Development of an ETL-Driven Dashboard for Analyzing Customer Behaviour towards Products of an E-Commerce Website

A Data Pipeline and BI Project using Pentaho DI and Power BI

Developed by:

Adarshram Nair

Email: nair.adarshram@gmail.com

LinkedIn: linkedin.com/in/adarshram-nair

Github: github.com/adarshram-nair

Date: June, 2025

Table of Contents

1. Aim of the Project

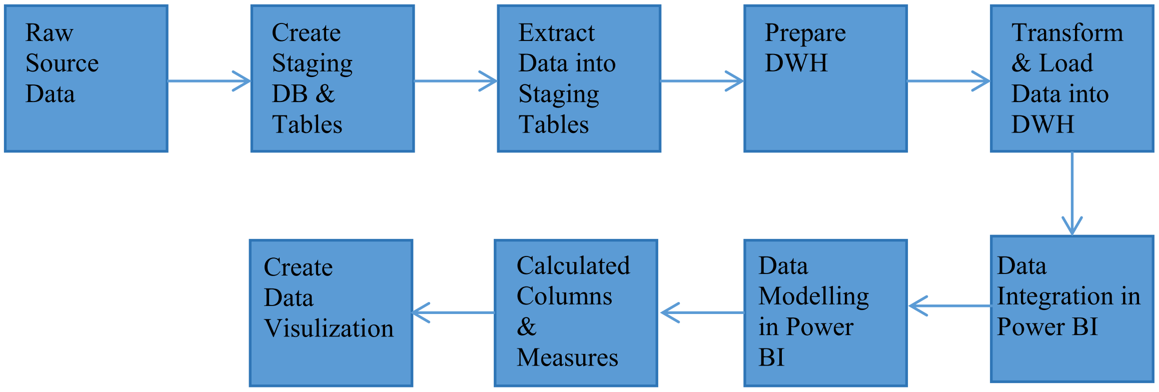
To develop a BI dashboard using Microsoft Power BI to analyze customer behaviour towards various products during the year 2024. The data for the dashboard will be Extracted, Transformed, and Loaded (ETL) into a Data Warehouse using an ETL tool - Pentaho Data Integration. Since this dashboard is for analyzing customer behaviour, Customers' private information (example: names) will not be extracted into the Data Warehouse or included in the visualizations.

2. Tools Used

- Database Management System - PostgreSQL and pgAdmin4
- Data Integration Tool - Pentaho Data Integration (PDI)
- Business Intelligence - Microsoft Power BI

3. Process Flow

Process flow of the project:


Raw Source Data
Create Staging DB & Tables
Extract Data into Staging Tables
Prepare DWH
Transform & Load Data into DWH
Data Integration in Power BI
Data Modelling in Power BI
Calculated Columns & Measures
Create Data Visulization

3.1 Raw Source Data

3.1.1 Data source

The source dataset is downloaded from Kaggle: www.kaggle.com/datasets/chadwambles/ecommerce-transactions. The dataset mimics transactions for a fictional eCommerce website named EverMart Online.

3.1.2 Available data

The dataset consists of 3 .csv files:

- Customers.csv (Columns: CustomerID, CustomerName, Region, SignupDate)
- Products.csv (Columns: ProductID, ProductName, Category, Price)
- Transactions.csv (Columns: TransactionID, CustomerID, ProductID, TransactionDate,Quantity, TotalValue, Price)

3.1.3 Usage of the dataset

These .csv files represent the transactional tables for this project and will be used for further processing.

3.1.4 Data type

The data type of the data is not considered while extracting the data into staging tables, hence every column will be considered as string data type. This is done to avoid errors during data type conversion caused by formatting issues in some records. Data type conversions and handling issues will be performed in data cleaning and validation steps during Transformation of data and Loading the data into the data warehouse.

3.1.5 Dataset analysis

- The 'CustomerID', 'ProductID' and 'TransactionID' are the primary keys of the three tables 'Customers', 'Products' and 'Transactions' respectively.
- The 'CustomerID' and 'ProductID' used in 'Transactions' table are foreign keys referencing the primary keys of the tables 'Customers' and 'Products' respectively.
- Every record in the 'Transactions' table represents one transaction. Each transaction only has one product (multiple quantities can be present). For example, If a customer orders 3 products with 2 quantities each, 3 transaction records with 3 unique 'TransactionID's will be generated.

3.2 Create Staging Database and Tables

The transactional data will be extracted to staging tables in staging database (DB). Following SQL statements are executed in PostgreSQL for creating the DB and tables.

3.2.1 Staging DB creation

Following SQL statement is executed in PostgreSQL for the creation of Staging DB.

```
CREATE DATABASE staging_ecom;
```

3.2.2 Staging table creation

The following SQL statement is executed in the staging_ecom DB to create the required tables.

```
--customers table creation
CREATE TABLE customers(
CustomerID varchar(255),
CustomerName varchar(255),
Region varchar(255),
SignupDate varchar(255)
);


--products table creation
CREATE TABLE Products(
ProductID varchar(255),
ProductName varchar(255),
Category varchar(255),
Price varchar(255)
);

--transactions table creation
CREATE TABLE Transactions(
TransactionID varchar(255),
CustomerID varchar(255),
ProductID varchar(255),
TransactionDate varchar(255),
Quantity varchar(255),
TotalValue varchar(255),
Price varchar(255)
);
```

3.3 Extract Data into Staging Tables

The following Pentaho Data Integration (PDI) Job is used for extracting the transactional data to the staging tables:

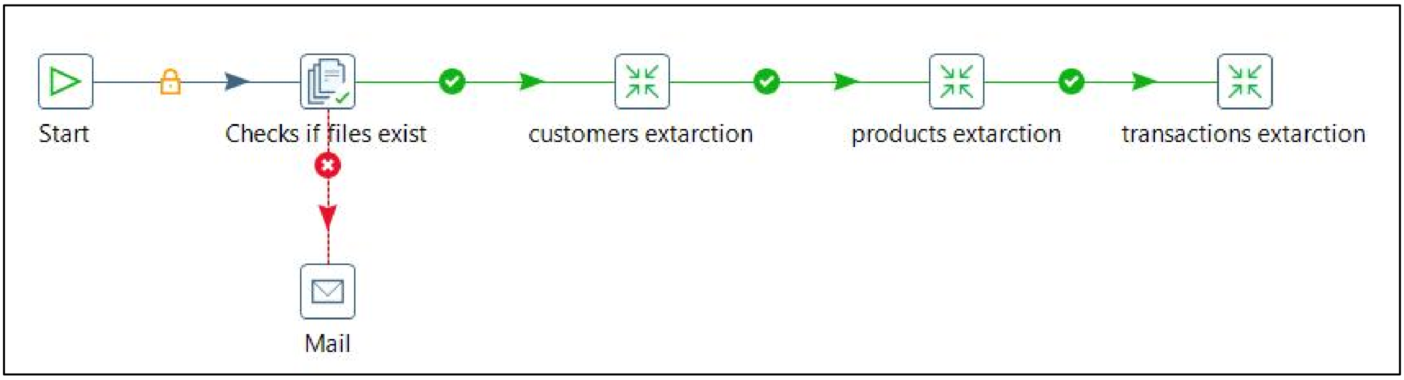


Fig 1: PDI job for extraction from .csv files to staging DB

3.3.1 Start

Step to start the PDI job execution.

3.3.2 Check if files Exist

This step verifies whether all the 3 .csv files are present in the required directory or not. If all files exists, it will move forward to step 3.3.4 (customer extraction) else it will go to step 3.3.3 (Mail).

3.3.3 Mail

This step sends an email to the concerned team notifying that the required files are not available

3.3.4 Customers Extraction

This step runs the ‘customers extraction’ PDI transformation. This transformation simply extracts data from the ‘Customers.csv’ file to the ‘Customers’ staging table.

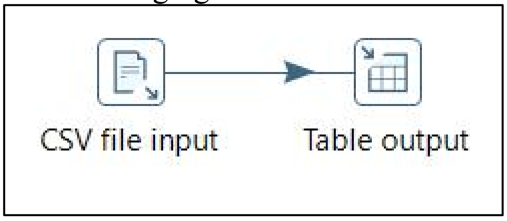


Fig 2: PDI transformation for etracting data from customers.csv to customers table in staging DB

3.3.5 Products Extraction

This step runs the ‘products extraction’ PDI transformation. Like the previous step, it just simply extracts data from the ‘Products.csv’ file to the ‘Products’ staging table.

3.3.6 Transactions Extraction

This step runs the ‘transactions extraction’ PDI transformation. Like the previous two steps, it just simply extracts data from the ‘Transactions.csv’ file to the ‘Transactions’ staging table.

3.4 Prepare Data Warehouse

To store the Transformed data, Data Warehouse (DWH) is created along with the required Tables in PostgreSQL. The data in the DWH will be used by the BI tool to build dashboards and reports. Fact tables, dimension tables and its types and columns for each table has to be determined before creation of the tables.

3.4.1 DWH DB creation

Following SQL statement is executed in PostgreSQL for the creation of DWH DB.

```
CREATE DATABASE dwh_ecom;
```

3.4.2 Determining facts and dimensions

Fact table: The table with the transaction records will be the fact table. It will be named as "**transactions_fact**" in the DWH.

Dimension tables: The tables containing the data of customers and products will be used as the dimension tables. These dimension tables will be named as "**customers_dim**" and "**products_dim**" respectively.

3.4.3 Type of dimensions

Both the customers and products dimensions will follow **SCD Type 2** for historical tracking.

3.4.4 Columns for dimension and fact tables

Following tables represent the columns for each tables in DWH. Since the dimension tables are following SCD Type 2, the necessary additional fields are added.

Columns of customers_dim table:

Sr. No.	Column Name	Description	Source/Logic	Constraints
1	customer_id	Customer id	From source table	
2	cus_surr_id	Surrogate id for customer	Created using the ETL tool	**Primary Key**
3	region	Region	From source table	
4	signup_date	Signup date	From source table	
5	data_version	Version of the data of the particular customer. There may be multiple versions for a customer, if any customer details are changed one or more times	Created using the ETL tool	
6	valid_from	Date from which the current row is valid	Created using the ETL tool	

7	valid_to	Date till which the current row is valid	Created using the ETL tool	
8	current	Show whether the particular row is the current row for the customer_id	Created using the ETL tool	

Columns of products_dim table:

Sr. No.	Column Name	Description	Source/Logic	Constraints
1	product_id	Product Id	From source table	
2	prod_surr_id	Surrogate id for the product	Created using the ETL tool	**Primary Key**
3	product_name	Product's name	From source table	
4	category	Category of the product	From source table	
5	price	Price of the product	From source table	
6	data_version	Version of the data of the particular product. There may be multiple versions for a product, if any product details are changed one or more times.	Created using the ETL tool	
7	valid_from	Date from which the current row is valid.	Created using the ETL tool	
8	valid_to	Date till which the current row is valid.	Created using the ETL tool	
9	current	Show whether the particular row is the current row for the product_id.	Created using the ETL tool	

Columns of transactions_fact table:

Sr. No.	Column Name	Description	Source/Logic	Constraints
1	transaction_id	Transaction id	From source table	**Primary Key**
2	cus_surr_id	Surrogate id of the customer	Retrieve using ETL tool by DB Lookup from 'customers_dim' table	**Foreign Key (customers_dim.cus_surr_id)**
3	prod_surr_id	Surrogate id of the product	Retrieve using ETL tool by DB Lookup from 'products_dim' table	**Foreign Key (products_dim.prod_surr_id)**
4	transaction_date	Date of transaction	Calculated Date from 'transactiondate' column of source table	

5	transaction_time	Time of transaction	Calculated Time from ‘transactiondate’ column of source table	
6	price	Price of the product	Retrieve using ETL tool by DB Lookup from ‘products_dim’ table	
7	quantity	Quantity of each product	From source table	

3.4.5 DWH table creation

The following SQL statement is executed in the dwh_ecom DB to create the required tables.

```
--create customers_dim table
CREATE TABLE customers_dim(
customer_id varchar(15),
cus_surr_id integer PRIMARY KEY,
region varchar(255),
signup_date date,
data_version integer,
valid_from timestamp,
valid_to timestamp,
current varchar(5)
);

--create products_dim table
CREATE TABLE products_dim(
product_id varchar(15),
prod_surr_id integer PRIMARY KEY,
product_name varchar(255),
category varchar(255),
price numeric,
data_version integer,
valid_from timestamp,
valid_to timestamp,
current varchar(5)
);

--create customers_dim table
CREATE TABLE transactions_fact(
transaction_id varchar(15) PRIMARY KEY,
cus_surr_id integer,
prod_surr_id integer,
transaction_date date,
transaction_time time,
price numeric,
quantity numeric,
CONSTRAINT trans_cus_fk FOREIGN KEY (cus_surr_id) REFERENCES
customers_dim(cus_surr_id) ON DELETE SET NULL,
CONSTRAINT trans_prod_fk FOREIGN KEY (prod_surr_id) REFERENCES
products_dim(prod_surr_id) ON DELETE SET NULL
);
```

3.5 Transform and Load Data into Data Warehouse

The following PDI Job is used for transforming and loading the data into the DWH

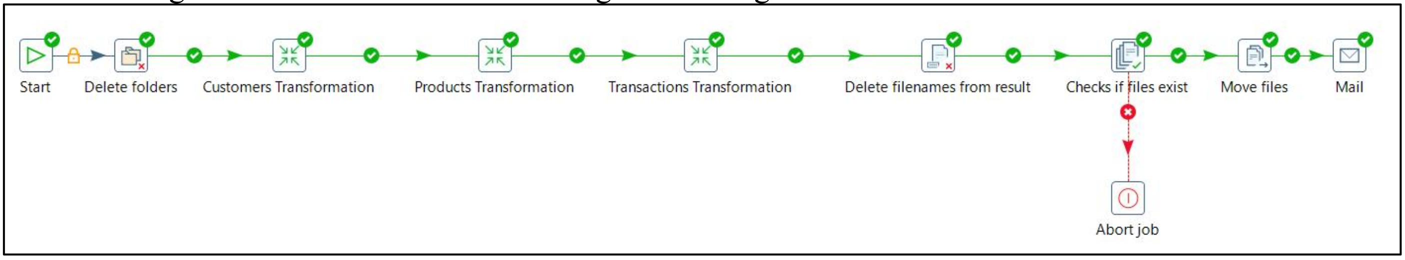


Fig 3: PDI job for transformation and loading data from staging DB to the DWH

3.5.1 Start

PDI job entry to start the PDI job execution.

3.5.2 Delete Folders

This job entry delete the folders created for error files by previous PDI transformations and jobs. This job entry is essential to avoid sending invalid errors to the concerned team.

3.5.3 Customers Transformation

This PDI job entry runs the ‘customers transformation’ PDI transformation. This transformation step will transform and load data from ‘customers’ table in the staging DB to ‘customers_dim’ table in the DWH. The loaded data will not include any private information of the customer such as customer’s name. Values for columns ‘cus_surr_id’, ‘data_version’, ‘valid_from’, ‘valid_to’ and ‘current’ for DWH will be added as per the specified criteria in the ‘Dimension Lookup/Update’ step. The ‘customers transformation’ is as follows:

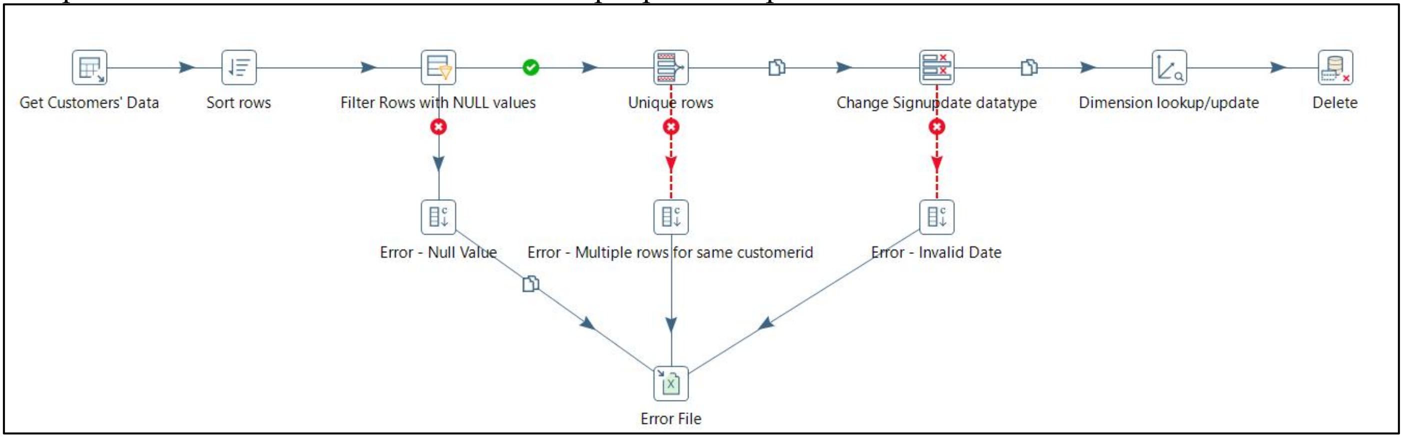


Fig 4: PDI transformation for transforming and loading customers’ data into the DWH

3.5.3.1 Get Customers’ Data

This step selects all columns and rows from the ‘customers’ table in staging DB.

3.5.3.2 Sort rows

This step sorts the rows as per the ‘customerid’ column in ascending order and then by ‘signupdate’ in descending order. Sorting is required for the ‘Unique rows’ step.

3.5.3.3 Filter rows with NULL values

This step is used to filter rows that doesn't have any NULL values for the columns: 'customerid', 'region' and 'signupdate'. These rows will be sent to the next step 'Unique rows' and the rows with NULL values for any of these columns will be sent to 'Error - NULL Value" step.

3.5.3.4 Unique rows

This step checks whether there exists any multiple rows for same 'customerid'. If no, then it passes the rows to the step 'Change Signupdate datatype'. For customer ids with multiple records, only the recent (based on sign up date) is forwarded to the 'Change Signupdate datatype' step and the others will be sent to 'Error - Multiple rows for same customerid' step.

3.5.3.5 Change Signupdate datatype

The 'signupdate' column is in string data type in format "yyyy-MM-dd". This step converts it to date data type and send to 'Dimension lookup/update' step. If any rows have incorrect format resulting in error while converting, will be sent to 'Error - Invalid Date' step.

3.5.3.6 Dimension Lookup/Update

This step inserts / updates the data to the 'customers_dim' table in the DWH. This step looks for records of every customer id with 'customer_id' in 'customers_dim' table of the DWH and inserts / updates accordingly as per SCD Type 2. Unique surrogate ids ('cus_surr_id') are added to the rows (even for rows with same customer id). Values for columns:'version', 'valid_from', 'valid_to' and 'current' are inserted/updated by PDI depending upon presence of old records for same customer id.

3.5.3.7 Error Steps (Addition of column for specifying the errors)

Steps 'Error - Null Value', 'Error - Multiple rows for same customerid' and 'Error - Invalid Date' adds a column 'Error Message' to the incoming data. This column specifies the exact issue with the record (For e.g. In 'Error - Multiple rows for same customerid' step, the error message inserted is 'Error - Multiple rows for same customerid'). This column is useful for the next step 'Error File'.

3.5.3.8 Error File

The rows with issues are updated in an MS Excel file (name: 'Customers_Errors.xlsx') and stored in the specified folder.

3.5.3.9 Delete

While using the 'Dimension lookup/update' step, a row with surrogate id 0 (zero) is added with other columns as NULL to the DWH table. This 'Delete' step helps to remove that row from the DWH table.

3.5.4 Products Transformation

This PDI job entry runs the 'products transformation' PDI transformation. This transformation step will transform and load data from 'products' table in the staging DB to 'products_dim' table in the DWH. Values for columns 'prod_surr_id', 'data_version', 'valid_from', 'valid_to' and 'current' for DWH will be added as per the specified criteria in the 'Dimension Lookup/Update' step. The 'products transformation' is as follows:

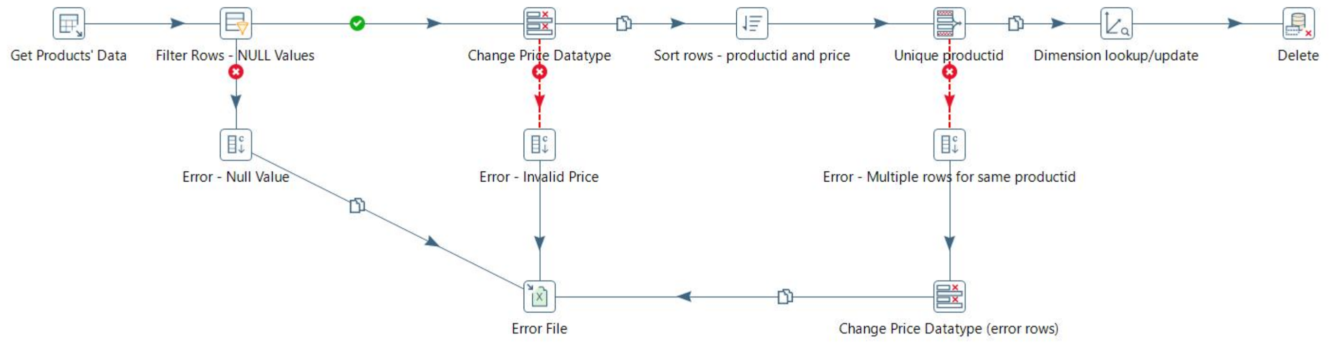


Fig 5: PDI transformation for transforming and loading products' data into the DWH

3.5.4.1 Get Products' Data

This steps selects all columns and rows from the 'products' table in staging DB.

3.5.4.2 Filter Rows - NULL Values

This step is used to filter rows that doesn't have any NULL values for the columns: 'productid', 'productname', 'category' and 'price'. These rows will be sent to the next step 'Change Price Datatype' and the rows with NULL values for any of these columns will be sent to 'Error - Null Value' step.

3.5.4.3 Change Price Datatype

The 'price' column is in string data type in format "##.##". This step converts it to 'BigNumber' data type and send to 'Sort rows - productid and price' step. If any rows have incorrect format resulting in error while converting, will be sent to 'Error - Invalid Price' step.

3.5.4.4 Sort rows - productid and price

This step sorts the rows as per the 'productid' column in ascending order and then by 'price' in descending order. Sorting is required for the next step.

3.5.4.5 Unique productid

This step checks whether there exists any multiple rows for same 'productid'. If no, then it passes the rows to the step 'Dimension lookup/update'. For product ids with multiple records, only the expensive (based on price) is forwarded to the 'Dimension lookup/update' step and the others will be sent to 'Error - Multiple rows for same productid' step.

3.5.4.6 Dimension lookup/update

This step inserts / updates the data to the 'products_dim' table in the DWH. This step looks for records of every product id with 'product_id' in 'products_dim' table of the DWH and inserts / updates accordingly as per SCD Type 2. Unique surrogate ids ('prod_surr_id') are added to the rows (even for rows with same product id). Values for columns:'version', 'valid_from', 'valid_to' and 'current' are inserted/updated by PDI depending upon presence of old records for same product id.

3.5.4.7 Error Steps (Addition of column for specifying the errors)

Steps ‘Error - Null Value’, ‘Error - Invalid Price’ and ‘Error - Multiple rows for same productid’ adds a column ‘Error Message’ to the incoming data. This column specifies the exact issue with the record (For e.g. In ‘Error - Multiple rows for same productid’ step, the error message inserted is ‘Multiple rows but same productid. Selected the most expensive product and returning other row/s.’). This column is useful for the ‘Error File’ step.

3.5.4.8 Change Price Datatype (error rows)

This step is only used for rows with errors identified after converting the price data type. This step converts the number data type to string data type to maintain uniform data type for every column within all rows of the error files. This will avoid any issues raised due to data types while writing to the MS Excel file.

3.5.4.9 Error File

The rows with issues are updated in an MS Excel file (name: ‘Products_Errors.xlsx’) and stored in the same folder of that of the ‘customers transformation’ error file.

3.5.4.10 Delete

As seen in the ‘Customers Transformation’, this ‘Delete’ step helps to remove the row with surrogate id 0 (zero) and other columns as NULL from the DWH table.

3.5.5 Transactions Transformation

This PDI job entry runs the ‘transactions transformation’ PDI transformation. This transformation step will transform and load data from ‘transactions’ table in the staging DB to ‘transactions_fact’ table in the DWH. During this transformation the customer ids and product ids from the staging data will be replaced by the surrogate keys generated for the rows for the dimension tables in the previous steps. Values for column ‘price’ will not be taken from the staging data but from ‘price’ column of ‘products_dim’ as per the surrogate id. The ‘transactions transformation’ is as follows:

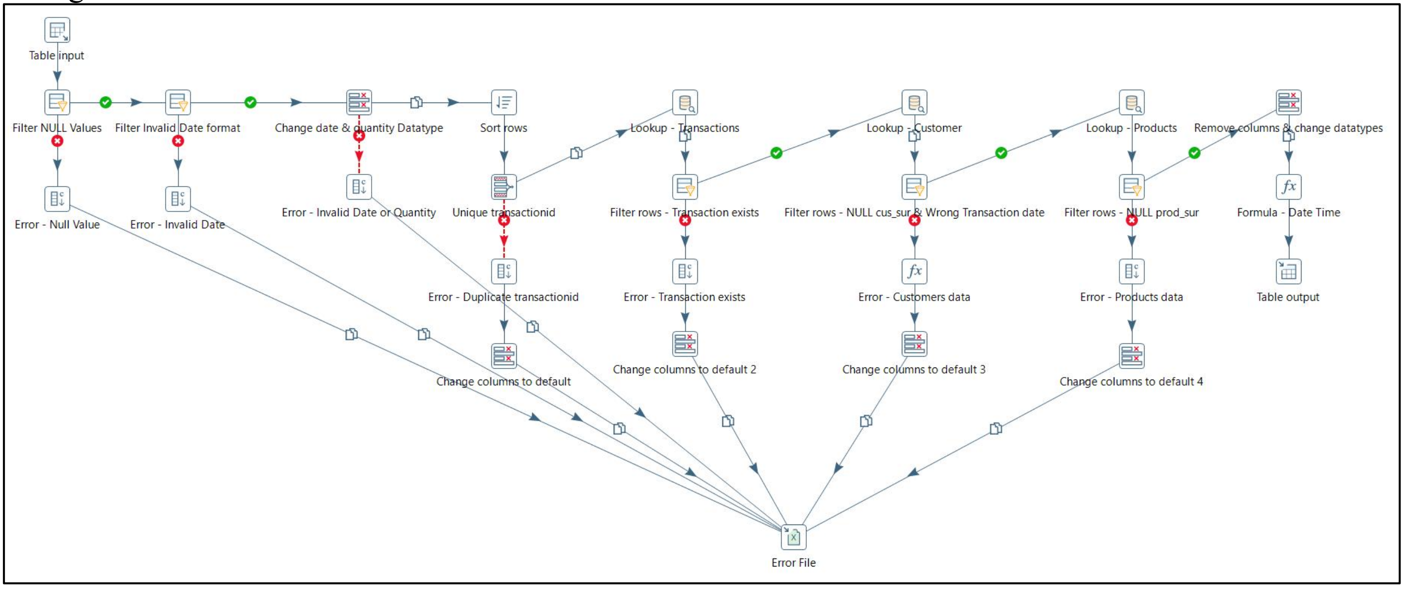


Table input
Filter NULL Values
Filter Invalid Date format
Change date & quantity Datatype
Sort rows
Lookup - Transactions
Lookup - Customer
Lookup - Products
Remove columns & change datatypes
Error - Null Value
Error - Invalid Date
Error - Invalid Date or Quantity
Unique transactionid
Filter rows - Transaction exists
Filter rows - NULL cus_sur & Wrong Transaction date
Filter rows - NULL prod_sur
Formula - Date Time
Error - Duplicate transactionid
Error - Transaction exists
Error - Customers data
Error - Products data
Table output
Change columns to default
Change columns to default 2
Change columns to default 3
Change columns to default 4
Error File

Fig 6: PDI transformation for transforming and loading transactions’ data into the DWH

3.5.5.1 Table input

This steps selects all columns and rows from the ‘transactions’ table in staging DB.

3.5.5.2 Filter NULL Values

This step is used to filter rows that doesn’t have any NULL values for the columns: ‘transactionid’, ‘customerid’, ‘productid’, ‘transactiondate’ and ‘quantity’. These rows will be sent to the next step ‘Filter Invalid Dateformat’ and the rows with NULL values for any of these columns will be sent to ‘Error - Null Value’ step.

3.5.5.3 Filter Invalid Date format

This step is used to filter rows with valid format for ‘transactiondate’ column. The ‘transactiondate’ column contains both date and time of the transaction. The date-time format used is “yyyy-MM-dd HH:mm:ss”. The rows with correct data will be sent to ‘Change date & quantity Datatype’ step and rows with invalid format will be sent to ‘Error - Invalid Date’ step. The regular expression used for this step is:

```
^(19|20|21)\d{2}-(0[1-9]|1[0-2])-(0[1-9]|[12]\d|3[01]) (0\d|1\d|2[0-
3]):([0-5]\d):([0-5]\d)$
```

3.5.5.4 Change date & quantity Datatype

This step converts the data type of ‘transactiondate’ and ‘‘quantity’ columns to date-time and ‘BigNumber’ data type respectively. The rows which are successfully processed will be send to ‘Sort rows’ step. If any rows have incorrect format resulting in error while converting, will be sent to ‘Error - Invalid Date or Quantity’ step.

3.5.5.5 Sort rows

This step sorts the rows as per the ‘transactionid’ and ‘transactiondate’ columns in ascending order. Sorting is required for the next step.

3.5.5.6 Unique transactionid

This step checks whether there exists any multiple rows for same ‘transactionid’. If no, then it passes the rows to the step ‘Lookup - Transactions’. For transaction ids with multiple records, only the first (based on transaction date/time) is forwarded to the ‘Lookup - Transactions’ step and the others will be sent to ‘Error - Duplicate transactionid’ step.

3.5.5.7 Lookup - Transactions

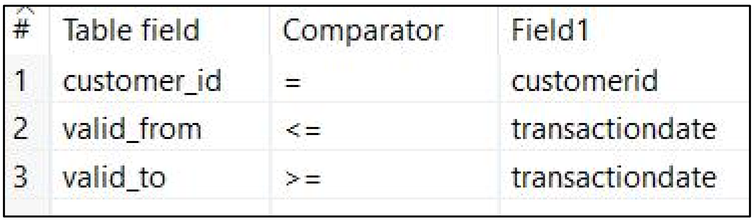
This step checks whether any transaction already exists in the DWH for the transaction ids in the stream. A new column ‘transaction_exists’ is added. If any transaction with the same id exists, then it will get the value of the transaction id and if it doesn’t exist, it will be NULL value. This can be used as a flag to filter rows with new transaction ids for the next step.

3.5.5.8 Filter rows - Transaction exists

This step sends rows of data with NULL value for column ‘transaction_exists’ (added in last step) to ‘Lookup - Customer’ step and the other rows are sent to ‘Error - Transaction exists’.

3.5.5.9 Lookup - Customer

This step retrieves the ‘cus_surr_id and ‘signup_date’ columns from the ‘customers_dim’ table in the DWH using the following conditions:



#	Table field	Comparator	Field1
1	customer_id	=	customerid
2	valid_from	<=	transactiondate
3	valid_to	>=	transactiondate

Fig 7: Customer surrogate id lookup conditions

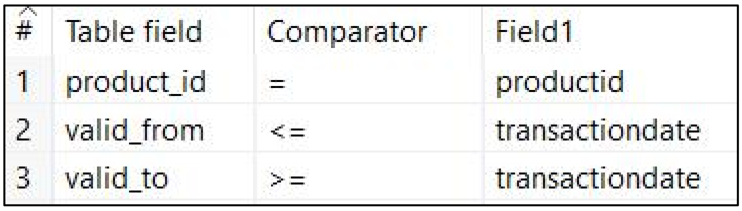
Here the first column contains the column names of the ‘customers_dim’ table and the third column contains the column names of the stream data. These conditions are used to get the correct surrogate id of the customer if more than one record is present for one customer id.

3.5.5.10 Filter rows - NULL cus_sur & Wrong Transaction date

This step filters rows with non NULL customer surrogate ids and transaction dates greater than the sign up date of the customer. The rows fulfilling these conditions will be sent to ‘Lookup - Products’ step and others will be sent to ‘Error - Customers data’ step.

3.5.5.11 Lookup - Products

This step retrieves the ‘prod_surr_id and ‘price’ columns from the ‘products_dim’ table in the DWH using the following conditions:



#	Table field	Comparator	Field1
1	product_id	=	productid
2	valid_from	<=	transactiondate
3	valid_to	>=	transactiondate

Fig 8: Product surrogate id lookup conditions

Here the first column contains the column names of the ‘products_dim’ table and the third column contains the column names of the stream data. These conditions are used to get the correct surrogate id of the product if more than one record is present for one customer id. The price column retrieved from the dimension table is named as ‘prod_lookup_price’ to avoid misinterpretation.

3.5.5.12 Filter rows - NULL prod_sur

This step filters rows with non NULL product surrogate ids. The rows fulfilling this condition will be sent to ‘Remove columns & change datatypes’ step and others will be sent to ‘Error - Products data’ step.

3.5.5.13 Remove columns & change datatypes

This step is used to remove columns which are not required for further processing and also changes data types of certain columns.
The following columns are removed in the step:

Sr. No.	Column name	Reason for removing
1	price	Will use 'prod_lookup_price' column instead of 'price'. The removal will help in reducing data redundancy.
2	totalvalue	'totalvalue' is equal to the product of the price and quantity. This can be evaluated in the BI tool if required. The removal will help in reducing data redundancy.
3	customerid	Surrogate id (cus_surr_id) will be used instaed of the customer id.
4	productid	Surrogate id (prod_surr_id) will be used instaed of the product id.
5	signupdate	Not required for further processing.

Data types of the following columns are changed for maintaining uniformity with the DWH database.

Sr. No.	Column name	Existing Data type	New Data type
1	cus_surr_id	BigNumber	Integer
2	prod_surr_id	BigNumber	Integer

3.5.5.14 Formula - Date Time

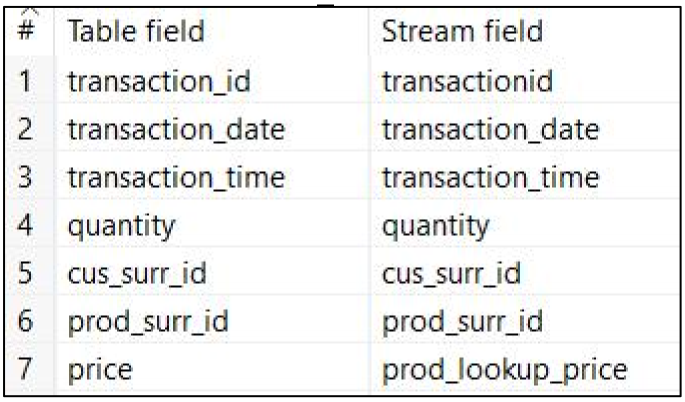
This step calculates two new columns 'Transaction_Date' and 'Transaction_Time' from the column 'transactiondate'. The formulae used are as follows:

➢ `Transaction_Date = DATE(YEAR([TransactionDate]); MONTH([TransactionDate]); DAY([TransactionDate]))`

➢ `Transaction_Time = TIME(HOUR([TransactionDate]); MINUTE([TransactionDate]); SECOND([TransactionDate]))`

3.5.5.15 Table output

This step inserts data into the fact table 'transactions_fact' in the DWH with the following mapping.



#	Table field	Stream field
1	transaction_id	transactionid
2	transaction_date	transaction_date
3	transaction_time	transaction_time
4	quantity	quantity
5	cus_surr_id	cus_surr_id
6	prod_surr_id	prod_surr_id
7	price	prod_lookup_price

Fig 9: Mapping stream data to transactions_fact table in the DWH for loading data

3.5.5.16 Error Steps (Addition of column for specifying the errors)

Steps ‘Error - Null Value’, ‘Error - Invalid Date’, ‘Error - Invalid Date or Quantity’, ‘Error - Duplicate transactionid’, ‘Error - Transaction exists’, ‘Error - Customers data’ and ‘Error - Products data’ adds a column ‘Error Message’ to the incoming data. This column specifies the exact issue with the record (For e.g. In ‘Error - Null Value’ step, the error message inserted is ‘Missing Value/s’). This column is useful for ‘Error File’ step.

3.5.5.17 Change columns to default (1 to 4)

These steps convert the data type of all columns (which where changed) to string data type to maintain uniform data type for every column within all rows of the error files. This will avoid any issues raised due to data types while writing to the MS Excel file.

3.5.5.18 Error File

The rows with issues are updated in an MS Excel file (‘Transactions_Errors.xlsx’) and stored in the same folder as the two previous transformations.

3.5.6 Delete filenames from result

In PDI transformations and jobs, while reading, writing or creating any files (.csv, .xlsx, .txt, etc.), the file names are added to result. This can be disabled by deselecting the option ‘Add filenames to result’ in the required steps of the transformations and jobs. But due to negligence, there may exist file names in the result. This PDI job entry helps in deleting filenames from result as per the given condition. Here no conditions are used resulting in deleting all filenames from result. This step is required because files names in result will be used to send only the error files using email.

3.5.7 Checks if files exist

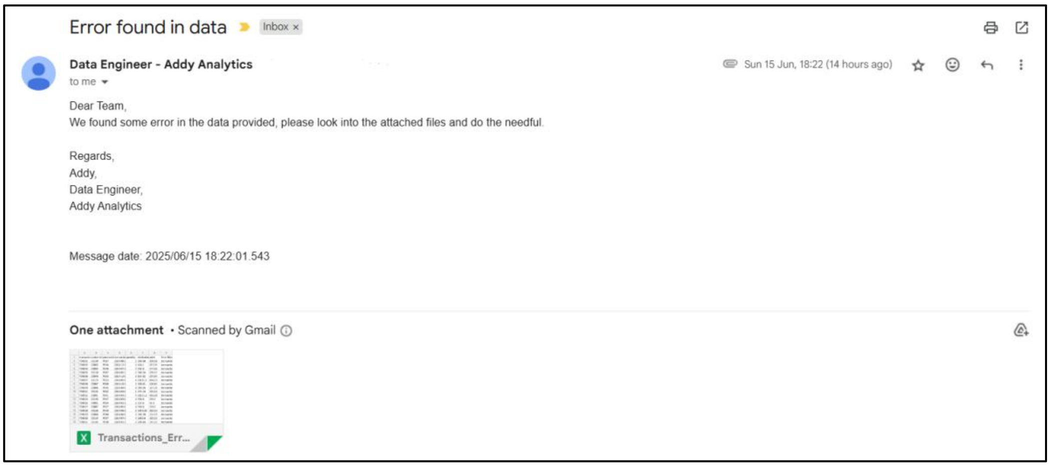
This PDI job entry makes sure that if any of the files exist in the specified folder. The specified folder is created for storing the MS Excel files containing the rows with error while performing the three transformations (‘customers transformation’, ‘products transformation’ and ‘transactions transformation’). If the folder contains any file, ‘Move files’ step is executed else if empty then ‘Abort job’ step is executed.

3.5.8 Move files

This PDI job entry , moves the error files to a new folder named as ‘Emailed Files’ to keep track of the files mailed to the concerned team. While moving the files, date and time of the execution of the step is added to the file name to keep track of the date and time. The option ‘Add files to result file name’ is checked to add these names to the result file. Only these files exist in the result file name as others have already been deleted in step ‘3.5.6 Delete filenames from result’.

3.5.9 Mail

In this PDI job entry, the MS Excel file(s) containing the rows with errors will be emailed to the concerned team with a custom message. To attach the file(s), the option ‘Attach file(s) to message?’ is enabled.
The recipient will receive an email with both the custom message and the file(s). The following screenshot shows an example of an email received by the concerned team.



Inbox ×
to me
Sun 15 Jun, 18:22 (14 hours ago)
One attachment • Scanned by Gmail
Transactions_Err...

Error found in data

Data Engineer - Addy Analytics

Dear Team,
We found some error in the data provided, please look into the attached files and do the needful.

Regards,
Addy,
Data Engineer,
Addy Analytics

Message date: 2025/06/15 18:22:01.543

Fig 10: Email received by the concerned team in case of any error in data set

3.5.10 Abort job

This PDI job entry aborts the job. This is done when there are no error files in the specified path.

3.6 Data Integration in Power BI

Tables from the DWH are imported into Power BI and custom tables are created within Power BI as required.

3.6.1 Importing data from DWH

The data stored in DWH is imported to Power BI using Power Query Editor. The imported tables are renamed as required and appropriate data types for each column are selected.
The 'customers_dim', 'products_dim' and 'transactions_fact' tables are imported to Power BI and renamed as 'Customers_Dimension', 'Products_Dimension' and 'Transactions_Fact' respectively.

3.6.2 Creating calendar table

A table consisting of dates and other required columns for the required date range is created in Power Query Editor. The 'Date' column serves as the primary key of the calendar table and will be referenced by other tables as a foreign key. The calendar table will help in time-based analysis.
The columns created in the calendar table are as follows:

Sr. No.	Column name	Column description	Data type
1	Date	Date	Date
2	Year	Year of the date in 'yyyy' format. Example: '**2019**' for date '01-01-2019'	Whole Number
3	Month Name	Month name without abbreviation. Example: '**January**' for date '01-01-2019'	Text
4	Day Name	Name of the weekday without abbreviation. Example: '**Tuesday**' for date '01-01-2019'	Text
5	Month Number	Number of the month of the year. Example: '**1**' for date '01-01-2019'	Whole Number
6	Start of Week	First day of the week for the date. Sunday is considered as starting of the week. Example: '**30-12-2018**' for date '01-01-2019'.	Date
7	Date in Text	Date in 'yyyy-MM-dd' format. Example: '**01-01-2019**'	Date

3.6.3 Creating table for measures

A dedicated table 'Measures Table' is created with no columns. This table will be used to store and organize all the calculated measures.

3.7 Data Modeling in Power BI

The dimension tables (including the calendar table) are joined with the fact table using ‘**One-to-Many**’ cardinality. The data model created in Power BI is as follows:

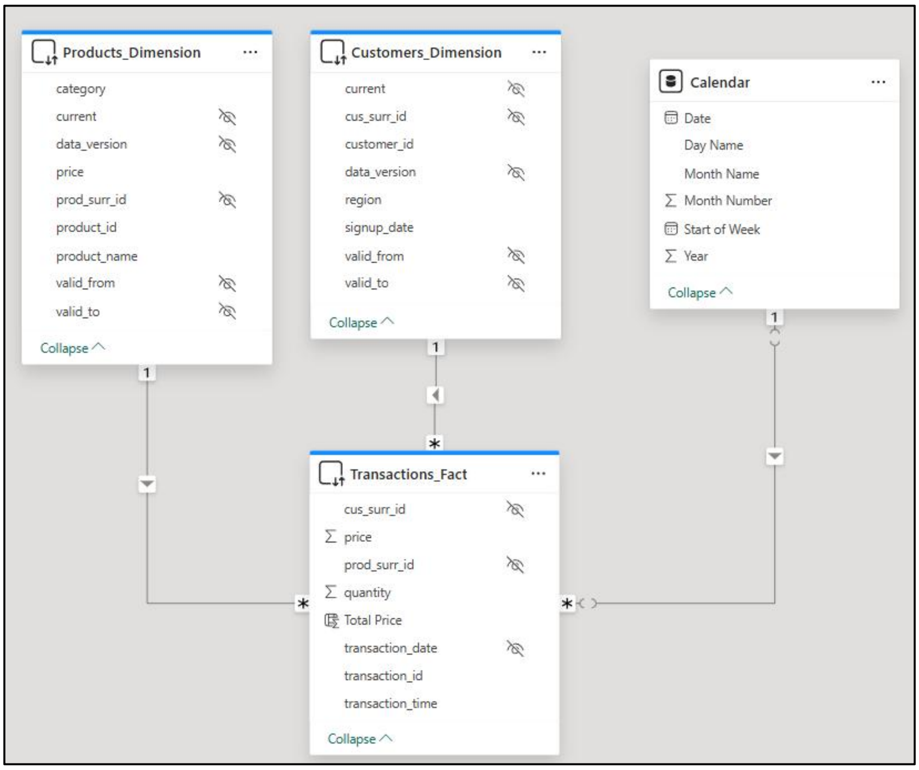


Fig 11: Data model in Power BI

The relationships among the tables are defined as:

Sr. No.	From: table (column)	To: table (column)
1	Customers_Dimension (cus_surr_id)	Transactions_Fact (cus_surr_id)
2	Products_Dimension (prod_surr_id)	Transactions_Fact (prod_surr_id)
3	Calendar (Date)	Transactions_Fact (transaction_date)

3.8 Calculated Columns and Measures

For creating visuals, there is a need for additional columns and aggregate measures in addition to the existing columns in the data model.

3.8.1 Calculated columns

It helps in creating columns using existing columns. The following calculated column is created in 'Transactions_Fact' table:

Sr. No.	**Calculated Column**	**Formula**
1	Total Price	`Transactions_Fact[quantity] * Transactions_Fact[price]`

3.8.2 Measures (explicit)

Both explicit and implicit measures can be used in visuals. For more control over the calculations, explicit measures are defined. The following explicit measures are defined:

Sr. No.	**Measure**	**Formula**

1	Active Customers	`CALCULATE(`
		`DISTINCTCOUNT(`
		`Customers_Dimension[customer_id]),`
		`Transactions_Fact`
		`)`

2	Active Customers Last Month	`CALCULATE([Active Customers], DATEADD('Calendar'[Date], -1, MONTH))`
3	Total Orders	`COUNT(Transactions_Fact[transaction_id])`
4	Orders Last Month	`CALCULATE([Total Orders], DATEADD('Calendar'[Date], -1, MONTH))`
5	Average Order per Customer	`DIVIDE([Total Orders], [Active Customers])`
6	Total Revenue	`SUMX(Transactions_Fact, Transactions_Fact[Total Price])`
7	Revenue Last Month	`CALCULATE([Total Revenue], DATEADD('Calendar'[Date], -1, MONTH))`

3.9 Create Data Visualization

The goal of the project is to develop a Power BI dashboard to analyze customer behaviour towards various products during the year 2024. To withstand this condition, a ‘Filter on all pages’ is set as ‘**Year is 2024**’. The dashboard is assumed to be viewed at the end of December 2024.

3.9.1 Visuals

Various visuals help in showing the story from different perspectives. The following are the visuals used in the dashboard to show the customer behaviour during the year 2024. The visuals are interactive, selecting any values of the visual will filter the dashboard data accordingly. The theme of the dashboard is chosen to be yellowish to align with the newly generated brand logo.

Fig 12: EverMart (e-commerce site) logo

3.9.1.1 KPI cards

KPI cards are used to show various important KPIs of the current month. Along with the values, the visual also displays the performance of the KPI for previous months using a trend axis and the goal for the current month. The ‘Goal’ is the target to beat the previous month’s performance. Following are the details of the KPI cards used in the dashboard:

Sr. No.	KPI Card	Measures used
1	Total Active Customers	1. For Main value: **Active Customers** 2. For Goal Value: **Active Customers Last Month**
2	Total Orders	1. For Main value: **Total Orders** 2. For Goal Value: **Orders Last Month**
3	Total Revenue	1. For Main value: **Total Revenue** 2. For Goal Value: **Revenue Last Month**

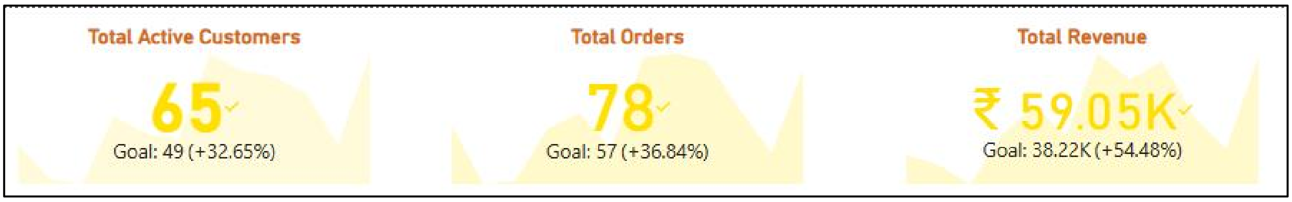


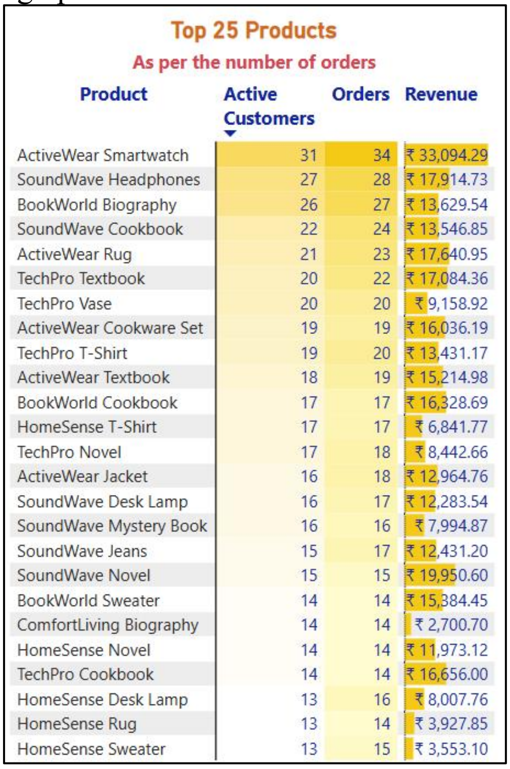
Fig 13: KPI card visuals used in dashboard

3.9.1.2 Matrix

The matrix visual is used to display the top 25 products of the year as per the number of orders. The matrix consists of the following columns:

Sr. No.	Matrix Columns	Description	Measure / Column used
1	Product	Name of the product.	Column: '**product_name**' from table: '**Products_Dimension**'
2	Active Customers	Total number of unique customers who bought the product.	Measure: '**Active Customers**'
3	Orders	Total number of orders placed for the product	Measure: '**Total Orders**'
4	Revenue	Total revenue generated by the product	Measure: '**Total Revenue**'

The matrix is sorted by 'Active Customers' count in descending order. Every value cell in the matrix has background colour effects depending upon the value.



Top 25 Products

As per the number of orders

Product	Active Customers	Orders	Revenue
ActiveWear Smartwatch	31	34	₹ 33,094.29
SoundWave Headphones	27	28	₹ 17,914.73
BookWorld Biography	26	27	₹ 13,629.54
SoundWave Cookbook	22	24	₹ 13,546.85
ActiveWear Rug	21	23	₹ 17,640.95
TechPro Textbook	20	22	₹ 17,084.36
TechPro Vase	20	20	₹ 9,158.92
ActiveWear Cookware Set	19	19	₹ 16,036.19
TechPro T-Shirt	19	20	₹ 13,431.17
ActiveWear Textbook	18	19	₹ 15,214.98
BookWorld Cookbook	17	17	₹ 16,328.69
HomeSense T-Shirt	17	17	₹ 6,841.77
TechPro Novel	17	18	₹ 8,442.66
ActiveWear Jacket	16	18	₹ 12,964.76
SoundWave Desk Lamp	16	17	₹ 12,283.54
SoundWave Mystery Book	16	16	₹ 7,994.87
SoundWave Jeans	15	17	₹ 12,431.20
SoundWave Novel	15	15	₹ 19,950.60
BookWorld Sweater	14	14	₹ 15,384.45
ComfortLiving Biography	14	14	₹ 2,700.70
HomeSense Novel	14	14	₹ 11,973.12
TechPro Cookbook	14	14	₹ 16,656.00
HomeSense Desk Lamp	13	16	₹ 8,007.76
HomeSense Rug	13	14	₹ 3,927.85
HomeSense Sweater	13	15	₹ 3,553.10

Fig 14: Matrix visual used in dashboard

3.9.1.3 Line and Clustered Column Chart

The line and clustered chart visual is used to depict the measures 'Total Revenue', 'Total Orders' and 'Active Customers'. The columns depict the revenue values and the lines depict the other two values. The visual shows the trend for every week of the year. The visual can be drilled up to view month wise and drilled down to view day wise. Upon hovering on any time period on the visual, user can find the values on the tooltip.

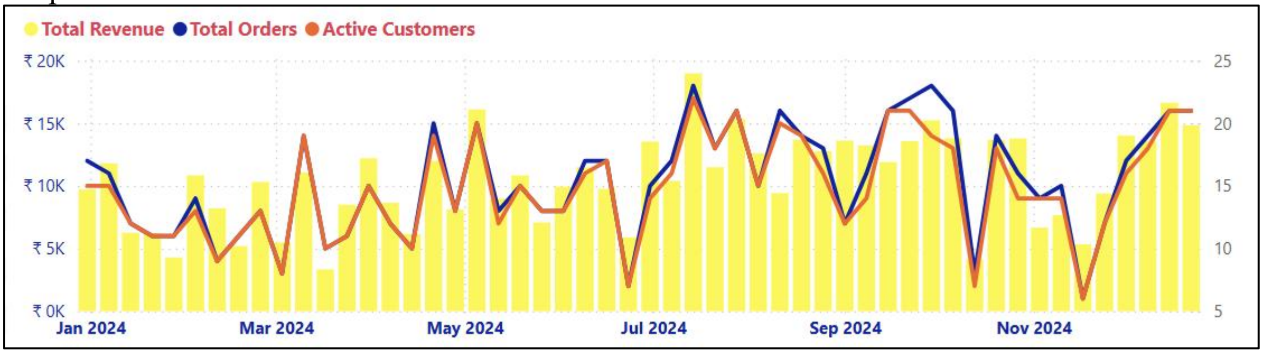


Fig 15: Line & clustered column chart visual used in dashboard

3.9.1.4 Pie chart

The pie chart visual displays the total orders by product categories. Upon hovering over any category, the tooltip provides insights.

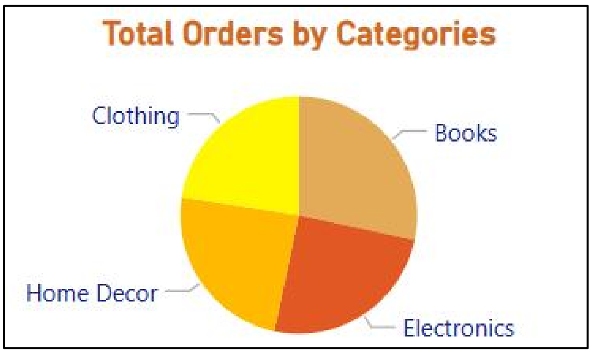


Fig 16: Pie chart visual used in dashboard

3.9.1.5 Treemap

The treemap visual depicts the total number of orders for each weekday. Upon hovering over any weekday, the tooltip provides insights.

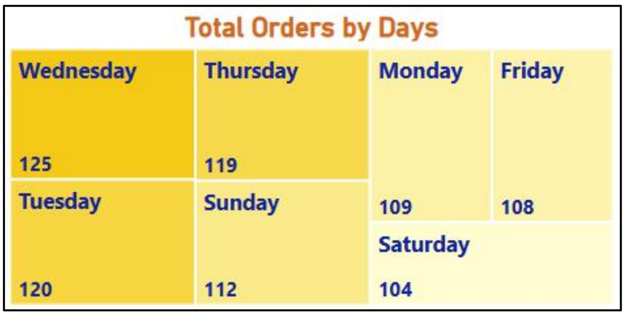


Fig 17: Treemap visual used in dashboard

3.9.2 Filter reset button

Upon clicking on visuals, the user can apply filters on the whole dashboard. To remove all the applied filters and return to the default view, user can use the 'Remove all applied filters' button available at the bottom right corner of the dashboard. This is achieved by adding the default page as a bookmark and then assigning the bookmark to the reset button.

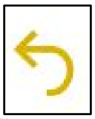


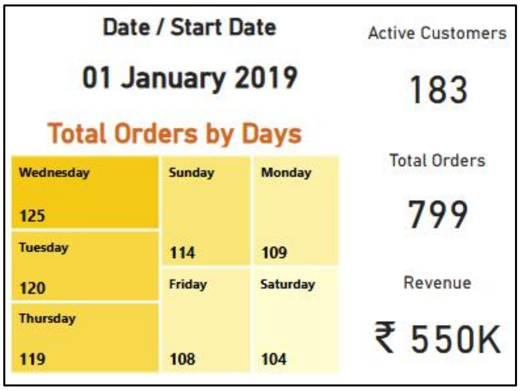
Fig 18: Filter reset button

3.9.3 Tooltips

Tooltips display values while hovering on the visuals. Custom tooltips provide essential insights in customized format for the required visual. For creating custom tooltips, tooltip pages have to be added and customized as required. The following are the two tooltips created for the visuals.

3.9.3.1 Tooltip for graph visual

The tooltip displays the date (or the start date of the week/month), count of active customers, count of total orders and total revenue generated for that date / period. It also includes a treemap to show the number of orders placed as per weekdays.



Date / Start Date

01 January 2019

Total Orders by Days

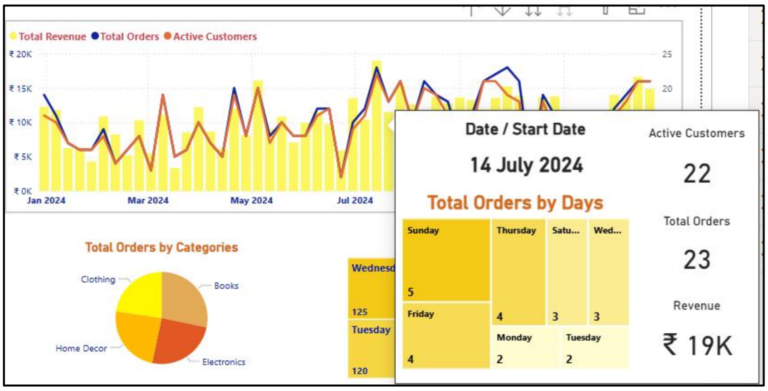
Wednesday 125	Sunday 114	Monday 109
Tuesday 120	Friday 108	Saturday 104
Thursday 119		

Active Customers 183

Total Orders 799

Revenue ₹ 550K

Fig 19: Tooltip for Graph visual



Date / Start Date

14 July 2024

Total Orders by Days

Sunday 5	Thursday 4	Satu... 3	Wed... 3
Friday 4	Monday 2	Tuesday 2	

Active Customers 22

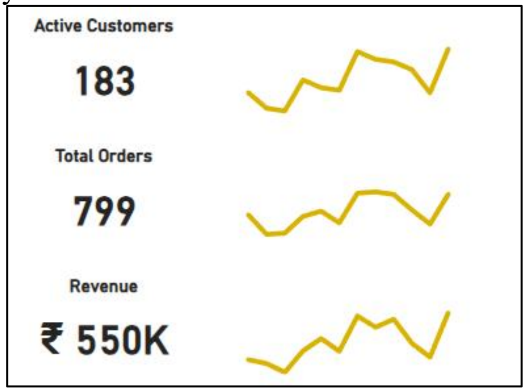
Total Orders 23

Revenue ₹ 19K

Fig 20: Example of tooltip for graph visual

3.9.3.2 Tooltip for matrix, pie chart and treemap visuals

The tooltip displays the count of active customers, count of total orders and total revenue generated for that selected product/category/weekday. It also shows the trends of these values using month as the X - axis.

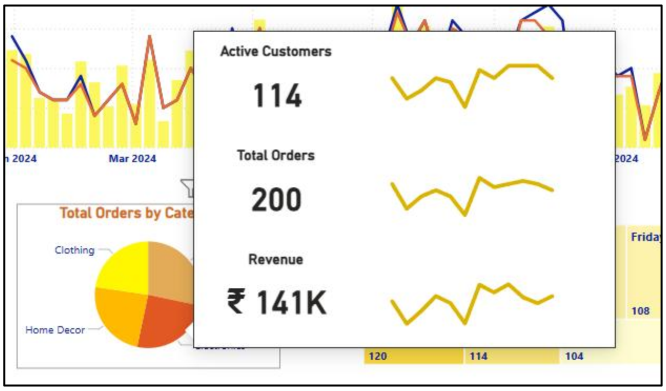


Active Customers 183

Total Orders 799

Revenue ₹ 550K

Fig 21: Tooltip for matrix, pie chart & treemap visuals



Active Customers 114

Total Orders 200

Revenue ₹ 141K

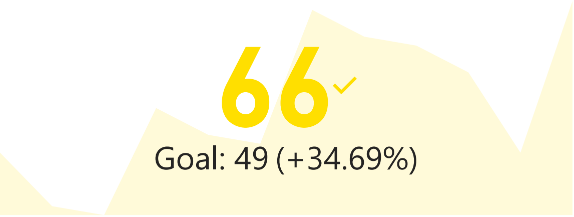
Fig 22: Example of tooltip for pie chart visual

3.9.4 Final dashboard

Following page represents the final dashboard page created with the above visuals:

EverMart

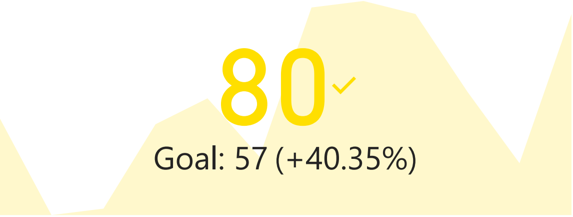
Total Active Customers



66 ✓

Goal: 49 (+34.69%)

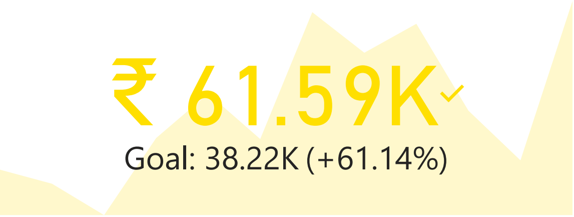
Total Orders



80 ✓

Goal: 57 (+40.35%)

Total Revenue



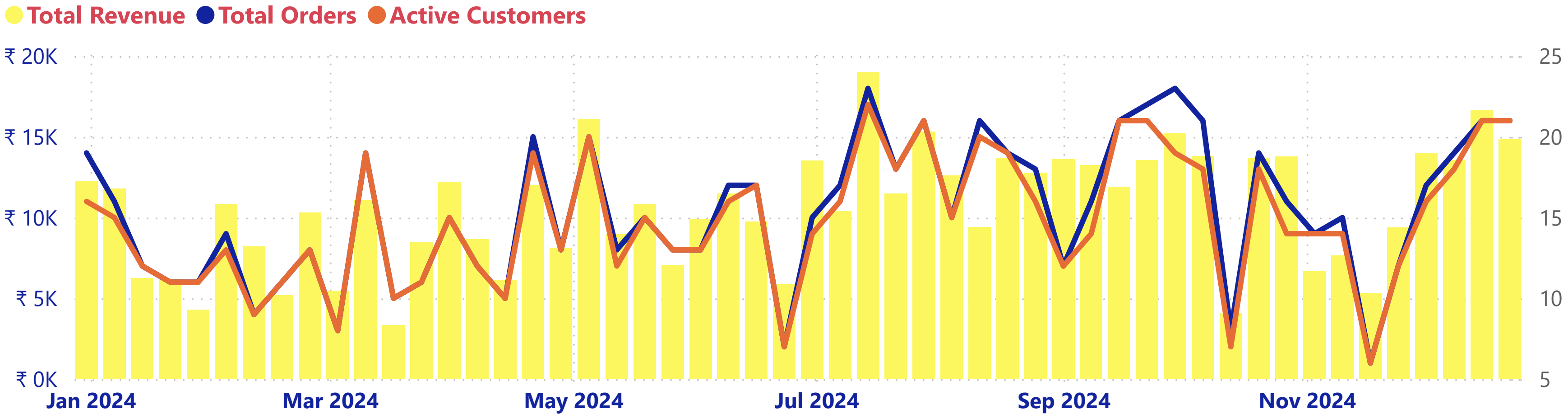
₹ 61.59K ✓

Goal: 38.22K (+61.14%)

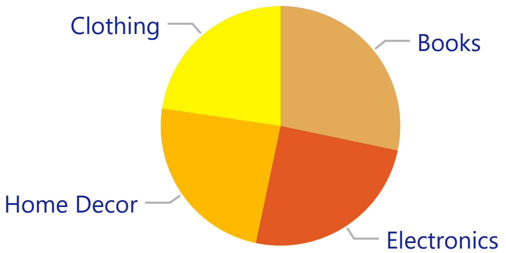
Top 25 Products

As per the number of orders

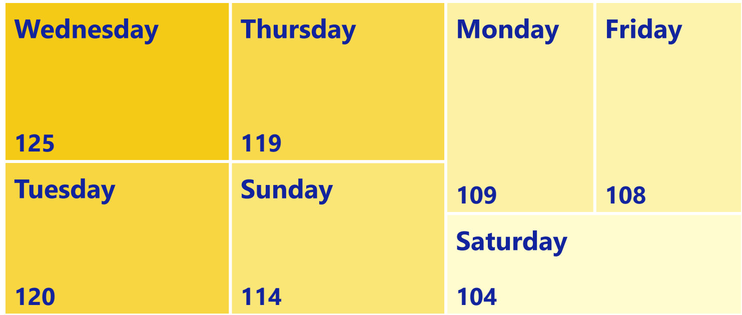
Product	Active Customers	Orders	Revenue
ActiveWear Smartwatch	31	34	₹ 33,094.29
SoundWave Headphones	27	28	₹ 17,914.73
BookWorld Biography	26	27	₹ 13,629.54
SoundWave Cookbook	22	24	₹ 13,546.85
ActiveWear Rug	21	23	₹ 17,640.95
TechPro Textbook	20	22	₹ 17,084.36
TechPro Vase	20	20	₹ 9,158.92
ActiveWear Cookware Set	19	19	₹ 16,036.19
TechPro T-Shirt	19	20	₹ 13,431.17
ActiveWear Textbook	18	19	₹ 15,214.98
BookWorld Cookbook	17	17	₹ 16,328.69
HomeSense T-Shirt	17	17	₹ 6,841.77
TechPro Novel	17	18	₹ 8,442.66
ActiveWear Jacket	16	19	₹ 14,550.12
SoundWave Desk Lamp	16	17	₹ 12,283.54
SoundWave Mystery Book	16	16	₹ 7,994.87
SoundWave Jeans	15	17	₹ 12,431.20
SoundWave Novel	15	15	₹ 19,950.60
BookWorld Sweater	14	14	₹ 15,384.45
ComfortLiving Biography	14	14	₹ 2,700.70
HomeSense Novel	14	14	₹ 11,973.12
TechPro Cookbook	14	14	₹ 16,656.00
HomeSense Desk Lamp	13	16	₹ 8,007.76
HomeSense Rug	13	14	₹ 3,927.85
HomeSense Sweater	13	15	₹ 3,553.10



Total Orders by Categories



Total Orders by Days



4. Summary of the Process

The goal was to develop a BI dashboard using Microsoft Power BI to analyze customer behaviour towards various products. Following files and tools were utilized to achieve the goal:

- **Transactional / Source Dataset from Kaggle**: Downloaded the dataset (.csv files) and treated it as the transactional data.

- **DBMS (PostgreSQL)**: Created required DBs and tables for both staging and DWH using PostgreSQL.

- **ETL Tool (Pentaho Data Integration - PDI)**: PDI helped in extracting data from transactional dataset to staging DB. PDI was extensively used in transforming and loading data from staging DB to the DWH.

- **BI Tool (Microsoft Power BI)**: Power BI was used to import data from the DWH, create additional tables, create data model and build the final dashboard.

5. Future Scope

Following are the future developments possible with the project:

- Scheduling of ETL process using batch files and schedulers like Microsoft Windows **Task Scheduler**.
- Development of customized and optimized dashboard layout for Mobile Phones.
- Addition of dashboard pages and visuals as per the requirement.
- Addition of data cleaning and validation steps if required.