Task 1: Data Cleaning and Preprocessing – Advanced Approach

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/conte
```

```
import pandas as pd

file_path = '/content/drive/MyDrive/KaggleV2-May-2016.csv'
df = pd.read_csv(file_path)

print("✅ Data Preview:")
print(df.head(), "\n")

print("📋 Dataset Info:")
print(df.info(), "\n")

print("📊 Descriptive Statistics:")
print(df.describe(include='all'))
```

```
 12  SMS_received    110527 non-null  int64
```

```
25%          0.000000        0.000000        0.000000        0.000000
```

```
25%          0.000000       0.000000       0.000000       0.000000
50%          0.000000       0.000000       0.000000       0.000000
75%          0.000000       0.000000       0.000000       0.000000
max          1.000000       1.000000       1.000000       4.000000

         SMS_received No-show
count   110527.000000  110527
unique            NaN       2
top               NaN      No
freq              NaN   88208
mean         0.321026     NaN
std          0.466873     NaN
min          0.000000     NaN
25%          0.000000     NaN
50%          0.000000     NaN
75%          1.000000     NaN
max          1.000000     NaN
```

Explanation:

.head() → first 5 rows

.info() → data types, non-null counts

.describe() → stats summary

Step 2: Identify Missing Values

```
missing_values = df.isnull().sum()
print(missing_values)

missing_percent = (df.isnull().sum() / len(df)) * 100
print(missing_percent)
```

```
PatientId         0
AppointmentID     0
Gender            0
ScheduledDay      0
AppointmentDay    0
Age               0
Neighbourhood     0
Scholarship       0
Hipertension      0
Diabetes          0
Alcoholism        0
Handcap           0
SMS_received      0
No-show           0
dtype: int64
PatientId         0.0
AppointmentID     0.0
Gender            0.0
ScheduledDay      0.0
AppointmentDay    0.0
Age               0.0
Neighbourhood     0.0
Scholarship       0.0
Hipertension      0.0
Diabetes          0.0
Alcoholism        0.0
Handcap           0.0
SMS_received      0.0
No-show           0.0
dtype: float64
```

Handling Missing Values:

1. Remove rows/columns with excessive missing values

```
df = df.loc[:, df.isnull().mean() < 0.5]

df.dropna(inplace=True)
```

2. Fill missing values intelligently

```
df['Age'] = df['Age'].fillna(df['Age'].median())
df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0] if not df['Gender'].mode().empty else

print(df.info())
print(df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 14 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   PatientId       110527 non-null  float64
 1   AppointmentID   110527 non-null  int64
 2   Gender          110527 non-null  object
 3   ScheduledDay    110527 non-null  object
 4   AppointmentDay  110527 non-null  object
 5   Age             110527 non-null  int64
 6   Neighbourhood   110527 non-null  object
 7   Scholarship     110527 non-null  int64
 8   Hipertension    110527 non-null  int64
 9   Diabetes        110527 non-null  int64
 10  Alcoholism      110527 non-null  int64
 11  Handcap         110527 non-null  int64
 12  SMS_received    110527 non-null  int64
 13  No-show         110527 non-null  object
dtypes: float64(1), int64(8), object(5)
memory usage: 11.8+ MB
None
      PatientId  AppointmentID Gender          ScheduledDay  \
0  2.987250e+13        5642903      F  2016-04-29T18:38:08Z
1  5.589978e+14        5642503      M  2016-04-29T16:08:27Z
2  4.262962e+12        5642549      F  2016-04-29T16:19:04Z
3  8.679512e+11        5642828      F  2016-04-29T17:29:31Z
4  8.841186e+12        5642494      F  2016-04-29T16:07:23Z

         AppointmentDay  Age      Neighbourhood  Scholarship  Hipertension  \
0  2016-04-29T00:00:00Z   62    JARDIM DA PENHA            0             1
1  2016-04-29T00:00:00Z   56    JARDIM DA PENHA            0             0
2  2016-04-29T00:00:00Z   62      MATA DA PRAIA            0             0
3  2016-04-29T00:00:00Z    8  PONTAL DE CAMBURI            0             0
4  2016-04-29T00:00:00Z   56    JARDIM DA PENHA            0             1

   Diabetes  Alcoholism  Handcap  SMS_received No-show
0         0           0        0             0      No
1         0           0        0             0      No
2         0           0        0             0      No
3         0           0        0             0      No
4         1           0        0             0      No
```

Notes for interviews:

.dropna() ⟶ removes missing data

.fillna() ⟶ imputes missing data

Median is better for skewed data; mean can be used if distribution is normal.

Step 3: Handle Duplicates

Tip: Always check subset of columns if duplicates are defined only by certain fields:

```
duplicates = df.duplicated().sum()
print(f"Duplicate rows: {duplicates}")

df.drop_duplicates(inplace=True)
```

```
Duplicate rows: 0
```

Tip: Always check subset of columns if duplicates are defined only by certain fields:

```
df.drop_duplicates(subset=['PatientId'], inplace=True)

print(f"Number of rows after dropping duplicates based on PatientId: {len(df)}")
```

```
Number of rows after dropping duplicates based on PatientId: 62299
```

Now have 62,299 unique Patient IDs in our dataset, which likely means duplicates (patients appearing more than once) were removed.

Step 4: Standardize Text Data

Common issues: inconsistent gender, country names, casing, extra spaces

```
df['Gender'] = df['Gender'].str.strip().str.lower()

df['Gender'] = df['Gender'].replace({'m': 'male', 'f': 'female', 'female ': 'female'})
```

Step 5: Handle Inconsistent Date Formats

```
df['ScheduledDay'] = pd.to_datetime(df['ScheduledDay'], errors='coerce')
df['AppointmentDay'] = pd.to_datetime(df['AppointmentDay'], errors='coerce')

invalid_scheduled = df['ScheduledDay'].isnull().sum()
invalid_appointment = df['AppointmentDay'].isnull().sum()

print(f"Invalid ScheduledDay: {invalid_scheduled}")
print(f"Invalid AppointmentDay: {invalid_appointment}")

df.dropna(subset=['ScheduledDay', 'AppointmentDay'], inplace=True)

df['scheduled_date'] = df['ScheduledDay'].dt.date
df['appointment_date'] = df['AppointmentDay'].dt.date
df['scheduled_time'] = df['ScheduledDay'].dt.time

df['waiting_days'] = (df['AppointmentDay'] - df['ScheduledDay']).dt.days
```

```
df['waiting_days'] = (df['AppointmentDay'] - df['ScheduledDay']).dt.days

print(df[['ScheduledDay', 'AppointmentDay', 'waiting_days']].head())
```

```
Invalid ScheduledDay: 0
Invalid AppointmentDay: 0
               ScheduledDay            AppointmentDay  waiting_days
0 2016-04-29 18:38:08+00:00 2016-04-29 00:00:00+00:00            -1
1 2016-04-29 16:08:27+00:00 2016-04-29 00:00:00+00:00            -1
2 2016-04-29 16:19:04+00:00 2016-04-29 00:00:00+00:00            -1
3 2016-04-29 17:29:31+00:00 2016-04-29 00:00:00+00:00            -1
4 2016-04-29 16:07:23+00:00 2016-04-29 00:00:00+00:00            -1
```

pd.to_datetime(..., errors='coerce') ⟶ safely converts string to datetime and replaces invalid values with NaT.

We remove rows where conversion failed.

We create extra features (scheduled_date, appointment_date, waiting_days) useful later for analysis.

Step 6: Rename Columns for Uniformity

```
df.columns = df.columns.str.lower().str.replace('-', '_').str.replace(' ', '_')

print("✅ Columns after renaming:")
print(df.columns)
```

```
✅ Columns after renaming:
Index(['patientid', 'appointmentid', 'gender', 'scheduledday',
       'appointmentday', 'age', 'neighbourhood', 'scholarship', 'hipertension',
       'diabetes', 'alcoholism', 'handcap', 'sms_received', 'no_show',
       'scheduled_date', 'appointment_date', 'scheduled_time', 'waiting_days'],
      dtype='object')
```

Reason: Clean column names help when writing models and avoid syntax issues.

Step 7: Check and Fix Data Types

```
df['age'] = df['age'].astype(int)

df['gender'] = df['gender'].astype('category')

df['no_show'] = df['no_show'].astype('category')

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 62299 entries, 0 to 110524
Data columns (total 18 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   patientid         62299 non-null  float64
 1   appointmentid     62299 non-null  int64
 2   gender            62299 non-null  category
 3   scheduledday      62299 non-null  datetime64[ns, UTC]
 4   appointmentday    62299 non-null  datetime64[ns, UTC]
 5   age               62299 non-null  int64
 6   neighbourhood     62299 non-null  object
 7   scholarship       62299 non-null  int64
 8   hipertension      62299 non-null  int64
 9   diabetes          62299 non-null  int64
 10  alcoholism        62299 non-null  int64
 11  handcap           62299 non-null  int64
```

```
 12  sms_received      62299 non-null  int64
 13  no_show           62299 non-null  category
 14  scheduled_date    62299 non-null  object
 15  appointment_date  62299 non-null  object
 16  scheduled_time    62299 non-null  object
 17  waiting_days      62299 non-null  int64
dtypes: category(2), datetime64[ns, UTC](2), float64(1), int64(9), object(4)
memory usage: 8.2+ MB
```

Step 8: Check and Fix Data Types

After your previous cleaning steps (removing invalid ages, fixing waiting days, etc.)

```
Start coding or generate with AI.
```