TANZANIA WATER WELL STATUS AND DISTRIBUTION: A PREDICTIVE ANALYSIS

Tanzania, officially the United Republic of Tanzania, is a country in East Africa within the African Great Lakes region. It boasts a rich tapestry of geographical features, bordering Uganda to the northwest; Kenya to the northeast; the Indian Ocean to the east; Mozambique and Malawi to the south; Zambia to the southwest; and Rwanda, Burundi, and the Democratic Republic of the Congo to the west. Mount Kilimanjaro, Africa's highest mountain, stands proudly in northeastern Tanzania. With a population of nearly 62 million according to the 2022 national census, Tanzania is the most populous country located entirely south of the equator.

Beyond its size and geographical diversity, Tanzania is a land of cultural richness. The country is home to over 120 ethnic groups, each with its own unique traditions and languages. Swahili, a Bantu language with Arabic influences, serves as the national language and lingua franca, uniting the people. Tanzania is religiously diverse as well, with Christianity being the largest religion, followed by Islam and traditional African religions.

Tanzania is a haven for wildlife enthusiasts. Its vast wilderness areas include the world-renowned Serengeti National Park, famous for the annual Great Migration of millions of wildebeest and zebra. The Ngorongoro Crater, a UNESCO World Heritage Site, shelters a unique ecosystem teeming with wildlife. Offshore, the spice island of Zanzibar offers a glimpse into Tanzania's historical ties with the Arab world, while Mafia Island boasts a marine park known for its stunning coral reefs and gentle giants, the whale sharks.

Access to clean water varies greatly across Tanzania. While urban areas have seen increased access to piped water, a significant portion of the rural population relies on groundwater sources. Many communities depend on wells, boreholes, and springs for their daily needs. The uneven distribution of water resources, coupled with seasonal variations in rainfall, can create challenges, particularly in arid and semi-arid regions. Government initiatives and NGO efforts are working to improve water infrastructure and expand access to clean water for all Tanzanians.

Research Statement

This project aims to delve deeper into the water issues faced by Tanzania. A key challenge lies in ensuring the functionality of water wells, which are vital for many rural communities. By leveraging machine learning, we propose the development of an algorithm that can predict the functionality of a water well. This algorithm will be designed to classify wells into three

categories: functional, non-functional, and functional but requiring repair. This information can

Research Objectives

In an effort to enhance the management and sustainability of water resources, this research focuses on two key objectives.

1. To find out the geographical distribution of the three classes of water pumps, to help inform which regions need more attention in terms of repair, maintenance, and replacement.
2. To build a Machine Learning classifier that will predict the condition of a water well (functional, functional-but-needs-repair, and non-functional), using data such as the kind of pump, when it was installed, the installer, the region, and so on.

Research questions

To guide this research, two central questions are posed.

1. What is the geographical distribution of the three classes of water pumps, and which regions need more attention for repair, maintenance, and replacement?
2. How can a Machine Learning classifier be developed to accurately predict the condition of water wells using data such as pump type, installation date, installer, and region?

Dataset description

To achive the above objectives the dataset used in this analysis is sourced from the DrivenData competition titled "Pump it Up: Data Mining the Water Table,". This focuses on predicting the functionality status of waterpoints in Tanzania, aiming to improve water resource management.

The primary components of the dataset are:

> X_train.csv 📂: Contains the independent variables for the training set, which are the features used to train the machine learning model.
>
> y_train.csv 📂: Includes the dependent variable 'status_group' for each entry in the training set, indicating the operational status of the water pumps.

For testing and submitting predictions, the dataset includes:

> Test set values 🧪: This file holds the independent variables that require predictions, similar to X_test in machine learning projects.

In [1]:
```
import pandas as pd
import folium
from IPython.display import display
```

In [2]:
```
# Load the independent variables for the training set from X_train.csv
# These variables will be used to train the machine learning model
X_train =pd.read_csv("X_train.csv")
y_train=pd.read_csv("y_train.csv")
```

In [3]:
```
# Displaying the summary information of the dataframe including the datatype of
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
```

This DataFrame contains 59,400 entries and 40 columns, each representing a different attribute of the data. The columns include various data types: integers (int64), floating-point numbers (float64), and strings (object). Key columns include 'id', 'amount_tsh' (total static head), 'date_recorded', 'funder', 'gps_height', 'installer', 'longitude', 'latitude', 'wpt_name' (waterpoint name), 'population', 'public_meeting', 'scheme_management', and 'scheme_name'. Some

columns have missing values, such as 'funder', 'installer', 'public_meeting', 'scheme_management', and 'scheme_name', indicated by the non-null count being less than 59,400. The data appears to be related to water points, likely including information on their location, management, and operational details.

In [4]:
```
# Calculate and display the number of missing values in each column of the Data

X_train.isnull().sum()
```

Out[4]:
```
id                         0
amount_tsh                 0
date_recorded              0
funder                  3637
gps_height                 0
installer               3655
longitude                  0
latitude                   0
wpt_name                   2
num_private                0
basin                      0
subvillage               371
region                     0
region_code                0
district_code              0
lga                        0
ward                       0
population                 0
public_meeting          3334
recorded_by                0
scheme_management       3878
scheme_name            28810
permit                  3056
construction_year          0
extraction_type            0
extraction_type_group      0
extraction_type_class      0
management                 0
management_group           0
payment                    0
payment_type               0
water_quality              0
quality_group              0
quantity                   0
quantity_group             0
source                     0
source_type                0
source_class               0
waterpoint_type            0
waterpoint_type_group      0
dtype: int64
```

The DataFrame has several columns with missing values. The columns 'funder' and 'installer' have 3,637 and 3,655 missing values, respectively. 'Wpt_name' is missing in 2 instances, while 'subvillage' has 371 missing values. The 'public_meeting' column is missing 3,334 values, and 'scheme_management' has 3,878 missing entries. The 'scheme_name' column has the most

missing values, with 28,810 entries absent. Additionally, the 'permit' column is missing 3,056 values. All other columns, including 'id', 'amount_tsh', 'date_recorded', 'gps_height', 'longitude',

In [5]:
```
# Display the shape of the y_train DataFrame to understand its dimensions
y_train.shape
```

Out[5]:
```
(59400, 2)
```

In [6]:
```
# Display the first few rows of the X_train DataFrame to preview the data
X_train.head()
```

Out[6]:

	id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt
0	69572	6000.0	2011-03-14	Roman	1390	Roman	34.938093	-9.856322	
1	8776	0.0	2013-03-06	Grumeti	1399	GRUMETI	34.698766	-2.147466	Z
2	34310	25.0	2013-02-25	Lottery Club	686	World vision	37.460664	-3.821329	M
3	67743	0.0	2013-01-28	Unicef	263	UNICEF	38.486161	-11.155298	Z Nar
4	19728	0.0	2011-07-13	Action In A	0	Artisan	31.130847	-1.825359	

5 rows × 40 columns

In [7]:
```
#displaying the first 5 columns of the y_train dataset
y_train.head()
```

Out[7]:

	id	status_group
0	69572	functional
1	8776	functional
2	34310	functional
3	67743	non functional
4	19728	functional

In [8]:
```
# Combine the X_train and y_train DataFrames along columns to create a single D
df = pd.concat([X_train , y_train ], axis=1)
# Display the first five rows of concatinated data
df.head()
```

Out[8]:

	id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt
0	69572	6000.0	2011-03-14	Roman	1390	Roman	34.938093	-9.856322	
1	8776	0.0	2013-03-06	Grumeti	1399	GRUMETI	34.698766	-2.147466	Z
2	34310	25.0	2013-02-25	Lottery Club	686	World vision	37.460664	-3.821329	M
3	67743	0.0	2013-01-28	Unicef	263	UNICEF	38.486161	-11.155298	Z Nar
4	19728	0.0	2011-07-13	Action In A	0	Artisan	31.130847	-1.825359	

5 rows × 42 columns

In [9]:
```
#Checking for missing values
df.isnull().sum()
```

Out[9]:
```
id                         0
amount_tsh                 0
date_recorded              0
funder                  3637
gps_height                 0
installer               3655
longitude                  0
latitude                   0
wpt_name                   2
num_private                0
basin                      0
subvillage               371
region                     0
region_code                0
district_code              0
lga                        0
ward                       0
population                 0
public_meeting          3334
recorded_by                0
scheme_management       3878
scheme_name            28810
permit                  3056
construction_year          0
extraction_type            0
extraction_type_group      0
extraction_type_class      0
management                 0
management_group           0
payment                    0
payment_type               0
water_quality              0
quality_group              0
quantity                   0
quantity_group             0
source                     0
source_type                0
source_class               0
waterpoint_type            0
waterpoint_type_group      0
id                         0
status_group               0
dtype: int64
```

In [10]:
```
# Displaying the summary information of the dataframe including the datatype of

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59400 entries, 0 to 59399
Data columns (total 42 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 
 40  id                     59400 non-null  int64  
 41  status_group           59400 non-null  object 
dtypes: float64(3), int64(8), object(31)
memory usage: 19.0+ MB
```

Analyzing Categorical Variables for Redundancy 🧐

This section is designed to explore and analyze the categorical variables in the dataset. By examining how these variables are grouped and their value counts, the aim is to identify potential duplications among the columns. The dataset includes a range of categorical variables such as 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, and status_group. This comprehensive review helps pinpoint redundant data, ensuring more efficient and accurate analysis 🗂️🔍.

In [11]:
```
# Check categorical columns
df.select_dtypes(include=['object']).columns
```

Out[11]:
```
Index(['date_recorded', 'funder', 'installer', 'wpt_name', 'basin',
       'subvillage', 'region', 'lga', 'ward', 'public_meeting', 'recorded_b
y',
       '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_typ
e',
       'source_class', 'waterpoint_type', 'waterpoint_type_group',
       'status_group'],
      dtype='object')
```

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

Out[12]:
```
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 [13]:
```
# Count the occurrences of each unique combination of 'management_group' and 'm
df[['management_group', 'management']].value_counts()
```

Out[13]:
```
management_group  management
user-group        vwc                40507
                  wug                 6515
                  water board         2933
                  wua                 2535
commercial        private operator    1971
parastatal        parastatal          1768
commercial        water authority      904
other             other                844
commercial        company              685
unknown           unknown              561
other             other - school        99
commercial        trust                 78
Name: count, dtype: int64
```

In [14]:
```
# Count the occurrences of each unique combination of 'water quality', 'quality
df[['water_quality', 'quality_group']].value_counts()
```

Out[14]:
```
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 [15]:
```
# Count the occurrences of each unique combination of 'water quality', 'quality
df[['quantity', 'quantity_group']].value_counts()
```

Out[15]:
```
quantity      quantity_group
enough        enough            33186
insufficient  insufficient      15129
dry           dry                6246
seasonal      seasonal           4050
unknown       unknown             789
Name: count, dtype: int64
```

In [16]:
```
# Count the occurrences of each unique combination of 'source_class', 'source_t
df[['source_class', 'source_type']].value_counts()
```

Out[16]:
```
source_class  source_type
groundwater   spring                17021
              shallow well          16824
              borehole              11949
surface       river/lake            10377
              rainwater harvesting   2295
              dam                     656
unknown       other                   278
Name: count, dtype: int64
```

In [17]:
```
# Count the occurrences of each unique combination of 'source_class', 'source_t
df[['source class', 'source type', 'source']].value counts()
```

Out[17]:
```
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 [18]:
```
# Count the occurrences of each unique combination of 'waterpoint_type', 'water
df[['waterpoint_type', 'waterpoint_type_group']].value_counts()
```

Out[18]:
```
waterpoint_type              waterpoint_type_group
communal standpipe           communal standpipe       28522
hand pump                    hand pump                17488
other                        other                     6380
communal standpipe multiple  communal standpipe        6103
improved spring              improved spring            784
cattle trough                cattle trough              116
dam                          dam                          7
Name: count, dtype: int64
```

In [19]:
```
# Count the occurrences of each unique combination of 'payment' and 'payment_ty
df[['payment', 'payment_type']].value_counts()
```

Out[19]:
```
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 [20]:
```
# Count the occurrences of each unique combination of 'water_quality' and 'qual

df[['water quality', 'quality group']].value counts()
```

Out[20]:
```
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 [21]:
```
# Count the occurrences of each unique combination of'quantity' and 'quantity_g

df[['quantity', 'quantity_group']].value_counts()
```

Out[21]:
```
quantity      quantity_group
enough        enough           33186
insufficient  insufficient     15129
dry           dry               6246
seasonal      seasonal          4050
unknown       unknown            789
Name: count, dtype: int64
```

In [22]:
```
# Count the occurrences of each unique combination of "source",'source_type',

df[["source",'source_type', 'source_class']].value_counts()
```

Out[22]:
```
source                source_type           source_class
spring                spring                groundwater     17021
shallow well          shallow well          groundwater     16824
machine dbh           borehole              groundwater     11075
river                 river/lake            surface          9612
rainwater harvesting  rainwater harvesting  surface          2295
hand dtw              borehole              groundwater       874
lake                  river/lake            surface           765
dam                   dam                   surface           656
other                 other                 unknown           212
unknown               other                 unknown            66
Name: count, dtype: int64
```

In [23]:
```
# Count the occurrences of each unique combination of 'extraction_type', 'extra

df[['extraction_type', 'extraction_type_group', 'extraction_type_class']].value
```

Out[23]:

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

Name: count, dtype: int64

In [24]:
```
# Count the occurrences of each unique combination of 'management', 'management

df[['management', 'management group']].value_counts()
```

Out[24]:
```
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 [25]:
```
# Display the first few entries of the 'recorded_by' column to preview the data

print(df.recorded_by.head())
print("**********")

# Print the unique values of the 'recorded_by' column to see all distinct entr

print(df.recorded_by.unique())
```

```
0    GeoData Consultants Ltd
1    GeoData Consultants Ltd
2    GeoData Consultants Ltd
3    GeoData Consultants Ltd
4    GeoData Consultants Ltd
Name: recorded_by, dtype: object
**********
['GeoData Consultants Ltd']
```

In [26]:
```
# Print the unique values of the 'wpt_name' column to identify all distinct nam
print("Unique values: ", df.wpt_name.unique())

# This indicates the proportion of missing data in this specific column

print("Mean: ", df.wpt_name.isnull(). mean())
```

```
Unique values:  ['none' 'Zahanati' 'Kwa Mahundi' ... 'Kwa Yahona Kuvala' 'Msh
oro'
 'Kwa Mzee Lugawa']
Mean:  3.367003367003367e-05
```

In [27]:
```
# Print the unique values of the 'scheme_name' column to identify all distinct

df.scheme_name.unique
```

Out[27]:
```
<bound method Series.unique of 0                                  Roman
1                                    NaN
2           Nyumba ya mungu pipe scheme
3                                    NaN
4                                    NaN
                      ...
59395            Losaa Kia water supply
59396       Ikondo electrical water sch
59397                                NaN
59398                                NaN
59399                                NaN
Name: scheme_name, Length: 59400, dtype: object>
```

Categorical columns to drop 

1. Date_recorded - This column is redundant because we already have the 'year of construction' which provides similar temporal information. Thus, 'Date_recorded' can be dropped.
2. Funder - Since 'Funder' and 'Installer' contain the same values and 'Installer' has a higher influence on how a water pump functions, you should drop the 'Funder' column.
3. Iga, Ward, sub_village - These can be effectively represented by the 'region' column, allowing you to drop 'Iga', 'Ward', and 'sub_village' to reduce redundancy.
4. Recorded_by - This column contains uniform information (e.g., 'GeoData Consultants Ltd') across all entries, providing no variability or additional insights, hence it can be dropped.
5. Scheme_name - Due to its high number of unique and missing values, 'scheme_name' can be dropped as it may complicate the model without adding significant value.
6. Payment - 'Payment' and 'Payment_type' are duplicative as they contain similar values. You can drop 'Payment' to eliminate redundancy.
7. Quality_group - Similar to 'water_quality', this column does not add additional information since both describe water quality. Therefore, 'Quality_group' can be dropped.
8. Extraction_type, extraction_type_group - These are duplicates to the information in 'Extraction_type_class'. Therefore, retain 'Extraction_type_class' and drop 'Extraction_type'.
9. Quantity_group - As it mirrors the values in 'Quantity', you can drop 'Quantity_group' to streamline the dataset.
10. Source, Source_type - These columns are duplicative and can be consolidated under 'Source_class', allowing you to drop 'Source' and 'Source_type'.
11. Wpt_name - This column has a large number of unique and missing values, which could lead to high dimensionality without much predictive power. It can be dropped.
12. Waterpoint_type - Can be represented by 'Waterpoint_type_group', thus 'Waterpoint_type' can be dropped to simplify the model.
13. Management- management is a category of the management_group and hence it does not add value to the model

A total of 17 columns will be dropped

Analyzing numerical variables for redundancy 🔍

In [28]:
```
# Check categorical columns
df.select_dtypes(include=['float', 'integer']).columns
```

Out[28]:
```
Index(['id', 'amount_tsh', 'gps_height', 'longitude', 'latitude',
       'num_private', 'region_code', 'district_code', 'population',
       'construction_year', 'id'],
      dtype='object')
```

In [29]:
```
df.head()
```

Out[29]:

	id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt
0	69572	6000.0	2011-03-14	Roman	1390	Roman	34.938093	-9.856322	
1	8776	0.0	2013-03-06	Grumeti	1399	GRUMETI	34.698766	-2.147466	Z
2	34310	25.0	2013-02-25	Lottery Club	686	World vision	37.460664	-3.821329	M
3	67743	0.0	2013-01-28	Unicef	263	UNICEF	38.486161	-11.155298	Z Nar
4	19728	0.0	2011-07-13	Action In A	0	Artisan	31.130847	-1.825359	

5 rows × 42 columns

Numerical columns to drop

1. region_code - This column is redundant because it largely mirrors the information provided by the 'region' column.
2. district_code - Similar to 'region_code', this column duplicates the geographical information already captured by the 'region' column.
3. num_private - This column is not useful for analysis as approximately 99% of its values are zeros, indicating a lack of variability and significance.

In [30]:
```
#listing the columns to drop
columns_to_drop = ['date_recorded', 'funder', 'wpt_name', 'subvillage', 'lga',
 'ward', 'recorded_by', 'scheme_name', 'extraction_type',
 'extraction_type_group', 'management', 'payment', 'quality_group',
 'quantity', 'source', 'source_type', 'waterpoint_type', 'num_private',
 'region_code', 'district_code']

# Drop the columns from dataset
df = df.drop(columns_to_drop, axis=1)
```

In [31]:
```
df.head()
```

Out[31]:

	id	amount_tsh	gps_height	installer	longitude	latitude	basin	region	populatio
0	69572	6000.0	1390	Roman	34.938093	-9.856322	Lake Nyasa	Iringa	1(
1	8776	0.0	1399	GRUMETI	34.698766	-2.147466	Lake Victoria	Mara	2(
2	34310	25.0	686	World vision	37.460664	-3.821329	Pangani	Manyara	2(
3	67743	0.0	263	UNICEF	38.486161	-11.155298	Ruvuma / Southern Coast	Mtwara	(
4	19728	0.0	0	Artisan	31.130847	-1.825359	Lake Victoria	Kagera	

5 rows × 22 columns

Dealing with missing values 🛠️

In [32]:
```
# Calculate the percentage of missing (null) values in each column of the DataF

df.isnull().mean().sort_values(ascending=False)
```

Out[32]:
```
scheme_management        0.065286
installer                0.061532
public_meeting           0.056128
permit                   0.051448
extraction_type_class    0.000000
id                       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
id                       0.000000
construction_year        0.000000
amount_tsh               0.000000
population               0.000000
region                   0.000000
basin                    0.000000
latitude                 0.000000
longitude                0.000000
gps_height               0.000000
status_group             0.000000
dtype: float64
```

In [33]:
```
# Display the count of each unique value in the 'scheme_management' column
df.scheme_management.value_counts()
```

Out[33]:
```
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 [34]:
```
# Display the count of each unique value in the 'installer' column

df.installer.value_counts()
```

Out[34]:
```
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 [35]:
```
df.public_meeting.value_counts()
```

Out[35]:
```
public_meeting
True     51011
False     5055
Name: count, dtype: int64
```

In [36]:
```
# Display the count of each unique value in the 'public_meeting' column

df.permit.value_counts()
```

Out[36]:
```
permit
True     38852
False    17492
Name: count, dtype: int64
```

Decision on missing values 🤔

1. Permit: Replace the missing values with the mode. Since "Yes" is the mode, we will fill the missing values with this value.
2. Public_meeting: Replace the missing values with the mode. Since "Yes" is the mode, we will fill the missing values with this value.
3. Drop the "installer" column because it has about 6.5 percent missing values. Its values are names of organizations with multiple entries, making it difficult to impute.
4. Drop "scheme_management" because 6.1 percent of the data is missing. Its values are also names of organizations with multiple entries, making it difficult to impute.

In [37]:
```
#Drop the 'installer' and 'scheme_management' columns from the DataFrame due to
# of missing values and the complexity of their data, which makes imputation di
df.dropna(subset=['installer', 'scheme_management'], inplace=True)


# Fill missing values in 'public_meeting' and 'permit' columns with True direct
df[['public_meeting', 'permit']] = df[['public_meeting', 'permit']].fillna(True
```

In [38]:
```
#Check is  the missing values have been replaced and dropped
df.isnull().sum()
```

Out[38]:
```
id                       0
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_class    0
management_group         0
payment_type             0
water_quality            0
quantity_group           0
source_class             0
waterpoint_type_group    0
id                       0
status_group             0
dtype: int64
```

In [39]:
```
df.nunique()
```

Out[39]:
```
id                       51926
amount_tsh                  97
gps_height                2428
installer                 2003
longitude                50207
latitude                 50209
basin                        9
region                      21
population                1020
public_meeting               2
scheme_management           11
permit                       2
construction_year           55
extraction_type_class        7
management_group             5
payment_type                 7
water_quality                8
quantity_group               5
source_class                 3
waterpoint_type_group        6
id                       51926
status_group                 3
dtype: int64
```

In [40]:
```
#Correcting the errors in the installer column
df['installer'] = df['installer'].replace(to_replace = ('villigers', 'villager
                                        'Villi', 'Village Council', 'Village
                                        'Village community', 'Villaers', 'Vi
                                        'Villege Council', 'Village council',
                                        'Villager', 'VILLAGER', 'Villagers',
                                        'Village water attendant', 'Village C
                                        'VILLAGE COUNCIL .ODA', 'VILLAGE COUN
                                        'VILLAG', 'VILLAGE', 'Village Governm
                                        'Village Govt', 'Village govt', 'VILL
                                        'Village water committee', 'Commu',
                                         'Comunity', 'Communit', 'Kijiji', 'S
                                        value ='Community')

df['installer'] = df['installer'].replace(to_replace = ('FinW', 'Fini water',
                                        'Finwater', 'FINN WATER', 'FinW', 'FW
                                        'FinWate', 'FINLAND', 'Fin Water', 'F
                                        value ='Finnish Government')

df['installer'] = df['installer'].replace(to_replace = ('RC CHURCH', 'RC Churc
                                        'RC church', 'RC CATHORIC', 'Roman Ch
                                        'Roman catholic', 'Roman Ca', 'Roman
                                        'ROMAN CATHOLIC', 'Kanisa', 'Kanisa K
                                        value ='Roman Catholic Church')

df['installer'] = df['installer'].replace(to_replace = ('Dmdd', 'DMDD'), value

df['installer'] = df['installer'].replace(to_replace = ('TASA', 'Tasaf', 'TASAF
                                        'TASSAF', 'TASAF'), value ='TASAF')

df['installer'] = df['installer'].replace(to_replace = ('RW', 'RWE'), value ='R

df['installer'] = df['installer'].replace(to_replace = ('SEMA CO LTD', 'SEMA Co

df['installer'] = df['installer'].replace(to_replace = ('DW E', 'DW#', 'DW$',
                                       'DWEB', 'DWE'), value ='DWE')

df['installer'] = df['installer'].replace(to_replace = ('No', 'NORA', 'Norad',
                                        value ='NORAD')

df['installer'] = df['installer'].replace(to_replace = ('Ox', 'OXFARM', 'OXFAM

df['installer'] = df['installer'].replace(to_replace = ('PRIV', 'Priva', 'Priva
                                        'Private individuals', 'PRIVATE INSTI
                                        'Private person', 'Private Technician
                                        value ='Private')
df['installer'] = df['installer'].replace(to_replace = ('Central government',
                                        'Cental Government','Tanzania governm
                                        'Centra Government', 'central governm
                                        'TANZANIA GOVERNMENT', 'TANZANIAN GOV
                                        'Centr', 'Centra govt', 'Tanzanian Go
                                        'Tanz', 'Tanza', 'GOVERNMENT',
                                        'GOVER', 'GOVERNME', 'GOVERM', 'GOVER
                                        'Governme', 'Governmen', 'Got', 'Serv
                                        'Central Government'),
                                        value = 'Central Government')
```

```
df['installer'] = df['installer'].replace(to_replace = ('IDARA', 'Idara ya maji
                                          'Ministry of water', 'Ministry of wat
                                          'MWE &', 'MWE', 'Wizara ya maji', 'WI
                                          'Ministry of Water'),
                                          value ='Ministry of Water')

df['installer'] = df['installer'].replace(to_replace = ('District COUNCIL', 'DI
                                          'Counc','District council','District
                                          'Council', 'COUN', 'Distri', 'Halmash
                                          'Halmashauri wilaya', 'District Counc
                                          value = 'District  Council')

df['installer'] = df['installer'].replace(to_replace = ('District water depar',
                                          'District water department', 'Distric
                                          value = 'District Water Department')



df['installer'] = df['installer'].replace(to_replace = ('Ch', 'CH', 'Chiko', 'C
                                            'China Goverment'), value ='Chinese

df['installer'] = df['installer'].replace(to_replace = ('Unisef','Unicef', 'UNI

df['installer'] = df['installer'].replace(to_replace = ('Wedeco','WEDEKO', 'WED

df['installer'] = df['installer'].replace(to_replace = ('Wo','WB', 'Word Bank',
                                          'WORDL BANK', 'World', 'world', 'WORL
                                          'world banks', 'World banks', 'WOULD
                                          value ='World Bank')

df['installer'] = df['installer'].replace(to_replace = ('Lga', 'LGA'), value =

df['installer'] = df['installer'].replace(to_replace = ('World Division', 'Worl
                                         'World vision', 'WORLD VISION', 'world
                                          'World Vision'),
                                          value ='World Vision')

df['installer'] = df['installer'].replace(to_replace = ('Local', 'Local technic
                                         'local  technician', 'LOCAL CONTRACT',
                                         'Local l technician', 'Local te', 'Loc
                                         'local technical tec', 'local technici
                                         'local technitian', 'Locall technician
                                         'Local Contractor'),
                                          value ='Local Contractor')

df['installer'] = df['installer'].replace(to_replace = ('DANID', 'DANNY', 'DANN
                                         'DANIDA CO', 'DANID', 'Danid', 'DANIAD
                                         'DENISH', 'DANIDA'),
                                          value ='DANIDA')

df['installer'] = df['installer'].replace(to_replace =('Adrs', 'Adra', 'ADRA'),

df['installer'] = df['installer'].replace(to_replace = ('Hesawa', 'hesawa', 'HE
                                          'HESAWQ', 'HESAWS', 'HESAWZ', 'hesawz
                                          'HESAWA'),
                                          value ='HESAWA')
```

```
df['installer'] = df['installer'].replace(to_replace = ('Jaica', 'JAICA', 'Jica
                                          'Japan', 'JAPAN', 'JAPAN EMBASSY', 'J
                                          'JIKA', 'jika', 'jiks', 'Embasy of Ja
                                          value ='JICA')

df['installer'] = df['installer'].replace(to_replace = ('KKT', 'KK', 'KKKT Chur
                                          'KKKT'), value ='KKKT')

df['installer'] = df['installer'].replace(to_replace = ('0', 'Not Known', 'not
```

In [41]:
```
df['installer'].value_counts(normalize=True)
```

Out[41]:
```
installer
DWE                   0.294149
Central Government    0.072565
Community             0.041521
DANIDA                0.033952
HESAWA                0.026730
                        ...
Upendo Group          0.000019
WA                    0.000019
Insititutiona         0.000019
kanisa                0.000019
SELEPTA               0.000019
Name: proportion, Length: 1770, dtype: float64
```

In [42]:
```
df.head()
```

Out[42]:

	id	amount_tsh	gps_height	installer	longitude	latitude	basin	region	populatio
0	69572	6000.0	1390	Roman Catholic Church	34.938093	-9.856322	Lake Nyasa	Iringa	1(
1	8776	0.0	1399	GRUMETI	34.698766	-2.147466	Lake Victoria	Mara	2(
2	34310	25.0	686	World Vision	37.460664	-3.821329	Pangani	Manyara	2(
3	67743	0.0	263	UNICEF	38.486161	-11.155298	Ruvuma / Southern Coast	Mtwara	(
5	9944	20.0	0	DWE	39.172796	-4.765587	Pangani	Tanga	

5 rows × 22 columns

In [43]:
```
df['installer'].value_counts(normalize=True).head(35)
```

Out[43]:
```
installer
DWE                         0.294149
Central Government          0.072565
Community                   0.041521
DANIDA                      0.033952
HESAWA                      0.026730
District  Council           0.022628
RWE                         0.020394
KKKT                        0.018141
Unknown                     0.015349
Finnish Government          0.013904
World Vision                0.013096
TCRS                        0.013057
CES                         0.011747
Ministry of Water           0.009802
Roman Catholic Church       0.009533
TASAF                       0.009109
JICA                        0.008031
LGA                         0.007934
NORAD                       0.007626
WEDECO                      0.007357
DMDD                        0.007203
World Bank                  0.006663
UNICEF                      0.006374
OXFAM                       0.006374
AMREF                       0.006317
TWESA                       0.006047
WU                          0.005797
ACRA                        0.005354
SEMA                        0.004949
DW                          0.004641
Da                          0.004314
Local Contractor            0.004275
Private                     0.004275
Sengerema Water Department  0.004121
Kiliwater                   0.004044
Name: proportion, dtype: float64
```

In [44]:
```
# Calculate the percentage of each installer
installer_counts = df['installer'].value_counts(normalize=True)

# Find installers that make up less than 0.005 of the total
small_installers = installer_counts[installer_counts < 0.005].index

# Replace these installers with 'other' in the DataFrame
df['installer'] = df['installer'].apply(lambda x: 'other' if x in small_instal
```

In [45]:
```
len(df.installer)
```

Out[45]:
```
51926
```

Data cleaning for the latitudes and longitudes

In [46]:
```
# Check unique values in longitude and latitude columns
df[['longitude', 'latitude']].value_counts()
```

Out[46]:
```
longitude  latitude
0.000000   -2.000000e-08    1671
37.543351  -6.963557e+00       2
32.972719  -2.528716e+00       2
37.314250  -7.177203e+00       2
32.977191  -2.516619e+00       2
                              ...
33.950414  -4.180370e+00       1
33.950440  -9.417821e+00       1
33.950689  -9.468759e+00       1
33.950875  -8.937886e+00       1
40.323402  -1.046327e+01       1
Name: count, Length: 50211, dtype: int64
```

In [47]:

```
# Filter the data
df_f = df[df['status_group'] == 'functional']
df_nf = df[df['status_group'] == 'non functional']
df_fnr = df[df['status_group'] == 'functional needs repair']

# Initialize a folium map centered at an average location
m = folium.Map(location=[-6.369028, 34.888822], zoom_start=6)

# Add CircleMarkers for the functional wells
for _, row in df_f.iterrows():
    folium.CircleMarker(
        location=[row['latitude'], row['longitude']],
        radius=3,  # Smaller radius for smaller points
        popup=row['status_group'],
        color='green',
        fill=True,
        fill_color='green'
    ).add_to(m)

# Add CircleMarkers for the non-functional wells
for _, row in df_nf.iterrows():
    folium.CircleMarker(
        location=[row['latitude'], row['longitude']],
        radius=3,  # Smaller radius for smaller points
        popup=row['status_group'],
        color='red',
        fill=True,
        fill_color='red'
    ).add_to(m)

# Add CircleMarkers for the functional needs repair wells
for _, row in df_fnr.iterrows():
    folium.CircleMarker(
        location=[row['latitude'], row['longitude']],
        radius=3,  # Smaller radius for smaller points
        popup=row['status_group'],
        color='orange',
        fill=True,
        fill_color='orange'
    ).add_to(m)

# Add a legend to the map
legend_html = '''
     <div style="position: fixed;
     bottom: 50px; left: 50px; width: 150px; height: 90px;
     border:2px solid grey; z-index:9999; font-size:14px;
     background-color:white;
     ">  Legend <br>
       <i class="fa fa-map-marker fa-2x" style="color:green"></i>  Fu
       <i class="fa fa-map-marker fa-2x" style="color:red"></i>  Non
       <i class="fa fa-map-marker fa-2x" style="color:orange"></i>  F
     </div>
     '''
m.get_root().html.add_child(folium.Element(legend_html))
```

```
# Display the map in the notebook
display(m)

# Optionally, save the map to an HTML file in the current directory
m.save('well_status_map.html')
```

In [48]:
```
#Checking the  percentage of points that hace zero longitudepercent
zero_longitude_percent = len(df[df['longitude'] == 0]) / len(df)
print(f"Percent of zero longitude entries: {zero_longitude_percent:.2%}")
```

```
Percent of zero longitude entries: 3.22%
```

Decision do delete the datapoints outside Tanzania

Based on the map above, some points are not located in Tanzania. Tanzania is situated at approximately latitude 6°23′31.20″ South and longitude 35°00′07.20″ East. Therefore, coordinates such as longitude 0.000000 and latitude -2.000000e-08 are not within Tanzania's boundaries. There are about 1,671 data entries with these coordinates. The decision was made to remove these points from the dataset because they do not correspond to locations within Tanzania. The data points are only 3.22 percent of the total dataset and therefore, they can be removed.

In [49]:
```
#Excluding rows with latitude 0
df = df[df['longitude'] != 0]
```

In [50]:
```
#Checking whether the change above was made
df[['longitude', 'latitude']].value_counts()
```

Out[50]:
```
longitude  latitude
32.993683  -2.494546     2
37.274352  -7.102004     2
32.919861  -2.476680     2
37.269036  -7.104923     2
32.993277  -2.510639     2
                        ..
33.950440  -9.417821     1
33.950689  -9.468759     1
33.950875  -8.937886     1
33.951194  -9.487462     1
40.323402  -10.463272    1
Name: count, Length: 50210, dtype: int64
```

Cleaning the construction year

The construction year contains zeros (0), so it is essential to address these values.

In [51]:
```
df["construction year"].value_counts()
```

Out[51]:

construction_year	
0	15347
2008	2515
2009	2392
2010	2330
2000	1502
2007	1485
2006	1381
2003	1214
2011	1172
2004	1059
1978	1023
2002	1009
2012	1003
2005	944
1999	922
1995	892
1998	867
1985	804
1984	722
1982	699
1972	692
1994	682
1990	654
1996	641
1974	639
1992	612
1980	595
1997	533
1993	526
1988	500
2001	497
1983	467
1975	425
1986	405
1976	379
1991	305
1970	304
1989	303
1987	285
1981	221
1977	186
1973	184
1979	173
2013	165
1971	128
1963	84
1967	83
1968	76
1969	59
1960	45
1964	40
1962	29
1961	21
1965	19

```
1966       16
Name: count, dtype: int64
```

Addressing the year of construction issue

To address the zeros in the construction year data for water sources, I decided to impute these values by assuming they represent natural springs that were never constructed. To maintain the integrity of my dataset and provide a plausible construction year for these entries, I replaced the zero values with the median construction year of the non-zero entries. The median is less affected by outliers and provides a representative figure that aligns with the typical construction years in the dataset. This approach ensures consistency and meaningfulness in the data, allowing for more accurate and reliable analysis in subsequent steps.

In [52]:
```
# Calculate the median construction year excluding zeros
median_year = int(df[df['construction_year'] != 0]['construction_year'].median(

# Replace zeros in the construction year with the median value
df['construction_year'] = df['construction_year'].replace(0, median_year)
```

In [53]:
```
#Checking to ensure that all zeros are removed
df["construction year"].value_counts()
```

Out[53]:
```
construction_year
2000    16849
2008     2515
2009     2392
2010     2330
2007     1485
2006     1381
2003     1214
2011     1172
2004     1059
1978     1023
2002     1009
2012     1003
2005      944
1999      922
1995      892
1998      867
1985      804
1984      722
1982      699
1972      692
1994      682
1990      654
1996      641
1974      639
1992      612
1980      595
1997      533
1993      526
1988      500
2001      497
1983      467
1975      425
1986      405
1976      379
1991      305
1970      304
1989      303
1987      285
1981      221
1977      186
1973      184
1979      173
2013      165
1971      128
1963       84
1967       83
1968       76
1969       59
1960       45
1964       40
1962       29
1961       21
1965       19
1966       16
Name: count, dtype: int64
```

Checking for duplicates

In [54]:
```
df.duplicated(keep = 'first'). sum()
```

Out[54]:
```
0
```

In [55]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 50255 entries, 0 to 59399
Data columns (total 22 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   id                     50255 non-null  int64
 1   amount_tsh             50255 non-null  float64
 2   gps_height             50255 non-null  int64
 3   installer              50255 non-null  object
 4   longitude              50255 non-null  float64
 5   latitude               50255 non-null  float64
 6   basin                  50255 non-null  object
 7   region                 50255 non-null  object
 8   population             50255 non-null  int64
 9   public_meeting         50255 non-null  bool
 10  scheme_management      50255 non-null  object
 11  permit                 50255 non-null  bool
 12  construction_year      50255 non-null  int64
 13  extraction_type_class  50255 non-null  object
 14  management_group       50255 non-null  object
 15  payment_type           50255 non-null  object
 16  water_quality          50255 non-null  object
 17  quantity_group         50255 non-null  object
 18  source_class           50255 non-null  object
 19  waterpoint_type_group  50255 non-null  object
 20  id                     50255 non-null  int64
 21  status_group           50255 non-null  object
dtypes: bool(2), float64(3), int64(5), object(12)
memory usage: 8.1+ MB
```

In [56]:
```
df.permit.head()
```

Out[56]:
```
0    False
1     True
2     True
3     True
5     True
Name: permit, dtype: bool
```

In [57]:
```
df
# Change the data type of public_meeting and permit columns to binary
#df[['public_meeting', 'permit']] = df[['public_meeting', 'permit']].astype(int
```

Out[57]:

	id	amount_tsh	gps_height	installer	longitude	latitude	basin	region	pop
0	69572	6000.0	1390	Roman Catholic Church	34.938093	-9.856322	Lake Nyasa	Iringa	
1	8776	0.0	1399	other	34.698766	-2.147466	Lake Victoria	Mara	
2	34310	25.0	686	World Vision	37.460664	-3.821329	Pangani	Manyara	
3	67743	0.0	263	UNICEF	38.486161	-11.155298	Ruvuma / Southern Coast	Mtwara	
5	9944	20.0	0	DWE	39.172796	-4.765587	Pangani	Tanga	
...	...	...	...	...	...	...	...	...	
59394	11164	500.0	351	other	37.634053	-6.124830	Wami / Ruvu	Morogoro	
59395	60739	10.0	1210	CES	37.169807	-3.253847	Pangani	Kilimanjaro	
59396	27263	4700.0	1212	other	35.249991	-9.070629	Rufiji	Iringa	
59398	31282	0.0	0	other	35.861315	-6.378573	Rufiji	Dodoma	
59399	26348	0.0	191	World Bank	38.104048	-6.747464	Wami / Ruvu	Morogoro	

50255 rows × 22 columns

In [58]:
```
#Check to see if the change was effected
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 50255 entries, 0 to 59399
Data columns (total 22 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   id                     50255 non-null  int64
 1   amount_tsh             50255 non-null  float64
 2   gps_height             50255 non-null  int64
 3   installer              50255 non-null  object
 4   longitude              50255 non-null  float64
 5   latitude               50255 non-null  float64
 6   basin                  50255 non-null  object
 7   region                 50255 non-null  object
 8   population             50255 non-null  int64
 9   public_meeting         50255 non-null  bool
 10  scheme_management      50255 non-null  object
 11  permit                 50255 non-null  bool
 12  construction_year      50255 non-null  int64
 13  extraction_type_class  50255 non-null  object
 14  management_group       50255 non-null  object
 15  payment_type           50255 non-null  object
 16  water_quality          50255 non-null  object
 17  quantity_group         50255 non-null  object
 18  source_class           50255 non-null  object
 19  waterpoint_type_group  50255 non-null  object
 20  id                     50255 non-null  int64
 21  status_group           50255 non-null  object
dtypes: bool(2), float64(3), int64(5), object(12)
memory usage: 8.1+ MB
```

In [59]:
```
df.  permit . head()
```

Out[59]:
```
0    False
1     True
2     True
3     True
5     True
Name: permit, dtype: bool
```

Part 2: Exploratory Data Analysis (EDA)

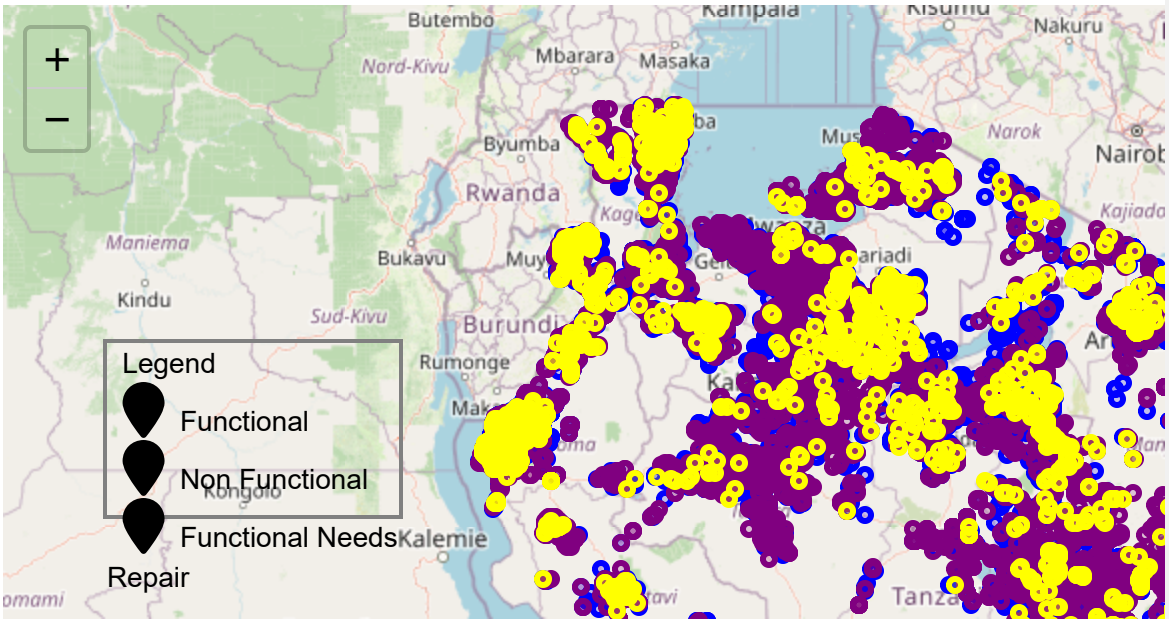
Functional Wells in Tanzania

In [60]:
```
# Filter the data
df_f = df[df['status_group'] == 'functional']
df_nf = df[df['status_group'] == 'non functional']
df_fnr = df[df['status_group'] == 'functional needs repair']

# Initialize a folium map centered at an average location
m = folium.Map(location=[-6.369028, 34.888822], zoom_start=6)

# Add CircleMarkers for the functional wells
for _, row in df_f.iterrows():
    folium.CircleMarker(
        location=[row['latitude'], row['longitude']],
        radius=3,  # Smaller radius for smaller points
        popup=row['status_group'],
        color='blue',
        fill=True,
        fill_color='blue'
    ).add_to(m)

# Add CircleMarkers for the non-functional wells
for _, row in df_nf.iterrows():
    folium.CircleMarker(
        location=[row['latitude'], row['longitude']],
        radius=3,  # Smaller radius for smaller points
        popup=row['status_group'],
        color='purple',
        fill=True,
        fill_color='purple'
    ).add_to(m)

# Add CircleMarkers for the functional needs repair wells
for _, row in df_fnr.iterrows():
    folium.CircleMarker(
        location=[row['latitude'], row['longitude']],
        radius=3,  # Smaller radius for smaller points
        popup=row['status_group'],
        color='yellow',
        fill=True,
        fill_color='yellow'
    ).add_to(m)

# Add a title to the map
title_html = '''
     <h3 align="center" style="font-size:20px"><b>Water Sources Status in Tanza
     '''
m.get_root().html.add_child(folium.Element(title_html))

# Add a legend to the map
legend_html = '''
     <div style="position: fixed;
     bottom: 50px; left: 50px; width: 150px; height: 90px;
     border:2px solid grey; z-index:9999; font-size:14px;
     background-color:white;
     ">  Legend <br>
       <i class="fa fa-map-marker fa-2x" style="color:blue"></i>  Fur
       <i class="fa fa-map-marker fa-2x" style="color:purple"></i>  N
       <i class="fa fa-map-marker fa-2x" style="color:yellow"></i>  F
```

```
    </div>
    '''
m.get_root().html.add_child(folium.Element(legend_html))

# Display the map in the notebook
display(m)

# Optionally, save the map to an HTML file in the current directory
m.save('well_status_map.html')
```

Make this Notebook Trusted to load map: File -> Trust Notebook

Water Sources Status in Tanzania



The above map shows the operational status of various water sources across the country. Functional water sources, marked in blue, are widely distributed throughout Tanzania, indicating a broad presence of operational water points. Non-functional water sources, shown in purple, are also prevalent and scattered across many regions, highlighting areas where water infrastructure might need attention or repair. Water sources that are functional but need repair, indicated in yellow, are similarly dispersed, often overlapping with regions containing both functional and non-functional sources. This distribution reveals areas of concern where maintenance and improvements are necessary to ensure reliable access to water. The map provides a comprehensive overview of the current status of water sources, aiding in identifying priority areas for infrastructure development and repair.

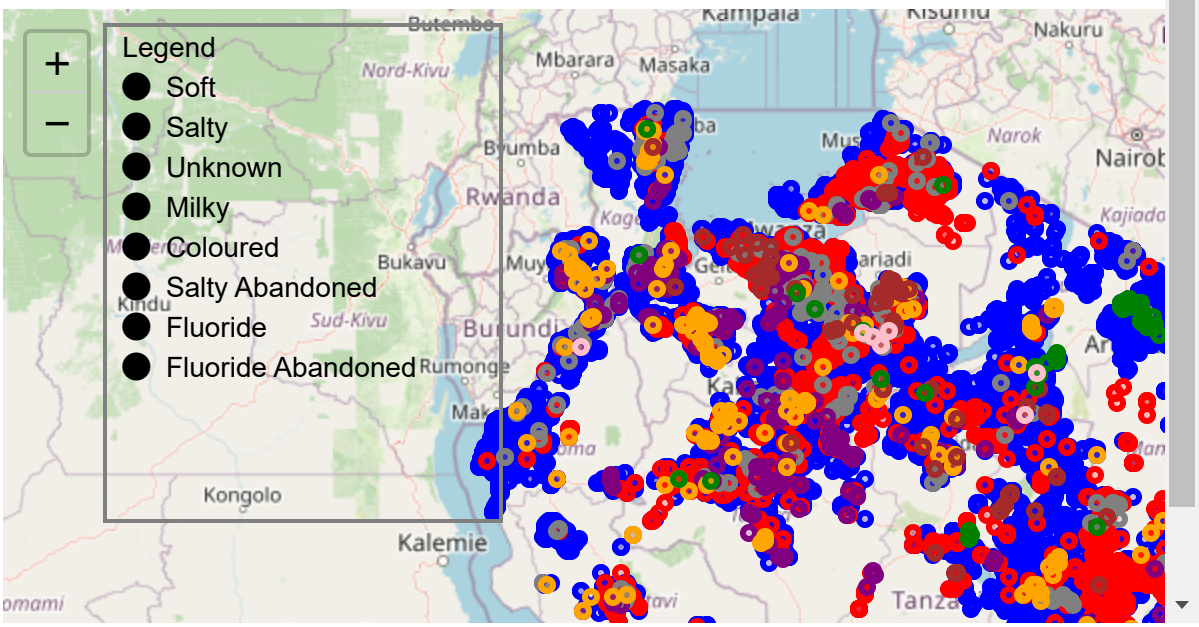
Water Quality Distribution in Tanzania

In [61]:
```
# Define a color dictionary for water quality
water_quality_colors = {
    'soft': 'blue',
    'salty': 'red',
    'unknown': 'grey',
    'milky': 'purple',
    'coloured': 'orange',
    'salty abandoned': 'brown',
    'fluoride': 'green',
    'fluoride abandoned': 'pink'
}

# Initialize a folium map centered at an average location
m = folium.Map(location=[-6.369028, 34.888822], zoom_start=6)

# Add CircleMarkers for each water quality type
for quality, color in water_quality_colors.items():
    df_quality = df[df['water_quality'] == quality]
    for _, row in df_quality.iterrows():
        folium.CircleMarker(
            location=[row['latitude'], row['longitude']],
            radius=3,  # Smaller radius for smaller points
            popup=f"{row['water_quality']} - {row['status_group']}",
            color=color,
            fill=True,
            fill_color=color
        ).add_to(m)

# Add a title to the map
title_html = '''
     <h3 align="center" style="font-size:20px"><b>Water Quality Distribution ir
     '''
m.get_root().html.add_child(folium.Element(title_html))

# Add a legend to the map
legend_html = '''
     <div style="position: fixed;
     bottom: 50px; left: 50px; width: 200px; height: 250px;
     border:2px solid grey; z-index:9999; font-size:14px;
     background-color:white;
     ">  Legend <br>
       <i class="fa fa-circle" style="color:blue"></i>  Soft<br>
       <i class="fa fa-circle" style="color:red"></i>  Salty<br>
       <i class="fa fa-circle" style="color:grey"></i>  Unknown<br>
       <i class="fa fa-circle" style="color:purple"></i>  Milky<br>
       <i class="fa fa-circle" style="color:orange"></i>  Coloured<br
       <i class="fa fa-circle" style="color:brown"></i>  Salty Abando
       <i class="fa fa-circle" style="color:green"></i>  Fluoride<br>
       <i class="fa fa-circle" style="color:pink"></i>  Fluoride Abar
     </div>
     '''
m.get_root().html.add_child(folium.Element(legend_html))

# Display the map in the notebook
display(m)

# Optionally, save the map to an HTML file in the current directory
```

```
m.save('water_quality_map.html')
```

Make this Notebook Trusted to load map: File -> Trust Notebook

Water Quality Distribution in Tanzania



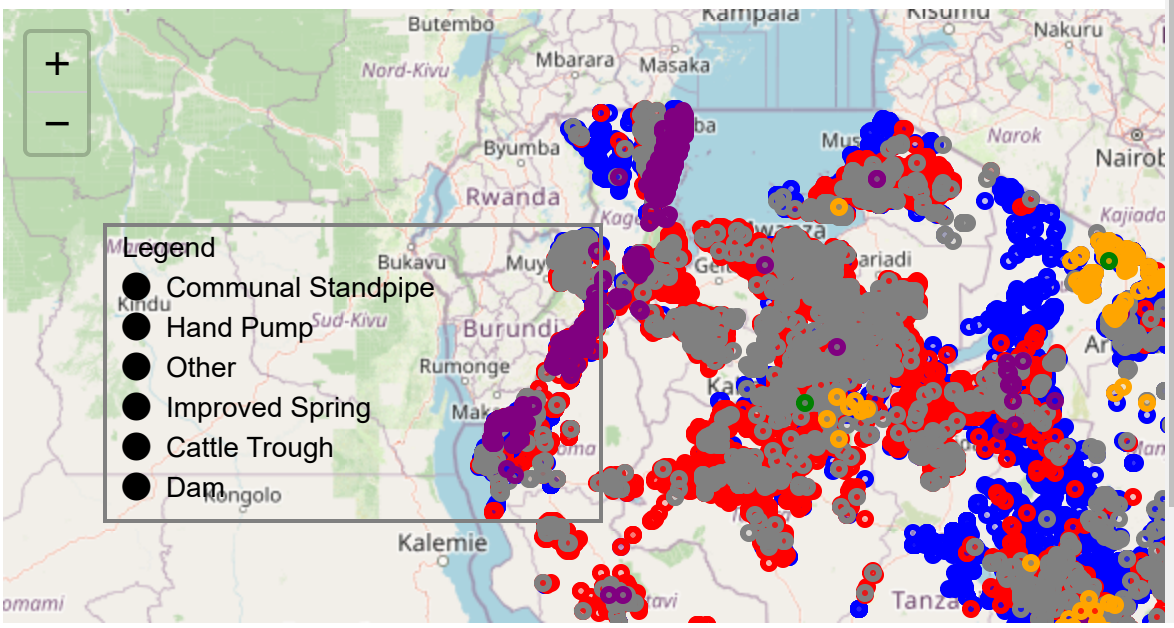
The map above shows the geographical spread of different water quality types across the country. Soft water sources, marked in blue, are widely distributed throughout Tanzania, indicating a broad presence of this water quality type. Salty water sources, shown in red, also have a significant presence, particularly in the eastern and northern regions. Unknown water quality sources, depicted in grey, are scattered but less frequent. Milky water sources, marked in purple, and coloured water sources, shown in orange, appear sporadically across the map. Salty abandoned sources, in brown, and fluoride water sources, in green, are less common but noticeable in specific areas. Fluoride abandoned sources, marked in pink, are the least frequent and appear in very few locations. This distribution highlights the variability in water quality across Tanzania and points to areas that may require targeted interventions for water quality improvement.

Waterpoint Type Distribution in Tanzania

In [62]:
```
# Define a color dictionary for waterpoint types
waterpoint_type_colors = {
    'communal standpipe': 'blue',
    'hand pump': 'red',
    'other': 'grey',
    'improved spring': 'purple',
    'cattle trough': 'orange',
    'dam': 'green'
}

# Initialize a folium map centered at an average location
m = folium.Map(location=[-6.369028, 34.888822], zoom_start=6)

# Add CircleMarkers for each waterpoint type
for waterpoint_type, color in waterpoint_type_colors.items():
    df_type = df[df['waterpoint_type_group'] == waterpoint_type]
    for _, row in df_type.iterrows():
        folium.CircleMarker(
            location=[row['latitude'], row['longitude']],
            radius=3,  # Smaller radius for smaller points
            popup=f"{row['waterpoint_type_group']} - {row['status_group']}",
            color=color,
            fill=True,
            fill_color=color
        ).add_to(m)

# Add a title to the map
title_html = '''
     <h3 align="center" style="font-size:20px"><b>Waterpoint Type Distribution
     '''
m.get_root().html.add_child(folium.Element(title_html))

# Add a legend to the map
legend_html = '''
     <div style="position: fixed;
     bottom: 50px; left: 50px; width: 250px; height: 150px;
     border:2px solid grey; z-index:9999; font-size:14px;
     background-color:white;
     ">  Legend <br>
       <i class="fa fa-circle" style="color:blue"></i>  Communal Star
       <i class="fa fa-circle" style="color:red"></i>  Hand Pump<br>
       <i class="fa fa-circle" style="color:grey"></i>  Other<br>
       <i class="fa fa-circle" style="color:purple"></i>  Improved Sp
       <i class="fa fa-circle" style="color:orange"></i>  Cattle Trou
       <i class="fa fa-circle" style="color:green"></i>  Dam<br>
     </div>
     '''
m.get_root().html.add_child(folium.Element(legend_html))

# Display the map in the notebook
display(m)

# Optionally, save the map to an HTML file in the current directory
m.save('waterpoint_type_map.html')
```

Make this Notebook Trusted to load map: File -> Trust Notebook

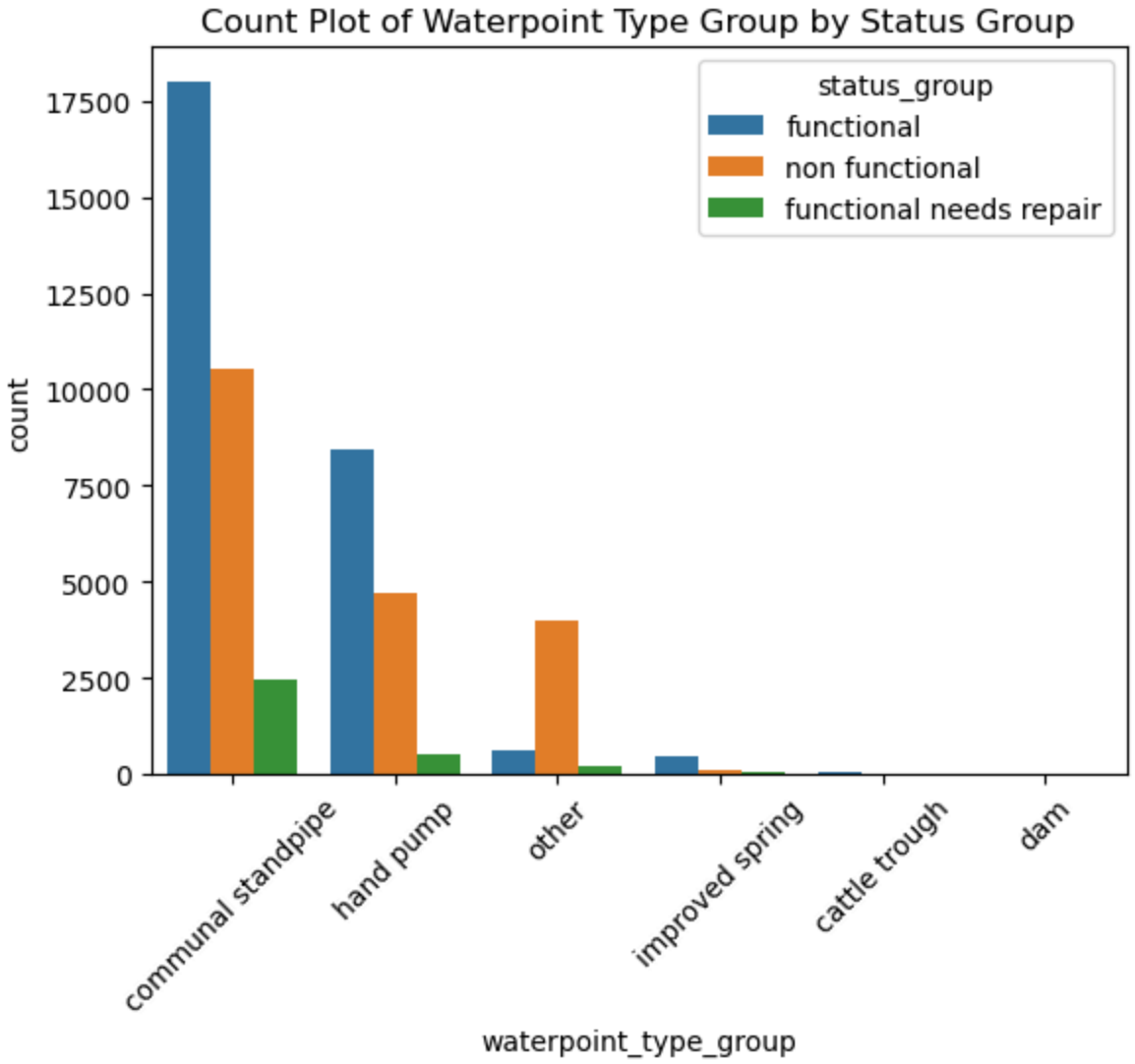
Waterpoint Type Distribution in Tanzania



The map above illustrates the spatial distribution of various waterpoint types across the country. Communal standpipes, shown in dark blue, are widely scattered, indicating their prevalence in many regions. Hand pumps, marked in dark green, are also common and spread across different areas. The grey markers represent other waterpoint types, which are numerous and dispersed throughout Tanzania. Improved springs, indicated in teal, are less frequent and primarily located in specific regions. Cattle troughs, shown in dark orange, and dams, marked in dark red, are the least common waterpoint types, with only a few scattered locations. This map highlights the diverse distribution of waterpoint types, revealing where each type is predominantly used and potentially indicating regional preferences or needs for specific waterpoint infrastructure.

In [63]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [64]:
```
sns.countplot(x='waterpoint_type_group', hue='status_group', data=df)
plt.title('Count Plot of Waterpoint Type Group by Status Group')
plt.xticks(rotation=45)
plt.show()
```



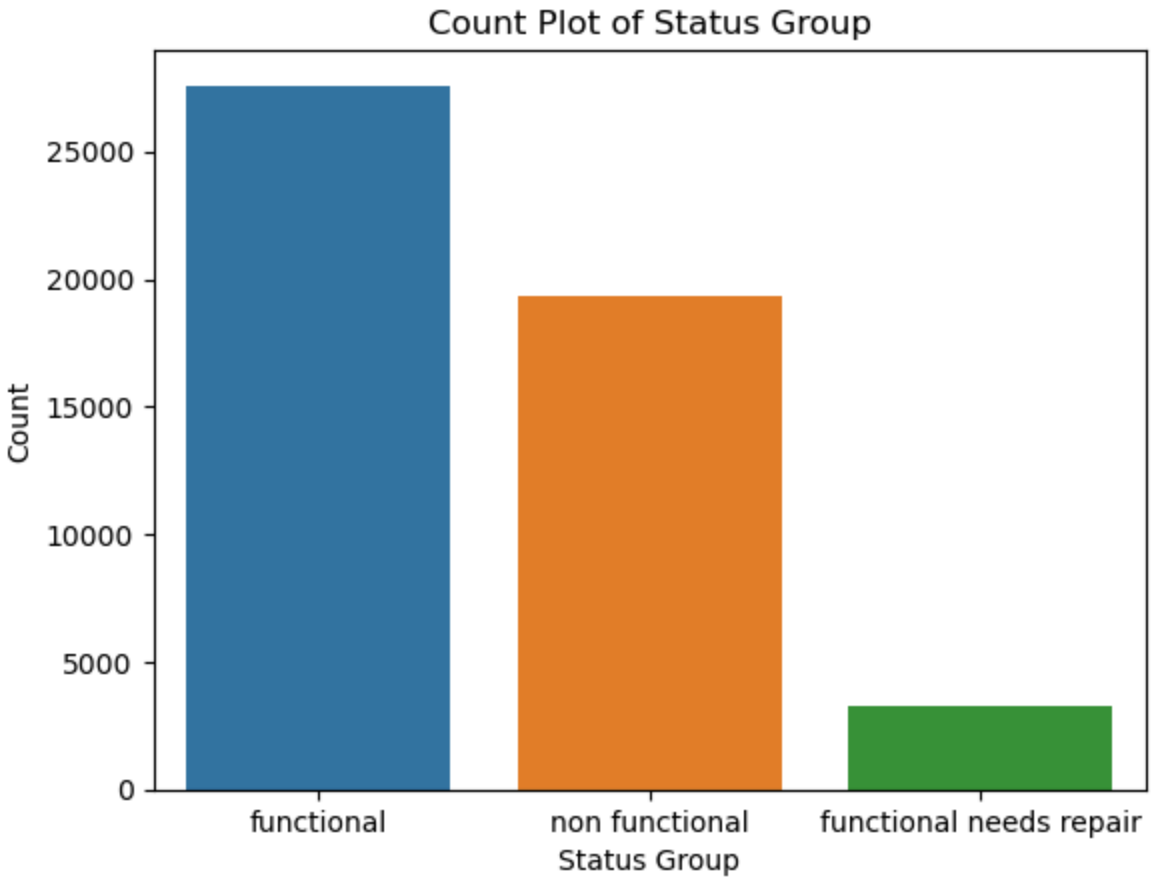
The bar chart above illustrates the distribution of various waterpoint types by their status. Communal standpipes are the most prevalent, with approximately 17,500 functional units, followed by 11,000 non-functional, and around 2,500 needing repair. Hand pumps are the next most common, with about 7,500 functional, 5,000 non-functional, and 1,000 requiring repair. Other waterpoint types, including improved springs, cattle troughs, and dams, have much lower counts across all statuses. The data emphasizes the need for significant maintenance efforts, particularly for communal standpipes and hand pumps, to improve functionality in Tanzania.

In [65]:
```
df.columns
```

Out[65]:
```
Index(['id', 'amount_tsh', 'gps_height', 'installer', 'longitude', 'latitud
e',
       'basin', 'region', 'population', 'public_meeting', 'scheme_managemen
t',
       'permit', 'construction_year', 'extraction_type_class',
       'management_group', 'payment_type', 'water_quality', 'quantity_group',
       'source_class', 'waterpoint_type_group', 'id', 'status_group'],
      dtype='object')
```

In [66]:
```
sns.countplot(data=df, x='status_group')
plt.title('Count Plot of Status Group')
plt.xlabel('Status Group')
plt.ylabel('Count')
```

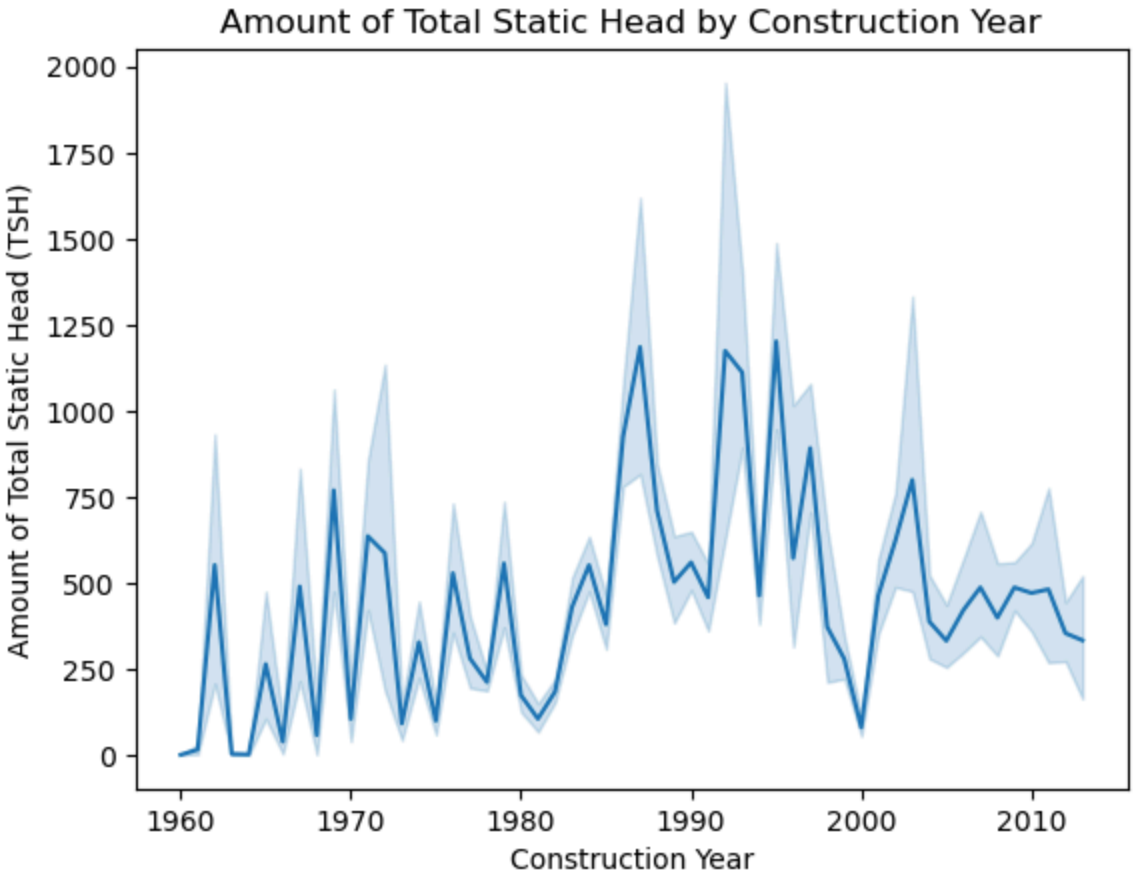
Out[66]:
```
Text(0, 0.5, 'Count')
```



The bar chart illustrates the count of different status groups for wells in Tanzania. The majority of wells are functional, with a count of approximately 27,000. Non-functional wells follow, with a count of around 19,000. The smallest group is wells that are functional but in need of repair, with a count of about 3,000. This distribution highlights the predominance of functional wells but also indicates a significant number of wells that are not operational or require maintenance.

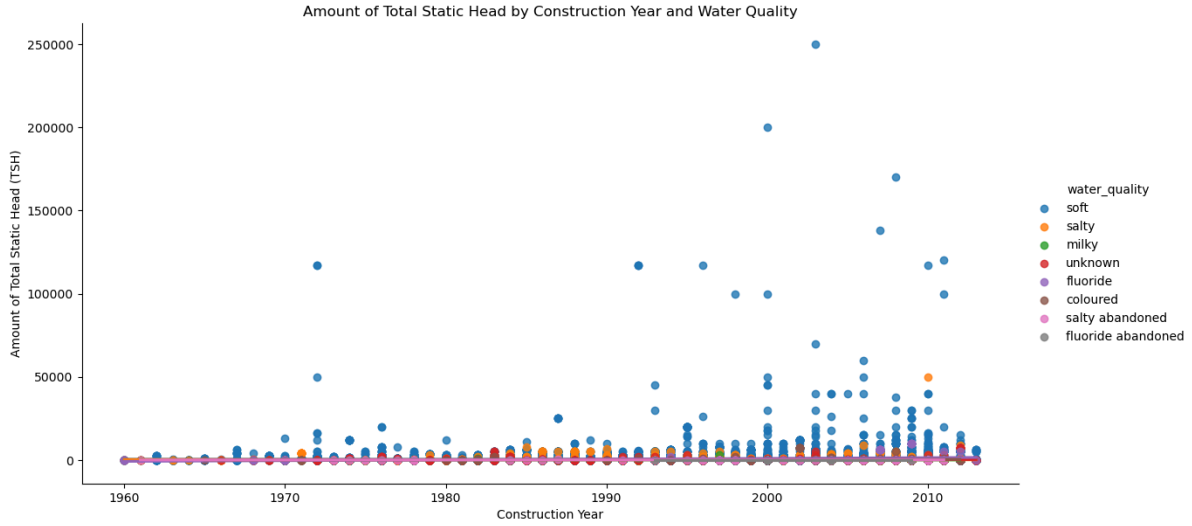
In [67]:
```
sns.lineplot(data=df, x='construction_year', y='amount_tsh')
plt.title('Amount of Total Static Head by Construction Year')
plt.xlabel('Construction Year')
plt.ylabel('Amount of Total Static Head (TSH)')
plt.show()
```

```
C:\Users\user\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWar
ning: use_inf_as_na option is deprecated and will be removed in a future vers
ion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\user\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWar
ning: use_inf_as_na option is deprecated and will be removed in a future vers
ion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



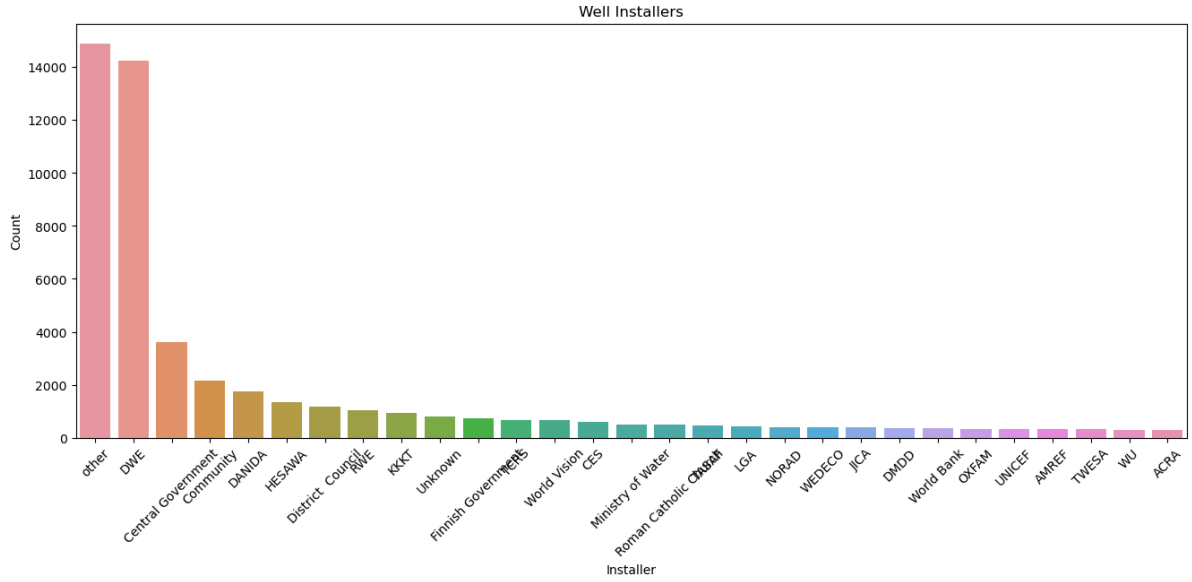
The line plot displays the relationship between the amount of total static head (TSH) and the construction year for wells in Tanzania. The data shows variability over the years, with several peaks and troughs. Notably, there are significant increases in TSH around the late 1980s and early 1990s, followed by a decline in the early 2000s. The shaded area around the line represents the confidence interval, indicating the variability in TSH values for each construction year.

In [68]:
```
# sccatter plot with LOWESS smoothing lines
sns.lmplot(data=df, x='construction_year', y='amount_tsh', hue='water_quality'

plt.title('Amount of Total Static Head by Construction Year and Water Quality'
plt.xlabel('Construction Year')
plt.ylabel('Amount of Total Static Head (TSH)')
plt.show()
```



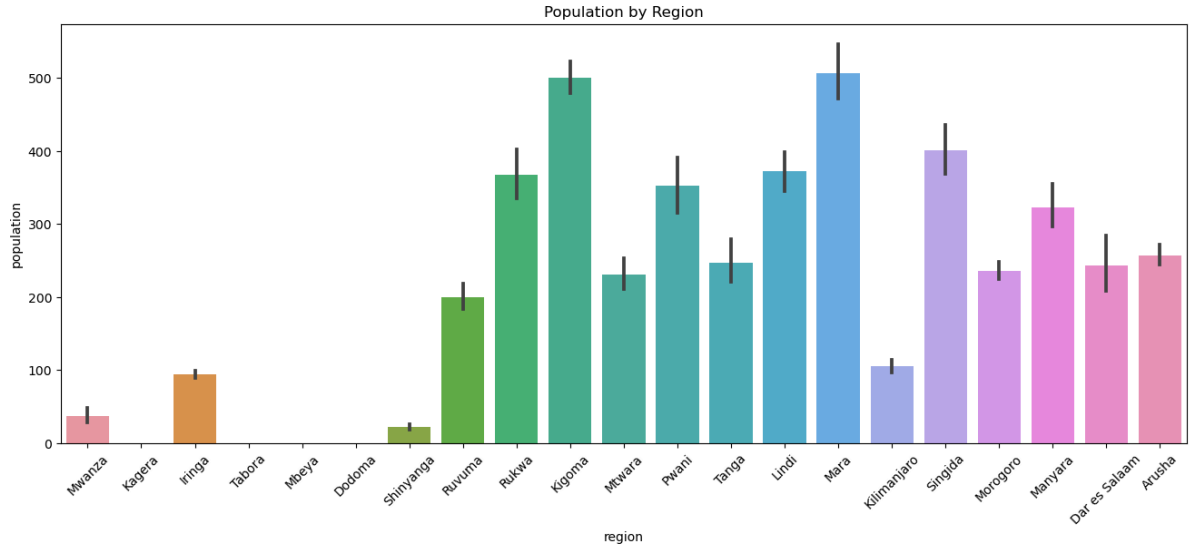
The scatter plot illustrates the relationship between the amount of total static head (TSH) and the construction year of wells, categorized by water quality.Wells constructed more recently tend to be more valuable due to the use of modern technology, better construction materials, and adherence to updated standards. Water quality also plays a crucial role in pricing; wells with high-quality water (e.g., soft water) are generally more valuable than those with lower quality water (e.g., salty or fluoride-abandoned). Therefore, wells that combine high TSH values, recent construction, and superior water quality are positioned to command higher prices.

In [69]:
```
# Set the figure size for better visibility
plt.figure(figsize=(16, 6))

sns.countplot(data=df, x='installer', order=df['installer'].value_counts().inde
plt.title('Well Installers')
plt.xlabel('Installer')
plt.ylabel('Count')

# Rotate x-axis labels to 45 degrees
plt.xticks(rotation=45)
plt.show()
```



The most prominent installers are Other and DWE, each responsible for installing over 14,000 wells, indicating their major roles in well installations. Following these, Central Government and Community have also made significant contributions, each installing several thousand wells. Other notable installers like DANIDA, HESAWA, and District Council each account for between 1,000 and 4,000 installations. Numerous other organizations, including KKKT, Unknown, and the Finnish Government, have fewer installations, contributing hundreds to just over a thousand wells each. This chart highlights both the dominance of a few key players and the diverse range of other organizations involved in well installations.

In [70]:
```
plt.figure(figsize=(16, 6))

# Sort the DataFrame by population in descending order
sorted_df = df.sort_values(by='population', ascending=True)

# Create the bar plot with regions sorted in descending order of population
sns.barplot(x='region', y='population', data=sorted_df, order=sorted_df['regio

# Rotate the x-axis labels to 45 degrees
plt.xticks(rotation=45)
plt.title('Population by Region')

# Display the plot
plt.show()
```



This bar plot above illustrates the population distribution across various regions. Each bar represents a region, with the height reflecting its population size. Mara stands out with the highest population, surpassing 500. Other regions with substantial populations include Kigoma, Iringa, Rukwa, and Ruvuma, each ranging between approximately 200 and 400. Regions such as Pwani, Tanga, Lindi, and Manyara show moderate population sizes, generally between 200 and 300. In contrast, Mwanza and Dodoma have the lowest populations, both falling below 100. The error bars provide an indication of the variability or uncertainty associated with these population estimates, adding a layer of context to the data presented.

In [71]:
```
original_file_path = r"C:\Users\user\Desktop\Checkpoints\Phase 3\Project\df_cle

df.to_csv(original_file_path, index=False)
```