Washington State Flu Pipeline

Team Members: Thomas England, Andrew Fuerst, Addison DeSalvo

This project builds a complete data pipeline to collect, process, and analyze influenza trends in Washington State using data from the Washington State Department of Health, the CDC, and the Washington State Census. Three datasets were selected for this pipeline: the CDC FluView live API, the Washington State Respiratory Illness Surveillance dataset known as RHINO, and the Washington State Census population density by county.

Each dataset supports a different part of the final structure of the produced database and web application, which includes five tables that focus on influenza activity or provide geographic or temporal reference points. The FluView API supplies a current week-by-week view of influenza-like illness activity at the statewide level. The RHINO dataset offers a more localized perspective by linking respiratory illness visits to specific counties, although the historical range is limited to only a few recent years. The Census dataset provides population baselines which allow clinical data to be normalized and compared across counties.

Dataset Overview

The three datasets used for this project are the FluView live API from the CDC, the Washington State Respiratory Illness Surveillance dataset known as RHINO, and Washington State Census population density data.

FluView is a weekly updating API from the CDC that tracks influenza-like illness activity at the statewide level. It allows a broad view of influenza trends throughout Washington on a week-by-week basis. The key features used in this project were region, the two-letter state code, epiweek, a six-digit time identifier where the first four digits are the year, and the last two are the week number, and wili, which is the weighted influenza-like illness value. Although the API contains more fields, these were the primary elements relevant to the tables created for the pipeline.

FluView URL: https://api.delphi.cmu.edu/epidata/fluview/

The second dataset is the Washington State Respiratory Illness Surveillance dataset, referred to as RHINO. It includes several important features and reports about influenza like illnesses, including influenza, COVID-19, and RSV, at the county level. It is updated regularly and provided in CSV format. The main fields used include Season, Week Start, Week End, Location, Respiratory Illness Category, and Care Type, which identifies whether the event was an emergency visit or hospitalization. An additional County column was

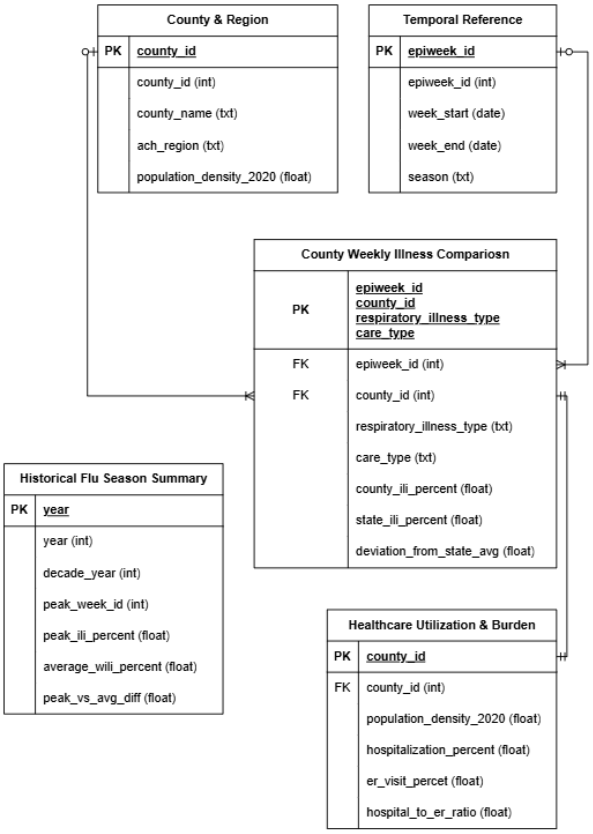
engineered by mapping the ACH region codes of the health providers in the Location feature to their corresponding counties.

RHINO URL: https://doh.wa.gov/data-statistical-reports/diseases-and-chronic-conditions/communicable-disease-surveillance-data/respiratory-illness-data-dashboard

The final dataset is the Washington State Census population density data, which reports county level population density for each decade. While structurally simple compared to the other two, this dataset provides the demographic baseline needed to interpret illness activity across regions. The main fields used were County Name and the population density columns listed by decade.

Census URL: https://data.wa.gov/Demographics/Population-Density-By-County-2000-2020/e6ip-wkqq

Database Schema



The database follows a star schema pattern, with fact tables arranged around shared dimension tables. The design used in this project includes two dimension tables, a temporal table, and a geographic table, along with three fact tables. The first fact table compares weekly county level illness rates to statewide averages. The second examines the local healthcare system burden by county. The third captures multiyear statewide influenza trends and identifies the peak illness week for each season.

This structure was chosen for two main reasons. The facts tables allow the pipeline to analyze influenza activity at two levels, statewide and local, while enabling direct comparison between them. The two dimension tables create a stable reference framework for both time and geography. This makes the system easier to expand in future work. New datasets can be added and linked through the shared temporal and geographic dimensions rather than rebuilding custom joins each time.

Table Definitions

In line with project specifications, five tables were developed. The two dimension tables are County and Region, which serves as the geographic reference, and the Temporal Reference table, which serves as the time reference. The three fact tables are the County Weekly Illness Comparison table, the Healthcare Utilization and Burden table, and the Historical Flu Season Summary table.

County and Region

The County and Region table is the central geographic reference for all other tables, combining fields from both the RHINO dataset and the Census dataset. Its features are county_id, county_name, ach_region, and population_density_2020.

County_id is a numerical identifier assigned to each Washington State county for consistent joins across tables. County_name is standardized to match the formatting used in both the Census and RHINO datasets. ACH_region represents the local accountable community of health associated with each county, taken from the RHINO data and mapped to a created County column to ensure the RHINO dataset could be joined to the Census dataset through county names. Population_density_2020 is taken from the Census dataset and provides a demographic baseline for interpreting illness activity, particularly because the RHINO dataset only begins in 2023.

The primary key of this table is county_id. It contains no foreign keys and is in third normal form as all non-key attributes depend solely on county_id. This table connects to the County Weekly Illness Comparison table in a one-to-many relationship.

Temporal Reference

The Temporal Reference table provides the unified time reference used throughout the pipeline and incorporates fields from both the RHINO dataset and the CDC FluView dataset. Its features are epiweek_id, week_start, week_end, and season.

Epiweek_id is the six digit value used in the FluView dataset to represent year and week. The first four digits are the year, and the last two are the week number. Week_start and Week_end come from the RHINO dataset and are included so that local county level data can be aligned with the CDC epiweek system. Season is also taken from the RHINO dataset and identifies the official illness season associated with each record.

The primary key of this table is epiweek_id. It contains no foreign keys and is in third normal form as all non-key attributes depend on epiweek_id. This table connects to the County Weekly Illness Comparison table in a one-to-many relationship.

County Weekly Illness Comparison

The County Weekly Illness Comparison table compares county level influenza activity to statewide averages for each epidemiological week. It is created by joining the RHINO and CDC FluView datasets through the County and Region table and the Temporal Reference table. Its purpose is to highlight how individual counties align with or diverge from the broader statewide average.

The fields include epiweek_id, taken from the Temporal Reference table, and county_id, taken from the County and Region table. Respiratory_illness_type identifies whether the illness is influenza, RSV, or COVID 19 and is taken from the RHINO dataset. Care_type, also from the RHINO dataset and specifies the type of clinical encounter associated with each record. County_ili_percent originates from the RHINO dataset as the One Week Percent field. State_ili_percent comes from the FluView dataset where it is reported as WILI, the weighted influenza-like illness percentage for Washington State. To support direct comparison between local and statewide activity, an additional field, deviation_from_state_avg, was engineered as the difference between county_ili_percent and state_ili_percent.

Because this table does not contain a single field that uniquely identifies each record, a composite primary key is used consisting of epiweek_id, county_id, respiratory_illness_type, and care_type. This ensures that each record is uniquely defined at the level of week, county, illness type, and encounter type, and keeps the table in third normal form. The foreign keys are epiweek_id, linking to the Temporal Reference table, and county_id, linking to the County and Region table. This table connects to the Healthcare Utilization and Burden table in a one-to-one relationship.

Healthcare Utilization and Burden

The Healthcare Utilization and Burden table provides a county-level view of how healthcare resources are used during respiratory illness activity. It combines data from the RHINO dataset with the demographic context from the Census data. The fields include county_id, taken from the County and Region table, and population_density_2020, taken from the Census dataset to give a basic demographic reference point when comparing illness burden across counties.

Three additional fields were engineered for this table. Hospitalization_percent was calculated by taking the percentage of respiratory illness cases in each county where the care type was hospitalization in the RHINO dataset. ER_visit_percent was calculated in the same way but limited to cases where the care type was emergency department visit.

Hospital_to_er_ratio represents the ratio of hospitalized cases to ER visit cases for each county, giving a sense of relative system strain.

The primary key of this table is county_id. As all attributes depend on the county identifier and no partial or transitive dependencies exist, the table is in third normal form.

Historical Flu Season Summary

The Historical Flu Season Summary table provides a statewide, year-level view of influenza trends rather than a week or county level view. It is created by joining elements of the FluView dataset with decade information from the Census dataset, and most of its fields are engineered. The first field, year, extracts the first four digits of the epiweek value from FluView to group all records belonging to the same year. Decade_year is taken from the Census dataset to identify the decade to which each year belongs, which supports potential future demographic analysis.

Peak_week_id identifies the week with the highest WILI value for each year by extracting the week component from the relevant epiweek. Average_wili_percent is the mean WILI value for all weeks in that year. Finally, peak_vs_avg_diff measures how much the peak WILI value deviates from the yearly average, giving a sense of how sharp or mild each season's peak was.

Year is the primary key of this table. It contains no foreign keys and no references to other tables. All non-key attributes depend solely on the year value, placing this table in third normal form. As there are no foreign keys in this table to connect it to other tables, it stands alone in the schema with no direct relationships.

Data Integration Strategy

The integration strategy for this project connects three separate datasets through two shared dimension tables. This structure allows statewide and county level influenza data to be compared consistently while keeping the system easy to expand.

The County and Region table provides the geographic reference point for all county level data. County names were standardized across the RHINO and Census datasets, so each record could be linked through a single county_id. This makes it possible to attach population density from the Census dataset to illness activity from RHINO and to use the same identifier throughout the fact tables.

The Temporal Reference table provides the unified time reference. CDC epiweeks were chosen as the standard time element and mapped to RHINO week start and week end

values so both datasets could use the same epiweek_id. Adding the season field also makes it possible to group records by the official illness season in future work.

With these dimensions in place, the fact tables link directly through them. The County Weekly Illness Comparison table uses county_id and epiweek_id to join RHINO percentages with statewide WILI values from FluView. The Healthcare Utilization and Burden table combines RHINO care type information with Census population density through county_id. The Historical Flu Season Summary table uses engineered year and decade fields to summarize statewide trends and does not require the shared dimensions.

Overall, the integration relies on standardized geographic and temporal identifiers, so the datasets can be compared without repeated manual joins.

Data Transformation Process

The data transformation process converts the three raw datasets into the cleaned and structured tables used in the final schema. Each dataset required specific preparation, and several fields were engineered to support analysis across time, geography, and illness type.

The RHINO dataset required the most cleaning. ACH regions were mapped to their corresponding counties using a custom ACH to county reference, and records listed as Statewide or Unassigned regions were removed. Because each ACH region includes multiple counties, the dataset was expanded by exploding the county list so each row represented a single county. The One Week Percent column was cleaned to convert empty values into nulls and standardize numeric values. A new epiweek_id field was created by combining the year and week values from the RHINO data so it could be matched to CDC epiweeks as the standard time element.

The Census dataset only needed minor preparation. County names were standardized to match the RHINO data, and the 2020 population density column was extracted as the base demographic feature of the schema. This allowed population context to be added to the healthcare burden table and to serve as a stable geographic reference.

The FluView dataset was pulled directly from the CDC API. Once loaded, it was filtered to the desired date range and key fields such as epiweek and WILI were extracted. These values served as the statewide averages used in the county comparison table and the historical summary table.

After cleaning, the two dimension tables were constructed. The County and Region table combine Census density values with the standardized county names and the ACH region assignments from RHINO. A county_id was added as the primary identifier. The Temporal

Reference table was created by taking the epiweek_id, week start and end dates, and season values from RHINO and converting the date fields to a standard format. This established a consistent time structure for the fact tables.

The three fact tables were then created. The County Weekly Illness Comparison table joined RHINO county level illness percentages with FluView statewide WILI values using epiweek_id and county_id. A field named deviation_from_state_average was engineered to show how each county compared to the state average for a given week. The Healthcare Utilization and Burden table combined population density with average percentages of hospitalizations and emergency visits for respiratory illnesses. From these values, hospitalization_percent, er_visit_percent, and the hospital_to_er_ratio fields were calculated. The Historical Flu Season Summary table extracted the year from the CDC epiweek values, assigned each year to a decade, identified the peak week for each year, and calculated the average seasonal WILI value along with the difference between the peak and the average.

Finally, each completed dataframe was loaded into PostgreSQL. All five tables were created according to the designed schema, and the cleaned data was ingested using COPY commands. This completed the transformation from raw source files into the structured database used by the pipeline.

Automation Workflow

The entire ingestion and processing pipeline is automated through a single Airflow DAG. The DAG coordinates every stage of the workflow, beginning with the collection of raw datasets, moving through transformation and table construction, and ending with ingestion into the PostgreSQL schema. All three external data sources are pulled automatically, specifically the Washington Department of Health RHINO dataset, the Washington State census dataset, and the CDC FluView API. Each task is wrapped as an Airflow decorated function, which lets the DAG handle retries, error handling, and execution order without manual supervision.

Airflow runs the pipeline daily. The RHINO dataset and the census file are retrieved directly from their published URLs, and FluView is pulled through the Delphi API. Because these calls always request the most recent available records, the pipeline automatically ingests new weeks of surveillance data as soon as they appear at the source. No additional versioning logic is needed. The API responses and CSV endpoints always return up to date content, and the downstream ingest step uses *ON CONFLICT DO NOTHING* to avoid duplicating rows that were already processed.

The automation trigger is strictly the Airflow scheduler. Once per day, Airflow fires the DAG, executes the extraction tasks in parallel, then moves on to constructing the five analytical tables, rebuilding schema objects when necessary, and loading new records through staged COPY commands. This keeps the relational database synchronized with all three external feeds without any manual runs. Errors can be determined by checking Airflow's logs. Specific errors that could arise would be a provider failing to respond, or a schema change breaks the ingest path. Under normal circumstances, however, the pipeline updates itself every day with no human involvement.

API Description

For an API, the project uses Flask which exposes the clean and transformed flu surveillance data. The API is organized around a small set of endpoints that surface the key analytical outputs of the pipeline. A basic root endpoint returns status information and a directory of available routes, while a dedicated health check verifies both API availability and database connectivity. All data served by the API is queried directly from the PostgreSQL tables using SQLAlchemy, which keeps the responses in sync with whatever the Airflow pipeline has most recently loaded.

The core functionality is provided through three report endpoints. The weekly trends endpoint summarizes recent statewide respiratory illness patterns by averaging county level influenza percentages and returning the most recent epiweeks and illness types. The healthcare impact endpoint aggregates hospitalization and ER visit rates by ACH region, along with population density and hospitalization to ER ratios. The historical summary endpoint returns long term seasonal metrics drawn from the historic table, including peak week identifiers, peak percentages, and differences between peak and average activity. Each report responds with two parts, a structured data list and a compact summary block designed to give consumers an immediate sense of the main findings.

In addition to the reports, the API provides a CSV export endpoint that allows users to download the tables directly. This makes it easy for external analysts to pull data into their own tooling without interacting with the database directly. The project also includes a browser based viewer that allows all these same functions to be done quite intuitively.

The responses are formatted as standard JSON objects, typically containing fields such as epiweek_id or respiratory_illness_type, and percentage values that the API formats for readability. A typical call, for example to /api/reports/weekly-trends, returns a short list of recent epiweeks with averaged influenza percentages and the number of reporting counties, along with a summary noting the most recent week and illness type. All database

reads are performed at request time, so any consumer calling the API is always interacting with the most current dataset produced by the automated pipeline.

Docker Implementation

The project is deployed through a fully containerized architecture built with Docker and orchestrated using a docker-compose stack. Each major component runs in its own container, which keeps the environment isolated, repeatable, and easy to rebuild. The system includes five coordinated services: PostgreSQL as the database layer, the Airflow webserver and scheduler for automation, the Flask API for data access, and a Jupyter notebook environment for exploratory work.

A single Dockerfile provides the base image for the Airflow, API, and Jupyter containers. It builds on Python 3.10, installs all required dependencies, and includes system packages such as the PostgreSQL client. The image copies the full codebase into the container, creates directories for logs and output files, and configures Airflow environment variables so that the scheduler and webserver detect and load the project's DAGs. The exposed ports make the key interfaces available on the host: Jupyter on 8888, the Airflow UI on 8080, and the Flask API on 5000.

The docker compose configuration defines how the services interact. The PostgreSQL container launches first and provides the shared database that all other services read from and write to. The Airflow webserver container initializes its metadata database, creates the default admin account if needed, and serves the UI for enabling or monitoring the DAG. The Airflow scheduler runs in its own container and executes the ETL pipeline at the interval specified in the DAG. Both Airflow containers share the same mounted project directory, so they can run the DAG code directly.

The Flask API container is built from the same base image to keep the environment consistent. It connects to the PostgreSQL service through Docker's internal network and begins serving endpoints as soon as the database becomes available. The Jupyter notebook container is also included for analysis and validation tasks, but it does not participate in the automated pipeline now that Airflow is responsible for ETL execution.

All services share a custom bridge network, which avoids manual IP configuration and ensures that containers can reference each other by service name. The design allows any component to be restarted independently while maintaining a stable environment for the pipeline. Bringing the system up with docker compose reproduces the entire stack without additional configuration, which makes the pipeline straightforward to run, test, or redeploy.

Instructions to Run the Entire Pipeline

The project is packaged so the full pipeline can be brought online with a single action. The only prerequisite is Docker and Docker Compose. Starting the stack with `docker-compose up -d` launches all required services: PostgreSQL for storage, the Airflow environment that orchestrates the ETL, and the Flask API that exposes the reports and dashboard. No additional configuration is needed because the environment variables and service wiring are handled directly in the compose file.

Once the containers are running, the ETL is executed entirely through Airflow. Airflow governs the data lifecycle from extraction to loading. After navigating to the Airflow UI, enabling the flu_data_airflow_v2.py DAG triggers the pipeline. The DAG downloads the latest RHINO, Census, and FluView data, applies all cleaning and transformation steps, recreates the database schema, and loads each of the five tables. Every refresh run produces a clean and fully updated dataset, which keeps the downstream API consistent with the most recent public health data.

After the DAG completes, the Flask service provides both the JSON endpoints and the interactive dashboard. Visiting the viewer page confirms that the ETL succeeded because all plots and CSV exports depend on valid database content. From this point on, the user can either allow Airflow to refresh the data automatically according to its schedule or manually trigger new runs through the Airflow interface. The system is fully functional as soon as the DAG completes, and no steps beyond standing up the containers and enabling the workflow are required.

Design Decisions and Challenges

This project was structured around building five tables from three different datasets, so the main design challenges centered on how to shape the raw data so it would fit into that framework without losing meaning. This was the reason for the star style layout. It kept the temporal and geographic references consistent while allowing the fact tables to focus on different surveillance questions. The separation also made it easier to reuse the same weeks and counties in multiple analyses instead of rebuilding them for every table.

Many of the coding challenges came from the source data rather than the schema. County names did not always match across datasets, and several counties appeared in the census data but not in the DOH RHINO data. Time alignment was another recurring problem. RHINO reports use week start and end dates, FluView reports everything as epiweeks, and Census data is decade based. Each one had to be reshaped into a shared epiweek_id before any comparisons could work. There were also irregular reporting

patterns, missing percentages, and categories that appeared in one dataset but not another. This was handled by enforcing consistent naming and validating keys before loading anything into the final tables. These decisions kept the downstream API and dashboard from having to fight with inconsistent inputs.

Conclusion and Possible Future Work

This project established a complete and repeatable pipeline for Washington State flu surveillance, pulling data from multiple sources, transforming it into structured tables, automating updates through Airflow, and making the results available through a functional API and dashboard. The system now provides a consistent foundation for tracking respiratory illness trends, evaluating healthcare utilization, and reviewing historical flu patterns.

Looking forward, the structure of the pipeline makes it easy to extend beyond the current scope. Additional illnesses, more granular care metrics, or new data sources could be added without altering the overall workflow. The API can support more advanced analytics or forecasting endpoints, and the automation layer can incorporate quality checks or alerting. The work completed here sets up a framework that could grow as surveillance needs do, making it useful not just for current analysis but possible future work.

Project Contributions

Thomas England

- Jupyter notebook development
- Python scripting for data processing
- Web API implementation
- Docker configuration and service integration

Andrew Fuerst

- Data engineering and table creation
- SQL query development
- Airflow installation and configuration
- DAG development and automation workflow design
- Docker configuration and service integration

Addison DeSalvo

- Proposal development
- Entity relationship diagram and schema design
- Table design and normalization logic

- Report writing and documentation