

Pimpri Chinchwad Education Trust's
Pimpri Chinchwad College of Engineering



Department of Computer Engineering

Formative Assessment – I

Report On

“Data Preprocessing & Warehouse Design”

Under

Data mining and Warehousing

Academic year 2024-25

Submitted By:
Ranjeet Nath Chaudhary
122B1B045

Submitted To:
Dr. Swati V. Shinde

Data Preprocessing

Data preprocessing is preparing and transforming raw data into a clean, structured, and usable format before it is fed into a machine learning model or used for analysis. Preprocessing ensures that the data is in a consistent format and free of noise, allowing models to perform more accurately

Techniques For Data Preprocessing

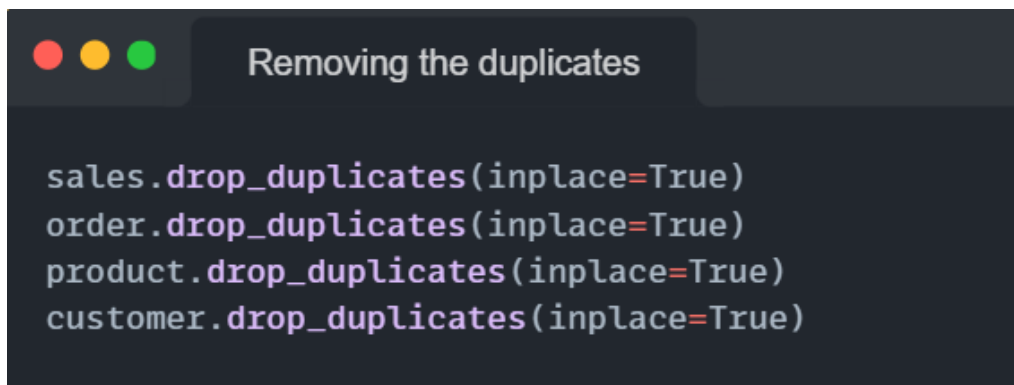
1. **Handling missing values:** - Techniques for identifying and addressing missing data to enhance dataset quality.
2. **Removing duplicates:** - Techniques for ensuring analysis accuracy and eliminating redundant records in the dataset.
3. **Encoding categorical variables** - Transferring categorical data into a numerical format for effective processing in machine learning.
4. **Handling outliers** - Identifying and treating outliers to improve the accuracy and reliability of data analysis.
5. **Normalization:** - Techniques used to scale the values of features in a dataset to a common range, typically between 0 and 1.

Dataset – Superstore Dataset

Steps performed:

1 - Removing the duplicates

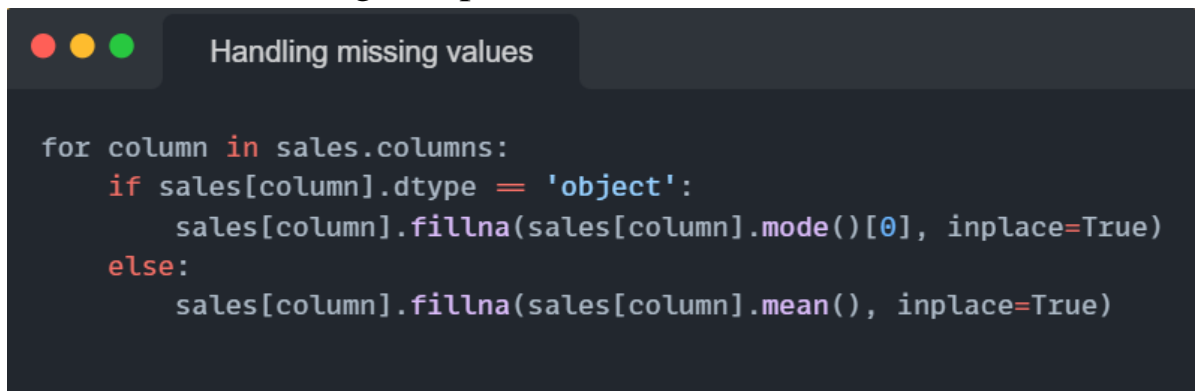
In this step, we check for duplicate rows in the sales, order, product, and customer Data Frames, remove them while keeping the first occurrence, and then verify that no duplicates remain to ensure clean data.



```
sales.drop_duplicates(inplace=True)
order.drop_duplicates(inplace=True)
product.drop_duplicates(inplace=True)
customer.drop_duplicates(inplace=True)
```

2 - Handling the missing values

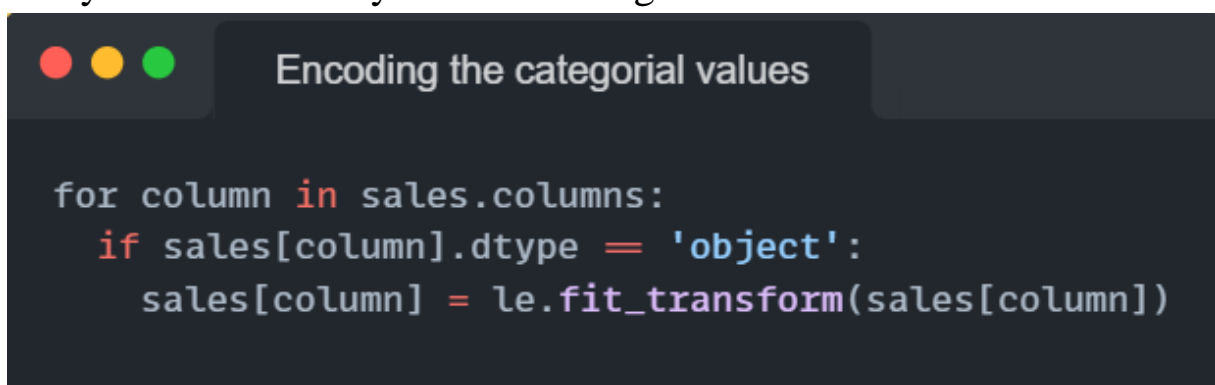
In this step, missing values are filled by replacing them with the mean for numerical columns and the mode for categorical columns in each Data Frame, ensuring complete data without bias.



```
for column in sales.columns:
    if sales[column].dtype == 'object':
        sales[column].fillna(sales[column].mode()[0], inplace=True)
    else:
        sales[column].fillna(sales[column].mean(), inplace=True)
```

3- Encoding the categorical values

In this step, we convert the categorical columns in the sales, order, product, and customer Data Frames into numerical values using Label Encoder. For each column with categorical data, we apply the encoder to assign a unique numeric value to each category, making the data ready for further analysis or modeling.



```
for column in sales.columns:
    if sales[column].dtype == 'object':
        sales[column] = le.fit_transform(sales[column])
```

4- Handling the outliers

This step uses the IQR method to handle outliers in the Sales, Quantity, Discount, and Profit columns of the sales Data Frame. It skips other Data Frames which has no numerical columns specified.

```
def handle_outliers_iqr(df, column):
    if not pd.api.types.is_numeric_dtype(df[column]):
        print(f"Column '{column}' is not numeric. Skipping outlier handling.")
        return # Skip non-numeric columns

    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    df[column] = np.where(df[column] < lower_bound, lower_bound, df[column])
    df[column] = np.where(df[column] > upper_bound, upper_bound, df[column])

numerical_columns_sales = ['Sales', 'Quantity', 'Discount', 'Profit']
for column in numerical_columns_sales:
    handle_outliers_iqr(sales, column)
```

5 - Normalizing using the Min-Max normalizing technique

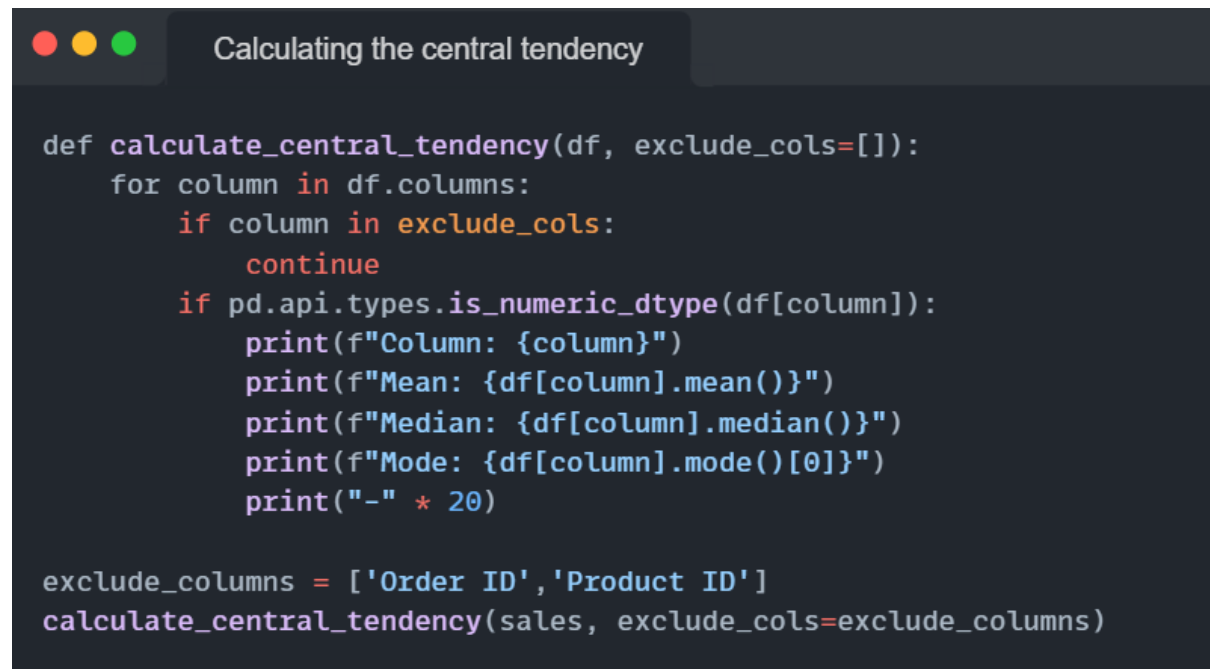
In this step, we normalize the Sales, Quantity, Discount, and Profit columns in the sales Data Frame using a Min-Max scaler. We transform the data to a standard scale, making it easier to analyze.

```
numerical_columns_sales = ['Sales', 'Quantity', 'Discount', 'Profit']

sales[numerical_columns_sales] = scaler.fit_transform(
    sales[numerical_columns_sales]
)
```

6 - Calculating the central tendency

In this step, a function is defined to calculate and print the mean, median, and mode for numeric columns in a Data Frame and provide meaningful statistical information.

A code editor window with a dark background and a title bar that says "Calculating the central tendency". The code is written in Python and defines a function to calculate the mean, median, and mode for numeric columns in a DataFrame, while excluding certain columns. The code is as follows:

```
def calculate_central_tendency(df, exclude_cols=[]):
    for column in df.columns:
        if column in exclude_cols:
            continue
        if pd.api.types.is_numeric_dtype(df[column]):
            print(f"Column: {column}")
            print(f"Mean: {df[column].mean()}")
            print(f"Median: {df[column].median()}")
            print(f"Mode: {df[column].mode()[0]}")
            print("-" * 20)

exclude_columns = ['Order ID', 'Product ID']
calculate_central_tendency(sales, exclude_cols=exclude_columns)
```

Data Warehouse

A data warehouse is a centralized storage system that organizes and manages large volumes of data from various sources. It helps optimize data organization for effective analysis and retrieval.

Data Modeling Schemas in Data Warehousing

Data modeling is the process of organizing data to make it easy to find and use. The types of schemas of data modeling are:

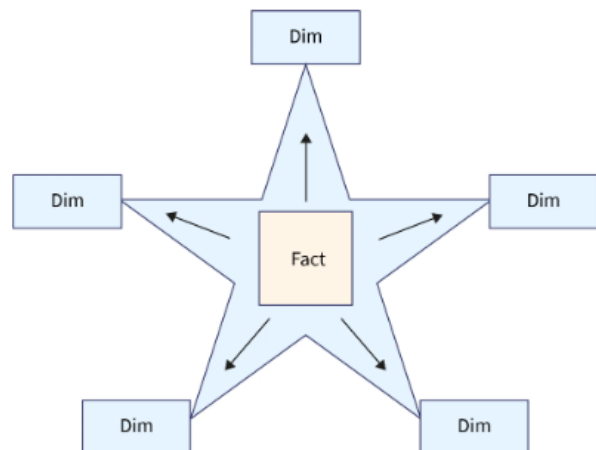
- Star Schema
- Snowflake Schema

Star Schema

In a star schema, data is organized into a central fact table that contains the measures of interest, surrounded by dimension tables that describe the attributes of the measures.

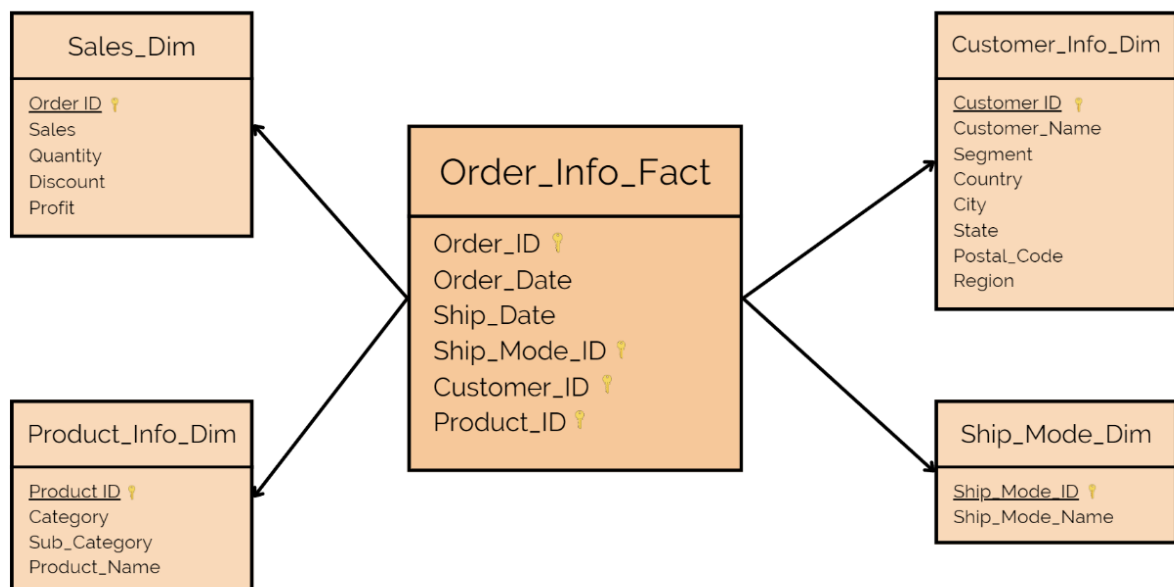
Characteristics:

- Simplified data model.
- Faster query performance due to fewer joins.



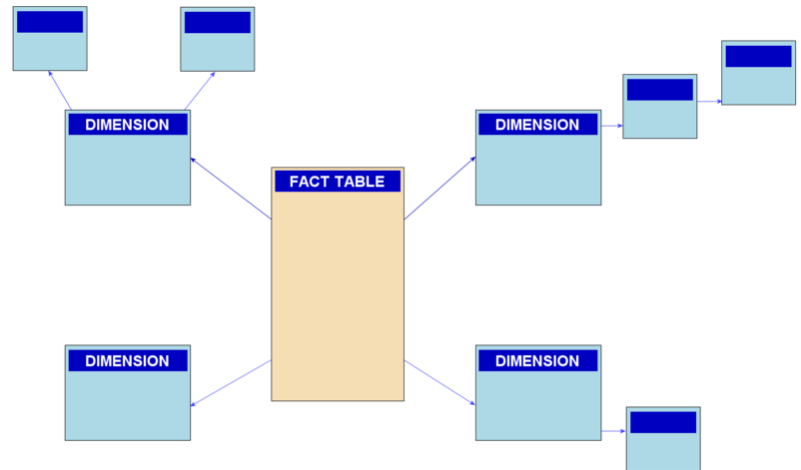
Star Schema of Dataset:

Star Schema



Snowflake Schema

A snowflake schema is a more normalized version of the star schema where dimension tables are split into sub-tables.

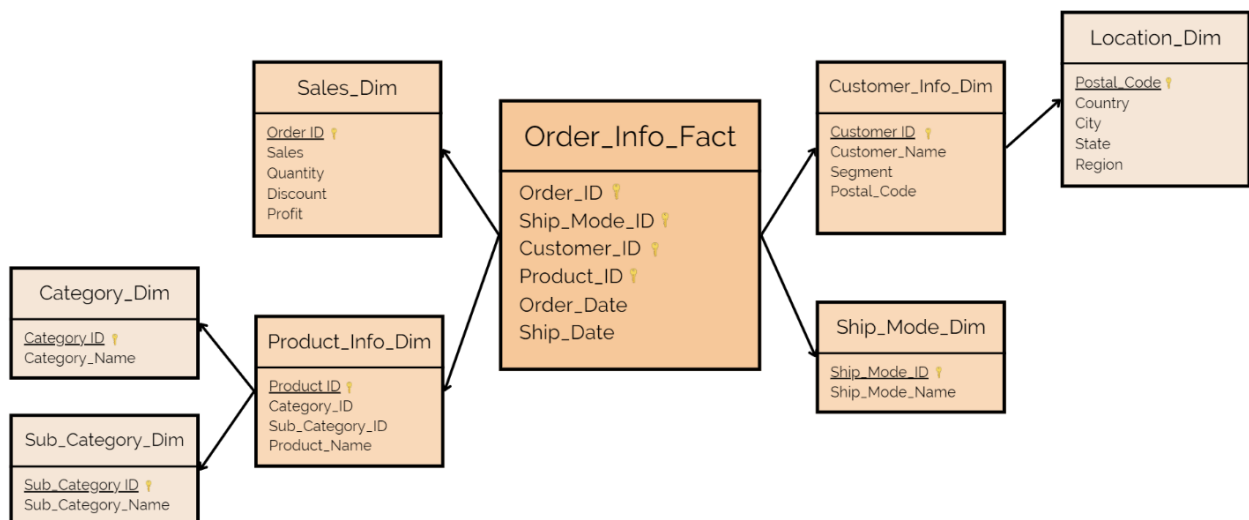


Characteristics:

- More complex due to additional tables.
- Reduces redundancy by normalizing data.

Snowflake Schema for Dataset:

Snowflake Schema



Conclusion

Hence, we transformed a collection of datasets into both star and snowflake schemas using data preprocessing and normalization techniques.