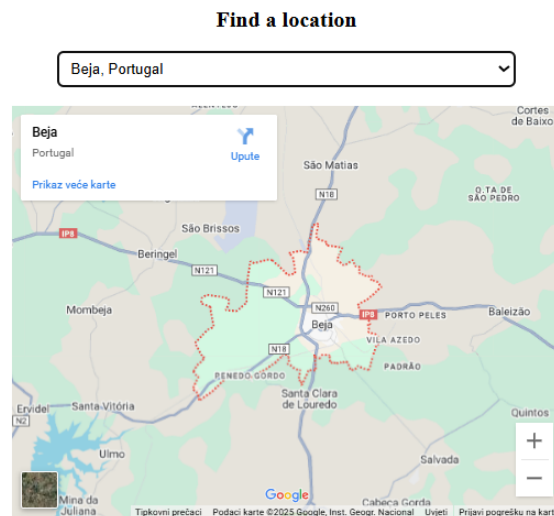
Za implementaciju su korištene sljedeće tehnologije:

- **Vue.js** – kao frontend JavaScript framework za izgradnju responzivnog i interaktivnog korisničkog sučelja.
- **Matplotlib** – za generiranje statičkih i dinamičkih grafova na strani servera, posebno prikladno za analize koje zahtijevaju preciznu kontrolu nad prikazom.
- **Metabase** – za jednostavno i brzo generiranje uvida i osnovnih vizualizacija direktno nad bazom, korišten pretežito tijekom razvoja i verifikacije podataka.

Ovaj pristup omogućio je kreiranje fleksibilnog dashboarda koji se može lako prilagoditi promjenama u modelu podataka, nadograditi dodatnim funkcionalnostima te povezati s analitičkim funkcijama specifičnim za političku domenu i praćenje izbornih trendova.

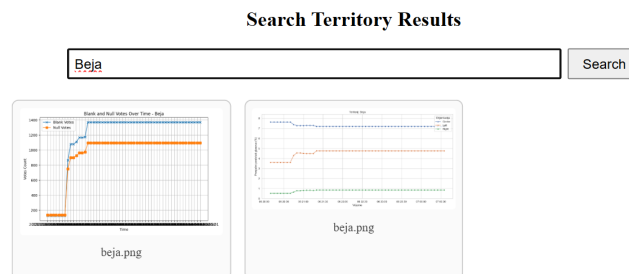
U nastavku će najprije biti prikazane funkcionalnosti dashboarda, a potom svi grafikoni i njihova interpretacija u kontekstu izbora.

- Implementiran je dinamički pronalazak pojedinih izbornih jedinica(`dim_country`) na interaktivnoj karti Portugala.



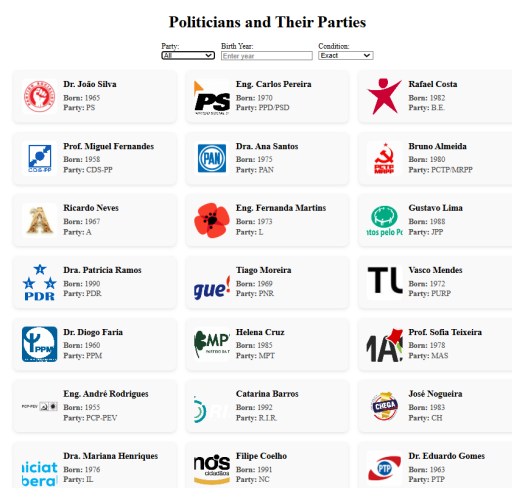
Slika 12: Interaktivna karta Portugala

- Omogućeno je pretraživanje rezultata po teritoriju, specifično prevladavajućih političkih orijentacija te pregled broja praznih i nevažećih glasova. Ta funkcionalnost odgovara OLAP Slice operaciji jer omogućava filtriranje po 1 dimenziji (teritorij)



Slika 13: Rezultati po teritoriju

- Moguće je pregledati sve kandidate, stranke kojima pripadaju i njihove autentične grbove te ih filtrirati po određenim kriterijima poput pripadnosti stranci ili godini rođenja. Ta funkcionalnost odgovara OLAP Dice operaciji jer omogućava filtriranje po više dimenzija istovremeno.

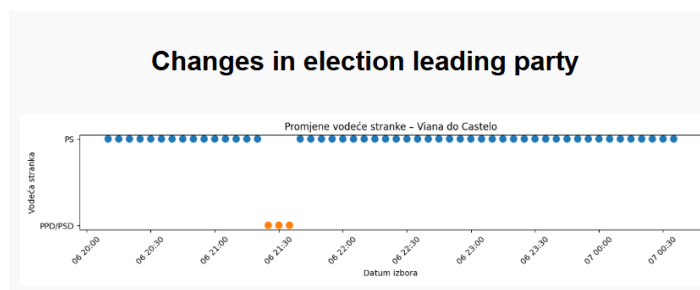


Slika 14: Pregled stranaka

- Zatim, kako je u skladištima podataka važno pratiti promjene kroz različite verzije podataka, sustav omogućuje pregled povijesnih rezultata izbora za svaki teritorij, pri čemu teritorij odgovara jednoj izbornoj jedinici. Iako podaci u odabranom datasetu ne prate promjene iz godine u godinu već samo kroz 1 izbore, omogućena je usporedba rezultata kroz različite izborne cikluse, što je dovoljno za analizu promjena u izbornim rezultatima tijekom vremena.

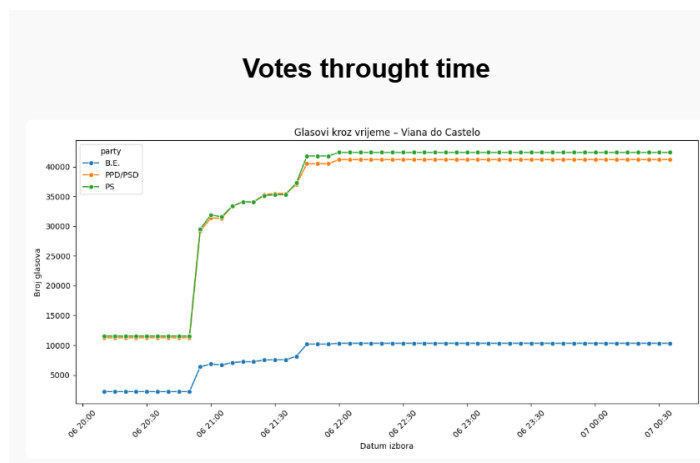
Za svaki teritorij moguće je vidjeti kako se mijenjao broj glasova po strankama kroz izbore, kako se ti brojevi izražavaju u postocima, koja je stranka bila vodeća u kojem ciklusu, te kako je izgledala konačna raspodjela mandata unutar teritorija. Ovakva analiza omogućuje korisnicima uvid u trendove i pomake u biračkom tijelu na razini izbornih jedinica, što može biti korisno za stranke, analitičare i istraživače.

Konkretno, ti su podaci obrađeni na 4 načina: promjenama u vodećoj poziciji kroz vrijeme, broju mandata po stranci, postotku glasova kroz vrijeme i broju glasova kroz vrijeme. Ti su podaci dostupni za svaku jedinicu, a primjera radi, odabran je teritorij Viana do Castelo



Slika 15: Pregled vodećih stranaka

Na grafikonu je vidljivo kako se mijenjala vodeća stranka kroz vrijeme te se može iščitati kako je gotovo cijelo vrijeme, osim kratkog perioda, u vodstvu bila stranka



Slika 18: Promjena broja glasova kroz vrijeme

Nadalje, iduća dva grafikona u kontekstu OLAP operacija predstavljaju analitički vrlo sličan pogled na podatke, budući da oba prikazuju promjenu kvantitativne vrijednosti kroz vrijeme za određenu stranku — jedan kroz apsolutni broj glasova, a drugi kroz postotak osvojenih glasova.

Ova analiza podrazumijeva primjenu kombinacije slice i drill-down operacija. Slice se koristi za filtriranje podataka prema određenoj stranci — fokusira se samo na jednu političku opciju iz cjelokupnog skupa podataka. Nakon toga, pomoću drill-down operacije, analitički pogled ide dublje u vremensku dimenziju, omogućujući korisniku da promatra kako se ta vrijednost mijenjala kroz pojedine izborne cikluse.

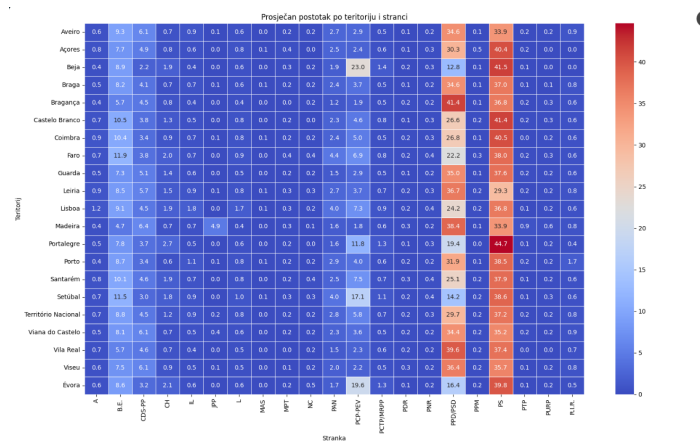
Uspoređivanjem oba grafikona paralelno, može se jasno uočiti način na koji se apsolutna brojka glasova preslikava u postotni udio, što dodatno doprinosi razumijevanju relativne snage stranke u kontekstu ukupnog broja birača. Time se ne samo promatra trend podrške određenoj stranci, već se omogućuje i dublja interpretacija rezultata, jer se broj glasova stavlja u odnos prema ukupnom biračkom tijelu — što može otkriti promjene koje na prvi pogled nisu očite samo iz apsolutnih vrijednosti.

Sljedeća primijenjena OLAP operacija jest pivot. Iako u ovom kontekstu nije naročito korisna u usporedbi s prikazima temeljenima na operacijama poput slice, dice ili drill-down, ipak može pružiti određeni uvid u podatke. Pivot, za razliku od prethodno navedenih operacija koje otkrivaju nove obrasce ili omogućuju filtraciju i agregaciju, prvenstveno služi za reorganizaciju prikaza podataka - tj. za zamjenu redaka i stupaca radi preglednosti ili naglašavanja određenih odnosa među dimenzijama.

U konkretnom slučaju, pivot operacija omogućuje da se, primjerice, dimenzija "stranka" prikaže kao stupci umjesto redaka, dok se izborne jedinice prikazu po redovima, ili obrnuto. Takva promjena može pomoći u vizualnom uspoređivanju više entiteta istovremeno, ali sama po sebi ne dodaje novu informaciju, već samo prezentira postojeće podatke iz druge perspektive. Upravo zato, iako tehnički spada u skup osnovnih OLAP operacija, pivot u ovom slučaju ne doprinosi inter-

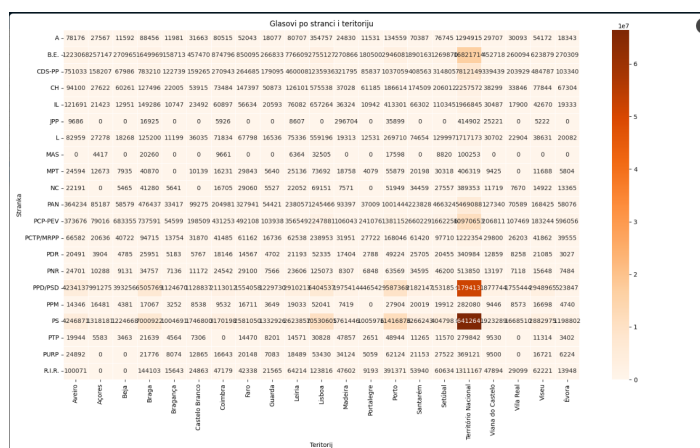
pretaciji podataka u istoj mjeri kao operacije koje filtriraju, agregiraju ili istražuju vremenske ili prostorne obrasce.

Svejedno, njezina vrijednost leži u mogućnosti da korisnicima s različitim preferencijama u načinu čitanja tabličnih podataka ponudi alternativni prikaz koji im može olakšati preglednost i usporedbu — ali ne i otkriti nešto što već nije bilo prisutno.



Slika 19: Prosječan postotak po teritoriju i stranci

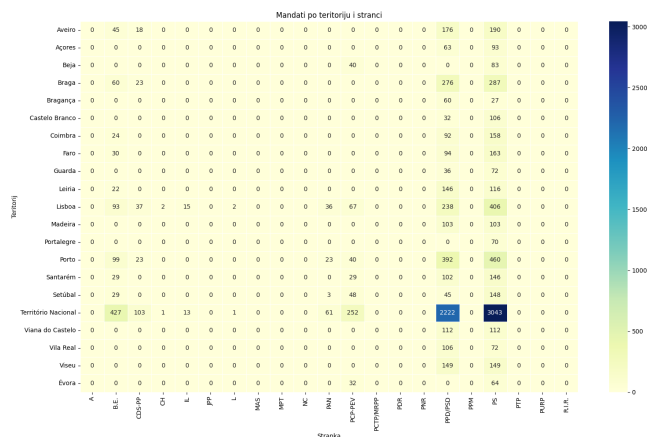
Prvi pivot grafikon omogućuje uvid u prosječan postotak koji je svaka stranka osvojila u svakom teritoriju kad se u obzir uzme cijeli vremenski period izbora. Prema boji je vidljivo kako su pojedine stranke "dominantne" te osvajaju značajno više mandata kroz sve teritorije u odnosu na konkurentske strane. (konkretno, stranke PS i PPD/PSD)



Slika 20: Broj glasova po stranci i teritoriju

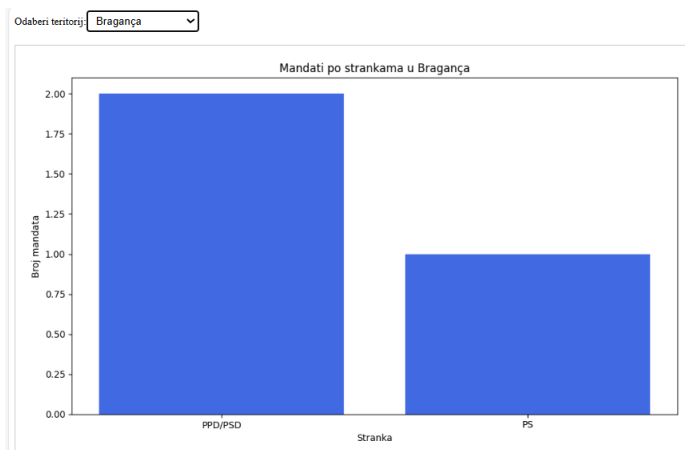
Drugi pivot grafikon omogućuje prikazuje broj glasova svake stranke na svakom teritoriju što daje uvid u neke ključne detalje. Konkretno, omogućuje da se uvidi koja je stranka dominantna na kojem pojedinom teritoriju te omogućuje

da se uvidi koji su teritoriji ključni u samim izborima (zbog svoje veličine i broja glasova/mandata koje oni daju).



Slika 21: Broj mandata po stranci i teritoriju

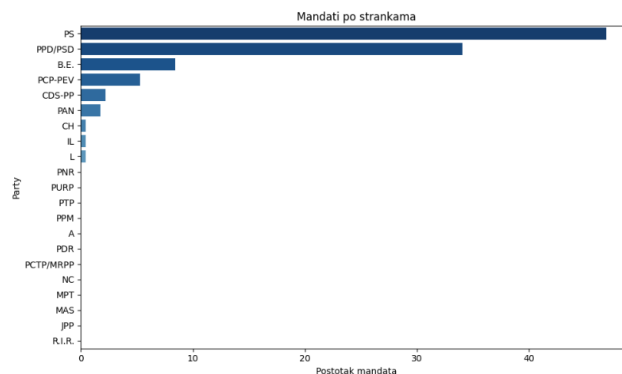
Za treći pivot koji prikazuje broj mandata po stranci i teritoriju vrijedi isto što i za drugi koji prikazuje broj glasova, no posebno su zanimljivi kada se promatraju u kombinaciji jer tada omogućuju uvid u čisto matematičko preslikavanje glasova u mandate (prije primjene D'Hondtove metode koja često nije "poštena" jer favorizira velike stranke, dok malim "lokalnim" strankama onemogućava značajnije političke uspjehe. Jer čak i ako osvoje dovoljan broj glasova, to često nije dovoljno za dobivanje mandata zato što D'Hondtova metoda ne vrednuje sve glasove u svim teritorijima jednako).



Slika 22: Slice konačnih rezultata po teritoriju

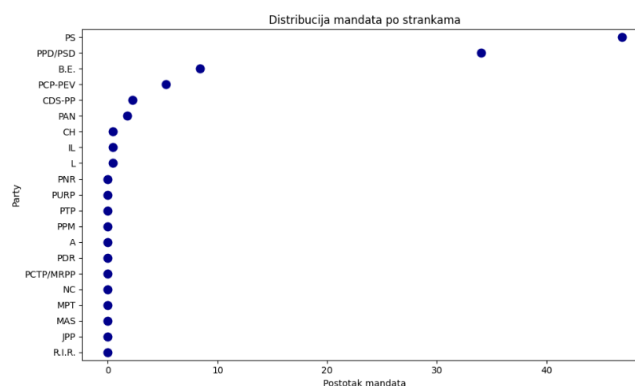
Zatim, omogućen je prikaz konačnih rezultata po teritoriju što je klasičan primjer slice operacije zato što ona uzima finalne rezultate i "filtrira" ih za 1 teritorij. Na sličan se način može prikazati dice kada bi se promatralo rezultate samo za određene stranke samo na određenim teritorijima. A ukoliko bi se uzelo ukupne rezultate svih stranaka i promatralo ih se (agregiralo) na razini cijele države, bio bi to roll-up.

Naravno, omogućen je i prikaz konačnih rezultata na razini cijele države, jer filtriranje i OLAP operacije ne bi imale smisla kada ne bi bilo moguće vidjeti "sveukupne" podatke. Prikaz postotka ukupnih osvojenih mandata prikazan je na 3 načina te omogućava promatranje istih podataka iz 3 perspektive.



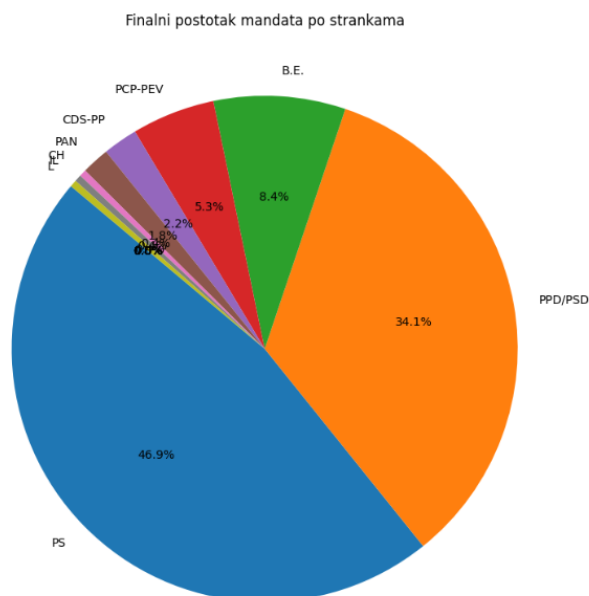
Slika 23: Stupčasti grafikon postotka mandata po strankama

Prvi grafikon omogućuje uvid u postotak osvojenih mandata po strankama. Dodatno, daje i uvid u to koje su sve stranke uopće ušle u parlament. Taj bi se grafikon mogao zarotirati i na način da se stranke nalaze na x-osi no, ponovno, ta operacija ne bi dala neke značajne nove uvide u same rezultate.



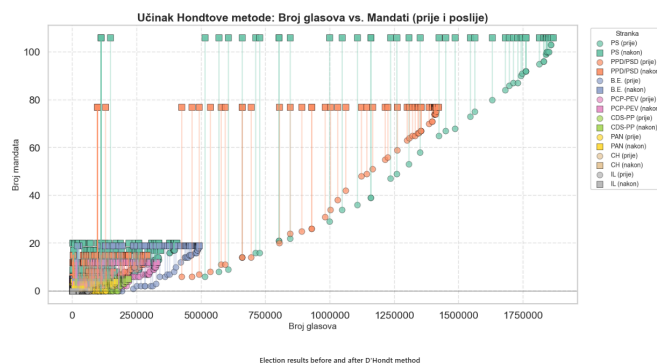
Slika 24: Distribucija mandata po strankama u postotcima

Drugi grafikon omogućuje uvid u distribuciju mandata po strankama što je jasan indikator toga da nisu sve stranke jednako "popularne" te da PS i PPD/PSD osvajaju znatno više mandata od konkurencije.



Slika 25: Pie chart konačnih rezultata

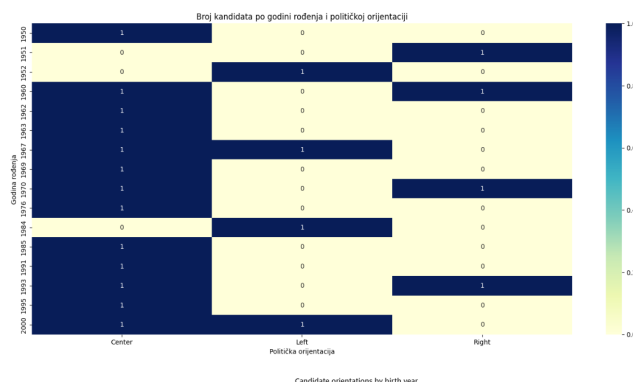
Pie chart konačnih rezultata daje točan i direktan uvid u odnose među rezultatima stranaka i pobjednike izbora. Vidljivo je kako je pobjednička stranka osvojila više nego sve ostale stranke zajedno (s izuzetkom drugoplasirane stranke).



Slika 26: Prikaz utjecaja D'Hondtove metode

D'Hondtova metoda je najčešće korišteni sustav za pretvorbu glasova u mandate u proporcionalnim izbornim sustavima. Radi se o matematičkom algoritmu koji teži osigurati razmjernu zastupljenost političkih opcija prema broju osvojenih glasova, no u praksi lagano favorizira veće stranke, budući da im omogućuje lakše osvajanje dodatnih mandata. Upravo zbog tog efekta, metoda se često smatra manje pravednom prema manjim strankama, iako je jednostavna za primjenu i stabilna u raspodjeli rezultata. Vidljivo je kako su manje stranke sve zbijene u donji lijevi kut dok se veće stranke prostiru prema gore-desno što označava da su dobile znatno više mandata.

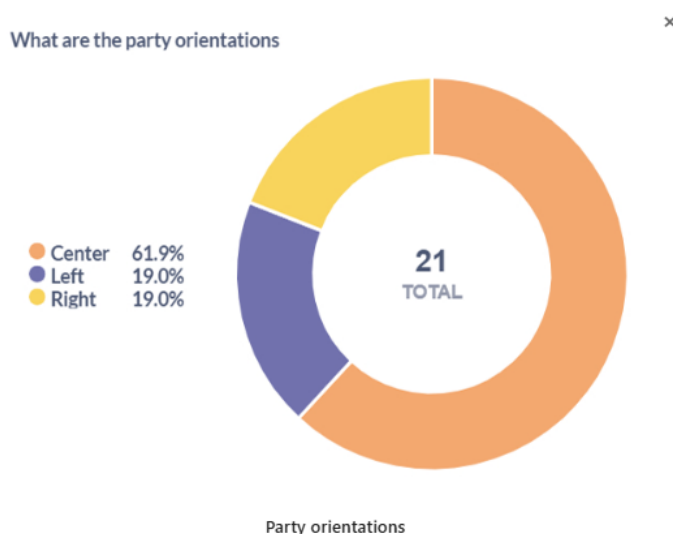
Iako prikaz nije rezultat klasične OLAP operacije, temelji se na konceptima agregacije i transformacije podataka koji su ključni za OLAP sustave. Prikaz omogućuje analitičarima uvid u učinkovitost stranačke konverzije glasova u mandate, što bi se u OLAP kontekstu moglo interpretirati kao analiza mjera izvedenih iz postojećih dimenzija.



Slika 27: Prikaz broja kandidata za kombinacije političke orijentacije i godine rođenja

Graf prikazuje broj kandidata prema godini rođenja i političkoj orijentaciji, u obliku heatmapa. Na y-osi su grupirane godine rođenja, a na x-osi političke orijentacije (lijevo, centar, desno). Svaka ćelija prikazuje koliko je kandidata pripadalo toj kombinaciji. Boje omogućuju brzo uočavanje obrazaca, poput većeg broja centrističkih kandidata među onima rođenima između 1960-1980.

U OLAP kontekstu, ovdje je primijenjena dice operacija, jer se istovremeno promatraju i filtriraju podaci prema dvjema dimenzijama: godina rođenja i politička orijentacija. Time se dobiva jasniji pregled odnosa između generacijske pripadnosti i ideološke orijentacije. Također, takav pristup potencijalno omogućava kreiranje predikcija navika glasanja pojedinih birača ovisno o godini rođenja.



Slika 28: Prikaz broja kandidata za kombinacije političke orijentacije i godine rođenja

Prikaz političkih orijentacija stranaka u obliku tortnog dijagrama temelji se na OLAP roll-up operaciji. U ovom slučaju, početni podaci sadrže 21 pojedinačnu stranku, koje se zatim agregiraju prema odabranoj dimenziji — političkoj orijentaciji (lijevo, centar, desno). Time se sve stranke svode na tri grupe, što omogućuje sažet i pregledan prikaz strukture političkog spektra.

title	number	gender
Prof.	1	M
Prof.*	3	F
Dr.	6	M
Sra.	2	F
Dra.	1	F
Eng.	4	M
Eng.*	2	F
Sr.	2	M

8 rows

Titles by gender

Slika 29: Prikaz broja titula po spolu

Tablični prikaz predstavlja OLAP slice operaciju po dimenziji spola kako bi se prikazao broj osoba za svaku titulu prema spolu. Dobiven je pregled titula raspoređenih na muške i ženske, s pripadajućim brojem za svaku kombinaciju. Ovi se podaci mogu i roll-upati kako bi se prikazao broj kandidata po tituli neovisno o spolu.

Title	Number
Prof.	4
Dr.	7
Sr.	4
Eng.	6

Tablica 2: Primjena roll-upa na podatke

Zaključak

U okviru ovog seminarskog rada izrađeno je skladište podataka temeljeno na stvarnim podacima o parlamentarnim izborima. Prvo su prikupljeni i analizirani dostupni podaci, nakon čega je definiran odgovarajući relacijski model koji opisuje odnose među entitetima. Na temelju njega izrađen je i dimenzijski model prilagođen analitičkom pristupu, s jasno definiranim činjenicama i dimenzijama.

Implementiran je ETL proces kojim su podaci iz izvornog formata transformirani i učitani u podatkovno skladište uz zadržavanje povijesne konzistencije. Posebna pažnja posvećena je složenijim dimenzijama koje zahtijevaju praćenje verzija i promjena kroz vrijeme. U svrhu analize podataka izrađene su vizualizacije koje prikazuju ključne pokazatelje poput izlaznosti birača, broja glasova i udjela pojedinih stranaka.

Kao završni korak, izrađena je interaktivna nadzorna ploča (dashboard) koja omogućuje pregled i usporedbu izbornih rezultata kroz različite dimenzije i vremenske periode. Cijeli proces demonstrira mogućnosti primjene podatkovnih tehnologija za učinkovitu analizu i praćenje izbornih podataka. Korišteni su alati poput Pythona, Pandasa, Matplotliba, Apache Sparka i VueJS-a.

Zaključno, uspostava integriranog skladišta podataka i implementacija sveobuhvatnog ETL procesa omogućili su efikasno upravljanje, analizu i prezentaciju izbornih podataka, čime su stvoreni preduvjeti za kvalitetnije donošenje odluka i daljnja istraživanja.