Project title: TANZANIA WATER WELL PUMP STATUS PREDICTION

PROJECT BY: EZRA KIPCHIRCHIR BETT

PROJECT OVERVIEW

The history of water supply in Tanzania reflects a complex narrative shaped by geographical, social, and economic factors. Tanzania, endowed with abundant water resources from its lakes, rivers, and underground aquifers, initially faced limited challenges in providing adequate water to its population. This initiative responds to the pressing need for efficient maintenance and management of water infrastructure to ensure reliable access to clean water across various regions in Tanzania. By analyzing historical data encompassing pump functionality, geographical attributes, maintenance records, and local demographics, this project seeks to predict potential pump failures, malfunctions, or required maintenance, enabling proactive interventions to sustain a consistent water supply. The project's ultimate goal is to create a scalable and adaptable predictive tool that aids decision-makers and maintenance teams in identifying potential issues with water pumps in advance. This predictive capability will empower stakeholders to optimize resource allocation, prioritize maintenance efforts, reduce downtime, and ensure sustainable water access for communities across Tanzania, thereby contributing to improved water infrastructure management and service delivery.
The project's innovative approach not only focuses on predictive analytics but also emphasizes community engagement and empowerment. Through partnerships with local stakeholders and communities, the project aims to integrate qualitative data, such as user feedback and community perceptions, into the predictive model. This inclusive strategy ensures that the tool not only relies on quantitative metrics but also incorporates the invaluable insights and experiences of those directly impacted by water pump functionality. By harnessing the collective knowledge of both data-driven insights and community perspectives, the project seeks to develop a holistic and robust solution that not only predicts pump statuses but also fosters a more participatory and sustainable approach to water infrastructure management in Tanzania.

INTRODUCTION

This a supervised Machine Learning project aimed at predicting conditions of water pumps located in water wells around the region of Tanzania. Tanzania is a developing country located in the East African region neighboring Kenya and Uganda. This machine learning model focuses on predicting whether a pump is functional, non functional, or functional and needs repair. Our research is based on a dataset collected by GeoData Consultants Ltd obtained from Taarifa platform that collects and shares data about water points and their status, including functionality and repairs. GeoData Consultants Ltd is a geo-spatial consultancy firm that specializes in providing comprehensive and innovative solutions using geographical data and spatial analysis techniques. Their expertise lies in leveraging geographic information systems (GIS), remote sensing, and other spatial technologies to offer valuable insights and solutions across various industries.
Since Tanzania is a developing country its water delivery infrastructure is still very low and

this leads to low supply of water to the Tanzanian residents. This project is aimed at using geodata about pumps in use and their attributes to provide insight to Tanzania government represented by the Ministry of water and sanitation if a water pump is functional, not functional or needs repair. This is to help towards curbing water shortage in the country and ensuring each and ever

Challenges

In a developing country like Tanzania, ensuring clean and accessible piped water faces numerous challenges. Some of these challenges include:
1. `Infrastructure Deficiency` : Many areas lack the necessary infrastructure for piped water systems. Remote or rural areas might not have pipelines or water treatment facilities, making it difficult to distribute clean water to these regions.
2. `Limited Access to Technology` : Developing countries may face limitations in accessing advanced technology for water treatment and distribution.
3. `Financial Constraints` : Funding constraints often limit the government's ability to invest in water infrastructure and maintenance.
4. `Population Growth` : Rapid population growth strains existing water resources and infrastructure. As urbanization increases, the demand for water rises, putting pressure on already limited water sources and distribution systems.
5 `Poor Maintenance` : Existing water infrastructure might suffer from poor maintenance due to a lack of resources or skilled personnel.

Proposed solutions

1.Build a predictive machine learning model that predicts whether a pump is functional , not functional or functioning but needs repair in orde to reduce water shortage due to failure of pumps
2.Identify features or properties that lead to these water related problems and adress them
3.Identify regions that suffer most from these challenges so we can have a map or ways on how to improve their status.
4.Investigate whether the infrastructure available is enough to meet the people's need

PROBLEM STATEMENT

The provision of clean and sustainable water access remains a critical challenge in Tanzania, with a vast network of water pumps serving communities across the country. However, the operational status of these pumps fluctuates, leading to intermittent water supply and hindering communities' access to safe drinking water. This project addresses the pressing need to develop a robust machine learning classification model capable of accurately predicting the operational status of water pumps in Tanzania.
The project scope focuses on harnessing historical data encompassing pump functionality, geographical features, maintenance records, and regional demographics to train and validate a predictive model. The objective is to accurately classify water pumps into different status categories, including 'functional,' 'functional needs repair,' and 'non-functional.' A successful model will enable stakeholders and decision-makers to anticipate potential pump failures or maintenance requirements proactively, thus optimizing resource allocation, reducing downtime, and ensuring sustained water access for Tanzanian communities.

DATA UNDERSTANDING

Data understanding involves comprehending the dataset's structure, contents, and potential insights, examining features and their relationships to extract valuable information for analysis or modeling purposes.
- `Source` - The dataset originates from Taarifa, a platform collecting reports on infrastructure and services, particularly focused on water points. This data was compiled by GeoData company limited.
- `Components` - This dataset consists of 59400 rows and 41 columns of data.

Column Description

amount_tsh - Total static head (amount water available to waterpoint)
date_recorded - The date the row was entered
funder - Who funded the well
gps_height - Altitude of the well
installer - Organization that installed the well
longitude - GPS coordinate
latitude - GPS coordinate
wpt_name - Name of the waterpoint if there is one
num_private -number of private water points
basin - Geographic water basin
subvillage - Geographic location
region - Geographic location
region_code - Geographic location (coded)
district_code - Geographic location (coded)
lga - Geographic location
ward - Geographic location
population - Population around the well
public_meeting - True/False
recorded_by - Group entering this row of data
scheme_management - Who operates the waterpoint
scheme_name - Who operates the waterpoint
permit - If the waterpoint is permitted
construction_year - Year the waterpoint was constructed
extraction_type - The kind of extraction the waterpoint uses
extraction_type_group - The kind of extraction the waterpoint uses
extraction_type_class - The kind of extraction the waterpoint uses
management - How the waterpoint is managed
management_group - How the waterpoint is managed
payment - What the water costs
payment_type - What the water costs
water_quality - The quality of the water
quality_group - The quality of the water
quantity - The quantity of water
quantity_group - The quantity of water
source - The source of the water
source_type - The source of the water
source_class - The source of the water
<br waterpoint_type - The kind of waterpoint

waterpoint_type_group - The kind of waterpoint

In [1]:
```
#import modules
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import descartes
from shapely.geometry import Point, Polygon
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder, StandardScaler, MinI
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from data_understanding import data_summary
import warnings
warnings.filterwarnings("ignore")
```

1.1 checking our data

In [2]:
```
path = "Test_set_values.csv"
summary = data_summary(path)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59400 entries, 0 to 59399
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   id            59400 non-null  int64 
 1   status_group  59400 non-null  object
dtypes: int64(1), object(1)
memory usage: 928.2+ KB
```

In [3]:
```
path_2 = "702ddfc5-68cd-4d1d-a0de-f5f566f76d91.csv"
summary_2 = data_summary(path_2)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14850 entries, 0 to 14849
Data columns (total 40 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   id                     14850 non-null  int64  
 1   amount_tsh             14850 non-null  float64
 2   date_recorded          14850 non-null  object 
 3   funder                 13980 non-null  object 
 4   gps_height             14850 non-null  int64  
 5   installer              13973 non-null  object 
 6   longitude              14850 non-null  float64
 7   latitude               14850 non-null  float64
 8   wpt_name               14850 non-null  object 
 9   num_private            14850 non-null  int64  
 10  basin                  14850 non-null  object 
 11  subvillage             14751 non-null  object 
 12  region                 14850 non-null  object 
 13  region_code            14850 non-null  int64  
 14  district_code          14850 non-null  int64  
 15  lga                    14850 non-null  object 
 16  ward                   14850 non-null  object 
 17  population             14850 non-null  int64  
 18  public_meeting         14029 non-null  object 
 19  recorded_by            14850 non-null  object 
 20  scheme_management      13881 non-null  object 
 21  scheme_name            7608 non-null   object 
 22  permit                 14113 non-null  object 
 23  construction_year      14850 non-null  int64  
 24  extraction_type        14850 non-null  object 
 25  extraction_type_group  14850 non-null  object 
 26  extraction_type_class  14850 non-null  object 
 27  management             14850 non-null  object 
 28  management_group       14850 non-null  object 
 29  payment                14850 non-null  object 
 30  payment_type           14850 non-null  object 
 31  water_quality          14850 non-null  object 
 32  quality_group          14850 non-null  object 
 33  quantity               14850 non-null  object 
 34  quantity_group         14850 non-null  object 
 35  source                 14850 non-null  object 
 36  source_type            14850 non-null  object 
 37  source_class           14850 non-null  object 
 38  waterpoint_type        14850 non-null  object 
 39  waterpoint_type_group  14850 non-null  object 
dtypes: float64(3), int64(7), object(30)
memory usage: 4.5+ MB
```

In [4]:
```
path_3 = "4910797b-ee55-40a7-8668-10efd5c1b960.csv"
data_summary(path_3)
```

```
<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
```

2. Merging our target classes DataFrame with training set DataFrame

We are using the id column to join the two DataFrames using the "inner" method

In [5]:
```
df_1 = pd.read_csv("Test_set_values.csv", index_col= 0)
df_2 = pd.read_csv("4910797b-ee55-40a7-8668-10efd5c1b960.csv", index_
df_3 = pd.read_csv("702ddfc5-68cd-4d1d-a0de-f5f566f76d91.csv", index_
```

Checking if columns of our two data sets are similar

In [6]:
```
assert (df_2.columns  == df_3.columns).any()
```

In [7]:
```
#creating a join
water_data = pd.merge(df_1, df_2, on= "id", how= "inner")
#seeing the first five rowa
water_data.head()
```

Out[7]:

id	status_group	amount_tsh	date_recorded	funder	gps_height	installer	longitude
69572	functional	6000.0	2011-03-14	Roman	1390	Roman	34.938093
8776	functional	0.0	2013-03-06	Grumeti	1399	GRUMETI	34.698766
34310	functional	25.0	2013-02-25	Lottery Club	686	World vision	37.460664
67743	non functional	0.0	2013-01-28	Unicef	263	UNICEF	38.486161
19728	functional	0.0	2011-07-13	Action In A	0	Artisan	31.130847

5 rows × 40 columns

Inspecting our new merged data frame

In [8]:
```
#
water_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 59400 entries, 69572 to 26348
Data columns (total 40 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   status_group           59400 non-null  object 
 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(6), object(31)
memory usage: 18.6+ MB
```

Summary statistics for our numerical columns in our dataset

In [9]:
```
#getting summary statistics for our numeric columns
water_data.describe()
```

Out[9]:

	amount_tsh	gps_height	longitude	latitude	num_private	region_cod
count	59400.000000	59400.000000	59400.000000	5.940000e+04	59400.000000	59400.00000
mean	317.650385	668.297239	34.077427	-5.706033e+00	0.474141	15.29700
std	2997.574558	693.116350	6.567432	2.946019e+00	12.236230	17.58740
min	0.000000	-90.000000	0.000000	-1.164944e+01	0.000000	1.00000
25%	0.000000	0.000000	33.090347	-8.540621e+00	0.000000	5.00000
50%	0.000000	369.000000	34.908743	-5.021597e+00	0.000000	12.00000
75%	20.000000	1319.250000	37.178387	-3.326156e+00	0.000000	17.00000
max	350000.000000	2770.000000	40.345193	-2.000000e-08	1776.000000	99.00000

Changing dates to DateTime object

In [10]:
```
#converting dates from objects to DateTime data types
water_data["construction_year"] = pd.to_datetime(water_data["constru
water_data["date_recorded"] = pd.to_datetime(water_data["date_record
```

3 Description of columns/ column understanding

Understanding what our columns contain and checking the similarities between colummns

In [11]:
```
#getting columns
water_data.columns
```

Out[11]:
```
Index(['status_group', 'amount_tsh', 'date_recorded', 'funder', 'gp
s_height',
       'installer', 'longitude', 'latitude', 'wpt_name', 'num_priva
te',
       'basin', 'subvillage', 'region', 'region_code', 'district_co
de', 'lga',
       'ward', 'population', 'public_meeting', 'recorded_by',
       'scheme_management', 'scheme_name', 'permit', 'construction_
year',
       'extraction_type', 'extraction_type_group', 'extraction_type
_class',
       'management', 'management_group', 'payment', 'payment_type',
       'water_quality', 'quality_group', 'quantity', 'quantity_grou
p',
       'source', 'source_type', 'source_class', 'waterpoint_type',
       'waterpoint_type_group'],
      dtype='object')
```

There are a few columns that are missing values but population and scheme_name are missing a lot(almost 21,000 records). Since scheme_managment provides the close to the same information according to the data documentation, we will drop scheme_name

In [12]:
```
#dropping scheme_name
water_data = water_data.drop(["scheme_name"], axis=1)
```

I have also noticed that some of the features have almost similar discriptions and values. We will be check these to see if the columns are duplicates and if so get rid of some of them to reduce feature dimensionality in our analysis.
I will write a helper function to help us get the feature name, number of unique values, the unique values and number of missing or Nan values. This function takes in a column or columns iterates through it and prints out the specified fields above

In [13]:
```
#Helper function
def column_checking(column):
    for i in column:
        print("Feature Name:", i)
        print("Number of Unique Values:", len(water_data[i].unique()
        print("Unique Values:", water_data[i].unique())
        print("Missing Values:", water_data[i].isna().sum())
        print('\n')
```

3.1 Funder and installer

These two columns talk about who funded the well and who installed the well

In [14]:
```
column_checking(["funder", "installer"])
```

```
Feature Name: funder
Number of Unique Values: 1897
Unique Values: ['Roman' 'Grumeti' 'Lottery Club' ... 'Dina' 'Brown'
'Samlo']
Missing Values: 3637


Feature Name: installer
Number of Unique Values: 2146
Unique Values: ['Roman' 'GRUMETI' 'World vision' ... 'Dina' 'brown'
'SELEPTA']
Missing Values: 3655
```

We are going to drop funder since it has less number of unique values. "installer" column has more unique values meaning it explains more of the data

In [15]:
```
water_data = water_data.drop("funder", axis = 1)
```

They both seem to have the same unique values and the funder column has more missing values than the installer column. The funder column also seems to have no impact in our analysis and we are going to drop it and use the installer column

3.2 Subvillage/ region/ region_code/ district_code/ lga/ ward

In [16]:
```
column_checking(["subvillage", "region", "region_code", "district_co
```

```
Feature Name: subvillage
Number of Unique Values: 19288
Unique Values: ['Mnyusi B' 'Nyamara' 'Majengo' ... 'Itete B' 'Maore
Kati' 'Kikatanyemba']
Missing Values: 371


Feature Name: region
Number of Unique Values: 21
Unique Values: ['Iringa' 'Mara' 'Manyara' 'Mtwara' 'Kagera' 'Tanga'
'Shinyanga' 'Tabora'
 'Pwani' 'Ruvuma' 'Kilimanjaro' 'Rukwa' 'Mwanza' 'Kigoma' 'Lindi' '
Dodoma'
 'Arusha' 'Mbeya' 'Singida' 'Morogoro' 'Dar es Salaam']
Missing Values: 0


Feature Name: region_code
Number of Unique Values: 27
Unique Values: [11 20 21 90 18  4 17 14 60 10  3 15 19 16 80  1  6
2 12 13  5  7 99 24
  9  8 40]
Missing Values: 0


Feature Name: district_code
Number of Unique Values: 20
Unique Values: [ 5  2  4 63  1  8  3  6 43  7 23 33 53 62 60 30 13
0 80 67]
Missing Values: 0
```

All the above columns represent the geographical location of water wells in Tanzania.The subvillage column contains 19288 unique values which makes it hard for analysis especially during one-hot encoding. Also this column is not relevant in our analysis since name of village has no influence in predicting the status of a pump. Both region code and district_code are in numbers making it hard to decipher information from it so we are going to drop them.This leaves with region column with 21 unique values and 0 missing data. We are going to keep this column as it is important for our analysis.

In [17]:
```
#dropping subvillage,region_code, district_code,
water_data = water_data.drop(["subvillage", "region_code", "district_
```

In [18]:
```
#checking lga column
column_checking(["lga"])
```

```
Feature Name: lga
Number of Unique Values: 125
Unique Values: ['Ludewa' 'Serengeti' 'Simanjiro' 'Nanyumbu' 'Karagw
e' 'Mkinga'
 'Shinyanga Rural' 'Kahama' 'Tabora Urban' 'Mkuranga' 'Namtumbo' 'M
aswa'
 'Siha' 'Meatu' 'Sumbawanga Rural' 'Njombe' 'Ukerewe' 'Bariadi' 'Sa
me'
 'Kigoma Rural' 'Moshi Rural' 'Lindi Rural' 'Rombo' 'Chamwino' 'Bag
amoyo'
 'Mafia' 'Arusha Rural' 'Kyela' 'Kondoa' 'Kilolo' 'Kibondo' 'Makete
'
 'Singida Rural' 'Masasi' 'Rungwe' 'Moshi Urban' 'Geita' 'Mbulu'
 'Bukoba Rural' 'Muheza' 'Lushoto' 'Meru' 'Iramba' 'Kilombero' 'Mba
rali'
 'Kasulu' 'Bukoba Urban' 'Korogwe' 'Bukombe' 'Morogoro Rural' 'Kish
apu'
 'Musoma Rural' 'Sengerema' 'Iringa Rural' 'Muleba' 'Dodoma Urban'
 'Ruangwa' 'Hanang' 'Misenyi' 'Missungwi' 'Songea Rural' 'Tanga' 'T
unduru'
 'Hai' 'Mwanga' 'Chato' 'Biharamulo' 'Ileje' 'Mpwapwa' 'Mvomero' 'B
unda'
 'Kiteto' 'Longido' 'Urambo' 'Mbozi' 'Sikonge' 'Ilala' 'Tarime' 'Te
meke'
 'Mbeya Rural' 'Magu' 'Manyoni' 'Igunga' 'Kilosa' 'Babati' 'Chunya'
 'Mufindi' 'Mtwara Rural' 'Ngara' 'Karatu' 'Mpanda' 'Kibaha'
 'Singida Urban' 'Newala' 'Nzega' 'Nkasi' 'Bahi' 'Mbinga' 'Ulanga'
 'Sumbawanga Urban' 'Morogoro Urban' 'Tandahimba' 'Kisarawe'
 'Mtwara Urban' 'Kilwa' 'Liwale' 'Kongwa' 'Uyui' 'Rufiji' 'Kwimba'
 'Monduli' 'Shinyanga Urban' 'Ngorongoro' 'Handeni' 'Rorya' 'Pangan
i'
 'Lindi Urban' 'Nachingwea' 'Kinondoni' 'Kigoma Urban' 'Ilemela' 'K
ilindi'
 'Arusha Urban' 'Songea Urban' 'Nyamagana']
Missing Values: 0
```

"LGA" stands for "Local Government Area." It refers to a specific administrative division or region within a country or a larger administrative boundary. Represents an administrative or governmental subdivision within a country, typically smaller than a state or province. We are going to keep it and use it in our EDA and drop it later when modeling

In [19]:
```
column_checking(["ward"])
```

```
Feature Name: ward
Number of Unique Values: 2092
Unique Values: ['Mundindi' 'Natta' 'Ngorika' ... 'Chinugulu' 'Nyamt
inga' 'Kinungu']
Missing Values: 0
```

In [20]:
```
water_data = water_data.drop("ward", axis= 1)
```

The "ward " column has 2092 distinct strings making it hard for our analysis.. We drop it.

3.4 extraction_type/extraction_type_group/extraction_type_class

often provide hierarchical information about how water is extracted or the mechanism used for extraction.

In [21]:
```
column_checking(["extraction_type","extraction_type_group", "extracti
```

```
Feature Name: extraction_type
Number of Unique Values: 18
Unique Values: ['gravity' 'submersible' 'swn 80' 'nira/tanira' 'ind
ia mark ii' 'other'
 'ksb' 'mono' 'windmill' 'afridev' 'other - rope pump' 'india mark
iii'
 'other - swn 81' 'other - play pump' 'cemo' 'climax' 'walimi'
 'other - mkulima/shinyanga']
Missing Values: 0


Feature Name: extraction_type_group
Number of Unique Values: 13
Unique Values: ['gravity' 'submersible' 'swn 80' 'nira/tanira' 'ind
ia mark ii' 'other'
 'mono' 'wind-powered' 'afridev' 'rope pump' 'india mark iii'
 'other handpump' 'other motorpump']
Missing Values: 0


Feature Name: extraction_type_class
Number of Unique Values: 7
Unique Values: ['gravity' 'submersible' 'handpump' 'other' 'motorpu
mp' 'wind-powered'
 'rope pump']
Missing Values: 0
```

All the three columns contain thr same data. Extraction_type is the super class and the other two are subsets. Extraction type has different man-powered mechanisms assigned as other many times and in the real sense they mean the same thing(There is no pump). "Extraction_type_group" column has "other" as a unique value meaning all the man-powered mechanisms has been summed up into one category. We are going to use this column for our analysis and drop the other two columns.

In [22]:
```
#dropping extraction_type and extraction_type_class
water_data = water_data.drop(["extraction_type", "extraction_type_cla
```

3.5 scheme_managemnet/ mnanagement/ management_group

####

In [23]:
```
#checking management, scheme_management, management
column_checking(["management", "scheme_management", "management_grou
```

```
Feature Name: management
Number of Unique Values: 12
Unique Values: ['vwc' 'wug' 'other' 'private operator' 'water board
' 'wua' 'company'
 'water authority' 'parastatal' 'unknown' 'other - school' 'trust']
Missing Values: 0


Feature Name: scheme_management
Number of Unique Values: 12
Unique Values: ['VWC' 'Other' nan 'Private operator' 'WUG' 'Water B
oard' 'WUA'
 'Water authority' 'Company' 'Parastatal' 'Trust' 'SWC']
Missing Values: 3878


Feature Name: management_group
Number of Unique Values: 5
Unique Values: ['user-group' 'other' 'commercial' 'parastatal' 'unk
nown']
Missing Values: 0
```

These columns also almost similar except for a few features in the data. The management column is complete with no missing values, "Scheme_management" the same except it has missing values so we drop it. The "management_group" column has summarized the whole categories in the management group so we are going to keep it and use it during one-hot encoding instead of management due to computational power available but we are gonna use them for analysis.

In [24]:
```
#dropping scheme_management
water_data = water_data.drop("scheme_management", axis = 1)
```

3.6 payment/ payment_type

In [25]:
```
#checking payment and payment_type
column_checking(["payment", "payment_type"])
```

```
Feature Name: payment
Number of Unique Values: 7
Unique Values: ['pay annually' 'never pay' 'pay per bucket' 'unknow
n'
 'pay when scheme fails' 'other' 'pay monthly']
Missing Values: 0


Feature Name: payment_type
Number of Unique Values: 7
Unique Values: ['annually' 'never pay' 'per bucket' 'unknown' 'on f
ailure' 'other'
 'monthly']
Missing Values: 0
```

These columns are similar except for how values have been named. We are going to drop payment and remain with payment_type

In [26]:
```
water_data = water_data.drop("payment", axis= 1)
```

3.7 water quality/ quality_group

In [27]:
```
#checking quality and quality_group
column_checking(["water_quality", "quality_group"])
```

```
Feature Name: water_quality
Number of Unique Values: 8
Unique Values: ['soft' 'salty' 'milky' 'unknown' 'fluoride' 'colour
ed' 'salty abandoned'
 'fluoride abandoned']
Missing Values: 0


Feature Name: quality_group
Number of Unique Values: 6
Unique Values: ['good' 'salty' 'milky' 'unknown' 'fluoride' 'colore
d']
Missing Values: 0
```

"water_quality" column has two more values talking about abandoned water. We will keep this column and analyze it better and drop the "quality_group" column

In [28]:
```
water_data = water_data.drop("quality_group", axis= 1)
```

3.8 quantity / quantity_group

In [29]:
```
#checking quantity and quantity_group
column_checking(["quantity", "quantity_group"])
```

```
Feature Name: quantity
Number of Unique Values: 5
Unique Values: ['enough' 'insufficient' 'dry' 'seasonal' 'unknown']
Missing Values: 0


Feature Name: quantity_group
Number of Unique Values: 5
Unique Values: ['enough' 'insufficient' 'dry' 'seasonal' 'unknown']
Missing Values: 0
```

They both convey the same thing so we drop one

In [30]:
```
water_data = water_data.drop("quantity_group", axis= 1)
```

3.9 source/ source_type/ source_class

In [31]:
```
#checking water source
column_checking(["source", "source_type", "source_class"])
```

```
Feature Name: source
Number of Unique Values: 10
Unique Values: ['spring' 'rainwater harvesting' 'dam' 'machine dbh'
'other'
 'shallow well' 'river' 'hand dtw' 'lake' 'unknown']
Missing Values: 0


Feature Name: source_type
Number of Unique Values: 7
Unique Values: ['spring' 'rainwater harvesting' 'dam' 'borehole' 'o
ther' 'shallow well'
 'river/lake']
Missing Values: 0


Feature Name: source_class
Number of Unique Values: 3
Unique Values: ['groundwater' 'surface' 'unknown']
Missing Values: 0
```

The "source column has ten distinct values making it a superclass of the "source_type" column. The "source_class" column only tells us if the water source is river or a well. i.e surface or ground. We are going to drop the "source_type" column and keep the other two but we are not going to use "source_class" in our models but it can come handy in exploratory analysis

In [32]:
```
water_data = water_data.drop("source_type", axis= 1)
```

3.10 waterpoint_type / waterpoint_type_group

In [33]:
```
column_checking(["waterpoint_type", "waterpoint_type_group", "wpt_nan
```

```
Feature Name: waterpoint_type
Number of Unique Values: 7
Unique Values: ['communal standpipe' 'communal standpipe multiple'
'hand pump' 'other'
 'improved spring' 'cattle trough' 'dam']
Missing Values: 0


Feature Name: waterpoint_type_group
Number of Unique Values: 6
Unique Values: ['communal standpipe' 'hand pump' 'other' 'improved
spring'
 'cattle trough' 'dam']
Missing Values: 0


Feature Name: wpt_name
Number of Unique Values: 37400
Unique Values: ['none' 'Zahanati' 'Kwa Mahundi' ... 'Kwa Yahona Kuv
ala' 'Mshoro'
 'Kwa Mzee Lugawa']
Missing Values: 2
```

"waterpoint_type" column has an extra feature than the other column so we are gonna drop it

In [34]:
```
#dropping water_type_group
water_data = water_data.drop("waterpoint_type_group", axis= 1)
```

3.11 funder / installer

Now that we are done with similar columns lets inspect other columns to know if they are relevant to us before we sort missing values. We are going to print the first five rows to see the columns we are remaing with

In [35]:
```
water_data.head()
```

Out[35]:

id	status_group	amount_tsh	date_recorded	gps_height	installer	longitude	latitud
69572	functional	6000.0	2011-03-14	1390	Roman	34.938093	-9.85632
8776	functional	0.0	2013-03-06	1399	GRUMETI	34.698766	-2.14746
34310	functional	25.0	2013-02-25	686	World vision	37.460664	-3.82132
67743	non functional	0.0	2013-01-28	263	UNICEF	38.486161	-11.15529
19728	functional	0.0	2011-07-13	0	Artisan	31.130847	-1.82535

5 rows × 26 columns

In [36]:
```
water_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 59400 entries, 69572 to 26348
Data columns (total 26 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   status_group           59400 non-null  object
 1   amount_tsh             59400 non-null  float64
 2   date_recorded          59400 non-null  datetime64[ns]
 3   gps_height             59400 non-null  int64
 4   installer              55745 non-null  object
 5   longitude              59400 non-null  float64
 6   latitude               59400 non-null  float64
 7   wpt_name               59398 non-null  object
 8   num_private            59400 non-null  int64
 9   basin                  59400 non-null  object
 10  region                 59400 non-null  object
 11  lga                    59400 non-null  object
 12  population             59400 non-null  int64
 13  public_meeting         56066 non-null  object
 14  recorded_by            59400 non-null  object
 15  permit                 56344 non-null  object
 16  construction_year      59400 non-null  datetime64[ns]
 17  extraction_type_group  59400 non-null  object
 18  management             59400 non-null  object
 19  management_group       59400 non-null  object
 20  payment_type           59400 non-null  object
 21  water_quality          59400 non-null  object
 22  quantity               59400 non-null  object
 23  source                 59400 non-null  object
 24  source_class           59400 non-null  object
 25  waterpoint_type        59400 non-null  object
dtypes: datetime64[ns](2), float64(3), int64(3), object(18)
memory usage: 12.2+ MB
```

3.12 Looking at other columns

amount_tsh(Total_statistic_head)

"Total Static Head" or "Total Static Head (TSH)." It refers to the total amount of water available or stored at a water point, typically in a static or non-moving state. This value signifies the water's pressure at the water point or the maximum height that the water can reach above the water source.

In [37]:
```
water_data["amount_tsh"].max()
```

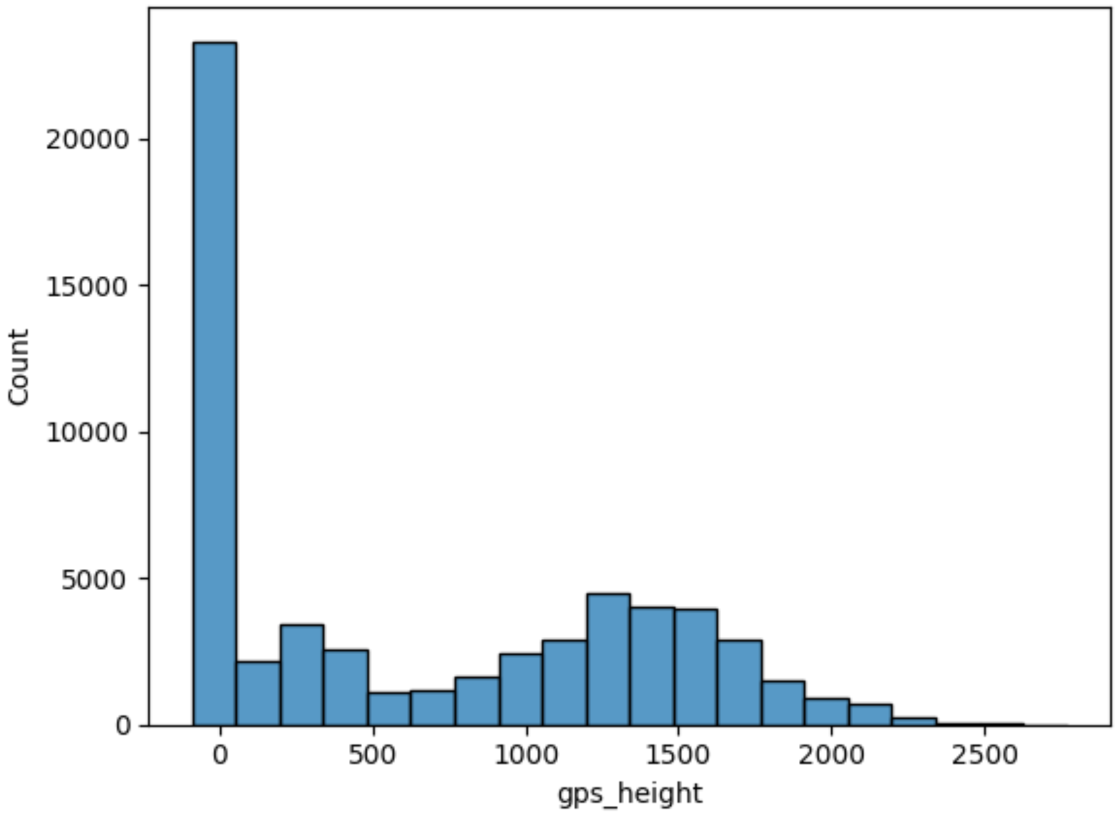
Out[37]:
```
350000.0
```

The total statistic per head column has 41639 records indicating the static_head per well is 0. We will have to further analyze this column with other columns like water quality to determine why the many zeros and if this column is relevant for our analysis

3.13 gps_height- Altitude of the well

This column typically represents the altitude or elevation of a well or a water point above sea level, recorded using GPS (Global Positioning System) technology. This value signifies the height or vertical distance of the water point from the Earth's mean sea level.t is usually measured in meters or feet, denoting the well's position in relation to sea level.

In [38]:
```
# water_data["gps_height"].value_counts()
sns.histplot(x = "gps_height", data = water_data, bins = 20);
```



This column is important in understanding the altitudes of our water wells and its relation tour our analysis. We will keep it for use in our EDA.

3.14 longitude/ latitude

This column typically represents the geographical coordinates of water wells or water points, recorded using GPS (Global Positioning System) technology. These coordinates provide the specific geographic location of the water points on the Earth's surface.We are going to use it in our exploratory analysis but not in our modeling so we keep it for now

3.15 num_private - Number of private water wells

This column in water-related datasets often represents the number of privately owned water points or the number of privately owned water sources associated with a particular water point

In [39]:
```
column_checking(["num_private"])
```

```
Feature Name: num_private
Number of Unique Values: 65
Unique Values: [   0   39    5   45    6    3  698   32   15    7
25  102    1   93
   14   34  120   17  213   47    8   41   80  141   20   35  131
4
   22   11   87   61   65  136    2  180   38   62    9   16   23
42
   24   12  668  672   58  150  280  160   50 1776   30   27   10
94
   26  450  240  755   60  111  300   55 1402]
Missing Values: 0
```

It purely contains numbers stating the number of wells owned privately in the region. This column might lack explicit information, making its interpretation challenging so we drop it.

In [40]:
```
water_data = water_data.drop("num_private", axis = 1)
```

3.16 wpt_name - Name of the waterpoint if there is one

It contains textual information that identifies or labels individual water points, wells, or water sources, if such names exist.

In [41]:
```
column_checking(["wpt_name"])
```

```
Feature Name: wpt_name
Number of Unique Values: 37400
Unique Values: ['none' 'Zahanati' 'Kwa Mahundi' ... 'Kwa Yahona Kuv
ala' 'Mshoro'
 'Kwa Mzee Lugawa']
Missing Values: 2
```

The column has 37400 records of unique water names in Tanzania. This column is helps us know the number of wells under study and their names but not useful in our analysis so we drop it

In [42]:
```
water_data = water_data.drop("wpt_name", axis= 1)
```

3.17 Public_meeting

The "public_meeting" column in water-related datasets typically represents whether there was a public meeting held to discuss water issues or water-related projects in the area associated with a particular water point.

In [43]:
```
water_data["public_meeting"].value_counts()
```

Out[43]:
```
public_meeting
True     51011
False     5055
Name: count, dtype: int64
```

We are going to keep this column for our analysis. It helps us further understand our analysis

3.18 recorded_by

Represents the entity or group responsible for entering or recording the data for each row in the dataset. This column often contains information about the organization, individual, or group that performed the data entry or data collection process.

In [44]:
```
water_data["recorded_by"].value_counts()
```

Out[44]:
```
recorded_by
GeoData Consultants Ltd    59400
Name: count, dtype: int64
```

This dataset was collected by the GeoData Consultants Ltd. Since we already know the group responsible for this data we can drop it since we wouldn't need it in our analysis

In [45]:
```
water_data = water_data.drop("recorded_by", axis= 1)
```

3.19 permit

What this column indicates is whether a water point or water source has the necessary legal or official permit or authorization to operate.

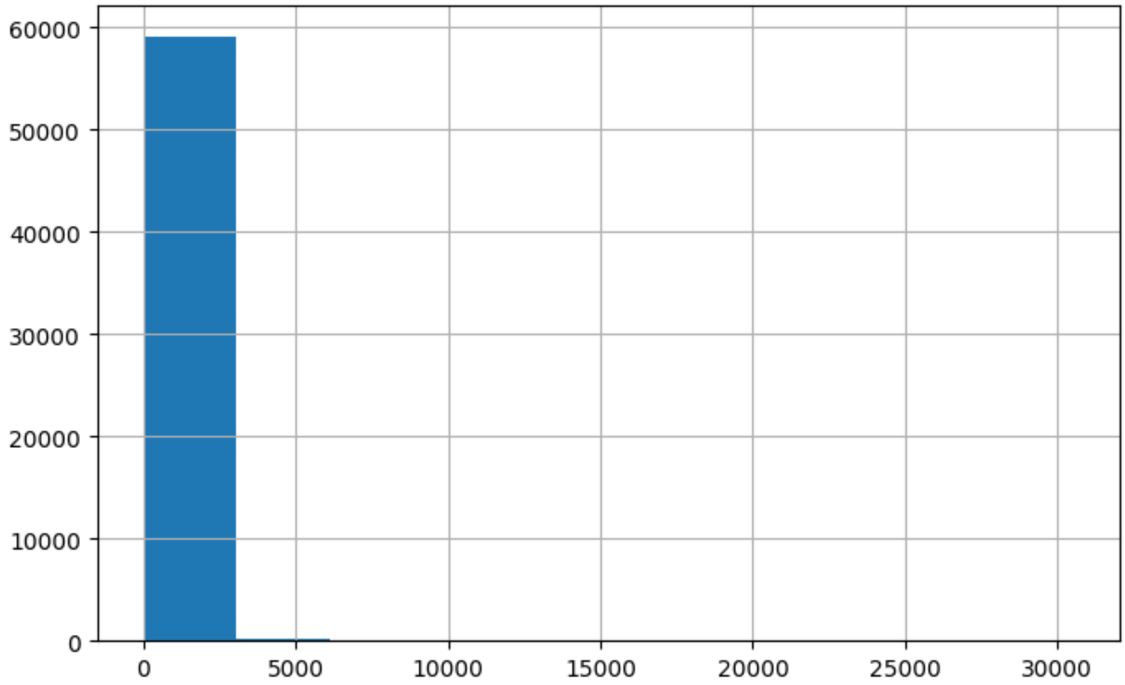
In [46]:
```
water_data["permit"].value_counts()
```

Out[46]:
```
permit
True     38852
False    17492
Name: count, dtype: int64
```

3.20 population

This column refers to the number of people residing within a certain proximity to the

In [47]:
```
water_data["population"].hist(figsize = (8,5));
```



It is important column since it helps in our analysis of predicting whether a pump is functional, non-functional or functional but needs repair

3.21 date_recorded, construction_year, id

This columns are not important or needed for our analysis so we drop them

In [48]:
```
water_data = water_data.drop(["date_recorded", "construction_year"],
```

Resetting our index so it can be on the same axis with other columns and dropping it since we don't need it in our analysis.

In [49]:
```
#reseting the index and dropping it
water_data = water_data.reset_index(drop=True)
```

Let us have a look on how our new data looks like after dropping the columns we did see not fit for our analysis. We are going to inspect the first five rows

In [50]:
```
#inspecting our new dataframe
water_data.head()
```

Out[50]:

	status_group	amount_tsh	gps_height	installer	longitude	latitude	basin	regio
0	functional	6000.0	1390	Roman	34.938093	-9.856322	Lake Nyasa	Iring
1	functional	0.0	1399	GRUMETI	34.698766	-2.147466	Lake Victoria	Mar
2	functional	25.0	686	World vision	37.460664	-3.821329	Pangani	Manyar
3	non functional	0.0	263	UNICEF	38.486161	-11.155298	Ruvuma / Southern Coast	Mtwar
4	functional	0.0	0	Artisan	31.130847	-1.825359	Lake Victoria	Kager

5 rows × 21 columns

In [51]:
```
#data info
water_data.info()
```

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

Summary statistics of our numeric columns

In [52]:
```
#summary statistics
water_data.describe()
```

Out[52]:

	amount_tsh	gps_height	longitude	latitude	population
count	59400.000000	59400.000000	59400.000000	5.940000e+04	59400.000000
mean	317.650385	668.297239	34.077427	-5.706033e+00	179.909983
std	2997.574558	693.116350	6.567432	2.946019e+00	471.482176
min	0.000000	-90.000000	0.000000	-1.164944e+01	0.000000
25%	0.000000	0.000000	33.090347	-8.540621e+00	0.000000
50%	0.000000	369.000000	34.908743	-5.021597e+00	25.000000
75%	20.000000	1319.250000	37.178387	-3.326156e+00	215.000000
max	350000.000000	2770.000000	40.345193	-2.000000e-08	30500.000000

The shape of our data

In [53]:
```
#shape
water_data.shape
```

Out[53]: (59400, 21)

We have 59400 rows and 21 columns in our data

4. Dealing with missing values in our dataset.

We have three columns with missing values which are "installer", "public_meeting" and "permit". All this columns are categorical so we can impute the missing values with the most frequent value in the columns. We are going to use a simple imputer from sklearn module to help us do this job

In [54]:
```
#columns to impute
columns_to_impute = ["installer","permit", "public_meeting"]
#using pipeline to fill missing values along with simple imputer
pipeline_imputer = Pipeline( steps= [
    ("imputer", SimpleImputer(strategy= "most_frequent"))
                            ])
#calling ColumnTransformer
preprocessor = ColumnTransformer(transformers=[
    ("imputer", pipeline_imputer, columns_to_impute)]
                                )
#fitting our data
imputed_data = preprocessor.fit_transform(water_data)
#putting our imputed columns into a DataFrame
imputed_data = pd.DataFrame(imputed_data, columns= (["installer", "p
```

We are going to drop the columns with missing values and replace them with our new imputed DataFrame. The new DataFrame contains the same values as the columns we are dropping but has been imputed and has no missing values.

In [55]:
```
#dropping the initial columns
water_data = water_data.drop(["installer", "permit", "public_meeting
#concat the two df
complete_data = pd.concat([water_data, imputed_data], axis= 1)
#inspecting new dataframe
complete_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59400 entries, 0 to 59399
Data columns (total 21 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   status_group           59400 non-null  object 
 1   amount_tsh             59400 non-null  float64
 2   gps_height             59400 non-null  int64  
 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   lga                    59400 non-null  object 
 8   population             59400 non-null  int64  
 9   extraction_type_group  59400 non-null  object 
 10  management             59400 non-null  object 
 11  management_group       59400 non-null  object 
 12  payment_type           59400 non-null  object 
 13  water_quality          59400 non-null  object 
 14  quantity               59400 non-null  object 
 15  source                 59400 non-null  object 
 16  source_class           59400 non-null  object 
 17  waterpoint_type        59400 non-null  object 
 18  installer              59400 non-null  object 
 19  permit                 59400 non-null  object 
 20  public_meeting         59400 non-null  object 
dtypes: float64(3), int64(2), object(16)
memory usage: 9.5+ MB
```

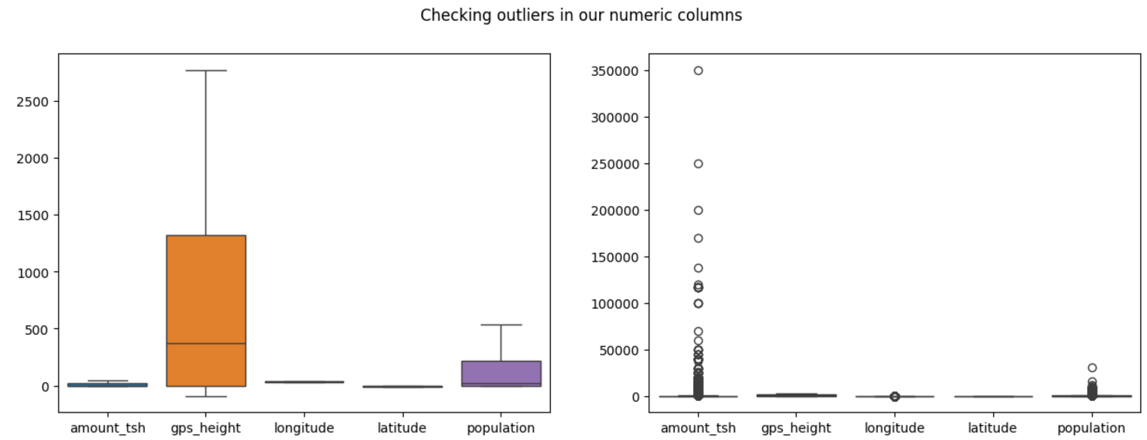
Perfecto! All our columns are now complete with none having any missing record. Our data is ready now for exploratory analysis but before that we are going to tweak some few columns.

In [56]:
```
#dict to replace
to_replace = {"Wami / Ruvu": "Wami", "Ruvuma / Southern Coast": "Ruvu
complete_data.loc[:, "basin"].replace(to_replace)
#replacing "other - school" with "school"
complete_data.loc[:, "management"].replace("other - school", "School
```

4.2 Inspecting outliers in our numerical columns

In [57]:
```
numeric = complete_data.select_dtypes("number")
fig, ax = plt.subplots(ncols=2, figsize = (15, 5))
plt.suptitle("Checking outliers in our numeric columns")
sns.boxplot(numeric, showfliers = False, ax= ax[0]);
sns.boxplot(numeric, showfliers = True, ax = ax[1])
```

Out[57]: <Axes: >



Our "amount_sh" column seems to have an outlier. The left plot shows how the numerical columns should behave without outliers and the right plot depicts the spread in our columns. We will sort the outlier in the affected column using the inter-quartile range method of solving outliers

In [58]:
```
#settting lower quantie
low_quant = complete_data["amount_tsh"].quantile(0.25)
#setting upper quantile
upp_quant = complete_data["amount_tsh"].quantile(0.75)
#inter-quartile range
IQR = upp_quant - low_quant
#creating the clipping points
low_limit = low_quant - 1.5 * IQR
upp_limit = upp_quant + 1.5 * IQR
#clipping the outlier
complete_data["amount_tsh"].clip(lower = low_limit, upper = upp_limi
```

In [59]:
```
complete_data["amount_tsh"].max()
```

Out[59]: 50.0

5. Exploratory Data Analysis (EDA)

Now that our data is clean and ready for use, we are going to start exploratory analysis of our data. Plotting graphs and visuals is the main thing we are doing here in order to further understand visually what our data is communicating.

5.1 Univariate EDA

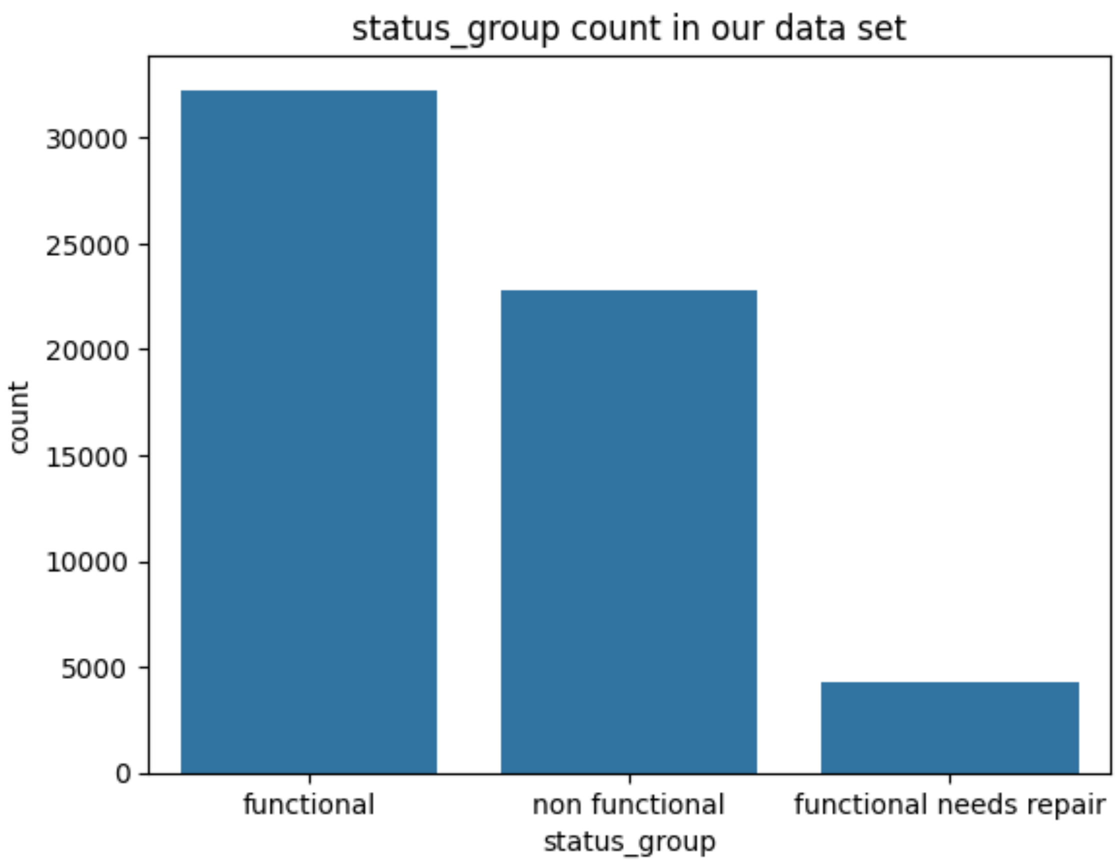
Writting two helper functions to helps us plot countplots for our univariate analysis.

In [60]:
```
#writting helper function to help us make x-axis countplots in our E
def sns_xcount(column , data):
    sns.countplot(x = column, data = data)
    plt.title(f"{column} count in our data set")
    plt.show();

#writting helper function to help us make y-axis countplots in our E
def sns_ycount(column , data):
    sns.countplot(y = column, data = data)
    plt.title(f"{column} count in our data set")
    plt.show();
```

The Distribution of our target class ("status_group)

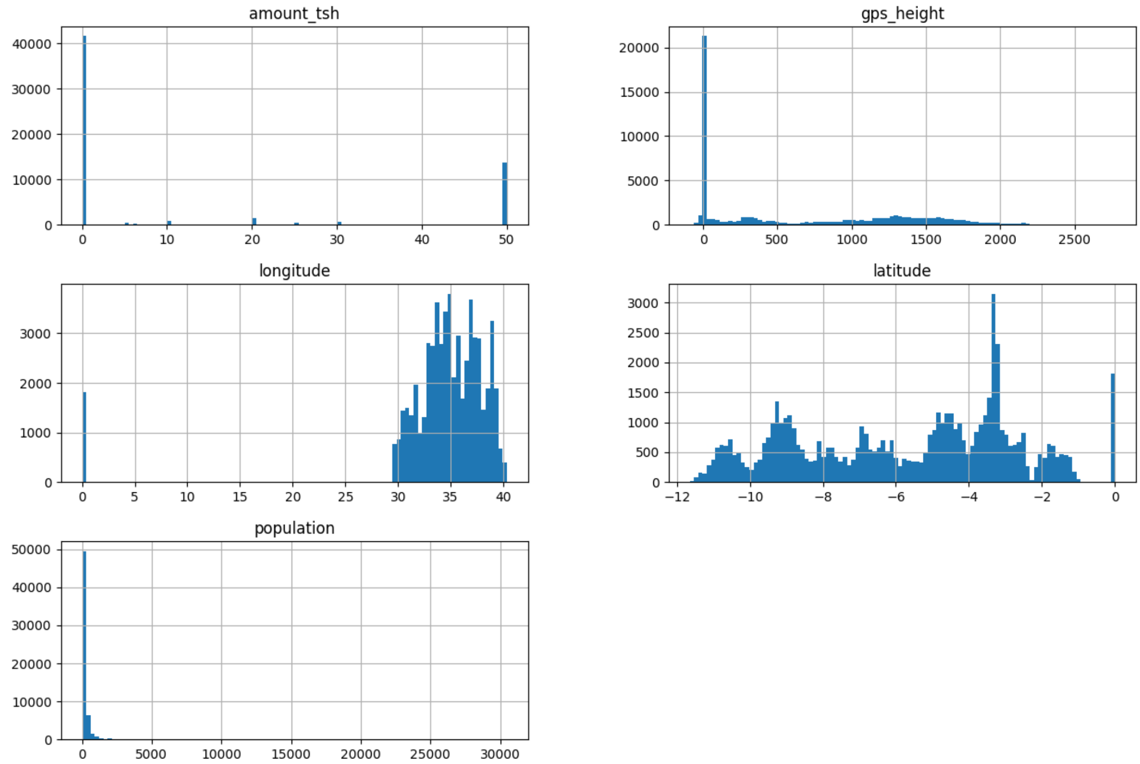
In [61]:
```
sns_xcount("status_group", complete_data)
```



We have three target class in our status column namely functional, non-functional and functional needs repair. The functional class is the highest with 32259 pumps followed by non-functional class with 22824 pumps and the least class is thefunctional needs repair with 4317 pumps

The spread of our numerical columns

In [62]:
```
#numeric columns histograms
complete_data.hist(figsize= (15, 10), bins= 100);
```



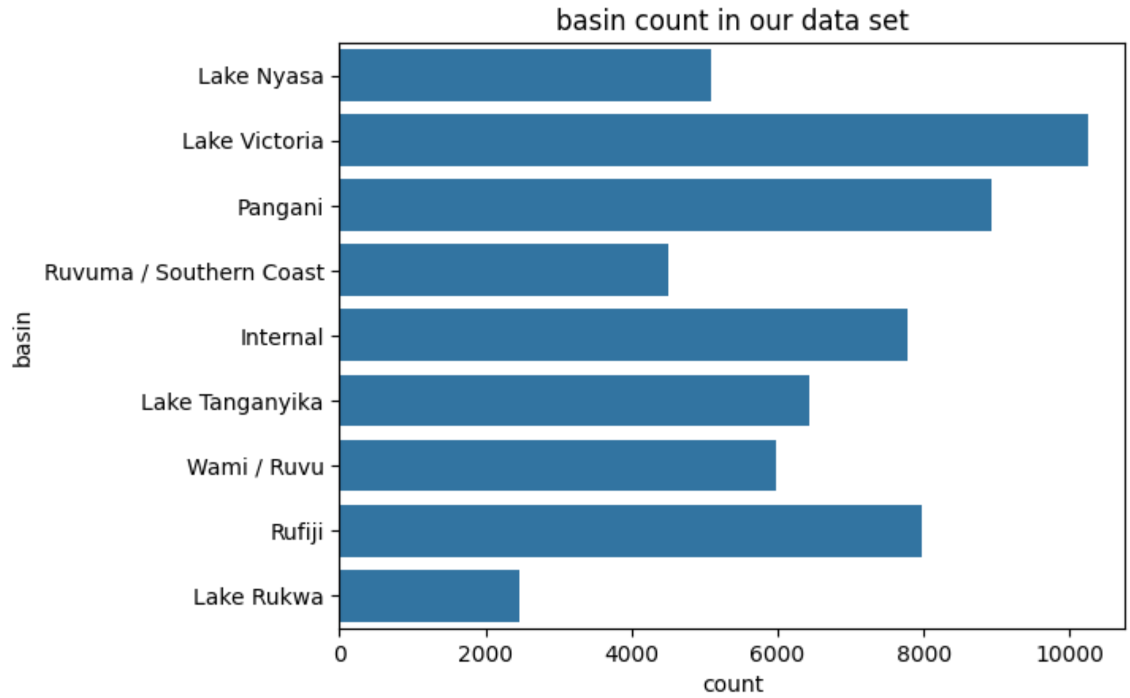
The "amount_tsh" column typically represents the pressure exerted by a water column due to gravity and is usually measured in meters or feet. The top-left plot shows us that 0 "amount_tsh" is the most common value in our data. A value of 0, it often indicates that there is no static head or minimal pressure available at the water source and situation might imply several scenarios such as; insufficient water, non-functioning pump missing data or unknown values. I will analyze it further in our multivariate analysis

Distribution of the altitude("gps_height) of the wells also has 0 as the highest group but has other values spread across a range of 0 to 2500metres above sea level.I will also analyze this column further since 0 might indicate a water source is at sea leavel or indicates missing values or unknown

The "population" variable typically refers to the estimated or recorded number of people living in the vicinity or area surrounding a well or a water source. Zero is the popular group in this column also and this can indicate many things such as unhabited area or missing values etc. I will also do further analysis on this column

basin - Geographic water basin

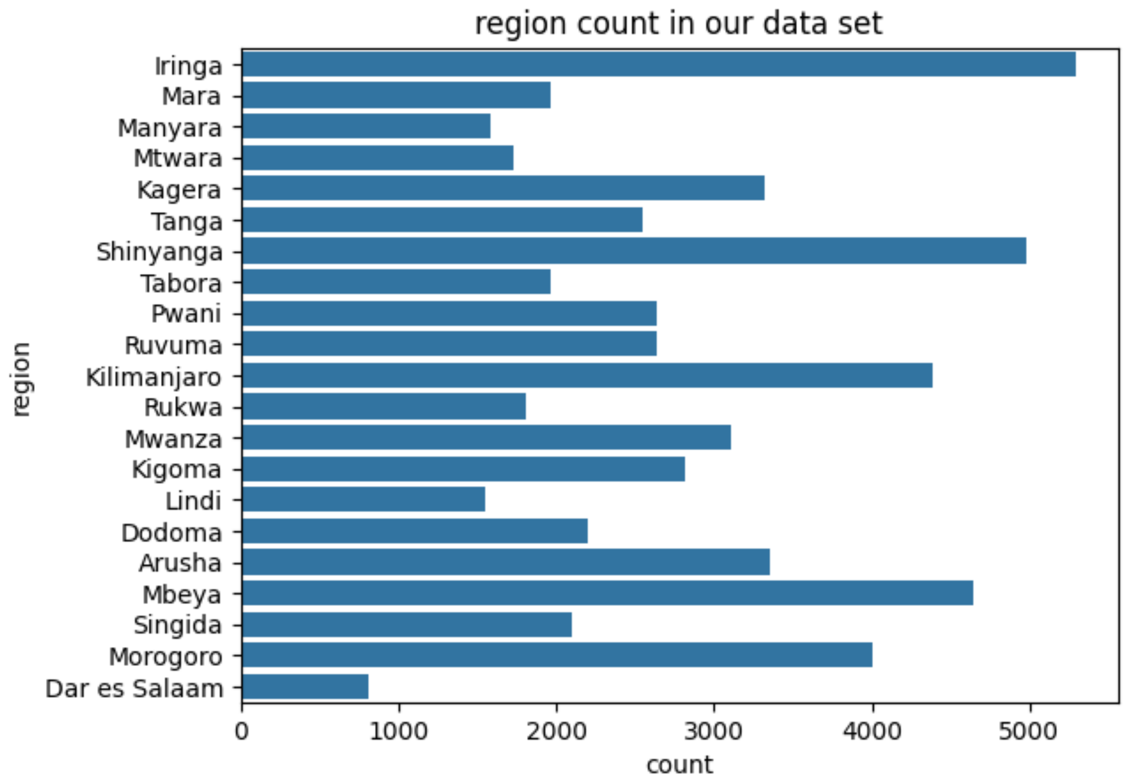
In [63]:
```
sns_ycount("basin", complete_data)
```



The water basins geographically located in Tanzania are nine in total. Lake Victoria is the largest in terms of body mass and Lake Rukwa holds the last position

The regions under study

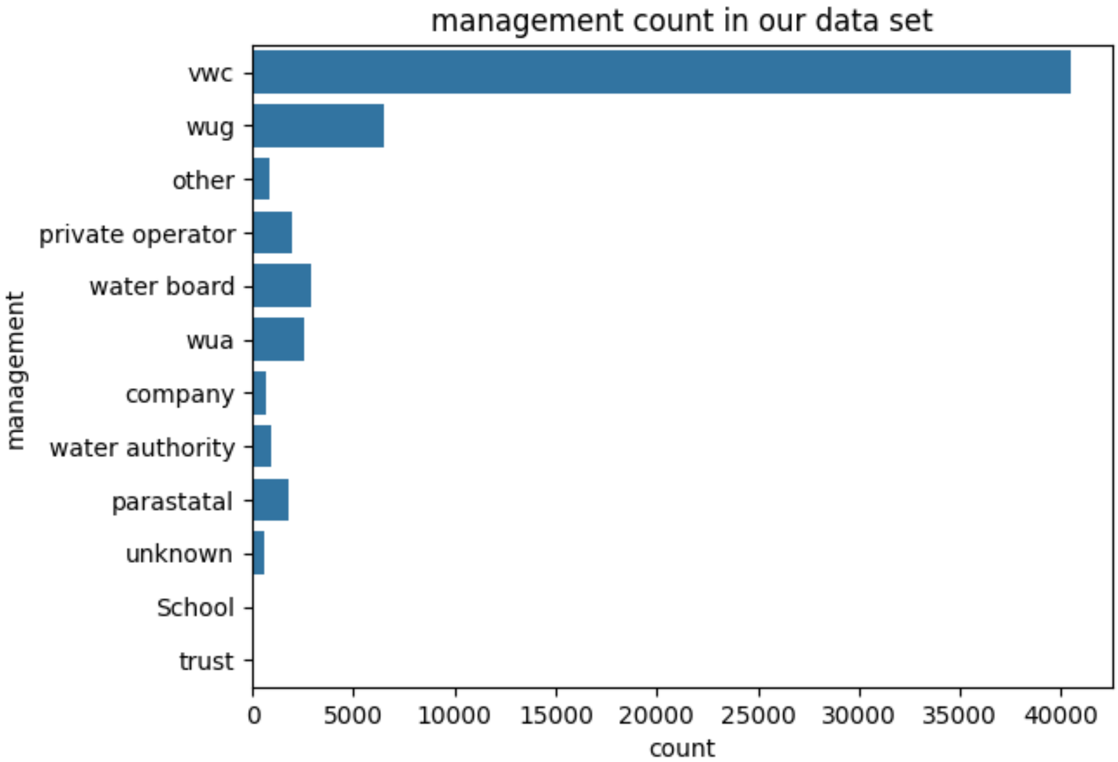
In [64]:
```
sns_ycount("region", complete_data)
```



We have 21 regions under study in our dataset. These regions represent where our water wells under research are located. Iringa has the most water points indicating that there is a large population living in this region. Dar es Salaam has the least count of water wells and this can indicate low population, insufficient water or poor water quality and quantity in this region.

Management

In [65]:
```
sns_ycount("management", complete_data);
```

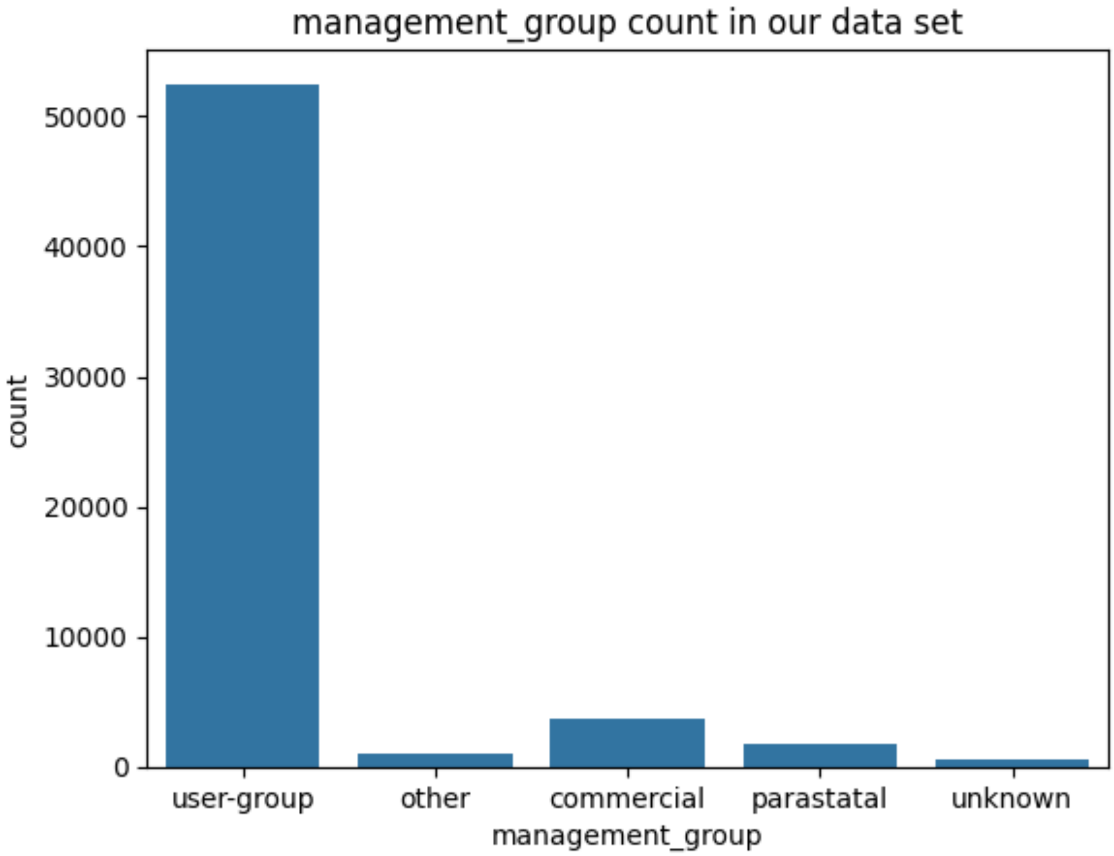


The management columns tells us about the people incharge of running the water wells located in Tanzania. Most water wells in Tanzania are managed by the Village Water Community(VWC). This makes a lot of sense sinse Tanzania is a communist country. School and trust hold the least share for the water well management. The abbreviations of the values are explained here;

1. VWC - Village Water Community
2. WUG - Water User Group
3. WUA - Water User Association Both WUG and WUA are mainly involved in activities such as irrigation, maintaining water infrastructure, resolving issues related to water usage among other things

Management groups

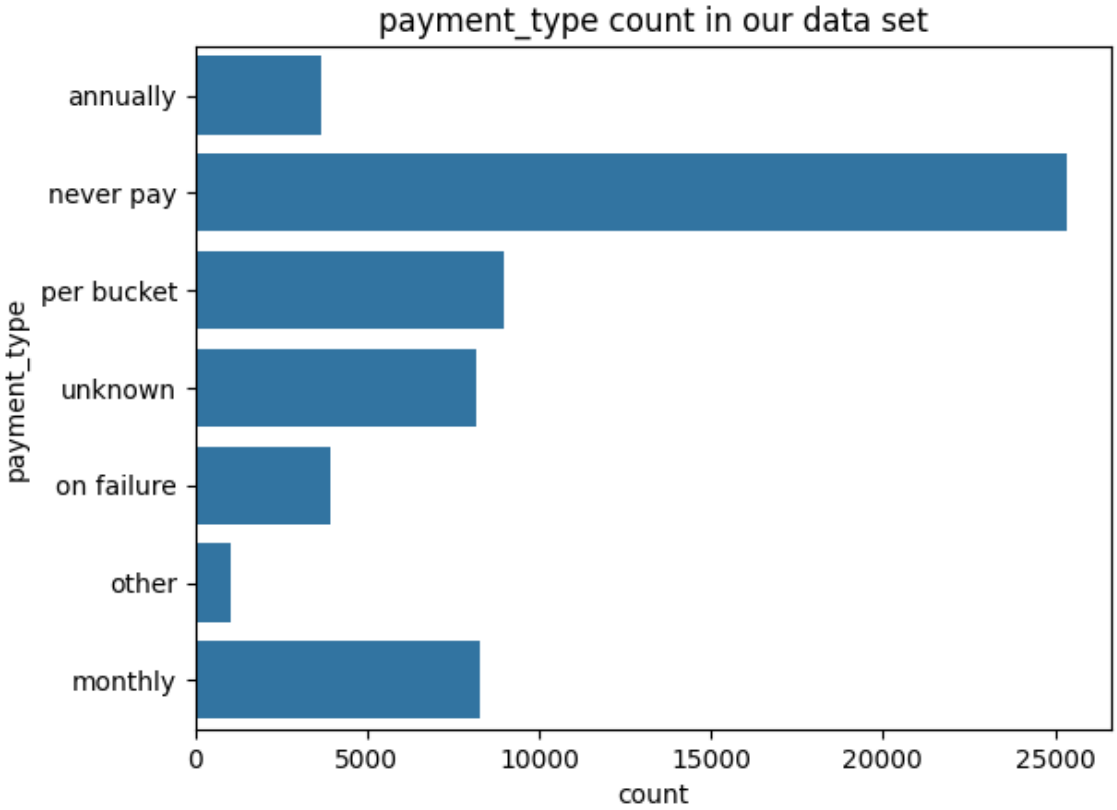
In [66]:
```
sns_xcount("management_group", complete_data)
```



When the management group are further categorised or grouped we obtain 5 major classes of management bodies. Communism being a norm in this country user-group category has the winning hand.

Payment type

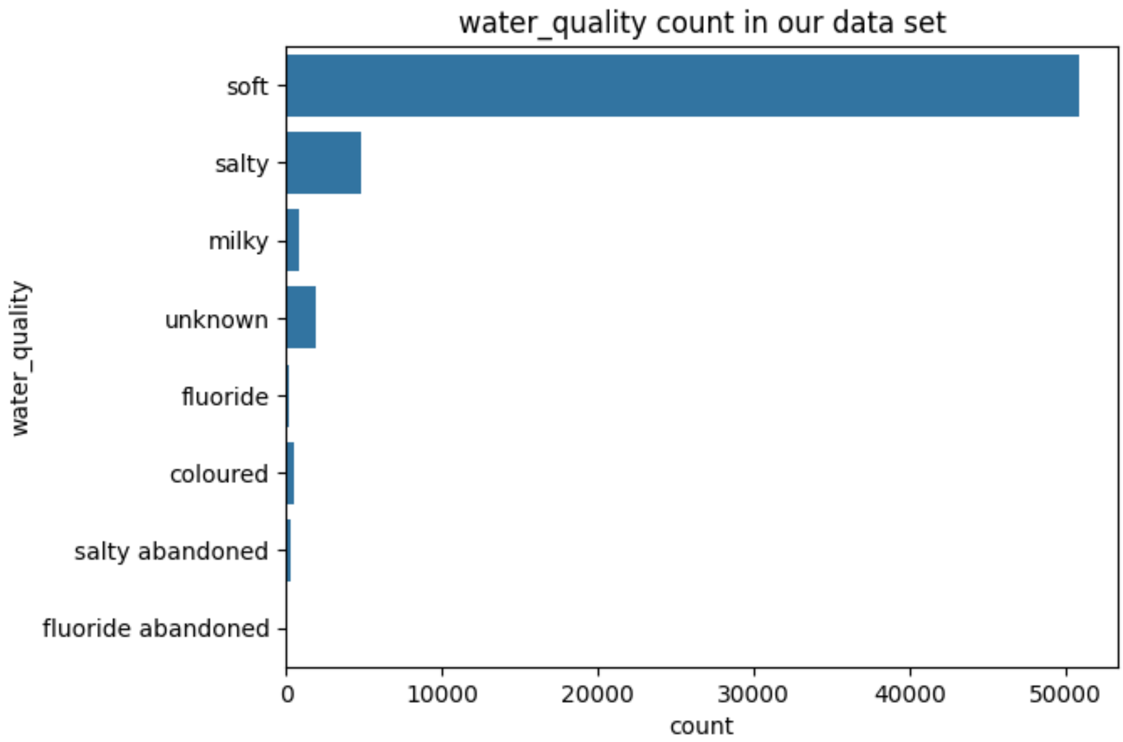
In [67]:
```
sns_ycount("payment_type", complete_data)
```



The means of payment for water in Tanzania has seven unique modes of payment. Most of the water wells are free and no charges are needed to access water. This is what the plot says as we can see most common mode of payment is "never pay". Our other plots on management concur with this as we can see most water wells are managed by the village community

Water quality

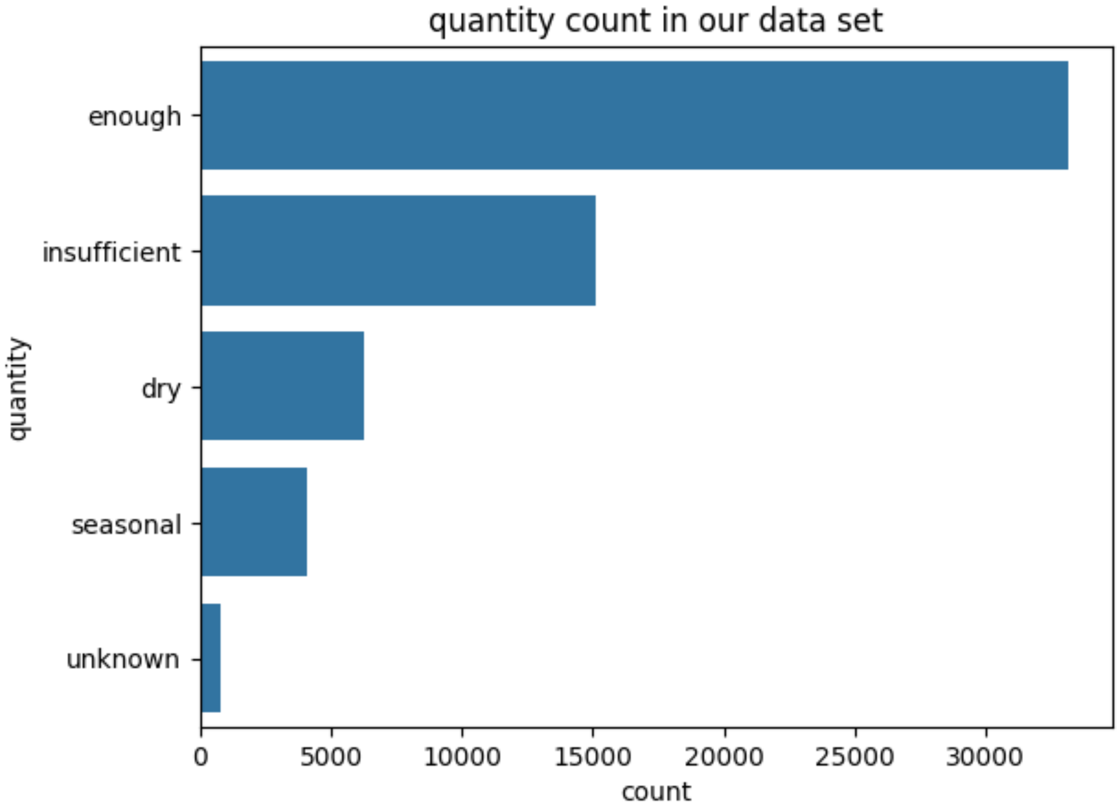
In [68]:
```
sns_ycount("water_quality", complete_data)
```



In terms of water quality, Tanzania is blessed with soft water. Soft water generally contains low concentrations of minerals like calcium and magnesium, leading to fewer issues related to limescale buildup in pipes and appliances. In Tanzania, having soft water can contribute to reduced instances of scale accumulation in plumbing systems and appliances, potentially minimizing maintenance and extending the lifespan of water-related infrastructure.

Water quantity

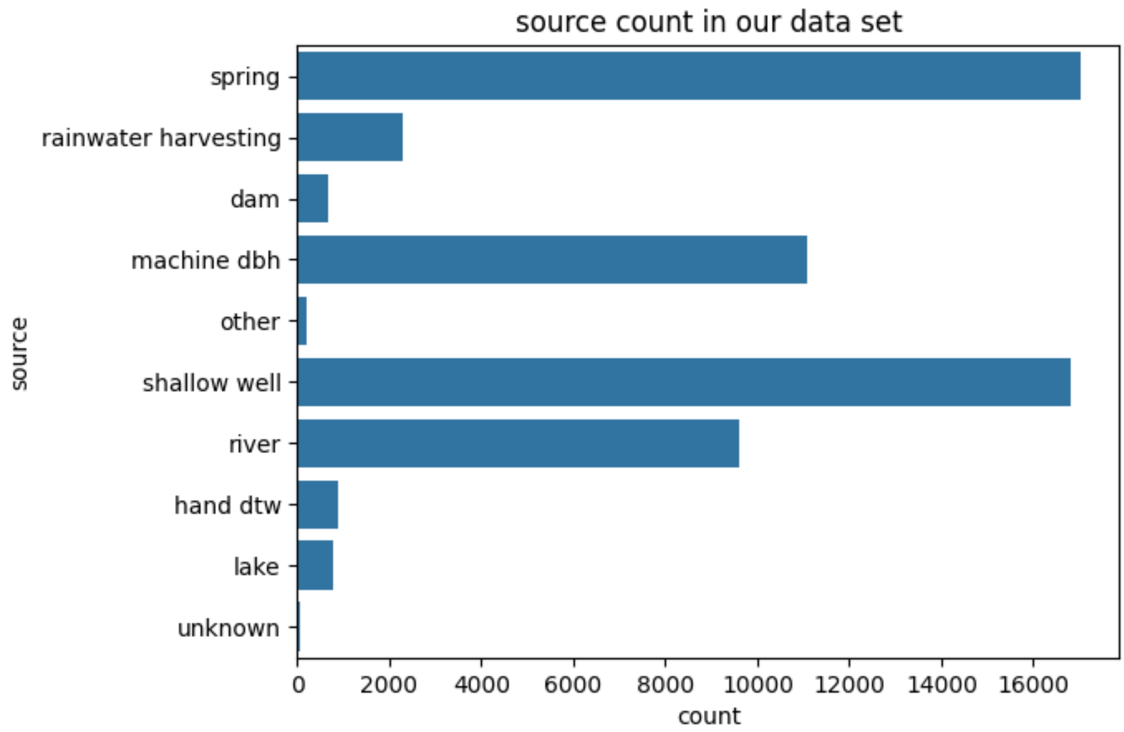
In [69]:
```
sns_ycount("quantity", complete_data)
```



From the dataset and visual above we can say that most water wells in Tanzania have enough water to sustain the population around it.We can see most wells holds enough capacity to meet the societies needs. It can also indicate that most of them are drought resistant. We can also see that almost a quarter of the wells contain insufficient water capacity to serve the people's needs

Source

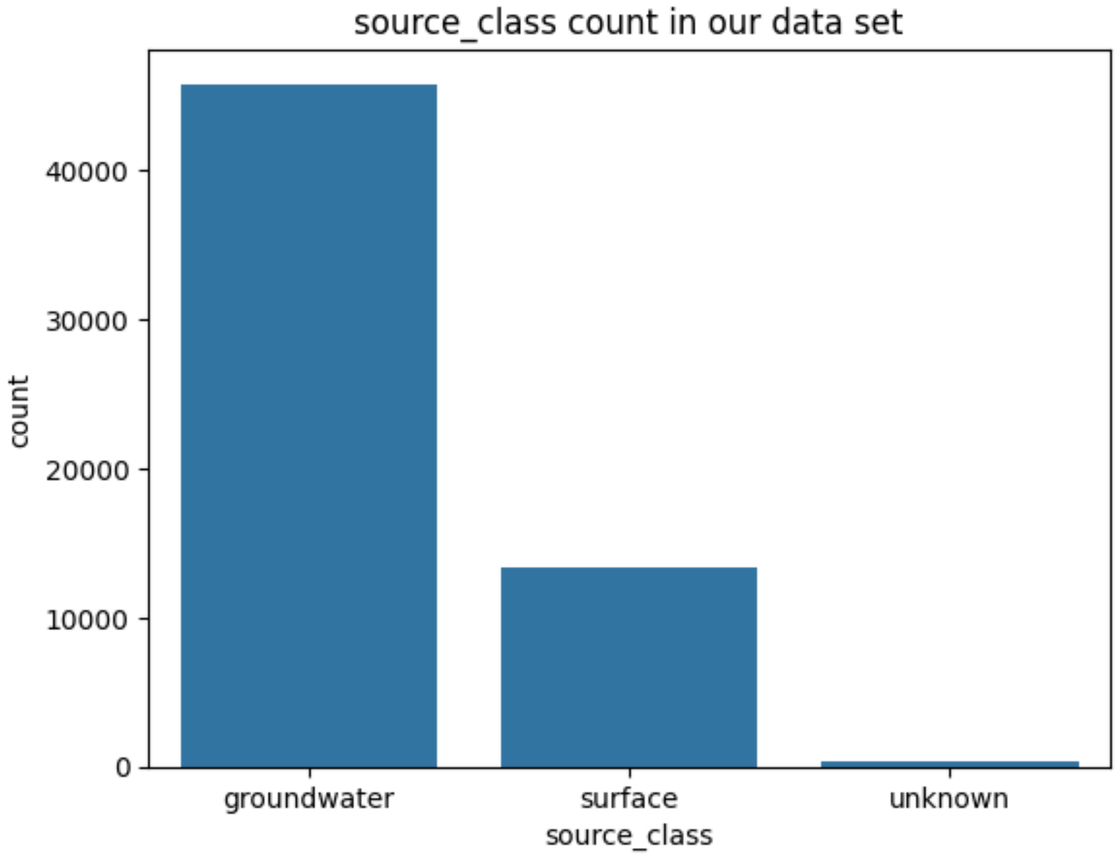
In [70]:
```
sns_ycount("source", complete_data)
```



Spring and shallow well are the most prolific water sources in Tanzania. Boreholes and rivers come in second and third respectively. Rain harvesting is a little bit lower and this can indicate that most households are not modern and rely on grass thatched houses making it impossible to harvest rain water

Source class

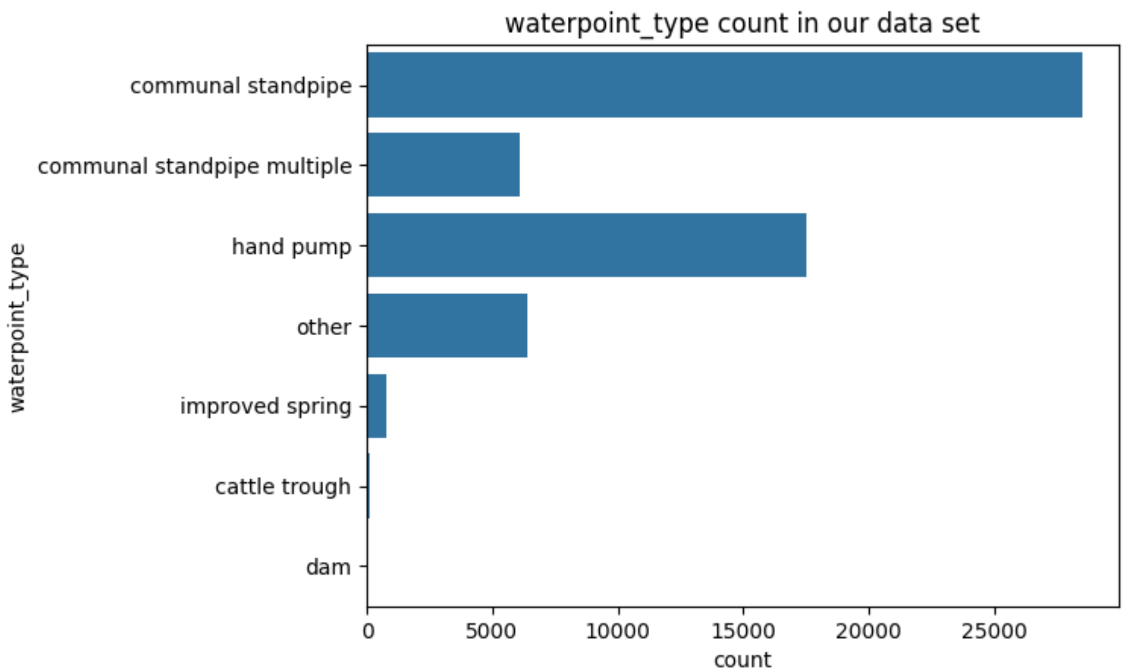
In [71]:
```
sns_xcount("source_class", complete_data)
```



Sources of water with similar type of occurence can be further grouped mainly into groundwater and surface. The unknown are significantly minimal

Waterpoint type

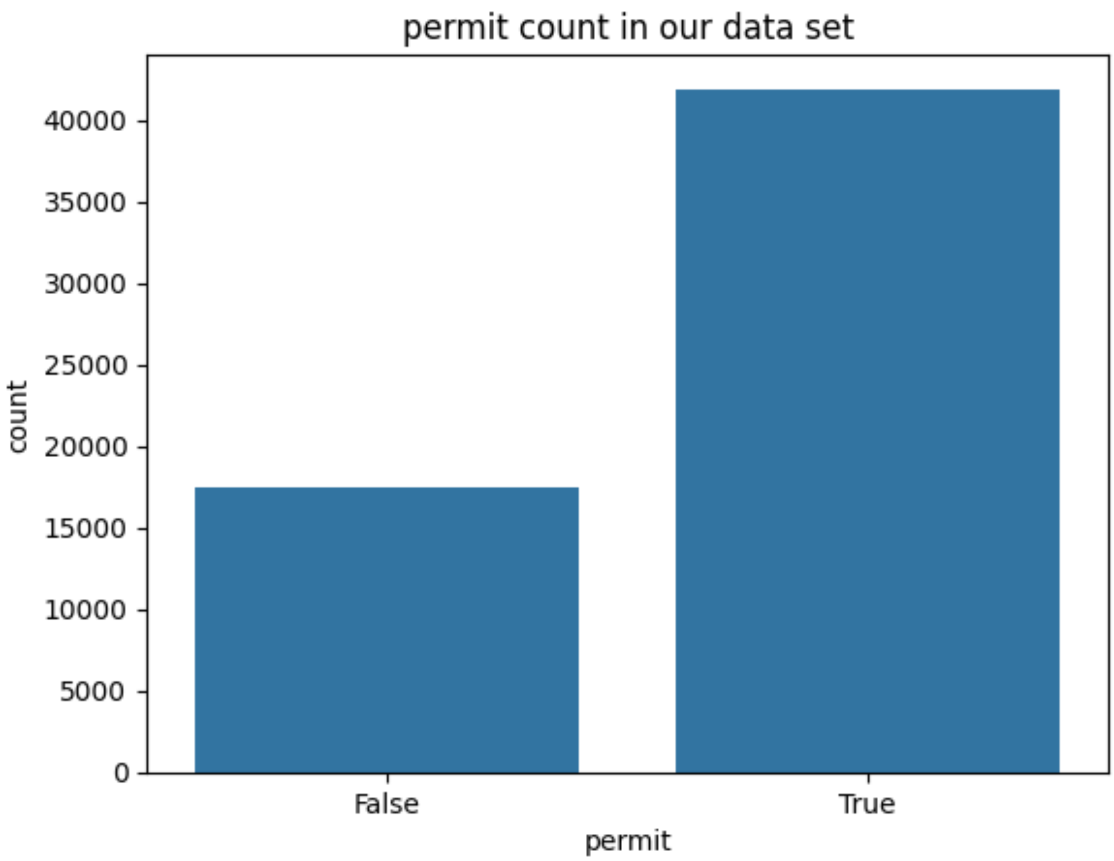
In [72]:
```
sns_ycount("waterpoint_type", complete_data)
```



As we have seen earlier that communism is the spirit in Tanzania, we can also see here that most water wells have communal standpipe as the infrastructure designed to provide access to data.The hand pump, communal standpipe multiple are also very popular designs of accessing water.

Permit

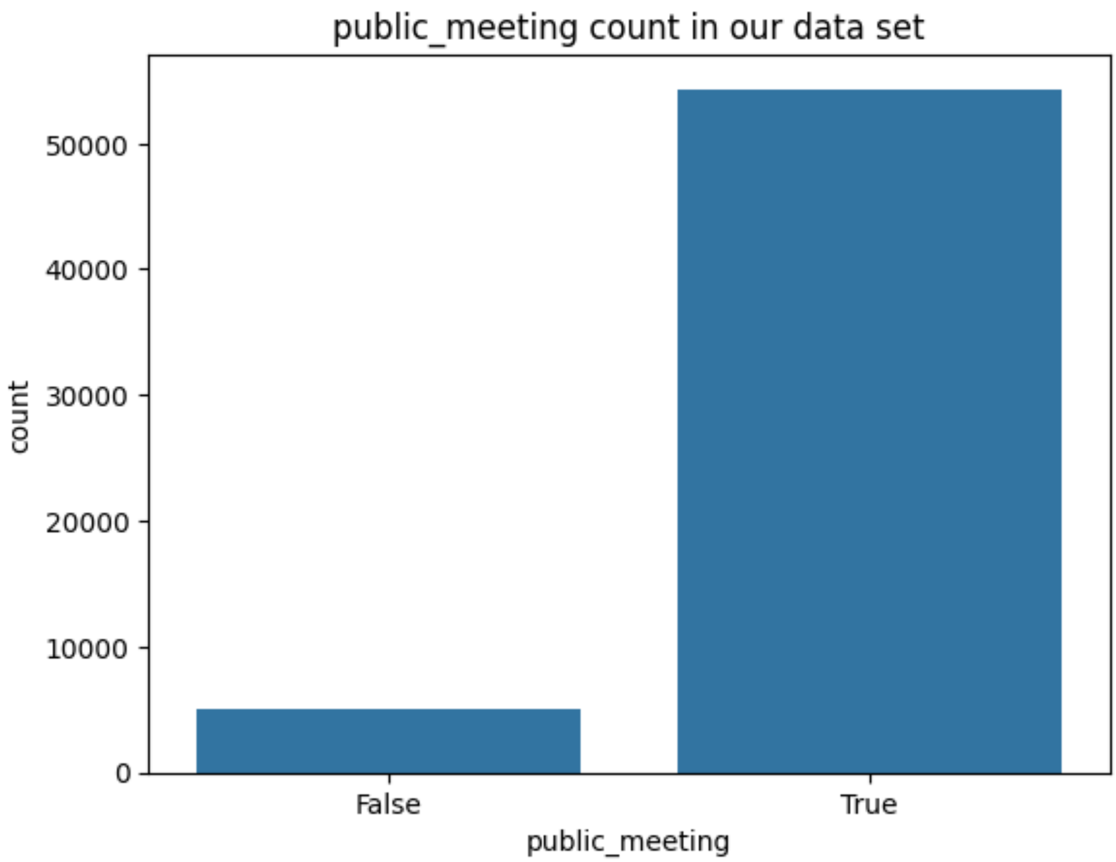
In [73]:
```
sns_xcount("permit", complete_data)
```



Most water wells in Tanzania have permits to operate and provide water. Our visual indicates that most water points have obtained the necessary permission, authorization, or legal approval to operate from the appropriate governing bodies. This may involve compliance with regulations and standards set by local or national authorities.The water points with no permits may indicate that the water source is operating without official permission, potentially raising concerns about its compliance with safety, health, or regulatory standards.

Public meeting

In [74]:
```
sns_xcount("public_meeting", complete_data)
```



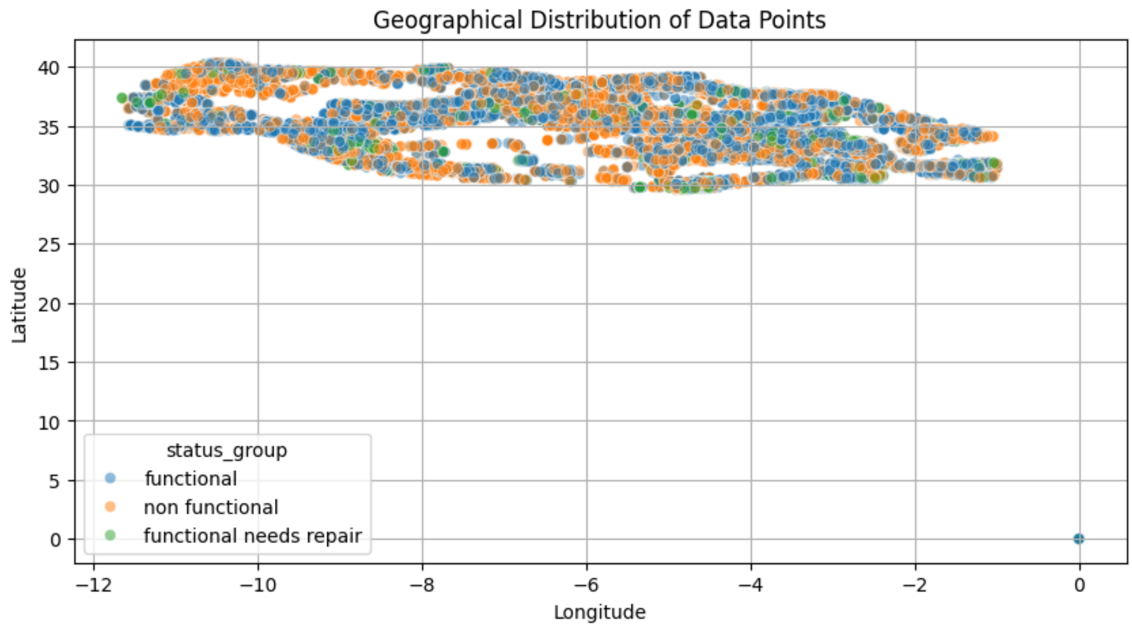
There is public participation whenever issues or concerns about the water points arise. From the data True means that a public meeting or community gathering has occurred to discuss matters related to water services, infrastructure, or projects such as waterquality, maintenance etc. False means there hasn't been a recorded public meeting or community gathering related to water-related issues or projects. This can indicate a lack of organized discussions or forums within the community about water related issues.

5.2 Multivariate EDA

The process here involves examining relationships, patterns, and interactions between multiple variables simultaneously within our dataset.

First let us plot the scatter map of our data and see the spread of water pumps across Tanzania's geographical land mass

In [75]:
```
#plotting latitudes and longitudes to check the geographcal represen
plt.figure(figsize=(10, 5))
sns.scatterplot(y ='longitude', x= 'latitude',alpha=0.5, c='blue',
                hue= "status_group", data = complete_data)
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Geographical Distribution of Data Points')
plt.grid(True)
plt.show()
```



From the scatter plot we can say that the pumps status across the region is almost even. The functional class is the highest followed by the non functional class and the ones that function but need repair are significantly low. The white spaces can be accounted as large water bodies such as lakes within the region

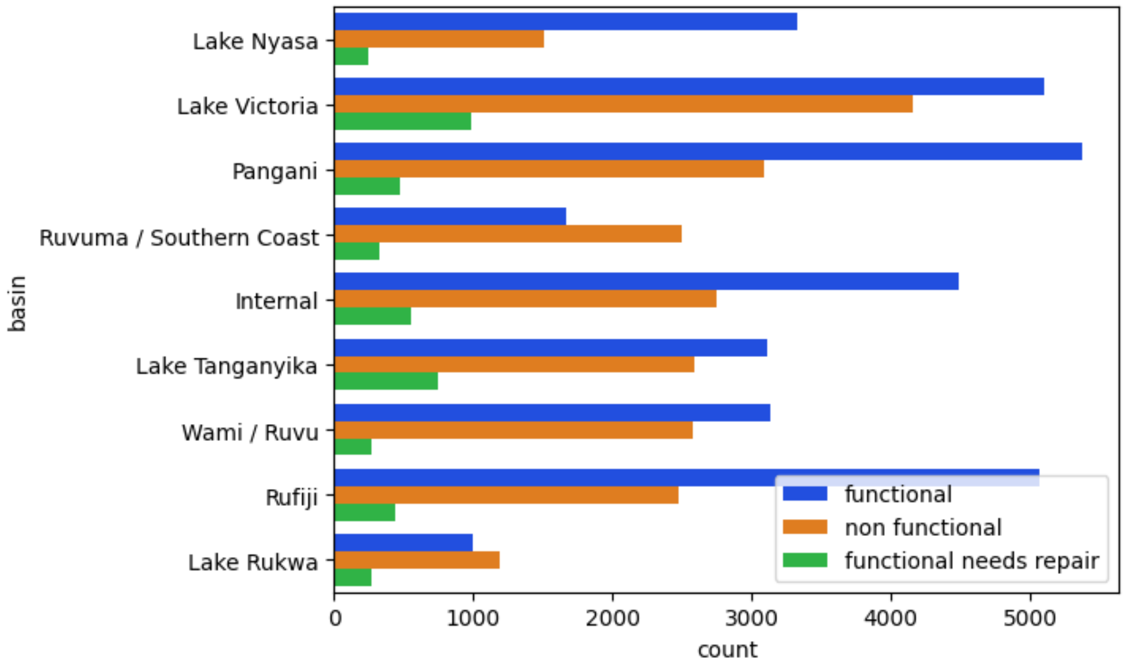
We are now going to write two helper functions to help us plot the relationship between two or more columns in our data.

In [76]:
```
#helper function for plotting multivariate countplots
def multivariate_ycountplot(column, data):
    sns.countplot(y = column, data = data, hue= "status_group", palet
    plt.legend(loc= "best")

def multivariate_xcountplot(column, data):
    sns.countplot(x = column, data = data,  hue= "status_group", pal
    plt.xticks(rotation = 45)
    plt.legend(loc= "best")
```

Relationship betweeen Status group and basin

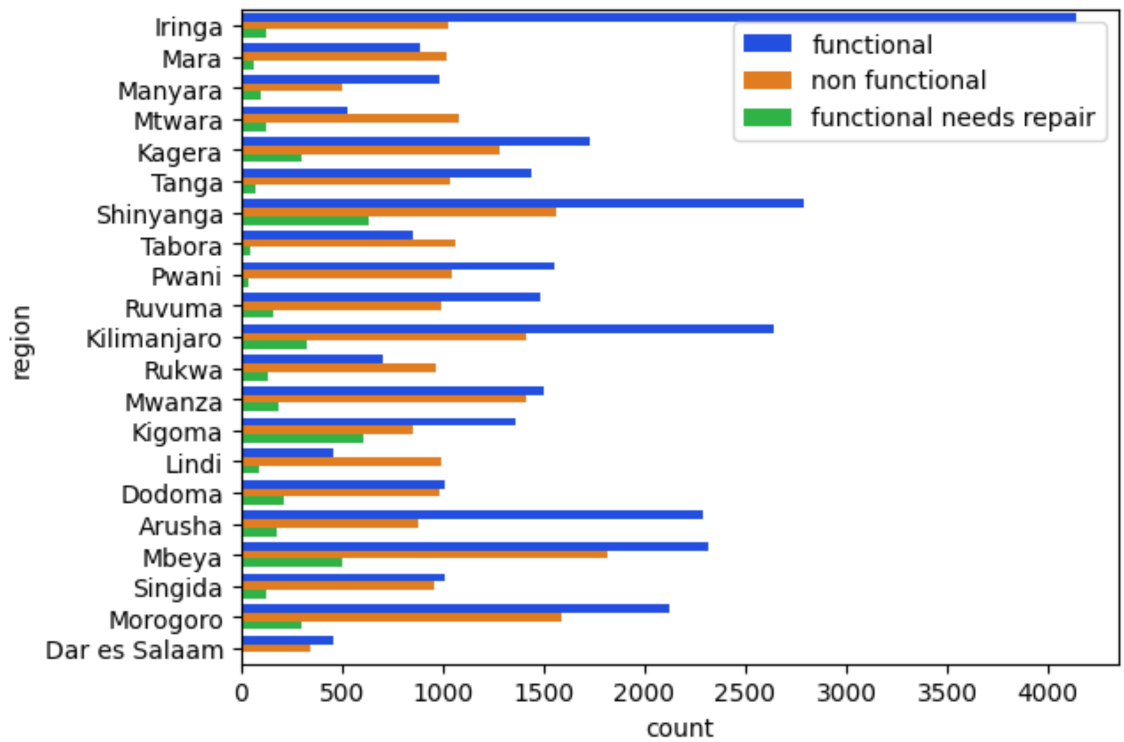
In [77]:
```
multivariate_ycountplot("basin", complete_data)
```



Lake Rufiji, Pangani basin and Lake Victoria leads as the basin bodies with the highest number pumps functioning around them and this can be an indication these are the main water sources in Tanzania. It also indicates high population around the regions of the basin

Relationship betweeen Status group and region

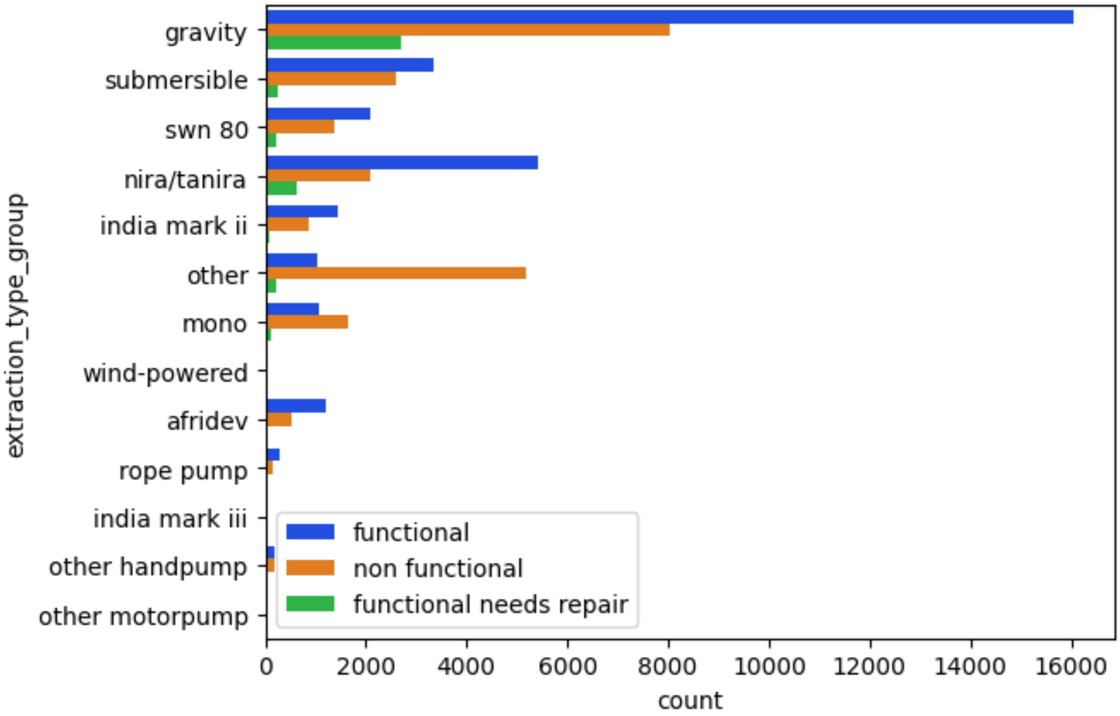
In [78]:
```
multivariate_ycountplot("region", complete_data)
```



Iringa region is the region with the most functioning pumps. Mbeya and Shinyanga regions are leading in pumps that are not functioning. Kigoma, Shinyanga and Kigoma also have highest number of pumps that need repair.

Relationship betweeen Status group and extraction_type_group

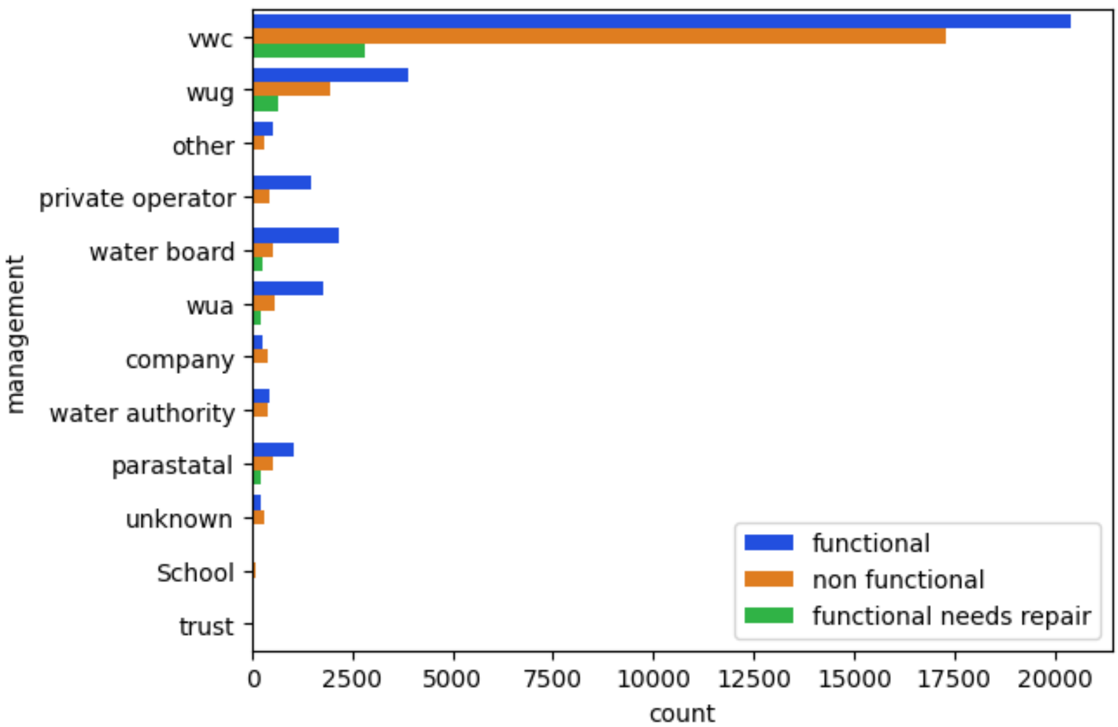
In [79]:
```
multivariate_ycountplot("extraction_type_group", complete_data)
```



Gravity is the main type of water extraction meaning many people fetch there water in streams and rivers. This indicates poor infrastructure of clean piped water in Tanzania.

Relationship betweeen Status group and management

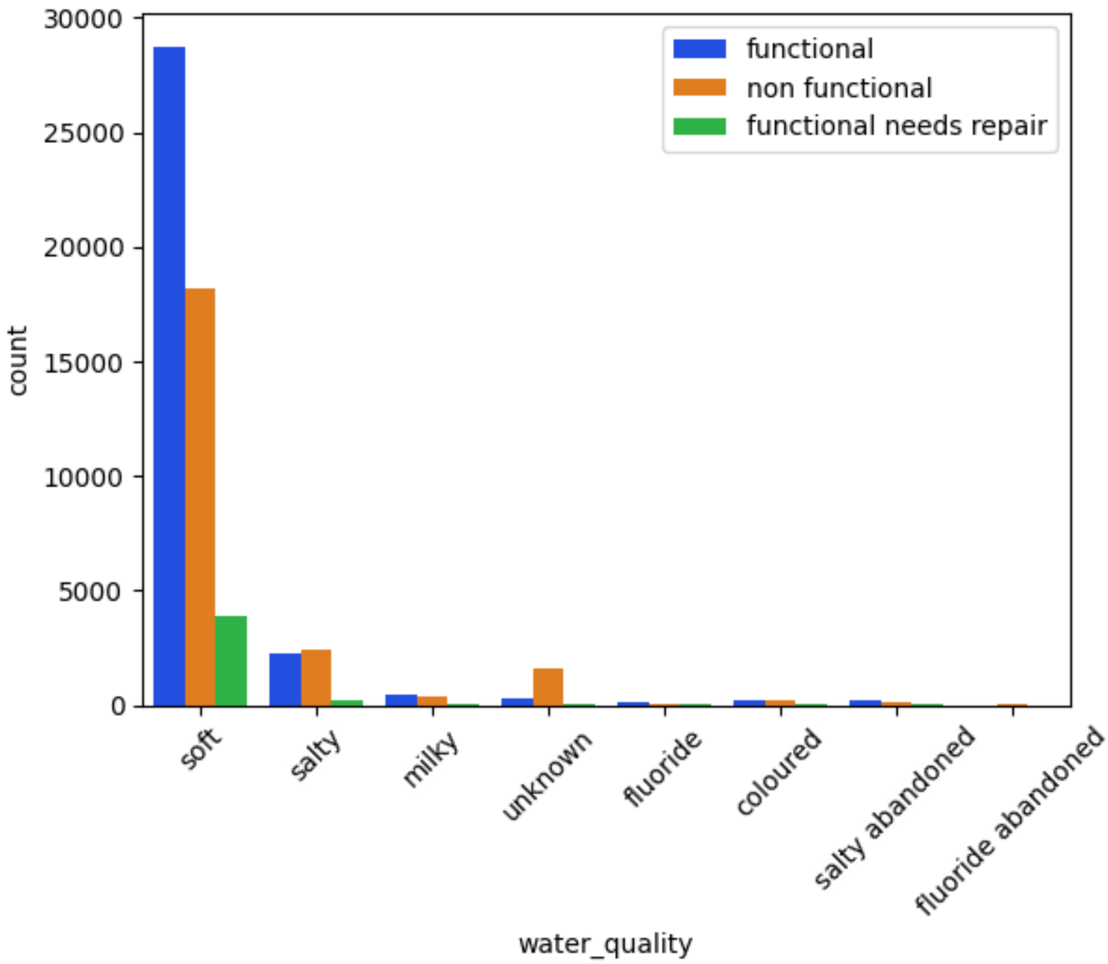
In [80]:
```
multivariate_ycountplot("management", complete_data)
```



Village Water Community managed wells have the highest number of functioning, non function and functioning but needs repair. This can indicate that some communities are best at managing their well's infrastructure and some of them don't know how to manage their water well

Relationship betweeen Status group and waterquality

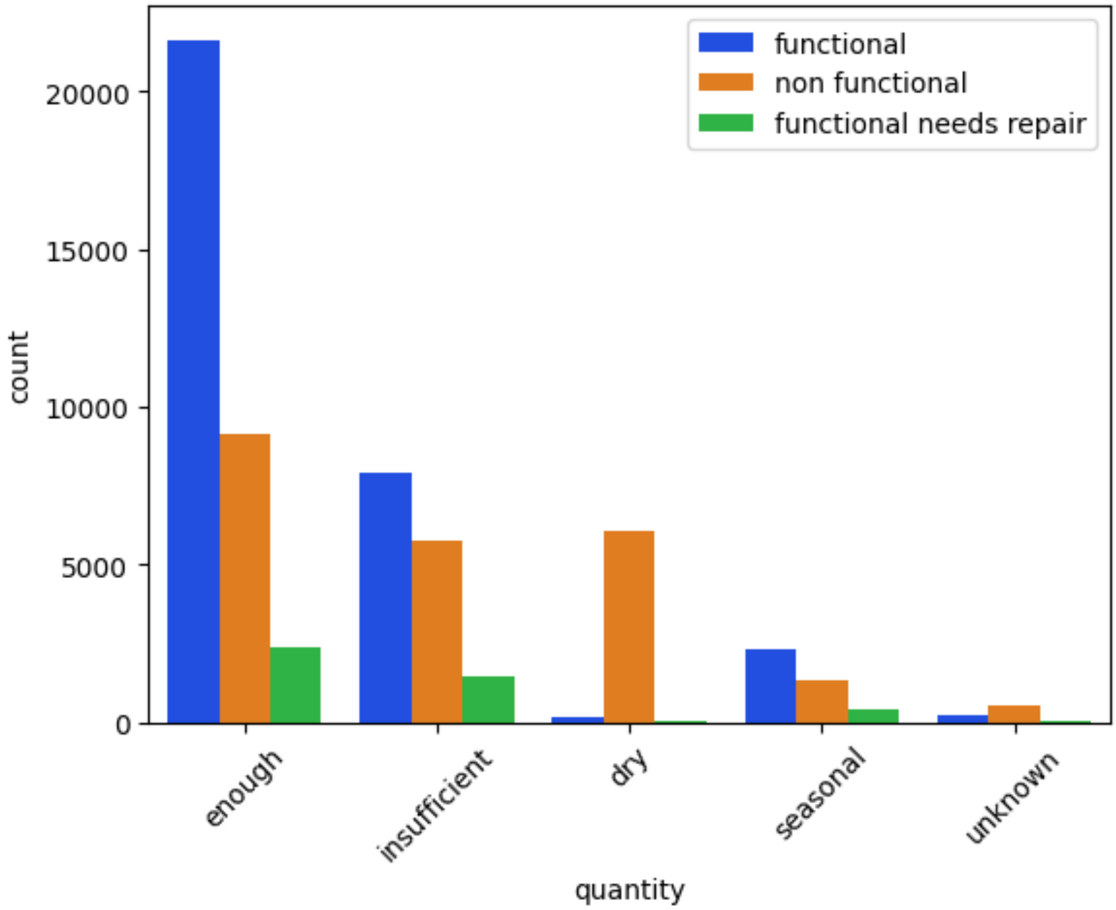
In [81]:
```
multivariate_xcountplot("water_quality", complete_data)
```



Soft water is the most used water in Tanzania as we can see it has the highest count of all three classes of the status group

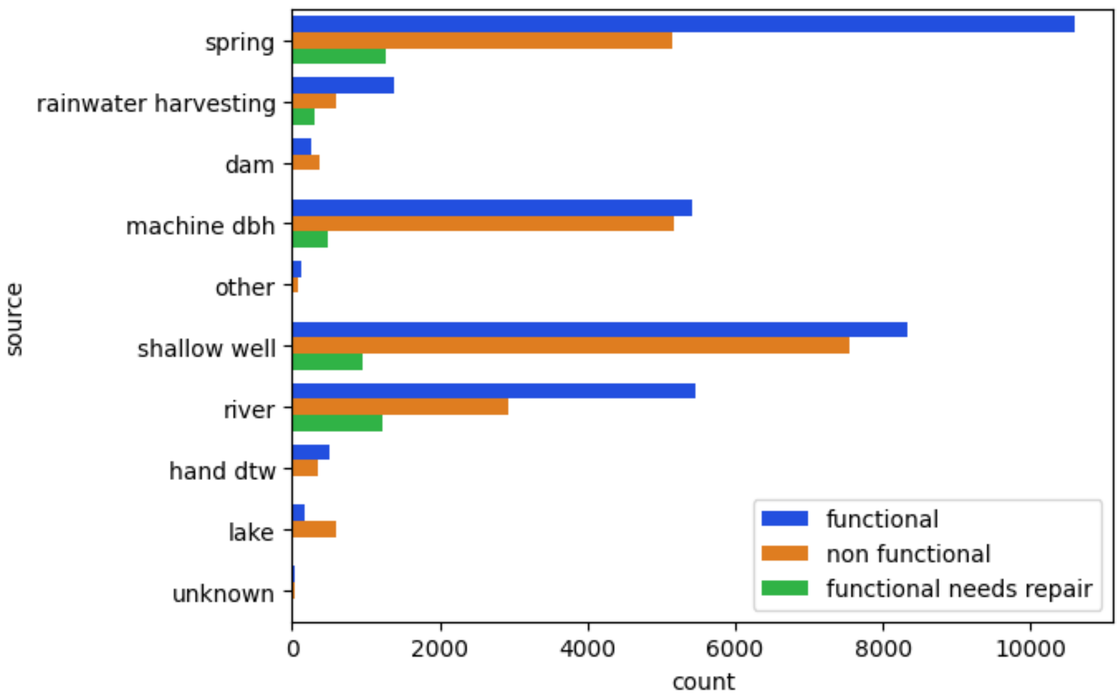
Relationship betweeen Status group and quantity

In [82]:
```
multivariate_xcountplot("quantity", complete_data)
```



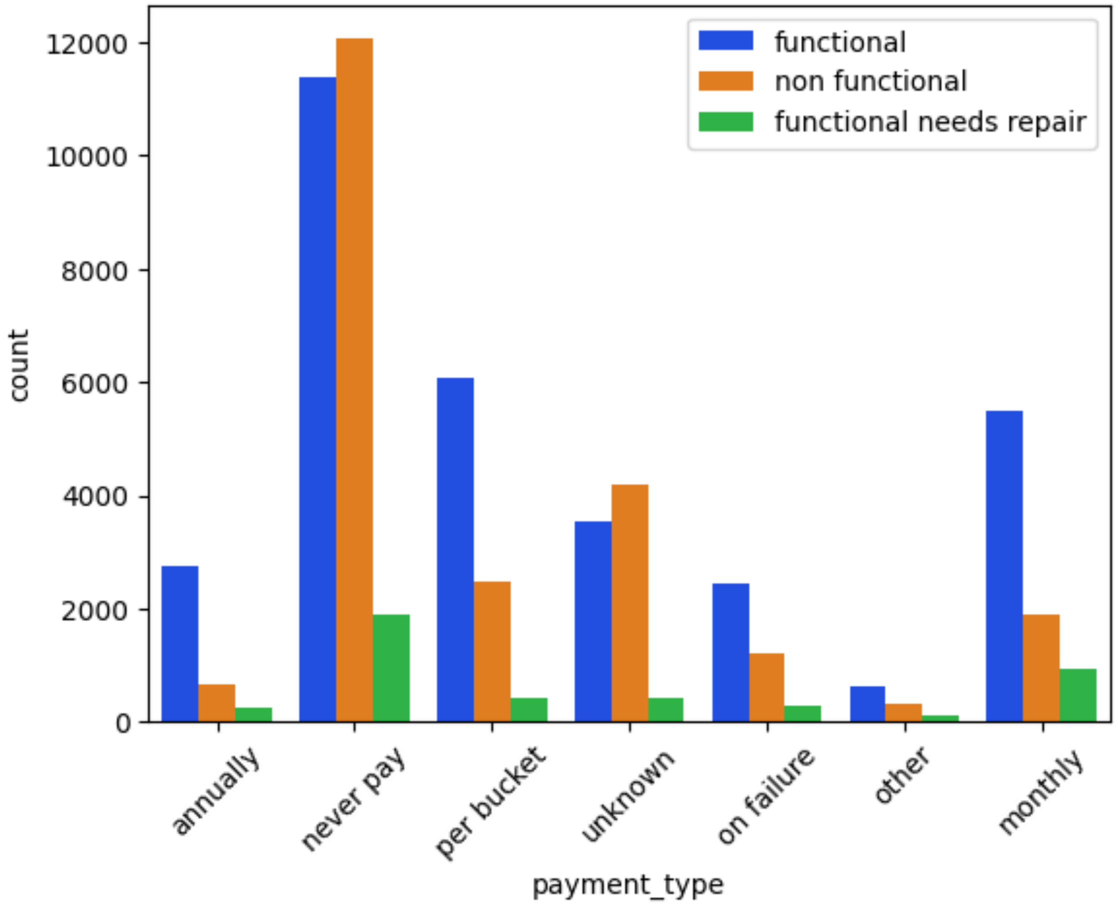
Relationship betweeen Status group and source

In [83]:
```
multivariate_ycountplot("source", complete_data)
```



Relationship betweeen Status group and payment_type

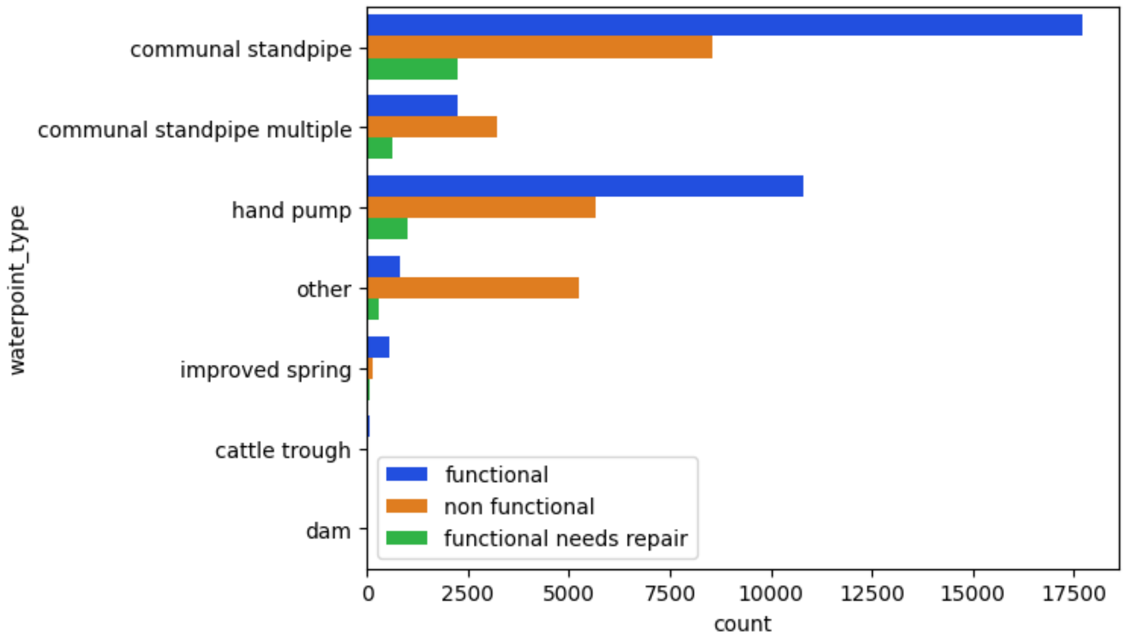
In [84]:
```
multivariate_xcountplot("payment_type", complete_data)
```



The neverpay water wells have the highest number of non functional pumps which can indicate lack of funds to repair the broken pumps in the region.

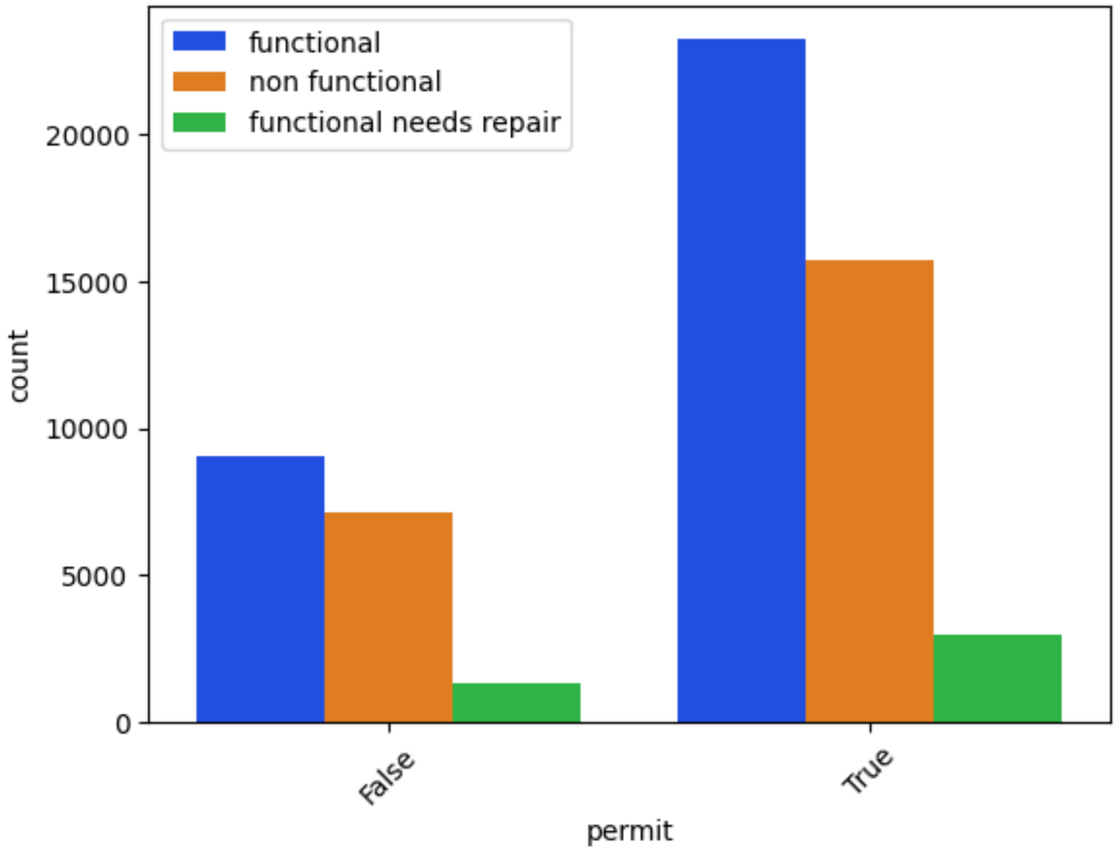
Relationship betweeen Status group and waterpoint_type

In [85]:
```
multivariate_ycountplot("waterpoint_type", complete_data)
```



Relationship betweeen Status group and permit

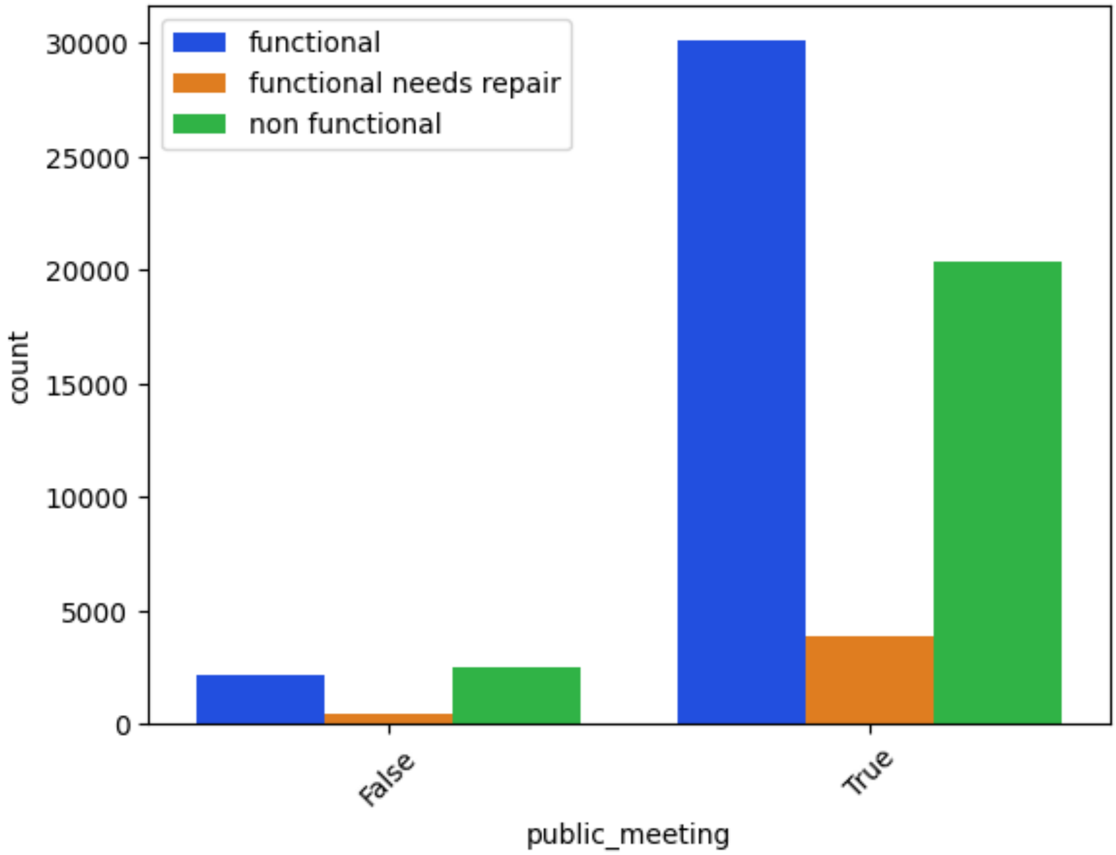
In [86]:
```
multivariate_xcountplot("permit", complete_data)
```



Water wells working with permits have the largest portion of functional pumps and non functional at the same time. Most wells in Tanzania operate under a working permit

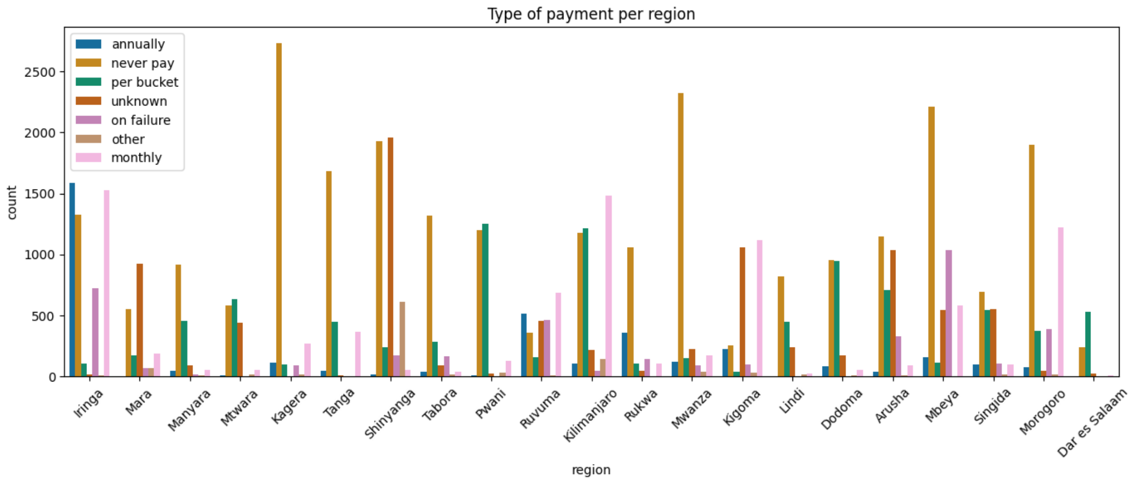
Relationship betweeen Status group and public_meeting

In [87]:
```
multivariate_xcountplot("public_meeting", complete_data)
```



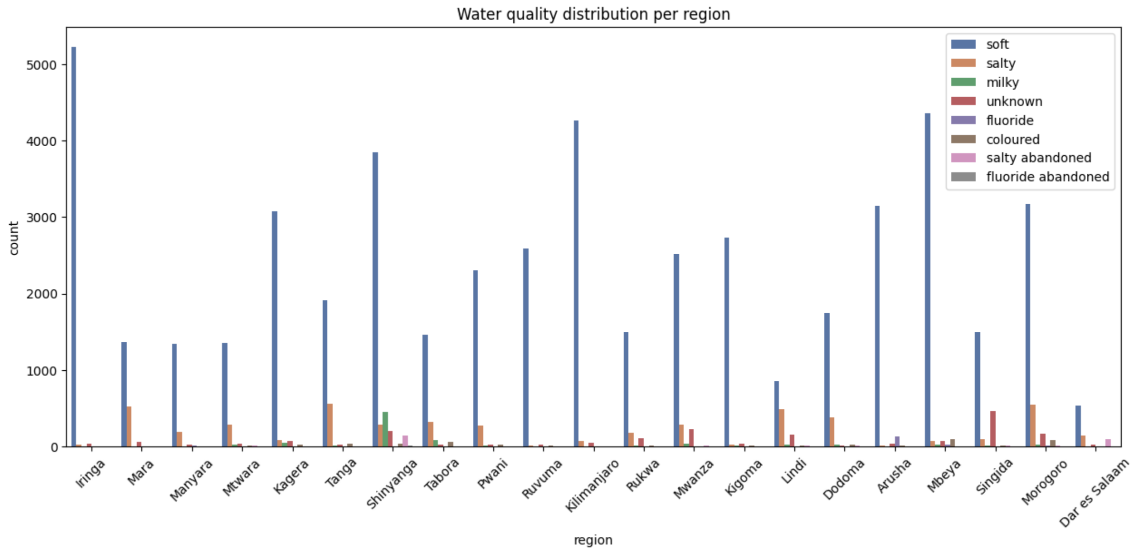
Knowing type of payment per region

In [88]:
```
plt.figure(figsize=(15, 5))
sns.countplot(x= "region", hue = "payment_type", data= complete_data
plt.xticks(rotation = 45);
plt.legend(loc= "best")
plt.title("Type of payment per region")
plt.show();
```



Kagera MWanza and Mbeya regions are the leading regions using the neverpay method. This tells us these regions are the leading when it comes to non functional pumps. There seems to be a collinearity between neverpay method and non functional pumps

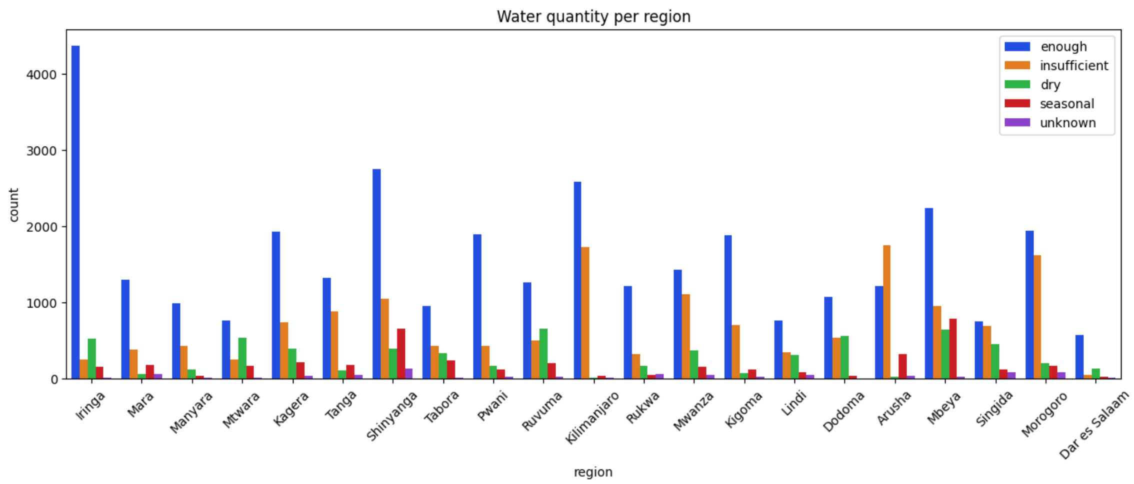
In [89]:
```
plt.figure(figsize=(15, 6))
sns.countplot(x= "region", hue = "water_quality", data= complete_data
plt.xticks(rotation = 45);
plt.legend(loc= "best")
plt.title("Water quality distribution per region")
plt.show();
```



Iringa, Tanga, Ruvuma and Arusha are the leading regions in terms of soft water quality. This a perfect exlanation why these regions have the highest number of water wells.

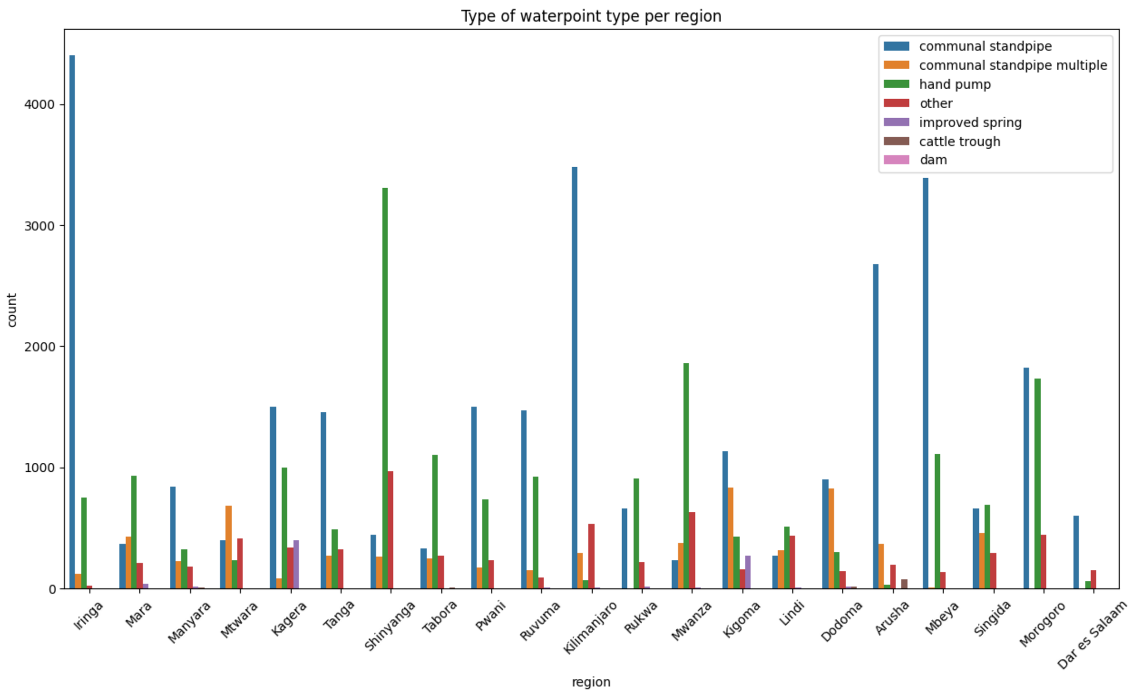
Relationship betweeen Region and water quantity

In [90]:
```
plt.figure(figsize=(15, 5))
sns.countplot(x= "region", hue = "quantity", data= complete_data, pa
plt.xticks(rotation = 45);
plt.legend(loc= "best")
plt.title("Water quantity per region")
plt.show();
```



Iringa, Shinyanga, Kilimanjaro and Mbeya are the regions with high count of sufficient water sources for its population. Arusha has the highest number of insufficient water wells followed by Kilimanjaro

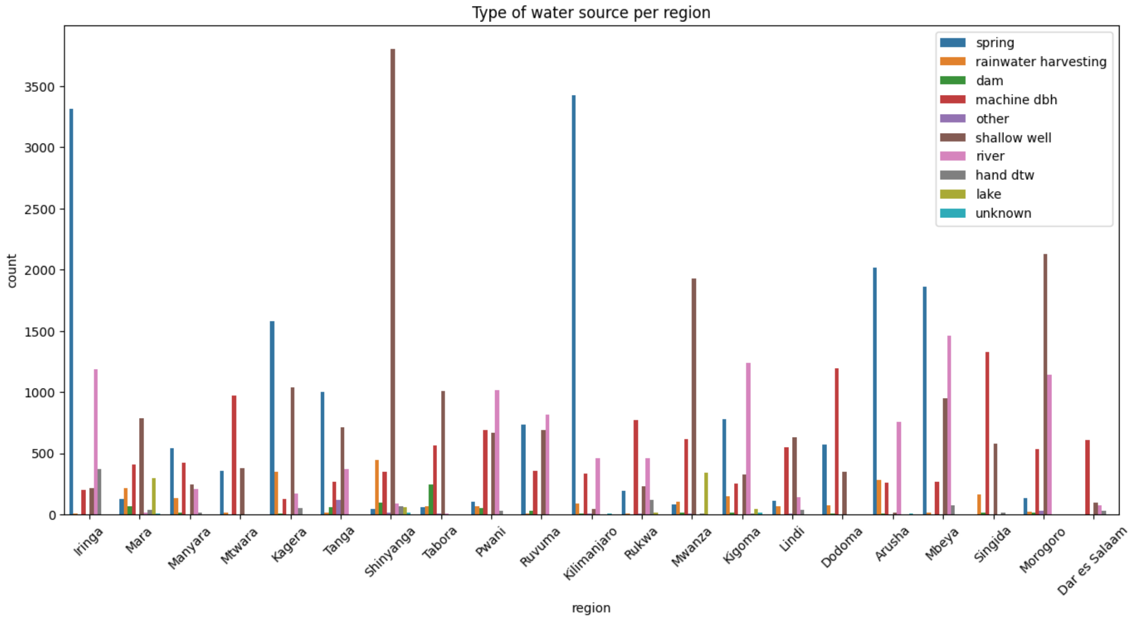
In [91]:
```
plt.figure(figsize=(15, 8))
sns.countplot(x= "region", hue = "waterpoint_type", data= complete_da
plt.xticks(rotation = 45);
plt.title("Type of waterpoint type per region")
plt.legend(loc= "best")
plt.show();
```



Communal standpipes are the most popular waterpoint type in Tanzania. Hand pump comes in second

Relationship betweeen Region and source of water

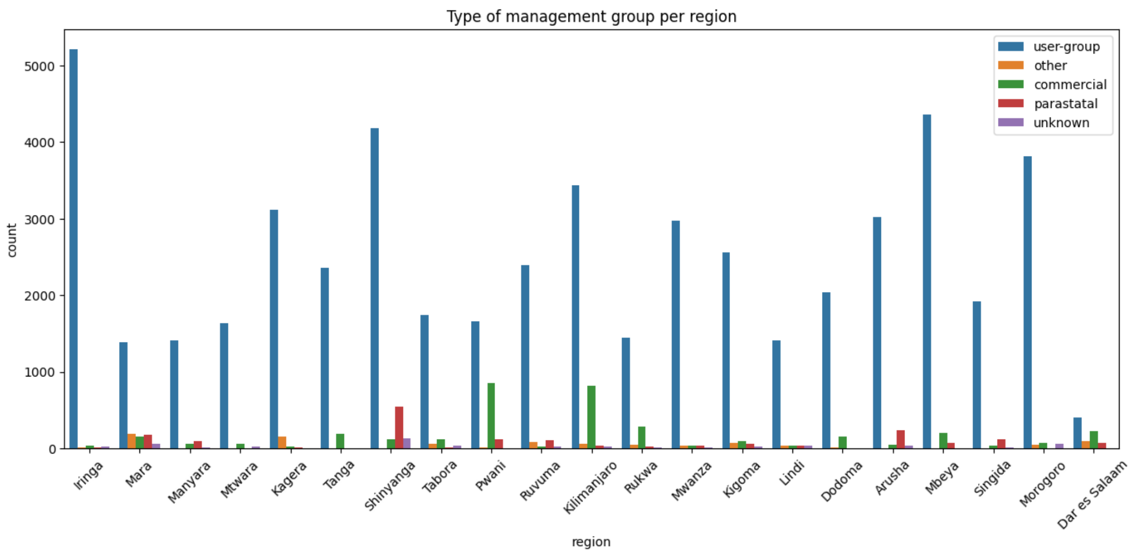
In [92]:
```
plt.figure(figsize=(15, 7))
sns.countplot(x= "region", hue = "source", data= complete_data)
plt.xticks(rotation = 45)
plt.title("Type of water source per region")
plt.legend(loc= "best")
plt.show();
```



Iringa and Ruvuma regions have springs as their main water sources. Shinyanga has shallow wells as its main source for water. Singida, Dodoma, Rukwa, Pwani, Mtwara have machine drilled borehole as the main source of water and this indicates these are commercial regions or towns with no natural water source or body near them.

Relationship betweeen Region and management group

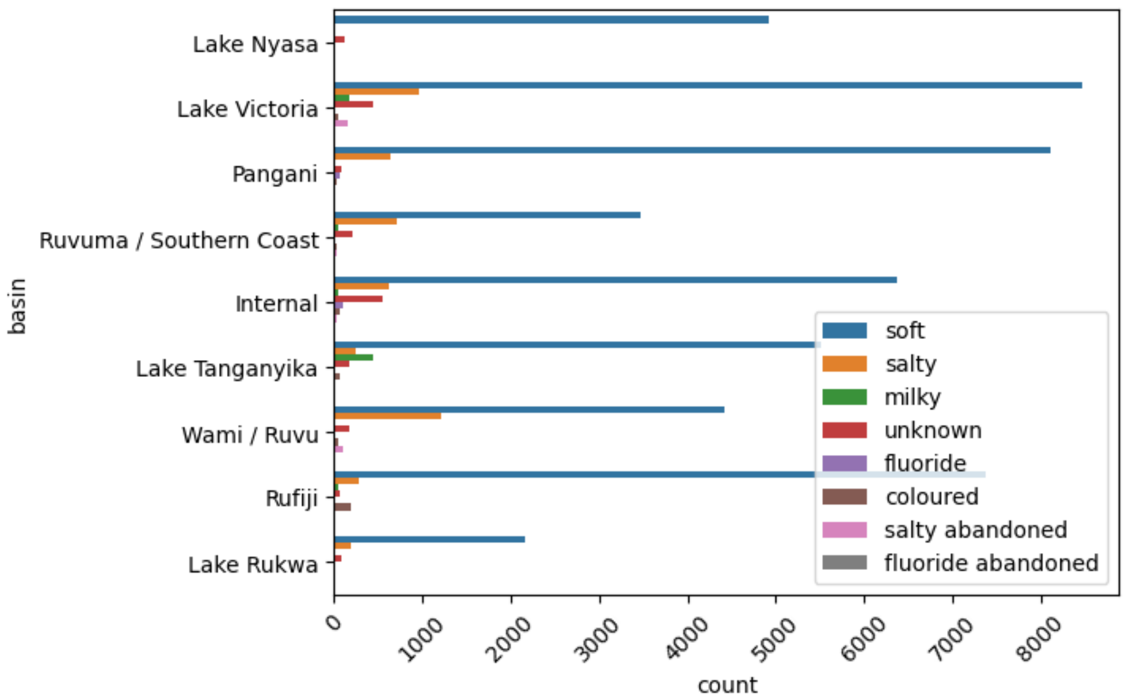
In [93]:
```
plt.figure(figsize=(15, 6))
sns.countplot(x= "region", hue = "management_group", data= complete_(
plt.xticks(rotation = 45)
plt.title("Type of management group per region")
plt.legend(loc= "best")
plt.show();
```



In all regions user-group is the main type of management group. This can be associated with the spirit of communism in Tanzania

Relationship between basin and water quality

In [94]:
```
sns.countplot(y = "basin", hue= "water_quality", data = complete_data
plt.legend(loc = "best")
plt.xticks(rotation = 45)
plt.show();
```



All water basins in Tanzania have soft water as the main component of its body. This can indicate economic activities like fishing is high in all the regions # We are now done with exploratory analysis and we have seen and understood our data well. Next stage is modelling and I will do that on another notebook to avoid confusion and break the monotony of reading a very long notebook. It also aids in organizing your work. I am going to inspect my data one last time before saving it as a csv file in order to capture the changes we have made and use this new saved data in my next notebook

In [95]:
```
#inspecting the data
complete_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59400 entries, 0 to 59399
Data columns (total 21 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   status_group           59400 non-null  object
 1   amount_tsh             59400 non-null  float64
 2   gps_height             59400 non-null  int64
 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   lga                    59400 non-null  object
 8   population             59400 non-null  int64
 9   extraction_type_group  59400 non-null  object
 10  management             59400 non-null  object
 11  management_group       59400 non-null  object
 12  payment_type           59400 non-null  object
 13  water_quality          59400 non-null  object
 14  quantity               59400 non-null  object
 15  source                 59400 non-null  object
 16  source_class           59400 non-null  object
 17  waterpoint_type        59400 non-null  object
 18  installer              59400 non-null  object
 19  permit                 59400 non-null  object
 20  public_meeting         59400 non-null  object
dtypes: float64(3), int64(2), object(16)
memory usage: 9.5+ MB
```

In [96]:
```
# saving our processed data as a csv file
complete_data.to_csv("modelling_data.csv", index= False)
```