```
# Upload Dataset
from google.colab import files
uploaded = files.upload()

# Load Dataset
import pandas as pd
df = pd.read_csv('/content/raw_sales.csv')
df.head()

# Dataset Info
df.info()
df.describe()

# Check for Missing Values and Duplicates
print("Missing Values:\n", df.isnull().sum())
print("Duplicates:", df.duplicated().sum())
df = df.drop_duplicates()

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns

df.hist(bins=30, figsize=(12, 10))
plt.tight_layout()
plt.show()

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

# Target and Features
target = 'price'
features = ['datesold', 'postcode', 'propertyType', 'bedrooms']
X = df[features]
y = df[target]

# Convert 'datesold'
X['datesold'] = pd.to_datetime(X['datesold'])
X['year'] = X['datesold'].dt.year
X['month'] = X['datesold'].dt.month
X = X.drop('datesold', axis=1)

# One-hot encode
X = pd.get_dummies(X, columns=['propertyType'], drop_first=True)

# Scale features
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-test split
from sklearn.model_selection import train_test_split
```

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

# Train model
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Evaluation
from sklearn.metrics import mean_squared_error, r2_score
y_pred = model.predict(X_test)
print("R² Score:", r2_score(y_test, y_pred))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))

# Predict new house price
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

df = pd.read_csv("raw_sales.csv")
df['datesold'] = pd.to_datetime(df['datesold'])
df['year'] = df['datesold'].dt.year
df['month'] = df['datesold'].dt.month
df = df.drop(columns=['datesold'])
X = df.drop(columns=['price'])
y = df['price']
categorical_cols = ['propertyType']
numerical_cols = ['bedrooms', 'postcode', 'year', 'month']

preprocessor = ColumnTransformer([
    ('num', StandardScaler(), numerical_cols),
    ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_cols)
])

pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('model', RandomForestRegressor(n_estimators=100, random_state=42))
])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
pipeline.fit(X_train, y_train)

# Gradio UI
import gradio as gr

def predict_house_price(bedrooms, postcode, propertyType, year, month):
    new_data = pd.DataFrame([{
        'bedrooms': bedrooms,
        'postcode': postcode,
        'propertyType': propertyType,
        'year': year,
        'month': month
    }])
    try:
```

```
        predicted_price = pipeline.predict(new_data)
        return "${:,.2f}".format(predicted_price[0])
    except Exception as e:
        return f"Error predicting price: {e}"

input_fields = [
    gr.Number(label="Bedrooms", value=3, precision=0),
    gr.Number(label="Postcode", value=3000, precision=0),
    gr.Dropdown(label="Property Type", choices=['House', 'Unit'], value='House'),
    gr.Number(label="Year", value=2024, precision=0),
    gr.Number(label="Month", value=1, precision=0),
]

gr.Interface(
    fn=predict_house_price,
    inputs=input_fields,
    outputs="text",
    title="House Price Predictor",
    description="Enter the details of a house to get a price prediction."
).launch(debug=True)
```