# Analiza Prodaje

Lucian Tin Udovičić
Broj indeksa : 0165073866
Mentor : doc. dr. sc. Goran Oreški
Sustavi poslovne inteligencije
Sveučilište Jurja Dobrile u Puli
20.5.2021. godine

# Sadržaj


**1 Uvod** **1**

**2 Podaci** **2**
2.1 Izvor . . . 2
2.2 Analiza . . . 2

**3 Transakcijski Model Podataka** **3**
3.1 ER Dijagram . . . 3
3.2 ER Model . . . 4
3.3 Punjenje Baze Podataka . . . 5
3.3.1 CSV i SQL . . . 5
3.3.2 Python . . . 5
3.3.3 Docker . . . 7

**4 Dimenzijski Model Podataka** **8**

**5 ETL Procesi** **9**
5.1 Pentaho ”Job” . . . 9
5.2 Transformacije . . . 11
5.2.1 Dimenzija Vrijeme . . . 11
5.2.2 Dimenzija Dostava . . . 11
5.2.3 Dimenzija Proizvod . . . 12
5.2.4 Dimenzija Lokacija . . . 12
5.2.5 Tablice Činjenica . . . 13
5.2.6 Spajanje Tablica Činjenica . . . 14

**6 Vizualizacija Podataka** **15**
6.1 Metabase . . . 15
6.1.1 Dizajn Upita . . . 15
6.1.2 Nadzorna ploča . . . 16
6.2 Vizualizacije . . . 16
6.2.1 Ukupna cijena narudžbe i profit u vremenskom intervalu . . . 16
6.2.2 Dobit od narudžbe po tržištu, po segmentu kupaca . . . 17
6.2.3 Prosječna razlika u zakazanim danima dostave i stvarnim danima dostave po načinu dostave . . . 18
6.2.4 Dobit po narudžbi po državi . . . 18
6.2.5 Ukupna količina prodanih proizvoda . . . 19

**7 Zakljucak** **20**

# 1 Uvod

Kako bismo koristili podatke za potporu odlučivanju moramo prvo stvoriti skladište podataka tako što integriramo podatke iz različitih izvora u jedinstvenu bazu [3]. U poglavlju 3 podaci se analiziraju i transformiraju za punjenje transakcijske baze podataka s podacima iz CSV datoteke. Rad prolazi kroz dizajn star sheme dimenzijskog modela i punjenje u poglavljima 3 i 4. Dimenzijski model se sastoji od 4 sporo mijenjajuće dimenzije i jedne usklađene dimenzije za spajanje dvije tablice činjenica. Dizajniran je tako da prati prodaju proizvoda te sadrži dimenzije koje opisuju taj poslovni proces. Tablice činjenica također sadrže atributi kojima možemo pratiti promjenu narudžbi kroz vrijeme. Vremenska dimenzija se sastoji od dva datuma, jedna opisuje datum stvaranja narudžbe dok drugi datum opisuje datum kada je narudžba dostavljena kupcu.

U poglavlju 5 proći ćemo kroz punjenje dimenzijskog modela uz pomoć Pentaho programa. Dimenzijski model se puni uz pomoć ETL procesa stvorenih u Pentaho-u, za punjenje se također koristi Pentaho "Job" kako bi se dimenzije mogle paralelno ažurirati. Također će se opisati SQL upit kojim se dohvaćaju aktivni zapisi u dimenzijskom modelu. U poglavlju 6 opisat će se kako se uz pomoć aplikacije Metabase stvaraju upiti za vizualizaciju podataka iz dimenzijskog modela koji su važni za donošenje odluka.

Za izradu projekta korišteno je više tehnologija, MySQL baza podataka, Docker Compose [2] za pokretanje baze podataka i alata za vizualizaciju, Pentaho za punjenje dimenzijskog modela i Metabase [5] alat za vizualizaciju. Transakcijska baza se puni uz pomoć Python skripte nakon što se MySQL container inicijalizira, nakon toga napunim dimenzije i tablice činjenica uz pomoć Pentaha. Projekt, zajedno s programskim kodom i podacima, je dostupan na GitHub-u [4].

# 2 Podaci

## 2.1 Izvor

Skup podataka je preuzet sa Kaggle-a [1], sastoji se od dvije CSV datoteke, jedna sadrži opise atributa dok druga sadrži denormalizirane podatke. Podaci opisuju poslovanje jedne tvrtke koja isporučuje svoje proizvode iz više manjih poslovnica.

## 2.2 Analiza

Skup podataka je analiziran uz pomoć Python Pandas biblioteke, provjerio sam dali ima nepostojećih vrijednosti, duplikata i raznolikost podataka. Također mi je bi važan format datuma kako bih ga uspješno unio u bazu podataka. Osim funkcija iz biblioteke Pandas, koristio sam i Matplotlib za vizualizaciju kategoričkih podataka kako bih si bolje predočio raznolikost tih podataka. Analizirani su tipovi podataka, format datuma, duplikati, nepostojeće vrijednosti i raznolikost podataka.

| | Tipovi | Jedinstvene | Nedostajuće |
|---|---|---|---|
| **Type** | object | 4 | 0 |
| **Days for shipping (real)** | int64 | 7 | 0 |
| **Days for shipment (scheduled)** | int64 | 4 | 0 |
| **Benefit per order** | float64 | 21998 | 0 |
| **Sales per customer** | float64 | 2927 | 0 |
| **Delivery Status** | object | 4 | 0 |
| **Late_delivery_risk** | int64 | 2 | 0 |
| **Category Id** | int64 | 51 | 0 |
| **Category Name** | object | 50 | 0 |
| **Customer City** | object | 563 | 0 |
| **Customer Country** | object | 2 | 0 |
| **Customer Email** | object | 1 | 0 |
| **Customer Fname** | object | 782 | 0 |
| **Customer Id** | int64 | 20652 | 0 |
| **Customer Lname** | object | 1109 | 8 |
| **Customer Password** | object | 1 | 0 |
| **Customer Segment** | object | 3 | 0 |
| **Customer State** | object | 46 | 0 |
| **Customer Street** | object | 7458 | 0 |
| **Customer Zipcode** | float64 | 995 | 3 |
| **Department Id** | int64 | 11 | 0 |
| **Department Name** | object | 11 | 0 |

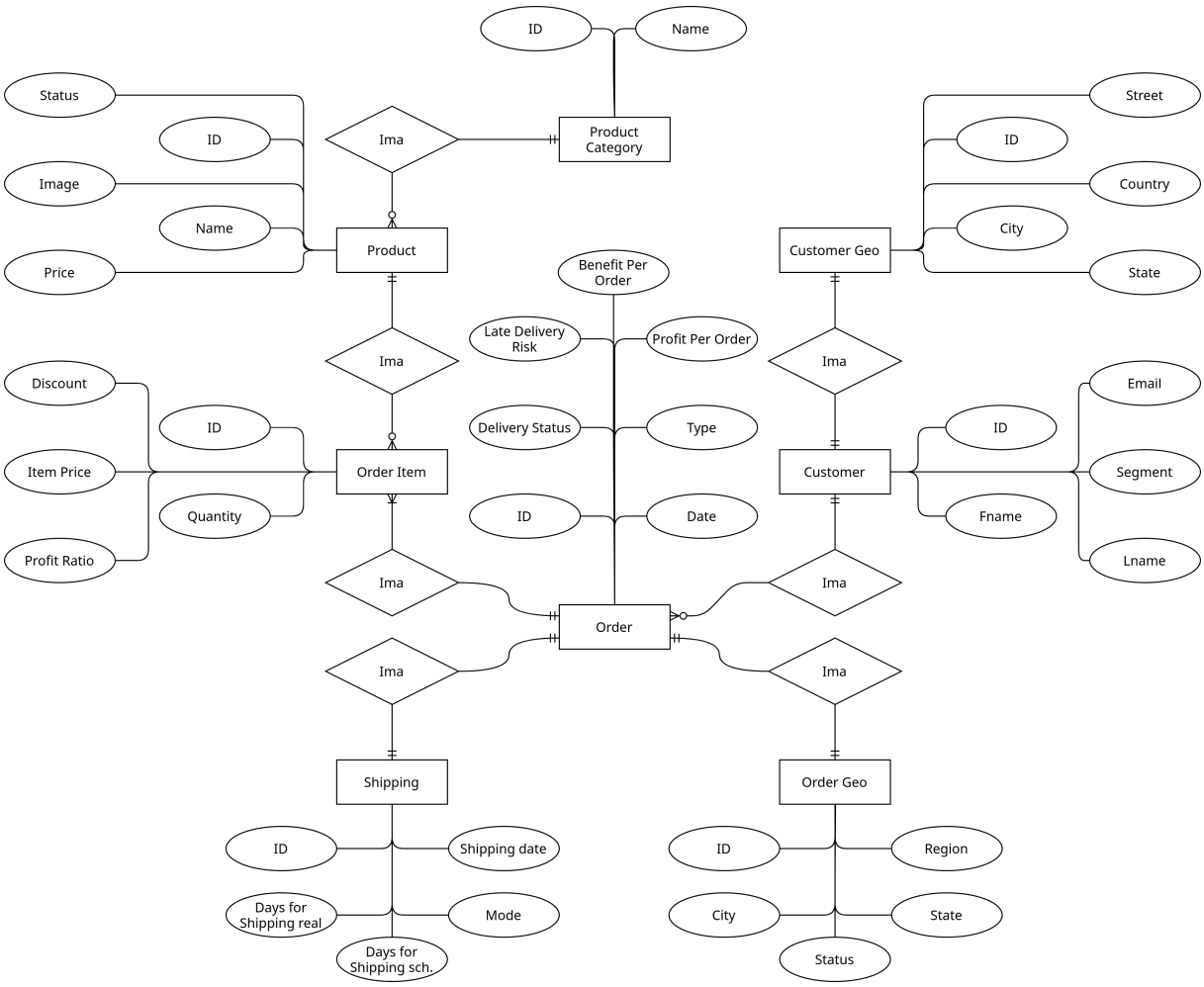
Slika 2.1: Analiza podataka

# 3 Transakcijski Model Podataka

Kako bi mogli napuniti skladište podataka prvo moramo stvoriti transakcijski model gdje se pohranjuju rezultati poslovnih procesa poput prodaja proizvoda, isporuka robe, itd. Nije opisano kako izgleda transakcijska baza podataka za preuzete podatke stoga sam stvorio svoju shemu. Odlučio sam iskoristiti što više atributa kako bih što vjerodostojnije izveo projekt.

## 3.1 ER Dijagram

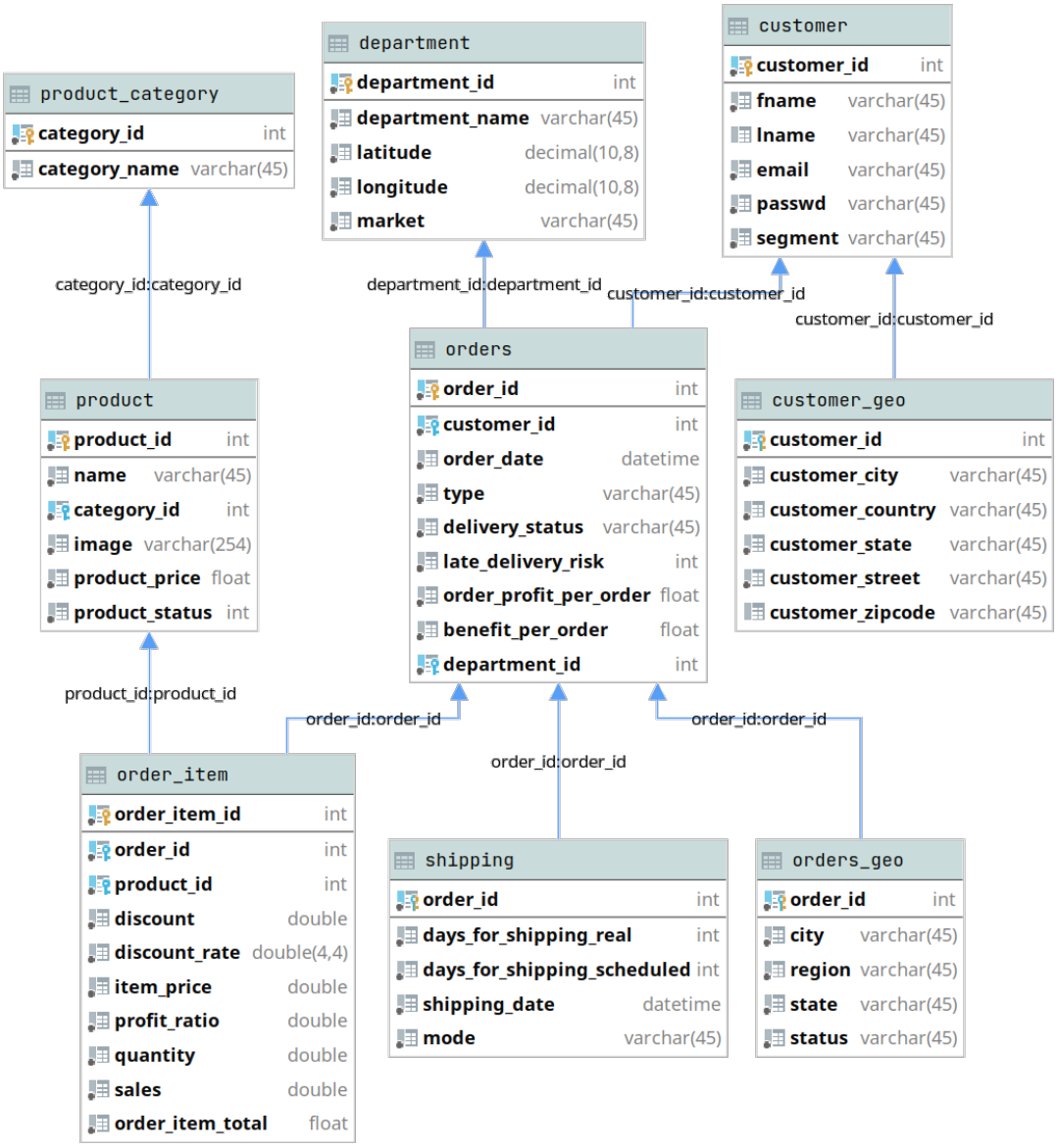
ER dijagram prikazuje konceptualni model baze podataka. Neki skupovi atributa koji opisuju isti entitet su rastavljeni na dva entiteta.




Slika 3.1: ER dijagram

## 3.2 ER Model

ER model se sastoji od 9 tablica koje opisuju poslovanje tvrtke koja prodaje neke proizvode. Svaka narudžba ima više stavki, poslovnicu, tablicu koja opisuje dostavu, kupca i njegovu geografsku lokaciju te svaka stavka sadrži opise o cijeni, količini, ukupnoj cijeni itd.




Slika 3.2: ER model

## 3.3 Punjenje Baze Podataka

### 3.3.1 CSV i SQL

ER model se sastoji od ER model se sastoji od ER model se sastoji od ER model se sastoji od ER model se sastoji od ER model se sastoji od ER model se sastoji od ER model se sastoji od ER model se sastoji od ER model se sastoji od

```
class Dataset:
    def __init__(self, path, delimiter, encoding):
        self.dataset = pd.read_csv(path, delimiter=delimiter, encoding=encoding)
        self.dataset.columns = dataset_db_col_names
        self.dataset = self.dataset.set_index('order_item_id', drop=False)

        self.dataset["shipping_date"] = self.change_date_format(self.dataset, "shipping_date")
        self.dataset["order_date"] = self.change_date_format(self.dataset, "order_date")

        self.tables = {}
        self.table_names = table_schema.keys()

        for table_name in self.table_names:
            self.tables[table_name] = self.dataset[table_schema[table_name]]

        self.tables["table_customer_geo"] = self.tables["table_customer_geo"].groupby("customer_id").first()
        self.tables["table_customer"] = self.tables["table_customer"].groupby("customer_id").first()
        self.tables["table_product"] = self.tables["table_product"].groupby("product_id").first()
        self.tables["table_department"] = self.tables["table_department"].groupby("department_id").first()
        self.tables["table_orders"] = self.tables["table_orders"].groupby("order_id").first()
        self.tables["table_orders_geo"] = self.tables["table_orders_geo"].groupby("order_id").first()

        self.tables["table_shipping"] = self.tables["table_shipping"].groupby("order_id").first()
        self.tables["table_product_category"] = self.tables["table_product_category"].drop_duplicates().set_index
        self.tables["table_order_item"] = self.tables["table_order_item"].set_index("order_item_id")

    def get_tables(self):
        return self.tables, table_dtypes, table_insertion_order

    def print_pandas_tables(self):
        for table_name in self.table_names:
            display(HTML(self.tables[table_name][:3].to_html()))

    def DD_MM_YYYY_to_YYYY_MM_DD(self, strng):
        strng = strng.split(" ")
        time = strng[1]
        date = strng[0].split('/')
        return date[2] + '-' + date[0] + '-' + date[1] + ' ' + time

    def change_date_format(self, df, col):
        return df[col].apply(lambda date: self.DD_MM_YYYY_to_YYYY_MM_DD(date + ":00"))
```

Slika 3.3: Python klasa za formatiranje podataka iz CSV datoteke

### 3.3.2 Python

Transakcijski model podataka se puni uz pomoću Python skripte koja osim punjenja tablica također stvara novog korisnika i sheme za transakcijski i dimenzijski model. Za spajanje s bazom iz Python-a je korištena biblioteka SqlAlchemy a za formatiranje podataka Pandas. Potrebno je bilo samo formatirati datum jer su ostali podaci bili dovoljno dobri za korištenje.

```
dataset_db_col_names = [
    "type", "days_for_shipping_real", "days_for_shipping_scheduled", "benefit_per_order", "sales_per_customer", "delivery_status",
    "late_delivery_risk", "category_id", "category_name", "customer_city", "customer_country", "email", "fname", "customer_id", "lname",
    "passwd", "segment", "customer_state", "customer_street", "customer_zipcode", "department_id", "department_name", "latitude", "longitude", "market",
    "city", "country", "order_customer_id", "order_date", "order_id", "order_item_cardprod_id", "discount", "discount_rate", "order_item_id", "item_price", "profit_ratio", "quantity", "sales", "order_it
    "region", "state", "status", "zipcode", "product_id", "product_category_id", "product_description", "image", "name", "product_price", "product_status", "shipping_date", "mode"]

table_schema = {
    "table_customer_geo": ["customer_id", "customer_city", "customer_country", "customer_state", "customer_street", "customer_zipcode"],
    "table_customer": ["customer_id", "fname", "lname", "email", "passwd", "segment"],
    "table_product": ["product_id", "name", "category_id", "image", "product_price", "product_status"],
    "table_shipping": ["order_id", "days_for_shipping_real", "days_for_shipping_scheduled", "shipping_date", "mode"],
    "table_product_category": ["category_id", "category_name"],
    "table_orders": ["order_id", "customer_id", "order_date", "type", "delivery_status", "late_delivery_risk", "order_profit_per_order", "benefit_per_order", "department_id"],
    "table_orders_geo": ["order_id", "city", "region", "state", "status"],
    "table_order_item": ["order_item_id", "order_id", "product_id", "discount", "discount_rate", "item_price", "profit_ratio", "quantity", "sales", "order_item_total"],
    "table_department": ["department_id", "department_name", "latitude", "longitude", "market"]
    # "table_department": ["department_id", "department_name", "market"]
}

table_dtypes = {
    "table_customer_geo": [types.Integer(), types.VARCHAR(length=45), types.VARCHAR(length=45), types.VARCHAR(length=45), types.VARCHAR(length=45), types.VARCHAR(length=45)],
    "table_customer": [types.Integer(), types.VARCHAR(length=45), types.VARCHAR(length=45), types.VARCHAR(length=45), types.VARCHAR(length=45), types.VARCHAR(length=45)],
    "table_product": [types.Integer(), types.VARCHAR(length=45), types.Integer(), types.VARCHAR(length=256), types.Float(), types.VARCHAR(length=45)],
    "table_shipping": [types.Integer(), types.Integer(), types.Integer(), types.DateTime(), types.VARCHAR(length=45)],
    "table_product_category": [types.Integer(), types.VARCHAR(length=45)],
    "table_orders": [types.Integer(), types.Integer(), types.DateTime(), types.VARCHAR(length=45), types.VARCHAR(length=45), types.Integer(), types.Float(), types.Float(), types.Integer()],
    "table_orders_geo": [types.Integer(), types.Unicode(length=45), types.Unicode(length=45), types.Unicode(length=45), types.VARCHAR(length=45)],
    "table_order_item": [types.Integer(), types.Integer(), types.Integer(), types.Float(), types.Float(), types.Float(), types.Float(), types.Float(), types.Float()],
    "table_department": [types.Integer(), types.VARCHAR(length=45), types.Float(precision='10,8'), types.Float(precision='10,8'), types.VARCHAR(length=45)]
    # "table_department": [types.Integer(), types.VARCHAR(length=45), types.VARCHAR(length=45)]
}

table_insertion_order = ["table_customer", "table_customer_geo", "table_product_category", "table_product", "table_department", "table_orders", "table_orders_geo", "table_order_item", "table_shipping"]
```

Slika 3.4: Opisi tablica u Pythonu

```
from db_setup import DbHandler

db_handler = DbHandler.DbHandler(dbtype="mysql", hostname="localhost", port="3308")


root_engine = db_handler.create_engine(username="root", password="4321")

script_file = open("./sql_script/alter_root.sql", "rt")
db_handler.run_script(engine=root_engine, script_file=script_file)

root_engine = db_handler.create_engine(username="root", password="1234")


# Create user "data" && tables

script_file = open("./sql_script/create_user_data.sql", "rt")
db_handler.run_script(engine=root_engine, script_file=script_file)


data_engine = db_handler.create_engine(username="data", password="1234")

script_file = open("./sql_script/tables_dim.sql", "rt")
results = db_handler.run_script(engine=data_engine, script_file=script_file)
print(results)


script_file = open("./sql_script/tables_init.sql", "rt")
results = db_handler.run_script(engine=data_engine, script_file=script_file)
print(results)


from db_setup import DataHandler

dataset_handler = DataHandler.Dataset(path=f'./data/DataCoSupplyChainDataset.csv', delimiter=',', encoding="ISO-8859-1")
# dataset_handler.print_pandas_tables()

tables, table_dtypes, table_insertion_order = dataset_handler.get_tables()


# Fill tables

# print(table_dtypes)

for table_name in table_insertion_order:
    print(table_name)
    print(tables[table_name][:10])
    dtype = dict(zip([tables[table_name].index.name] + tables[table_name].columns.tolist(), table_dtypes[table_name]))
    db_handler.fill_table(df=tables[table_name], table_name=table_name[6:], engine=data_engine,
                          schema="data_co_schema", dtypes=dtype)
```

Slika 3.5: Python kod za inicijalizaciju baze podataka

Kako bih si olakšao punjenje baze stvorio sam dvije Python klase, jedna mi služi za stvaranje objekta koji ostvaruje vezu s bazom i izvršava SQL skripte dok druga klasa ima definirane SqlAlchemy tipove podataka, učitava CSV datoteku, izvršava čišćenje podataka i formatiranje podataka za transakcijski model.

### 3.3.3 Docker

Docker Compose datoteka sadrži opis tri containera koja se nalaze na istoj virtualnoj mreži kako bi mogli međusobno komunicirati preko IP adrese i sadrže portove koji su vidljivi računalu. Koristio sam Docker jer mi je bilo puno lakše testirati ispravnost koda, nisam morao brisati sheme i sl. nego bi samo pokrenuo containere i Pyhthon skriptu te bih time dobio novu "čistu" bazu na kojoj mogu testirati ETL procese.

```yaml
db:
  image: mysql
  command: --default-authentication-plugin=mysql_native_password
  restart: always
  environment:
    MYSQL_ROOT_PASSWORD: 4321
  ports:
    - "3308:3306"

adminer:
  image: adminer
  restart: always
  ports:
    - 5001:8080
  deploy:
    resources:
      limits:
        cpus: '0.2'
        memory: '1G'

metabase:
  image: luciantin/metabase_unlimited:latest
  restart: always
  ports:
    - 3000:3000
```

Slika 3.6: Docker Compose definicije containera

# 4 Dimenzijski Model Podataka

Kako bi mogli pratiti promijene podataka, svaka tablica sadrži atribute ”date_from” i ”date_to” pomoću kojih možemo odrediti od kada i do kada je neki zapis aktivan. Modelirano je tako da dimenzijski model podržava promijene narudžba i stavki narudžba osim samo njihovih dimenzija jer je prodaja proizvoda dinamička, kupac se može predomisliti nakon što naruči proizvode te ovim modelom možemo pratiti te promijene i bolje razumjeti potrebe kupaca. Dimenzijski model se sastoji od četiri dimenzije, dvije tablice činjenica i jedne tablice koja služi za spajanje tablica činjenica. Svaka tablica sadrži primarni ključ kojim je svaki zapis povezan s izvorišnom bazom podataka te se time može lakše ažurirati stanje.


Slika 4.1: Dijagram servisa

Dimenzija ”Location” opisuje adresu dostave, adresu kupca, tip tržišta kupca (dali je to privatni korisnik, tvrtka, itd.) te poslovnicu koja je obradila tu narudžbu. Dimenzija ”Time” sadrži dva datuma, kada je korisnik naručio proizvode i kada je proizvod isporučen. Dimenzija ”Product” opisuje tip proizvoda te je povezana s tablicom činjenica ”Fact_Item” i služi za opis tog proizvoda. Dimenzija ”Shipping” opisuje koliko je dana bilo potrebno za isporuku proizvoda i sadrži način dostave. Tablica činjenica ”Fact_Order” opisuje narudžbe, profit i ukupnu vrijednost proizvoda, dok tablica činjenica ”Fact_Item” opisuje pojedinu stavku narudžbe. Stavka narudžbe ima atribute koji opisuju popust, ostvareni profit i količinu proizvoda. Za postavljanje baze podataka koristim tri SQL datoteke sa DDL i DCL naredbama, dvije sadrže naredbe za izradu sheme dok jedna SQL datoteka sadrži naredbe za stvaranje korisnika, izmjenu lozinke root korisnika i postavljanje pravila korisnika koji upravlja transakcijskom i dimenzijskom shemom.

# 5 ETL Procesi

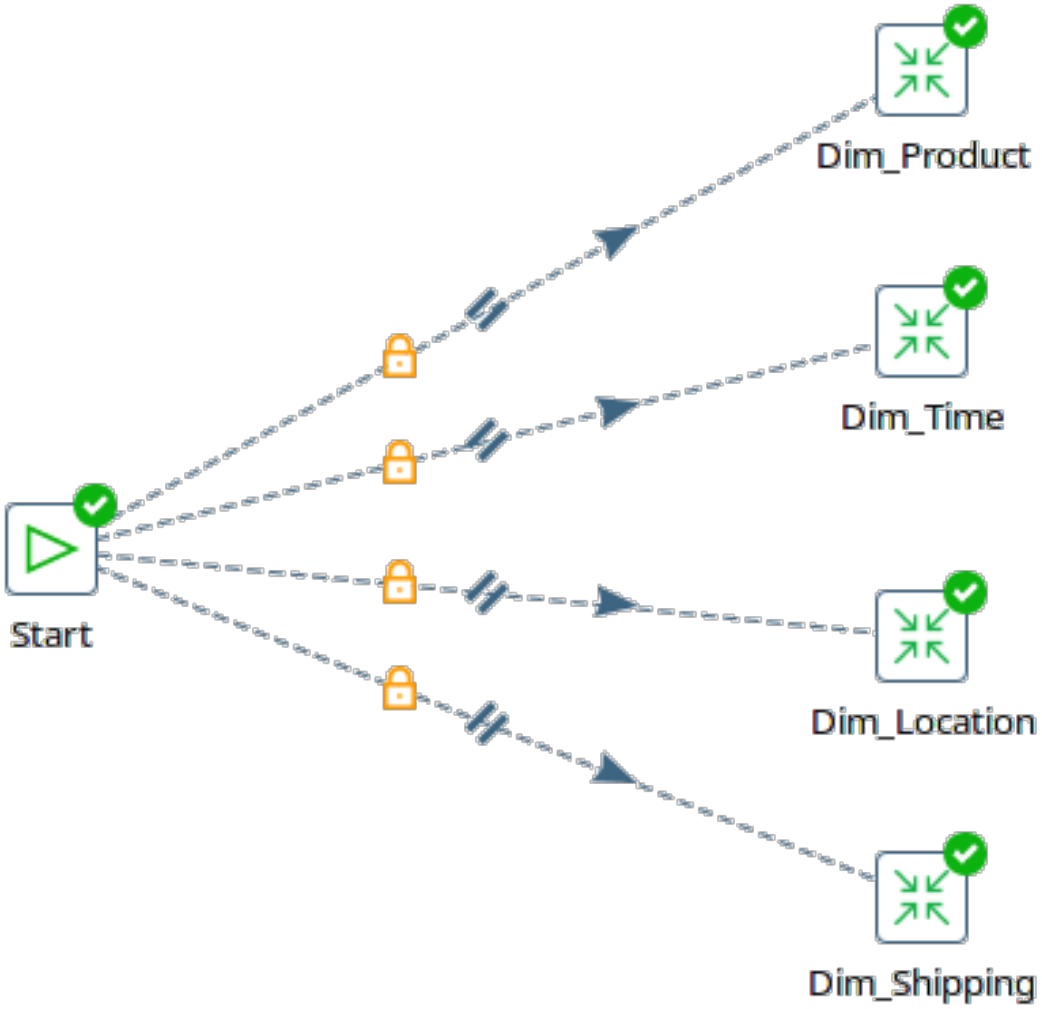
ETL procesi služe za ekstrakciju, transformaciju i punjenje dimenzijskog modela, sve dimenziju su sporo mijenjajući dimenzije tipa 2, sadrže verziju dimenzijskog unosa u atributu ”version” i datume koji opisuju interval u kojem je zapis aktivan . Ažuriranje dimenzijskog modela se izvršava pomoću elementa ”Dimension lookup/update” koji se brine o unosu datuma, verzije i tehničkog ključa [6]. Svaka tablica dimenzijskog modela sadrži primarni ključ kojim je zapis povezan s zapisom u transakcijskoj bazi te se time lakše ažuriraju vrijednosti.

## 5.1 Pentaho ”Job”

Sve ETL transformacije se izvršavaju u dva Pentaho Job-a. Sveukupno se koriste 6 transformacija za punjenje dimenzijskog modela, 4 transformacije za punjenje dimenzijskih tablica, jedna transformacija za punjenje obje tablice činjenica i jedna transformacija za punjenje tablice kojom se spajaju tablice činjenica.

Dimenzije se pune paralelno jer ne ovise jedna o drugoj, slika 5.1 . Nakon što se taj Pentaho job izvrši onda se krene na sljedeće transformacije u glavnom jobu, prikazan slikom 5.2 .


Slika 5.1: Glavni Pentaho ”Job”

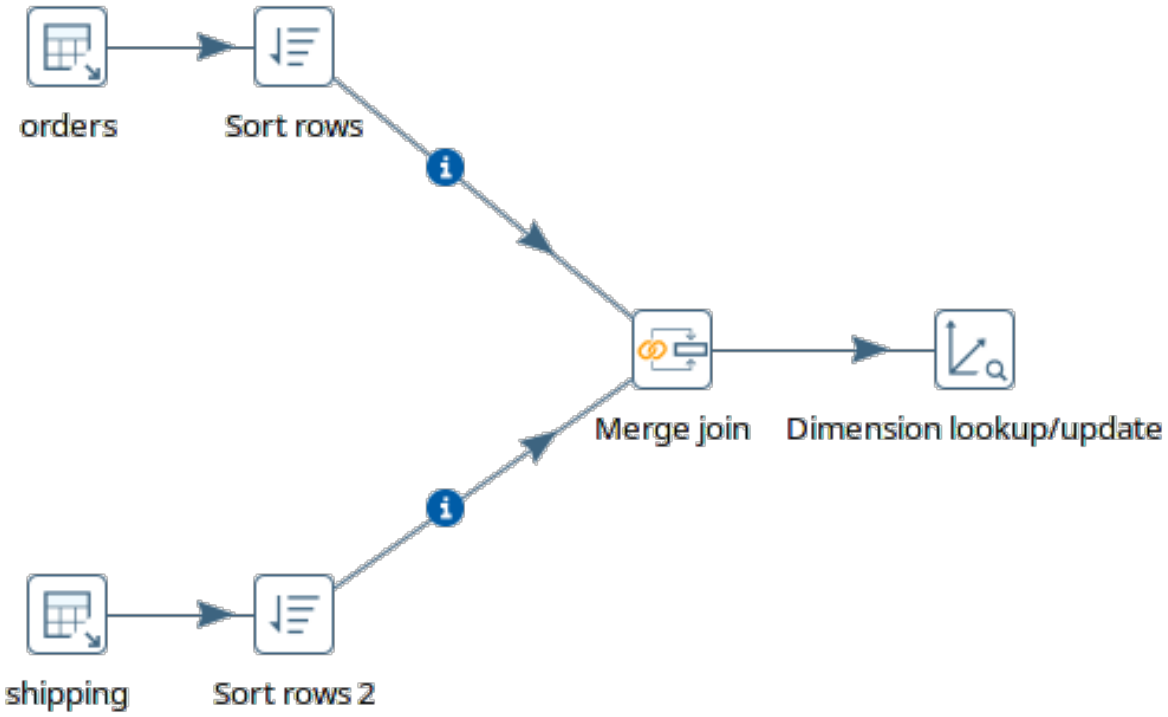



Slika 5.2: Punjenje dimenzija

## 5.2 Transformacije

### 5.2.1 Dimenzija Vrijeme

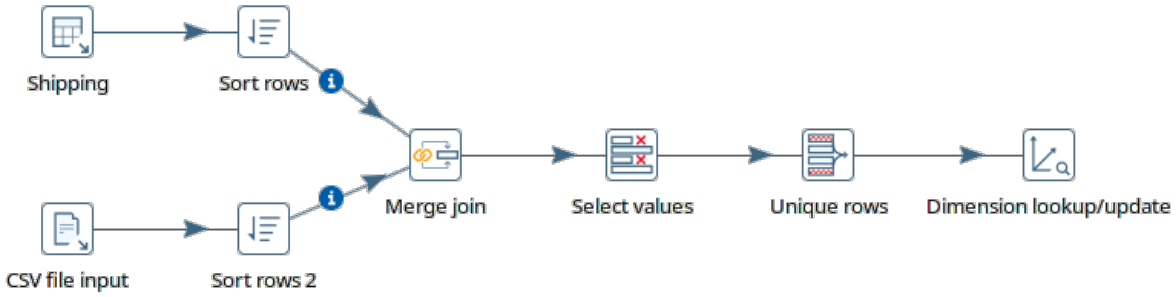
Vremenska dimenzija sadrži dva atributa koji opisuju vrijeme kada je narudžba donesena i kada je dostavljena. Atributi se nalaze u različitima tablicama pa se moraju prvo tablice spojiti prije punjenja dimenzije.




Slika 5.3: Punjenje vremenske dimenzije

### 5.2.2 Dimenzija Dostava

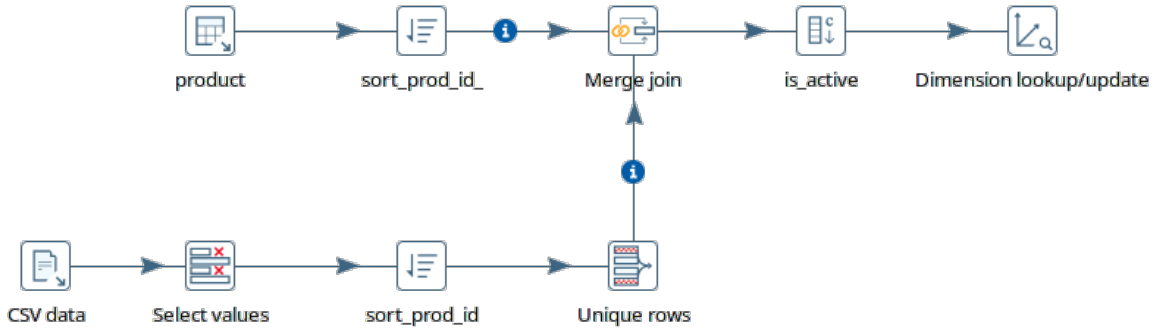
Sadrži atribute koji opisuju dostavu pojedine narudžbe, način dostave, dani potrebni za dostavu te rizik za zakašnjenje dostave. Puni se uz pomoć podataka iz tablice ”Shipping” i CSV datoteke, ti podaci se tada sortiraju, spoje i selektiraju se samo potrebni atributu. Dimenzija se puni samo s jedinstvenim vrijednostima zato što je CSV datoteka denormalizirana te se, po primarnom ključu narudžba, moraju maknuti duplikati koji su nastali zbog toga što svaka narudžba ima više stavki.




Slika 5.4: Punjenje dimenzije dostava

### 5.2.3 Dimenzija Proizvod

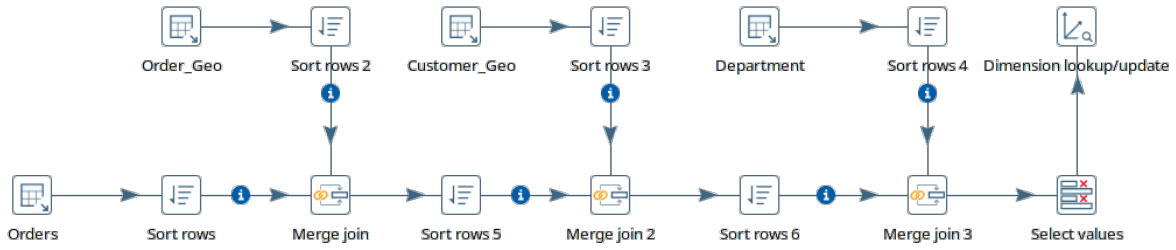
Dimenzija proizvod opisuje proizvod u tablici činjenica za stavke narudžba. Sadrži naziv proizvoda, cijenu i naziv kategorije koj taj proizvod pripada. Osim što spajanja podataka iz CSV datoteke i tablice, dodao sam i atribut koji označuje jeli proizvod trenutno aktivan ali nije bilo potrebno jer se isto može postići s datumima koji se unose u ”Dimension lookup/update” elementu.




Slika 5.5: Punjenje dimenzije proizvod

### 5.2.4 Dimenzija Lokacija

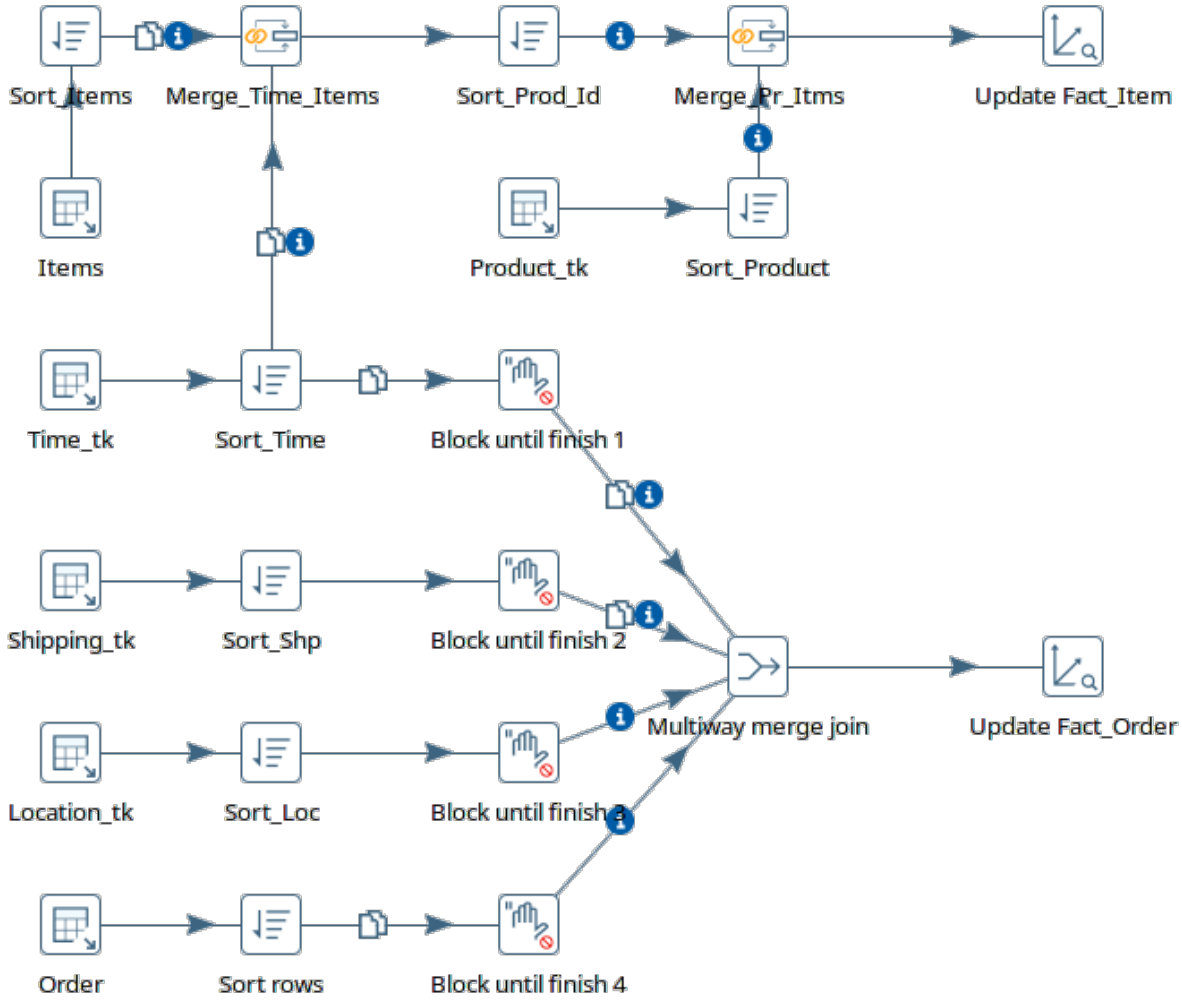
Slika 5.2.4 prikazuje Pentaho transformaciju koja prikuplja lokacije iz više tablica te ih spaja s primarnim ključem narudžba. Podaci se sortiraju ovisno o ključu pomoću kojeg se spajaju jer se spajaju pomoću različitih primarnih ključeva. Prije punjenja, odaberu se vrijednosti kojima će se napuniti dimenzija te se nakon toga dimenzija napuni zajedno s tehničkim ključem, datumima i verzijom.




Slika 5.6: Punjenje dimenzije lokacija

### 5.2.5 Tablice Činjenica

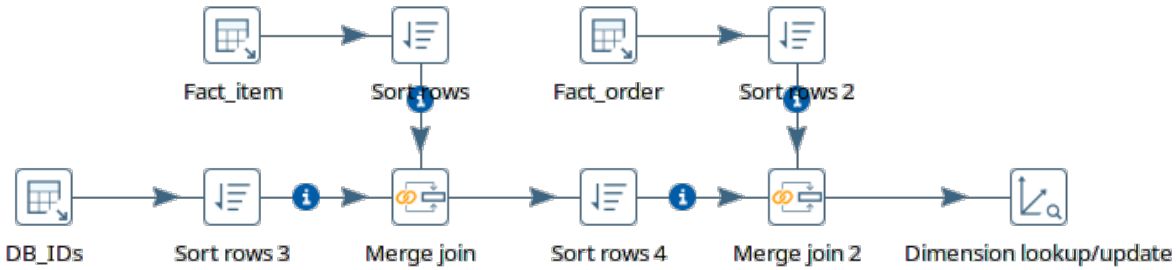
Kao što je ranije navedeno, koriste se dvije tablice činjenica, jedna prati ukupan profit i ukupnu cijenu za svaku narudžbu dok druga tablica činjenica prati količinu, , cijenu i popust stavke narudžbe. Obje sadrže atribute za praćenje od kad i do kada je aktivan pojedini zapis jer se pomoću toga mogu pratiti interesi kupaca, dali su odlučili nakon narudžbe zamijeniti neku stavku. Tablice činjenica sadrže primarni ključ kojim se stvara veza na transakcijski zapis iz kojega je preuzeta pojedina činjenica. Tablice činjenica se pune u istoj transformaciji jer je potrebno manje vremena zbog kopiranja sortiranih podataka dimenzijske tablice vremena. Umjesto da se dvaput sortira 160,000+ zapisa, oni se sortiraju jedanput, kopiraju i vežu s obje tablice činjenica.




Slika 5.7: Dijagram servisa

### 5.2.6 Spajanje Tablica Činjenica

Kako bi se mogle tablice činjenica povezati, stvorena je tablica koja služi za povezivanje svakog tehničkog ključa tablice činjenica narudžba sa tehničkim ključevima tablice stavki. Tablica je usklađena dimenzija kojom se može iz jedne tablice činjenica doći do druge, no ona ne sadrži nikakve atribute koji opisuju tu vezu. Kao i ostale tablice dimenzijskog modela, i ova tablica je napravljena tako da prati povijesne podatke. Slika 5.2.6 prikazuje izgled tog ETL procesa, prvo se dohvate originalni podaci iz transakcijske baze kako bi se mogli spojiti točni tehnički ključevi. Zatim se dohvaćeni primarni ključevi narudžba i stavki spajaju s posebno dohvaćenom tablicom činjenica stavki narudžbe te se nakon toga spajaju s tablicom činjenica za narudžbe i onda se dimenzijski model ažurira.




Slika 5.8: Dijagram servisa

# 6 Vizualizacija Podataka

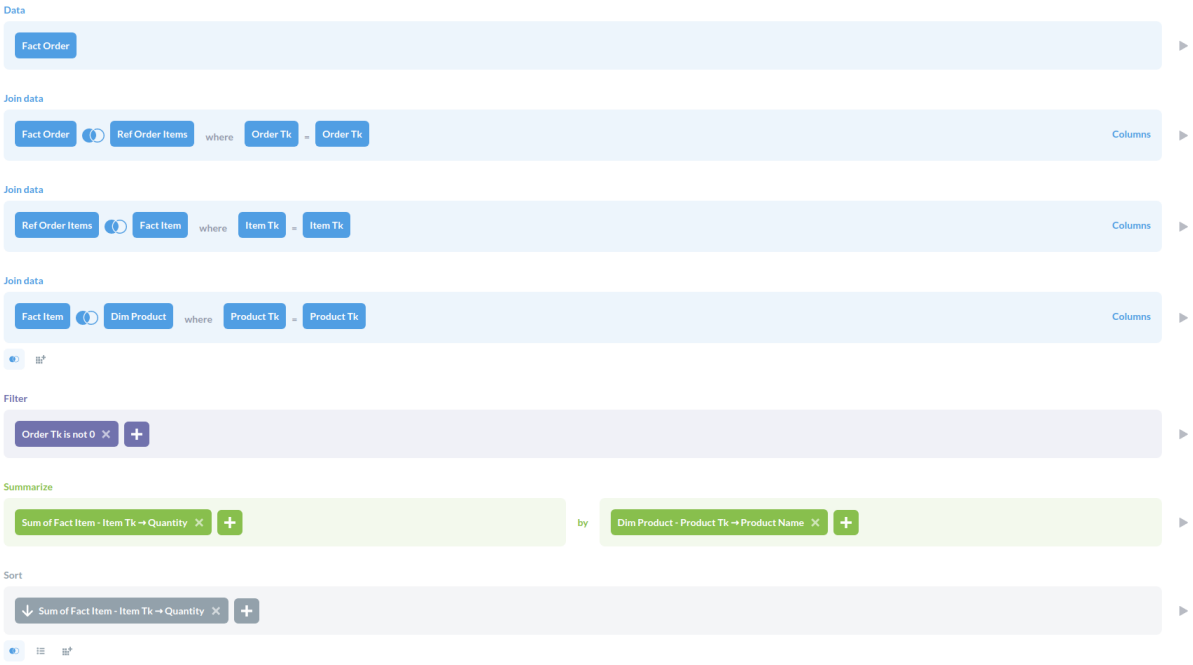
Podaci se vizualiziraju kako bi pomogli u odlučivanju, stoga sam odlučio vizualizirati par različitih upita koji mislim da su važni za donošenje odluka o poslovanju tvrtke. Za vizualizaciju je korišten Metabase jer je otvoren kod, te je to web aplikacija koja se može lokalno pokrenuti uz pomoć Docker-a.

## 6.1 Metabase

Kako bih mogao koristiti Metabase sa velikom količinom podataka, morao sam zamijeniti par vrijednosti koje određuju gornju granicu dohvaćanja zapisa. Stvorio sam svoju Metabase sliku i postavio sam ju na DokcerHub. Metabase se sastoji od više elemenata, nadzorna ploča služi za prikazivanje više grafova dok su grafovi stvoreni uz pomoć posebnog alata koji omogućuje lakše postavljanje upita i odabira tipa grafa.

### 6.1.1 Dizajn Upita

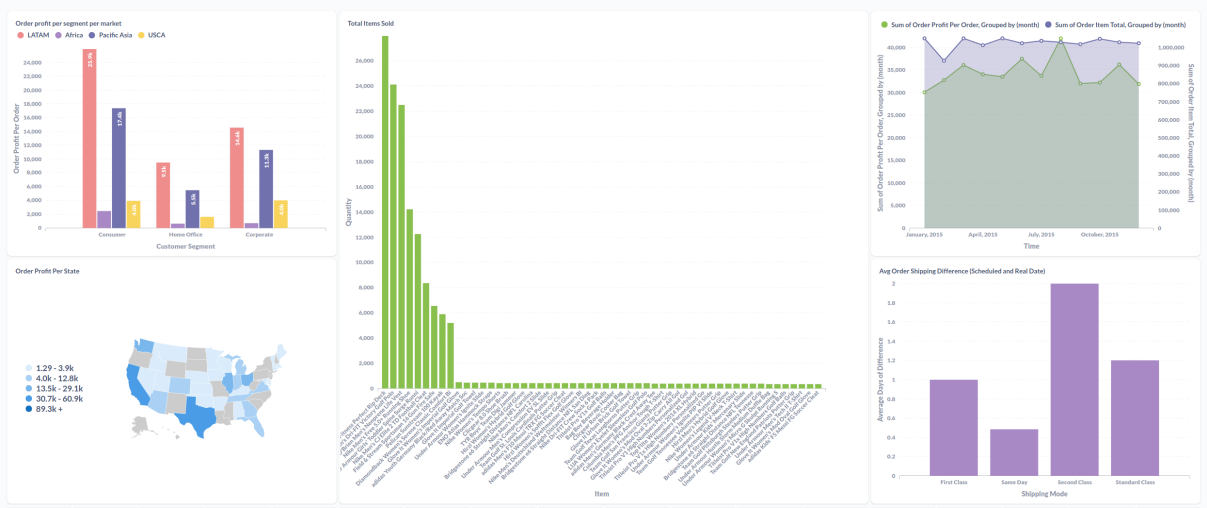
Upiti se stvaraju tako da se odabere početna tablica nakon čega se podaci mogu spojiti s drugim tablicama, može se stvoriti novi atribut od dobijenih atributa te se još podaci mogu grupirati, filtrirati ili se može ograničiti broj dohvaćenih zapisa. Slika 6.1.1 prikazuje upit kojim se dohvaćaju podaci o ukupnoj prodaji svakog proizvoda.




Slika 6.1: Upit kojim se dohvaćaju podaci o ukupnoj prodaji proizvoda

### 6.1.2 Nadzorna ploča

Nakon što se graf spremi, on se može dodati na neki od nadzornih ploča gdje mu se može promijeniti veličina, boja, pozicija te se određenim tipovima grafova može dodati više serija podataka, kako bi se prikazalo više grafova u jednom.



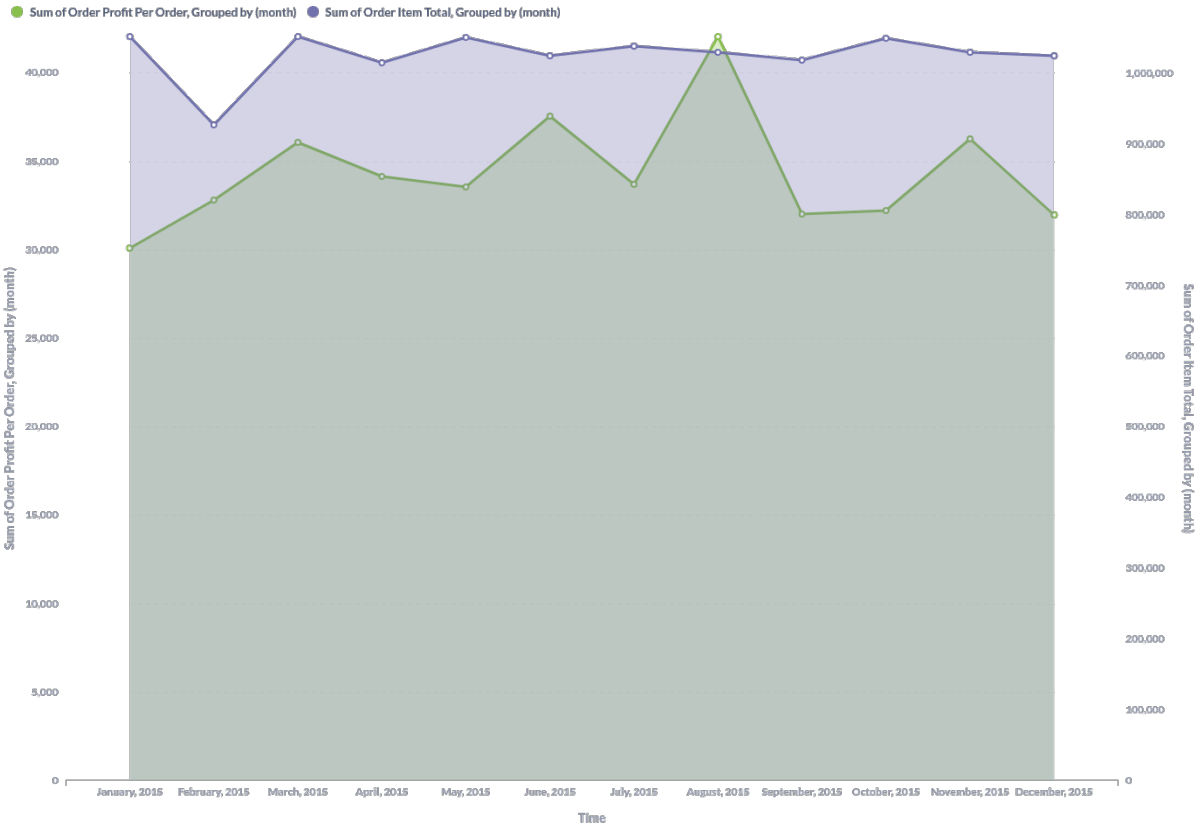

Slika 6.2: Dijagram servisa

## 6.2 Vizualizacije

Cilj vizualizacija podataka je zadovoljiti potrebe menadžera stoga sam odabrao prikazati na geografskoj karti frekvenciju narudžba u pojedinim državama SAD-a, prikazati koji se proizvodi najčešće naručuju, profit narudžba ovisno o segmentu kupaca u svakom tržištu, ukupnu cijenu narudžbe i ostvarenoga profita u nekom vremenskom intervalu i prosječnu razliku između stvarnih dana potrebnih za dostavu i zakazanih dana, grupirano po tipu dostave.

### 6.2.1 Ukupna cijena narudžbe i profit u vremenskom intervalu

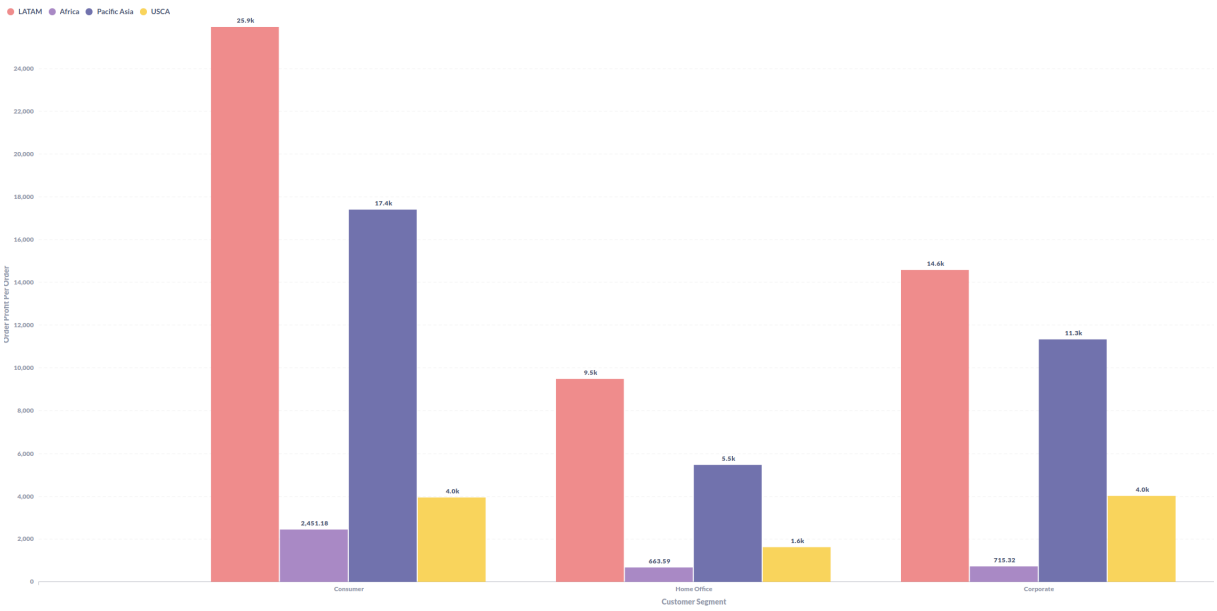
Slika 6.2.1 prikazuje ukupnu cijenu narudžbe ljubičastom bojom te je na tu vremensku seriju dodan drugi graf koji prikazuje ukupan profit. Vremensko razdoblje je jedna godina te su podaci sumirani za svaki mjesec u toj godini.




Slika 6.3: Dijagram servisa

### 6.2.2 Dobit od narudžbe po tržištu, po segmentu kupaca

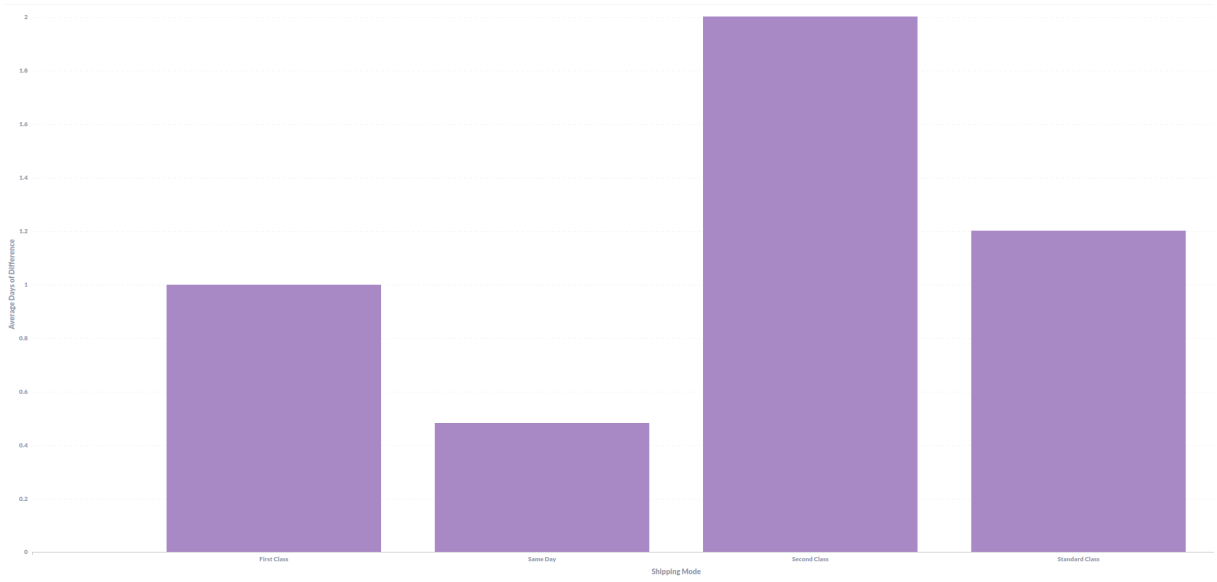
Slika 6.2.2 prikazuje ostvarenu dobit po tržištu, te su podaci, unutar svakog tržišta, grupirani po segmentu kupaca.




Slika 6.4: Dijagram servisa

### 6.2.3 Prosječna razlika u zakazanim danima dostave i stvarnim danima dostave po načinu dostave

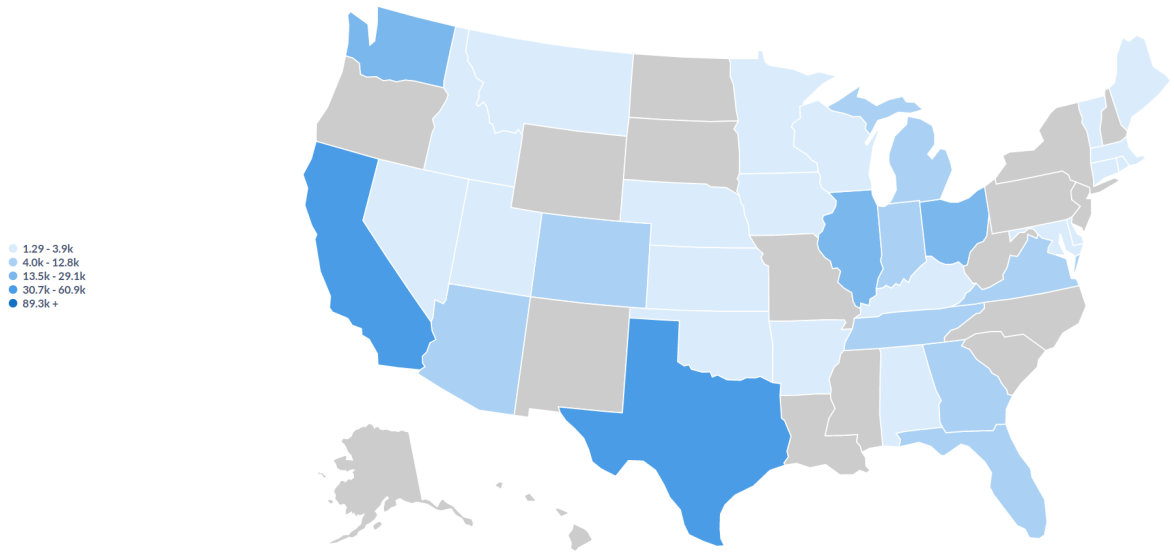
Graf na slici 6.2.3, za svaki tip dostave, prikazuje prosječnu razliku između zakazanih dana potrebnih za dostavu i stvarnih dana koliko je trebalo da se narudžba dostavi. Ovime se može vidjeti dali tvrtke, uz pomoć kojih se dostavljaju proizvodi, pružaju dovoljno dobru uslugu našim kupcima.




Slika 6.5: Dijagram servisa

### 6.2.4 Dobit po narudžbi po državi

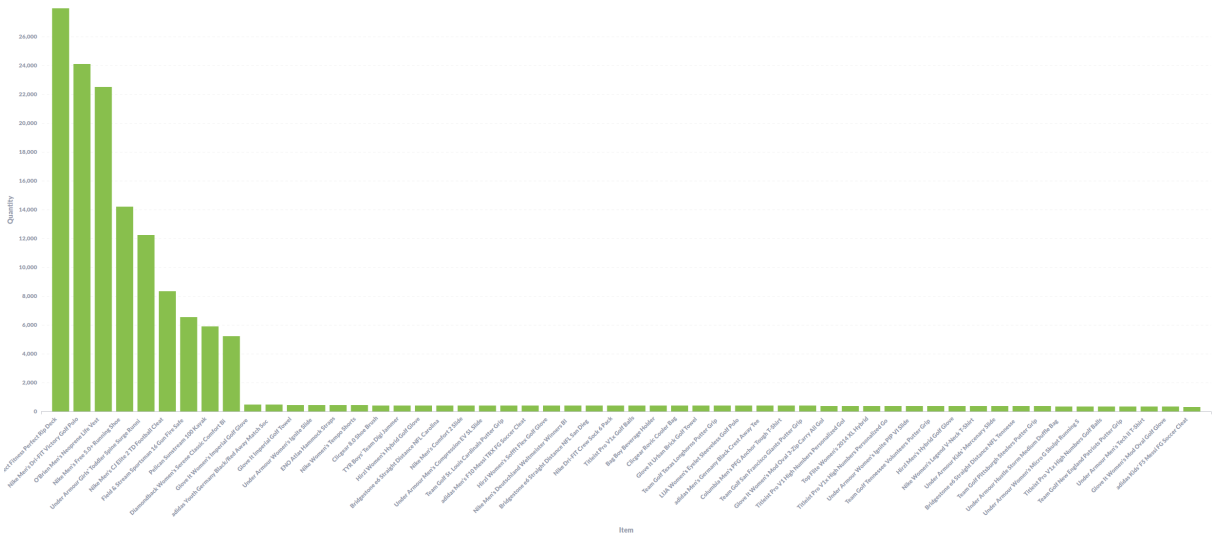
Kako bismo vidjeli u kojim državama najbolje poslujemo a u kojima najlošije, na karti SAD-a sam prikazao sumu profita za svaku državu.




Slika 6.6: Dijagram servisa

### 6.2.5 Ukupna količina prodanih proizvoda

Kako bi se moglo vidjeti koji su proizvodi najuspješniji, vizualizirao sam ukupnu količinu prodanih proizvoda.




Slika 6.7: Dijagram servisa

# 7 Zakljucak

Kroz ovaj rad prikazan je proces stvaranja skladišta podataka, punjenje i vizualizacija podataka. Projekt je izrađen uz pomoć raznih alata i programskih rješenja kako bi se dobila nadzorna ploču u kojoj su vidljivi podaci o poslovanju tvrtke. Denormalizirani podaci, koji opisuju rad tvrtke, pune se uz pomoć Python-a koji također transformira te podatke kako bi se mogli prenijeti u transakcijsku bazu podataka. ETL procesi su dizajnirani tako da se podaci mogu lagano ažurirati na novije stanje, te se za punjenje dimenzijskog modela koriste podaci iz različitih izvora. Program za vizualizaciju podatak je morao biti izmjenjen kako bi mogao prikazati velike količine podataka te je također korišten Docker kako bi se projekt mogao pokrenuti na raznim računalima bez utjecaja nečije okoline.

# Reference


[1] *DataCo SMART SUPPLY CHAIN FOR BIG DATA ANALYSIS*. kaggle.com. URL: https://www.kaggle.com/shashwatwork/dataco-smart-supply-chain-for-big-data-analysis (visited on 05/20/2021).

[2] *docker/compose*. GitHub, May 2021. URL: https://github.com/docker/compose (visited on 05/20/2021).

[3] R. Kimball and M. Ross. *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. Wiley, 2013. ISBN: 9781118732281. URL: https://books.google.hr/books?id=4rFXzk8wAB8C.

[4] *luciantin-faks/faks-SPI*. GitHub, May 2021. URL: https://github.com/luciantin-faks/faks-SPI (visited on 05/20/2021).

[5] *metabase/metabase*. GitHub, May 2021. URL: https://github.com/metabase/metabase (visited on 05/20/2021).

[6] *Pentaho - Loading fact table with SCD type 2 dimension*. www.howtobuildsoftware.com. URL: https://www.howtobuildsoftware.com/index.php/how-do/6cV/pentaho-data-warehouse-loading-fact-table-with-scd-type-2-dimension (visited on 05/20/2021).