There are several feature engineering options based on domain knowledge to create variables that capture the severity of weather conditions necessary to cause flight delays.

These features move beyond single variables (like just temperature) and instead represent combined operational hazards.

### 1. Flight Category Index (Visibility and Ceiling) ☁️

This is arguably the most important factor for airport operations. Instead of using raw visibility or cloud cover, you can create a single categorical feature that mirrors the flight rules pilots must follow. This requires parsing the cloud ceiling height from the SkyCondition string (e.g., 'OVC**008**' means an overcast ceiling at 800 feet).

* **Feature: Flight\_Category**
* **Why it matters**: This directly quantifies the operational impact. **L-IFR** conditions cause significant delays and cancellations, while **VFR** conditions rarely do.
* **How to create it**:
  + **L-IFR (Low Instrument Flight Rules)**: Visibility < 1 mile OR cloud ceiling < 500 feet. **(Highest Severity)**
  + **IFR (Instrument Flight Rules)**: 1 <= Visibility < 3 miles OR 500 <= cloud ceiling < 1,000 feet. **(High Severity)**
  + **MVFR (Marginal Visual Flight Rules)**: 3 <= Visibility <= 5 miles OR 1,000 <= cloud ceiling <= 3,000 feet. **(Moderate Severity)**
  + **VFR (Visual Flight Rules)**: Visibility > 5 miles AND cloud ceiling > 3,000 feet. **(Low Severity)**

You can encode this as an ordinal variable, for example: L-IFR = 3, IFR = 2, MVFR = 1, VFR = 0.

### 2. Wind Severity and Gusts 🌬️

Pilots are concerned with both the sustained wind speed and the sudden, unpredictable changes from gusts.

* **Feature: Wind\_Severity\_Score**
* **Why it matters**: High or gusty winds, especially crosswinds, can exceed aircraft safety limits for takeoff and landing.
* **How to create it**:
  + 0 (Normal): WindSpeed < 15 knots AND ValueForWindCharacter is not 'G'.
  + 1 (High Wind): 15 <= WindSpeed < 25 knots AND ValueForWindCharacter is not 'G'.
  + 2 (Severe/Gusty): WindSpeed >= 25 knots OR ValueForWindCharacter contains 'G' (for Gust).

### 3. Icing and Freezing Precipitation Risk ❄️

Aircraft icing is extremely dangerous. The risk is highest when temperatures are near freezing *and* there is visible moisture in the air.

* **Feature: Icing\_Risk\_Flag**
* **Why it matters**: Icing conditions require planes to undergo lengthy de-icing procedures on the ground, causing delays. They are also hazardous in the air.
* **How to create it**: Create a binary flag that is 1 (high risk) if the following conditions are met, and 0 otherwise:
  + DryBulbCelsius is between -15°C and 2°C.
  + **AND** RelativeHumidity is high (e.g., > 80%) OR WeatherType indicates visible moisture like rain (RA), fog (FG), or mist (BR). The presence of freezing rain (FZRA) or freezing drizzle (FZDZ) in WeatherType should automatically trigger this flag.

### 4. Convective and Storm Activity ⛈️

Thunderstorms are a catch-all for the worst flying weather: turbulence, lightning, hail, and extreme wind shear.

* **Feature: Is\_Thunderstorm\_Activity** has\_thunderstorm
* **Why it matters**: Airports will halt ground operations and flights will not take off, land, or fly through areas with thunderstorms. It's a very strong predictor of delays.
* **How to create it**: This is a simple but powerful binary flag derived from WeatherType.
  + Is\_Thunderstorm\_Activity = 1 if WeatherType contains "TS" (e.g., 'TS', 'TSRA').
  + Is\_Thunderstorm\_Activity = 0 otherwise.
* **Feature: Precipitation\_Intensity** feature\_precipitation\_intensity
* **Why it matters**: Heavy precipitation (+) drastically reduces visibility and can flood runways, making it much more impactful than light (-) precipitation.
* **How to create it**: Parse the WeatherType string.
  + Precip\_Intensity = 1 if it starts with + (e.g., +RA).
  + Precip\_Intensity = -1 if it starts with - (e.g., -SN).
  + Precip\_Intensity = 0 otherwise.  
    You should also create separate binary flags for the type of precipitation (e.g., Is\_Snow, Is\_Rain) as snow is generally more disruptive.

has\_thunderstorm, has\_rain, has\_snow, has\_hail, has\_precipitation

Based on the flight data you've provided, here are the columns that will help explain flight delays and how you can use the linked weather data.

It appears your new dataset has **already been linked** with weather data, which is excellent. The columns prefixed with origin\_ and dest\_ are your weather variables.

### Key Variables in Your Flight Data to Explain Delays

#### 1. The Target Variable (What You Want to Predict)

You have several great options for the variable you want to predict (your target).

* **DEP\_DELAY / DEP\_DELAY\_NEW**: This measures the total departure delay in minutes. It's a good general-purpose target to model all types of delays.
* **WEATHER\_DELAY**: This is the perfect target variable if you want to build a model that *specifically* predicts the portion of a delay caused by weather.

#### 2. The Most Important Predictor Variables

These are the features from your flight data that will be most predictive of delays.

* **Temporal Features**: Time is a critical factor.
  + **DEP\_TIME\_BLK or CRS\_DEP\_TIME**: The time of day is crucial. Delays tend to accumulate throughout the day (a "cascading effect").
  + **MONTH and DAY\_OF\_WEEK**: These capture seasonal trends (e.g., summer thunderstorms, winter snow) and weekly traffic patterns.
* **Geospatial Features**: Where the flight is going is essential.
  + **ORIGIN and DEST**: These are your most important categorical features. Some airports are inherently more prone to congestion and weather delays (e.g., Chicago O'Hare in the winter).
* **Operational Features**:
  + **OP\_UNIQUE\_CARRIER**: The airline operating the flight can influence delays due to different hub operations and schedules.
  + **DISTANCE**: The length of the flight.

### How to Use the Linked Weather Data

Your dataset contains weather information for **both the origin and destination airports**. This is powerful because delays can be caused by bad weather at either end of the journey.

#### 1. Using Origin Weather Data (origin\_ columns)

These variables explain delays caused by conditions at the departure airport. You should apply the feature engineering techniques we discussed earlier to these columns.

* **Key Columns**:
  + origin\_HourlyVisibility
  + origin\_HourlyWindSpeed & origin\_HourlyWindGustSpeed
  + origin\_HourlyPresentWeatherType (This corresponds to WeatherType from the first dataset)
  + origin\_HourlySkyConditions
  + origin\_HourlyDryBulbTemperature & origin\_HourlyRelativeHumidity
* **Example Feature Engineering**:
  + Create an **origin\_Icing\_Risk** flag using origin\_HourlyDryBulbTemperature and origin\_HourlyRelativeHumidity.
  + Create an **origin\_Is\_Thunderstorm** flag from origin\_HourlyPresentWeatherType.
  + Create an **origin\_Flight\_Category** (L-IFR, IFR, etc.) using origin\_HourlyVisibility and origin\_HourlySkyConditions.

#### 2. Using Destination Weather Data (dest\_ columns)

This is a critical, and often overlooked, source of departure delays. A flight will not be allowed to take off if its destination airport is closed or has severely restricted operations.

* **Key Columns**:
  + dest\_HourlyVisibility
  + dest\_HourlyWindSpeed
  + dest\_HourlyPresentWeatherType
* **How it works**: If the dest\_HourlyPresentWeatherType shows a thunderstorm (TS), the national air system may implement a "ground delay program," holding the flight at its origin gate.
* **Example Feature Engineering**:
  + Create a **dest\_Is\_Unlandable** flag that is 1 if destination weather is severe (e.g., very low visibility, thunderstorms, or high crosswinds). This feature can be a very strong predictor of NAS\_DELAY and DEP\_DELAY.

By combining the flight's own characteristics (time of day, route) with the engineered weather features from **both the origin and destination**, you can build a powerful and accurate model to explain flight delays.

### How to Create a 'Late Aircraft' Feature

The key is to track each specific aircraft, identified by its TAIL\_NUM, throughout its flying day.

#### 1. The Logic: Calculating the Turnaround Buffer

For each flight in your dataset, you need to find:

1. The **actual arrival time** of the *previous* flight operated by that same aircraft (TAIL\_NUM).
2. The **scheduled departure time** of the *current* flight.
3. The difference between these two times, which represents the real, available time for turnaround.

#### 2. The Implementation Steps

You can implement this using tools like pandas in Python.

1. **Group by Aircraft and Date**: First, group your entire dataset by TAIL\_NUM and FL\_DATE\_2. This isolates the daily schedule for each individual plane.
2. **Sort by Time**: Within each group, sort the flights by CRS\_DEP\_TIME (Scheduled Departure Time) to ensure they are in chronological order.
3. **Calculate Previous Flight's Arrival Delay**: You'll need the arrival delay of the previous flight. Let's assume you have an ARR\_DELAY column (which is standard). You can use a shift() operation to get the ARR\_DELAY from the row above (the previous flight).
   * df['PREV\_AIRCRAFT\_ARR\_DELAY'] = df.groupby(['TAIL\_NUM', 'FL\_DATE\_2'])['ARR\_DELAY'].shift()
4. **Handle the First Flight of the Day**: The shift() operation will result in a NaN (or null value) for the first flight of the day for each plane. This is correct, as there is no preceding flight. You should fill these NaN values with 0, assuming the aircraft starts its day on time.
   * df['PREV\_AIRCRAFT\_ARR\_DELAY'].fillna(0, inplace=True)

The resulting feature, **PREV\_AIRCRAFT\_ARR\_DELAY**, is now a powerful predictor. A positive value indicates the plane was already behind schedule when it arrived at the gate, significantly increasing the probability that its next flight will also be delayed.

### A Crucial Warning: Avoid Data Leakage

Your dataset contains a column named LATE\_AIRCRAFT\_DELAY.

**Do not use this column as a feature to predict the delay of the same flight.**

This column tells you *how many minutes of the delay were officially attributed* to a late-arriving aircraft. This value is determined **after** the flight has already occurred and been delayed. Using it as a predictor is a classic error known as **data leakage** or **target leakage**, as you are giving the model information from the future that it would not have at the time of prediction.

Your engineered feature, PREV\_AIRCRAFT\_ARR\_DELAY, avoids this because it only uses information from the *past* (the previous flight) to predict the future (the current flight).