First of all, I cover the dbt, which I believe matches well with IMAGO business, data stack and targets, then I go through orchestration and then the cloud warehouses:

- **Current data stack fits well with dbt:**
    - The current data stack is based on SQL, SSIS and Tableau, which is a good fit environment for dbt.
    - Tableau connects to dbt data models easily.

- **Extended data sources:**
    - If the current data sources get more variety other than csv and exel as mentioned in the data challenge, and require coding, the ETL expansion will be much easier through dbt.

- **Lack of version control and collaboration:**
    - SSIS is more like a visual and drag-and-drop tool, which makes version controlling, tracking changes and onboarding new members difficult.
    - SSIS jobs are manually deployed and are not CI/CD friendly.
    - There is no collaboration through SSIS as git versioning is also not possible and all xml files get corrupted if two people work at the same time on the same component.
    - SSIS is a powerful tool, but it always needs data engineers to work with and not cross-functional collaboration. If we want to add back office team, developers and other people to unify the data catalog and remove nonconformities more easily, dbt is a better and easier tool for data democracy cause it offers:
        - transparent data models
        - lineage graphs
        - a very powerful documentation in yml files which is a powerful tool for cross-collaboration.

- **Lack of data quality:**
    - At the moment, so many data nonconformities have made the data dirty. Capturing and flagging them is more difficult and manual in SSIS, whereas

anomaly flags, status fields, and data contracts can all be enforced directly in dbt models + tests :

- Null checks
- Range checks
- Relationship integrity (invoice ↔ position ↔ customer)

Proposed model on dbt:

Based on the dbt best practices, I would propose a 4 layer model:

- **sources:**

  Represents the YML file containing the sources and configs:

```
version: 2
sources:
 - name: imago_raw
   database: imago_data_infra
   schema: raw_data
   tables:
     - name: invoices
     - name: positions
     - name: customers
```

- **stage:**

  Standardizes and cleans raw data.

- **warehouse:**

  Models the data into fact and dim, Also Anomaly detection, inferred data and constraint checks for nonconformity dashboards happen here.

- **marts:**

  Analyst playground and Tableau feed will get created here.

Below is the schematic version of such a model:

models/

```
├── sources/
│   ├── src.yml
│
├── stage/
│   ├── stg_invoices.sql
│   ├── stg_positions.sql
│   └── stg_customers.sql
│
├── warehouse/
│   ├── wrhouse_dim_region.sql
│   ├── wrhouse_dim_customer.sql
│   ├── wrhouse_fact_invoice_position.sql
│   ├── wrhouse_fact_invoice.sql
│   └── …..
│
├── marts/
│   ├── mart_revenue_overview.sql
│   ├── mart_anomaly_summary.sql
│   └── mart_customer_content_revenue.sql
```

What Should Stay As-Is (For Now):

Data Ingestion from Excel, CSV or Access sources:

- These can stay SSIS-based until a new input method (e.g., API, form submission) is implemented.

Tableau Dashboards:

- New nonconformity dashboards must be visualized and subscribed.
- No changes required for the previous dashboards. Only the underlying data models would change.

Migration Risks and Considerations:

- **New nonconformities arrival:**
    - Since dbt is a cross-collaboration tool, data analysts are also welcome to have access to dbt and prepare their own data, this might lead to dirty data and inconsistent KPIs unless:
        - Invest in dbt onboarding workshops for both analysts and engineers and make sure everyone is fully aware and sensitive to unified definitions and KPIs.
    - There is a chance of a logic mismatch occurring between the two tools (SSIS and dbt):
        - Run both in parallel temporarily and reconcile the outputs
- **Upstream data instability:**
    - Make ETL validations explicit and use audit dashboards to catch anomalies early.
- **Dependency conflicts:**

    SSIS can handle both transformation and orchestration, while dbt handles only transformation. A very careful orchestration strategy must be applied to dbt to ensure the valid data flow.
- **Performance Differences**

dbt runs on the target database (e.g. SQL Server, Snowflake) and depends on the performance of the SQL engine, while SSIS has been tuned with intermediate cache steps. Therefore, the tuned use of materializations and optimized SQL become more important.

Orchestration:

Since the dbt scheduling feature comes only with the premium version, so far, the etl part is covered and now some orchestration tool is needed to run it as a schedule and make sure that the components and models run in a way that does not create inconsistency or inferred data. The order that the layers run really matters.

I can propose two orchestration tools for the current use case:

Airflow:

Airflow is the most popular choice for scheduling and orchestrating for modern data stacks and works perfectly with dbt.

Here is the roadmap to use Airflow as an orchestration layer:

- First, I schedule all the extract jobs on SSIS on Airflow.
- I think and design the orchestration flow. All logical steps must be ordered properly. Also the flow must stop on some serious errors and never propagate till the end. For example:
    - The warehouse folder should never get entered if there is an error in the staging.
    - I should make sure that the position data loads after the invoice.
    - Facts load only after the loads of all dimensions.
    - All dbt tests should be run both after transformation and also loading into the warehouse layer.
- Sending notifications tothe back office, developers and BI teams can be set through Airflow email.
- Auto retries can also be set to avoid extra errors and logging.
- Tableau refresh can also be scheduled as the last step of the flow to have the latest data on the visual layer.

What Should Stay As-Is (for Now)

- Data extraction from Access, Excel, or legacy SSIS can stay as they are.
- Tableau reports and refresh schedule can also remain the same. But the revenues that won't get shown up in dashboards can maybe get fixed here by a robust job run.

Risks:

- **Logic conflicts between both systems:**

  I should make sure the exact same logic applies in airflow to prevent new conflicts occure. Best case scenario would be to keep both systems running for a while to be able to compare and avoid future inconvenience.

- **Upstream data instability:**

  Since the raw data is at the moment exel and csv and most of the time, manual works get done on these sources, there is a high chance that the very very first step of ETL crashes. SSIS might ignore some of these small conflicts but since Airflow is stricter than SSIS, the orchestration will fail fast when something is missing, malformed, or doesn't match the expected schema.

- **Python basic knowledge:**

  All the people who are supposed to interact with airflow must get onboarded on Python principles and basic knowledge.

Dagster:

Since **dbt is at the heart of our ETL**, and all my challenge is considered on **data and analytics engineering** and not a general job scheduling workflow, I think Dagster might seem a better and faster solution:

- Dagster automatically finds and detects dbt models (all stages, dims, facts and marts), and show them as a node in UI. This enables us to run only a specific node for a test and then finally run the whole flow.

- Dagster allows you to define expected data types for inputs and outputs. For example, expecting Nettobetrag to always be a non-null float. Which can have a huge contribution to the nonconformities I found earlier on the challenge.

- Since we are in a transition mode to a modern data stack, alot of things must get done. So Dagster could reduce a bunch of workload cause Airflow is more configuration-heavy and in the need of devops. Dagster is easier with a more modern UI and more pythonic use. So we reduce infrastructure maintenance and can invest more time on data quality and validation as its the current focus of the challenge.

Cloud data warehouse:

Snowflake could be a good choice over the current SQL data warehouse if:

- Several teams want to have heavy queries parallel to each other. Like If finance is running a complex revenue report while BI is refreshing Tableau.
- Zero infrastructure and more effort on data modeling and data conflict resolution is needed
- New data types like Json, Parquet or blob is gonna be needed for storing.

Given the fact that IMAGO is a BI-heavy company and Image, video and related metadata are gonna be an inseparable part of data, Snowflake is a very good choice.

How I would propose migrating to Snowflake:

- Set up Snowflake as the central data warehouse.
- Extract existing data from current sources (SQL Server, Excel/Access exports) using SSIS, Fivetran, or dbt seed files.
- Load raw data into Snowflake staging tables for invoices, positions, and customers.
- Point dbt to Snowflake and build modular transformations:
  - staging: cleaning and transforming data
  - warehouse: build and fill facts and dims, apply anomaly flags and join logic
  - marts: create Tableau-ready tables and analyst playgrounds
- Connect Tableau directly to Snowflake

- Define roles, privileges, row-level security and other security principles
- Hold onboarding sessions for all end users like analysts, finance etc.

Migration Risks and Considerations:

- Data types, precision, or encoding may differ slightly from SQL Server and require extra validation after all.
- Inefficient queries or unmonitored usage could create huge costs.
- Integration updates must be applied on dbt and Tableau for Snowflake.