Here are **separate Python code blocks** for each of the tasks we listed. These can be used in a Jupyter Notebook after loading your cleaned dataset (jamb\_survey\_cleaned.csv).

1. **Load the dataset into Python for analysis**

import pandas as pd

# Load original raw file (update path if needed)

raw\_df = pd.read\_csv("Nigerian JAMB UTME Exam Taker Survey (2020-2025) (Responses) - Form Responses 1.csv")

# Save raw data as CSV

raw\_df.to\_csv("jamb\_survey\_raw.csv", index=False)

1. **Clean and Transform the Dataset**

# Load the raw data again

df = pd.read\_csv("jamb\_survey\_raw.csv")

# Rename columns for clarity

df.columns = [

"Timestamp", "Eligibility", "Gender", "Age", "Exam\_Year", "State",

"School\_Type", "Highest\_Score", "Prep\_Method", "Prep\_Challenges",

"Confidence\_Level", "Difficult\_Subjects", "CBT\_Center\_Difficulty",

"CBT\_Familiarity", "Exam\_Day\_Challenges", "Center\_Facilities\_Rating",

"Technical\_Issues", "Prep\_Improvement", "Advice", "Additional\_Comments"

]

# Keep only respondents who took JAMB

df = df[df["Eligibility"].str.strip().str.lower() == "yes"]

# Drop rows missing essential fields

essential\_fields = ["Gender", "Age", "Exam\_Year", "State", "School\_Type", "Highest\_Score"]

df = df.dropna(subset=essential\_fields)

# Convert types

df["Exam\_Year"] = df["Exam\_Year"].astype(int)

df["Highest\_Score"] = pd.to\_numeric(df["Highest\_Score"], errors='coerce')

df = df.dropna(subset=["Highest\_Score"])

# Create Success column

df["Success"] = df["Highest\_Score"] >= 200

# Reset index

df.reset\_index(drop=True, inplace=True)

1. **Save the Cleaned Dataset**

# Save cleaned dataset

df.to\_csv("jamb\_survey\_cleaned.csv", index=False)

1. **Import The Cleaned Data Set**

import pandas as pd

import matplotlib.pyplot as plt

# Load cleaned dataset

df = pd.read\_csv('jamb\_survey\_cleaned.csv')

# Ensure 'Success' is boolean

df['Success'] = df['Success'].astype(bool)

# Calculate success rate per year

success\_by\_year = df.groupby('Exam\_Year')['Success'].mean() \* 100

# Plot

success\_by\_year.plot(kind='bar', color='skyblue', ylabel='Success Rate (%)', title='Success Rate by Year')

plt.xticks(rotation=0)

plt.tight\_layout()

plt.show()

**5. Identify Trends and Potential Causes of Poor Performance**

This explores the **relationship between preparation methods, challenges, and failure**.

import seaborn as sns

# Countplot: Preparation method vs. success

sns.countplot(data=df, x='Prep\_Method', hue='Success')

plt.title('Preparation Method vs Success')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

# Countplot: School type vs success

sns.countplot(data=df, x='School\_Type', hue='Success')

plt.title('School Type vs Success')

plt.tight\_layout()

plt.show()

**3. Correlations Between Background Factors and Exam Outcomes**

This simplifies correlation with encoded features.

# Encode categorical features

df\_encoded = pd.get\_dummies(df[['Gender', 'School\_Type', 'CBT\_Familiarity', 'Confidence\_Level']], drop\_first=True)

df\_encoded['Success'] = df['Success'].astype(int)

# Compute correlations

correlations = df\_encoded.corr()['Success'].sort\_values(ascending=False)

print("Correlation with Success:\n", correlations)

**7. Predictive Modeling: Classify Success/Failure**

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report

# Select and encode features

features = ['Gender', 'Age', 'Exam\_Year', 'School\_Type', 'Confidence\_Level']

df\_model = pd.get\_dummies(df[features + ['Success']], drop\_first=True)

X = df\_model.drop('Success', axis=1)

y = df\_model['Success']

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Random Forest model

model = RandomForestClassifier(random\_state=42)

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

# Evaluation

print(classification\_report(y\_test, y\_pred))

**8. Forecast Success/Failure Rates for 2026–2030**

A basic time-series-like projection using trend extrapolation.

import numpy as np

from sklearn.linear\_model import LinearRegression

# Average success rate by year

yearly = df.groupby('Exam\_Year')['Success'].mean().reset\_index()

X = yearly[['Exam\_Year']]

y = yearly['Success']

# Linear Regression

lr = LinearRegression()

lr.fit(X, y)

# Predict 2026–2030

future\_years = pd.DataFrame({'Exam\_Year': list(range(2026, 2031))})

future\_preds = lr.predict(future\_years)

# Plot

plt.plot(X, y, label='Historical', marker='o')

plt.plot(future\_years, future\_preds, label='Forecast', marker='x', linestyle='--')

plt.title('Forecasted JAMB Success Rate (2026–2030)')

plt.xlabel('Year')

plt.ylabel('Success Rate')

plt.legend()

plt.tight\_layout()

plt.show()

# Display predictions

forecast\_df = future\_years.copy()

forecast\_df['Predicted\_Success\_Rate'] = future\_preds \* 100

print(forecast\_df)