```
!pip install pyvis
```

```
Collecting pyvis
  Downloading pyvis-0.3.2-py3-none-any.whl.metadata (1.7 kB)
Requirement already satisfied: ipython>=5.3.0 in /usr/local/lib/python3.11/
Requirement already satisfied: jinja2>=2.9.6 in /usr/local/lib/python3.11/d
Requirement already satisfied: jsonpickle>=1.4.1 in /usr/local/lib/python3.
Requirement already satisfied: networkx>=1.11 in /usr/local/lib/python3.11/
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.1
Collecting jedi>=0.16 (from ipython>=5.3.0->pyvis)
  Downloading jedi-0.19.2-py2.py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: decorator in /usr/local/lib/python3.11/dist-
Requirement already satisfied: pickleshare in /usr/local/lib/python3.11/dis
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.11/
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0
Requirement already satisfied: pygments in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: backcall in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: matplotlib-inline in /usr/local/lib/python3.
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.11/dis
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11
Requirement already satisfied: parso<0.9.0,>=0.8.4 in /usr/local/lib/python
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.11
Requirement already satisfied: wcwidth in /usr/local/lib/python3.11/dist-pa
Downloading pyvis-0.3.2-py3-none-any.whl (756 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 756.0/756.0 kB 13.8 MB/s eta 0:
Downloading jedi-0.19.2-py2.py3-none-any.whl (1.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 60.7 MB/s eta 0:00:0
Installing collected packages: jedi, pyvis
Successfully installed jedi-0.19.2 pyvis-0.3.2
```

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

```
Mounted at /content/drive
```

```
import pandas as pd
import numpy as np
import os
import warnings
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelEncoder
import plotly.express as px
from tqdm import tqdm
from tqdm.auto import tqdm
from google.colab import drive
import os
import time
import plotly.graph_objects as go
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
import seaborn as sns
```

```
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.float_format', '{:.2f}'.format)
```

```
df=pd.read_csv('/content/drive/MyDrive/FEasmaa/king_last.csv')
```

```
df.isnull().sum().sum()
```

```
np.int64(130048023)
```

```
df['AGE_GROUP'] = (df['AGE_YEARS'] // 10) * 10

columns_with_nan = df.columns[df.isnull().any()].tolist()
columns_with_nan = [col for col in columns_with_nan if col != 'AGE_GROUP']

for col in tqdm(columns_with_nan, desc="Imputing missing values"):
    df[col] = df.groupby('AGE_GROUP')[col].transform(lambda x: x.fillna(x.mean(
```

```
Imputing missing values: 100% 36/36 [00:14<00:00, 2.50it/s]
```

```
df.isnull().sum().sum()
```

```
np.int64(10743429)
```

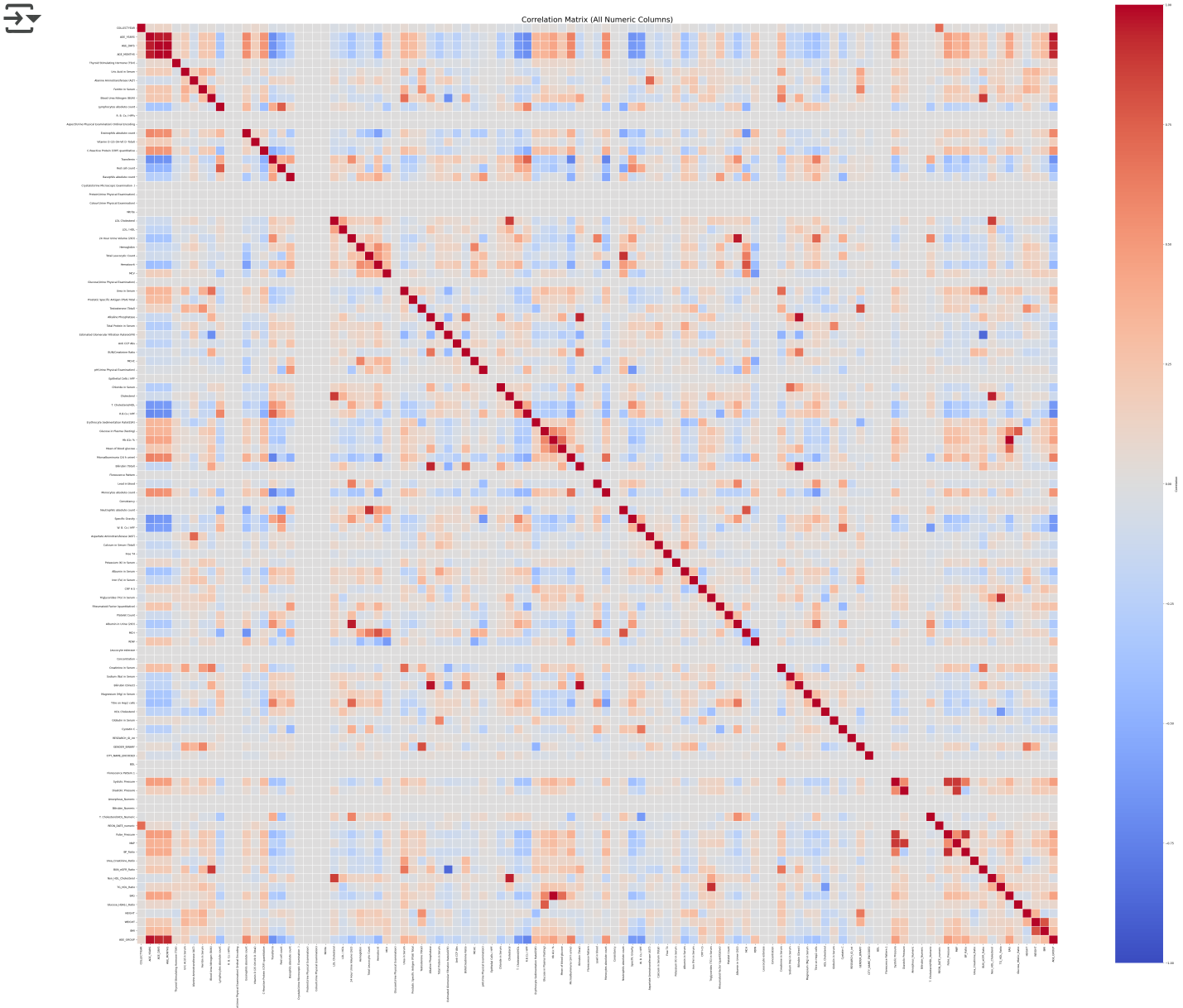
```
for col in tqdm(columns_with_nan, desc="Imputing missing values with group fall
    overall_mean = df[col].mean()
    df[col] = df.groupby('AGE_GROUP')[col].transform(lambda x: x.fillna(x.mean(
```

```
Imputing missing values with group fallback: 100% 36/36 [00:14<00:00, 2.48it/s]
```

```
df.isnull().sum().sum()
```

```
np.int64(0)
```

```
excluded_columns = ['Unnamed: 0', 'RESEARCH_ID', 'SAMPLE_ID', 'REGN_DATE']
df_numeric = df.drop(columns=excluded_columns, errors='ignore').select_dtypes(i


df_for_corr = df_numeric.replace(0, np.nan).fillna(df_numeric.mean(numeric_only


all_columns = df_for_corr.columns.tolist()
corr = df_for_corr.corr()
corr = corr.reindex(index=all_columns, columns=all_columns)
corr_filled = corr.fillna(0).round(2)

plt.figure(figsize=(60, 50))
sns.heatmap(
    corr_filled,
    cmap='coolwarm',
    vmin=-1,
    vmax=1,
    annot=False,
    square=True,
    linewidths=0.3,
    cbar_kws={'label': 'Correlation'}
)


plt.title('Correlation Matrix (All Numeric Columns)', fontsize=26)
plt.xticks(rotation=90, fontsize=9)
plt.yticks(fontsize=9)
plt.tight_layout()


plt.show()

plt.savefig('/content/correlation_matrix2_full.png', dpi=600, bbox_inches='tigh
```



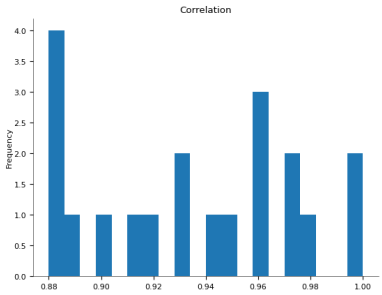
<Figure size 640x480 with 0 Axes>

```
top_corr = (
    corr_filled.where(np.triu(np.ones(corr_filled.shape), k=1).astype(bool))
    .stack()
    .reset_index()
    .rename(columns={'level_0': 'Feature 1', 'level_1': 'Feature 2', 0: 'Correl
    .sort_values(by='Correlation', key=abs, ascending=False)
)

top_corr.head(20)
```

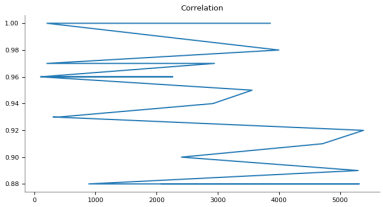
	Feature 1	Feature 2	Correlation
3858	Hb A1c %	EAG	1.00
207	AGE_DAYS	AGE_MONTHS	1.00
3999	Bilirubin (Total)	Bilirubin (Direct)	0.98
206	AGE_YEARS	AGE_GROUP	0.97
2945	Alkaline Phosphatase	Bilirubin (Direct)	0.97
105	AGE_YEARS	AGE_MONTHS	0.96
2263	24 Hour Urine Volume (263)	Albumin in Urine (263)	0.96
104	AGE_YEARS	AGE_DAYS	0.96
3561	Cholesterol	Non_HDL_Cholesterol	0.95
2920	Alkaline Phosphatase	Bilirubin (Total)	0.94
409	AGE_MONTHS	AGE_GROUP	0.93
308	AGE_DAYS	AGE_GROUP	0.93
5383	Pulse_Pressure	BP_Ratio	0.92
4712	Triglycerides (TG) in Serum	TG_HDL_Ratio	0.91
2407	Total Leucocytic Count	Neutrophils absolute count	0.90
5295	Systolic Pressure	MAP	0.89
891	Blood Urea Nitrogen (BUN)	BUN_eGFR_Ratio	0.88
5312	Diastolic Pressure	MAP	0.88
5294	Systolic Pressure	Pulse_Pressure	0.88

3294	Systolic Pressure	Pulse_Pressure	0.88
2076	LDL Cholesterol	Cholesterol	0.88

Distributions

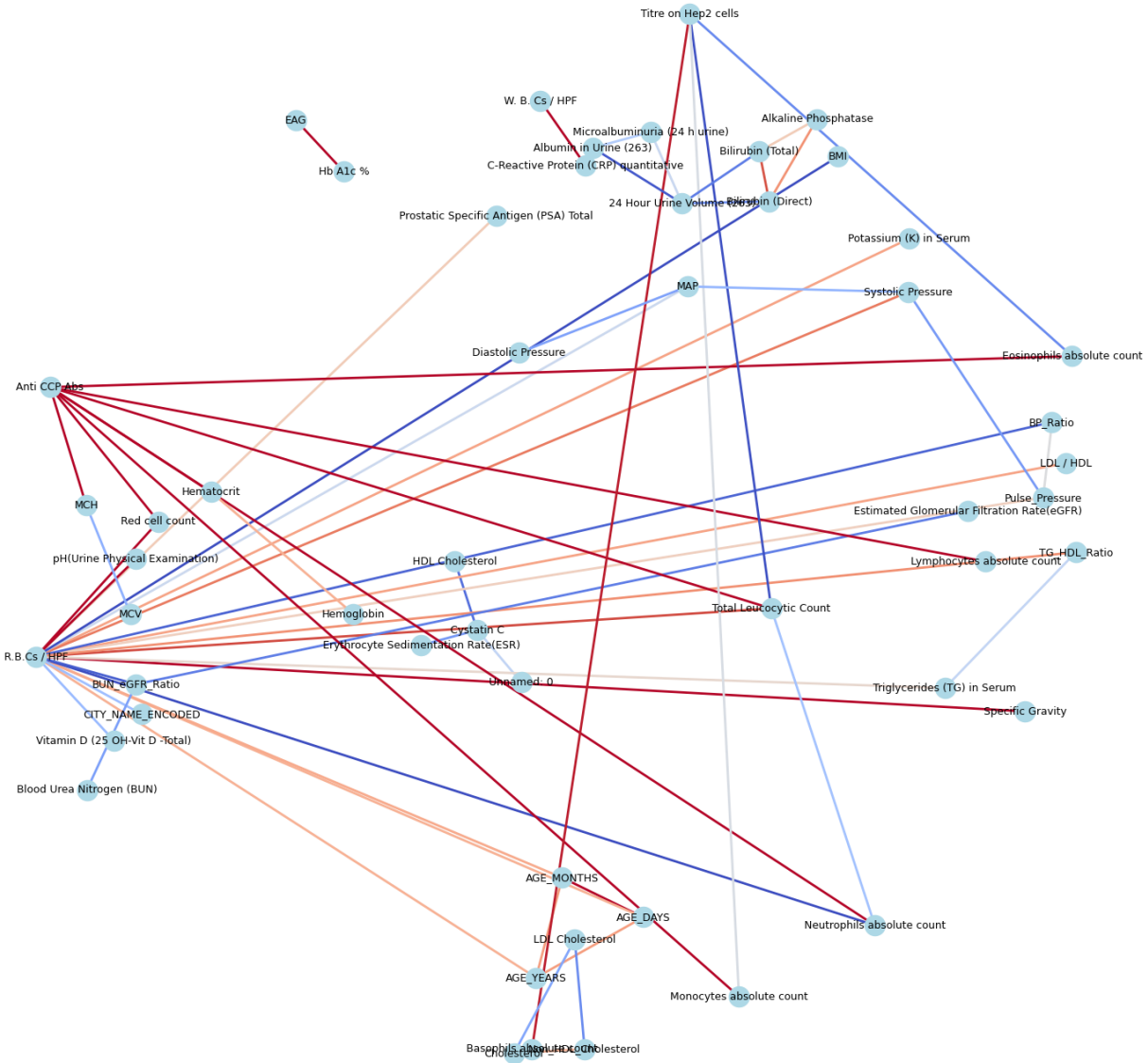


Values



ERROR:root:Did not find quickchart key chart-b017035e-
ERROR:root:Did not find quickchart key chart-b017035e-

```
threshold = 0.85
high_corr_pairs = [
    (col1, col2)
    for col1 in corr_filled.columns
    for col2 in corr_filled.columns
    if col1 != col2 and abs(corr_filled.loc[col1, col2]) >= threshold
]


G = nx.Graph()
for col1, col2 in high_corr_pairs:
    G.add_edge(col1, col2, weight=corr_filled.loc[col1, col2])

plt.figure(figsize=(15, 15))
pos = nx.spring_layout(G, k=0.3)
edges = G.edges(data=True)
weights = [abs(edge[2]['weight']) * 5 for edge in edges]

nx.draw(
    G, pos, with_labels=True, edge_color=weights, width=2.0,
    node_color='lightblue', font_size=9, edge_cmap=plt.cm.coolwarm
)
plt.title('Correlation Network')
plt.show()
```

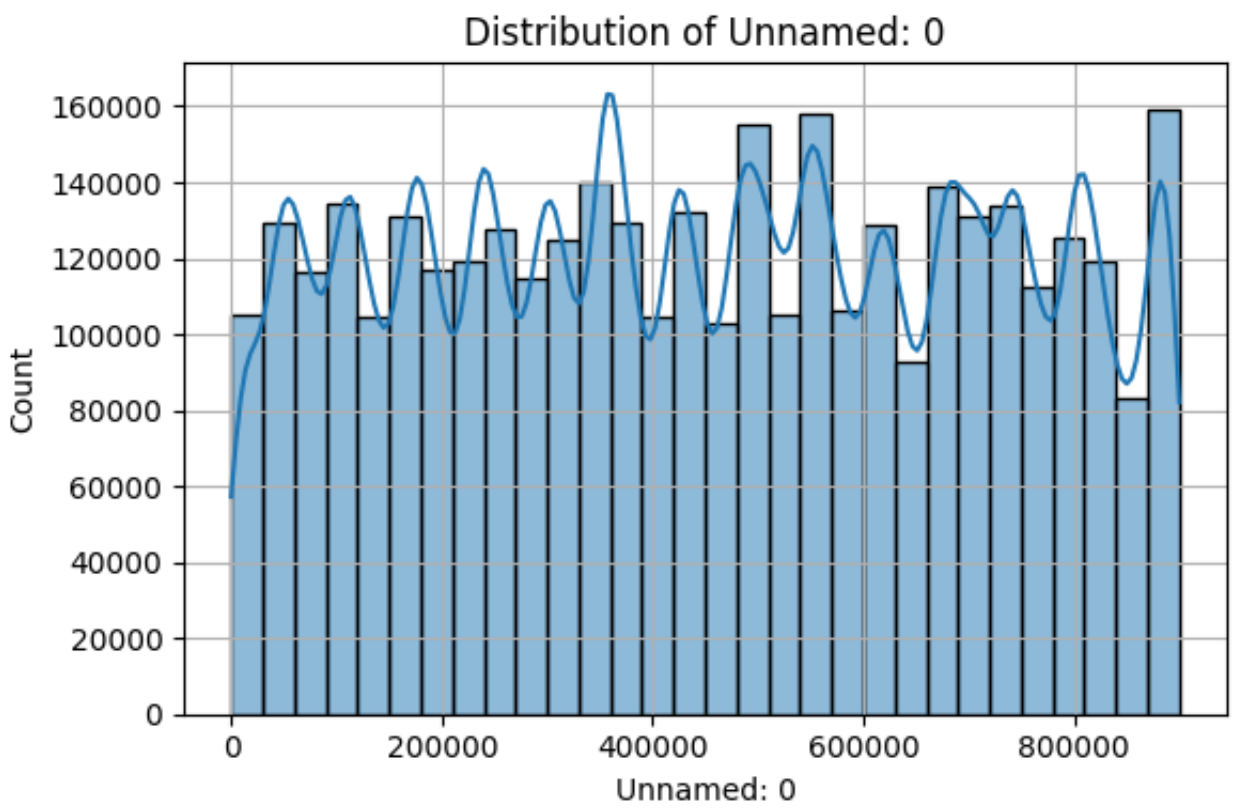

Titre on Hep2 cells
W. B. Cs / HPF
EAG
Alkaline Phosphatase
Microalbuminuria (24 h urine)
Albumin in Urine (263)
Bilirubin (Total)
BMI
Hb A1c %
C-Reactive Protein (CRP) quantitative
24 Hour Urine Volume (2030)
Bilirubin (Direct)
Prostatic Specific Antigen (PSA) Total
Potassium (K) in Serum
MAP
Systolic Pressure
Diastolic Pressure
Eosinophils absolute count
Anti CCP Abs
BP_Ratio
LDL / HDL
Hematocrit
MCH
Red cell count
Pulse_Pressure
Estimated Glomerular Filtration Rate(eGFR)
pH(Urine Physical Examination)
HDL Cholesterol
TG_HDL_Ratio
Lymphocytes absolute count
MCV
Hemoglobin
Total Leucocytic Count
Cystatin C
Erythrocyte Sedimentation Rate(ESR)
R.B.Cs / HPF
BUN_eGFR_Ratio
Unnamed: 0
Triglycerides (TG) in Serum
Specific Gravity
CITY_NAME_ENCODED
Vitamin D (25 OH-Vit D -Total)
Blood Urea Nitrogen (BUN)
AGE_MONTHS
AGE_DAYS
Neutrophils absolute count
LDL Cholesterol
AGE_YEARS
Monocytes absolute count

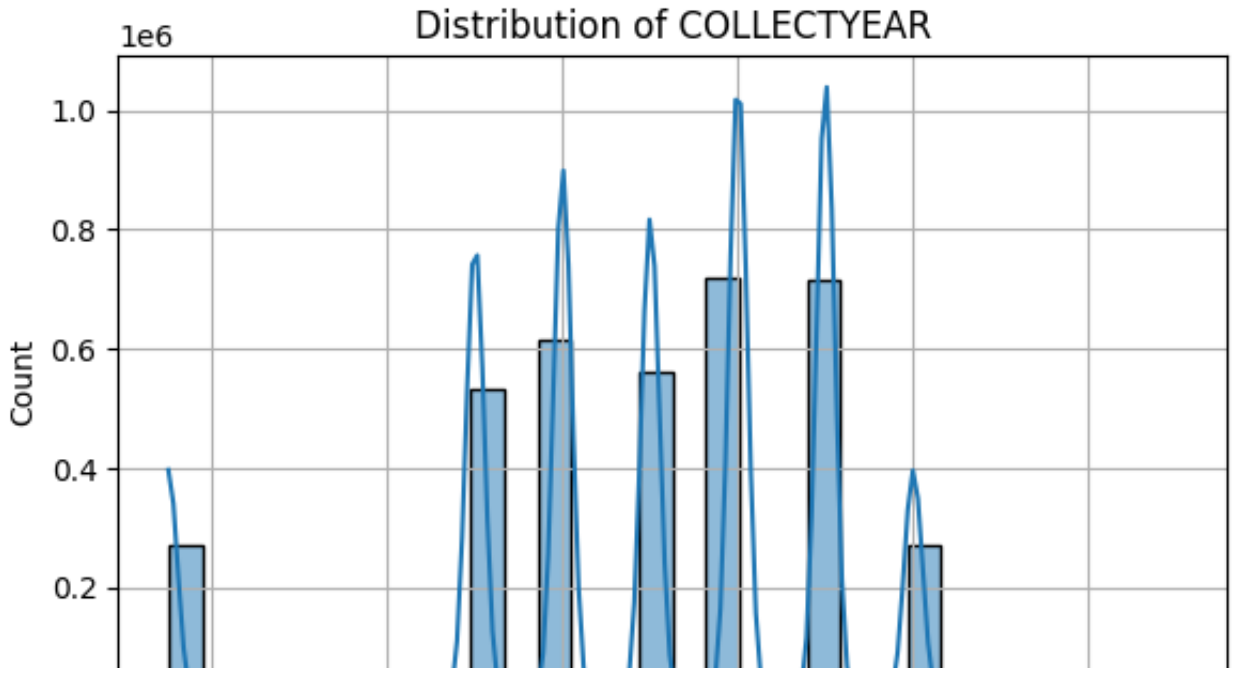
```
excluded_columns = ['Unnamed: 0', 'RESEARCH_ID', 'SAMPLE_ID', 'REGN_DATE']
df_numeric = df.drop(columns=excluded_columns, errors='ignore').select_dtypes(i


df_for_corr = df_numeric.replace(0, np.nan).fillna(df_numeric.mean(numeric_only


all_columns = df_for_corr.columns.tolist()
corr = df_for_corr.corr()
corr = corr.reindex(index=all_columns, columns=all_columns)
corr_filled = corr.fillna(0).round(2)


fig = go.Figure(data=go.Heatmap(
    z=corr_filled.values,
    x=all_columns,
    y=all_columns,
    colorscale='RdBu',
    zmin=-1,
    zmax=1,
    colorbar=dict(title='Correlation'),
    hovertemplate="<b>X:</b> %{x}<br><b>Y:</b> %{y}<br><b>Correlation:</b> %{z}
))


fig.update_layout(
    title='Interactive Correlation Matrix ',
    autosize=False,
    width=2200,
    height=2200,
    template='plotly_white',
    font=dict(family='Arial', size=11),
    margin=dict(l=150, r=150, t=100, b=150),
    xaxis=dict(
        tickangle=45,
        automargin=True,
        tickfont=dict(size=9),
    ),
```

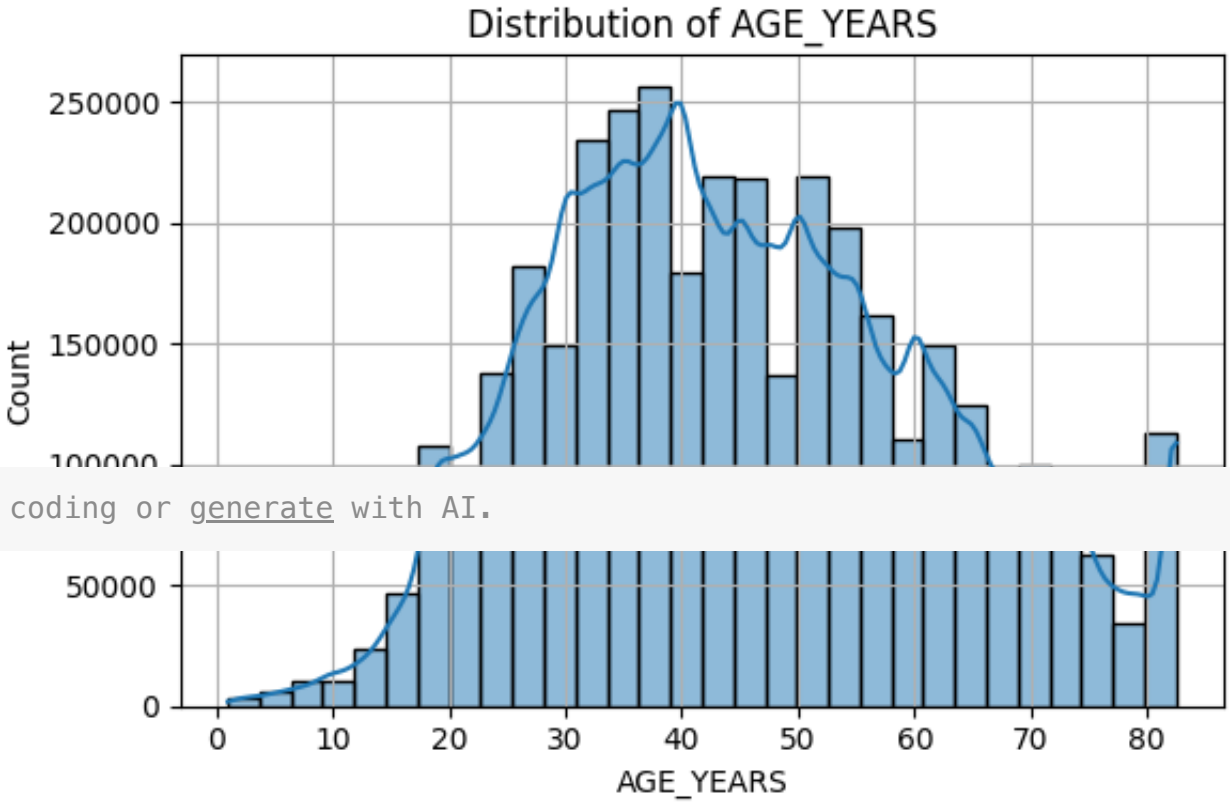
```
    yaxis=dict(
        automargin=True,
        tickfont=dict(size=9),
    )
)


fig.show()


fig.write_html("/content/interactive.html")
```

```
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
categorical_cols = df.select_dtypes(include='object').columns


for col in numeric_cols:
    plt.figure(figsize=(6, 4))
    sns.histplot(df[col], kde=True, bins=30)
    plt.title(f"Distribution of {col}")
    plt.xlabel(col)
    plt.ylabel("Count")
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

```
for col in categorical_cols:
    plt.figure(figsize=(8, 4))
    value_counts = df[col].value_counts().head(20)
    sns.barplot(x=value_counts.values, y=value_counts.index)
    plt.title(f"Top Categories in {col}")
    plt.xlabel("Count")
    plt.ylabel(col)
    plt.tight_layout()
    plt.show()
```







Start coding or generate with AI.