5) Data Understanding.

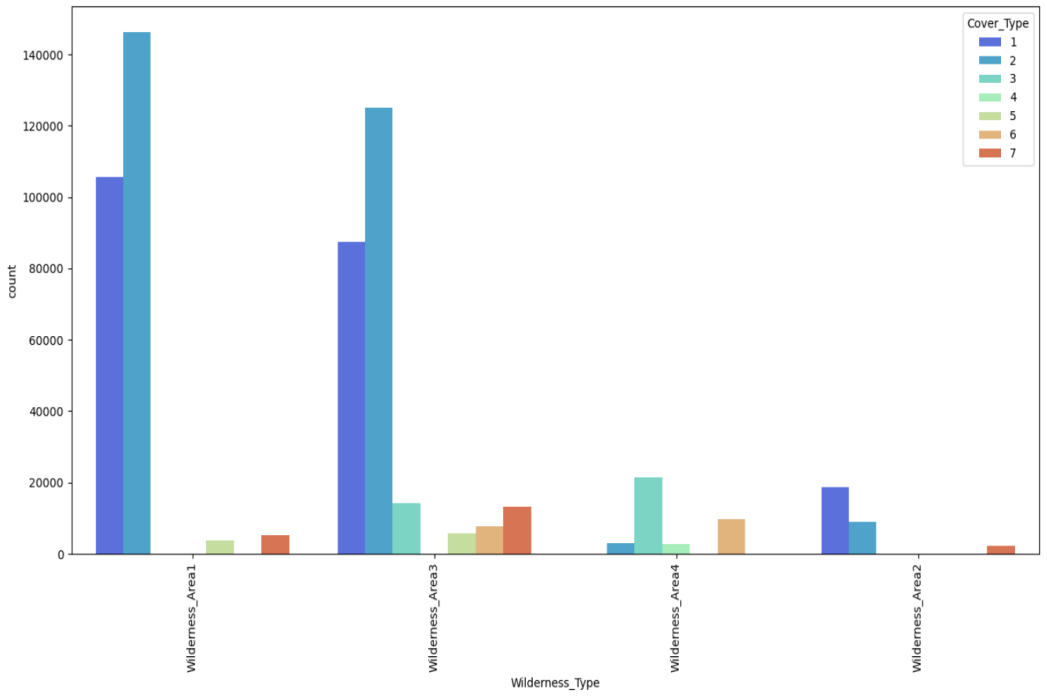
a) Exploratory Data Analysis:

Exploratory Data Analysis is made using the AWS SageMaker Jupyter Notebooks.

Explorartory Data Analysis

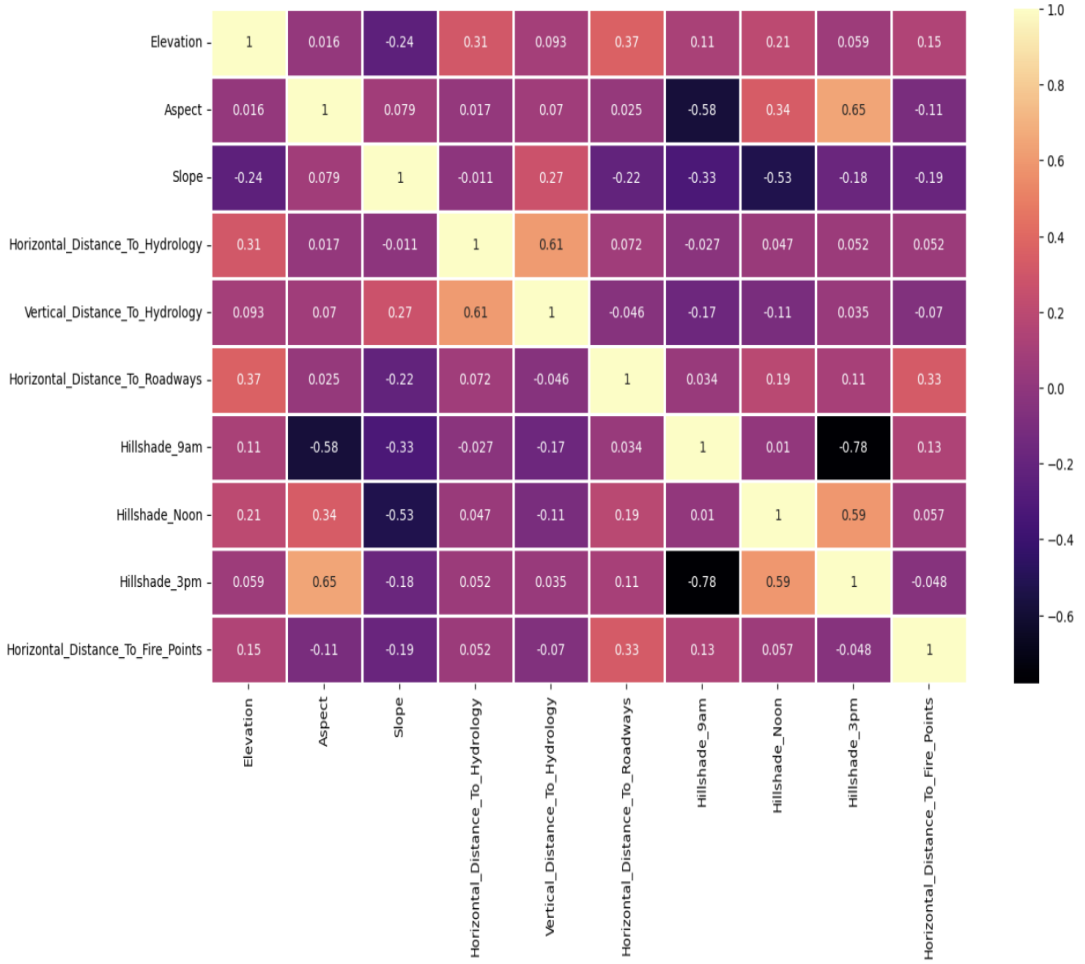
In [14]:
```
def rev_code(row):
    for c in Wilderness_data.columns:
        if row[c]==1:
            return c

data['Wilderness_Type']=Wilderness_data.apply(rev_code, axis=1)
plt.figure(figsize=(16,8))
sns.countplot(x='Wilderness_Type', hue='Cover_Type',data=data, palette="rainbow")
plt.xticks(rotation=90)
```

Out[14]:
```
(array([0, 1, 2, 3]),
 [Text(0, 0, 'Wilderness_Area1'),
  Text(1, 0, 'Wilderness_Area3'),
  Text(2, 0, 'Wilderness_Area4'),
  Text(3, 0, 'Wilderness_Area2')])
```



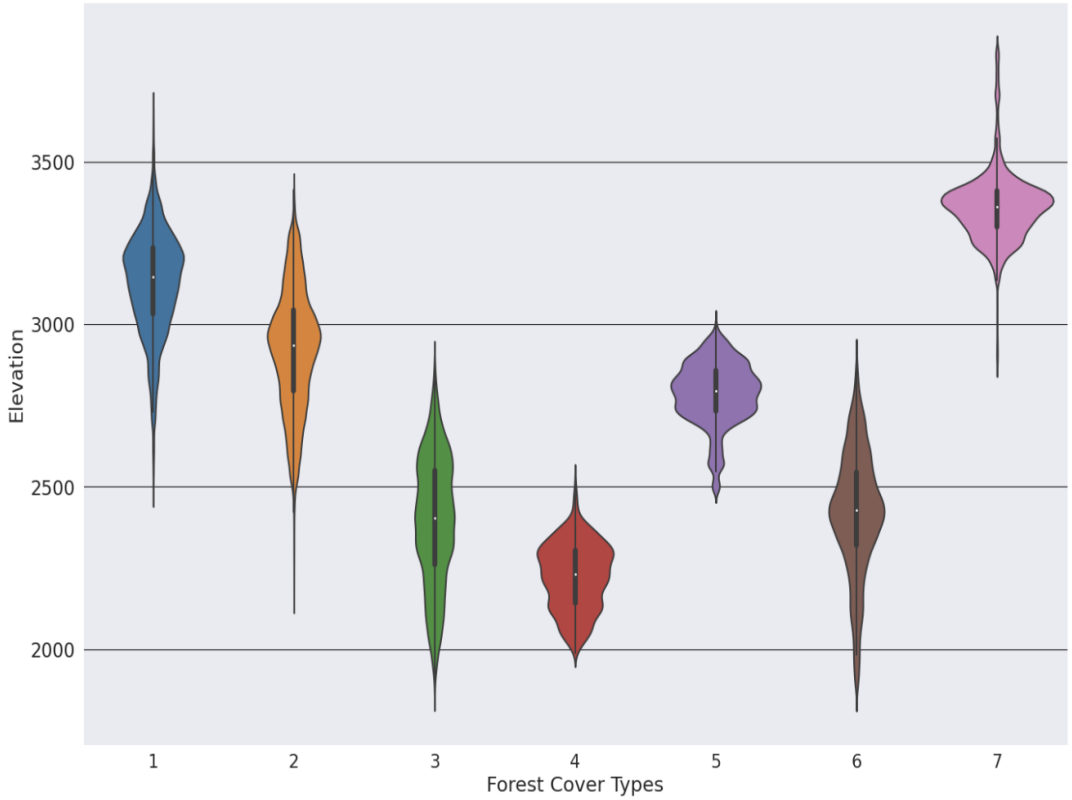
In [15]:
```
plt.figure(figsize=(15,8))
sns.heatmap(continous_data.corr(),cmap='magma',linecolor='white',linewidths=1,annot=True)
```

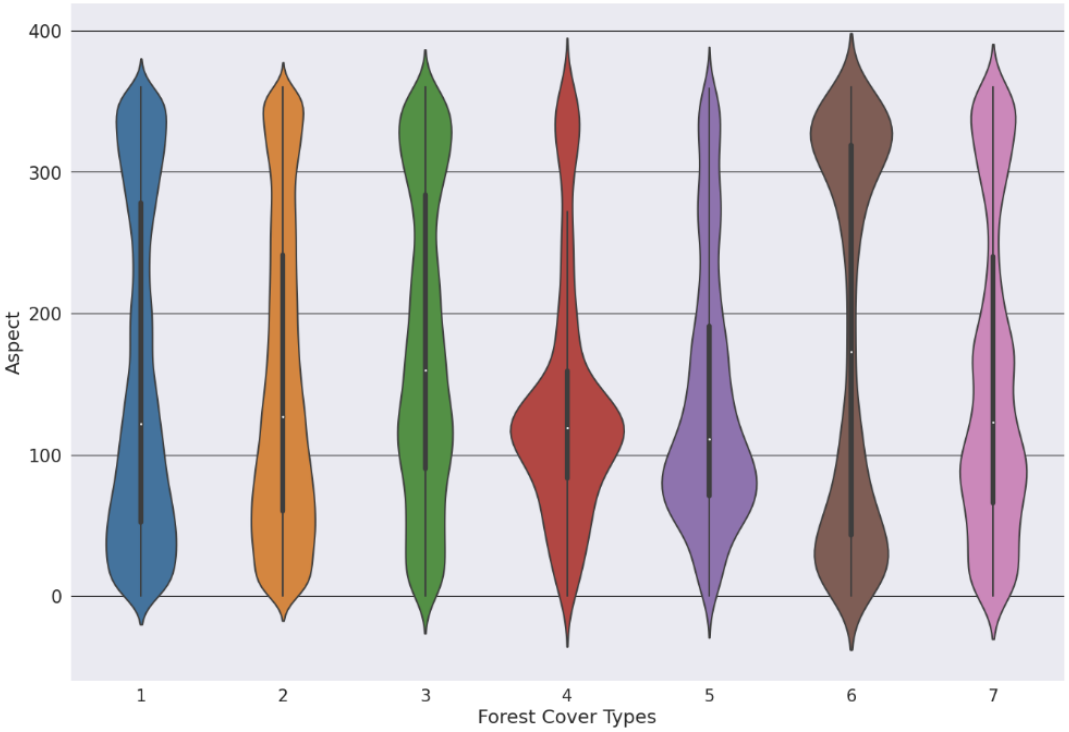
Out[15]: <AxesSubplot: >

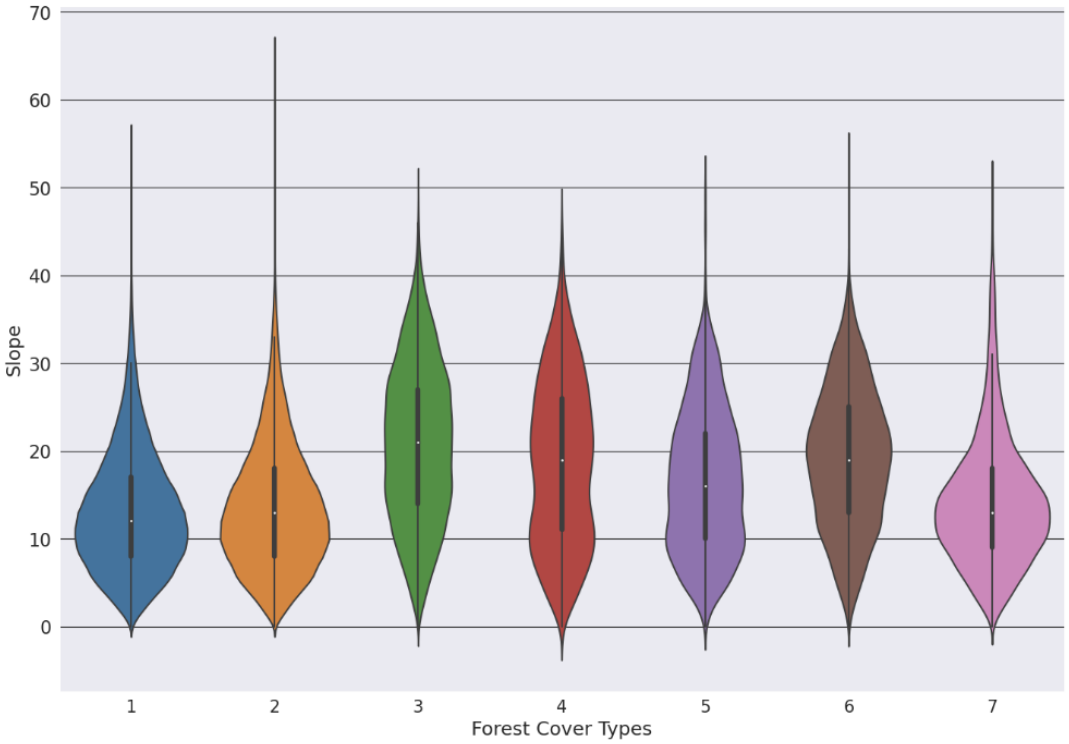


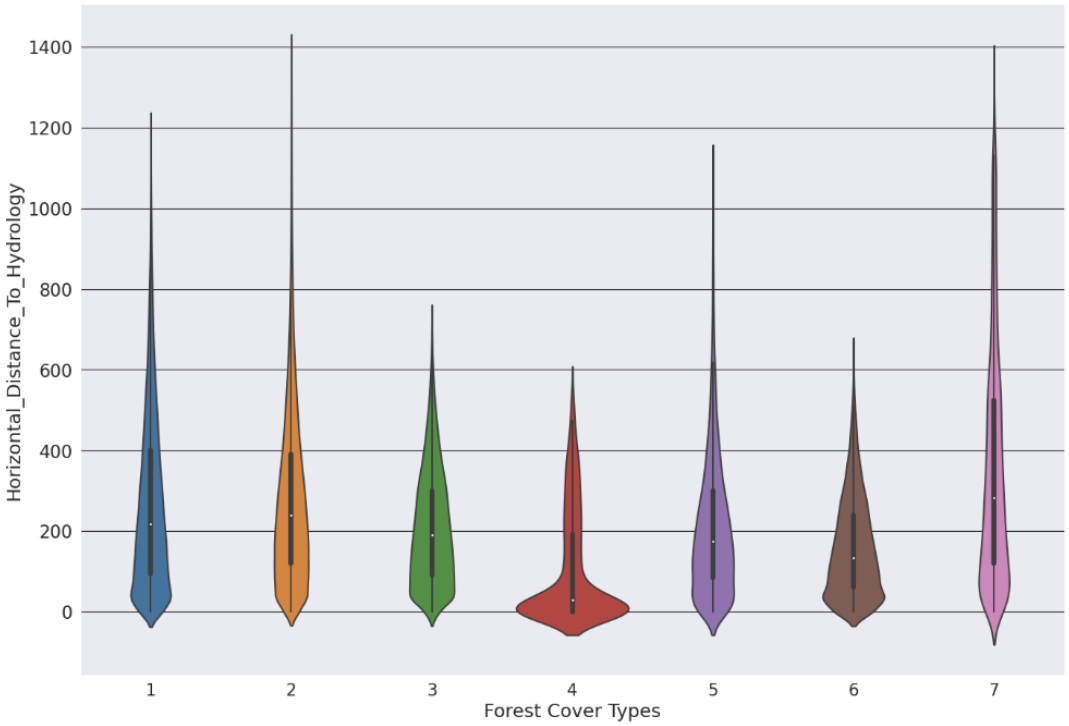
Some of the features in the dataset exhibit a strong correlation with each other. Specifically, the Hillshade_9am and Hillshade_3pm features are highly correlated, as well as the Aspect and Hillshade_3pm features.

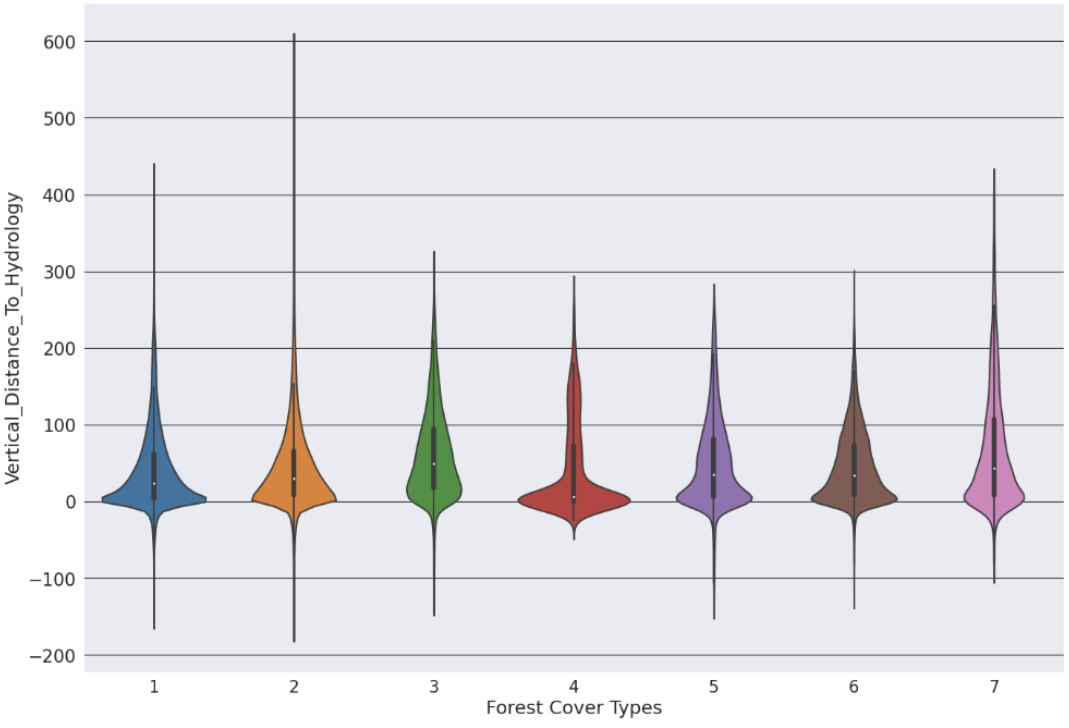
In [26]:
```
sns.set_style("darkgrid", {'grid.color': '.1'})

target = data['Cover_Type']
features = continous_data.columns


# loop for plotting Violin Plot for each features in the data
for i in range(0, len(features)):

    plt.subplots(figsize=(16, 11))
    sns.violinplot(data=continous_data, x=target, y = features[i])
    plt.xticks(size = 15)
    plt.yticks(size = 16)

    # Horizontal axis Label
    plt.xlabel('Forest Cover Types', size = 17)
    # Vertical axis Label
    plt.ylabel(features[i], size = 17)
    plt.show()
```

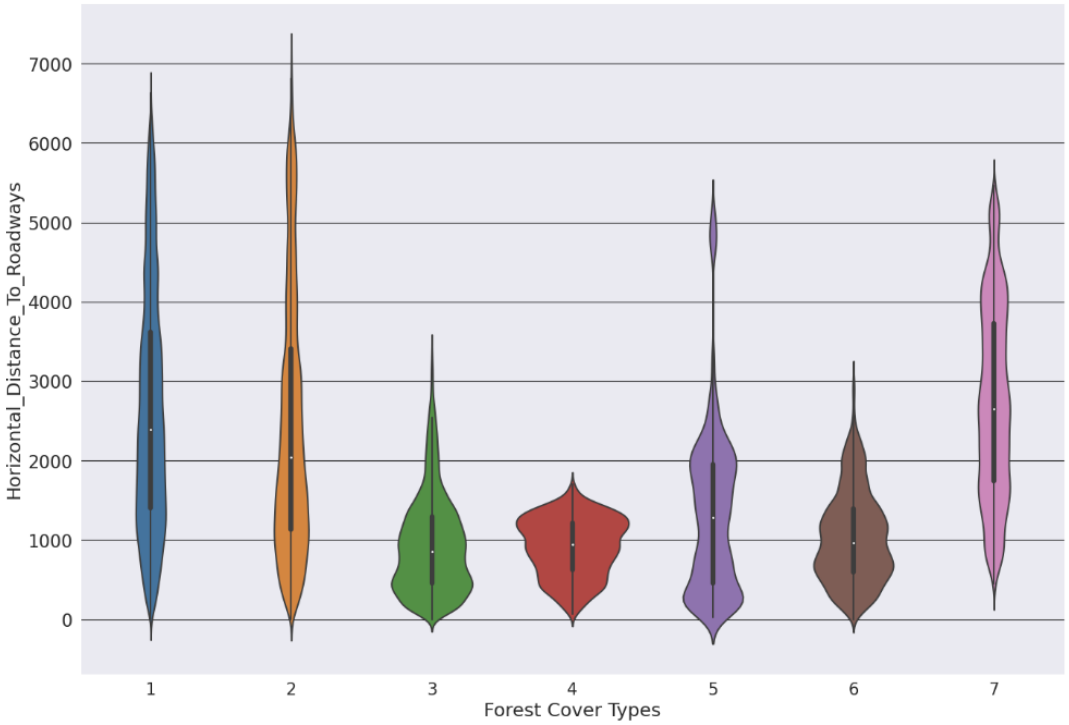


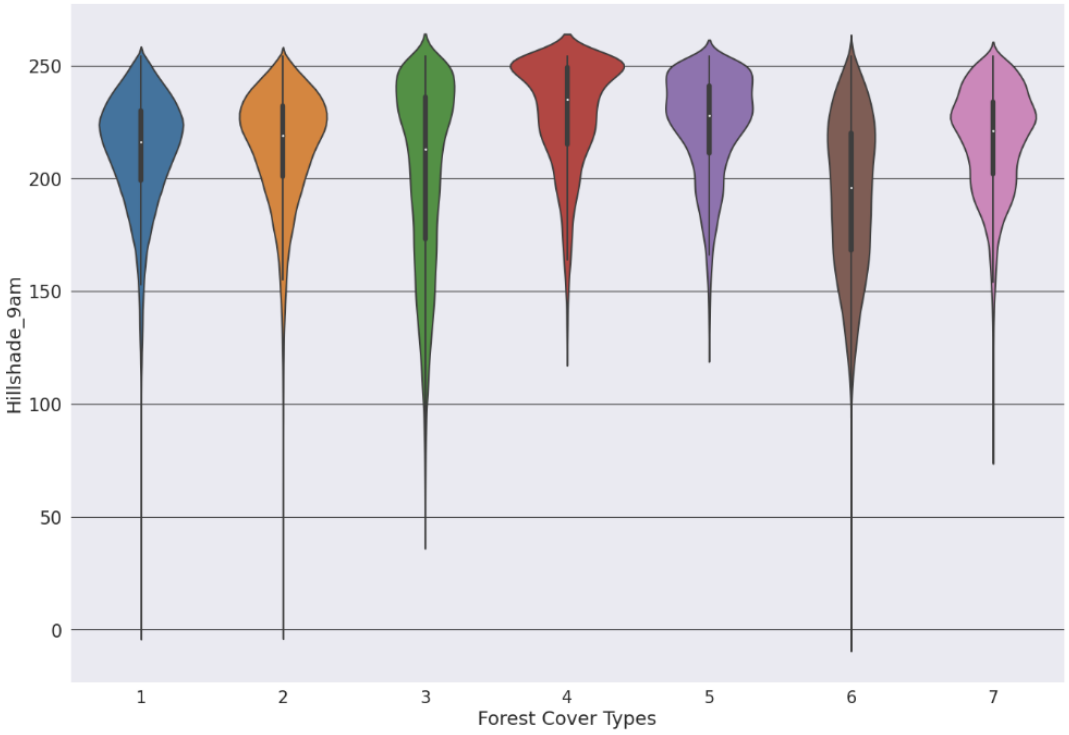

400
300
200
100
0
Aspect
1
2
3
4
5
6
7
Forest Cover Types

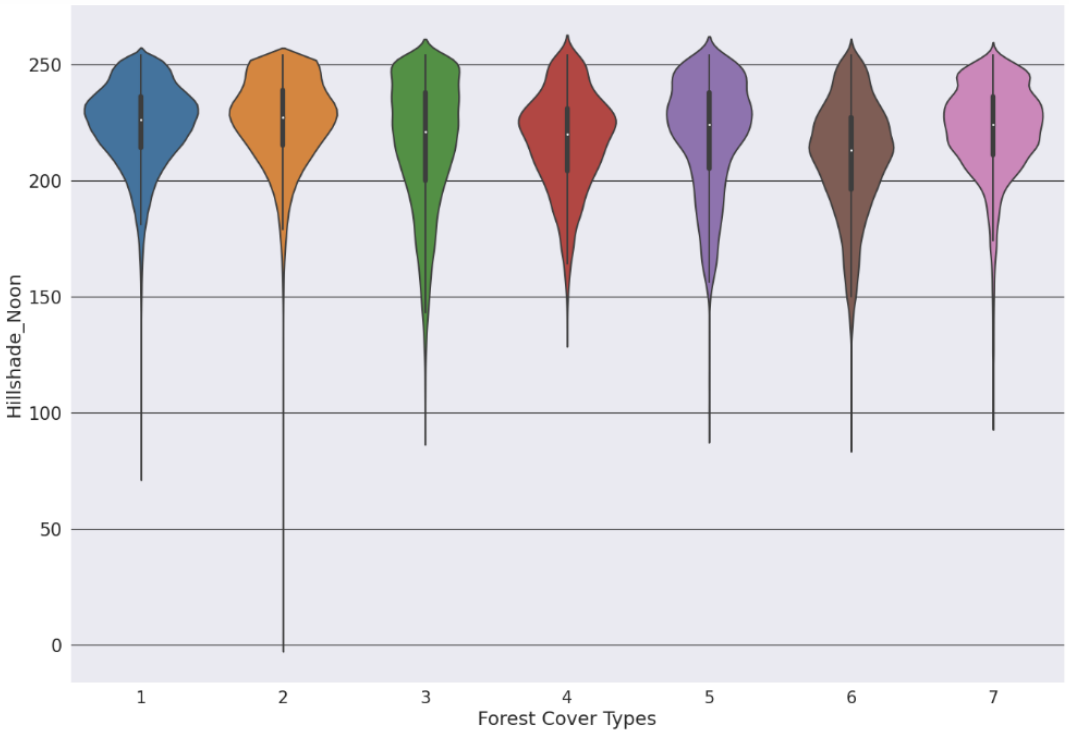

70
60
50
40
30
20
10
0
Slope
1
2
3
4
5
6
7
Forest Cover Types

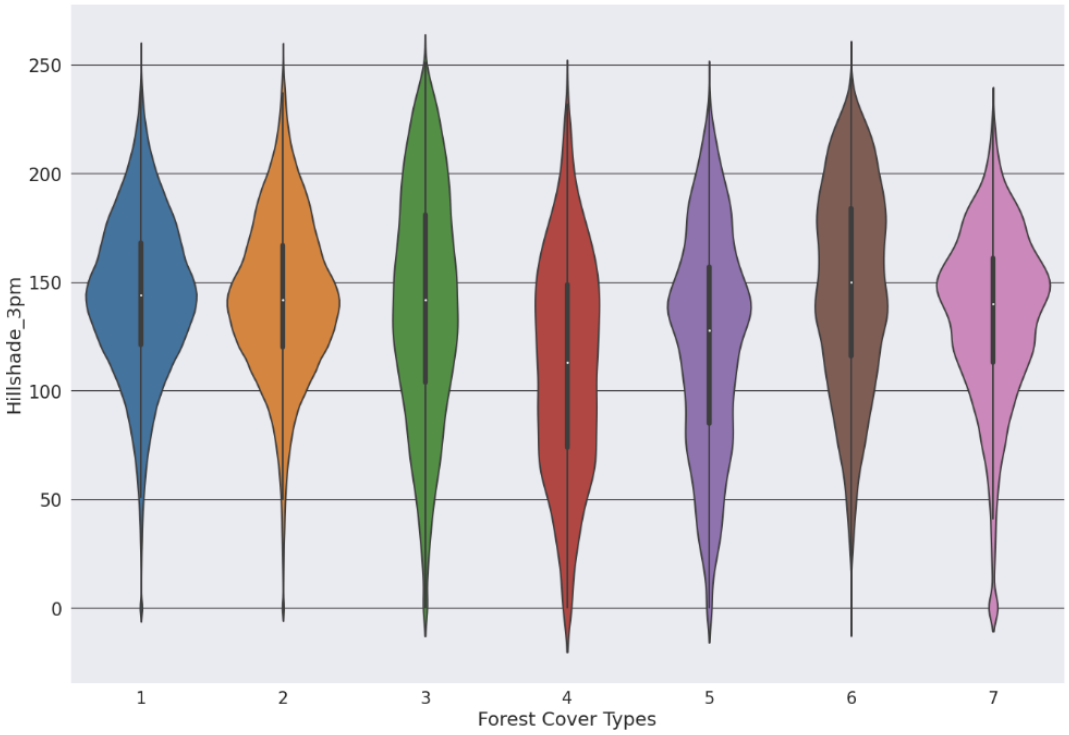

Horizontal_Distance_To_Hydrology
1400
1200
1000
800
600
400
200
0
1
2
3
4
5
6
7
Forest Cover Types

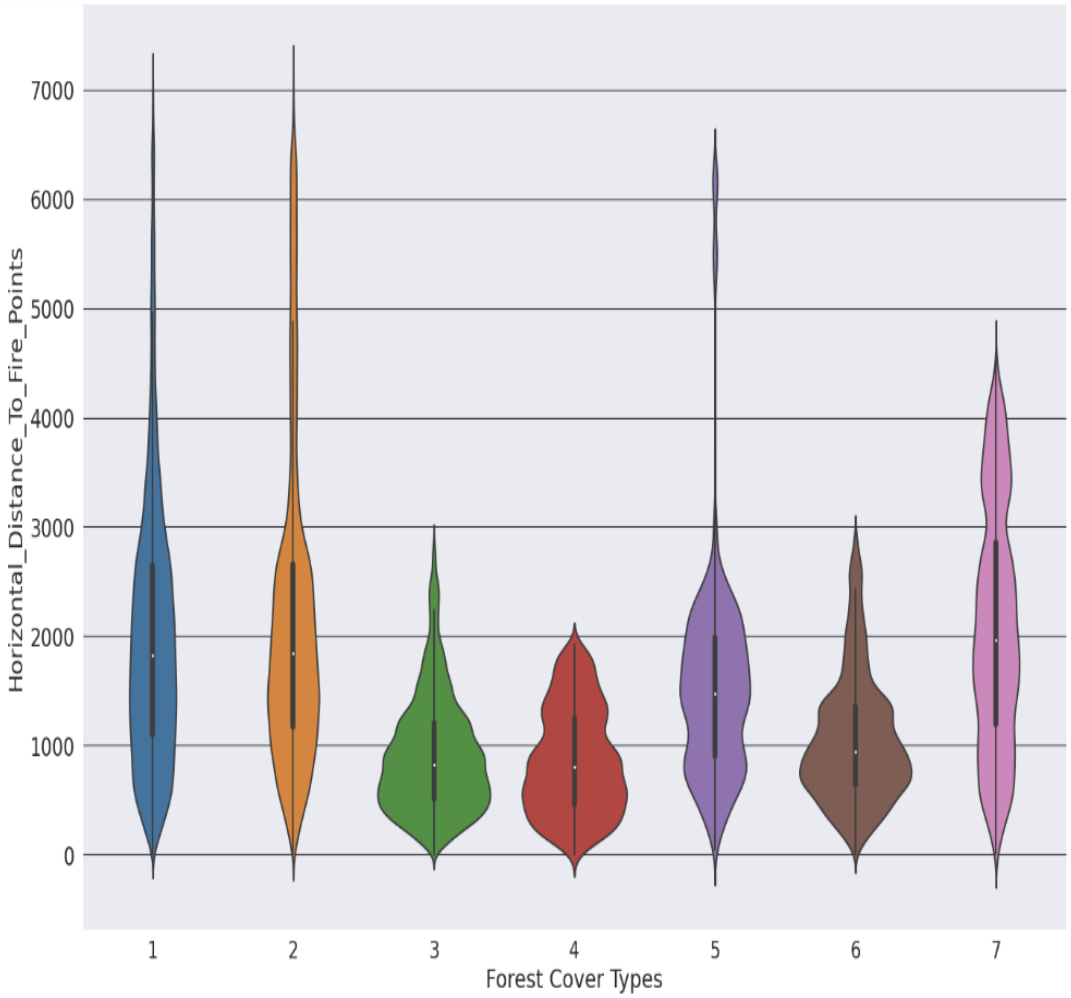

Vertical_Distance_To_Hydrology
600
500
400
300
200
100
0
−100
−200
1
2
3
4
5
6
7
Forest Cover Types


Horizontal_Distance_To_Roadways
7000
6000
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
Forest Cover Types


Hillshade_9am
250
200
150
100
50
0
1
2
3
4
5
6
7
Forest Cover Types


Hillshade_Noon
250
200
150
100
50
0
1
2
3
4
5
6
7
Forest Cover Types


Hillshade_3pm
250
200
150
100
50
0
1
2
3
4
5
6
7
Forest Cover Types



-The different features of the forest cover types have varying distributions and range of values.

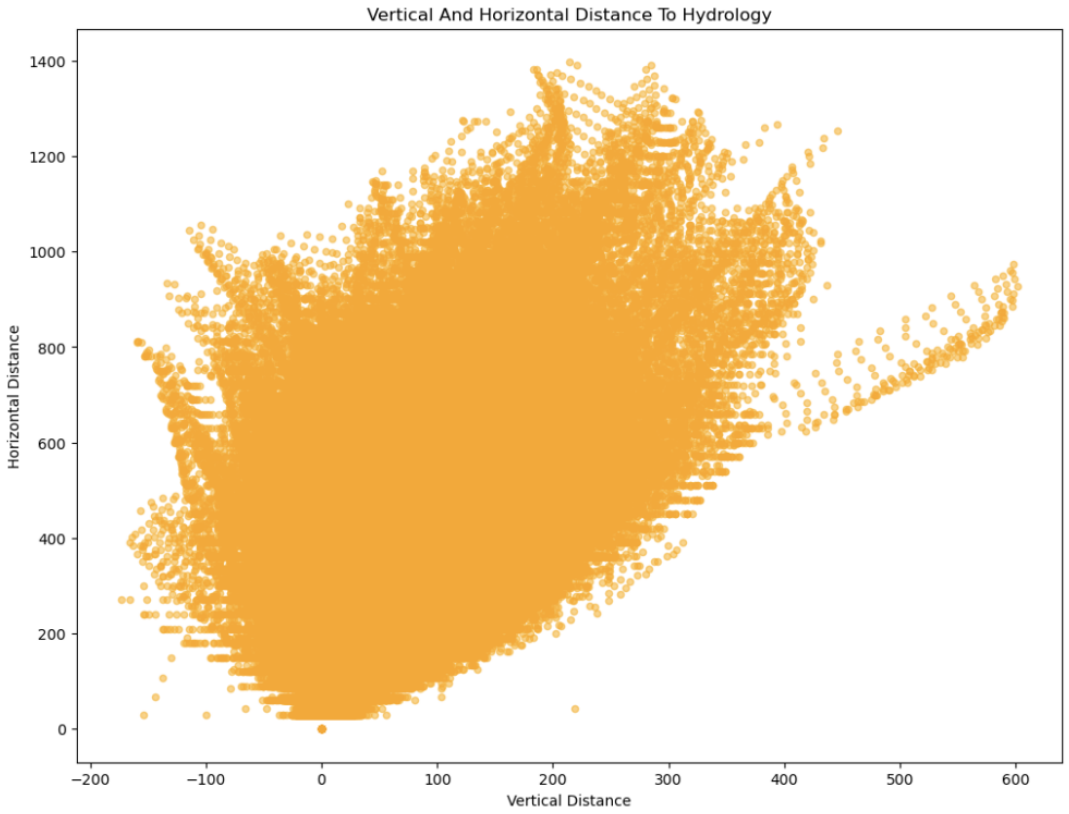
-Elevation is an important feature as it distinguishes between the different forest classes, with class 7 having the highest elevations. Aspect has a normal distribution for all classes.

-Slope has the least maximum value and is mainly observed in Forest Cover Type 2.

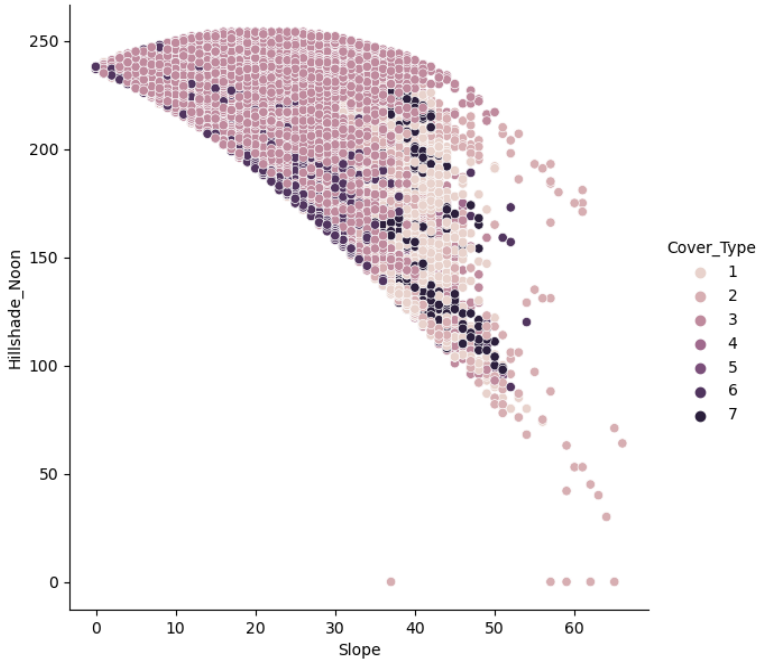
-The horizontal and vertical distances to hydrology are positively skewed, with most values concentrated towards 0-50m. The highest value in the vertical distance to hydrology feature is observed in Forest Cover Type 2, while it has the least minimum value across all features.

-Hillshade_9am and Hillshade_Noon are negatively skewed, while Hillshade_3pm has a normal distribution across all classes.

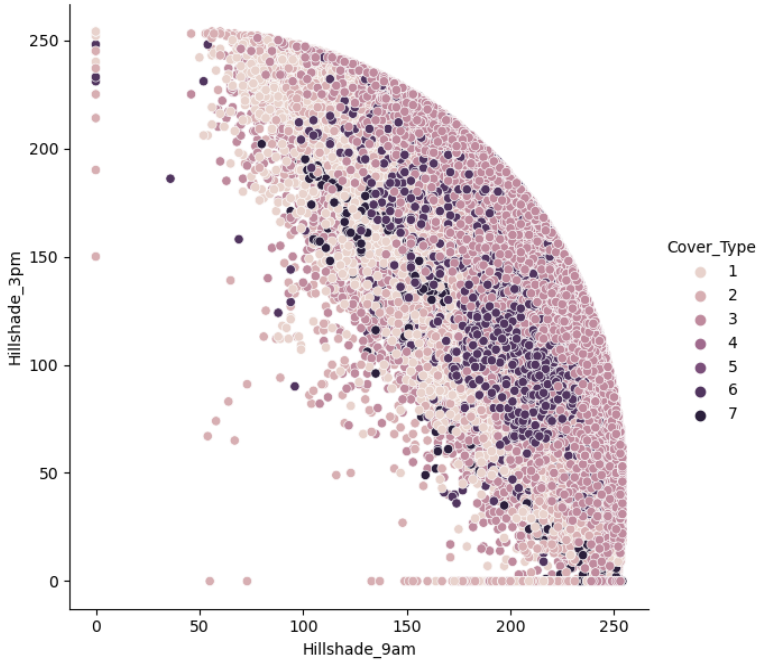
In [16]:
```
data.plot(kind='scatter', x='Vertical_Distance_To_Hydrology',
              y='Horizontal_Distance_To_Hydrology', alpha=0.5,
              color='orange', figsize = (12,9))

plt.title('Vertical And Horizontal Distance To Hydrology')
plt.xlabel("Vertical Distance")
plt.ylabel("Horizontal Distance")

plt.show()
```



In [17]:
```
sns.pairplot(data, hue="Cover_Type", height=6, x_vars="Slope",y_vars="Hillshade_Noon" )
plt.show()
```

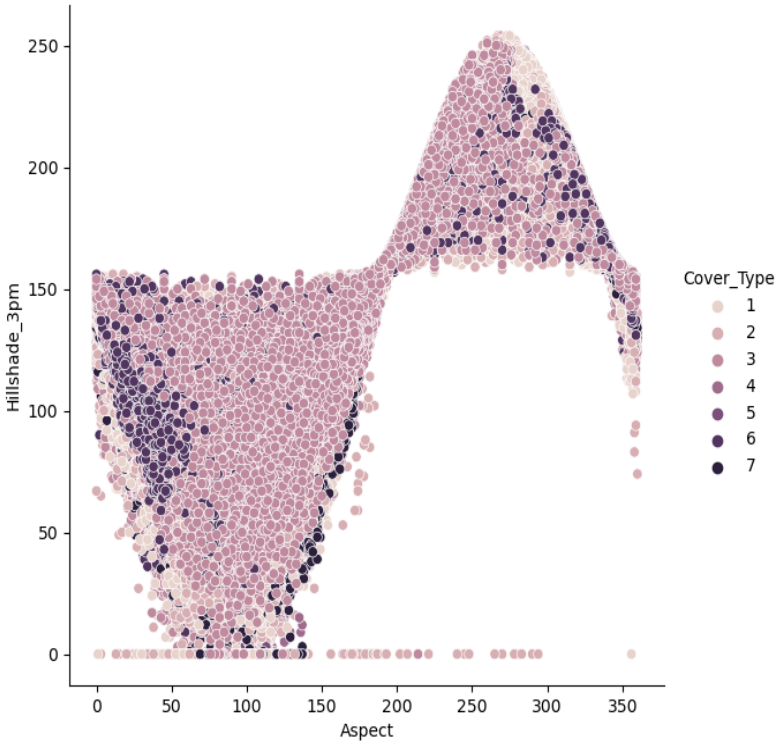


In [18]:
```
sns.pairplot(data, hue="Cover_Type", height=6, x_vars="Hillshade_9am",y_vars="Hillshade_3pm" )
plt.show()
```




```
Hillshade patterns give a nice ellipsoid patterns with each other
```

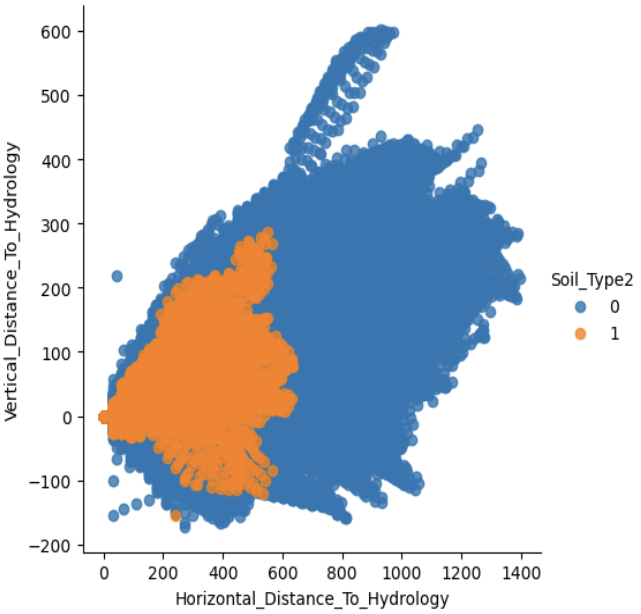
In [19]:
```
sns.pairplot(data, hue="Cover_Type", height=6, x_vars="Aspect",y_vars="Hillshade_3pm" )
plt.show()
```



Aspect and Hillshades attributes form a sigmoid pattern

In [21]:
```
sns.lmplot(x='Horizontal_Distance_To_Hydrology', y='Vertical_Distance_To_Hydrology', data=data, hue='Soil_Type2',fit_re
```

Out[21]:
```
<seaborn.axisgrid.FacetGrid at 0x7f50ea7c7af0>
```



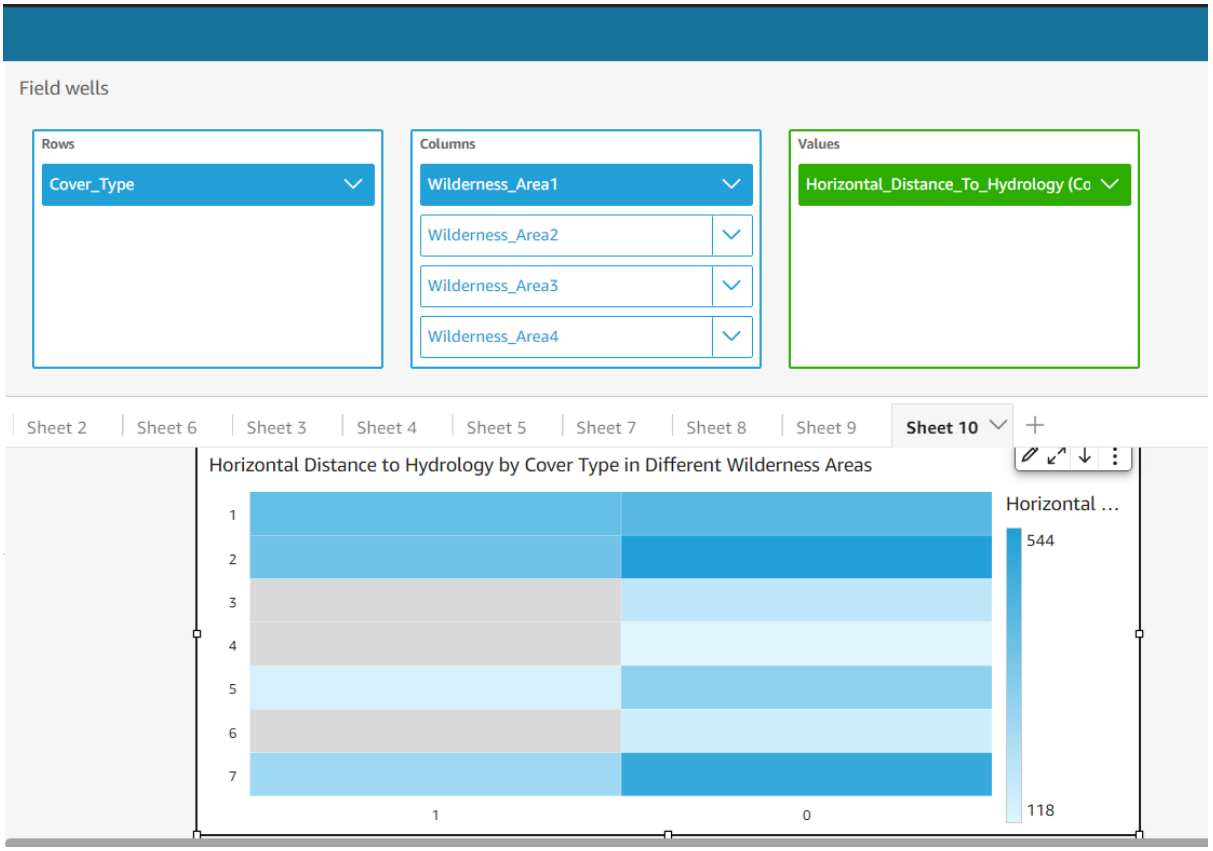
From the above plot, it is clear that Soil type 2 has lower Vertical and Horizontal distance to hydrology.
They are high correlated.

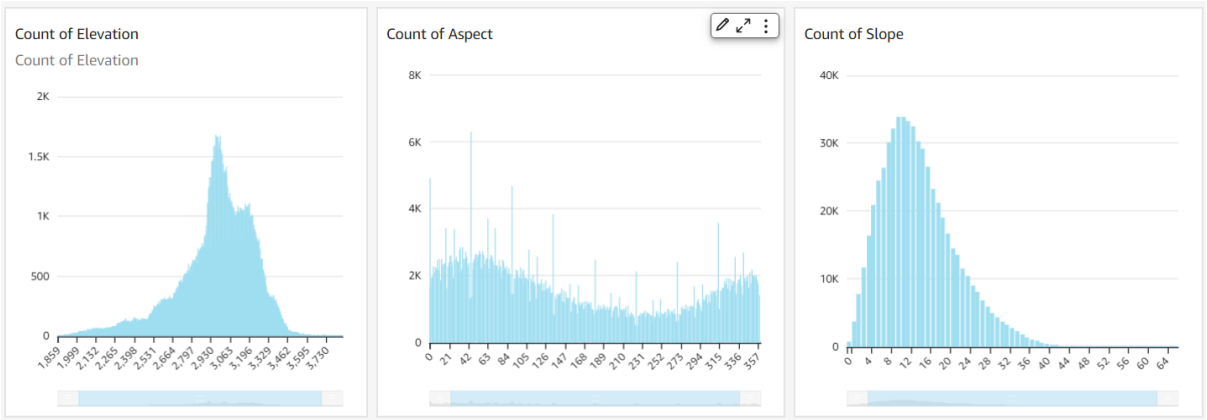
b) Dashboard

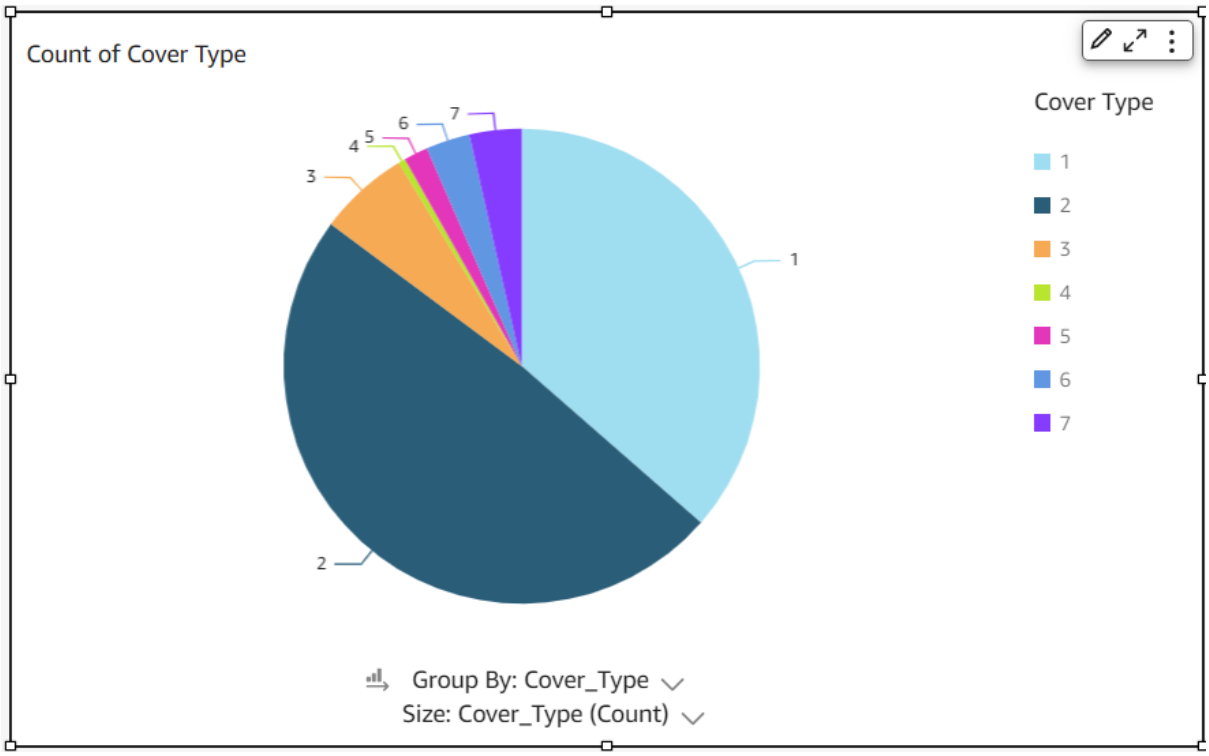
Dashboard is created with the help of AWS QuickSight. AWS QuickSight is a cloud-based business intelligence service that allows users to create interactive dashboards, perform ad-hoc analysis, and generate reports from various data sources.

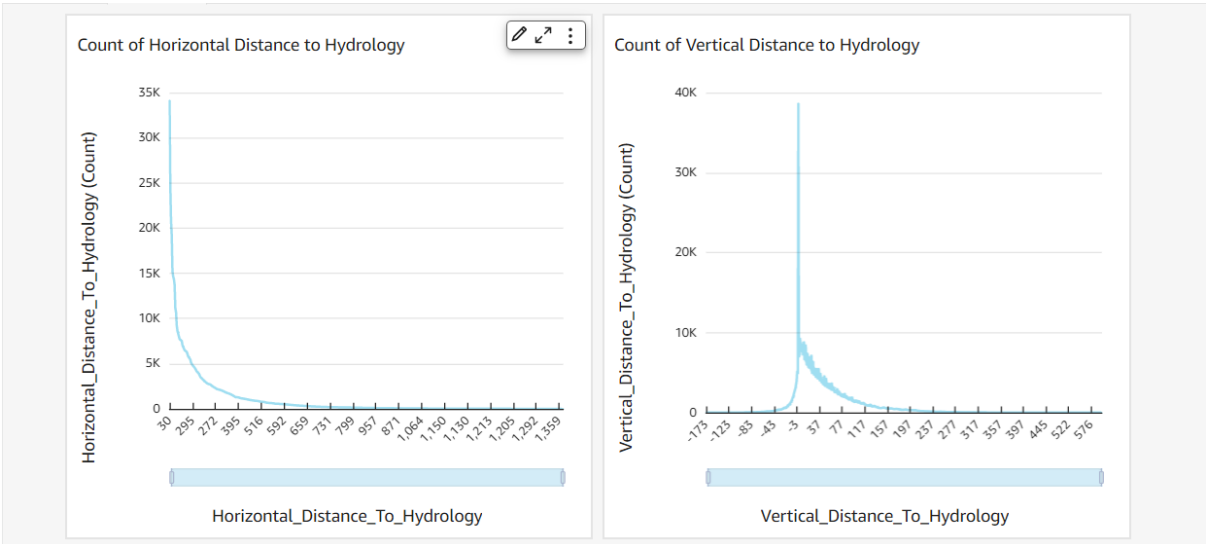
This heat map shows the relationship between different cover types and wilderness areas

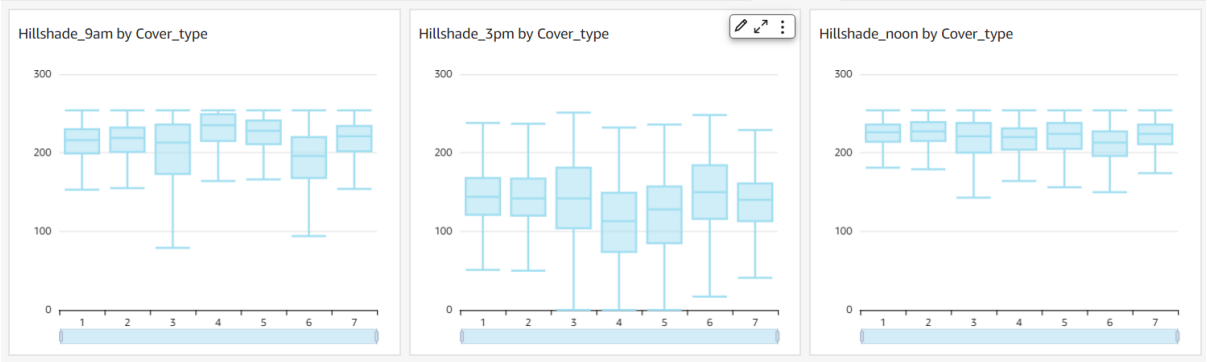
When we select wilderness area 1 heap map is generated for all cover types with wilderness area as 1. The same applies for wilderness area 2,3 and 4

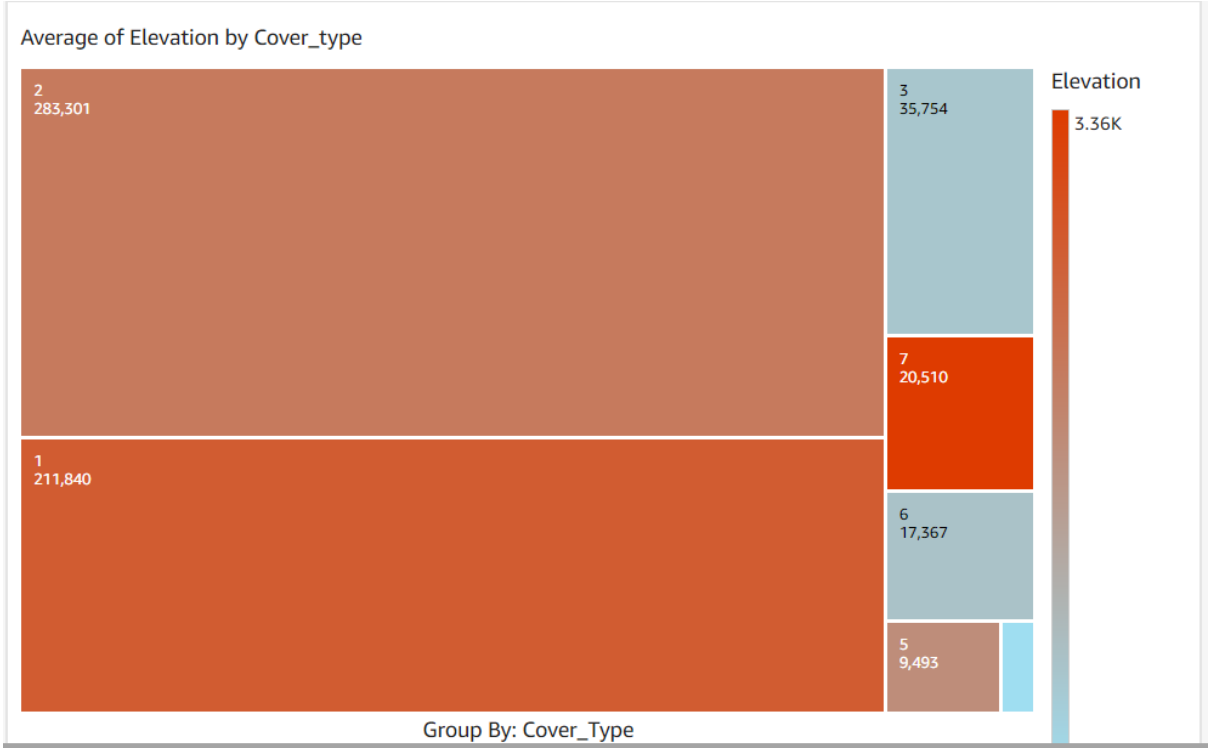



Count of Elevation
Count of Elevation
2K
1.5K
1K
500
0
1,859
1,999
2,132
2,265
2,398
2,531
2,664
2,797
2,930
3,063
3,196
3,329
3,462
3,595
3,730
Count of Aspect
8K
6K
4K
2K
0
0
21
42
63
84
105
126
147
168
189
210
231
252
273
294
315
336
357
Count of Slope
40K
30K
20K
10K
0
0
4
8
12
16
20
24
28
32
36
40
44
48
52
56
60
64


Count of Cover Type
Cover Type
1
2
3
4
5
6
7
Group By: Cover_Type
Size: Cover_Type (Count)


Count of Horizontal Distance to Hydrology
Horizontal_Distance_To_Hydrology (Count)
35K
30K
25K
20K
15K
10K
5K
0
30
295
272
395
516
592
659
731
799
957
871
1,064
1,150
1,130
1,213
1,205
1,292
1,359
Horizontal_Distance_To_Hydrology
Count of Vertical Distance to Hydrology
Vertical_Distance_To_Hydrology (Count)
40K
30K
20K
10K
0
-173
-123
-83
-43
-3
37
77
117
157
197
237
277
317
357
397
445
522
576
Vertical_Distance_To_Hydrology


Hillshade_9am by Cover_type
Hillshade_3pm by Cover_type
Hillshade_noon by Cover_type
300
200
100
0
1
2
3
4
5
6
7


Average of Elevation by Cover_type
2
283,301
1
211,840
3
35,754
7
20,510
6
17,367
5
9,493
Elevation
3.36K
Group By: Cover_Type

6) Data Preparation

In [5]:
```
print(data.columns)
```

```
Index(['Elevation', 'Aspect', 'Slope', 'Horizontal_Distance_To_Hydrology',
       'Vertical_Distance_To_Hydrology', 'Horizontal_Distance_To_Roadways',
       'Hillshade_9am', 'Hillshade_Noon', 'Hillshade_3pm',
       'Horizontal_Distance_To_Fire_Points', 'Wilderness_Area1',
       'Wilderness_Area2', 'Wilderness_Area3', 'Wilderness_Area4',
       'Soil_Type1', 'Soil_Type2', 'Soil_Type3', 'Soil_Type4', 'Soil_Type5',
       'Soil_Type6', 'Soil_Type7', 'Soil_Type8', 'Soil_Type9', 'Soil_Type10',
       'Soil_Type11', 'Soil_Type12', 'Soil_Type13', 'Soil_Type14',
       'Soil_Type15', 'Soil_Type16', 'Soil_Type17', 'Soil_Type18',
       'Soil_Type19', 'Soil_Type20', 'Soil_Type21', 'Soil_Type22',
       'Soil_Type23', 'Soil_Type24', 'Soil_Type25', 'Soil_Type26',
       'Soil_Type27', 'Soil_Type28', 'Soil_Type29', 'Soil_Type30',
       'Soil_Type31', 'Soil_Type32', 'Soil_Type33', 'Soil_Type34',
       'Soil_Type35', 'Soil_Type36', 'Soil_Type37', 'Soil_Type38',
       'Soil_Type39', 'Soil_Type40', 'Cover_Type'],
      dtype='object')
```

In [6]:
```
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 581012 entries, 0 to 581011
Data columns (total 55 columns):
 #   Column                              Non-Null Count   Dtype
---  ------                              --------------   -----
 0   Elevation                           581012 non-null  int64
 1   Aspect                              581012 non-null  int64
 2   Slope                               581012 non-null  int64
 3   Horizontal_Distance_To_Hydrology    581012 non-null  int64
 4   Vertical_Distance_To_Hydrology      581012 non-null  int64
 5   Horizontal_Distance_To_Roadways     581012 non-null  int64
 6   Hillshade_9am                       581012 non-null  int64
 7   Hillshade_Noon                      581012 non-null  int64
 8   Hillshade_3pm                       581012 non-null  int64
 9   Horizontal_Distance_To_Fire_Points  581012 non-null  int64
 10  Wilderness_Area1                    581012 non-null  int64
 11  Wilderness_Area2                    581012 non-null  int64
 12  Wilderness_Area3                    581012 non-null  int64
 13  Wilderness_Area4                    581012 non-null  int64
 14  Soil_Type1                          581012 non-null  int64
 15  Soil_Type2                          581012 non-null  int64
 16  Soil_Type3                          581012 non-null  int64
 17  Soil_Type4                          581012 non-null  int64
 18  Soil_Type5                          581012 non-null  int64
 19  Soil_Type6                          581012 non-null  int64
 20  Soil_Type7                          581012 non-null  int64
 21  Soil_Type8                          581012 non-null  int64
 22  Soil_Type9                          581012 non-null  int64
 23  Soil_Type10                         581012 non-null  int64
 24  Soil_Type11                         581012 non-null  int64
 25  Soil_Type12                         581012 non-null  int64
 26  Soil_Type13                         581012 non-null  int64
 27  Soil_Type14                         581012 non-null  int64
 28  Soil_Type15                         581012 non-null  int64
 29  Soil_Type16                         581012 non-null  int64
 30  Soil_Type17                         581012 non-null  int64
 31  Soil_Type18                         581012 non-null  int64
 32  Soil_Type19                         581012 non-null  int64
 33  Soil_Type20                         581012 non-null  int64
 34  Soil_Type21                         581012 non-null  int64
 35  Soil_Type22                         581012 non-null  int64
 36  Soil_Type23                         581012 non-null  int64
 37  Soil_Type24                         581012 non-null  int64
 38  Soil_Type25                         581012 non-null  int64
 39  Soil_Type26                         581012 non-null  int64
 40  Soil_Type27                         581012 non-null  int64
 41  Soil_Type28                         581012 non-null  int64
 42  Soil_Type29                         581012 non-null  int64
 43  Soil_Type30                         581012 non-null  int64
 44  Soil_Type31                         581012 non-null  int64
 45  Soil_Type32                         581012 non-null  int64
 46  Soil_Type33                         581012 non-null  int64
 47  Soil_Type34                         581012 non-null  int64
 48  Soil_Type35                         581012 non-null  int64
 49  Soil_Type36                         581012 non-null  int64
 50  Soil_Type37                         581012 non-null  int64
 51  Soil_Type38                         581012 non-null  int64
 52  Soil_Type39                         581012 non-null  int64
 53  Soil_Type40                         581012 non-null  int64
 54  Cover_Type                          581012 non-null  int64
dtypes: int64(55)
memory usage: 243.8 MB
None
```

In [7]:
```
#check for null values
data.isnull().sum()
```

Out[7]:
```
Elevation                           0
Aspect                              0
Slope                               0
Horizontal_Distance_To_Hydrology    0
Vertical_Distance_To_Hydrology      0
Horizontal_Distance_To_Roadways     0
Hillshade_9am                       0
Hillshade_Noon                      0
Hillshade_3pm                       0
Horizontal_Distance_To_Fire_Points  0
Wilderness_Area1                    0
Wilderness_Area2                    0
Wilderness_Area3                    0
Wilderness_Area4                    0
Soil_Type1                          0
Soil_Type2                          0
Soil_Type3                          0
Soil_Type4                          0
Soil_Type5                          0
Soil_Type6                          0
Soil_Type7                          0
Soil_Type8                          0
Soil_Type9                          0
Soil_Type10                         0
Soil_Type11                         0
Soil_Type12                         0
Soil_Type13                         0
Soil_Type14                         0
Soil_Type15                         0
Soil_Type16                         0
Soil_Type17                         0
Soil_Type18                         0
Soil_Type19                         0
Soil_Type20                         0
Soil_Type21                         0
Soil_Type22                         0
Soil_Type23                         0
Soil_Type24                         0
Soil_Type25                         0
Soil_Type26                         0
Soil_Type27                         0
Soil_Type28                         0
Soil_Type29                         0
Soil_Type30                         0
Soil_Type31                         0
Soil_Type32                         0
Soil_Type33                         0
Soil_Type34                         0
Soil_Type35                         0
Soil_Type36                         0
Soil_Type37                         0
Soil_Type38                         0
Soil_Type39                         0
Soil_Type40                         0
Cover_Type                          0
dtype: int64
```

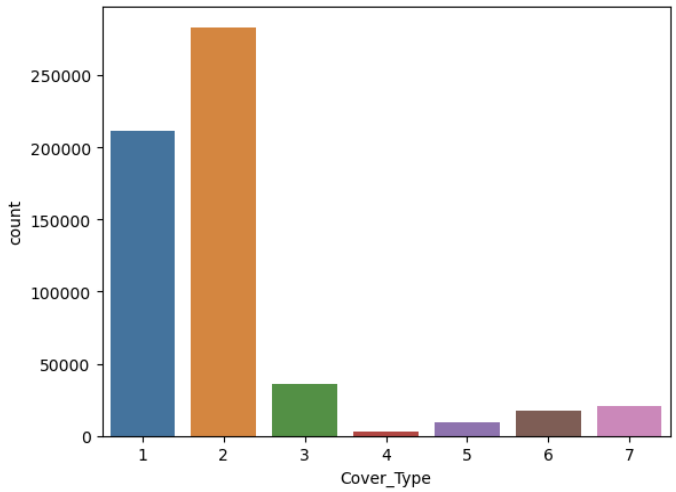
In [8]:
```
data.head()
```

Out[8]:

	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology	Horizontal_Distance_To_Roadways	Hillshade_9am	Hillshade_Noon
0	2596	51	3	258	0	510	221	232
1	2590	56	2	212	-6	390	220	235
2	2804	139	9	268	65	3180	234	238
3	2785	155	18	242	118	3090	238	238
4	2595	45	2	153	-1	391	220	234

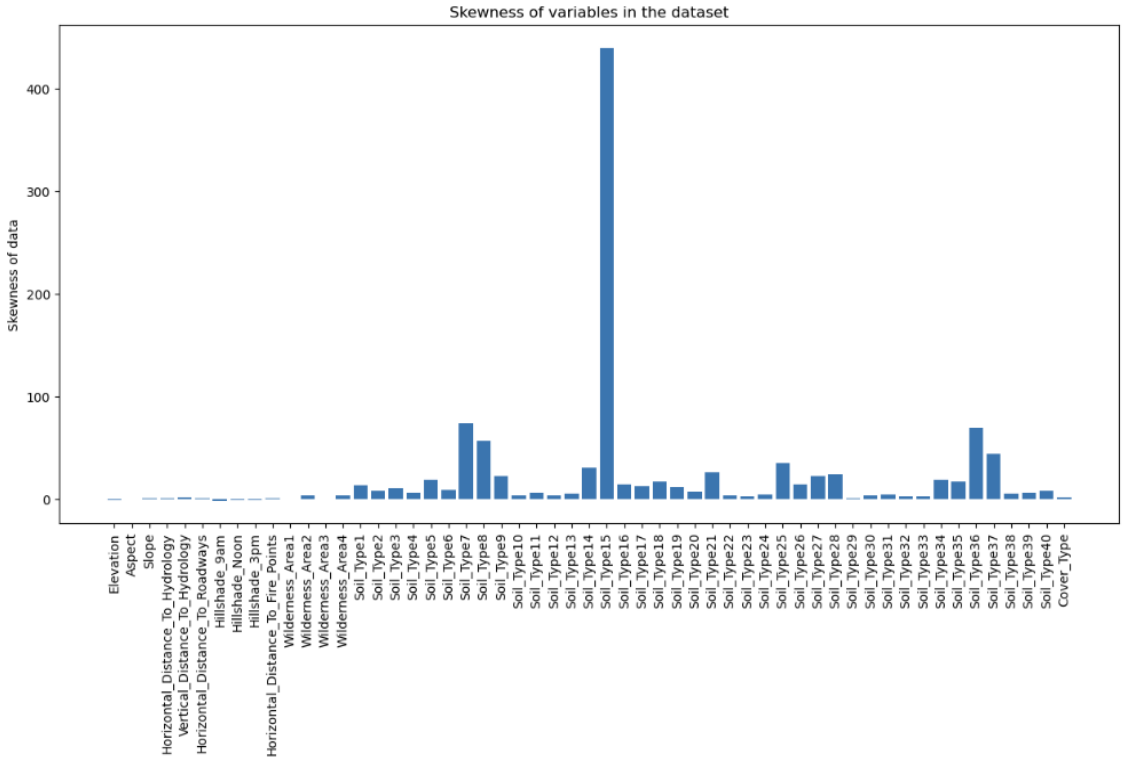
In [23]:
```
# Distribution of Cover_Type

sns.countplot(data=data, x='Cover_Type')
```

Out[23]:
```
<AxesSubplot: xlabel='Cover_Type', ylabel='count'>
```



it looks like "Cover_Type 2" is the most common type, accounting for almost 49% of the dataset, followed by "Cover_Type 1" at 36.46%. "Cover_Type 4" is the least common, at only 0.47%.

In [10]:
```
#lets look at the skewness of the data

skewness = data.skew()

# Create a bar plot using Matplotlib
fig, ax = plt.subplots(figsize=(15, 7))
ax.bar(x=range(len(skewness)), height=skewness)
ax.set_xticks(range(len(skewness)))
ax.set_xticklabels(data.columns, rotation=90)
ax.set_ylabel('Skewness of data')
ax.set_title('Skewness of variables in the dataset')
plt.show()
```



The results show that some variables, especially those related to soil types, are highly skewed. This information is useful because it can help identify potential problems with the dataset, such as the existence of soil types with very few occurrences.

In [11]:
```
binary_data=data.loc[:,'Wilderness_Area1':'Soil_Type40']

continous_data=data.loc[:,'Elevation':'Horizontal_Distance_To_Fire_Points']

Wilderness_data=data.loc[:,'Wilderness_Area1': 'Wilderness_Area4']

Soiltype_data=data.loc[:,'Soil_Type1':'Soil_Type40']
```

In [12]:
```
for col in binary_data:
    count=binary_data[col].value_counts()
    print(col,count)
```

```
Wilderness_Area1 0    320216
1    260796
Name: Wilderness_Area1, dtype: int64
Wilderness_Area2 0    551128
1     29884
Name: Wilderness_Area2, dtype: int64
Wilderness_Area3 0    327648
1    253364
Name: Wilderness_Area3, dtype: int64
Wilderness_Area4 0    544044
1     36968
Name: Wilderness_Area4, dtype: int64
Soil_Type1 0    577981
1      3031
Name: Soil_Type1, dtype: int64
Soil_Type2 0    573487
1      7525
Name: Soil_Type2, dtype: int64
Soil_Type3 0    576189
```

In [13]:
```
print('Soil Type','count')
for col in binary_data:
    count=binary_data[col].value_counts()[1] #considering all one's among 1 and 0's in each soil type
    if count < 1000:
        print(col,count)
```

```
Soil Type  Occurence_count
Soil_Type7 105
Soil_Type8 179
Soil_Type14 599
Soil_Type15 3
Soil_Type21 838
Soil_Type25 474
Soil_Type28 946
Soil_Type36 119
Soil_Type37 298
```

The code identifies that some soil types have a very low occurrence count. Although a dataset doesn't necessarily have to be balanced, statistically speaking, one would expect around 14,500 observations per soil type with a total of 40 soil types and over 500,000 records. However, the code highlights that there are soil types with a smaller number of observations. Therefore, it suggests that it may be necessary to remove the features with a very small sample size to avoid data imbalances.

Removing columns with Standars Deviation zero, because they don't help in prediction process

Data cleaning

In [*]:
```
rem = []

#Add constant columns as they don't help in prediction process
for c in data.columns:
    if data[c].std() == 0: #standard deviation is zero
        rem.append(c)

#drop the columns
df.drop(rem,axis=1,inplace=True)
```