

Sveučilište Jurja Dobrile u Puli  
Fakultet informatike u Puli

Anastazija Širol

# Analiza prijevara u transakcijama

Seminarski rad

Pula, Svibanj, 2025

Sveučilište Jurja Dobrile u Puli  
Fakultet informatike u Puli

Anastazija Širol

# Analiza prijevara u transakcijama

Seminarski rad

JMBAG: 0303106956/redoviti student

Studijski smjer: Informatika

Kolegij: Skladišta i rudarenje podataka

Mentor: izv. prof. dr. sc. Goran Oreški

Pula, Svibanj, 2025



Sveučilište Jurja Dobrile u Puli

## IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisana Anastazija Širol, ovime izjavljujem da je ovaj seminarski rad rezultat isključivo mogega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio seminarskog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Studentica: Anastazija

Pula, 19.5.2025.

# Sadržaj

<b>1</b>	<b>Uvod</b>	<b>2</b>
<b>2</b>	<b>Eksplorativna analiza podataka</b>	<b>3</b>
2.1	Odabir dataset-a . . . . .	3
2.2	Analiza dataset-a . . . . .	3
2.3	Opis dataset-a . . . . .	6
<b>3</b>	<b>Implementacija relacijskog modela podataka</b>	<b>7</b>
3.1	ER dijagram . . . . .	7
3.2	Predprocesiranje podataka . . . . .	8
3.3	Shema relacijskog modela podataka . . . . .	8
3.4	Punjenje relacijskog modela podataka . . . . .	10
3.5	Provjera podataka . . . . .	14
<b>4</b>	<b>Implementacija dimenzijskog modela podataka</b>	<b>15</b>
<b>5</b>	<b>Kreiranje ETL procesa</b>	<b>17</b>
5.1	Ekstrakcija podataka . . . . .	17
5.2	Transformacija podataka . . . . .	17
5.3	Učitavanje podataka . . . . .	24
<b>6</b>	<b>Analiza podataka i vizualizacija</b>	<b>26</b>
6.1	Analiza iznosa transakcija . . . . .	26
6.2	Analiza rizika . . . . .	30
6.3	Analiza prijevvara . . . . .	34
<b>7</b>	<b>Zaključak</b>	<b>38</b>

# 1 Uvod

U suvremenom poslovnom okruženju, donošenje odluka temeljenih na podacima postalo je ključno za stjecanje konkurentске prednosti. Uzimajući u obzir velike količine podataka, organizacije se oslanjaju na sustave **poslovne inteligencije** (Business Intelligence, BI) [2] koja objedinjuje skup alata omogućujući prikupljanje i obradu podataka s ciljem dobivanja informacija za donošenje poslovnih odluka. [1]

U ovom radu simuliran je razvoj sustava poslovne inteligencije koristeći javno dostupan skup podataka s platforme Kaggle pod nazivom Fraud Detection Transactions Dataset, koji sadrži financijske transakcije, uključujući i one označene kao prijevare. Cilj rada bio je demonstrirati kako se kroz proces poslovne inteligencije može razviti sustav za analizu prijevara u transakcijama.

Rad obuhvaća nekoliko ključnih faza: eksplorativnu analizu podataka, implementaciju relacijskog i dimenzijskog modela podataka, kreiranje ETL procesa te analizu podataka i vizualizaciju.

Skup podataka preuzet je u obliku CSV datoteke koja je potom podijeljena u dva dijela: jedan dio je ostao u CSV formatu, dok je drugi dio transformiran u relacijski model. Za kreiranje shema relacijskog i dimenzijskog modela korišten je alat Lucidchart, koji je omogućio pregledno i jasno modeliranje strukture podataka. Za implementaciju i punjenje relacijskog modela u MySQL bazi podataka korišten je programski jezik Python, dok je za punjenje dimenzijskog modela u MySQL bazi podataka korišten Apache Spark, koji je omogućio obradu i integraciju podataka iz oba izvora. Na kraju, vizualizacija podataka izvedena je pomoću alata Tableau, čime je omogućeno intuitivno i interaktivno prikazivanje uvida dobivenih iz podataka.

Ovaj rad predstavlja praktični primjer razvoja sustava poslovne inteligencije, u ovom slučaju namijenjenom za analizu prijevara vezanih uz transakcije, a sam cilj rada je stvoriti skladište podataka i kreirati interaktivni dashboard koji će pružiti pregled ključnih metrika vezanih uz detekciju prijevara. Ovakav vizualni prikaz omogućuje korisnicima sustava poslovne inteligencije brže prepoznavanje potencijalno sumnjivih ponašanja i donošenje učinkovitih odluka u svrhu prevencije transakcijskih prijevara.

## 2 Eksplorativna analiza podataka

**Eksplorativna analiza podataka** (Exploratory Data Analysis, EDA) predstavlja temeljni korak u svakom procesu analize podataka te je ključna za razumijevanje strukture i karakteristika skupa podataka. EDA omogućuje identifikaciju obrazaca, anomalija, odnosa među varijablama i potencijalnih pogrešaka u podacima. Ovaj pristup pomaže pri odluci o daljnjoj obradi i modeliranju podataka. [3]

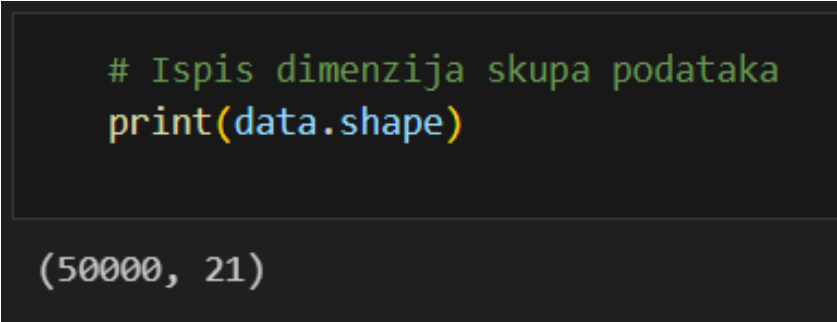
### 2.1 Odabir dataset-a

Proces odabira skupa podataka proveden je putem specijaliziranih internetskih platformi koje nude besplatne i javno dostupne podatke za analitičke i znanstvene svrhe. U ovom radu korišten je skup podataka preuzet u CSV formatu s platforme Kaggle, dostupan na sljedećoj poveznici: <https://www.kaggle.com/datasets/samayashar/fraud-detection-transactions-dataset>. Odabrani skup sadrži informacije o financijskim transakcijama te uključuje niz atributa koji omogućuju identifikaciju potencijalno prijevornih aktivnosti.

### 2.2 Analiza dataset-a

Nakon odabira skupa podataka, bilo je potrebno provesti temeljitu analizu kako bi se utvrdilo ispunjava li postavljene kriterije za daljnju obradu i modeliranje.

Prvi kriterij odnosio se na veličinu skupa podataka. Bilo je potrebno da sadrži najmanje 50 000 redaka i najmanje 20 stupaca. Kako je vidljivo na prikazanoj slici, odabrani skup podataka zadovoljava ovaj uvjet jer uključuje 50 000 redaka i 21 stupac.



```
# Ispis dimenzija skupa podataka
print(data.shape)

(50000, 21)
```

Slika 1: Dimenzije skupa podataka

Nadalje, analizirana je raznolikost podataka unutar atributa. Utvrđeno je da svi atributi sadrže dovoljan broj različitih vrijednosti, čime je ispunjen uvjet raznolikosti, što je ključno za kvalitetnu analizu i modeliranje.

```
# Ispis broja jedinstvenih vrijednosti po stupcima
print(data.nunique())
```

Transaction_ID	50000
User_ID	8963
Transaction_Amount	21763
Transaction_Type	4
Timestamp	47724
Account_Balance	49867
Device_Type	3
Location	5
Merchant_Category	5
IP_Address_Flag	2
Previous_Fraudulent_Activity	2
Daily_Transaction_Count	14
Avg_Transaction_Amount_7d	31420
Failed_Transaction_Count_7d	5
Card_Type	4
Card_Age	239
Transaction_Distance	47546
Authentication_Method	4
Risk_Score	9931
Is_Weekend	2
Fraud_Label	2
dtype: int64	

Slika 2: Jedinstvene vrijednosti

Skup podataka također uključuje vremensku dimenziju, odnosno atribut koji označava vrijeme izvršenja transakcije, što je bio jedan od nužnih preduvjeta.

	Transaction_ID	User_ID	Transaction_Amount	Transaction_Type	\
0	TXN_33553	USER_1834	39.79	POS	
1	TXN_9427	USER_7875	1.19	Bank Transfer	
2	TXN_199	USER_2734	28.96	Online	
3	TXN_12447	USER_2617	254.32	ATM Withdrawal	
4	TXN_39489	USER_2014	31.28	POS	

	Timestamp	Account_Balance	Device_Type	Location	\
0	2023-08-14 19:30:00	93213.17	Laptop	Sydney	
1	2023-06-07 04:01:00	75725.25	Mobile	New York	
2	2023-06-20 15:25:00	1588.96	Tablet	Mumbai	
3	2023-12-07 00:31:00	76807.20	Tablet	New York	
4	2023-11-11 23:44:00	92354.66	Mobile	Mumbai	

	Merchant_Category	IP_Address_Flag	Previous_Fraudulent_Activity	\
0	Travel	0	0	
1	Clothing	0	0	
2	Restaurants	0	0	
3	Clothing	0	0	
4	Electronics	0	1	

	Daily_Transaction_Count	Avg_Transaction_Amount_7d	\
0	7	437.63	
1	13	478.76	
2	14	50.01	
...			

	Authentication_Method	Risk_Score	Is_Weekend	Fraud_Label
1	Password	0.0959	0	1
2	Biometric	0.8400	0	1
3	OTP	0.7935	0	1
4	Password	0.3819	1	1

Slika 3: Vremenska dimenzija

Uz to, dataset obuhvaća kvantitativne (numeričke) i kvalitativne (kategorizirane) podatke, što je jasno vidljivo iz prikazanih primjera čime se omogućuje bolja obrada podataka.

```
# Ispis tipova podataka po stupcima (analiza kvantitativnih i kvalitativnih varijabli)
print(data.dtypes)
```

Transaction_ID	object
User_ID	object
Transaction_Amount	float64
Transaction_Type	object
Timestamp	object
Account_Balance	float64
Device_Type	object
Location	object
Merchant_Category	object
IP_Address_Flag	int64
Previous_Fraudulent_Activity	int64
Daily_Transaction_Count	int64
Avg_Transaction_Amount_7d	float64
Failed_Transaction_Count_7d	int64
Card_Type	object
Card_Age	int64
Transaction_Distance	float64
Authentication_Method	object
Risk_Score	float64
Is_Weekend	int64
Fraud_Label	int64
dtype:	object

Slika 4: Kvantitativni i kvalitativni podaci



Konačno, važno je bilo i provjeriti kvalitetu podataka u smislu potpunosti. U ovom slučaju, skup podataka ne sadrži nedostajuće vrijednosti (NULL), što dodatno potvrđuje njegovu prikladnost za uporabu u sustavu poslovne inteligencije.

```
# Ispis imena stupaca i nedostajućih vrijednosti
print(data.isna().sum())
```

Transaction_ID	0
User_ID	0
Transaction_Amount	0
Transaction_Type	0
Timestamp	0
Account_Balance	0
Device_Type	0
Location	0
Merchant_Category	0
IP_Address_Flag	0
Previous_Fraudulent_Activity	0
Daily_Transaction_Count	0
Avg_Transaction_Amount_7d	0
Failed_Transaction_Count_7d	0
Card_Type	0
Card_Age	0
Transaction_Distance	0
Authentication_Method	0
Risk_Score	0
Is_Weekend	0
Fraud_Label	0
dtype: int64	

Slika 5: Nedostajuće vrijednosti

Osim provjere osnovnih kriterija, dodatno su provjereni nazivi svih stupaca, kako bi se osiguralo bolje razumijevanje strukture i sadržaja skupa podataka prije provođenja daljnjih transformacija.

## 2.3 Opis dataset-a

Skup podataka obuhvaća 50 000 transakcija s pripadnom informacijom o detekciji prijevara. Svaku transakciju izvršava korisnik putem kartice, a poznata je i kategorija trgovca kome je transakcija namijenjena. Transakcija može biti obavljena s jedne od pet lokacija, putem jednog od tri različita uređaja i koristeći jednu od četiri metode autentifikacije.

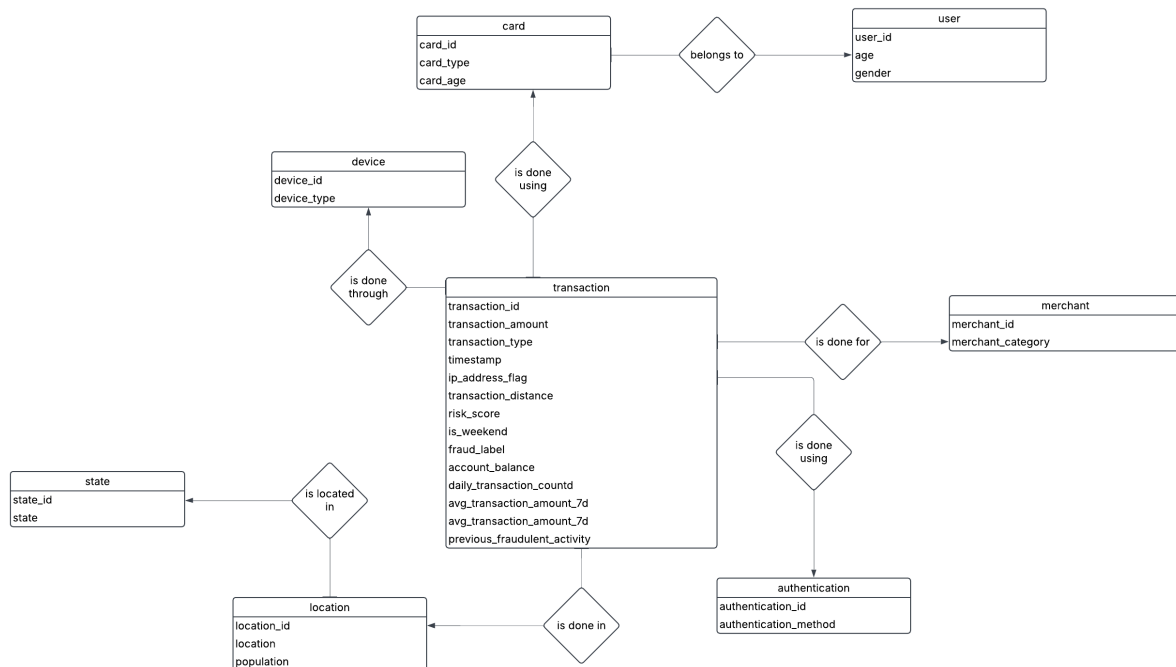
### 3 Implementacija relacijskog modela podataka

Relacijski model podataka u ovom radu oblikovan je kao **transakcijski sustav** (Operational System, OPS), koji je osmišljen za svakodnevnu obradu i pohranu transakcija. Za takve je sustave karakteristično da ne pohranjuju povijesne podatke, već sadržavaju trenutno stanje poslovnih procesa. [1]

#### 3.1 ER dijagram

U početnoj fazi implementacije kreiran je ER dijagram koji prikazuje strukturu relacijskog modela podataka. Središnja tablica sustava je **transaction**, koja sadrži sve attribute povezane s pojedinom transakcijom. Sve ostale tablice povezane su s njom izravno ili neizravno vezom jedan naprema više.

Tablica **device** opisuje uređaj korišten za izvršenje transakcije. Tablica **card** predstavlja karticu kojom je transakcija obavljena, a ona je povezana vezom više naprema jedan s tablicom **user**, koja definira korisnika te kartice. Nadalje, tablica **merchant** sadrži informacije o trgovcu kod kojeg je transakcija provedena, dok tablica **authentication** označava metodu autentifikacije korištenu prilikom izvršenja transakcije. Na samom kraju, tablica **location** definira grad u kojem je transakcija obavljena. Ona je povezana s tablicom **state**, koja sadrži podatke o državi kojoj taj grad pripada. Veza između ovih dviju tablica također je tipa više naprema jedan.



Slika 6: ER dijagram relacijskog modela

## 3.2 Predprocesiranje podataka

Sljedeći korak bio je postupak predprocesiranja podataka, koji je proveden korištenjem programskog jezika Python. Nakon učitavanja skupa podataka, provedene su osnovne pripreme: uklanjanje mogućih nedostajućih vrijednosti, standardizacija naziva stupaca korištenjem malih slova i zamjenom razmaka znakom donje crte radi konzistentnosti.

S obzirom na to da se u izvornom skupu podataka nalazilo samo pet gradova, svaki iz različite države, kreirani su dodatni popisi gradova za svaku od tih pet država. Potom je razvijena funkcija koja nasumično zamjenjuje nazive gradova u podacima, vodeći računa da zamjenski grad i dalje pripada istoj državi kao i izvorni, čime se osigurava veća raznolikost u podacima uz očuvanje logičke dosljednosti.

```
# Definiranje popisa gradova po državama
australian_cities = ['Sydney', 'Melbourne']
indian_cities = ['Mumbai', 'Delhi', 'Bangalore', 'Hyderabad', 'Pune']
japanese_cities = ['Tokio', 'Nagoya']
uk_cities = ['London', 'Birmingham']
us_cities = ['New York', 'Los Angeles', 'Chicago', 'Houston', 'San Francisco']

df['location'] = df['location'].apply(lambda city: 'Tokio' if city == 'Tokyo' else city)

# Definiranje nasumičnog mijenjenja gradova u istim državama
def randomize_location(df, city_column, city_list):
    num_to_change = int(len(df) * 0.6)
    rows_to_change = random.sample(range(len(df)), num_to_change)

    for row in rows_to_change:
        current_city = df.iloc[row][city_column]
        if current_city in city_list:
            new_city = random.choice([city for city in city_list if city != current_city])
            df.at[row, city_column] = new_city

    return df

df = randomize_location(df, 'location', australian_cities)
df = randomize_location(df, 'location', indian_cities)
df = randomize_location(df, 'location', japanese_cities)
df = randomize_location(df, 'location', uk_cities)
df = randomize_location(df, 'location', us_cities)
```

Slika 7: Ažuriranje gradova

Nakon toga, skup podataka podijeljen je u dva dijela u omjeru 80:20, pri čemu je svaki dio spremljen u zasebnu datoteku.

## 3.3 Shema relacijskog modela podataka

Za izradu sheme relacijskog modela podataka bilo je potrebno uspostaviti vezu s MySQL bazom podataka te unutar nje kreirati tablice prema prethodno definiranom ER dijagramu. Osim već poznatih atributa, pri definiranju tablica bilo je potrebno dodati i strane ključeve, kojima se uspostavlja veze između povezanih entiteta unutar baze.

```

# Definiranje sheme baze podataka
class State(Base):
    __tablename__ = 'state'
    id = Column(Integer, primary_key=True, autoincrement=True)
    name = Column(String(100), nullable=False, unique=True)

class Location(Base):
    __tablename__ = 'location'
    id = Column(Integer, primary_key=True, autoincrement=True)
    name = Column(String(100), nullable=True, unique=True)
    population = Column(Integer, nullable=False, default=0)
    state_fk = Column(Integer, ForeignKey('state.id'))

class User(Base):
    __tablename__ = 'user'
    id = Column(String(50), primary_key=True)
    age = Column(Integer, nullable=False)
    gender = Column(String(10), nullable=False)

class Card(Base):
    __tablename__ = 'card'
    id = Column(String(50), primary_key=True)
    card_type = Column(String(50), nullable=False)
    card_age = Column(Integer, nullable=False)
    user_fk = Column(String(50), ForeignKey('user.id'))

class Device(Base):
    __tablename__ = 'device'
    id = Column(Integer, primary_key=True, autoincrement=True)
    device_type = Column(String(50), nullable=False, unique=True)

class Authentication(Base):
    __tablename__ = 'authentication'
    id = Column(Integer, primary_key=True, autoincrement=True)
    authentication_method = Column(String(100), nullable=False, unique=True)

class Merchant(Base):
    __tablename__ = 'merchant'
    id = Column(Integer, primary_key=True, autoincrement=True)
    merchant_category = Column(String(100), nullable=False, unique=True)

class Transaction(Base):
    __tablename__ = 'transaction'
    id = Column(String(50), primary_key=True)
    transaction_amount = Column(DECIMAL(20,2), nullable=False)
    transaction_type = Column(String(50), nullable=False)
    timestamp = Column(DateTime, nullable=False)
    ip_address_flag = Column(Boolean, nullable=False)
    transaction_distance = Column(DECIMAL(20,2), nullable=False)
    risk_score = Column(Float, nullable=False)
    is_weekend = Column(Boolean, nullable=False)
    fraud_label = Column(Boolean, nullable=False)
    account_balance = Column(DECIMAL(20,2), nullable=False)
    daily_transaction_count = Column(Integer, nullable=False)
    avg_transaction_amount_7d = Column(Float, nullable=False)
    failed_transaction_count_7d = Column(Integer, nullable=False)
    previous_fraudulent_activity = Column(Boolean, nullable=False)

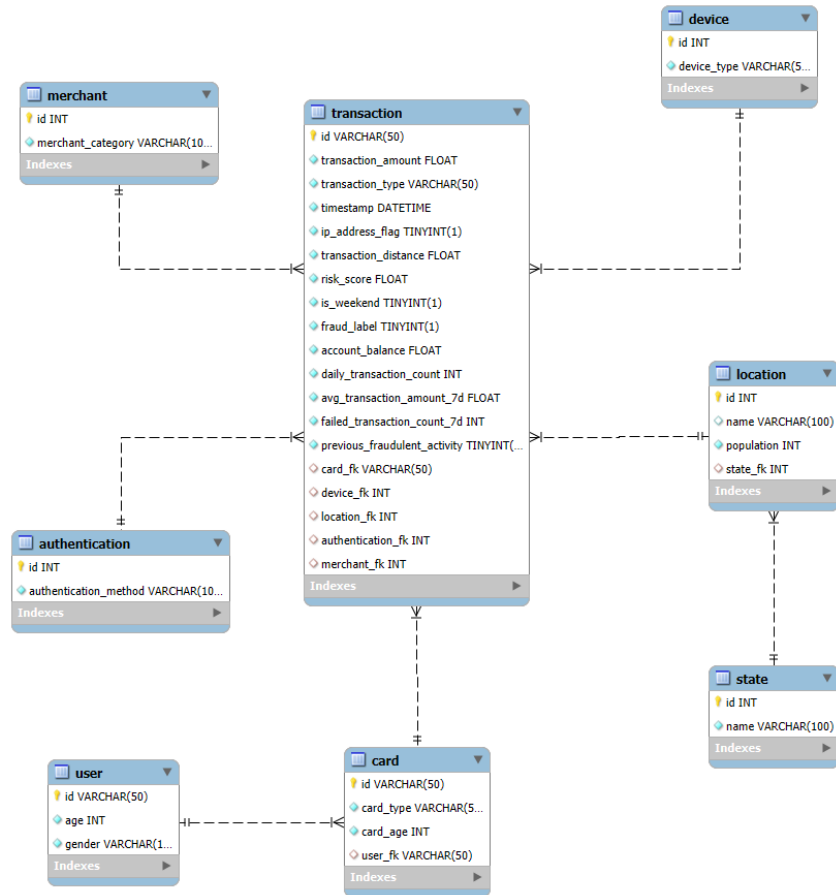
    card_fk = Column(String(50), ForeignKey('card.id'))
    device_fk = Column(Integer, ForeignKey('device.id'))
    location_fk = Column(Integer, ForeignKey('location.id'))
    authentication_fk = Column(Integer, ForeignKey('authentication.id'))
    merchant_fk = Column(Integer, ForeignKey('merchant.id'))

```

Slika 8: Shema relacijskog modela podataka

Nakon pokretanja koda, u bazi podataka su se uspješno kreirale tablice koje su u početku bile prazne. Na temelju tih tablica generiran je EER dijagram koristeći MySQL

Reverse Engineering funkcionalnost. Kao što je prikazano na slici, dijagram je vrlo sličan izvornom ER dijagramu, ali sadrži i dodatne attribute, strane ključeve, koji su u MySQL-u ključni za uspostavljanje odnosa među tablicama.



Slika 9: EER dijagram

### 3.4 Punjenje relacijskog modela podataka

Popunjavanje relacijskog modela podataka provedeno je korištenjem 80 posto podataka koji su tijekom faze predprocesiranja spremljeni u zasebnu CSV datoteku. Svaka tablica punila se zasebno, izravno iz te datoteke. Većina potrebnih atributa već je bila sadržana u CSV-u, no za pojedine tablice bilo je potrebno primijeniti dodatne metode kako bi se osigurao ispravan unos i očuvala dosljednost podataka.

Prva takva tablica bila je tablica **user** za koju je bilo potrebno definirati dvije dodatne funkcije koje su nasumično dodjeljivale godine starosti i spol svakom korisniku. Ti podaci nisu bili dostupni u izvornom CSV-u, a njihovo dodavanje imalo je za cilj obogatiti skup podataka. Za razliku od većine ostalih tablica, u kojima se identifikator automatski generira, u ovom slučaju se id preuzima iz CSV datoteke jer je riječ o jedinstvenom identifikatoru u obliku stringa.

```

# Umetanje: user
import random

def generate_age():
    return random.randint(18, 80)

def generate_gender():
    return random.choice(['Male', 'Female'])

df_for_db = df.copy()
user_set = set(df_for_db['user_id'])

user_list = []

for user_id in user_set:
    user_dict = {
        'id': user_id,
        'age': generate_age(),
        'gender': generate_gender()
    }
    user_list.append(user_dict)

session.bulk_insert_mappings(User, user_list)
session.commit()

user_map = {user.id: user.id for user in session.query(User).all()}

```

Slika 10: Popunjavanje tablice user

Sljedeća tablica koja je zahtijevala dodatne prilagodbe bila je tablica card. Umjesto automatski generiranog identifikatora, bilo je potrebno ručno generirati id tako da se kombinacijom atributa user id, card type i card age, povezanim donjom crtom, dobije jedinstvena vrijednost. Ovakav pristup omogućio je jasno i precizno povezivanje središnje tablice transaction s točno određenom karticom, s obzirom na to da je upravo kombinacija tih triju atributa jedinstvena za svaku karticu u skupu podataka.

```

# Umetanje: card
cards = df[['card_type', 'card_age', 'user_id']].drop_duplicates()

cards['user_fk'] = cards['user_id'].map(user_map)
cards['id'] = cards['user_id'].astype(str) + "_" + cards['card_type'] + "_" + cards['card_age'].astype(str)

cards = cards.drop(columns=['user_id'])

session.bulk_insert_mappings(Card, cards.to_dict(orient="records"))
session.commit()

card_map = {c.id: c.id for c in session.query(Card).all()}

```

Slika 11: Popunjavanje tablice card

Budući da informacije o državama nisu bile sadržane u izvornom CSV-u, za potrebe popunjavanja tablice state države su definirane ručno, na temelju poznatih gradova prisutnih u skupu podataka.

```

# Umetanje: state
countries_data = [
    {"name": "Australia"},
    {"name": "India"},
    {"name": "Japan"},
    {"name": "United Kingdom"},
    {"name": "United States"}
]

countries_list = []
for i, country in enumerate(countries_data):
    country_entry = {
        "name": country["name"]
    }
    countries_list.append(country_entry)

session.bulk_insert_mappings(State, countries_list)
session.commit()

state_map = {s.name: s.id for s in session.query(State).all()}

```

Slika 12: Popunjavanje tablice `state`

Dodatne nadogradnje morale su se napraviti i prilikom punjenja tablice `location`, s obzirom na to da informacije o broju stanovnika nisu postojale u izvornom CSV-u. Podaci o populaciji dohvaćeni su automatski s Wikipedije pomoću web scrapinga, pri čemu se koristila biblioteka BeautifulSoup za izvlačenje informacija iz tablice najvećih svjetskih metropola. Paralelno s time, za svaki je grad ručno određeno kojoj državi pripada, s ciljem da se kasnije može uspostaviti veza s tablicom `state`. Gradovi bez pridružene države ili broja stanovnika dodatno su obrađeni — u slučaju nedostajuće populacije upisana je vrijednost 0, dok su gradovi bez pripadajuće države isključeni iz daljnje obrade.

```

# Umetanje: location
def fetch_city_population():
    url = 'https://sh.wikipedia.org/wiki/Lista_najvećih_svjetskih_metropola'
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')

    city_population = {}
    table = soup.find('table', {'class': 'wikitable'})
    rows = table.find_all('tr')[1:]

    for row in rows:
        cols = row.find_all('td')
        if len(cols) >= 3:
            city = cols[1].text.strip()
            raw_population = cols[2].text.strip()
            clean_population = re.sub(r'[\d\,]', '', raw_population)
            clean_population = int(''.join(clean_population.split()))
            city_population[city] = clean_population

    return city_population

city_population = fetch_city_population()

australian_cities = ['Sydney', 'Melbourne']
indian_cities = ['Mumbai', 'Delhi', 'Bangalore', 'Hyderabad', 'Pune']
japanese_cities = ['Tokio', 'Nagoya']
uk_cities = ['London', 'Birmingham']
us_cities = ['New York', 'Los Angeles', 'Chicago', 'Houston', 'San Francisco']

def map_cities_to_states(city_list, state_name):
    return {city: state_name for city in city_list}

city_to_state = {}
city_to_state.update(map_cities_to_states(australian_cities, 'Australia'))
city_to_state.update(map_cities_to_states(indian_cities, 'India'))
city_to_state.update(map_cities_to_states(japanese_cities, 'Japan'))
city_to_state.update(map_cities_to_states(uk_cities, 'United Kingdom'))
city_to_state.update(map_cities_to_states(us_cities, 'United States'))

df['population'] = df['location'].map(city_population)
df['state_name'] = df['location'].map(city_to_state)

df['population'] = df['population'].fillna(0)

df = df.dropna(subset=['state_name'])

missing_states = df[~df['state_name'].isin(state_map)].drop_duplicates(subset=['state_name'])

if not missing_states.empty:
    print("Upozorenje: Sljedeći gradovi nemaju pridružene države:")
    print(missing_states[['location', 'state_name']])

df['state_fk'] = df['state_name'].map(state_map)

locations = df[['location', 'state_fk', 'population']].drop_duplicates().rename(columns={'location': 'name'})

session.bulk_insert_mappings(Location, locations.to_dict(orient="records"))
session.commit()

location_map = {l.name: l.id for l in session.query(Location).all()}

```

Slika 13: Popunjavanje tablice location

Na samom kraju, prilikom popunjavanja tablice `transaction`, svi potrebni atributi već su postojali u izvornom CSV-u. Strani ključevi, potrebni za povezivanje s ostalim tablicama, umetnuti su korištenjem prethodno definiranih mapiranja koja su omogućila dohvaćanje odgovarajućih id-eva za svaki povezani zapis.

Nakon izvršavanja skripti za popunjavanje, tablice u MySQL bazi podataka uspješno su ispunjene relevantnim podacima iz prethodno predprocesiranog CSV-a. Na sljedećoj slici prikazan je primjer jedne od popunjenih tablica.

	id	transaction_amount	transaction_type	timestamp	ip_address_flag	transaction_distance	risk_score	is_weekend	fraud_label	account_balance	daily_trans
▶	TXN_0	95.97	ATM Withdrawal	2023-01-20 06:55:00	0	4610.43	0.3773	0	0	93915.02	14
	TXN_1	55.41	ATM Withdrawal	2023-11-27 16:08:00	0	2284.15	0.5871	0	0	91495.28	7
	TXN_10	283.26	POS	2023-01-10 05:17:00	0	4273.45	0.6237	1	0	48484.20	1
	TXN_100	57.24	ATM Withdrawal	2023-11-20 10:32:00	0	3446.99	0.0785	0	0	89219.38	14
	TXN_1000	58.43	ATM Withdrawal	2023-08-21 22:32:00	0	2968.75	0.1635	0	0	40090.04	3
	TXN_10000	16.37	Bank Transfer	2023-09-03 18:32:00	0	1198.43	0.8101	0	0	30837.25	6
	TXN_10001	27.52	Online	2023-04-04 08:40:00	0	1335.29	0.776	1	1	18273.83	2
	TXN_10002	19.87	ATM Withdrawal	2023-06-26 09:17:00	0	1607.64	0.9484	0	1	4380.72	1
	TXN_10003	99.68	POS	2023-01-06 00:18:00	0	448.65	0.0801	1	0	87541.23	4
	TXN_10004	49.88	Bank Transfer	2023-10-10 06:47:00	0	2156.48	0.0827	0	1	40649.87	5

Slika 14: Tablica transaction s podacima



### 3.5 Provjera podataka

Posljednji korak ove faze bio je provjeriti je li uvoz podataka u bazu podataka uspješno izvršen. Ta validacija provedena je usporedbom podataka iz predprocesirane CSV datoteke i onih pohranjenih u relacijskoj bazi podataka.

Usporedba se odvijala u dva koraka. Prvo su se provjeravali nazivi stupaca kako bi se osiguralo da su svi relevantni atributi prisutni u oba izvora podataka. Nakon toga, sadržaj podataka iz CSV-a i baze uspoređivao se po vrijednostima. Kako bi usporedba bila precizna, poduzete su određene prilagodbe: numerički atributi pretvoreni su u decimalni oblik i zaokruženi na dvije decimale, a atribut vremenskog zapisa konvertiran je u standardni datetime format. Također, redovi su sortirani prema id-u transakcije kako bi se osiguralo da je redoslijed zapisa jednak.

```
def test_columns(self):
    expected_columns = [
        'transaction_id', 'user_id', 'transaction_amount', 'transaction_type', 'timestamp', 'account_balance',
        'device_type', 'ip_address_flag', 'location', 'merchant_category', 'card_type', 'card_age',
        'authentication_method', 'transaction_distance', 'risk_score', 'is_weekend', 'fraud_label',
        'daily_transaction_count', 'avg_transaction_amount_7d', 'failed_transaction_count_7d',
        'previous_fraudulent_activity'
    ]

    df_filtered_columns = [col for col in self.df.columns if col in expected_columns]
    db_filtered_columns = [col for col in self.db_df.columns if col in expected_columns]

    self.assertEqual(df_filtered_columns, db_filtered_columns)

def test_dataframes(self):
    expected_columns = [
        'transaction_id', 'user_id', 'transaction_amount', 'transaction_type', 'timestamp', 'account_balance',
        'device_type', 'ip_address_flag', 'location', 'merchant_category', 'card_type', 'card_age',
        'authentication_method', 'transaction_distance', 'risk_score', 'is_weekend', 'fraud_label',
        'daily_transaction_count', 'avg_transaction_amount_7d', 'failed_transaction_count_7d',
        'previous_fraudulent_activity'
    ]

    common_columns = list(set(expected_columns) & set(self.df.columns) & set(self.db_df.columns))

    df_filtered = self.df[common_columns].sort_values(by="transaction_id").reset_index(drop=True)
    db_df_filtered = self.db_df[common_columns].sort_values(by="transaction_id").reset_index(drop=True)

    numeric_columns = ['transaction_amount', 'account_balance', 'transaction_distance']

    for col in numeric_columns:
        if col in df_filtered.columns:
            df_filtered[col] = df_filtered[col].astype(float).round(2)
            db_df_filtered[col] = db_df_filtered[col].astype(float).round(2)

    if "timestamp" in df_filtered.columns:
        df_filtered["timestamp"] = pd.to_datetime(df_filtered["timestamp"], errors='coerce')
        db_df_filtered["timestamp"] = pd.to_datetime(db_df_filtered["timestamp"], errors='coerce')

    try:
        assert_frame_equal(df_filtered, db_df_filtered, check_dtype=False, rtol=1e-3)
    except AssertionError:
        print("Razlike između CSV-a i baze podataka:")
        print(df_filtered.compare(db_df_filtered))
        raise
```

Slika 15: Usporedba podataka

Provedbom ove provjere došlo se do zaključka da je uvoz podataka uspješno izvršen te da se podaci iz predprocesirane CSV datoteke u potpunosti podudaraju s onima pohranjenima u relacijskoj bazi podataka. Time je potvrđena ispravnost postupka punjenja baze i osigurana kvaliteta podataka za daljnju analizu.

## 4 Implementacija dimenzijskog modela podataka

**Dimenzijski model** podataka je pristup dizajnu skladišta podataka koji koristi fakt i dimenzijske tablice. Fakt tablice sadrže kvantitativne podatke o poslovnim događajima, dok dimenzijske tablice pružaju kontekstualne informacije za te događaje.

Temelji se na jednostavnom konceptu koji omogućuje korisnicima brzo i intuitivno pretraživanje podataka, što ga čini vrlo pogodnim za poslovnu inteligenciju i analizu podataka. Može se implementirati kroz dvije osnovne sheme: star shemu i snowflake shemu.

**Star shemu** karakterizira jedna centralna fakt tablica koja je direktno povezana sa svim dimenzijskim tablicama, koje su u pravilu denormalizirane kako bi sadržavale sve potrebne attribute na jednom mjestu. Denormalizacija smanjuje kompleksnost i broj spajanja u SQL upitima, što rezultira jednostavnijom interpretacijom modela.

**Snowflake shema** je tip dimenzijskog modela u kojoj su dimenzijske tablice dodatno normalizirane i podijeljene u manje tablice, čime se uklanja redundancija podataka, ali se povećava složenost upita i smanjuje performanse.

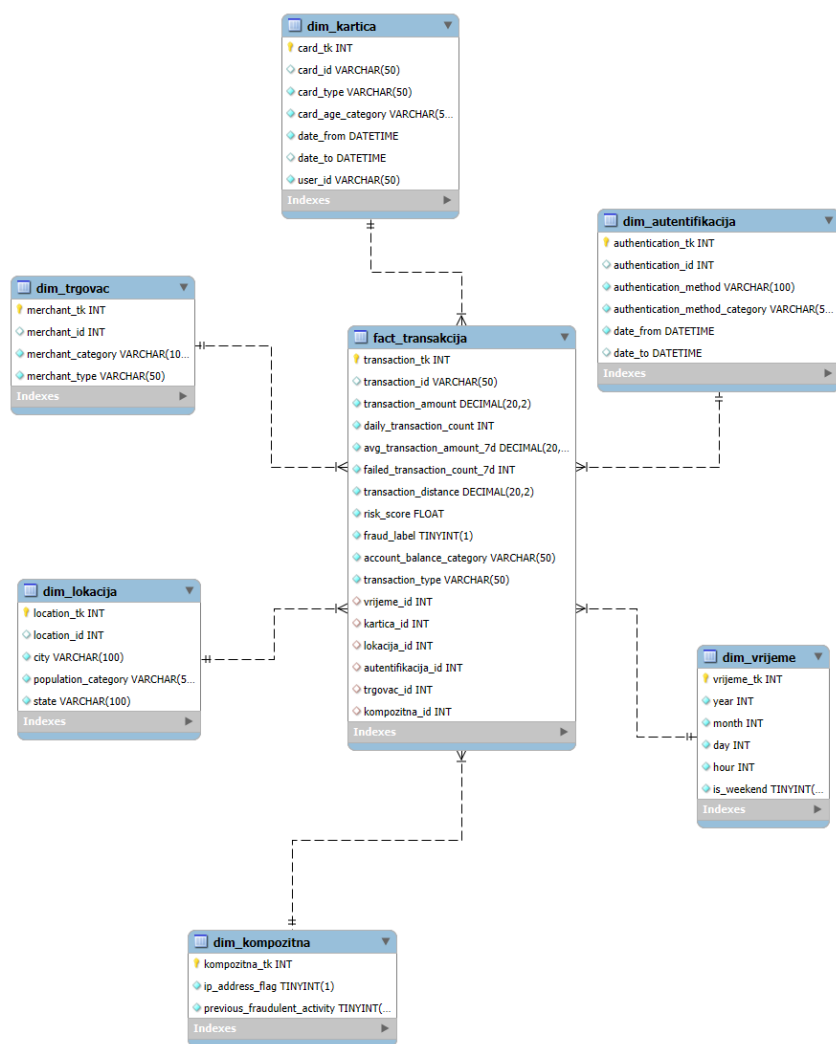
Star shema je često preferirani izbor u poslovnoj inteligenciji zbog svoje jednostavnosti i učinkovitosti u analitičkim upitima. U praksi, dodatna redundancija u dimenzijskim tablicama nije problem jer je cilj brza analiza podataka. Zbog toga se star shema koristi u većini slučajeva kada se gradi skladište podataka, uključujući i ovaj rad. [2]

Dimenzijski model podataka razvijen u ovom radu temelji se na jednoj fact tablici, šest dimenzijskih tablica te dvije degenerirane dimenzije.

Svaka dimenzijska tablica uključuje dodatni, automatski generirani tehnički ključ, koji omogućuje jednoznačno identificiranje svakog retka i osigurava stabilnu vezu s fact tablicom, neovisno o promjenama u prirodnim ključevima. Većina atributa u dimenzijskim tablicama preuzeta je iz relacijskog modela, no u pojedinim slučajevima kvantitativne vrijednosti zamijenjene su kvalitativnim kategorijama, a dodani su i novi atributi kako bi se omogućila bogatija analiza i detaljniji uvid u podatke.

Vremenska dimenzija izdvojena je kao zasebna tablica, pri čemu je originalni timestamp rastavljen na sastavne dijelove: `year`, `month`, `day`, `hour`. Time se omogućuje fleksibilnije grupiranje i analiziranje podataka kroz vrijeme.

Nadalje, u nekim dimenzijskim tablicama identificirane su sporo mijenjajuće dimenzije. Ove promjene praćene su uvođenjem atributa `date from` i `date to`, čime se omogućuje vremensko praćenje važenja određenih vrijednosti i analize koje uzimaju u obzir promjene tijekom vremena.



Slika 16: EER dijagram - dimenzijski

## 5 Kreiranje ETL procesa

**ETL** (Extract, Transform, Load) proces predstavlja ključan korak u izgradnji skladišta podataka jer omogućuje prikupljanje, obradu i pohranu podataka iz različitih izvora u jedinstveni centralizirani sustav. Proces se sastoji od tri glavne faze:

**Ekstrakcija** (Extract) – dohvaćanje podataka iz različitih izvora

**Transformacija** (Transform) – čišćenje podataka, usklađivanje formata, standardizacija i priprema za učitavanje

**Učitavanje** (Load) – unos obrađenih podataka u skladište podataka radi daljnje analize i izvještavanja

ETL proces omogućuje da se svi podaci koji dolaze iz različitih izvora spoje u jedno zajedničko skladište podataka.

**Skladišta podataka** (Data Warehouses, DW) služe kao centralizirani sustavi koji integriraju podatke iz različitih izvora, omogućujući analizu i izvještavanje. [2]

U ovom radu za implementaciju ETL procesa korišten je Apache Spark, alat za obradu velikih količina podataka. Apache Spark omogućuje brzo izvođenje složenih transformacija nad velikim skupovima podataka, a korišten je unutar Docker okruženja, čime je osigurana jednostavna konfiguracija.

U glavnoj `main` datoteci redom su se pozivale sve potrebne funkcije za izvođenje ETL procesa, pri čemu su te funkcije bile organizirane i definirane u zasebnim datotekama. Ovakva modularna struktura omogućila je bolju preglednost, ponovnu iskoristivost koda i jednostavnije održavanje svakog koraka unutar ETL procesa – ekstrakcije, transformacije i učitavanja podataka.

### 5.1 Ekstrakcija podataka

Za početak je bilo potrebno provesti ekstrakciju podataka. Kao što je prethodno navedeno, podaci su dolazili iz dva izvora: CSV datoteke koja je sadržavala 20 posto podataka te relacijske baze podataka u kojoj se nalazilo preostalih 80 posto. Procesom ekstrakcije dohvaćeni su svi podaci iz oba izvora, čime je omogućena njihova daljnja obrada u ETL procesu.

### 5.2 Transformacija podataka

Nakon obavljene ekstrakcije, pristupilo se fazi transformacije podataka, koja je bila najzahtjevniji i vremenski najintenzivniji dio procesa. Razlog tome leži u potrebi za čišćenjem podataka, standardizacijom formata, identifikacijom i uklanjanjem nekonzistentnosti, kao i u integraciji podataka iz dvaju različitih izvora. Također, tijekom transformacije bilo je potrebno osigurati da svi podaci budu strukturirani i prilagođeni formatu dimenzijskog modela kako bi se mogli ispravno učitati u skladište podataka.

Transformacija podataka provedena je tako da su se podaci najprije pripremili i prilagodili za svaku dimenzijsku tablicu pojedinačno, a potom je izvršena transformacija podataka za fact tablicu. Ovakav pristup omogućio je pravilno povezivanje dimenzijskih tablica s fact tablicom, čime je osigurana konzistentnost i integritet podataka unutar dimenzijskog modela.

Za dimenziju autentifikacija, transformacija podataka provedena je tako što su iz baze podataka i CSV datoteke odabrani relevantni stupci. Budući da se identifikator autentifikacije nalazi isključivo u bazi, za podatke iz CSV-a u taj je stupac umetnuta vrijednost null. Dodatno je u dimenzijsku tablicu uveden novi atribut koji kategorizira metode autentifikacije u dvije skupine: Secure i Insecure. Ova kategorizacija izrađena je pomoću funkcije jer navedeni atribut nije postojao ni u jednom od izvora podataka. Također su definirani vremenski atributi `date from` i `date to` radi praćenja sporo mijenjajućih dimenzija, konkretno promjena u kategorizaciji metoda autentifikacije. Vrijednost atributa `date from` postavlja se u trenutku pokretanja skripte, dok se `date to` ostavlja praznim sve dok se ne pojavi nova verzija zapisa. Ovakav pristup omogućuje precizno praćenje valjanosti i promjena podataka kroz vrijeme.

```
def transform_authentifikacija_dim(authentication_df, csv_authentication_df=None):
    # Baza podataka
    db_auth_df = (
        authentication_df
        .select(
            col("id").cast(LongType()).alias("authentication_id"),
            initcap(trim(col("authentication_method"))).alias("authentication_method")
        )
    )

    # CSV podaci
    if csv_authentication_df:
        csv_df = (
            csv_authentication_df
            .selectExpr("authentication_method")
            .withColumn("authentication_method", initcap(trim(col("authentication_method"))))
            .withColumn("authentication_id", lit(None).cast(LongType()))
        )

        db_auth_df = db_auth_df.unionByName(csv_df)

    # Brisanje duplikata
    db_auth_df = db_auth_df.dropDuplicates(["authentication_method"])

    # Definiranje kategorija metoda autentifikacija
    db_auth_df = db_auth_df.withColumn(
        "authentication_method_category",
        when(col("authentication_method").isin("Biometric", "Otp"), "Secure")
        .when(col("authentication_method").isin("Password", "Pin"), "Insecure")
        .otherwise("Unknown")
    )
```

Slika 17: Transformacija podataka za dimenziju autentifikacija

Sljedeća dimenzijska tablica odnosila se na karticu. Transformacija podataka za ovu dimenziju provedena je tako da su se iz baze podataka i CSV datoteke dohvatili svi relevantni atributi, uključujući i `card age`. Iako se taj atribut ne pohranjuje izravno u dimenzijskoj tablici, bio je ključan jer se na temelju njega dodao novi atribut - kategorija starosti kartice. Ovaj atribut omogućio je klasifikaciju kartica u tri kvalitativne skupine:

New, Stable i Long Term, čime je izvorni numerički podatak pretvoren u kvalitativan podatak pogodan za analizu. Nakon definiranja kategorija, atribut `card age` je uklonjen iz tablice. Kao i u prethodnoj dimenziji, dodani su i vremenski atributi `date from` i `date to` radi praćenja sporo mijenjajućih dimenzija, konkretno promjena u kategoriji starosti kartice. Vrijednost atributa `date from` postavlja se na trenutni datum i vrijeme, dok `date to` ostaje prazan sve dok se ne pojavi nova verzija zapisa. Ovakav pristup osigurava povijesnu točnost i praćenje promjena kroz vrijeme.

```
def transform_kartica_dim(card_df, user_df, csv_card_df=None):
    c = card_df.alias("c")
    u = user_df.alias("u")

    # Baza podataka
    db_cards_df = (
        c.join(u, col("c.user_fk") == col("u.id"), "left")
        .select(
            col("c.id").cast(StringType()).alias("card_id"),
            col("c.card_type").alias("card_type"),
            col("c.card_age").alias("card_age"),
            col("u.id").cast(StringType()).alias("user_id")
        )
    )

    # CSV podaci
    if csv_card_df:
        csv_cards_df = (
            csv_card_df
            .select(
                col("card_type"),
                col("card_age").cast(int),
                col("user_id").cast(StringType())
            )
            .withColumn("card_id", lit(None).cast(StringType()))
        )

        db_cards_df = db_cards_df.select("card_id", "card_type", "card_age", "user_id") \
            .unionByName(csv_cards_df)

    # Definiranje kategorija starosti kartica
    db_cards_df = db_cards_df.withColumn(
        "card_age_category",
        when(col("card_age") < 60, "New")
        .when((col("card_age") >= 60) & (col("card_age") < 120), "Stable")
        .otherwise("Long term")
    )

    # Brisanje duplikata
    db_cards_df = db_cards_df.dropDuplicates(["card_type", "user_id", "card_age"])

    # Uklanjanje card_age
    db_cards_df = db_cards_df.drop("card_age")

    window = Window.orderBy("card_type", "user_id", "card_id")
    db_cards_df = (
        db_cards_df
        .withColumn("card_tk", row_number().over(window))
        .withColumn("date_from", current_timestamp())
        .withColumn("date_to", lit(None).cast("timestamp"))
    )
```

Slika 18: Transformacija podataka za dimenziju kartica

Kompozitna dimenzijska tablica u ovom slučaju sadržavala je dva boolean atributa koji nisu međusobno izravno povezani, ali su objedinjeni s ciljem analitičke učinkovitosti i preglednosti. Kreiranjem ove dimenzije omogućeno je stvaranje kartezijevog produkta vrijednosti tih dvaju atributa, što rezultira s ukupno četiri moguće kombinacije – po jedna za svaku kombinaciju vrijednosti oba atributa. Transformacija ove tablice bila je relativno

jednostavna jer su oba atributa već bili dostupni i u CSV datoteci i u relacijskoj bazi podataka. Također, nije bilo potrebe za dodavanjem dodatnih atributa niti za naprednijom obradom podataka.

```
def transform_kompozitna_dim(composite_df, csv_composite_df=None):  
  
    # Baza podataka  
    db_df = (  
        composite_df  
        .select(  
            col("ip_address_flag").cast("boolean"),  
            col("previous_fraudulent_activity").cast("boolean")  
        )  
    )  
  
    # CSV podaci  
    if csv_composite_df:  
        csv_df = (  
            csv_composite_df  
            .select(  
                col("ip_address_flag").cast("boolean"),  
                col("previous_fraudulent_activity").cast("boolean")  
            )  
        )  
  
        combined_df = db_df.unionByName(csv_df)  
  
    # Brisanje duplikata  
    combined_df = combined_df.dropDuplicates(["ip_address_flag", "previous_fraudulent_activity"])  
  
    # Definiranje finalne tablice  
    window = Window.orderBy("ip_address_flag", "previous_fraudulent_activity")  
    final_df = (  
        combined_df  
        .withColumn("kompozitna_tk", row_number().over(window))  
        .select(  
            "kompozitna_tk",  
            "ip_address_flag",  
            "previous_fraudulent_activity"  
        )  
    )
```

Slika 19: Transformacija podataka za kompozitnu dimenziju

Za dimenziju lokacija prikupljeni su svi relevantni atributi iz CSV datoteke i relacijske baze podataka. Budući da u CSV datoteci nisu postojali podaci o populaciji i državi, ti su atributi inicijalno postavljeni na null. Međutim, to nije predstavljalo problem jer su se u tablicu pohranjivali isključivo jedinstveni gradovi, a svi gradovi prisutni u CSV-u već su postojali u bazi. Time je omogućeno da se nakon spajanja podataka svi gradovi povežu s kompletnim informacijama, čime se u konačnoj dimenzijskoj tablici izbjegla pojava null vrijednosti. Atribut populacija poslužio je za generiranje dodatnog kvalitativnog atributa – kategorije veličine grada. Na temelju vrijednosti populacije gradovi su razvrstani u četiri skupine: *mega city*, *large city*, *medium city*, *small city*. Nakon što je ta transformacija provedena, izvorni atribut uklonjen je iz tablice kako bi se zadržala samo kvalitativna interpretacija.

```

def transform_kompozitna_dim(composite_df, csv_composite_df=None):

    # Baza podataka
    db_df = (
        composite_df
        .select(
            col("ip_address_flag").cast("boolean"),
            col("previous_fraudulent_activity").cast("boolean")
        )
    )

    # CSV podaci
    if csv_composite_df:
        csv_df = (
            csv_composite_df
            .select(
                col("ip_address_flag").cast("boolean"),
                col("previous_fraudulent_activity").cast("boolean")
            )
        )

        combined_df = db_df.unionByName(csv_df)

    # Brisanje duplikata
    combined_df = combined_df.dropDuplicates(["ip_address_flag", "previous_fraudulent_activity"])

    # Definiranje finalne tablice
    window = Window.orderBy("ip_address_flag", "previous_fraudulent_activity")
    final_df = (
        combined_df
        .withColumn("kompozitna_tk", row_number().over(window))
        .select(
            "kompozitna_tk",
            "ip_address_flag",
            "previous_fraudulent_activity"
        )
    )

```

Slika 20: Transformacija podataka za dimenziju lokacija

U dimenzijskoj tablici `trgovac`, uz postojeće attribute preuzete iz CSV datoteke i relacijske baze podataka, dodan je novi atribut `merchant type` koji je nastao primjenom funkcije nad postojećim atributom `merchant category`. Funkcija je svrstavala postojeće kategorije u šire skupine: `everyday`, `luxury`, `consumable`. Cilj ove transformacije bio je uvođenje hijerarhije unutar podataka, čime se omogućava detaljnija i fleksibilnija analiza u kasnijim fazama izvještavanja.



```
def transform_trgovac_dim(merchant_df, csv_merchant_df=None):
    # Baza podataka
    db_merchant_df = (
        merchant_df
        .select(
            col("id").cast(LongType()).alias("merchant_id"),
            initcap(trim(col("merchant_category"))).alias("merchant_category")
        )
    )

    # CSV podaci
    if csv_merchant_df:
        csv_df = (
            csv_merchant_df
            .selectExpr("merchant_category")
            .withColumn("merchant_category", initcap(trim(col("merchant_category"))))
            .withColumn("merchant_id", lit(None).cast(LongType()))
        )
        db_merchant_df = db_merchant_df.unionByName(csv_df)

    # Brisanje duplikata
    db_merchant_df = db_merchant_df.dropDuplicates(["merchant_category"])

    # Definiranje tipa trgovca
    db_merchant_df = db_merchant_df.withColumn(
        "merchant_type",
        when(col("merchant_category").isin("Groceries"), "Everyday")
        .when(col("merchant_category").isin("Travel", "Restaurants"), "Luxury")
        .when(col("merchant_category").isin("Electronics", "Clothing"), "Consumable")
        .otherwise("Unknown")
    )
```

Slika 21: Transformacija podataka za dimenziju trgovac

Posljednja dimenzijska tablica za koju je provedena transformacija podataka bila je dimenzija vrijeme. Iz baze i CSV datoteke preuzeta su dva ključna atributa: informacija o tome je li transakcija izvršena tijekom vikenda te timestamp transakcije. Budući da je cilj bio strukturirati podatke prema year, month, day, hour, bilo je potrebno izdvojiti te komponente iz timestamp vrijednosti. Međutim, zbog problema s vremenskim zonama, pristup parsiranja timestampa izravno, nije davao dobre rezultate pa se vrijednost najprije morala pretvoriti u string, a zatim su se iz nje izdvojili pojedinačni elementi.

```
def transform_vrijeme_dim(date_df, csv_date_df=None):
    def prepare_df(df):
        return (
            df
            .withColumn("timestamp_str", col("timestamp").cast("string")) # Osiguravanje da je datum string zbog problema s vremenskim zonama
            .select(
                col("timestamp_str").alias("timestamp"),
                col("is_weekend").cast("boolean")
            )
            .withColumn("year", substring("timestamp", 1, 4).cast("int"))
            .withColumn("month", substring("timestamp", 6, 2).cast("int"))
            .withColumn("day", substring("timestamp", 9, 2).cast("int"))
            .withColumn("hour", substring("timestamp", 12, 2).cast("int"))
        )

    # Baza podataka
    db_df = prepare_df(date_df)

    # CSV podaci
    if csv_date_df:
        csv_df = prepare_df(csv_date_df)
        combined_df = db_df.unionByName(csv_df)
```

Slika 22: Transformacija podataka za dimenziju vrijeme

Nakon što su završile sve transformacije podataka za dimenzijske tablice, pristupilo se pripremi i transformaciji podataka za fact tablicu, što je bio najzahtjevniji i vremenski najintenzivniji dio procesa transformacije. Prvi korak uključivao je deduplikaciju dimenzijskih tablica odnosno uklanjanje viškova redaka s identičnim vrijednostima. U slučaju-

vima gdje su postojale višestruke pojave istih kombinacija atributa unutar dimenzijskih tablica, bilo je potrebno zadržati samo prvi redak kako bi se osigurala jednoznačnost tehničkih ključeva. Ova deduplikacija bila je ključna kako bi se spriječilo stvaranje pogrešnih veza prilikom spajanja s fact tablicom i kako bi broj redova odgovarao stvarnom broju transakcija bez višestrukih uparivanja istih dimenzija.

```
# Dedupliciranje dimenzija (uzimanje prvog retka ako ih ima više)
w_kartica = Window.partitionBy("card_type", "card_age_category", "user_id").orderBy("card_tk")
dim_kartica_dedup = dim_kartica.withColumn("rn", row_number().over(w_kartica)).filter(col("rn") == 1).drop("rn")

w_lokacija = Window.partitionBy("city").orderBy("location_id")
dim_lokacija_dedup = dim_lokacija.withColumn("rn", row_number().over(w_lokacija)).filter(col("rn") == 1).drop("rn")

w_auth = Window.partitionBy("authentication_method").orderBy("authentication_id")
dim_auth_dedup = dim_authentifikacija.withColumn("rn", row_number().over(w_auth)).filter(col("rn") == 1).drop("rn")

w_trgovac = Window.partitionBy("merchant_category").orderBy("merchant_id")
dim_trgovac_dedup = dim_trgovac.withColumn("rn", row_number().over(w_trgovac)).filter(col("rn") == 1).drop("rn")

w_kompozitna = Window.partitionBy("ip_address_flag", "previous_fraudulent_activity").orderBy("kompozitna_tk")
dim_kompozitna_dedup = dim_kompozitna.withColumn("rn", row_number().over(w_kompozitna)).filter(col("rn") == 1).drop("rn")

w_vrijeme = Window.partitionBy("year", "month", "day", "hour").orderBy("vrijeme_tk")
dim_vrijeme_dedup = dim_vrijeme.withColumn("rn", row_number().over(w_vrijeme)).filter(col("rn") == 1).drop("rn")
```

Slika 23: Deduplikacija dimenzijskih tablica

Nakon provedene deduplikacije, pristupilo se dohvaćanju podataka iz CSV datoteke i relacijske baze podataka. U ovom koraku prikupljeni su svi potrebni atributi za formiranje fact tablice. Kako bi se osigurala ispravne veze prema dimenzijskim tablicama, za svaki redak bilo je potrebno pronaći odgovarajuće tehničke ključeve dimenzija. To je postignuto spajanjem tablica na temelju jednog ili više atributa iz izvora (CSV/baza podataka) koji odgovaraju vrijednostima u dimenzijskim tablicama. Kada bi se pronašao redak u dimenzijskoj tablici s podudarnim vrijednostima, iz njega se dohvaćao tehnički ključ ili id koji se potom unosio kao strani ključ u fact tablicu.

Jedna od dvije degenerirane dimenzije nastala je na način da je preuzet atribut `account balance`, na temelju kojeg su kreirane kategorije: `low`, `average`, `above average`. Time je kvantitativni podatak pretvoren u kvalitativni.

```

# Baza podataka
joined_db = (
    db_trans_df.alias("t")
    .join(dim_kartica.alias("k"), col("t.card_fk") == col("k.card_id"), "left")
    .join(dim_lokacija_dedup.alias("l"), col("t.location_fk") == col("l.location_id"), "left")
    .join(dim_auth_dedup.alias("a"), col("t.authentication_fk") == col("a.authentication_id"), "left")
    .join(dim_trgovac_dedup.alias("m"), col("t.merchant_fk") == col("m.merchant_id"), "left")
    .join(dim_vrijeme_dedup.alias("d"),
        (year(col("t.timestamp")) == col("d.year")) &
        (month(col("t.timestamp")) == col("d.month")) &
        (dayofmonth(col("t.timestamp")) == col("d.day")) &
        (hour(col("t.timestamp")) == col("d.hour")), "left")
    .join(dim_kompozitna_dedup.alias("co"),
        (col("t.ip_address_flag") == col("co.ip_address_flag")) &
        (col("t.previous_fraudulent_activity") == col("co.previous_fraudulent_activity")), "left")
    .withColumn("account_balance_category", when(col("t.account_balance") < 1000, "Low")
        .when((col("t.account_balance") >= 1000) & (col("t.account_balance") < 5000), "Average")
        .otherwise("Above Average"))
    .select(
        col("t.id").alias("transaction_id"),
        col("t.transaction_amount"),
        col("t.daily_transaction_count"),
        col("t.avg_transaction_amount_7d"),
        col("t.failed_transaction_count_7d"),
        col("t.transaction_distance"),
        col("t.transaction_type"),
        col("t.fraud_label").cast("boolean").alias("fraud_label"),
        col("t.risk_score"),
        col("account_balance_category"),
        col("k.card_id"),
        col("l.location_id"),
        col("a.authentication_id"),
        col("m.merchant_id"),
        col("d.vrijeme_id"),
        col("co.kompozitna_id")
    )
)

```

Slika 24: Primjer transformacije podataka za fact tablicu

Važno je istaknuti da je u svim dimenzijskim i fact tablici dodan automatski generirani tehnički ključ, koji služi kao jedinstveni identifikator svakog retka i omogućuje pouzdano povezivanje između tablica unutar skladišta podataka. Ovaj pristup olakšava upravljanje podacima i održava integritet veze između činjenica i dimenzija.

Također, na kraju svake faze transformacije podataka provedena je kontrola broja zapisa kako bi se osigurala potpuna i točna obrada podataka. Ova provjera služi da potvrdi da je broj unesenih podataka u skladu s očekivanim brojem iz izvornog dataset-a.

Primjer takve provjere prikazan je na slici ispod, gdje se provjerava da fact tablica sadrži točno 50.000 redaka, što odgovara ukupnom broju transakcija u početnom dataset-u.

```

# Provjera broja podataka
assert fact_df.count() == 50000, "Number of transactions from step one of the project."

```

Slika 25: Primjer provjere broja podataka

## 5.3 Učitavanje podataka

Nakon završetka transformacije podataka, pristupilo se učitavanju pripremljenih podataka u postojeću bazu podataka koja je dizajnirana prema dimenzijskom modelu.

Na slici je prikazan primjer jedne tablice s unesenim podacima, koji predstavljaju konačni rezultat cjelokupnog ETL procesa.

Result Grid										
Filter Rows:										
Edit: Export/Import: Wrap Cell Contents: Fetch rows:										
nt_7d	transaction_distance	risk_score	fraud_label	account_balance_category	transaction_type	vrijeme_id	kartica_id	lokacija_id	autentifikacija_id	trgovac_id
	4610.43	0.3773	0	Above Average	ATM Withdrawal	2595	47479	9	4	1
	2284.15	0.5871	0	Above Average	ATM Withdrawal	45237	38591	11	4	3
	4273.45	0.6237	0	Above Average	POS	1218	27090	4	2	2
	3446.99	0.0785	0	Above Average	ATM Withdrawal	44257	44453	5	1	1
	2968.75	0.1635	0	Above Average	ATM Withdrawal	31976	11061	13	2	1
	1198.43	0.8101	0	Above Average	Bank Transfer	33738	5034	2	1	1
	1335.29	0.776	1	Above Average	Online	12801	32360	9	3	1
	1607.64	0.9484	1	Average	ATM Withdrawal	24079	15879	11	4	1
	498.65	0.0801	0	Above Average	POS	656	14557	6	3	1
	1156.90	0.007	1	Above Average	Bank Transfer	41172	24572	8	2	1

Output: Apply

#	Time	Action	Message
1	18:05:12	USE dw_dimensional	0 row(s) affected
2	18:05:16	SELECT * FROM fact_transakcija LIMIT 0, 1000	1000 row(s) returned

Slika 26: Primjer popunjene tablice nakon ETL procesa

## 6 Analiza podataka i vizualizacija

Nakon uspješne implementacije cjelokupnog procesa izgradnje skladišta podataka slijedi završna faza koja uključuje analizu i vizualizaciju podataka. Ova faza omogućuje korisnicima da iz postojećih podataka izvuku korisne uvide i temeljem njih donesu poslovne odluke.

**Analiza i vizualizacija** podataka ključni su koraci u procesu poslovne inteligencije, omogućujući organizacijama da interpretiraju velike količine podataka na intuitivan i pregledan način.

Ključnu ulogu u analitičkoj fazi rada ima Online Analytical Processing (OLAP) – tehnologija osmišljena za učinkovitu multidimenzionalnu analizu podataka. OLAP sustavi omogućuju izvođenje složenih upita nad velikim količinama podataka te korisnicima pružaju mogućnost da podatke pregledavaju i analiziraju iz različitih perspektiva.

**OLAP** se odnosi na sustave koji podržavaju prikaz tekstualnih i numeričkih podataka iz skladišta podataka, u svrhu donošenja odluka. Ovi alati omogućuju brzo i fleksibilno dobivanje odgovora na analitičke upite, pri čemu je struktura podataka optimizirana za analizu. Tijekom rada korištene su neke od osnovnih OLAP operacija:

- SLICE – filtriranje podataka unutar jedne dimenzije na određenu vrijednost ili raspon vrijednosti
- DICE – filtriranje podataka prema više dimenzija istovremeno, čime se formira ciljano manji podskup podataka
- ROLL-UP – sažimanje i agregacija podataka prelaskom na višu razinu hijerarhije unutar dimenzije čime se smanjuje količina detalja i povećava preglednost
- DRILL-DOWN – prelazak na detaljnije razine hijerarhije ili dodavanje dodatnih dimenzija
- PIVOT – mijenjanje redoslijeda dimenzija u prikazu podataka rotirajući podatkovnu kocku kako bi se omogućio prikaz podataka iz druge perspektive [4]

Za vizualizaciju podataka u ovom radu korišten je alat Tableau, jedno od najpoznatijih rješenja za poslovnu inteligenciju. Tableau omogućuje izradu interaktivnih grafikona, izvještaja i dashboarda te se jednostavno povezuje s različitim izvorima podataka. [5] Zahvaljujući intuitivnom sučelju i snažnim vizualnim mogućnostima, Tableau se pokazao kao izuzetno prikladan za analizu složenih podataka dobivenih iz skladišta.

### 6.1 Analiza iznosa transakcija

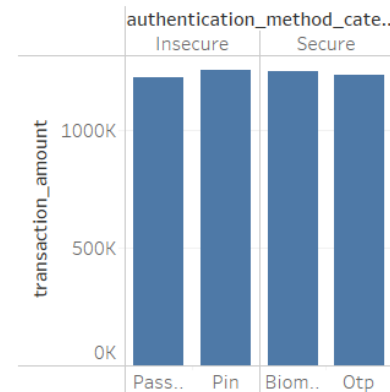
Prvi je dashboard fokusiran na analizu iznosa transakcija kako bi se otkrili obrasci i trendovi u potrošnji korisnika. Sadrži četiri grafička prikaza, pri čemu svaki prikazuje iznos

transakcija iz druge perspektive. Te se perspektive odnose na prikaz iznosa transakcija prema vremenu, namjeni, autentifikaciji i lokacijama na kojima su obavljene.

#### Iznos transakcija prema lokaciji

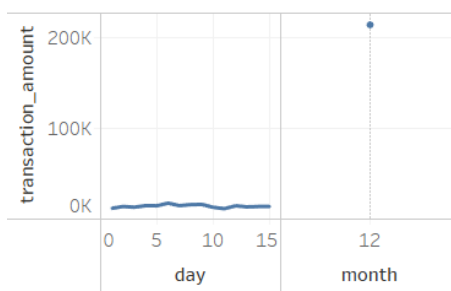
population_c...	city	state				
		Australia	India	Japan	United Ki..	United St..
Large City	Bangalore		155.061			
	Birmingham			595.220		
	Chicago				142.851	
	Houston				149.829	
	Hyderabad		143.701			
	Melbourne	603.050				
	Nagoya			599.077		
	Pune		147.062			
	San Francisco					149.961
Mega City	Sydney	393.916				
	Delhi		143.426			
	London			390.871		
	Los Angeles				158.227	
	Mumbai		395.802			
	New York					399.431
	Tokio			403.065		

#### Iznos transakcija prema autentifikaciji

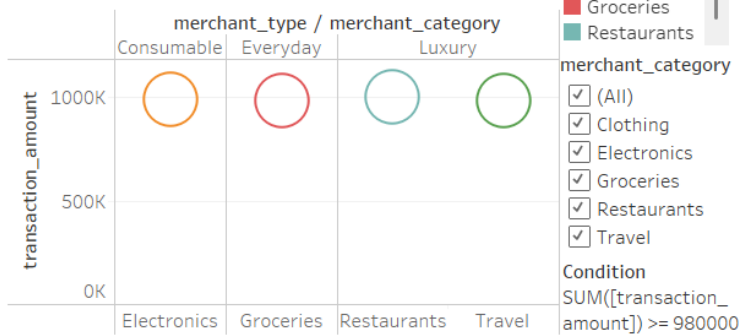


## ANALIZA IZNOSA TRANSAKCIJA

#### Iznos transakcija prema vremenu



#### Iznos transakcija prema namjeni



Slika 27: Dashbord analize iznosa transakcija

Grafički prikaz transakcijskih iznosa prema lokaciji izrađen je korištenjem **drill-down** pristupa, koji omogućuje postupno istraživanje podataka od razine država prema gradovima. Dodatno je prikazan i atribut koji označava kojoj kategoriji populacije pripada svaki grad. Prikaz je strukturiran kao **pivot**-tablica: gradovi i njihove populacijske kategorije nalaze se u redovima, dok stupce predstavljaju pojedine države. Vrijednosti unutar tablice prikazuju ukupne iznose transakcija.

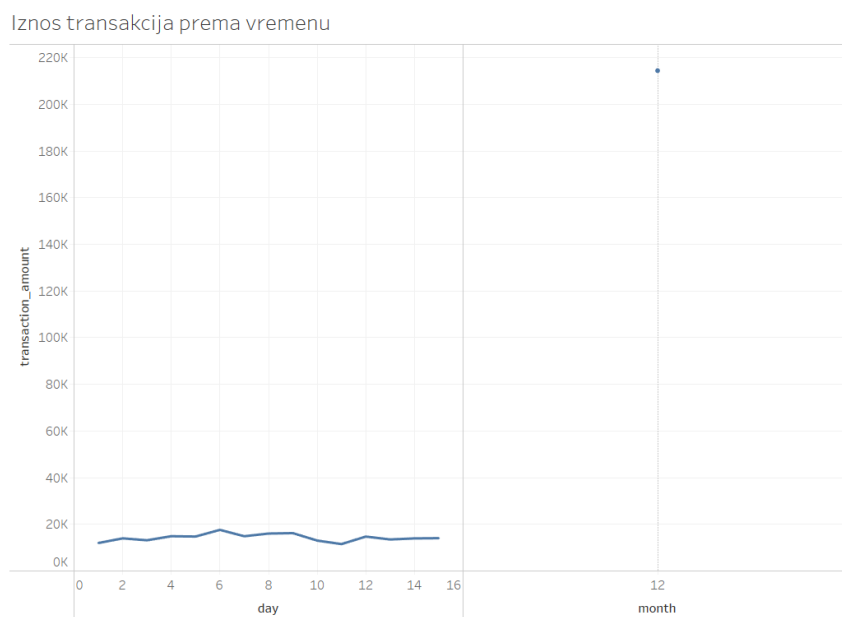
## Iznos transakcija prema lokaciji

population_category	city	state				
		Australia	India	Japan	United Kingdom	United States
Large City	Bangalore		155.061			
	Birmingham				595.220	
	Chicago					142.851
	Houston					149.829
	Hyderabad		143.701			
	Melbourne	603.050				
	Nagoya			599.077		
	Pune		147.062			
	San Francisco					149.961
Mega City	Sydney	393.916				
	Delhi		143.426			
	London				390.871	
	Los Angeles					158.227
	Mumbai		395.802			
	New York					399.431
	Tokio			403.065		

Slika 28: Iznos transakcija prema lokaciji

Cilj ovog prikaza je omogućiti brzu identifikaciju lokacija s neuobičajeno visokim ili niskim iznosima transakcija. Prikaz olakšava uočavanje odstupanja jer omogućuje izravnu usporedbu različitih lokacija i populacijskih kategorija unutar iste države ili među državama. Time se analitičarima omogućuje da ciljano usmjere daljnju analizu na sumnjive gradove ili regije.

Grafički prikaz iznosa transakcija prema vremenu izrađen je korištenjem djelomičnog **drill-down** pristupa, pri čemu su prikazani podaci na razini mjeseca i dana. Prikaz je oblikovan kao **pivot** jer su zamijenjene pozicije mjeseca i dana u odnosu na uobičajenu poredak. Uz to, primijenjen je i **dice** pristup budući da su prikazane samo određene vrijednosti, konkretno prvih 15 dana u 12. mjesecu.



Slika 29: Iznos transakcija prema vremenu

Cilj ovog prikaza bio je analizirati iznose transakcija tijekom blagdanskih razdoblja,

s naglaskom na prvu polovicu prosinca. U tom vremenu potrošnja prirodno raste, što stvara kontekst u kojem se potencijalne prijevare mogu lakše prikriti unutar pojačanog broja i vrijednosti transakcija. Ovakav vremenski prikaz omogućuje analitičarima da prepoznaju neuobičajena odstupanja od očekivanih sezonskih obrazaca te usmjere pažnju na transakcije koje ne prate tipične blagdanske trendove.

Grafički prikaz iznosa transakcija prema namjeni izrađen je kao **drill-down** gdje su prikazani tip i kategorija trgovca kojem je transakcija namijenjena. Također, primijenjen je **slice** filter kojim je zadano da iznos transakcije mora biti najmanje 980 000, što je rezultiralo smanjenjem prikaza jer jedna od pet kategorija nije zadovoljila taj uvjet.

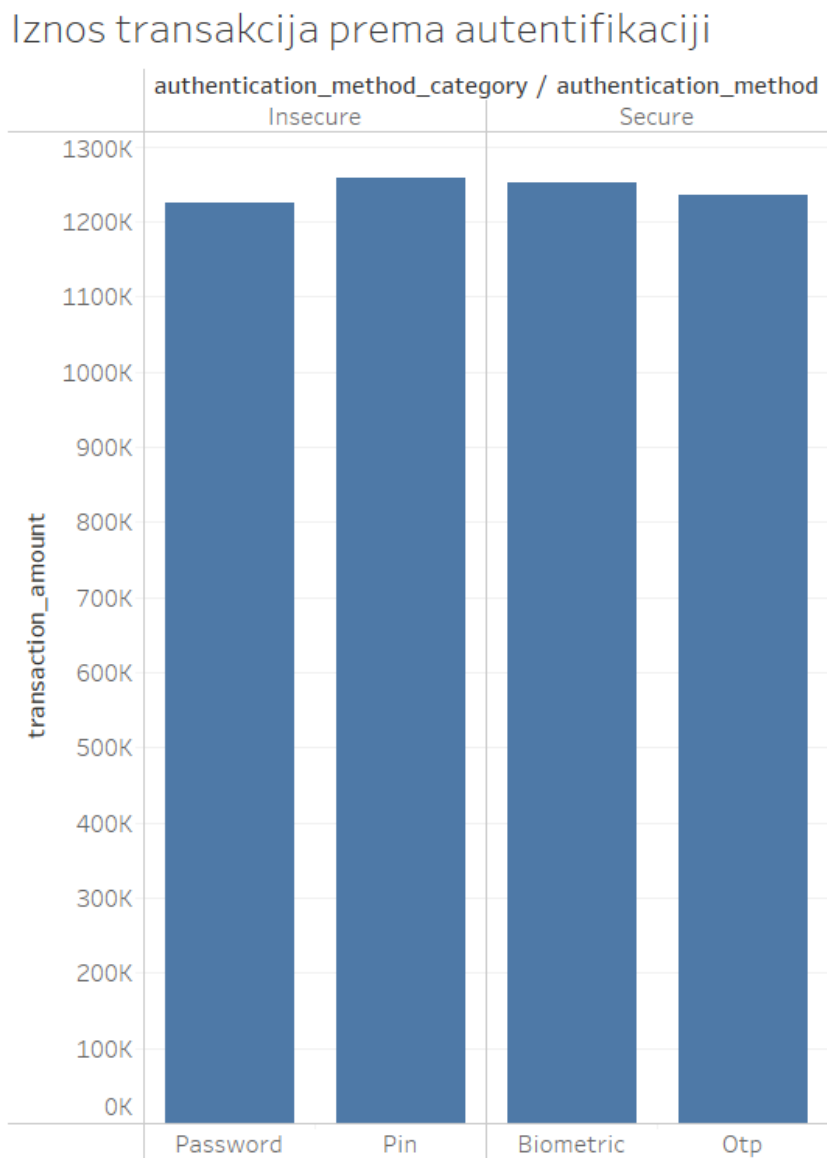


Slika 30: Iznos transakcija prema namjeni

Cilj ovog prikaza je identificirati iznose transakcija prema namjeni i vrstama namjene, filtriranjem na veće transakcije kako bi se uočile ključne kategorije s visokim financijskim aktivnostima. Ovaj pristup pomaže u prepoznavanju potencijalno rizičnih obrazaca u potrošnji, što može biti korisno upravljanje rizicima.

Posljednji grafički prikaz na ovom dashboardu prikazuje iznos transakcija prema načinu autentifikacije. Primijenjen je **drill-down** pristup koji omogućuje detaljniju analizu iznosa transakcija prema korištenoj metodi autentifikacije i pripadajućoj kategoriji.





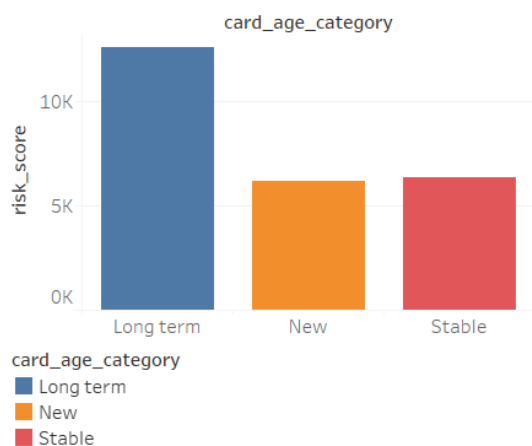
Slika 31: Iznos transakcija prema autentifikaciji

Cilj ovog prikaza je pratiti iznose transakcija u odnosu na različite metode autentifikacije kako bi se identificirale potencijalne nepravilnosti ili rizici povezani s određenim načinima potvrde identiteta. Time se olakšava otkrivanje neuobičajenih obrazaca koji mogu ukazivati na pokušaje prijave.

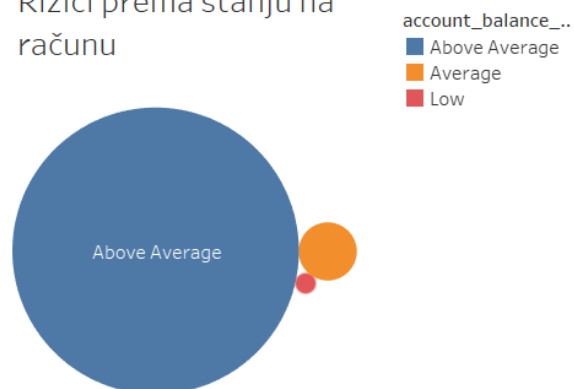
## 6.2 Analiza rizika

Dashboard za analizu rizika fokusira se na prepoznavanje sumnjivih transakcija kroz različite dimenzije podataka. Sastoji se od četiri grafička prikaza, od kojih svaki analizira rizike na temelju različitih atributa kao što su vrsta kartice, stanje na računu te sigurnosni flagovi povezani s transakcijama.

Rizici prema kartici

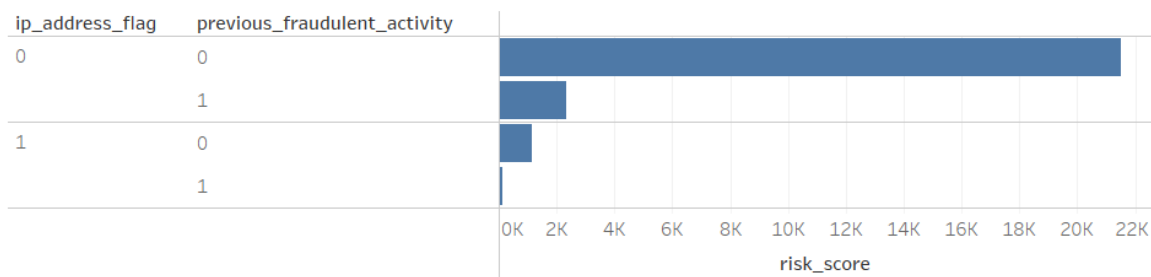


Rizici prema stanju na računu



## ANALIZA RIZIKA

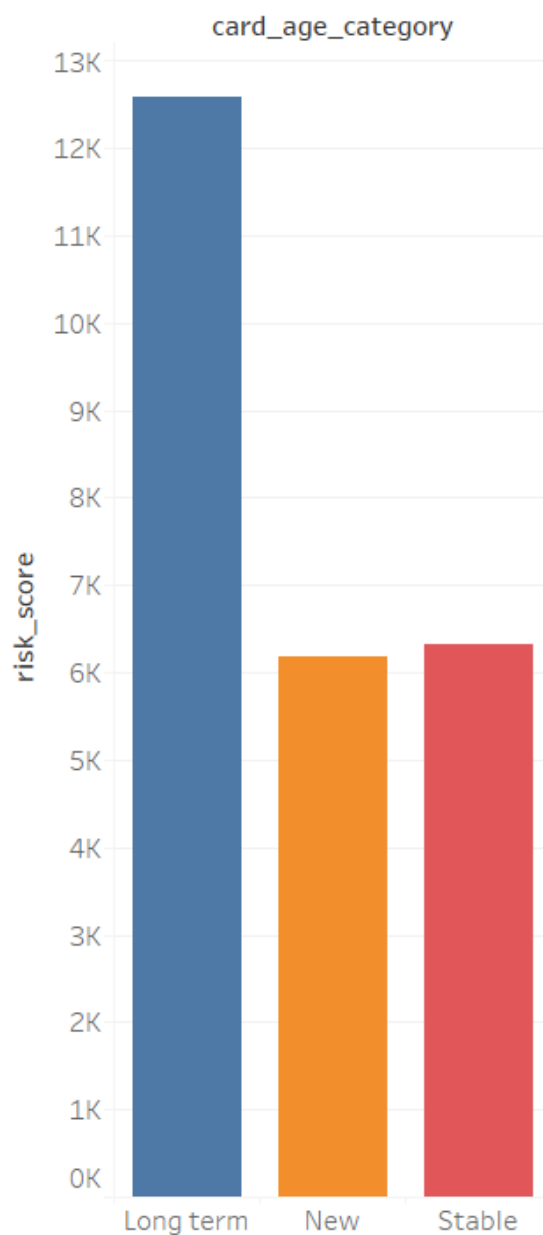
Rizici prema flagovima



Slika 32: Dashbord analize rizika

Grafički prikaz rizika prema kartici izrađen je korištenjem roll-up pristupa, pri čemu su prikazane samo kategorije kartica, bez detalja o pojedinim tipovima. Na taj način omogućena je analiza rizika na višoj, sažetijoj razini apstrakcije.

## Rizici prema kartici

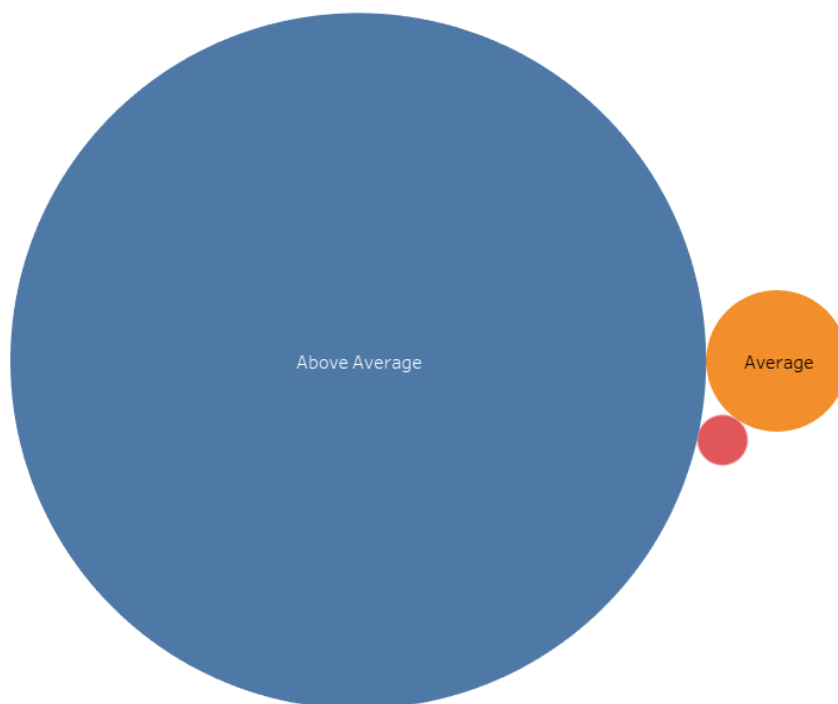


Slika 33: Rizici prema kartici

Cilj ovog prikaza je omogućiti brzu identifikaciju kategorija kartica koje su povezane s višim razinama rizika. Agregiranjem podataka na razinu kategorije olakšava se uočavanje potencijalno rizičnih skupina kartica, što može pomoći u donošenju strateških odluka o daljnjem praćenju ili ograničavanju određenih kartičnih kategorija.

Grafički prikaz rizika prema stanju na računu uključuje degeneriranu dimenziju, pri čemu se analizira ukupan rizik za svaku kategoriju stanja računa. Ovakav prikaz omogućuje uvid u to postoje li određeni rasponi stanja na računu koji su učestalije povezani s rizičnim transakcijama.

## Rizici prema stanju na računu

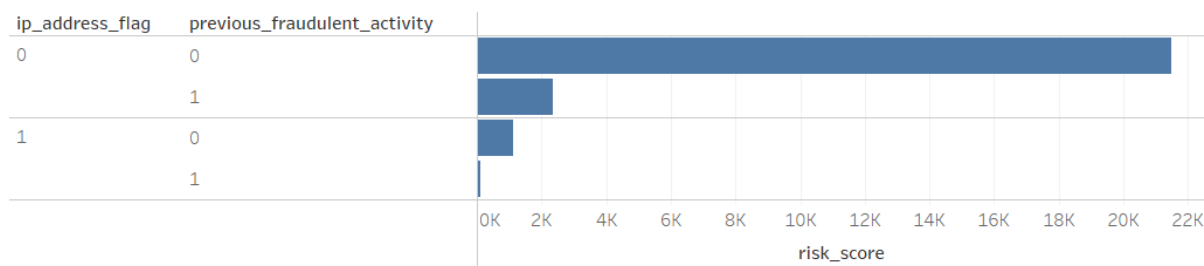


Slika 34: Rizici prema stanju na računu

Cilj ovog prikaza je utvrditi postoji li povezanost između stanja na računu i razine rizika transakcija. Analizom ukupnog rizika po kategorijama stanja omogućuje se identifikacija potencijalno rizičnih skupina korisnika, u ovom slučaju onih s iznad prosječnim saldom, što može pomoći u ranijem otkrivanju sumnjivih aktivnosti.

Grafički prikaz rizika prema flagovima prikazuje razine rizika na temelju oznaka koje se odnose na IP adresu i prethodnu prijevarnu aktivnost korisnika. Prikaz je oblikovan kao *pivot*-tablica jer su zamijenjene pozicije atributa po kojima se vrši analiza i samog rizika koji se promatra.

### Rizici prema flagovima



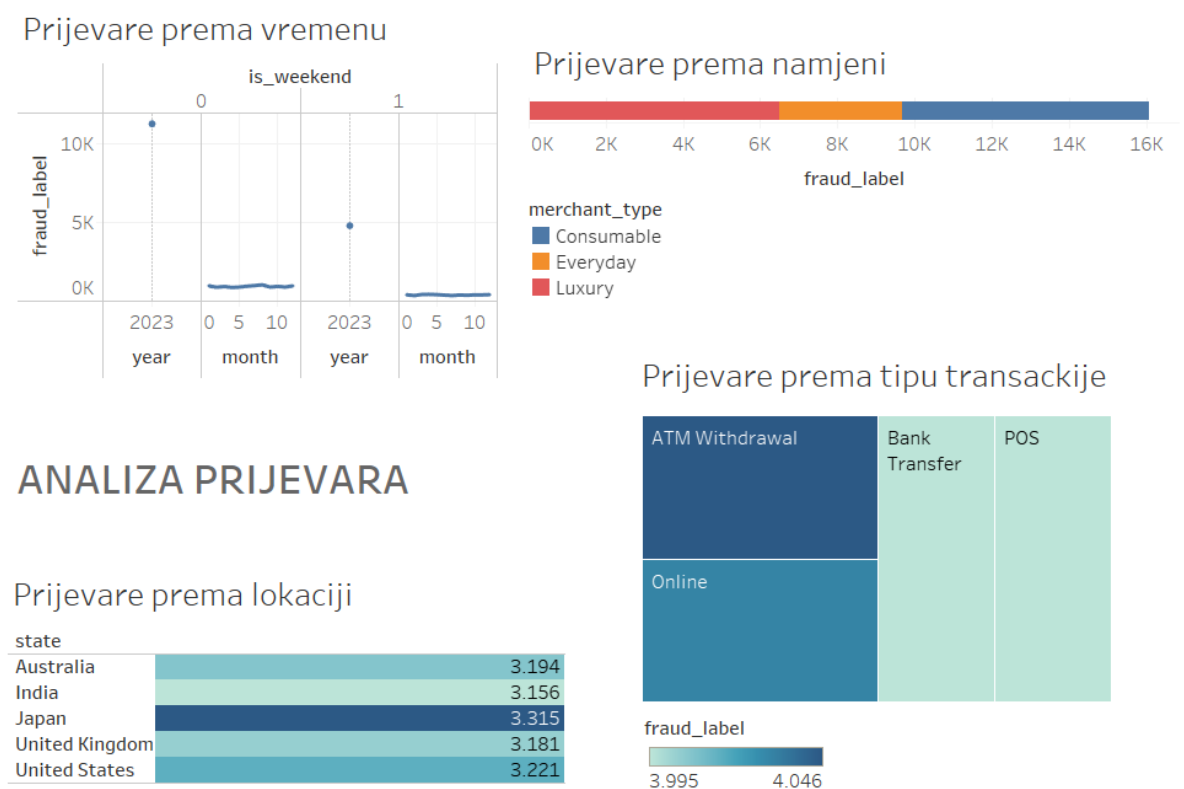
Slika 35: Rizici prema flagovima

Cilj ovog prikaza je analizirati kako sigurnosni flagovi, sumnjive IP adrese i prethodne

prijevarne aktivnosti utječu na razinu rizika transakcija. Ovakva analiza omogućuje prepoznavanje kombinacija faktora koje se češće pojavljuju uz rizične transakcije, što može pomoći u unapređenju mehanizama za detekciju prijevara.

### 6.3 Analiza prijevara

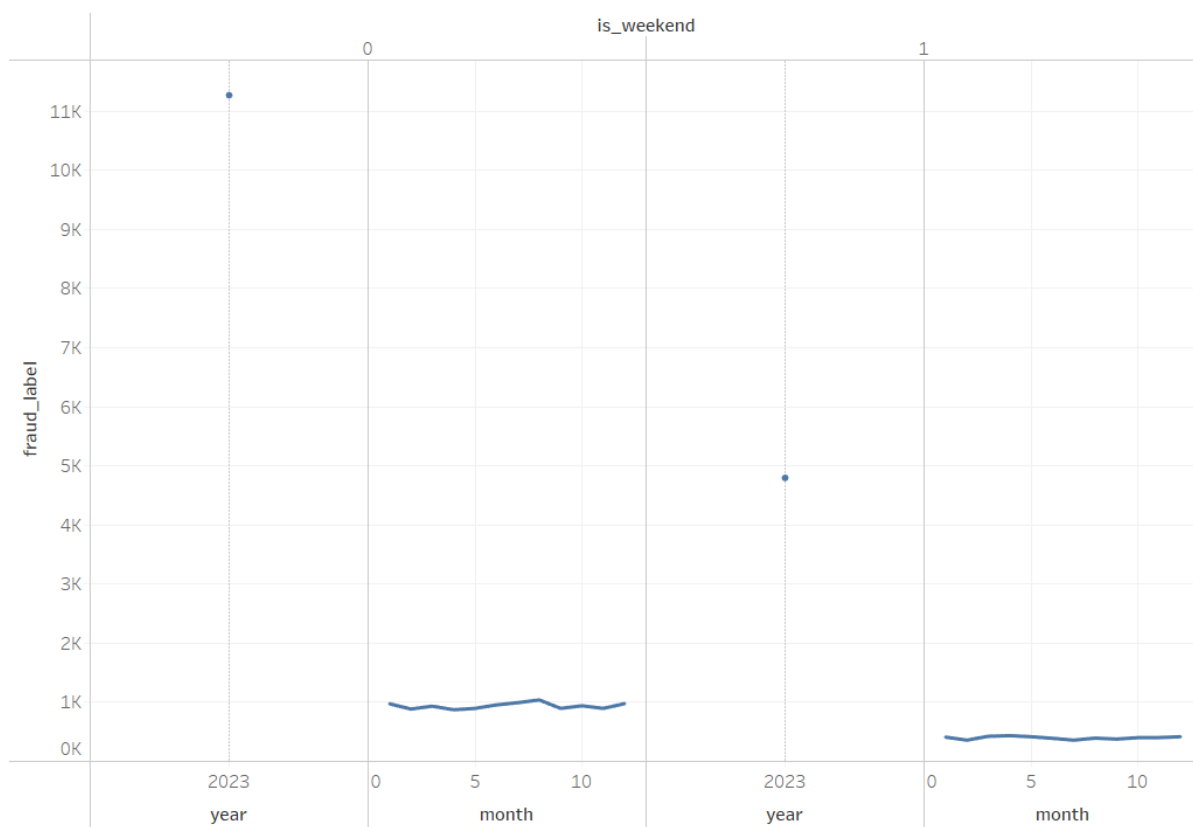
Posljednji dashboard predstavlja središnji dio analize jer je izravno usmjeren na identifikaciju i razumijevanje obrazaca prijevornih transakcija. Sadrži četiri grafička prikaza koja obuhvaćaju ključne dimenzije povezane s prijevarama: vrijeme, namjenu, lokaciju i tip transakcije. Kroz sve prethodne analize oblikovan je kontekst potreban za ovaj završni korak, vizualizaciju prijevara. Time se omogućuje preciznije prepoznavanje faktora koji doprinose pojavi sumnjivih aktivnosti te se analitičarima pruža alat za donošenje informiranih odluka u prevenciji prijevara.



Slika 36: Dashbord analize prijevara

Grafički prikaz prijevara prema vremenu predstavlja djelomični **drill-down** jer ne ide do krajnje razine vremenske hijerarhije. Prikazane su prijevare agregirane po mjesecima i godinama, uz dodatnu podjelu na radne dane i vikende.

### Prijevare prema vremenu

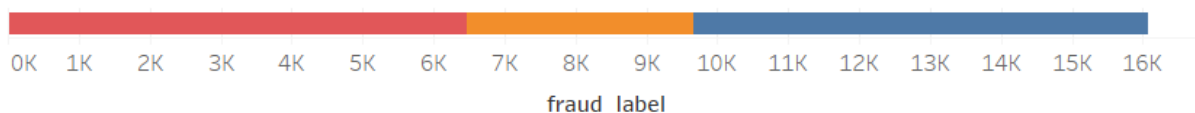


Slika 37: Prijevare prema vremenu

Cilj ovog prikaza je identificirati vremenske obrasce u pojavi prijevvara te razlikovati učestalost prijevvara između radnih dana i vikenda. Takva analiza pomaže u prepoznavanju perioda s povećanim rizikom, čime se može unaprijediti pravovremena prevencija i nadzor.

Grafički prikaz prijevvara prema namjeni prikazuje učestalost prijevvara u odnosu na tip trgovca. Prikaz je izrađen kao **pivot**-tablica jer su zamijenjene pozicije atributa, tip trgovca kao kriterij analize i broj prijevvara kao vrijednost, kako bi se omogućila preglednija usporedba između različitih kategorija. Budući da se prikazuje samo tip bez pripadajuće kategorije trgovca, uočava se i primjena **roll-up** operacije, kojom se agregiraju podaci na višu razinu hijerarhije dimenzije.

### Prijevare prema namjeni



Slika 38: Prijevare prema namjeni

Cilj ovog prikaza je utvrditi koje kategorije trgovaca bilježe veću učestalost prijevvarnih transakcija. Na taj način moguće je prepoznati rizične sektore unutar trgovinske mreže i

usmjeriti dodatnu kontrolu na najukritičnije tipove trgovaca.

Grafički prikaz prijevара prema lokaciji izrađen je primjenom `roll-up` pristupa, pri čemu se podaci agregiraju na razinu država, bez detaljnije razrade na razinu gradova. Time se omogućuje pregled distribucije prijevара na višoj geografskoj razini.

### Prijevare prema lokaciji

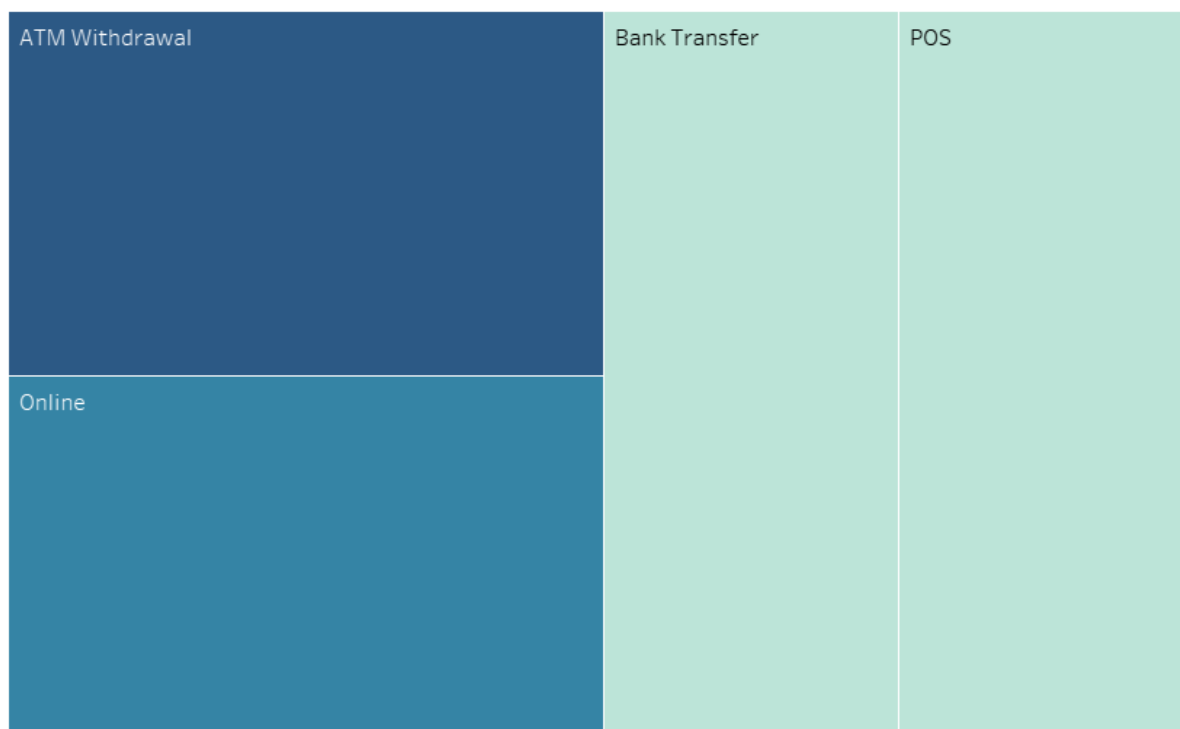
state	
Australia	3.194
India	3.156
Japan	3.315
United Kingdom	3.181
United States	3.221

Slika 39: Prijevare prema lokaciji

Cilj ovog prikaza je dobiti uvid u geografsku raspodjelu prijevара na razini država kako bi se identificirale lokacije s povećanom učestalošću sumnjivih transakcija. Takav pregled pomaže u prepoznavanju zemalja koje zahtijevaju pojačan nadzor ili dodatne mjere sigurnosti u sustavu plaćanja.

Grafički prikaz prijevара prema tipu transakcije temelji se na degeneriranoj dimenziji, pri čemu se tip transakcije koristi kao jedini atribut za analizu. Prikaz pokazuje ukupan broj prijevара za svaki tip transakcije, omogućujući jasan uvid u to koje vrste transakcija su najčešće povezane s prijevarama.

## Prijevare prema tipu transakcije



Slika 40: Prijevare prema tipu transakcije

Cilj ovog prikaza je identificirati koje vrste transakcija su najčešće povezane s prijevarama. Analizom učestalosti prijevara prema tipu transakcije omogućuje se usmjeravanje sigurnosnih mjera na transakcijske obrasce koji predstavljaju veći rizik.

Svi izrađeni dashboardi zajednički omogućuju dublje razumijevanje transakcijskog ponašanja kako bi se na temelju toga mogli donijeti zaključci vezani uz potencijalne prijevare. Prikazivanjem podataka iz različitih perspektiva omogućuje se uočavanje neuobičajenih obrazaca, odstupanja i rizičnih kombinacija atributa. Na taj način analitičari mogu prepoznati gdje i kada se prijevare najčešće događaju, te koji su čimbenici s njima povezani. Dashboardi time postaju alat za podršku u ranom otkrivanju prijevara i donošenju učinkovitijih strategija prevencije.



## 7 Zaključak

Provedenom analizom prijevara u transakcijama, uz korištenje sustava poslovne inteligencije i kreiranih dashboarda, uspješno je demonstrirana učinkovitost transformacije sirovih transakcijskih podataka u korisne uvide. Kroz sustavno prikupljanje i obradu podataka formirano je skladište podataka koje služi kao temelj za daljnju analizu i vizualizaciju.

Izrađeni dashboardi pokazali su se dobrim alatom za brzu identifikaciju anomalija i potencijalnih prijetnji. Ovi vizualni prikazi omogućavaju analitičarima da s preciznošću lociraju visokorizične transakcije i razumiju temeljne uzroke prijevara. Stoga se može zaključiti da sustavi poslovne inteligencije ne samo da pružaju podršku u ranom otkrivanju prijevara, već su i temelj za razvoj strategija zaštite, što izravno doprinosi jačanju sigurnosti poslovanja.

## Literatura

- [1] Oreški, G. (2025). *Uvod u poslovnu inteligenciju*. PDF. Merlin – sustav za e-učenje, Fakultet informatike u Puli.
- [2] Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling* (3rd ed.). Wiley.
- [3] Komorowski, M., Marshall, D. C., Saliccioli, J. D., & Crutain, Y. (2016). *Exploratory Data Analysis*.
- [4] Chaudhuri, S., & Dayal, U. (1997). Online Analytical Processing (OLAP) for Decision Support.
- [5] Tableau Software. (2025). *Tableau Documentation: Get Started Tutorial*.