```
import findspark
findspark.init()

from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession, functions as F

from pyspark.sql.functions import split, udf, col, regexp_replace,
concat_ws, regexp_extract
from pyspark.sql.types import StringType, IntegerType, DateType,
TimestampType
from datetime import datetime
from pyspark.sql.functions import udf, to_timestamp
import re

spark = (SparkSession\
    .builder \
    .appName('Logs') \
    .getOrCreate())

sc = spark.sparkContext

base_df = spark.read.text("Z:\Datasets\Linux_2k.log")
base_df.printSchema()
```

```
root
 |-- value: string (nullable = true)
```

```
base_df.show(5, truncate=False)
```

```
+-----------------------------------------------------------------------
-----------------------------------------------------------------------
----------------------+
|value
|
+-----------------------------------------------------------------------
-----------------------------------------------------------------------
----------------------+
|Jun 14 15:16:01 combo sshd(pam_unix)[19939]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
|
|Jun 14 15:16:02 combo sshd(pam_unix)[19937]: check pass; user unknown
|
|Jun 14 15:16:02 combo sshd(pam_unix)[19937]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
|
|Jun 15 02:04:59 combo sshd(pam_unix)[20882]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-
ip.hinet.net  user=root|
|Jun 15 02:04:59 combo sshd(pam_unix)[20884]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-
```

```
ip.hinet.net  user=root|
+------------------------------------------------------------------------
------------------------------------------------------------------------
---------------------+
only showing top 5 rows
```

```
type(base_df)
```

```
pyspark.sql.dataframe.DataFrame
```

```
print((base_df.count(), len(base_df.columns)))
```

```
(2000, 1)
```

```
sample_logs = [item['value'] for item in base_df.take(10)]
```

```
sample_logs
```

```
['Jun 14 15:16:01 combo sshd(pam_unix)[19939]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4 ',
 'Jun 14 15:16:02 combo sshd(pam_unix)[19937]: check pass; user
unknown',
 'Jun 14 15:16:02 combo sshd(pam_unix)[19937]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4 ',
 'Jun 15 02:04:59 combo sshd(pam_unix)[20882]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-
ip.hinet.net  user=root',
 'Jun 15 02:04:59 combo sshd(pam_unix)[20884]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-
ip.hinet.net  user=root',
 'Jun 15 02:04:59 combo sshd(pam_unix)[20883]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-
ip.hinet.net  user=root',
 'Jun 15 02:04:59 combo sshd(pam_unix)[20885]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-
ip.hinet.net  user=root',
 'Jun 15 02:04:59 combo sshd(pam_unix)[20886]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-
ip.hinet.net  user=root',
 'Jun 15 02:04:59 combo sshd(pam_unix)[20892]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-
ip.hinet.net  user=root',
 'Jun 15 02:04:59 combo sshd(pam_unix)[20893]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-
ip.hinet.net  user=root']
```

```
#to extract timestamp
```

```
time_pattern = r"(.*?)\b\d{2}:\d{2}:\d{2}\b"
```

```
timestamp = [re.search(time_pattern, item).group(1)
             if re.search(time_pattern, item)
             else 'no match'
              for item in sample_logs ]

def extract_timestamp(entry):
    time_pattern = r'(\b\d{2}:\d{2}:\d{2}\b)'
    match = re.search(time_pattern, entry)
    return match.group(1) if match else 'no match'

#create a UDF
extract_timestamp_UDF = udf(extract_timestamp, StringType())

base_df = base_df.withColumn("date time",
extract_timestamp_UDF('value'))

base_df.show(5 ,truncate=False)
```

```
+-----------------------------------------------------------------------
-----------------------------------------------------------------------
----------------------+---------+
|value
|date time|
+-----------------------------------------------------------------------
-----------------------------------------------------------------------
----------------------+---------+
|Jun 14 15:16:01 combo sshd(pam_unix)[19939]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
|15:16:01 |
|Jun 14 15:16:02 combo sshd(pam_unix)[19937]: check pass; user unknown
|15:16:02 |
|Jun 14 15:16:02 combo sshd(pam_unix)[19937]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
|15:16:02 |
|Jun 15 02:04:59 combo sshd(pam_unix)[20882]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-
ip.hinet.net  user=root|02:04:59 |
|Jun 15 02:04:59 combo sshd(pam_unix)[20884]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-
ip.hinet.net  user=root|02:04:59 |
+-----------------------------------------------------------------------
-----------------------------------------------------------------------
----------------------+---------+
only showing top 5 rows
```

```
base_df = base_df.drop('date time')

base_df.printSchema()
```

```
root
 |-- value: string (nullable = true)
```

```
# This for taking the date also
def extract_date(sub_string):
    return sub_string.split('combo')[0]
def date_time(string, word="combo"):
    pattern = rf'^(.*?)(?=\b{re.escape(word)})'

    result = re.search(pattern, string)

    if result:
        extracted_text = result.group(1)
        return extracted_text
    else:
        return None

extract_date_UDF = udf(extract_date, StringType())


base_df = base_df.withColumn('Date time', extract_date_UDF('value'))

base_df = base_df.drop('Message')

base_df.show(5)
```

```
+--------------------+----------------+
|               value|       Date time|
+--------------------+----------------+
|Jun 14 15:16:01 c...|Jun 14 15:16:01 |
|Jun 14 15:16:02 c...|Jun 14 15:16:02 |
|Jun 14 15:16:02 c...|Jun 14 15:16:02 |
|Jun 15 02:04:59 c...|Jun 15 02:04:59 |
|Jun 15 02:04:59 c...|Jun 15 02:04:59 |
+--------------------+----------------+
only showing top 5 rows
```

```
from pyspark.sql.functions import udf, to_timestamp
from pyspark.sql.types import ArrayType, StringType
import re

# Define the error_extract function
def error_extract(sub_string):
    pattern = r'(?<=]:)(.*?)(?=;)'
    matches = re.findall(pattern, sub_string)
    return matches

# Register the error_extract function as a UDF
```

```
extract_error_UDF = udf(error_extract, ArrayType(StringType()))

# Apply the UDF to the DataFrame column
base_df = base_df.withColumn('Message', extract_error_UDF('value'))

# Show the DataFrame
base_df.show(5, truncate=False)
```

```
+-----------------------------------------------------------------------
-----------------------------------------------------------------------
----------------------+----------------+-------------------------+
|value
|Date time       |Message                    |
+-----------------------------------------------------------------------
-----------------------------------------------------------------------
----------------------+----------------+-------------------------+
|Jun 14 15:16:01 combo sshd(pam_unix)[19939]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
|Jun 14 15:16:01 |[ authentication failure]|
|Jun 14 15:16:02 combo sshd(pam_unix)[19937]: check pass; user unknown
|Jun 14 15:16:02 |[ check pass]              |
|Jun 14 15:16:02 combo sshd(pam_unix)[19937]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
|Jun 14 15:16:02 |[ authentication failure]|
|Jun 15 02:04:59 combo sshd(pam_unix)[20882]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-
ip.hinet.net  user=root|Jun 15 02:04:59 |[ authentication failure]|
|Jun 15 02:04:59 combo sshd(pam_unix)[20884]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-
ip.hinet.net  user=root|Jun 15 02:04:59 |[ authentication failure]|
+-----------------------------------------------------------------------
-----------------------------------------------------------------------
----------------------+----------------+-------------------------+
only showing top 5 rows
```

```
pattern = r'(?<=;)(.*)'

# Extract everything after the first occurrence of ;


def error_user(sub_string):
    pattern = r'(?<=;)(.*)'
    matches = re.findall(pattern, sub_string)
    return matches

# Register the error_extract function as a UDF
extract_user_UDF = udf(error_user, ArrayType(StringType()))

base_df = base_df.withColumn('User Info', extract_user_UDF('value'))
```

```
base_df.show(5, truncate=True)
```

```
+--------------------+----------------+--------------------
+--------------------+
|               value|       Date time|             Message|
User Info|
+--------------------+----------------+--------------------
+--------------------+
|Jun 14 15:16:01 c...|Jun 14 15:16:01 |[ authentication ...|[ logname=
uid=0 ...|
|Jun 14 15:16:02 c...|Jun 14 15:16:02 |       [ check pass]|
[ user unknown]|
|Jun 14 15:16:02 c...|Jun 14 15:16:02 |[ authentication ...|[ logname=
uid=0 ...|
|Jun 15 02:04:59 c...|Jun 15 02:04:59 |[ authentication ...|[ logname=
uid=0 ...|
|Jun 15 02:04:59 c...|Jun 15 02:04:59 |[ authentication ...|[ logname=
uid=0 ...|
+--------------------+----------------+--------------------
+--------------------+
only showing top 5 rows
```

```
#check the schema
base_df.printSchema()
```

```
root
 |-- value: string (nullable = true)
 |-- Date time: string (nullable = true)
 |-- Message: array (nullable = true)
 |    |-- element: string (containsNull = true)
 |-- User Info: array (nullable = true)
 |    |-- element: string (containsNull = true)
```

```
base_df
```

```
DataFrame[value: string, Date time: string, Message: array<string>,
User Info: array<string>]
```

```
df = base_df.withColumn('timestamp', to_timestamp("Date time"))
```

```
#there were unwanted '[]' in our data this code is for removing them
base_df = base_df.withColumn('Message', concat_ws('','Message'))
```

```
base_df = base_df.withColumn('Message', regexp_replace('Message', '\
[|\]', ''))
```

```
base_df = base_df.withColumn('User Info', concat_ws('','User Info'))
```

```
base_df = base_df.withColumn('User Info', regexp_replace('User Info',
'\[|\]', ''))

base_df.show(3, truncate=False)
```

```
+-------------------------------------------------------------------------
-------------------------------------------------------------
+----------------+----------------------
+-----------------------------------------------------------------+
|value
|Date time       |Message                 |User Info
|
+-------------------------------------------------------------------------
-------------------------------------------------------------
+----------------+----------------------
+-----------------------------------------------------------------+
|Jun 14 15:16:01 combo sshd(pam_unix)[19939]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4 |Jun 14
15:16:01 | authentication failure| logname= uid=0 euid=0 tty=NODEVssh
ruser= rhost=218.188.2.4 |
|Jun 14 15:16:02 combo sshd(pam_unix)[19937]: check pass; user unknown
|Jun 14 15:16:02 | check pass            | user unknown
|
|Jun 14 15:16:02 combo sshd(pam_unix)[19937]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4 |Jun 14
15:16:02 | authentication failure| logname= uid=0 euid=0 tty=NODEVssh
ruser= rhost=218.188.2.4 |
+-------------------------------------------------------------------------
-------------------------------------------------------------
+----------------+----------------------
+-----------------------------------------------------------------+
only showing top 3 rows
```

```
base_df = base_df.withColumn('Date', regexp_extract('Date time', '^\
w{3} \d{2}', 0))

#regular expression to get the time
base_df = base_df.withColumn('Time', regexp_extract('Date time', '\
d{2}:\d{2}:\d{2}', 0))
base_df.show(3, truncate=False)
```

```
+-------------------------------------------------------------------------
-------------------------------------------------------------
+----------------+----------------------
+-----------------------------------------------------------------+------
+--------+
|value
|Date time       |Message                 |User Info
|Date  |Time    |
```

```
+-------------------------------------------------------------------
-------------------------------------------------------------
+----------------+----------------------
+-------------------------------------------------------------+------
+--------+
|Jun 14 15:16:01 combo sshd(pam_unix)[19939]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4 |Jun 14
15:16:01 | authentication failure| logname= uid=0 euid=0 tty=NODEVssh
ruser= rhost=218.188.2.4 |Jun 14|15:16:01|
|Jun 14 15:16:02 combo sshd(pam_unix)[19937]: check pass; user unknown
|Jun 14 15:16:02 | check pass            | user unknown
|Jun 14|15:16:02|
|Jun 14 15:16:02 combo sshd(pam_unix)[19937]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4 |Jun 14
15:16:02 | authentication failure| logname= uid=0 euid=0 tty=NODEVssh
ruser= rhost=218.188.2.4 |Jun 14|15:16:02|
+-------------------------------------------------------------------
-------------------------------------------------------------
+----------------+----------------------
+-------------------------------------------------------------+------
+--------+
only showing top 3 rows
```

----------Pandas Below ----------

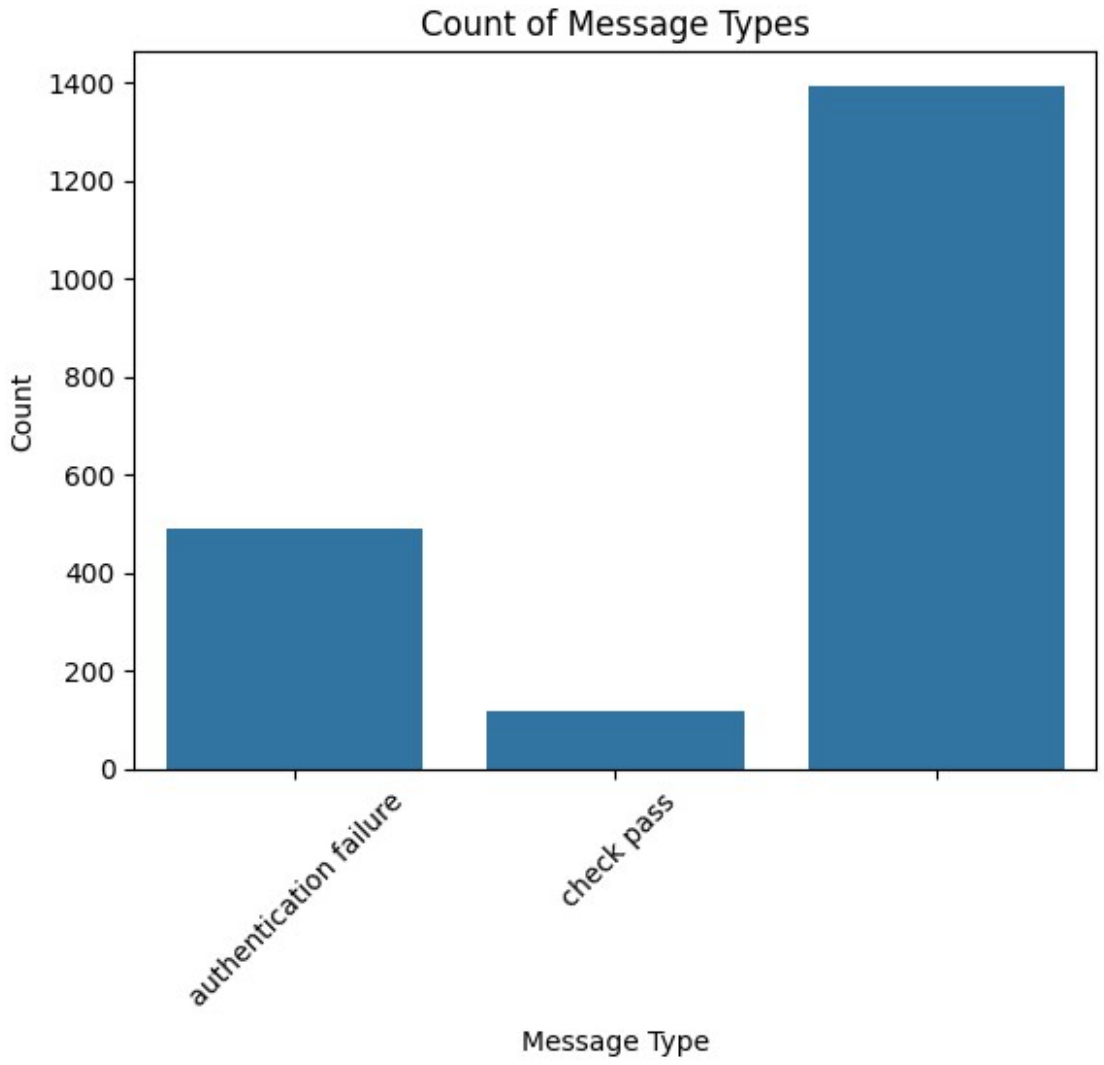
```
result_pdf = base_df.select("*").toPandas()

result_pdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   value      2000 non-null   object
 1   Date time  2000 non-null   object
 2   Message    2000 non-null   object
 3   User Info  2000 non-null   object
 4   Date       2000 non-null   object
dtypes: object(5)
memory usage: 78.3+ KB
```

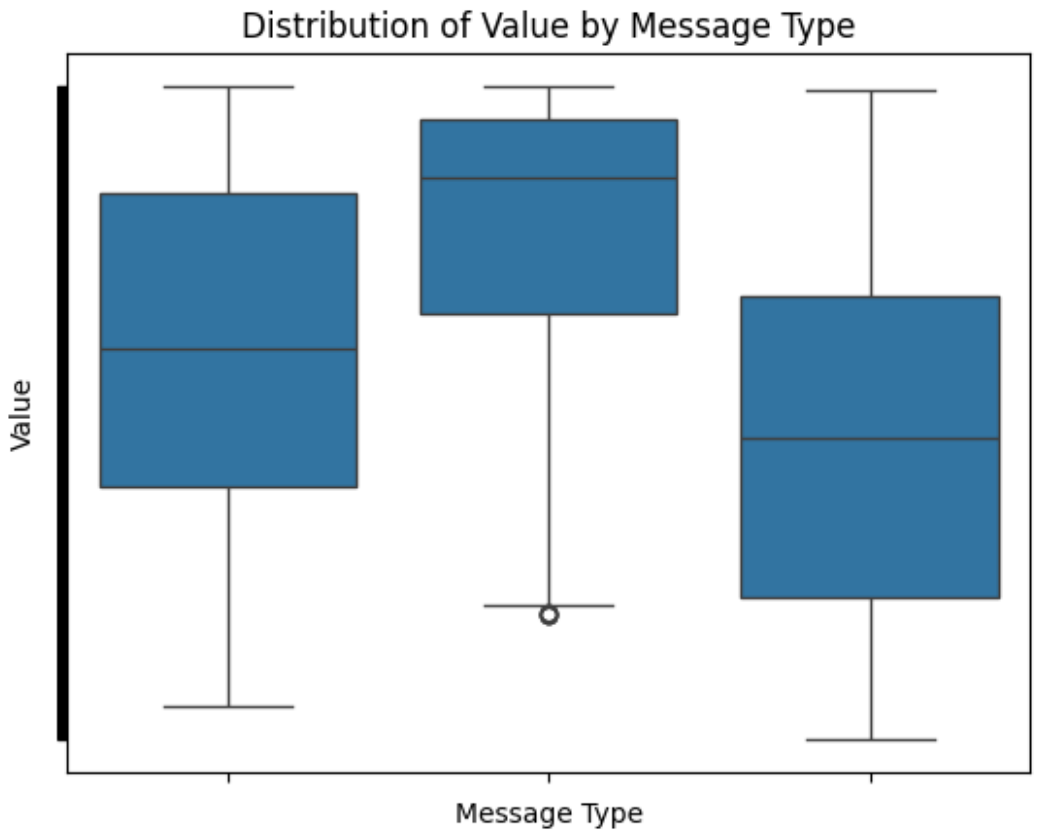
Below are some visualizations for getting an insight of our data,

As this data is lacking a decent number of values, I have not decided to go with a null value approach to handle null enteries

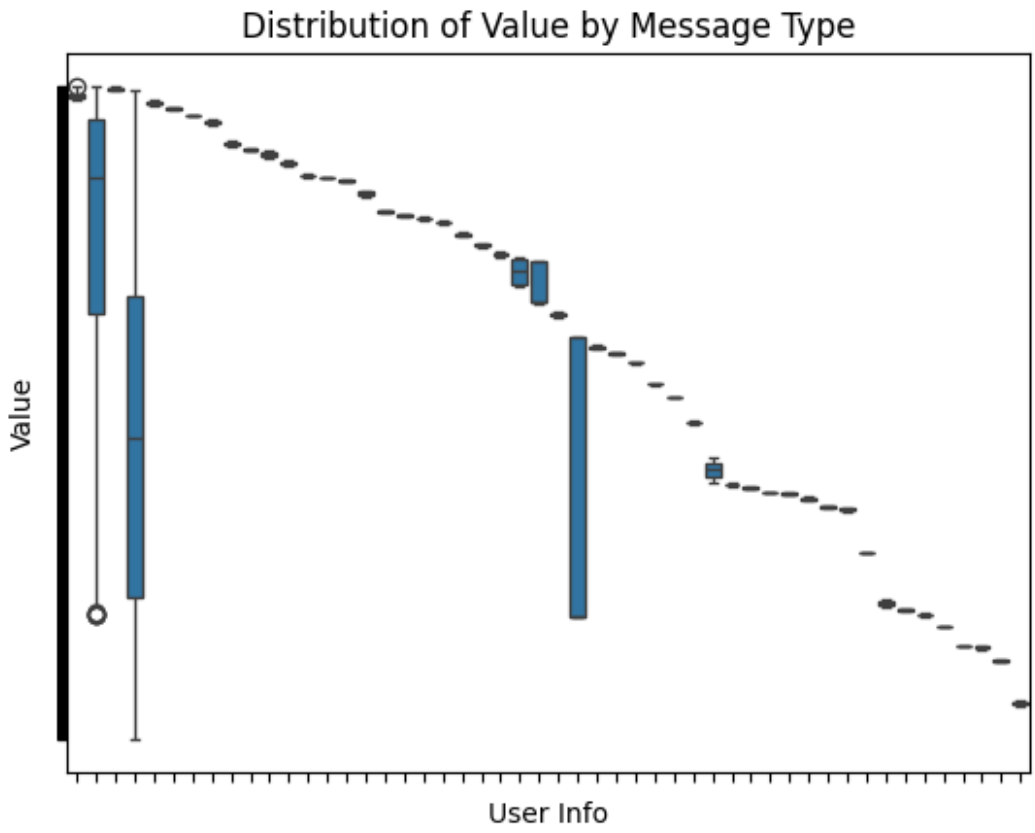
```
sns.countplot(data=result_pdf, x='Message')
plt.xlabel('Message Type')
plt.ylabel('Count')
plt.title('Count of Message Types')
plt.xticks(rotation=45)
plt.show()
```




```
# Plot a boxplot of 'value' grouped by 'Message'
sns.boxplot(data=result_pdf, x='Message', y='value')
plt.xlabel('Message Type')
plt.ylabel('Value')
plt.title('Distribution of Value by Message Type')
plt.xticks(rotation=45)
plt.gca().set_yticklabels([])
plt.gca().set_xticklabels([])
plt.show()
```




```
# Plot a boxplot of 'value' grouped by 'Message'
sns.boxplot(data=result_pdf, x='User Info', y='value')
plt.xlabel('User Info')
plt.ylabel('Value')
plt.title('Distribution of Value by Message Type')
plt.xticks(rotation=45)
plt.gca().set_yticklabels([])
plt.gca().set_xticklabels([])
plt.show()
```




```
result_pdf.head()
```

```
                                                  value          Date time
\
0  Jun 14 15:16:01 combo sshd(pam_unix)[19939]: a...  Jun 14 15:16:01

1  Jun 14 15:16:02 combo sshd(pam_unix)[19937]: c...  Jun 14 15:16:02

2  Jun 14 15:16:02 combo sshd(pam_unix)[19937]: a...  Jun 14 15:16:02

3  Jun 15 02:04:59 combo sshd(pam_unix)[20882]: a...  Jun 15 02:04:59

4  Jun 15 02:04:59 combo sshd(pam_unix)[20884]: a...  Jun 15 02:04:59


                     Message
User Info  \
0   authentication failure   logname= uid=0 euid=0 tty=NODEVssh ruser=
rho...
1               check pass                                          user
unknown
2   authentication failure   logname= uid=0 euid=0 tty=NODEVssh ruser=
rho...
3   authentication failure   logname= uid=0 euid=0 tty=NODEVssh ruser=
```

```
rho...
4   authentication failure   logname= uid=0 euid=0 tty=NODEVssh ruser=
rho...
```

```
     Date
0  Jun 14
1  Jun 14
2  Jun 14
3  Jun 15
4  Jun 15
```

```
result_pdf['Message'].unique()
```

```
array([' authentication failure', ' check pass', ''], dtype=object)
```

```
result_pdf['Date'].unique()
```

```
array(['Jun 14', 'Jun 15', 'Jun 16', 'Jun 17', 'Jun 18', 'Jun 19',
       'Jun 20', 'Jun 21', 'Jun 22', 'Jun 23', 'Jun 24', 'Jun 25',
       'Jun 26', 'Jun 27', 'Jun 28', 'Jun 29', 'Jun 30', '', 'Jul 10',
       'Jul 11', 'Jul 12', 'Jul 13', 'Jul 14', 'Jul 15', 'Jul 16',
       'Jul 17', 'Jul 18', 'Jul 19', 'Jul 20', 'Jul 21', 'Jul 22',
       'Jul 23', 'Jul 24', 'Jul 25', 'Jul 26', 'Jul 27'],
dtype=object)
```