Assignment 2 Report

Air Quality Data Warehouse

188.429 Business Intelligence (VU 4.0), 2025W
Group 020

Student A: Muhammad Sajid Bashir (52400204)
Student B: Eman Shahin (12432813)

2025-11-29

1. ETL Summary

Dimension Population

`dim_timemonth` (ETL01): Generated using `generate_series()` for 2023-2024. Includes year/quarter/month numbers, month names, and days_in_month. Primary key format: YYYYMM (e.g., 202401).

`dim_city` (ETL02): Loaded from staging with region classification (Central/Eastern/Western Europe). Used `DISTINCT` on (country, city) to eliminate duplicates. Auto-generated surrogate keys.

`dim_param` (ETL03): Loaded parameters with category/purpose groupings. **Note:** "Particulate matter" category is case-sensitive (lowercase 'm').

`dim_alertpeak` (ETL04): Five fixed rows with keys 1000-1004 (None→Yellow→Orange→Red→Crimson) for stable sorting.

Fact Table Computation

`ft_param_city_month` (ETL05): Monthly aggregation using `DATE_TRUNC('month', recorded_at)`. Eight measures computed: - `reading_events_count`: `COUNT(*)` - `devices_reporting_count`: `COUNT(DISTINCT device_id)` - `recordedvalue_avg`: `AVG(recorded_value)` - `recordedvalue_p95`: `PERCENTILE_CONT(0.95)` - `exceed_days_any`: Days with alert thresholds exceeded - `data_volume_kb_sum`: `SUM(data_volume_kb)` - `data_quality_avg`: `AVG(data_quality)` - `missing_days`: `days_in_month - COUNT(DISTINCT DATE(recorded_at))`

Notable Decisions and Pitfalls

Monthly grain: Balances query performance with temporal detail. Daily would create millions of rows; monthly keeps table manageable while supporting trend analysis.

Validation: Post-ETL checks verified referential integrity and measure validity.

2. Answers to Business Questions

Selected Questions

SQL – Student A: Q01, Q02, Q04, Q05, Q06, Q07, Q08
SQL – Student B: Q09, Q10, Q11, Q13, Q14, Q15, Q16, Q17
MDX – Student A: Q01, Q02, Q03, Q12, Q18
MDX – Student B: Q09, Q10, Q19, Q20, Q21

Query Implementation Reflections

i. CROSSJOIN fix in MDX (Q01, Q02, Q03, Q12, Q19, Q20, Q21): Initial queries placed measures in `WHERE` clause, returning only hierarchical indices. Fixed by using `CROSSJOIN({ [Measures].[...] }, dimension_members)` on COLUMNS axis. This Atoti-specific pattern differs from SQL's implicit measure selection and was the most common debugging issue.

ii. Case-sensitive filtering (Q18): MDX query for "Particulate Matter" returned empty results. Database stores "Particulate matter" (lowercase 'm'). SQL is case-insensitive; MDX `FILTER()` with `.Name` is case-sensitive. Lesson: Always verify exact dimension value casing.

iii. Top N queries — SQL vs. MDX (Q06, Q09, Q14): SQL uses `ORDER BY ... LIMIT 10` with explicit sorting. MDX uses `TOPCOUNT(set, N, measure)` which is more concise but requires understanding measure context. MDX feels more declarative for ranking operations.

iv. Pivot tables — SQL complexity vs. MDX elegance (Q01, Q02, Q20): SQL required `CASE WHEN month_name = 'Jan' THEN value END` for each column (verbose 12-column pivots). MDX naturally expresses cross-tabs with dimensions on ROWS/COLUMNS axes. For multi-dimensional reports, MDX is significantly cleaner.

v. Hierarchical filtering (Q02 Austria, Q21 Eastern Europe): SQL used `JOIN dim_city WHERE country_name = 'Austria'`. MDX navigated hierarchies:`FILTER(...,ANCESTOR([dim_city].[Geo].CurrentMembe = "Austria")`. MDX hierarchy traversal is powerful but steeper learning curve.

vi. Month ordering (all time queries): SQL relied on `month_num` column. MDX required `CustomOrder(first_elements=["Jan","Feb",...,"Dec"])` configuration to override alphabetical default. This global setting simplified all subsequent MDX queries.

3. Reflection and Lessons Learned

Student A: Muhammad Sajid Bashir (52400204)

This assignment deepened my understanding of dimensional modeling and OLAP architecture. Working on dimension ETL (ETL01-ETL04) taught me the importance of surrogate keys and fixed lookup tables for stable analytics. The monthly grain decision required balancing detail with performance — too granular (daily) creates storage issues, too coarse (yearly) loses analytical value. The MDX learning curve was steep. Initially, I struggled with CROSSJOIN syntax and hierarchy navigation. Debugging Q18's case-sensitivity issue highlighted the importance of exact dimension value matching. However, once mastered, MDX proved more elegant than SQL for multi-dimensional queries, especially pivot tables and Top N rankings. Teamwork was essential — Student B validated fact measures while I focused on dimension correctness. We jointly debugged Atoti configuration (month ordering) and Docker issues. Using Jupyter for ETL orchestration kept SQL scripts modular while Python handled workflow logic. This assignment shifted my view of data warehousing from "bigger databases" to purpose-built analytical systems.

Student B: Eman Shahin (12432813)

Assignment 2 showed me the power of pre-aggregated analytics. The fact table (ETL05) pre-calculates eight measures, enabling sub-second dashboard refreshes versus querying raw OLTP. Understanding semi-additive measures (avg, p95) versus fully additive ones (counts, sums) clarified when to use SUM vs. MEAN in OLAP aggregation. Building Atoti dashboards revealed data quality patterns — missing days analysis (Q02, Q19) exposed collection gaps, while alert distributions (Q20) showed Crimson alerts concentrated in Health Risk categories. These insights were immediate with OLAP but would require complex SQL in OLTP. The toolchain integration (DBeaver→staging, Python→ETL, Atoti→dashboards, Docker→PostgreSQL) taught me workflow orchestration. Each tool serves a purpose: DBeaver for schema design, Jupyter for reproducible ETL, Atoti for interactive analysis. Student A and I divided work by dimension vs. fact, then merged results. This assignment transformed my understanding — data warehousing is an entire analytical ecosystem, not just storage.