# Uber Transactions Analysis Script Summary

## Purpose of the Script

This code performs end-to-end analysis and modeling on Uber transaction data, involving:  
  
• Exploratory data analysis (EDA)  
• Visualizations  
• Feature engineering  
• Regression modeling with evaluation

## 📦 1. Library Imports

import warnings  
warnings.simplefilter(action='ignore', category=FutureWarning)  
  
Suppresses warnings related to deprecated features.  
  
Other libraries:  
- pandas, numpy: data manipulation and computations  
- matplotlib, seaborn: plotting  
- statsmodels, sklearn: statistical and machine learning modeling

## 📁 2. Data Loading & Preprocessing

df = pd.read\_csv("/lakehouse/default/Files/Uber\_csv/uber\_transactions\_extracted.csv")  
  
Loads the Uber transaction CSV file.  
  
  
df = df.sort\_values("Processed\_Date")  
df['Processed\_Date'] = pd.to\_datetime(df['Processed\_Date'])  
  
Sorts by date and converts date column to datetime format.

## 🔍 3. Data Exploration

df.head(), df.tail()  
df.drop(columns='Processed\_Date').describe().T  
  
Inspects sample rows and summary statistics of numerical columns.

## 📆 4. Monthly Aggregation and Line Plot

monthly\_summary = df.groupby([...]).agg({'Balance': 'sum'})  
  
Aggregates Balance by Year and Month.  
  
  
sns.lineplot(data=monthly\_summary, x='Month', y='Balance', hue='Year')  
  
Plots monthly total balance grouped by year.

## 📊 5. Extended Aggregation: Monthly, Quarterly, Yearly

Adds:  
- Average\_Balance  
- Transaction\_Count  
  
Then creates:  
- yearly\_summary: summed and averaged stats by year  
- quarterly\_summary: total balances by quarter, plotted with labels

## 📈 6. Precise Time-Series Plot

monthly\_summary['Date'] = pd.to\_datetime(...)  
sns.lineplot(data=monthly\_summary, x='Date', y='Total\_Balance')  
  
Converts month/year to a proper datetime index for accurate plotting.

## 📅 7. Weekday-Based Summary

day\_summary = df.groupby('Weekday').agg(...)  
  
Calculates mean values of key metrics per weekday.

## 📌 8. Distributions & Relationships

sns.histplot() shows earnings distribution  
  
sns.scatterplot() visualizes relationships:  
- Earnings vs Balance  
- Earnings vs Expenses  
  
sns.heatmap() shows numeric correlations

## 🛠️ 9. Feature Engineering

df2['Net\_Earnings'] = df2['Your earnings'] - df2['Refunds & Expenses']  
  
Introduces Net\_Earnings as the target variable.  
  
  
df3 = pd.get\_dummies(..., drop\_first=True)  
  
Applies one-hot encoding to categorical columns.

## 📐 10. Linear Regression with OLS

ols = sm.OLS(y, sm.add\_constant(x))  
  
Fits linear model using statsmodels to inspect p-values.

## 🧹 Backward Feature Elimination

Iteratively removes features with p-value > 0.05.

## 🤖 11. Train/Test Modeling

model = LinearRegression()  
model.fit(X\_train, y\_train)  
  
Trains linear model using Scikit-learn.  
  
  
model.score(...)  
  
Checks model accuracy for training and testing sets.

## 📏 12. Evaluation Metrics

Uses:  
- R² score  
- Mean Squared Error (MSE)  
- Root Mean Squared Error (RMSE)  
- Mean Absolute Error (MAE)  
  
Plots predicted vs actual values with reference line.

## 📌 13. Final Output

compare = pd.DataFrame({"Actual\_test\_value" : y\_test, "Predicted\_values": y\_pred})  
  
Creates comparison DataFrame for evaluation.