**Documentation: US Market Analysis Data Project**

**Overview**

The goal of this project is to download, process, and analyse FOIA data related to market analysis from the ICE FOIA Library. The data, spanning multiple years, will be loaded into SQL Server tables for analytical purposes. The process involves creating year-based tables, generating fact tables, and creating views for comprehensive analysis.

**Step 1: Download Data**

1. **Navigate to Data Source:**
   * Go to the [ICE FOIA Library](https://www.ice.gov/foia/library).
   * Select **Category = Monthly FOI Releases**.
2. **Filter and Download:**
   * Filter records related to FOI request 2023-ICFO-43657.
   * Download all files with the following criteria:
     + **FY (Financial Year)**: 2017, 2018, 2019, 2020, 2021, 2022, 2023
     + **Part**: All available parts.

**Step 2: Load Data Using C#**

1. **Prepare Data Files:**
   * Convert downloaded files into CSV format if required.
2. **Load Data into Development SQL Server:**
   * **Server:** smineudevsqls01.9bdb410b845c.database.windows.net
   * **Database:** BI\_DEV
   * Use a C# script to load each file into separate year-based tables in SQL Server.
   * Create tables for each year to manage performance and efficiency.  
       
       
     A screenshot of a computer

     Description automatically generated
3. **Verify and Validate Data in Development Server:**
   * Ensure that data is correctly loaded and matches the expected format.
   * Run validation queries to ensure data integrity.
4. **Transfer Data to Production SQL Server:**
   * **Server:** smineuprdsqls01.09ed840669e8.database.windows.net
   * **Database:** BI
   * Use data export/import tools or SQL Server Integration Services (SSIS) to transfer data from BI\_DEV to BI.

**Step 3: Create Year-Based Tables**

1. **Table Naming Convention:**
   * Create tables for each year following the pattern:
     + uma.US\_Market\_Analysis\_2017
     + uma.US\_Market\_Analysis\_2018
     + uma.US\_Market\_Analysis\_2019
     + ...
     + uma.US\_Market\_Analysis\_2023

A screenshot of a computer

Description automatically generated

1. **Schema:**
   * The schema for each year-based table should match the structure of the data files.

**Step 4: Data Transformation**

1. **Convert String Values:**
   * Convert string values in each year-based table to integer values according to the uma.KeyValues table.
   * Update tables to replace string representations with their corresponding integer values.

**Step 5: Create Fact Tables**

1. **Fact Table: Admissions**

sql

Copy code

-- Drop the table if it already exists

DROP TABLE IF EXISTS fact.Admissions;

-- Create the fact table with a surrogate primary key

CREATE TABLE fact.Admissions (

ID INT IDENTITY(1,1) PRIMARY KEY,

MarketYear INT NOT NULL,

School\_Name INT NULL,

Campus\_State INT NULL,

Country\_of\_Birth INT NULL,

Status\_Code INT NULL,

Education\_Level INT NULL,

Total\_Admissions INT NOT NULL,

Min\_Record\_ID BIGINT NOT NULL,

Max\_Record\_ID BIGINT NOT NULL

);

-- Add indexes to columns

CREATE INDEX idx\_MarketYear ON fact.Admissions(MarketYear);

CREATE INDEX idx\_School\_Name ON fact.Admissions(School\_Name);

CREATE INDEX idx\_Campus\_State ON fact.Admissions(Campus\_State);

CREATE INDEX idx\_Country\_of\_Birth ON fact.Admissions(Country\_of\_Birth);

CREATE INDEX idx\_Status\_Code ON fact.Admissions(Status\_Code);

CREATE INDEX idx\_Education\_Level ON fact.Admissions(Education\_Level);

-- Insert data into the fact table

INSERT INTO fact.Admissions (

MarketYear,

School\_Name,

Campus\_State,

Country\_of\_Birth,

Status\_Code,

Education\_Level,

Total\_Admissions,

Min\_Record\_ID,

Max\_Record\_ID

)

SELECT

2017 AS MarketYear,

School\_Name,

Campus\_State,

Country\_of\_Birth,

Status\_Code,

Student\_Edu\_Level\_Desc AS Education\_Level,

COUNT(PK\_ID) AS Total\_Admissions,

MIN(PK\_ID) AS Min\_Record\_ID,

MAX(PK\_ID) AS Max\_Record\_ID

FROM [BI\_UMA].[uma].[US\_Market\_Analysis\_2017]

GROUP BY

School\_Name,

Campus\_State,

Country\_of\_Birth,

Status\_Code,

Student\_Edu\_Level\_Desc

UNION ALL

-- Repeat for each year up to 2023

1. **Fact Table: Employment**

sql

Copy code

-- Drop the table if it already exists

DROP TABLE IF EXISTS fact.Employment;

-- Create the fact table

CREATE TABLE fact.Employment (

ID INT IDENTITY(1,1) PRIMARY KEY,

MarketYear INT NOT NULL,

School\_Name INT NULL,

Campus\_State INT NULL,

Country\_of\_Birth INT NULL,

Employer\_Name INT NULL,

Employer\_City INT NULL,

Employer\_State INT NULL,

Employment\_OPT\_Type INT NULL,

Total\_Admissions INT NOT NULL,

Min\_Record\_ID BIGINT NOT NULL,

Max\_Record\_ID BIGINT NOT NULL

);

-- Add indexes to every column

CREATE INDEX idx\_School\_Name ON fact.Employment(School\_Name);

CREATE INDEX idx\_Campus\_State ON fact.Employment(Campus\_State);

CREATE INDEX idx\_Country\_of\_Birth ON fact.Employment(Country\_of\_Birth);

CREATE INDEX idx\_Employer\_Name ON fact.Employment(Employer\_Name);

CREATE INDEX idx\_Employer\_City ON fact.Employment(Employer\_City);

CREATE INDEX idx\_Employer\_State ON fact.Employment(Employer\_State);

CREATE INDEX idx\_Employment\_OPT\_Type ON fact.Employment(Employment\_OPT\_Type);

-- Insert data into the fact table

INSERT INTO fact.Employment (

MarketYear,

School\_Name,

Campus\_State,

Country\_of\_Birth,

Employer\_Name,

Employer\_City,

Employer\_State,

Employment\_OPT\_Type,

Total\_Admissions,

Min\_Record\_ID,

Max\_Record\_ID

)

SELECT

2017 AS MarketYear,

School\_Name,

Campus\_State,

Country\_of\_Birth,

Employer\_Name,

Employer\_City,

Employer\_State,

Employment\_OPT\_Type,

COUNT(PK\_ID) AS Total\_Admissions,

MIN(PK\_ID) AS Min\_Record\_ID,

MAX(PK\_ID) AS Max\_Record\_ID

FROM [BI\_UMA].[uma].[US\_Market\_Analysis\_2017]

GROUP BY

School\_Name,

Campus\_State,

Country\_of\_Birth,

Employer\_Name,

Employer\_City,

Employer\_State,

Employment\_OPT\_Type

UNION ALL

-- Repeat for each year up to 2023  
  
 

**Step 6: Create Views**

1. **View: Admissions**

sql

Copy code

-- Drop the view if it already exists

IF OBJECT\_ID('vw.Admissions', 'V') IS NOT NULL

BEGIN

DROP VIEW vw.Admissions;

END

GO

-- Create the view

CREATE VIEW vw.Admissions AS

SELECT

f.ID,

f.MarketYear,

School\_Name = kv1.KeyValue,

Campus\_State = kv2.KeyValue,

Country\_of\_Birth = kv3.KeyValue,

Status\_Code = kv4.KeyValue,

Education\_Level = kv5.KeyValue,

f.Total\_Admissions,

f.Min\_Record\_ID,

f.Max\_Record\_ID

FROM

fact.Admissions f

LEFT JOIN uma.KeyValues kv1 ON kv1.DataId = f.School\_Name

LEFT JOIN uma.KeyValues kv2 ON kv2.DataId = f.Campus\_State

LEFT JOIN uma.KeyValues kv3 ON kv3.DataId = f.Country\_of\_Birth

LEFT JOIN uma.KeyValues kv4 ON kv4.DataId = f.Status\_Code

LEFT JOIN uma.KeyValues kv5 ON kv5.DataId = f.Education\_Level;

GO  
  
  
2.**View: Employment**

sql

Copy code

-- Drop the view if it already exists

IF OBJECT\_ID('vw.Employment', 'V') IS NOT NULL

BEGIN

DROP VIEW vw.Employment;

END

GO

-- Create the view

CREATE VIEW vw.Employment AS

SELECT

f.ID,

f.MarketYear,

School\_Name = kv1.KeyValue,

Campus\_State = kv2.KeyValue,

Country\_of\_Birth = kv3.KeyValue,

Employer\_Name = kv4.KeyValue,

Employer\_City = kv5.KeyValue,

Employer\_State = kv6.KeyValue,

Employment\_OPT\_Type = kv7.KeyValue,

f.Total\_Admissions,

f.Min\_Record\_ID,

f.Max\_Record\_ID

FROM

fact.Employment f

LEFT JOIN uma.KeyValues kv1 ON kv1.DataId = f.School\_Name

LEFT JOIN uma.KeyValues kv2 ON kv2.DataId = f.Campus\_State

LEFT JOIN uma.KeyValues kv3 ON kv3.DataId = f.Country\_of\_Birth

LEFT JOIN uma.KeyValues kv4 ON kv4.DataId = f.Employer\_Name

LEFT JOIN uma.KeyValues kv5 ON kv5.DataId = f.Employer\_City

LEFT JOIN uma.KeyValues kv6 ON kv6.DataId = f.Employer\_State

LEFT JOIN uma.KeyValues kv7 ON kv7.DataId = f.Employment\_OPT\_Type;

GO  
  
 A close up of words

Description automatically generated

**Step 7: Verify and Test**

1. **Check Record Counts:**
   * Verify record counts from fact tables and views to ensure data integrity.

sql

Copy code

SELECT COUNT(1) FROM fact.Admissions;

SELECT COUNT(1) FROM vw.Admissions;

SELECT COUNT(1) FROM fact.Employment;

SELECT COUNT(1) FROM vw.Employment;  
  
**Step 8: Review Data Accuracy:**

* + Ensure that the data in the views matches expected values and is correctly transformed.
  + Perform test queries to validate the accuracy of the views.

**Appendix**

1. **Table and Column Definitions:**

**Year-Based Tables:**

* + **Schema:** [uma].[US\_Market\_Analysis\_YYYY]
  + **Columns:** Match the fields from the downloaded files, including necessary identifiers.

**Fact Tables:**

* + **fact.Admissions:**
    - ID: Surrogate primary key.
    - MarketYear: Year of the data.
    - School\_Name, Campus\_State, Country\_of\_Birth, Status\_Code, Education\_Level: Integer representations of categorical data.
    - Total\_Admissions: Total count of admissions.
    - Min\_Record\_ID, Max\_Record\_ID: Range of record IDs for the admissions.
  + **fact.Employment:**
    - ID: Surrogate primary key.
    - MarketYear: Year of the data.
    - School\_Name, Campus\_State, Country\_of\_Birth, Employer\_Name, Employer\_City, Employer\_State, Employment\_OPT\_Type: Integer representations of categorical data.
    - Total\_Admissions: Total count of admissions.
    - Min\_Record\_ID, Max\_Record\_ID: Range of record IDs for the employment records.

1. **Views:**
   * **vw.Admissions:** Displays admissions data with string values replaced by corresponding key values from uma.KeyValues.
   * **vw.Employment:** Displays employment data with string values replaced by corresponding key values from uma.KeyValues.

This documentation provides a structured approach for the entire process, from data download to analysis. Each step includes SQL commands and guidelines for creating tables, transforming data, and setting up views. Make sure to adjust scripts as necessary for different years and specific data nuances.