Data Schema Documentation

Overview

This document provides detailed information about the data files, schemas, field descriptions, and contents generated for the Åland sample data integration project.

The data consists of two main systems:

1. **Tourism System** - Monthly CSV data
2. **Grocery Store Chain Sales System** - Daily JSON data with three separate files

Both systems can be linked via Åland municipality identifiers for data integration testing.

Directory Structure

The data files are organized in the following structure:

```
data/
├── DATA_SCHEMA.md                  # This file (for easy reference)
├── tourism/
│   └── tourism_data.csv            # Monthly tourism statistics
└── grocery/
    ├── stores.json                 # Store and municipality reference dat
    ├── products.json               # Product catalog
    └── grocery_sales_*.json       # Daily product-level sales (one file p
```

Tourism Data Schema (CSV)

File Location: `data/tourism/tourism_data.csv`

File Format: CSV with header row

File Name Pattern: `tourism_data.csv`

Record Count: 4,992 records (approximately 300 months × 16 municipalities)

Time Period: January 2000 to December 2025 (monthly granularity)

Schema

Column Name	Data Type	Description	Example
municipality_code	String	Unique identifier for Åland municipality	"MH"
municipality_name	String	Full name of the municipality	"Mariehamn"
year	Integer	Year of the record	2023
month	Integer	Month number (1-12)	7
date	String (YYYY-MM-DD)	First day of the month	"2023-07-01"
visitor_count	Integer	Total number of visitors for the month	12500
accommodation_type	String	Type of accommodation	"hotel", "guesthouse", "camping"
origin_country	String	Primary origin country of visitors	"Sweden", "Finland", "Norway", etc.
revenue	Float	Tourism revenue in EUR	125000.50

Notes

- Each row represents one month for one municipality
- Multiple accommodation types may be represented as separate rows or aggregated
- Revenue is in EUR (Euro)
- Seasonal patterns: Summer months (June-August) have higher visitor counts
- Long-term trends: Gradual growth over the 25-year period

Sample Data

```
municipality_code,municipality_name,year,month,date,visitor_count,accommo
MH,Mariehamn,2023,7,2023-07-01,12500,hotel,Sweden,125000.50
JO,Jomala,2023,7,2023-07-01,6500,guesthouse,Finland,78000.00
```

Stores Data Schema (JSON)

File Location: `data/grocery/stores.json`

File Format: JSON array of objects

Record Count: 15 stores (variable based on municipality population)

Schema

Field Name	Data Type	Description	Example
store_id	String	Unique identifier for the grocery store	"STORE_001"
municipality_code	String	Unique identifier for Åland municipality	"MH"
municipality_name	String	Full name of the municipality	"Mariehamn"
store_location	String	Physical address or location name	"Main Street 15"
store_name	String	Name of the store	"Åland Grocery Mariehamn"

Notes

- Reference data file containing all store locations
- Links stores to municipalities
- Used as lookup for grocery_sales.json
- Number of stores per municipality varies based on population
- **Mariehamn**: 6 stores, **Jomala**: 4 stores, **Finström**: 3 stores (more stores than other municipalities)

Sample Data

```
[
  {
    "store_id": "STORE_001",
    "municipality_code": "MH",
    "municipality_name": "Mariehamn",
    "store_location": "Main Street 15",
    "store_name": "Åland Grocery Mariehamn"
  }
]
```

Products Data Schema (JSON)

File Location: `data/grocery/products.json`

File Format: JSON array of objects

Record Count: 37 products (variable product catalog)

Schema

Field Name	Data Type	Description	Example
product_id	String	Unique identifier for the product	"PROD_001"
product_name	String	Name of the product	"Organic Tomatoes"
product_category	String	Category of the product	"produce", "dairy", "meat", "beverages", "bakery", "frozen", "canned"
unit_price	Float	Price per unit in EUR	3.50
unit_type	String	Unit of measurement	"kg", "liter", "piece", "pack"
supplier	String	Product supplier name	"Local Farm Co"

Notes

- Product catalog/reference data
- Contains product details and categorization
- Used as lookup for grocery_sales.json
- Products span multiple categories

Sample Data

```
[
  {
    "product_id": "PROD_001",
    "product_name": "Organic Tomatoes",
    "product_category": "produce",
    "unit_price": 3.50,
    "unit_type": "kg",
    "supplier": "Local Farm Co"
  }
]
```

Grocery Sales Data Schema (JSON)

File Location: `data/grocery/grocery_sales_*.json` (one file per year: 2000-2025)

File Format: JSON array of objects

File Name Pattern: `grocery_sales_YYYY.json` (e.g., `grocery_sales_2023.json`)

Record Count: Very large (approximately 3.6 million total records)

- Daily product-level transactions
- 9,131 days (2000-01-01 to 2025-12-31)
- Multiple stores × multiple products per day

Time Period: January 1, 2000 to December 31, 2025 (daily granularity)

Schema

Field Name	Data Type	Description	Example
store_id	String	Foreign key to stores.json	"STORE_001"
product_id	String	Foreign key to products.json	"PROD_001"
date	String (YYYY-MM-DD)	Date of the sale	"2023-07-15"
sales_amount	Float	Total sales revenue in EUR	1250.75
units_sold	Integer	Total number of units sold	357

Notes

- Each object represents sales of one specific product at one store for one day
- Product-level granularity (not category-level)
- Daily data (not monthly)
- Year and month can be extracted from date field
- Links to stores.json via `store_id`
- Links to products.json via `product_id`
- Revenue is in EUR (Euro)
- Not all products are sold every day (60-80% of products per day)

Sample Data

```
[
  {
    "store_id": "STORE_001",
    "product_id": "PROD_001",
    "date": "2023-07-15",
    "sales_amount": 1250.75,
    "units_sold": 357
  }
]
```

Linkage Keys

Tourism to Grocery Sales

- **Join Key**: `municipality_code` + `year` + `month`
- Tourism is monthly, grocery sales are daily
- To join:
    1. Extract `municipality_code` from `stores.json` using `store_id` from grocery sales
    2. Extract `year` and `month` from `date` field in grocery sales
    3. Join with tourism data on `municipality_code` + `year` + `month`
- Aggregate daily grocery sales by month for correlation analysis

Grocery Sales Internal Links

- **store_id** → `stores.json` (to get `municipality_code`, `municipality_name`)
- **product_id** → `products.json` (to get `product_name`, `product_category`, `unit_price`)
- **date** field contains full date (YYYY-MM-DD), year and month can be extracted

Composite Keys

- **Tourism**: `municipality_code` + `year` + `month`
- **Grocery Sales**: `store_id` + `product_id` + `date`

Data Characteristics

Time Period

- **Start Date**: January 1, 2000
- **End Date**: December 31, 2025

- **Total Duration**: 25 years, 9,131 days

Record Counts

- **Tourism Data**: ~4,992 records (300 months × 16 municipalities)
- **Stores**: 15 stores
- **Products**: 37 products
- **Grocery Sales**: ~3.6 million records (daily product-level transactions)

Granularity

- **Tourism**: Monthly
- **Grocery Sales**: Daily at product level

Data Quality

- Consistent municipality identifiers across both systems
- Seasonal patterns: Tourism peaks in summer months, grocery sales correlate accordingly
- Long-term trends: Gradual growth over 25-year period (2% per year for tourism, 1.5% per year for grocery)
- Day-of-week patterns: Weekend sales (Friday-Sunday) are 30% higher than weekdays
- Product-level granularity: Individual product sales, not aggregated by category

Åland Municipalities

The data covers all 16 Åland municipalities with their codes:

Code	Municipality Name	Population (approx)
MH	Mariehamn	11,866
JO	Jomala	5,642
FI	Finström	2,564
LE	Lemland	2,203
HA	Hammarland	1,615
SA	Saltvik	1,801
SU	Sund	1,000
GE	Geta	508
VA	Vårdö	450

Code	Municipality Name	Population (approx)
FO	Föglö	567
KU	Kumlinge	315
BR	Brändö	488
KO	Kökar	252
SO	Sottunga	101
EC	Eckerö	952
LU	Lumparland	384

Example Queries

Example 1: Join Grocery Sales with Store Information

```
import json

# Load stores
with open('data/grocery/stores.json', 'r') as f:
    stores = {s['store_id']: s for s in json.load(f)}

# Load sales
with open('data/grocery/grocery_sales_2023.json', 'r') as f:
    sales = json.load(f)

# Join
for sale in sales:
    store = stores[sale['store_id']]
    print(f"Store: {store['store_name']} in {store['municipality_name']}"
    print(f"Sales: {sale['sales_amount']} EUR on {sale['date']}")
```

Example 2: Aggregate Grocery Sales by Municipality and Month

```
import json
from collections import defaultdict

# Load stores
with open('data/grocery/stores.json', 'r') as f:
```

```python
    stores = {s['store_id']: s for s in json.load(f)}

# Load sales for a year
with open('data/grocery/grocery_sales_2023.json', 'r') as f:
    sales = json.load(f)

# Aggregate by municipality and month
monthly_sales = defaultdict(float)

for sale in sales:
    store = stores[sale['store_id']]
    municipality = store['municipality_code']
    year, month = sale['date'][:7].split('-')  # Extract YYYY-MM
    key = (municipality, year, month)
    monthly_sales[key] += sale['sales_amount']

# Print results
for (municipality, year, month), total in sorted(monthly_sales.items()):
    print(f"{municipality} {year}-{month}: {total:.2f} EUR")
```

Example 3: Correlate Tourism with Grocery Sales

```python
import csv
import json
from collections import defaultdict

# Load tourism data
tourism = {}
with open('data/tourism/tourism_data.csv', 'r') as f:
    reader = csv.DictReader(f)
    for row in reader:
        key = (row['municipality_code'], row['year'], row['month'])
        tourism[key] = {
            'visitor_count': int(row['visitor_count']),
            'revenue': float(row['revenue'])
        }

# Load stores
with open('data/grocery/stores.json', 'r') as f:
    stores = {s['store_id']: s for s in json.load(f)}

# Load and aggregate grocery sales
```

```
with open('data/grocery/grocery_sales_2023.json', 'r') as f:
    sales = json.load(f)

monthly_grocery = defaultdict(float)
for sale in sales:
    store = stores[sale['store_id']]
    municipality = store['municipality_code']
    year, month = sale['date'][:7].split('-')
    key = (municipality, year, month)
    monthly_grocery[key] += sale['sales_amount']

# Correlate
for key in sorted(tourism.keys()):
    if key in monthly_grocery:
        print(f"{key[0]} {key[1]}-{key[2]}:")
        print(f"  Tourism: {tourism[key]['visitor_count']} visitors, {tou
        print(f"  Grocery: {monthly_grocery[key]:.2f} EUR")
        print()
```

Example 4: Product Category Analysis

```
import json

# Load products
with open('data/grocery/products.json', 'r') as f:
    products = {p['product_id']: p for p in json.load(f)}

# Load sales
with open('data/grocery/grocery_sales_2023.json', 'r') as f:
    sales = json.load(f)

# Aggregate by category
category_sales = defaultdict(float)
for sale in sales:
    product = products[sale['product_id']]
    category = product['product_category']
    category_sales[category] += sale['sales_amount']

# Print results
for category, total in sorted(category_sales.items(), key=lambda x: x[1],
    print(f"{category}: {total:.2f} EUR")
```