



Azure Data Pipeline for COVID-19 Reporting and Analysis

By: Radwa Esmail

CovidTrack Data Solution

Overview:

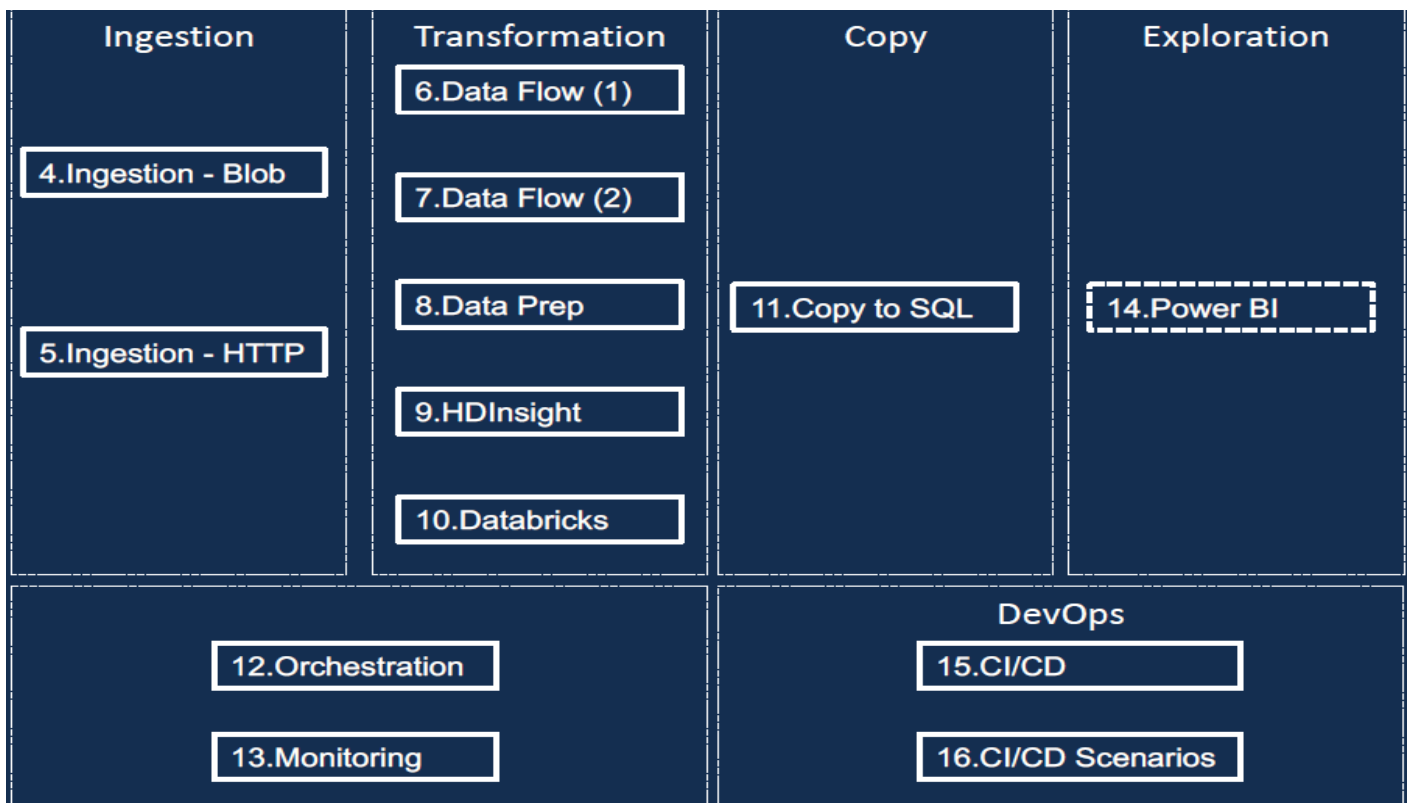
This project implements a data engineering solution using Azure Data Factory (ADF) to report COVID-19 trends and prepare data for future predictions. The solution ingests, processes, and stores data using various Azure technologies to create an end-to-end data pipeline.

Objectives:

- Ingest COVID-19 data from various sources
- Process and transform the data for analysis and reporting
- Implement ADF pipelines for data integration and transformation
- Orchestrate workflows and create a dependency between processes
- Monitor and manage pipelines using Azure monitoring tools

Technologies Used:

- Azure Data Factory (ADF): The core orchestration tool for creating and managing data pipelines.
- Azure Data Lake Storage Gen2 & Azure Blob Storage: For scalable storage of raw and processed data.
- Azure SQL Database: For storing processed, structured data.
- Azure Databricks: For advanced data transformation using notebooks.
- Microsoft Power BI: For generating reports and visualizing trends from processed data.
- Azure DevOps (CI/CD): For continuous integration and deployment of ADF artifacts across environments.

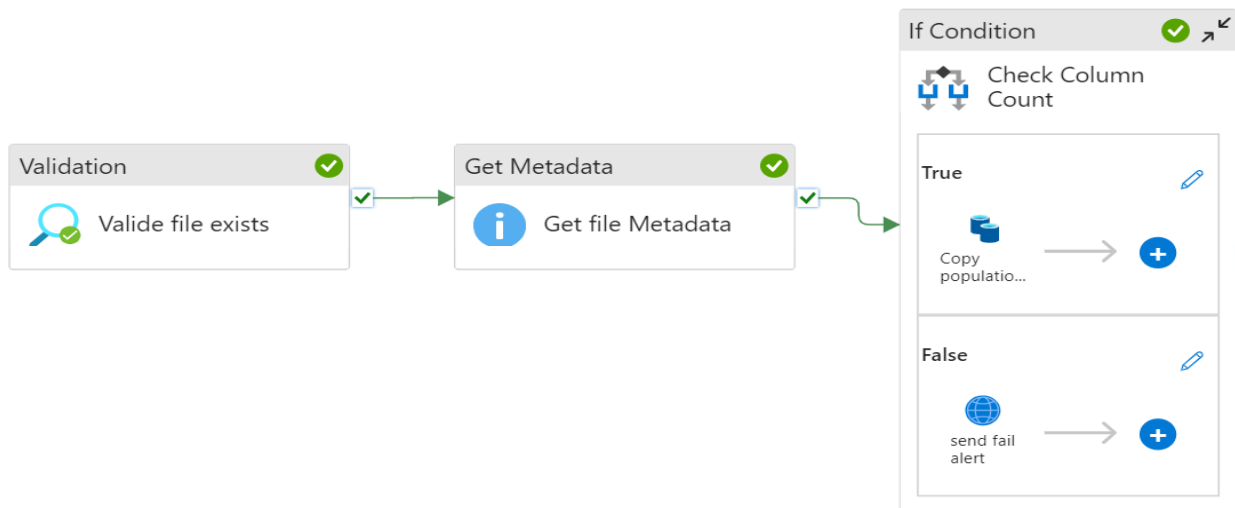


Data Ingestion:

Data is sourced from:

1. Azure Blob Storage and Azure Data Lake Gen2

Ingest "population by age" for all EU Countries into the Data Lake to support the machine learning models to predict increase in Covid-19 mortality rates

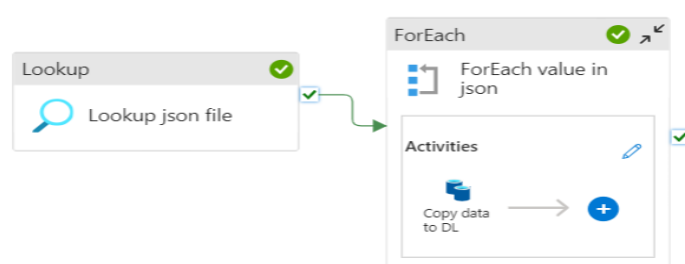


2. HTTP clients (COVID-19 APIs or public datasets)

To ingest COVID-19 data from external sources, the ADF pipeline integrates data via HTTP clients using public datasets. As a testing solution, we utilized variables and parameters to automate the data copying process.

Automation with JSON Configuration:

We developed a dynamic pipeline that reads from a JSON file containing the source URLs and destination filenames. The pipeline automates data retrieval and copying by reading each JSON entry and fetching the data to store in Azure Data Lake.



Parameters



Variables

Settings


Output

Pipeline run ID:

a5f0035c-f765-4741-a470-724defc86474



Pipeline status

 Succeeded

[View debug run consumption](#)



All status

List

[Monitor in Azure Metrics](#)

[Export to CSV](#)

Showing 1 - 7 of 7 items

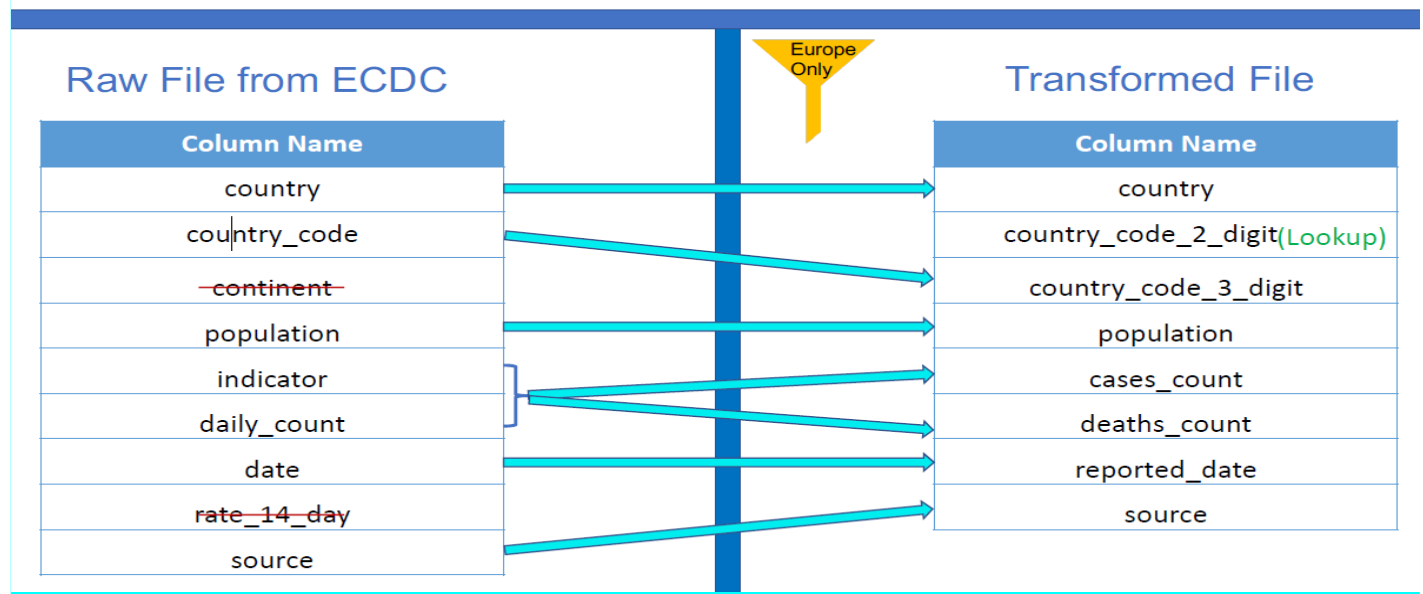
Activity name	Activity status	Activity type	Run start	Duration	Integration runtime
Copy data to DL	 Succeeded	Copy data	9/16/2024, 3:25:15 PM	13s	AutoResolveIntegration
Copy data to DL	 Succeeded	Copy data	9/16/2024, 3:24:59 PM	14s	AutoResolveIntegration

Data Transformation:

Data transformation is managed through ADF Mapping Data Flows, covering:

- Source transformation: Reading data from storage.
- Transformation steps: Filtering, selecting, pivoting, lookup, conditional split, derived columns, aggregation, joins, and sinks.
- Debugging and Issue Resolution: Implementing mechanisms to monitor and debug failures or discrepancies in data.

Transform Cases & Deaths Data



Cases and Deaths for Europe

The first transformation focused on the cases_deaths.csv dataset, applying a series of steps to filter, transform, and prepare the data for analysis.

1. Filtering:

Filtered the dataset to include only records for the Europe continent. Used the case_deaths_uk_ind_only.csv file to test and debug the filtering logic. Further filtered records where the country_code is not null.

2. Select Transformation:

Renamed and removed unnecessary columns as required for the output.

3. Pivot Transformation:

Applied a pivot transformation to aggregate the data, calculating the sum of confirmed cases and deaths by using the indicator and daily_count fields.

4. Lookup Transformation:

Used a separate lookup file to retrieve the `country_code_2_digit` by performing a join on the `country` field.

5. Final Select and Sink:

Performed a final select transformation to output the necessary columns with the correct naming conventions.

Loaded the transformed data into the target sink for further analysis and reporting.



Hospital Admissions (Daily and Weekly)

The second transformation focused on processing the `hospital_admissions.csv` file with separate flows for daily and weekly data, followed by data enrichment and aggregation.

1. Initial Data Load and Lookup:

Loaded the hospital admissions data from the data lake.

Used a Lookup Transformation to join the `country_code_2_digit` column by matching the `country` field.

2. Select Transformation:

Removed unnecessary columns and renamed the required ones for clarity.

3. Conditional Split:

Split the data based on the `indicator` field into two streams:

Daily Stream: Filtered for indicators `Daily hospital occupancy` and `Daily ICU occupancy`.

Weekly Stream: All other indicator values.

4. Daily Stream:

Applied a Pivot Transformation to aggregate the data, calculating the sum of values for unique indicators, resulting in:

`hospital_occupancy_count`.

`icu_occupancy_count`.

5. Weekly Stream:

Added a second source for a date dimension to enrich the weekly data. The original data lacked start and end dates for each week. Created a Derived Column in the date dimension using the expression `year + '-W' + lpad(week_of_year, 2, '0')` to match the `week_year` format (2020-W06).

Performed an Aggregate Transformation to calculate:
`max(date)` as `reported_week_end_date`.

`min(date)` as `reported_week_start_date`, grouped by the derived `year+'-W'+lpad(week_of_year,2,'0')` column.

Joined the date dimension data to the weekly hospital admissions stream.

6. Weekly Stream Aggregation:

Used a Pivot Transformation to calculate the sum of values for each indicator, resulting in:

Weekly new hospital admissions per 100k_count.

Weekly new ICU admissions per 100k_count.

7. Final Output:

Performed a Select Transformation to include only the necessary columns for the weekly report, renaming columns as required.

Sorted the daily and weekly final data by date (descending) and country (ascending).

Loaded the transformed data into the output sink.

The screenshot displays a data engineering tool interface with a data flow pipeline and a data preview table.

Data Flow Pipeline:

- hospital_admissions...** (Source) connects to **countrycodeslo...** (Lookup).
- countrycodeslo...** connects to **selecthospitala...** (Select).
- selecthospitala...** connects to **daily** (Table).
- daily** connects to **pivotdailyforoc...** (Pivot).
- pivotdailyforoc...** connects to **sortdailydata** (Sort).
- sortdailydata** connects to **selectfinaldaily...** (Select).
- selectfinaldaily...** connects to **dailydatasink** (Sink).
- daily** also connects to **weekly** (Table).
- weekly** connects to **joinwithdate** (Join).
- joinwithdate** connects to **pivotweeklyfor...** (Pivot).
- pivotweeklyfor...** connects to **selectfinalweekl...** (Select).
- selectfinalweekl...** connects to **sortfinalweekly...** (Sort).
- sortfinalweekly...** connects to **10 Columns** (Sink).
- 10 Columns** connects to **firstandlastday** (Table).
- firstandlastday** connects to **joinwithdate**.
- ccodeslookup** (Lookup) connects to **daily**.
- datadimlookup** (Lookup) connects to **reportedyearwe...** (Table).
- reportedyearwe...** connects to **firstandlastday**.

Data Preview Table:

country	country_code_...	country_code_...	population	reported_year_...	reported_week_...	reported_week_...	new_hospital_o...	new_icu_occup
Belgium	BE	BEL	11455519	2020-W43	2020-10-18	2020-10-24	28.3793340135...	NULL
Belgium	BE	BEL	11455519	2020-W42	2020-10-11	2020-10-17	15.4510677342...	NULL
Croatia	HR	HRV	4076246	2020-W42	2020-10-11	2020-10-17	12.8304327069...	NULL

Data Processing:

- **Azure Databricks:** Advanced transformations are implemented through Databricks notebooks. Pipelines invoke Databricks activities for heavy computation and data manipulation tasks.
- **Databricks Cluster Setup:**
 1. Created a Databricks cluster to perform the necessary data transformations.
 2. Mounted the required datasets, including population data and the country codes lookup.

The screenshot shows the Databricks web interface. The notebook 'mount_storage' is open, displaying three code blocks. Block 1 (07:10 PM) defines OAuth configurations for Azure storage. Block 2 (2 minutes ago) uses `dbutils.fs.mount` to mount the 'populationdataoutput' dataset. Block 3 (1 minute ago) uses `dbutils.fs.mount` to mount the 'processeddatabricks' dataset. The notebook is running on a cluster named 'radwa esmaiel's Cluster'.

- **Notebook Execution:**
 1. Developed a Python notebook for executing advanced transformations and data processing tasks.
 2. The notebook was triggered as an execution activity within an ADF pipeline, ensuring seamless integration between data pipelines and Databricks for large-scale processing.

The screenshot shows the ADF pipeline execution view. A 'Notebook' activity named 'excute transformation' (note the typo) is shown with a green checkmark indicating success. The pipeline run ID is 935a7cf2-5c5c-4da8-a0de-907848a7b407. The pipeline status is 'Succeeded'. The activity details table shows the activity name, status, type, run start time, duration, and integration runtime.

Activity name	Activity status	Activity type	Run start	Duration	Integration runtime
excute transformation	Succeeded	Notebook	9/17/2024, 7:32:26 PM	50s	AutoResolveIntegrator

Data Persistence: Azure SQL Database

To persist the transformed data, we utilized Azure SQL Database for structured storage and querying. Below are the key steps followed:

- **Azure SQL Server and Database Creation:**

Set up an SQL Server and created a database within Azure to store the processed data.

- **Table Creation:**

Created the following tables in the database to hold COVID-19 reporting data:

covid_reporting.cases_and_deaths

covid_reporting.hospital_admissions_daily

covid_reporting.hospital_admissions_weekly

covid_reporting.population_by_age

- **Data Ingestion:**

Data from the transformed files was copied into the SQL database tables using manual mapping to ensure data integrity.

After ingestion, queries were executed to verify that the data was correctly inserted into the respective tables.

```
1 SELECT TOP (1000) * FROM [covid_reporting].[hospital_admissions_weekly]
```

Results Messages

country_code_3_digit	year_week	week_start_date	week_end_date	hospital_occupancy	icu_occupancy
BEL	2020-W43	2020-10-18	2020-10-24	28.37933401	
CZE	2020-W43	2020-10-18	2020-10-24	81.95459070	14.92985784
DNK	2020-W43	2020-10-18	2020-10-24	2.79017809	

Data Orchestration:

- **Branching and Chaining:** Pipelines are controlled using branching activities such as If Condition, ForEach, Validation, etc., to ensure smooth flow.
- **Triggers:** Event-based, scheduled, and tumbling window triggers manage pipeline execution, ensuring timely and consistent data refreshes.
- **Dependencies:** The project creates dependencies between pipelines to manage orchestrated data flow from ingestion to transformation.

A fully automated pipeline was created to handle the ingestion, transformation, and loading of population data, following these key steps:

- 1. Ingestion and Validation:**

The pipeline begins by executing the ingestion process for the population file.

A validation step checks if the file exists in the specified location.

A Get Metadata activity ensures that the column count matches the expected number of columns in the file.

An If Condition activity is used to verify successful validation. If the validation passes, the file is copied from Azure Blob Storage to Azure Data Lake.
- 2. Transformation:**

After the ingestion step, the pipeline triggers a Databricks notebook for performing the required data transformations on the population data. The notebook is integrated into the pipeline using an Execute Databricks Activity.
- 3. Loading into SQL Database:**

Once the data is transformed, the pipeline executes the final step, which copies the transformed data into an Azure SQL Database table, specifically `population_by_age`.
- 4. Event Trigger:**

An event-driven trigger was added to the ingestion pipeline, using a Storage Event Trigger. This trigger automatically executes the full pipeline as soon as a blob is added to the population container.
- 5. Verification:**

The table was queried, and the data was confirmed to be correctly inserted.

Query 1 × Query 2 ×

Run

Cancel query

Save query

Export data as

Show all

Open Copilot

Results

Messages

digit	country_code_3_digit	age_group_0_14	age_group_15_24	age_group_25_49	age_group_50_64	age_group_65_99
	ALB	17.20	15.50	33.00	20.20	11.40
	AND	13.90	10.60	39.40	22.50	10.20
	ARM	20.20	11.80	36.90	19.10	9.00
	AUT	14.40	10.90	34.00	21.70	13.80
	AZE	22.40	14.10	39.10	17.60	5.30
	BLR	16.90	9.90	36.60	21.30	11.30
	BEL	16.90	11.40	32.70	20.10	13.30
	BGR	14.40	8.90	35.00	20.40	16.50

✓ Validate▶ Debug⚡ Trigger (1)

Execute Pipeline

Execute ingestion...
population_ingestion...

Execute Pipeline

Execute...
databricks_covidrepo...

Execute Pipeline

copy transformed...
databricks_sqlized_pi...

ParametersVariablesSettingsOutput

Pipeline run ID: b6919088-ffa3-4051-8ac9-c00d50492ed9

Pipeline status ✓ Succeeded

View debug run consumption

All status

Monitor in Azure MetricsExport to CSV

Showing 1 - 3 of 3 items

Activity name	Activity status	Activity type	Run start	Duration	Integration runtime
copy transformed data to DB ...	✓ Succeeded	Execute Pipeline	9/17/2024, 11:05:06 PM	22s	
Execute transformation Pipeli...	✓ Succeeded	Execute Pipeline	9/17/2024, 10:59:40 PM	5m 27s	
Execute ingestion Pipeline	✓ Succeeded	Execute Pipeline	9/17/2024, 10:59:13 PM	28s	

Tumbling Window Triggers:

Tumbling Window Triggers were implemented:

The first trigger initiates ingestion from HTTP sources using links specified in a JSON file.

The second trigger, dependent on the first, runs the pipeline for data flow transformation.

The third trigger, dependent on the second, runs the pipeline that copies the transformed data to the SQL database.

Filter by name

Annotations : Any

Showing 1 - 8 of 8 items

Name	Type	Status	Related	Annotations
http_automated_trigger	Schedule	✓ Started	0	
population_fullpipeline_trigger	Storage events	✓ Started	1	
casesanddeath_sqltrigger	Tumbling window	✓ Started	1	
dailysql_trigger	Tumbling window	✓ Started	1	
hospitaltransformation_trigger	Tumbling window	✓ Started	3	
http_ingestion_trigger	Tumbling window	✓ Started	3	
transformation_trigger	Tumbling window	✓ Started	2	
weekkysql_trigger	Tumbling window	✓ Started	1	

Power BI Reports

The dashboard provides comprehensive insights into COVID-19's impact across various countries in Europe continent, with a focus on cases, hospitalizations, and population demographics. Key metrics and visualizations help highlight the pandemic's severity, healthcare strain, and population distribution by age group.

1. Transformations and Data Preparation

Unpivoting: Age group data from the `population_by_age` table was unpivoted to create a more readable and analyzable structure.

Value Replacement: Some values were replaced for better readability, ensuring that labels and data are clear.

Slicers: Slicers for country and date were implemented to allow dynamic filtering, providing a more interactive user experience.

Country_dim: integrated a conformed dimension for country and population details, linking it to all other tables. This dimension provides consistency across your data, ensuring that all metrics (such as cases, hospitalizations, and age group data) are aligned with accurate country and population information.

2. Key Measures and Visualizations

Case Fatality Rate (CFR) Analysis:

Measure: Calculated as $\text{deaths_count} / \text{cases_count} * 100$, providing the percentage of deaths relative to confirmed cases for each country.

Top 10 Countries by Hospital Occupancy vs Population:

Measure: Showcased the top 10 countries with the highest ratio of hospital occupancy to total population, highlighting the countries most impacted by healthcare strain.

Top 10 Countries by Total Cases vs Population:

Measure: Displayed the top 10 countries with the highest ratio of total COVID-19 cases to population size.

Total Cases and Total Deaths:

Measure:

Total Cases: Sum of `cases_count` per country.

Total Deaths: Sum of `deaths_count` per country.

Hospital Admissions vs ICU Occupancy:

Measure: Compared the daily hospital admissions (`hospital_admissions_daily`) against the ICU occupancy (`icu_occupancy_count`) for better insights into healthcare demand.

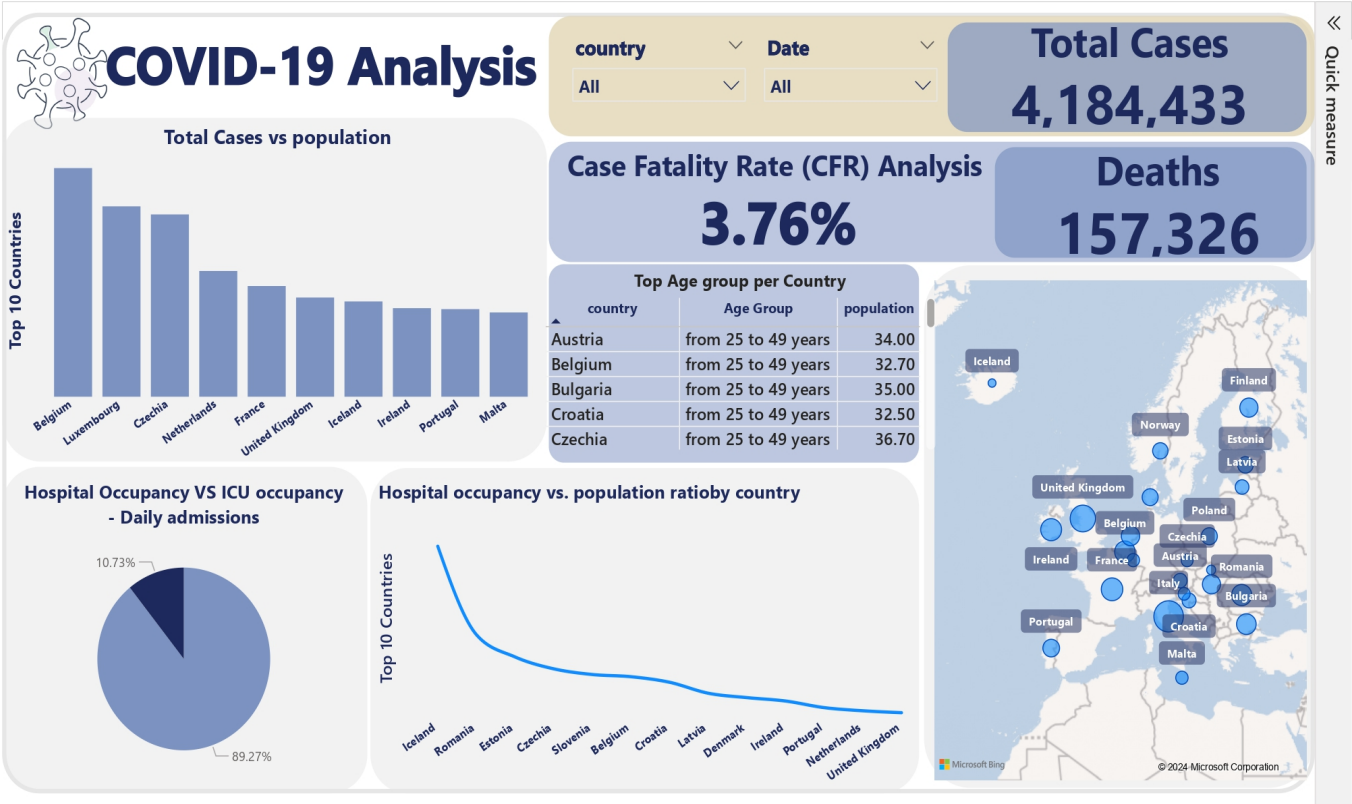
Map Visualization:

Visualization: Map with bubble sizes.

Bubble Size: Represented the total number of cases per country, providing a geographic perspective on the spread of COVID-19.

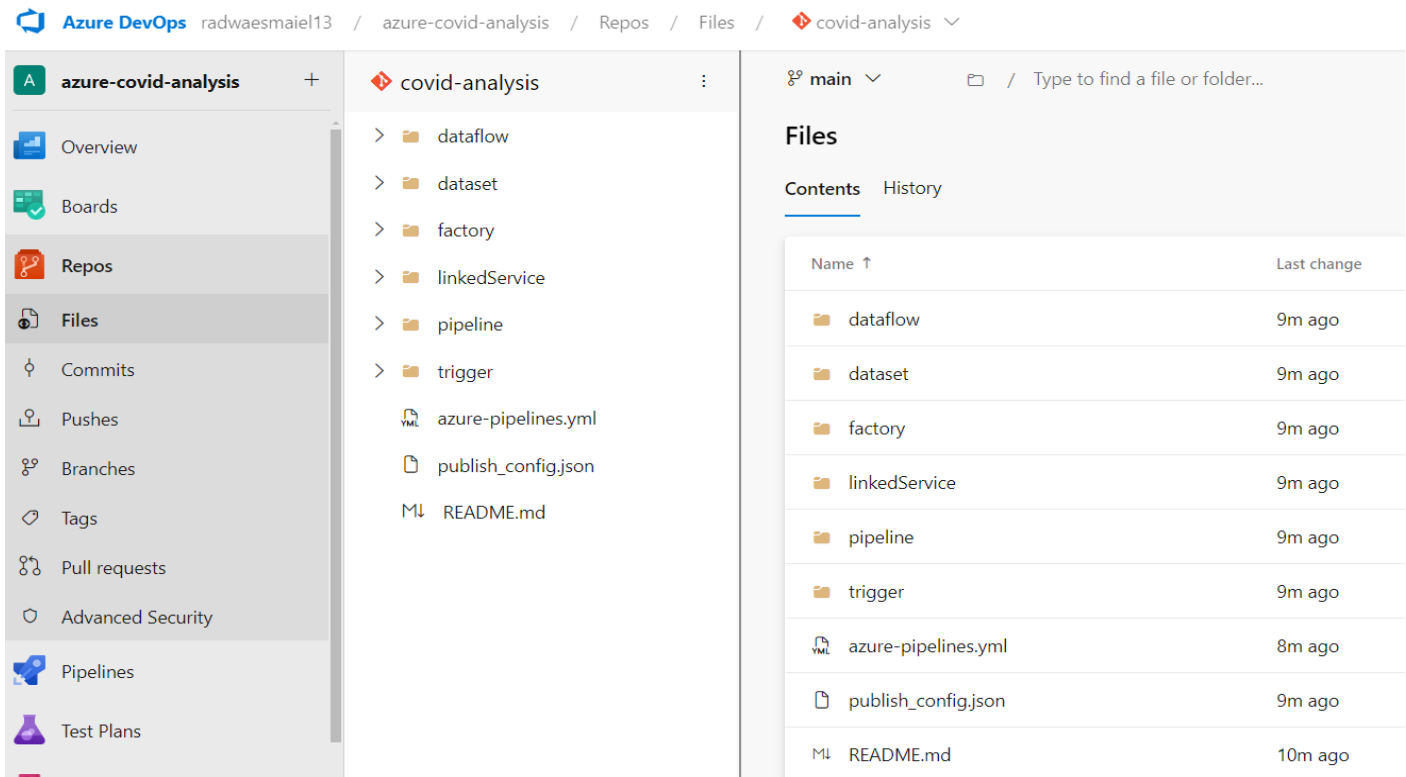
Top Age Group by Population for Each Country:

Visualization: Table, Content: Displayed each country with its highest age group by population, offering insights into demographic distribution and potential age-related impacts of the pandemic.



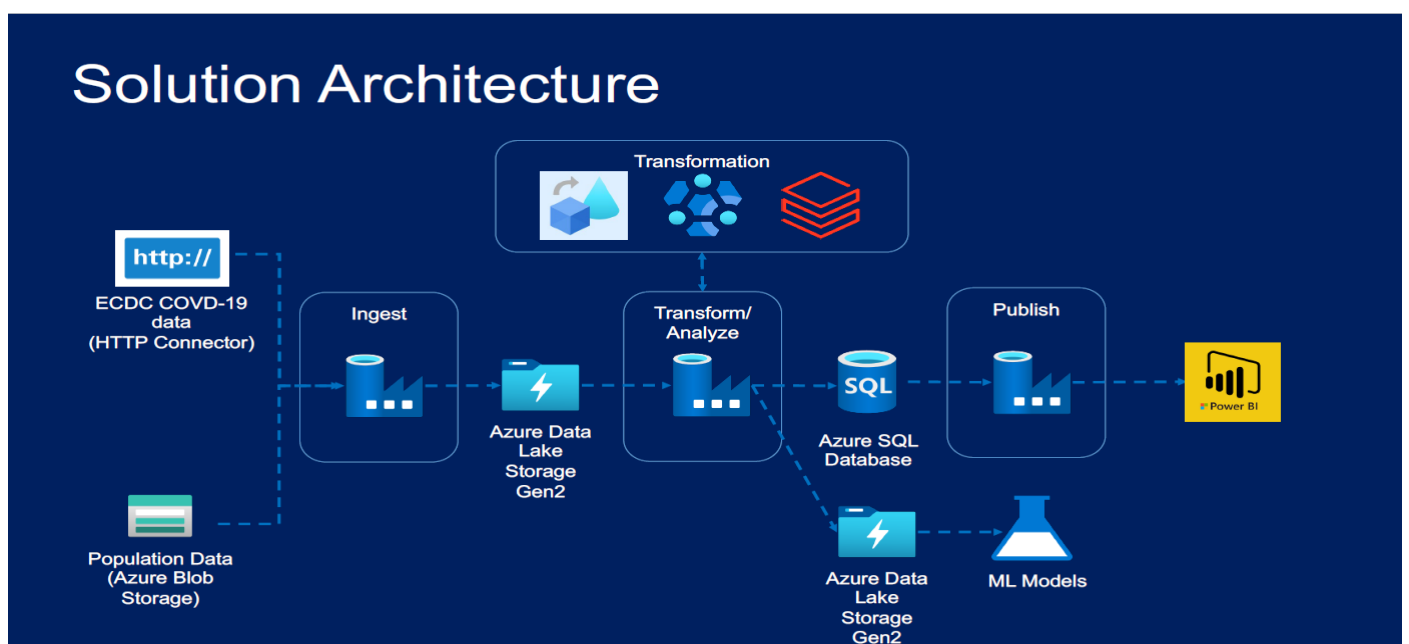
CI/CD Integration:

- **Azure DevOps:** The project implements CI/CD pipelines for releasing ADF artifacts.
- **Version Control and Release Pipelines:** Azure DevOps Git and release pipelines ensure a smooth code management process for higher environment releases.



Project Outcome:

Upon completion, this project offers a robust framework for ingesting, processing, and reporting on COVID-19 data trends. The solution is scalable and can be extended to support additional transformations, storage solutions, and machine learning models for predictive analysis.



course and use case:

<https://www.udemy.com/course/learn-azure-data-factory-from-scratch/learn/lecture/27591718#overview>



Certificate no: UC-ba32521b-1b81-4c18-9a13-58c070f9757c
Certificate url: ude.my/UC-ba32521b-1b81-4c18-9a13-58c070f9757c
Reference Number: 0004

CERTIFICATE OF COMPLETION

Azure Data Factory For Data Engineers - Project on Covid19

Instructors **Ramesh Retnasamy**

Radwa Esmaiel

Date **Sept. 18, 2024**

Length **13 total hours**

GitHub: link of full project and Data :

<https://github.com/RadwaEsamiel/Azure-COVID-Analysis/tree/main>