

Analiza prodaje namirnica

*Predmetni projekat iz predmeta
Sistemi baza podataka*



Opis skupa podataka

Skup podataka sadrži transakcione podatke o prodaji prehrambenih proizvoda u periodu od četiri meseca. Sastoji se od sedam povezanih CSV datoteka koje obuhvataju detalje o prodajama, proizvodima, kupcima, zaposlenima, kao i geografske informacije o gradovima i državama.

Sedam ključnih datoteka:

- sales.csv: Centralna datoteka sa svim prodajnim transakcijama.
- products.csv: Detalji o proizvodima, uključujući cene, klase i kategorije.
- customers.csv: Demografski i geografski podaci o kupcima.
- employees.csv: Informacije o zaposlenima koji su realizovali prodaje.
- categories.csv: Definicije kategorija proizvoda.
- cities.csv & countries.csv: Geografski podaci za segmentaciju tržišta.

Link ka skupu podataka: <https://www.kaggle.com/datasets/andrexibiza/grocery-sales-dataset>

Semantika obeležja

Tabela categories

- **CategoryID:** Jedinstveni identifikator za svaku kategoriju proizvoda.
- **CategoryName:** Naziv kategorije proizvoda.

Tabela cities

- **CityID:** Jedinstveni identifikator za svaki grad.
- **CityName:** Naziv grada.
- **Zipcode:** Poštanski broj.
- **CountryID:** Referenca (strani ključ) na odgovarajuću državu.

Semantika obeležja

Tabela countries

- **CountryID:** Jedinstveni identifikator za svaku državu.
- **CountryName:** Naziv države.
- **CountryCode:** Dvoslovni kod države (npr. RS).

Tabela customers

- **CustomerID:** Jedinstveni identifikator za svakog kupca.
- **FirstName:** Ime kupca.
- **MiddleInitial:** Srednje slovo (inicijal) kupca.
- **LastName:** Prezime kupca.
- **cityID:** Grad kupca (referenca na tabelu 'cities').
- **Address:** Adresa stanovanja kupca.

Semantika obeležja

Tabela employees

- **EmployeeID:** Jedinstveni identifikator za svakog zaposlenog.
- **FirstName:** Ime zaposlenog.
- **MiddleInitial:** Srednje slovo (inicijal) zaposlenog.
- **LastName:** Prezime zaposlenog.
- **BirthDate:** Datum rođenja zaposlenog.
- **Gender:** Pol zaposlenog.
- **CityID:** Jedinstveni identifikator za grad (referenca na tabelu 'cities').
- **HireDate:** Datum zaposlenja.

Tabela products

- **ProductID:** Jedinstveni identifikator za svaki proizvod.
- **ProductName:** Naziv proizvoda.
- **Price:** Cena po jedinici proizvoda.
- **CategoryID:** Jedinstveni identifikator kategorije (referenca).
- **Class:** Klasa proizvoda (Low, Medium, High).
- **ModifyDate:** Datum poslednje izmene.
- **Resistant:** Kategorija otpornosti proizvoda.
- **IsAllergic:** Označava da li je proizvod alergen.
- **VitalityDays:** Klasifikacija vitalnosti proizvoda (npr. rok trajanja u danima).

Semantika obelezja

Tabela sales

- **SalesID:** Jedinstveni identifikator za svaku prodaju.
- **SalesPersonID:** Zaposleni odgovoran za prodaju (referenca).
- **CustomerID:** Kupac koji je izvršio kupovinu (referenca).
- **ProductID:** Prodati proizvod (referenca).
- **Quantity:** Broj prodatih jedinica.

- **Discount:** Primjenjeni popust na prodaju.
- **TotalPrice:** Konačna cena prodaje nakon popusta.
- **SalesDate:** Datum i vreme prodaje.
- **TransactionNumber:** Jedinstveni identifikator transakcije.

AGREGACIJE

Tijana

- Koji su ukupni prihodi (TotalPrice) i prosečni popusti (Discount) za svaku kategoriju proizvoda po mesecima? Identifikovati najprofitabilnije kategorije i kategorije sa najvećim prosečnim popustom.
- Koji su top 5 najprodavanijih proizvoda (po Quantity i TotalPrice) u celom periodu, i koliki je procenat tih proizvoda označen kao IsAllergic?
- Kako se menjala ukupna prodaja (TotalPrice) i broj transakcija po danima u nedelji tokom celog perioda? Postoje li dani sa značajno većom ili manjom prodajom?
- Postoji li korelacija između Class (klase proizvoda) i Resistant (otpornosti proizvoda) atributa i njihove ukupne prodaje (TotalPrice)? Koji Class/Resistant kombinacije ostvaruju najveći prihod?
- Koje su top 3 kategorije proizvoda koje se najviše prodaju (po TotalPrice) u svakom od top 5 gradova po ukupnoj prodaji?

AGREGACIJE

Milica

- Koliki je prosečan broj kupovina i prosečna ukupna potrošnja (TotalPrice) po kupcu u različitim gradovima? Identifikovati gradove sa "najlojalnijim" (najviše kupovina/potrošnje) kupcima.
- Koji su top 3 zaposlena (SalesPerson) po ukupnoj prodaji (TotalPrice)? Uporediti njihove demografske podatke (pol, datum zaposlenja) sa prosekom svih zaposlenih.
- Koliko se proizvoda označenih kao IsAllergic prodaje u svakom gradu, po Quantity i TotalPrice? Koji gradovi imaju najveću prodaju alergijskih proizvoda?
- Kakav je prosečan 'sadržaj korpe' po različitim gradovima? Analizirati prosečan broj stavki po transakciji i prosečnu vrednost jedne transakcije za svaki grad.
- Izracunati za svaki grad i svaku kategoriju koliko se proizvoda iz date kategorije prodalo po gradu. Koja kategorija proizvoda je najprodavanija?

Inicijalna logička šema

Kolekcija sales

- SalesID
- Quantity
- Discount
- TotalPrice
- SalesDate
- TransactionNumber
- ProductID (referenca ka odgovarajućem proizvodu)
- CustomerID (referenca ka odgovarajućem kupcu)
- SalesPersonID (referenca ka odgovarajućem prodavcu)

Kolekcija products

- ProductID
- ProductName
- Price
- ModifyDate
- Resistant
- Class
- IsAllergic
- VitalityDays
- Category: { CategoryID, CategoryName }

Inicijalna logička šema

Kolekcija customers

- CustomerID
- FirstName
- LastName
- MiddleInitial
- Address
- Location: { CityID, CityName, Zipcode, Country : { CountryID, CountryName, CountryCode } }

Kolekcija employees

- EmployeeID
- FirstName
- LastName
- MiddleInitial
- BirthDate
- Gender
- HireDate
- Location: { CityID, CityName}

Skripta za pretprocesiranje podataka u Python-u

```
1 import pandas as pd
2 import os
3
4
5 DATA_PATH = "."
6 SALES_FILE = "sales.csv"
7 PRODUCTS_FILE = "products.csv"
8 OUTPUT_FILE = "sales_cleaned.csv"
9
10
11 def clean_and_calculate_sales_data():
12
13     sales_path = os.path.join(DATA_PATH, SALES_FILE)
14     products_path = os.path.join(DATA_PATH, PRODUCTS_FILE)
15     output_path = os.path.join(DATA_PATH, OUTPUT_FILE)
16
17     print("--- Učitavanje podataka ---")
18     try:
19         print(f"Učitavanje fajla: {sales_path}...")
20         df_sales = pd.read_csv(sales_path)
21         print(f"Učitavanje fajla: {products_path}...")
22         df_products = pd.read_csv(products_path)
23         print("Fajlovi uspešno učitani.")
24     except FileNotFoundError as e:
25         print(f"Greška: Fajl nije pronađen. {e}")
26     return
27
28
29     print("\n--- Čišćenje i priprema podataka ---")
30
31     print("Čišćenje 'SalesDate' kolone...")
32     df_sales['SalesDate'] = pd.to_datetime(df_sales['SalesDate'], errors='coerce')
33
34     initial_rows = len(df_sales)
35     df_sales.dropna(subset=['SalesDate'], inplace=True)
36     rows_dropped = initial_rows - len(df_sales)
37     if rows_dropped > 0:
38         print(f"Izbačeno {rows_dropped} redova sa nevalidnim ili praznim datumom.")
39
40     df_prices = df_products[['ProductID', 'Price']].copy()
41
42     print("\n--- Spajanje podataka o prodaji i cena proizvoda ---")
43
44     df_merged = pd.merge(df_sales, df_prices, on='ProductID', how='left')
45
46     missing_prices = df_merged['Price'].isnull().sum()
```

```
45     missing_prices = df_merged['Price'].isnull().sum()
46     if missing_prices > 0:
47         print(
48             f"UPOZORENJE: Pronađeno {missing_prices} prodaja za koje ne postoji cena proizvoda. TotalPrice će biti 0 za njih.")
49         df_merged['Price'].fillna(0, inplace=True)
50
51
52     print("\n--- Izračunavanje nove 'TotalPrice' kolone ---")
53
54     df_merged['TotalPrice'] = (df_merged['Price'] - (df_merged['Price'] * df_merged['Discount'])) * df_merged[
55         'Quantity']
56
57     df_merged['TotalPrice'] = df_merged['TotalPrice'].round(2)
58
59     print("Nova 'TotalPrice' kolona je uspešno izračunata.")
60     print("Primer prvih 5 izračunatih vrednosti:")
61     print(df_merged[['ProductID', 'Price', 'Discount', 'Quantity', 'TotalPrice']].head())
62
63
64     final_df = df_merged.drop(columns=['Price'])
65
66
67     print(f"\n--- Čuvanje očišćenog fajla na lokaciji: {output_path} ---")
68
69     final_df.to_csv(output_path, index=False)
70
71     print("\nČišćenje i izračunavanje podataka je završeno!")
72
73
74     if __name__ == "__main__":
75         clean_and_calculate_sales_data()
```

Skripta za kreiranje i popunjavanje kolekcija u Python-u

```
* main.py > import_collection
1  import pandas as pd
2  from pymongo import MongoClient
3  from pymongo.errors import ConnectionFailure
4  import os
5
6  MONGO_URI = "mongodb://localhost:27017/"
7  DB_NAME = "grocery_sales_db"
8  DATA_PATH = "."
9
10 PRODUCTS_COLLECTION = "products"
11 CUSTOMERS_COLLECTION = "customers"
12 EMPLOYEES_COLLECTION = "employees"
13 SALES_COLLECTION = "sales"
14
15
16 def connect_to_mongo(uri):
17     print("Povezivanje na MongoDB...")
18     try:
19         client = MongoClient(uri)
20         # Provera konekcije
21         client.admin.command('ping')
22         print("MongoDB konekcija uspešna.")
23         return client
24     except ConnectionFailure as e:
25         print(f"Greška pri povezivanju na MongoDB: {e}")
26         return None
27
28
29 def preprocess_lookup_data(path):
30     print("Priprema podataka za ugnježđivanje (lookup)...")
31
32     try:
33         categories_df = pd.read_csv(os.path.join(path, 'categories.csv'))
34         cities_df = pd.read_csv(os.path.join(path, 'cities.csv'))
35         countries_df = pd.read_csv(os.path.join(path, 'countries.csv'))
36     except FileNotFoundError as e:
37         print(f"Greška: CSV fajl nije pronađen. Proverite putanju. {e}")
38         return None, None
39
40     categories_map = {
41         row['CategoryID']: {
42             'CategoryID': row['CategoryID'],
43             'CategoryName': row['CategoryName']
44         }
45     }
46     for index, row in categories_df.iterrows():
47         categories_map[row['CategoryID']] = {
48             'CategoryID': row['CategoryID'],
49             'CategoryName': row['CategoryName']
50         }
```

```
48     locations_df = pd.merge(cities_df, countries_df, on='CountryID', how='left')
49     locations_map = {
50         row['CityID']: {
51             'CityID': row['CityID'],
52             'CityName': row['CityName'],
53             'Zipcode': row['Zipcode'],
54             'Country': {
55                 'CountryID': row['CountryID'],
56                 'CountryName': row['CountryName'],
57                 'CountryCode': row['CountryCode']
58             }
59         }
60     }
61     for index, row in locations_df.iterrows():
62         locations_map[row['CityID']] = {
63             'CityID': row['CityID'],
64             'CityName': row['CityName'],
65             'Zipcode': row['Zipcode'],
66             'Country': {
67                 'CountryID': row['CountryID'],
68                 'CountryName': row['CountryName'],
69                 'CountryCode': row['CountryCode']
70             }
71         }
72
73     print("Lookup podaci uspešno pripremljeni.")
74     return categories_map, locations_map
75
76
77 def import_collection(db, collection_name, csv_file, lookup_maps=None):
78     print(f"\n--- Obrada kolekcije: {collection_name} ---")
79
80     try:
81         df = pd.read_csv(csv_file)
82     except FileNotFoundError:
83         print(f"Fajl {csv_file} nije pronađen. Preskačem.")
84         return
85
86     print(f"Brisanje postojeće kolekcije '{collection_name}'...")
87     db[collection_name].delete_many({})
88
89     documents_to_insert = []
90
91     if collection_name == PRODUCTS_COLLECTION:
92         categories_map = lookup_maps.get('categories', {})
93         for index, row in df.iterrows():
94             doc = {
95                 '_id': row['ProductID'],
96                 'ProductName': row['ProductName'],
97                 'Price': row['Price'],
98                 'Class': row['Class'],
99                 'ModifyDate': pd.to_datetime(row['ModifyDate']),
100                'Resistant': row['Resistant'],
101                'IsAllergic': row['IsAllergic'],
102                'VitalityDays': row['VitalityDays'],
103                'Category': categories_map.get(row['CategoryID'], {})
```

Skripta za kreiranje i popunjavanje kolekcija u Python-u

```
    category = categories_map.get(row['CategoryID'], {})
94    documents_to_insert.append(doc)

95 elif collection_name == CUSTOMERS_COLLECTION:
96     locations_map = lookup_maps.get('locations', {})
97     for index, row in df.iterrows():
98         doc = {
99             '_id': row['CustomerID'],
100            'FirstName': row['FirstName'],
101            'MiddleInitial': row.get('MiddleInitial'),
102            'LastName': row['LastName'],
103            'Address': row['Address'],
104            'Location': locations_map.get(row['CityID'], {})
105        }
106        documents_to_insert.append(doc)

107 elif collection_name == EMPLOYEES_COLLECTION:
108     locations_map = lookup_maps.get('locations', {})
109     for index, row in df.iterrows():
110         full_location = locations_map.get(row['CityID'], {})
111         employee_location = {
112             'CityID': full_location.get('CityID'),
113             'CityName': full_location.get('CityName')
114         }
115         doc = {
116             '_id': row['EmployeeID'],
117             'FirstName': row['FirstName'],
118             'MiddleInitial': row.get('MiddleInitial'),
119             'LastName': row['LastName'],
120             'BirthDate': pd.to_datetime(row['BirthDate']),
121             'Gender': row['Gender'],
122             'HireDate': pd.to_datetime(row['HireDate']),
123             'Location': employee_location
124         }
125         documents_to_insert.append(doc)

126 elif collection_name == SALES_COLLECTION:
127     print("Učitavanje očišćenog sales fajla...")
128     df = pd.read_csv(csv_file)
129     df['SalesDate'] = pd.to_datetime(df['SalesDate'])

130     documents_to_insert = df.to_dict('records')

131 if documents_to_insert:
132     if collection_name == SALES_COLLECTION:
133         print(f"Učitavanje {len(documents_to_insert)} dokumenta u '{collection_name}'...")
134         db[collection_name].insert_many(documents_to_insert)
135         print("Uspešno završeno.")
136     else:
137         print(f"Nema dokumenata za ubacivanje u '{collection_name}'.")
```

```
138
139
140 if documents_to_insert:
141     print(f"Ubacivanje {len(documents_to_insert)} dokumenata u '{collection_name}'...")
142     db[collection_name].insert_many(documents_to_insert)
143     print("Uspešno završeno.")
144 else:
145     print(f"Nema dokumenata za ubacivanje u '{collection_name}'.")
```

```
146
147
148 def main():
149     client = connect_to_mongo(MONGO_URI)
150     if not client:
151         return
152
153     db = client[DB_NAME]
154
155     categories_map, locations_map = preprocess_lookup_data(DATA_PATH)
156     if categories_map is None:
157         return
158
159     lookup_maps = {
160         'categories': categories_map,
161         'locations': locations_map
162     }
163
164     import_collection(db, PRODUCTS_COLLECTION, os.path.join(DATA_PATH, 'products.csv'), lookup_maps)
165     import_collection(db, CUSTOMERS_COLLECTION, os.path.join(DATA_PATH, 'customers.csv'), lookup_maps)
166     import_collection(db, EMPLOYEES_COLLECTION, os.path.join(DATA_PATH, 'employees.csv'), lookup_maps)
167     import_collection(db, SALES_COLLECTION, os.path.join(DATA_PATH, 'sales_cleaned.csv'))
168
169     print("\nSvi podaci su uspešno importovani u MongoDB bazu!")
170     client.close()
171
172
173 if __name__ == "__main__":
174     main()
```

Primeri agregacija: Inicijalna Šema

- Koji su ukupni prihodi (TotalPrice) i prosečni popusti (Discount) za svaku kategoriju proizvoda po mesecima? Identifikovati najprofitabilnije kategorije i kategorije sa najvećim prosečnim popustom.

```
db.getCollection("sales").aggregate([
  {
    $lookup: {
      from: 'products',
      localField: 'ProductID',
      foreignField: '_id',
      as: 'productInfo'
    }
  },
  {
    $unwind: '$productInfo'
  },
  {
    $group: {
      _id: {
        year: { $year: '$SalesDate' },
        month: { $month: '$SalesDate' },
        category: '$productInfo.Category.CategoryName'
      },
      totalRevenue: {
        $sum: '$TotalPrice'
      },
      averageDiscount: {
        $avg: '$Discount'
      }
    }
  },
  {
    $sort: {
      '_id.year': 1,
      '_id.month': 1,
      'totalRevenue': -1
    }
  },
  {
    $project: [
      '_id: 0',
      'year: "$_id.year"',
      'month: "$_id.month"',
      'category: "$_id.category"',
      'totalRevenue: { $round: ['$totalRevenue', 2] }',
      'averageDiscount: { $round: ['$averageDiscount', 2] }'
    ]
  }
])
```

sales > month					
year	month	category	totalRevenue	averageDiscount	
2018	1	Confections	132815151.2	0.03	
2018	1	Meat	117126751.91	0.03	
2018	1	Poultry	104779714.13	0.03	
2018	1	Cereals	101202038.15	0.03	
2018	1	Snails	88950230.11	0.03	
2018	1	Produce	87453213.52	0.03	
2018	1	Beverages	87044911.04	0.03	
2018	1	Dairy	84041017.34	0.03	
2018	1	Seafood	78656103.25	0.03	
2018	1	Grain	77379043.21	0.03	
2018	1	Shell fish	71287780.01	0.03	
2018	2	Confections	119023000.67	0.03	
2018	2	Meat	105350957.86	0.03	
2018	2	Poultry	94272655.42	0.03	
2018	2	Cereals	91660786.12	0.03	
2018	2	Snails	79984140.07	0.03	
2018	2	Produce	78989345.79	0.03	
2018	2	Beverages	78851002.5	0.03	

Vreme izvršavanja: 126s

Primeri agregacija: Inicijalna šema

- Postoji li korelacija između Class (klase proizvoda) i Resistant (otpornosti proizvoda) atributa i njihove ukupne prodaje (TotalPrice)? Koji Class/Resistant kombinacije ostvaruju najveći prihod?

```
db.getCollection("sales").aggregate[]

{
  $lookup: {
    from: 'products',
    localField: 'ProductID',
    foreignField: '_id',
    as: 'productInfo'
  },
  {
    $unwind: '$productInfo'
  },
  {
    $group: {
      _id: {
        productClass: '$productInfo.Class',
        productResistance: '$productInfo.Resistant'
      },
      totalRevenue: {
        $sum: '$TotalPrice'
      },
      totalQuantitySold: {
        $sum: '$Quantity'
      }
    }
  },
  {
    $sort: {
      'totalRevenue': -1
    }
  },
  {
    $project: {
      _id: 0,
      productCombination: {
        class: '$_id.productClass',
        resistance: '$_id.productResistance'
      },
      totalRevenue: { $round: ['$totalRevenue', 2] },
      totalQuantitySold: 1
    }
  }
}
```

sales		
totalQuantitySold	productCombination	totalRevenue
10578443	{ 2 fields }	568385602.61
10406124	{ 2 fields }	528162483.18
10596247	{ 2 fields }	514432090.33
9808944	{ 2 fields }	503401339.43
9241481	{ 2 fields }	482217598.13
9635073	{ 2 fields }	469084134.83
9802321	{ 2 fields }	431140897.65
7329458	{ 2 fields }	411187019.91
9605456	{ 2 fields }	381230916.7

sales > productCombination > class	resistance
High	Durable
Low	Durable
Medium	Durable
Medium	Unknown
Low	Weak
High	Weak
Low	Unknown
High	Unknown
Medium	Weak

Primeri agregacija: Inicijalna šema

- Izracunati za svaki grad i svaku kategoriju koliko se proizvoda iz date kategorije prodalo po gradu. Koja kategorija proizvoda je najprodavanija?

```
db.getCollection("sales").aggregate([
  {
    $lookup: {
      from: 'products',
      localField: 'ProductID',
      foreignField: '_id',
      as: 'productInfo'
    }
  },
  { $unwind: '$productInfo' },
  {
    $lookup: {
      from: 'customers',
      localField: 'CustomerID',
      foreignField: '_id',
      as: 'customerInfo'
    }
  },
  { $unwind: '$customerInfo' },
  {
    $group: {
      _id: {
        city: '$customerInfo.Location.CityName',
        category: '$productInfo.Category.CategoryName'
      },
      totalItemsSold: { $sum: '$Quantity' }
    }
  },
  {
    $sort: {
      '_id.city': 1,
      'totalItemsSold': -1
    }
  },
  {
    $project: {
      _id: 0,
      city: '$_id.city',
      category: '$_id.category',
      totalItemsSold: 1
    }
  }
])
```

Vreme izvšavanja: 343s

sales > city		
totalItemsSold	city	category
111650	Akron	Confections
96971	Akron	Meat
90490	Akron	Poultry
87652	Akron	Cereals
81858	Akron	Produce
72936	Akron	Snails
72121	Akron	Beverages
69935	Akron	Seafood
68882	Akron	Shell fish
68800	Akron	Dairy
54592	Akron	Grain
116982	Albuquerque	Confections
103186	Albuquerque	Meat
97226	Albuquerque	Poultry
92256	Albuquerque	Cereals
87116	Albuquerque	Produce
77448	Albuquerque	Beverages
75594	Albuquerque	Snails
75010	Albuquerque	Seafood
74365	Albuquerque	Dairy
74155	Albuquerque	Shell fish
56579	Albuquerque	Grain
119210	Anaheim	Confections
101066	Anaheim	Meat
96647	Anaheim	Poultry
92380	Anaheim	Cereals
87957	Anaheim	Produce
1 document selected		

Primeri agregacija: Inicijalna Šema

- Koliko se proizvoda označenih kao IsAllergic prodaje u svakom gradu, po Quantity i TotalPrice? Koji gradovi imaju najveću prodaju alergijskih proizvoda?

```
db.getCollection("sales").aggregate([
  {
    $group: {
      _id: '$TransactionNumber',
      transactionValue: { $sum: '$TotalPrice' },
      itemsInTransaction: { $sum: 1 },
      customerID: { $first: '$CustomerID' }
    }
  },
  {
    $lookup: {
      from: 'customers',
      localField: 'customerID',
      foreignField: '_id',
      as: 'customerInfo'
    }
  },
  {
    $match: { 'customerInfo': { $ne: [] } }
  },
  { $unwind: '$customerInfo' },
  {
    $group: {
      _id: '$customerInfo.Location.CityName',
      avgTransactionValue: { $avg: '$transactionValue' },
      avgItemsPerTransaction: { $avg: '$itemsInTransaction' },
      totalTransactions: { $sum: 1 }
    }
  },
  {
    $sort: {
      'avgTransactionValue': -1
    }
  },
  {
    $project: {
      _id: 0,
      city: '$_id',
      avgTransactionValue: { $round: ['$avgTransactionValue', 2] },
      avgItemsPerTransaction: { $round: ['$avgItemsPerTransaction', 2] },
      totalTransactions: 1
    }
  }
])
```

sales > totalQuantityOfAllergens		
totalQuantityOfAllergens	city	totalRevenueFromAllergens
342179	Tucson	17746144.27
337159	Jackson	17702270.45
335205	Sacramento	17405623.93
332159	Fort Wayne	17336577.21
328895	San Antonio	17233594.89
330199	Indianapolis	17187931.63
333492	Columbus	17138997.81
330186	Charlotte	17069814.36
323004	Rochester	17028509.89
325305	Phoenix	16974941.22
323167	Spokane	16953668.98
322088	Newark	16942599.44
322264	St. Petersburg	16897381.62
324045	Yonkers	16895168.83
325354	Greensboro	16894196.18
322403	New York	16890000.09
323748	Baton Rouge	16870530.47
322256	Las Vegas	16861100.71

Vreme izvršavanja: 401s

Optimizovana logička říšma

Kolekcija denormalized_sales

```
• SalesID  
• SalesDate  
• Quantity  
• Discount  
• TotalPrice  
• TransactionNumber  
• Product: {  
    ProductID,  
    ProductName,  
    CategoryName,  
    IsAllergic,  
    Class,  
    Resistant  
},  
• Customer: {  
    CustomerID,  
    CityName  
},  
• SalesPerson: {  
    SalesPersonID  
}
```

Kolekcija employees

```
• EmployeeID  
• FirstName  
• LastName  
• MiddleInitial  
• BirthDate  
• Gender  
• HireDate  
• Location: { CityID, CityName }
```

Skripta za kreiranje i popunjavanje optimizovanih kolekcija u Python-u

```
import pandas as pd
from pymongo import MongoClient
from pymongo.errors import ConnectionFailure
import os

MONGO_URI = "mongodb://student:ftn@localhost:27017/?authSource=admin"
DB_NAME = "grocery_sales_db"
DATA_PATH = "."

PRODUCTS_COLLECTION = "products"
CUSTOMERS_COLLECTION = "customers"
EMPLOYEES_COLLECTION = "employees"
SALES_COLLECTION = "sales"
SALES_DENORMALIZED_COLLECTION = "sales_denormalized"

def connect_to_mongo(uri):
    """Povezuje se na MongoDB i proverava konekciju."""
    print("Povezivanje na MongoDB...")
    try:
        client = MongoClient(uri)
        client.admin.command('ping')
        print("MongoDB konekcija uspešna.")
        return client
    except ConnectionFailure as e:
        print(f"Greška pri povezivanju na MongoDB: {e}")
        return None

def prepare_data_maps(path):
    """
    Priprema sve potrebne podatke iz CSV fajlova i pretvara ih u rečnike (mapu)
    radi brzog pristupa tokom denormalizacije.
    """
    print("\n--- Priprema lookup mapa za denormalizaciju ---")
    try:
        categories_df = pd.read_csv(os.path.join(path, 'categories.csv'))
        cities_df = pd.read_csv(os.path.join(path, 'cities.csv'))
        countries_df = pd.read_csv(os.path.join(path, 'countries.csv'))
        products_df = pd.read_csv(os.path.join(path, 'products.csv'))
        customers_df = pd.read_csv(os.path.join(path, 'customers.csv'))

        locations_df = pd.merge(cities_df, countries_df, on='CountryID', how='left')
        locations_map = {
            row['CityID']: {
                'CityID': row['CityID'],
                'CityName': row['CityName'],
                'Zipcode': row['Zipcode'],
                'Country': {
                    'CountryID': row['CountryID'],
                    'CountryName': row['CountryName'],
                    'CountryCode': row['CountryCode']
                }
            }
            for index, row in locations_df.iterrows()
        }
        print("Mapa lokacija kreirana.")

        categories_map = {
            row['CategoryID']: {'CategoryName': row['CategoryName']}
            for index, row in categories_df.iterrows()
        }
        print("Mapa kategorija kreirana.")

        products_map = {
            row['ProductID']: {
                'ProductName': row['ProductName'],
                'CategoryName': categories_map.get(row['CategoryID'], {}).get('CategoryName'),
                'IsAllergic': row['IsAllergic'],
                'Class': row['Class'],
                'Resistant': row['Resistant']
            }
            for index, row in products_df.iterrows()
        }
        print("Mapa proizvoda kreirana.")

        customers_map = {
            row['CustomerID']: {
                'CustomerID': row['CustomerID'],
                'CityName': locations_map.get(row['CityID'], {}).get('CityName')
            }
            for index, row in customers_df.iterrows()
        }
        print("Mapa kupaca kreirana.")

        print("Sve lookup mape su uspešno pripremljene.")

        return {
            'categories': categories_map,
            'locations': locations_map,
            'products': products_map,
            'customers': customers_map
        }
    except FileNotFoundError as e:
        print(f"Greška: CSV fajl nije pronađen. Proverite putanju. {e}")
        return None
    
```

```
def prepare_data_maps(path):
    'CountryID': row['CountryID'],
    'CountryName': row['CountryName'],
    'CountryCode': row['CountryCode']

    }
    for index, row in locations_df.iterrows():
}
print("Mapa lokacija kreirana.")

categories_map = {
    row['CategoryID']: {'CategoryName': row['CategoryName']}
    for index, row in categories_df.iterrows()
}
print("Mapa kategorija kreirana.")

products_map = {
    row['ProductID']: {
        'ProductName': row['ProductName'],
        'CategoryName': categories_map.get(row['CategoryID'], {}).get('CategoryName'),
        'IsAllergic': row['IsAllergic'],
        'Class': row['Class'],
        'Resistant': row['Resistant']
    }
    for index, row in products_df.iterrows()
}
print("Mapa proizvoda kreirana.")

customers_map = {
    row['CustomerID']: {
        'CustomerID': row['CustomerID'],
        'CityName': locations_map.get(row['CityID'], {}).get('CityName')
    }
    for index, row in customers_df.iterrows()
}
print("Mapa kupaca kreirana.")

print("Sve lookup mape su uspešno pripremljene.")

return {
    'categories': categories_map,
    'locations': locations_map,
    'products': products_map,
    'customers': customers_map
}

except FileNotFoundError as e:
    print(f"Greška: CSV fajl nije pronađen. Proverite putanju. {e}")
    return None
    
```

```
def import_collection(db, collection_name, csv_file, Lookup_maps={}):
    print(f"\n--- Obrada kolekcije: {collection_name} ---")
    try:
        df = pd.read_csv(csv_file)
    except FileNotFoundError:
        print(f"Fajl {csv_file} nije pronađen. Preskačem.")
        return

    print(f"Brisanje postojeće kolekcije '{collection_name}'...")
    db[collection_name].delete_many({})

    documents_to_insert = []

    if collection_name == PRODUCTS_COLLECTION:
        categories_map = lookup_maps.get('categories', {})
        for index, row in df.iterrows():
            documents_to_insert.append({
                '_id': row['ProductID'], 'ProductName': row['ProductName'], 'Price': row['Price'],
                'Class': row['Class'], 'ModifyDate': pd.to_datetime(row['ModifyDate']),
                'Resistant': row['Resistant'], 'IsAllergic': row['IsAllergic'],
                'VitalityDays': row['VitalityDays'],
                'Category': categories_map.get(row['CategoryID'], {})
            })

    elif collection_name == CUSTOMERS_COLLECTION:
        locations_map = lookup_maps.get('locations', {})
        for index, row in df.iterrows():
            documents_to_insert.append({
                '_id': row['CustomerID'], 'FirstName': row['FirstName'],
                'MiddleInitial': row.get('MiddleInitial'), 'LastName': row['LastName'],
                'Address': row['Address'], 'Location': locations_map.get(row['CityID'], {})
            })

    elif collection_name == EMPLOYEES_COLLECTION:
        locations_map = lookup_maps.get('locations', {})
        for index, row in df.iterrows():
            full_location = locations_map.get(row['CityID'], {})
            documents_to_insert.append({
                '_id': row['EmployeeID'], 'FirstName': row['FirstName'],
                'MiddleInitial': row.get('MiddleInitial'), 'LastName': row['LastName'],
                'BirthDate': pd.to_datetime(row['BirthDate']), 'Gender': row['Gender'],
                'HireDate': pd.to_datetime(row['HireDate']),
                'Location': {'CityID': full_location.get('CityID'), 'CityName': full_location.get('CityName')}
            })

    elif collection_name == SALES_COLLECTION:
        df['SalesDate'] = pd.to_datetime(df['SalesDate'])
        documents_to_insert = df.to_dict('records')
    
```

Skripta za kreiranje i popunjavanje optimizovanih kolekcija u Python-u

```
def import_collection(db, collection_name, csv_file, lookup_maps={}):
    elif collection_name == SALES_DENORMALIZED_COLLECTION:
        print("Priprema obogaćenih (denormalizovanih) sales dokumenata...")
        products_map = lookup_maps.get('products', {})
        customers_map = lookup_maps.get('customers', {})

        df['SalesDate'] = pd.to_datetime(df['SalesDate'])

        for index, row in df.iterrows():
            product_info = products_map.get(row['ProductID'], {})
            customer_info = customers_map.get(row['CustomerID'], {})

            doc = {
                'SalesID': row['SalesID'],
                'SalesDate': row['SalesDate'],
                'Quantity': row['Quantity'],
                'Discount': row['Discount'],
                'TotalPrice': row['TotalPrice'],
                'TransactionNumber': row['TransactionNumber'],
                'Product': {
                    'ProductID': row['ProductID'],
                    'ProductName': product_info.get('ProductName'),
                    'CategoryName': product_info.get('CategoryName'),
                    'IsAllergic': product_info.get('IsAllergic'),
                    'Class': product_info.get('Class'),
                    'Resistant': product_info.get('Resistant')
                },
                'Customer': {
                    'CustomerID': row['CustomerID'],
                    'CityName': customer_info.get('CityName')
                },
                'SalesPerson': {
                    'SalesPersonID': row['SalesPersonID']
                }
            }
            documents_to_insert.append(doc)

        if documents_to_insert:
            print(f"Ubacivanje {len(documents_to_insert)} dokumenata u '{collection_name}'...")
            db[collection_name].insert_many(documents_to_insert, ordered=False)
            print("Uspešno završeno.")
        else:
            print(f"Nema dokumenata za ubacivanje u '{collection_name}'.")

    else:
```

```
def main():
    client = connect_to_mongo(MONGO_URI)
    if not client:
        return

    db = client[DB_NAME]

    lookup_maps = prepare_data_maps(DATA_PATH)
    if not lookup_maps:
        print("Greška pri pripremi mapa. Prekidam izvršavanje.")
        return

    import_collection(db, PRODUCTS_COLLECTION, os.path.join(DATA_PATH, 'products.csv'), lookup_maps)
    import_collection(db, CUSTOMERS_COLLECTION, os.path.join(DATA_PATH, 'customers.csv'), lookup_maps)
    import_collection(db, EMPLOYEES_COLLECTION, os.path.join(DATA_PATH, 'employees.csv'), lookup_maps)
    import_collection(db, SALES_COLLECTION, os.path.join(DATA_PATH, 'sales_cleaned.csv'))

    import_collection(db, SALES_DENORMALIZED_COLLECTION, os.path.join(DATA_PATH, 'sales_cleaned.csv'), lookup_maps)

    print("\nSvi podaci su uspešno importovani u MongoDB bazu!")
    print(f"Kreirana je optimizovana kolekcija: '{SALES_DENORMALIZED_COLLECTION}'")
    client.close()

if __name__ == "__main__":
    main()
```

Primeri agregacija: Optimizovana šema

- Koji su ukupni prihodi (TotalPrice) i prosečni popusti (Discount) za svaku kategoriju proizvoda po mesecima? Identifikovati najprofitabilnije kategorije i kategorije sa najvećim prosečnim popustom.

```
db.getCollection("sales_denormalized").aggregate([
  {
    $group: {
      _id: {
        year: { $year: '$SalesDate' },
        month: { $month: '$SalesDate' },
        category: '$Product.CategoryName'
      },
      totalRevenue: { $sum: '$TotalPrice' },
      averageDiscount: { $avg: '$Discount' }
    }
  },
  {
    $sort: {
      '_id.year': 1,
      '_id.month': 1,
      'totalRevenue': -1
    }
  },
  {
    $project: {
      _id: 0,
      year: '$_id.year',
      month: '$_id.month',
      category: '$_id.category',
      totalRevenue: { $round: ['$totalRevenue', 2] },
      averageDiscount: { $round: ['$averageDiscount', 2] }
    }
  }
]).explain("executionStats")
```

Vreme izvršavanja: **6.12s**

Primeri agregacija: Optimizovana šema

- Postoji li korelacija između Class (klase proizvoda) i Resistant (otpornosti proizvoda) atributa i njihove ukupne prodaje (TotalPrice)? Koji Class/Resistant kombinacije ostvaruju najveći prihod?

```
db.getCollection("sales_denormalized").aggregate([
  {
    $group: {
      _id: {
        productClass: '$Product.Class',
        productResistance: '$Product.Resistant'
      },
      totalRevenue: { $sum: '$TotalPrice' },
      totalQuantitySold: { $sum: '$Quantity' }
    }
  },
  {
    $sort: { 'totalRevenue': -1 }
  },
  {
    $project: {
      _id: 0,
      productCombination: {
        class: '$_id.productClass',
        resistance: '$_id.productResistance'
      },
      totalRevenue: { $round: ['$totalRevenue', 2] },
      totalQuantitySold: 1
    }
  }
])
```

Vreme izvršavanja: 6.48s

Primeri agregacija: Inicijalna [✓]šema

- Izracunati za svaki grad i svaku kategoriju koliko se proizvoda iz date kategorije prodalo po gradu. Koja kategorija proizvoda je najprodavanija?

```
db.getCollection("sales_denormalized").aggregate([
  {
    $group: {
      _id: {
        city: '$Customer.CityName',
        category: '$Product.CategoryName'
      },
      totalItemsSold: { $sum: '$Quantity' },
    }
  },
  {
    $sort: {
      '_id.city': 1,
      'totalItemsSold': -1
    }
  },
  {
    $project: {
      _id: 0,
      city: '$_id.city',
      category: '$_id.category',
      totalItemsSold: 1,
    }
  }
])
```

Vreme izvršavanja: 4.12s

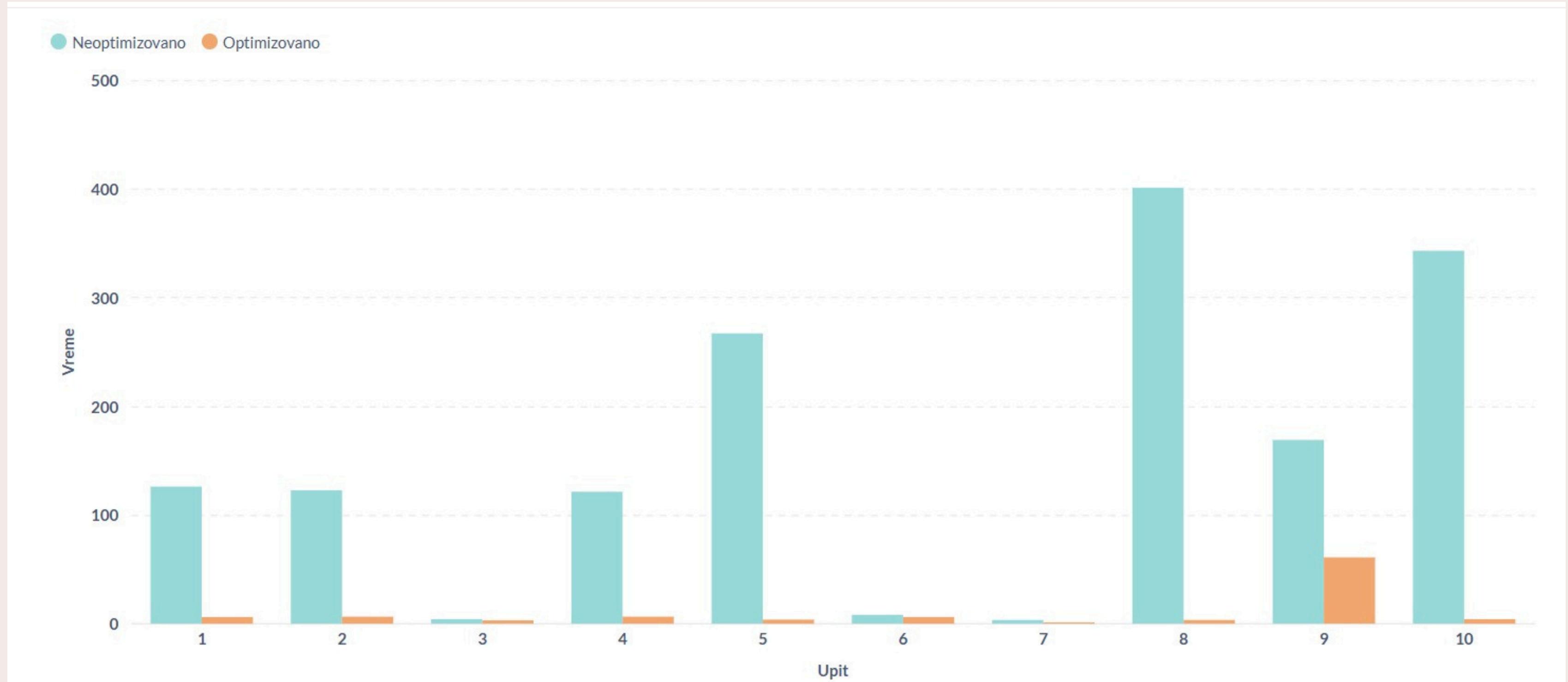
Primeri agregacija: Inicijalna [✓]šema

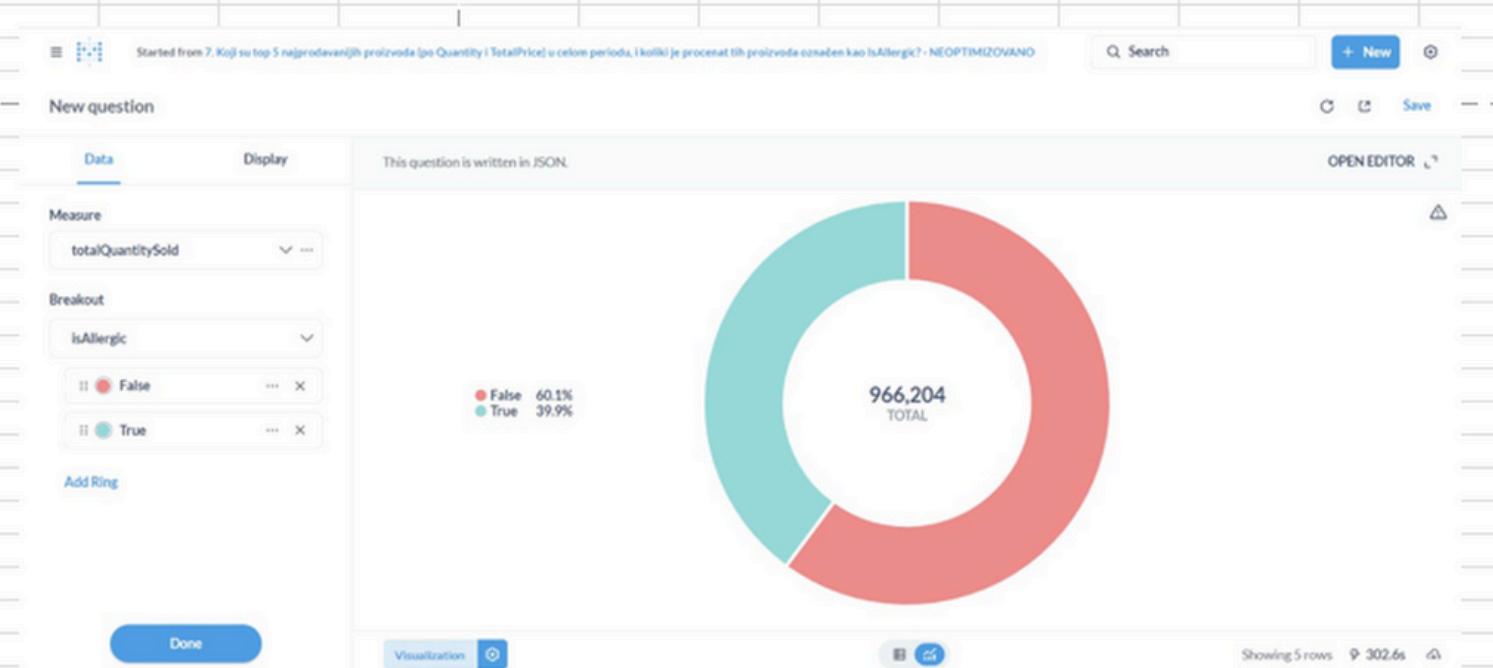
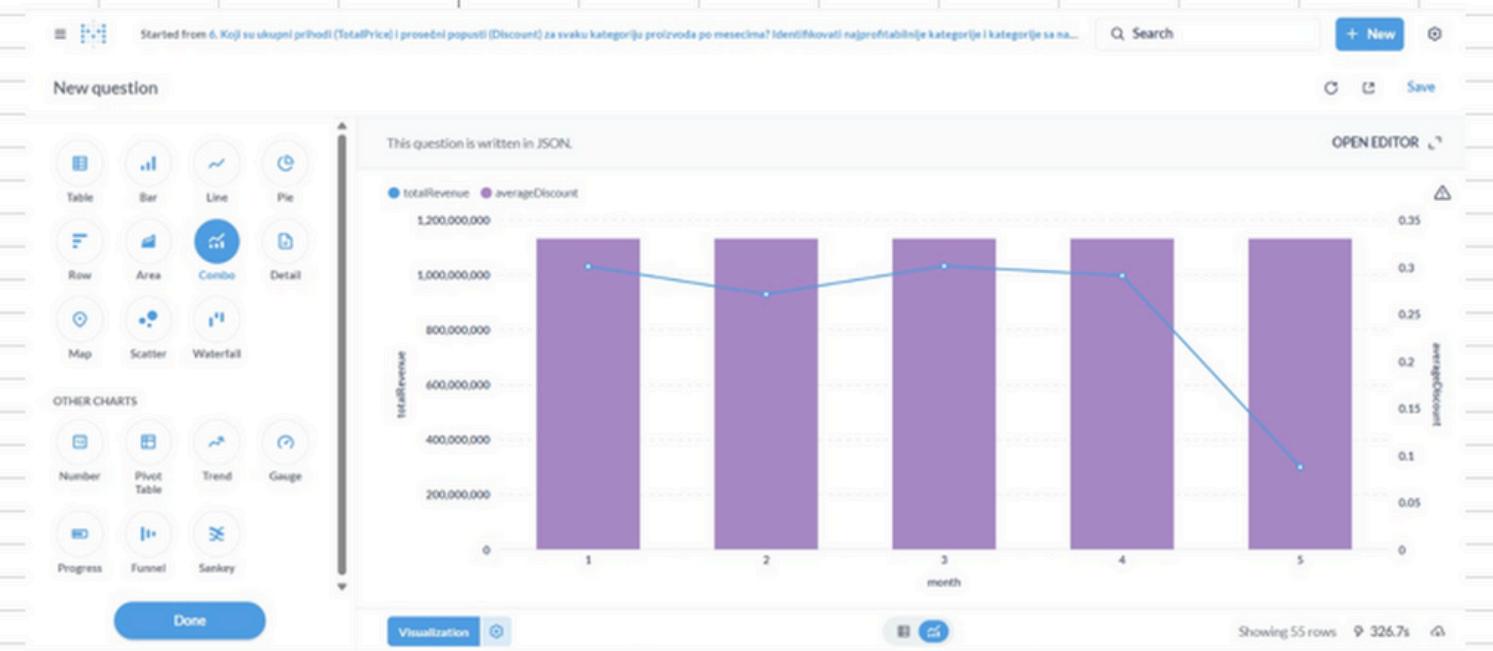
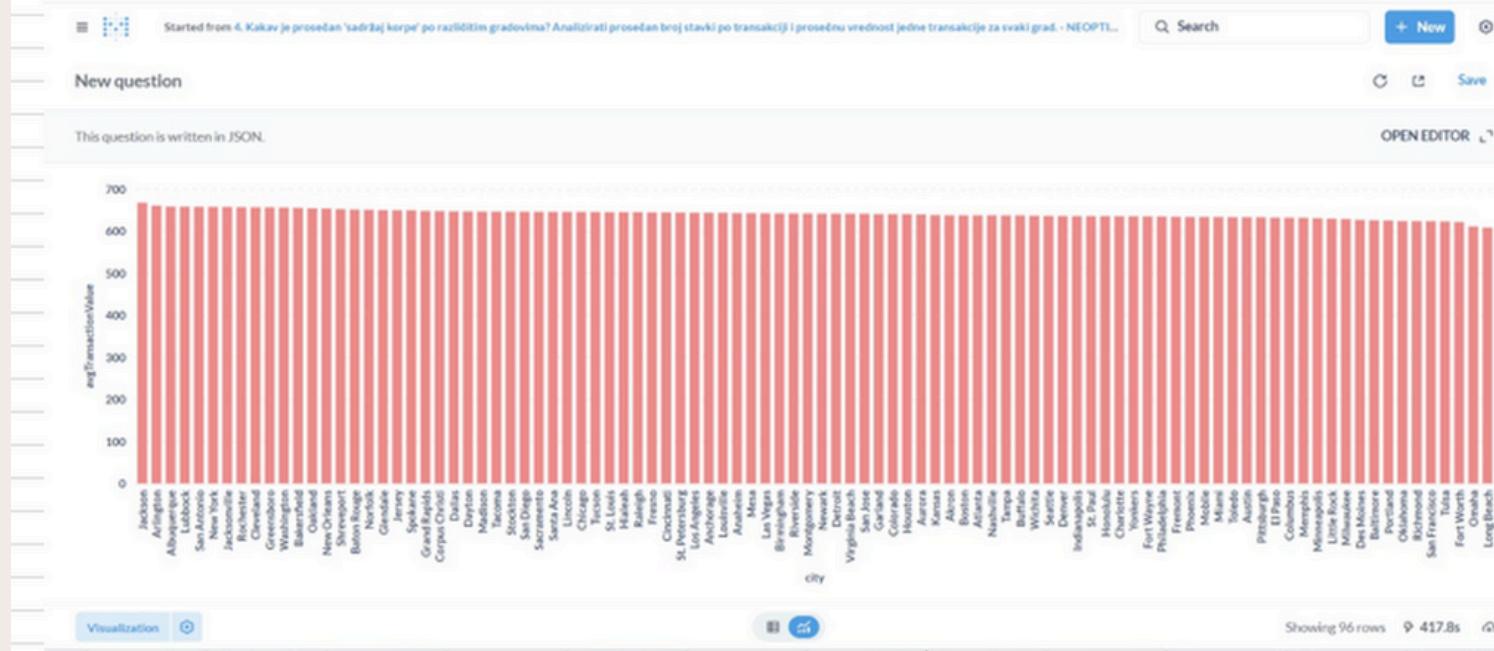
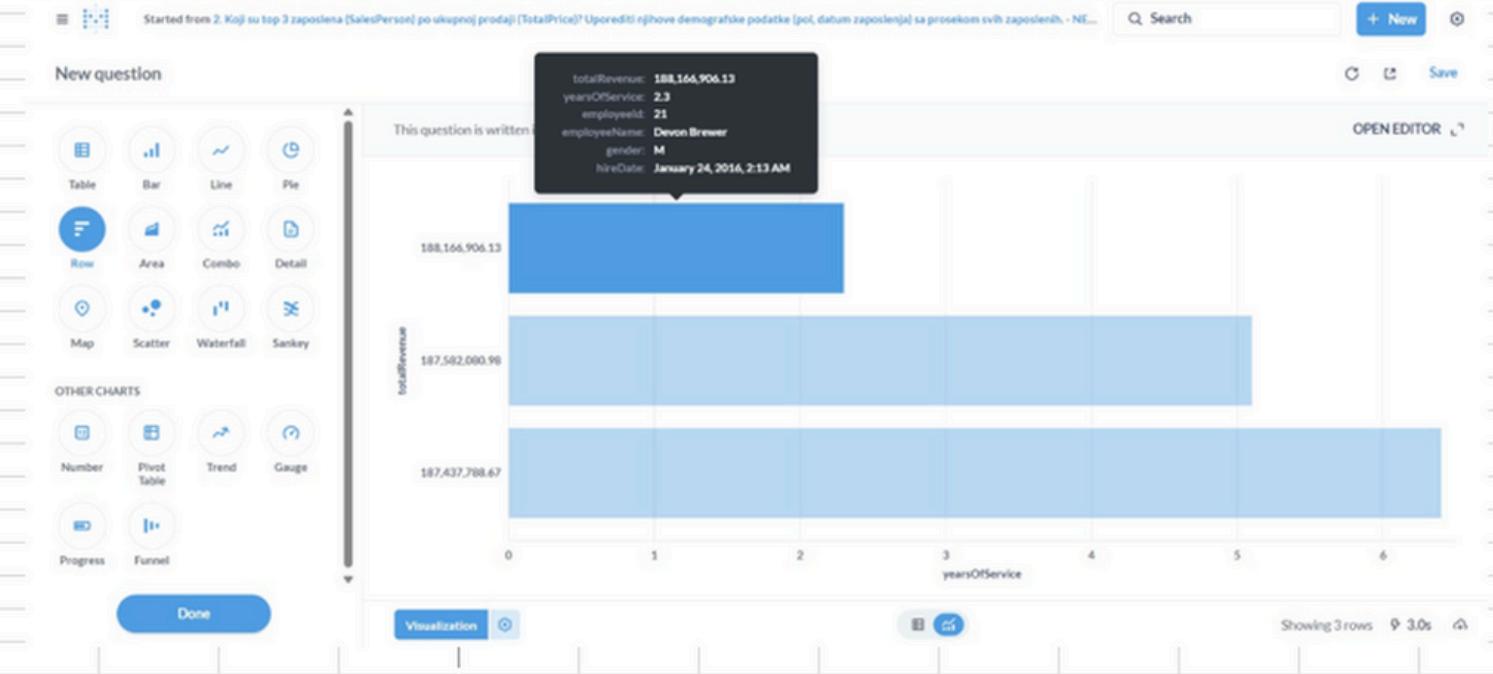
- Koliko se proizvoda označenih kao IsAllergic prodaje u svakom gradu, po Quantity i TotalPrice? Koji gradovi imaju najveću prodaju alergijskih proizvoda?

```
db.getCollection("sales_denormalized").aggregate([
  {
    $match: {
      'Product.IsAllergic': 'True'
    }
  },
  {
    $group: {
      _id: '$Customer.CityName',
      totalRevenueFromAllergens: { $sum: '$TotalPrice' },
      totalQuantityOfAllergens: { $sum: '$Quantity' }
    }
  },
  {
    $sort: { 'totalRevenueFromAllergens': -1 }
  },
  {
    $project: {
      _id: 0,
      city: '_id',
      totalRevenueFromAllergens: { $round: ['$totalRevenueFromAllergens', 2] },
      totalQuantityOfAllergens: 1
    }
  }
])
```

Vreme izvršavanja: 3.37s

Poređenje performansi





Milica Bošnjak INI8/2021

ČLANOVI TIMA

Tijana Petrović INIO/2021