Convert columns to correct formats

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

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

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
from sklearn.preprocessing import LabelEncoder, OrdinalEncoder
import plotly.express as px
```

```
!pip install category_encoders
```

```
Collecting category_encoders
  Downloading category_encoders-2.8.0-py3-none-any.whl.metadata (7.9 kB)
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.11/d
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: scikit-learn>=1.6.0 in /usr/local/lib/python
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/pytho
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-p
Downloading category_encoders-2.8.0-py3-none-any.whl (85 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.7/85.7 kB 2.1 MB/s eta 0:00:
Installing collected packages: category_encoders
Successfully installed category_encoders-2.8.0
```

```
df=pd.read_csv("/content/drive/MyDrive/FE/FE462.csv")
```

```
<ipython-input-4-ade62a7a98d2>:1: DtypeWarning: Columns (14,15,16,17,19,20,
  df=pd.read_csv("/content/drive/MyDrive/FE/FE462.csv")
```

```
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
```

The df.shape attribute in Pandas returns the dimensions of a DataFrame as a tuple (number of rows, number of columns).

```
df.shape
```

```
(900000, 102)
```

```
df.dtypes
```

	0
Unnamed: 0	int64
RESEARCH_ID	object
SAMPLE_ID	object
COLLECTYEAR	float64
REGN_DATE	object
GENDER_NAME	object
AGE_YEARS	float64
AGE_DAYS	float64
AGE_MONTHS	float64
CITY_NAME	object
HEIGHT	int64
WEIGHT	float64
BMI	float64
'Thyroid Stimulating Hormone (TSH)'	object
'Uric Acid in Serum'	object
'Alanine Aminotransferase (ALT)'	object
'Ferritin In Serum'	object
'Blood Urea Nitrogen (BUN)'	object
'Lymphocytes absolute count'	float64
'B. B. Co. / HBEs'	object

R. B. Cs / HPFs	object
'Aspect(Urine Physical Examination)'	object
'Eosinophils absolute count'	float64
'Vitamin D (25 OH-Vit D -Total)'	object
'C-Reactive Protein (CRP) quantitative'	object
'Transferrin'	object
'Height.'	float64
'Red cell count'	object
'Basophils absolute count'	float64
'Crystals(Urine Microscopic Examination :)'	object
'Protein(Urine Physical Examination)'	object
'Colour(Urine Physical Examination)'	object
'LDL / HDL'	float64
'24 Hour Urine Volume (263)'	float64
'MCV'	float64
'Glucose(Urine Physical Examination)'	object
'Urea in Serum'	object
'Prostatic Specific Antigen (PSA) Total'	object
'Testosterone (Total)'	object
'Alkaline Phosphatase'	object
'Total Protein in Serum'	object
'Estimated Glomerular Filtration Rate(eGFR)'	object
'Anti CCP Abs'	object
' BUN/Creatinine Ratio'	object
'Blood pressure'	object

```
all_null_columns = df.columns[df.isnull().all()]
all_null_columns
```

```
Index(['''Height.''', '''Non-HDL Cholesterol''', '''Weight.''', '''BMI'''],
dtype='object')
```

```
df.drop(columns=all_null_columns, inplace=True)
```

```
df.columns = df.columns.str.replace("'", "")
```

```
df.columns
```

```
Index(['Unnamed: 0', 'RESEARCH_ID', 'SAMPLE_ID', 'COLLECTYEAR',
       'REGN_DATE',
       'GENDER_NAME', 'AGE_YEARS', 'AGE_DAYS', 'AGE_MONTHS', 'CITY_NAME',
       'HEIGHT', 'WEIGHT', 'BMI', 'Thyroid Stimulating Hormone (TSH)',
       'Uric Acid in Serum', 'Alanine Aminotransferase (ALT)',
       'Ferritin in Serum', 'Blood Urea Nitrogen (BUN)',
       'Lymphocytes absolute count', 'R. B. Cs / HPFs',
       'Aspect(Urine Physical Examination)', 'Eosinophils absolute count',
       'Vitamin D (25 OH-Vit D -Total)',
       'C-Reactive Protein (CRP) quantitative', 'Transferrin',
       'Red cell count', 'Basophils absolute count',
       'Crystals(Urine Microscopic Examination :)',
       'Protein(Urine Physical Examination)',
       'Colour(Urine Physical Examination)', 'Nitrite', 'LDL Cholesterol',
       'LDL/HDL', 'Urine 24 hour Urine Volume(263)', 'Hemoglobin',
       'Total Leucocytic Count', 'Hematocrit', 'MCV',
       'Glucose(Urine Physical Examination)', 'Urea in Serum',
       'Prostatic Specific Antigen (PSA) Total', 'Testosterone (Total)',
       'Alkaline Phosphatase', 'Total Protein in Serum',
       'Estimated Glomerular Filtration Rate(eGFR)', 'Anti CCP Abs',
       ' BUN/Creatinine Ratio', 'Blood pressure', 'Ketones', 'MCHC',
       'pH(Urine Physical Examination)', 'Amorphous Elements',
       'Blood and Haemoglobin', 'Epithelial Cells / HPF',
       'Casts(Urine Microscopic Examination :)', 'Bilirubin',
       'Chloride in Serum', 'Cholesterol', 'T. Cholesterol/HDL',
       'Urobilinogen', 'R.B.Cs / HPF', 'Erythrocyte Sedimentation
Rate(ESR)',
       'Glucose in Plasma (Fasting)', 'Hb A1c %', 'Mean of blood glucose
',
       'Microalbuminuria (24 h urine)', 'Bilirubin (Total)',
       'Florescence Pattern', 'Lead in blood', 'Monocytes absolute count',
       'Consistancy', 'Neutrophils absolute count', 'Specific Gravity',
       'W. B.Cs / HPF', 'Aspartate Aminotransferase (AST)',
       'Calcium in Serum (Total)', 'Free T4', 'Potassium (K) in Serum',
       'Albumin in Serum', 'Iron (Fe) in Serum', 'CRP H.S',
       'Triglycerides (TG) in Serum', 'Rheumatoid Factor (quantitative)',
       'Platelet Count', 'Albumin in Urine(263)', 'MCH', 'RDW',
       'W.B.Cs / HPF', 'Leucocyte esterase', 'Concentration',
       'Creatinine in Serum', 'Sodium (Na) in Serum', 'Bilirubin
(Direct)',
       'Magnesium (Mg) in Serum', 'Titre on Hep2 cells', 'HDL
Cholesterol',
       'Globulin in Serum', 'Cystatin C'],
      dtype='object')
```

```
df= df.sort_values(by='RESEARCH_ID')
```

```
df.head()
```

	Unnamed: 0	RESEARCH_ID	SAMPLE_ID	COLLECTYEAR	REGN_DATE	GENDER_
15940	15940	R015-23-1	21157313R015-23-1	2015.0	2015-02-08	I
6921	6921	R015-23-10	17154493R015-23-10	2015.0	2015-01-18	I
23108	23108	R015-23-100	251515977R015-23-100	2015.0	2015-02-23	FEI
145172	145172	R015-23-100	251549513R015-23-100	2015.0	2015-12-30	FEI
168477	168477	R015-23-1000	171615838R015-23-1000	2016.0	2016-03-01	FEI

```
df = df.rename(columns={'Unnamed: 0': 'ID'})
```

```
df=df.set_index('ID')
```

'Sodium (Na) in Serum'	object
'Bilirubin (Direct)'	object
'Magnesium (Mg) in Serum'	object
'Titre on Hep2 cells'	object
'HDL Cholesterol'	object
'Globulin in Serum'	float64
'Cystatin C'	object

dtype: object

```
df.head()
```

ID	RESEARCH_ID	SAMPLE_ID	COLLECTYEAR	REGN_DATE	GENDER_NAME	AGE_
15940	R015-23-1	21157313R015-23-1	2015.0	2015-02-08	MALE	
6921	R015-23-10	17154493R015-23-10	2015.0	2015-01-18	MALE	
23108	R015-23-100	251515977R015-23-100	2015.0	2015-02-23	FEMALE	
145172	R015-23-100	251549513R015-23-100	2015.0	2015-12-30	FEMALE	
168477	R015-23-1000	171615838R015-23-1000	2016.0	2016-03-01	FEMALE	

This column does not need to be converted to a digital column because it is usually not used in Moodle

```
df["RESEARCH_ID"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: RESEARCH_ID
Non-Null Count   Dtype
--------------   -----
900000 non-null  object
dtypes: object(1)
memory usage: 13.7+ MB
```

```
df['RESEARCH_ID_int'] = df['RESEARCH_ID'].astype(str).str.replace(r'R|-', '', r
df[['RESEARCH_ID_int']].head(20)
```

ID	RESEARCH_ID_int
15940	015231
6921	0152310
23108	01523100
145172	01523100
168477	015231000
801640	0152310000
689903	01523100000
688264	01523100000
687419	01523100000
684844	01523100001
787224	01523100001
689542	01523100002
263272	01523100002
689543	01523100002
711716	01523100003
729838	01523100004
56769	01523100004
86414	01523100005
676308	01523100005
840785	01523100005

```
#pattern_analysis = df["RESEARCH_ID"].astype(str).apply(lambda x: re.findall(r'
```

```
#pattern_analysis
```

```
df.head()
```

ID	RESEARCH_ID	SAMPLE_ID	COLLECTYEAR	REGN_DATE	GENDER_NAME	AGE_
15940	R015-23-1	21157313R015-23-1	2015.0	2015-02-08	MALE	
6921	R015-23-10	17154493R015-23-10	2015.0	2015-01-18	MALE	
23108	R015-23-100	251515977R015-23-100	2015.0	2015-02-23	FEMALE	
145172	R015-23-100	251549513R015-23-100	2015.0	2015-12-30	FEMALE	
168477	R015-23-1000	171615838R015-23-1000	2016.0	2016-03-01	FEMALE	

In this step, a column was converted to int while preserving the original column because it contains a pattern that could be important.

```
df['COLLECTYEAR'].isna().sum()
```

```
61962
```

```
df['COLLECTYEAR'] = pd.to_datetime(df['COLLECTYEAR'].astype('Int64'), format='%
```

```
#df.loc[df['COLLECTYEAR'] == 0, 'COLLECTYEAR'] = df['REGN_DATE'].dt.year
```

```
df['COLLECTYEAR'].head()
```

ID	COLLECTYEAR
15940	2015-01-01
6921	2015-01-01
23108	2015-01-01
145172	2015-01-01
168477	2016-01-01

dtype: datetime64[ns]

```
df['COLLECTYEAR'].isnull().sum()
```

```
61962
```

```
df['COLLECTYEAR'].value_counts()
```

COLLECTYEAR	count
2019-01-01	174895
2018-01-01	173113
2016-01-01	159215
2015-01-01	138885
2017-01-01	127094
2020-01-01	64628
2021-01-01	144
2022-01-01	46
2023-01-01	18

dtype: int64

```
df['COLLECTYEAR'] = pd.to_datetime(df['COLLECTYEAR'])

df['COLLECTYEAR'] = df['COLLECTYEAR'].dt.year
```

```
df['COLLECTYEAR'] = df['COLLECTYEAR'].fillna(0).astype('int64')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-b93ec9259c44> in <cell line: 0>()
----> 1 df['COLLECTYEAR'] = df['COLLECTYEAR'].fillna(0).astype('int64')

NameError: name 'df' is not defined
```

```
df['COLLECTYEAR'] = df['COLLECTYEAR'].astype('int64')
```

```
df['COLLECTYEAR'].value_counts()
```

COLLECTYEAR	count
2019	174895
2018	173113
2016	159215
2015	138885
2017	127094
2020	64628
0	61962
2021	144
2022	46
2023	18

dtype: int64

```
df['REGN_DATE'] = pd.to_datetime(df['REGN_DATE'])
```

```
df['REGN_DATE'].head()
```

ID	REGN_DATE
15940	2015-02-08
6921	2015-01-18
23108	2015-02-23
145172	2015-12-30
168477	2016-03-01

dtype: datetime64[ns]

```
df['REGN_DATE'].value_counts()
```

REGN_DATE	count
2018-09-29	3263
2018-09-22	3225
2015-01-01	3179
2018-09-20	3042
2019-09-23	2947
2018-09-23	2936
2019-09-28	2813
2019-09-22	2733
2019-09-30	2666
2018-09-24	2647
2018-09-30	2490
2018-09-19	2434
2019-09-21	2368
2018-09-15	2164
2019-09-29	2132
2018-09-18	2058

2018-09-17	1974
2017-09-24	1926
2018-09-27	1851
2019-09-26	1733
2018-09-16	1731
2019-09-14	1698
2015-12-31	1598
2016-12-31	1548
2019-09-19	1499
2017-09-23	1461
2018-09-25	1395
2018-12-31	1310
2018-09-26	1297
2019-02-16	1280

```
df['GENDER_BINARY'] = df['GENDER_NAME'].map({'MALE': 1, 'FEMALE': 0})
```

```
df['AGE_YEARS'] = df['AGE_YEARS'].astype('Int64')
```

2019-04-30	1100
2018-04-14	1092
2016-10-29	1084

```
df['AGE_DAYS'] = df['AGE_DAYS'].fillna(0).astype('Int64')
df[['AGE_DAYS']].head(20)
```

ID	AGE_DAYS
15940	20089
6921	12785
23108	15706
145172	16016
168477	13515
801640	10593
689903	24203
688264	24204
687419	24203
684844	17533
787224	17685
689542	21046
263272	20090
689543	21047
711716	29221
729838	11338
56769	9861
86414	13706
676308	0
840785	15292

2019-03-16	926
2015-09-22	922
2017-03-11	917
2016-03-26	916
2019-09-12	914

```
df['AGE_MONTHS'] = df['AGE_MONTHS'].fillna(0).astype('Int64')
df[['AGE_MONTHS']].head(20)
```

ID	AGE_MONTHS
15940	670
6921	426
23108	524
145172	534
168477	451
801640	353
689903	807
688264	807
687419	807
684844	584
787224	589
689542	702
263272	670
689543	702
711716	974
729838	378
56769	329
86414	457
676308	0
840785	510
2016-12-03	842
2015-02-28	842

There is bias in AGE_YEARS, AGE_DAYS, AGE_MONTHS

```
df["AGE_YEARS"] = df["AGE_YEARS"].abs()
df["AGE_DAYS"] = df["AGE_DAYS"].abs()
df["AGE_MONTHS"] = df["AGE_MONTHS"].abs()
```

```
df.loc[((df["AGE_YEARS"] * 365 - df["AGE_DAYS"]).abs() > 365), "AGE_DAYS"] = df
df.loc[((df["AGE_YEARS"] * 12 - df["AGE_MONTHS"]).abs() > 12), "AGE_MONTHS"] =
mismatches_after_fix = {
    " AGE_YEARS و AGE_DAYS  ": ((df["AGE_YEARS"] * 365 - df["AGE_DAYS"]).abs()
    " AGE_YEARS و AGE_MONTHS: ((df["AGE_YEARS"] * 12 - df["AGE_MONTHS"]).abs()
}
print(mismatches_after_fix)
```

Show hidden output

2019-11-09	822
2015-12-28	821
2019-09-15	817
2016-06-05	814
2015-10-31	813
2018-12-27	806
2015-09-17	805
2019-04-20	801
2015-12-26	800
2019-04-28	799
2019-04-13	798
2017-12-30	797
2017-11-04	795
2019-12-31	792
2017-10-14	791
2020-06-20	791
2018-10-27	790
2020-06-03	785
2015-02-14	784
2019-11-02	781
2019-12-28	780
2019-09-11	778
2016-02-06	778
2018-02-03	776
2018-09-12	776

##الفصل الذي تم فيه اخذ التحليل لمعرفت تأثير الفحوص على الصحه

ظهرت حمة 258,530 مرة (Jeddah).

ظهرت حمة واحدة فقط (Abu Dhabi).

```
df['CITY_NAME'] = df['CITY_NAME'].str.strip().str.title()
df['CITY_NAME'] = df['CITY_NAME'].astype('category')
df['CITY_NAME'].head(20)
```

ID	CITY_NAME
15940	Hail
6921	Abha
23108	Makkah
145172	Makkah
168477	Abha
801640	Madinah
689903	Jeddah
688264	Jeddah
687419	Jeddah
684844	Riyadh
787224	Riyadh
689542	Tabouk
263272	Tabouk
689543	Tabouk
711716	Makkah
729838	Riyadh
56769	Riyadh
86414	Jeddah
676308	Jeddah
840785	Jeddah

dtype: category

2019-10-26	731
2019-09-10	727
2015-09-05	727
2020-02-22	726

"Western Lab Management" "**Auditing**"

```
invalid_cities = ["Auditing", "Central Lab Management", "Western Lab Management
df['CITY_NAME'] = df['CITY_NAME'].replace(invalid_cities, "Administrative Locat
```

```
<ipython-input-43-1cbf2f9389e1>:2: FutureWarning: The behavior of Series.re
  df['CITY_NAME'] = df['CITY_NAME'].replace(invalid_cities, "Administrative
```

2020-06-21	725
2016-11-19	725
2020-06-06	722
2018-10-29	722
2019-09-07	722
2016-03-14	719
2016-03-13	718
2019-02-14	718
2015-12-19	717
2019-11-16	716
2020-06-02	712
2017-12-16	712
2017-02-11	712
2020-06-04	710
2018-06-02	708
2019-03-02	708
2015-05-02	707
2019-03-28	706
2016-07-31	705
2016-03-28	703
2017-05-13	702
2015-04-11	701
2016-03-16	701
2019-04-04	701
2016-11-14	700
2016-12-28	700
2018-05-30	700
2019-08-31	699

```
label_encoder = LabelEncoder()
df['CITY_NAME_ENCODED'] = label_encoder.fit_transform(df['CITY_NAME'])
df[['CITY_NAME', 'CITY_NAME_ENCODED']].head(20)
```

ID	CITY_NAME	CITY_NAME_ENCODED
15940	Hail	12
6921	Abha	0
23108	Makkah	21
145172	Makkah	21
168477	Abha	0
801640	Madinah	20
689903	Jeddah	14
688264	Jeddah	14
687419	Jeddah	14
684844	Riyadh	25
787224	Riyadh	25
689542	Tabouk	28
263272	Tabouk	28
689543	Tabouk	28
711716	Makkah	21
729838	Riyadh	25
56769	Riyadh	25
86414	Jeddah	14
676308	Jeddah	14
840785	Jeddah	14

2017-04-02 683

```
def clean_and_preserve_tsh(value):
    if pd.isna(value):
        return np.nan

    value_str = str(value).strip()
```

```
    if value_str.startswith("<"):
        try:
            return float(value_str[1:]) - 0.0001
        except ValueError:
            return np.nan


    if value_str.startswith(">"):
        try:
            return float(value_str[1:]) + 0.0001
        except ValueError:
            return np.nan


    cleaned_value = re.sub(r'[\*\(\)]', '', value_str)

    try:
        return float(cleaned_value)
    except ValueError:
        return np.nan

df["Thyroid Stimulating Hormone (TSH)"] = df["Thyroid Stimulating Hormone (TSH)
df["Thyroid Stimulating Hormone (TSH)"].dtype, df["Thyroid Stimulating Hormone
```

```
(dtype('float64'),
 ID
 15940     11.14
 6021       0.84
 23108      0.75
 145172     1.04
 168477     1.62
 2016-04    2.56
 689903      NaN
 688264      NaN
 687419     1.38
 684844     4.85
 787224     0.19
 689542     1.47
 263272      NaN
 689543      NaN
 729838     2.33
 30469      1.89
 86414       NaN
 676308      NaN
 840785      NaN
 Name: Thyroid Stimulating Hormone (TSH), dtype: float64)
```

2018-03-12 673
2019-08-03 673
2015-11-28 672
2016-09-27 672
2016-04-16 672
2016-03-30 672
2016-12-27 671
2020-06-24 669
2015-05-16 669
2017-01-21 667
2020-06-11 666
2015-12-27 663
2020-03-02 662
2017-04-01 660
2019-03-09 660

```
df["Uric Acid in Serum"] = df["Uric Acid in Serum"].replace(9.999999e+99, np.na
df["Uric Acid in Serum"] = pd.to_numeric(df["Uric Acid in Serum"], errors='coer
df["Uric Acid in Serum"].dtype, df["Uric Acid in Serum"].head(20)
```

```
(dtype('float64'),
ID
15940      7.2
6921       NaN
25188      4.4
145172     3.7
108477     3.4
801640     NaN
689903     NaN
688264     NaN
687419     6.8
684844     4.9
787224     5.5
695542     6.6
263272     NaN
689543     NaN
711716     5.8
729858     5.2
56769      6.0
86414      NaN
676308     NaN
840785     NaN
Name: Uric Acid in Serum, dtype: float64)
```

2020-06-07	654
2017-01-14	652
2018-06-30	651
2018-04-03	649
2016-11-15	648
2016-05-30	647
2017-12-31	647
2018-12-26	647
2020-01-15	646
2019-03-24	646
2019-12-14	643
2015-08-29	642
2017-01-07	642
2019-04-02	642
2016-04-11	641
2020-05-31	641
2019-12-07	641
2019-07-27	640
2016-08-28	640
2018-10-18	639
2018-05-05	637
2016-03-24	637
2015-09-14	637
2018-12-24	637
2016-09-26	635
2018-04-21	635
2016-12-05	634

```
df.head()
```

ID	RESEARCH_ID	SAMPLE_ID	COLLECTYEAR	REGN_DATE	GENDER_NAME	AGE_
15940	R015-23-1	21157313R015-23-1	2015	2015-02-08	MALE	
6921	R015-23-10	17154493R015-23-10	2015	2015-01-18	MALE	
23108	R015-23-100	251515977R015-23-100	2015	2015-02-23	FEMALE	
145172	R015-23-100	251549513R015-23-100	2015	2015-12-30	FEMALE	
168477	R015-23-1000	171615838R015-23-1000	2016	2016-03-01	FEMALE	

2017-09-16	626
2017-02-18	624
2015-06-13	624
2016-08-21	624
2019-06-29	624
2020-06-15	624
2018-04-02	623
2016-04-04	622
2020-06-09	622
2018-03-28	622
2019-02-07	622
2015-06-16	621
2016-04-26	621
2019-03-11	621
2016-05-14	621
2016-08-23	620

```
df["Alanine Aminotransferase (ALT)"] = pd.to_numeric(df["Alanine Aminotransfera
df["Alanine Aminotransferase (ALT)"].dtype, df["Alanine Aminotransferase (ALT)"
```

```
(dtype('float64'),
15940     15.0
23108     23.0
145172    24.0
168477    31.0
801640     NaN
689903     NaN
688264     NaN
687419     NaN
684844    19.0
679224    23.0
689542    13.0
203272     NaN
689543     NaN
711710    15.0
729838    13.0
56769     24.0
86414      NaN
676308     NaN
840785     7.0
Name: Alanine Aminotransferase (ALT), dtype: float64)
```

2016-11-12	619
2020-01-04	619
2015-11-30	618
2017-04-26	617
2018-11-24	616
2016-03-12	616
2017-12-02	616
2019-04-03	616
2018-11-29	616
2016-08-18	615
2016-05-31	615
2016-03-06	615
2019-07-29	613
2016-07-24	613
2017-09-09	613
2019-01-12	612
2020-01-14	612
2016-06-01	612
2016-02-27	611
2016-04-27	611
2018-12-25	610
2019-04-16	608
2017-12-09	608
2020-03-08	607
2018-03-11	606
2019-02-12	606
2016-05-28	606
2017-05-24	606

```
df["Ferritin In Serum"] = pd.to_numeric(df["Ferritin In Serum"], errors='coerce
df["Ferritin In Serum"].dtype, df["Ferritin In Serum"].head(20)
```

```
(dtype('float64'),
 ID
 19940       NaN
 6921        NaN
 23108       NaN
 145172      NaN
 168477    10.56
 801640      NaN
 689903    160.80
 688264      NaN
 687419      NaN
 787224      NaN
 689542     7.71
 263272      NaN
 689543      NaN
 711716      NaN
 729838      NaN
 56769       NaN
 86414       NaN
 676208      NaN
 840785      NaN
 Name: Ferritin In Serum, dtype: float64)
```

2019-03-26	605
2019-10-10	604
2019-11-23	604
2020-05-30	603
2016-08-30	603
2019-08-01	602
2018-04-13	601
2019-03-21	601
2016-04-06	601
2018-05-29	601
2015-06-15	601
2017-12-23	600
2019-01-26	600
2018-05-16	599
2017-04-08	599
2020-06-23	598
2018-09-01	598
2019-07-06	598
2016-05-16	597
2016-04-10	597
2018-11-10	597
2018-10-14	596
2016-04-21	596
2019-07-13	596
2020-03-01	595
2019-09-08	595
2015-12-24	595
2017-01-28	594

```
df.head()
```

ID	RESEARCH_ID	SAMPLE_ID	COLLECTYEAR	REGN_DATE	GENDER_NAME	AGE_
15940	R015-23-1	21157313R015-23-1	2015	2015-02-08	MALE	
6921	R015-23-10	17154493R015-23-10	2015	2015-01-18	MALE	
23108	R015-23-100	251515977R015-23-100	2015	2015-02-23	FEMALE	
145172	R015-23-100	251549513R015-23-100	2015	2015-12-30	FEMALE	
168477	R015-23-1000	171615838R015-23-1000	2016	2016-03-01	FEMALE	

```
#df.to_csv('/content/drive/MyDrive/FE/FE462_processed.csv', index=False)
```

2018-07-02	586
2016-10-20	586
2018-04-04	586
2019-06-22	585
2016-08-03	585
2019-05-28	585
2018-03-22	585
2019-02-13	585
2016-11-07	584
2015-12-12	584
2019-03-12	584
2019-02-05	583
2016-08-14	583

```
df["Blood Urea Nitrogen (BUN)"] = pd.to_numeric(df["Blood Urea Nitrogen (BUN)"]
df["Blood Urea Nitrogen (BUN)"].dtype, df["Blood Urea Nitrogen (BUN)"].head(20)
```

```
(dtype('float64'),
 ID
 15940     NaN
 6921      NaN
 23108     NaN
 145172    NaN
 168477    5.0
 689903    NaN
 688264    NaN
 687419    9.0
 684844    13.4
 787224    NaN
 689542    11.1
 263272    NaN
 689543    NaN
 711716    16.7
 729838    NaN
 267691    NaN
 86414     NaN
 676508    NaN
 840785    NaN
 Name: Blood Urea Nitrogen (BUN), dtype: float64)
```

```
df["Blood Urea Nitrogen (BUN)"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Blood Urea Nitrogen (BUN)
Non-Null Count   Dtype
--------------   -----
361245 non-null  float64
dtypes: float64(1)
memory usage: 46.0 MB
```

2016-12-06	582
2016-06-30	582
2019-12-21	581
2018-10-07	580
2016-03-07	580
2016-08-08	579
2018-10-16	578
2019-11-28	578
2019-08-24	578
2016-08-07	578
2016-08-04	578
2015-12-21	578
2016-03-08	577
2020-01-25	577
2019-04-15	577
2016-05-29	574
2015-03-21	574
2016-08-17	573
2019-03-19	573
2016-11-27	573
2016-05-07	573
2017-08-27	572
2016-08-22	571
2020-03-05	571
2015-12-05	571
2016-09-05	571
2019-03-17	570

```
df.head()
```

ID	RESEARCH_ID	SAMPLE_ID	COLLECTYEAR	REGN_DATE	GENDER_NAME	AGE_
15940	R015-23-1	21157313R015-23-1	2015	2015-02-08	MALE	
6921	R015-23-10	17154493R015-23-10	2015	2015-01-18	MALE	
23108	R015-23-100	251515977R015-23-100	2015	2015-02-23	FEMALE	
145172	R015-23-100	251549513R015-23-100	2015	2015-12-30	FEMALE	
168477	R015-23-1000	171615838R015-23-1000	2016	2016-03-01	FEMALE	

2018-02-27	566

```
df["R. B. Cs / HPFs"].value_counts()
```

R. B. Cs / HPFs	count
0-1	39
1	23
1 - 3	15
0 - 2	14
1-3	12
0-2	11
1-2	10
3-5	10
2	10
2-4	9
2-3	4

6	3
0.2	3
5	3
6-8	3
20-25	2
2 - 4	2
4	2
10-12	2
3	2
8-10	2
3 - 5	2
1 - 3	1
4.6	1
15-20	1
16-18	1
10	1
5-8	1
0 - 1	1
6 - 8	1
2 - 4	1
11-13	1
3-4	1
> 100	1
6-7	1
12	1
35.40	1
5-7	1
2 - 4	1
70	1
8	1

```
def process_rbcs_values(value):
    if pd.isna(value) or str(value).strip() == "":
        return np.nan

    value = str(value).strip()


    if ">" in value:
        try:
            return float(re.sub(r'[^\d.]', '', value))
        except ValueError:
            return np.nan


    if "–" in value:
        try:
            parts = [float(x) for x in re.findall(r'\d+\.?\d*', value)]
            return max(parts)
        except ValueError:
            return np.nan


    try:
        return float(value)
    except ValueError:
        return np.nan


df["R. B. Cs / HPFs"] = df["R. B. Cs / HPFs"].apply(process_rbcs_values)
df["R. B. Cs / HPFs"].fillna(0, inplace=True)
```

```
<ipython-input-56-a8ce81b06806>:30: FutureWarning: A value is trying to be
The behavior will change in pandas 3.0. This inplace method will never work
For example, when doing 'df[col].method(value, inplace=True)', try using 'd
  df["R. B. Cs / HPFs"].fillna(0, inplace=True)
```

2017-03-12	546
2019-04-09	545
2016-09-03	545
2019-11-03	544
2016-08-15	544
2016-10-18	544
2018-05-14	544
2016-03-10	544
2016-08-25	543
2019-01-19	542

```
df["R. B. Cs / HPFs"].value_counts()
```

R. B. Cs / HPFs	count
0.0	899794
1.0	63
2.0	46
3.0	34
5.0	16
4.0	16
8.0	6
6.0	4
0.2	3
10.0	3
12.0	3
7.0	2
25.0	2
4.6	1
20.0	1
18.0	1
13.0	1
35.4	1
100.0	1
70.0	1
1.3	1

dtype: int64

2017-03-08	536
2016-05-02	536
2018-12-23	535

```
df["Aspect(Urine Physical Examination)"].value_counts()
```

Aspect(Urine Physical Examination)	count
Clear	145
Slightly Turbid	35
clear	17
Turbid	9

dtype: int64

2019-05-01	531
2020-01-11	530

Ordinal Encoding

```
encoding_map = {
    'Clear': 0,
    'clear': 0,
    'Slightly Turbid': 1,
    'Turbid': 2
}

df["Aspect(Urine Physical Examination)"] = df["Aspect(Urine Physical Examinatic
df.rename(columns={"Aspect(Urine Physical Examination)": "Aspect(Urine Physical
```

2018-07-14	528

```
df["Aspect(Urine Physical Examination) Ordinal Encoding"].value_counts()
```

Aspect(Urine Physical Examination) Ordinal Encoding	count
0.0	162
1.0	35
2.0	9

dtype: int64

```
df[["Eosinophils absolute count"]].value_counts()
```

Eosinophils absolute count	count

0.150	12
0.160	12
0.200	12
0.130	11
0.110	11
0.180	9
0.060	9
0.120	8
0.220	8
0.100	8
0.190	7
0.300	7
0.090	6
0.140	6
0.050	6
0.270	6
0.250	6
0.210	5
0.080	5
0.070	5
0.170	5
0.030	4
0.240	4
0.320	4
0.260	4
0.230	3
0.520	3
0.370	3
0.390	3
0.360	2

```
df["Vitamin D (25 OH-Vit D -Total)"].value_counts()
```

Vitamin D (25 OH-Vit D -Total)	count
11.2	2934
10.5	2918
10.6	2908
11.4	2863
11.3	2860
11.7	2848
10.2	2846
11.1	2817
11.6	2793
10.9	2785
12.5	2781
11.9	2774
11.8	2771
12.7	2770
10.4	2764
11.5	2762
12.9	2757
12.6	2747
10.3	2737
12.1	2731
12.2	2730
12.3	2716
10.8	2714
13.2	2700
12.4	2689

10.7	2676
13.1	2655
13.5	2644
13.3	2643
13.8	2631

```
df.dtypes
```

	0
RESEARCH_ID	object
SAMPLE_ID	object
COLLECTYEAR	int64
REGN_DATE	datetime64[ns]
GENDER_NAME	object
AGE_YEARS	Int64
AGE_DAYS	Int64
AGE_MONTHS	Int64
CITY_NAME	category
HEIGHT	int64
WEIGHT	float64
BMI	float64
Thyroid Stimulating Hormone (TSH)	float64
Uric Acid in Serum	float64
Alanine Aminotransferase (ALT)	float64
Ferritin In Serum	float64
Blood Urea Nitrogen (BUN)	float64
Lymphocytes absolute count	float64
R. B. Cs / HPFs	float64
Aspect(Urine Physical Examination) Ordinal Encoding	float64
Eosinophils absolute count	float64
Vitamin D (25 OH-Vit D -Total)	object

C-Reactive Protein (CRP) quantitative	object
Transferrin	object
Red cell count	object
Basophils absolute count	float64
Crystals(Urine Microscopic Examination :)	object
Protein(Urine Physical Examination)	object
Colour(Urine Physical Examination)	object
Nitrite	object
LDL Cholesterol	object
LDL/HDL	float64

```
df["Vitamin D (25 OH-Vit D -Total)"].value_counts()
```

Vitamin D (25 OH-Vit D -Total)	count
11.2	2934
10.5	2918
10.6	2908
11.4	2863
11.3	2860
11.7	2848
10.2	2846
11.1	2817
11.6	2793
10.9	2785
12.5	2781
11.9	2774
11.8	2771
12.7	2770
10.4	2764
11.5	2762
12.9	2757

12.6	2747
10.3	2737
12.1	2731
12.2	2730
12.3	2716
10.8	2714
13.2	2700
12.4	2689
10.7	2676
13.1	2655
13.5	2644
13.3	2643
13.8	2631
13.4	2619
12.8	2611
10.1	2605
13.6	2595
14.2	2578
13.7	2575
14.3	2561
13.9	2521
14.6	2520
14.1	2519

This column contains numerical values. When it is converted from object to float, the percentage of missing values increases, and this should not happen.

The solution is to determine the values that became nan after processing and identify the pattern in them

Extra spaces before or after numbers.

Unexpected symbols such as (?, #, ,, %, >, <, *).

Values written in words (such as "twenty" instead of "20").

Values that contain units of measurement (such as "10 ng/ml" instead of "10").

```
df["Vitamin D (25 OH-Vit D -Total)"].value_counts()
```

Vitamin D (25 OH-Vit D -Total)	count
11.2	2934
10.5	2918

10.6	2908
11.4	2863
11.3	2860
11.7	2848
10.2	2846
11.1	2817
11.6	2793
10.9	2785
12.5	2781
11.9	2774
11.8	2771
12.7	2770
10.4	2764
11.5	2762
12.9	2757
12.6	2747
10.3	2737
12.1	2731
12.2	2730
12.3	2716
10.8	2714
13.2	2700
12.4	2689
10.7	2676
13.1	2655
13.5	2644
13.3	2643
13.8	2631
13.4	2619
12.8	2611

```
column_name = 'Vitamin D (25 OH-Vit D -Total)'
```

```
column_name = 'Vitamin D (25 OH-Vit D -Total)'
df[column_name] = df[column_name].astype(str)
unique_chars = set("".join(df[column_name]))
non_numeric_chars = [char for char in unique_chars if not char.isdigit() and ch
```

```
non_numeric_chars
```

```
['>', ')', 'H', '(', '<', ':', '*', '-', 'n', 'a']
```

```
df["Vitamin D (25 OH-Vit D -Total)"].unique()
```

```
array(['17.5', '10.8', '26.8', ..., '36.96', '103.1*', '30.33'],
      dtype=object)
```

قيم تحتوي على رموز مثل "<4.00", ">100", "(3.8)*", "4.0*", "*4.00", وغيرها. قيم مكررة أو غير مفهومة مثل "21.221.2", "17.217.2", "15.415.4", "8.388.38". قيم تحتوي على وحدات قياس غير واضحة أو قد تشير إلى خطأ في الإدخال. بعض القيم المشبوهة قد تكون مفصولة بمسافات غير ضرورية أو تحتوي على رموز إضافية

13.6	2595
14.6	2520
14.1	2519
14.0	2448
14.8	2446
14.7	2442
11.0	2434
14.9	2430
15.1	2414
12.0	2406
15.3	2406
15.2	2384
15.8	2363
15.4	2362
15.5	2357
15.6	2356
15.7	2333
10.0	2332
16.4	2308
16.2	2303
15.9	2297
16.6	2291
13.0	2283
16.1	2258

```
column_name = 'Vitamin D (25 OH-Vit D -Total)'

def clean_value(value):
    if pd.isna(value) or value == "":
        return pd.NA

    value = str(value).strip()
    value = value.replace(",", ".")


    if value.startswith("<"):
        numeric_value = re.sub(r'[^0-9.]', '', value)
        return float(numeric_value.split('.')[0]) - 1 if numeric_value else pd.


    if value.startswith(">"):
        numeric_value = re.sub(r'[^0-9.]', '', value)
        return float(numeric_value.split('.')[0]) - 1 if numeric_value else pd.


    value = re.sub(r'[^0-9.]', '', value)
    value = value.split('.')[0]

    return float(value) if value else pd.NA


df[column_name] = df[column_name].apply(clean_value)
```

```
df["Vitamin D (25 OH-Vit D -Total)"].dtypes
```

```
dtype('O')
```

```
df["Vitamin D (25 OH-Vit D -Total)"] = pd.to_numeric(df["Vitamin D (25 OH-Vit D
```

```
df["Vitamin D (25 OH-Vit D -Total)"] .info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Vitamin D (25 OH-Vit D -Total)
Non-Null Count   Dtype
--------------   -----
568468 non-null  float64
dtypes: float64(1)
memory usage: 46.0 MB
```

```
df.info()
```

17.6	2009
17.9	1983
18.8	1943
19.2	1926
18.7	1916
18.2	1906
19.8	1896

```
<class 'pandas.core.frame.DataFrame'>
Index: 900000 entries, 15940 to 679524
Data columns (total 100 columns):
 #   Column                                                Non-Null Count
---  ------                                                --------------
 0   RESEARCH_ID                                           900000 non-null
 1   SAMPLE_ID                                             900000 non-null
 2   COLLECTYEAR                                           900000 non-null
 3   REGN_DATE                                             900000 non-null
 4   GENDER_NAME                                           900000 non-null
 5   AGE_YEARS                                             838038 non-null
 6   AGE_DAYS                                              900000 non-null
 7   AGE_MONTHS                                            900000 non-null
 8   CITY_NAME                                             900000 non-null
 9   HEIGHT                                                900000 non-null
 10  WEIGHT                                                900000 non-null
 11  BMI                                                   900000 non-null
 12  Thyroid Stimulating Hormone (TSH)                     544701 non-null
 13  Uric Acid in Serum                                    548513 non-null
 14  Alanine Aminotransferase (ALT)                        579129 non-null
 15  Ferritin In Serum                                     150771 non-null
 16  Blood Urea Nitrogen (BUN)                             361245 non-null
 17  Lymphocytes absolute count                            216 non-null
 18  R. B. Cs /HPFs                                        900000 non-null
 19  Aspect(Urine Physical Examination) Ordinal Encoding   206 non-null
 20  Eosinophils absolute count                            216 non-null
 21  Vitamin D (25 OH-Vit D -Total)                        568468 non-null
 22  C-Reactive Protein (CRP) quantitative                 11311 non-null
 23  Transferrin                                           1719 non-null
 24  Red cell count                                        216 non-null
 25  Basophils absolute count                              216 non-null
 26  Crystals(Urine Microscopic Examination :)             206 non-null
 27  Protein(Urine Physical Examination)                   206 non-null
 28  Colour(Urine Physical Examination)                    206 non-null
 29  Nitrite                                               206 non-null
 30  LDL Cholesterol                                       306442 non-null
 31  LDL / HDL                                             243 non-null
 32  24 Hour Urine Volume (263)                            7 non-null
 33  Hemoglobin                                            219 non-null
 34  Total Leucocytic Count                                216 non-null
 35  Hematocrit                                            219 non-null
 36  MCV                                                   216 non-null
 37  Glucose(Urine Physical Examination)                   206 non-null
 38  Urea in Serum                                         212283 non-null
 39  Prostatic Specific Antigen (PSA) Total                119269 non-null
 40  Testosterone (Total)                                  123295 non-null
 41  Alkaline Phosphatase                                  288565 non-null
 42  Total Protein in Serum                                281111 non-null
 43  Estimated Glomerular Filtration Rate(eGFR)            298895 non-null
 44  Anti CCP Abs                                          6173 non-null
 45   BUN/Creatinine Ratio                                 275271 non-null
 46  Blood pressure                                        294405 non-null
```

```
47  Ketones                                  206 non-null
48  MCHC                                     216 non-null
49  pH(Urine Physical Examination)           206 non-null
50  Amorphous Sediments                      206 non-null
51  Blood and Haemoglobin                    206 non-null
52  Epithelial Cells / HPF                   206 non-null
```

```
column_to_save = df[["C-Reactive Protein (CRP) quantitative"]]


column_to_save.to_csv('column_data.csv', index=False, encoding='utf-8')
```

```
df["BDL"] = df["C-Reactive Protein (CRP) quantitative"].apply(lambda x: 1 if is

df["C-Reactive Protein (CRP) quantitative"] = df["C-Reactive Protein (CRP) quar

df["C-Reactive Protein (CRP) quantitative"] = pd.to_numeric(df["C-Reactive Prot
```

```
df["C-Reactive Protein (CRP) quantitative"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679624
Series name: C-Reactive Protein (CRP) quantitative
Non-Null Count  Dtype
--------------  -----
11077 non-null  float64
dtypes: float64(1)
memory usage: 46.0 MB
```

22.5	1603
22.0	1602
22.9	1601
22.6	1582
23.4	1467
23.7	1466
23.5	1465
21.0	1447
23.9	1436
24.3	1433
22.0	1416
24.2	1414
24.1	1409
24.6	1403
24.5	1354
24.4	1349
24.8	1338
24.0	1335
24.7	1323
9.8	1322
25.2	1317

So that I don't lose the information that it is under the limit, and this is a value that cannot be read to determine it accurately, but I cannot transform the column in this way. I create a column that determines that it is under the limit, and the main column I put in the average value that is under 0.

How do you choose an alternative value?

Use 0.05 mg/L → (the midpoint of 0 and 0.1) and it is considered a reasonable estimate when no additional data are available.

Use 0.054 mg/L → (the average of the actual values below 0.1 in the sample, which is more accurate).

Use 0.1 mg/L → (less accurate because it gives an upper limit rather than an estimate of the actual value).

```
df["Transferrin"].isna().sum()
```

```
898281
```

```
df["Transferrin"] = df["Transferrin"].astype(str)
df["Transferrin"] = df["Transferrin"].str.replace(r"[^\d.]", "", regex=True)
df["Transferrin"] = pd.to_numeric(df["Transferrin"], errors='coerce')
```

```
df["Transferrin"].isnull().sum()
```

```
898281
```

```
df["Transferrin"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Transferrin
Non-Null Count  Dtype
--------------  -----
1719 non-null   float64
dtypes: float64(1)
memory usage: 46.0 MB
```

In this column, the missing values are coded as Absent. I will code them as zero so that they do not take on more importance when multiplying them by w.

Absent: 899,992 times (due to a large number of missing values)

Few Calcium Oxalate Crystals: 2 times

Uric Acid Crystals: 2 times

Calcium Oxalate Crystals: 1 time

Uric Acid Crystals (+): 1 time

Uric Acid Crystals (some): 1 time

Uric Acid Crystals (few): 1 time

```
df["Crystals(Urine Microscopic Examination :)"] = df["Crystals(Urine Microscopi
```

```
column_name = "Crystals(Urine Microscopic Examination :)"

df[column_name] = df[column_name].replace({
    'Uric Acid Crystal(few)': 'Uric Acid Crystal - Few',
    'Uric Acid Crystal Some': 'Uric Acid Crystal - Some',
    'Uric Acid Crystal (+)': 'Uric Acid Crystal - (+)',
    'Few Calcium Oxalate Crystal': 'Calcium Oxalate Crystal - Few'
})

print("Unique values after replacement:", df[column_name].unique())


ordered_categories = [
    "Absent",
    "Uric Acid Crystal - Few",
    "Uric Acid Crystal - Some",
    "Uric Acid Crystal - (+)",
    "Uric Acid Crystal",  # Added missing category
    "Calcium Oxalate Crystal - Few",
    "Calcium Oxalate Crystal"  # Added missing category
]


encoder = OrdinalEncoder(categories=[ordered_categories], dtype=float, handle_u
df["Crystals(Urine Microscopic Examination :)"] = encoder.fit_transform(df[[col
```

```
Unique values after replacement: ['Absent' 'Calcium Oxalate Crystal' 'Uric
 'Uric Acid Crystal - (+)' 'Uric Acid Crystal'
 'Calcium Oxalate Crystal - Few' 'Uric Acid Crystal - Few']
```

```
df["Crystals(Urine Microscopic Examination :)"] .info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Crystals(Urine Microscopic Examination :)
Non-Null Count   Dtype
--------------   -----
900000 non-null  float64
dtypes: float64(1)
memory usage: 13.7 MB
```

27.8	991
28.5	968
8.1	963
8.2	953
28.1	952
28.6	936
7.9	931
28.4	931
28.8	930
29.3	927
<4.00	923
7.8	921

```
df["Crystals(Urine Microscopic Examination :)"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Crystals(Urine Microscopic Examination :)
Non-Null Count   Dtype
--------------   -----
900000 non-null  float64
dtypes: float64(1)
memory usage: 46.0 MB
```

Protein(Urine Physical Examination) "Aspect(Urine Physical Examination)"

Feature	Aspect(Urine Physical Examination)	Protein(Urine Physical Examination)
Column Name	Aspect(Urine Physical Examination)	Protein(Urine Physical Examination)
Description	Describes the physical appearance of urine, which can indicate infections, kidney problems, or other health conditions.	Measures the presence of protein in urine, which is an important marker for kidney function and possible renal diseases.
Possible Values	"Clear", "Slightly Turbid", "Turbid"	"Absent", "Present (+)", "Present (++)", "Present (+++)", "Trace"
Most Frequent Value	Clear	Absent
Clinical Significance	Turbidity in urine can indicate the presence of bacteria, white blood cells, or protein, suggesting infections or kidney diseases.	Proteinuria (presence of protein in urine) can indicate kidney dysfunction, diabetes, high blood pressure, or other systemic diseases.
Potential Causes of Abnormal Values	Urinary tract infections (UTIs), kidney disease, excessive white blood cells or protein, or dehydration.	Diabetes, hypertension, kidney inflammation (glomerulonephritis), lupus, or chronic kidney disease.

Detailed Comparison of Urine Physical Examination Tests

Feature	Aspect (Urine Physical Examination)	Protein (Urine Physical Examination)
Column Name	Aspect(Urine Physical Examination)	Protein(Urine Physical Examination)
Description	Describes the physical appearance of urine, which can indicate infections, kidney problems, or other health conditions.	Measures the presence of protein in urine, an important marker for kidney function and possible renal diseases.
Possible Values	"Clear", "Slightly Turbid", "Turbid"	"Absent", "Present (+)", "Present (++)", "Present (+++)", "Trace"
Most Frequent Value	Clear	Absent
Clinical Significance	Turbidity in urine can indicate the presence of bacteria, white blood cells, or protein, suggesting infections or kidney diseases.	Proteinuria (presence of protein in urine) can indicate kidney dysfunction, diabetes, high blood pressure, or other systemic diseases.
Potential Causes of Abnormal Values	Urinary tract infections (UTIs), kidney disease, excessive white blood cells or protein, or dehydration.	Diabetes, hypertension, kidney inflammation (glomerulonephritis), lupus, or chronic kidney disease.

```
protein_mapping = {
    "Absent": 0,
    "Trace": 0.5,
    "Present (+)": 1,
    "Present (++)": 2,
    "Present (+++)": 3,
    "Present(+)": 1,
    "Present(++)": 2
}


df["Protein(Urine Physical Examination)"] = df["Protein(Urine Physical Examinat
```

Absent	0.0
Trace	0.5
Present (+)	1.0
Present (++)	2.0
Present (+++)	3.0

```
df["Protein(Urine Physical Examination)"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Protein(Urine Physical Examination)
Non-Null Count  Dtype
--------------  -----
206 non-null    float64
dtypes: float64(1)
memory usage: 46.0 MB
```

33.1	636
32.6	636
31.0	634
6.8	616
6.9	607
6.7	593
33.7	592
33.5	586
33.9	582
32.0	574
6.6	571

```
df["Colour(Urine Physical Examination)"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Colour(Urine Physical Examination)
Non-Null Count  Dtype
--------------  -----
206 non-null    object
dtypes: object(1)
memory usage: 46.0+ MB
```

```
df["Colour(Urine Physical Examination)"].unique()
```

```
array([nan, 'Yellow', 'Pale Yellow', 'yellow', 'Amber Yellow',
       'Yellowish'], dtype=object)
```

Double-click (or enter) to edit

```
df["Colour(Urine Physical Examination)"] = df["Colour(Urine Physical Examinatic
df["Colour(Urine Physical Examination)"].unique()
```

```
array([nan, 'Yellow', 'Pale Yellow', 'Amber Yellow', 'Yellowish'],
      dtype=object)
```

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

```
df.shape
```

```
(900000, 101)
```

```
colour_mapping = {
    "Pale Yellow": 1,
    "Yellow": 2,
    "Yellowish": 3,
    "Amber Yellow": 4
}

df["Colour(Urine Physical Examination)"] = df["Colour(Urine Physical Examinatic
```

```
df["Colour(Urine Physical Examination)"].unique()
```

```
array([nan,  2.,  1.,  4.,  3.])
```

33.4	558
32.9	557
34.5	554
34.4	551
33.6	548
34.9	539
34.1	538
34.2	535
6.5	531
34.8	515
7.0	512
35.2	505
34.6	495
35.8	489
35.6	476
34.0	469
36.3	461

```
df["Colour(Urine Physical Examination)"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Colour(Urine Physical Examination)
Non-Null Count  Dtype
--------------  -----
206 non-null    float64
dtypes: float64(1)
memory usage: 46.0 MB
```

```
df['Nitrite'].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Nitrite
Non-Null Count  Dtype
--------------  -----
206 non-null    object
dtypes: object(1)
memory usage: 46.0+ MB
```

```
df['Nitrite'].unique()
```

```
array([nan, 'Absent', 'Present (+++)', 'pos', 'Present', 'Present (+)'],
      dtype=object)
```

```
nitrite_mapping = {
    "Absent": 0,
    "Present (+)": 1,
    "Present": 2,
    "pos": 2,
    "Present (+++)": 3
}

df["Nitrite"] = df["Nitrite"].map(nitrite_mapping)
```

```
df["Nitrite"].unique()
```

```
array([nan,  0.,  3.,  2.,  1.])
```

```
df['LDL Cholesterol'].unique()
```

```
array([nan, 111.0, '177', '123', 91.0, '173', 114.0, 127.0, '90', 93.0,
       '85', 113.0, 123.0, 180.0, '100', '144', '131', 88.0, 83.0,
'145',
       '153', '133', '109', '122', '165', '127', 109.0, '117', 81.0,
```

```
       '139', '146', '104', '136', '95', 182.0, '63', '111', '116',
'149',
       '114', '198', '72', '203', 99.0, '97', '94', '140', '157', '103',
       124.0, '129', '171', '113', '135', '156', '130', 102.0, 173.0,
       '168', '164', '187', 87.0, '134', '125', '98', '101', '110',
'68',
       '154', 137.0, '159', 120.0, '150', '138', 142.0, 105.0, '126',
       '180', '87', '142', '78', '160', 118.0, '188', '80', '61', '66',
       107.0, 135.0, 150.0, 147.0, '205', '218', 130.0, '79', 116.0,
       '204', '119', '121', '83', '81', 76.0, '106', '107', 96.0, 149.0,
       95.0, 100.0, 186.0, 94.0, '93', '124', 110.0, 122.0, 73.0, '222',
       72.0, 80.0, 86.0, 153.0, '183', 112.0, 90.0, '252', '89', 131.0,
       155.0, '115', '167', 132.0, '137', 64.0, 62.0, '181', '92',
'176',
       '190', '182', '141', 139.0, 98.0, '86', '172', 84.0, '213',
133.0,
       238.0, '128', '166', 166.0, '158', '74', 58.0, '120', 161.0,
129.0,
       185.0, '161', 71.0, 67.0, 163.0, 54.0, 82.0, '70', '151', '118',
       '58', '143', 74.0, 159.0, 97.0, '155', '99', '162', '152', 106.0,
       '189', '163', '226', '169', 20.0, 144.0, '102', '82', '132',
50.0,
       '105', '191', 179.0, '64', 121.0, 162.0, 168.0, 203.0, '88',
'91',
       134.0, '73', 126.0, 198.0, '211', 101.0, '71', 140.0, 193.0,
'148',
       '179', 136.0, 57.0, 164.0, '77', '51', '147', 181.0, 175.0, 85.0,
       '193', 115.0, 117.0, '49', '215', 156.0, '84', '108', 63.0,
145.0,
       '96', '240', '201', '217', 143.0, '67', '41', '174', 128.0,
'112',
       191.0, '32', '55', 65.0, 53.0, '178', '230', '208', '175', 165.0,
       47.0, 140.0, 169.0, '170', 125.0, 79.0, 157.0, 151.0, '35', '76',
       '40', '62', 89.0, '224', 104.0, '255', '192', '200', '216',
146.0,
       92.0, 138.0, 75.0, '53', '60', '195', 184.0, 170.0, 69.0, 158.0,
       219.0, '75', 141.0, 152.0, 59.0, 172.0, '229', 46.0, '56', 171.0,
       148.0, 119.0, 66.0, '223', 48.0, 35.0, 77.0, 49.0, '210', 209.0,
       160.0, 194.0, 37.0, '59', '186', '232', 195.0, 167.0, '69', 52.0,
       '52', '206', 103.0, 41.0, 178.0, 239.0, '194', 70.0, 189.0,
108.0,
       177.0, 218.0, '65', 183.0, 188.0, '199', 78.0, 154.0, '196',
213.0,
       206.0, '197', 192.0, 216.0, 40.0, 250.0, 68.0, '184', 176.0,
207.0,
       '47', '29', 60.0, 24.0, '57', '185', 56.0, '39', '267', '48',
51.0,
       '214', '46', 43.0, '54', 61.0, '50', 229.0, '258', 38.0, '43',
       220.0, '202', 230.0, 190.0, 232.0, '238', 223.0, 187.0, '221',
       247.0, '207', 222.0, 201.0, '285', '228', 44.0, '275', 204.0,
'34',
       227.0, 246.0, 208.0, '45', 202.0, '220', '231', 55.0, 233.0,
205.0,
```

36.0	356
38.7	354
13	353
39.3	352
17	347
39.4	347
38.9	347
15	341
5.9	341
39.5	340
38.5	339
38.6	338
5.7	332
39.1	332
39.2	331
41.1	328
39.6	326
40.9	325
39.9	320
39.0	319
40.4	318
39.7	318
39.8	316
40.7	315
40.1	313
40.2	311
16	311
40.3	307
18	303

45.0 244.0 '251' '391*' 308.0 248.0 '219' '212' 197.0

```
df["LDL Cholesterol"] = pd.to_numeric(df["LDL Cholesterol"], errors="coerce").f
```

```
df["LDL Cholesterol"] = df["LDL Cholesterol"].astype(str).str.replace(r'[^\d]',
df["LDL Cholesterol"] = df["LDL Cholesterol"].replace('', np.nan)
df["LDL Cholesterol"].unique()
```

```
array(['0', '111', '177', '123', '91', '173', '114', '127', '90', '93',
       '85', '113', '180', '100', '144', '131', '88', '83', '145', '153',
       '133', '109', '122', '165', '117', '81', '139', '146', '104',
       '136', '95', '182', '63', '116', '149', '198', '72', '203', '99',
       '97', '94', '140', '157', '103', '124', '129', '171', '135', '156',
       '130', '102', '168', '164', '187', '87', '134', '125', '98', '101',
       '110', '68', '154', '137', '159', '120', '150', '138', '142',
       '105', '126', '78', '160', '118', '188', '80', '61', '66', '107',
       '147', '205', '218', '79', '204', '119', '121', '76', '106', '96',
       '186', '73', '222', '86', '183', '112', '252', '89', '155', '115',
       '167', '152', '64', '62', '181', '92', '176', '190', '141', '172',
       '84', '213', '238', '128', '166', '158', '74', '58', '161', '185',
       '71', '67', '163', '54', '82', '70', '151', '143', '162', '152',
       '189', '226', '169', '20', '50', '191', '179', '211', '193', '148',
       '57', '77', '51', '175', '49', '215', '108', '240', '201', '217',
       '41', '174', '32', '55', '65', '53', '178', '230', '208', '47',
       '170', '35', '40', '224', '255', '192', '200', '216', '75', '60',
       '195', '124', '69', '219', '59', '229', '46', '56', '223', '48',
       '210', '209', '194', '37', '232', '52', '206', '239', '199', '196',
       '197', '250', '207', '29', '225', '39', '267', '214', '43', '258',
       '38', '220', '202', '221', '247', '285', '228', '44', '275', '34',
       '227', '246', '45', '231', '253', '251', '248', '212', '264',
       '278', '22', '287', '26', '27', '21', '3', '263', '225', '241',
       '242', '28', '235', '237', '245', '271', '290', '253', '25', '33',
       '236', '249', '298', '387', '36', '254', '259', '260', '243', '30',
       '31', '262', '297', '284', '268', '13', '266', '261', '370', '276',
       '257', '234', '244', '345', '327', '42', '291', '288', '11', '309',
       '269', '256', '293', '283', '12', '273', '270', '265', '280',
       '279', '277', '7', '296', '383', '16', '391', '15', '286', '313',
       '282', '23', '319', '292', '332', '321', '300', '281', '274',
       '272', '317', '14', '342', '19', '315', '295', '360', '17', '289',
       '4', '308', '323', '494', '344', '294', '305', '299', '484', '318',
       '337', '9223372036854775808', '9', '10', '334', '5', '311', '18',
       '8', '333', '329', '317', '374', '351', '349', '346', '331', '301',
       '340', '307', '534', '383', '303', '302', '314'], dtype=object)
```

Double-click (or enter) to edit

```
for value in df["LDL Cholesterol"].unique():
    if isinstance(value, (int, np.integer)):
      if value > 999:
        print(value)
    elif isinstance(value, str):
      try:
        num = int(value)
        if num > 999:
          print(num)
      except ValueError:
        pass
```

```
9223372036854775808
```

42.0	239
43.3	237

```
def first_three_digits(value):
    if isinstance(value, (int, np.integer)):
        return int(str(value)[:3]) if value > 99 else value
    elif isinstance(value, str):
        try:
            num = int(value)
            return int(str(num)[:3]) if num > 99 else num
        except ValueError:
            return value
    else:
        return value

df["LDL Cholesterol"] = df["LDL Cholesterol"].apply(first_three_digits)
```

25	219
43.9	219
45.4	214
45.7	214
46.5	213
43.5	213
44.5	212
45.6	212
44.7	211
44.9	207
45.1	204
43.2	204

```
df["LDL Cholesterol"].unique()
```

```
array([  0, 111, 177, 123,  91, 173, 114, 127,  90,  93,  85, 113, 180,
       100, 144, 131,  88,  83, 145, 153, 133, 109, 122, 165, 117,  81,
       139, 146, 104, 136,  95, 182,  63, 116, 149, 198,  72, 203,  99,
        97,  94, 140, 157, 103, 124, 129, 171, 135, 156, 130, 102, 168,
       164, 187,  87, 134, 125,  98, 101, 110,  68, 154, 137, 159, 120,
       150, 138, 142, 105, 126,  78, 160, 118, 188,  80,  61,  66, 107,
       147, 205, 218,  79, 204, 119, 121,  76, 106,  96, 186,  73, 222,
        86, 183, 112, 252,  89, 155, 115, 167, 132,  64,  62, 181,  92,
       176, 196, 141, 172,  84, 213, 238, 128, 166, 158,  74,  58, 161,
       185,  71,  67, 163,  54,  82,  70, 151, 143, 162, 152, 189, 226,
       169,  20,  50, 191, 179, 211, 193, 148,  57,  77,  51, 175,  49,
       215, 108, 240, 201, 217,  41, 174,  32,  55,  65,  53, 178, 230,
       208,  47, 170,  35,  40, 224, 255, 192, 200, 216,  75,  60, 195,
       184,  69, 219,  59, 229,  46, 56, 223,  48, 210, 209, 194,  37,
       232,  52, 206, 239, 199, 196, 197, 250, 207,  29,  24,  39, 267,
       214,  43, 258,  38, 220, 202, 221, 247, 285, 228,  44, 275,  34,
       227, 246,  45, 231, 233, 251, 248, 212, 264, 278,  22, 287,  26,
        27,  23,   3, 263, 225, 241, 242,  28, 235, 237, 245, 271, 290,
       253,  25,  33, 236, 249, 298, 387,  36, 254, 259, 260, 243,  30,
        31, 262, 297, 284, 268,  13, 266, 261, 370, 276, 257, 234, 244,
       345, 327,  42, 291, 288,  11, 309, 269, 256, 293, 283,  12, 273,
       270, 265, 280, 279, 277,   7, 296, 382,  16, 391,  15, 286, 313,
       282,  23, 319, 292, 332, 321, 300, 281, 274, 272, 377,  14, 342,
        19, 315, 295, 360,  17, 289,   4, 308, 323, 494, 344, 294, 305,
       299, 484, 318, 337, 922,   9, 10, 334,   5, 311,  18,   8, 333,
       329, 317, 374, 351, 349, 346, 331, 301, 340, 307, 534, 383, 303,
       302, 314])
```

```
df[""'LDL / HDL'""].unique()
```

```
array([nan, 4.4, 2.7, 2. , 2.3, 3.3, 3.2, 3. , 4.2, 2.5, 2.1, 1.7, 4. ,
       1.8, 2.6, 3.4, 1.6, 2.9, 2.8, 4.3, 1.4, 3.8, 5.1, 3.9, 1.5, 1. ,
       1.3, 1.2, 3.5, 3.7, 2.2, 3.1, 1.9, 5.8, 5. , 4.7, 3.6, 2.4, 5.2,
       0.8, 4.6, 4.5, 4.9, 5.3, 6. , 0.5, 6.5, 6.4, 1.1])
```

```
df[""'LDL / HDL'""].value_counts()
```

LDL / HDL	count
3.0	16
2.7	12
3.3	12
2.0	12
2.5	10

2.3	9
2.4	9
1.7	8
2.6	8
1.8	8
2.9	8
4.0	8
2.8	8
1.9	7
2.1	7
1.6	6
3.6	6
3.7	6
3.2	5
4.4	5
1.4	5
3.8	5
3.1	5
3.5	5
3.4	4
4.2	4
1.3	4
1.2	4
5.1	4
3.9	3
1.5	3
4.3	3
2.2	3
1.0	3

```
df["24 Hour Urine Volume (263)"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: 24 Hour Urine Volume (263)
Non-Null Count  Dtype
--------------  -----
7 non-null      float64
dtypes: float64(1)
memory usage: 46.0 MB
```

```
df['Hemoglobin'].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Hemoglobin
Non-Null Count  Dtype
--------------  -----
219 non-null    float64
dtypes: float64(1)
memory usage: 46.0 MB
```

```
df.dtypes
```

	0
RESEARCH_ID	object
SAMPLE_ID	object
COLLECTYEAR	int64
REGN_DATE	datetime64[ns]
GENDER_NAME	object
AGE_YEARS	Int64
AGE_DAYS	Int64
AGE_MONTHS	Int64
CITY_NAME	category
HEIGHT	int64
WEIGHT	float64
BMI	float64
Thyroid Stimulating Hormone (TSH)	float64
Uric Acid in Serum	float64

Alanine Aminotransferase (ALT)	float64
Ferritin In Serum	float64
Blood Urea Nitrogen (BUN)	float64
Lymphocytes absolute count	float64
R. B. Cs / HPFs	float64
Aspect(Urine Physical Examination) Ordinal Encoding	float64
Eosinophils absolute count	float64
Vitamin D (25 OH-Vit D -Total)	float64
C-Reactive Protein (CRP) quantitative	float64
Transferrin	float64
Red cell count	object
Basophils absolute count	float64
Crystals(Urine Microscopic Examination :)	float64
Protein(Urine Physical Examination)	float64
Colour(Urine Physical Examination)	float64
Nitrite	float64
LDL Cholesterol	int64
LDL/HDL	float64

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 900000 entries, 13940 to 679524
Columns: 101 entries, RESEARCH_ID to BDL
dtypes: Int64(3), category(1), datetime64[ns](1), float64(32), int64(6), ob
memory usage: 729.2+ MB
```

Hemoglobin	float64
Total Leucocytic Count	float64
Hematocrit	float64
MCV	float64
Glucose(Urine Physical Examination)	object
Urea in Serum	object
Prostatic Specific Antigen (PSA) Total	object
Testosterone (Total)	object
Alkaline Phosphatase	object
Total Protein in Serum	object
Estimated Glomerular Filtration Rate (eGFR)	object

```
df["Glucose(Urine Physical Examination)"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Glucose(Urine Physical Examination)
Non-Null Count  Dtype
--------------  -----
206 non-null    object
dtypes: object(1)
memory usage: 46.0+ MB
```

```
df["Glucose(Urine Physical Examination)"].unique()
```

```
array([nan, 'Present (++)', 'Absent', 'Present (+++)', 'Present (+)'],
      dtype=object)
```

```
glucose_mapping = {
    "Absent": 0,
    "Present (+)": 1,
    "Present (++)": 2,
    "Present (+++)": 3
    }
df["Glucose(Urine Physical Examination)"] = df["Glucose(Urine Physical Examinat
df["Glucose(Urine Physical Examination)"].unique()
```

```
array([nan,  2.,  0.,  3.,  1.])
```

```
df["Glucose(Urine Physical Examination)"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Glucose(Urine Physical Examination)
Non-Null Count  Dtype
--------------  -----
206 non-null    float64
dtypes: float64(1)
memory usage: 46.0 MB
```

BUN/Creatinine Ratio	object
Blood pressure	object
Ketones	object
MCHC	float64
pH(Urine Physical Examination)	float64
Blood and Haemoglobin	object
Epithelial Cells / HPF	object
R.B.Cs / HPF	object
Hb A1c %	object
Mean of blood glucose	object
Microalbuminuria (24 h urine)	object
Bilirubin (Total)	object
Florescence Pattern	object
Lead in blood	object
Monocytes absolute count	float64
Consistancy	object
Neutrophils absolute count	float64
Specific Gravity	object
W. B. Cs / HPF	object

```
df[["Urea in Serum"]].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 900000 entries, 15940 to 679524
Data columns (total 1 columns):
 #   Column          Non-Null Count   Dtype 
---  ------          --------------   ----- 
 0   Urea in Serum   211282 non-null  object
dtypes: object(1)
memory usage: 4810.1+ MB
```

```
df[["Urea in Serum"]].value_counts()
```

Urea in Serum	count
21	12671
19	11938
24	11857
26	10527
17	9535
28	8999
30	7148
22	7084
23	6853
20	6838
15	6759
25	6403
18	6325
16	5811
27	5394
32	5387
29	4379
34	4011
31	3627

Calcium in Serum (Total)	object
Free T4	object
Potassium (K) in Serum	object
Albumin in Serum	object
Iron (Fe) in Serum	object
CRP H.S	object

13	3613
14	2912
36	2845
33	2596
21.0	2289
19.0	2140
24.0	2035
35	1990
26.0	1935
39	1896
17.0	1828

```
def clean_and_convert(value):
    if isinstance(value, str):
        # إزالة الرموز غير الرقمية مثل *, >, <, ()
        value = re.sub(r"[><*(),]", "", value).strip()

        # التعامل مع القيم المكررة الخاطئة (مثل 2828، 4949 #
        if len(value) > 2 and value[:2] == value[2:]:
            value = value[:2]  # الاحتفاظ بالجزء الأول فقط

        # محاولة تحويل القيم إلى رقمية
        try:
            return float(value)
        except ValueError:
            return None

    return value

df["Urea in Serum"] = df["Urea in Serum"].apply(clean_and_convert)
df["Urea in Serum"].unique()
```

38	1206
27.0	1077
32.0	1004
43	966
40	877
29.0	870

```
array([13.  ,  nan,  27.  ,  30.  ,  26.  ,  24.  ,  23.  ,  28.  ,
        25.  ,  21.  ,  34.  ,  22.  ,  39.  ,  20.  ,  19.  ,  14.  ,
        46.  ,  29.  ,  15.  ,  17.  ,  16.  ,  18.  ,  13.  ,  12.  ,
        54.  ,  11.  ,  31.  ,  32.  ,  53.  ,  43.  ,  42.  ,  56.  ,
        36.  ,  49.  ,  35.  , 154.  ,  38.  ,  40.  ,  62.  ,  37.  ,
        10.  ,  75.  , 160.  ,  81.  , 167.  , 143.  ,  89.  ,  70.  ,
        41.  ,  52.  ,  51.  , 124.  ,  57.  , 112.  ,   8.  ,  66.  ,
       156.  ,  97.  , 147.  , 184.  ,  44.  ,   9.  ,  47.  ,  82.  ,
        61.  , 190.  ,  86.  ,  48.  ,  58.  ,  55.  ,  78.  ,  79.  ,
        94.  ,  72.  ,  67.  ,  63.  , 100.  ,  73.  , 102.  , 105.  ,
        76.  , 187.  ,   7.  ,  95.  , 131.  ,  64.  ,  50.  ,  98.  ,
       140.  ,  83.  ,  45.  ,  87.  ,  90.  , 205.  , 126.  ,  99.  ,
        65.  ,  59.  ,  80.  ,  91.  , 101.  ,  74.  , 139.  , 203.  ,
       130.  , 128.  ,  69.  , 116.  , 182.  ,  60.  , 103.  , 194.  ,
       268.  , 159.  , 161.  , 134.  , 109.  , 151.  , 145.  , 140.  ,
       107.  , 108.  ,   4.55, 220.  , 137.  , 152.  , 132.  , 197.  ,
       216.  , 224.  , 133.  ,  93.  , 104.  ,  68.  ,  77.  ,   6.  ,
       163.  , 180.  ,  71.  , 243.  ,  88.  , 138.  , 135.  , 115.  ,
       166.5 , 118.  , 199.  , 256.8 , 250.  , 289.  , 211.  ,  84.  ,
       155.  , 198.  ,  96.  ,  92.  , 230.  , 122.  , 123.  , 148.  ,
       249.  , 158.  , 176.  , 171.  , 288.55, 168.  , 227.  , 106.  ,
       141.  , 165.  , 210.  , 202.  , 172.  , 192.  , 183.  , 146.  ,
       120.  , 235.  , 231.  , 200.  , 111.  ,  85.  , 129.  , 179.  ,
         7.9 , 164.  , 142.  , 125.  , 284.  , 213.  , 272.  , 255.  ,
       114.  , 259.  , 127.  , 181.  , 117.  , 110.  , 233.  , 157.  ,
       252.  , 242.  , 170.  , 319.  , 150.  , 162.  , 219.  , 196.  ,
       113.  , 245.  , 144.  , 173.  , 169.  , 177.  , 237.  , 186.  ,
       153.  , 121.  , 253.  , 223.  , 175.  , 248.  , 178.  , 119.  ,
       400.  ,   4.  , 225.  , 313.  , 136.  , 188.  , 347.  , 221.  ,
       189.  , 214.  , 464.  , 193.  , 149.  , 195.  , 234.  ,   5.  ,
       353.  , 333.  , 287.  , 174.  , 208.  , 204.  , 254.  , 222.  ,
       202.  , 257.  , 166.  , 185.  , 238.  , 274.  , 258.  , 293.  ,
       266.  , 226.  , 217.  , 191.  ,   4.1 , 215.  , 362.  , 244.  ,
       386.  ,   2.  , 270.  , 239.  , 263.  , 265.  , 206.  , 207.  ,
       229.  , 337.  , 256.  ,   1.  , 218.  , 228.  , 261.  , 411.  ,
       267.  , 218.14, 308.  , 288.  , 212.  , 209.  , 240.24, 338.  ,
       364.3 , 349.9 , 360.9 , 372.  , 351.  , 396.  , 304.  , 377.  ,
       201.  , 265.36, 285.  , 241.  , 281.  , 393.4 , 378.  , 260.  ,
       276.  , 271.  , 280.  , 315.  , 294.  , 307.  ,   6.74, 345.  ,
       251.  , 236.  , 309.  , 492.8 , 277.  ])
```

```
df["Prostatic Specific Antigen (PSA) Total"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Prostatic Specific Antigen (PSA) Total
Non-Null Count   Dtype
--------------   -----
119269 non-null  object
dtypes: object(1)
memory usage: 46.0+8MB
```

```
df["Prostatic Specific Antigen (PSA) Total"].unique()
```

```
array([nan, '1.243', '1.371', ..., '*515.774', '40.065', 3.825],
      dtype=object)
```

```
df["Prostatic Specific Antigen (PSA) Total"].value_counts()
```

Prostatic Specific Antigen (PSA) Total	count
0.41	294
0.43	293
0.42	290
0.45	282
0.53	280
0.50	280
0.48	279
0.36	273
0.46	269
0.37	268
0.54	266
0.49	260
0.40	260
0.57	259
0.56	257
0.58	255
0.52	254
0.47	252
0.39	251
0.44	250
0.35	246
0.34	246
0.59	245

0.51	244
0.62	237
0.55	233
0.38	232
0.63	231
0.60	229
0.33	224
0.70	195
0.26	185
0.77	182
0.27	180
0.72	180
0.76	174
0.28	171
0.78	168

```
def clean_and_convert_decimal(value):
    if isinstance(value, str):
        # إزالة الرموز غير الرقمية مثل *, <, >, () مع الاحتفاظ بالمعلومات الأساسية
        value = re.sub(r"[><*(),]", "", value).strip()

        # التحقق من القيم المكررة الخاطئة مثل "2828" أو "4949" وتصحيحها
        if len(value) > 2 and value[:2] == value[2:]:
            value = value[:2]  # الاحتفاظ بالجزء الأول فقط

        # محاولة تحويل القيم إلى عدد عشري مع الحفاظ على الدقة
        try:
            return float(value)
        except ValueError:
            return np.nan

    return value


df["Prostatic Specific Antigen (PSA) Total"] = df["Prostatic Specific Antigen (
df["Prostatic Specific Antigen (PSA) Total"].unique()
```

```
array([    nan,   1.243,   1.371, ..., 453.237, 515.774,  40.065])
```

```
df["Prostatic Specific Antigen (PSA) Total"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Prostatic Specific Antigen (PSA) Total
Non-Null Count   Dtype
--------------   -----
119237 non-null  float64
dtypes: float64(1)
memory usage: 46.0 MB
```

```
df["Testosterone (Total)"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Testosterone (Total)
Non-Null Count   Dtype
--------------   -----
123295 non-null  object
dtypes: object(1)
memory usage: 46.0+ MB
```

```
df["Testosterone (Total)"].unique()
```

```
array([nan, '4.89', '3.49', ..., '0.042', '19.09', '12.390'],
dtype=object)
```

```
df["Testosterone (Total)"].value_counts()
```

Testosterone (Total)	count
4.42	281
4.37	278
4.22	273
4.23	272
4.31	261
4.48	261
5.02	260
4.28	259
4.34	259
4.02	258
4.52	257
4.77	257
4.17	255
4.01	255
3.77	254
4.32	253

4.26	253
4.53	252
4.36	252
4.25	252
4.56	251
4.83	251
4.62	251
4.27	250
4.49	249
3.97	249
4.45	248
4.11	248
4.43	247
3.95	246

Start coding or generate with AI.

```
df.shape
```

```
(900000, 101)
```

4.19	244
3.81	243
3.93	243
4.74	243
4.58	243
3.79	242
4.05	242
4.93	241
3.94	241
3.72	239
4.41	239
4.91	237
4.97	237

```
def clean_and_convert_decimal(value):
    if isinstance(value, str):
        # إزالة الرموز غير الرقمية مثل *, <, >, () مع الاحتفاظ بالمعلومات الأساسية
        value = re.sub(r"[><*(),]", "", value).strip()

        # التحقق من القيم المكررة الخاطئة مثل "2828" أو "4949" وتصحيحها
        if len(value) > 2 and value[:2] == value[2:]:
            value = value[:2]  # الاحتفاظ بالجزء الأول فقط

        # محاولة تحويل القيم إلى عدد عشري مع الحفاظ على الدقة
        try:
            return float(value)
        except ValueError:
            return np.nan

    return value

df["Testosterone (Total)"] = df["Testosterone (Total)"].apply(clean_and_convert
df["Testosterone (Total)"].unique()
```

```
array([   nan,  4.89 ,  3.49 , ...,  7.865,  0.042, 19.09 ])
```

```
df.shape
```

```
(900000, 101)
```

```
df["Testosterone (Total)"].unique()
```

```
array([   nan,  4.89 ,  3.49 , ...,  7.865,  0.042, 19.09 ])
```

```
df["Florescence Pattern"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Florescence Pattern
Non-Null Count  Dtype
--------------  -----
2337 non-null   object
dtypes: object(1)
memory usage: 46.0+ MB
```

3.76	234
4.63	233
4.13	233
4.84	233
3.55	232
4.07	231
4.78	231
5.26	231
4.86	231
3.62	230
4.38	230
3.96	230
3.91	230
4.51	229

```
df["Florescence Pattern"].unique()
```

```
array([nan, 'Speckeled', '-', 'Speckled', 'Homogenous', 'Nuclear dots',
       'Nucleolar', 'Fine Speckeled', 'Centromere',
       'Speckeled&Nuclear dot pattern', 'Nuclear Membrane', 'Rim',
       'Dense Fine Speckled', '--', 'Fine Speckled',
       'Homogenous &Speckled', 'Homogenous &Centromere',
       'Nuclear Membrane & Homogenous', 'Homogenous &Nucleolar',
       'Homogenous & Nucleolar', 'Speckled & Few Nuclear dot pattern',
       'PCNA ( Pleomorphic speckled nucleoplasmic pattern )',
       'Speckled & homogenous', 'Speckled & Homogenous',
       'Homogenous & Speckled', 'Speckled&Nucleolar  pattern',
       'Speckled& Few Nuclear dot pattern'], dtype=object)
```

```
corrections = {
    "Speckeled": "Speckled",
    "Spekled": "Speckled",
    "Fine Speckeled": "Fine Speckled",
    "Homogenous &Speckled": "Homogenous & Speckled",
    "Homogenous &Nucleolar": "Homogenous & Nucleolar",
    "Speckled&Nucleolar  pattern": "Speckled & Nucleolar",
    "Speckled& Few Nuclear dot pattern": "Speckled & Few Nuclear dot pattern",
}

def clean_florescence(value):
    if pd.isna(value) or value in ['-', '--']:
        return np.nan  # تعويض القيم غير المفهومة بـ NaN

    # تصحيح الإملاء إذا كانت القيمة تحتاج إلى ذلك
    value = corrections.get(value, value)

    # إزالة المسافات الزائدة حول "&"
    value = " & ".join([x.strip() for x in value.split("&")])

    # تحويل الأنماط المركبة إلى قوائم
    patterns = sorted(value.split(" & "))
    return " & ".join(patterns)

df["Florescence Pattern"] = df["Florescence Pattern"].apply(clean_florescence)
```

4.55	229
5.24	229
4.79	228
4.72	228
3.85	228
4.81	228
4.87	228
3.64	228
3.98	223
3.49	223
4.03	223
5.05	223
4.68	223
5.38	223

```
df["Florescence Pattern"].unique()
```

```
array([nan, 'Speckled', 'Homogenous', 'Nuclear dots', 'Nucleolar',
       'Fine Speckled', 'Centromere', 'Nuclear dot pattern & Speckeled',
       'Nuclear Membrane', 'Rim', 'Dense Fine Speckled',
       'Homogenous & Speckled', 'Centromere & Homogenous',
       'Homogenous & Nuclear Membrane', 'Homogenous & Nucleolar',
       'Few Nuclear dot pattern & Speckled',
       'PCNA ( Pleomorphic speckled nucleoplasmic pattern )',
       'Speckled & homogenous', 'Nucleolar & Speckled'], dtype=object)
```

هناك قيم مفقودة في العمود، مما يستدعي معالجة البيانات المفقودة، إما بحذفها أو تعويضها بناءً على استراتيجية مناسبة. قيم غير ذات معنى ("-", "--"): وجود رموز غير واضحة مثل '-' و '--' قد يشير إلى بيانات مفقودة مشفرة بشكل غير قياسي. يفضل استبدال هذه القيم بـ NaN لسهولة المعالجة. الأنماط الرئيسية في البيانات: هناك أنماط فردية مثل 'Nucleolar' 'Speckled' 'Rim' 'Homogenous' 'Centromere' 'Nuclear dots' 'Nuclear Membrane' 'PCNA (Pleomorphic speckled nucleoplasmic pattern)' الأنماط المركبة: هناك قيم مدمجة تعبر عن أكثر من نمط واحد، مثل 'Speckled&Nuclear dot pattern' 'Spekled & homogenous' (كتابة غير متسقة لكلمة "Speckled") 'Homogenous &Speckled' (عدم تناسق في المسافات) 'Nuclear Membrane & Homogenous' 'Speckled & Few Nuclear dot pattern' مشكلات تتعلق بعدم تناسق الكتابة: وجود اختلافات في كتابة نفس المصطلح، مثل 'Speckled' مقابل 'Speckled' 'Spekled' مقابل 'Speckeled' اختلاف ترتيب الأنماط في القيم المركبة، مثل 'Speckled & Homogenous' مقابل 'Homogenous & Speckled' بعض القيم تحتوي على مسافات زائدة مثل 'Homogenous &Nucleolar' (عدم وجود مسافة بعد &)

```
df["Florescence Pattern"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Florescence Pattern
Non-Null Count  Dtype
--------------  -----
1133 non-null   object
dtypes: object(1)
memory usage: 46.0+ MB
```

4.54	223
4.85	223
3.65	222
3.35	222
3.82	222
4.92	222
4.39	222
4.67	222
3.67	221
4.15	221
3.92	221
3.74	221
4.99	220
3.78	220
3.39	220
4.21	220
5.51	219
4.89	218
4.71	218
5.22	218
3.46	218
3.73	218
3.53	218
4.61	218
5.09	216
4.88	216
4.82	216

```
df["Florescence Pattern"].value_counts()
```

Florescence Pattern	count
Speckled	798
Homogenous	156
Nucleolar	114
Centromere	10
Nuclear Membrane	10
Dense Fine Speckled	9
Nuclear dots	8
Rim	6
Homogenous & Speckled	6
Fine Speckled	5
Nuclear dot pattern & Speckeled	2
Homogenous & Nucleolar	2
Few Nuclear dot pattern & Speckled	2
Centromere & Homogenous	1
Homogenous & Nuclear Membrane	1
PCNA (Pleomorphic speckled nucleoplasmic pattern)	1
Spekled & homogenous	1
Nucleolar & Speckled	1

dtype: int64

3.42	209
3.29	209
5.34	209
4.70	208
5.14	208
3.44	208
4.96	208

```
df["Florescence Pattern"].fillna("Unknown", inplace=True)
encoder = LabelEncoder()
df["Florescence Pattern "] = encoder.fit_transform(df["Florescence Pattern"])
df[["Florescence Pattern"]].head()
```

```
<ipython-input-136-57d1c23b1e71>:1: FutureWarning: A value is trying to be
The behavior will change in pandas 3.0. This inplace method will never work

For example, when doing 'df[col].method(value, inplace=True)', try using 'd

  df["Florescence Pattern"].fillna("Unknown", inplace=True)
```

ID	Florescence Pattern
15940	Unknown
6921	Unknown
23108	Unknown
145172	Unknown
168477	Unknown

```
df["Lead in blood"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Lead in blood
Non-Null Count  Dtype
--------------  -----
719 non-null    object
dtypes: object(1)
memory usage: 46.0+ MB
```

```
df["Anti CCP Abs"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Anti CCP Abs
Non-Null Count  Dtype
--------------  -----
6173 non-null   object
dtypes: object(1)
memory usage: 46.0+ MB
```

```
df["Anti CCP Abs"].value_counts()
```

```
df['Anti CCP Abs'].value_counts()
```

Anti CCP Abs	count
<0.5	905
< 0.5	673
0.5	510
0.3	450
0.4	393
0.0	368
0.6	337
0.2	326
0.1	248
> 200.0	233
0.7	209
0.8	157
0.9	123
>200.0	102
1.1	80
1.0	74
1.2	70
1.3	48
1.4	39
1	28
1.6	27
1.5	24
1.8	20
1.9	19
1.7	18
2.3	16
2.2	14

2.1	12
2.8	12
2.0	11
2.4	9
3.4	8
4.3	8
< 0.50	8
3.9	7
3.5	7
3.3	6
2.5	6
3.1	5
3.0	5
2.7	5
7.4	4
4.6	4
7.5	4
4.1	4
4.5	4
5.5	3
3.7	3
5.0	3
6.0	3
22.5	3
5.3	3
8.6	3
4.8	3

```
df["Anti CCP Abs"].unique()
```

```
array([nan, '<0.5', '0.2', '0.5', '0.4', '< 0.5', '0.6', '0.3', '0.7',
       '0.0', '1.9', '1.1', '136', '1.0', '0.1', '111.1', '97.6', '0.8',
       '69.4', '>1200.0', '0.9', '> 200.0', '16.8', '47.8', '49.0',
'4.6',
       '11.7', '>200.0', '1.3', '1.5', '2.0', '1.2', '9.8', '4.2', '7.0',
       '70.6', '144.7', '132.5', '87.8', '25.9', '1.4', '82.1', '94.7',
       '140.5', '1.6', '2.1', '25.8', '5.5', '8.0', '13.0', '4.3',
       '100.5', '2.4', '7.4', '6.5', '6.3', '51.6', '13.3', '3.6',
       '174.3', '1.8', '17.7', '4.5', '< 0.50', '33.6', '2.7', '9.7',
       '38.8', '3.1', '13.2', '16.2', '22.5', '3.5', '136.1', '146.1',
       '15.3', '97.0', '64.4', '44.4', '2.3', '3.9', '2.2', '6.4', '1',
       '109.4', '10.0', '37.9', '66.5', '11.0', '8.1', '79.1', '51.2',
       '50.2', '5.8', '> 1200.0', '77.4', '2.8', '8.8', '2.6', '124.6',
       '71.2', '7.8', '8.4', '4.7', '4.4', '67.5', '26.9', '6.1', '5.1',
       '26.2', '3.0', '37.8', '3.4', '24.6', '13.9', '177.7', '153.1',
       '19.6', '109.3', '162.6', '2.5', '170.2', '3.8', '4.0', '8.5',
       '40.0', '16.5', '37.3', '2', '87.6', '17.6', '5.2', 'Negative
0.1',
       '3.3', '97.7', '65.9', '4.8', '78.9', '54.3', '8.6', '7.2', '5',
       '181.9', '74.7', '185.9', '94.3', '117.2', '4.1', '101.4',
'0.53',
       '91.5', '188.4', '15.0', '93.2', '38.3', '90.8', '7.3', '7.5',
       '6.6', '13.4', '4.9', '106.7', '39.7', '134.9', '9.9', '14.5',
       '192.0', '18.1', '38.9', '9.4', '6', '114.2', '15.2', '54.5',
       '47.4', '68.9', '16.4', '5.4', '81.9', '112.2', '19', '5.3',
       '41.6', '59.5', '49.6', '72.9', '57.3', '83.7', '33.5', '51.4',
       '166.4', '78.7', '68.6', '159.5', '42.5', '40.4', '72.0', '168.6',
       '44.7', '29.0', '158.7', '10.9', '48.9', '39.4', '11.1', '167.0',
       '18.6', '48.5', '12.9', '75.1', '118.0', '167.2', '110.5', '3',
       '20.2', '67.9', '169.9', '39.6', '45.7', '139.0', '80.1', '75.9',
       '119.7', '187.3', '195.0', '6.9', '47.9', '121.7', '123.9',
'24.5',
       '18.9', '20.8', '67.1', '167.6', '6.0', '66.8', '40.7', '134.5',
       '85.5', '28.5', '152.6', '189.9', '76.1', '93.1', '9.2', '192.7',
       '10.3', '77.2', '2.9', '186', '15.9', '133.4', '22.6', '199.0',
       '37.4', '*339.6', '118.2', '117.3', '13.6', '11.3', '5.7',
'101.2',
       '48.4', '30.0', '19.1', '17.2', '13.5', '107.4', '21.4', '11.9',
       '538.9*', '103.9', '47', '10.6', '32', '23', '0.70', '74.0',
       '32.0', '173.2', '143.5', '64.9', '26.7', '85.4', '46.9', '8',
       '20.6', '188.3', '99.6', '189.4', '164.2', '107.2', '32.5',
'12.0',
       '23.6', '105.1', '104.0', '49.7', '87.5', '14.1', '50.1', '56.0',
       '193.1', '12.2', '69.3', '32.7', '117.8', '5.6', '17.9', '187.2',
```

```
'84.8', '30.8', '6.7', '7.6', '38.6', '173.3', '5.0', '11.4',
'3.7', '16.6', '148.0', '74.1', '193.9', '81.8', '51.9', '17.3',
'64.5', '125.8', '60.8', '981.9*', '107.8', '141.1', '147.1',
'90.4', '10.1', '136.4', '55.5', '34.5', '55.1', '171.9',
'131.3',
'149.9', '4', '3.2', '102.8', '> 200.0> 200.0', '78', '43.7',
'35.5', '71.4', '88.2', '140.9', '127.2', '22.2', '68.1',
'129.1',
'48.7', '139.9', '50.3', '117.7', '196.0', '14.8', '15.6',
'61.4',
'124.1', '24.1', '111.8', '24.2', '24.0', '*907.9', '66.6',
'115.3', '109.7', '60.4', '167.9', '54.8', '151.4', '46.8',
'23.9',
'14.3', '73.2', '21.5', '25.2', '176.3', '158.8', '165.4',
'183.7',
```

5.2	3
3.8	3
24.6	3
5.8	3
5.1	3
8.8	3
7.3	3
6.3	3
8.0	3
99.6	2
9.9	2
4.9	2
58.1	2
0.53	2
5	2

Start coding or generate with AI.

```
df["Anti CCP Abs"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Anti CCP Abs
Non-Null Count  Dtype
--------------  -----
6173 non-null   object
dtypes: object(1)
memory usage: 46.0+ MB
```

```
df["Anti CCP Abs"].value_counts()
```

Anti CCP Abs	count
<0.5	905
< 0.5	673
0.5	510
0.3	450
0.4	393
0.0	368
0.6	337
0.2	326
0.1	248

> 200.0	233
0.7	209
0.8	157
0.9	123
>200.0	102
1.1	80
1.0	74
1.2	70
1.3	48
1.4	39
1	28
1.6	27
1.5	24
1.8	20
1.9	19
1.7	18
2.3	16
2.2	14
2.1	12
2.8	12
2.0	11
2.4	9
3.4	8
4.3	8
< 0.50	8
3.9	7
3.5	7
3.3	6

```
df["Anti CCP Abs"].unique()
```

```
array([nan, '<0.5', '0.2', '0.5', '0.4', '< 0.5', '0.6', '0.3', '0.7',
       '0.0', '1.9', '1.1', '136', '1.0', '0.1', '111.1', '97.6', '0.8',
       '69.4', '>1200.0', '0.9', '> 200.0', '16.8', '47.8', '49.0',
'4.6',
       '11.7', '>200.0', '1.3', '1.5', '2.0', '1.2', '9.8', '4.2', '7.0',
       '70.6', '144.7', '132.5', '87.8', '25.9', '1.4', '82.1', '94.7',
       '140.5', '1.6', '2.1', '25.8', '5.5', '8.0', '13.0', '4.3',
       '100.5', '2.4', '7.4', '6.5', '6.3', '51.6', '13.3', '3.6',
       '174.3', '1.8', '17.7', '4.5', '< 0.50', '33.6', '2.7', '9.7',
       '38.8', '3.1', '13.2', '16.2', '22.5', '3.5', '136.1', '146.1',
       '15.3', '97.0', '64.4', '44.4', '2.3', '3.9', '2.2', '6.4', '1',
       '109.4', '10.0', '37.9', '66.5', '11.0', '8.1', '79.1', '51.2',
```

```
'50.2', '5.8', '> 1200.0', '77.4', '2.8', '8.8', '2.6', '124.6',
2.5      6
       71.2', '7.8', '8.4', '4.7', '4.4', '67.5', '26.9', '6.1', '5.1',
       '26.2', '3.0', '37.8', '3.4', '24.6', '13.9', '177.7', '153.1',
3.1      5
       9.6', '109.3', '162.6', '2.5', '170.2', '3.8', '4.0', '8.5',
       '40.0', '16.5', '37.3', '2', '87.6', '17.6', '5.2', 'Negative
3.0    5
0.1',
       13.3', '97.7', '65.9', '4.8', '78.9', '54.3', '8.6', '7.2', '5',
2.7    5
       '181.9', '74.7', '185.9', '94.3', '117.2', '4.1', '101.4',
'0.53'7.4      4
       '91.5', '188.4', '15.0', '93.2', '38.3', '90.8', '7.3', '7.5',
4.6    4
       6.6', '13.4', '4.9', '106.7', '39.7', '134.9', '9.9', '14.5',
       '192.0', '18.1', '38.9', '9.4', '6', '114.2', '15.2', '54.5',
7.5    4
       47.4', '68.9', '16.4', '5.4', '81.9', '112.2', '19', '5.3',
       '41.6', '59.5', '49.6', '72.9', '57.3', '83.7', '33.5', '51.4',
4.1    4
       66.4', '78.7', '68.6', '159.5', '42.5', '40.4', '72.0', '168.6',
       '44.7', '29.0', '158.7', '10.9', '48.9', '39.4', '11.1', '167.0',
4.5    4
       '18.6', '48.5', '12.9', '75.1', '118.0', '167.2', '110.5', '3',
       '20.2', '67.9', '169.9', '39.6', '45.7', '139.0', '80.1', '75.9',
5.5    3
       '119.7', '187.3', '195.0', '6.9', '47.9', '121.7', '123.9',
'24.5'3.7      3
       '18.9', '20.8', '67.1', '167.6', '6.0', '66.8', '40.7', '134.5',
5.0    3
       85.5', '28.5', '152.6', '189.9', '76.1', '93.1', '9.2', '192.7',
       '10.3', '77.2', '2.9', '186', '15.9', '133.4', '22.6', '199.0',
6.0    3
       37.4', '*339.6', '118.2', '117.3', '13.6', '11.3', '5.7',
'101.2',
22.5   3
       '48.4', '30.0', '19.1', '17.2', '13.5', '107.4', '21.4', '11.9',
5.3    3
       1538.9*', '103.9', '47', '10.6', '32', '23', '0.70', '74.0',
       '32.0', '173.2', '143.5', '64.9', '26.7', '85.4', '46.9', '8',
8.6    3
       '20.6', '188.3', '99.6', '189.4', '164.2', '107.2', '32.5',
'12.0',
4.8    3
       '23.6', '105.1', '104.0', '49.7', '87.5', '14.1', '50.1', '56.0',
       '193.1', '12.2', '69.3', '32.7', '117.8', '5.6', '17.9', '187.2',
5.2    3
       84.8', '30.8', '6.7', '7.6', '38.6', '173.3', '5.0', '11.4',
       '3.7', '16.6', '148.0', '74.1', '193.9', '81.8', '51.9', '17.3',
3.8    3
       64.5', '125.8', '60.8', '981.9*', '107.8', '141.1', '147.1',
       '90.4', '10.1', '136.4', '55.5', '34.5', '55.1', '171.9',
24.6   3
'131.3',
       '149.9', '4', '3.2', '102.8', '> 200.0> 200.0', '78', '43.7',
5.8    3
       '35.5', '71.4', '88.2', '140.9', '127.2', '22.2', '68.1',
'129.1'7.1     3
       '48.7', '139.9', '50.3', '117.7', '196.0', '14.8', '15.6',
'61.4',8.8     3
       '124.1', '24.1', '111.8', '24.2', '24.0', '*907.9', '66.6',
7.3    3
       115.3', '109.7', '60.4', '167.9', '54.8', '151.4', '46.8',
'23.9'
6.3    3
       '14.3', '73.2', '21.5', '25.2', '176.3', '158.8', '165.4',
'183.7'8.0     3
```

```
df["Anti CCP Abs"].unique()
```

```
array([nan, '<0.5', '0.2', '0.5', '0.4', '< 0.5', '0.6', '0.3', '0.7',
       '0.0', '1.9', '1.1', '136', '1.0', '0.1', '111.1', '97.6', '0.8',
       '69.4', '>1200.0', '0.9', '> 200.0', '16.8', '47.8', '49.0',
```

```
'4.6',
'1.7', '>200.0', '1.3', '1.5', '2.0', '1.2', '9.8', '4.2', '7.0',
'70.6', '144.7', '132.5', '87.8', '25.9', '1.4', '82.1', '94.7',
'140.5', '1.6', '2.1', '25.8', '5.5', '8.0', '13.0', '4.3',
'100.5', '2.4', '7.4', '6.5', '6.3', '51.6', '13.3', '3.6',
'174.3', '1.8', '17.7', '4.5', '< 0.50', '33.6', '2.7', '9.7',
'38.8', '3.1', '13.2', '16.2', '22.5', '3.5', '136.1', '146.1',
'15.3', '97.0', '64.4', '44.4', '2.3', '3.9', '2.2', '6.4', '1',
'109.4', '10.0', '37.9', '66.5', '11.0', '8.1', '79.1', '51.2',
'50.2', '5.8', '> 1200.0', '77.4', '2.8', '8.8', '2.6', '124.6',
'71.2', '7.8', '8.4', '4.7', '4.4', '67.5', '26.9', '6.1', '5.1',
'26.2', '3.0', '37.8', '3.4', '24.6', '13.9', '177.7', '153.1',
'9.6', '109.3', '162.6', '2.5', '170.2', '3.8', '4.0', '8.5',
'40.0', '16.5', '37.3', '2', '87.6', '17.6', '5.2', 'Negative
0.1',
'3.3', '97.7', '65.9', '4.8', '78.9', '54.3', '8.6', '7.2', '5',
'181.9', '74.7', '185.9', '94.3', '117.2', '4.1', '101.4',
'0.53',
'91.5', '188.4', '15.0', '93.2', '38.3', '90.8', '7.3', '7.5',
'6.6', '13.4', '4.9', '106.7', '39.7', '134.9', '9.9', '14.5',
'192.0', '18.1', '38.9', '9.4', '6', '114.2', '15.2', '54.5',
'47.4', '68.9', '16.4', '5.4', '81.9', '112.2', '19', '5.3',
'41.6', '59.5', '49.6', '72.9', '57.3', '83.7', '33.5', '51.4',
'66.4', '78.7', '68.6', '159.5', '42.5', '40.4', '72.0', '168.6',
'44.7', '29.0', '158.7', '10.9', '48.9', '39.4', '11.1', '167.0',
'118.6', '48.5', '12.9', '75.1', '118.0', '167.2', '110.5', '3',
'20.2', '67.9', '169.9', '39.6', '45.7', '139.0', '80.1', '75.9',
'119.7', '187.3', '195.0', '6.9', '47.9', '121.7', '123.9',
'24.5',
'18.9', '20.8', '67.1', '167.6', '6.0', '66.8', '40.7', '134.5',
'85.5', '28.5', '152.6', '189.9', '76.1', '93.1', '9.2', '192.7',
'10.3', '77.2', '2.9', '186', '15.9', '133.4', '22.6', '199.0',
'37.4', '*339.6', '118.2', '117.3', '13.6', '11.3', '5.7',
'101.2',
'48.4', '30.0', '19.1', '17.2', '13.5', '107.4', '21.4', '11.9',
'538.9*', '103.9', '47', '10.6', '32', '23', '0.70', '74.0',
'32.0', '173.2', '143.5', '64.9', '26.7', '85.4', '46.9', '8',
'20.6', '188.3', '99.6', '189.4', '164.2', '107.2', '32.5',
'12.0',
'23.6', '105.1', '104.0', '49.7', '87.5', '14.1', '50.1', '56.0',
'193.1', '12.2', '69.3', '32.7', '117.8', '5.6', '17.9', '187.2',
'84.8', '30.8', '6.7', '7.6', '38.6', '173.3', '5.0', '11.4',
'3.7', '16.6', '148.0', '74.1', '193.9', '81.8', '51.9', '17.3',
'64.5', '125.8', '60.8', '981.9*', '107.8', '141.1', '147.1',
'90.4', '10.1', '136.4', '55.5', '34.5', '55.1', '171.9',
'131.3',
'149.9', '4', '3.2', '102.8', '> 200.0> 200.0', '78', '43.7',
'35.5', '71.4', '88.2', '140.9', '127.2', '22.2', '68.1',
'129.1',
'48.7', '139.9', '50.3', '117.7', '196.0', '14.8', '15.6',
'61.4',
'124.1', '24.1', '111.8', '24.2', '24.0', '*907.9', '66.6',
'115.3', '109.7', '60.4', '167.9', '54.8', '151.4', '46.8',
```

9.9 2
4.9 2
58.1 2
0.53 2
5 2
51.4 2
17.6 2
2 2
8.5 2
4.0 2
6.8 2
9.6 2
153.1 2
62.7 2
6.1 2
7.2 2
20.6 2
46.9 2
143.5 2
173.2 2
12.0 2
13.5 2
30.0 2
5.7 2
17.9 2
78.7 2
68.2 2
15.9 2
2.9 2
16.6 2

'23.9', '14.3', '73.2', '21.5', '25.2', '176.3', '158.8', '165.4',

```
df["Anti CCP Abs"] = df["Anti CCP Abs"].replace("Negative 0.1", "0")
```

```
df["Anti CCP Abs"] = df["Anti CCP Abs"].astype(str).str.replace(r'[^\d]', '', r
```

```
df["Anti CCP Abs"] .unique()
```

148.0	2
49.0	2
6.4	2
109.4	2
7.0	2
4.2	2
9.8	2
79.1	2
25.9	2
64.4	2
50.2	2
15.3	2
13.3	2
16.2	2
13.2	2
145.7	2
124.6	2
4.4	2
8.4	2
25.8	2
4.7	2
7.8	2
68.9	2
71.2	2
4	1
131.3	1

```
array(['', '05', '02', '04', '06', '03', '07', '00', '19', '11', '136',
       '10', '01', '1111', '976', '08', '694', '12000', '09', '2000',
       '168', '478', '490', '46', '17', '13', '15', '20', '12', '98',
       '42', '70', '706', '1447', '1325', '878', '259', '14', '821',
       '947', '1405', '16', '21', '258', '55', '80', '130', '43', '1005',
       '24', '74', '65', '63', '516', '133', '36', '1743', '18', '177',
       '45', '050', '336', '27', '97', '388', '31', '132', '162', '225',
       '35', '1361', '1461', '153', '970', '644', '444', '23', '39', '22',
       '64', '1', '1094', '100', '379', '665', '110', '81', '791', '512',
       '502', '58', '774', '28', '88', '26', '1246', '712', '78', '84',
       '47', '44', '675', '269', '61', '51', '262', '30', '378', '34',
       '246', '139', '1777', '1531', '96', '1093', '1626', '25', '1702',
       '38', '40', '85', '400', '165', '373', '2', '876', '176', '52',
       '0', '33', '977', '659', '48', '789', '543', '86', '72', '5',
       '1819', '747', '1859', '943', '1172', '41', '1014', '053', '915',
       '1884', '150', '932', '383', '908', '73', '75', '66', '134', '49',
       '1067', '397', '1349', '99', '145', '1920', '181', '389', '94',
       '6', '1142', '152', '545', '474', '689', '164', '54', '819',
       '1122', '53', '416', '595', '496', '729', '573', '837', '335',
       '514', '664', '787', '686', '1595', '425', '404', '720', '1686',
       '447', '290', '1587', '109', '489', '394', '111', '1670', '186',
       '485', '129', '751', '1180', '1672', '1105', '3', '202', '679',
       '1699', '396', '457', '1390', '801', '759', '1197', '1873', '1950',
       '69', '479', '1217', '1239', '245', '189', '208', '671', '1676',
       '60', '668', '407', '1345', '855', '285', '1526', '1899', '761',
       '931', '92', '1927', '103', '772', '29', '159', '1334', '226',
       '1990', '374', '3396', '1182', '1173', '113', '57', '1012', '484',
       '300', '191', '172', '135', '1074', '214', '119', '5389', '1039',
       '106', '32', '070', '740', '320', '1732', '1435', '649', '267',
       '854', '469', '8', '206', '1883', '996', '1894', '1642', '1072',
       '825', '120', '236', '1051', '1040', '497', '875', '141', '501',
       '560', '1931', '122', '693', '327', '1178', '56', '179', '1872',
       '848', '308', '67', '76', '386', '1733', '50', '114', '37', '166',
       '1480', '741', '1939', '818', '519', '173', '645', '1258', '608',
       '9819', '1078', '1411', '1471', '904', '101', '1364', '555', '345',
       '551', '1719', '1313', '1499', '4', '1028', '20002000', '437',
       '355', '714', '882', '1409', '1272', '222', '681', '1291', '487',
       '1399', '503', '1177', '1960', '148', '156', '614', '1241', '241',
       '1118', '242', '240', '9079', '666', '1153', '1097', '604', '1679',
       '1548', '1514', '468', '239', '143', '732', '215', '252', '1763',
       '1588', '1654', '1837', '1417', '1522', '827', '1251', '228',
       '713', '163', '1260', '1864', '9168', '9231', '707', '7930', '221',
       '680', '71', '1548', '1257', '627', '597', '138', '572', '538',
       '935', '682', '719', '638', '1716', '937', '1333', '906', '1416',
       '142', '193', '68', '1572', '1546', '993', '1755', '1469', '1618',
       '1457', '1342', '1232', '1431', '1643', '155', '20000', '581',
       '692', '790', '1791', '1836', '5070', '1932', '1817', '1611',
       '458', '5263', '094', '494', '422', '657', '1454', '1852', '279',
       '1022', '764', '710', '893', '830', '1103', '803', '851', '2545',
       '901', '249', '1440'], dtype=object)
```

```
df.shape
```

```
(900000, 102)
```

> 200.0	1
102.8	1
60.8	1
127.2	1
187.2	1
94.7	1
140.5	1
5.6	1
117.8	1
97.6	1
32.7	1
69.3	1
12.2	1
64.5	1
56.0	1
50.1	1
14.1	1
87.5	1
57.2	1
49.7	1
23.6	1
104.0	1
105.1	1
193.1	1
125.8	1
111.1	1
17.3	1
51.9	1
81.8	1

```
df["Anti CCP Abs"] = df["Anti CCP Abs"].astype(str).str.lstrip('0')
df["Anti CCP Abs"].unique()
```

```
array(['', '5', '2', '4', '6', '3', '7', '19', '11', '136', '10', '1',
       '1111', '976', '8', '694', '12000', '9', '2000', '168', '478',
       '490', '46', '17', '13', '15', '20', '12', '98', '42', '70', '706',
       '1447', '1325', '878', '259', '14', '821', '947', '1405', '16',
       '21', '258', '55', '80', '130', '43', '1005', '24', '74', '65',
       '63', '516', '133', '36', '1743', '18', '177', '45', '50', '336',
       '27', '97', '388', '31', '132', '162', '225', '35', '1361', '1461',
       '153', '970', '644', '444', '23', '39', '22', '64', '1094', '100',
       '379', '665', '110', '81', '791', '512', '502', '58', '774', '28',
       '88', '26', '1246', '712', '78', '84', '47', '44', '675', '269',
       '61', '51', '262', '30', '378', '34', '246', '139', '1777', '1531',
       '96', '1093', '1626', '25', '1702', '38', '40', '85', '400', '165',
       '373', '876', '176', '52', '33', '977', '659', '48', '789', '543',
       '86', '72', '1819', '747', '1859', '943', '1172', '41', '1014',
       '53', '915', '1884', '150', '932', '383', '908', '73', '75', '66',
       '134', '49', '1067', '397', '1349', '99', '145', '1920', '181',
       '389', '94', '1142', '152', '545', '474', '689', '164', '54',
       '6819', '1122', '416', '595', '496', '729', '573', '837', '335',
       '514', '664', '787', '686', '1595', '425', '404', '720', '1686',
       '447', '290', '1587', '109', '489', '394', '111', '1670', '186',
       '485', '129', '751', '1180', '1672', '1105', '202', '679', '1699',
       '396', '457', '1390', '801', '759', '1197', '1873', '1950', '69',
       '479', '1217', '1239', '245', '189', '208', '671', '1676', '60',
       '668', '407', '1345', '855', '285', '1526', '1899', '761', '931',
       '92', '1927', '103', '772', '29', '159', '1334', '226', '1990',
       '374', '3396', '1182', '1173', '113', '57', '1012', '484', '300',
       '191', '172', '135', '1074', '214', '119', '5389', '1039', '106',
       '32', '740', '320', '1732', '1435', '649', '267', '854', '469',
       '206', '1883', '996', '1894', '1642', '1072', '325', '120', '236',
       '1051', '1040', '497', '875', '141', '501', '560', '1931', '122',
       '693', '327', '1178', '56', '179', '1872', '848', '308', '67',
       '76', '386', '1733', '114', '37', '166', '1480', '741', '1939',
       '818', '519', '173', '645', '1258', '608', '9819', '1078', '1411',
       '1471', '904', '101', '1364', '555', '345', '551', '1719', '1313',
       '1499', '1028', '20002000', '437', '355', '714', '882', '1409',
       '1272', '222', '681', '1291', '487', '1399', '503', '1177', '1960',
       '148', '156', '614', '1241', '241', '1118', '242', '240', '9079',
       '666', '1153', '1097', '604', '1679', '548', '1514', '468', '239',
       '143', '732', '215', '252', '1763', '1588', '1654', '1837', '1417',
       '1522', '827', '1251', '228', '713', '163', '1260', '1864', '9168',
       '9231', '707', '7930', '221', '680', '71', '1548', '1257', '627',
       '597', '138', '572', '538', '935', '682', '719', '638', '1716',
       '937', '1333', '906', '1416', '142', '193', '68', '1572', '1546',
       '993', '1755', '1469', '1618', '1457', '1342', '1232', '1431',
       '1643', '155', '20000', '581', '692', '790', '1791', '1836',
       '5070', '1932', '1817', '1611', '458', '5263', '494', '422', '657',
       '1454', '1852', '279', '1022', '764', '710', '893', '830', '1103',
       '803', '851', '2545', '901', '249', '1440'], dtype=object)
```

```
df["Anti CCP Abs"] = df["Anti CCP Abs"].replace({"'": 0, '"': 0})
```

7.90	1
69.2	1
164.3	1
> 200.00	1
45.8	1
47.8	1
16.8	1
59.7	1
>1200.0	1
69.4	1
13.8	1
24.9	1
90.1	1
254.5*	1
27.9	1
80.3	1
110.3	1
125.7	1
83.0	1
89.3	1
71.0	1
154.8	1
76.4	1
7.1	1
102.2	1
85.1	1
73.2	1
14.3	1

```
df["Anti CCP Abs"].unique()
```

```
array(['', '5', '2', '4', '6', '3', '7', '19', '11', '136', '10', '1',
       '1111', '976', '8', '694', '12000', '9', '2000', '168', '478',
       '490', '46', '17', '13', '15', '20', '12', '98', '42', '70', '706',
       '1447', '1325', '878', '259', '14', '821', '947', '1405', '16',
       '21', '258', '55', '80', '130', '43', '1005', '24', '74', '65',
       '63', '516', '133', '36', '1743', '18', '177', '45', '50', '336',
       '27', '97', '388', '31', '132', '162', '225', '35', '1361', '1461',
       '153', '970', '644', '444', '23', '39', '22', '64', '1094', '100',
       '379', '665', '110', '81', '791', '512', '502', '58', '774', '28',
       '88', '26', '1246', '712', '78', '84', '47', '44', '675', '269',
       '61', '51', '262', '30', '378', '34', '246', '139', '1777', '1531',
       '96', '1093', '1626', '25', '1702', '38', '40', '85', '400', '165',
       '373', '876', '176', '52', '33', '977', '659', '48', '789', '543',
       '86', '72', '1819', '747', '1859', '943', '1172', '41', '1014',
       '53', '915', '1884', '150', '932', '383', '908', '73', '75', '66',
       '134', '49', '1067', '397', '1349', '99', '145', '1920', '181',
       '389', '94', '1142', '152', '545', '474', '689', '164', '54',
       '819', '1122', '416', '595', '496', '729', '573', '837', '335',
       '514', '664', '787', '686', '1595', '425', '404', '720', '1686',
       '447', '290', '1587', '109', '489', '394', '111', '1670', '186',
       '485', '129', '751', '1180', '1672', '1105', '202', '679', '1699',
       '396', '457', '1390', '801', '759', '1197', '1873', '1950', '69',
       '479', '1217', '1239', '245', '189', '208', '671', '1676', '60',
       '668', '407', '1345', '855', '285', '1526', '1899', '761', '931',
       '92', '1927', '103', '772', '29', '159', '1334', '226', '1990',
       '374', '3396', '1182', '1173', '113', '57', '1012', '484', '300',
       '191', '172', '135', '1074', '214', '119', '5389', '1039', '106',
       '82', '740', '320', '1732', '1435', '649', '267', '854', '469',
       '206', '1883', '996', '1894', '1642', '1072', '325', '120', '236',
       '1051', '1040', '497', '875', '141', '501', '560', '1931', '122',
       '693', '327', '1178', '56', '179', '1872', '848', '308', '67',
       '76', '386', '1733', '114', '37', '166', '1480', '741', '1939',
       '818', '519', '173', '645', '1258', '608', '9819', '1078', '1411',
       '1471', '904', '101', '1364', '555', '345', '551', '1719', '1313',
       '1499', '1028', '20002000', '437', '355', '714', '882', '1409',
       '1272', '222', '681', '1291', '487', '1399', '503', '1177', '1960',
       '148', '156', '614', '1241', '241', '1118', '242', '240', '9079',
       '666', '1153', '1097', '604', '1679', '548', '1514', '468', '239',
       '143', '732', '215', '252', '1763', '1588', '1654', '1837', '1417',
       '1522', '827', '1251', '228', '713', '163', '1260', '1864', '9168',
       '9231', '707', '7930', '221', '680', '71', '1548', '1257', '627',
       '597', '138', '572', '538', '935', '682', '719', '638', '1716',
       '937', '1333', '906', '1416', '142', '193', '68', '1572', '1546',
       '993', '1755', '1469', '1618', '1457', '1342', '1232', '1431',
       '1643', '155', '20000', '581', '692', '790', '1791', '1836',
       '5070', '1932', '1817', '1611', '458', '5263', '494', '422', '657',
       '1454', '1852', '279', '1022', '764', '710', '893', '830', '1103',
       '603', '851', '2545', '901', '249', '1440'], dtype=object)
```

46.8	1
151.4	1
54.8	1
167.9	1
60.4	1
70.6	1
15.5	1
115.3	1
144.7	1
*907.3	1
24.0	1
24.2	1
111.8	1
24.1	1
124.1	1
61.4	1
15.6	1
109.7	1
22.1	1
793.0*	1
70.7	1
923.1*	1
*916.8	1
186.4	1
126.0	1
16.3	1
507.8*	1
71.3	1
21.5	1

```
df["Anti CCP Abs"] = df["Anti CCP Abs"].replace("20002000", "200")
```

```
df["Anti CCP Abs"] = pd.to_numeric(df["Anti CCP Abs"], errors='coerce').fillna(
```

```
df["Anti CCP Abs"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Anti CCP Abs
Non-Null Count   Dtype
--------------   -----
900000 non-null  int64
dtypes: int64(1)
memory usage: 46.0 MB
```

```
df["Alkaline Phosphatase"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Alkaline Phosphatase
Non-Null Count   Dtype
--------------   -----
288565 non-null  object
dtypes: object(1)
memory usage: 46.0+ MB
```

```
df["Alkaline Phosphatase"].value_counts()
```

Al
62
66
65
63
64
60
59
61
68
67

```
df["Alkaline Phosphatase"].unique()
```

```
array([ 105 ,  144 ,  119 ,  102 ,  34 , 63.0,  107 , 66.0, 54.0,  153 ,
       '96', '293', '110', '114', '128', '< 36', '216', '108', '169',
       '171', '31', '35', 55.0, 74.0, '< 33.0', 58.0, '*27', 91.0,
'156',
      61.0, '124', '138', '118', '123', 89.0, '192', 79.0, '181',
'280',
       '342', 68.0, '130', '163', '258', '131', '175', '120', '127',
       '134', '162', '140', '249', '116', '172', 53.0, 96.0, '30',
'149',
       '150', '< 28', '*23', 64.0, 65.0, 87.0, 86.0, '<28.5', 80.0,
103.0,
       '198', '151', '176', '137', 106.0, '230', 60.0, '304', '332',
       '274', '147', 97.0, 107.0, 9393.0, 85.0, 75.0, '159', 81.0, 72.0,
```

```
       '191', '233', '126', 59.0, '136', '29', 77.0, '427', '260', 43.0,
       129.0, '270', '185', '206', 50.0, 70.0, '212', 62.0, '278',
'326',
       57.0, '323', '299', '320', '23*', 105.0, 38.0, '152', 67.0, 90.0,
       '135', '272', '179', '222', '196', '170', '442', '182', '200',
       '195', '341', '157', '288', '263', '285', '142', '283', '164',
       '238', '146', '275', '329', '413', '377', '507', '426', '354',
       '297', '464', '428', '448', '305', '322', '418', '313', '409',
       '349', '451', '553', '500', '343', 92.0, '247', '183', '234',
       '831', '691', '207', '286', '208', '132', '165', '257', 41.0,
       '203', 93.0, 52.0, '242', '154', '193', '201', '292', '194',
'20',
       '273', '394', '166', '160', '190', '215', '189', '158', '282',
       '700', 82.0, '375', '187', '374', '174', '209', 139.0, 99.0,
'186',
       '28', '177', '356', '276', 48.0, '298', '<28', 47.0, 69.0, 78.0,
       '265', 56.0, 84.0, '291', '< 23', 95.0, '225', '<30', '358',
'188',
       '232', '521', '213', '373', '325', '303', 102.0, 46.0, '240',
       '461', '316', '312', 49.0, '145', '218', '167', '197', '221',
       133.0, 73.0, '< 30', '307', '168', 94.0, 30.0, 35.0, 45.0, '239',
       '155', '236', '26*', '248', '226', '411', '348', '235', '205',
       '460', 76.0, '25', 112.0, '241', 198.0, '204', 98.0, '340',
'220',
       '229', 120.0, 40.0, 113.0, 37.0, 122.0, '16.0', '237', '184',
'381',
       '760', '360', '180', '217', 146.0, '219', '231', '211', '228',
       217.0, 174.0, '268', '541', '210', '455', 140.0, '390', '264',
       '27*', '199', 306.0, '266', '22*', '924', '555', '515', '*28',
       118.0, 130.0, '267', '317', '300', 123.0, 100.0, 44.0, '369',
       '540', '245', '< 24.0', '>1000', '975', '997', '24*', 167.0,
'223',
       7070.0, '408', 144.0, 137.0, '*26', '324', '301', '214', 128.0,
       104.0, '454', '444', 108.0, '173', '328', '470', '*22', '*11',
       '279', '224', '259', '308', '284', '253', '261', '347', '281',
       '250', '490', '367', '254', '368', '27', 131.0, 101.0, '505',
       '*24', '21*', 148.0, 111.0, 121.0, 220.0, '296', '441', '269',
       36.0, '624', '302', '28*', '450', '26', '334', 124.0, '319',
       '1,564*', '883', 114.0, 184.0, 126.0, 31.0, '417', 109.0, '338',
       '23', 158.0, '406', '458', '376', '262', '765', '287', '310',
       117.0, '> 1100', '785', '365', '518', '552', '601', '396', '473',
       '488', '506', 141.0, '333', '474', '430', '311', 132.0, '837',
       '519', 42.0, 115.0, '29*', '22', 345.0, 207.0, '362', '327',
'498',
       '290', '255', '295', 145.0, '243', 169.0, '403', 142.0, '361',
       '874', '>1447', 125.0, 127.0, 150.0, '318', '378', '251', '16'
```

```
df.shape
```

```
(900000, 102)
```

```
df["Alkaline Phosphatase"] = df["Alkaline Phosphatase"].astype(str).str.strip()
df["Alkaline Phosphatase"] = df["Alkaline Phosphatase"].str.replace(r'[^0-9<>.]


def process_threshold(value):
    if isinstance(value, str):
        if re.match(r'<\s*\d+', value):
            return float(value.replace("<", "").strip()) * 0.9
        elif re.match(r'>\s*\d+', value):
            return float(value.replace(">", "").strip()) * 1.1
    return value

df["Alkaline Phosphatase"] = df["Alkaline Phosphatase"].apply(process_threshold

df["Alkaline Phosphatase"] = pd.to_numeric(df["Alkaline Phosphatase"], errors='
```

Show hidden output

```
df["Alkaline Phosphatase"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Alkaline Phosphatase
Non-Null Count   Dtype
--------------   -----
288565 non-null  float64
dtypes: float64(1)
memory usage: 46.0 MB
```

```
df.shape
```

```
(900000, 102)
```

```
df["Total Protein in Serum"].unique()
```

```
array([nan, 7.1, 7.2, 7.0, 6.5, '7.0', 7.3, 6.9, '6.0', '6.7', 7.4, 6.7,
       7.6, '8.1', 7.8, '7.6', 7.9, 7.7, '7.2', 8.4, '7.4', 7.5, '*14.1',
       '7.3', '6.6', 6.3, 8.0, '7.5', 6.8, 5.7, '6.4', 8.3, 6.4, 6.6,
       '6.5', 6.2, 5.9, '7.9', '6.9', 8.5, '7.7', '7.1', '7', '8.0', 8.6,
       6.0, '8.9', 8.1, '7.8', 8.2, '8.2', 8.7, '6.8', '8.5', '6.2',
       '8.3', 9.1, 6.1, '6.3', '8', 5.8, '8.6', 5.5, '6.1', 4.9, '5.9',
       '5.7', '6', 9.4, 5.2, '5.6', 8.8, '5.4', 9.0, 5.6, '8.7', '5.8',
       8.9, '8.4', 5.4, 66.5, 4.0, '8.8', 9.2, 9.6, '5.2', '9.2', 5.3,
       9.5, 5.0, '9.0', 4.7, '5.1', '7..6', '6.3*', '5.3', 5.1, '4.4',
       '9.1', '>10', '4.8', 4.5, '5.0', 4.8, 9.9, '*5.6', 4.3, 4.1, '4.0',
       '6.96.9', '9.3', 9.3, 4.4, '4.7', '5.5', 4.6, '4.5', 4.2, 3.8,
       '10.4*', 9.8, '4.2', '3.4*', '6.66.6', 82.0, 9.7, 74.1, '(10.7)',
       '9', 3.4, '4.9', '5', '9.5', '9.7', '*11.2', 3.3, '9.4', '>9.5',
       '4.6', '*10.7', 10.3, '3.9', 3.9, 3.7, '*10.9', '*9.6', '*10',
       '*10.2', '9.9*', '5.9*', '9.9', '8.18.1', '4.3', '*10.8', '7.37.3',
       '-', '7.47.4', '*10.1', '(11.3)', '(10.2)'], dtype=object)
```

```
df["Total Protein in Serum"] = df["Total Protein in Serum"].astype(str).str.stri

df["Total Protein in Serum"] = df["Total Protein in Serum"].str.replace(r'[\*\(\

def clean_format(value):
    if isinstance(value, str):
        return re.sub(r'(\d+)\.(\d+)', r'\1.\2', value)
    return value

df["Total Protein in Serum"] = df["Total Protein in Serum"].apply(clean_format)

df["Total Protein in Serum"] = pd.to_numeric(df["Total Protein in Serum"], error

print(f"عدد الصفوف بعد التحويل: {df.shape[0]}")
print(f"عدد القيم المفقودة: {df['Total Protein in Serum'].isna().sum()}")
```

Show hidden output

```
df.shape
```

```
(900000, 102)
```

```
df["Total Protein in Serum"] .info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Total Protein in Serum
Non-Null Count   Dtype
--------------   -----
281103 non-null  float64
dtypes: float64(1)
memory usage: 46.0 MB
```

```
df["Estimated Glomerular Filtration Rate(eGFR)"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Estimated Glomerular Filtration Rate(eGFR)
Non-Null Count   Dtype
--------------   -----
298895 non-null  object
dtypes: object(1)
memory usage: 46.0+ MB
```

```
df.shape
```

```
(900000, 102)
```

```
df["Estimated Glomerular Filtration Rate(eGFR)"].unique()
```

```
array([nan, '>60', '>75', '52', '60', '57', '59', '46', '55', '41', '38',
       '33', '53', '45', '37', '75', '49', '47', '34', '13', '10', '8',
       '42', '26', '54', '22', '40', '39', '50', '17', '23', '28', '27',
       '43', '15', '58', '62', '48', '31', '51', '29', '32', '56', '9',
       '16', '25', '24', '36', '44', '19', '74', '72', '71', '66', '18',
       '65', '21', '68', '11', '20', '6', '64', '61', '12', '35', '14',
       '3', '73', '4', '69', '70', '5', '7', '67', '30', '63', '<10',
       '>90', '> 60', '2', '1', '89', '28.5', '87', '86', '<30', '<20'],
      dtype=object)
```

```
df.shape
```

```
(900000, 102)
```

```
df["Estimated Glomerular Filtration Rate(eGFR)"] = df["Estimated Glomerular Filt

def convert_egfr(value):
    if isinstance(value, str):
        if re.match(r'>\s*\d+', value):
            return float(value.replace(">", "").strip()) + 5
        elif re.match(r'<\s*\d+', value):
            return float(value.replace("<", "").strip()) - 5
    return value

df["Estimated Glomerular Filtration Rate(eGFR)"] = df["Estimated Glomerular Filt

df["Estimated Glomerular Filtration Rate(eGFR)"] = pd.to_numeric(df["Estimated G
```

Show hidden output

```
df.shape
```

```
(900000, 102)
```

```
df["Estimated Glomerular Filtration Rate(eGFR)"] .info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Estimated Glomerular Filtration Rate(eGFR)
Non-Null Count   Dtype
--------------   -----
298895 non-null  float64
dtypes: float64(1)
memory usage: 46.0 MB
```

```
df["Red cell count"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Red cell count
Non-Null Count  Dtype
--------------  -----
216 non-null    object
dtypes: object(1)
memory usage: 46.0+ MB
```

```
#df["Red cell count"].unique()
```

54.0

167

82.0

<28

84.0

55.0

170

171

81.0

79.0

181

174

177

176

```
df.shape
```

```
(900000, 102)
```

```
df["Alkaline Phosphatase"].unique()
```

```
array([         nan, 6.700000e+01, 9.400000e+01, 7.100000e+01,
       9.100000e+01, 5.300000e+01, 9.500000e+01, 6.900000e+01,
       1.030000e+02, 9.800000e+01, 1.060000e+02, 1.010000e+02,
       1.780000e+02, 1.120000e+02, 8.300000e+01, 8.800000e+01,
       5.400000e+01, 4.400000e+01, 5.200000e+01, 1.210000e+02,
       5.000000e+01, 5.800000e+01, 5.700000e+01, 6.600000e+01,
       5.500000e+01, 7.400000e+01, 8.600000e+01, 8.100000e+01,
       5.100000e+01, 5.600000e+01, 8.000000e+01, 7.600000e+01,
       4.900000e+01, 9.700000e+01, 7.500000e+01, 8.500000e+01,
       8.900000e+01, 6.500000e+01, 8.700000e+01, 1.290000e+02,
       6.400000e+01, 3.300000e+01, 9.900000e+01, 3.200000e+01,
       3.900000e+01, 7.300000e+01, 6.000000e+01, 1.090000e+02,
       5.900000e+01, 8.400000e+01, 7.700000e+01, 9.000000e+01,
       4.200000e+01, 4.300000e+01, 4.600000e+01, 6.300000e+01,
       4.500000e+01, 1.220000e+02, 8.200000e+01, 1.480000e+02,
       9.200000e+01, 7.900000e+01, 7.800000e+01, 4.700000e+01,
       1.610000e+02, 6.800000e+01, 6.100000e+01, 1.110000e+02,
       1.040000e+02, 4.100000e+01, 4.800000e+01, 1.100000e+02,
       7.000000e+01, 3.600000e+01, 3.700000e+01, 6.200000e+01,
       7.200000e+01, 2.020000e+02, 1.390000e+02, 9.300000e+01,
       1.330000e+02, 3.800000e+01, 1.250000e+02, 1.410000e+02,
       1.130000e+02, 1.000000e+02, 4.000000e+01, 1.150000e+02,
       1.430000e+02, 1.170000e+02, 2.460000e+02, 1.050000e+02,
       1.440000e+02, 1.190000e+02, 1.020000e+02, 3.400000e+01,
       1.070000e+02, 1.530000e+02, 9.600000e+01, 2.930000e+02,
       1.140000e+02, 1.280000e+02, 3.240000e+01, 2.160000e+02,
       1.080000e+02, 1.690000e+02, 1.710000e+02, 3.100000e+01,
       3.500000e+01, 2.970000e+01, 2.700000e+01, 1.560000e+02,
       1.240000e+02, 1.380000e+02, 1.180000e+02, 1.230000e+02,
       1.920000e+02, 1.810000e+02, 2.800000e+02, 3.420000e+02,
       1.300000e+02, 1.630000e+02, 2.580000e+02, 1.310000e+02,
       1.750000e+02, 1.200000e+02, 1.270000e+02, 1.340000e+02,
       1.620000e+02, 1.400000e+02, 2.490000e+02, 1.160000e+02,
       1.720000e+02, 3.000000e+01, 1.490000e+02, 1.500000e+02,
       2.520000e+01, 2.300000e+01, 2.565000e+01, 1.980000e+02,
       1.510000e+02, 1.760000e+02, 1.370000e+02, 2.300000e+02,
       3.040000e+02, 3.320000e+02, 2.740000e+02, 1.470000e+02,
       9.393000e+03, 1.590000e+02, 1.910000e+02, 2.330000e+02,
       1.260000e+02, 1.360000e+02, 2.900000e+01, 4.270000e+02,
       2.600000e+02, 2.700000e+02, 1.850000e+02, 2.060000e+02,
       2.120000e+02, 2.780000e+02, 3.260000e+02, 3.230000e+02,
       2.990000e+02, 3.200000e+02, 1.520000e+02, 1.350000e+02,
       2.720000e+02, 1.790000e+02, 2.220000e+02, 1.960000e+02,
       1.700000e+02, 4.420000e+02, 1.820000e+02, 2.000000e+02,
       1.950000e+02, 3.410000e+02, 1.570000e+02, 2.880000e+02,
       2.630000e+02, 2.850000e+02, 1.420000e+02, 2.830000e+02,
```

```
       1.640000e+02, 2.380000e+02, 1.460000e+02, 2.750000e+02,
       3.290000e+02, 4.130000e+02, 3.770000e+02, 5.070000e+02,
       4.260000e+02, 3.540000e+02, 2.970000e+02, 4.640000e+02,
       4.280000e+02, 4.480000e+02, 3.050000e+02, 3.220000e+02,
       4.180000e+02, 3.130000e+02, 4.090000e+02, 3.490000e+02,
       4.510000e+02, 5.530000e+02, 5.000000e+02, 3.430000e+02,
       2.470000e+02, 1.830000e+02, 2.340000e+02, 8.310000e+02,
       6.910000e+02, 2.070000e+02, 2.860000e+02, 2.080000e+02,
       1.320000e+02, 1.650000e+02, 2.570000e+02, 2.030000e+02,
       2.420000e+02, 1.540000e+02, 1.930000e+02, 2.010000e+02,
       2.920000e+02, 1.940000e+02, 2.000000e+01, 2.730000e+02,
       3.940000e+02, 1.660000e+02, 1.600000e+02, 1.900000e+02,
       2.150000e+02, 1.890000e+02, 1.580000e+02, 2.820000e+02,
```

```
df["Alkaline Phosphatase"] = df["Alkaline Phosphatase"].astype(float)
```

```
df["Alkaline Phosphatase"].unique()
```

```
array([         nan, 6.700000e+01, 9.400000e+01, 7.100000e+01,
       9.100000e+01, 5.300000e+01, 9.500000e+01, 6.900000e+01,
       1.030000e+02, 9.800000e+01, 1.060000e+02, 1.010000e+02,
       1.780000e+02, 1.120000e+02, 8.300000e+01, 8.800000e+01,
       5.400000e+01, 4.400000e+01, 5.200000e+01, 1.210000e+02,
       5.000000e+01, 5.800000e+01, 5.700000e+01, 6.600000e+01,
       5.500000e+01, 7.400000e+01, 8.600000e+01, 8.100000e+01,
       5.100000e+01, 5.600000e+01, 8.000000e+01, 7.600000e+01,
       4.900000e+01, 9.700000e+01, 7.500000e+01, 8.500000e+01,
       8.900000e+01, 6.500000e+01, 8.700000e+01, 1.290000e+02,
       6.400000e+01, 3.300000e+01, 9.900000e+01, 3.200000e+01,
       3.900000e+01, 7.300000e+01, 6.000000e+01, 1.090000e+02,
       5.900000e+01, 8.400000e+01, 7.700000e+01, 9.000000e+01,
       4.200000e+01, 4.300000e+01, 4.600000e+01, 6.300000e+01,
       4.500000e+01, 1.220000e+02, 8.200000e+01, 1.480000e+02,
       9.200000e+01, 7.900000e+01, 7.800000e+01, 4.700000e+01,
       1.610000e+02, 6.800000e+01, 6.100000e+01, 1.110000e+02,
       1.040000e+02, 4.100000e+01, 4.800000e+01, 1.100000e+02,
       7.000000e+01, 3.600000e+01, 3.700000e+01, 6.200000e+01,
       7.200000e+01, 2.020000e+02, 1.390000e+02, 9.300000e+01,
       1.330000e+02, 3.800000e+01, 1.250000e+02, 1.410000e+02,
       1.130000e+02, 1.000000e+02, 4.000000e+01, 1.150000e+02,
       1.430000e+02, 1.170000e+02, 2.460000e+02, 1.050000e+02,
       1.440000e+02, 1.190000e+02, 1.020000e+02, 3.400000e+01,
       1.070000e+02, 1.530000e+02, 9.600000e+01, 2.930000e+02,
       1.140000e+02, 1.280000e+02, 3.240000e+01, 2.160000e+02,
       1.080000e+02, 1.690000e+02, 1.710000e+02, 3.100000e+01,
       3.500000e+01, 2.970000e+01, 2.700000e+02, 1.560000e+02,
       1.240000e+02, 1.380000e+02, 1.180000e+02, 1.230000e+02,
       1.920000e+02, 1.810000e+02, 2.800000e+02, 3.420000e+02,
       1.300000e+02, 1.630000e+02, 2.580000e+02, 1.310000e+02,
       1.750000e+02, 1.200000e+02, 1.270000e+02, 1.340000e+02,
       1.620000e+02, 1.400000e+02, 2.490000e+02, 1.160000e+02,
       1.720000e+02, 3.000000e+01, 1.490000e+02, 1.500000e+02,
```

```
       2.520000e+01, 2.300000e+01, 2.565000e+01, 1.980000e+02,
       1.510000e+02, 1.760000e+02, 1.370000e+02, 2.300000e+02,
       3.040000e+02, 3.320000e+02, 2.740000e+02, 1.470000e+02,
       9.393000e+03, 1.590000e+02, 1.910000e+02, 2.330000e+02,
       1.260000e+02, 1.360000e+02, 2.900000e+01, 4.270000e+02,
       2.600000e+02, 2.700000e+02, 1.850000e+02, 2.060000e+02,
       2.120000e+02, 2.780000e+02, 3.260000e+02, 3.230000e+02,
       2.990000e+02, 3.200000e+02, 1.520000e+02, 1.350000e+02,
       2.720000e+02, 1.790000e+02, 2.220000e+02, 1.960000e+02,
       1.700000e+02, 4.420000e+02, 1.820000e+02, 2.000000e+02,
       1.950000e+02, 3.410000e+02, 1.570000e+02, 2.880000e+02,
       2.630000e+02, 2.850000e+02, 1.420000e+02, 2.830000e+02,
       1.640000e+02, 2.380000e+02, 1.460000e+02, 2.750000e+02,
       3.290000e+02, 4.130000e+02, 3.770000e+02, 5.070000e+02,
       4.260000e+02, 3.540000e+02, 2.970000e+02, 4.640000e+02,
       4.280000e+02, 4.480000e+02, 3.050000e+02, 3.220000e+02,
       4.180000e+02, 3.130000e+02, 4.090000e+02, 3.490000e+02,
       4.510000e+02, 5.530000e+02, 5.000000e+02, 3.430000e+02,
       2.470000e+02, 1.830000e+02, 2.340000e+02, 8.310000e+02,
       6.910000e+02, 2.070000e+02, 2.860000e+02, 2.080000e+02,
       1.320000e+02, 1.650000e+02, 2.570000e+02, 2.030000e+02,
       2.420000e+02, 1.540000e+02, 1.930000e+02, 2.010000e+02,
       2.920000e+02, 1.940000e+02, 2.000000e+02, 2.730000e+02,
       3.940000e+02, 1.660000e+02, 1.600000e+02, 1.900000e+02,
       2.150000e+02, 1.890000e+02, 1.580000e+02, 2.820000e+02,
```

```
df["Alkaline Phosphatase"] = df["Alkaline Phosphatase"].apply(lambda x: int(x)
```

```
df.shape
```

(900000, 102)

```
df["Alkaline Phosphatase"].unique()
```

100.0
92.0
208
206
102.0
161.0
232
105.0
38.0
28
207
217
204
107.0
216
273
226
210
108.0
246
224
234
219
214
240
211

```
array([nan, 67, 94, 71, 91, 53, 95, 69, 103, 98, 106, 101, 178, 112, 83,
       88, 54, 44, 52, 121, 50, 58, 57, 66, 55, 74, 86, 81, 51, 56, 80,
       76, 49, 97, 75, 85, 89, 65, 87, 129, 64, 33, 99, 32, 39, 73, 60,
       109, 59, 84, 77, 90, 42, 43, 46, 63, 45, 122, 82, 148, 92, 79, 78,
       47, 161, 68, 61, 111, 104, 41, 48, 110, 70, 36, 37, 62, 72, 202,
       139, 93, 133, 38, 125, 141, 113, 100, 40, 115, 143, 117, 246, 105,
       144, 119, 102, 34, 107, 153, 96, 293, 126, 128, 32.4, 216, 108,
       169, 171, 31, 35, 29.7, 27, 156, 124, 138, 118, 123, 192, 181, 280,
       342, 130, 163, 258, 131, 175, 120, 127, 134, 162, 140, 249, 116,
       172, 30, 149, 150, 25.2, 23, 25.650000000000002, 198, 151, 176,
       137, 230, 304, 332, 274, 147, 9393, 159, 191, 233, 126, 136, 29,
       427, 260, 270, 185, 206, 212, 278, 326, 323, 299, 320, 152, 135,
       272, 179, 222, 196, 170, 442, 182, 200, 195, 341, 157, 288, 263,
       285, 142, 283, 164, 238, 146, 275, 329, 413, 377, 507, 426, 354,
       297, 464, 428, 448, 305, 322, 418, 313, 409, 349, 451, 553, 500,
       343, 247, 183, 234, 831, 691, 207, 286, 208, 132, 165, 257, 203,
       242, 154, 193, 201, 292, 194, 20, 273, 394, 166, 160, 190, 215,
       189, 158, 282, 700, 375, 187, 374, 174, 209, 186, 28, 177, 356,
       276, 298, 265, 291, 20.7, 225, 358, 188, 232, 521, 213, 373, 325,
       303, 240, 461, 316, 312, 145, 218, 167, 197, 221, 307, 168, 239,
       155, 236, 26, 248, 226, 411, 348, 235, 205, 460, 25, 241, 204, 340,
       220, 229, 237, 184, 381, 760, 360, 180, 217, 219, 231, 211, 228,
       268, 541, 210, 455, 390, 264, 199, 306, 266, 22, 924, 555, 515,
       267, 317, 300, 369, 540, 245, 21.6, 1100, 975, 997, 24, 223, 7070,
       408, 324, 301, 214, 454, 444, 173, 328, 470, 11, 279, 224, 259,
       308, 284, 253, 261, 347, 281, 250, 490, 367, 254, 368, 505, 21,
       296, 441, 269, 624, 302, 450, 334, 319, 664, 883, 417, 338, 406,
       458, 376, 262, 765, 287, 310, 1210, 785, 365, 518, 552, 601, 396,
       473, 488, 506, 333, 474, 430, 311, 837, 319, 345, 362, 327, 498,
       290, 255, 295, 243, 403, 361, 874, 1447, 318, 378, 251, 16, 388,
       244, 314, 384, 392, 252, 18, 309, 227, 289, 635, 605, 321, 379,
       337, 351, 331, 357, 535, 412, 363, 22.5, 457, 330, 446, 389, 1359,
       463, 6060, 1312, 402, 754, 17, 1197, 393, 973, 387, 315, 401, 609,
       607, 790, 277, 510, 421, 372, 256, 2833, 1135, 615, 429, 569, 400,
       434, 844, 355, 466, 359, 761, 679, 597, 9, 14, 385, 443, 581, 1596,
       757, 491, 1268, 432, 364, 764, 476, 775, 346, 350, 352, 339, 437,
       6868, 382, 829, 534, 7575, 537, 576, 529, 561, 1205, 19, 404, 554,
       271, 756, 423, 479, 994, 344, 386, 666, 353, 492, 5656, 1164, 634,
       641, 974, 903, 482, 538, 447, 399, 425, 422, 433, 294, 424, 520,
       495, 456, 568, 4.5, 630, 815, 595, 435, 810, 380, 420, 714, 604,
       806, 567, 550, 594, 536, 708, 477, 608, 445, 611, 453, 502, 395,
       335, 459, 906, 699, 15, 1273, 678, 1150, 414, 642, 1320, 467, 579,
       1114, 336, 436, 588, 398, 407, 745, 956, 556, 952, 465, 866, 501,
       513, 680, 4747, 6666, 370, 497, 525, 514, 688, 410, 1309,

999999900000000005568454386808673874983305472438300736670640772187451167140
       9696, 469, 1163, 587, 1128, 671, 1259, 1665, 415, 566, 5353, 807,
       468, 582, 371, 925, 7373, 481, 571, 941, 111111, 524, 383, 26.9,
       6363, 967, 1701, 9292, 6, 918, 695, 503, 499, 762, 528, 808, 789,
       776, 1571, 557, 103103, 530, 543, 702, 517, 899, 516, 391, 523,
       546, 618], dtype=object)
```

```
df["Alkaline Phosphatase"] = pd.to_numeric(df["Alkaline Phosphatase"], errors='c

df["Alkaline Phosphatase"] = df["Alkaline Phosphatase"].apply(
    lambda x: int(x) if isinstance(x, float) and x.is_integer() else x
)
```

```
df.shape
```

```
(900000, 102)
```

```
df["Alkaline Phosphatase"].unique()
```

```
array([111.0, 266, 242, 298, 259, 275, '25*', '< 23', 25, 262, 110.0,
       '<30', 278, 303, 252, 288, 227, '*25', 248, 282, 312, 254, 253,
```

```
array([nan, 67, 94, 71, 91, 53, 95, 69, 103, 98, 106, 101, 178, 112, 83,
       88, 54, 44, 52, 121, 50, 58, 57, 66, 55, 74, 86, 81, 51, 56, 80,
       76, 49, 97, 75, 85, 89, 65, 87, 129, 64, 73, 99, 32, 39, 73, 60,
       109, 59, 84, 77, 90, 42, 43, 46, 63, 45, 122, 82, 148, 92, 79, 78,
       47, 161, 68, 61, 111, 104, 41, 48, 110, 70, 36, 37, 62, 72, 202,
       139, 93, 133, 38, 125, 141, 113, 100, 40, 115, 143, 117, 246, 105,
       144, 119, 102, 34, 107, 153, 96, 293, 114, 128, 32.4, 216, 108,
       169, 171, 31, 35, 29.7, 27, 156, 124, 138, 118, 123, 192, 181, 280,
       342, 130, 163, 258, 131, 175, 120, 127, 134, 162, 140, 249, 116,
       172, 30, 149, 150, 25.2, 23, 25.650000000000002, 198, 151, 176,
       137, 230, 304, 332, 274, 147, 9393, 159, 191, 233, 126, 136, 29,
       427, 260, 270, 185, 206, 212, 278, 326, 323, 299, 320, 152, 135,
       272, 179, 222, 196, 170, 442, 182, 200, 195, 341, 157, 288, 263,
       285, 142, 283, 164, 238, 146, 275, 329, 313, 377, 507, 426, 354,
       297, 464, 428, 448, 305, 322, 418, 313, 409, 349, 451, 553, 500,
       343, 247, 183, 234, 831, 691, 207, 286, 208, 132, 165, 257, 203,
       242, 154, 193, 201, 292, 194, 20, 273, 394, 166, 160, 190, 215,
       189, 158, 282, 700, 375, 187, 374, 174, 209, 186, 28, 177, 356,
       276, 298, 265, 291, 20.7, 225, 358, 188, 232, 521, 213, 373, 325,
       303, 240, 461, 316, 312, 145, 218, 167, 197, 221, 307, 168, 239,
       155, 236, 26, 248, 226, 411, 348, 235, 205, 460, 25, 241, 204, 340,
       220, 229, 237, 184, 381, 760, 360, 180, 217, 219, 231, 211, 228,
       268, 541, 210, 455, 390, 264, 199, 306, 266, 22, 924, 555, 515,
       267, 317, 300, 369, 540, 245, 21.6, 1100, 975, 997, 24, 223, 7070,
       408, 324, 301, 214, 454, 444, 173, 328, 470, 11, 279, 224, 259,
       308, 284, 253, 261, 347, 281, 250, 490, 367, 254, 368, 505, 21,
       296, 441, 269, 624, 302, 450, 334, 319, 1564, 883, 417, 338, 406,
       458, 376, 262, 765, 287, 310, 1210, 785, 365, 518, 552, 601, 396,
       473, 488, 506, 333, 474, 430, 311, 837, 519, 345, 362, 327, 498,
       290, 255, 295, 243, 403, 361, 874, 1447, 318, 378, 251, 16, 388,
       244, 314, 384, 392, 252, 18, 309, 227, 289, 635, 605, 321, 379,
       337, 351, 331, 357, 535, 412, 363, 22.5, 457, 330, 446, 389, 1359,
       463, 6060, 1312, 402, 754, 17, 1197, 393, 973, 387, 315, 401, 609,
       607, 790, 277, 510, 421, 372, 256, 2833, 1135, 615, 429, 569, 400,
       434, 844, 355, 466, 359, 761, 679, 597, 9, 14, 385, 443, 581, 1596,
       757, 491, 1268, 432, 364, 764, 476, 775, 346, 350, 352, 339, 437,
       6868, 382, 829, 534, 7575, 537, 576, 529, 561, 1205, 19, 404, 554,
       271, 756, 423, 479, 994, 344, 386, 666, 353, 492, 5656, 1164, 634,
       641, 974, 903, 482, 538, 447, 399, 425, 422, 433, 294, 424, 520,
       495, 456, 568, 4.5, 630, 815, 595, 435, 810, 380, 420, 714, 604,
       806, 567, 550, 594, 536, 708, 477, 608, 445, 611, 453, 502, 395,
       335, 459, 906, 699, 15, 1273, 678, 1150, 414, 642, 1320, 467, 579,
       1114, 336, 436, 588, 398, 407, 745, 956, 556, 952, 465, 866, 501,
       513, 680, 4747, 6666, 370, 497, 525, 514, 688, 410, 1309,
       9999999000000000055684543868086738749833054724383007366706407721874511671409696, 469, 1163, 587, 1128, 671, 1259, 1865, 415, 566, 5353, 807,
       468, 582, 371, 925, 7373, 481, 571, 941, 111111, 524, 383, 26.9,
       6363, 967, 1701, 9292, 6, 918, 695, 503, 499, 762, 528, 808, 789,
       776, 1571, 557, 103103, 530, 543, 702, 517, 899, 516, 391, 523,
       546, 618], dtype=object)
```

```
df["Total Protein in Serum"].unique()
```

```
array([ nan,  7.1,  7.2,  7. ,  6.5,  7.3,  6.9,  6. ,  6.7,  7.4,  7.6,
        8.1,  7.8,  7.9,  7.7,  8.4,  7.5, 14.1,  6.6,  6.3,  8. ,  6.8,
        5.7,  6.4,  8.3,  6.2,  5.9,  8.5,  8.6,  8.9,  8.2,  8.7,  9.1,
        6.1,  5.8,  5.5,  4.9,  9.4,  5.2,  5.6,  8.8,  5.4,  9. , 66.5,
        4. ,  9.2,  9.6,  5.3,  9.5,  5. ,  4.8,  5.1,  4.4, 10. ,  4.8,
        4.5,  9.9,  4.3,  4.1,  9.3,  4.6,  4.2,  3.8, 10.4,  9.8,  3.4,
       82. ,  9.7, 74.1, 10.7, 11.2,  3.3, 10.3,  3.9,  3.7, 10.9, 10.2,
       10.8, 10.1, 11.3])
```

```
df["Total Protein in Serum"] = pd.to_numeric(df["Total Protein in Serum"], error

df["Total Protein in Serum"] = df["Total Protein in Serum"].apply(
    lambda x: int(x) if isinstance(x, (float, int)) and not np.isnan(x) and x.is
)
```

```
df.shape
```

```
(900000, 102)
```

```
df["Total Protein in Serum"].unique()
```

```
array([ nan,  7.1,  7.2,  7. ,  6.5,  7.3,  6.9,  6. ,  6.7,  7.4,  7.6,
        8.1,  7.8,  7.9,  7.7,  8.4,  7.5, 14.1,  6.6,  6.3,  8. ,  6.8,
        5.7,  6.4,  8.3,  6.2,  5.9,  8.5,  8.6,  8.9,  8.2,  8.7,  9.1,
        6.1,  5.8,  5.5,  4.9,  9.4,  5.2,  5.6,  8.8,  5.4,  9. , 66.5,
        4. ,  9.2,  9.6,  5.3,  9.5,  5. ,  4.8,  5.1,  4.4, 10. ,  4.8,
        4.5,  9.9,  4.3,  4.1,  9.3,  4.6,  4.2,  3.8, 10.4,  9.8,  3.4,
       82. ,  9.7, 74.1, 10.7, 11.2,  3.3, 10.3,  3.9,  3.7, 10.9, 10.2,
       10.8, 10.1, 11.3])
```

```
df["Total Protein in Serum"] = df["Total Protein in Serum"].astype(float)
```

```
df["Total Protein in Serum"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Total Protein in Serum
Non-Null Count   Dtype
--------------   -----
281103 non-null  float64
dtypes: float64(1)
memory usage: 46.0 MB
```

```
df["Estimated Glomerular Filtration Rate(eGFR)"].unique()
```

```
array([ nan, 65. , 80. , 52. , 60. , 57. , 59. , 46. , 55. , 41. , 38. ,
       33. , 53. , 45. , 37. , 75. , 49. , 47. , 34. , 13. , 10. ,  8. ,
       42. , 26. , 54. , 22. , 40. , 39. , 50. , 17. , 23. , 28. , 27. ,
       43. , 15. , 58. , 62. , 48. , 31. , 51. , 29. , 32. , 56. ,  9. ,
       16. , 25. , 24. , 36. , 44. , 19. , 74. , 72. , 71. , 66. , 18. ,
       21. , 68. , 11. , 20. ,  6. , 64. , 61. , 12. , 35. , 14. ,  3. ,
       73. ,  4. , 69. , 70. ,  5. ,  7. , 67. , 30. , 63. , 95. ,  2. ,
        1. , 89. , 28.5, 87. , 86. ])
```

```
df.shape
```

```
(900000, 102)
```

```
df["Estimated Glomerular Filtration Rate(eGFR)"] = pd.to_numeric(df["Estimated G

df["Estimated Glomerular Filtration Rate(eGFR)"] = df["Estimated Glomerular Filt
    lambda x: int(x) if isinstance(x, (float, int)) and not np.isnan(x) and x.is
)
```

```
df["Estimated Glomerular Filtration Rate(eGFR)"].unique()
```

```
array([ nan, 65. , 80. , 52. , 60. , 57. , 59. , 46. , 55. , 41. , 38. ,
       33. , 53. , 45. , 37. , 75. , 49. , 47. , 34. , 13. , 10. ,  8. ,
       42. , 26. , 54. , 22. , 40. , 39. , 50. , 17. , 23. , 28. , 27. ,
       43. , 15. , 58. , 62. , 48. , 31. , 51. , 29. , 32. , 56. ,  9. ,
       16. , 25. , 24. , 36. , 44. , 19. , 74. , 72. , 71. , 66. , 18. ,
       21. , 68. , 11. , 20. ,  6. , 64. , 61. , 12. , 35. , 14. ,  3. ,
       73. ,  4. , 69. , 70. ,  5. ,  7. , 67. , 30. , 63. , 95. ,  2. ,
        1. , 89. , 28.5, 87. , 86. ])
```

```
df["Estimated Glomerular Filtration Rate(eGFR)"]=df["Estimated Glomerular Filtr
```

```
df.columns
```

```
Index(['RESEARCH_ID', 'SAMPLE_ID', 'COLLECTYEAR', 'REGN_DATE',
'GENDER_NAME',
       'AGE_YEARS', 'AGE_DAYS', 'AGE_MONTHS', 'CITY_NAME', 'HEIGHT',
       ...
       'Magnesium (Mg) in Serum', 'Titre on Hep2 cells', 'HDL
Cholesterol',
       'Globulin in Serum', 'Cystatin C', 'RESEARCH_ID_int',
'GENDER_BINARY',
       'CITY_NAME_ENCODED', 'BDL', 'Florescence Pattern '],
      dtype='object', length=102)
```

```
df.shape
```

```
(900000, 102)
```

```
df["Blood pressure"].unique()
```

```
array([nan, '-', '125/81', ..., '162 / 91', '118/51', '76/41'],
      dtype=object)
```

```
def clean_blood_pressure(value):
    if isinstance(value, str):
        value = value.strip()
        if value == "-" or value == "" or value.lower() == "nan":
            return None
        if " / " in value:
            value = value.replace(" / ", "/")
    return value

df["Blood pressure"] = df["Blood pressure"].apply(clean_blood_pressure)
```

```
def split_blood_pressure(value):
    if isinstance(value, str) and "/" in value:
        try:
            systolic, diastolic = map(int, value.split("/"))
            return systolic, diastolic
        except ValueError:
            return None, None
    return None, None

df[["Systolic Pressure", "Diastolic Pressure"]] = df["Blood pressure"].apply(lam
df.drop(columns=["Blood pressure"], inplace=True)
```

```
df["Systolic Pressure"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Systolic Pressure
Non-Null Count   Dtype
--------------   -----
292833 non-null  float64
dtypes: float64(1)
memory usage: 46.0 MB
```

```
df.shape
```

```
(900000, 103)
```

```
df["Diastolic Pressure"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Diastolic Pressure
Non-Null Count   Dtype
--------------   -----
292833 non-null  float64
dtypes: float64(1)
memory usage: 46.0 MB
```

```
df.dtypes
```

	0
RESEARCH_ID	object
SAMPLE_ID	object
COLLECTYEAR	int64
REGN_DATE	datetime64[ns]
GENDER_NAME	object
AGE_YEARS	Int64
AGE_DAYS	Int64
AGE_MONTHS	Int64
CITY_NAME	category
HEIGHT	int64
WEIGHT	float64

BMI	float64
Thyroid Stimulating Hormone (TSH)	float64
Uric Acid in Serum	float64
Alanine Aminotransferase (ALT)	float64
Ferritin In Serum	float64
Blood Urea Nitrogen (BUN)	float64
Lymphocytes absolute count	float64
R. B. Cs / HPFs	float64
Aspect(Urine Physical Examination) Ordinal Encoding	float64
Eosinophils absolute count	float64
Vitamin D (25 OH-Vit D -Total)	float64
C-Reactive Protein (CRP) quantitative	float64
Transferrin	float64
Red cell count	object
Basophils absolute count	float64
Crystals(Urine Microscopic Examination :)	float64
Protein(Urine Physical Examination)	float64
Colour(Urine Physical Examination)	float64
Nitrite	float64
LDL Cholesterol	int64
LDL / HDL	float64
24 Hour Urine Volume (263)	float64
Hemoglobin	float64
Total Leucocytic Count	float64
Hematocrit	float64
MCV	float64
Glucose(Urine Physical Examination)	float64
Urea in Serum	float64
Prostatic Specific Antigen (PSA) Total	float64
Testosterone (Total)	float64

Alkaline Phosphatase	object
Total Protein in Serum	float64
Estimated Glomerular Filtration Rate(eGFR)	float64
Anti CCP Abs	int64
BUN/Creatinine Ratio	object
Ketones	object
MCHC	float64
pH(Urine Physical Examination)	float64
Amorphous Elements	object
Blood and Haemoglobin	object
Epithelial Cells / HPF	object
Casts(Urine Microscopic Examination :)	object
Bilirubin	object
Chloride in Serum	object
Cholesterol	object
T. Cholesterol/HDL	object
Urobilinogen	object
R.B.Cs / HPF	object
Erythrocyte Sedimentation Rate(ESR)	object
Glucose in Plasma (Fasting)	object
Hb A1c %	object
Mean of blood glucose	object
Microalbuminuria (24 h urine)	object
Bilirubin (Total)	object
Florescence Pattern	object
Lead in blood	object
Monocytes absolute count	float64
Consistancy	object
Neutrophils absolute count	float64

Specific Gravity	object
W. B. Cs / HPF	object
Aspartate Aminotransferase (AST)	object
Calcium in Serum (Total)	object
Free T4	object
Potassium (K) in Serum	object
Albumin in Serum	object
Iron (Fe) in Serum	object
CRP H.S	object
Triglycerides (TG) in Serum	object
Rheumatoid Factor (quantitative)	object
Platelet Count	object
Albumin in Urine (263)	float64
MCH	float64
RDW	object
W.B.Cs / HPF	object
Leucocyte esterase	object
Concentration	object
Creatinine in Serum	object
Sodium (Na) in Serum	object
Bilirubin (Direct)	object
Magnesium (Mg) in Serum	object
Titre on Hep2 cells	object
HDL Cholesterol	object
Globulin in Serum	float64
Cystatin C	object
RESEARCH_ID_int	object
GENDER_BINARY	int64
CITY_NAME_ENCODED	int64
BDL	int64

Florescence Pattern	int64
Systolic Pressure	float64
Diastolic Pressure	float64

dtype: object

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 900000 entries, 15940 to 679524
Columns: 103 entries, RESEARCH_ID to Diastolic Pressure
dtypes: Int64(3), category(1), datetime64[ns](1), float64(40), int64(8), ob
memory usage: 742.9+ MB
```

```
df['Red cell count'].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Red cell count
Non-Null Count  Dtype
--------------  -----
216 non-null    object
dtypes: object(1)
memory usage: 46.0+ MB
```

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

```
df[' BUN/Creatinine Ratio'].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name:  BUN/Creatinine Ratio
Non-Null Count   Dtype
--------------   -----
275271 non-null  object
dtypes: object(1)
memory usage: 46.0+ MB
```

```
df[' BUN/Creatinine Ratio'].unique()
```

```
array([nan, 10.0, 16.8, 17.3, '15.5', 12.7, 14.6, 23.0, 19.4, 12.2, 9.6,
       14.8, 12.9, 17.7, 11.3, 24.4, 20.8, 10.8, 17.5, 17.6, 18.6, 21.7,
       16.2, 15.2, 10.5, 15.7, 15.4, 12.0, 26.7, 18.0, '11.1', 11.1,
13.8,
```

```
       13.6, 30.2, 16.5, 11.4, 16.3, 30.8, 16.1, 11.6, 13.9, 15.0, 15.8,
       20.0, 23.3, 12.4, 14.7, 26.2, 10.1, 7.4, 20.5, 16.4, 12.3, 14.0,
       16.6, 19.5, 21.5, 9.9, 13.7, 24.1, 14.2, 12.1, 13.0, 11.9, 14.9,
       14.1, 6.3, 11.8, 9.5, 13.5, 13.1, 13.3, 18.1, 18.9, 24.8, 18.7,
       '12.7', 19.7, 16.7, 22.7, 19.1, 12.6, 14.3, 11.2, 34.1, 37.1,
19.0,
       21.8, 24.0, 13.4, 17.2, 21.9, 14.4, 17.4, 9.8, 27.8, 42.9, 8.5,
       9.2, 8.9, 7.8, 17.1, 14.5, 11.7, 12.5, 21.4, 26.5, 19.2, 7.5,
17.9,
       10.3, 41.8, 15.1, 7.1, 9.4, 22.2, 10.7, 15.6, 15.9, 27.9, 10.9,
       35.0, 19.3, 16.9, '12.0', 9.7, 16.0, 21.0, 12.8, 24.5, 39.5,
20.2,
       29.8, 20.7, 8.8, 21.6, 7.3, 26.1, 9.1, 29.1, 21.3, 10.6, 20.6,
       25.3, 17.0, 34.5, 13.2, 23.6, 32.9, 32.4, 25.8, 30.4, 18.4, 25.1,
       15.3, 24.2, 26.3, '11.3', 26.4, 32.7, 10.4, 21.1, 8.7, 23.9,
25.0,
       17.8, 18.5, 28.6, 18.8, 19.8, 11.0, 18.2, 8.4, 11.5, 15.5, 18.3,
       22.1, 8.6, '14.6', 20.3, 5.2, 6.8, 20.4, 19.6, 31.3, 23.2, 10.2,
       '11.9', 36.2, 7.9, 8.2, 24.9, '15.2', '14.2', 28.0, 31.1, '18.8',
       28.8, 22.4, 9.0, 25.9, '25.5', 22.9, 29.6, '30.9', 31.2, 23.1,
       29.3, 28.2, 23.4, 21.2, 29.2, 27.6, 28.5, 27.2, 8.3, 24.3, 28.9,
       22.5, 30.0, 7.6, 25.4, 25.6, 9.3, 8.0, 6.4, 26.9, 28.7, 26.0,
8.1,
       27.7, 26.6, 28.4, 61.2, 65.0, 7.2, 23.8, 20.1, 7.0, 31.7, '20.5',
       7.7, 22.6, 19.9, 22.3, 34.8, '13.5', 5.9, 39.8, 29.4, 26.8, 33.3,
       33.0, '9.4', '11.0', '14.5', '9.1', '13.2', '24.9', 22.8, 38.6,
       29.5, '20.0', '6.3', 35.9, 20.9, 34.3, 24.7, 30.6, '20.8', 68.6,
       37.5, 33.7, 23.7, 23.5, 28.3, '10.2', '13.1', 27.1, 30.5, '25.7',
       '19.0', 3.1, 25.5, '26.0', '18.3', 24.6, 27.0, 29.7, '20.3',
32.0,
       27.3, 25.7, 6.7, 5.4, 5.7, 35.3, '17.4', 32.1, '28.6', 31.5,
32.3,
       28.1, '9.5', 22.0, 6.2, 57.9, 35.5, 34.4, '13.6', 31.4, 6.1,
45.2,
       33.8, 27.5, 36.3, 31.6, 31.0, '27.3', '28.2', '17.8', 37.3, 60.0,
       '14.9', 5.8, '16.5', '16.4', '10.4', '17.9', '34.0', '8.2',
'18.5',
       5.0, '9.7', 25.2, '15.6', 39.3, 6.5, 27.4, '12.5', 36.7, '14.7',
       30.3, 34.2, 5.5, '9.9', 32.8, '16.7', '12.4', 31.8, 32.2, 34.0,
       30.7, 33.4, 40.0, 4.2, '20.2', 34.9, '19.7', '14.3', 3.6, '14.8',
       '15.8', '8.9', '12.2', 29.0, 55.2, '14.1', '20.1', '13.0', 37.8,
       '12.3', '17.2', '22.5', 43.8, 6.9, 42.0, '16.8', '10.3', '15.4',
       '18.2', 30.9, '12.8', '12.9', 4.8, 4.1, '19.1', '13.3', '15.0',
       39.6, 6.6, '11.2', '22.9', '11.7', 33.5, 5.3, 41.7, 32.5, 36.5,
       37.9, 36.9, '23.5', '15.7', 58.9, '23.0', '16.9', '12.1', 42.5,
       '19.3', 4.9, 34.6, 36.1, 29.9, '29.2', '17.5', 5.6, '12.6', 39.7,
       '26.5', 6.0, 33.6, 33.9, '14.4', 4.6, 30.1, 37.0, '20.7', 57.4,
       31.9, 35.1, 49.1, 38.7, 35.8, 40.3, '18.7', 42.3, '18.9', 44.4,
       '11.4', '13.9', '23.3', '15.3', '13.4', 36.0, '19.9', 47.0,
'16.2',
       36.4, 32.6, 36.8, '23.8', '21.6', '10.9', '16.1', '17.6', '9.0',
       35.6, '21.5', '8.8', 44.8, 33.2, 4.4, '19.2', '15.9', '11.8',
48.0,
```

```
414
426
443
387
501
143.0
410
680
618
513
530
16*
543
702
197.0
208.0
517
5353.0
807
468
582
925
7373.0
481
671
941
111111.0
524
26.9
6363.0
```

48.3 48.4 '26.7' '19.4' 36.6 '13.8' '22.2' 47.9 33.1

```
def clean_bun_creatinine(value):
    if isinstance(value, str):
        value = value.strip()
        if value in ["-", "nan", ".", ""]:
            return np.nan
    try:
        value = float(value)
        return int(value) if value.is_integer() else value
    except ValueError:
        return np.nan

df[" BUN/Creatinine Ratio"] = df[" BUN/Creatinine Ratio"].apply(clean_bun_creat
```

6666.0

```
df.shape
```

```
(900000, 103)
```

525

514

688

1309*

*21

161.0

99999990000000000000000000000000000000000000000000000000000000000000

<33.0

9696.0

202.0

469

251.0

1163

213.0

499

762

528

808

789

```
print(df["Casts(Urine Microscopic Examination :)"].unique())

casts_mapping = {
    "Absent": 0,
    "Few": 1,
    "Occasional": 1,
    "Moderate": 2,
    "Many": 3,
    "Not Performed": np.nan,
    "–": np.nan,
    " ": np.nan,
}

df["Casts(Urine Microscopic Examination :)"] = df["Casts(Urine Microscopic Exami
df["Casts(Urine Microscopic Examination :)"] = df["Casts(Urine Microscopic Exami
df["Casts(Urine Microscopic Examination :)"].fillna(0, inplace=True)
df["Casts(Urine Microscopic Examination :)"] = df["Casts(Urine Microscopic Exami
print(df["Casts(Urine Microscopic Examination :)"].unique())
print(df["Casts(Urine Microscopic Examination :)"].info())
```

```
[nan 'Absent' 'Fine Granular Casts' 'Coarse Granular Casts']
[0]
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Casts(Urine Microscopic Examination :)
Non-Null Count   Dtype
--------------   -----
900000 non-null  int64
dtypes: int64(1)
memory usage: 46.0 MB
None
<ipython-input-210-5ccab8dd3e76>:17: FutureWarning: A value is trying to be
The behavior will change in pandas 3.0. This inplace method will never work

For example, when doing 'df[col].method(value, inplace=True)', try using 'd

  df["Casts(Urine Microscopic Examination :)"].fillna(0, inplace=True)
```

```
df.shape
```

```
(900000, 103)
```

429

569

157.0

844

160.0

210.0

<25

457.0

289.0

446

1,359*

6060.0

1312*

754

296.0

1197

```
df["Casts(Urine Microscopic Examination :)"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Casts(Urine Microscopic Examination :)
Non-Null Count   Dtype
--------------   -----
900000 non-null  int64
dtypes: int64(1)
memory usage: 46.0 MB
```

```
df["Bilirubin"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Bilirubin
Non-Null Count  Dtype
--------------  -----
206 non-null    object
dtypes: object(1)
memory usage: 46.0+ MB
```

```
df['Chloride in Serum'].unique()
```

```
array([nan, 104.0, 109.0, '101', 102.0, 103.0, 105.0, '107', 108.0, 101.0,
       '103', '98', '102', 107.0, 99.0, 100.0, '100', 106.0, 98.0, '108',
       '104', '105', '90', '97', '106', 89.0, '99', 95.0, '96', 93.0,
       113.0, 96.0, 97.0, '109', 110.0, '111', '110', '94', 94.0, '93',
       92.0, '95', 87.0, 91.0, 84.0, '87', 111.0, '92', 86.0, '88', '91',
       90.0, '115*', 85.0, 72.0, '83', 112.0, 88.0, 103103.0, 114.0, '71',
       '112', 82.0, '113', '80', 122.0, 104104.0, '115', '123*', 79.0,
       '78', 120.0, 121.0, 115.0, '66', '116', 100100.0, '89', '118',
       '< 98', 106106.0, 78.0, 118.0, 83.0, 77.0, '114', 73.0, '84',
       9999.0, 102102.0, 116.0, '85', '<75', '86', 119.0, 101101.0, '76',
       '80*', '116*', '82', 80.0, '81', '117', '121', 76.0, 117.0, '77',
       81.0, 105105.0, '*125', '75', 9797.0, 98.1, '127*', 75.0, '73',
       124.0, '79', '100.70', '109*', '117*', 9898.0, 101.1, '75*', 74.0,
       '126*', '120', '74', 107107.0, 137.0, '118', 70.0, 71.0, '97*',
       '124*', '76*', '122', '*129', '*126', '*128', '*123', '68', '72',
       68.0, '119*', '*134', '119', '71*', '121*', '123', '78*', 69.0,
       '74*', '79*'], dtype=object)
```

```
def clean_chloride(value):
    if isinstance(value, str):
        value = value.strip()
        value = value.replace("*", "").replace("<", "")
        if value.isdigit():
            return float(value)
    return value


df["Chloride in Serum"] = df["Chloride in Serum"].apply(clean_chloride)
df["Chloride in Serum"] = pd.to_numeric(df["Chloride in Serum"], errors='coerce'
df["Chloride in Serum"] = df["Chloride in Serum"].apply(lambda x: int(x) if pd.n
```

```
df["Chloride in Serum"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Chloride in Serum
Non-Null Count   Dtype
--------------   -----
292027 non-null  float64
dtypes: float64(1)
memory usage: 46.0 MB
```

```
df['Cholesterol'].unique()
```

```
array([nan, 172.0, '266', 185.0, 179.0, 227.0, 206.0, '176', 200.0,
148.0,
       '150', 139.0, '144', 170.0, '242', 187.0, '198', 193.0, '186',
       '155', 201.0, 215.0, '166', '221', 207.0, 219.0, 190.0, '162',
       '153', 220.0, '181', 177.0, 156.0, '215', '170', 213.0, 166.0,
       '189', '139', 191.0, 238.0, 106.0, '165', 195.0, 267.0, 149.0,
       137.0, 164.0, 256.0, 120.0, '102', 162.0, 168.0, '120', 165.0,
       '223', 272.0, '217', 194.0, 231.0, '241', 208.0, 188.0, 221.0,
       211.0, 161.0, '248', '239', '252', '160', 203.0, 178.0, 192.0,
       253.0, '172', 152.0, 197.0, 217.0, '213', '207', 199.0, 205.0,
       186.0, 216.0, 255.0, '145', 143.0, '199', 210.0, 142.0, 163.0,
       251.0, '201', '184', 138.0, 146.0, '108', 118.0, '188', '192',
       204.0, 198.0, 258.0, '295', '206', '147', 180.0, 289.0, 218.0,
       275.0, 196.0, '197', 131.0, 116.0, 182.0, 147.0, 181.0, 176.0,
       235.0, '154', '156', 189.0, '228', 308.0, 171.0, '136', 130.0,
       230.0, '123', 263.0, 299.0, '161', '195', 254.0, 309.0, '235',
       242.0, '225', '212', '219', '151', '116', 154.0, 245.0, 264.0,
       '284', 229.0, '135', 140.0, '175', 241.0, '246', 167.0, '226',
       '163', '255', 287.0, 317.0, 225.0, 232.0, 124.0, '179', 112.0,
       214.0, '191', 202.0, 265.0, '249', 243.0, 135.0, '169', '127',
       '133', '171', '115', 141.0, 236.0, '164', 136.0, '200', '159',
       122.0, 183.0, 234.0, '143', 226.0, 237.0, '220', '180', '218',
       240.0, 224.0, 158.0, 249.0, '208', 283.0, '286', '258', 73.0,
```

```
       '205', '204', '130', 159.0, '178', 129.0, '168', '267', '256',
       222.0, '202', 151.0, '240', '231', '304', 175.0, '149', '174',
       '243', '190', 117.0, 126.0, '289', 212.0, '282', '121', '224',
       '185', '233', 115.0, '214', '132', 150.0, 134.0, 209.0, 153.0,
       '142', 144.0, '229', 262.0, 257.0, '167', '196', '194', 223.0,
       '244', 125.0, '131', '211', 160.0, 121.0, 291.0, '222', 252.0,
       '216', '230', '265', '138', 246.0, '203', 261.0, '187', '152',
       '278', '177', '117', '97', '272', 145.0, 113.0, '113', '209',
       '148', 157.0, 169.0, 301.0, '238', 280.0, '260', 268.0, 380.0,
       363.0, 173.0, 228.0, 127.0, '182', '137', '263', 128.0, '245',
       '316', 184.0, 133.0, '236', 250.0, 294.0, 269.0, '232', 278.0,
       174.0, 288.0, '193', 107.0, 247.0, '141', 270.0, '146', '250',
       132.0, 244.0, '140', '134', 248.0, 274.0, 314.0, '292', 279.0,
       '237', '280', '253', '158', 123.0, '181', 290.0, '157', '112',
       '173', '122', '297', 297.0, 266.0, '114', '234', 233.0, 87.0,
       '254', '118', 273.0, '126', 259.0, 300.0, '274', 239.0, '210',
       104.0, '119', 114.0, '257', 155.0, '78', 319.0, '125', '261',
       284.0, '275', 276.0, 295.0, '283', 310.0, '262', 110.0, 330.0,
       '251', '294', '247', '313', 92.0, '270', '269', '99', '86',
'259',
       292.0, 397.0, '109', 282.0, '264', 281.0, '81', 105.0, 111.0,
       '101', 101.0, 109.0, '110', 271.0, '371', '296', '276', '89',
       '227', '129', '88', '322', 316.0, 285.0, '103', 188188.0, '321',
       260.0, '111', '271', '268', 119.0, '91', 296.0, 340.0, 337.0,
       345.0, 108.0, '128', 95.0, 100.0, '326', 374.0, '.', '325',
'281',
       328.0, 332.0, 298.0, 293.0, '124', 86.0, '273', 350.0, 286.0,
       277.0, 98.0, 313.0, '300', 327.0, 307.0, '90', '306', '303',
304.0,
       '288', 439.0, '305', 302.0, '298', '299', 322.0, '342', '98',
       333.0, 56.0, 103.0, '83', 306.0, 88.0, '341', '287', 366.0,
'293',
       312.0, '323', '290', 5.06, 323.0, '279', '277', '350', '349',
       102.0, 97.0, '311', 62.0, '92', 365.0, '106', '310', 311.0,
'302',
       '312', 315.0, '96', 324.0, 305.0, 78.0, '105', 359.0, 85.0
```

458
765
71100
785
470
131.0
148.0
220.0
666
492
5656.0
1164
634
641
974
229.0
154.0
903
538
447
399
433
520
495
568
250.0
292.0
595
975
997

```
def clean_cholesterol(value):
    if isinstance(value, str):
        value = value.strip()
        value = value.replace("*", "").replace("<", "").replace(">", "")
        if value.isdigit():
            return float(value)
    return value


df["Cholesterol"] = df["Cholesterol"].apply(clean_cholesterol)


df["Cholesterol"] = pd.to_numeric(df["Cholesterol"], errors='coerce')
def clean_cholesterol(value):
    if isinstance(value, str):
        value = value.strip()
        value = value.replace("*", "").replace("<", "").replace(">", "")
        if value.isdigit():
            return float(value)
    return value

df["Cholesterol"] = df["Cholesterol"].apply(clean_cholesterol)
df["Cholesterol"] = pd.to_numeric(df["Cholesterol"], errors='coerce')

df["Cholesterol"] = df["Cholesterol"].apply(lambda x: int(x) if pd.notna(x) anc
```

```
df["Cholesterol"] = pd.to_numeric(df["Cholesterol"], errors='coerce').astype(fl
```

```
df['T. Cholesterol/HDL'].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: T. Cholesterol/HDL
Non-Null Count  Dtype
--------------  -----
243 non-null    object
dtypes: object(1)
memory usage: 46.0+ MB
```

149.0

205.0

708

189.0

608

445

611.0

395

459

906

699

```
df['T. Cholesterol/HDL'].unique()
```

```
array([nan, 5.9, 4.5, 3.2, 3.1, 3.6, 5.2, 5.1, 5.0, 4.1, 5.3, 3.8, 3.3,
       3.7, 4.3, 3.0, 5.6, 4.7, 7.7, 4.0, 2.8, 6.4, 4.4, '(12.2)', 4.2,
       6.1, 6.5, 2.6, 5.5, 8.1, 2.9, 2.1, 2.7, 3.9, 6.0, '4.8', 4.6, 2.2,
       2.4, 7.5, 2.0, 5.7, 4.9, 6.7, 8.9, 4.8, 6.6, 3.5, 3.4, 5.8, 2.3,
       7.6, 7.1, 6.8, 6.3, 7.2, 7.0, 5.4, 1.6, 8.7, 6.9, 6.2],
      dtype=object)
```

```
df.shape
```

```
(900000, 103)
```

```
def clean_and_convert_urobilinogen(value):
    if isinstance(value, str):
        value = re.sub(r"[><*(),]", "", value).strip()
        try:
            return float(value)
        except ValueError:
            return np.nan
    return value

df["Urobilinogen"] = df["Urobilinogen"].apply(clean_and_convert_urobilinogen)
```

```
df["Urobilinogen"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Urobilinogen
Non-Null Count  Dtype
--------------  -----
0 non-null      float64
dtypes: float64(1)
memory usage: 46.0 MB
```

```
df['R.B.Cs / HPF'].unique()
```

```
array([nan, '0 - 1', '3 - 5', '1-3', 'Absent'], dtype=object)
```

```
df['R.B.Cs / HPF'].head()
```

ID	R.B.Cs / HPF
15940	NaN
6921	NaN
23108	NaN
145172	NaN
168477	NaN

dtype: object

```
def clean_esr_value(value):
    if isinstance(value, str):
        value = value.strip()


        value = re.sub(r'^0+(\d+)', r'\1', value)


        if value.startswith('>'):
            value = value.replace('>', '').strip()


        if '/' in value:
            return np.nan


        try:
            num_value = float(value)
            return int(num_value) if num_value.is_integer() else num_value
        except ValueError:
            return np.nan

    return value

df['Erythrocyte Sedimentation Rate(ESR)'] = df['Erythrocyte Sedimentation Rate(E

unique_values_after_cleaning = df['Erythrocyte Sedimentation Rate(ESR)'].unique(
```

0.166	5
0.215	5

```
unique_values_after_cleaning
```

```
array([1.600e+01,        nan, 1.800e+01, 2.200e+01, 1.000e+01, 8.000e+00,
       7.700e+01, 1.700e+01, 7.000e+00, 3.400e+01, 1.200e+01, 3.000e+00,
       4.000e+00, 5.000e+00, 4.200e+01, 2.500e+01, 1.400e+01, 3.200e+01,
       1.300e+01, 2.000e+00, 4.300e+01, 3.100e+01, 9.000e+00, 3.600e+01,
       6.000e+00, 4.000e+01, 1.500e+01, 2.300e+01, 3.000e+01, 1.900e+01,
       2.000e+01, 5.000e+01, 3.500e+01, 2.700e+01, 3.300e+01, 4.700e+01,
       2.600e+01, 2.900e+01, 2.100e+01, 7.400e+01, 5.500e+01, 5.800e+01,
       2.400e+01, 5.300e+01, 3.700e+01, 4.500e+01, 1.100e+01, 6.500e+01,
       8.500e+01, 2.800e+01, 4.800e+01, 6.200e+01, 7.100e+01, 9.100e+01,
       6.100e+01, 5.200e+01, 6.700e+01, 6.000e+01, 4.100e+01, 2.020e+03,
       5.400e+01, 1.515e+03, 1.000e+00, 7.500e+01, 1.100e+02, 1.150e+02,
       8.400e+01, 9.600e+01, 6.900e+01, 5.700e+01, 7.000e+01, 1.250e+02,
       3.800e+01, 9.500e+01, 4.600e+01, 6.300e+01, 4.400e+01, 6.800e+01,
       8.600e+01, 3.900e+01, 5.600e+01, 1.616e+03, 6.060e+03, 8.800e+01,
       9.000e+01, 4.900e+01, 7.800e+01, 7.900e+01, 1.080e+02, 6.600e+01,
       7.200e+01, 6.400e+01, 1.000e+02, 5.900e+01, 9.900e+01, 5.100e+01,
       8.000e+01, 1.200e+02, 1.050e+02, 1.170e+02, 2.727e+03, 8.300e+01,
       8.100e+01, 1.212e+03, 9.200e+01, 2.828e+03, 3.030e+03, 7.300e+01,
       1.010e+03, 9.800e+01, 2.525e+03, 1.400e+02, 1.030e+02, 1.130e+02,
       9.300e+01, 1.140e+02, 1.040e+02, 6.565e+03, 1.818e+03, 3.737e+03,
       1.350e+02, 8.080e+02, 8.200e+01, 4.444e+03, 2.323e+03, 1.160e+02,
       9.400e+01, 3.535e+03, 1.313e+03, 1.919e+03, 1.220e+02, 2.222e+03,
       8.700e+01, 7.600e+01, 9.700e+01, 1.300e+02, 1.070e+02, 6.262e+03,
       8.000e+01, 1.180e+02, 1.280e+02, 1.010e+02, 4.343e+03, 1.414e+03,
       1.320e+02, 1.060e+02, 0.000e+00, 1.717e+03, 1.120e+02, 3.939e+03,
       1.180e+02, 1.500e+02, 3.030e+02, 2.929e+03, 4.242e+03, 1.230e+02,
       1.020e+02, 1.240e+02, 1.290e+02, 4.141e+03, 1.270e+02, 1.450e+02,
       5.050e+02, 4.545e+03, 1.110e+02, 9.090e+02, 1.370e+02, 2.121e+03,
       1.090e+02, 1.330e+02, 3.838e+03, 3.333e+03, 2.424e+03, 1.111e+03,
       1.340e+02, 6.161e+03, 1.210e+02, 4.949e+03, 1.360e+02, 3.232e+03,
       4.040e+03, 2.626e+03, 1.310e+02, 3.636e+03, 1.430e+02, 1.620e+02,
       1.650e+02, 8.383e+03, 1.260e+02, 1.420e+02, 7.070e+02, 5.050e+03,
       4.040e+02, 4.747e+03, 3.434e+03, 5.353e+03, 3.131e+03, 1.720e+02,
       5.656e+03, 3.000e-01, 8.080e+03, 2.020e+02, 1.460e+02, 6.363e+03])
```

```
df['Erythrocyte Sedimentation Rate(ESR)'].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Erythrocyte Sedimentation Rate(ESR)
Non-Null Count   Dtype  
--------------   -----  
144256 non-null  float64
dtypes: float64(1)
memory usage: 46.0 MB
```

```
def clean_esr_value(value):
    if isinstance(value, str):
        value = value.strip()


        value = re.sub(r'^0+(\d+)', r'\1', value)


        if value.startswith('>'):
            value = value.replace('>', '').strip()


        if '/' in value:
            return np.nan


        try:
            num_value = float(value)
            return int(num_value) if num_value.is_integer() else num_value
        except ValueError:
            return np.nan

    return value


df['Glucose in Plasma (Fasting)'] = df['Glucose in Plasma (Fasting)'].apply(cle
```

```
df['Glucose in Plasma (Fasting)'].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Glucose in Plasma (Fasting)
Non-Null Count   Dtype
--------------   -----
523853 non-null  float64
dtypes: float64(1)
memory usage: 46.0 MB
```

```
def clean_esr_value(value):
    if isinstance(value, str):
        value = value.strip()

        value = re.sub(r'^0+(\d+)', r'\1', value)


        if value.startswith('>'):
            value = value.replace('>', '').strip()
```

```
        if '/' in value:
            return np.nan

        try:
            num_value = float(value)
            return int(num_value) if num_value.is_integer() else num_value
        except ValueError:
            return np.nan

    return value

df['Hb A1c %'] = df['Hb A1c %'].apply(clean_esr_value)
df['Mean of blood glucose ']=df['Mean of blood glucose '].apply(clean_esr_value)
df['Microalbuminuria (24 h urine)']=df['Microalbuminuria (24 h urine)'].apply(cl
df['Bilirubin (Total)']=df['Bilirubin (Total)'].apply(clean_esr_value)
df['Aspartate Aminotransferase (AST)']=df['Aspartate Aminotransferase (AST)'].ap
df['Calcium in Serum (Total)']=df['Calcium in Serum (Total)'].apply(clean_esr_va
df[ 'Rheumatoid Factor (quantitative)']=df[ 'Rheumatoid Factor (quantitative)'].
df['Sodium (Na) in Serum']=df['Sodium (Na) in Serum'].apply(clean_esr_value)
df['Bilirubin (Direct)']=df['Bilirubin (Direct)'].apply(clean_esr_value)
df['Albumin in Serum']=df['Albumin in Serum'].apply(clean_esr_value)
df['HDL Cholesterol']=df['HDL Cholesterol'].apply(clean_esr_value)
df['Platelet Count']=df['Platelet Count'].apply(clean_esr_value)
df['W.B.Cs / HPF']=df['W.B.Cs / HPF'].apply(clean_esr_value)
df['Magnesium (Mg) in Serum']=df['Magnesium (Mg) in Serum'].apply(clean_esr_valu
df['Creatinine in Serum']=df['Creatinine in Serum'].apply(clean_esr_value)
df["Triglycerides (TG) in Serum"]=df["Triglycerides (TG) in Serum"].apply(clean_
df["CRP H.S"]=df["CRP H.S"].apply(clean_esr_value)
df["Iron (Fe) in Serum"]=df["Iron (Fe) in Serum"].apply(clean_esr_value)
df["Potassium (K) in Serum"]=df["Potassium (K) in Serum"].apply(clean_esr_value)
df[ 'Free T4']=df[ 'Free T4'].apply(clean_esr_value)
df['Red cell count']=df['Red cell count'].apply(clean_esr_value)
df['T. Cholesterol/HDL']=df['T. Cholesterol/HDL'].apply(clean_esr_value)
df['Total Protein in Serum']=df['Total Protein in Serum'].apply(clean_esr_value)
df['W. B. Cs / HPF']=df['W. B. Cs / HPF'].apply(clean_esr_value)
df['RDW']=df['RDW'].apply(clean_esr_value)
```

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

	0
RESEARCH_ID	0
SAMPLE_ID	0
COLLECTYEAR	0

REGN_DATE	0
GENDER_NAME	0
AGE_YEARS	61962
AGE_DAYS	0
AGE_MONTHS	0
CITY_NAME	0
HEIGHT	0
WEIGHT	0
BMI	0
Thyroid Stimulating Hormone (TSH)	355299
Uric Acid in Serum	351487
Alanine Aminotransferase (ALT)	320871
Ferritin In Serum	749229
Blood Urea Nitrogen (BUN)	538755
Lymphocytes absolute count	899784
R. B. Cs / HPFs	0
Aspect(Urine Physical Examination) Ordinal Encoding	899794
Eosinophils absolute count	899784
Vitamin D (25 OH-Vit D -Total)	331532
C-Reactive Protein (CRP) quantitative	888923
Transferrin	898281
Red cell count	899790
Basophils absolute count	899784
Crystals(Urine Microscopic Examination :)	0
Protein(Urine Physical Examination)	899794
Colour(Urine Physical Examination)	899794
Nitrite	899794
LDL Cholesterol	0
LDL / HDL	899757
24 Hour Urine Volume (263)	899993

Hemoglobin	899781
Total Leucocytic Count	899784
Hematocrit	899781
MCV	899784
Glucose(Urine Physical Examination)	899794
Urea in Serum	687717
Prostatic Specific Antigen (PSA) Total	780763
Testosterone (Total)	776737
Alkaline Phosphatase	611435
Total Protein in Serum	618897
Estimated Glomerular Filtration Rate(eGFR)	601105
Anti CCP Abs	0
BUN/Creatinine Ratio	624730
Ketones	899794
MCHC	899784
pH(Urine Physical Examination)	899794
Amorphous Elements	899794
Blood and Haemoglobin	899794
Epithelial Cells / HPF	899794
Casts(Urine Microscopic Examination :)	0
Bilirubin	899794
Chloride in Serum	607973
Cholesterol	551734
T. Cholesterol/HDL	899758
Urobilinogen	900000
R.B.Cs / HPF	899995
Erythrocyte Sedimentation Rate(ESR)	755744
Glucose in Plasma (Fasting)	376147
Hb A1c %	549093

Mean of blood glucose	549685
Microalbuminuria (24 h urine)	890469
Bilirubin (Total)	581615
Florescence Pattern	0
Lead in blood	899281
Monocytes absolute count	899784
Consistancy	899995
Neutrophils absolute count	899784
Specific Gravity	899794
W. B. Cs / HPF	899947
Aspartate Aminotransferase (AST)	325714
Calcium in Serum (Total)	361075
Free T4	510513
Potassium (K) in Serum	593976
Albumin in Serum	468039
Iron (Fe) in Serum	746343
CRP H.S	586365
Triglycerides (TG) in Serum	556776
Rheumatoid Factor (quantitative)	899498
Platelet Count	899908
Albumin in Urine (263)	899992
MCH	899784
RDW	899787
W.B.Cs / HPF	900000
Leucocyte esterase	899794
Concentration	899995
Creatinine in Serum	306509
Sodium (Na) in Serum	597269
Bilirubin (Direct)	581936
Magnesium (Mg) in Serum	882887

Titre on Hep2 cells	897663
HDL Cholesterol	594795
Globulin in Serum	624329
Cystatin C	899992
RESEARCH_ID_int	0
GENDER_BINARY	0
CITY_NAME_ENCODED	0
BDL	0
Florescence Pattern	0
Systolic Pressure	607167
Diastolic Pressure	607167

dtype: int64

```
df.dtypes
```

	0
RESEARCH_ID	object
SAMPLE_ID	object
COLLECTYEAR	int64
REGN_DATE	datetime64[ns]
GENDER_NAME	object
AGE_YEARS	Int64
AGE_DAYS	Int64
AGE_MONTHS	Int64
CITY_NAME	category
HEIGHT	int64
WEIGHT	float64
BMI	float64
Thyroid Stimulating Hormone (TSH)	float64
Uric Acid in Serum	float64

Alanine Aminotransferase (ALT)	float64
Ferritin In Serum	float64
Blood Urea Nitrogen (BUN)	float64
Lymphocytes absolute count	float64
R. B. Cs / HPFs	float64
Aspect(Urine Physical Examination) Ordinal Encoding	float64
Eosinophils absolute count	float64
Vitamin D (25 OH-Vit D -Total)	float64
C-Reactive Protein (CRP) quantitative	float64
Transferrin	float64
Red cell count	float64
Basophils absolute count	float64
Crystals(Urine Microscopic Examination :)	float64
Protein(Urine Physical Examination)	float64
Colour(Urine Physical Examination)	float64
Nitrite	float64
LDL Cholesterol	int64
LDL / HDL	float64
24 Hour Urine Volume (263)	float64
Hemoglobin	float64
Total Leucocytic Count	float64
Hematocrit	float64
MCV	float64
Glucose(Urine Physical Examination)	float64
Urea in Serum	float64
Prostatic Specific Antigen (PSA) Total	float64
Testosterone (Total)	float64
Alkaline Phosphatase	object
Total Protein in Serum	float64
Estimated Glomerular Filtration Rate(eGFR)	float64

Anti CCP Abs	int64
BUN/Creatinine Ratio	float64
Ketones	object
MCHC	float64
pH(Urine Physical Examination)	float64
Amorphous Elements	object
Blood and Haemoglobin	object
Epithelial Cells / HPF	object
Casts(Urine Microscopic Examination :)	int64
Bilirubin	object
Chloride in Serum	float64
Cholesterol	float64
T. Cholesterol/HDL	float64
Urobilinogen	float64
R.B.Cs / HPF	object
Erythrocyte Sedimentation Rate(ESR)	float64
Glucose in Plasma (Fasting)	float64
Hb A1c %	float64
Mean of blood glucose	float64
Microalbuminuria (24 h urine)	float64
Bilirubin (Total)	float64
Florescence Pattern	object
Lead in blood	object
Monocytes absolute count	float64
Consistancy	object
Neutrophils absolute count	float64
Specific Gravity	object
W. B. Cs / HPF	float64
Aspartate Aminotransferase (AST)	float64

Calcium in Serum (Total)	object
Free T4	float64
Potassium (K) in Serum	float64
Albumin in Serum	float64
Iron (Fe) in Serum	float64
CRP H.S	float64
Triglycerides (TG) in Serum	float64
Rheumatoid Factor (quantitative)	float64
Platelet Count	float64
Albumin in Urine (263)	float64
MCH	float64
RDW	float64
W.B.Cs / HPF	float64
Leucocyte esterase	object
Concentration	object
Creatinine in Serum	object
Sodium (Na) in Serum	float64
Bilirubin (Direct)	object
Magnesium (Mg) in Serum	float64
Titre on Hep2 cells	object
HDL Cholesterol	float64
Globulin in Serum	float64
Cystatin C	object
RESEARCH_ID_int	object
GENDER_BINARY	int64
CITY_NAME_ENCODED	int64
BDL	int64
Florescence Pattern	int64
Systolic Pressure	float64
Diastolic Pressure	float64

dtype: object

```
mapping = {
    'Absent': 0,
    'Present (+)': 1,
    'Present (++)': 2,
}
df['Ketones'] = df['Ketones'].map(mapping)
```

```
df['Ketones'].unique()
```

```
array([nan,  0.,  1.,  2.])
```

```
def convert_amorphous(value):
    if pd.isna(value):
        return np.nan
    elif 'Absent' in value:
        return 0.0
    elif 'Few' in value or '(+)' in value or ' +' in value or '( + )' in value:
        return 0.25
    elif 'Many' in value or '(++)' in value or '( + + )' in value:
        return 0.5
    elif '(+++)' in value or 'High' in value:
        return 0.75
    else:
        return 0.25


df['Amorphous_Numeric'] = df['Amorphous Elements'].apply(convert_amorphous)
```

```
print(df.select_dtypes(include='object').columns)
```

```
Index(['RESEARCH_ID', 'SAMPLE_ID', 'GENDER_NAME', 'Alkaline Phosphatase',
       'Amorphous Elements', 'Blood and Haemoglobin', 'Epithelial Cells / H
       'Bilirubin', 'R.B.Cs / HPF', 'Florescence Pattern', 'Lead in blood',
       'Consistancy', 'Specific Gravity', 'Calcium in Serum (Total)',
       'Leucocyte esterase', 'Concentration', 'Creatinine in Serum',
       'Bilirubin (Direct)', 'Titre on Hep2 cells', 'Cystatin C',
       'RESEARCH_ID_int'],
      dtype='object')
```

```
import pandas as pd
import numpy as np
import re

def clean_titre_value(value):
    if isinstance(value, str):
        value = value.strip()


        match = re.search(r'1:(\d+)', value)
        if match:
            numeric_value = int(match.group(1))

            if 'Negative' in value:
                return numeric_value - 1
            return numeric_value

    return np.nan
df['Titre on Hep2 cells'] = df['Titre on Hep2 cells'].apply(clean_titre_value)
```

```
df['Titre on Hep2 cells'].unique()
```

```
array([  nan,   40.,   80.,   39.,  160.,  640., 1280.,  320., 2560.])
```

```
df['Lead in blood'].unique()
```

```
array([nan, '< 2.4', '<2.4', '4.83', 'Not detected', '< 3', '< 0.77',
       '1.00', '1.35', '8.32', '3.76', '2.73', '3.78', '3.50', '2.88',
       '< 2.5', '5.79', '20.04', '22.77', '16.71', '<1', '11.99'],
      dtype=object)
```

0.636	3
0.110	3
0.441	3
4.824	3
0.440	3
5.994	3
0.381	3
12.55	3
0.484	3
15.11	3
0.412	3
4.564	3
5.551	3
3.856	3
16.53	3

```
def convert_to_float(value):
    if pd.isna(value):
        return np.nan

    if isinstance(value, str):
        value = value.strip()

        if value.startswith('<'):
            return float(value.replace('<', '').strip())

        if value.lower() == "not detected":
            return np.nan

        try:
            return float(value)
        except ValueError:
            return np.nan

    return value


df['Lead in blood'] = df['Lead in blood'].apply(convert_to_float)
```

```
df['Lead in blood'].info()
```

```
<class 'pandas.core.series.Series'>
Index: 900000 entries, 15940 to 679524
Series name: Lead in blood
Non-Null Count  Dtype
--------------  -----
710 non-null    float64
dtypes: float64(1)
memory usage: 46.0 MB
```

```
label_encoder = LabelEncoder()
df['Florescence Pattern'] = label_encoder.fit_transform(df['Florescence Patterr
print(df[['Florescence Pattern']].head())
```

```
        Florescence Pattern
ID
15940                    18
6921                     18
23108                    18
145172                   18
168477                   18
```

```
df['Florescence Pattern'].unique()
```

```
array([18, 16,  5, 11, 12,  4,  0, 10,  9, 15,  2,  8,  1,  6,  7,  3, 14,
       17, 13])
```

14.56	3
0.512	3

```
def standardize_amorphous(value):
    if pd.isna(value):
        return np.nan

    value = value.lower().replace("(", "").replace(")", "").replace(":", "").str
    value = value.replace("  ", " ")

    if 'absent' in value:
        return "Absent"
    elif 'many' in value:
        return "Very High"
    elif 'few' in value:
        return "Very Low"
    elif '+++' in value:
        return "Highest"
    elif '++' in value:
        return "High"
    elif '+' in value:
        return "Medium"
    else:
        return "Low"

df['Amorphous Elements'] = df['Amorphous Elements'].apply(standardize_amorphous)


mapping = {
    'Absent': 0.0,
    'Very Low': 0.1,
    'Low': 0.2,
    'Medium': 0.3,
    'High': 0.5,
    'Very High': 0.75,
    'Highest': 1.0
}

df['Amorphous Elements'] = df['Amorphous Elements'].map(mapping)
```

0.485	3
7.355	3
0.051	3
0.415	3

```
def standardize_blood_hb(value):
    if pd.isna(value):
        return np.nan
    value = value.lower().replace("(", "").replace(")", "").replace(":", "").str
    value = value.replace("  ", " ")

    if 'absent' in value:
        return "Absent"
    elif 'trace' in value:
        return "Trace"
    elif '+++++' in value:
        return "Very High"
    elif '++++' in value:
        return "High"
    elif '+++' in value:
        return "Medium"
    elif '++' in value:
        return "Low"
    elif '+' in value:
        return "Very Low"
    else:
        return "Very Low"

df['Blood and Haemoglobin'] = df['Blood and Haemoglobin'].apply(standardize_bloo


mapping = {
    'Absent': 0.0,
    'Trace': 0.1,
    'Very Low': 0.2,
    'Low': 0.3,
    'Medium': 0.5,
    'High': 0.7,
    'Very High': 1.0
}
df['Blood and Haemoglobin'] = df['Blood and Haemoglobin'].map(mapping)
```

```
df[ 'Cystatin C'].unique()
```

```
array([nan, 82.5, '215', 137.3, 192.6, 206.1, '>400', 228.5],
dtype=object)
```

0.080	3
0.145	3
0.437	3
0.143	3
5.373	3
3.843	3

```
df[ 'Cystatin C'].unique()
```

```
array([nan, 82.5, '215', 137.3, 192.6, 206.1, '>400', 228.5],
      dtype=object)
```

0.528	3
0.064	3

```
bilirubin_mapping = {
    'Absent': 0.0,
    'Present (+)': 1.0,
    'Present(+)': 1.0
}

df['Bilirubin_Numeric'] = df['Bilirubin'].map(bilirubin_mapping)
df['T. Cholesterol/HDL_Numeric'] = pd.to_numeric(df['T. Cholesterol/HDL'].astyp
```

18.40	3

```
df.dtypes
```

	0
RESEARCH_ID	object
SAMPLE_ID	object
COLLECTYEAR	int64
REGN_DATE	datetime64[ns]
GENDER_NAME	object
AGE_YEARS	Int64
AGE_DAYS	Int64
AGE_MONTHS	Int64
CITY_NAME	category
HEIGHT	int64
WEIGHT	float64
BMI	float64
Thyroid Stimulating Hormone (TSH)	float64
Uric Acid in Serum	float64
Alanine Aminotransferase (ALT)	float64
Ferritin In Serum	float64
Blood Urea Nitrogen (BUN)	float64
Lymphocytes absolute count	float64

R. B. Cs / HPFs	float64
Aspect(Urine Physical Examination) Ordinal Encoding	float64
Eosinophils absolute count	float64
Vitamin D (25 OH-Vit D -Total)	float64
C-Reactive Protein (CRP) quantitative	float64
Transferrin	float64
Red cell count	float64
Basophils absolute count	float64
Crystals(Urine Microscopic Examination :)	float64
Protein(Urine Physical Examination)	float64
Colour(Urine Physical Examination)	float64
Nitrite	float64
LDL Cholesterol	int64
LDL / HDL	float64
24 Hour Urine Volume (263)	float64
Hemoglobin	float64
Total Leucocytic Count	float64
Hematocrit	float64
MCV	float64
Glucose(Urine Physical Examination)	float64
Urea in Serum	float64
Prostatic Specific Antigen (PSA) Total	float64
Testosterone (Total)	float64
Alkaline Phosphatase	object
Total Protein in Serum	float64
Estimated Glomerular Filtration Rate(eGFR)	float64
Anti CCP Abs	int64
BUN/Creatinine Ratio	float64
Ketones	float64

MCHC	float64
pH(Urine Physical Examination)	float64
Amorphous Elements	float64
Blood and Haemoglobin	float64
Epithelial Cells / HPF	object
Casts(Urine Microscopic Examination :)	int64
Bilirubin	object
Chloride in Serum	float64
Cholesterol	float64
T. Cholesterol/HDL	float64
Urobilinogen	float64
R.B.Cs / HPF	object
Erythrocyte Sedimentation Rate(ESR)	float64
Glucose in Plasma (Fasting)	float64
Hb A1c %	float64
Mean of blood glucose	float64
Microalbuminuria (24 h urine)	float64
Bilirubin (Total)	float64
Florescence Pattern	int64
Lead in blood	float64
Monocytes absolute count	float64
Consistancy	object
Neutrophils absolute count	float64
Specific Gravity	object
W. B. Cs / HPF	float64
Aspartate Aminotransferase (AST)	float64
Calcium in Serum (Total)	object
Free T4	float64
Potassium (K) in Serum	float64
Albumin in Serum	float64

Iron (Fe) in Serum	float64
CRP H.S	float64
Triglycerides (TG) in Serum	float64
Rheumatoid Factor (quantitative)	float64
Platelet Count	float64
Albumin in Urine (263)	float64
MCH	float64
RDW	float64
W.B.Cs / HPF	float64
Leucocyte esterase	object
Concentration	object
Creatinine in Serum	object
Sodium (Na) in Serum	float64
Bilirubin (Direct)	object
Magnesium (Mg) in Serum	float64
Titre on Hep2 cells	float64
HDL Cholesterol	float64
Globulin in Serum	float64
Cystatin C	object
RESEARCH_ID_int	object
GENDER_BINARY	int64
CITY_NAME_ENCODED	int64
BDL	int64
Florescence Pattern	int64
Systolic Pressure	float64
Diastolic Pressure	float64
Amorphous_Numeric	float64
Bilirubin_Numeric	float64
T. Cholesterol/HDL_Numeric	float64

dtype: object

```
df['RESEARCH_ID_int'] = pd.to_numeric(df['RESEARCH_ID_int'], errors='coerce').a
```

```
mapping = {
    'Absent': 0,
    'Present (+)': 1,
    'Present (++)': 2,
}
df['Ketones'] = df['Ketones'].map(mapping)
```

```
df.shape
```

```
(900000, 196)
```

```
df.dy
```

```
import numpy as np

def convert_amorphous(value):
    if pd.isna(value):
        return np.nan
    if isinstance(value, str):
        if 'Absent' in value:
            return 0.0
        elif 'Few' in value or '(+)' in value or '+' in value or '( + )' in val
            return 0.25
        elif 'Many' in value or '(++)' in value or '( + + )' in value:
            return 0.5
        elif '(+++)' in value or 'High' in value:
            return 0.75
    return 0.25

df['Amorphous Elements'] = df['Amorphous Elements'].apply(convert_amorphous)
```

6.235	3
0.763	3
5.530	3
3.914	3
5.621	3
0.611	3
3.789	3
6.877	3

```
def standardize_amorphous(value):
    if pd.isna(value):
        return np.nan

    if isinstance(value, str):
        value = value.lower().replace("(", "").replace(")", "").replace(":", ""
        value = value.replace("  ", " ")

        if 'absent' in value:
            return "Absent"
        elif 'many' in value:
            return "Very High"
        elif 'few' in value:
            return "Very Low"
        elif '+++' in value:
            return "Highest"
        elif '++' in value:
            return "High"
        elif '+' in value:
            return "Medium"
        else:
            return "Low"
    else:
        return value

df['Amorphous Elements'] = df['Amorphous Elements'].apply(standardize_amorphous


mapping = {
    'Absent': 0.0,
    'Very Low': 0.1,
    'Low': 0.2,
    'Medium': 0.3,
    'High': 0.5,
    'Very High': 0.75,
    'Highest': 1.0
}

df['Amorphous Elements'] = df['Amorphous Elements'].map(mapping)
```

0.522	3
0.094	3
3.460	3
0.460	3
0.03	3

```
import numpy as np
import pandas as pd

def standardize_blood_hb(value):
    if pd.isna(value):
        return np.nan
    if not isinstance(value, str):
        return value

    value = value.lower().replace("(", "").replace(")", "").replace(":", "").str
    value = value.replace("  ", " ")

    if 'absent' in value:
        return "Absent"
    elif 'trace' in value:
        return "Trace"
    elif '+++++' in value:
        return "Very High"
    elif '++++' in value:
        return "High"
    elif '+++' in value:
        return "Medium"
    elif '++' in value:
        return "Low"
    elif '+' in value:
        return "Very Low"
    else:
        return "Very Low"

df['Blood and Haemoglobin'] = df['Blood and Haemoglobin'].apply(standardize_bloo


mapping = {
    'Absent': 0.0,
    'Trace': 0.1,
    'Very Low': 0.2,
    'Low': 0.3,
    'Medium': 0.5,
    'High': 0.7,
    'Very High': 1.0
}

df['Blood and Haemoglobin'] = df['Blood and Haemoglobin'].map(mapping)
```

13.48	3
7.116	3

```
bilirubin_mapping = {
    'Absent': 0.0,
    'Present (+)': 1.0,
    'Present(+)': 1.0
}

df['Bilirubin_Numeric'] = df['Bilirubin'].map(bilirubin_mapping)
df['T. Cholesterol/HDL_Numeric'] = pd.to_numeric(df['T. Cholesterol/HDL'].astyp
```

0.564	3

```
df['T. Cholesterol/HDL'] = df['T. Cholesterol/HDL'].astype(str).str.replace(r'|
```

```
def convert_rbc_value(value):
    if pd.isna(value):
        return np.nan
    elif value == 'Absent':
        return 0
    elif '-' in value:
        nums = [float(x) for x in value.replace(' ', '').split('-')]
        return np.mean(nums)
    else:
        return float(value)

df['R.B.Cs / HPF'] = df['R.B.Cs / HPF'].apply(convert_rbc_value)
```

13.96	3

```
df['Consistancy'] = df['Consistancy'].map({
    'Semiformed': 0,
    'Formed': 1
}).astype(float)
```

```
def clean_specific_gravity(value):
    if pd.isna(value):
        return np.nan

    value = str(value).replace(',', '.')
    if value.isdigit() and len(value) == 4:
        value = f"1.{value[1:]}"

    return float(value)
df['Specific Gravity'] = df['Specific Gravity'].apply(clean_specific_gravity)
```

0.508	3
0.966	3

```
def clean_wbc_value(value):
    if pd.isna(value):
        return np.nan

    value = str(value).replace(' ', '')

    if '-' in value:
        nums = [float(x) for x in value.split('-')]
        return np.mean(nums)

    return float(value)

df['W.B.Cs / HPF'] = df['W.B.Cs / HPF'].apply(clean_wbc_value)
```

```
def clean_rdw(value):
    if pd.isna(value) or value == '.':
        return np.nan
    return float(value)


df['RDW'] = df['RDW'].apply(clean_rdw)
```

```
print(df.columns)
```

```
Index(['RESEARCH_ID', 'SAMPLE_ID', 'COLLECTYEAR', 'REGN_DATE', 'GENDER_NAME',
       'AGE_YEARS', 'AGE_DAYS', 'AGE_MONTHS', 'CITY_NAME', 'HEIGHT',
       ...
       'RESEARCH_ID_int', 'GENDER_BINARY', 'CITY_NAME_ENCODED', 'BDL',
       'Florescence Pattern ', 'Systolic Pressure', 'Diastolic Pressure',
       'Amorphous_Numeric', 'Bilirubin_Numeric', 'T. Cholesterol/HDL_Numeri',
      dtype='object', length=106)
```

```
print(df.columns.tolist())
```

```
['RESEARCH_ID', 'SAMPLE_ID', 'COLLECTYEAR', 'REGN_DATE', 'GENDER_NAME', 'AG
```

```
print('RDW' in df.columns)
```

```
True
```

```
mapping = {
    'Absent': 0.0,
    'Present (+)': 1.0,
    'Present(+)': 1.0,
    'Present (++)': 2.0,
    'Present(++)': 2.0,
    'Present (+++)': 3.0,
    'Trace': 0.5
}

df['Leucocyte esterase'] = df['Leucocyte esterase'].map(mapping).astype(float)
```

12.45	2

```
df['Concentration'] = df['Concentration'].replace({
    'Negative': 0.0,
    '.': np.nan
}).astype(float)
```

```
<ipython-input-271-f79ef632a17f>:1: FutureWarning: Downcasting behavior in
  df['Concentration'] = df['Concentration'].replace({
```

12.22	2
13.56	2

```
def clean_cystatin(value):
    if pd.isna(value):
        return np.nan
    value = str(value).strip()
    if value.startswith('>'):
        return 399.9
    return float(value)

df['Cystatin C'] = df['Cystatin C'].apply(clean_cystatin)
```

6.826	2

```
df.shape
```

```
(900000, 100)
```

7.485	2
7.666	2

```
df.to_csv("processed2.3.csv", index=False, encoding="utf-8-sig", float_format='
```

<0.02	2
6.978	2
0.511	2
5.418	2
0.633	2
5.031	2

5.031	2
0.728	2
2.458	2
4.415	2
4.717	2
6.153	2
0.676	2
3.452	2
6.519	2