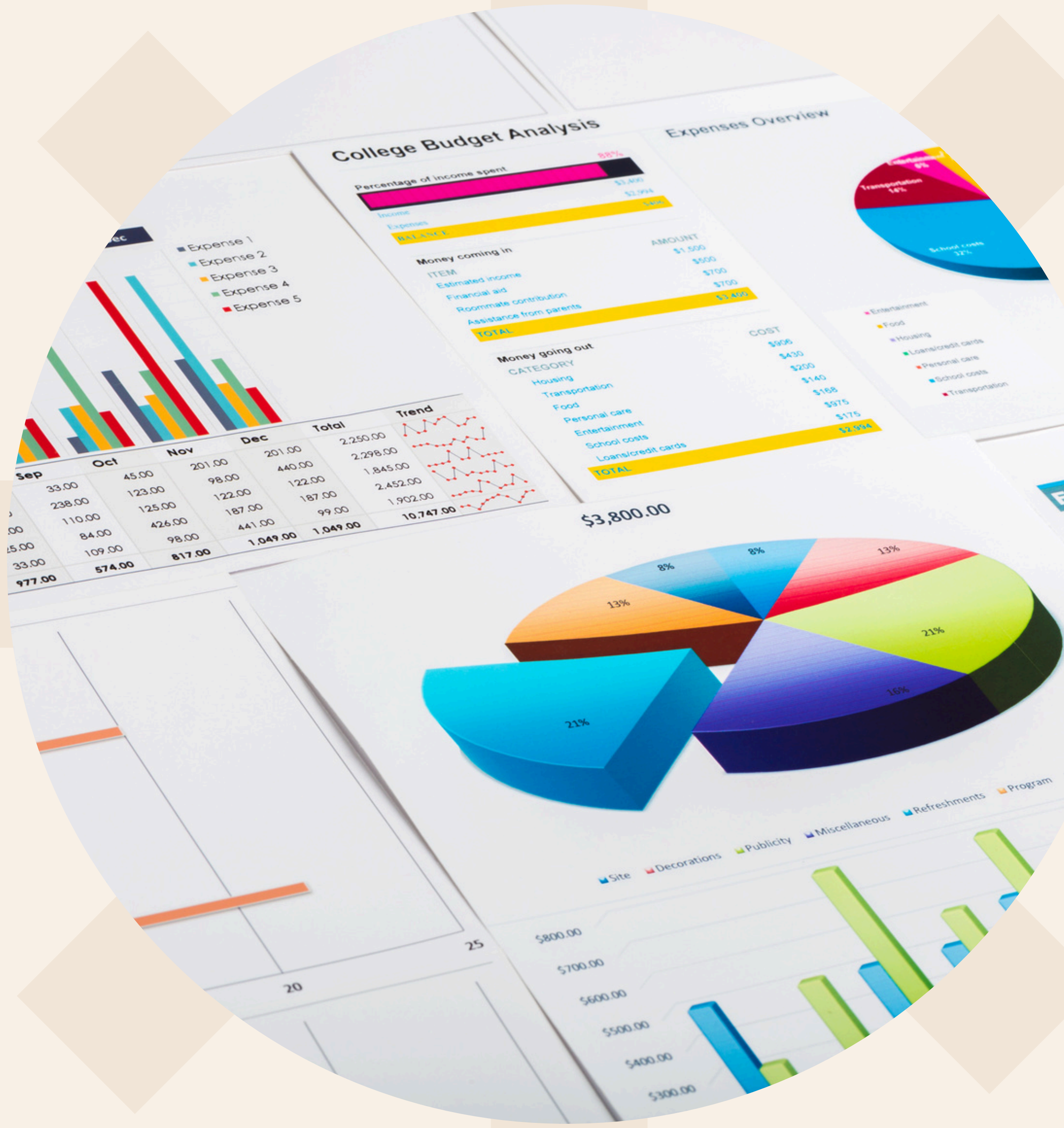
Exploratory Data Analysis - Warehouse And Retail Sales

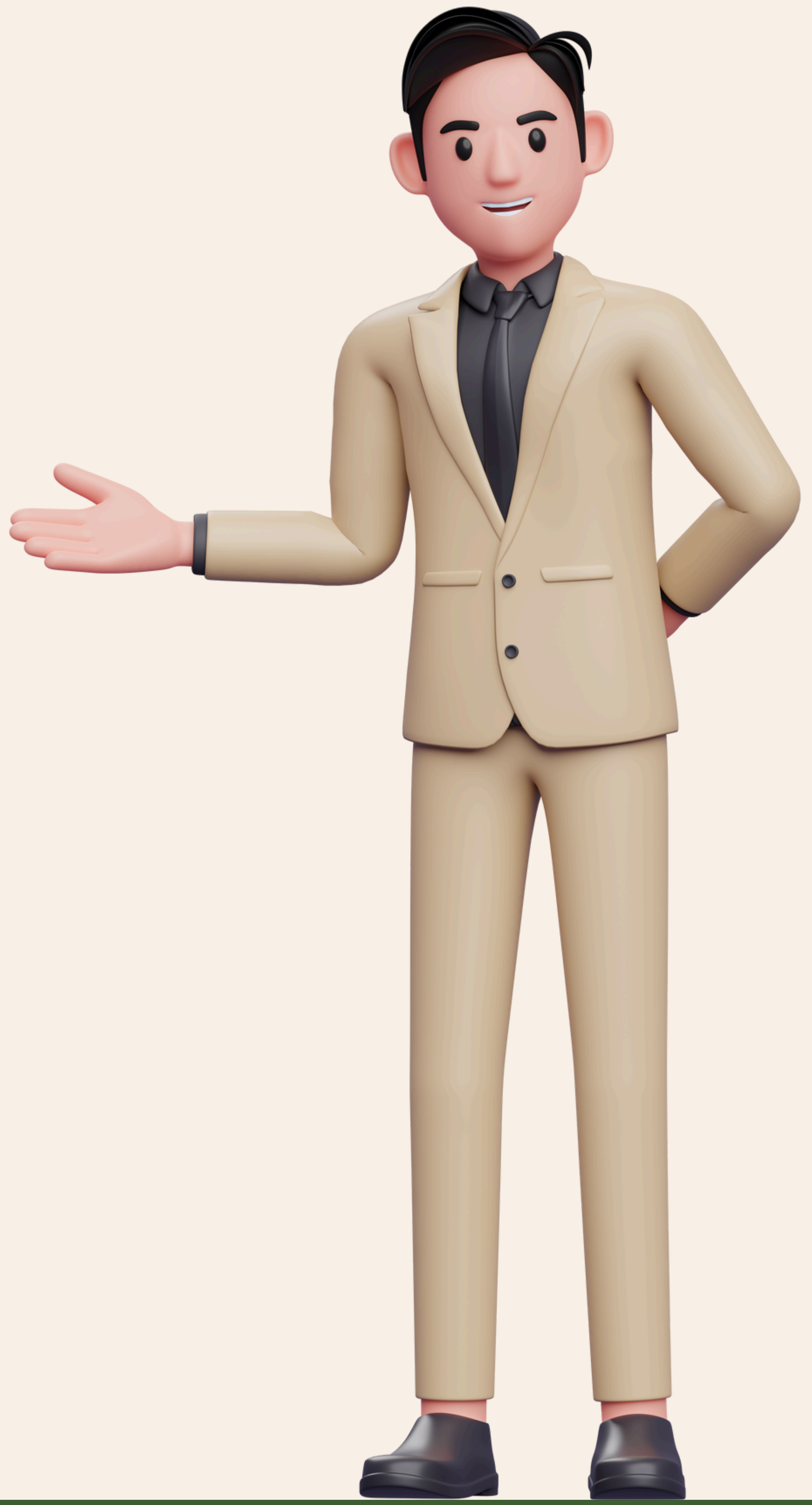


Marcell Franclin

Kelas: Faculty Of Data | #DigitalSkillFair40
Program @dibimbing.id

INTRODUCTION

In today's data-driven world, conducting thorough data analysis is essential for extracting meaningful insights that can support better decision-making. This project focuses on performing Exploratory Data Analysis (EDA) and data pre-processing on the Warehouse and Retail Sales dataset using Python. As part of the preprocessing stage, several key steps are carried out, including the removal of duplicate records, handling of missing values, and encoding of categorical variables. These steps are crucial to ensure that the dataset is clean, consistent, and ready for further analysis or modeling. By preparing the data properly, we can enhance the accuracy and reliability of any insights or predictions derived from it.



Project Goals

The main goals of this project are:

- **To explore the structure and characteristics of the Warehouse and Retail Sales dataset** through Exploratory Data Analysis (EDA).
- **To clean and prepare the data** by removing duplicate entries and handling missing values to improve data quality.
- **To encode categorical variables** so the dataset can be properly used in future analytical or machine learning processes.
- **To uncover initial insights or patterns** that may support business understanding and decision-making.
- **To establish a solid, clean datase**t as a foundation for further modeling or predictive analysis.

Data Overview

	YEAR	MONTH	SUPPLIER	ITEM CODE	ITEM DESCRIPTION	ITEM TYPE	RETAIL SALES	RETAIL TRANSFERS	WAREHOUSE SALES
0	2020	1	REPUBLIC NATIONAL DISTRIBUTING CO	100009	BOOTLEG RED - 750ML	WINE	0.00	0.0	2.00
1	2020	1	PWSWN INC	100024	MOMENT DE PLAISIR - 750ML	WINE	0.00	1.0	4.00
2	2020	1	RELIABLE CHURCHILL LLLP	1001	S SMITH ORGANIC PEAR CIDER - 18.7OZ	BEER	0.00	0.0	1.00
3	2020	1	LANTERNA DISTRIBUTORS INC	100145	SCHLINK HAUS KABINETT - 750ML	WINE	0.00	0.0	1.00
4	2020	1	DIONYSOS IMPORTS INC	100293	SANTORINI GAVALA WHITE - 750ML	WINE	0.82	0.0	0.00
...	...	...	...	...	...	...	...	...	...
307640	2020	9	LEGENDS LTD	99753	DUTCHESS DE BOURGOGNE NR - 750ML	BEER	0.00	0.0	5.00
307641	2020	9	ANHEUSER BUSCH INC	9997	HOEGAARDEN 4/6NR - 12OZ	BEER	66.12	37.0	240.75
307642	2020	9	COASTAL BREWING COMPANY LLC	99970	DOMINION OAK BARREL STOUT 4/6 NR - 12OZ	BEER	2.25	0.0	0.00
307643	2020	9	BOSTON BEER CORPORATION	99990	SAM ADAMS SUMMER VARIETY 12PK NR	BEER	20.50	0.0	0.00
307644	2020	9	NaN	WC	WINE CREDIT	REF	0.00	0.0	-70.00

307645 rows × 9 columns

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 478 entries, 0 to 477
Data columns (total 22 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   brand                      478 non-null    object
 1   model                      477 non-null    object
 2   top_speed_kmh              478 non-null    int64
 3   battery_capacity_kWh       478 non-null    float64
 4   battery_type               478 non-null    object
 5   number_of_cells            276 non-null    float64
 6   torque_nm                  471 non-null    float64
 7   efficiency_wh_per_km       478 non-null    int64
 8   range_km                   478 non-null    int64
 9   acceleration_0_100_s       478 non-null    float64
 10  fast_charging_power_kw_dc  477 non-null    float64
 11  fast_charge_port           477 non-null    object
 12  towing_capacity_kg         452 non-null    float64
 13  cargo_volume_l             477 non-null    object
 14  seats                      478 non-null    int64
 15  drivetrain                 478 non-null    object
 16  segment                    478 non-null    object
 17  length_mm                  478 non-null    int64
 18  width_mm                   478 non-null    int64
 19  height_mm                  478 non-null    int64
 20  car_body_type              478 non-null    object
 21  source_url                 478 non-null    object
dtypes: float64(6), int64(7), object(9)
memory usage: 82.3+ KB
```

Data Overview

The Warehouse and Retail Sales dataset consists of **307,645 rows and 9 columns**, capturing transaction records of various alcoholic beverages. Each row represents a single transaction, containing information such as **YEAR and MONTH, SUPPLIER name, product details** (including item code, description, and type), and **sales data** (retail sales, retail transfers, and warehouse sales). The dataset also includes some missing and potentially invalid values, which will be handled during the pre-processing phase.

Handling Missing Values and Duplicates

Missing Value Summary

```
# mengecek missing value
df.isna().sum()
```

	0
YEAR	0
MONTH	0
SUPPLIER	167
ITEM CODE	0
ITEM DESCRIPTION	0
ITEM TYPE	1
RETAIL SALES	3
RETAIL TRANSFERS	0
WAREHOUSE SALES	0

dtype: int64

Based on the result of the df.isna().sum() function, out of 9 columns in the dataset, 3 columns contain missing values:

- SUPPLIER with 167 missing entries
- ITEM TYPE with 1 missing entry
- RETAIL SALES with 3 missing entries

The remaining 6 columns are free of missing data, making them ready for further analysis after data cleaning is applied.

Descriptive Statistics

```python
# cek statiscial summary
df.describe()
```

	YEAR	MONTH	RETAIL SALES	RETAIL TRANSFERS	WAREHOUSE SALES
count	307645.000000	307645.000000	307642.000000	307645.000000	307645.000000
mean	2018.438525	6.423862	7.024071	6.936465	25.294597
std	1.083061	3.461812	30.986238	30.237195	249.916798
min	2017.000000	1.000000	-6.490000	-38.490000	-7800.000000
25%	2017.000000	3.000000	0.000000	0.000000	0.000000
50%	2019.000000	7.000000	0.320000	0.000000	1.000000
75%	2019.000000	9.000000	3.267500	3.000000	5.000000
max	2020.000000	12.000000	2739.000000	1990.830000	18317.000000

This summary shows the basic descriptive statistics for numerical columns such as RETAIL SALES, RETAIL TRANSFERS, and WAREHOUSE SALES.

- The dataset spans from 2017 to 2020 with all months represented.
- The mean values of sales columns are relatively low compared to the maximum values, indicating many small-value transactions.
- Several columns contain zero values, which appear frequently based on the 25th and 50th percentiles.

This helps provide a quick overview of the data before performing further cleaning.

Descriptive Statistics

```
# cek statiscial summary
df['SUPPLIER'].describe()
```

	SUPPLIER
count	307478
unique	396
top	REPUBLIC NATIONAL DISTRIBUTING CO
freq	20995

dtype: object

There are 396 unique suppliers in the dataset, with a total of 307,478 entries.
The most frequent supplier is REPUBLIC NATIONAL DISTRIBUTING CO, appearing 20,995 times.

```
# cek statiscial summary
df['ITEM TYPE'].describe()
```

	ITEM TYPE
count	307644
unique	8
top	WINE
freq	187640

dtype: object

There are 8 unique item types in the dataset, with a total of 307,644 entries.
The most common item type is WINE, appearing 187,640 times.

Handling Missing Values

```
# Mengatasi missing value

df['SUPPLIER'].fillna('Unknown', inplace=True) # handle missing value pada kolom supplier
df['ITEM TYPE'].fillna(df['ITEM TYPE'].mode()[0], inplace=True) # handle missing value pada kolom ITEM TYPE
df['RETAIL SALES'].fillna(df['RETAIL SALES'].median(), inplace=True) # handle missing value pada kolom RETAIL SALES
```

```
/tmp/ipython-input-14-2026777179.py:5: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation in

  df['RETAIL SALES'].fillna(df['RETAIL SALES'].median(), inplace=True) # handle missing value pada kolom RETAIL SALES
```

To ensure data quality, missing values in several columns were handled using appropriate imputation techniques:

- SUPPLIER: Missing entries were filled with "Unknown", assuming these records had no specified supplier.
- ITEM TYPE: Missing values were replaced with the most frequent item type (mode), assuming it represents the most likely value.
- RETAIL SALES: Missing values were filled using the median, to avoid distortion from extreme values and better represent typical sales.

Handling Duplicate Data

4. Mengatasi Duplikat Data

```
[ ] # Cek duplikat
    print("Jumlah duplikat:", df.duplicated().sum())
```

```
Jumlah duplikat: 0
```

Hasil Tidak ada data duplikat jadi tidak perlu drop data duplikat

Duplicate records were checked using the .duplicated() function, and the result showed zero duplicates in the dataset. This means the data is already clean in terms of duplication, and no further action was necessary for this step.

Data Preprocessing: Encoding

Categorical Data Overview

```python
# Cek nilai unik dan distribusi
print("SUPPLIER Unique Values:", df['SUPPLIER'].nunique())
print("\nITEM TYPE Distribution:")
print(df['ITEM TYPE'].value_counts(normalize=True)*100)


plt.figure(figsize=(10,6))
df['ITEM TYPE'].value_counts().plot(kind='bar', color='skyblue')
plt.title('Distribusi Tipe Produk')
plt.ylabel('Jumlah')
plt.xticks(rotation=45)
plt.show()
```

```
SUPPLIER Unique Values: 397

ITEM TYPE Distribution:
ITEM TYPE
WINE            60.992703
LIQUOR          21.098994
BEER            13.786345
KEGS             3.297957
NON-ALCOHOL      0.620195
STR_SUPPLIES     0.131645
REF              0.041281
DUNNAGE          0.030880
Name: proportion, dtype: float64
```

There are **397 unique suppliers** in the dataset. The **ITEM TYPE column contains 8 unique categories**, with the most common being WINE (61%), followed by LIQUOR (21%) and BEER (14%). Since ITEM TYPE has a small number of distinct categories and no inherent order, it was encoded using **One-Hot Encoding**. On the other hand, the SUPPLIER column has high cardinality with 396 unique values, so it was encoded using **Frequency Encoding** to simplify the data while preserving its distribution.

Encoding Results

```
# One-Hot Encoding untuk ITEM TYPE
df = pd.get_dummies(df, columns=['ITEM TYPE'], prefix='Type')

# Frequency Encoding untuk SUPPLIER
supplier_freq = df['SUPPLIER'].value_counts(normalize=True)
df['SUPPLIER_FREQ'] = df['SUPPLIER'].map(supplier_freq)

# Hasil encoding
print(df[['SUPPLIER', 'SUPPLIER_FREQ']].head())
```

```
                         SUPPLIER  SUPPLIER_FREQ
0  REPUBLIC NATIONAL DISTRIBUTING CO       0.068244
1                        PWSWN INC       0.009410
2             RELIABLE CHURCHILL LLLP       0.022659
3          LANTERNA DISTRIBUTORS INC       0.011718
4               DIONYSOS IMPORTS INC       0.013590
```

To convert categorical columns into numerical format:

- ITEM TYPE was encoded using One-Hot Encoding because it only has 8 distinct categories without any order.
- SUPPLIER was encoded using Frequency Encoding due to its high cardinality (396 unique values). Each supplier is now represented by the frequency of its occurrence in the dataset.

Encoding Results

```
print("\nMissing Value Setelah Encoding:")
print(df.isnull().sum())
```

```
Missing Value Setelah Encoding:
YEAR                  0
MONTH                 0
SUPPLIER              0
ITEM CODE             0
ITEM DESCRIPTION      0
RETAIL SALES          0
RETAIL TRANSFERS      0
WAREHOUSE SALES       0
Type_BEER             0
Type_DUNNAGE          0
Type_KEGS             0
Type_LIQUOR           0
Type_NON-ALCOHOL      0
Type_REF              0
Type_STR_SUPPLIES     0
Type_WINE             0
SUPPLIER_FREQ         0
dtype: int64
```

After the encoding process, we checked for missing values in each column. The result shows zero missing values across all columns, meaning the data is complete. So, the encoding steps (One-Hot for ITEM TYPE and Frequency Encoding for SUPPLIER) worked without causing any data loss.

Contact

Marcell Franclin

+62 882 1446 0991

marcellfranclin@gmail.com

Thank You