# Week 4 - Handling data

## 1. Exploratory Data Analysis (EDA)

- **Loading data** and checking columns, missing values, data types:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load datasets
orders = pd.read_csv('/mnt/data/olist_orders_dataset.csv')
order_items = pd.read_csv('/mnt/data/olist_order_items_dataset.csv')
products = pd.read_csv('/mnt/data/olist_products_dataset.csv')
customers = pd.read_csv('/mnt/data/olist_customers_dataset.csv')
sellers = pd.read_csv('/mnt/data/olist_sellers_dataset.csv')
payments = pd.read_csv('/mnt/data/olist_order_payments_dataset.csv')
reviews = pd.read_csv('/mnt/data/olist_order_reviews_dataset.csv')
geo = pd.read_csv('/mnt/data/olist_geolocation_dataset.csv')
product_cat =
pd.read_csv('/mnt/data/product_category_name_translation.csv')

# Check missing values
print(products.isnull().sum())
print(orders.isnull().sum())
```

## 2. Data Cleaning and Preprocessing

- Handle missing values:

```
# Fill missing product measurements with median values
products['product_weight_g'].fillna(products['product_weight_g'].medi
an(), inplace=True)
products['product_length_cm'].fillna(products['product_length_cm'].me
dian(), inplace=True)
products['product_height_cm'].fillna(products['product_height_cm'].me
dian(), inplace=True)
products['product_width_cm'].fillna(products['product_width_cm'].medi
an(), inplace=True)

# Remove orders with no delivery date (not completed)
orders = orders.dropna(subset=['order_delivered_customer_date'])
```

- Handle outliers:

```
# Example: Remove products with weight > 99th percentile
weight_threshold = products['product_weight_g'].quantile(0.99)
products = products[products['product_weight_g'] <= weight_threshold]
```

- Convert dates:

```
date_columns = ['order_purchase_timestamp', 'order_approved_at',
'order_delivered_carrier_date', 'order_delivered_customer_date']
for col in date_columns:
    orders[col] = pd.to_datetime(orders[col])
```

- Feature creation: order completion time:

```
      orders['delivery_time_days'] =
      (orders['order_delivered_customer_date'] -
      orders['order_approved_at']).dt.days
```

## 3. Feature Engineering & Dataset Merging

```
# Merge product and product category
products = products.merge(product_cat, on='product_category_name',
how='left')

# Merge order items with products
order_items_products = order_items.merge(products, on='product_id',
how='left')

# Merge with orders
df = order_items_products.merge(orders, on='order_id', how='left')

# Merge with payments (useful to see if payment installments affect
delivery time)
df = df.merge(payments.groupby('order_id').agg({'payment_installments':
'mean', 'payment_value': 'sum'}).reset_index(), on='order_id', how='left')

# Merge with reviews (useful to see quality)
df = df.merge(reviews.groupby('order_id').agg({'review_score':
'mean'}).reset_index(), on='order_id', how='left')

# Add customer state/city
df = df.merge(customers[['customer_id', 'customer_state',
'customer_city']], on='customer_id', how='left')
```

## 4. Feature Engineering & Dataset Merging

```
# Convert categorical variables
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df['product_category_name_english'] =
df['product_category_name_english'].fillna('unknown')
df['product_category_encoded'] =
le.fit_transform(df['product_category_name_english'])
df['customer_state_encoded'] = le.fit_transform(df['customer_state'])
df['customer_city_encoded'] = le.fit_transform(df['customer_city'])
```

## 5. Final Dataset Preparation

```
# Keep only necessary columns
final_data = df[['product_weight_g', 'product_length_cm',
'product_height_cm', 'product_width_cm','payment_installments',
'payment_value', 'review_score','product_category_encoded',
'customer_state_encoded', 'delivery_time_days']]
```

```
# Drop missing delivery times or negative values if any
final_data = final_data[final_data['delivery_time_days'] > 0]
```

## 6. Train-Test Split

```
from sklearn.model_selection import train_test_split

X = final_data.drop('delivery_time_days', axis=1)
y = final_data['delivery_time_days']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

## 7. Data Visualization

```
# Correlation heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(final_data.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()

# Delivery time distribution
plt.figure(figsize=(8, 5))
sns.histplot(final_data['delivery_time_days'], bins=30, kde=True)
plt.title('Distribution of Delivery Time (in days)')
plt.show()

# Boxplot for product weight and delivery time
plt.figure(figsize=(8, 5))
sns.boxplot(x='product_category_encoded', y='delivery_time_days',
data=final_data)
plt.title('Delivery Time by Product Category')
plt.show()
```