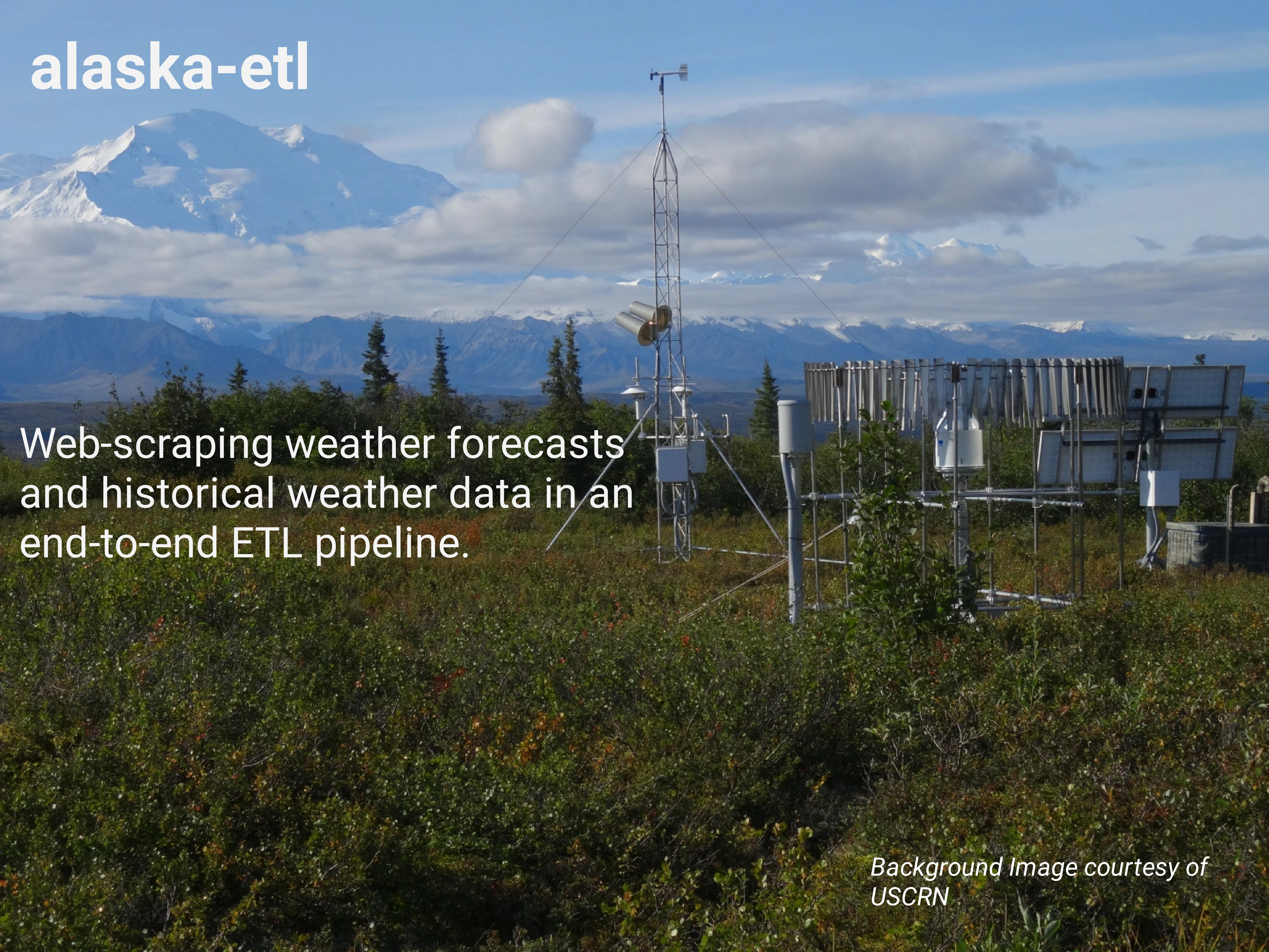
alaska-etl



Web-scraping weather forecasts and historical weather data in an end-to-end ETL pipeline.

Background Image courtesy of USCRN

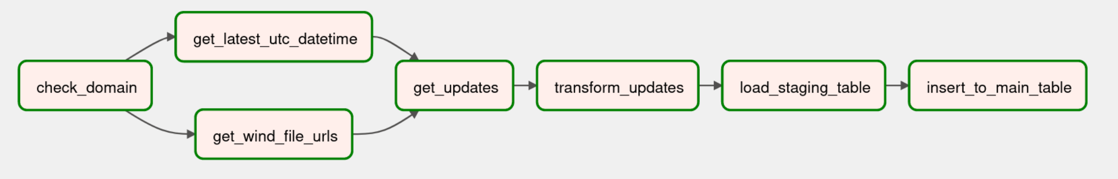
Project Overview (github)

"How surprising is next week's weather forecast?"
"How accurate have the forecasts been?"

Historic data — USCRN (hourly & subhourly)
Forecasts — NWS (hourly)

Technologies used:

Airflow



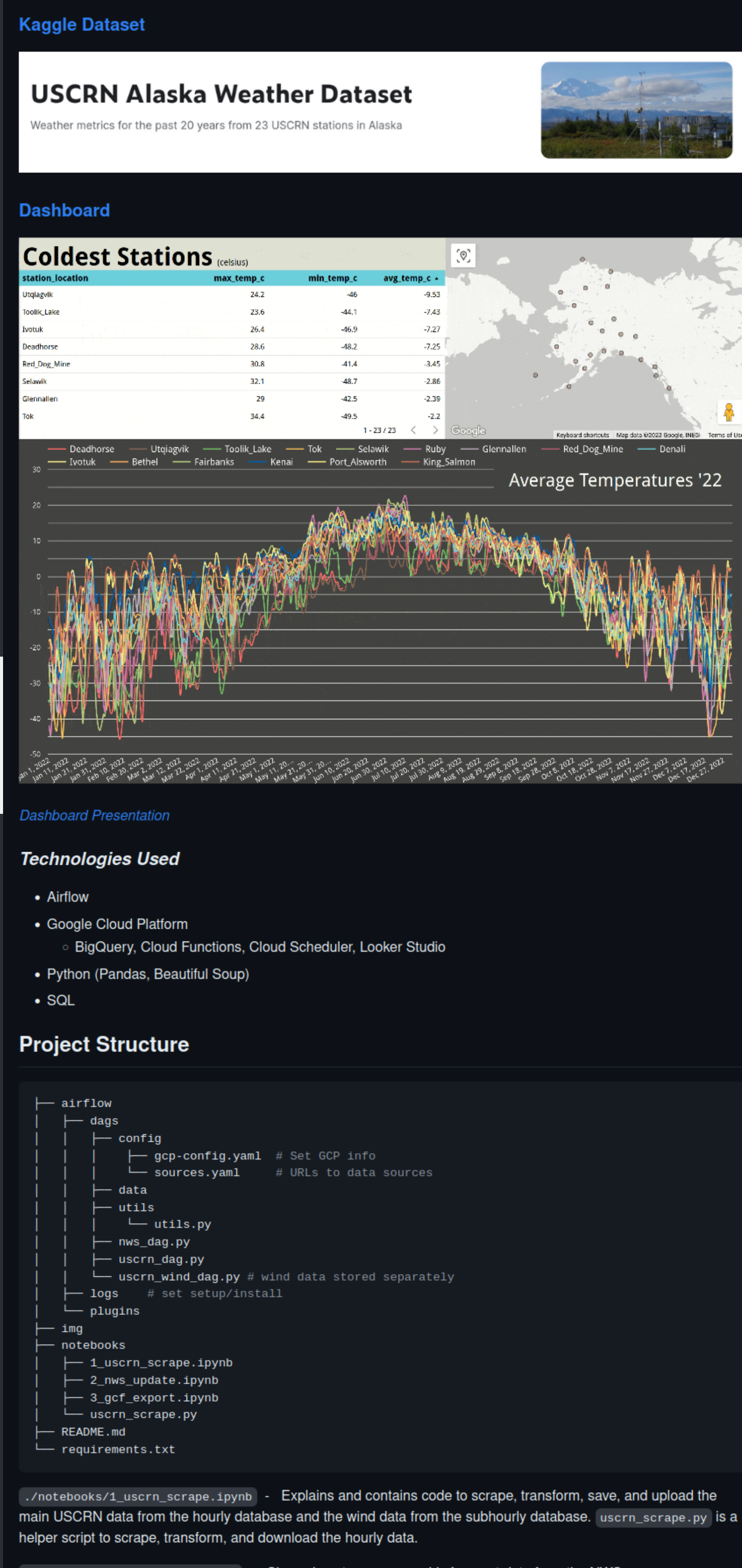
Docker

Python

pandas

BeautifulSoup

Google Cloud Platform
- Google Cloud Functions
- Google Cloud Scheduler
- BigQuery
- Looker Studio



Kaggle Dataset

USCRN Alaska Weather Dataset

Weather metrics for the past 20 years from 23 USCRN stations in Alaska

Dashboard

Coldest Stations (celsius)

station_location	max_temp_c	min_temp_c	avg_temp_c ▲
Utqiagvik	24.2	-46	-9.53
Toolik_Lake	23.6	-44.1	-7.43
Ivotuk	26.4	-46.9	-7.27
Deadhorse	28.6	-48.2	-7.25
Red_Dog_Mine	30.8	-41.4	-3.45
Selawik	32.1	-48.7	-2.86
Glennallen	29	-42.5	-2.39
Tok	34.4	-49.5	-2.2

1 - 23 / 23

Dashboard Presentation

Technologies Used

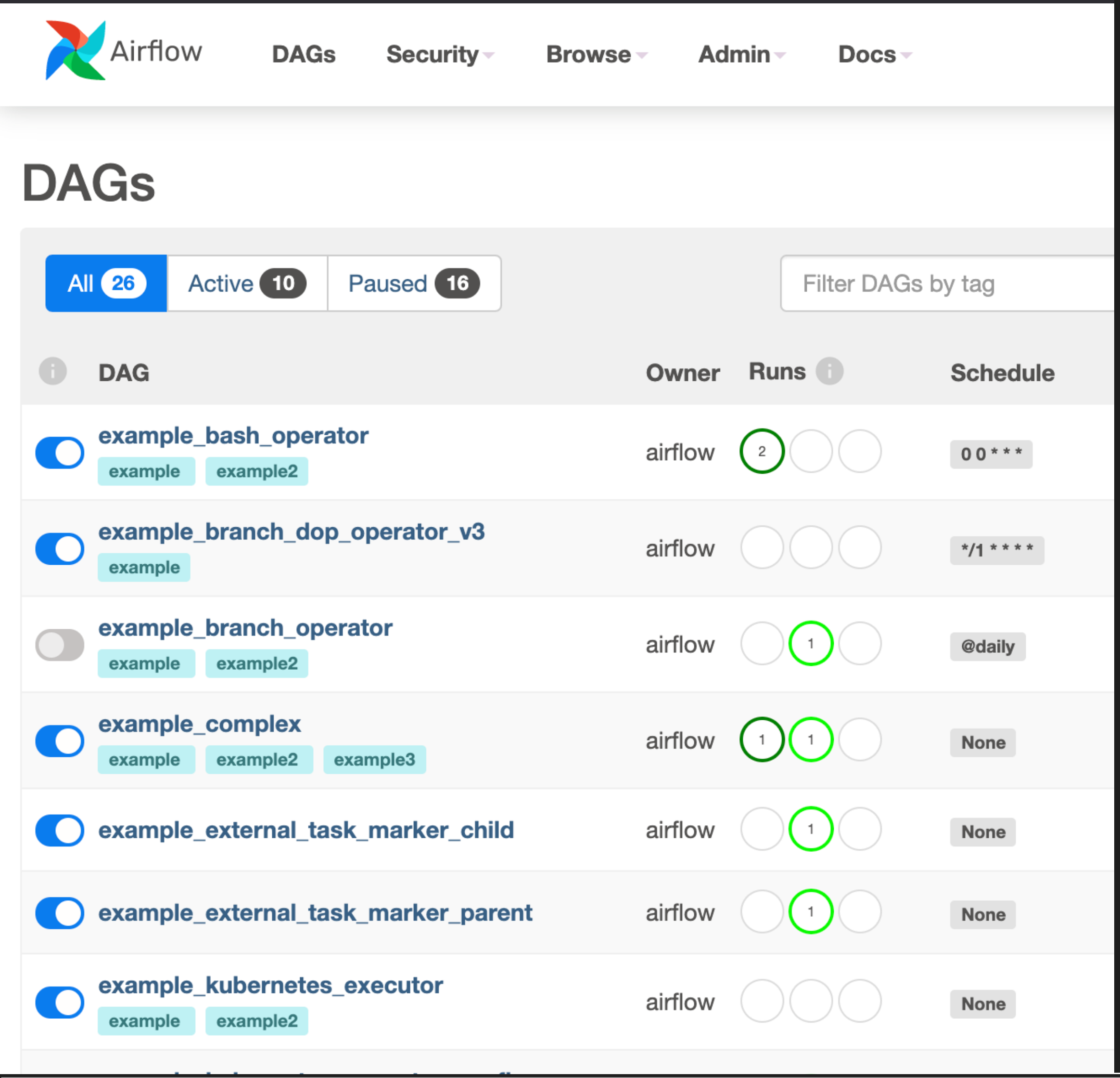
- Airflow
- Google Cloud Platform
  - BigQuery, Cloud Functions, Cloud Scheduler, Looker Studio
- Python (Pandas, Beautiful Soup)
- SQL

Project Structure

```
├── airflow
│   ├── dags
│   │   ├── config
│   │   │   ├── gcp-config.yaml   # Set GCP info
│   │   │   └── sources.yaml      # URLs to data sources
│   │   ├── data
│   │   ├── utils
│   │   │   └── utils.py
│   │   ├── nws_dag.py
│   │   ├── uscrn_dag.py
│   │   └── uscrn_wind_dag.py # wind data stored separately
│   ├── logs    # set setup/install
│   └── plugins
├── img
├── notebooks
│   ├── 1_uscrn_scrape.ipynb
│   ├── 2_nws_update.ipynb
│   ├── 3_gcf_export.ipynb
│   └── uscrn_scrape.py
├── README.md
└── requirements.txt
```

`./notebooks/1_uscrn_scrape.ipynb` - Explains and contains code to scrape, transform, save, and upload the main USCRN data from the hourly database and the wind data from the subhourly database. `uscrn_scrape.py` is a helper script to scrape, transform, and download the hourly data.

Airflow

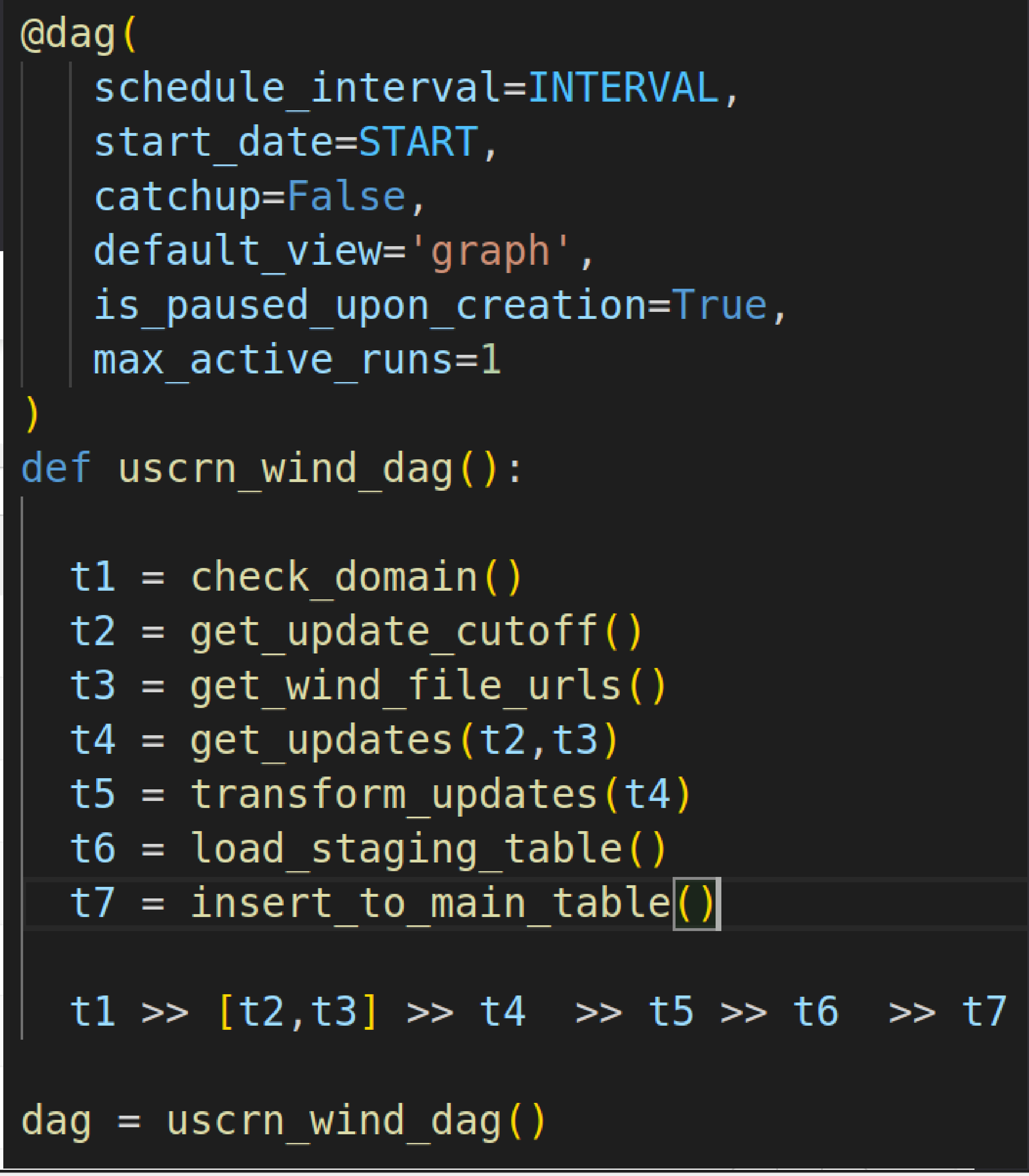


DAGs

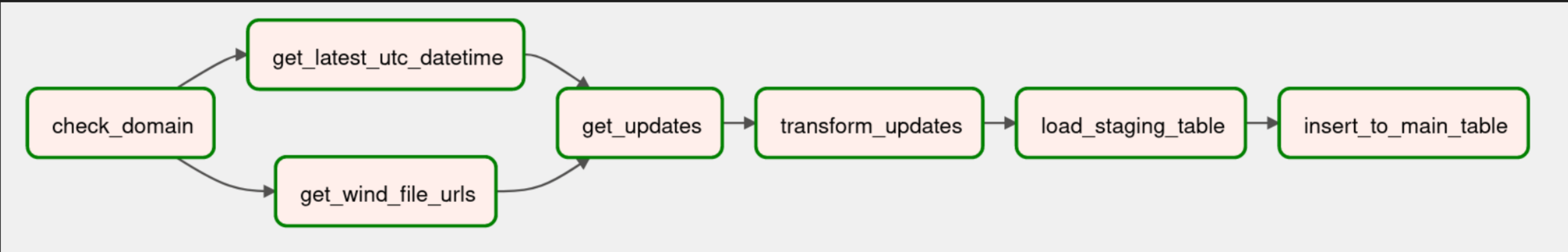
All 26 | Active 10 | Paused 16

Filter DAGs by tag

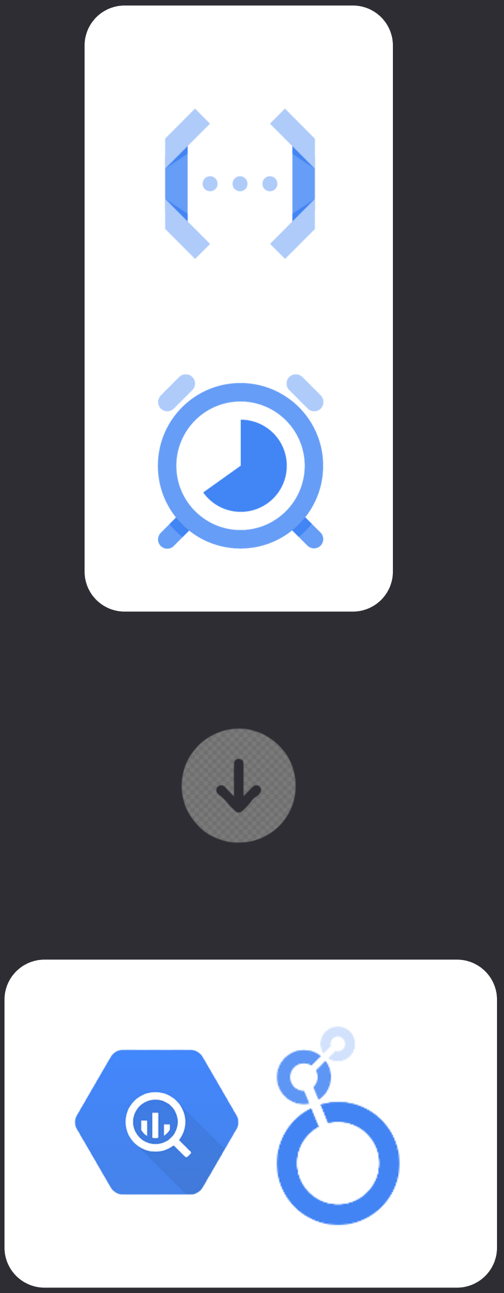
DAG	Owner	Runs	Schedule
example_bash_operator example example2	airflow	2 ○ ○	0 0 * * *
example_branch_dop_operator_v3 example	airflow	○ ○ ○	*/1 * * * *
example_branch_operator example example2	airflow	○ 1 ○	@daily
example_complex example example2 example3	airflow	1 1 ○	None
example_external_task_marker_child	airflow	○ 1 ○	None
example_external_task_marker_parent	airflow	○ 1 ○	None
example_kubernetes_executor example example2	airflow	○ ○ ○	None

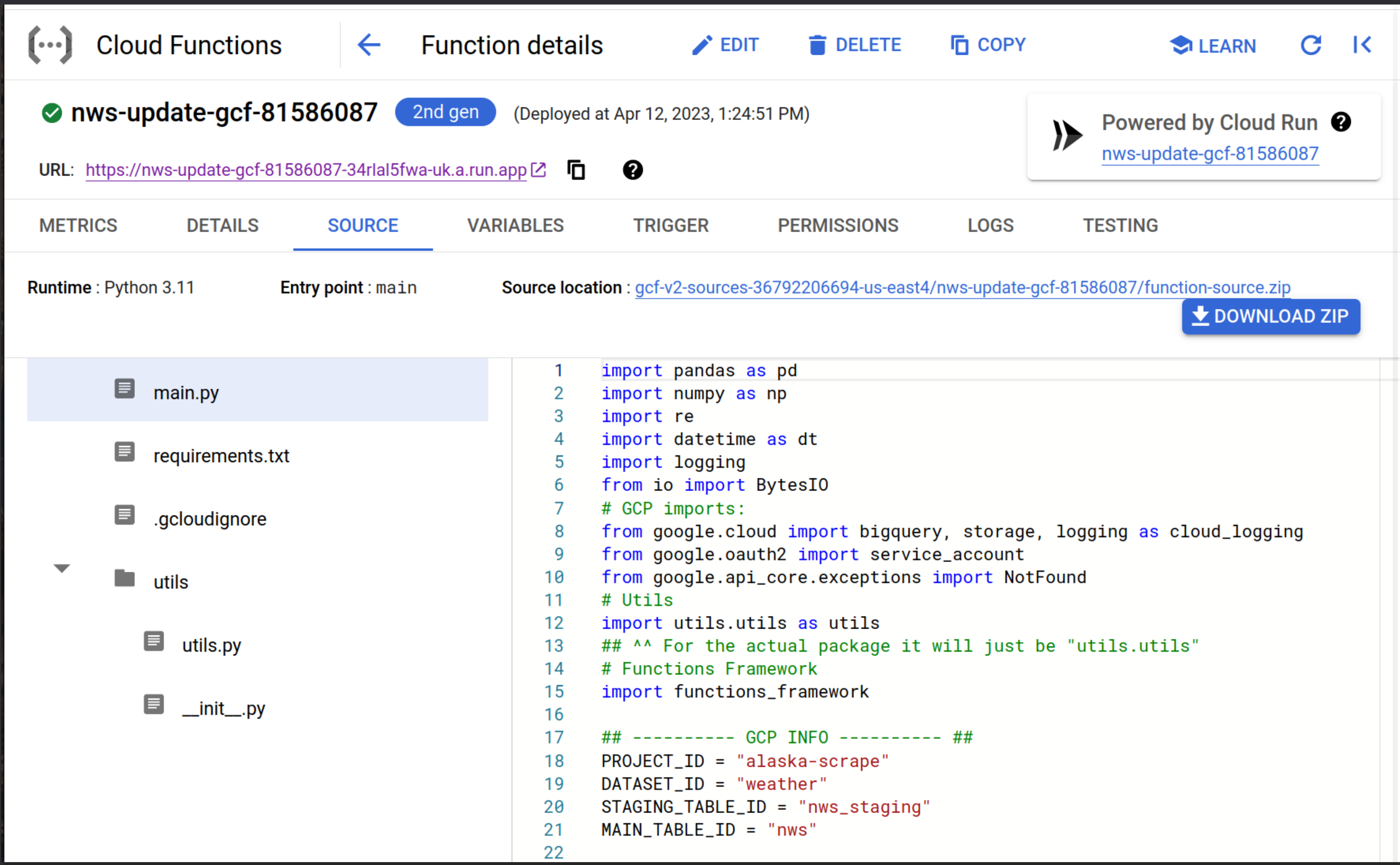


```
@dag(
    schedule_interval=INTERVAL,
    start_date=START,
    catchup=False,
    default_view='graph',
    is_paused_upon_creation=True,
    max_active_runs=1
)
def uscrn_wind_dag():

  t1 = check_domain()
  t2 = get_update_cutoff()
  t3 = get_wind_file_urls()
  t4 = get_updates(t2,t3)
  t5 = transform_updates(t4)
  t6 = load_staging_table()
  t7 = insert_to_main_table()

  t1 >> [t2,t3] >> t4  >> t5 >> t6  >> t7

dag = uscrn_wind_dag()
```



Cloud Functions + Cloud Scheduler





nws-update-gcf-81586087 2nd gen (Deployed at Apr 12, 2023, 1:24:51 PM)

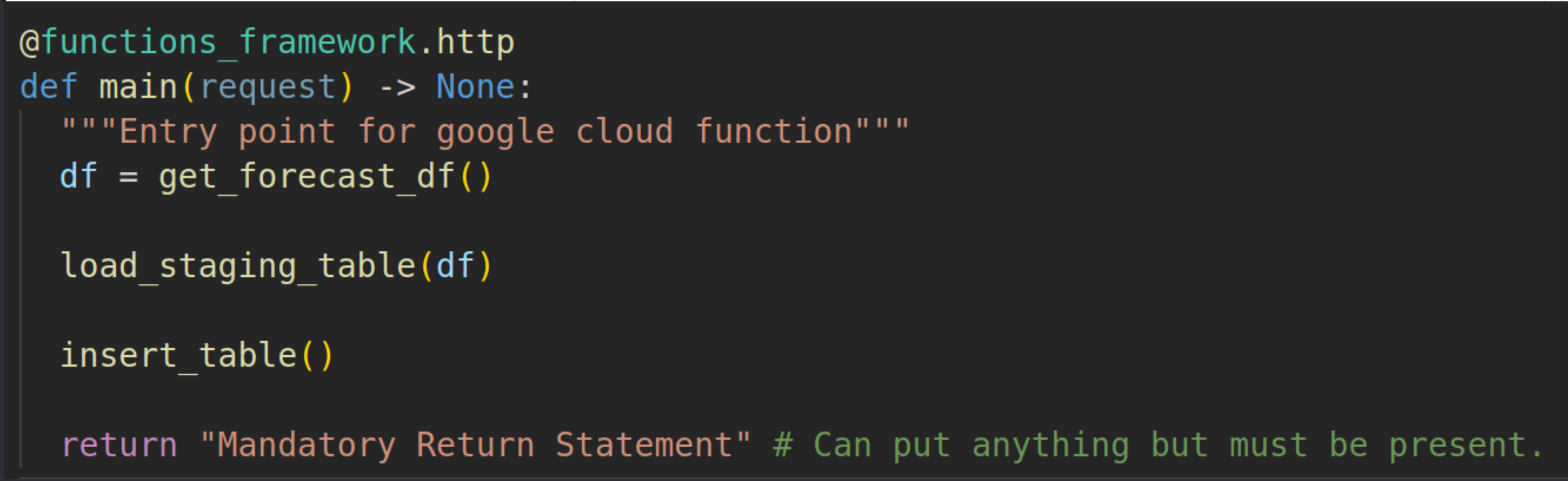
URL: https://nws-update-gcf-81586087-34rlal5fwa-uk.a.run.app

Powered by Cloud Run
nws-update-gcf-81586087

Runtime : Python 3.11 **Entry point** : main **Source location** : gcf-v2-sources-36792206694-us-east4/nws-update-gcf-81586087/function-source.zip

- main.py
- requirements.txt
- .gcloudignore
- utils
  - utils.py
  - __init__.py

```python
import pandas as pd
import numpy as np
import re
import datetime as dt
import logging
from io import BytesIO
# GCP imports:
from google.cloud import bigquery, storage, logging as cloud_logging
from google.oauth2 import service_account
from google.api_core.exceptions import NotFound
# Utils
import utils.utils as utils
## ^^ For the actual package it will just be "utils.utils"
# Functions Framework
import functions_framework

## ---------- GCP INFO ---------- ##
PROJECT_ID = "alaska-scrape"
DATASET_ID = "weather"
STAGING_TABLE_ID = "nws_staging"
MAIN_TABLE_ID = "nws"

```



```python
@functions_framework.http
def main(request) -> None:
  """Entry point for google cloud function"""
  df = get_forecast_df()

  load_staging_table(df)

  insert_table()

  return "Mandatory Return Statement" # Can put anything but must be present.
```

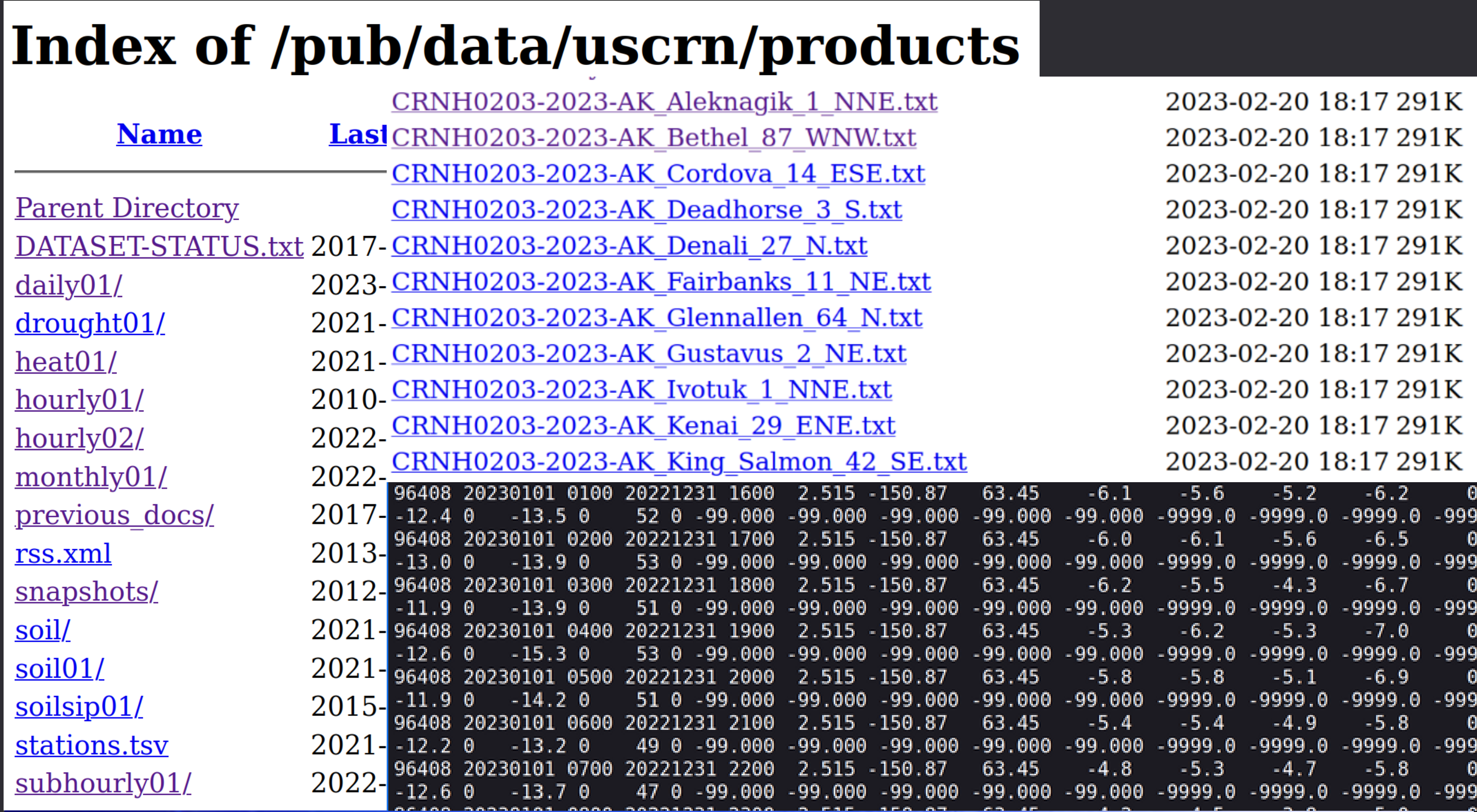
Data Sources

USCRN

- 20 years of data
- 22 AK weather stations

NWS

- Weekly hourly forecast
- Matched to stations

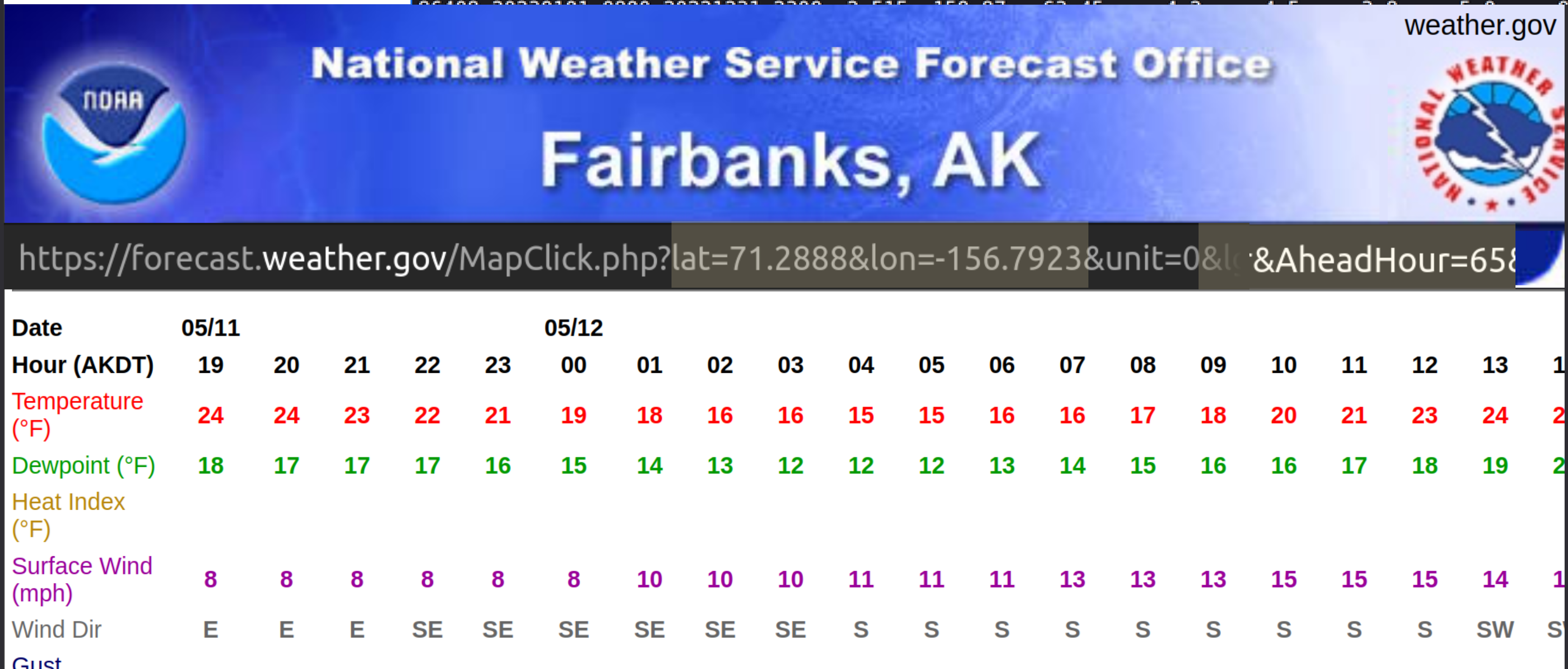


Index of /pub/data/uscrn/products

Name	Last
Parent Directory	
DATASET-STATUS.txt	2017-
daily01/	2023-
drought01/	2021-
heat01/	2021-
hourly01/	2010-
hourly02/	2022-
monthly01/	2022-
previous_docs/	2017-
rss.xml	2013-
snapshots/	2012-
soil/	2021-
soil01/	2021-
soilsip01/	2015-
stations.tsv	2021-
subhourly01/	2022-

CRNH0203-2023-AK_Aleknagik_1_NNE.txt	2023-02-20 18:17 291K
CRNH0203-2023-AK_Bethel_87_WNW.txt	2023-02-20 18:17 291K
CRNH0203-2023-AK_Cordova_14_ESE.txt	2023-02-20 18:17 291K
CRNH0203-2023-AK_Deadhorse_3_S.txt	2023-02-20 18:17 291K
CRNH0203-2023-AK_Denali_27_N.txt	2023-02-20 18:17 291K
CRNH0203-2023-AK_Fairbanks_11_NE.txt	2023-02-20 18:17 291K
CRNH0203-2023-AK_Glennallen_64_N.txt	2023-02-20 18:17 291K
CRNH0203-2023-AK_Gustavus_2_NE.txt	2023-02-20 18:17 291K
CRNH0203-2023-AK_Ivotuk_1_NNE.txt	2023-02-20 18:17 291K
CRNH0203-2023-AK_Kenai_29_ENE.txt	2023-02-20 18:17 291K
CRNH0203-2023-AK_King_Salmon_42_SE.txt	2023-02-20 18:17 291K

```
96408 20230101 0100 20221231 1600  2.515 -150.87   63.45    -6.1     -5.6     -5.2     -6.2     0
-12.4 0   -13.5 0    52 0 -99.000 -99.000 -99.000 -99.000 -99.000 -9999.0 -9999.0 -9999.0 -999
96408 20230101 0200 20221231 1700  2.515 -150.87   63.45    -6.0     -6.1     -5.6     -6.5     0
-13.0 0   -13.9 0    53 0 -99.000 -99.000 -99.000 -99.000 -99.000 -9999.0 -9999.0 -9999.0 -999
96408 20230101 0300 20221231 1800  2.515 -150.87   63.45    -6.2     -5.5     -4.3     -6.7     0
-11.9 0   -13.9 0    51 0 -99.000 -99.000 -99.000 -99.000 -99.000 -9999.0 -9999.0 -9999.0 -999
96408 20230101 0400 20221231 1900  2.515 -150.87   63.45    -5.3     -6.2     -5.3     -7.0     0
-12.6 0   -15.3 0    53 0 -99.000 -99.000 -99.000 -99.000 -99.000 -9999.0 -9999.0 -9999.0 -999
96408 20230101 0500 20221231 2000  2.515 -150.87   63.45    -5.8     -5.8     -5.1     -6.9     0
-11.9 0   -14.2 0    51 0 -99.000 -99.000 -99.000 -99.000 -99.000 -9999.0 -9999.0 -9999.0 -999
96408 20230101 0600 20221231 2100  2.515 -150.87   63.45    -5.4     -5.4     -4.9     -5.8     0
-12.2 0   -13.2 0    49 0 -99.000 -99.000 -99.000 -99.000 -99.000 -9999.0 -9999.0 -9999.0 -999
96408 20230101 0700 20221231 2200  2.515 -150.87   63.45    -4.8     -5.3     -4.7     -5.8     0
-12.6 0   -13.7 0    47 0 -99.000 -99.000 -99.000 -99.000 -99.000 -9999.0 -9999.0 -9999.0 -999
```

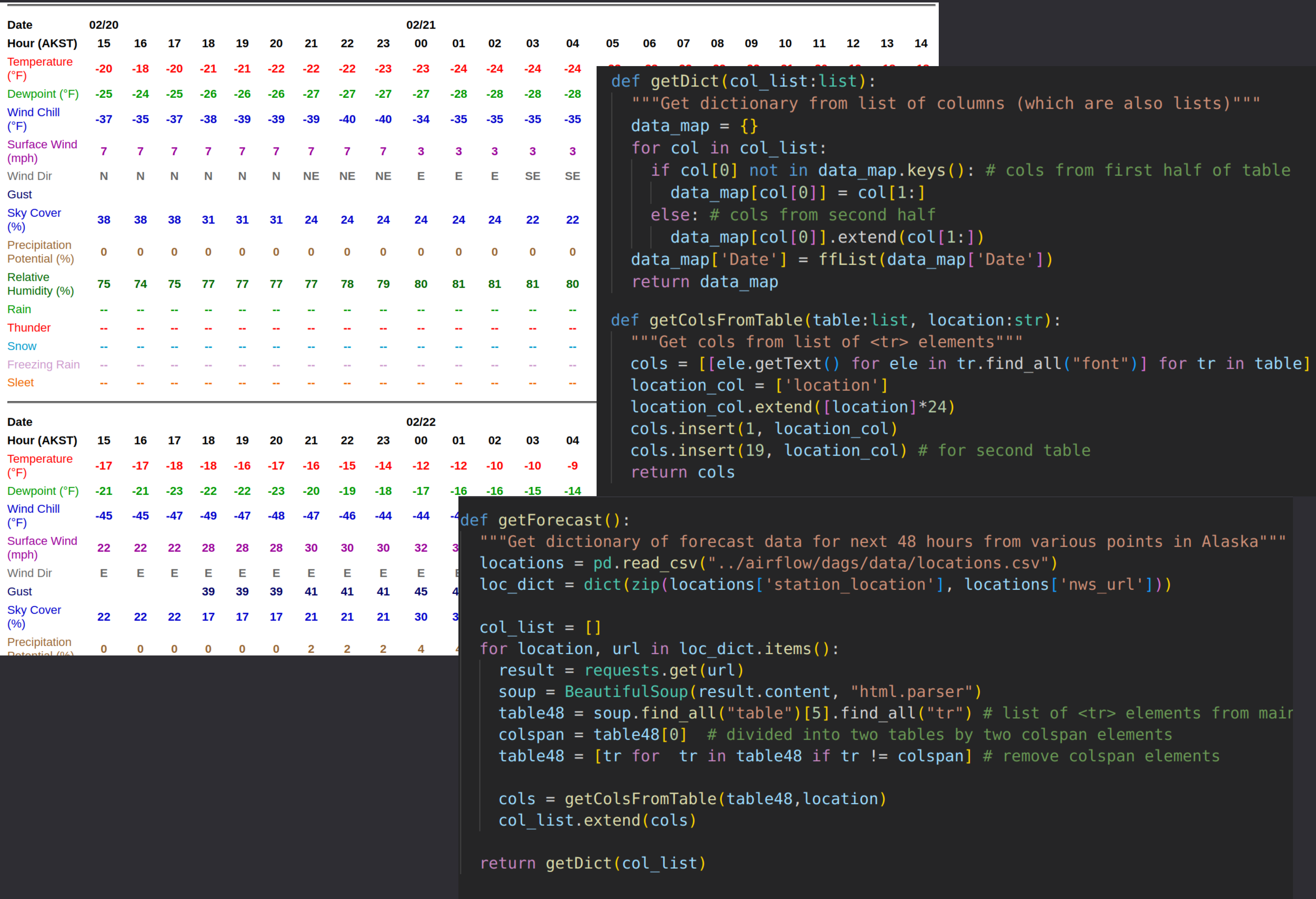


Fairbanks, AK

https://forecast.weather.gov/MapClick.php?lat=71.2888&lon=-156.7923&unit=0&l &AheadHour=658

Date	05/11					05/12														
Hour (AKDT)	19	20	21	22	23	00	01	02	03	04	05	06	07	08	09	10	11	12	13	1
Temperature (°F)	24	24	23	22	21	19	18	16	16	15	15	16	16	17	18	20	21	23	24	2
Dewpoint (°F)	18	17	17	17	16	15	14	13	12	12	12	13	14	15	16	16	17	18	19	2
Heat Index (°F)																				
Surface Wind (mph)	8	8	8	8	8	8	10	10	10	11	11	11	13	13	13	15	15	15	14	1
Wind Dir	E	E	E	SE	SE	SE	SE	SE	SE	S	S	S	S	S	S	S	S	S	SW	S
Gust																				

NWS - Forecast Data



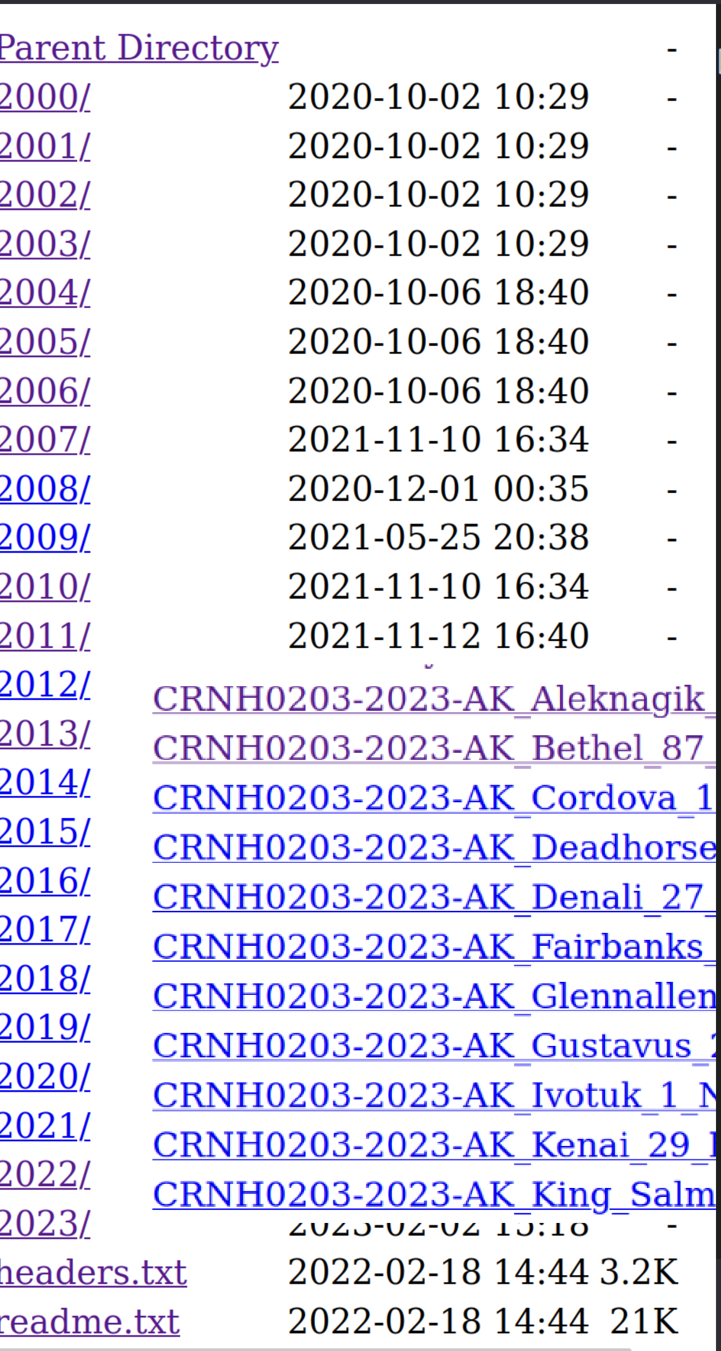
Date	02/20									02/21														
Hour (AKST)	15	16	17	18	19	20	21	22	23	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14
Temperature (°F)	-20	-18	-20	-21	-21	-22	-22	-22	-23	-23	-24	-24	-24	-24										
Dewpoint (°F)	-25	-24	-25	-26	-26	-26	-27	-27	-27	-27	-28	-28	-28	-28										
Wind Chill (°F)	-37	-35	-37	-38	-39	-39	-39	-40	-40	-34	-35	-35	-35	-35										
Surface Wind (mph)	7	7	7	7	7	7	7	7	7	3	3	3	3	3										
Wind Dir	N	N	N	N	N	N	NE	NE	NE	E	E	E	SE	SE										
Gust																								
Sky Cover (%)	38	38	38	31	31	31	24	24	24	24	24	24	22	22										
Precipitation Potential (%)	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
Relative Humidity (%)	75	74	75	77	77	77	77	78	79	80	81	81	81	80										
Rain	--	--	--	--	--	--	--	--	--	--	--	--	--	--										
Thunder	--	--	--	--	--	--	--	--	--	--	--	--	--	--										
Snow	--	--	--	--	--	--	--	--	--	--	--	--	--	--										
Freezing Rain	--	--	--	--	--	--	--	--	--	--	--	--	--	--										
Sleet	--	--	--	--	--	--	--	--	--	--	--	--	--	--										

Date										02/22				
Hour (AKST)	15	16	17	18	19	20	21	22	23	00	01	02	03	04
Temperature (°F)	-17	-17	-18	-18	-16	-17	-16	-15	-14	-12	-12	-10	-10	-9
Dewpoint (°F)	-21	-21	-23	-22	-22	-23	-20	-19	-18	-17	-16	-16	-15	-14
Wind Chill (°F)	-45	-45	-47	-49	-47	-48	-47	-46	-44	-44				
Surface Wind (mph)	22	22	22	28	28	28	30	30	30	32				
Wind Dir	E	E	E	E	E	E	E	E	E	E				
Gust				39	39	39	41	41	41	45				
Sky Cover (%)	22	22	22	17	17	17	21	21	21	30				
Precipitation Potential (%)	0	0	0	0	0	0	2	2	2	4				

```python
def getDict(col_list:list):
    """Get dictionary from list of columns (which are also lists)"""
    data_map = {}
    for col in col_list:
        if col[0] not in data_map.keys(): # cols from first half of table
            data_map[col[0]] = col[1:]
        else: # cols from second half
            data_map[col[0]].extend(col[1:])
    data_map['Date'] = ffList(data_map['Date'])
    return data_map

def getColsFromTable(table:list, location:str):
    """Get cols from list of <tr> elements"""
    cols = [[ele.getText() for ele in tr.find_all("font")] for tr in table]
    location_col = ['location']
    location_col.extend([location]*24)
    cols.insert(1, location_col)
    cols.insert(19, location_col) # for second table
    return cols
```

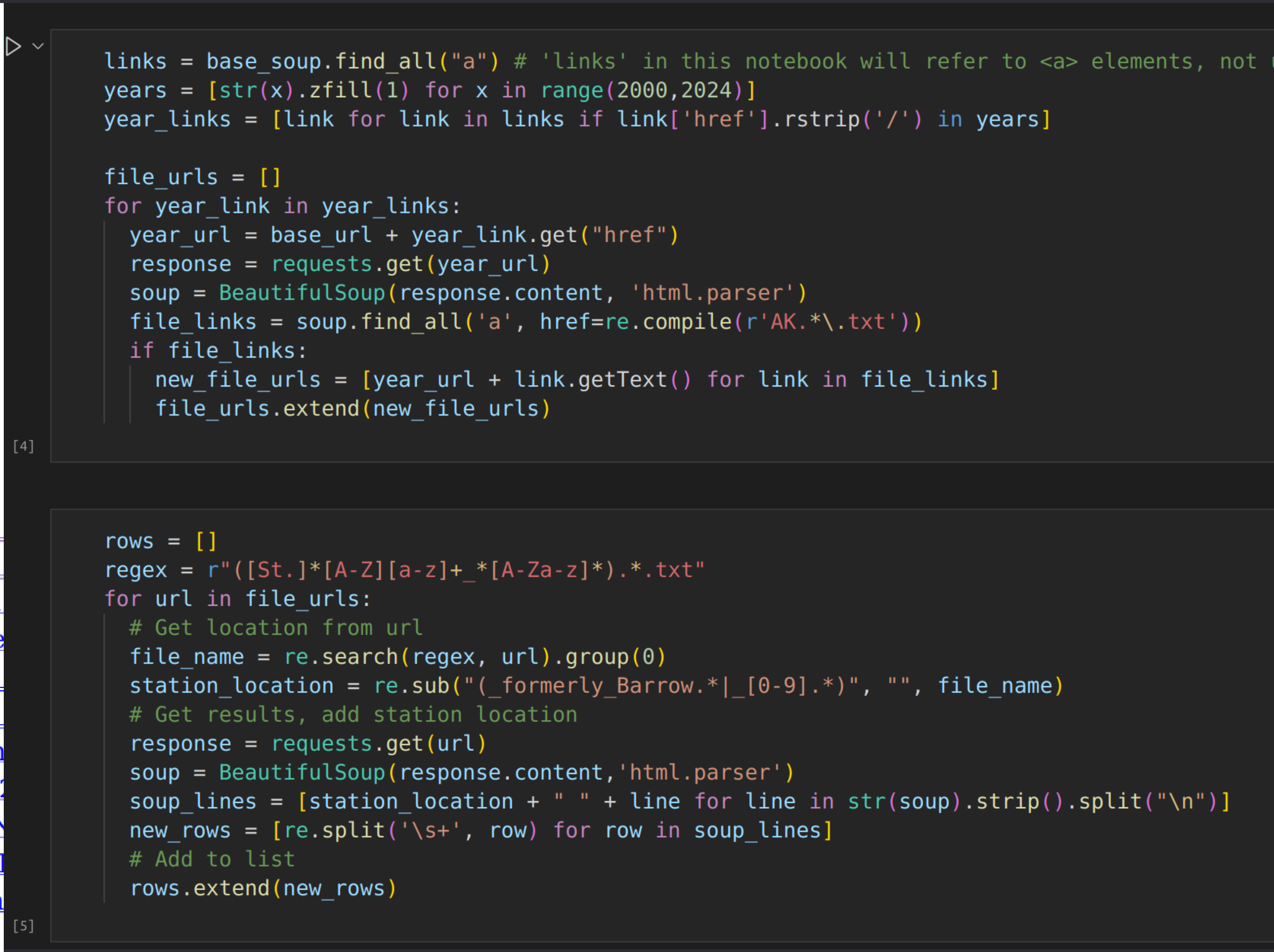
```python
def getForecast():
    """Get dictionary of forecast data for next 48 hours from various points in Alaska"""
    locations = pd.read_csv("../airflow/dags/data/locations.csv")
    loc_dict = dict(zip(locations['station_location'], locations['nws_url']))

    col_list = []
    for location, url in loc_dict.items():
        result = requests.get(url)
        soup = BeautifulSoup(result.content, "html.parser")
        table48 = soup.find_all("table")[5].find_all("tr") # list of <tr> elements from mair
        colspan = table48[0]  # divided into two tables by two colspan elements
        table48 = [tr for  tr in table48 if tr != colspan] # remove colspan elements

        cols = getColsFromTable(table48,location)
        col_list.extend(cols)

    return getDict(col_list)
```

USCRN - Hourly Data



Parent Directory		-
2000/	2020-10-02 10:29	-
2001/	2020-10-02 10:29	-
2002/	2020-10-02 10:29	-
2003/	2020-10-02 10:29	-
2004/	2020-10-06 18:40	-
2005/	2020-10-06 18:40	-
2006/	2020-10-06 18:40	-
2007/	2021-11-10 16:34	-
2008/	2020-12-01 00:35	-
2009/	2021-05-25 20:38	-
2010/	2021-11-10 16:34	-
2011/	2021-11-12 16:40	-
2012/		
2013/		
2014/		
2015/		
2016/		
2017/		
2018/		
2019/		
2020/		
2021/		
2022/		
2023/	2023-02-02 15:18	-
headers.txt	2022-02-18 14:44	3.2K
readme.txt	2022-02-18 14:44	21K

CRNH0203-2023-AK_Aleknagik_
CRNH0203-2023-AK_Bethel_87_
CRNH0203-2023-AK_Cordova_1
CRNH0203-2023-AK_Deadhorse
CRNH0203-2023-AK_Denali_27_
CRNH0203-2023-AK_Fairbanks_
CRNH0203-2023-AK_Glennallen
CRNH0203-2023-AK_Gustavus_
CRNH0203-2023-AK_Ivotuk_1_N
CRNH0203-2023-AK_Kenai_29_
CRNH0203-2023-AK_King_Salm



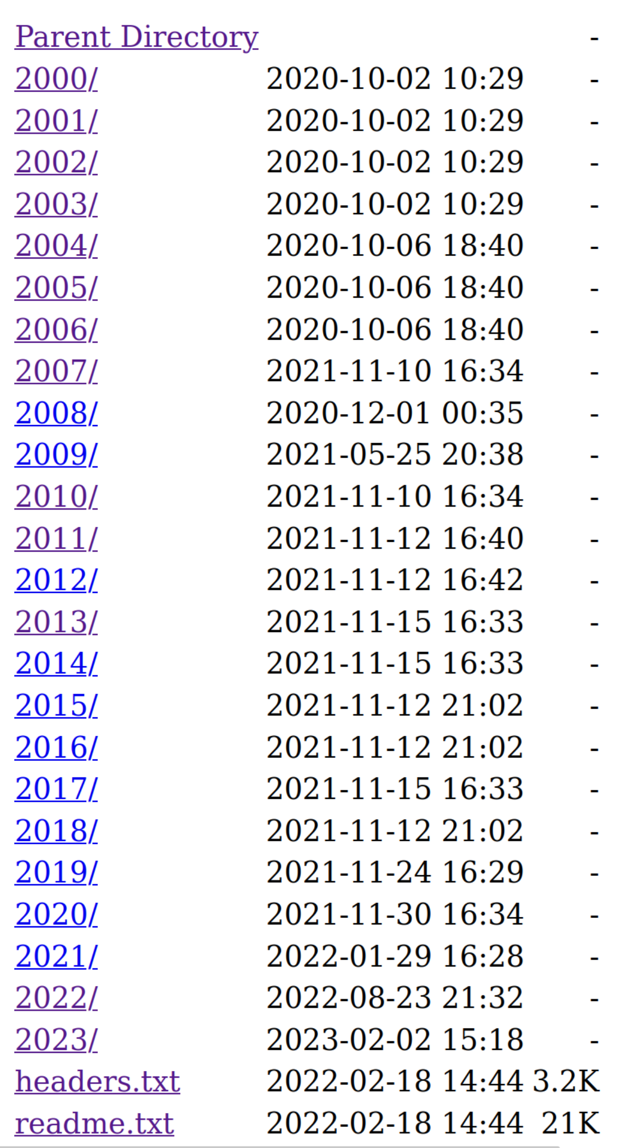
```
links = base_soup.find_all("a") # 'links' in this notebook will refer to <a> elements, not
years = [str(x).zfill(1) for x in range(2000,2024)]
year_links = [link for link in links if link['href'].rstrip('/') in years]

file_urls = []
for year_link in year_links:
  year_url = base_url + year_link.get("href")
  response = requests.get(year_url)
  soup = BeautifulSoup(response.content, 'html.parser')
  file_links = soup.find_all('a', href=re.compile(r'AK.*\.txt'))
  if file_links:
    new_file_urls = [year_url + link.getText() for link in file_links]
    file_urls.extend(new_file_urls)
```

[4]

```
rows = []
regex = r"([St.]*[A-Z][a-z]+_*[A-Za-z]*).*.txt"
for url in file_urls:
  # Get location from url
  file_name = re.search(regex, url).group(0)
  station_location = re.sub("(_formerly_Barrow.*|_[0-9].*)", "", file_name)
  # Get results, add station location
  response = requests.get(url)
  soup = BeautifulSoup(response.content,'html.parser')
  soup_lines = [station_location + " " + line for line in str(soup).strip().split("\n")]
  new_rows = [re.split('\s+', row) for row in soup_lines]
  # Add to list
  rows.extend(new_rows)
```

[5]

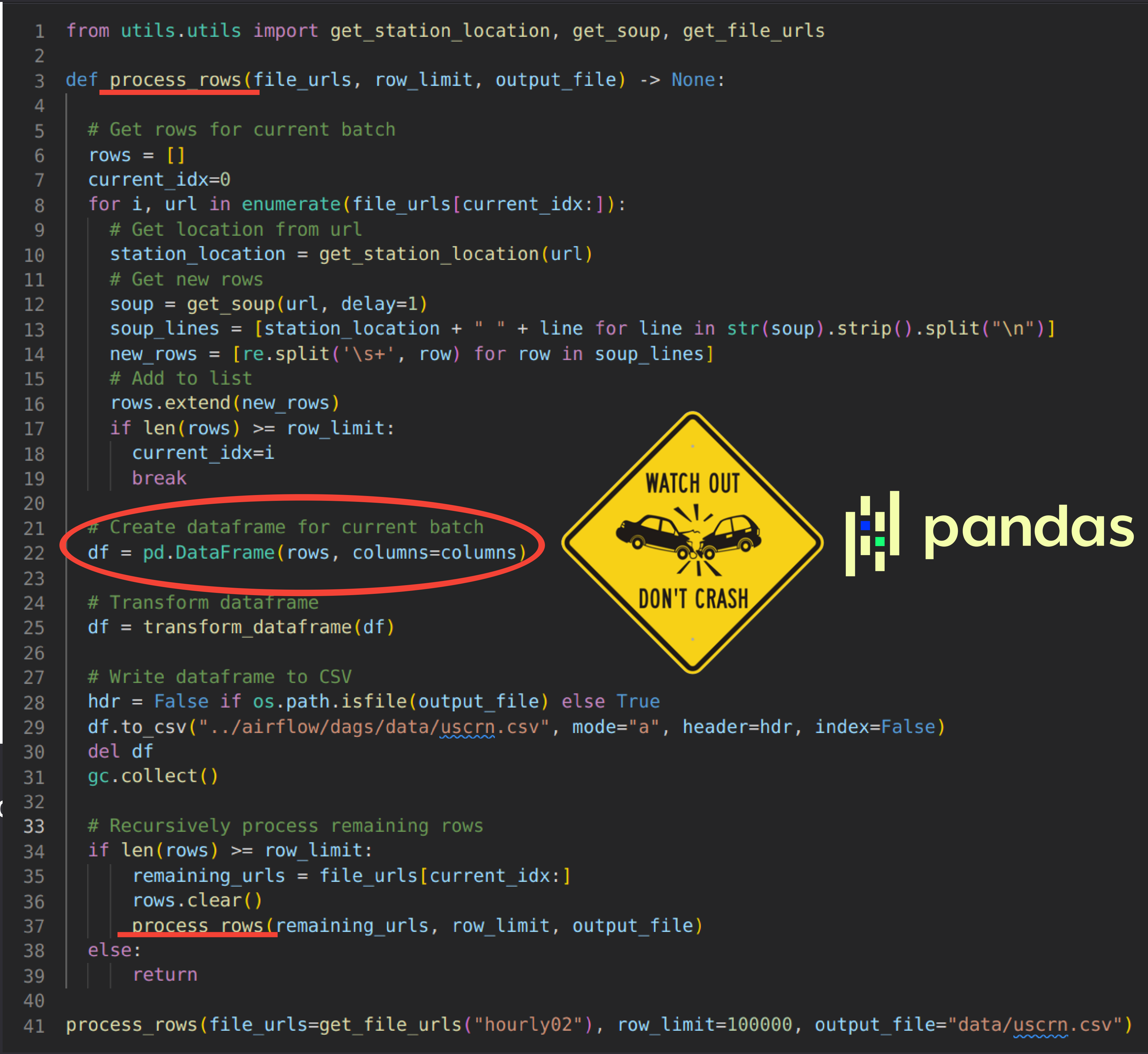
USCRN - Hourly Data



Parent Directory		-
2000/	2020-10-02 10:29	-
2001/	2020-10-02 10:29	-
2002/	2020-10-02 10:29	-
2003/	2020-10-02 10:29	-
2004/	2020-10-06 18:40	-
2005/	2020-10-06 18:40	-
2006/	2020-10-06 18:40	-
2007/	2021-11-10 16:34	-
2008/	2020-12-01 00:35	-
2009/	2021-05-25 20:38	-
2010/	2021-11-10 16:34	-
2011/	2021-11-12 16:40	-
2012/	2021-11-12 16:42	-
2013/	2021-11-15 16:33	-
2014/	2021-11-15 16:33	-
2015/	2021-11-12 21:02	-
2016/	2021-11-12 21:02	-
2017/	2021-11-15 16:33	-
2018/	2021-11-12 21:02	-
2019/	2021-11-24 16:29	-
2020/	2021-11-30 16:34	-
2021/	2022-01-29 16:28	-
2022/	2022-08-23 21:32	-
2023/	2023-02-02 15:18	-
headers.txt	2022-02-18 14:44	3.2K
readme.txt	2022-02-18 14:44	21K

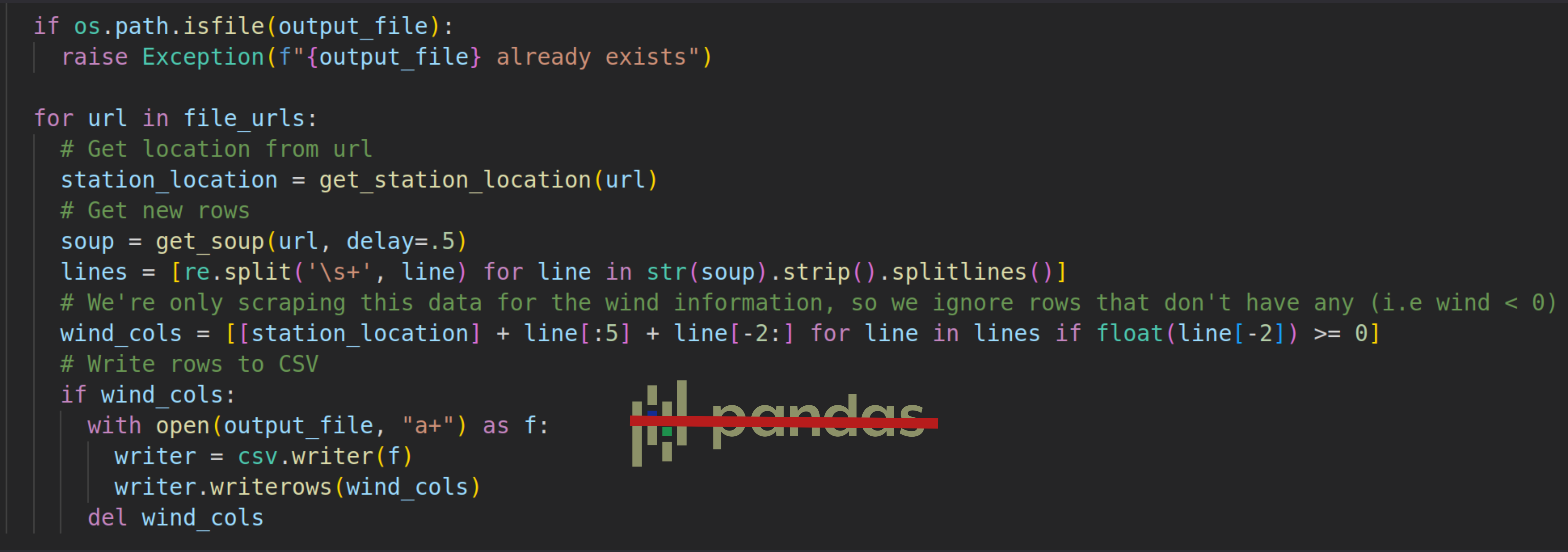
Too much data to read in and manipulate all at once with Pandas.

-> Recursion and batch processing



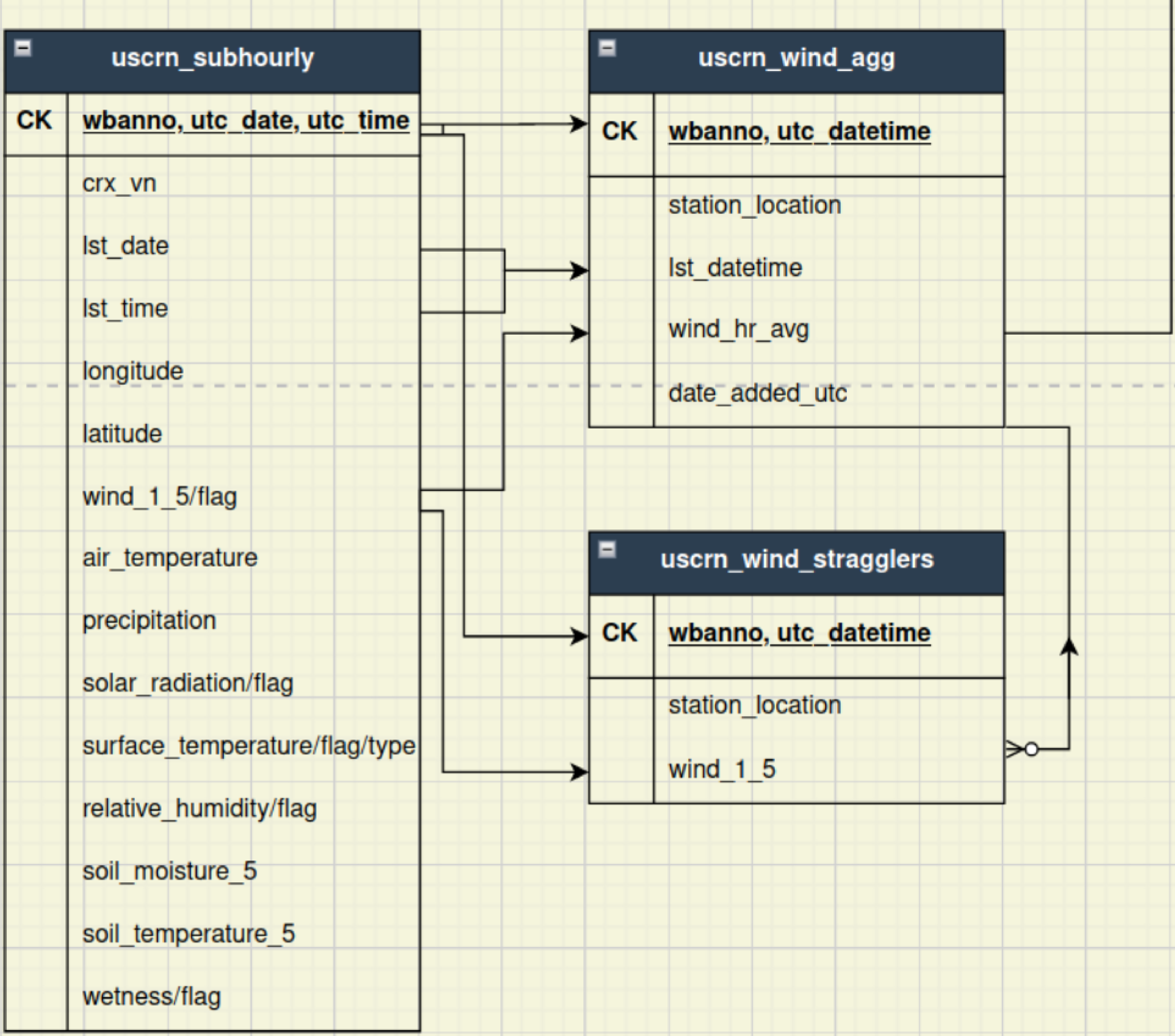
```python
from utils.utils import get_station_location, get_soup, get_file_urls

def process_rows(file_urls, row_limit, output_file) -> None:

  # Get rows for current batch
  rows = []
  current_idx=0
  for i, url in enumerate(file_urls[current_idx:]):
    # Get location from url
    station_location = get_station_location(url)
    # Get new rows
    soup = get_soup(url, delay=1)
    soup_lines = [station_location + " " + line for line in str(soup).strip().split("\n")]
    new_rows = [re.split('\s+', row) for row in soup_lines]
    # Add to list
    rows.extend(new_rows)
    if len(rows) >= row_limit:
      current_idx=i
      break

  # Create dataframe for current batch
  df = pd.DataFrame(rows, columns=columns)

  # Transform dataframe
  df = transform_dataframe(df)

  # Write dataframe to CSV
  hdr = False if os.path.isfile(output_file) else True
  df.to_csv("../airflow/dags/data/uscrn.csv", mode="a", header=hdr, index=False)
  del df
  gc.collect()

  # Recursively process remaining rows
  if len(rows) >= row_limit:
    remaining_urls = file_urls[current_idx:]
    rows.clear()
    process_rows(remaining_urls, row_limit, output_file)
  else:
    return

process_rows(file_urls=get_file_urls("hourly02"), row_limit=100000, output_file="data/uscrn.csv")
```

USCRN - Subhourly Data



```python
if os.path.isfile(output_file):
  raise Exception(f"{output_file} already exists")

for url in file_urls:
  # Get location from url
  station_location = get_station_location(url)
  # Get new rows
  soup = get_soup(url, delay=.5)
  lines = [re.split('\s+', line) for line in str(soup).strip().splitlines()]
  # We're only scraping this data for the wind information, so we ignore rows that don't have any (i.e wind < 0)
  wind_cols = [[station_location] + line[:5] + line[-2:] for line in lines if float(line[-2]) >= 0]
  # Write rows to CSV
  if wind_cols:
    with open(output_file, "a+") as f:
      writer = csv.writer(f)
      writer.writerows(wind_cols)
    del wind_cols
```

~~pandas~~



uscrn_subhourly

CK	**wbanno, utc_date, utc_time**
	crx_vn
	lst_date
	lst_time
	longitude
	latitude
	wind_1_5/flag
	air_temperature
	precipitation
	solar_radiation/flag
	surface_temperature/flag/type
	relative_humidity/flag
	soil_moisture_5
	soil_temperature_5
	wetness/flag

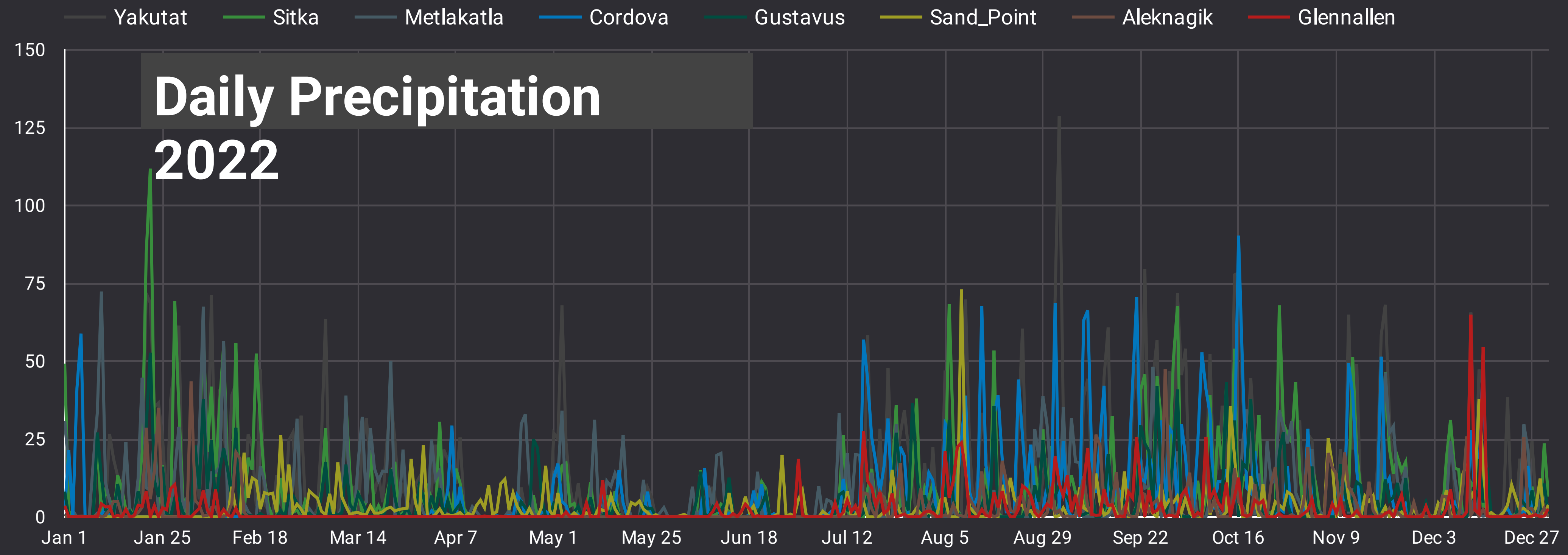
uscrn_wind_agg

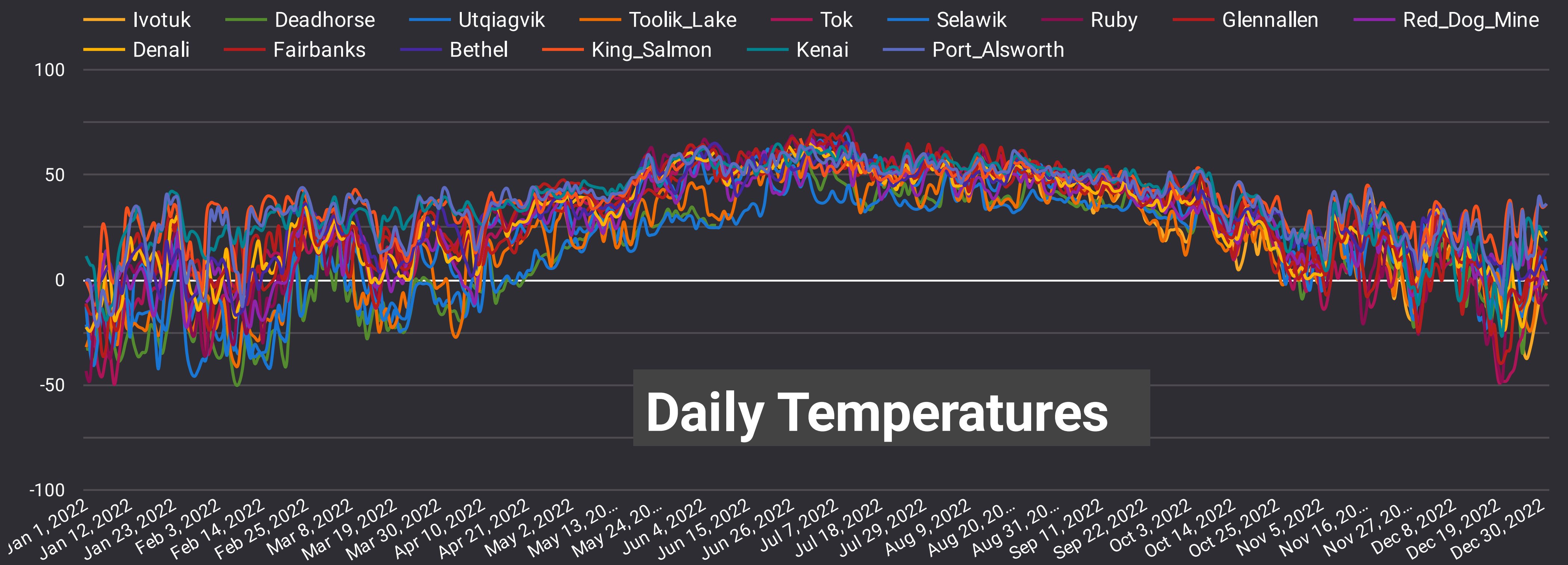
CK	**wbanno, utc_datetime**
	station_location
	lst_datetime
	wind_hr_avg
	date_added_utc

uscrn_wind_stragglers

CK	**wbanno, utc_datetime**
	station_location
	wind_1_5

Visualizations


Daily Precipitation 2022


Daily Temperatures

Data Health

Duplicate Rows
0

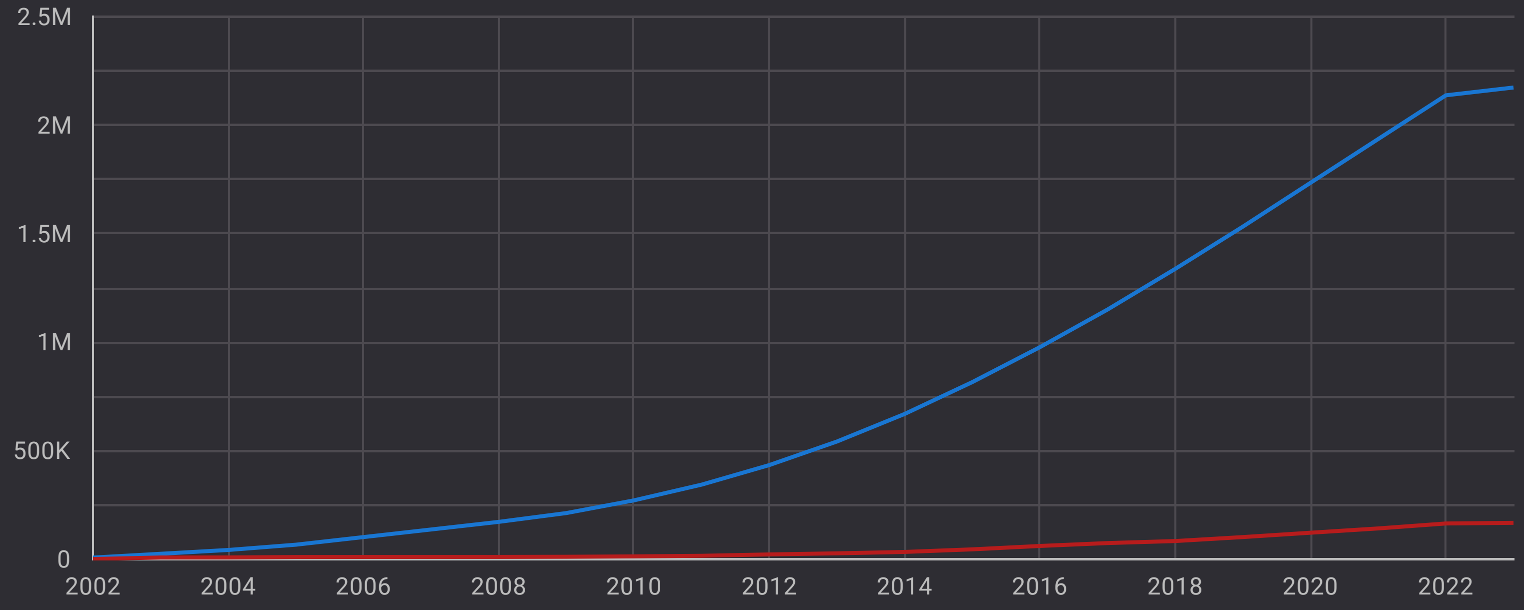
Null Values
0

USCRN: Completeness *(By Field)*

station_loca...	t_calc	t_max	t_min	t_hr_avg	p_calc	rh_hr_avg	solarad	solarad_max	solarad_min	sur_temp	sur_temp_max	sur_temp_min
Yakutat	97.94%	97.93%	97.95%	97.92%	94.79%	97.92%	98.06%	98.06%	98.06%	98.06%	98.06%	98.06%
Utqiagvik	97.74%	97.72%	97.77%	97.60%	93.55%	78.44%	98.12%	78.56%	78.56%	98.12%	78.56%	78.56%
Toolik_Lake	99.69%	99.68%	99.68%	99.68%	84.31%	99.72%	99.80%	99.80%	99.80%	99.73%	99.73%	99.73%
Tok	87.97%	87.97%	87.97%	87.95%	85.39%	88.48%	88.65%	88.65%	88.65%	88.63%	88.63%	88.63%
St._Paul	96.39%	96.42%	96.42%	96.04%	98.82%	75.56%	98.86%	94.07%	94.07%	98.65%	93.86%	93.86%
Sitka	99.65%	99.64%	99.66%	99.62%	99.95%	77.07%	99.98%	94.31%	94.31%	99.99%	94.31%	94.31%
Selawik	94.80%	94.80%	94.80%	94.71%	94.66%	95.07%	95.13%	95.13%	95.13%	95.07%	95.07%	95.07%

(All Fields)

station_location	rows_with_missing	pct_missing ▾
Bethel	14,858	37.23
Cordova	14,133	28.75
Ruby	17,499	23.42
Deadhorse	13,682	17.96
Aleknagik	4,950	16.58
Toolik_Lake	7,635	15.73
Tok	15,320	15.26
Ivotuk	11,130	14.41



NWS: Null Values

pct_null_wind_chill_f	nulls_other_cols ▾
16.69%	0

Duplicate Rows
0

NWS Rows Added

location	date_added_utc ▾	Rows Added
Ruby	May 12, 2023, 6:47:07 AM	144
King_Salm...	May 12, 2023, 6:47:07 AM	144
Sand_Point	May 12, 2023, 6:47:07 AM	144
Gustavus	May 12, 2023, 6:47:07 AM	144
Tok	May 12, 2023, 6:47:07 AM	144

1 - 100 / 805 ‹ ›

USCRN Rows Added

station_location	date_added_utc ▾	Rows Added
Fairbanks	Mar 7, 2023, 3:33:40 AM	180,339
Cordova	Mar 7, 2023, 3:33:40 AM	49,159
Sand_Point	Mar 7, 2023, 3:33:40 AM	118,726
Yakutat	Mar 7, 2023, 3:33:40 AM	57,192
Bethel	Mar 7, 2023, 3:33:40 AM	39,914

1 - 23 / 23 ‹ ›

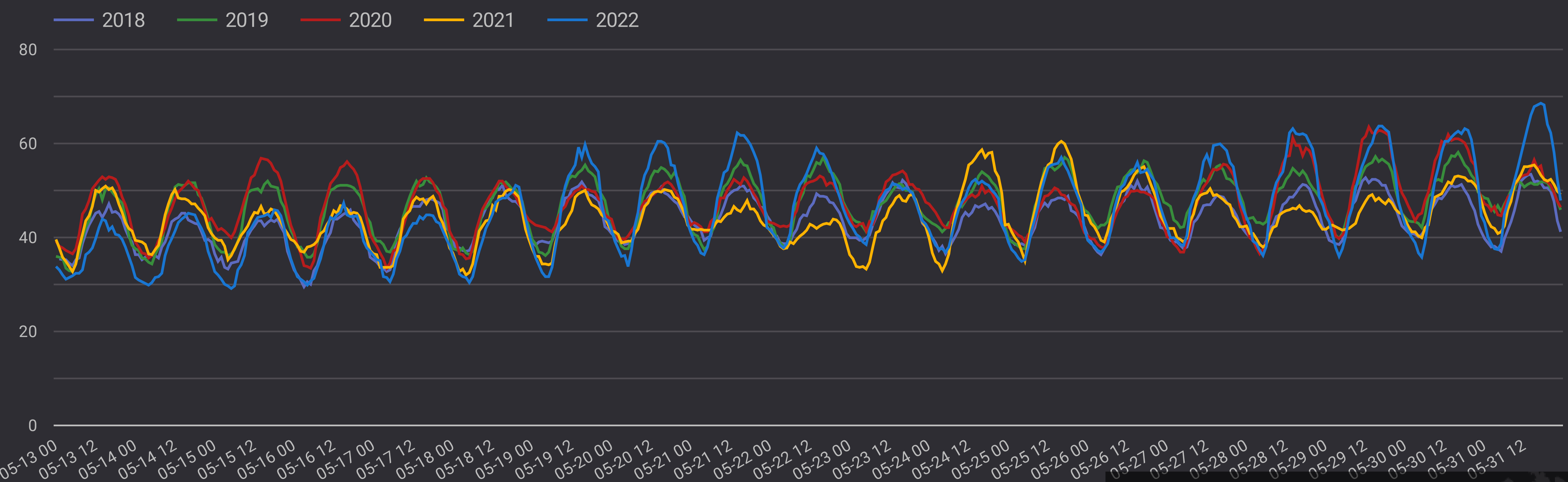
This Week's Temperatures

station_location ▾

date ▾	week
May 13, 20…	Week 19

Historic Temperature Averages (high, low, average)

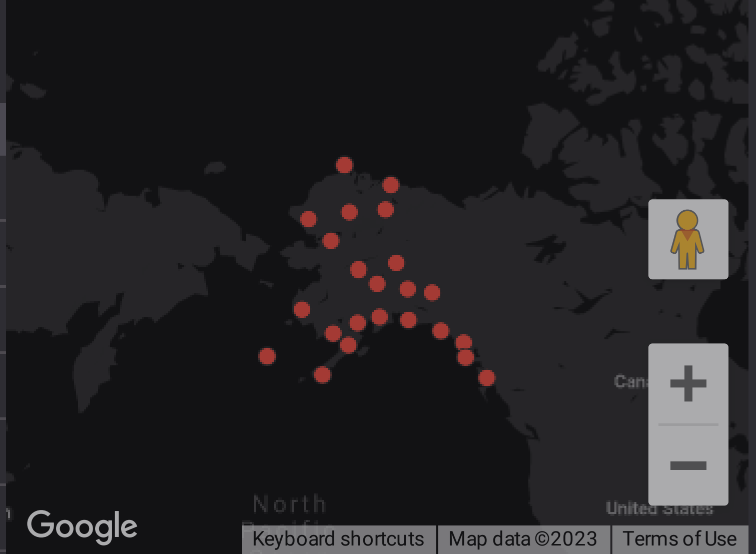
station_location	sunday	monday	tuesday	wednesday	thursday	friday	saturday
Fairbanks	(77.5, 53.4, 66.5)	(75.2, 52.5, 64.2)	(71.1, 47.7, 60.2)	(71.2, 44.6, 58.1)	(72.3, 52.5, 62.2)	(75.6, 49.3, 63.2)	(77.4, 52.9, 65.7)
Tok	(76.5, 38.8, 60.9)	(73.2, 39.2, 60.9)	(69.6, 29.1, 50.6)	(72, 27.1, 52.7)	(77.2, 29.7, 56.7)	(80.4, 30.9, 59.9)	(82.6, 30.7, 61.5)
Port_Alsworth	(70.9, 27.1, 51)	(73.4, 30.9, 52.9)	(72, 34, 53.9)	(70.3, 28.6, 51.1)	(75.7, 29.3, 54.3)	(77.9, 30, 57.4)	(75.9, 37, 57.7)



Current Temperature Forecast (Next 6 days)**

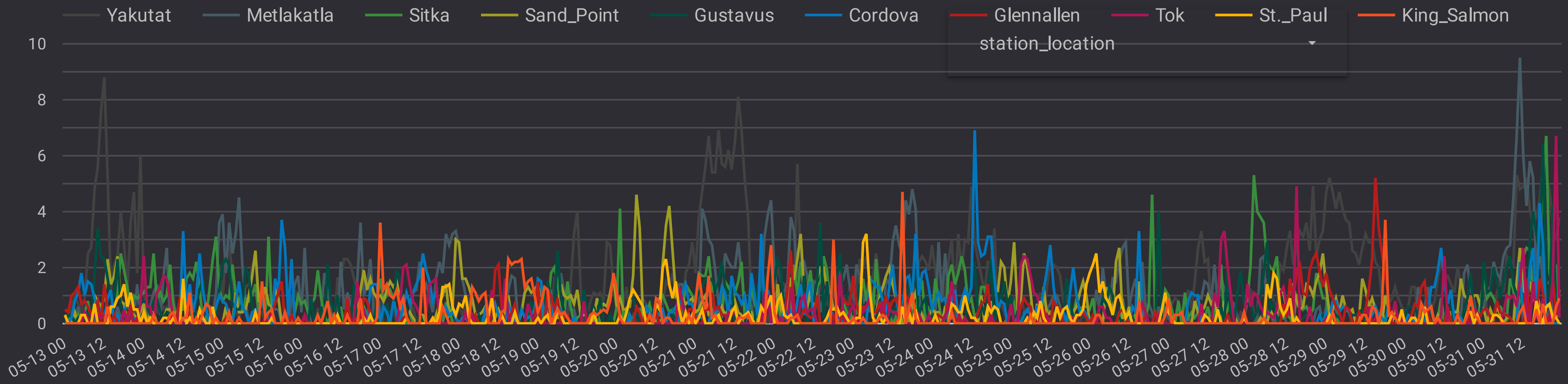
**NWS Does not provide hourly forecasts a full week in advance

location	today	tomorrow	day_after_tom…	four_days_out	five_days_out	six_days_out
Ruby	(56, 32, 46)	(61, 38, 50.7)	(62, 39, 51.3)	(56, 36, 46.9)	(56, 53, 55)	null
Deadhorse	(30, 18, 22.2)	(35, 28, 31.6)	(37, 29, 33.8)	(29, 25, 26.3)	(28, 25, 26.7)	null
Utqiagvik	(28, 19, 24.2)	(36, 28, 32)	(34, 29, 32)	(28, 23, 25.1)	(28, 27, 27.6)	null
Red_Dog_Mi…	(34, 26, 30.1)	(40, 34, 35.9)	(39, 32, 34.5)	(37, 29, 32.8)	(35, 34, 34.9)	null
Ivotuk	(44, 30, 36.4)	(44, 38, 41.3)	(44, 31, 37.5)	(36, 25, 30)	(38, 34, 36.6)	null
Sand_Point	(47, 39, 42.5)	(46, 41, 43)	(44, 40, 41.5)	(42, 39, 39.8)	(41, 40, 40.1)	null



Precipitation This Week:

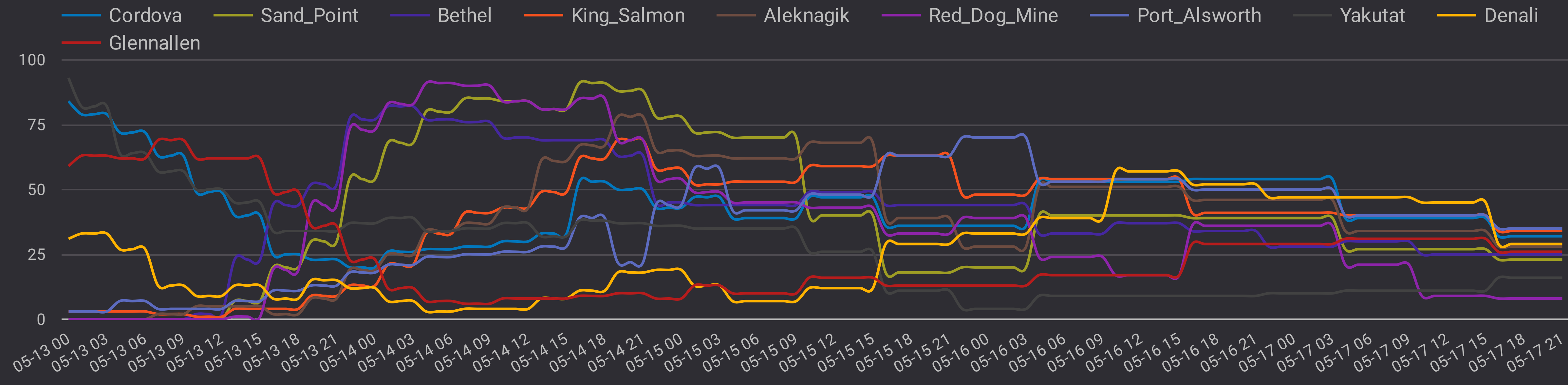
Historic Hourly Precipitation Averages (mm)



Historic Daily Averages

station_location ▾	sunday	monday	tuesday	wednesday	thursday	friday	saturday
Yakutat	42.3	11.2	7.4	4.1	28.5	5.5	16.5
Utqiagvik	12.6	27.4	81.1	29.9	43.5	4.1	4
Toolik_Lake	0.8	1.3	2.5	2.5	14.1	2.8	2.5
Tok	3.4	10.7	15.4	5.8	1	1.8	5.3

Hourly Precipitation Forecast (%)


Cordova
Sand_Point
Bethel
King_Salmon
Aleknagik
Red_Dog_Mine
Port_Alsworth
Yakutat
Denali
Glennallen