```
pwd
```

```
'C:\\Users\\sonuk'
```

```
import pandas as pd

india_df = pd.read_excel(r"C:\Users\sonuk\Downloads\IN_Data (1).xlsx")
japan_df = pd.read_excel(r"C:\Users\sonuk\Downloads\JPN Data
(2).xlsx")

japan_head = japan_df.head()
india_head = india_df.head()

japan_info = japan_df.dtypes
india_info = india_df.dtypes

japan_head, japan_info, india_head, india_info
```

```
(           ID  CURR_AGE GENDER     ANN_INCOME  AGE_CAR  PURCHASE
 0  00001Q15YJ        50      M  445344.000000      439         0
 1  00003I71CQ        35      M  107634.000000      283         0
 2  00003N47FS        59      F  502786.666667      390         1
 3  00005H41DE        43      M  585664.000000      475         0
 4  00007E17UM        39      F  705722.666667      497         1,
 ID              object
 CURR_AGE         int64
 GENDER          object
 ANN_INCOME     float64
 AGE_CAR          int64
 PURCHASE         int64
 dtype: object,
            ID  CURR_AGE GENDER  ANN_INCOME   DT_MAINT
 0  20710B05XL        54      M     1425390 2018-04-20
 1  89602T51HX        47      M     1678954 2018-06-08
 2  70190Z52IP        60      M      931624 2017-07-31
 3  25623V15MU        55      F     1106320 2017-07-31
 4  36230I68CE        32      F      748465 2019-01-27,
 ID                    object
 CURR_AGE               int64
 GENDER                object
 ANN_INCOME             int64
 DT_MAINT      datetime64[ns]
 dtype: object)
```

```
from datetime import datetime

# Step 1: Create age_car category for Japanese dataset
def categorize_age_car(days):
    if days < 200:
        return '<200'
    elif 200 <= days <= 360:
```

```
        return '200-360'
    elif 360 < days <= 500:
        return '360-500'
    else:
        return '>500'

japan_df['AGE_CAR_CATEGORY'] =
japan_df['AGE_CAR'].apply(categorize_age_car)

# Step 2: Calculate AGE_CAR for Indian dataset using 1st July 2019 as
reference
reference_date = pd.to_datetime('2019-07-01')
india_df['AGE_CAR'] = (reference_date - india_df['DT_MAINT']).dt.days

# Step 3: Apply same binning to Indian dataset
india_df['AGE_CAR_CATEGORY'] =
india_df['AGE_CAR'].apply(categorize_age_car)

# Display results
japan_df[['AGE_CAR', 'AGE_CAR_CATEGORY']].head(),
india_df[['DT_MAINT', 'AGE_CAR', 'AGE_CAR_CATEGORY']].head()
```

```
(   AGE_CAR AGE_CAR_CATEGORY
 0      439          360-500
 1      283          200-360
 2      390          360-500
 3      475          360-500
 4      497          360-500,
    DT_MAINT  AGE_CAR AGE_CAR_CATEGORY
 0 2018-04-20      437          360-500
 1 2018-06-08      388          360-500
 2 2017-07-31      700             >500
 3 2017-07-31      700             >500
 4 2019-01-27      155             <200)
```

```
# Re-import libraries and reload the files after user re-uploaded them
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime


# Set up visual style
sns.set(style="whitegrid")

# Create subplots for EDA -get 2 rows and 2 columns → total 4
subplots.
fig, axes = plt.subplots(2, 2, figsize=(14, 10)) #fig-complete , axes-
2D NumPy array of subplot axes (individual plots)
```

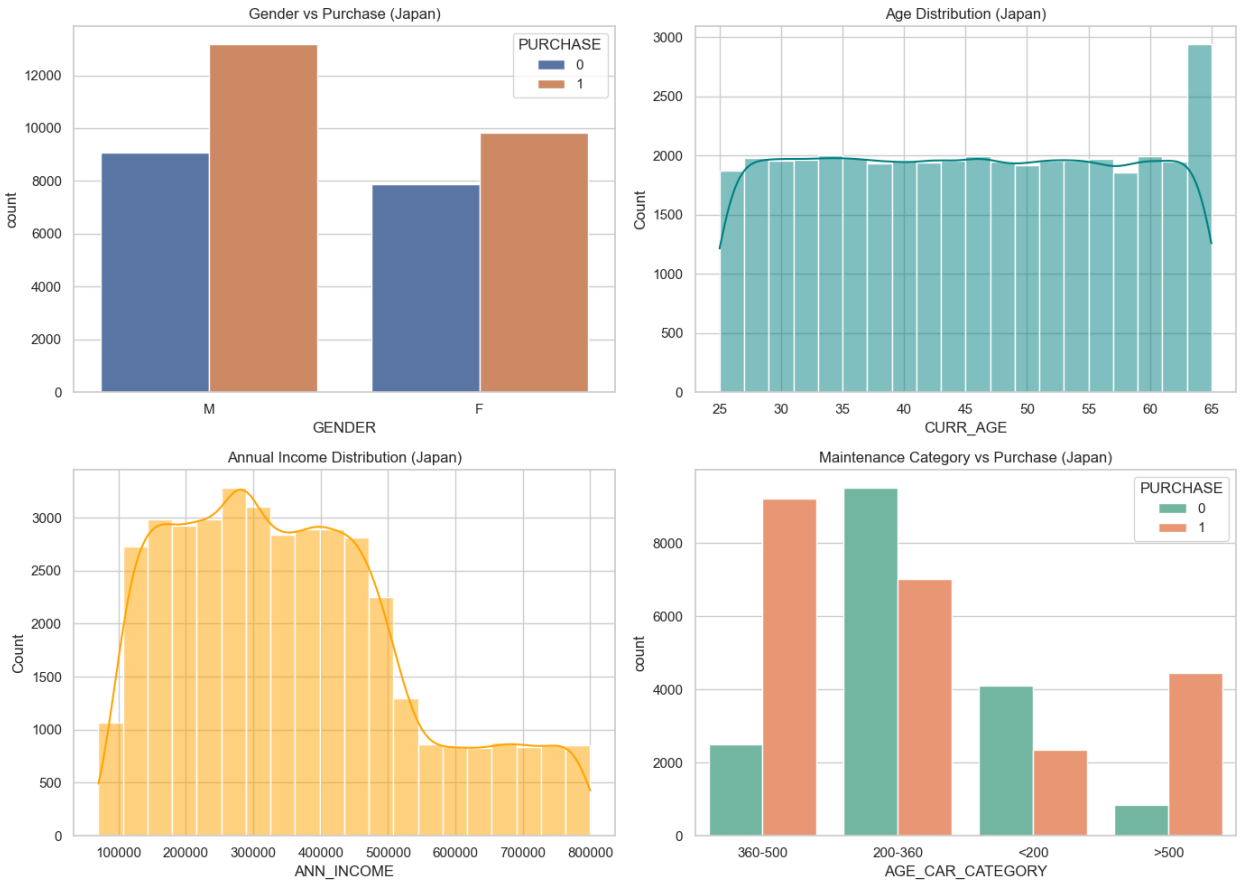
```
# 1. Gender Distribution vs Purchase- countplot always plots counts on
y-axis
sns.countplot(data=japan_df, x='GENDER', hue='PURCHASE', ax=axes[0,
0]) # y-axis shows the count (frequency) of records in each category
#hue- splits each bar into segments for purchase=0 and 1
axes[0, 0].set_title("Gender vs Purchase (Japan)")

# 2. Age Distribution - used hist so bins , kde-Kernel Density
Estimate creates smooth curve to know shape of distribution
sns.histplot(data=japan_df, x='CURR_AGE', kde=True, bins=20,
ax=axes[0, 1], color='teal')
axes[0, 1].set_title("Age Distribution (Japan)")

# 3. Annual Income Distribution
sns.histplot(data=japan_df, x='ANN_INCOME', kde=True, bins=20,
ax=axes[1, 0], color='orange')
axes[1, 0].set_title("Annual Income Distribution (Japan)")

# 4. AGE_CAR_CATEGORY vs Purchase
sns.countplot(data=japan_df, x='AGE_CAR_CATEGORY', hue='PURCHASE',
ax=axes[1, 1], palette='Set2') # set2 is color palettes soft pastel
like
axes[1, 1].set_title("Maintenance Category vs Purchase (Japan)")

plt.tight_layout()
plt.show()
```




```
# Exporting cleaned datasets for Tableau and further use
japan_df.to_excel(r"C:\Users\sonuk\Downloads\JPN Data (2).xlsx",
index=False)
india_df.to_excel(r"C:\Users\sonuk\Downloads\IN_Data (1).xlsx",
index=False)

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Select features and target
X = japan_df[['CURR_AGE', 'GENDER', 'ANN_INCOME', 'AGE_CAR_CATEGORY']]
y = japan_df['PURCHASE']

# Define which columns are categorical
categorical_cols = ['GENDER', 'AGE_CAR_CATEGORY']

# Preprocessing for categorical features
preprocessor = ColumnTransformer(
    transformers=[('cat', OneHotEncoder(drop='first'),
```

```
categorical_cols)],
    remainder='passthrough'  # numerical columns remain unchanged
)

# Create pipeline with preprocessing + logistic regression
model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(max_iter=1000))
])

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Train the model
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)

# Evaluation
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Confusion Matrix:
[[3421 1592]
 [2202 4785]]

Classification Report:
              precision    recall  f1-score   support

           0       0.61      0.68      0.64      5013
           1       0.75      0.68      0.72      6987

    accuracy                           0.68     12000
   macro avg       0.68      0.68      0.68     12000
weighted avg       0.69      0.68      0.69     12000
```

```
# Step 1: Select the same features from Indian data
X_india = india_df[['CURR_AGE', 'GENDER', 'ANN_INCOME',
'AGE_CAR_CATEGORY']]

# Step 2: Predict using the trained model
india_df['PREDICTED_PURCHASE'] = model.predict(X_india)

# Step 3: Count predicted buyers
predicted_buyers = india_df['PREDICTED_PURCHASE'].sum()
```

```
total_customers = len(india_df)

print(f"□ Predicted buyers: {predicted_buyers} out of
{total_customers} total customers")

# Step 4: Export to Excel for Tableau
india_df.to_excel(r"C:\Users\sonuk\Downloads\IN_Data (1).xlsx",
index=False)
```

```
□ Predicted buyers: 67014 out of 70000 total customers
```