Wether_Forecast_Analysis.ipynb

```
#%%
import pandas as pd
import requests_cache
from retry_requests import retry
import openmeteo_requests
import requests
import time

#%%

#%%

#%%


# List of state capitals
states_and_capitals = [
    {"State": "Andhra Pradesh", "Capital": "Amaravati"},
    {"State": "Arunachal Pradesh", "Capital": "Itanagar"},
    {"State": "Assam", "Capital": "Dispur"},
    {"State": "Bihar", "Capital": "Patna"},
    {"State": "Chhattisgarh", "Capital": "Raipur"},
    {"State": "Goa", "Capital": "Panaji"},
    {"State": "Gujarat", "Capital": "Gandhinagar"},
    {"State": "Haryana", "Capital": "Chandigarh"},
    {"State": "Himachal Pradesh", "Capital": "Shimla"},
    {"State": "Jharkhand", "Capital": "Ranchi"},
    {"State": "Karnataka", "Capital": "Bengaluru"},
    {"State": "Kerala", "Capital": "Thiruvananthapuram"},
    {"State": "Madhya Pradesh", "Capital": "Bhopal"},
    {"State": "Maharashtra", "Capital": "Mumbai"},
    {"State": "Manipur", "Capital": "Imphal"},
    {"State": "Meghalaya", "Capital": "Shillong"},
    {"State": "Mizoram", "Capital": "Aizawl"},
    {"State": "Nagaland", "Capital": "Kohima"},
    {"State": "Odisha", "Capital": "Bhubaneswar"},
    {"State": "Punjab", "Capital": "Chandigarh"},
    {"State": "Rajasthan", "Capital": "Jaipur"},
    {"State": "Sikkim", "Capital": "Gangtok"},
    {"State": "Tamil Nadu", "Capital": "Chennai"},
    {"State": "Telangana", "Capital": "Hyderabad"},
    {"State": "Tripura", "Capital": "Agartala"},
    {"State": "Uttar Pradesh", "Capital": "Lucknow"},
    {"State": "Uttarakhand", "Capital": "Dehradun"},
    {"State": "West Bengal", "Capital": "Kolkata"},
    {"State": "Jammu and Kashmir", "Capital": "Srinagar"},
    {"State": "Ladakh", "Capital": "Leh"}
]

df = pd.DataFrame(states_and_capitals)

# Empty columns to fill
df["Latitude"] = None
df["Longitude"] = None
df["Timezone"] = None

# Fetch geocoding info
for index, row in df.iterrows():
    try:
        city = row["Capital"]
        url = f"https://geocoding-api.open-meteo.com/v1/search?name={city}&count=1"
        response = requests.get(url).json()

        result = response["results"][0]
        df.at[index, "Latitude"] = result["latitude"]
        df.at[index, "Longitude"] = result["longitude"]
        df.at[index, "Timezone"] = result["timezone"]
    except Exception as e:
        print(f"Failed for {row['Capital']}: {e}")
    time.sleep(1)

# Save to CSV
df.to_csv("indian_capital_geodata.csv", index=False)
print("Done! File saved as indian_capital_geodata.csv")

#%%

#%%
# In this step we load Weather Forecast Data into our Indian_Capital_Geodata

# Step 1: Load state capitals with geo-coordinates
df = pd.read_csv("indian_capital_geodata.csv")

# Step 2: Setup retry + cache session
cache_session = requests_cache.CachedSession('.cache', expire_after=3600)
retry_session = retry(cache_session, retries=5, backoff_factor=0.5)

# Step 3: Prepare list to collect forecast data
all_data = []

# Step 4: Loop through each capital city
for i, row in df.iterrows():
    city = row['Capital']
    state = row['State']
    lat = row['Latitude']
    lon = row['Longitude']
    timezone = row['Timezone']

    url = (
        f"https://api.open-meteo.com/v1/forecast?"
        f"latitude={lat}&longitude={lon}"
```

```
        f"&daily=temperature_2m_max,temperature_2m_min,rain_sum,uv_index_max,precipitation_probability_max"
        f"&timezone={timezone}&forecast_days=14"
    )

    try:
        response = retry_session.get(url).json()

        # Extract weather data
        days = response['daily']['time']
        temp_max = response['daily']['temperature_2m_max']
        temp_min = response['daily']['temperature_2m_min']
        rain = response['daily']['rain_sum']
        uv = response['daily']['uv_index_max']
        rain_chance = response['daily']['precipitation_probability_max']

        for j in range(len(days)):
            all_data.append({
                "State": state,
                "Capital": city,
                "Date": days[j],
                "Temp_Max": temp_max[j],
                "Temp_Min": temp_min[j],
                "Rain_Sum": rain[j],
                "UV_Index": uv[j],
                "Precipitation_Probability": rain_chance[j]
            })

        print(f"✅ Retrieved: {city}")

    except Exception as e:
        print(f"❌ Skipped {city}: {e}")

    time.sleep(3)  # Just to be respectful to API rate limits

# Step 5: Save combined dataset
df_forecast = pd.DataFrame(all_data)
df_forecast.to_csv("capital_weather_forecast.csv", index=False)
print("✅ Saved as capital_weather_forecast.csv")


#%%
df_forecast.to_csv("capital_weather_forecast.csv", index=False)
print("✅ Saved as capital_weather_forecast.csv")

#%%

#%%

#%%
# In this Block of code we try to get the weather data for past 45 years

# Load capital data
df = pd.read_csv("indian_capital_geodata.csv")

# Setup retry + caching
cache_session = requests_cache.CachedSession('.cache', expire_after=3600)
retry_session = retry(cache_session, retries=5, backoff_factor=0.5)

# Date range
start_date = "1980-01-01"
end_date = "2025-06-30"

all_data = []

# Loop through cities
for i, row in df.iterrows():
    city = row['Capital']
    state = row['State']
    lat = row['Latitude']
    lon = row['Longitude']
    timezone = row['Timezone']

    url = (
        f"https://archive-api.open-meteo.com/v1/archive?"
        f"latitude={lat}&longitude={lon}"
        f"&start_date={start_date}&end_date={end_date}"
        f"&daily=temperature_2m_max,temperature_2m_min,rain_sum,uv_index_max"
        f"&timezone={timezone}"
    )

    while True:
        try:
            response = retry_session.get(url).json()

            # Check rate limit exceeded
            if 'error' in response and 'Minutely API request limit exceeded' in response.get('reason', ''):
                print(f"🚦 Rate limit hit for {city}, waiting 60 seconds...")
                time.sleep(60)
                continue  # Try again after wait

            if 'daily' not in response:
                print(f"⚠️ No 'daily' data for {city}. Full response: {response}")
                break  # Exit this city loop and go to next city

            # Extract weather data
            days = response['daily']['time']
            temp_max = response['daily']['temperature_2m_max']
            temp_min = response['daily']['temperature_2m_min']
            rain = response['daily']['rain_sum']
            uv = response['daily']['uv_index_max']

            for j in range(len(days)):
                all_data.append({
                    "State": state,
```

```
                    "Capital": city,
                    "Date": days[j],
                    "Temp_Max": temp_max[j],
                    "Temp_Min": temp_min[j],
                    "Rain_Sum": rain[j],
                    "UV_Index": uv[j]
                })

            print(f"✅ Retrieved: {city}")
            break  # Exit while loop if successful

        except Exception as e:
            print(f"❌ Failed for {city}: {e}")
            break  # Exit if permanent error

    time.sleep(3)

# Save the final result
df_historic = pd.DataFrame(all_data)
df_historic.to_csv("capital_weather_historic.csv", index=False)
print("✅ All data saved to capital_weather_historic.csv")

#%%

#%%
# In this block of code we will save only those

# Load full capital list
df_all = pd.read_csv("indian_capital_geodata.csv")  # Full list of capitals

# Load already completed dataset
df_done = pd.read_csv("capital_weather_historic.csv")

# Extract unique cities already done
completed_cities = set(df_done['Capital'].unique())

# Filter only those cities which are NOT done yet
df_failed = df_all[~df_all['Capital'].isin(completed_cities)].copy()

# Save this list as failed_cities.csv
df_failed.to_csv("failed_cities.csv", index=False)

print(f"✅ Generated failed_cities.csv with {len(df_failed)} cities left to retry.")

#%%

#%%
import pandas as pd
import requests
import requests_cache
from retry_requests import retry
import time
import os

# Setup caching + retry logic
cache_session = requests_cache.CachedSession('weather_cache_resume', backend='sqlite', expire_after=86400)
retry_session = retry(cache_session, retries=5, backoff_factor=0.5)

# Load list of capitals
df_capitals = pd.read_csv("indian_capital_geodata.csv")

# Load already retrieved data if exists
if os.path.exists("capital_weather_historic.csv"):
    df_done = pd.read_csv("capital_weather_historic.csv")
    done_cities = set(df_done["Capital"].unique())
else:
    df_done = pd.DataFrame()
    done_cities = set()

# Date range
start_date = "1980-01-01"
end_date = "2025-06-30"

# Empty list to collect new data
new_data = []

# Loop through only pending cities
for i, row in df_capitals.iterrows():
    city = row["Capital"]

    if city in done_cities:
        print(f"⏭️ Skipping already done: {city}")
        continue

    state = row["State"]
    lat = row["Latitude"]
    lon = row["Longitude"]
    timezone = row["Timezone"]

    url = (
        f"https://archive-api.open-meteo.com/v1/archive?"
        f"latitude={lat}&longitude={lon}"
        f"&start_date={start_date}&end_date={end_date}"
        f"&daily=temperature_2m_max,temperature_2m_min,rain_sum,uv_index_max"
        f"&timezone={timezone}"
    )

    while True:
        try:
            response = retry_session.get(url).json()

            if 'error' in response and 'request limit' in response.get('reason', '').lower():
                print(f"🚦 Rate limit hit for {city}, waiting 60 seconds...")
                time.sleep(60)
```

```
                continue

            if 'daily' not in response:
                print(f"⚠️ No 'daily' data for {city}. Skipping...")
                break

            # Extract and append data
            days = response['daily']['time']
            temp_max = response['daily']['temperature_2m_max']
            temp_min = response['daily']['temperature_2m_min']
            rain = response['daily']['rain_sum']
            uv = response['daily']['uv_index_max']

            city_data = []
            for j in range(len(days)):
                city_data.append({
                    "State": state,
                    "Capital": city,
                    "Date": days[j],
                    "Temp_Max": temp_max[j],
                    "Temp_Min": temp_min[j],
                    "Rain_Sum": rain[j],
                    "UV_Index": uv[j]
                })

            # Append directly to file
            pd.DataFrame(city_data).to_csv("capital_weather_historic.csv", mode='a', index=False, header=not os.path.exists("capital_weather_historic.
            print(f"✅ Retrieved and saved: {city}")
            break

        except Exception as e:
            print(f"❌ Error for {city}: {e}")
            break

    time.sleep(3)
```

#%%

#%%