CSC12107 – INFORMATION SYSTEMS FOR BUSINESS INTELLIGENCE

# Building and Mining Data Warehouse

Project 2024 – 2025

21HTTT2 – Ho Chi Minh City University of Science - VNUHCM

Instructors: MSc. Nguyen Ngoc Minh Chau, MSc. Tiet Gia Hong, MSc. Ho Thi Hoang Vy

Group 11

✦

Ho Chi Minh City, November 2024

# Table of Contents

## Contents

# Introduction

- GitHub repository: https://github.com/kru01/ISBI_AirQualityDW (Won't be publicly available before the final submission).
- YouTube playlist: https://youtube.com/playlist?list=PLnfnJ5OTnHtOGSF_7IGO4POBs6xLR9M6w&si=PDYsodbme1MaR0Ba

**Table 1.** Student information and Task assignment table.

| ID | Name | Task | Status (%) |
|---|---|---|---|
| 21127004 | Tran Nguyen An Phong | *Stage to NDS:*<br>- Data cleaning and transformation.<br>- Create NDS schema and db.<br>- ETL for STATE.<br>*Report:*<br>- Sect. 2.2.1 – Data cleaning and transformation.<br>- Sect. 2.3 – DDS schema design and ETL.<br>- Sect. 3.2, 3.3 – Dimensions. | 100 |
| 21127135 | Diep Huu Phuc | *Source to Stage:*<br>- Create METADATA and STAGE dbs.<br>- ETL for 10_STATE_AQI.<br>*NDS to DDS:*<br>- Create DDS schema and db.<br>- ETL for DIM_GEO.<br>*Cube and OLAP:*<br>- Build cube.<br>- MDX for Q1 – 4, and 10.<br>*Report:*<br>- Sect. 4 – Scheduling. | 100 |
| 21127296 | Dang Ha Huy | *Source to Stage:*<br>- ETL for 2B_USCOUNTIES.<br>*NDS to DDS:*<br>- Generate DIM_DATE.<br>- ETL for FACT_AIR_QUALITY.<br>*Report:*<br>- Sect. 2, 2.1 – Preface, Source (.csv) to Stage.<br>- Sect. 3, 3.1 – Measures and Calc Members. | 100 |
| 21127385 | Pham Uyen Nhi | *Stage to NDS:*<br>- ETL for COUNTY, and AIR_QUALITY.<br>*Cube and OLAP:*<br>- MDX for Q5, 6, 8, 9, 11, and 12.<br>*Report:*<br>- Sect. 1 – Data Description. | 100 |

| | | | |
|---|---|---|---|
| | | - Sect. 2.2.2 – NDS schema design and ETL. | |

## Installation

Regarding our setup,

- **SQL Server 2019** and **SSMS 19**.
- **Visual Studio 2022** for **SSIS**, and **VS2019** for **SSAS** and **Data Mining**.
- For all server connections, we just use a "**.**" (**localhost**) and Windows authentication.

1 – After successful extraction, our project should have the following core components,

- **Data** – Folder storing the .csv(s), preserved exactly as when they are provided.
- **Docs** – Folder storing the Project Assignment and Report.
- **SQL** – Folder storing 5 scripts for warehouse creation, and one for deletion.
- **SSAS_Group11_2019** – Folder storing VS2019 solution for SSAS.
- **SSIS_Group11** – Folder storing VS2022 solution for SSIS.
- **OLAP_Visual.pbix** – Power BI Desktop file for analysis graphs and charts.
- **OLAP.mdx** – MDX query file for analysis questions.

2 – Run scripts in the **SQL** folder in the following order to set up the Data Warehouse,

1_create_metadata → 2_create_stage → 4_create_nds → 5_create_dds

- The **3_clean_data** must only be run after completing ETL from Source to Stage.

3 – For **SSIS**, if your server is not on **localhost**, all the packages' **Connection Managers** must be changed. Otherwise,

- In **CsvSrc_Stage**, edit the **10_STATE_AQI - LOOP CSVs** and change the **Folder** path in the **Collection** tab.
- Also in the same package, reselect the flat file connection **2b_uscounties_csv**'s **File name** field.

4 – For **SSAS**, double-click **21BI11 DDS.ds** in the **Solution Explorer**, navigate to the **Impersonation Information** tab then change the Windows **User name** and **Password**.

- Be aware that this is the password associated with your **Microsoft account** (or the personal **Outlook** mail in our case), *not whatever PIN or password for the Windows Lock Screen*.

# 1 Data Description

All .csv files under **Air Quality Data** folder possess identical data structures. Each entry describes the AQI value, measured by some criteria, in a state's county on a specific date. It comprises of,

- **State Name**, and **county Name** – (string) Name of the state, and county where the monitor resides.
- **State Code**, and **County Code** – (int) ID of said state, and county.
- **Date** – (string) The date that the measure was taken. Although this technically should belong to type **date**, we shall see clearer in data cleaning that the 2023 entries have erroneous formats.
- **AQI** – (int) The result that was calculated by the monitor.
- **Category** – (string) The six-category classification based on the ranges of AQI values. From Good (0 – 50), Moderate (51 – 100), Unhealthy for Sensitive Groups (101 – 150), Unhealthy (151 – 200), Very Unhealthy (201 – 300), to Hazardous ($\geq$ 301).
- **Defining Parameter** – (string) Name of the pollutant used as a AQI measuring basis by the monitor.
- **Defining Site** – (string) If multiple sites, the ID (or name) of the one with maximum AQI value in the state's county.
- **Number of Sites Reporting** – (int) The number of monitors used, and by extension the number of sites.
- **Created**, and **Last Updated** – (datetime) The moment that the record is created, and last updated.

**(2B)uscounties.csv** is an unrelated compilation of *almost* (also explained in data cleaning) all counties in the US.

- **county**, and **county_ascii** – (string) Name of the county, and its ascii representation.
- **county_full** – (string) The full formal specification of the name.
- **county_fips** – (string) Five-digit ID of the county, only certainly unique within a state.
- **state_id** – (string) Two-letter ID name for the state, from USPS postal abbreviation.
- **state_name** – (string) Name of the state containing the county.
- **lat**, and **lng** – (float) Latitude, and longitude of the county.
- **population** – (int) An estimate of the county's population.

Because of different code pages between the files and our machines, to conserve as much originality as possible when translating type **string** to SQL, we prefer using **NVARCHAR** in the database and **Unicode string** during ETL's Data Conversion.

# 2 Data Warehouse

To facilitate Incremental Load and Auditing (e.g., tracking data sources, update time, etc.), we will maintain a database named **21BI11_METADATA**.

- **Three tables** corresponding to the phases (STAGE, NDS, DDS) with each row storing the phase's table name and its **LSET** (Last Successful Extraction Time). CET (Current Extraction Time) is not included since it can be gotten through GETDATE() during ETL.
- **SRC_SYS** table to define the source systems (the .csv files), and **SCD_STATUS** for state (inactive, active) when handling Slowly Changing Dimension. As cross-database foreign key is not possible, these tables are purely for reference, although we can still cross-database join to filter out invalid sources, or states, when needed.

## 2.1 Source (.csv) to Stage

**21BI11_STAGE** has two tables which are,

- **10_STATE_AQI** has the same columns as the .csv(s) in Air_Quality_Data and mimics their data types. Except, **DATE** is set as **NVARCHAR** to preserve data integrity, the main reason for this will be discussed further when cleaning data.
- **2B_USCOUNTIES** also shares the same columns with (2B)uscounties.csv but with the addition of CREATED and LAST_UPDATED during ETL.
- For tracing purposes, a **SOURCE_ID** column will also be added to both tables.

The ETL process goes through the following steps,

1. **Get LSET, CET** for the corresponding table from the metadata.
2. **Truncate the table** to clear old data and reset identity column.
3. **Load only new data** (in the time range between LSET and CET) from Source to Stage.
4. **Update LSET** metadata to CET for the table.

Firstly, with **10_STATE_AQI**, we use a **Foreach Loop Container** to iterate through the .csv(s). And because **Flat File Source** must be used for .csv, we can't extract data through a SQL command (like with Excel Source). That's why a **Conditional Split** is employed to check if the data is new, and only then is it loaded. A **Data Conversion** node is also required for the data to be compliant with the Stage's schema. In addition, a counter variable is increased with each loop to provide the current file index for **Derived Column** to make SOURCE_ID.

As for **2B_USCOUNTIES**, it is much simpler since we only need to ensure compatible data types and the inclusion of CREATED, LAST_UPDATED, and SOURCE_ID.

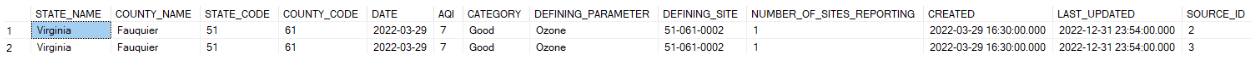
## 2.2 Stage to NDS

### 2.2.1 Data cleaning and transformation

*Refer to **SQL/3_clean_data.sql** for the detailed process and implementation.*

To start off, **STAGE.10_STATE_AQI**'s 2023 entries have a **DATE** format of **DD-MM-YYYY**, while the rest is **YYYY-MM-DD**. With an **UPDATE** and some string manipulations, all the affected rows can be corrected. Next, there exist trailing whitespaces in certain columns. Specifically, all **COUNTY_NAME** rows of the state **Alaska**. Nevertheless, for good measure, every **NVARCHAR** column will be trimmed during the ETL flow.

Initially, the air quality data in **STAGE.10_STATE_AQI** has 194971 rows, but if we try to query **distinct** entries while ignoring SOURCE_ID, the number is now 194964. Taking a sample for confirmation, we can conclude that the 7 affected rows have exactly matching data, albeit from different sources.



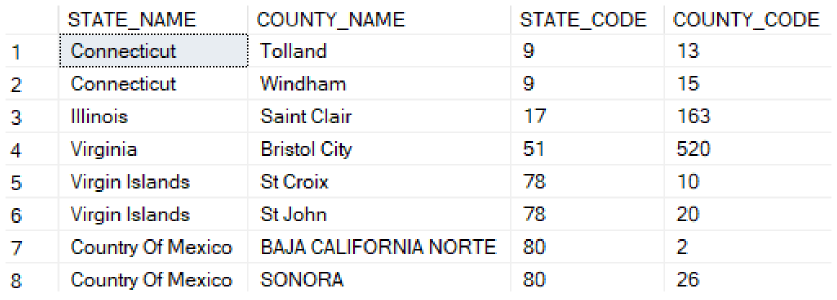
| | STATE_NAME | COUNTY_NAME | STATE_CODE | COUNTY_CODE | DATE | AQI | CATEGORY | DEFINING_PARAMETER | DEFINING_SITE | NUMBER_OF_SITES_REPORTING | CREATED | LAST_UPDATED | SOURCE_ID |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Virginia | Fauquier | 51 | 61 | 2022-03-29 | 7 | Good | Ozone | 51-061-0002 | 1 | 2022-03-29 16:30:00.000 | 2022-12-31 23:54:00.000 | 2 |
| 2 | Virginia | Fauquier | 51 | 61 | 2022-03-29 | 7 | Good | Ozone | 51-061-0002 | 1 | 2022-03-29 16:30:00.000 | 2022-12-31 23:54:00.000 | 3 |

**Figure 221a.** Query for STATE_CODE=51, COUNTY_CODE=61, and DATE=2022-03-29.

Since identical data does not offer any value, we will only retain one record, this can be addressed during ETL to NDS. As for **STAGE.2B_USCOUNTIES**, no duplication is found.

Now on the topic of mismatched data, we have verified that both 10_STATE_AQI's **CATEGORY** and **DEFINING_PARAMETER** conform with the AQI basics table (given in the Assignment), and WHO's pollution measures. However, there are many misclassifications between **AQI** and **CATEGORY**, e.g., rows with AQI < 201 but labeled as "Very Unhealthy," or AQI < 301 but as "Hazardous." For a thorough solution, all entries will be recategorized based on AQI basics table.

Likewise, certain **(State, County)** pairs from 10_STATE_AQI don't align with (or exist in) 2B_USCOUNTIES.



| | STATE_NAME | COUNTY_NAME | STATE_CODE | COUNTY_CODE |
|---|---|---|---|---|
| 1 | Connecticut | Tolland | 9 | 13 |
| 2 | Connecticut | Windham | 9 | 15 |
| 3 | Illinois | Saint Clair | 17 | 163 |
| 4 | Virginia | Bristol City | 51 | 520 |
| 5 | Virgin Islands | St Croix | 78 | 10 |
| 6 | Virgin Islands | St John | 78 | 20 |
| 7 | Country Of Mexico | BAJA CALIFORNIA NORTE | 80 | 2 |
| 8 | Country Of Mexico | SONORA | 80 | 26 |

**Figure 221b.** List of 10_STATE_AQI's (State, County) pairs that can't be identified in 2B_USCOUNTIES.

First, after investigating, we learn that **Connecticut**'s **Tolland** is part of the **Southeastern Connecticut Planning Region**, and **Windham** the **Capitol Planning Region**. These regions do exist in 2B_USCOUNTIES, and despite consisting of many small towns and cities, our data does not reach that level of detail. So, we might as well map Tolland and Windham to their respective regions' names.

Moving on, **Illinois**' **Saint Clair** should be changed to **St. Clair**. And **Virginia**'s "**Bristol City**" is the county's full name, but for 10_STATE_AQI, all the counties use shortened name, thus, it will be contracted to **Bristol**.

Lastly, the **Country of Mexico** is not a state of the US, therefore, all its entries will be removed during ETL to NDS. Yet, **Virgin Islands** is certainly one, so to prevent the loss of its AQI data leading to our skewed estimate of US's air quality, we consider manually creating data for **the state and its counties** in **2B_USCOUNTIES**.

- Data for **St. Croix** and **St. John** will be compiled from multiple sources. Still, the USPS doesn't have an abbreviation for Virgin Islands, since it lies outside the US's Customs Territory. Fortunately, we can use **VI** for its **STATE_ID** as there isn't a state denoted with that ID in 2B_USCOUNTIES.
- Since this new data is handcrafted, we will update **METADATA.SRC_SYS** to include a new entry recording which script is the origin.

The final **row count** for **NDS.AIR_QUALITY** should be,

194964 (unique records) – 1472 (of Mexico) = **193492**.

### 2.2.2 NDS schema design and ETL

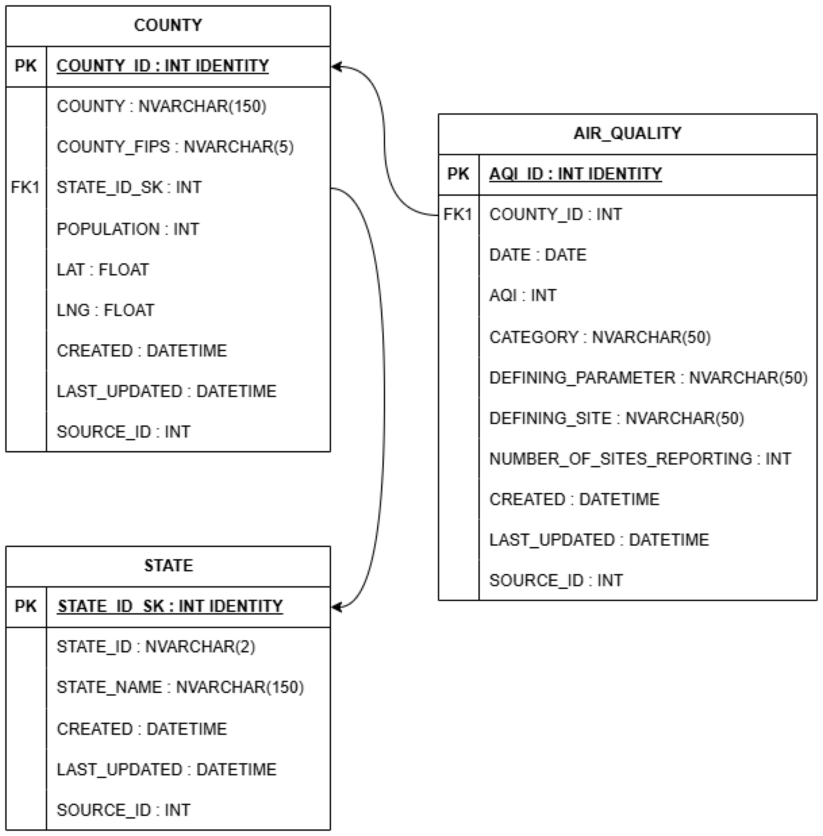


**Figure 222a.** Detailed NDS schema, with foreign keys and data types.

The schema can be derived from normalizing STAGE's to at least 3NF, removing certain superfluous columns, then attaching Surrogate Keys (SK).

- **STAGE.10_STATE_AQI**'s **STATE_CODE** and **COUNTY_CODE** are discarded since they don't correlate with anything in **STAGE.2B_USCOUNTIES**, and the latter's **STATE_ID** and **COUNTY_FIPS** together can uniquely identify a county.
- **STAGE.2B_USCOUNTIES**'s **COUNTY_ASCII** and **COUNTY_FULL** are also redundant as the first is identical to **COUNTY**, while the second adds very little information. Moreover, **COUNTY** alone is sufficient for it matches the shortened **COUNTY_NAME** that **STAGE.10_STATE_AQI** uses to designate a county.

Worth noting as well is that, in Fig. 222a, the absence of STATUS columns is because SCD isn't used during this phase. If a record exists in the NDS, it'll be directly updated. The ETL process must go from **STATE**, **COUNTY**, then **AIR_QUALITY** due to foreign key constraints.

#### *2.2.2a STATE*

STATE will be populated by **STAGE.2B_USCOUNTIES**, we will first remove any repeating state (that might have come from different sources) by **sorting STATE_ID**, and trim trailing spaces through **Derived Column**. Then with **Lookup**, check (by its id) whether the state is already in **NDS.STATE**. If it does, we grab the **STATE_ID_SK** to update its entry, otherwise, with the new **CREATED** and **LAST_UPDATED** appended, the state is inserted.

#### *2.2.2b COUNTY*

Data for COUNTY also comes from **STAGE.2B_USCOUNTIES**, yet, a county is only unique within its state, i.e., by **(STATE_ID, COUNTY_FIPS)** alone can the county be identified. In addition, COUNTY has a foreign key to **NDS.STATE** but notice that, for the latter did come from the same STAGE's table, their STATE_IDs will be identical. This enables the use of **Merge Join**, for slightly better performance than **Lookup**, as extra rows won't be produced.

From **STAGE.2B_USCOUNTIES**, trimmed rows are sorted by (STATE_ID, COUNTY_FIPS) to filter duplicate counties, and prepare for merge join with sorted states from NDS.STATE. Joining will net us the foreign key **STATE_ID_SK**, which is next combined with **COUNTY_FIPS** for the **Lookup** to decide whether to **update-or-insert** process.

#### *2.2.2c AIR_QUALITY*

Unlike the previous two, AIR_QUALITY's entries are sourced from STAGE.10_STATE_AQI, if we attempt to merge join for the foreign key to **NDS.COUNTY**, surplus rows will arise due to both sides' inconsistent data.

From STAGE.10_STATE_AQI, trimmed rows are sorted by everything **except SOURCE_ID** to eliminate redundancy. Since here we don't have state's id and county's FIPS, a **Lookup** is performed with **(STATE_NAME, COUNTY_NAME)** on a joined table of NDS's STATE and COUNTY to get the corresponding **COUNTY_ID**.

Now, owing to the STAGE's table not having an obvious natural key, it proves difficult to judge what qualifier to use for the **update-or-insert Lookup**. However, after some inspections, we've determined that (STATE_NAME, COUNTY_NAME, **DATE, DEFINING_PARAMETER**) can uniquely identify any entry. And in AIR_QUALITY, (STATE_NAME, COUNTY_NAME) translates to solely **COUNTY_ID**.

## 2.3 NDS to DDS

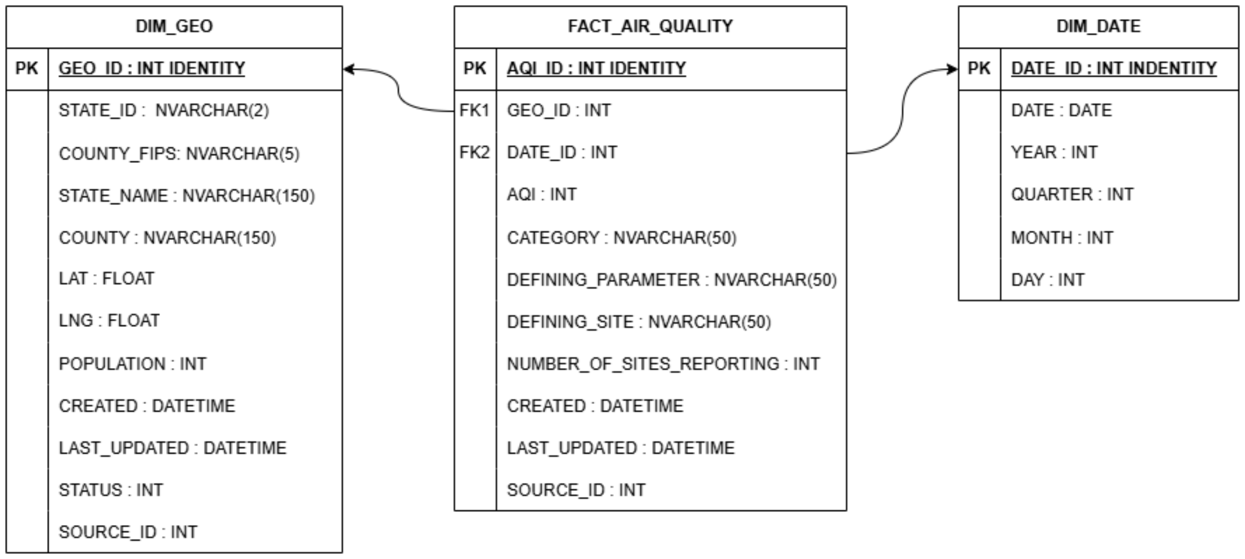


**Figure 223a.** Base DDS Star schema, with foreign keys and data types.

By comprehending requirements and de-normalizing NDS's, we arrive at this schema. It is designated as "**Base**" since here we don't include any potential **measure**, or **calculated member**. They will be documented later during Cube building, where we also delve into their related questions.

Currently, our DDS can satisfy both **Geography** and **Date** dimensions' hierarchies. Apart from the obvious merging of **STATE** into **COUNTY** to form DIM_GEO, we would like to note some degenerate dimensions (or **fact dimensions**) in FACT_AIR_QUALITY.

- **CATEGORY** and **DEFINING_PARAMETER**, per a lack of additional information, are put straight into the fact table to reduce joining works. They will be used to form the appropriate dimensions when configuring our cube.
- For **DEFINING_SITE**, despite technically also a fact dimension, is not demanded in any section of the project.

Additionally, the missing **STATUS** column in FACT_AIR_QUALITY is because we consider these facts to represent a moment in time, i.e., once an entry is recorded then it is final. To alter a past data would be highly uncommon.

### 2.3.1 DIM_GEO

The source for DIM_GEO is a joint of **NDS**'s **COUNTY** and **STATE**, where all rows with timestamp in the range of $(LSET, CET]$ and their relevant columns are extracted. Because changes are tracked separately for COUNTY and STATE, we must check if any of their **CREATED**s and **LAST_UPDATED**s satisfy.

Moving on to **Derived Column**, we append **STATUS** with value **1** (for SCD) and reassign LAST_UPDATED to be the latest date between COUNTY's and STATE's. To clarify, for this dimension, every row essentially represents a specific county (of a state), so if this county is updated in the NDS, only one DDS record is modified accordingly. But if it's an NDS state, then we want to alter all related DDS rows (i.e., all counties of that state).

Finally, we configure the **SCD** node with business keys **(STATE_ID, COUNTY_FIPS)**, and the following attributes,

- **Historical**: STATE_NAME, COUNTY.
- **Changing**: LAT, LNG, POPULATION.

For changing attributes, outdated records won't be updated, and for historical, a STATUS of **1** signifies **current** and **0 expired**. Inferred member support is left enabled (as default) for potential Early Arriving Facts.

### 2.3.2 DIM_DATE and FACT_AIR_QUALITY

Considering the time gaps that will appear if **DIM_DATE** is only made up of dates from Sources, leading to inflexibility, we choose to populate it through a stored procedure **USP_DIM_DATE_GEN**. This procedure, declared and ran along with the DDS's creation, generates all dates within 2 given years, breaks them down into date parts (year, quarter, month, day), and inserts the resulting rows into DIM_DATE.

When querying new rows (i.e., those having CREATED or LAST_UPDATED in $(LSET, CET]$) from **NDS.AIR_QUALITY** for the fact table, we must join it with **COUNTY** and **STATE** to also attain the **COUNTY_FIPS** and **STATE_ID**, both of which will form a pair to be used for looking up **GEO_ID** from DDS.DIM_GEO. Then similarly, we use **DATE** to look up **DATE_ID** from DDS.DIM_DATE and insert the completed entry.

# 3 SSAS Cube

When first initialized from the DDS, our Cube only has 2 valid measures which are,

- **AQI** – From the original AQI column of FACT_AIR_QUALITY, this serves a base to make other measure and is almost never used directly in query.
- **FACT AIR QUALITY Count** – Automatically generated by SSAS, it is simply a row count of the fact table. Since every entry in fact belongs to a date, the count can also double as a **Count of Days**, which appears in analysis questions (Q) 3, 4, 11, and 12.

Working through the requirements, we start to formulate the following, through the **New Measure** wizard,

- **Minimum** and **Maximum AQI** (Q1, 9) – Derived from AQI by taking min and max.

- **SUM AQI** and **SUM AQI SQUARED** – Derived from AQI and AQI_SQUARED by summing. Both are purely supports for **STDEV AQI**, which will be explained shortly.

## 3.1 Calculated Members

Q2, 3, 5, 6, 8, 9, and 10 all demand the AQI mean, but after ascertaining that using the **Average over time** does not produce desired results, we opt to manually compute **AVG AQI** by simply dividing **SUM AQI** by **FACT AIR QUALITY Count**, making sure to set the output as NULL if the count is 0 as well.

Now, **STDEV AQI** for Q2, and 9 is a tricky one since although the Stdev function does exist, not only is it operationally expensive but also requires a fixed Set_Expression, meaning no context-based filtered querying. To avoid these complexities, we choose to just apply the standard deviation formula,

$$\sqrt{\frac{\sum(X^2) - \frac{(\sum X)^2}{N}}{N}} = \left(\frac{SUM\ AQI\ SQUARED - \frac{SUM\ AQI \times SUM\ AQI}{Count}}{Count}\right)^{0.5}$$

Notice that we have **SUM AQI** and **FACT AIR QUALITY Count**, and power of 0.5 as there is no square root function, but we are still lacking **SUM AQI SQUARED**. For it is impossible to make a new measure from a calculated member, the sole route left for us is adding an **AQI_SQUARED** column to **DDS.FACT_AIR_QUALITY** and updating the **NDS_DDS** ETL package duly. Afterwards, the cube is reprocessed, and the New Measure wizard is once again employed to make SUM AQI SQUARED. Of course, we also treat the result when the count becomes 0 as NULL.

## 3.2 DIM GEO and DIM DATE

In **DIM GEO**, we define **Hierarchy GEO**'s level going from **STATE NAME** to **COUNTY** and link the **Attribute Relationships** suitably, their **RelationshipType**(s) are put as **Rigid**. For each attribute's properties, the **KeyColumns** is arranged to conform with the hierarchy, and **NameColumn** is set to said attribute's name.

Likewise, all steps are repeated for **DIM DATE** and **Hierarchy DATE** (with the level of YEAR, QUARTER, MONTH, DAY, and DATE). Furthermore, we expect SSAS to recognize this as a Time Dimension, so the property **Type** for DIM DATE is switched to **Time** and the attributes to the appropriate type that each represents.

Lastly, to tackle Q10, **DAYLIGHT_SAVING** is covered in this dimension, we also add the corresponding column to **DDS.DIM_DATE** with the **BIT** type and amend the date generating procedure to provide it data, being 1 (True) between Mar. 12th, 2023, and Nov. 5th, 2023.

### 3.3 Fact (Degenerate) dimensions

As mentioned in Sect. 2.3, out of the 3 dimensions, CATEGORY, DEFINING_PARAMETER, and DEFINING_SITE, only the former two are asked during analysis. Thus, they ascend to standalone dimensions **DIM CATEGORY** and **DIM DEFINING PARAMETER** with the key columns and attributes being merely themselves, and no related table.

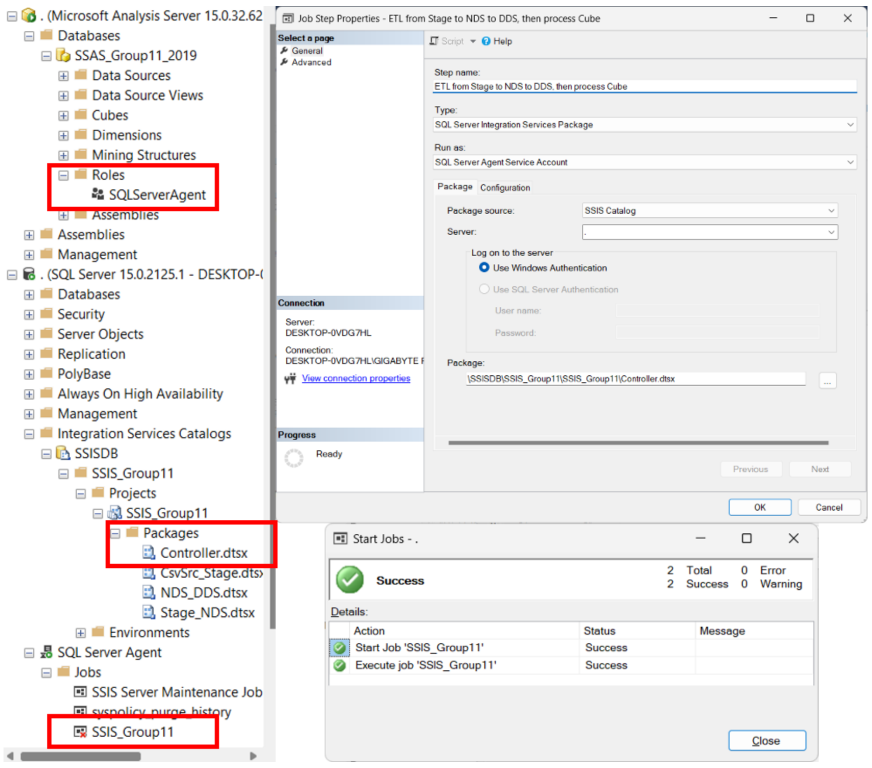
## 4 Scheduling

Scheduling can be facilitated with the help of **SSIS Catalog** and **SQL Server Agent**. Back in SSIS, we've created a new **Controller** package to guide the flow, starting from executing the packages **Stage_NDS** to **NDS_DDS** and ending with **processing** the whole **SSAS** database, through the Execute Package Task and Analysis Services Processing Task nodes.

- The exclusion of **CsvSrc_Stage** can be attributed to our defining Stage as a storage for messy and inconsistent data, compiled from many sources, before getting manually transformed, and prepared for loading to inner data stores. In that subsequent sources may vary in data quality, we should not automate this process.

After, the entire SSIS project is deployed to a premade catalog **SSISDB** in Integration Services Catalogs. We then make a new Job in SQL Server Agent with a singular step pointing to the Controller package, which now lies in the SSIS Catalog's SSISDB. Our job is scheduled to run Weekly at 12AM on Sunday.

Now, prior to starting the job, one important point is granting SSAS's **Process database** and **Read definition** privileges to the **NT SERVICE\SQLSERVERAGENT** as it will be the account attempting to process the Cube.



```
⊟ . (Microsoft Analysis Server 15.0.32.62
  ⊟ Databases
    ⊟ SSAS_Group11_2019
      ⊞ Data Sources
      ⊞ Data Source Views
      ⊞ Cubes
      ⊞ Dimensions
      ⊞ Mining Structures
      ⊟ Roles
          SQLServerAgent
      ⊞ Assemblies
  ⊞ Assemblies
  ⊞ Management
⊟ . (SQL Server 15.0.2125.1 - DESKTOP-(
  ⊞ Databases
  ⊞ Security
  ⊞ Server Objects
  ⊞ Replication
  ⊞ PolyBase
  ⊞ Always On High Availability
  ⊞ Management
  ⊟ Integration Services Catalogs
    ⊟ SSISDB
      ⊟ SSIS_Group11
        ⊟ Projects
          ⊟ SSIS_Group11
            ⊟ Packages
                Controller.dtsx
                CsvSrc_Stage.dtsx
                NDS_DDS.dtsx
                Stage_NDS.dtsx
        ⊞ Environments
  ⊟ SQL Server Agent
    ⊟ Jobs
        SSIS Server Maintenance Job
        syspolicy_purge_history
        SSIS_Group11
```

Job Step Properties - ETL from Stage to NDS to DDS, then process Cube

Step name: ETL from Stage to NDS to DDS, then process Cube
Type: SQL Server Integration Services Package
Run as: SQL Server Agent Service Account
Package source: SSIS Catalog
Server: .
Log on to the server: Use Windows Authentication
Package: \SSISDB\SSIS_Group11\SSIS_Group11\Controller.dtsx

Connection — Server: DESKTOP-0VDG7HL; Connection: DESKTOP-0VDG7HL\GIGABYTE
Progress: Ready

Start Jobs - .

Success — 2 Total, 2 Success, 0 Error, 0 Warning

| Action | Status | Message |
|---|---|---|
| Start Job 'SSIS_Group11' | Success | |
| Execute job 'SSIS_Group11' | Success | |

**Figure 4.** Left – All correctly setup components (in red) in SSMS's Object Explorer. Top Right – Configurations of the Job's ETL and Cube process step. Bottom Right – A successful run of the Job.

## 5 OLAP and Reporting

## 6 Data Mining

## Conclusion

## References

- **Slides, Resources, and Guides** provided during both Lecture and Lab sessions.
- Past lecture sessions and project of this course in 2023: https://www.youtube.com/@BinhPham-lh4rv

## 2 Data Warehouse

### 2.1 Source (.csv) to Stage

- Load files from folder to table: https://youtu.be/OkA85uaUiM0?si=spEKdtvuqHnB1Z42

### 2.2 Stage to NDS

- Refer to **SQL/3_clean_data.sql** file for certain links and their usages.
- On comparing Merge join and Lookup: https://stackoverflow.com/a/6739121
- On inner join returning more rows than expected: https://blog.sqlauthority.com/2012/02/09/sql-server-inner-join-returning-more-records-than-exists-in-table/
- On uniquely identifying records in air quality data: https://aqs.epa.gov/aqsweb/airdata/FileFormats.html#_monitors

### 2.3 NDS to DDS

- On updating fact tables: https://stackoverflow.com/a/67745448
- On getting all dates between two dates in MSSQL: https://stackoverflow.com/a/23291758
- On Unknown and Inferred Members, Early Arriving Fact or Late Arriving Dimension: https://youtu.be/weNidVsl6WQ?si=_iH7XTDCMT6IjJEs

## 3 SSAS Cube

- On SSAS's common warnings: https://www.sqlshack.com/warnings-in-ssas-cubes/
- On creating Average/Mean measure: https://insightextractor.com/2014/08/27/how-to-create-an-average-aggregation-in-sql-server-analysis-services/
- On creating Standard Deviation measure: https://dba.stackexchange.com/a/154385

## 4 Scheduling

- On deploying SSIS packages and scheduling: https://youtu.be/eNxbMwUGl1g?si=hdRvZUvIer9LAJkw&t=9423
- On scheduling the processing of SSAS Cube: https://youtu.be/UebBb64MAT4?si=Qp_F7DbfArwShZIe