```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from lifelines import KaplanMeierFitter

# Load data
df = pd.read_csv('COVID clinical trials.csv')

# 1. Data Cleaning & Feature Engineering
df['Enrollment'] = pd.to_numeric(df['Enrollment'], errors='coerce')
df['Start Date'] = pd.to_datetime(df['Start Date'], errors='coerce')
df['Completion Date'] = pd.to_datetime(df['Completion Date'], errors='coerce')
df['Duration_Days'] = (df['Completion Date'] - df['Start Date']).dt.days

# Handle negative or zero durations (data quality check)
df['Duration_Days'] = df['Duration_Days'].apply(lambda x: x if x > 0 else np.nan)

# Binary variable for Completion
df['Is_Completed'] = (df['Status'] == 'Completed').astype(int)

# Simple Funder mapping
def simplify_funder(funder_str):
    if pd.isna(funder_str): return 'Other'
    if 'Industry' in funder_str: return 'Industry'
    if 'NIH' in funder_str: return 'NIH'
    if 'U.S. Fed' in funder_str: return 'U.S. Fed'
    return 'Other'
df['Funder_Category'] = df['Funded Bys'].apply(simplify_funder)

# 2. Basic Stats: Descriptive & Missing Values
missing_data = df.isnull().mean() * 100
summary_stats = df[['Enrollment', 'Duration_Days']].describe()

# 3. Intermediate: Correlation
# Note: Phases is ordinal, we can map it
phase_map = {'Early Phase 1': 0, 'Phase 1': 1, 'Phase 1|Phase 2': 1.5, 'Phase 2': 2,
'Phase 2|Phase 3': 2.5, 'Phase 3': 3, 'Phase 4': 4}
df['Phase_Ordinal'] = df['Phases'].map(phase_map)
corr_matrix = df[['Enrollment', 'Duration_Days', 'Phase_Ordinal',
'Is_Completed']].corr()

# 4. Advanced: Hypothesis Testing
# Chi-Square: Funder vs Completion
contingency_table = pd.crosstab(df['Funder_Category'], df['Is_Completed'])
chi2, p_chi2, dof, expected = stats.chi2_contingency(contingency_table)

# Mann-Whitney U: Enrollment (Completed vs Terminated)
# Why? Enrollment is likely not normal.
group_comp = df[df['Status'].isin(['Completed', 'Terminated'])]
u_stat, p_mann = stats.mannwhitneyu(
    group_comp[group_comp['Status'] == 'Completed']['Enrollment'].dropna(),
    group_comp[group_comp['Status'] == 'Terminated']['Enrollment'].dropna()
)

# 5. Advanced: Survival Analysis (Time to Completion)
# Event = Completed (1), Censored = everything else (0)
df_km = df.dropna(subset=['Duration_Days'])
kmf = KaplanMeierFitter()
# We cap duration at a reasonable level (e.g., 2 years) to avoid the extreme
errors found earlier
df_km_filtered = df_km[df_km['Duration_Days'] < 1500]

# Visualizations
plt.figure(figsize=(10, 6))
missing_data.sort_values(ascending=False).plot(kind='bar', color='salmon')
plt.title('Percentage of Missing Values by Column')
plt.ylabel('Percentage (%)')
plt.tight_layout()
plt.savefig('missing_values.png')

plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.tight_layout()
plt.savefig('correlation_heatmap.png')

plt.figure(figsize=(10, 6))
kmf.fit(df_km_filtered['Duration_Days'],
event_observed=df_km_filtered['Is_Completed'], label='Completion Probability')
kmf.plot_survival_function()
plt.title('Survival Analysis: Time to Trial Completion (Kaplan-Meier)')
plt.xlabel('Days')
plt.ylabel('Probability of Not Yet Completed')
plt.grid(True)
plt.savefig('survival_analysis.png')

# Print results for interpretation
print("Summary Stats:\n", summary_stats)
print("\nMissing Values (%):\n", missing_data)
print("\nChi-Square Test (Funder vs Completion): p-value =", p_chi2)
print("Mann-Whitney U Test (Enrollment Completed vs Terminated): p-value =",
p_mann)
```