In [1]:
```
!pip install folium
```

```
Requirement already satisfied: folium in c:\users\user\anaconda3\lib\site-packages (0.17.0)
Requirement already satisfied: branca>=0.6.0 in c:\users\user\anaconda3\lib\site-packages (from folium) (0.7.2)
Requirement already satisfied: jinja2>=2.9 in c:\users\user\anaconda3\lib\site-packages (from folium) (3.1.3)
Requirement already satisfied: numpy in c:\users\user\anaconda3\lib\site-packages (from folium) (1.26.4)
Requirement already satisfied: requests in c:\users\user\anaconda3\lib\site-packages (from folium) (2.31.0)
Requirement already satisfied: xyzservices in c:\users\user\anaconda3\lib\site-packages (from folium) (2022.9.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\user\anaconda3\lib\site-packages (from jinja2>=2.9->foliu
m) (2.1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\user\anaconda3\lib\site-packages (from requests->
folium) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\user\anaconda3\lib\site-packages (from requests->folium) (3.
4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\user\anaconda3\lib\site-packages (from requests->foliu
m) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\user\anaconda3\lib\site-packages (from requests->foliu
m) (2024.2.2)
```

In [2]:
```
# importing relevant libraries
import pandas as pd
import folium
from IPython.display import display
```

In [3]:
```
# loading the training set variables (indipendent and dependent variables)
x_train = pd.read_csv('x_train.csv')
y_train = pd.read_csv('y_train.csv')
```

In [4]:
```
x_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59400 entries, 0 to 59399
Data columns (total 40 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   id                     59400 non-null  int64  
 1   amount_tsh             59400 non-null  float64
 2   date_recorded          59400 non-null  object 
 3   funder                 55763 non-null  object 
 4   gps_height             59400 non-null  int64  
 5   installer              55745 non-null  object 
 6   longitude              59400 non-null  float64
 7   latitude               59400 non-null  float64
 8   wpt_name               59398 non-null  object 
 9   num_private            59400 non-null  int64  
 10  basin                  59400 non-null  object 
 11  subvillage             59029 non-null  object 
 12  region                 59400 non-null  object 
 13  region_code            59400 non-null  int64  
 14  district_code          59400 non-null  int64  
 15  lga                    59400 non-null  object 
 16  ward                   59400 non-null  object 
 17  population             59400 non-null  int64  
 18  public_meeting         56066 non-null  object 
 19  recorded_by            59400 non-null  object 
 20  scheme_management      55522 non-null  object 
 21  scheme_name            30590 non-null  object 
 22  permit                 56344 non-null  object 
 23  construction_year      59400 non-null  int64  
 24  extraction_type        59400 non-null  object 
 25  extraction_type_group  59400 non-null  object 
 26  extraction_type_class  59400 non-null  object 
 27  management             59400 non-null  object 
 28  management_group       59400 non-null  object 
 29  payment                59400 non-null  object 
 30  payment_type           59400 non-null  object 
 31  water_quality          59400 non-null  object 
 32  quality_group          59400 non-null  object 
 33  quantity               59400 non-null  object 
 34  quantity_group         59400 non-null  object 
 35  source                 59400 non-null  object 
 36  source_type            59400 non-null  object 
 37  source_class           59400 non-null  object 
 38  waterpoint_type        59400 non-null  object 
 39  waterpoint_type_group  59400 non-null  object 
dtypes: float64(3), int64(7), object(30)
memory usage: 18.1+ MB
```

The dataset has 59400 rows and 40 columns. Some columns are of type int while others are float. Majority are objects. Some columns;'funder', 'installer', 'public_meeting','scheme_management', and 'scheme_name' have missing values,

In [5]:
```
# previewing the data
x_train.head()
```

Out[5]:

	id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_name	num_private	...	payment_type	water_qua
0	69572	6000.0	2011-03-14	Roman	1390	Roman	34.938093	-9.856322	none	0	...	annually	
1	8776	0.0	2013-03-06	Grumeti	1399	GRUMETI	34.698766	-2.147466	Zahanati	0	...	never pay	
2	34310	25.0	2013-02-25	Lottery Club	686	World vision	37.460664	-3.821329	Kwa Mahundi	0	...	per bucket	
3	67743	0.0	2013-01-28	Unicef	263	UNICEF	38.486161	-11.155298	Zahanati Ya Nanyumbu	0	...	never pay	
4	19728	0.0	2011-07-13	Action In A	0	Artisan	31.130847	-1.825359	Shuleni	0	...	never pay	

5 rows × 40 columns

In [6]:
```
# combining the data sets
df=pd.concat([x_train,y_train],axis=1)
df.head()
```

Out[6]:

	id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_name	num_private	...	quality_group	quantity
0	69572	6000.0	2011-03-14	Roman	1390	Roman	34.938093	-9.856322	none	0	...	good	enough
1	8776	0.0	2013-03-06	Grumeti	1399	GRUMETI	34.698766	-2.147466	Zahanati	0	...	good	insufficient
2	34310	25.0	2013-02-25	Lottery Club	686	World vision	37.460664	-3.821329	Kwa Mahundi	0	...	good	enough
3	67743	0.0	2013-01-28	Unicef	263	UNICEF	38.486161	-11.155298	Zahanati Ya Nanyumbu	0	...	good	dry
4	19728	0.0	2011-07-13	Action In A	0	Artisan	31.130847	-1.825359	Shuleni	0	...	good	seasonal

5 rows × 42 columns

Exploring categorical variables

In [7]:
```
# finding the categorical variables

df.select_dtypes(include=['object']).columns
```

Out[7]:
```
Index(['date_recorded', 'funder', 'installer', 'wpt_name', 'basin',
       'subvillage', 'region', 'lga', 'ward', 'public_meeting', 'recorded_by',
       'scheme_management', 'scheme_name', 'permit', 'extraction_type',
       'extraction_type_group', 'extraction_type_class', 'management',
       'management_group', 'payment', 'payment_type', 'water_quality',
       'quality_group', 'quantity', 'quantity_group', 'source', 'source_type',
       'source_class', 'waterpoint_type', 'waterpoint_type_group',
       'status_group'],
      dtype='object')
```

In [8]:
```
#Count the occurrences of each unique combination of 'funder' and 'installer'
df[['funder','installer']].value_counts()
```

Out[8]:
```
funder                  installer
Government Of Tanzania  DWE                     4254
                        Government              1607
Hesawa                  DWE                     1296
Danida                  DANIDA                  1046
Rwssp                   DWE                      914
                                                ...
Masai Land              MASAI LAND                 1
Maseka Community        Maseka community           1
Masese                  Masese                     1
Mashaka                 DWE                        1
Zingibali Secondary     Zingibali Secondary        1
Name: count, Length: 3697, dtype: int64
```

In [9]:
```
#Count the occurrences of each unique combination of 'scheme management' and 'scheme name'
df[['scheme_management', 'scheme_name']].value_counts()
```

Out[9]:
```
scheme_management  scheme_name
VWC                K                             571
WUA                Chalinze wate                 404
VWC                DANIDA                        378
                   M                             331
                   Borehole                      285
                                                ...
                   Mradi wa maji wa Maposeni       1
                   Mradi wa maji wa Kilagano       1
                   Mradi wa maji wa Kakola         1
                   Mradi wa maji wa  Wino          1
Water authority    water supply at Kalebejo        1
Name: count, Length: 3069, dtype: int64
```

In [10]:
```
#Count the occurrences of each unique combination of 'payment' and 'payment_type'
df[['payment', 'payment_type']].value_counts()
```

Out[10]:
```
payment                payment_type
never pay              never pay       25348
pay per bucket         per bucket       8985
pay monthly            monthly          8300
unknown                unknown          8157
pay when scheme fails  on failure       3914
pay annually           annually         3642
other                  other            1054
Name: count, dtype: int64
```

In [11]:
```
#Count the occurrences of each unique combination of 'management' and 'management_group'
df[['management', 'management_group']].value_counts()
```

Out[11]:
```
management        management_group
vwc               user-group          40507
wug               user-group           6515
water board       user-group           2933
wua               user-group           2535
private operator  commercial           1971
parastatal        parastatal           1768
water authority   commercial            904
other             other                 844
company           commercial            685
unknown           unknown               561
other - school    other                  99
trust             commercial             78
Name: count, dtype: int64
```

In [12]:
```
#Count the occurrences of each unique combination of 'water_quality' and 'quality_group'
df[['water_quality', 'quality_group']].value_counts()
```

Out[12]:
```
water_quality       quality_group
soft                good             50818
salty               salty             4856
unknown             unknown           1876
milky               milky              804
coloured            colored            490
salty abandoned     salty              339
fluoride            fluoride           200
fluoride abandoned  fluoride            17
Name: count, dtype: int64
```

In [13]:
```
#Count the occurrences of each unique combination of 'quantity' and 'quantity_group'
df[['quantity', 'quantity_group']].value_counts()
```

Out[13]:
```
quantity      quantity_group
enough        enough            33186
insufficient  insufficient      15129
dry           dry                6246
seasonal      seasonal           4050
unknown       unknown             789
Name: count, dtype: int64
```

In [14]:
```
#Count the occurrences of each unique combination of 'source_class' and 'source'and 'source_type'
df[['source_class', 'source_type','source']].value_counts()
```

Out[14]:
```
source_class  source_type           source
groundwater   spring                spring                  17021
              shallow well          shallow well            16824
              borehole              machine dbh             11075
surface       river/lake            river                    9612
              rainwater harvesting  rainwater harvesting     2295
groundwater   borehole              hand dtw                  874
surface       river/lake            lake                      765
              dam                   dam                       656
unknown       other                 other                     212
                                    unknown                    66
Name: count, dtype: int64
```

In [15]:
```
#Count the occurrences of each unique combination of 'waterpoint_type_group' and 'waterpoint_type'
df[['waterpoint_type_group', 'waterpoint_type']].value_counts()
```

Out[15]:
```
waterpoint_type_group  waterpoint_type
communal standpipe     communal standpipe             28522
hand pump              hand pump                      17488
other                  other                           6380
communal standpipe     communal standpipe multiple     6103
improved spring        improved spring                  784
cattle trough          cattle trough                    116
dam                    dam                                7
Name: count, dtype: int64
```

In [16]:
```
#Count the occurrences of each unique combination of 'extraction_type_group' and 'extraction_type'
df[['extraction_type_group', 'extraction_type']].value_counts()
```

Out[16]:
```
extraction_type_group  extraction_type          
gravity                gravity                      26780
nira/tanira            nira/tanira                   8154
other                  other                         6430
submersible            submersible                   4764
swn 80                 swn 80                        3670
mono                   mono                          2865
india mark ii          india mark ii                 2400
afridev                afridev                       1770
submersible            ksb                           1415
rope pump              other - rope pump              451
other handpump         other - swn 81                 229
wind-powered           windmill                       117
india mark iii         india mark iii                  98
other motorpump        cemo                            90
other handpump         other - play pump               85
                       walimi                          48
other motorpump        climax                          32
other handpump         other - mkulima/shinyanga        2
Name: count, dtype: int64
```

categorical columns to drop due to redundancy

1. Date_recorded- we have year of construction with similar information
2. Funder which has similar information with installer
3. lga, Ward, sub_village to keep the region column
4. Scheme_name due to its high value of unique and missing values
5. payment
6. quality_group
7. extraction_type
8. source
9. source_type
10. wpt_name- has many unique and missing values
11. waterpoint_type
12. management

Analysing numerical variables

In [17]:
```
df.select_dtypes(include=['float', 'integer']).columns
```

Out[17]:
```
Index(['id', 'amount_tsh', 'gps_height', 'longitude', 'latitude',
       'num_private', 'region_code', 'district_code', 'population',
       'construction_year', 'id'],
      dtype='object')
```

Columns to drop

1. region_code which is a duplicate of region
2. district_code which is similar to region
3. num_private, most values are zeros hence lack variability

Dealing with missing values

In [18]:
```
columns_to_drop = ['id','recorded_by','date_recorded', 'funder', 'wpt_name', 'subvillage', 'lga','ward', 'scheme_name'
                   'management', 'payment', 'quality_group', 'quantity', 'source',
                   'source_type', 'waterpoint_type', 'num_private', 'region_code', 'district_code']

df = df.drop(columns_to_drop, axis=1)  # Dropping columns and reassigning to df
df.head()  # Displaying the first few rows
```

Out[18]:

latitude	basin	region	population	public_meeting	scheme_management	...	construction_year	extraction_type	extraction_type_class	mana
-9.856322	Lake Nyasa	Iringa	109	True	VWC	...	1999	gravity	gravity	
-2.147466	Lake Victoria	Mara	280	NaN	Other	...	2010	gravity	gravity	
-3.821329	Pangani	Manyara	250	True	VWC	...	2009	gravity	gravity	
-11.155298	Ruvuma / Southern Coast	Mtwara	58	True	VWC	...	1986	submersible	submersible	
-1.825359	Lake Victoria	Kagera	0	True	NaN	...	0	gravity	gravity	

In [19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59400 entries, 0 to 59399
Data columns (total 21 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   amount_tsh             59400 non-null  float64
 1   gps_height             59400 non-null  int64  
 2   installer              55745 non-null  object 
 3   longitude              59400 non-null  float64
 4   latitude               59400 non-null  float64
 5   basin                  59400 non-null  object 
 6   region                 59400 non-null  object 
 7   population             59400 non-null  int64  
 8   public_meeting         56066 non-null  object 
 9   scheme_management      55522 non-null  object 
 10  permit                 56344 non-null  object 
 11  construction_year      59400 non-null  int64  
 12  extraction_type        59400 non-null  object 
 13  extraction_type_class  59400 non-null  object 
 14  management_group       59400 non-null  object 
 15  payment_type           59400 non-null  object 
 16  water_quality          59400 non-null  object 
 17  quantity_group         59400 non-null  object 
 18  source_class           59400 non-null  object 
 19  waterpoint_type_group  59400 non-null  object 
 20  status_group           59400 non-null  object 
dtypes: float64(3), int64(3), object(15)
memory usage: 9.5+ MB
```

In [20]:
```
df.isnull().mean().sort_values (ascending =False)
```

Out[20]:
```
scheme_management        0.065286
installer                0.061532
public_meeting           0.056128
permit                   0.051448
extraction_type          0.000000
waterpoint_type_group    0.000000
source_class             0.000000
quantity_group           0.000000
water_quality            0.000000
payment_type             0.000000
management_group         0.000000
extraction_type_class    0.000000
amount_tsh               0.000000
construction_year        0.000000
gps_height               0.000000
population               0.000000
region                   0.000000
basin                    0.000000
latitude                 0.000000
longitude                0.000000
status_group             0.000000
dtype: float64
```

In [24]:
```
# unique values in 'scheme_management' column
df['scheme_management'].value_counts()
```

Out[24]:
```
scheme_management
VWC                 36793
WUG                  5206
Water authority      3153
WUA                  2883
Water Board          2748
Parastatal           1680
Private operator     1063
Company              1061
Other                 766
SWC                    97
Trust                  72
Name: count, dtype: int64
```

In [25]:
```
# unique values in 'installer' column
df['installer'].value_counts()
```

Out[25]:
```
installer
DWE                17402
Government          1825
RWE                 1206
Commu               1060
DANIDA              1050
                   ...
Wizara  ya maji        1
TWESS                  1
Nasan workers          1
R                      1
SELEPTA                1
Name: count, Length: 2145, dtype: int64
```

In [26]:
```
# unique values in 'public_meeting' column
df['public_meeting'].value_counts()
```

Out[26]:
```
public_meeting
True     51011
False     5055
Name: count, dtype: int64
```

In [27]:
```
# unique values in 'permit' column
df['permit'].value_counts()
```

Out[27]:
```
permit
True     38852
False    17492
Name: count, dtype: int64
```

Decision on missing values

1.replace the missing value with the mode ['yes'] on permit an public meeting columns

2. Drop the istaller and scheme_management column. It is difficultt to impute

In [29]:
```
df.dropna(subset=['installer','scheme_management'], inplace=True)
```

In [30]:
```
columns_to_impute = ['public_meeting', 'permit']

for column in columns_to_impute:
    df[column].fillna(df[column].mode()[0], inplace=True)
```

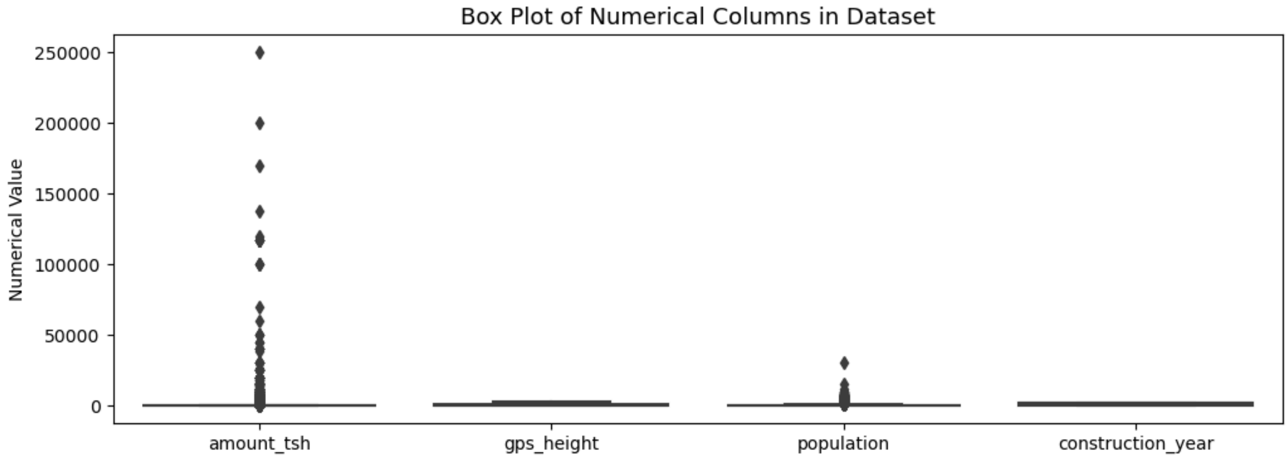
In [31]:
```
df.isnull().sum()
```

Out[31]:
```
amount_tsh               0
gps_height               0
installer                0
longitude                0
latitude                 0
basin                    0
region                   0
population               0
public_meeting           0
scheme_management        0
permit                   0
construction_year        0
extraction_type          0
extraction_type_class    0
management_group         0
payment_type             0
water_quality            0
quantity_group           0
source_class             0
waterpoint_type_group    0
status_group             0
dtype: int64
```

In [33]:
```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn import datasets, linear_model, metrics
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay, f1_score, recall_score, p

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import warnings
warnings.filterwarnings('ignore')
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from imblearn.over_sampling import SMOTE
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline
from imblearn.pipeline import Pipeline
from imblearn.over_sampling import SMOTE
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay

from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay
from imblearn.pipeline import Pipeline
from imblearn.over_sampling import SMOTE
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier
```

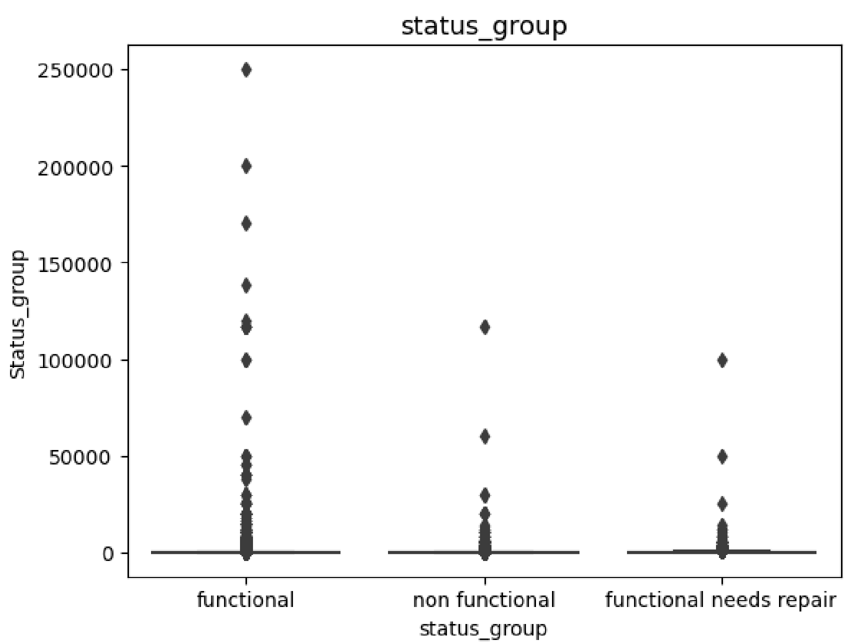
In [34]:
```
numerical_cols = ['amount_tsh', 'gps_height', 'population', 'construction_year']
plt.figure(figsize=(12, 4))
sns.boxplot(data=[df[col] for col in numerical_cols])
plt.title("Box Plot of Numerical Columns in Dataset", fontsize=13)
plt.ylabel("Numerical Value")
plt.xticks(range(0,4), numerical_cols)
```

Out[34]:
```
([<matplotlib.axis.XTick at 0x20e875bcc50>,
  <matplotlib.axis.XTick at 0x20e875b2890>,
  <matplotlib.axis.XTick at 0x20e8762a650>,
  <matplotlib.axis.XTick at 0x20e8763b910>],
 [Text(0, 0, 'amount_tsh'),
  Text(1, 0, 'gps_height'),
  Text(2, 0, 'population'),
  Text(3, 0, 'construction_year')])
```



In [35]:
```
sns.boxplot(y='amount_tsh', x="status_group", data=df)
plt.title("status_group", fontsize=13)
plt.ylabel("amount -TSH ")
plt.ylabel("Status_group")
```

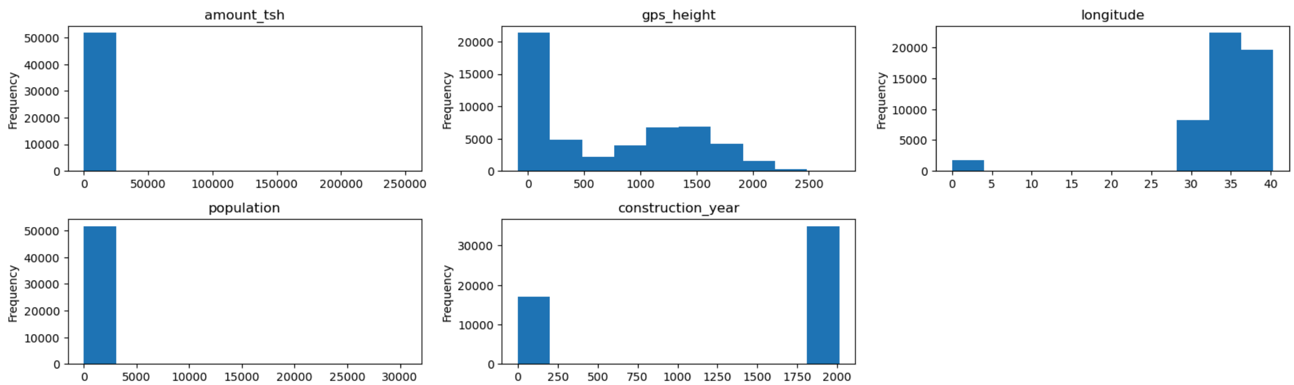
Out[35]:
```
Text(0, 0.5, 'Status_group')
```



Based on the provided figures, it is advisable not to remove outliers in the amount_tsh column as they likely represent real variations in water availability across different wells. These outliers are present in all status_group categories (functional, non-functional, and functional needs repair), suggesting they carry significant insights into the conditions and performance of the wells. Removing them could result in a loss of valuable information and an incomplete understanding of the dataset. Instead, transformations such as log scaling can mitigate the impact of outliers while preserving the integrity and richness of the data, ensuring robust and comprehensive analysis.

checking normal distribution in continous columns

In [36]:
```
# Histogram of continuous variables
continuous = ['amount_tsh','gps_height','longitude', 'population','construction_year']
fig = plt.figure(figsize=(16, 7))
for i, col in enumerate(continuous):
    ax = plt.subplot(3, 3, i+1)
    df[col].plot(kind='hist', ax=ax, title=col)
plt.tight_layout()
```



label encode and onehot encoder

In [37]:
```
label_mapping = {False: 0, True: 1}
df["public_meeting"] = df["public_meeting"].map(label_mapping)
df["permit"] = df["permit"].map(label_mapping)
```

In [38]:
```
label_mapping_s = {"non functional": 0, "functional needs repair": 1, "functional": 2}
df["status_group"] = df["status_group"].replace(label_mapping_s)
```

In [39]:
```
columns_encode=df[["installer", "basin", "region", "scheme_management", "management_group",
         "extraction_type_class", "payment_type", 'water_quality',"quantity_group",
         "source_class", "waterpoint_type_group"]]
```

In [40]:
```
columns_to_encode = ["installer", "basin", "region", "scheme_management",
                     "management_group", "extraction_type_class", "payment_type",
                     'water_quality', "quantity_group", "source_class",
                     "waterpoint_type_group"]

# Create dummy variables for all specified columns
df_encoded = pd.get_dummies(df, columns=columns_to_encode, drop_first=True, dtype=int)
```

In [41]:
```
df_store=df_encoded.copy()
df_encoded.head()
```

Out[41]:

lation	public_meeting	permit	construction_year	extraction_type	status_group	...	quantity_group_insufficient	quantity_group_seasonal	quantity_grc
109	1	0	1999	gravity	2	...	0	0	
280	1	1	2010	gravity	2	...	1	0	
250	1	1	2009	gravity	2	...	0	0	
58	1	1	1986	submersible	0	...	0	0	
1	1	1	2009	submersible	2	...	0	0	

In [50]:
```
df_store.drop(['extraction_type'],axis=1)
```

Out[50]:

	amount_tsh	gps_height	longitude	latitude	population	public_meeting	permit	construction_year	status_group	installer_0	...	quantity
0	6000.0	1390	34.938093	-9.856322	109	1	0	1999	2	0	...	
1	0.0	1399	34.698766	-2.147466	280	1	1	2010	2	0	...	
2	25.0	686	37.460664	-3.821329	250	1	1	2009	2	0	...	
3	0.0	263	38.486161	-11.155298	58	1	1	1986	0	0	...	
5	20.0	0	39.172796	-4.765587	1	1	1	2009	2	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
59394	500.0	351	37.634053	-6.124830	89	1	1	2007	0	0	...	
59395	10.0	1210	37.169807	-3.253847	125	1	1	1999	2	0	...	
59396	4700.0	1212	35.249991	-9.070629	56	1	1	1996	2	0	...	
59398	0.0	0	35.861315	-6.378573	0	1	1	0	2	0	...	
59399	0.0	191	38.104048	-6.747464	150	1	1	2002	2	0	...	

51926 rows × 2083 columns

standard scaler

In [53]:
```
scaled_columns=["amount_tsh", "gps_height", "population"]

# Initialize the StandardScaler
scaler = StandardScaler()

# Fit and transform the specified columns
df_encoded[scaled_columns] = scaler.fit_transform(df_encoded[scaled_columns])
```

Reingineering or data transformation -¶
Transforming the status_group column

2 = functional water points ,

1 = functional but needs repair water points,

0 = non-functinal water points

We collect functional and functional but needs help target together and make them 1, non-functional is 0.

In [51]:
```
df_encoded["status_group"] = df_encoded["status_group"].apply(lambda x: 1 if x in [1, 2] else 0)
```

In [54]:
```
df_encoded.corr()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[54], line 1
----> 1 df_encoded.corr()

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:10704, in DataFrame.corr(self, method, min_periods, numeric_o
nly)
  10702 cols = data.columns
  10703 idx = cols.copy()
> 10704 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  10706 if method == "pearson":
  10707     correl = libalgos.nancorr(mat, minp=min_periods)

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:1889, in DataFrame.to_numpy(self, dtype, copy, na_value)
   1887 if dtype is not None:
   1888     dtype = np.dtype(dtype)
-> 1889 result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
   1890 if result.dtype is not dtype:
   1891     result = np.array(result, dtype=dtype, copy=False)

File ~\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:1656, in BlockManager.as_array(self, dtype, cop
y, na_value)
   1654         arr.flags.writeable = False
   1655 else:
-> 1656     arr = self._interleave(dtype=dtype, na_value=na_value)
   1657     # The underlying data was copied within _interleave, so no need
   1658     # to further copy if copy=True or setting na_value
   1660 if na_value is lib.no_default:

File ~\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:1715, in BlockManager._interleave(self, dtype, n
a_value)
   1713     else:
   1714         arr = blk.get_values(dtype)
-> 1715     result[rl.indexer] = arr
   1716     itemmask[rl.indexer] = 1
   1718 if not itemmask.all():

ValueError: could not convert string to float: 'gravity'
```

In [ ]: