Åland Data Integration Project

Project Overview

This project involves building a data integration pipeline and data warehouse for analyzing tourism, grocery sales, and demographic data from Åland municipalities. The goal is to create a comprehensive data warehouse that enables analytical insights across multiple data domains.

Purpose

Build a data pipeline that integrates:

- **Tourism data**: Monthly visitor statistics and revenue by municipality (CSV format)
- **Grocery sales data**: Daily product-level sales transactions from a grocery store chain (JSON format)
- **Demographic data**: Population statistics from ÅSUB (Ålands statistik- och utredningsbyrå) via REST API

The integrated data will be stored in a SQL database using a star schema design, enabling powerful analytical queries and visualizations.

Data Sources

1. Tourism Data (CSV)

- **Location**: `data/tourism/tourism_data.csv`
- **Format**: CSV with monthly granularity
- **Time Period**: January 2000 to December 2025
- **Content**:
  - Municipality-level monthly statistics
  - Visitor counts, accommodation types, origin countries
  - Tourism revenue in EUR
- **Schema**: See `data/DATA_SCHEMA.md` for detailed field descriptions

2. Grocery Sales Data (JSON)

- **Location**: `data/grocery/`

- **Format**: Multiple JSON files
  - `stores.json` : Store and municipality reference data
  - `products.json` : Product catalog with categories and pricing
  - `grocery_sales_*.json` : Daily product-level sales transactions (one file per year)
- **Time Period**: January 1, 2000 to December 31, 2025 (daily granularity)
- **Content**:
  - Daily sales at product level
  - Multiple stores across 16 Åland municipalities
  - Sales amounts and units sold
- **Schema**: See `data/DATA_SCHEMA.md` for detailed field descriptions

3. Demographic Data (REST API)

- **Source**: ÅSUB (Ålands statistik- och utredningsbyrå)
- **Endpoint**: Population Statistics API
- **Content**: Population statistics by municipality, year, age, and gender
- **Time Period**: 1975-2024 (students should fetch relevant years for their analysis)
- **Format**: PX-Web API (students will need to query this REST endpoint)

Requirements

Analytical Questions

The data warehouse must be designed to answer the following questions through SQL queries:

1. **How has sales per capita changed over time?**
   - Requires joining grocery sales, population data, and time dimensions
   - Calculate sales per capita by municipality and time period
2. **Is there a correlation between sales and tourism statistics?**
   - Analyze relationship between tourism visitor counts/revenue and grocery sales
   - Consider temporal alignment (monthly tourism vs daily sales)
3. **Which municipalities have the highest sales per capita, and how does this relate to tourism?**
   - Compare sales per capita across municipalities
   - Investigate correlation with tourism metrics
4. **What are the seasonal patterns in both tourism and grocery sales?**
   - Identify seasonal trends and patterns

- Compare tourism seasonality with sales seasonality

5. **How do different product categories perform across municipalities?**
    - Analyze product category sales by location
    - Identify regional preferences or patterns
6. **Which stores are the top performers, and what factors contribute to their success?**
    - Store-level performance analysis
    - Investigate factors like location, municipality, size
7. **How has tourism revenue changed over time by municipality?**
    - Time-series analysis of tourism trends
    - Municipality-level comparisons
8. **What is the relationship between population size and total grocery sales?**
    - Demographic correlation analysis
    - Population growth vs sales growth
9. **Are there differences in sales patterns between weekdays and weekends?**
    - Temporal pattern analysis
    - Day-of-week effects on sales
10. **How do product category sales correlate with tourism seasons?**
    - Cross-domain correlation analysis
    - Seasonal product preferences during tourism peaks

Important: The data warehouse should be designed to answer **all** of the above questions through SQL queries. However, students are only required to create **visualizations for 2 of these questions**. The choice of which 2 questions to visualize is up to the students.

Project Goals

Primary Objectives

1. **Build a Data Pipeline**
    - Extract data from CSV files (tourism data)
    - Extract data from JSON files (grocery sales data)
    - Extract data from ÅSUB REST API (demographic data)
    - Transform and load data into a SQL database

2. **Implement Data Warehouse in SQL Database**
   - Choose a SQL database (PostgreSQL, MySQL, SQLite, SQL Server, etc.)
   - Implement medallion architecture:
     - **Gold Layer**: **Required**
       - Final transformed data in star schema format
     - **Bronze Layer**: **Recommended**
       - Raw data ingestion layer (highly recommended for staging and data lineage)
     - **Silver Layer**: **Optional**
       - Cleaned and validated data stage (students may choose to implement this for practice, but it is not required)

3. **Design Star Schema**
   - Create fact tables for measurable events (sales, tourism)
   - Create dimension tables (time, location/municipality, products, stores, demographics)
   - Ensure proper relationships and referential integrity

4. **Create Visualizations**
   - Build visualizations for 2 selected analytical questions
   - Use any visualization tool (Tableau, Power BI, Python/Matplotlib, R, etc.)

Design Process

Students should start with a thorough design process before implementation:

1. **Data Analysis**
   - Understand the structure and content of each data source
   - Identify relationships and keys for joining data
   - Document data quality issues or considerations

2. **Star Schema Design**
   - Identify facts (measurable events) and dimensions (descriptive attributes)
   - Design fact tables with appropriate grain (level of detail)
   - Design dimension tables with all necessary attributes
   - Create entity-relationship diagrams
   - Consider aggregation strategies

3. **ETL/ELT Design**
   - Plan extraction methods for each data source
   - Design transformation logic (cleaning, joining, aggregating)

- Plan loading strategy (see Loading Patterns section - full refresh is recommended for this project)
- Consider data refresh frequency

4. **Query Design**

    - Plan SQL queries for each analytical question
    - Identify required joins and aggregations
    - Consider performance optimization

5. **Visualization Planning**

    - Select 2 questions to visualize
    - Design appropriate chart types
    - Plan data preparation for visualizations

Technical Architecture

Medallion Architecture

The project should follow the medallion architecture pattern:

- **Bronze Layer (Recommended)**: Raw, unprocessed data as ingested from sources
    - **Highly recommended** for this project as it provides a staging area for raw data ingestion
    - Allows for data lineage tracking and easier debugging
    - Enables reprocessing of data without re-extracting from source systems
- **Silver Layer (Optional)**: Cleaned and validated data, ready for transformation
- **Gold Layer (Required)**: Final transformed data in star schema, optimized for analytics

Note: The **gold layer** with star schema is **mandatory**. The **bronze layer** is **highly recommended** as it provides a good practice for data ingestion and staging. The silver layer is optional, but students are encouraged to implement it if they want to practice the full medallion architecture.

Star Schema Design

The gold layer should implement a star schema with:

- **Fact Tables**:

    - Sales fact (grocery sales transactions)
    - Tourism fact (monthly tourism statistics)
    - Potentially combined fact tables depending on design choices

- **Dimension Tables**:

    - Time dimension (date, month, quarter, year, day of week, etc.)
    - Location/Municipality dimension (municipality codes, names, population data)
    - Product dimension (product details, categories, suppliers)
    - Store dimension (store information, locations)
    - Demographics dimension (population statistics by time and location)

Data Flow

```
Data Sources → Extraction → [Bronze Layer] → [Silver Layer] → Gold Layer
     ↓              ↓              ↓                ↓
  CSV/JSON      ETL/ELT      Raw Data      Cleaned Data        Transforme
  REST API      Process
```

Loading Patterns

When loading data into the data warehouse, there are two main approaches:

- **Full Refresh (Truncate and Load)**
    - Delete all existing data from target tables
    - Load all data from source systems
    - Simple to implement and ensures data consistency
    - Suitable for smaller datasets or when historical data changes are rare
    - May be slower for large datasets
- **Incremental Refresh (Upsert/Merge)**
    - Only load new or changed records since the last load
    - Requires tracking changes (e.g., timestamps, change data capture)
    - More complex to implement but more efficient for large datasets
    - Requires careful handling of updates and deletes

For this project: A **full refresh** approach is sufficient and recommended. Since this is a learning project with sample data, students should focus on building a working pipeline rather than optimizing for incremental loads. Full refresh is simpler to implement, easier to debug, and adequate for the dataset sizes in this project.

SQL Database

Students can choose any SQL database that supports:

- Standard SQL syntax

- Foreign key constraints
- Indexing capabilities
- Common data types (dates, numbers, strings)

Recommended options:

- PostgreSQL
- MySQL/MariaDB
- SQLite (for simplicity, though limited for large datasets)
- SQL Server
- Oracle

Deliverables

1. **Data Pipeline Code**
    - Scripts/code for extracting data from all sources
    - Transformation logic
    - Loading scripts
2. **Database Schema**
    - SQL DDL scripts for creating tables
    - Documentation of the star schema design
    - Entity-relationship diagrams (recommended)
3. **SQL Queries (OPTIONAL)**
    - Queries that answer all 10 analytical questions
    - Well-documented and organized
    - Optional, but good for learning about writing analytical queries
4. **Visualizations**
    - 2 visualizations answering selected questions
    - Clear titles, labels, and insights
5. **Documentation (OPTIONAL)**
    - README explaining the project structure
    - Design decisions and rationale
    - Instructions for running the pipeline
    - Data dictionary
    - Optional

Getting Started

1. **Review the Data**
   - Examine the sample data files in `data/` directory
   - Read `data/DATA_SCHEMA.md` for detailed schema information
   - Explore the ÅSUB API endpoint
2. **Start with Design**
   - Don't jump straight into coding
   - Create a comprehensive design document
   - Design your star schema on paper/diagram first
   - Plan your ETL/ELT process
3. **Implement Incrementally**
   - Start with one data source
   - Build and test your pipeline
   - Add additional sources one at a time
   - Iterate and refine
4. **Test Your Queries**
   - Verify all analytical questions can be answered
   - Check data quality and completeness
   - Optimize query performance
5. **Create Visualizations**
   - Select 2 questions that interest you
   - Design clear, informative visualizations
   - Ensure visualizations tell a story

Resources

- **Data Schema Documentation**: `data/DATA_SCHEMA.md`
- **ÅSUB Statistics Portal**: https://pxweb.asub.ax/
- **Sample Data**: Located in `data/` directory

Notes

- The sample data is generated with realistic patterns and correlations (e.g., grocery sales correlate with tourism)
- Municipality data should be fetched from the ÅSUB API, not from the sample data files
- Focus on building a robust, well-designed data warehouse rather than just getting data loaded
- Design decisions should be documented and justified
- Code should be clean, well-commented, and maintainable

Good luck with your project!