TASK REPORT

1. Description of the solution step by step

At the beginning I imported all the necessary libraries that will support the process of data processing and visualization Then I implemented a function (load_json_data) to load the contents of files from a specific location. I used the read_text and map method from the dask bag library to load data in the appropriate format from many json files. Then I converted it to dask dataframe using the to_dataframe method. I decided on the dask bag because the data is loaded faster than using the read_json method in dask. It does this in parallel and in small memory. Depending on the name of the file, its contents are cumulated (added) to the appropriate data frame. I also chose the most important attributes to reduce the data set.

Dask *is a flexible library for parallel computing in Python.* Dask is composed of parallel arrays, data frames, and lists that extend common interfaces like *NumPy, Pandas, or Python iterators* to larger-than-memory or distributed environments. These parallel collections run on top of dynamic task schedulers.

There are two types of files in json format in the source location, one applies to logs from Egnyte Cloud Server and the other from Egnyte Connect domain. For the second set of data I used the read_json method from the Dask library to load data. I used a mechanism which detects all files related to logs from Connect domain from the source and loads them into one dataframe.

In the next step, I created a function (basic_statistics) that is used for both datasets. It returns basic statistics on the data set. These are information such as the number of rows and columns and the list of columns with types. To perform these operations, I used the shape and dtypes methods on the data frame from the dask library.

In the third step I have created several different functions for both datasets separately. Each of them implements a different advanced statistic and displays the corresponding graph. In the implementation, I used the methods from dask and the matplotlib library to create simple graphs showing the result of the statistics. I also used a mechanism to check whether the input data (dataframe) has the required columns. In the 'main' function, I did the error and exception handling.

This is only a substitute for what could be done with the data. I focus only on basic statistics.

2. Test cases to check the quality of the data

High-quality data are the precondition for analyzing and using big data and for guaranteeing the value of the data. There are a variety of features that can be explored.

Completeness refers to whether there are any gaps in the data from what was expected to be collected, and what was actually collected. Completeness of key attributes, as well as their values is very important.

In our case, it would be good to check the completeness of the data in the columns with the time stamp, action and information about the working group along with the location

(country). These attributes are required to perform an analysis of the most frequently performed actions, time of day, when it is the most traffic, and to check which clients are the most active. When this data is missing, correct data analysis is not possible due to missing or incomplete key attributes. Data completeness check should be performed while loading data.

Correctness – checks whether the data matches the set rules. All data may be validated against the set of well-defined valid values. This can be used to detect new or incompatible values. Data values are standardized according to a data model and/or data type. All characteristics of the data must be correct – including business rules, relations, dates, definitions, etc.

In the data sets provided, the data validity could be checked for dates (timestamp, subscription_date), or they have an appropriate, standardized format. In addition, you could check if the values in the space_used column are given in the same unit and if the value has the right type.

Uniqueness – means that data denoting the same should be unique. There should not be duplicates in the data, as this may lead to incorrect analysis results. Each data record should be unique, otherwise the risk of accessing outdated information increases.

To ensure the uniqueness of data in our data sets, we could check the uniqueness of the event identifier (event_id). In addition, we could check if there are different terms for the same thing in the country column (long name / abbreviated name), action or plan name.

3. Analysis and visualization of statistics

2.1. Data (logs) from Egnyte Cloud Server

- Basic statistics

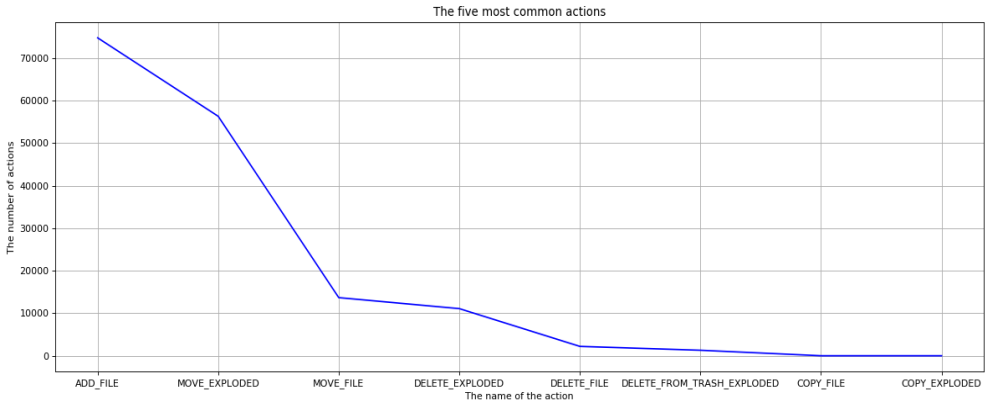
Dataframe column names:

```
Index(['action', 'app_inferred', 'os_inferred', 'country', 'datacenter',
       'event_id', 'automated_action', 'space_used', 'timestamp',
       'workgroup_id'],
      dtype='object')
```

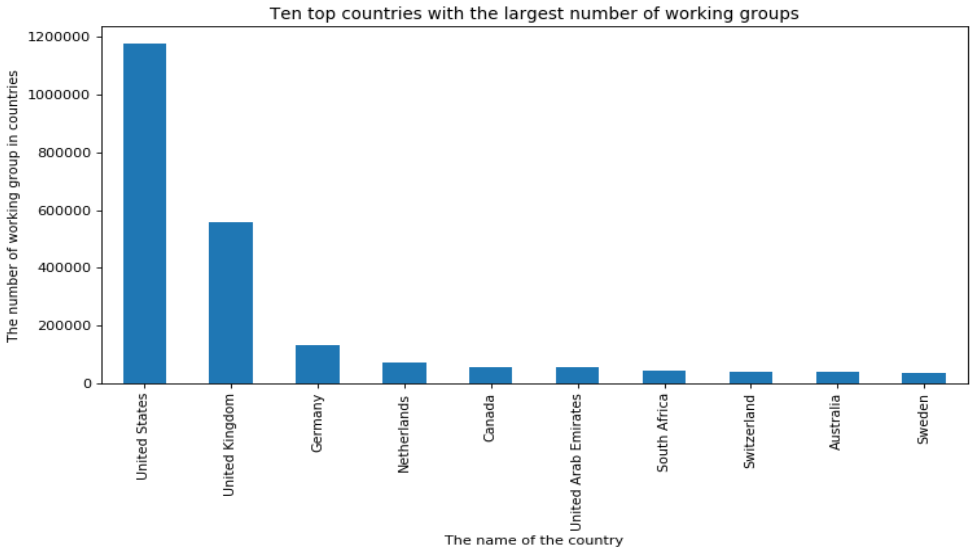
Dataframe column names along with the type of content:

```
action              object
app_inferred        object
os_inferred         object
country             object
datacenter          object
event_id            object
automated_action      bool
space_used          object
timestamp           object
workgroup_id        object
dtype: object
```

- Advanced statistics
    - The graph shows the distribution of shares in logs. The chart shows which actions are most often performed. As you can see in the chart, it's 'ADD_FILE' and 'MOVE_EXPLODED'. This information may be useful for automating specific activities and increasing resources.



    - The graph presents the distribution of observations divided into the countries in which the action was made. Thanks to this, you can easily see in which country the most actions are performed. This is important because of the balanced load, adjusting the mechanisms of access to data, bandwidth, and memory.



- The next chart shows the distribution of actions performed on individual hours during the day. Thanks to this, you can find out at what times the most activities are performed. This can be useful when the company wants to be prepared for high traffic and counteract problems resulting, for example, from overloading. Then the company knows when to add more resources.

2.2. Data (logs) from Egnyte Connect domain

- Basic statistics

Dataframe column names:

```
Index(['account_type', 'data_center', 'domain_status', 'plan_name',
       'pu_bought', 'pu_used', 'storage_bought_MB', 'storage_used_MB',
       'subscription_date', 'trial_end_date', 'trial_start_date',
       'workgroup_id'],
      dtype='object')
```

Dataframe column names along with the type of content:

```
account_type          object
data_center           object
domain_status         object
plan_name             object
pu_bought            float64
pu_used              float64
storage_bought_MB    float64
storage_used_MB      float64
subscription_date     object
trial_end_date        object
trial_start_date      object
workgroup_id          object
dtype: object
```

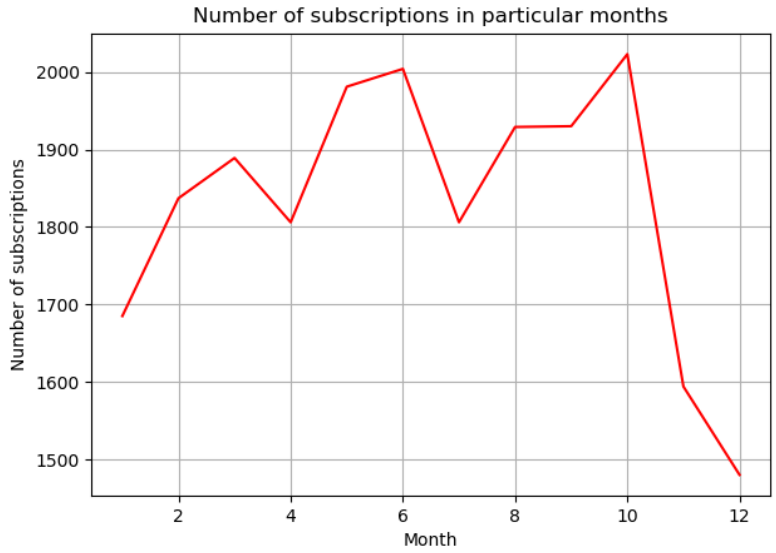
Information about a DataFrame including the index dtype and column dtypes, non-nul values and memory usage:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21964 entries, 26114 to 400478
Data columns (total 12 columns):
account_type          21964 non-null object
data_center           21964 non-null object
domain_status         21964 non-null object
plan_name             21964 non-null object
pu_bought             21964 non-null float64
pu_used               21964 non-null float64
storage_bought_MB     21964 non-null float64
storage_used_MB       21964 non-null float64
subscription_date     21964 non-null object
trial_end_date        21964 non-null object
trial_start_date      21964 non-null object
workgroup_id          21964 non-null object
dtypes: float64(4), object(8)
memory usage: 2.2+ MB
```

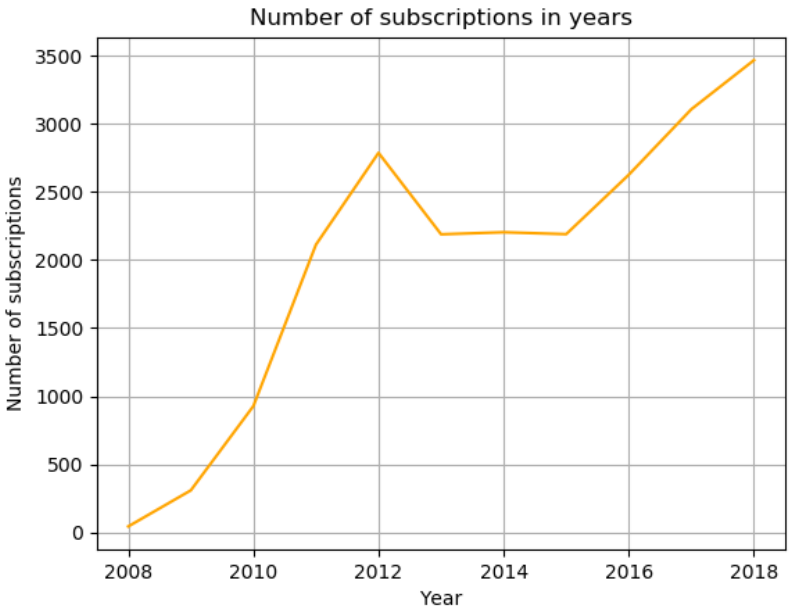
Descriptive statistics that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding NaN values:

	account_type	data_center	domain_status	plan_name	pu_bought	pu_used	storage_bought_MB	storage_used_MB	workgroup_id
count	21964	21964	21964	21964	21964.000000	21964.000000	2.196400e+04	2.196400e+04	21964
unique	9	3	7	415	NaN	NaN	NaN	NaN	21964
top	Regular	AVL	active	Office	NaN	NaN	NaN	NaN	c4468088-e00c-42ea-b98c-59d20c4c81b2
freq	20128	10167	18832	10757	NaN	NaN	NaN	NaN	1
mean	NaN	NaN	NaN	NaN	564.549445	53.063285	5.925713e+08	8.832032e+05	NaN
std	NaN	NaN	NaN	NaN	9840.871161	1558.682787	7.594657e+09	1.089321e+07	NaN
min	NaN	NaN	NaN	NaN	1.000000	1.000000	0.000000e+00	0.000000e+00	NaN
25%	NaN	NaN	NaN	NaN	5.000000	3.000000	1.000000e+06	4.902750e+03	NaN
50%	NaN	NaN	NaN	NaN	10.000000	5.000000	5.000000e+06	6.340300e+04	NaN
75%	NaN	NaN	NaN	NaN	21.000000	12.000000	5.000000e+06	3.171055e+05	NaN
max	NaN	NaN	NaN	NaN	1000000.000000	199500.000000	1.000000e+11	9.815875e+08	NaN

- Advanced statistics
  - The chart shows the subsequent months of the year along with the number of subscriptions. As you can see, the most subscriptions are bought in the middle of the year and in the autumn, when the year slowly ends. At the end of the year there is the least subscription, which is usually associated with the closing of expenses for a given year. The fall is also recorded during the holiday season.



- The graph shows how the number of purchased subscriptions has changed over the years. As you can see, the trend is growing.



4. Bibliography

In the process of implementing the solution, I used mainly the DASK and Matplotlib library documentation. In addition, I got acquainted with several articles/tutorials on the Internet in the field of using the DASK library in data processing.