Analiza Prodaje

DataCo SMART SUPPLY CHAIN

Kolegij: Sustavi poslovne inteligencije , Student: Lucian Tin Udovičić , Mentor: doc.dr.sc. Goran Oreški

Sadržaj

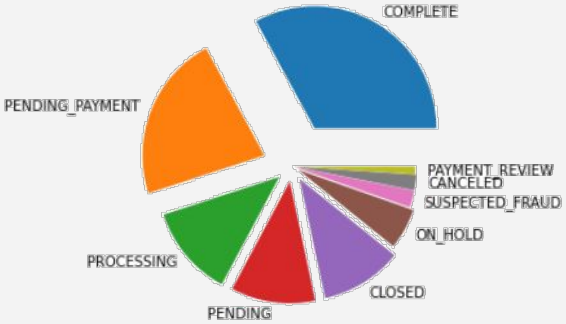
Uvod

- Dataset : DataCo SMART SUPPLY CHAIN
- Prodaja proizvoda
- Skladište podataka
- Potpora odlučivanju

Analiza Podataka

- 53 Atributa
- 160,000 + redova
- 1 CSV datoteka, denormalizirana

- Malo nepostojećih vrijednosti
- Puno datuma, raznoliki



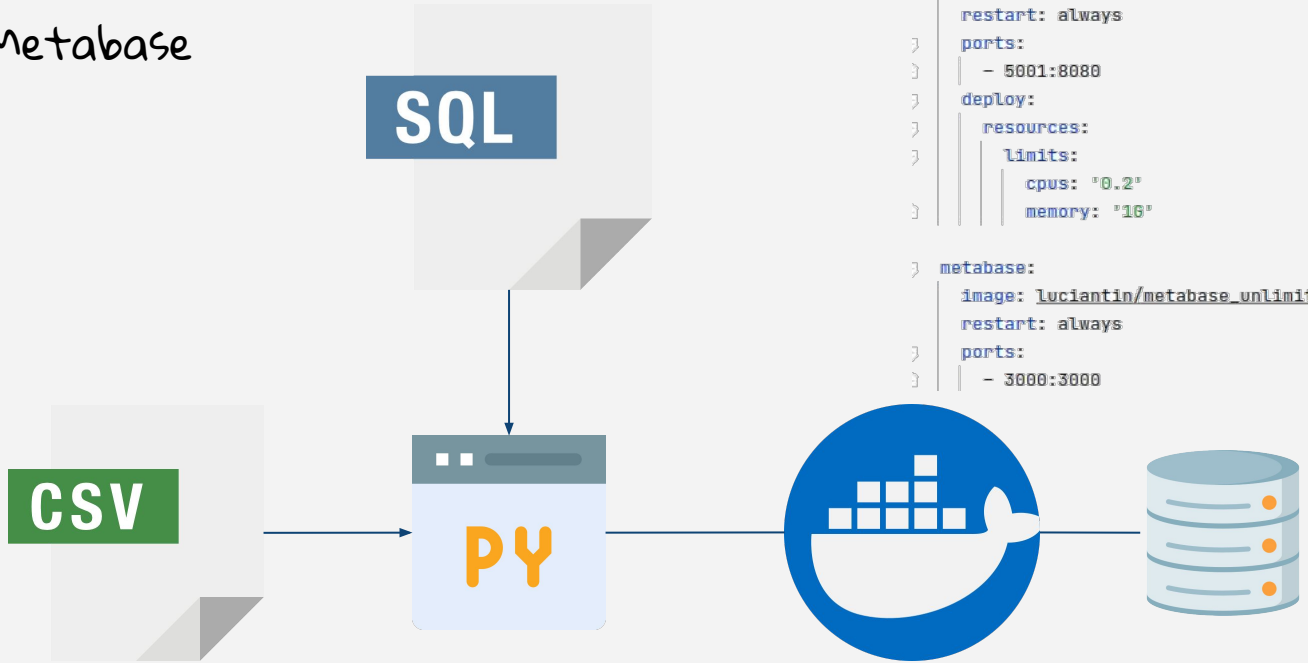
	Tipovi	Jedinstvene	Nedostajuće
Type	object	4	0
Days for shipping (real)	int64	7	0
Days for shipment (scheduled)	int64	4	0
Benefit per order	float64	21998	0
Sales per customer	float64	2927	0
Delivery Status	object	4	0
Late_delivery_risk	int64	2	0
Category Id	int64	51	0
Category Name	object	50	0
Customer 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

```
tip = data.dtypes.to_frame().rename(columns = {0:'Tipovi'})  data: {Dat
jed = data.nunique().to_frame().rename(columns = {0:'Jedinstvene '})  d
ned = data.isna().sum().to_frame().rename(columns = {0:'Nedostajuće'})

tip .join(jed).join(ned)  tip: {DataFrame: (53, 1)}  jed: {DataFrame: (
```

Baza Podataka

- MySQL
- Python
- Metabase

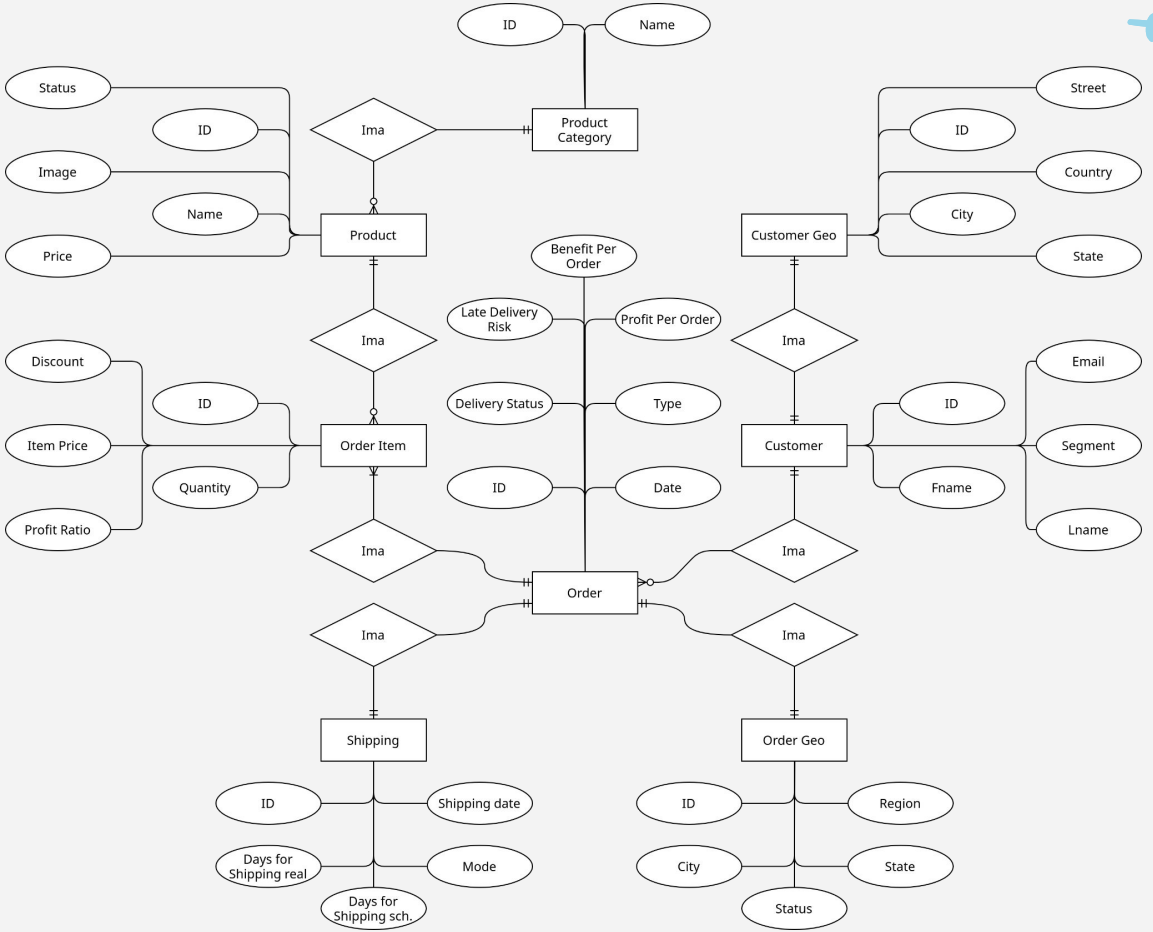
```yaml
db:
  image: mysql
  command: --default-authentication-plugin=mysql_native_password
  restart: always
  environment:
    MYSQL_ROOT_PASSWORD: 4321
  ports:
    - "3308:3306"

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
```



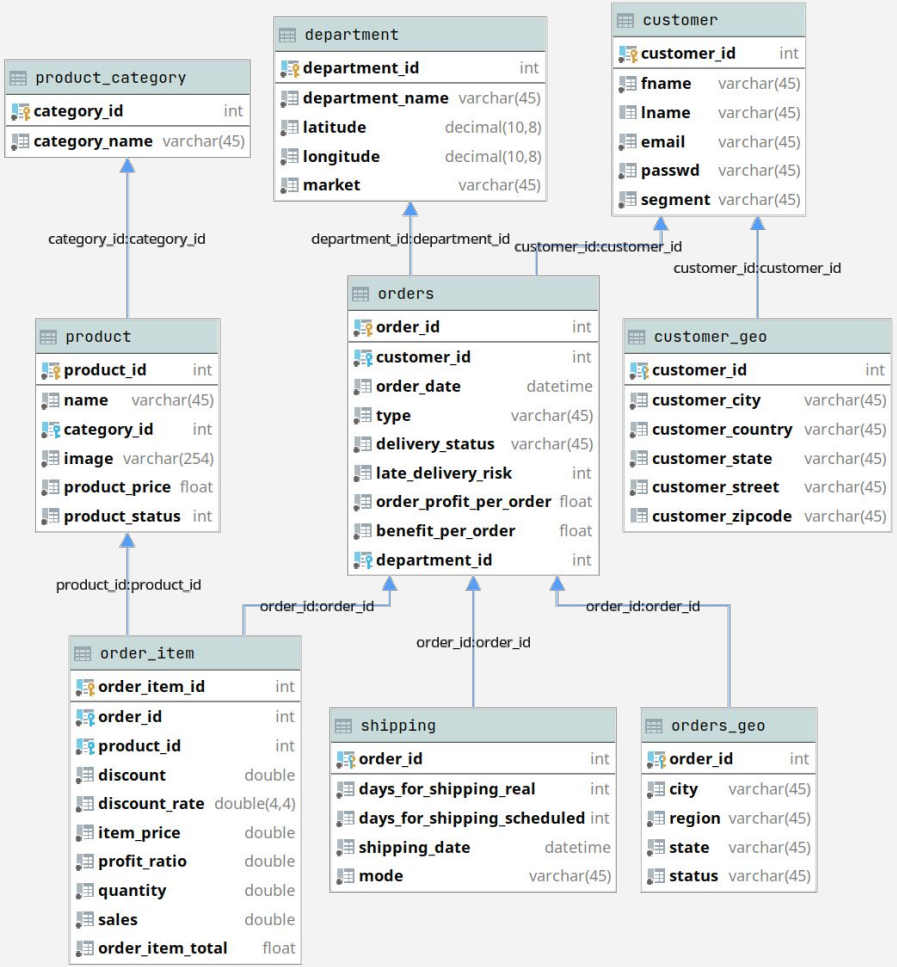
```python
from db_setup import DbHandler

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)
```

```python
table_dtypes = {
    "table_customer_geo": [types.Integer(), types.VARCHAR(length=45), types.VARCHAR(length=45), types.VARCHAR(length=4
    "table_customer": [types.Integer(), 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=254), types.Flo
    "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(lengt
    "table_orders_geo": [types.Integer(), types.Unicode(length=45), types.Unicode(length=45), types.Unicode(length=45)
    "table_order_item": [types.Integer(), types.Integer(), types.Integer(), types.Float(), types.Float(), types.Float(
    "table_department": [types.Integer(), types.VARCHAR(length=45), types.Float(precision='10,8'), types.Float(precisi
    # "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_dep

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"))
```

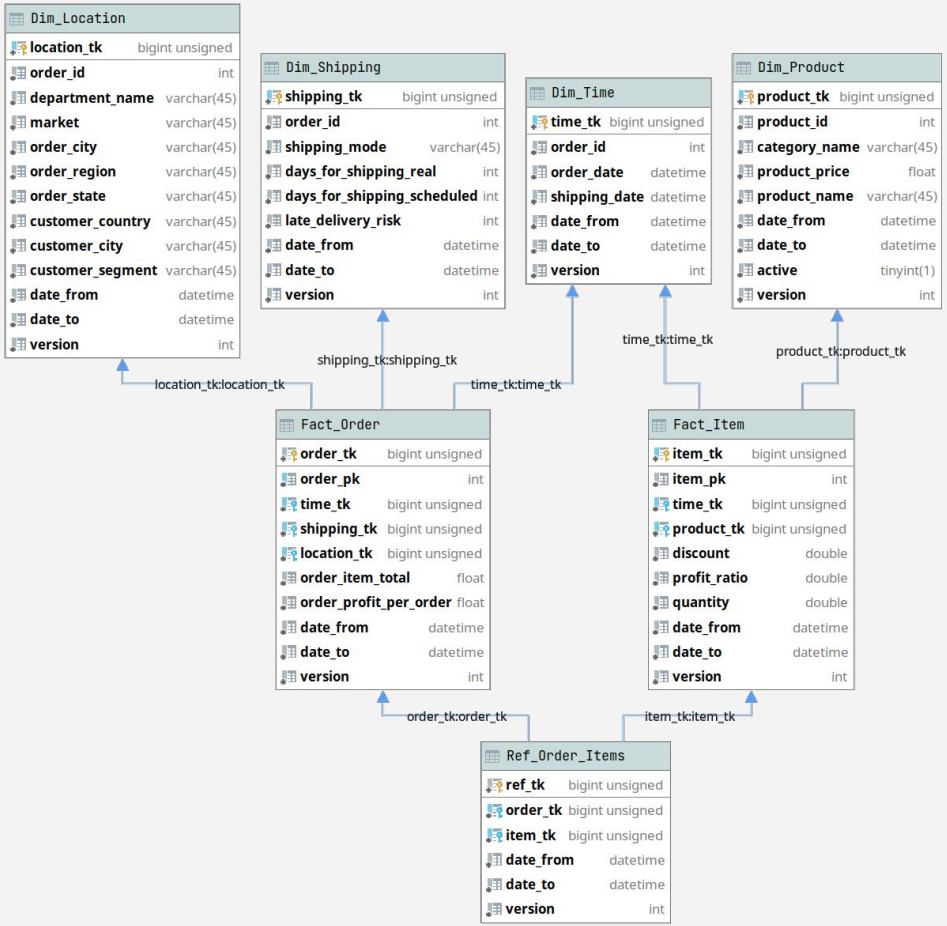
ER Dijagram


ID
Name
Status
ID
Ima
Product Category
Image
Name
Price
Product
Street
ID
Country
City
State
Customer Geo
Benefit Per Order
Late Delivery Risk
Profit Per Order
Ima
Ima
Discount
ID
Item Price
Quantity
Profit Ratio
Order Item
Delivery Status
Type
ID
Date
Email
ID
Segment
Fname
Lname
Customer
Ima
Ima
Order
Ima
Ima
Shipping
Order Geo
ID
Shipping date
Days for Shipping real
Mode
Days for Shipping sch.
ID
Region
City
State
Status

ER Model


product_category
category_id int
category_name varchar(45)
department
department_id int
department_name varchar(45)
latitude decimal(10,8)
longitude decimal(10,8)
market varchar(45)
customer
customer_id int
fname varchar(45)
lname varchar(45)
email varchar(45)
passwd varchar(45)
segment varchar(45)
category_id:category_id
department_id:department_id
customer_id:customer_id
customer_id:customer_id
product
product_id int
name varchar(45)
category_id int
image varchar(254)
product_price float
product_status int
orders
order_id int
customer_id int
order_date datetime
type varchar(45)
delivery_status varchar(45)
late_delivery_risk int
order_profit_per_order float
benefit_per_order float
department_id int
customer_geo
customer_id int
customer_city varchar(45)
customer_country varchar(45)
customer_state varchar(45)
customer_street varchar(45)
customer_zipcode varchar(45)
product_id:product_id
order_id:order_id
order_id:order_id
order_id:order_id
order_item
order_item_id int
order_id int
product_id int
discount double
discount_rate double(4,4)
item_price double
profit_ratio double
quantity double
sales double
order_item_total float
shipping
order_id int
days_for_shipping_real int
days_for_shipping_scheduled int
shipping_date datetime
mode varchar(45)
orders_geo
order_id int
city varchar(45)
region varchar(45)
state varchar(45)
status varchar(45)

Dimenzijski Model



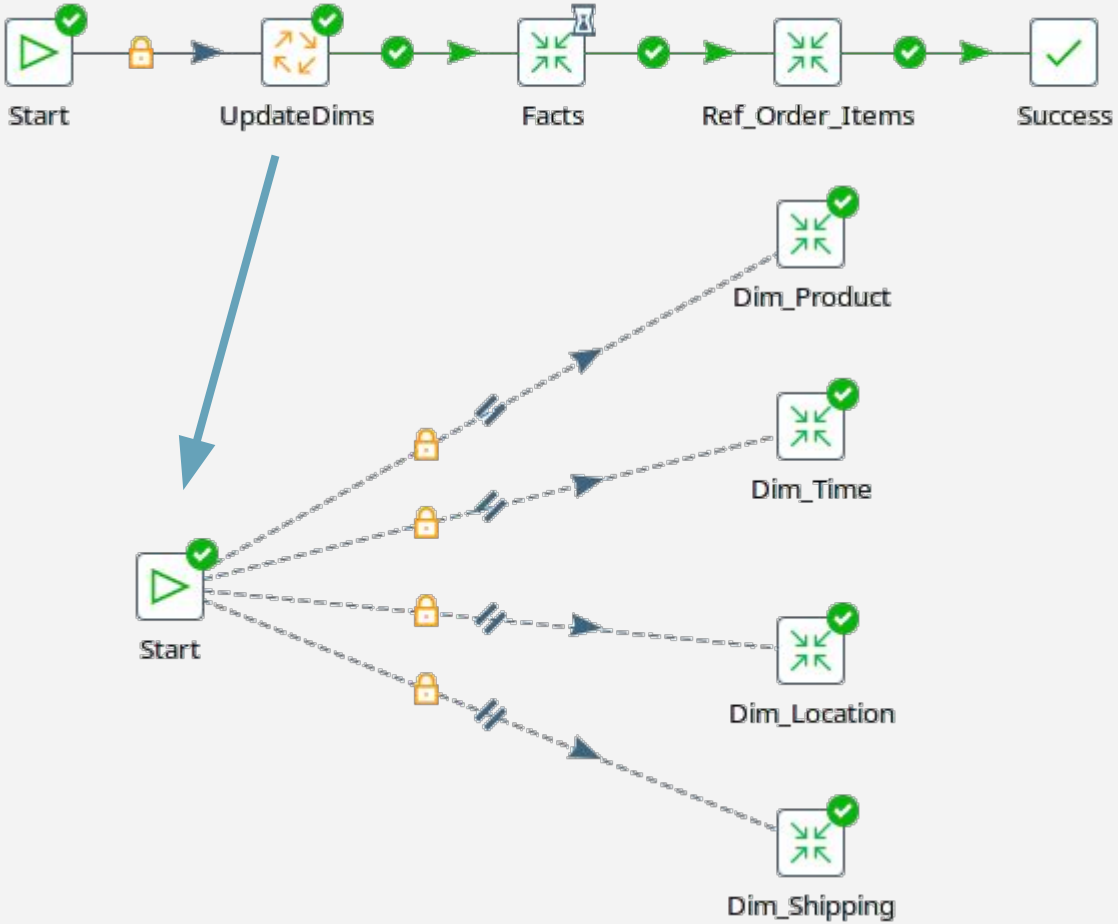
ETL

- “Job” za ažuriranje skladišta podataka
  - Glavni puni tablice činjenica
  - Paralelno puni dimenzijske tablice

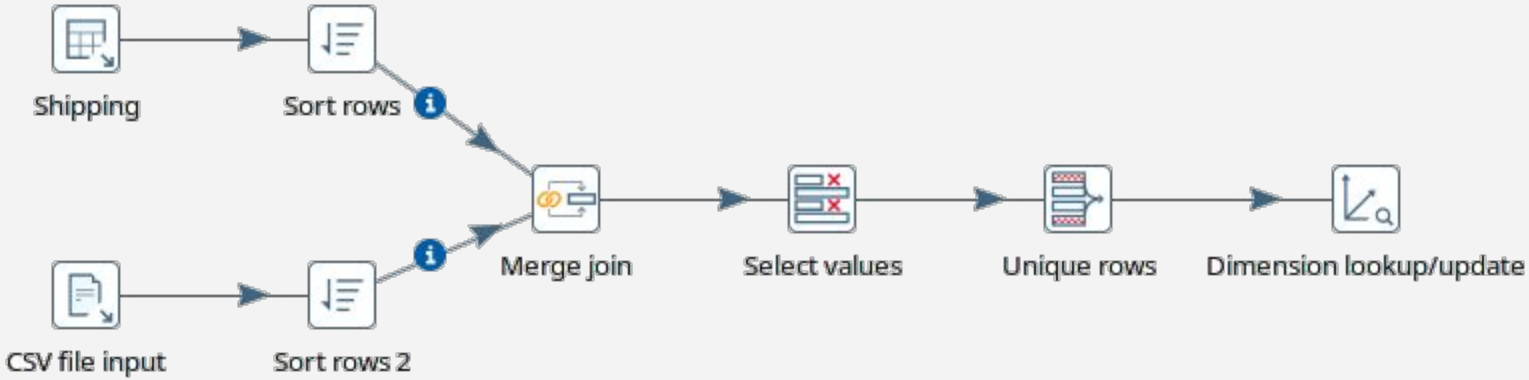
- Sveukupno 6 Transformacija

- SCD Type 2 + revision number (verzija)

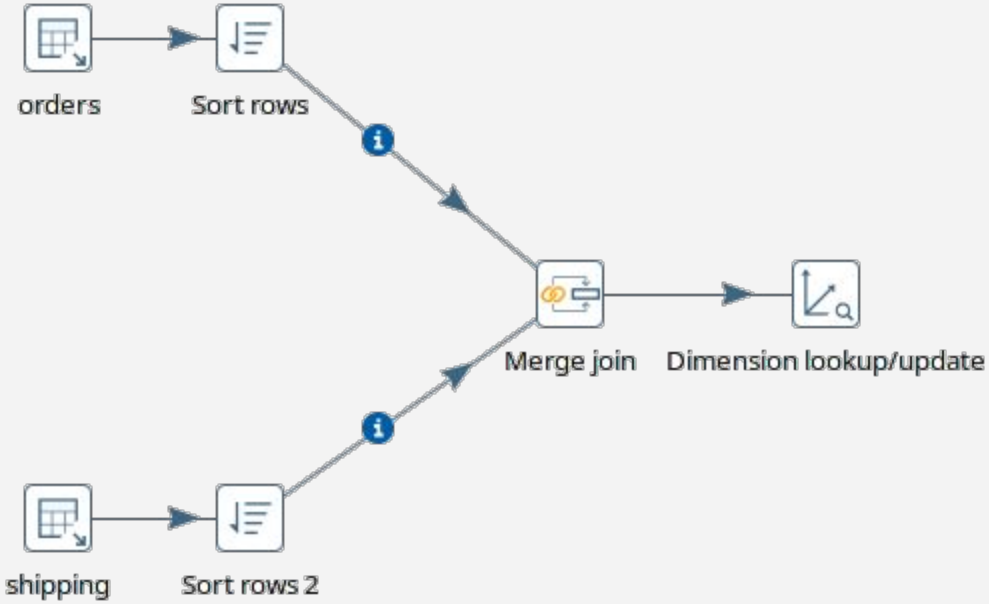
Pentaho JOB


Start
UpdateDims
Facts
Ref_Order_Items
Success
Dim_Product
Dim_Time
Start
Dim_Location
Dim_Shipping

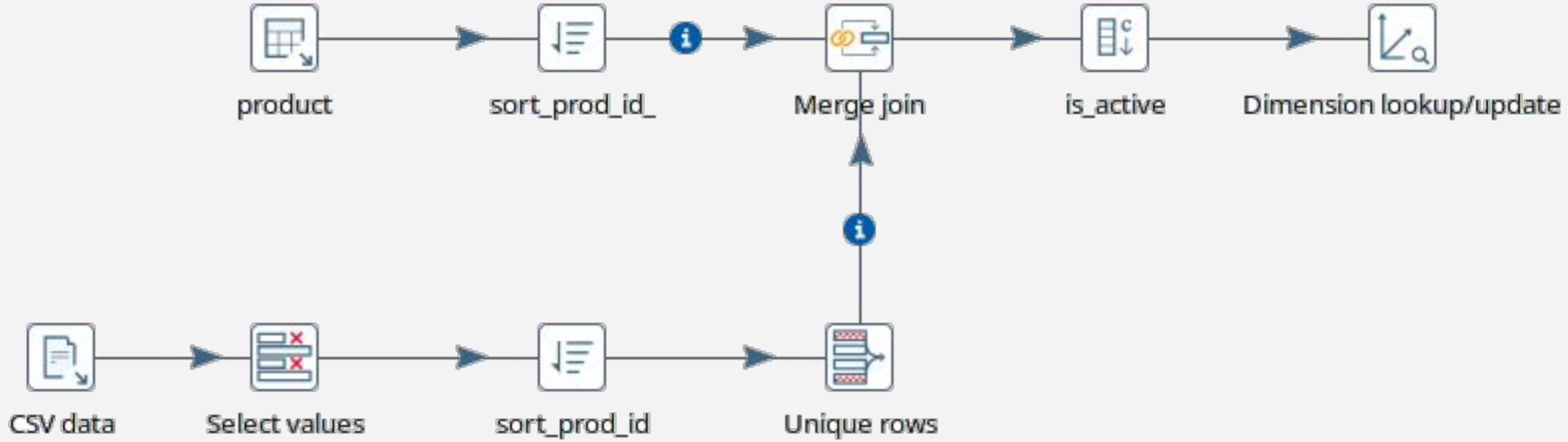
Shipping


Shipping
Sort rows
CSV file input
Sort rows 2
Merge join
Select values
Unique rows
Dimension lookup/update

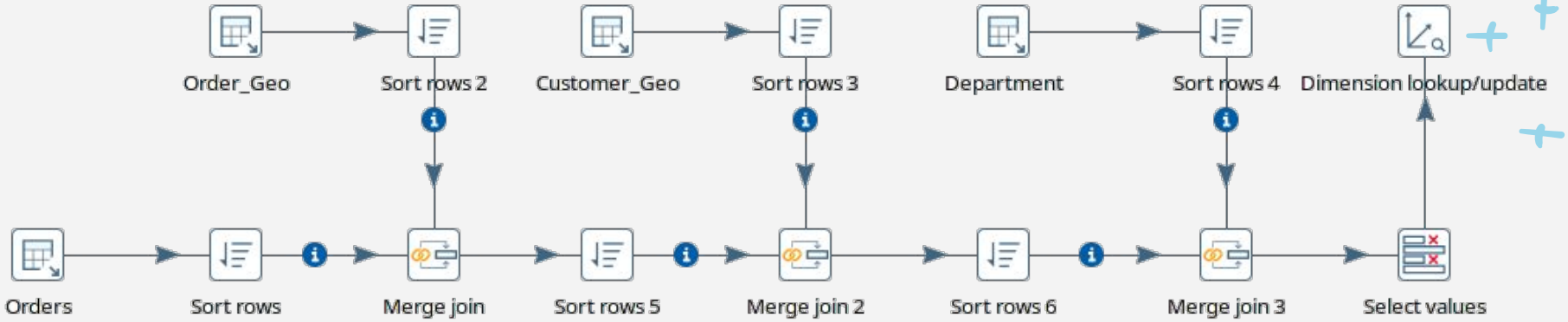
Time


orders
Sort rows
Merge join
Dimension lookup/update
shipping
Sort rows 2

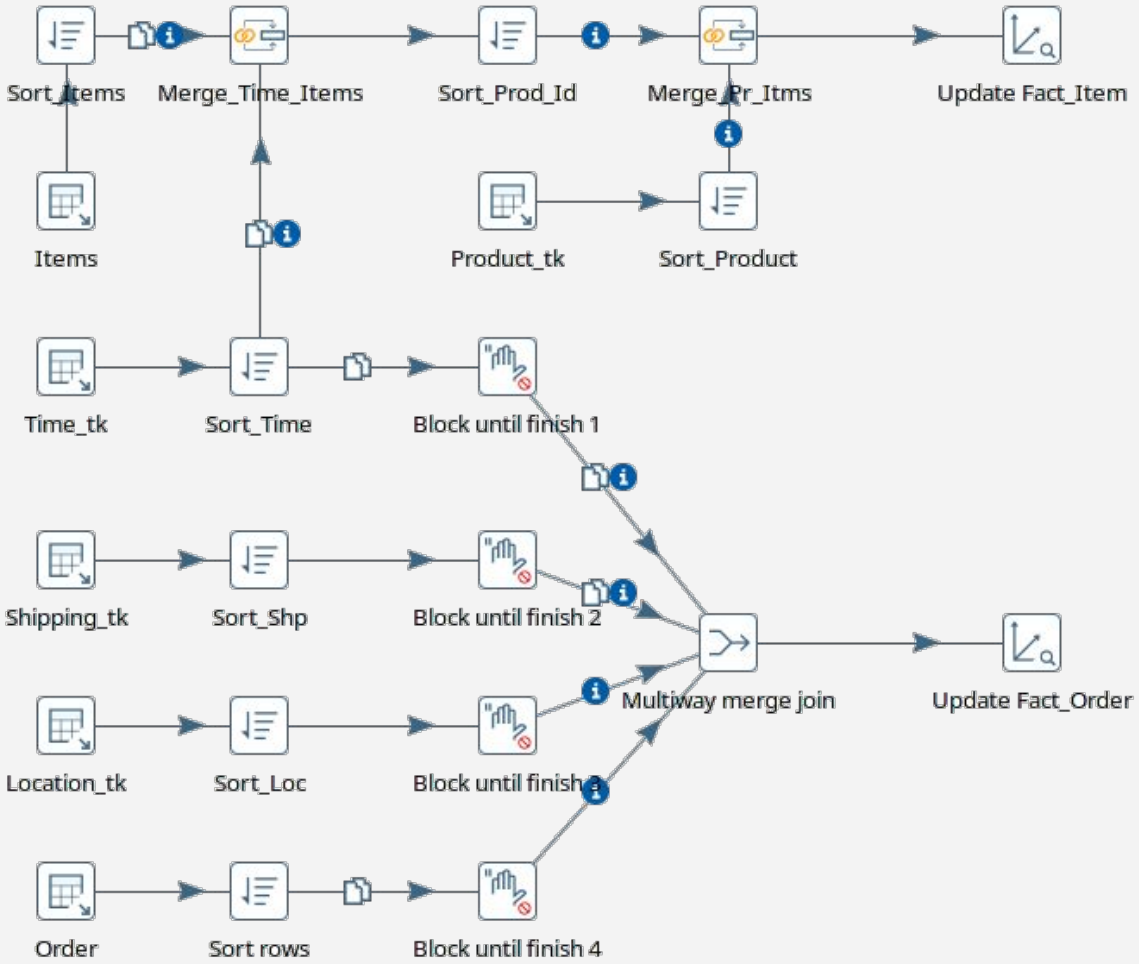
Product


product
sort_prod_id_
Merge join
is_active
Dimension lookup/update
CSV data
Select values
sort_prod_id
Unique rows

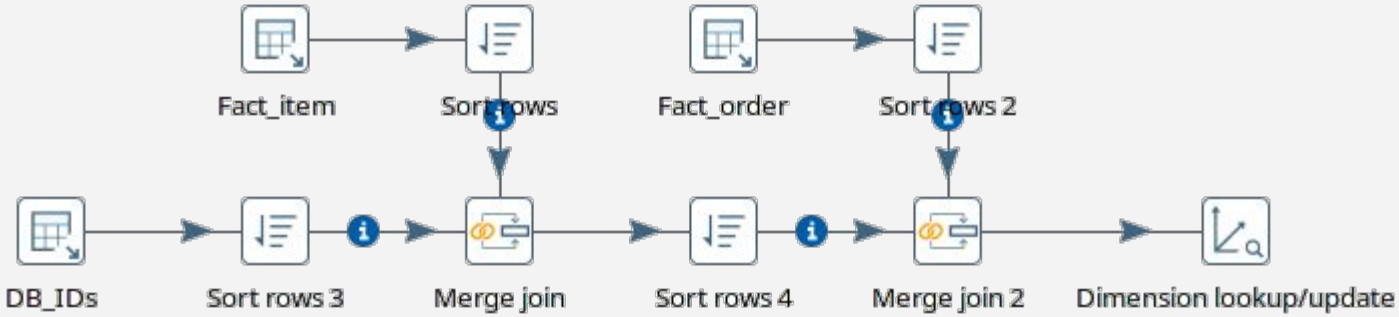
Location


Order_Geo
Sort rows 2
Customer_Geo
Sort rows 3
Department
Sort rows 4
Dimension lookup/update
Orders
Sort rows
Merge join
Sort rows 5
Merge join 2
Sort rows 6
Merge join 3
Select values

Facts


Sort_Items
Merge_Time_Items
Sort_Prod_Id
Merge_Pr_Itms
Update Fact_Item
Items
Product_tk
Sort_Product
Time_tk
Sort_Time
Block until finish 1
Shipping_tk
Sort_Shp
Block until finish 2
Multiway merge join
Update Fact_Order
Location_tk
Sort_Loc
Order
Sort rows
Block until finish 4

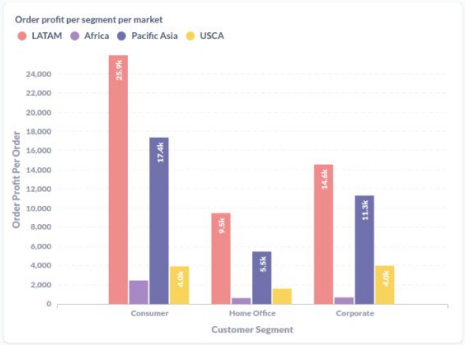
Reference

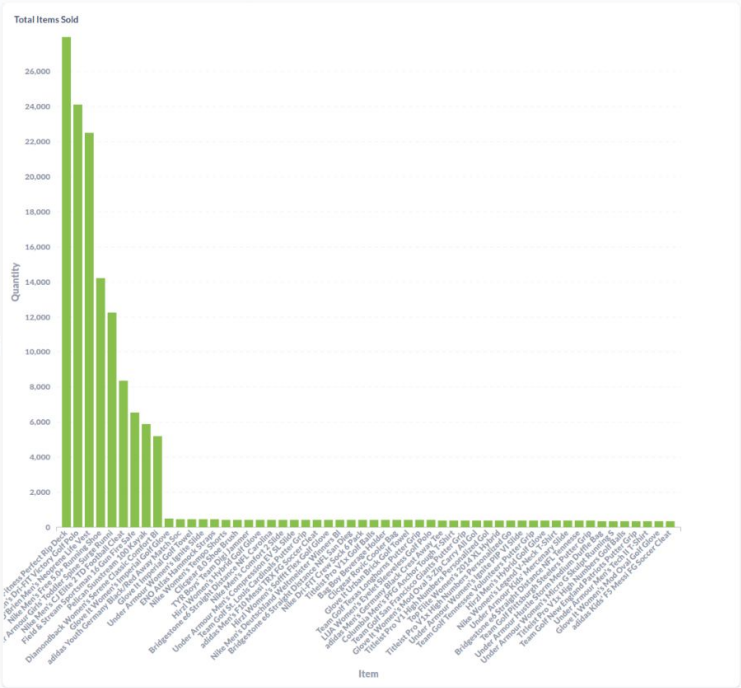


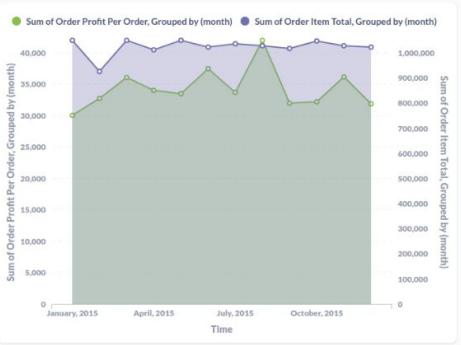

```
SELECT A.product_id, A.product_tk, A.date_from FROM
        data_co_schema_dim.Dim_Product as A
        LEFT JOIN
        (SELECT product_id, MAX(date_from) as date_from FROM
        data_co_schema_dim.Dim_Product GROUP BY product_id ORDER BY product_id) as B
        ON A.product_id = B.product_id
        WHERE A.date_from = B.date_from
        ORDER BY A.product_id;
```

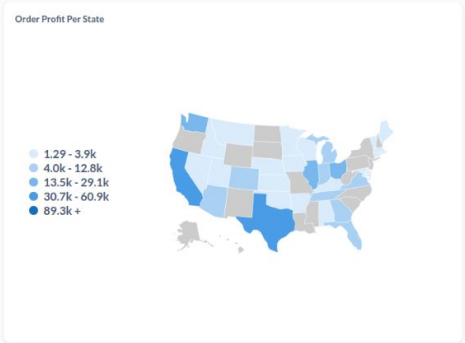
Vizualizacija Podataka

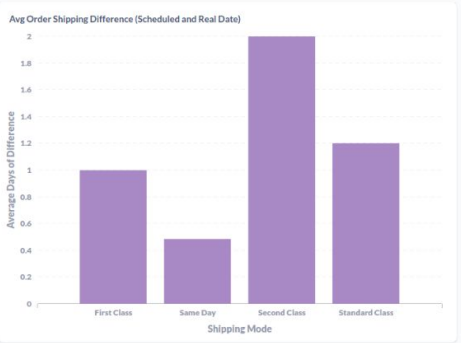
Metabase Dashboard za vizualizaciju podataka





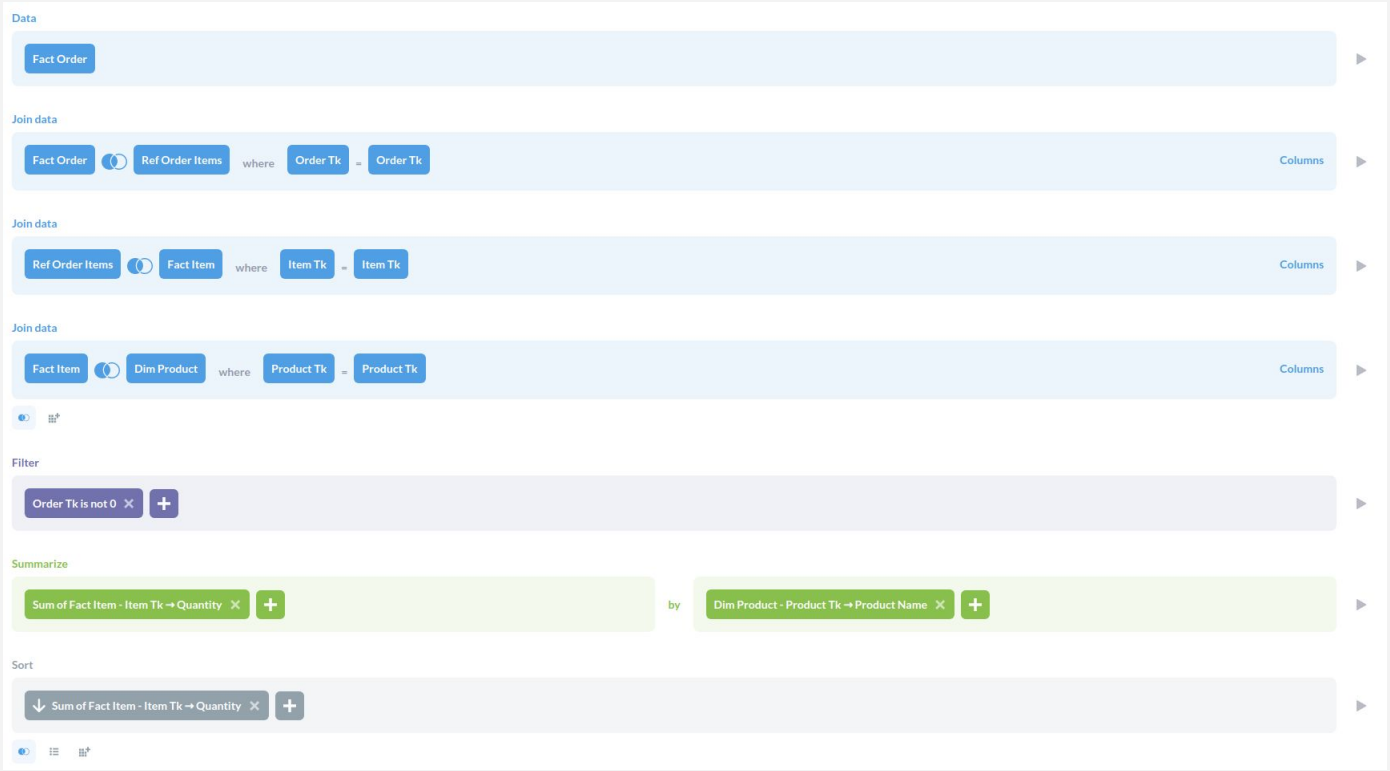






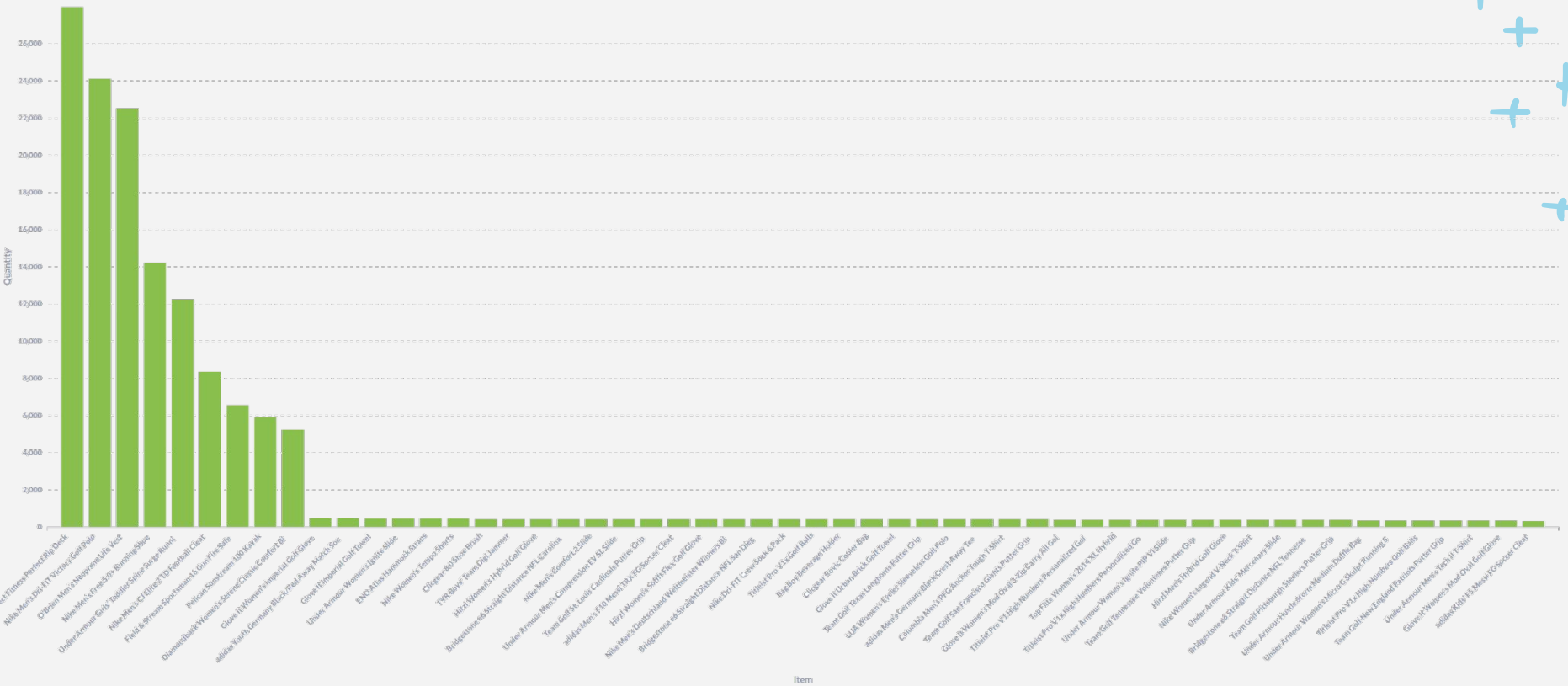
Vizualizacija Podataka

Metabase Editor



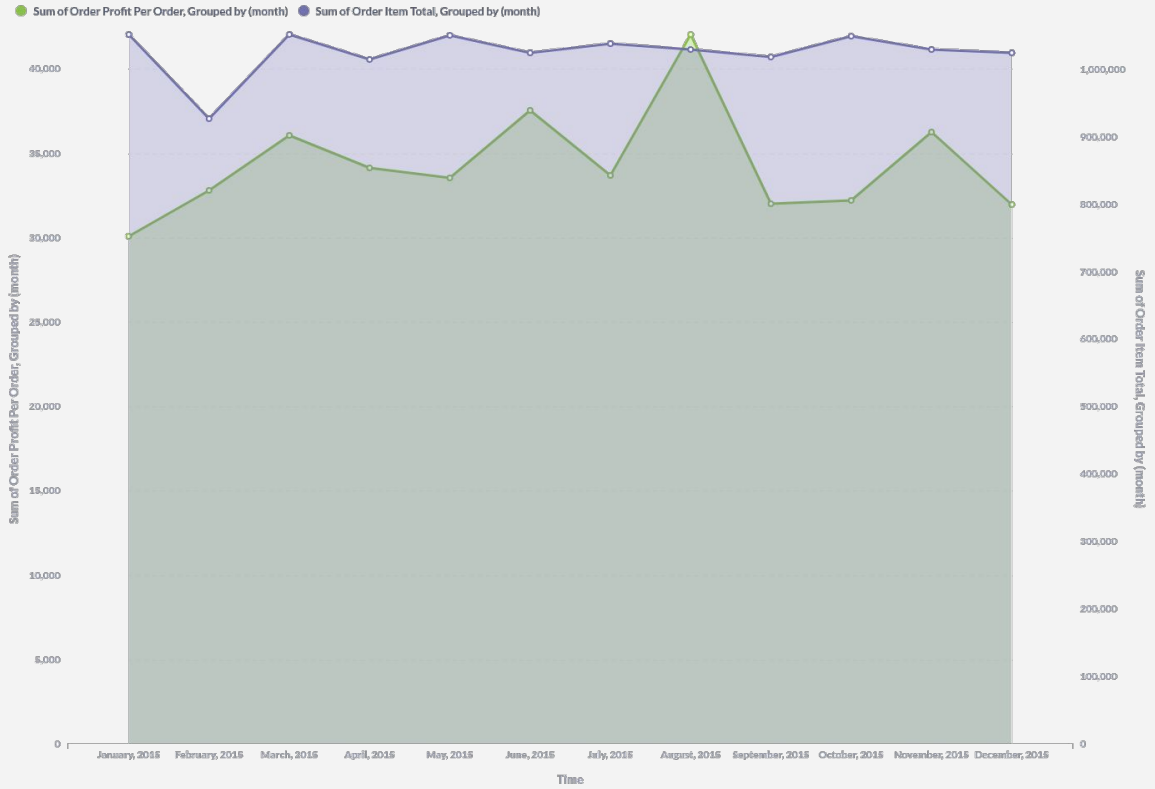
Vizualizacija Podataka

Total items sold



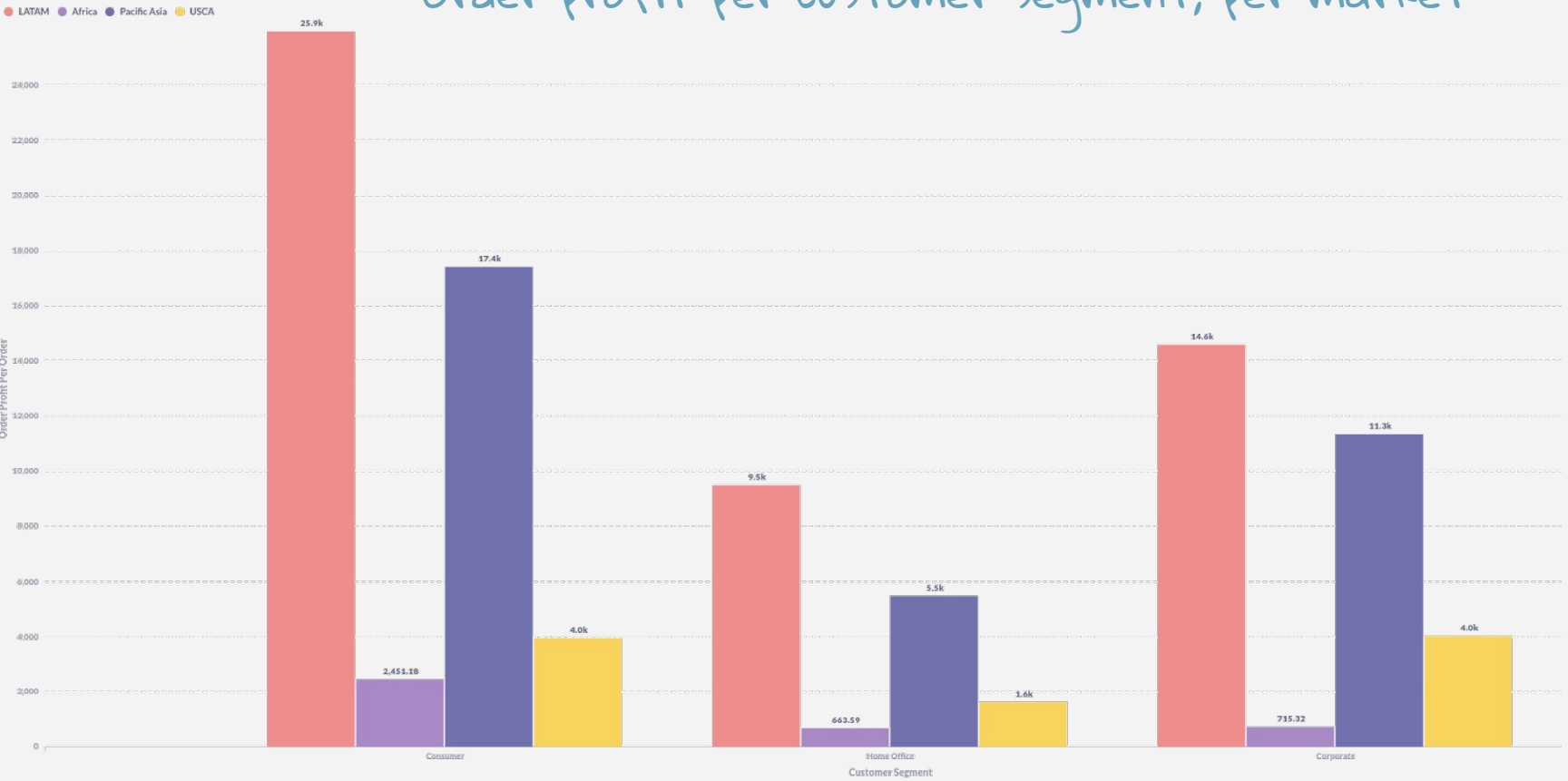
Vizualizacija Podataka

Ukupna cijena narudžbe i profit u nekom vremenskom intervalu



Vizualizacija Podataka

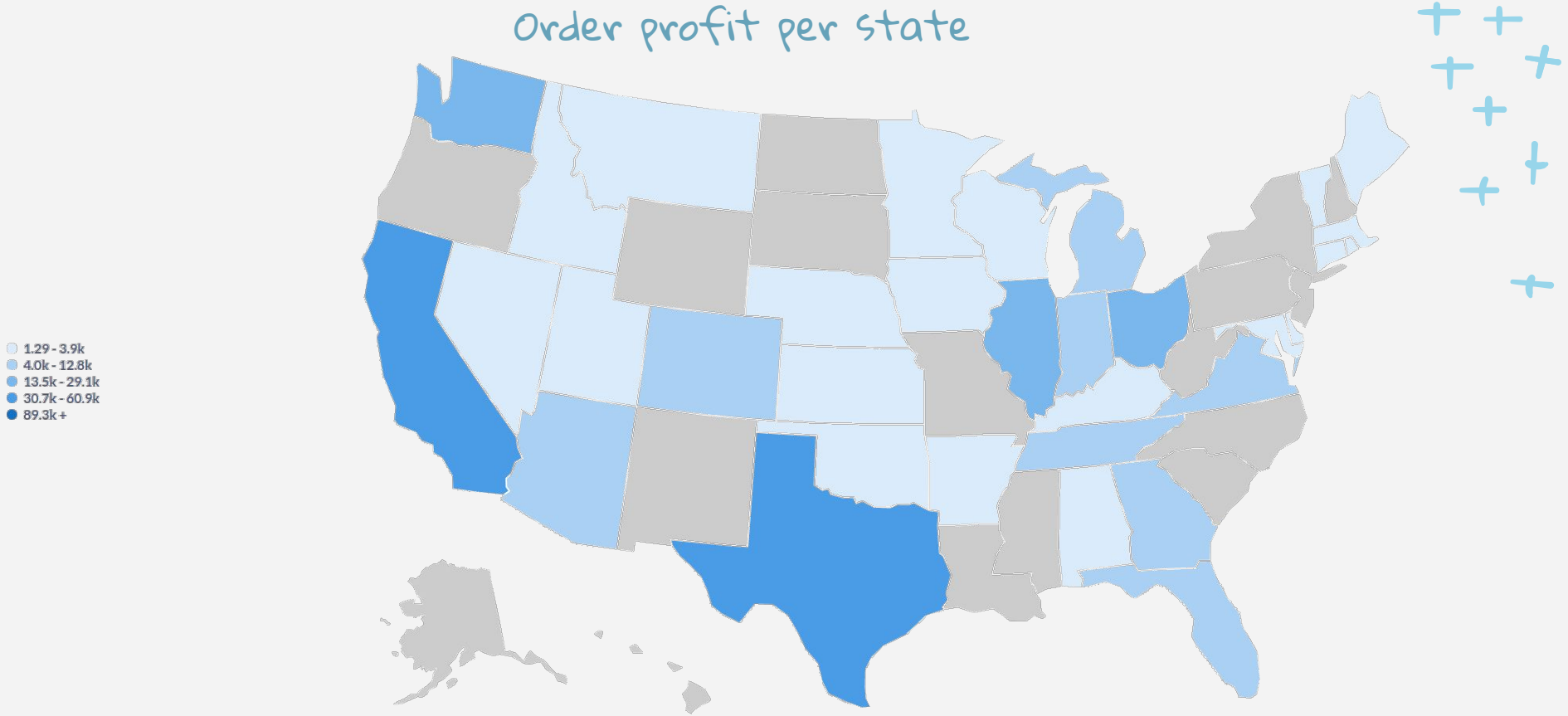
Order profit per customer segment, per market



Vizualizacija Podataka

Average difference of scheduled shipping days and real shipping days, per shipping mode

Vizualizacija Podataka



Zaključak

- Programsko okruženje
- Stvaranje Transakcijskog modela
- Stvaranje Dimenzijskog modela
- Pentaho punjenje
- Metabase Vizualizacija

Izvori :

- Metabase, Pristupljeno 10.5.2021, Dostupno na : https://github.com/metabase/metabase

- “Loading fact table with SCD type 2 dimension”, Pristupljeno 10.5.2021, Dostupno na : https://www.howtobuildsoftware.com/index.php/how-do/6cV/pentaho-data-warehouse-loading-fact-table-with-scd-type-2-dimension

- “How to store data in fact table with multiple products in an order in data warehouse”, Pristupljeno 10.5.2021, Dostupno na : https://stackoverflow.com/questions/35694914/how-to-store-data-in-fact-table-with-multiple-products-in-an-order-in-data-wareh