In [1]:
```
import pandas as pd
import numpy as np
import os
from tqdm import tqdm

## To keep the plots even after closing the notebook
import plotly.io as pio
pio.renderers.default='notebook'
```

In [2]:
```
folder_path = 'output/'
# list all files in the directory
files = os.listdir(folder_path)

# filter files that start with 'Take_2' and end with '.csv'
filtered_files = [file for file in files if file.startswith('Take_2') and fi
filtered_files.sort()
# print the list of filtered files
filtered_files
```

Out[2]:
```
['Take_2_2006Fall_2017Spring_GOES_meteo_combined_14836.csv',
 'Take_2_2006Fall_2017Spring_GOES_meteo_combined_14840.csv',
 'Take_2_2006Fall_2017Spring_GOES_meteo_combined_14841.csv',
 'Take_2_2006Fall_2017Spring_GOES_meteo_combined_94815.csv']
```

In [3]:
```
len(filtered_files)
```

Out[3]:
```
4
```

In [4]:
```
filename = filtered_files[1]

file_path = 'output/'

df_temp_table = pd.read_csv(file_path+filename)

df_temp_table.head(5)
```

```
/tmp/ipykernel_1468242/2204204280.py:5: DtypeWarning:

Columns (19,28,29,30) have mixed types. Specify dtype option on import or s
et low_memory=False.
```

Out[4]:

	Date_UTC	Time_UTC	Date_CST	Time_CST	File_name_for_1D_lake	File_n
0	2006-10-01	00:00	2006-09-30	18:00	goes11.2006.10.01.0000.v01.nc-var1-t0.csv	T_goes11.2006.
1	2006-10-01	01:00	2006-09-30	19:00	goes11.2006.10.01.0100.v01.nc-var1-t0.csv	T_goes11.2006
2	2006-10-01	02:00	2006-09-30	20:00	goes11.2006.10.01.0200.v01.nc-var1-t0.csv	T_goes11.2006.
3	2006-10-01	03:00	2006-09-30	21:00	goes11.2006.10.01.0300.v01.nc-var1-t0.csv	T_goes11.2006.
4	2006-10-01	04:00	2006-09-30	22:00	goes11.2006.10.01.0400.v01.nc-var1-t0.csv	T_goes11.2006.

5 rows × 31 columns

In [5]:
```
station_ID_num = filename.split('_')[-1].split('.')[0]
print(station_ID_num)
```

```
14840
```

In [6]:
```
# df_usable_data = df_temp_table[df_temp_table['data_usable'] == True]
df_usable_data = df_temp_table.copy()

df_usable_data.head(5)
```

Out[6]:

	Date_UTC	Time_UTC	Date_CST	Time_CST	File_name_for_1D_lake	File_n
0	2006-10-01	00:00	2006-09-30	18:00	goes11.2006.10.01.0000.v01.nc-var1-t0.csv	T_goes11.2006.
1	2006-10-01	01:00	2006-09-30	19:00	goes11.2006.10.01.0100.v01.nc-var1-t0.csv	T_goes11.2006
2	2006-10-01	02:00	2006-09-30	20:00	goes11.2006.10.01.0200.v01.nc-var1-t0.csv	T_goes11.2006.
3	2006-10-01	03:00	2006-09-30	21:00	goes11.2006.10.01.0300.v01.nc-var1-t0.csv	T_goes11.2006.
4	2006-10-01	04:00	2006-09-30	22:00	goes11.2006.10.01.0400.v01.nc-var1-t0.csv	T_goes11.2006.

5 rows × 31 columns

In [7]:
```
column_names_list = df_usable_data.columns.tolist()
print(column_names_list)
```

```
['Date_UTC', 'Time_UTC', 'Date_CST', 'Time_CST', 'File_name_for_1D_lake',
'File_name_for_2D_lake', 'Lake_data_1D', 'data_usable', 'cloud_count', 'clo
ud_exist', 'Temp (F)', 'RH (%)', 'Dewpt (F)', 'Wind Spd (mph)', 'Wind Direc
tion (deg)', 'Peak Wind Gust(mph)', 'Low Cloud Ht (ft)', 'Med Cloud Ht (f
t)', 'High Cloud Ht (ft)', 'Visibility (mi)', 'Atm Press (hPa)', 'Sea Lev P
ress (hPa)', 'Altimeter (hPa)', 'Precip (in)', 'Wind Chill (F)', 'Heat Inde
x (F)', 'Unnamed: 18', 'precip_work_zone', 'is_snow_precip', 'is_precip',
'does_snow_24_120']
```

Experiment

Go get the possible continuous range of precipitation!

In [8]:
```
# Print the size of `df_usable_data`
print("The size of `df_usable_data` is:", df_usable_data.shape)
print("The size of `df_temp_table` is:", df_temp_table.shape)
```

```
The size of `df_usable_data` is: (48121, 31)
The size of `df_temp_table` is: (48121, 31)
```



```
ndex of snow precip
# is_snow_precip = df_usable_data['is_snow_precip']
```

```
# # Find the starting row index number of each continuous set of True values
# start_idx_single_lib = is_snow_precip[is_snow_precip.diff() == True].index

# # Print the starting row index number of each continuous set of True value
# print("The starting row index number of each continuous set of True values
```

In [10]:

```
# # This is the true way to go, min of 3 hours of snow
# is_snow_precip = df_usable_data['is_snow_precip']

# # Find continuous sets of at least three True values in the 'is_snow_preci
# sets = []
# set_start = None
# for idx, value in is_snow_precip.items():
#     if value:
#         if set_start is None:
#             set_start = idx
#         if idx == len(is_snow_precip) - 1:
#             sets.append((set_start, idx))
#     else:
#         if set_start is not None:
#             sets.append((set_start, idx - 1))
#             set_start = None

# # Filter out sets with less than three True values and store the starting
# start_idx_lib = [s[0] for s in sets if s[1] - s[0] >= 2]

# # Print the starting row index number of each continuous set of at least t
# print("The starting row index number of each continuous set of at least th
```

In [11]:

```
start_precip_index_lib = []
end_precip_index_lib = []

# Convert the column to a list for easier manipulation
precip_values = df_usable_data['precip_work_zone'].tolist()

start_index = None

# Iterate over the list
for i in range(len(precip_values)):
    # Check if the current value is greater than 0.00
    if precip_values[i] > 0.00:
        # If start_index is None, this is the start of a new sequence
        if start_index is None:
            start_index = i
    else:
        # If start_index is not None, then we've reached the end of a sequen
        if start_index is not None:
            # Check if sequence length is at least 3
            if i - start_index >= 3:
                start_precip_index_lib.append(start_index)
                end_precip_index_lib.append(i - 1)
            # Reset start_index
            start_index = None
```

```
# Check for a sequence that ends at the very last element
if start_index is not None and len(precip_values) - start_index >= 3:
    start_precip_index_lib.append(start_index)
    end_precip_index_lib.append(len(precip_values) - 1)
```

In [12]:
```
print('# of starting index = ', len(start_precip_index_lib))
print('# of ending index = ', len(end_precip_index_lib))
```

```
# of starting index =  470
# of ending index =  470
```

In [13]:
```
duration_of_precip_event = [end - start + 1 for start, end in zip(start_prec
print('# of duration list = ', len(duration_of_precip_event))
```

```
# of duration list =  470
```

In [14]:
```
print(duration_of_precip_event)
```

```
[11, 4, 3, 7, 7, 4, 13, 4, 9, 3, 7, 4, 8, 4, 4, 3, 5, 8, 8, 8, 6, 3, 4, 4,
3, 7, 5, 3, 6, 3, 11, 6, 5, 10, 5, 3, 4, 7, 5, 5, 3, 6, 3, 7, 7, 5, 4, 6,
4, 3, 5, 3, 3, 7, 3, 3, 6, 13, 5, 7, 4, 4, 5, 4, 5, 5, 5, 3, 3, 4, 4, 6, 4,
3, 5, 9, 4, 5, 6, 3, 7, 5, 6, 11, 3, 6, 10, 3, 5, 3, 7, 4, 4, 9, 3, 5, 6,
5, 9, 4, 7, 10, 3, 5, 3, 6, 4, 7, 4, 7, 6, 10, 3, 7, 10, 16, 9, 10, 3, 5,
3, 3, 5, 5, 12, 4, 7, 12, 11, 10, 6, 3, 4, 13, 3, 9, 3, 4, 3, 13, 3, 4, 17,
3, 8, 3, 6, 10, 4, 4, 4, 6, 3, 3, 3, 19, 6, 4, 3, 13, 5, 6, 6, 5, 7, 10, 3,
3, 5, 4, 3, 3, 4, 5, 9, 5, 4, 3, 12, 5, 3, 5, 3, 5, 7, 16, 4, 4, 4, 3, 6,
9, 5, 3, 9, 3, 3, 4, 11, 6, 4, 3, 6, 16, 8, 7, 6, 22, 4, 3, 6, 8, 12, 11,
3, 4, 15, 9, 8, 18, 3, 7, 10, 5, 3, 11, 5, 10, 5, 6, 4, 6, 8, 3, 6, 10, 12,
3, 6, 5, 3, 4, 7, 3, 3, 6, 5, 3, 7, 3, 4, 6, 6, 3, 15, 5, 4, 9, 8, 4, 8, 4,
7, 13, 3, 6, 3, 3, 4, 10, 3, 4, 10, 4, 3, 3, 5, 6, 15, 3, 4, 3, 7, 3, 10,
5, 3, 22, 4, 6, 4, 15, 6, 7, 3, 5, 6, 4, 4, 7, 5, 3, 5, 3, 4, 4, 5, 7, 6,
4, 3, 4, 5, 20, 6, 6, 12, 4, 7, 6, 4, 6, 8, 3, 6, 3, 3, 12, 4, 3, 8, 4, 3,
8, 4, 15, 9, 5, 9, 4, 4, 9, 9, 3, 3, 9, 4, 7, 3, 3, 16, 11, 8, 4, 6, 3, 6,
3, 7, 6, 3, 4, 3, 3, 7, 3, 25, 5, 3, 4, 5, 19, 7, 5, 3, 10, 5, 6, 7, 3, 25,
4, 3, 6, 4, 5, 3, 3, 6, 7, 5, 3, 5, 3, 4, 5, 3, 12, 7, 11, 4, 3, 17, 4, 6,
4, 14, 18, 3, 5, 18, 3, 3, 6, 4, 8, 3, 4, 3, 3, 3, 5, 10, 3, 5, 3, 6, 5, 3,
9, 6, 11, 7, 3, 19, 3, 4, 6, 4, 3, 9, 3, 4, 8, 3, 19, 3, 6, 12, 7, 16, 3,
8, 7, 4, 6, 10, 3, 17, 8, 13, 3, 3, 11, 3, 8, 4, 3, 8, 13]
```

In [15]:
```
# Initialise the two lists
valid_start_index_lib = []
valid_end_index_lib = []

# Iterate through the pairs of start and end indices
for start, end in zip(start_precip_index_lib, end_precip_index_lib):
    # Compute the sum of numbers between start and end (inclusive) in the 'p
    total_precip = df_usable_data['precip_work_zone'].iloc[start:end+1].sum(

    # Check if total_precip is greater or equal to 0.01*(end-start+1)
    if total_precip >= 0.01*(end-start+1):
        # If the condition is met, append the start and end indices to the r
        valid_start_index_lib.append(start)
        valid_end_index_lib.append(end)
```

In [16]:
```
print('# of valid starting index = ', len(valid_start_index_lib))
print('# of valid ending index = ', len(valid_end_index_lib))
```

```
# of valid starting index =  470
# of valid   ending index =  470
```

In [17]:
```
valid_duration_of_precip_event = [end - start + 1 for start, end in zip(vali
print('# of valid duration list = ', len(valid_duration_of_precip_event))
```

```
# of valid duration list =  470
```

In [18]:
```
print(valid_duration_of_precip_event)
```

```
[11, 4, 3, 7, 7, 4, 13, 4, 9, 3, 7, 4, 8, 4, 4, 3, 5, 8, 8, 8, 6, 3, 4, 4,
3, 7, 5, 3, 6, 3, 11, 6, 5, 10, 5, 3, 4, 7, 5, 5, 3, 6, 3, 7, 7, 5, 4, 6,
4, 3, 5, 3, 3, 7, 3, 3, 6, 13, 5, 7, 4, 4, 5, 4, 5, 5, 5, 3, 3, 4, 4, 6, 4,
3, 5, 9, 4, 5, 6, 3, 7, 5, 6, 11, 3, 6, 10, 3, 5, 3, 7, 4, 4, 9, 3, 5, 6,
5, 9, 4, 7, 10, 3, 5, 3, 6, 4, 7, 4, 7, 6, 10, 3, 7, 10, 16, 9, 10, 3, 5,
3, 3, 5, 5, 12, 4, 7, 12, 11, 10, 6, 3, 4, 13, 3, 9, 3, 4, 3, 13, 3, 4, 17,
3, 8, 3, 6, 10, 4, 4, 4, 6, 3, 3, 3, 19, 6, 4, 3, 13, 5, 6, 6, 5, 7, 10, 3,
3, 5, 4, 3, 3, 4, 5, 9, 5, 4, 3, 12, 5, 3, 5, 3, 5, 7, 16, 4, 4, 4, 3, 6,
9, 5, 3, 9, 3, 3, 4, 11, 6, 4, 3, 6, 16, 8, 7, 6, 22, 4, 3, 6, 8, 12, 11,
3, 4, 15, 9, 8, 18, 3, 7, 10, 5, 3, 11, 5, 10, 5, 6, 4, 6, 8, 3, 6, 10, 12,
3, 6, 5, 3, 4, 7, 3, 3, 6, 5, 3, 7, 3, 4, 6, 6, 3, 15, 5, 4, 9, 8, 4, 8, 4,
7, 13, 3, 6, 3, 3, 4, 10, 3, 4, 10, 4, 3, 3, 5, 6, 15, 3, 4, 3, 7, 3, 10,
5, 3, 22, 4, 6, 4, 15, 6, 7, 3, 5, 6, 4, 4, 7, 5, 3, 5, 3, 4, 4, 5, 7, 6,
4, 3, 4, 5, 20, 6, 6, 12, 4, 7, 6, 4, 6, 8, 3, 6, 3, 3, 12, 4, 3, 8, 4, 3,
8, 4, 15, 9, 5, 9, 4, 4, 9, 9, 3, 3, 9, 4, 7, 3, 3, 16, 11, 8, 4, 6, 3, 6,
3, 7, 6, 3, 4, 3, 3, 7, 3, 25, 5, 3, 4, 5, 19, 7, 5, 3, 10, 5, 6, 7, 3, 25,
4, 3, 6, 4, 5, 3, 3, 6, 7, 5, 3, 5, 3, 4, 5, 3, 12, 7, 11, 4, 3, 17, 4, 6,
4, 14, 18, 3, 5, 18, 3, 3, 6, 4, 8, 3, 4, 3, 3, 3, 5, 10, 3, 5, 3, 6, 5, 3,
9, 6, 11, 7, 3, 19, 3, 4, 6, 4, 3, 9, 3, 4, 8, 3, 19, 3, 6, 12, 7, 16, 3,
8, 7, 4, 6, 10, 3, 17, 8, 13, 3, 3, 11, 3, 8, 4, 3, 8, 13]
```

In [19]:
```
# start_idx_lib = []
# num_T_lib = []

# prev_value = False
# counter = 0
# start_idx = -1

# for idx, row in df_usable_data.iterrows():
#     if row['is_precip']:
#         counter += 1
#         if not prev_value:
#             start_idx = idx
#     else:
#         if counter >= 3:
#             start_idx_lib.append(start_idx)
#             num_T_lib.append(counter)
#         counter = 0
#     prev_value = row['is_precip']

# # Check if the loop ended with a valid sequence of True values
# if counter >= 3:
#     start_idx_lib.append(start_idx)
#     num_T_lib.append(counter)

# print("Starting indices:", start_idx_lib)
# print("Number of True values in each sequence:", num_T_lib)
```

In [20]:
```
## Inspection use only

df_usable_data[df_usable_data['precip_work_zone']==0.0001].shape
```

Out[20]:
```
(256, 31)
```

In [21]:
```
## Inspection use only

df_usable_data['precip_work_zone'].value_counts()
```

Out[21]:
```
0.0000    43856
0.0100     1751
0.0200      723
0.0300      368
0.0400      260
0.0001      256
0.0500      188
0.0600      131
0.0700      110
0.0800      103
0.0900       55
0.1100       53
0.1000       43
0.1200       34
0.1300       31
0.1400       24
0.1500       16
0.1800       16
0.1900       15
0.1600       14
0.1700       10
0.2600        7
0.2000        5
0.2300        5
0.2100        4
0.2500        4
0.3200        4
0.2800        3
0.2700        3
0.3100        3
0.2400        3
0.3000        3
0.3300        2
0.3500        2
0.2200        2
0.2900        2
0.3700        1
0.3800        1
0.4500        1
0.4200        1
0.4300        1
0.4000        1
0.4100        1
0.6000        1
0.5200        1
0.5400        1
Name: precip_work_zone, dtype: int64
```

In [22]:
```
start_idx_lib = valid_start_index_lib.copy()

print(len(start_idx_lib))
```

```
470
```

In [23]:
```
num_I_lib = valid_duration_of_precip_event.copy()
print(len(num_I_lib))
```

```
470
```

In [24]:
```
i = 0
while i < len(num_T_lib):
    if num_T_lib[i] > 48:
        del start_idx_lib[i]
        del num_T_lib[i]
    else:
        i += 1
```

In [25]:
```
print(len(start_idx_lib))
```

```
470
```

In [26]:
```
print(df_usable_data.columns)
```

```
Index(['Date_UTC', 'Time_UTC', 'Date_CST', 'Time_CST', 'File_name_for_1D_la
ke',
       'File_name_for_2D_lake', 'Lake_data_1D', 'data_usable', 'cloud_coun
t',
       'cloud_exist', 'Temp (F)', 'RH (%)', 'Dewpt (F)', 'Wind Spd (mph)',
       'Wind Direction (deg)', 'Peak Wind Gust(mph)', 'Low Cloud Ht (ft)',
       'Med Cloud Ht (ft)', 'High Cloud Ht (ft)', 'Visibility (mi)',
       'Atm Press (hPa)', 'Sea Lev Press (hPa)', 'Altimeter (hPa)',
       'Precip (in)', 'Wind Chill (F)', 'Heat Index (F)', 'Unnamed: 18',
       'precip_work_zone', 'is_snow_precip', 'is_precip', 'does_snow_24_12
0'],
      dtype='object')
```

In [27]:
```
cloud_count = df_usable_data['cloud_count']

# Find continuous sets of at least two values larger than 720 in the 'cloud_
cloud_sets = []
cloud_set_start = None
for idx, value in cloud_count.items():
    if value > 720:
        if cloud_set_start is None:
            cloud_set_start = idx
        if idx == len(cloud_count) - 1:
            cloud_sets.append((cloud_set_start, idx))
    else:
        if cloud_set_start is not None:
            cloud_sets.append((cloud_set_start, idx - 1))
            cloud_set_start = None

# Filter out sets with less than two values larger than 720 and store the st
cloud_start_idx_lib = [s[0] for s in cloud_sets if s[1] - s[0] >= 3]

# Print the starting row index number of each continuous set of at least 3 v
print("The starting row index number of each continuous set of at least 3 va
```

```
The starting row index number of each continuous set of at least 3 values l
arger than 720 is: [20, 140, 354, 594, 691, 758, 974, 1263, 1287, 1311, 150
3, 1647, 1695, 1719, 1839, 1887, 1935, 2055, 2103, 2175, 2247, 2271, 2295,
2319, 2343, 2368, 2391, 2415, 2439, 2463, 2610, 4051, 4099, 4146, 4382, 443
6, 4460, 4578, 4844, 4865, 4913, 4982, 5033, 5151, 5178, 5439, 5967, 6207,
6255, 6543, 6615, 6639, 6663, 6783, 6999, 7167, 8706, 8774, 8972, 9019, 918
3, 9498, 9594, 9665, 10815, 10887, 10911, 10935, 10959, 11007, 11031, 1107
9, 11103, 11151, 11343, 11947, 13142, 14054, 14127, 14394, 14630, 14847, 14
871, 15279, 15375, 15399, 15471, 15519, 15543, 15567, 15639, 15735, 16531,
16842, 16868, 17156, 17275, 17299, 17510, 17585, 17611, 17633, 17683, 1770
6, 17873, 17897, 18038, 18302, 18402, 18470, 19311, 19359, 19599, 19647, 19
743, 19767, 19815, 19911, 19935, 20511, 20798, 20872, 20894, 21495, 21590,
21806, 21830, 21854, 21878, 21903, 21951, 21974, 21999, 22023, 22047, 2207
1, 22094, 22166, 22262, 22382, 22406, 22431, 22454, 22574, 22695, 22718, 22
742, 22887, 22983, 23006, 23199, 23342, 23390, 23534, 23559, 23606, 23774,
23871, 23919, 23966, 23991, 24014, 24159, 24305, 24327, 24713, 25025, 2627
0, 26297, 26441, 26464, 30638, 32992, 33040, 35007, 35586, 35610, 37240, 39
375, 39618, 39953, 40170, 40290, 40314, 40432, 40457, 40914, 40960, 41704,
43767, 44587, 44610, 44659, 44732, 44803, 45736, 45759, 45855, 45904, 4597
9, 46312, 47011, 47132, 47152, 47540, 47755]
```

In [28]:

```python
cloud_start_lst = []
precip_start_lst = []

for precip_idx in start_idx_lib:
    cloud_idx = None
    for idx in range(len(cloud_start_idx_lib)):
        if cloud_start_idx_lib[idx] >= precip_idx:
            break
        cloud_idx = cloud_start_idx_lib[idx]
    cloud_start_lst.append(cloud_idx)
    precip_start_lst.append(precip_idx)

# Print the corresponding cloud_start_idx and precip_start_idx values
print("cloud_start_lst:", cloud_start_lst)
print("precip_start_lst:", precip_start_lst)
```

cloud_start_lst: [20, 20, 20, 140, 140, 354, 354, 354, 354, 354, 354, 594,
974, 1311, 1311, 1311, 1311, 1311, 1719, 1935, 1935, 1935, 2103, 2175, 229
5, 2463, 2463, 2610, 2610, 2610, 2610, 2610, 2610, 2610, 2610, 2610, 2610,
4099, 4099, 4099, 4146, 4146, 4460, 4578, 4578, 4578, 4578, 4865, 5178, 543
9, 5439, 5439, 5439, 5439, 5439, 5439, 5967, 5967, 5967, 6255, 6663, 6663,
6783, 6783, 6783, 6999, 6999, 7167, 7167, 7167, 7167, 7167, 7167, 7167, 716
7, 7167, 7167, 7167, 7167, 7167, 7167, 7167, 8706, 8706, 8774, 8774, 8774,
8774, 9019, 9183, 9183, 9183, 9183, 9183, 9183, 9498, 9594, 9665, 9665, 966
5, 9665, 9665, 9665, 9665, 9665, 9665, 9665, 9665, 9665, 9665, 9665, 9665,
9665, 9665, 9665, 9665, 9665, 10815, 11103, 11151, 11151, 11151, 11151, 113
43, 11947, 11947, 11947, 11947, 11947, 11947, 11947, 11947, 11947, 11947, 1
1947, 13142, 13142, 13142, 13142, 13142, 13142, 13142, 13142, 13142, 13142,
13142, 13142, 13142, 13142, 14127, 14127, 14394, 14394, 14394, 14630, 1463
0, 14871, 14871, 14871, 15471, 15735, 15735, 15735, 16531, 16531, 16868, 17
510, 18038, 18038, 18038, 18470, 18470, 18470, 18470, 18470, 18470, 18470,
18470, 18470, 19359, 19647, 19767, 19815, 19935, 19935, 19935, 19935, 1993
5, 19935, 19935, 19935, 19935, 20511, 20511, 20894, 20894, 20894, 20894, 20
894, 20894, 20894, 21590, 21590, 21590, 22166, 22166, 22262, 22262, 22454,
22454, 22574, 22574, 22742, 22742, 22742, 22742, 23199, 23342, 23390, 2360
6, 23871, 23919, 24014, 24014, 24014, 24327, 24327, 24327, 24327, 24327, 24
327, 24327, 24713, 25025, 25025, 25025, 25025, 25025, 25025, 25025, 25025,
25025, 25025, 25025, 25025, 25025, 25025, 26297, 26297, 26464, 26464, 2646
4, 26464, 26464, 26464, 26464, 26464, 26464, 26464, 26464, 26464, 26464, 26
464, 26464, 26464, 26464, 26464, 26464, 26464, 26464, 26464, 26464, 26464,
26464, 26464, 26464, 26464, 26464, 26464, 26464, 26464, 26464, 26464, 2646
4, 26464, 26464, 26464, 26464, 26464, 26464, 26464, 26464, 26464, 26464, 26
464, 26464, 26464, 26464, 26464, 26464, 26464, 26464, 26464, 26464, 26464,
26464, 30638, 30638, 30638, 30638, 30638, 30638, 30638, 30638, 30638, 3063
8, 30638, 30638, 30638, 30638, 30638, 30638, 30638, 30638, 30638, 30638, 30
638, 30638, 30638, 30638, 30638, 30638, 30638, 30638, 30638, 33040, 33040,
33040, 33040, 33040, 33040, 33040, 33040, 33040, 33040, 33040, 33040, 3304
0, 33040, 33040, 33040, 35007, 35007, 35007, 35007, 35007, 35610, 35610, 35
610, 35610, 35610, 35610, 35610, 35610, 35610, 35610, 35610, 35610, 35610,
35610, 35610, 35610, 35610, 35610, 35610, 37240, 37240, 37240, 37240, 3724
0, 37240, 37240, 37240, 37240, 37240, 37240, 37240, 37240, 37240, 37240, 37
240, 37240, 37240, 39375, 39618, 39618, 39618, 39953, 39953, 39953, 40314,
40457, 40457, 40457, 40457, 40960, 40960, 40960, 40960, 40960, 40960, 4096
0, 40960, 41704, 41704, 41704, 41704, 41704, 41704, 41704, 41704, 41704, 41
704, 41704, 41704, 41704, 41704, 41704, 41704, 41704, 41704, 41704, 41704,
41704, 41704, 43767, 43767, 43767, 43767, 43767, 43767, 43767, 43767, 4480
3, 44803, 44803, 44803, 44803, 44803, 44803, 44803, 44803, 44803, 45759, 45
759, 45979, 45979, 45979, 46312, 46312, 46312, 46312, 47152, 47152, 47152,
47540, 47755, 47755, 47755, 47755]
precip_start_lst: [31, 50, 85, 248, 257, 372, 385, 433, 503, 513, 553, 627,
982, 1374, 1412, 1440, 1444, 1472, 1732, 1964, 1981, 2003, 2120, 2199, 230
5, 2475, 2548, 2728, 2899, 3537, 3546, 3561, 3636, 3643, 3765, 3771, 3841,
4117, 4128, 4138, 4279, 4377, 4561, 4600, 4698, 4737, 4787, 4898, 5247, 545
8, 5591, 5616, 5783, 5857, 5887, 5933, 5980, 6084, 6196, 6499, 6733, 6759,
6817, 6879, 6980, 7087, 7127, 7246, 7411, 7440, 7445, 7459, 7517, 7604, 760
8, 7713, 8001, 8075, 8429, 8441, 8514, 8663, 8746, 8754, 8806, 8882, 8930,
8941, 9115, 9220, 9327, 9343, 9350, 9380, 9405, 9555, 9648, 9678, 9688, 977
0, 9794, 9819, 9894, 10070, 10221, 10226, 10293, 10353, 10393, 10416, 1042
9, 10536, 10548, 10563, 10665, 10770, 10790, 10860, 11122, 11175, 11224, 11
261, 11293, 11369, 11965, 12029, 12206, 12334, 12543, 12568, 12608, 12621,
[illegible] 13268, 13316, 13330, 13613, 13645, 1366
1, 13690, 13732, 13743, 13750, 13830, 13843, 14309, 14334, 14451, 14476, 14

```
491, 14664, 14786, 15031, 15167, 15181, 15495, 15899, 16121, 16284, 16633,
16643, 17012, 17522, 18052, 18056, 18109, 18548, 18753, 18758, 18818, 1894
0, 18965, 18974, 18984, 19223, 19471, 19695, 19781, 19840, 19961, 20045, 20
107, 20218, 20343, 20381, 20386, 20454, 20474, 20588, 20629, 20923, 21054,
21070, 21207, 21216, 21323, 21335, 21594, 21600, 21642, 22173, 22203, 2231
0, 22319, 22528, 22542, 22584, 22663, 22786, 22807, 22844, 22848, 23230, 23
389, 23404, 23651, 23883, 23959, 24040, 24084, 24102, 24356, 24384, 24407,
24471, 24549, 24607, 24619, 24728, 25136, 25180, 25373, 25493, 25514, 2555
9, 25583, 25604, 25687, 25784, 25797, 26050, 26062, 26220, 26321, 26334, 26
475, 26519, 26555, 26564, 26571, 26576, 26599, 26660, 26668, 26687, 26700,
26782, 26805, 26856, 27265, 27561, 27745, 27898, 27903, 27972, 28074, 2810
3, 28150, 28183, 28195, 28205, 28213, 28387, 28585, 28706, 28759, 28944, 29
011, 29058, 29111, 29118, 29144, 29151, 29159, 29244, 29269, 29291, 29366,
29384, 29444, 29540, 29586, 29642, 29655, 29723, 29831, 29841, 29847, 3010
7, 30119, 30303, 30308, 30699, 30755, 30980, 31025, 31105, 31182, 31345, 31
354, 31384, 31392, 31497, 31621, 31752, 31773, 31853, 32025, 32158, 32182,
32282, 32292, 32415, 32600, 32636, 32677, 32688, 32712, 32816, 32924, 3294
9, 33072, 33213, 33357, 33410, 33449, 33487, 33587, 33679, 33767, 33944, 33
980, 34056, 34332, 34691, 34753, 34895, 35046, 35085, 35127, 35319, 35377,
35719, 35819, 35865, 35914, 35959, 36037, 36049, 36056, 36100, 36186, 3628
7, 36314, 36373, 36400, 36642, 36982, 37021, 37031, 37221, 37262, 37283, 37
325, 37389, 37692, 37952, 38018, 38197, 38357, 38379, 38388, 38476, 38488,
38631, 38678, 39202, 39308, 39339, 39553, 39842, 39915, 39937, 40018, 4004
2, 40096, 40373, 40717, 40724, 40832, 40837, 41126, 41136, 41309, 41372, 41
445, 41492, 41523, 41536, 41757, 41765, 41784, 41993, 42005, 42035, 42040,
42194, 42309, 42357, 42385, 42616, 42896, 43020, 43027, 43210, 43315, 4337
5, 43554, 43576, 43661, 43727, 43787, 43889, 44036, 44123, 44365, 44371, 44
381, 44536, 44939, 44948, 45038, 45074, 45092, 45166, 45309, 45470, 45590,
45600, 45769, 45829, 46181, 46191, 46224, 46343, 46424, 46560, 46681, 4719
6, 47261, 47378, 47671, 47774, 47931, 47989, 48083]
```

In [29]:
```
print('# of cloud_start_lst = ', len(cloud_start_lst))
print('# of precip_start_lst = ', len(precip_start_lst))
```

```
# of cloud_start_lst =  470
# of precip_start_lst =  470
```

In [30]:
```
df_usable_data[1460:1485]
```

Out[30]:

	Date_UTC	Time_UTC	Date_CST	Time_CST	File_name_for_1D_lake	Fil
1460	2006-11-30	20:00	2006-11-30	14:00	goes11.2006.11.30.2000.v01.nc-var1-t0.csv	T_goes11.2(
1461	2006-11-30	21:00	2006-11-30	15:00	goes11.2006.11.30.2100.v01.nc-var1-t0.csv	T_goes11.2
1462	2006-11-30	22:00	2006-11-30	16:00	goes11.2006.11.30.2200.v01.nc-var1-t0.csv	T_goes11.2(
1463	2006-11-30	23:00	2006-11-30	17:00	goes11.2006.11.30.2300.v01.nc-var1-t0.csv	T_goes11.2(
1464	2006-12-01	00:00	2006-11-30	18:00	goes11.2006.12.01.0000.v01.nc-var1-t0.csv	T_goes11.2(
1465	2006-12-01	01:00	2006-11-30	19:00	goes11.2006.12.01.0100.v01.nc-var1-t0.csv	T_goes11.2
1466	2006-12-01	02:00	2006-11-30	20:00	goes11.2006.12.01.0200.v01.nc-var1-t0.csv	T_goes11.2(
1467	2006-12-01	03:00	2006-11-30	21:00	goes11.2006.12.01.0300.v01.nc-var1-t0.csv	T_goes11.2(
1468	2006-12-01	04:00	2006-11-30	22:00	goes11.2006.12.01.0400.v01.nc-var1-t0.csv	T_goes11.2(
1469	2006-12-01	05:00	2006-11-30	23:00	goes11.2006.12.01.0500.v01.nc-var1-t0.csv	T_goes11.2(
1470	2006-12-01	06:00	2006-12-01	00:00	goes11.2006.12.01.0600.v01.nc-var1-t0.csv	T_goes11.2(
1471	2006-12-01	07:00	2006-12-01	01:00	goes11.2006.12.01.0700.v01.nc-var1-t0.csv	T_goes11.2(

	Date_UTC	Time_UTC	Date_CST	Time_CST	File_name_for_1D_lake	Fil
1472	2006-12-01	08:00	2006-12-01	02:00	goes11.2006.12.01.0800.v01.nc-var1-t0.csv	T_goes11.2(
1473	2006-12-01	09:00	2006-12-01	03:00	goes11.2006.12.01.0900.v01.nc-var1-t0.csv	T_goes11.2(
1474	2006-12-01	10:00	2006-12-01	04:00	goes11.2006.12.01.1000.v01.nc-var1-t0.csv	T_goes11.2
1475	2006-12-01	11:00	2006-12-01	05:00	goes11.2006.12.01.1100.v01.nc-var1-t0.csv	T_goes11.2
1476	2006-12-01	12:00	2006-12-01	06:00	goes11.2006.12.01.1200.v01.nc-var1-t0.csv	T_goes11.2
1477	2006-12-01	13:00	2006-12-01	07:00	goes11.2006.12.01.1300.v01.nc-var1-t0.csv	T_goes11.2
1478	2006-12-01	14:00	2006-12-01	08:00	goes11.2006.12.01.1400.v01.nc-var1-t0.csv	T_goes11.2
1479	2006-12-01	15:00	2006-12-01	09:00	goes11.2006.12.01.1500.v01.nc-var1-t0.csv	T_goes11.2
1480	2006-12-01	16:00	2006-12-01	10:00	goes11.2006.12.01.1600.v01.nc-var1-t0.csv	T_goes11.2
1481	2006-12-01	17:00	2006-12-01	11:00	goes11.2006.12.01.1700.v01.nc-var1-t0.csv	T_goes11.2
1482	2006-12-01	18:00	2006-12-01	12:00	goes11.2006.12.01.1800.v01.nc-var1-t0.csv	T_goes11.2

	Date_UTC	Time_UTC	Date_CST	Time_CST	File_name_for_1D_lake	Fil
1483	2006-12-01	19:00	2006-12-01	13:00	goes11.2006.12.01.1900.v01.nc-var1-t0.csv	T_goes11.2
1484	2006-12-01	20:00	2006-12-01	14:00	goes11.2006.12.01.2000.v01.nc-var1-t0.csv	T_goes11.2(

25 rows × 31 columns

In [31]:
```
# for precip_start, cloud_start in zip(precip_start_lst, cloud_start_lst):
# #     print(precip_start)
#     difference = precip_start - cloud_start
#     print(difference)
```

In [32]:
```
if cloud_start_lst[0] is None:
    print('Is None.')
    cloud_start_lst[0] = 0

if cloud_start_lst[1] is None:
    print('Is None.')
    cloud_start_lst[1] = 0



duration_between_cloud_snow = [precip_start_lst[i] - cloud_start_lst[i] + 1
print(duration_between_cloud_snow)
```

```
[12, 31, 66, 109, 118, 19, 32, 80, 150, 160, 200, 34, 9, 64, 102, 130, 134,
162, 14, 30, 47, 69, 18, 25, 11, 13, 86, 119, 290, 928, 937, 952, 1027, 103
4, 1156, 1162, 1232, 19, 30, 40, 134, 232, 102, 23, 121, 160, 210, 34, 70,
20, 153, 178, 345, 419, 449, 495, 14, 118, 230, 245, 71, 97, 35, 97, 198, 8
9, 129, 80, 245, 274, 279, 293, 351, 438, 442, 547, 835, 909, 1263, 1275, 1
348, 1497, 41, 49, 33, 109, 157, 168, 97, 38, 145, 161, 168, 198, 223, 58,
55, 14, 24, 106, 130, 155, 230, 406, 557, 562, 629, 689, 729, 752, 765, 87
2, 884, 899, 1001, 1106, 1126, 46, 20, 25, 74, 111, 143, 27, 19, 83, 260, 3
88, 597, 622, 662, 675, 1023, 1111, 1179, 14, 38, 127, 175, 189, 472, 504,
520, 549, 591, 602, 609, 689, 702, 183, 208, 58, 83, 98, 35, 157, 161, 297,
311, 25, 165, 387, 550, 103, 113, 145, 13, 15, 19, 72, 79, 284, 289, 349, 4
71, 496, 505, 515, 754, 113, 49, 15, 26, 27, 111, 173, 284, 409, 447, 452,
520, 540, 78, 119, 30, 161, 177, 314, 323, 430, 442, 5, 11, 53, 8, 38, 49,
58, 75, 89, 11, 90, 45, 66, 103, 107, 32, 48, 15, 46, 13, 41, 27, 71, 89, 3
0, 58, 81, 145, 223, 281, 293, 16, 112, 156, 349, 469, 490, 535, 559, 580,
663, 760, 773, 1026, 1038, 1196, 25, 38, 12, 56, 92, 101, 108, 113, 136, 19
7, 205, 224, 237, 319, 342, 393, 802, 1098, 1282, 1435, 1440, 1509, 1611, 1
640, 1687, 1720, 1732, 1742, 1750, 1924, 2122, 2243, 2296, 2481, 2548, 259
5, 2648, 2655, 2681, 2688, 2696, 2781, 2806, 2828, 2903, 2921, 2981, 3077,
3123, 3179, 3192, 3260, 3368, 3378, 3384, 3644, 3656, 3840, 3845, 62, 118,
343, 388, 468, 545, 708, 717, 747, 755, 860, 984, 1115, 1136, 1216, 1388, 1
521, 1545, 1645, 1655, 1778, 1963, 1999, 2040, 2051, 2075, 2179, 2287, 231
2, 33, 174, 318, 371, 410, 448, 548, 640, 728, 905, 941, 1017, 1293, 1652,
1714, 1856, 40, 79, 121, 313, 371, 110, 210, 256, 305, 350, 428, 440, 447,
491, 577, 678, 705, 764, 791, 1033, 1373, 1412, 1422, 1612, 23, 44, 86, 15
0, 453, 713, 779, 958, 1118, 1140, 1149, 1237, 1249, 1392, 1439, 1963, 206
9, 2100, 179, 225, 298, 320, 66, 90, 144, 60, 261, 268, 376, 381, 167, 177,
350, 413, 486, 533, 564, 577, 54, 62, 81, 290, 302, 332, 337, 491, 606, 65
4, 682, 913, 1193, 1317, 1324, 1507, 1612, 1672, 1851, 1873, 1958, 2024, 2
1, 123, 270, 357, 599, 605, 615, 770, 137, 146, 236, 272, 290, 364, 507, 66
8, 788, 798, 11, 71, 203, 213, 246, 32, 113, 249, 370, 45, 110, 227, 132, 2
0, 177, 235, 329]
```

In [33]:
```
print('# of element in cloud_start_lst:', len(cloud_start_lst))
print('# of element in precip_start_lst:', len(precip_start_lst))
```

```
# of element in cloud_start_lst: 470
# of element in precip_start_lst: 470
```

In [34]:
```
# Get the corresponding cloud_start
df_cloud_start_rows = df_usable_data.iloc[cloud_start_lst]

# Get the corresponding precip start
df_precip_start_rows = df_usable_data.iloc[precip_start_lst]
```

In [35]:
```
df_cloud_start_rows = df_cloud_start_rows.reset_index(drop=True)

df_cloud_start_rows.head(5)
```

Out[35]:

	Date_UTC	Time_UTC	Date_CST	Time_CST	File_name_for_1D_lake	File_ı
0	2006-10-01	20:00	2006-10-01	14:00	goes11.2006.10.01.2000.v01.nc-var1-t0.csv	T_goes11.2006
1	2006-10-01	20:00	2006-10-01	14:00	goes11.2006.10.01.2000.v01.nc-var1-t0.csv	T_goes11.2006
2	2006-10-01	20:00	2006-10-01	14:00	goes11.2006.10.01.2000.v01.nc-var1-t0.csv	T_goes11.2006
3	2006-10-06	20:00	2006-10-06	14:00	goes11.2006.10.06.2000.v01.nc-var1-t0.csv	T_goes11.2006
4	2006-10-06	20:00	2006-10-06	14:00	goes11.2006.10.06.2000.v01.nc-var1-t0.csv	T_goes11.2006

5 rows × 31 columns

In [36]:
```
df_precip_start_rows = df_precip_start_rows.reset_index(drop=True)


df_precip_start_rows.head(5)
```

Out[36]:

	Date_UTC	Time_UTC	Date_CST	Time_CST	File_name_for_1D_lake	File_r
0	2006-10-02	07:00	2006-10-02	01:00	goes11.2006.10.02.0700.v01.nc-var1-t0.csv	T_goes11.2006
1	2006-10-03	02:00	2006-10-02	20:00	goes11.2006.10.03.0200.v01.nc-var1-t0.csv	T_goes11.2006.
2	2006-10-04	13:00	2006-10-04	07:00	goes11.2006.10.04.1300.v01.nc-var1-t0.csv	T_goes11.2006
3	2006-10-11	08:00	2006-10-11	02:00	None	
4	2006-10-11	17:00	2006-10-11	11:00	goes11.2006.10.11.1700.v01.nc-var1-t0.csv	T_goes11.2006

5 rows × 31 columns

In [37]:
```
df_cloud_start_rows.shape
```

Out[37]: (470, 31)

In [38]:
```
df_precip_start_rows.shape
```

Out[38]: (470, 31)

In [39]:
```
df_cloud_start_rows = df_cloud_start_rows.rename(columns={col: 'Cloud_Start_
df_cloud_start_rows.head(5)
```

Out[39]:

	Cloud_Start_Date_UTC	Cloud_Start_Time_UTC	Cloud_Start_Date_CST	Cloud_Start_Time_
0	2006-10-01	20:00	2006-10-01	
1	2006-10-01	20:00	2006-10-01	
2	2006-10-01	20:00	2006-10-01	
3	2006-10-06	20:00	2006-10-06	
4	2006-10-06	20:00	2006-10-06	

5 rows × 31 columns

In [40]:
```
df_precip_start_rows = df_precip_start_rows.rename(columns={col: 'Precip_Sta
df_precip_start_rows.head(5)
```

Out[40]:

	Precip_Start_Date_UTC	Precip_Start_Time_UTC	Precip_Start_Date_CST	Precip_Start_Tim
0	2006-10-02	07:00	2006-10-02	
1	2006-10-03	02:00	2006-10-02	
2	2006-10-04	13:00	2006-10-04	
3	2006-10-11	08:00	2006-10-11	
4	2006-10-11	17:00	2006-10-11	

5 rows × 31 columns

In [41]:
```
df_combined_timing = pd.DataFrame({
    'Cloud_Start_Date_UTC': df_cloud_start_rows['Cloud_Start_Date_UTC'],
    'Cloud_Start_Time_UTC': df_cloud_start_rows['Cloud_Start_Time_UTC'],
    'Cloud_Start_Date_CST': df_cloud_start_rows['Cloud_Start_Date_CST'],
    'Cloud_Start_Time_CST': df_cloud_start_rows['Cloud_Start_Time_CST'],
    'Precip_Start_Date_UTC': df_precip_start_rows['Precip_Start_Date_UTC'],
    'Precip_Start_Time_UTC': df_precip_start_rows['Precip_Start_Time_UTC'],
    'Precip_Start_Date_CST': df_precip_start_rows['Precip_Start_Date_CST'],
    'Precip_Start_Time_CST': df_precip_start_rows['Precip_Start_Time_CST']
})
df_combined_timing.head(5)
```

Out[41]:

	Cloud_Start_Date_UTC	Cloud_Start_Time_UTC	Cloud_Start_Date_CST	Cloud_Start_Time_
0	2006-10-01	20:00	2006-10-01	
1	2006-10-01	20:00	2006-10-01	
2	2006-10-01	20:00	2006-10-01	
3	2006-10-06	20:00	2006-10-06	
4	2006-10-06	20:00	2006-10-06	

In [42]:
```
df_combined_timing['Duration_of_Snow'] = num_T_lib
df_combined_timing.head(5)
```

Out[42]:

	Cloud_Start_Date_UTC	Cloud_Start_Time_UTC	Cloud_Start_Date_CST	Cloud_Start_Time_
0	2006-10-01	20:00	2006-10-01	
1	2006-10-01	20:00	2006-10-01	
2	2006-10-01	20:00	2006-10-01	
3	2006-10-06	20:00	2006-10-06	
4	2006-10-06	20:00	2006-10-06	

In [43]:
```
df_combined_timing['Duration_of_cloud_formation'] = duration_between_cloud_s
# duration_between_cloud_snow
```

In [44]:
```
df_combined_timing = df_combined_timing[df_combined_timing['Duration_of_clou
```

In [45]:
```
# df_combined_timing = df_combined_timing[df_combined_timing['Duration_of_cl
```

In [46]:
```
df_combined_timing['Precip_year'] = df_combined_timing['Precip_Start_Date_CS
df_combined_timing.head(5)
```

Out[46]:

	Cloud_Start_Date_UTC	Cloud_Start_Time_UTC	Cloud_Start_Date_CST	Cloud_Start_Time_
0	2006-10-01	20:00	2006-10-01	
1	2006-10-01	20:00	2006-10-01	
2	2006-10-01	20:00	2006-10-01	
5	2006-10-15	18:00	2006-10-15	
6	2006-10-15	18:00	2006-10-15	

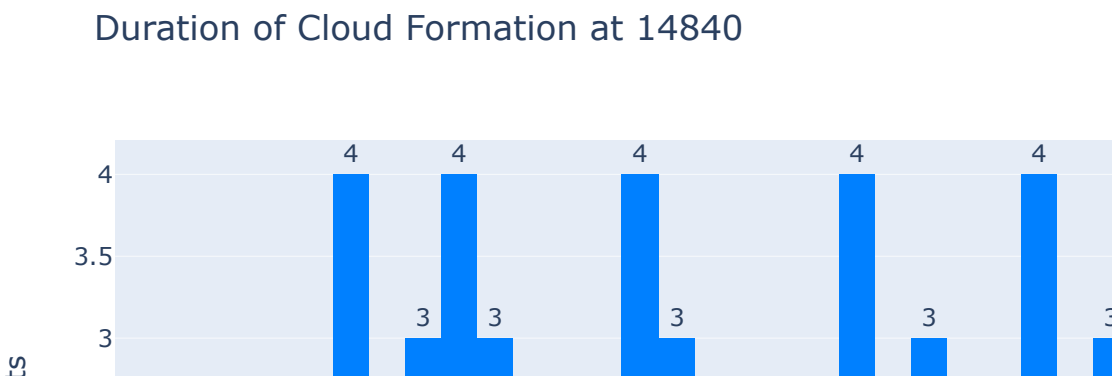
In [47]:
```
df_combined_timing.shape
```

Out[47]:
```
(93, 11)
```

In [48]:
```
import plotly.express as px

fig = px.histogram(df_combined_timing, x="Duration_of_cloud_formation", nbir
cloud formation", yaxis_title="# of prec
fig.update_traces(marker_color="rgb(0, 128, 255)", texttemplate='%{y}', text
```

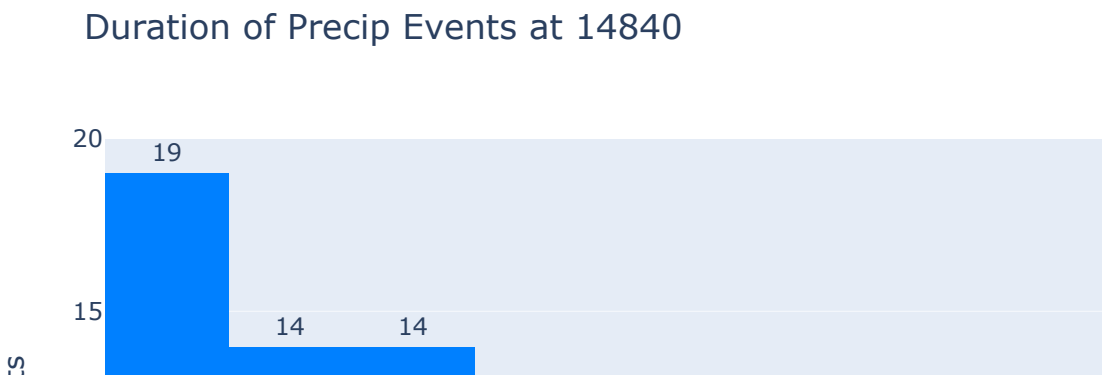
```
fig.update_layout(
   xaxis = dict(
      tickmode = 'linear',
      tick0 = 1,
      dtick = 1
   )
)
fig.show()
fig.write_image('hist_'+station_ID_num+'_duration_of_cloud_formation.png')
```



In [49]:
```
import plotly.express as px

fig = px.histogram(df_combined_timing, x="Duration_of_Snow", nbins = 36, tit
fig.update_layout(xaxis_title="# of hours of precip", yaxis_title="# of Prec
fig.update_traces(marker_color="rgb(0, 128, 255)", texttemplate='%{y}', text
fig.update_layout(
   xaxis = dict(
      tickmode = 'linear',
      tick0 = 1,
      dtick = 1
   )
)
```

```
fig.show()
fig.write_image('output/histo_plot_precip/hist_'+station_ID_num+'.png')
```



In [50]:
```
df_combined_timing = df_combined_timing.rename(columns={'Duration_of_Snow':
df_combined_timing = df_combined_timing.rename(columns={'Duration_of_cloud_f
```

In [51]:
```
output_dir = 'output/histo_precip/'
output_csv_name = station_ID_num+'_Precip_Events.csv'

output_file_path = os.path.join(output_dir, output_csv_name)

df_combined_timing.to_csv(output_file_path, index=False)
```

In [52]:
```
output_csv_name
```

Out[52]:
```
'14840_Precip_Events.csv'
```

In [53]:
```
column_names = df_combined_timing.columns.tolist()
column_names
```

Out[53]:
```
['Cloud_Start_Date_UTC',
 'Cloud_Start_Time_UTC',
 'Cloud_Start_Date_CST',
 'Cloud_Start_Time_CST',
 'Precip_Start_Date_UTC',
 'Precip_Start_Time_UTC',
 'Precip_Start_Date_CST',
 'Precip_Start_Time_CST',
 'Duration_of_All_Precip',
 'Duration_of_cloud_formation_all_precip',
 'Precip_year']
```

In [ ]: