Week 4 - Implementation: Intermediate layer

The objective of the fourth week is to create the intermediate layer for the project, but only for the listed companies and repositories based on the Company Details sheet.

General

This week the task and the project description will be less in detail than it was in the staging layer. It focuses more on the business side and gives a more general overview about the technical part, creates more space to explore and solve the project tasks with more freedom.

As for the development principles and dbt best practices, the same rules and tips stand.

Since it is necessary to establish relationships between the models in this layer, the first step in the implementation should be to finish the Company Details task and only after it is advisable to proceed to the other tasks.

Company Details (Google Sheet)

Objective

Create an intermediate table for the Company Details sheet that uses the snapshot materialization method.

Description

The Company Details sheet can change over time. As mentioned earlier, this spreadsheet is a business input and it is possible that business users may change its content from time to time. Add, update or remove organizations, change the repositories or just change the list of used tags. The task in this case is to ensure that all these changes are stored in the intermediate table, but it is possible to use only the latest, current version of the Company Details sheet.

To accomplish the task, we created a new tab in each student's Company Details sheet called v2, which contains the next version of the spreadsheet. In order to implement the task, there should be one active tab which is used by Airbyte and the other tabs can store the versions. This should be changed by the students themselves throughout the development cycle.

At the end of the task, the intermediate table needs to contain the first version and then the v2 loaded as well.

To follow and stay up to date, but keep the previous versions of the Company Details list, a historical data handling should be used here. More info here.

dbt has a built-in solution for handling and managing slowly changing dimensions. This is called snapshots. Snapshot uses SCD-Type 2 method.

Hint

It will be explained later, but for the project to work properly, it is necessary that an organization in the Company Details sheet that has already been deleted, should not be included in the current version. To do this there is a configuration setting in the snapshot operation that needs to be used. Solution here.

Additional task

Consider changing the primary key. If the primary key was generated inside the dbt model previously, it can cause problems with the snapshot methodology. dbt suggests to use surrogate keys instead of auto-incrementing IDs. There is a macro in one of the dbt packages which may be useful for this purpose.

GitHub

Objective

At the end of this sprint week, there should be one intermediate GitHub table with the above columns joined with Company Details sheet, materialized as an incremental model in dbt with proper tests and documentation.

Description

- **Company Details join**

For the intermediate layer in GitHub, only work with the repositories that exist in the Company Details sheet. If an organization does not have a repository account, then the GitHub data for that organization is not required. If the organization has a repository account in the Company Details sheet, then it should match the GitHub repository account. But if the repository name is empty in the Company Details sheet, it means that all the available repositories should be used for that GitHub account.

Join the Company Details to the GitHub dataset through those two fields, repository account and repository name. And make sure that you also bring the organization name to the intermediate table from the Company Details sheet.

As the intermediate Company Details table is now historical, which contains all previous versions of the sheet, make sure to use only the actual version of the Company Details.

- **List of columns**

From the GitHub dataset in the intermediate layer only a few columns are needed.

Column name	Description
_pk	Primary key for the table
repository_account	There is a delimiter in the repo.name field. It should be split into two columns. It is used as join fields with the Company Details
repository_name	There is a delimiter in the repo.name field. It should be split into two columns. It is used as join fields with the Company Details
user_id	It is the actor.id in the source
event_id	ID column in the source
type	Same in source
organization_name	Coming from the Company Details
created_at_datetime_utc	Same in source

Payload field is a large JSON file. Since the staging model is just a view, it is not a problem to include the field, but it should not be used in the intermediate layer, even in the queries.

Materialization

Use incremental materialization to create the intermediate table where only the new day will be processed.

Test

Create a custom test where the data quality is validated for each new daily source table. Make sure that the new source table contains only the events for the actual day, that means the *created_at* field equals in every row with the name of the table.

Pay attention when creating the test that the intermediate model should run even if there are rows with wrong dates inside. It means it should be given only warning and not error severity in dbt. The log schema with the wrong events has to be created as well.

Hint

If it is needed, it is possible to change the staging model to prepare the data in a different way and change the structure for the intermediate layer needs.

Stack Overflow

Objective

At the end of this sprint week, there should be two intermediate Stack Overflow tables with the above columns joined with Company Details sheet, materialized as an incremental model in dbt with proper tests and documentation.

Description

For the Stack Overflow data, the *tag* field will be used to mark and catch the matching questions and answers with the organizations listed in the Company Details sheet.

Only work with the questions and answers that have tag matches with the organizations in the Company Details sheet.

In that dataset, all the staging Stack Overflow columns should be brought and materialized in the intermediate layer, except the column named *body*, because it is a large JSON, which is unnecessary as it would take up too much space.

It is also necessary here to add the organization column as well from the Company Details sheet.

Materialization

Use incremental materialization to create the intermediate tables where only the newly coming staging data will be processed.

Hint

Create and use the tags from both sides as arrays. Array operations, like array intercept or unnest, are easier to use and take less process cost. It is also possible that there will be no match for some of the companies.

If it is needed, it is possible to change the staging model to prepare the data in a different way and change the structure for the intermediate layer needs.

If the characteristics and the connection between the two resource tables is considered, then if they are processed in the right order, it is possible to achieve the output from less resources.

Additional task

dbt cloud

Setup up your project in dbt cloud as one developer seat is always free with dbt. Do all the configurations and use our project’s GitHub repository to transfer your project into dbt cloud. Make sure that your local project is in sync with dbt cloud.

To create your project in dbt cloud, it is more essential to have separated and updated branches in the project’s GitHub repository. After you created your dbt cloud project, get started with dbt jobs.

Jobs

Try one-time jobs first through dbt cloud and see what information and feedback is available on the interface for the runs.

Once you're more familiar with the interface, create a separate job for each layer with a weekly schedule, listing the models that belong to that layer with tests included. At this stage of the project there should be a staging and an intermediate job.

There are several ways to refer to a dbt job. One way is to list them directly, but a more elegant way is to tag the models and use the tags in the job command. Tags can configured in the yml files or in the configuration block of each model.