Appendix 1 - Preliminary analysis

```
!pip install pandas numpy seaborn matplotlib
```

Load and Clean the Data

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from google.colab import files

# Load the dataset
df = pd.read_csv("cstads_2122_pumf.csv")

# Dataset overview
print("Original DataFrame:\n")
print(df.shape)
df.describe()
df.info()

# Check the first few rows
print("\n", df.head())

# Remove duplicates if any
df = df.drop_duplicates()

# Standardize categorical values (example: converting 'Male'/'M' variations)
df['DVGENDER'] = df['DVGENDER'].replace({1: 'Female', 2: 'Male'})
df['DVURBAN'] = df['DVURBAN'].replace({1: 'Urban', 2: 'Rural'})
df['PROVID'] = df['PROVID'].replace({10: 'NL', 11: 'PEI', 12: 'NS', 24: 'QC', 35: 'ON', 46: 'MN', 47: 'SK', 48: 'AB', 59:'BC'})

# Identify missing values
df.replace({96: pd.NA, 98: pd.NA, 99: pd.NA, 996: pd.NA, 999: pd.NA}, inplace=True)
print("\nMissing Values:\n", df.isnull().sum())

# Calculate the percentage of missing values per column
missing_percentage = df.isnull().mean() * 100

# Drop columns with more than 75% missing values
df_cleaned = df.loc[:, missing_percentage <= 75]

# Cleaned dataset overview
print("\nCleaned DataFrame (columns with more than 75% missing values removed):\n")
print(df_cleaned.shape)
df_cleaned.describe()
df_cleaned.info()

# Check the first few rows
print("\n", df_cleaned.head())

# Identify missing values
print("\nMissing Values:\n", df_cleaned.isnull().sum())

# Move cleaned dataframe to df for convenience
df = df_cleaned
```

```
Original DataFrame:

(61096, 168)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61096 entries, 0 to 61095
Columns: 168 entries, SEQID to DVAVCIGD
dtypes: float64(1), int64(167)
memory usage: 78.3 MB

    SEQID  PROVID  GRADE  DVGENDER  DVURBAN  DVRES  DVORIENT  DVDESCRIBE  \
0  18338      11      9         2        1      3         2           1
1  16111      24     10         1        1      1         2           4
2  20587      10      7         2        1      1         2           1
3  54568      12      8        99        2      1         1           1
4  40991      10      7         1        2      1         2           1
```

```
      WTPUMF  GH_010  ...  BUL_100  BUL_110  BUL_120  DVTY1ST  DVTY2ST  \
0   1.164716       2  ...        2        2        4        3        7
1  36.141169       3  ...        2        2        1        3        7
2   1.448578       2  ...        2        2        1        3        7
3   3.036660       4  ...        2        2        2        3        7
4   3.745233       2  ...       99       99       99        3        7

   DVLAST30  DVAMTSMK  DVCIGWK  DVNDSMK  DVAVCIGD
0         2        96      996       96        96
1         2        96      996       96        96
2         2        96      996       96        96
3         2        96      996       96        96
4         2        96      996       96        96

[5 rows x 168 columns]

Missing Values:
 SEQID          5
PROVID         0
GRADE          0
DVGENDER    5025
DVURBAN        0
            ...
DVLAST30     375
DVAMTSMK   57291
DVCIGWK    56879
DVNDSMK    56876
DVAVCIGD   56876
Length: 168, dtype: int64

Cleaned DataFrame (columns with more than 75% missing values removed):

(61096, 138)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61096 entries, 0 to 61095
Columns: 138 entries, SEQID to DVLAST30
dtypes: float64(1), int64(1), object(136)
memory usage: 64.3+ MB

    SEQID PROVID  GRADE DVGENDER DVURBAN DVRES DVORIENT DVDESCRIBE     WTPUMF  \
0  18338    PEI      9     Male   Urban     3        2          1   1.164716
```

Generate Summary Statistics

```
# Summary statistics for numerical variables
print("Summary Statistics:\n", df.describe())

# Count unique values for categorical variables
print("\nCategorical Data Distribution:\n", df[['DVGENDER']].apply(pd.Series.value_counts))
print("\n", df[['PROVID']].apply(pd.Series.value_counts))
print("\n", df[['GRADE']].apply(pd.Series.value_counts))
print("\n", df[['DVURBAN']].apply(pd.Series.value_counts))
```

```
Summary Statistics:
               GRADE        WTPUMF
count  61096.000000  61096.000000
mean       9.141548     35.353820
std        1.622969     56.903468
min        7.000000      0.507389
25%        8.000000      6.075865
50%        9.000000     25.260619
75%       10.000000     38.635371
max       12.000000   1561.260211

Categorical Data Distribution:
           DVGENDER
DVGENDER
Male         28903
Female       27168

         PROVID
PROVID
QC        10863
AB         9260
NL         7032
NS         6999
BC         6885
ON         6745
SK         5596
PEI        4616
MN         3100
```

```
         GRADE
GRADE
8      12500
7      12436
9      11055
10     10218
11      8859
12      6028

            DVURBAN
DVURBAN
Urban      49577
Rural      11519
```

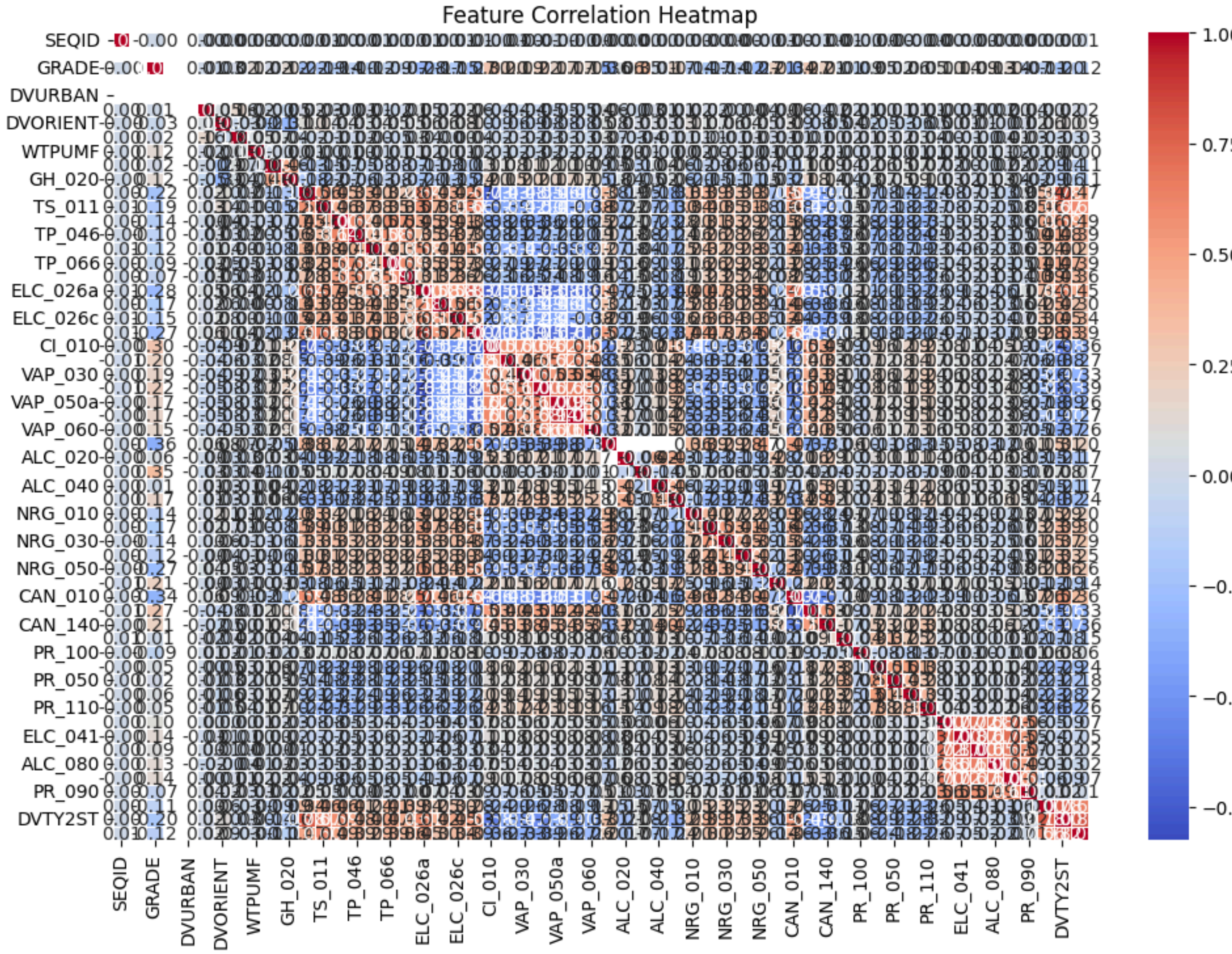
Perform Correlation Analysis

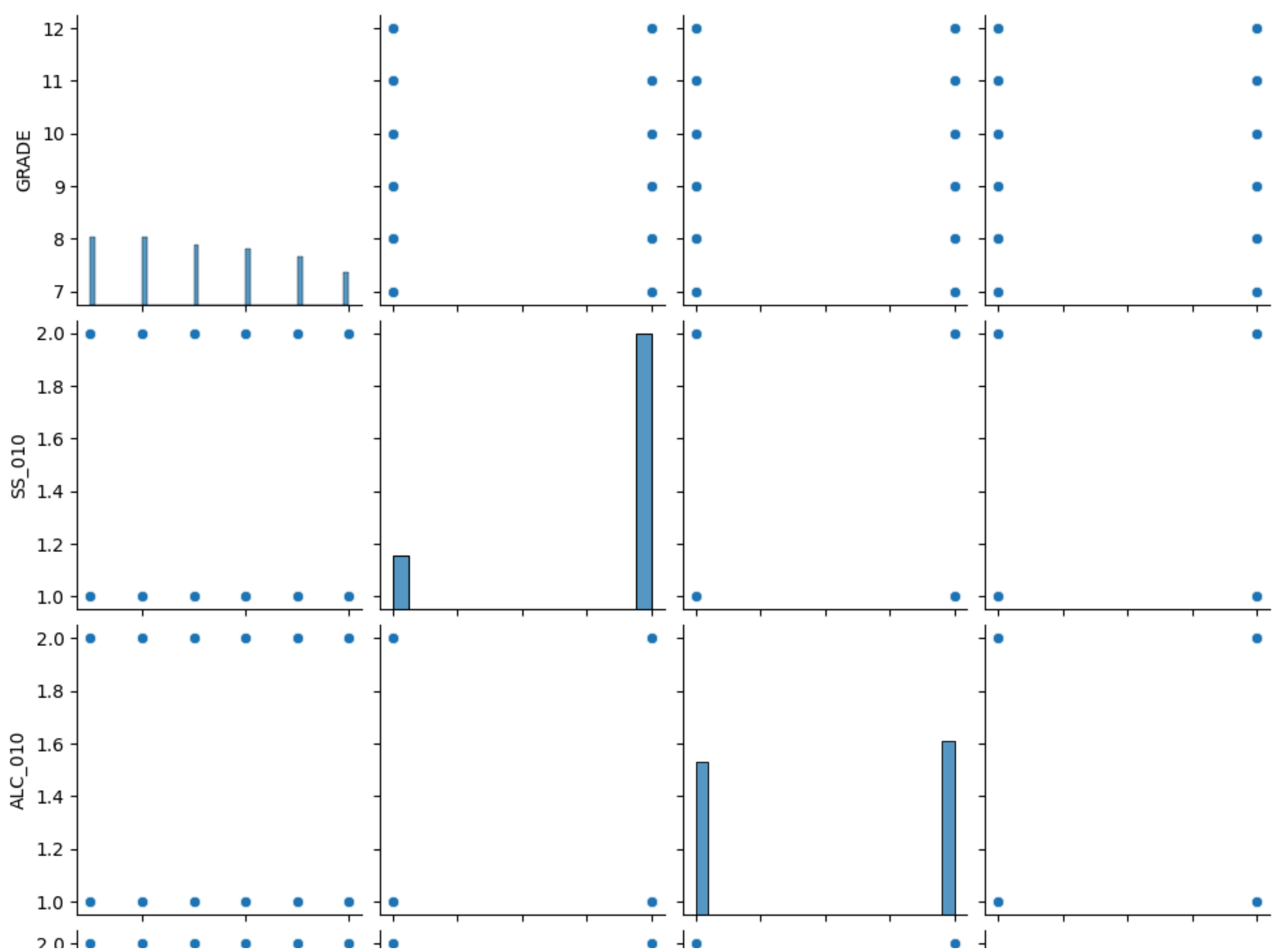
```
import re

# Define the prefixes to exclude
exclude_prefixes = ["UND", "MET", "XTC", "HAL", "HER", "COC", "SYN", "BZP", "TNB", "TRP", "GLU", "SAL",
                    "SLP", "STI", "DEX", "GRV", "SED", "POLY", "PH", "DR", "BEH", "BUL"]

# Create a regex pattern to match column names starting with these prefixes
pattern = re.compile(r'^(?:' + '|'.join(exclude_prefixes) + r').*')

# Identify columns to drop
columns_to_drop = [col for col in df.columns if pattern.match(col)]

# Drop the unwanted columns
df_filtered = df.drop(columns=columns_to_drop)

# Print the removed columns
print("Removed columns:", columns_to_drop)

# Remaining dataset overview
print("\nCleaned DataFrame (columns not associated with smoking/alcohol/cannabis removed):\n")
print(df_filtered.shape)
df_filtered.describe()
df_filtered.info()

# Move cleaned dataframe to df for convenience
df = df_filtered
```

```
Removed columns: ['UND_010', 'UND_020', 'MET_010', 'XTC_010', 'HAL_010', 'HER_010', 'COC_010', 'SYN_010', 'BZP_010', 'TNB_010', 'TRP_

Cleaned DataFrame (columns not associated with smoking/alcohol/cannabis removed):

(61096, 58)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61096 entries, 0 to 61095
Data columns (total 58 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   SEQID       61091 non-null  object
 1   PROVID      61096 non-null  object
 2   GRADE       61096 non-null  int64
 3   DVGENDER    56071 non-null  object
 4   DVURBAN     61096 non-null  object
 5   DVRES       59793 non-null  object
 6   DVORIENT    50683 non-null  object
 7   DVDESCRIBE  58548 non-null  object
 8   WTPUMF      61096 non-null  float64
 9   GH_010      60248 non-null  object
 10  GH_020      60160 non-null  object
 11  SS_010      61011 non-null  object
 12  TS_011      60834 non-null  object
 13  TP_016      59237 non-null  object
 14  TP_046      58940 non-null  object
 15  TP_056      59433 non-null  object
 16  TP_066      59263 non-null  object
 17  TP_086      59242 non-null  object
 18  ELC_026a    59670 non-null  object
 19  ELC_026b    58644 non-null  object
 20  ELC_026c    58606 non-null  object
 21  VAP_010     60487 non-null  object
 22  CI_010      59877 non-null  object
 23  VAP_020     59557 non-null  object
 24  VAP_030     58820 non-null  object
 25  VAP_040     58388 non-null  object
```

```
 26  VAP_050a    59872 non-null  object
 27  VAP_050b    58900 non-null  object
 28  VAP_060     59607 non-null  object
 29  ALC_010     60525 non-null  object
 30  ALC_020     26907 non-null  object
 31  ALC_030     27582 non-null  object
 32  ALC_040     26722 non-null  object
 33  ALC_050     27874 non-null  object
 34  NRG_010     60047 non-null  object
 35  NRG_020     59039 non-null  object
 36  NRG_030     58854 non-null  object
 37  NRG_040     58833 non-null  object
 38  NRG_050     58906 non-null  object
 39  ALC_075     25943 non-null  object
 40  CAN_010     60573 non-null  object
 41  CAN_130     59884 non-null  object
 42  CAN_140     59272 non-null  object
 43  BS_010      59810 non-null  object
 44  PR_100      60001 non-null  object
 45  PR_030      59984 non-null  object
 46  PR_050      59860 non-null  object
```

```python
# Compute the correlation matrix
# Convert columns to numeric, errors='coerce' will handle non-numeric values
correlation_matrix = df.apply(pd.to_numeric, errors='coerce').corr()

# Heatmap visualization
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.show()

# Pairplot for key variables
# Convert columns to numeric for pairplot as well
sns.pairplot(df[['GRADE', 'SS_010', 'ALC_010', 'CAN_010']].apply(pd.to_numeric, errors='coerce'))
plt.show()
```


Feature Correlation Heatmap
SEQID
GRADE
DVURBAN
DVORIENT
WTPUMF
GH_020
TS_011
TP_046
TP_066
ELC_026a
ELC_026c
CI_010
VAP_030
VAP_050a
VAP_060
ALC_020
ALC_040
NRG_010
NRG_030
NRG_050
CAN_010
CAN_140
PR_100
PR_050
PR_110
ELC_041
ALC_080
PR_090
DVTY2ST
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75


GRADE
SS_010
ALC_010