```
from google.colab import files

import pandas as pd

from scipy import stats

from sklearn.linear_model import LinearRegression

from sklearn.metrics import r2_score


# Upload file

uploaded = files.upload()


# Read and clean file

df = None

for filename in uploaded:

    try:

        if filename.endswith('.csv'):

            df = pd.read_csv(filename)

        elif filename.endswith(('.xls', '.xlsx')):

            df = pd.read_excel(filename, engine='openpyxl')

        else:

            raise ValueError("Unsupported file format.")

        print(f"✅ File loaded: {filename}")


        df.columns = df.columns.str.strip().str.lower().str.replace(" ", "_")

        print("\n🧾 Cleaned Column Names:", df.columns.tolist())

        display(df.head())

        break

    except Exception as e:

        print("❌ Failed to read file:", e)
```

```
if df is not None:
    print("\n📢 Available Statistical Tests:")
    print("1️⃣T-Test (comparing two groups' means)")
    print("2️⃣F-Test (variance comparison)")
    print("3️⃣Chi-Square Test (categorical association)")
    print("4️⃣Multiple Linear Regression")

    choice = input("\n👉 Enter the number of the test you want to run (1/2/3/4): ").strip()

    dep = input("🎯 Enter dependent variable: ").strip().lower().replace(" ", "_")
    group = input("🧑‍🤝‍🧑 Enter independent/grouping variable(s) (comma-separated for multiple): ").strip().lower()
    group_vars = [g.strip().replace(" ", "_") for g in group.split(",")]

    # Check validity
    if dep not in df.columns:
        print(f"❌ '{dep}' is not a valid column name.")
    elif not set(group_vars).issubset(set(df.columns)):
        print(f"❌ Some independent/grouping variables are not valid: {group_vars}")
    else:
        dep_is_numeric = pd.api.types.is_numeric_dtype(df[dep])
        group_types = {col: pd.api.types.is_numeric_dtype(df[col]) for col in group_vars}

        print("\n🔍 Variable Types:")
        print(f" - Dependent '{dep}': {'Numeric' if dep_is_numeric else 'Categorical'}")
        for col, is_num in group_types.items():
            print(f" - '{col}': {'Numeric' if is_num else 'Categorical'}")
        # Decision logic
        if choice == "1":  # T-Test
            if not dep_is_numeric:
                print("❌ T-Test requires the dependent variable to be numeric.")
            elif len(group_vars) != 1 or not pd.api.types.is_categorical_dtype(df[group_vars[0]])
and not df[group_vars[0]].dtype == 'object':
                print("❌ T-Test requires one categorical grouping variable with exactly 2 unique
groups.")
            else:
                g = group_vars[0]
                unique_vals = df[g].dropna().unique()
                print(f"💡 Grouping variable '{g}' has categories: {unique_vals.tolist()}")
                if len(unique_vals) != 2:
                    print("❌ T-Test needs exactly 2 groups.")
                else:
                    samples = [df[df[g] == val][dep].dropna() for val in unique_vals]
                    t_stat, p_val = stats.ttest_ind(samples[0], samples[1], equal_var=False)
                    print(f"✅ T-Test Result:\nT-statistic = {t_stat:.4f}, P-value = {p_val:.4f}")

        elif choice == "2":  # F-Test
            if not dep_is_numeric:
                print("❌ F-Test requires the dependent variable to be numeric.")
            elif len(group_vars) != 1 or not pd.api.types.is_categorical_dtype(df[group_vars[0]])
and not df[group_vars[0]].dtype == 'object':
                print("❌ F-Test requires one categorical grouping variable.")
            else:
                g = group_vars[0]
                unique_vals = df[g].dropna().unique()
                print(f"💡 Grouping variable '{g}' has categories: {unique_vals.tolist()}")
                if len(unique_vals) != 2:
                    print("❌ F-Test is only valid for two groups.")
                else:
                    samples = [df[df[g] == val][dep].dropna() for val in unique_vals]
                    f_stat = stats.levene(samples[0], samples[1]).statistic
                    p_val = stats.levene(samples[0], samples[1]).pvalue
                    print(f"✅ F-Test Result:\nF-statistic = {f_stat:.4f}, P-value = {p_val:.4f}")

        elif choice == "3":  # Chi-Square Test
            if dep_is_numeric or any(group_types[col] for col in group_vars):
                print("❌ Chi-Square Test requires all variables to be categorical.")
            else:
                contingency = pd.crosstab(df[dep], df[group_vars[0]])
                chi2, p, dof, expected = stats.chi2_contingency(contingency)
                print(f"✅ Chi-Square Test Result:\nChi2 = {chi2:.4f}, p-value = {p:.4f}, Degrees of
freedom = {dof}")

        elif choice == "4":  # Multiple Regression
            if not dep_is_numeric:
                print("❌ Regression requires the dependent variable to be numeric.")
            else:
                df_model = df[[dep] + group_vars].dropna()
                df_encoded = pd.get_dummies(df_model, columns=group_vars, drop_first=True)
                X = df_encoded.drop(columns=[dep])
                y = df_encoded[dep]
                model = LinearRegression().fit(X, y)
                predictions = model.predict(X)

                print(f"\n✅ Multiple Regression Results")
                print(f"📈 R² Score: {r2_score(y, predictions):.4f}")
                for col, coef in zip(X.columns, model.coef_):
                    print(f"   - {col}: {coef:.4f}")
                print(f"📍 Intercept: {model.intercept_:.4f}")

        else:
            print("❌ Invalid selection. Please choose 1, 2, 3, or 4.")
```