BID3000 Final Report

Business Intelligence and Data Warehouse

Group Exam – Autumn 2025

Universitetet i Sørøst-Norge

Name	Candidate Number	Student Number
Sarina Chui Ling Ng	7013	26609
Khalil Ahmad Jabbar	7023	265927

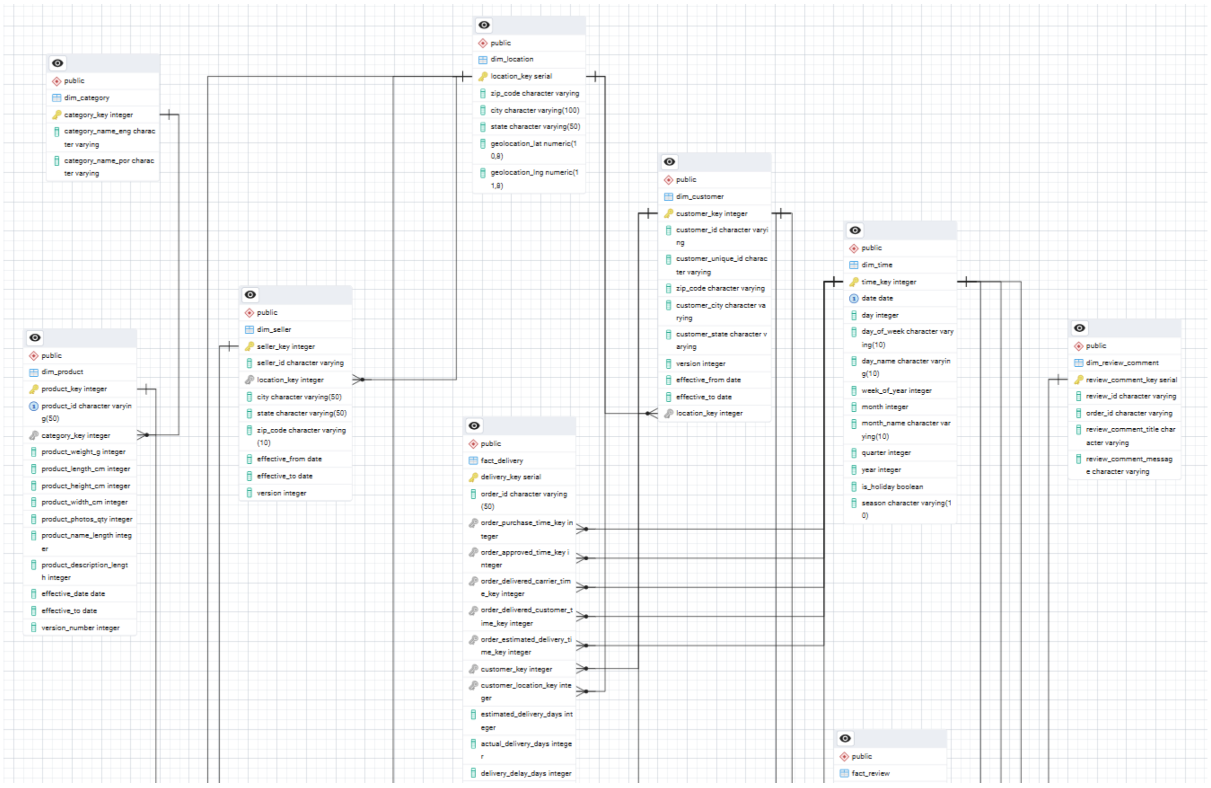
Contents

1. Warehouse Design Decisions

1.1 Star schema architectural overview

With a star schema, fact tables are in the center with the dimensional tables around, so it looks like a star (Sharda, Delen, & Turban, 2024). We have Kimball-style star schema, it is standard for an analytical standpoint in the industry (Kimball & Ross, 2013). Fact tables contain quantitative data about business events and transactions. Each row represents a business event in a specific grain. The foreign keys link to dimension tables. Dimension tables provide who, what and where? They can contain categorical and textual attributes used for grouping and filtering.

1.2 Our implementations (ERD)



public
dim_category
category_key integer
category_name_eng character varying
category_name_por character varying
public
dim_location
location_key serial
zip_code character varying
city character varying(100)
state character varying(50)
geolocation_lat numeric(10,8)
geolocation_lng numeric(11,8)
public
dim_customer
customer_key integer
customer_id character varying
customer_unique_id character varying
zip_code character varying
customer_city character varying
customer_state character varying
version integer
effective_from date
effective_to date
location_key integer
public
dim_time
time_key integer
date date
day integer
day_of_week character varying(10)
day_name character varying(10)
week_of_year integer
month integer
month_name character varying(10)
quarter integer
year integer
is_holiday boolean
season character varying(10)
public
dim_review_comment
review_comment_key serial
review_id character varying
order_id character varying
review_comment_title character varying
review_comment_message character varying
public
dim_product
product_key integer
product_id character varying(50)
category_key integer
product_weight_g integer
product_length_cm integer
product_height_cm integer
product_width_cm integer
product_photos_qty integer
product_name_length integer
product_description_length integer
effective_date date
effective_to date
version_number integer
public
dim_seller
seller_key integer
seller_id character varying
location_key integer
city character varying(50)
state character varying(50)
zip_code character varying(10)
effective_from date
effective_to date
version integer
public
fact_delivery
delivery_key serial
order_id character varying(50)
order_purchase_time_key integer
order_approved_time_key integer
order_delivered_carrier_time_key integer
order_delivered_customer_time_key integer
order_estimated_delivery_time_key integer
customer_key integer
customer_location_key integer
estimated_delivery_days integer
actual_delivery_days integer
delivery_delay_days integer
public
fact_review

The full ERD file is located in the Documentation and Screenshots folder.

We have four fact tables that represent different business processes and are fit with the task and to the brazilian e-commerce dataset by kaggle. (Data source: https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce)

Fact_order_lines = Sales transactions

- Individual product sales at line-item level. It contains freight_value, price and quantity.
- Links up to product, customer, seller, time and location.
- Relevant for total revenue, detailed sales metrics and ect.

Fact_delivery = Delivery logistics

- Tracks and evaluates logistics and fulfillment performance with delivery dates.
- Links up to customer, seller, time and location dimensions.
- Since it has transit time and on time metrics, it enables delivery performance analysis and customer satisfaction measurement.

Fact_payment = Payment processing

- Tracks payment transactions details including payment value, installment and payment type.
- Links up to customer and time dimension.
- Supports payment method analysis.

Fact_review = Customers reviews

- Customer reviews (1-5 stars). Links to product, customer, and time dimension.
- It is used for quality assessment and customer satisfaction analysis.

These fact tables are connected with 6-dimension tables that also fit with the dataset from kaggle.

dim_product = what product was sold?

- Contains product name, physical dimensions (weight, height, width and length), photo counts, description length and category reference.

dim_category = what type of product is it?

- Product categorization with category names in both Portuguese and English for reporting support.

dim_customer = who bought it?

- Customer master with historical tracking. Contains customer Ids, unique identifier, location reference and validity dates (effective_from, effective_to) for tracking customer changes over time.

dim_seller = who sold it?

- Seller information includes seller ID, location reference, and validity dates for historical seller tracking.

dim_location = where did the sale happen?

- Brazilian geographic hierarchy containing zip code, state, city and coordinates (latitude, longitude) for spatial analysis.
-

dim_time = when did it happen?

- Important calendar dimension with date components, year, month day of week and week.

dim_review_comment

- Collect all reviews that have a comment title or comment message
- Enable detailed analysis of reviews

This gives an overview of how a star schema works. Single hops in queries from fact to dim make them simple. Since it is denormalized we get fast performance.

1.3 Description of primary keys, foreign keys and indexing strategies

1.3.1 The primary keys for each dimension table

dim_category: **category_key** (integer) – Auto-incrementing surrogate key

dim_customer: **customer_key** (integer) – Auto-incrementing surrogate key, with natural key customer ID from source

dim_location: **location_key** (integer) – surrogate key for geographic data

dim_product: **product_key (**integer) – surrogate key, with natural key product id from source

dim_seller: **seller_key** (integer) – surrogate key,with natural key seller id from source

dim_time: **time_key** (integer) – surrogate key

dim_review_comment: **review_comment_key** (integer) – surrogate key, with natural key which is a combination of review_id and order_id

1.3.2 The primary keys for each fact table

Fact_order_lines: **order_line_key (**integer) – surrogate key , with natural key which is a combination of order_item_id and order_id

Fact_payment: **payment_key** (integer) – surrogate key, with a natural key which is a combination of order_id and payment sequential

Fact_review: **review_key** (integer) – surrogate key, with a natural key which is a combination of review_id and order_id

Fact_delivery: **delivery_key** (integer) – surrogate key, with a natural key order_id from source

1.3.3 Foreign keys for fact tables

Fact_order_lines links to

- Dim_product (product_key)
- Dim_seller (seller_key)
- Dim_time (time_key)
- Dim_location (location_key)

Fact_delivery links to

- Dim_customer (customer_key)
- Dim_seller (seller_key)
- Dim_time(time_key)
- Dim_location (location_key)

Fact_payemnt link to

- Dim_customer (customer_key)
- Dim_time (time_key)

Fact_review links to

- Dim_product (product_key)
- Dim_customer (customer_key)
- dim_time (time_key)

Dimension foreign keys

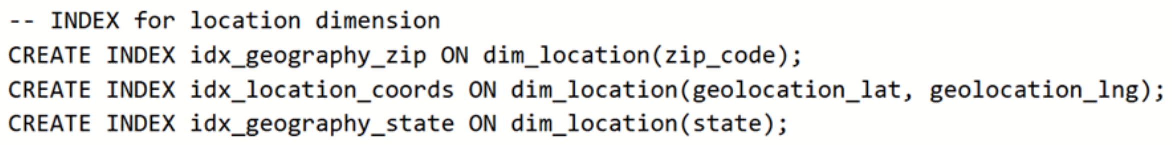
Dim_seller – dim_locaiton (location_key)

Dim_customer – dim_location (location_key)

Dim_product – dim_category (category_key)

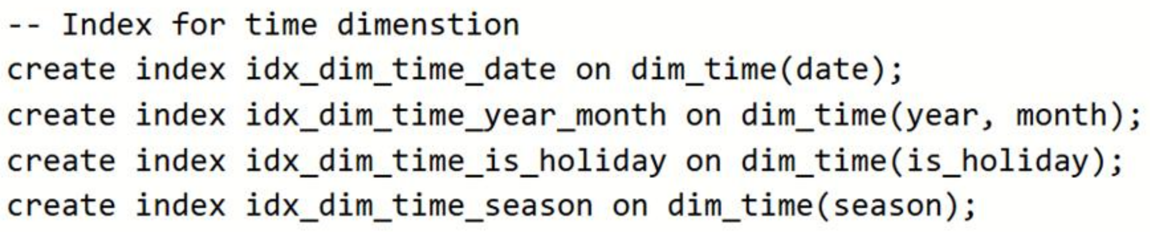
1.3.4 Indexing Strategies

Our indexing strategy optimize analytical query performance by placing indexes on frequently filtered columns, all foreign keys, and attributes used in common query patterns.



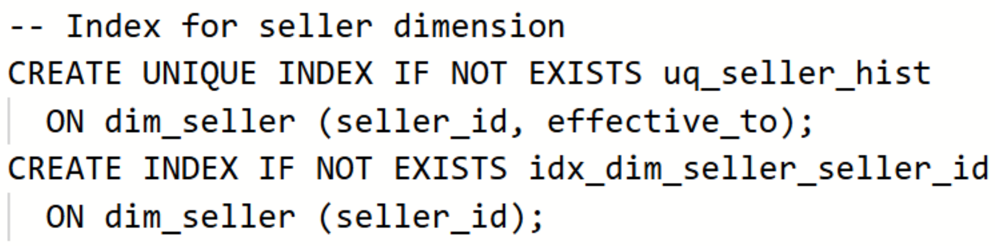
```
-- INDEX for location dimension
CREATE INDEX idx_geography_zip ON dim_location(zip_code);
CREATE INDEX idx_location_coords ON dim_location(geolocation_lat, geolocation_lng);
CREATE INDEX idx_geography_state ON dim_location(state);
```

- Zip code index support location lookups
- Coordinates enable geographical radius searches
- State index facilitates regional aggregations.



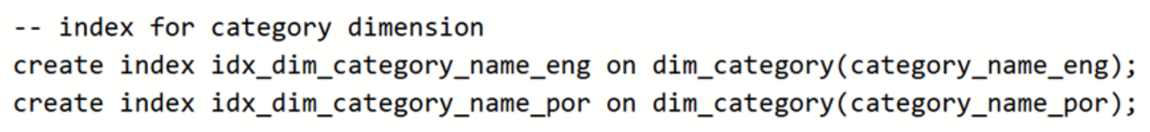
```
-- Index for time dimenstion
create index idx_dim_time_date on dim_time(date);
create index idx_dim_time_year_month on dim_time(year, month);
create index idx_dim_time_is_holiday on dim_time(is_holiday);
create index idx_dim_time_season on dim_time(season);
```

- Date index supports direct date filtering.
- Year-month index monthly and quarterly aggregations
- Holiday indexes enable us to analyze patterns vs non holiday
- Season indexes enable us to analyze patterns through seasons



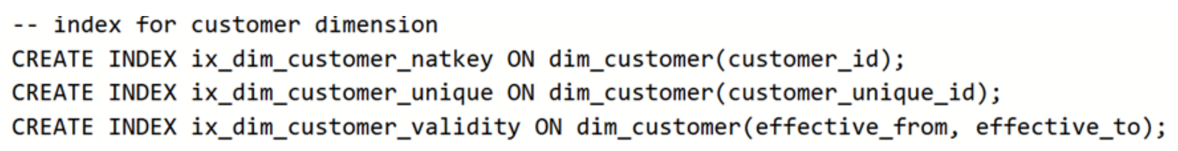
```
-- Index for seller dimension
CREATE UNIQUE INDEX IF NOT EXISTS uq_seller_hist
  ON dim_seller (seller_id, effective_to);
CREATE INDEX IF NOT EXISTS idx_dim_seller_seller_id
  ON dim_seller (seller_id);
```

- Seller_id index supports natural key lookups and source system joins.
- SCD Type 2 integrity is forced for unique historical analysis



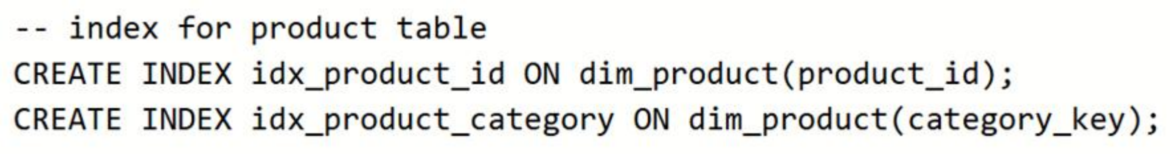
```
-- index for category dimension
create index idx_dim_category_name_eng on dim_category(category_name_eng);
create index idx_dim_category_name_por on dim_category(category_name_por);
```

- Index support categorical filtering in both English and Portuguese
- Easier for international BI reports



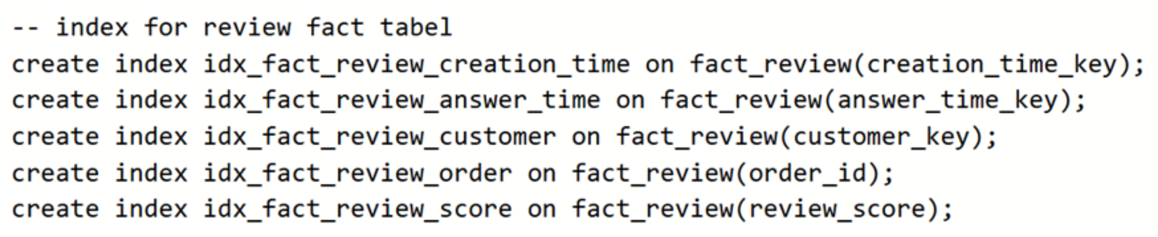
```
-- index for customer dimension
CREATE INDEX ix_dim_customer_natkey ON dim_customer(customer_id);
CREATE INDEX ix_dim_customer_unique ON dim_customer(customer_unique_id);
CREATE INDEX ix_dim_customer_validity ON dim_customer(effective_from, effective_to);
```

- Unique id index support customer deduplication ‘
- Natural key index enables source system reconciliation



```
-- index for product table
CREATE INDEX idx_product_id ON dim_product(product_id);
CREATE INDEX idx_product_category ON dim_product(category_key);
```

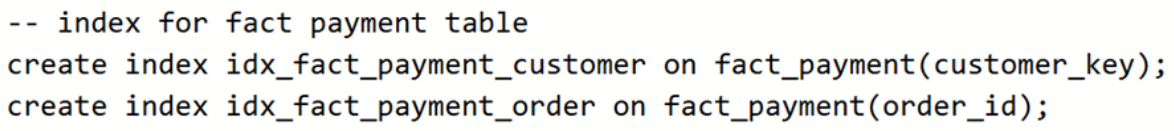
- Natural key lookups and source system reconciliation
- Optimizes product-to-category joins and category level filtering.



```
-- index for review fact tabel
create index idx_fact_review_creation_time on fact_review(creation_time_key);
create index idx_fact_review_answer_time on fact_review(answer_time_key);
create index idx_fact_review_customer on fact_review(customer_key);
create index idx_fact_review_order on fact_review(order_id);
create index idx_fact_review_score on fact_review(review_score);
```

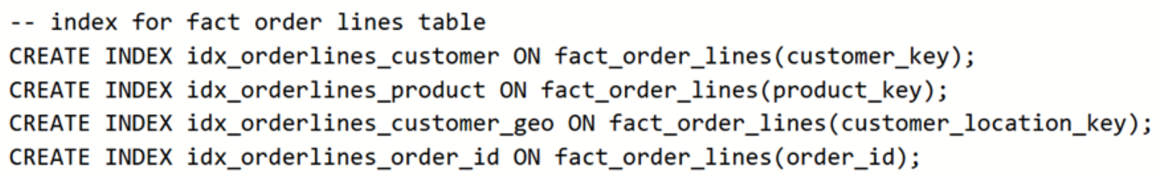
- Index support time-bases analysis, when reviews were created and answered
- Customer index enables customer review history queries
- Order_id index links reviews to transaction

- Score index optimizes filtering by rating



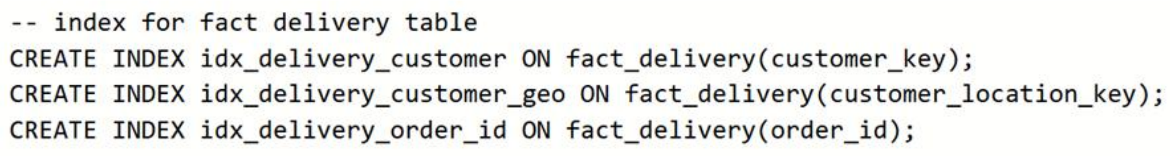
```sql
-- index for fact payment table
create index idx_fact_payment_customer on fact_payment(customer_key);
create index idx_fact_payment_order on fact_payment(order_id);
```

- Customer index enables customer payment pattern analysis and payment history queries.
- Order_id index links payments to orders for financial reconciliation and supports queries joining payment data with other order-related facts.



```sql
-- index for fact order lines table
CREATE INDEX idx_orderlines_customer ON fact_order_lines(customer_key);
CREATE INDEX idx_orderlines_product ON fact_order_lines(product_key);
CREATE INDEX idx_orderlines_customer_geo ON fact_order_lines(customer_location_key);
CREATE INDEX idx_orderlines_order_id ON fact_order_lines(order_id);
```

- Product and customer indexes optimize most common joins to dimension tables
- Geographical indexes support regional analysis
- Order_id index enables order-based queries and joins to other fact tables (payment, delivery, review)



```sql
-- index for fact delivery table
CREATE INDEX idx_delivery_customer ON fact_delivery(customer_key);
CREATE INDEX idx_delivery_customer_geo ON fact_delivery(customer_location_key);
CREATE INDEX idx_delivery_order_id ON fact_delivery(order_id);
```

- Customer index optimizes customer delivery history queries
- Geographical index supports regional delivery performance analysis

- Order_id index links deliveries to orders and enables joins with other fact tables for order-to-delivery analysis.

1.4 Design rationale and grain definition

1.4.1 Fact table 1: fact_order_lines

The grain definition is one row per individual product within an order.

Each row represents one product type purchased in one specific order, if a customer orders three different products in one order, it will mean three rows in the table. Moreover, if a customer orders 2 units of a product, it will be one row, but two in quantity.

The grain is chosen with Kimball mythology; we store data at the lowest level possible. We can also aggregate levels as we like. This makes sure that we have flexibility in the fact table. We can also analyze data in detail, for example the revenue based on items and geography, which items sell more than one unit in average order. The possibilities are many for analyzing, which makes this grain great.

We can furthermore calculate and count, which is essential for data analysis.

- **Quantity** - units of product sold
- **Price** - unit price for product
- **Freight_value** - shipping cost

1.4.2 Fact table 2: fact_delivery

The grain definition is one row per delivary process for each complete order

Every order gets one record of delivery; it will track the entire road from confirmed to delivery.

This grain is very important, as we can aim for the goal of optimizing the logistics. With data we can see if the customers are getting their deliveries on time or not. This grain is totally different from sales; it is meant for logistics. We can analyze from both seller and geographical standpoint, who is fastest and slowest? How many deliveries are due? The possibilities for analyzing are many.

- **Estimated_delivary_days –** Estimation for delivery days
- **Actual_delivary_days –** The actual delivery days
- **Delivery_delay_days –** Delays for the delivery days
- **Total_freight_value –** Total shipping cost for entire order

1.4.3 Fact table 3: fact_payment

The grain definition is one row per payment transaction

Each row represents a payment transaction. In Brazilian e-commerce, one order can have multiple payment methods. This means one order can generate multiple rows in the fact table.

The grain is essential because payment patterns in Brazil are unique – installment payments are very common. With the grain we can analyze payment behavior in detail. Which payment methods are preferred? How many installments do customers usually make? The possibilities are many.

- **Payment_value –** Amount paid in transaction
- **Payment_installments –** Number of installments for this payment
- **Payment_sequential** – The sequence number if multiple payments exist for one order
- **Payment_type –** Method used, (credit card. Debit card, voucher, boleto)

1.4.4 Fact table 4: fact_review

The grain definition is one row per customer review of a product.

Each row represents one review written by one customer about one product. Reviews are optional, meaning not every purchase generates a review. One can accumulate many reviews over time from different customers.

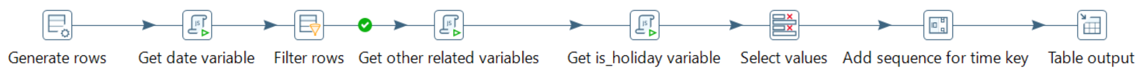
This grain is important for quality control and customer satisfaction analysis. With this data we can identify problematic products, understand what customers appreciate, and see how delivery performance affects customer satisfaction. We can analyze review patterns. Which products get the best ratings? Do delivery days correlate with poor reviews? The possibilities are many.

- **Review_Score –** Customer rating 1-5
- **Review_count –** Can be counted to see total reviews per product
- **Avarage_review_score –** Can be calculated per product, category or seller
- **Review_response_time –** Time between review creation and seller response

2. ETL process

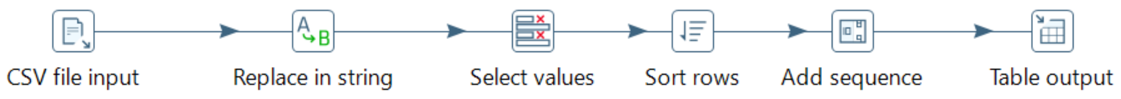
We use Pentaho Data Integration (PDI) to build ETL transformations. We have created a separate transformation for each table in our database, resulting in a total of 11 transformations. Below, we will show screenshots of each transformation, along with step by step explanations.

2.1 dim_time table transformation



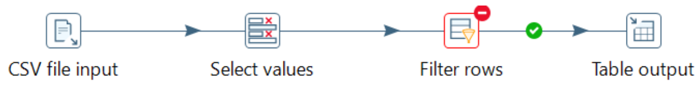
This is the transformation for the dim_time table. It begins by generating a row for the date field, then uses JavaScript to create date variables for the period from 2016-01-01 to 2020-12-31, effectively generating all the required date records. Next, we filter out any rows with null values, and use JavaScript to derive additional variables such as month, year, is_holiday, and others. After that, we use the "Select values" step to choose the final set of columns needed in the database. Finally, we create surrogate key by adding a sequence and write the data into the database.

2.2 dim_category table transformation



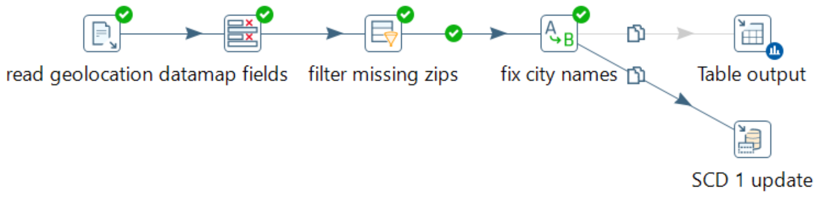
This is the transformation for the dim_category table. It begins by reading data from a CSV file as the source. Next, as part of the data cleansing process, we select UTF-8 as the file encoding to ensure that non-English and special characters are read correctly. We also use the "Replace in String" step to replace underscores with spaces and normalize the text fields. Finally, we select the required columns for the database, sort the rows, add a sequenced surrogate key, and write the data into the database.

2.3 dim_review_comments table transformation



It begins by reading data from a CSV file as the source. UTF-8 is selected as the file encoding method to ensure data quality. Next, it selects the columns needed for the database, filters out rows that do not have review comments, and finally writes the data into the database.

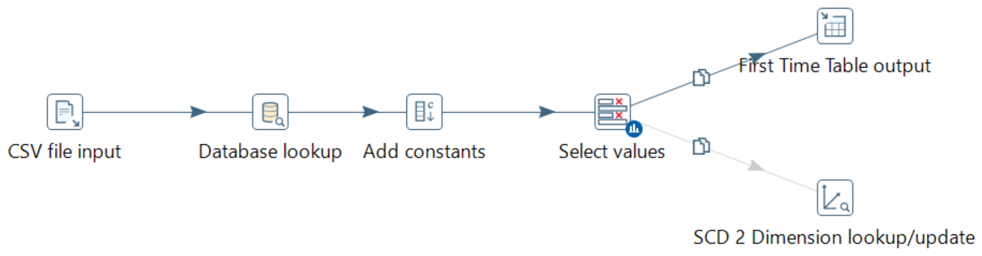
2.4 dim_location table transformation



This transformation is quite similar to the previous one in terms of extracting data and selecting columns. However, it filters out rows where the zip code is null and normalizes the data by handling special characters before writing to the table output. SCD Type 1 is implemented in this transformation. We perform a lookup on the ZIP code and update the corresponding state, city, and latitude values accordingly.

During the initial data load, we use the Table Output step for efficiency. Afterwards, if there are any changes in the dataset, we can disable the Table Output step and use the SCD Type 1 step to update the existing records.

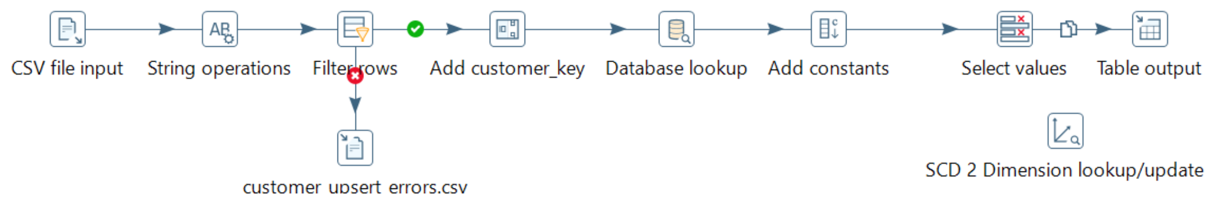
2.5 dim_seller table transformation



This transformation starts by reading data from a CSV file as the source. Next, it performs a database lookup to retrieve the location_key based on the zip code. We then add constants such as effective_from, effective_to, and version to support SCD updates. After that, the final columns are selected, and the data is written to the target table. To enable SCD Type 2, we implement an SCD 2 dimension lookup. In this lookup, the process searches for rows with the same seller_id and creates a new row with updated information, such as zip code, location key, and other relevant fields. The previous row remains in the table but is deactivated by updating the effective_to date.

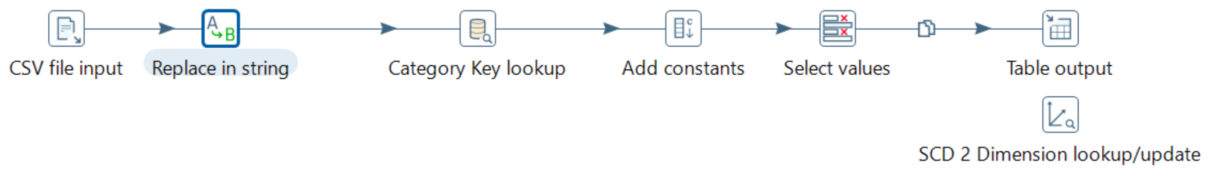
The “First Time Table Output” and “SCD 2 Dimension Lookup” steps conflict with each other when used simultaneously in the same transformation. Therefore, we use “First Time Table Output” to load the data initially, delete the step, and then apply the “SCD 2 Dimension Lookup” step for subsequent updates.

2.6 dim_customer tabel transformation



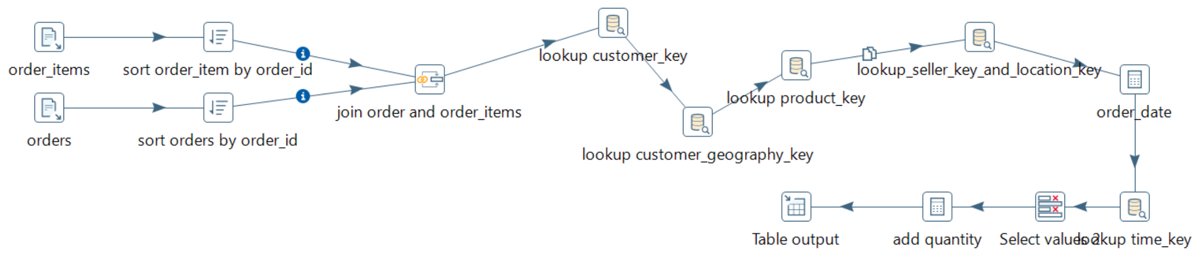
This transformation is very similar to the previous one. However, it includes a “String Operation” step to normalize the data by trimming values and unifying the case format to ensure data quality. It also handles missing data by filtering out rows that do not have customer_unique_id, city, or post_code. In addition, it implements error handling by sending rows with empty data to a separate CSV file. SCD Type 2 is implemented in this transformation.

2.7 dim_product table transformation



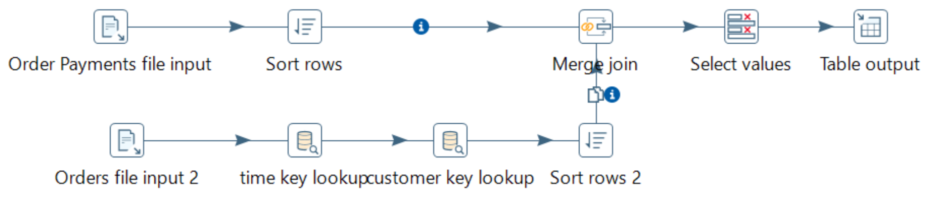
This transformation is very similar to the previous one. In addition, it includes UTF-8 as file encoding method and a “Replace in String” step to replace underscores with spaces, supporting data cleansing. It also performs a category key lookup by comparing category names. SCD Type 2 is implemented in this transformation.

2.8 fact_order_lines table transformation



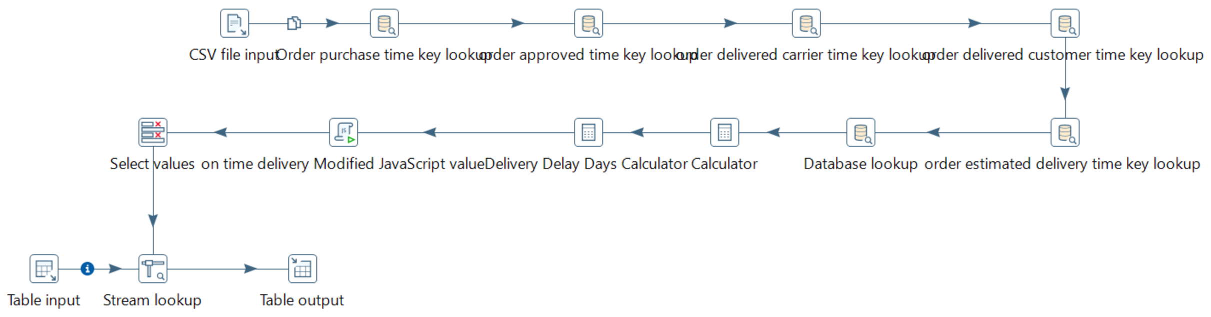
This transformation starts by extracting data from two CSV files. It then sorts the data by order_id and joins the two tables together. Next, it performs lookups for customer_key, location_key, seller_key, time_key, and other relevant fields. It also adds quantity to the order before writing the data to the target table.

2.9 fact_payment table transformation



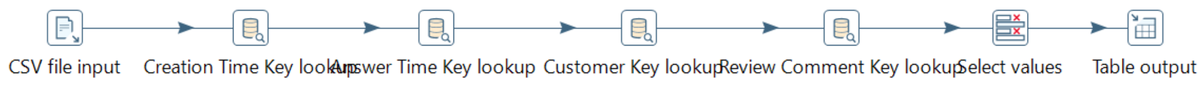
This transformation is similar to the previous one. It extracts data from two CSV files, sorts them, and joins the tables together. It then selects the columns to be included and writes the data to the target table.

2.10 fact_delivery table transformation


CSV file input
Order purchase time key lookup
order approved time key lookup
order delivered carrier time key lookup
order delivered customer time key lookup
order estimated delivery time key lookup
Database lookup
Calculator
Delivery Delay Days Calculator
Modified JavaScript value
on time delivery
Select values
Table input
Stream lookup
Table output

This transformation starts by extracting data from a CSV file. It performs multiple lookups for different time_key values and calculates delivery delay days by comparing the actual delivery date with the estimated delivery date. The data is then enriched by looking up the freight value in the database using the order_id, and finally, the data is written to the target table.

2.11 fact_review table transformation



This transformation starts by extracting data from a CSV file. It then performs lookups for time_key, customer_key, and comment_key. Finally, it selects the columns to be included and writes the data to the database.

3. Data quality issues identified and resolution strategies

During the ETL process, we identified and addressed several data quality issues. The first issue was inconsistent formatting, particularly in the category and location fields. This was mainly caused by the presence of special characters and non-English characters. To resolve this, we set the encoding format to UTF-8 to ensure all data is loaded correctly. Additionally, we applied string operations to further enhance data quality, for example, normalizing text to uppercase/lowercase, removing underscores, and trimming unnecessary spaces.

Another issue we encountered was null values. Rows missing critical fields such as customer_unique_id or zip_code were filtered out and exported to a separate CSV file containing error records. Null values were also handled based on the context of each table. For example, in the dim_review_comment table, we only included reviews that contained actual comments. Therefore, we filtered out any reviews with empty comment titles or comment fields to maintain data quality.

Another issue we encountered was a referential integrity issue. After loading the data, we found some null values in the category_key field of the dim_product table. This occurred because certain categories were missing from the dim_category table, resulting in missing foreign key references in the product data. To resolve this, we added the missing categories to the dim_category table, generated category keys for them, and updated the corresponding category keys in the dim_product table. This ensured referential integrity between the two tables.

4. Performance considerations and optimization techniques applied

4.1 Database performance optimizations with index strategy

Indexing was implemented to optimize query performance across all fact and dimension tables. All primary keys are automatically indexed, and strategic indexes are placed on.

- **Foreign keys in fact tables –** Optimize join operations between facts and dimensions
- **Frequently filtered columns –** Indexes on state, date, zip code and review score support common WHERE clause patterns.
- **Geographical analysis –** Composite index on lat og lng supports radius-based queries for regional analysis.

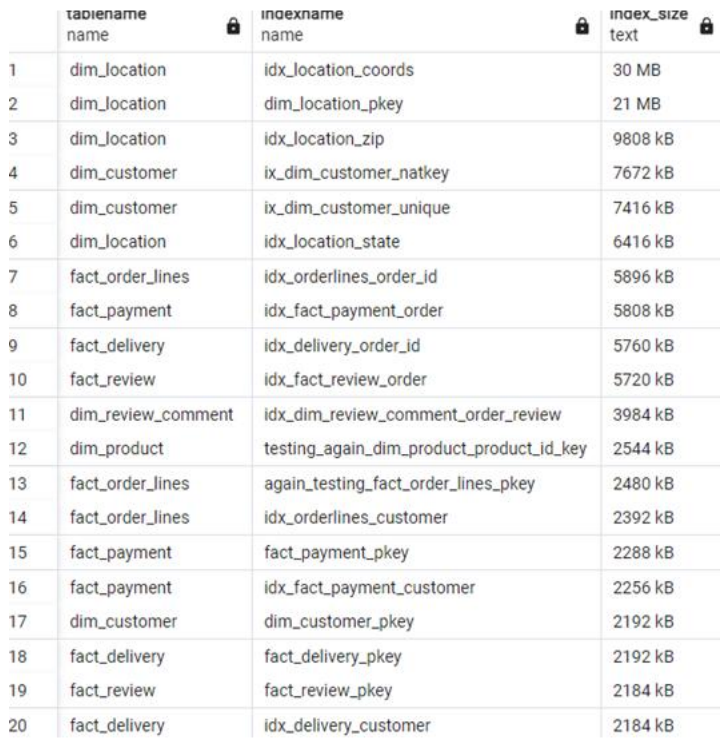
Indexes add approximately 50% storage overhead (13mb indexes on 12 mb fact_order_lines data). Which is okay for the dramatic query performance improvement achieved.

Efficient data types minimize storage and improve query performance. Example use of serial, integer, decimal, varchar and dates.

4.2 Constraints implementation

Check constraint on geographic coordinates ensure data validity (latitude -90 to 90, longitude -180 to 180)

UNIQUE constraint on natural keys (customer-id, product_id) prevents duplicates at source.



	tablename name	indexname name	index_size text
1	dim_location	idx_location_coords	30 MB
2	dim_location	dim_location_pkey	21 MB
3	dim_location	idx_location_zip	9808 kB
4	dim_customer	ix_dim_customer_natkey	7672 kB
5	dim_customer	ix_dim_customer_unique	7416 kB
6	dim_location	idx_location_state	6416 kB
7	fact_order_lines	idx_orderlines_order_id	5896 kB
8	fact_payment	idx_fact_payment_order	5808 kB
9	fact_delivery	idx_delivery_order_id	5760 kB
10	fact_review	idx_fact_review_order	5720 kB
11	dim_review_comment	idx_dim_review_comment_order_review	3984 kB
12	dim_product	testing_again_dim_product_product_id_key	2544 kB
13	fact_order_lines	again_testing_fact_order_lines_pkey	2480 kB
14	fact_order_lines	idx_orderlines_customer	2392 kB
15	fact_payment	fact_payment_pkey	2288 kB
16	fact_payment	idx_fact_payment_customer	2256 kB
17	dim_customer	dim_customer_pkey	2192 kB
18	fact_delivery	fact_delivery_pkey	2192 kB
19	fact_review	fact_review_pkey	2184 kB
20	fact_delivery	idx_delivery_customer	2184 kB

4.3 ETL Performance Optimization (Pentaho)

Table output steps use **batch insert mode** (use batch=Y), which groups multiple insert statements into single batch operation. This reduces

- Transaction overhead
- Overall load time by 5-10x compared to row-by-row inserts.

Commit size optimization

- 1000 rows are optimal for our dataset size – fast commits without excessive memory usage. 100 is too low (transaction overhead), 100 000 is too large (memory pressure)

Row buffer configuration

Row buffer size: 10000 rows enable efficient streaming between transformation steps.

- Reduces I/O wait time between steps

CSV Input Optimization

CSV buffer size, 50000 bytes optimizes file reading.

- Enables efficient reading of large csv files.

Data Quality Filtering

The filter rows step eliminates invalid data before database insert.

- Invalid rows logged to error files for review
- Reduces ETL rollback and failure
- NOT NULL on critical fields such as Pk's is important for data integrity

5. SQL Analytical Queries - Query results and analytical outputs

We have developed a total of 10 analytical SQL queries to demonstrate key dimensional modeling concepts. These queries help us better understand business performance, for example, by analyzing year-over-year revenue growth, customer lifetime value, 7 day moving averages of daily revenue, and other performance KPI calculations. Detailed query results can be found in the database folder. The following are analytical output to each query:

1A. The output shows year over year revenue growth by month. Revenue increased significantly in 2017, with growth of more than 2,000% in September and more than 60,000% in December compared to the same month in the previous year. Growth in 2018 slowed down, eventually reaching negative growth in September 2018.

1B. The output compares each month's average revenue to the overall average revenue, serving as an indicator of how each month performs relative to the average month. It reveals a seasonal pattern, with November performing significantly better than the rest of the year. In contrast, September, October, January, and February show weaker sales. Businesses can use this insight to optimize marketing timing and inventory planning.

2A. This table shows total revenue by year, month, and product category. It indicates that business revenue increased each year from 2016 to 2018. It also highlights popular categories such as Watches & Gifts, Health & Beauty, Sports & Leisure, and computers & accessories. It is easy to compare sales performance across different months as well.

2B. This is a hierarchical dimension analysis showing product-level revenue with subtotals per category and a grand total. The top three categories generating the highest revenue over the entire period are Health & Beauty, Watches & Gifts, and Bed, Bath & Table. The analysis also makes it possible to see which products generate the most revenue within each category. This provides valuable insights into popular products and can inform product purchasing and inventory decisions.

3A. This query ranks customers by revenue within each state and assigns them to percentile bands (1–100) based on their revenue. It displays each customer’s total revenue along with their rank within the state, making it easy to identify top performing customers in each region.

3B. This query shows the 7 day moving average of daily revenue as well as the year to date cumulative revenue. The days with the highest 7 day moving averages are 11/29/2017, 11/30/2017, 11/28/2017, 11/27/2017, and 11/26/2017, indicating that late November (likely the Black Friday period) represents a peak revenue period. The company can use this insight to plan marketing campaigns, allocate additional staff, and increase inventory accordingly.

4A. This query identifies products that have more than three late deliveries, an average review score below 3, and were sold to customers in the state “SP.” The results are a list of product IDs that highlight products likely to have delivery issues and cause customer dissatisfaction in this specific region. The company can use this information to inspect these products, address the underlying problems, and improve customer satisfaction in “SP”.

4B. This query identifies products with above average prices within their category. The result highlights premium products in each category. It can also help detect mispriced products. The company can use this price information to optimize its pricing structure, adjust discounts, and make more informed pricing strategy decisions.

5A. This query identifies the top 20 customers by lifetime revenue, along with their order behavior and location. The results show that most top customers made only one order, a few made two to three orders, and one customer made four orders. Five of these customers are from the SP region, suggesting an opportunity to target marketing campaigns in SP and to apply different sales strategies, such as retention and upselling, based on the customers' purchasing behavior.

5B. This query calculates monthly key performance indicators (KPIs) including order volume, average order value, on-time delivery rate, and average review score. The results show that the number of orders grew rapidly in 2017 and stabilized in 2018. Average order value and on-time delivery rates remained relatively stable over time. Review scores were notably lower in November and December 2017, likely due to the higher volume of orders during these peak months. This suggests that the company should optimize its logistics operations during peak seasons to maintain customer satisfaction.

6. Python Analytics Integration - Business interpretation of findings

We connected a Python script to a PostgreSQL data warehouse hosted on Azure using a dummy account called "test." To prepare the data for analysis, we structured the dataset at the order_id level, with each row containing relevant information such as order_date, customer_key, quantity, product_value, freight_value, and more. The Python script then generated the data into a file named analysis_data.csv. We conducted descriptive, predictive, and prescriptive analyses, and the business interpretations of the findings are presented as follows:

6.1 Descriptive analysis – on time delivery analysis

This refers to "Appendix A". It shows a descriptive statistic containing mean, standard deviation, minimum, 25th percentile, median, 75th percentile and maximum for "items","product_value","freight_value","order_total_value", "weight_kg","avg_distance_km","paid_total","avg_review_score". These statistics summarize the distribution of our numeric data. The data shows that, in general, orders contain only one item, are lightweight, and cover relatively short distances. However, there are occasional orders with very high values, heavy weights, and long shipping distances, resulting in large standard deviations and high maximum values, particularly for product value and average distance.

The correlation table indicates that the average review score has the highest correlation with delivery delay, suggesting that delays are likely to lead to lower review scores. Improving delivery performance could therefore have a significant positive impact on customer review ratings.

6.2 Descriptive analysis – regional sales analysis

This refers to Appendix B, which shows the top 10 states by revenue. SP, RG, and MG are the top three states. This suggests that the company could consider expanding its services in semi-established regions such as RG and MG to further increase revenue.

6.3 Predictive analysis – customer churn analysis

This refers to Appendix C. This model predicts whether a customer is going to churn (leave) or stay. Class 1 represents churners, while Class 0 represents non-churners. The model has a precision of 87.1%, meaning that when it predicts churn, it is correct in 87.1% of cases. It also has a recall of 98.3%, which indicates that it successfully identifies almost all churners. The data further shows that average freight is the factor with the highest predictive power for churn.

6.4 Predictive analysis - Price Elasticity Analysis

This refers to Appendix D. The data shows that customers are very sensitive to price changes. On average, a 1% price cut results in a 2.8% decrease in demand.

6.5 Prescriptive analysis - Freight Cost Optimization

This refers to Appendix E. The model divides distance into five different bands and suggests an 8% decrease in the delivery rate per kilometer for bands that have an on time delivery percentage of 90% or higher. The first four bands show a reduced delivery rate after optimization, while the last band (>1000 km) remains unchanged because its on time delivery percentage is below 90%.

6.6 Prescriptive analysis: Customer Segmentation Strategy

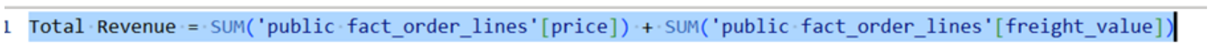
This refers to Appendix F. The model divides customers into nine groups using a 3×3 (value × frequency) matrix. It then suggests different actions, such as VIP programs, re-engagement campaigns, upselling, bundling and more, to target each customer segment and increase revenue.

6.7 Prescriptive Model: Delivery Performance Optimization

This refers to Appendix G. The data shows that AL, RR, MA, SE, CE, and PI are the top five states with the highest risk of delivery delays. The analysis includes recommended actions, such as upgrading logistics partners, to help reduce these delays.

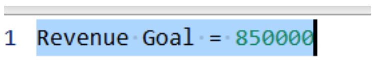
7. Documentation of DAX calculations and design decisions

7.1 Dax calculations



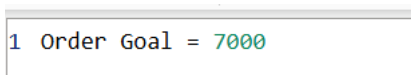
```
Total Revenue = SUM('public fact_order_lines'[price]) + SUM('public fact_order_lines'[freight_value])
```

The total revenue is price + freight_value. Both are ine fact_order_lines



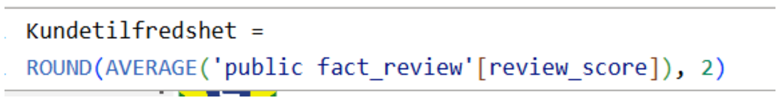
```
Revenue Goal = 850000
```

For revenue goal kpi. We have to hit 850 000 total revenue each month.



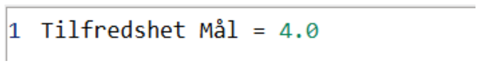
```
Order Goal = 7000
```

Total orders goal kpi, we have to hit 7000 each month



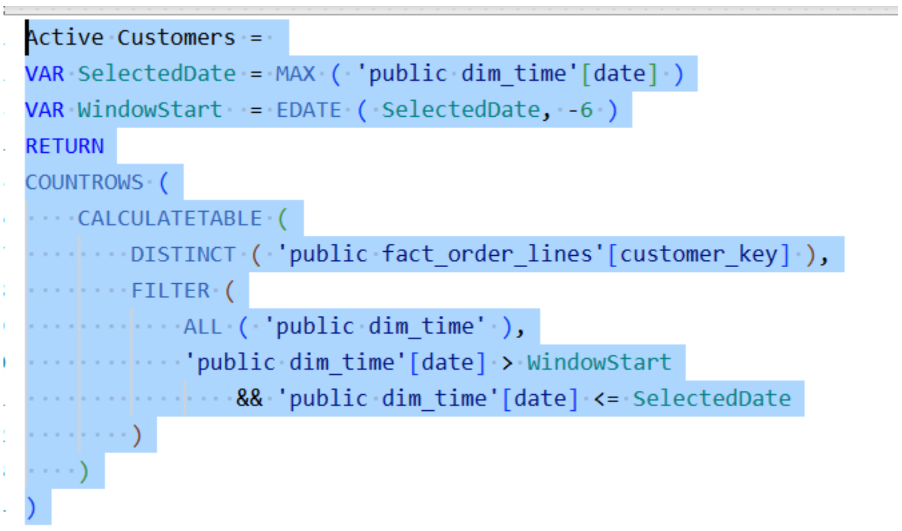
```
Kundetilfredshet =
ROUND(AVERAGE('public fact_review'[review_score]), 2)
```

For customer satisfaction kpi, this gives us AVARAGE review score



```
Tilfredshet Mål = 4.0
```

For Customer Satisfaction kpi, we have to hit 4 review score

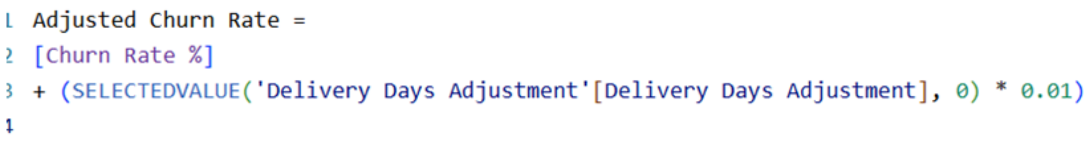


```
Active Customers =
VAR SelectedDate = MAX ( 'public dim_time'[date] )
VAR WindowStart  = EDATE ( SelectedDate, -6 )
RETURN
COUNTROWS (
    CALCULATETABLE (
        DISTINCT ( 'public fact_order_lines'[customer_key] ),
        FILTER (
            ALL ( 'public dim_time' ),
            'public dim_time'[date] > WindowStart
                && 'public dim_time'[date] <= SelectedDate
        )
    )
)
```

This calculates the number of active customers over past 6 months relative to the currently selected date.

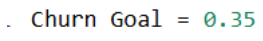
- SelectedDate – Captures the latest date in the current filter conext (date slicer)
- WindowStart – Edate() to move 6 months back from the selected date
- Filter() – Applies a rolling 6-month date window to the entire date table (ALL(‘public dim_time’) removes existing filters
- Distinct(Customer_key) – Ensures each customer is counted only once, even if they have multiple orders
- Countrows – Counts how many distinct customers are active within that 6-month window

1 of my advanced dax calculatins



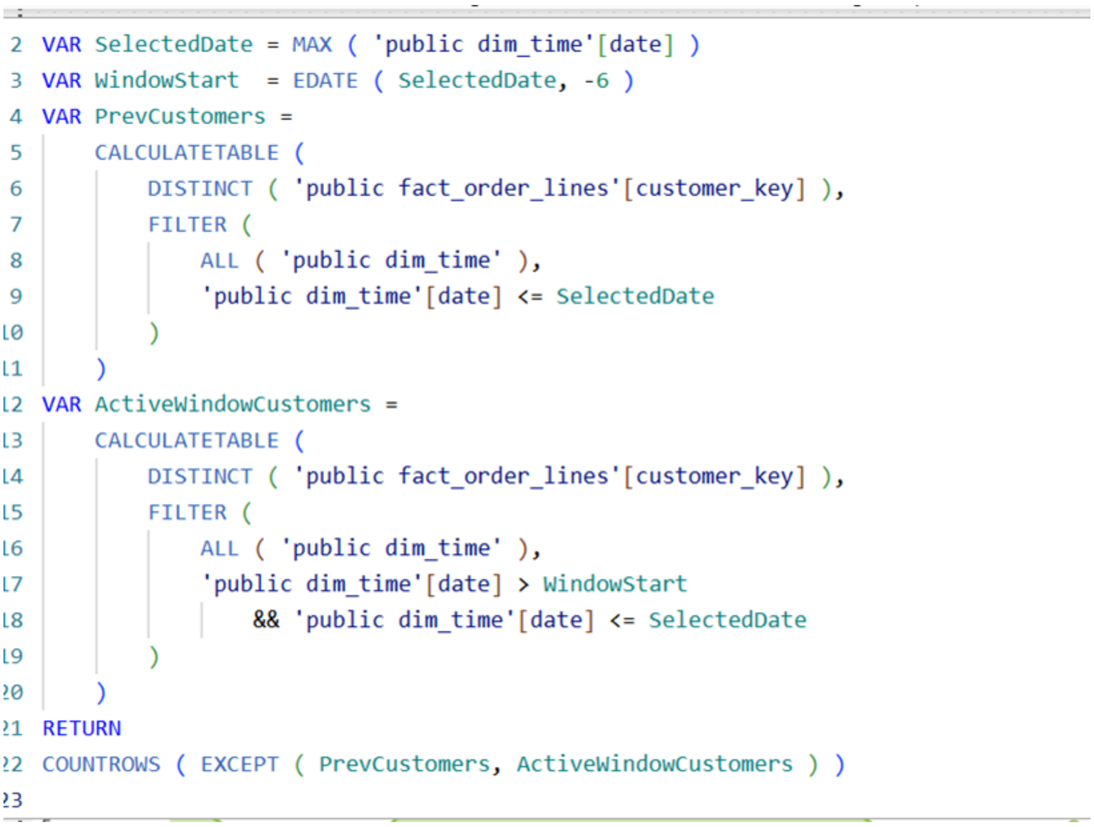
```
Adjusted Churn Rate =
[Churn Rate %]
+ (SELECTEDVALUE('Delivery Days Adjustment'[Delivery Days Adjustment], 0) * 0.01)
```

This measure belongs to the what if analysis. It adjusts customer churn rate based on delivery days. If deliery day gets 1 day more, churn rate will increase by 1%.



```
Churn Goal = 0.35
```

Churn KPI, we cannot have above 35% churned customers. I have reversed the KPI, so lower is better



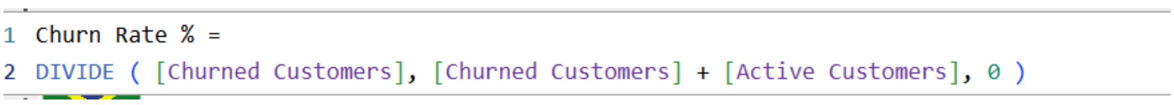
```
VAR SelectedDate = MAX ( 'public dim_time'[date] )
VAR WindowStart  = EDATE ( SelectedDate, -6 )
VAR PrevCustomers =
    CALCULATETABLE (
        DISTINCT ( 'public fact_order_lines'[customer_key] ),
        FILTER (
            ALL ( 'public dim_time' ),
            'public dim_time'[date] <= SelectedDate
        )
    )
VAR ActiveWindowCustomers =
    CALCULATETABLE (
        DISTINCT ( 'public fact_order_lines'[customer_key] ),
        FILTER (
            ALL ( 'public dim_time' ),
            'public dim_time'[date] > WindowStart
                && 'public dim_time'[date] <= SelectedDate
        )
    )
RETURN
COUNTROWS ( EXCEPT ( PrevCustomers, ActiveWindowCustomers ) )
```

This measure calculates how many customers have churned, meaning customers who made purchases before the current 6-month period within it.

I have used **Claude** for this advanced dax.

- PrevCustomers – Creates a table of all customers who have ever purchased before the selected date
- Except() – Returns customers present PrevCustoemrs but not in ActiveWindowCustomers (those who stopped buying)

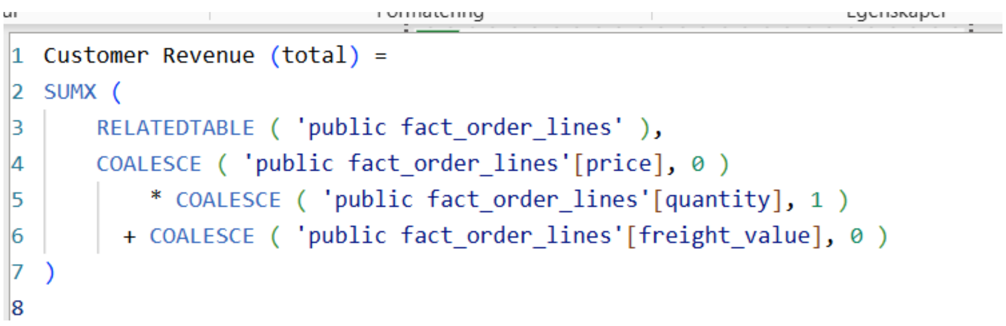
This is my second advanced Dax calculation created by the help of **Claude**



```
Churn Rate % =
DIVIDE ( [Churned Customers], [Churned Customers] + [Active Customers], 0 )
```

Percentage of customers who have churned. Formula logic, churned/ churned + active.

Relies on heawier dax calculation but is «easy» as itself.



```
Customer Revenue (total) =
SUMX (
    RELATEDTABLE ( 'public fact_order_lines' ),
    COALESCE ( 'public fact_order_lines'[price], 0 )
        * COALESCE ( 'public fact_order_lines'[quantity], 1 )
      + COALESCE ( 'public fact_order_lines'[freight_value], 0 )
)
```

First measure calculates total revenue per customer, like said before. Total revenue is sales and freight income.

The second measure decides if the custoemr is high value or low value. If he/she have bought more than in their total revenue 1000 they become high value.

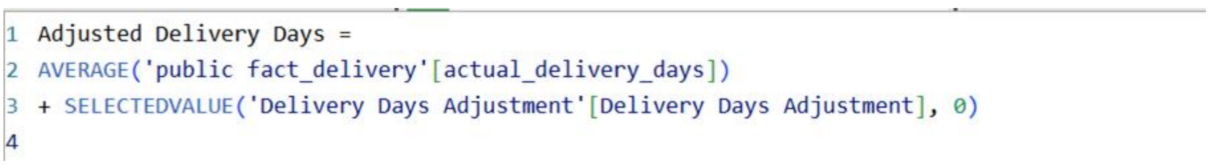
Sumx() – Iterates over all related order line records for each customer

RELATEDABLE() – Retrieves all orders linked to the current customer context

COALESCE() – Replaces any missing (BLANK) values with default of (0 or 1), preventing calc errors

Calc logic – price * quantity + freight produces the total value of each order line

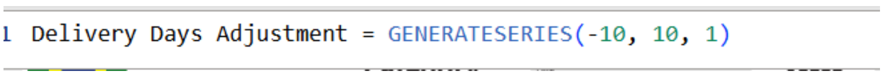
These advanced dax calculations have i used for analytical deep dive, Claude have been used for troubleshooting to save time.



```
Adjusted Delivery Days =
AVERAGE('public fact_delivery'[actual_delivery_days])
+ SELECTEDVALUE('Delivery Days Adjustment'[Delivery Days Adjustment], 0)
```

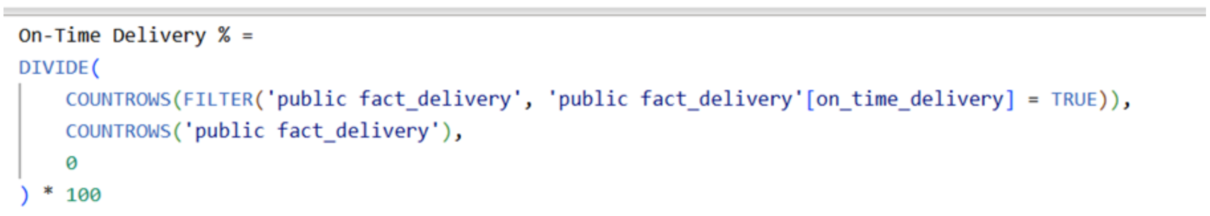
For what if analysis

- Selected Value – reads the user-selected adjustment value from what if slider



```
Delivery Days Adjustment = GENERATESERIES(-10, 10, 1)
```

What if parameter, you can range delivery day values fra -10 to 10, and it increments by 1



```
On-Time Delivery % =
DIVIDE(
    COUNTROWS(FILTER('public fact_delivery', 'public fact_delivery'[on_time_delivery] = TRUE)),
    COUNTROWS('public fact_delivery'),
    0
) * 100
```

This measure simply calculates the percentage of deliveries completed on or before the promised date.

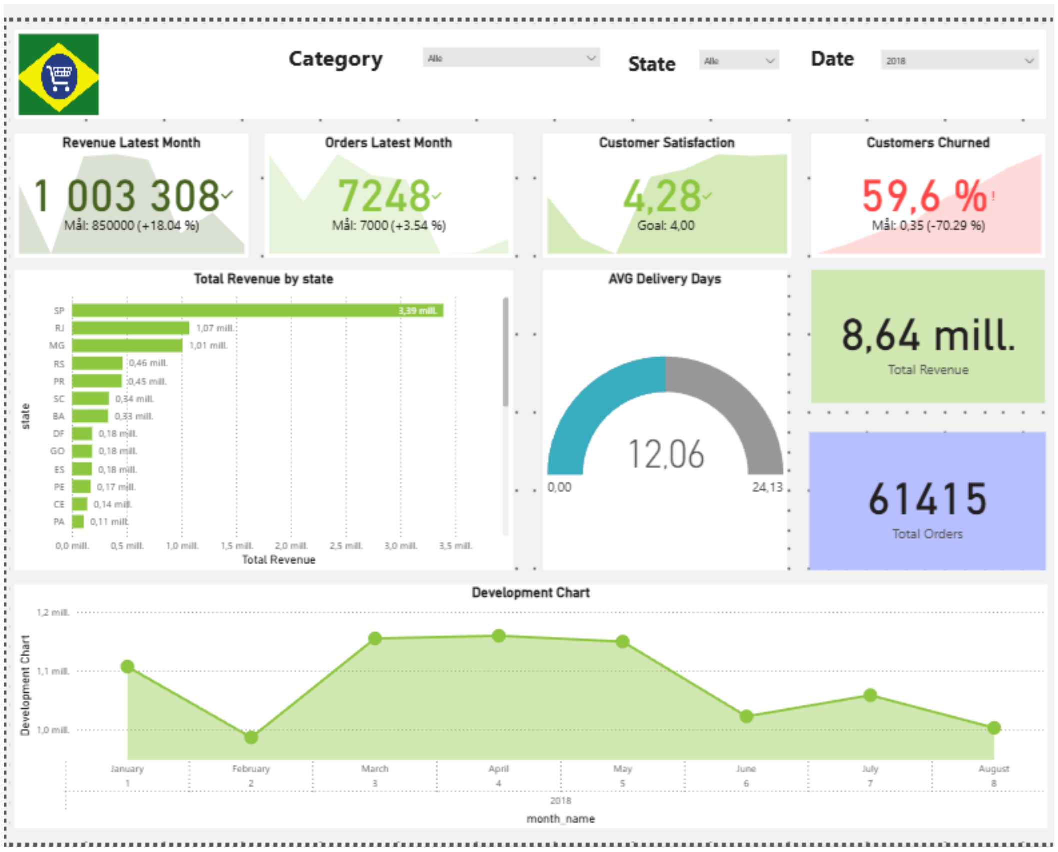
Filter() – Selects only the rows in the delivery table where on_time_delivery is TRUE

COUNTROWS() – Counts how many deliveries meet the condition

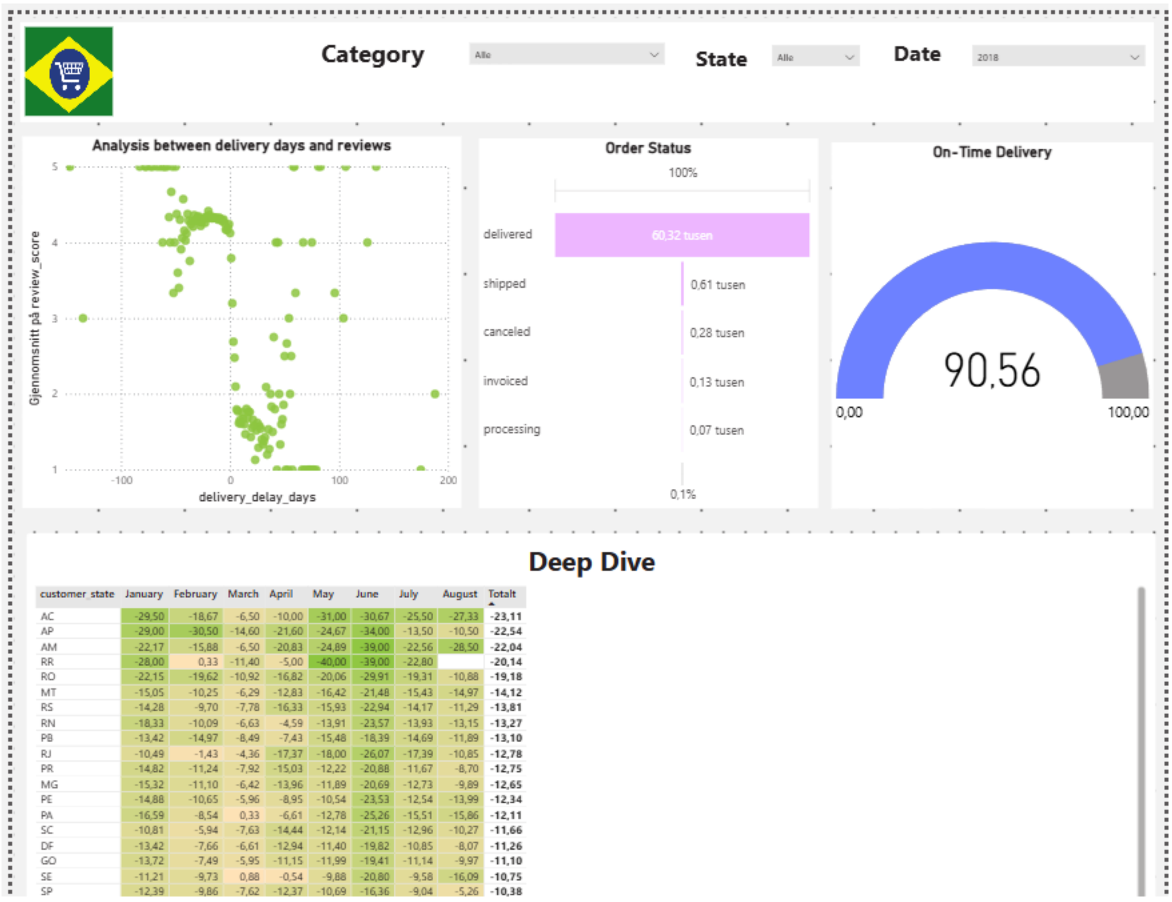
DIVIDE() – Divides the number of on-time deliveries by the total numbers of deliveries

*100 – converts to procentage format

7.2 Design decisions



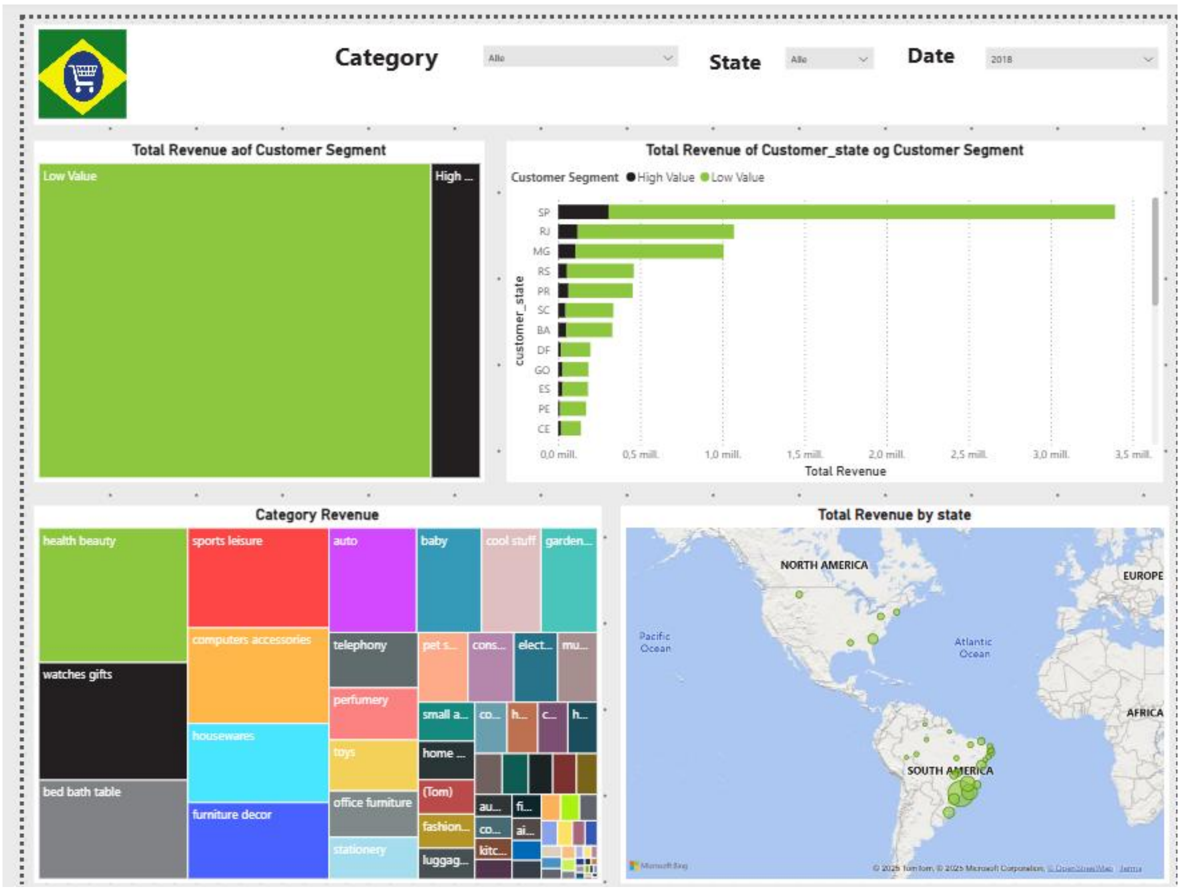
For the summary page, I first tried different colors, but it was for heavy on the eyes. Light colors make it easy to read, therefore easier to understand. I tried to choose colors that were easy and matched with the rest of the visuals. It starts with the header; I have chosen this header on all four pages. It has a picture I found a picture on google that looks like Brazillian E-commerce, the header has three relevant slices with names. Category, State and Date. I chose gardins for the slicers as they are clean and industry standard. For the body, I wanted to have the most relevant KPIs first so they can get attention. I also chose to have with two relevant cards, one for total revenue and one for total orders. These cards sum up whichever slicers we slide between, but the KPIS is just the latest month that is chosen I the slicer. For avg delivery days I have a visual measure, very simple that gives of overview and that is what we are looking for. Finally, we have bigger visualizations for revenue for total states and a development chart. This is also industry standard, we need to see development in a way we get info, and it is easy on the eyes. Thats why i got shadows and dots, as i believe it makes it much easier to look over. Overall, a clean and industry standard dashboard. This sums up overall the other pages as well.

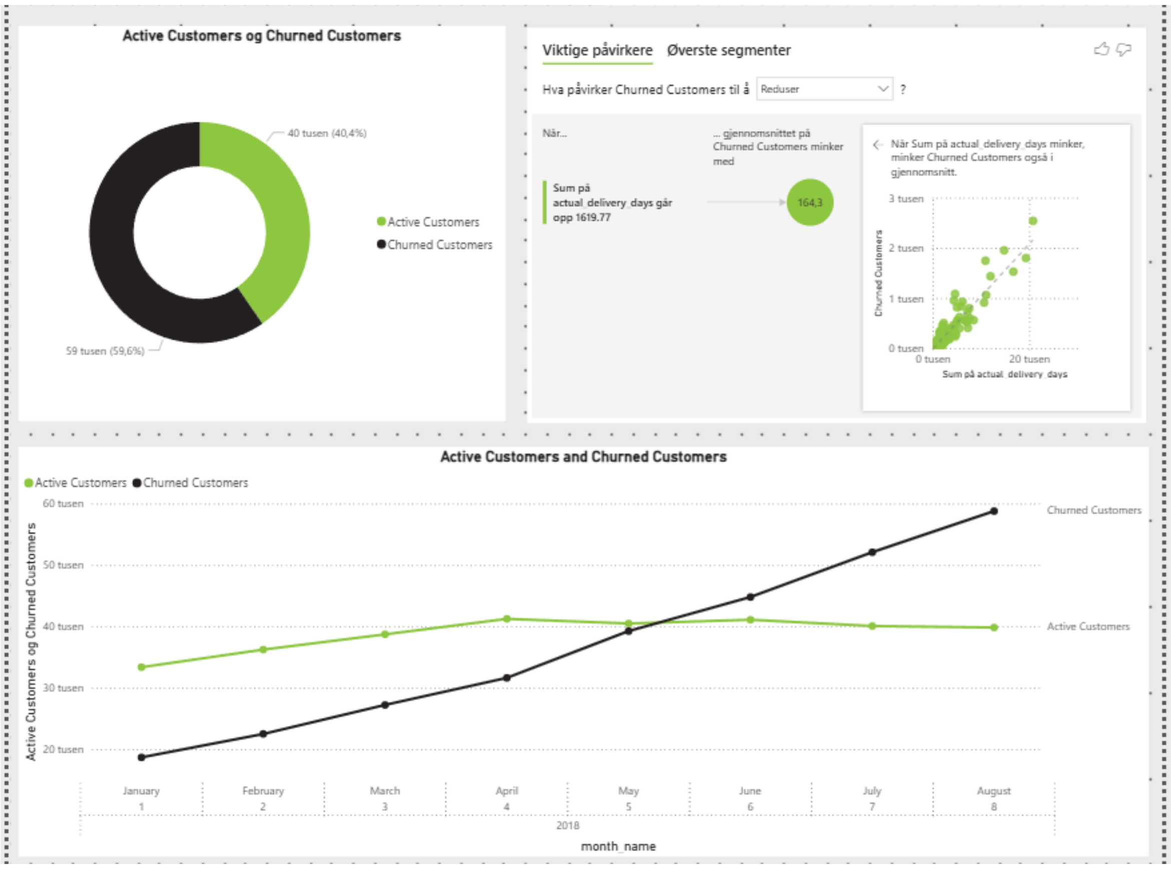


Deep Dive

customer_state	January	February	March	April	May	June	July	August	Totalt
AC	-29,50	-18,67	-6,50	-10,00	-31,00	-30,67	-25,50	-27,33	**-23,11**
AP	-29,00	-30,50	-14,60	-21,60	-24,67	-34,00	-13,50	-10,50	**-22,54**
AM	-22,17	-15,88	-6,50	-20,83	-24,89	-39,00	-22,56	-28,50	**-22,04**
RR	-28,00	0,33	-11,40	-5,00	-40,00	-39,00	-22,80		**-20,14**
RO	-22,15	-19,62	-10,92	-16,82	-20,06	-29,91	-19,31	-10,88	**-19,18**
MT	-15,05	-10,25	-6,29	-12,83	-16,42	-21,48	-15,43	-14,97	**-14,12**
RS	-14,28	-9,70	-7,78	-16,33	-15,93	-22,94	-14,17	-11,29	**-13,81**
RN	-18,33	-10,09	-6,63	-4,59	-13,91	-23,57	-13,93	-13,15	**-13,27**
PB	-13,42	-14,97	-8,49	-7,43	-15,48	-18,39	-14,69	-11,89	**-13,10**
RJ	-10,49	-1,43	-4,36	-17,37	-18,00	-26,07	-17,39	-10,85	**-12,78**
PR	-14,82	-11,24	-7,92	-15,03	-12,22	-20,88	-11,67	-8,70	**-12,75**
MG	-15,32	-11,10	-6,42	-13,96	-11,89	-20,69	-12,73	-9,89	**-12,65**
PE	-14,88	-10,65	-5,96	-8,95	-10,54	-23,53	-12,54	-13,99	**-12,34**
PA	-16,59	-8,54	0,33	-6,61	-12,78	-25,26	-15,51	-15,86	**-12,11**
SC	-10,81	-5,94	-7,63	-14,44	-12,14	-21,15	-12,96	-10,27	**-11,66**
DF	-13,42	-7,66	-6,61	-12,94	-11,40	-19,82	-10,85	-8,07	**-11,26**
GO	-13,72	-7,49	-5,95	-11,15	-11,99	-19,41	-11,14	-9,97	**-11,10**
SE	-11,21	-9,73	0,88	-0,54	-9,88	-20,80	-9,58	-16,09	**-10,75**
SP	-12,39	-9,86	-7,62	-12,37	-10,69	-16,36	-9,04	-5,26	**-10,38**

Here we have the operational dashboard. The same header. As we can see, like the dashboard. There is space between the visuals, the sides and even the header. This is to follow industry standards again so the eyes can read intresting facts. This was PointDiagram was perfect for an analysis between delivery days and reviews and a treechart for order status. I had to have a metrics/heat map with since it is the most relevant for the industry. I chose to have differen format for colours, the ones that are very green have delivered before estimation. Then it gets lighter to red, the red ones is when it is late delivery. Using green and red is powerful when it comes to looking the right and the wrong. A measure again, where we can easy see how many deliveries are on time. We can choose the filters for what we want to analyze.





Hva påvirker Churned Customers til å Reduser ?

Når...

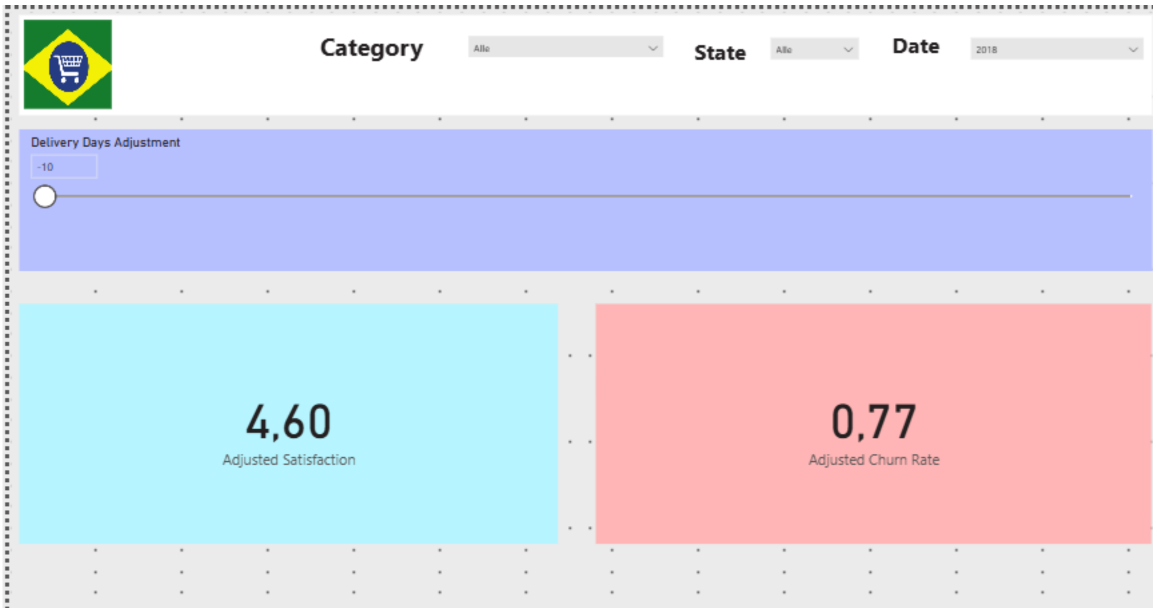
... gjennomsnittet på Churned Customers minker med

Sum på actual_delivery_days går opp 1619.77 → 164,3

Når Sum på actual_delivery_days minker, minker Churned Customers også i gjennomsnitt.

This is the analytical deep dive page. As we can see, it uses much more space and is bigger than the dashboards. Again, the same header. We start with seeing Total revenue for customers and they are either high or low customers, and under we have a treemap that shows us the most relevant categories. And to the right we have a map. This gives us an overview and deep dive of our customers, how much they shop, what they shop for and if they pay high prices and low prices.

The second part we have active and churned customers. Even an AI visual that helps us with the analysis. It clearly says that delivery days and churned customers are linked. The fewer the days, the fewer customers churned. And at the bottom a line graph for churned and active customers. I tried to make this page as basic as possible, as it has some advanced calculations. I have not changed the colours, basic white. Black and green for comparing high and low customers, black and green for churned and active customers. It just makes sense, so why change it? The page is easy to read and understand and gives us an analytical deep dive for customers and sales.



This is the what if analysis page, it felt to empty. So, I decided to give it light colors so it does not get too boring. The header as before is like the rest to keep integrity. And nice light colors so it grabs attention.

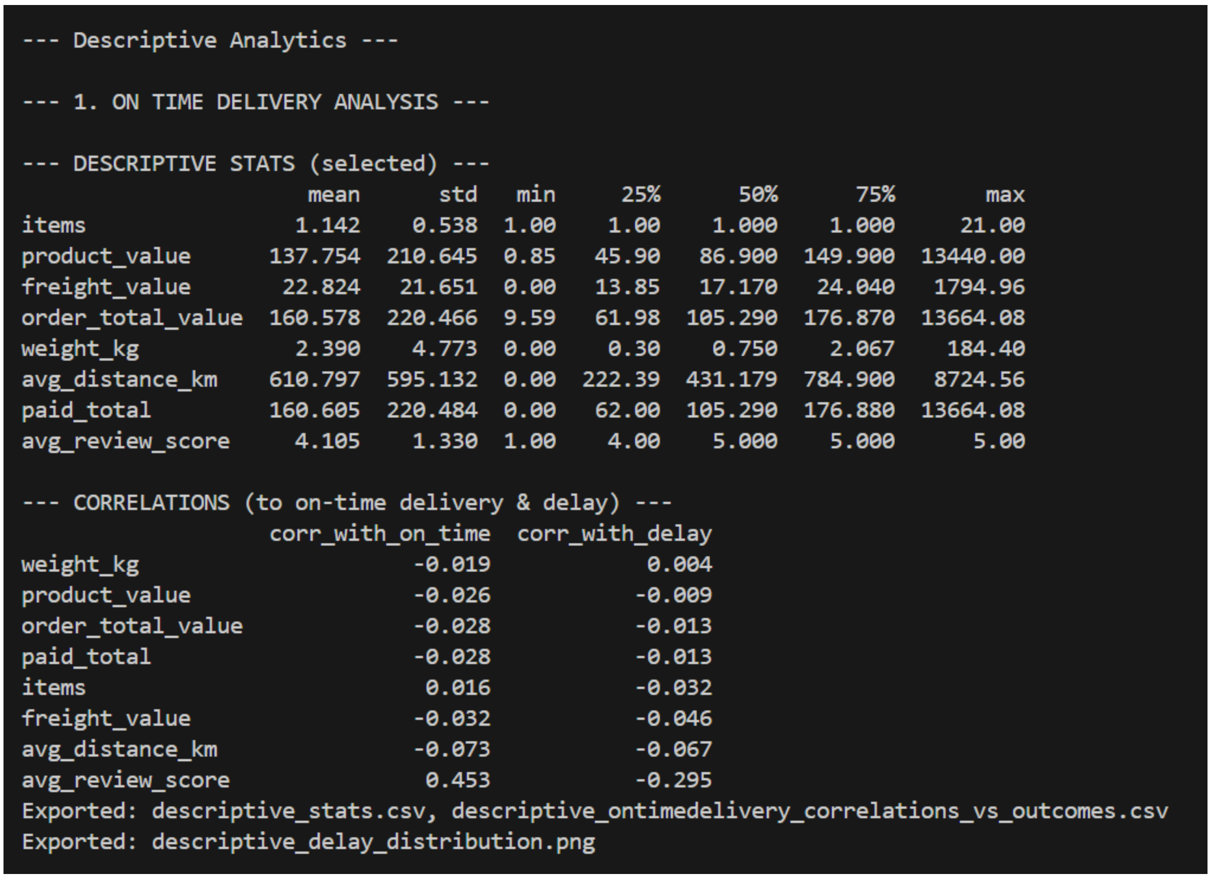
8. Reference

Sharda, R., Delen, D., & Turban, E. (2024). *Business intelligence, analytics, data science, and AI: A managerial perspective*. Pearson.

Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling* (3rd ed.). John Wiley & Sons, Incorporated.

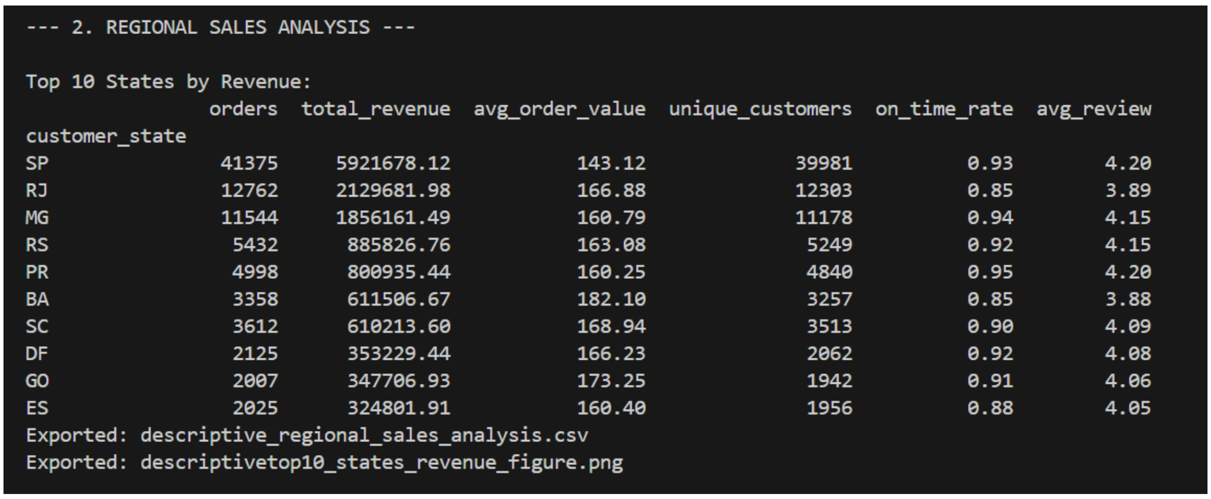
9. APPENDIX

APPENDIX A



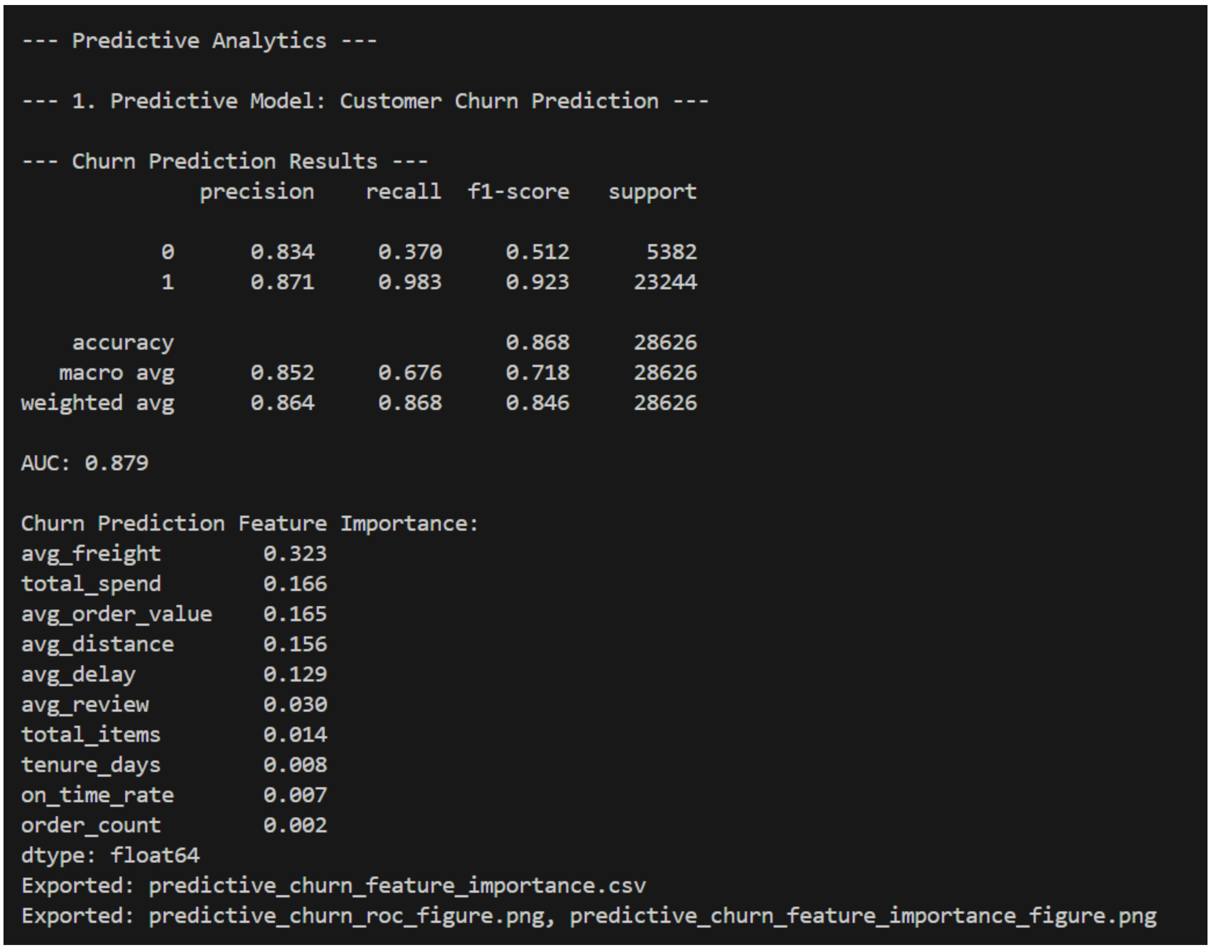
```
--- Descriptive Analytics ---

--- 1. ON TIME DELIVERY ANALYSIS ---

--- DESCRIPTIVE STATS (selected) ---
                      mean      std   min     25%      50%      75%       max
items                1.142    0.538  1.00    1.00    1.000    1.000     21.00
product_value      137.754  210.645  0.85   45.90   86.900  149.900  13440.00
freight_value       22.824   21.651  0.00   13.85   17.170   24.040   1794.96
order_total_value  160.578  220.466  9.59   61.98  105.290  176.870  13664.08
weight_kg            2.390    4.773  0.00    0.30    0.750    2.067    184.40
avg_distance_km    610.797  595.132  0.00  222.39  431.179  784.900   8724.56
paid_total         160.605  220.484  0.00   62.00  105.290  176.880  13664.08
avg_review_score     4.105    1.330  1.00    4.00    5.000    5.000      5.00

--- CORRELATIONS (to on-time delivery & delay) ---
                   corr_with_on_time  corr_with_delay
weight_kg                     -0.019            0.004
product_value                 -0.026           -0.009
order_total_value             -0.028           -0.013
paid_total                    -0.028           -0.013
items                          0.016           -0.032
freight_value                 -0.032           -0.046
avg_distance_km               -0.073           -0.067
avg_review_score               0.453           -0.295
Exported: descriptive_stats.csv, descriptive_ontimedelivery_correlations_vs_outcomes.csv
Exported: descriptive_delay_distribution.png
```

APPENDIX B



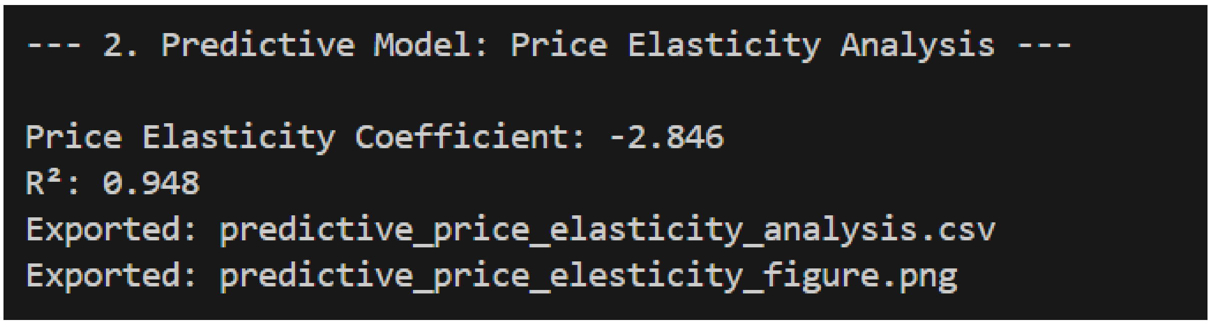
```
--- 2. REGIONAL SALES ANALYSIS ---

Top 10 States by Revenue:
               orders  total_revenue  avg_order_value  unique_customers  on_time_rate  avg_review
customer_state
SP              41375     5921678.12           143.12             39981          0.93        4.20
RJ              12762     2129681.98           166.88             12303          0.85        3.89
MG              11544     1856161.49           160.79             11178          0.94        4.15
RS               5432      885826.76           163.08              5249          0.92        4.15
PR               4998      800935.44           160.25              4840          0.95        4.20
BA               3358      611506.67           182.10              3257          0.85        3.88
SC               3612      610213.60           168.94              3513          0.90        4.09
DF               2125      353229.44           166.23              2062          0.92        4.08
GO               2007      347706.93           173.25              1942          0.91        4.06
ES               2025      324801.91           160.40              1956          0.88        4.05
Exported: descriptive_regional_sales_analysis.csv
Exported: descriptivetop10_states_revenue_figure.png
```

APPENDIX C



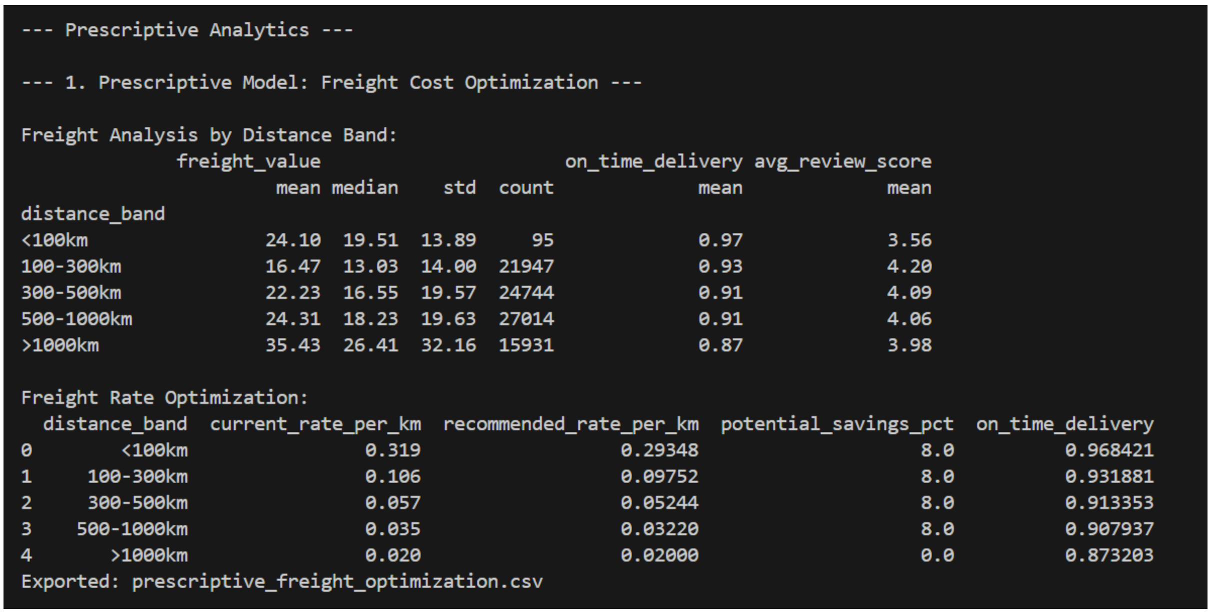
```
--- Predictive Analytics ---

--- 1. Predictive Model: Customer Churn Prediction ---

--- Churn Prediction Results ---
              precision    recall  f1-score   support

           0      0.834     0.370     0.512      5382
           1      0.871     0.983     0.923     23244

    accuracy                          0.868     28626
   macro avg      0.852     0.676     0.718     28626
weighted avg      0.864     0.868     0.846     28626

AUC: 0.879

Churn Prediction Feature Importance:
avg_freight        0.323
total_spend        0.166
avg_order_value    0.165
avg_distance       0.156
avg_delay          0.129
avg_review         0.030
total_items        0.014
tenure_days        0.008
on_time_rate       0.007
order_count        0.002
dtype: float64
Exported: predictive_churn_feature_importance.csv
Exported: predictive_churn_roc_figure.png, predictive_churn_feature_importance_figure.png
```

APPENDIX D



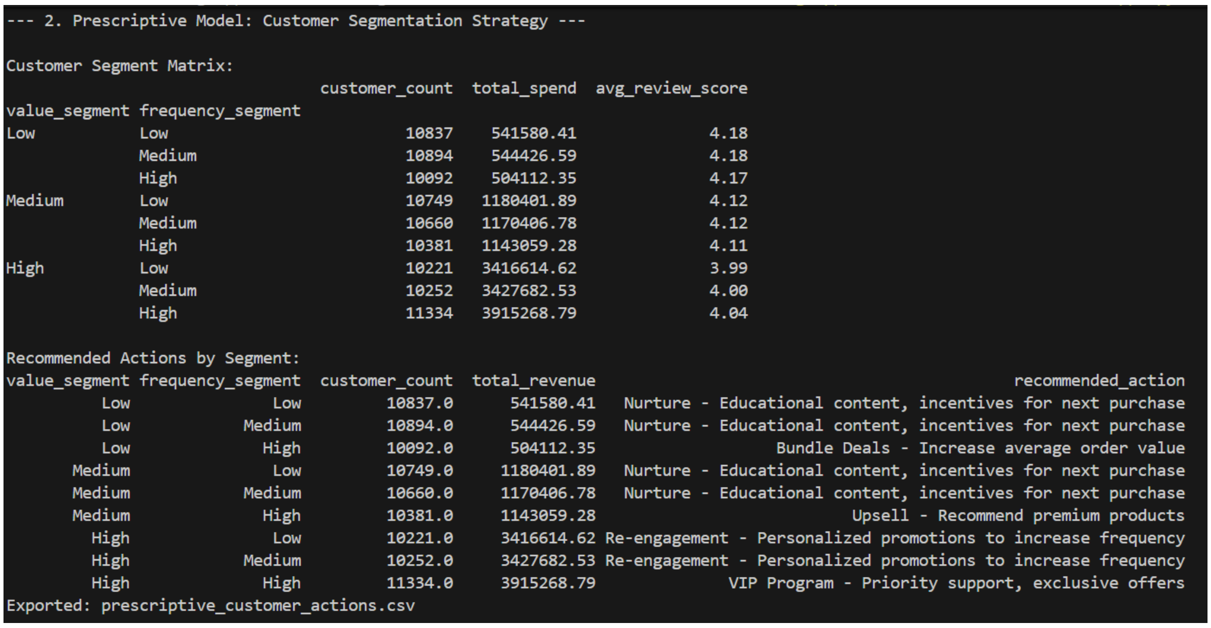
```
--- 2. Predictive Model: Price Elasticity Analysis ---

Price Elasticity Coefficient: -2.846
R²: 0.948
Exported: predictive_price_elasticity_analysis.csv
Exported: predictive_price_elesticity_figure.png
```

APPENDIX E



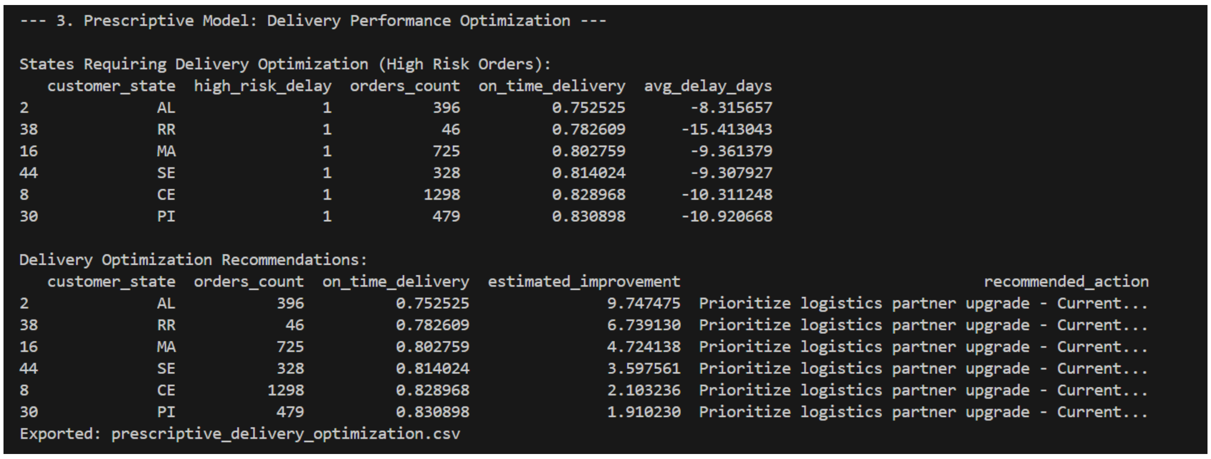
```
--- Prescriptive Analytics ---

--- 1. Prescriptive Model: Freight Cost Optimization ---

Freight Analysis by Distance Band:
              freight_value                     on_time_delivery avg_review_score
                       mean median    std  count             mean             mean
distance_band
<100km                24.10  19.51  13.89     95             0.97             3.56
100-300km             16.47  13.03  14.00  21947             0.93             4.20
300-500km             22.23  16.55  19.57  24744             0.91             4.09
500-1000km            24.31  18.23  19.63  27014             0.91             4.06
>1000km               35.43  26.41  32.16  15931             0.87             3.98

Freight Rate Optimization:
  distance_band  current_rate_per_km  recommended_rate_per_km  potential_savings_pct  on_time_delivery
0        <100km                0.319                  0.29348                    8.0          0.968421
1     100-300km                0.106                  0.09752                    8.0          0.931881
2     300-500km                0.057                  0.05244                    8.0          0.913353
3    500-1000km                0.035                  0.03220                    8.0          0.907937
4       >1000km                0.020                  0.02000                    0.0          0.873203
Exported: prescriptive_freight_optimization.csv
```

APPENDIX F



```
--- 2. Prescriptive Model: Customer Segmentation Strategy ---

Customer Segment Matrix:
                               customer_count  total_spend  avg_review_score
value_segment frequency_segment
Low           Low                       10837    541580.41              4.18
              Medium                    10894    544426.59              4.18
              High                      10092    504112.35              4.17
Medium        Low                       10749   1180401.89              4.12
              Medium                    10660   1170406.78              4.12
              High                      10381   1143059.28              4.11
High          Low                       10221   3416614.62              3.99
              Medium                    10252   3427682.53              4.00
              High                      11334   3915268.79              4.04

Recommended Actions by Segment:
value_segment frequency_segment  customer_count  total_revenue                                        recommended_action
          Low               Low         10837.0      541580.41   Nurture - Educational content, incentives for next purchase
          Low            Medium         10894.0      544426.59   Nurture - Educational content, incentives for next purchase
          Low              High         10092.0      504112.35                       Bundle Deals - Increase average order value
       Medium               Low         10749.0     1180401.89   Nurture - Educational content, incentives for next purchase
       Medium            Medium         10660.0     1170406.78   Nurture - Educational content, incentives for next purchase
       Medium              High         10381.0     1143059.28                            Upsell - Recommend premium products
         High               Low         10221.0     3416614.62 Re-engagement - Personalized promotions to increase frequency
         High            Medium         10252.0     3427682.53 Re-engagement - Personalized promotions to increase frequency
         High              High         11334.0     3915268.79                 VIP Program - Priority support, exclusive offers
Exported: prescriptive_customer_actions.csv
```

APPENDIX G



```
--- 3. Prescriptive Model: Delivery Performance Optimization ---

States Requiring Delivery Optimization (High Risk Orders):
   customer_state  high_risk_delay  orders_count  on_time_delivery  avg_delay_days
2              AL                1           396          0.752525       -8.315657
38             RR                1            46          0.782609      -15.413043
16             MA                1           725          0.802759       -9.361379
44             SE                1           328          0.814024       -9.307927
8              CE                1          1298          0.828968      -10.311248
30             PI                1           479          0.830898      -10.920668

Delivery Optimization Recommendations:
   customer_state  orders_count  on_time_delivery  estimated_improvement                                 recommended_action
2              AL           396          0.752525               9.747475  Prioritize logistics partner upgrade - Current...
38             RR            46          0.782609               6.739130  Prioritize logistics partner upgrade - Current...
16             MA           725          0.802759               4.724138  Prioritize logistics partner upgrade - Current...
44             SE           328          0.814024               3.597561  Prioritize logistics partner upgrade - Current...
8              CE          1298          0.828968               2.103236  Prioritize logistics partner upgrade - Current...
30             PI           479          0.830898               1.910230  Prioritize logistics partner upgrade - Current...
Exported: prescriptive_delivery_optimization.csv
```