Matillion AI Data Games Virtual Workshop Lab Manual

Thank you for joining the Matillion AI Data Games Virtual Workshop!

You will be processing a data set to explore the **2016 Summer Olympics**, and to discover more about the most successful competitors.

During the lab, you will be developing ETL pipelines inside the Matillion Data Productivity Cloud, and taking advantage of Matillion's virtual data engineer - Maia - to help →

The lab will start with some simple and foundational data engineering tasks, and will quickly build on this to include more sophisticated techniques and analysis. You will be able to experience hands-on using code-optional data engineering, data transformation and integration, and concluding by working with generative AI inside an ETL pipeline.

There is a standard set of three pipelines that the host will briefly demonstrate at the start of the session.

During the main part of the lab, the host will build the main ETL pipeline from scratch, with plenty of time and commentary. You will have access to the same development environment and will be able to follow along and build the same pipeline yourself. You're also free to go off piste and try some variations! The aim is to give you the opportunity to become familiar with the platform while building an interesting analysis.

After the lab, you will retain access to your work in Matillion for several days. During this time you will be free to experiment independently, with more data samples - including the ability to upload your own data. You will retain access to all the generative AI functionality including with Snowflake Cortex and OpenAI.

This document contains the instructions you will need:

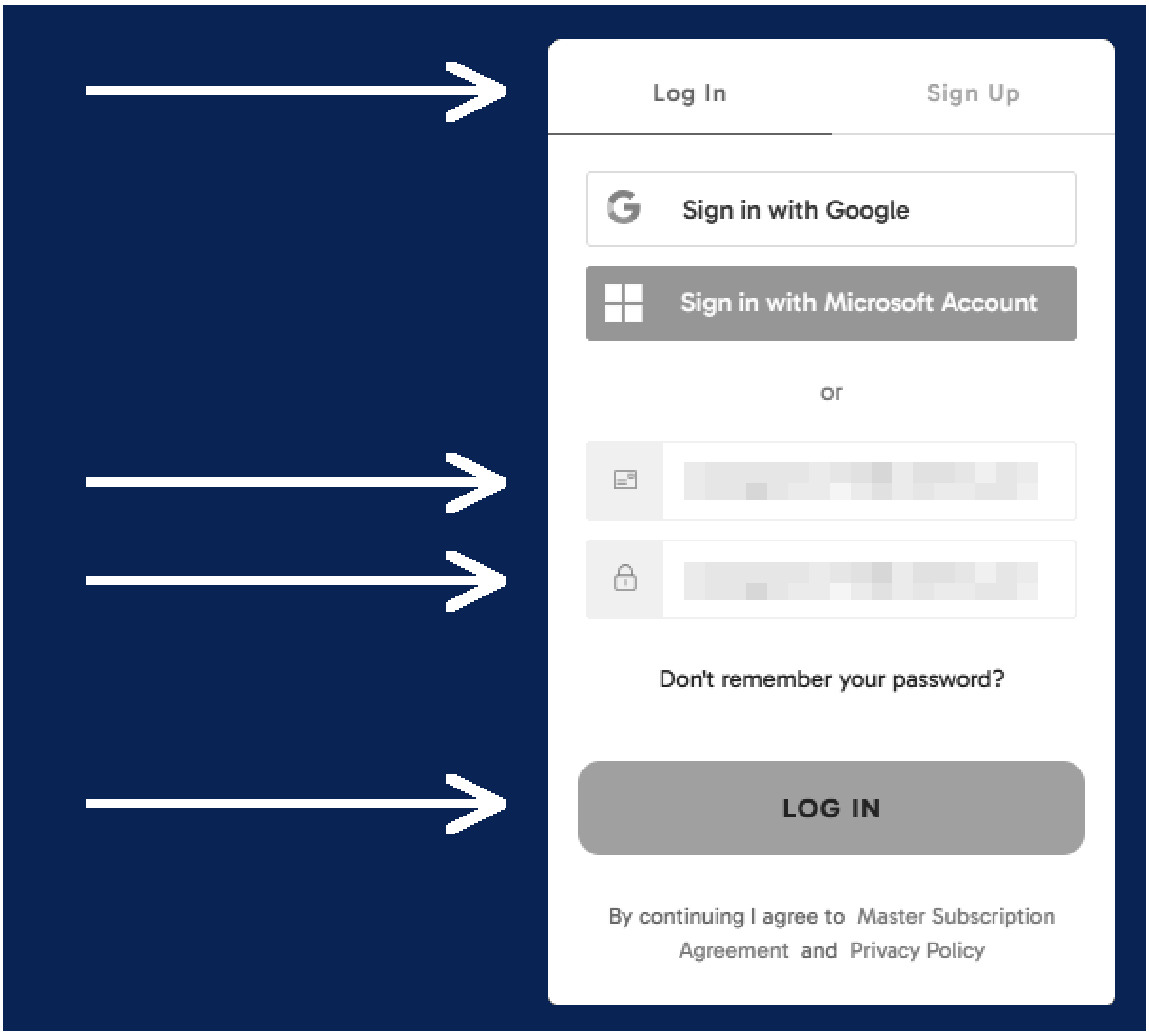
- to participate **during** the event
- to optionally continue using Matillion **after** the event

Your unique Matillion username

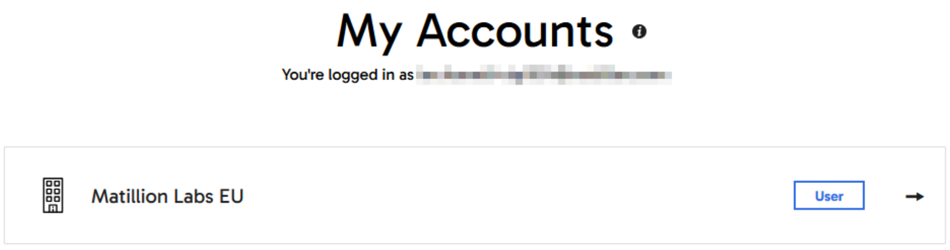
When you join the event you will be provided with a Matillion username and password.

This identification is unique to you for the duration of the lab, and for several days afterwards.

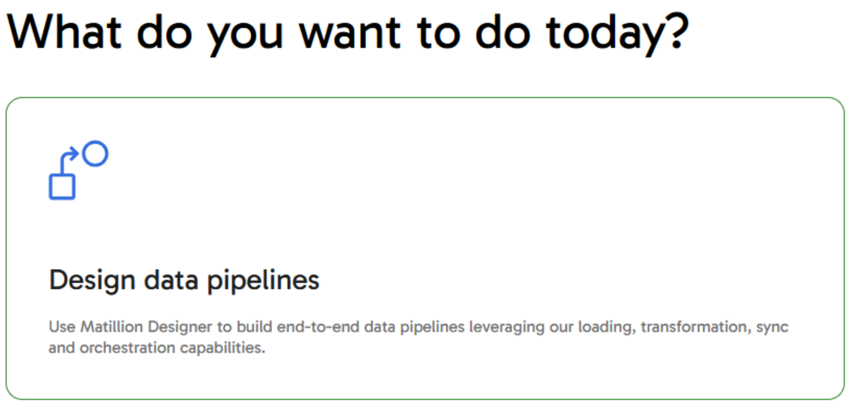
Open a web browser and log In at https://hub.matillion.com


Log In
Sign Up
Sign in with Google
Sign in with Microsoft Account
or
Don't remember your password?
LOG IN
By continuing I agree to Master Subscription Agreement and Privacy Policy

At the "My Accounts" screen, click on **Matillion Labs EU** to continue.



Once in the Matillion Labs project, follow the big "Design data pipelines" link.


What do you want to do today?

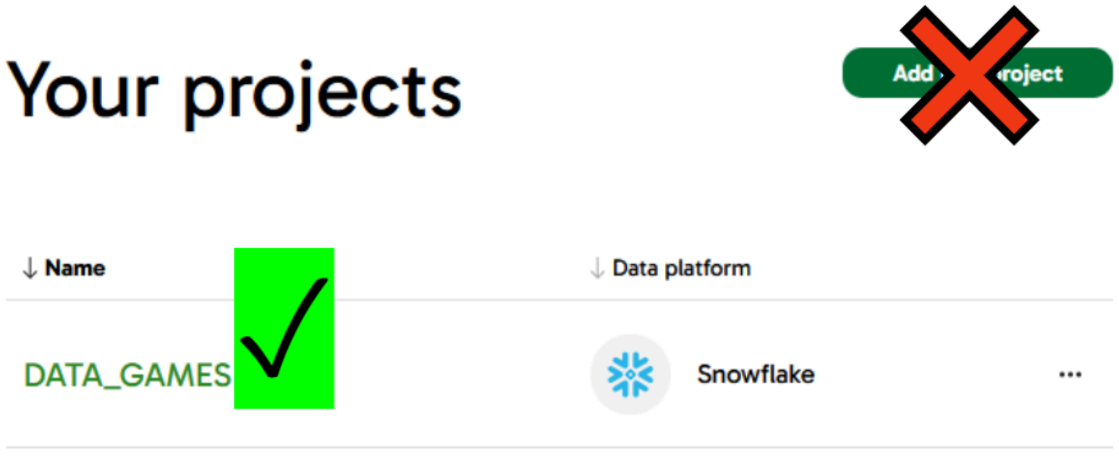
Design data pipelines

Use Matillion Designer to build end-to-end data pipelines leveraging our loading, transformation, sync and orchestration capabilities.

You will see AI Data Games listed in your projects.

❌ **Do not create a new project!**

✅ **Click into the Data Games project**



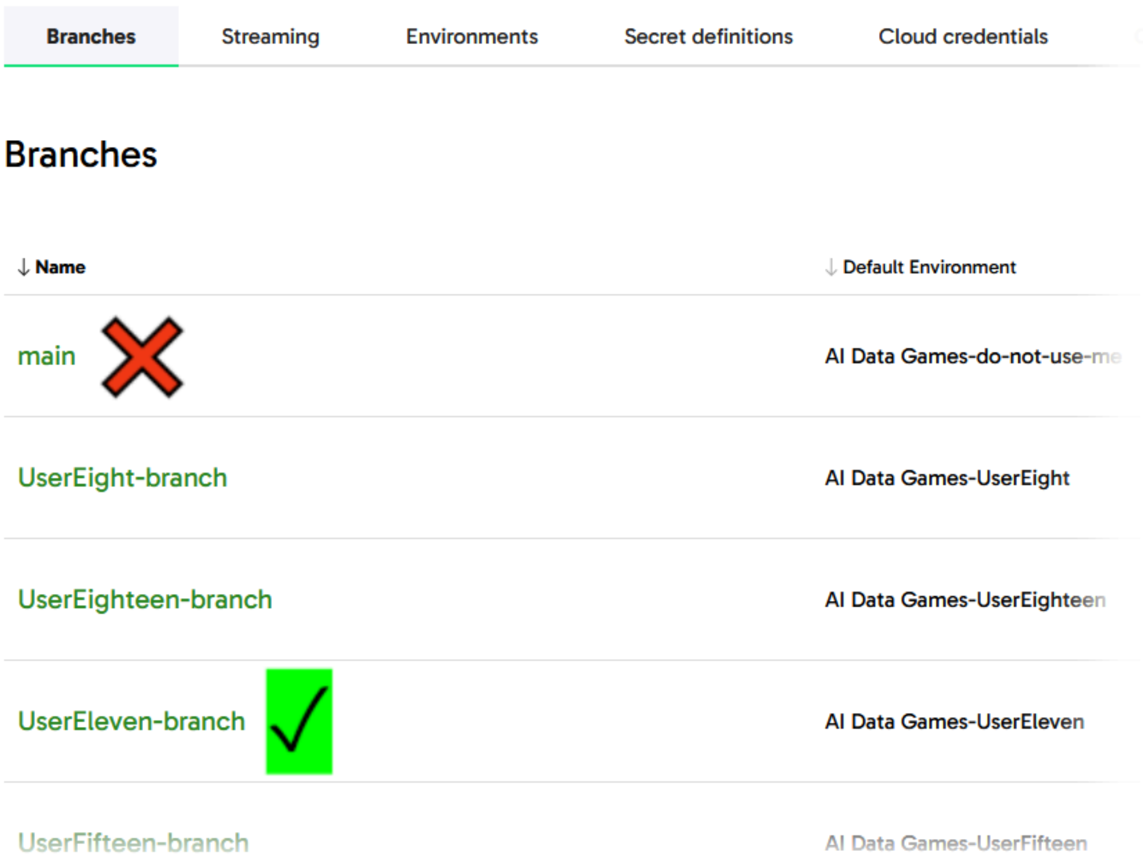
Work in your own Branch

This Project is shared among many attendees, so once you are connected;

❌ Please do not use any of the git operations (commit, push, pull etc)

❌ Please do not work in the **main** branch

In the Your projects / AI Data Games screen, **click on the Branch name that matches your user name.**



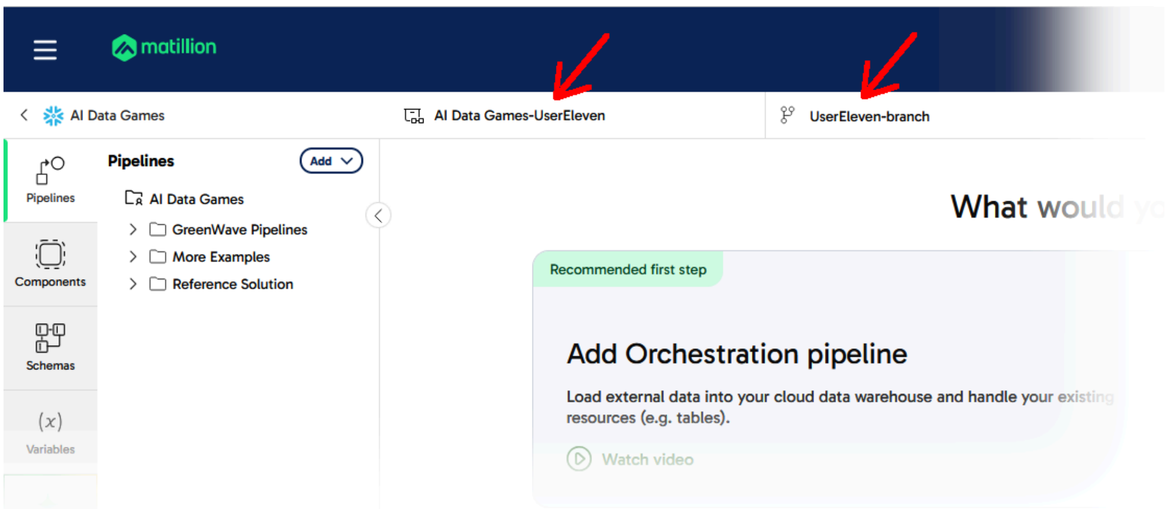
Branches | Streaming | Environments | Secret definitions | Cloud credentials

Branches

↓ Name	↓ Default Environment
main	AI Data Games-do-not-use-me
UserEight-branch	AI Data Games-UserEight
UserEighteen-branch	AI Data Games-UserEighteen
UserEleven-branch	AI Data Games-UserEleven
UserFifteen-branch	AI Data Games-UserFifteen

Example: User 11 should choose this branch

Check that your screen looks something like this:



Example for User 11

Now you're ready to begin the lab!

During the event - Build the Data Games AI pipelines

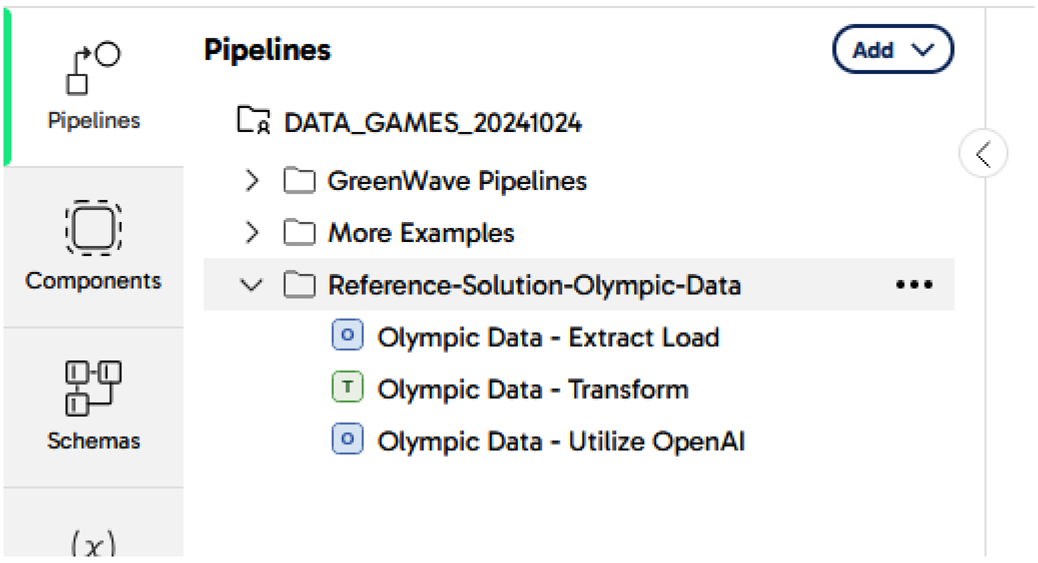
The lab will be in two parts:

- In the first part you will build three pipelines that work on a set of Olympic athlete data
- In the second part you will be able to work and experiment independently with more data samples. The Matillion team will be online and ready to help or answer questions

Lab part 1 - building the Olympic data pipelines

You will build these pipelines yourself, watching as the instructor does the same on screen.

A reference set of pipelines is available under the Reference Solution folder. You can examine or copy from these if you get stuck!



Pipelines

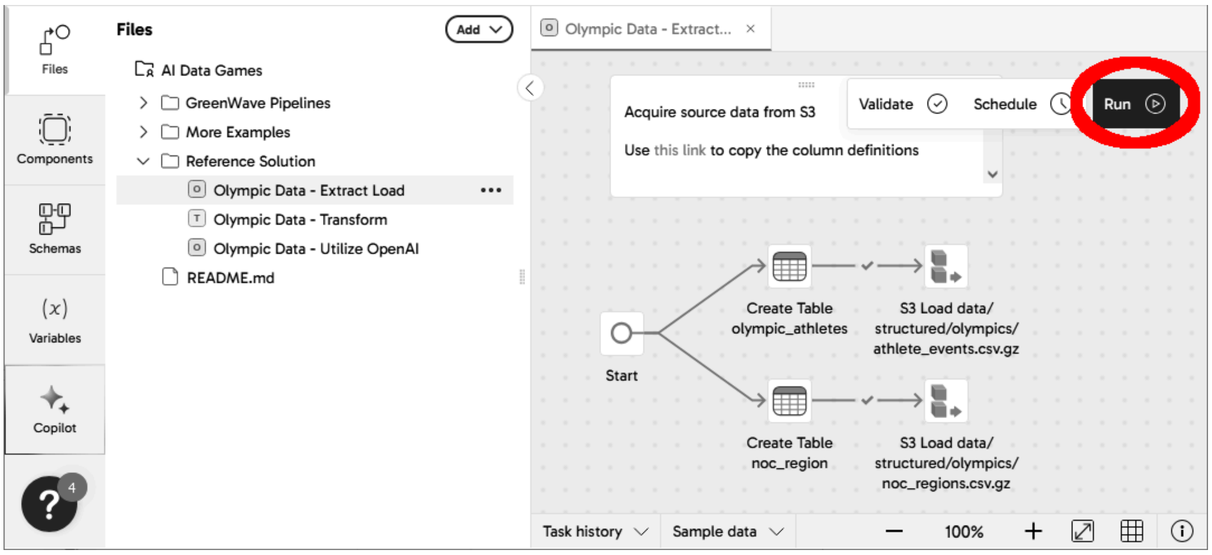
DATA_GAMES_20241024

- GreenWave Pipelines
- More Examples
- Reference-Solution-Olympic-Data
  - Olympic Data - Extract Load
  - Olympic Data - Transform
  - Olympic Data - Utilize OpenAI

Olympic Data - Extract Load

This is an **orchestration** pipeline that brings the source data into Snowflake, from two files in cloud storage.

Locate this pipeline under the **Reference Solution** folder, and click once to open it in the editor.

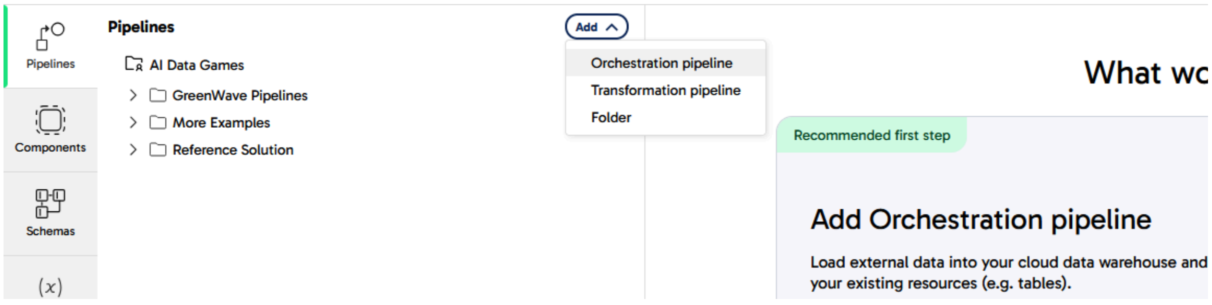

Files
Components
Schemas
(x)
Variables
Copilot
Files
Add
AI Data Games
GreenWave Pipelines
More Examples
Reference Solution
Olympic Data - Extract Load
Olympic Data - Transform
Olympic Data - Utilize OpenAI
README.md
Olympic Data - Extract...
Acquire source data from S3
Use this link to copy the column definitions
Validate
Schedule
Run
Start
Create Table olympic_athletes
S3 Load data/ structured/olympics/ athlete_events.csv.gz
Create Table noc_region
S3 Load data/ structured/olympics/ noc_regions.csv.gz
Task history
Sample data
100%

Press **Run** to load the data.

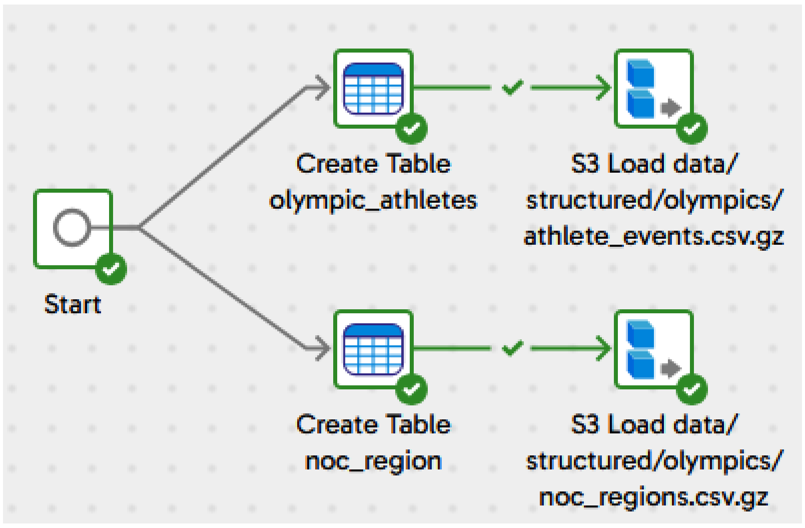
The lab will return to this pipeline later.

Meanwhile skip ahead to the next subsection: "Olympic Data - Transform" on page 9.

To replicate this extract-and-load functionality, start by adding a new Orchestration pipeline, using one of the on-screen options.



Give the new pipeline a name, and it will open in the editor. As you follow the lab, it will end up looking like this:

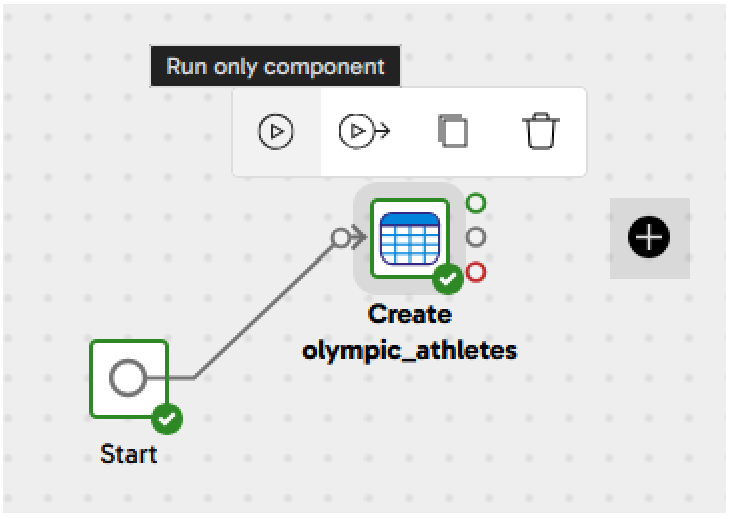

Start
Create Table olympic_athletes
S3 Load data/ structured/olympics/ athlete_events.csv.gz
Create Table noc_region
S3 Load data/ structured/olympics/ noc_regions.csv.gz

Choose a Create Table from the components menu, under DDL, and drag it onto the screen. Connect it downstream of the Start component and configure it as follows:

- Name: **Create olympic_athletes**
- Create/Replace: **Replace**
- New Table Name: **olympic_athletes**
- Columns: switch the view to text mode and copy and paste from this link (Go to the "Lab Helpers" section at the end of https://exchange.matillion.com/data-productivity-cloud/pipeline/olympic-athlete-data-analysis/), or from the reference pipelines

Check that the component validates cleanly, showing a green border.

Run the component on its own to create the new table.



Now use the + sign to add an S3 Load component after the Create Table.

- S3 Object Prefix: **devrel.matillion.com** (ignore any errors you see at this point)
- Pattern: **data/structured/olympics/athlete_events.csv.gz**
- Target Table: **olympic_athletes** (the one you just created)

- File Type: **CSV**
- Compression: **GZIP**
- Field Delimiter: **,** (a single comma)
- Skip Header: **1**
- Field Optionally Enclosed: **"** (one double quote)

Run the S3 Load component on its own, and verify in the Task History that 271116 records were loaded. Remember Snowflake will silently ignore attempts to load the same file more than once into the same table. If you need to reload the data, use a "Run From" starting at Create olympic_athletes, which will recreate the target table first.

Moving on to the next source data, add another Create Table component connected downstream of the Start component:

- Name: **Create Table noc_region**
- Create/Replace: **Replace**
- New Table Name: **noc_region**
- Columns: again it's easiest to copy and paste from this link while in text mode

Run the component on its own to create the noc_region table.

The source for this table is also in S3 cloud storage, so add another S3 Load component, configured as follows:

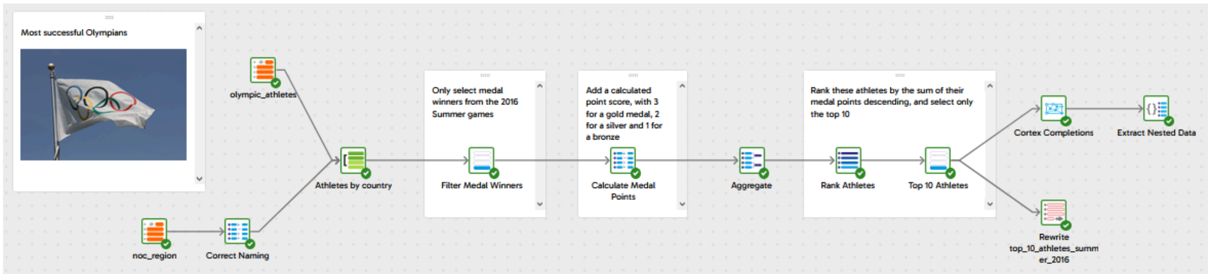
- S3 Object Prefix: **devrel.matillion.com** (ignore any errors you see at this point)
- Pattern: **data/structured/olympics/noc_regions.csv.gz**
- Target Table: **noc_region**
- File Type: **CSV**
- Compression: **GZIP**
- Record Delimiter: **\r** (one CR character)
- Field Delimiter: **,** (a single comma)
- Skip Header: **1**
- Field Optionally Enclosed: **"** (one double quote)

Run this second S3 Load component on its own, and verify in the Task History shows 230 records loaded.

Once the two tables have been created and populated with data, the Extract and Load steps are complete. This has been the "E" and "L" of ELT, so next it's on to Transformation.

Olympic Data - Transform

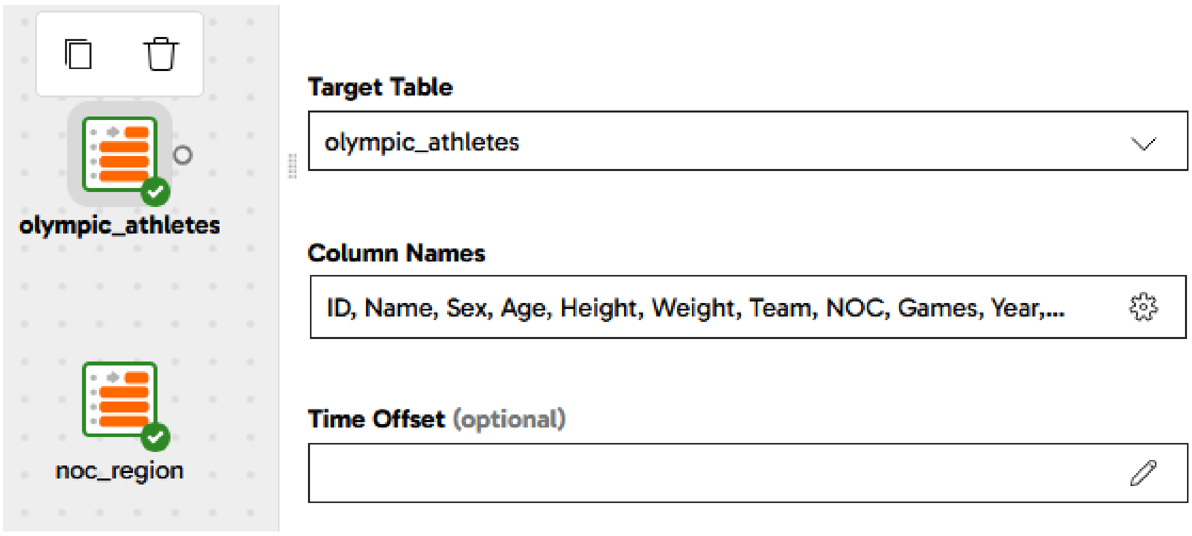
This is a **transformation** pipeline that works on the data you have just loaded into Snowflake.


Most successful Olympians
olympic_athletes
noc_region
Correct Naming
Athletes by country
Only select medal winners from the 2016 Summer games
Filter Medal Winners
Add a calculated point score, with 3 for a gold medal, 2 for a silver and 1 for a bronze
Calculate Medal Points
Aggregate
Rank these athletes by the sum of their medal points descending, and select only the top 10
Rank Athletes
Top 10 Athletes
Cortex Completions
Extract Nested Data
Rewrite top_10_athletes_summer_2016

Use the on-screen options to add a new Transformation pipeline.

Every Matillion transformation pipeline begins with components that read data somehow. This lab will use two Table Input components, which you can find under "Read" in the list of components.

Set the Target Table to olympic_athletes and noc_region respectively, and in each case choose all the columns. Your pipeline should look something like this:



Target Table

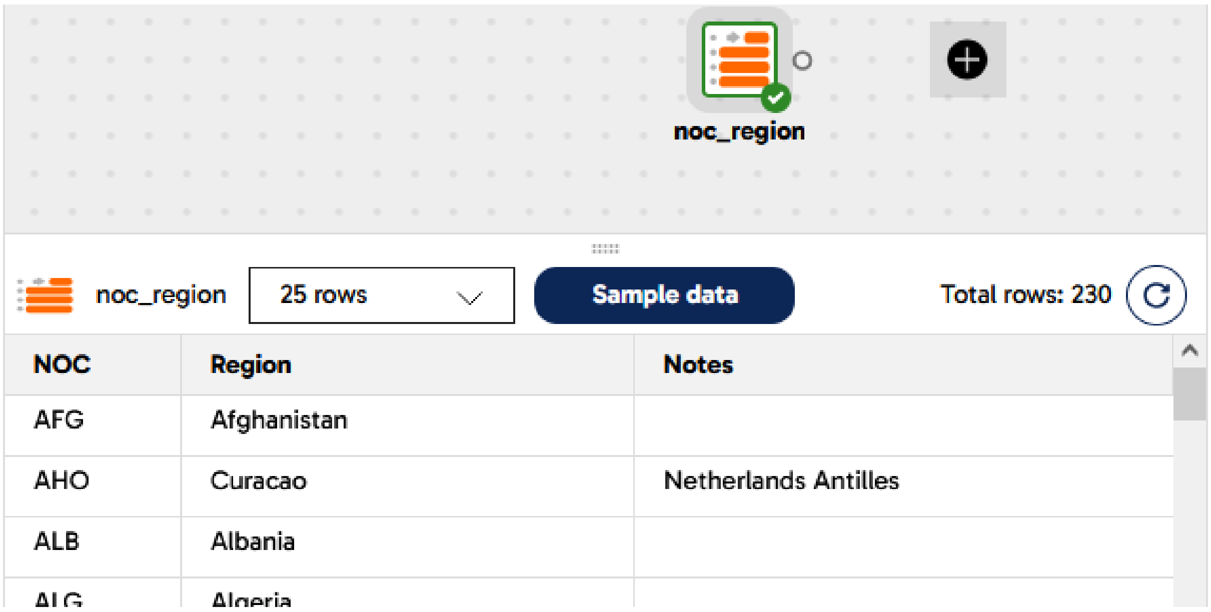
olympic_athletes

Column Names

ID, Name, Sex, Age, Height, Weight, Team, NOC, Games, Year,...

Time Offset (optional)

One of the great things about transformation pipelines is that you can keep in touch with your data as it goes through steps of transformation. Use the Sample Data panel at the bottom of the screen, and press the **Sample Data** button.



NOC	Region	Notes
AFG	Afghanistan	
AHO	Curacao	Netherlands Antilles
ALB	Albania	
ALG	Algeria	

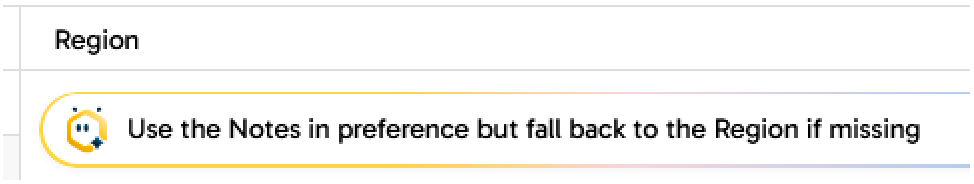
The Sample Data panel is available for almost every transformation component. It's good practice to take advantage of it at every stage during development.

This lab requires noc_region to act as a simple lookup table for athletes. But as you can see from the screenshot above it has both a Region and a Notes column, sometimes both containing values.

To fix this, add a Calculator component after noc_region. The Calculator is a general purpose data transformation tool, which can perform any SQL based operation. Configure the Calculator as follows:

- Name: **Correct Naming**
- Include Input Columns: **No** (we want only the columns added in the Calculator)
- Calculations:
    - NOC: "NOC"
    - Name: COALESCE("Notes", "Region")

If you get stuck, please take a look at this video to help configure the Calculator component. You may also choose to use Maia to create the expression. Use a prompt like this:


Region

Use the Notes in preference but fall back to the Region if missing

Verify that it's now a simple key/value table using the Sample Data panel.

One of the main ways to integrate data in a transformation pipeline is to use a Join component. Drag one of these into place downstream of the **olympic_athletes** and the new **Correct Naming** component, and connect it to both as sources. Configure the Join as follows:

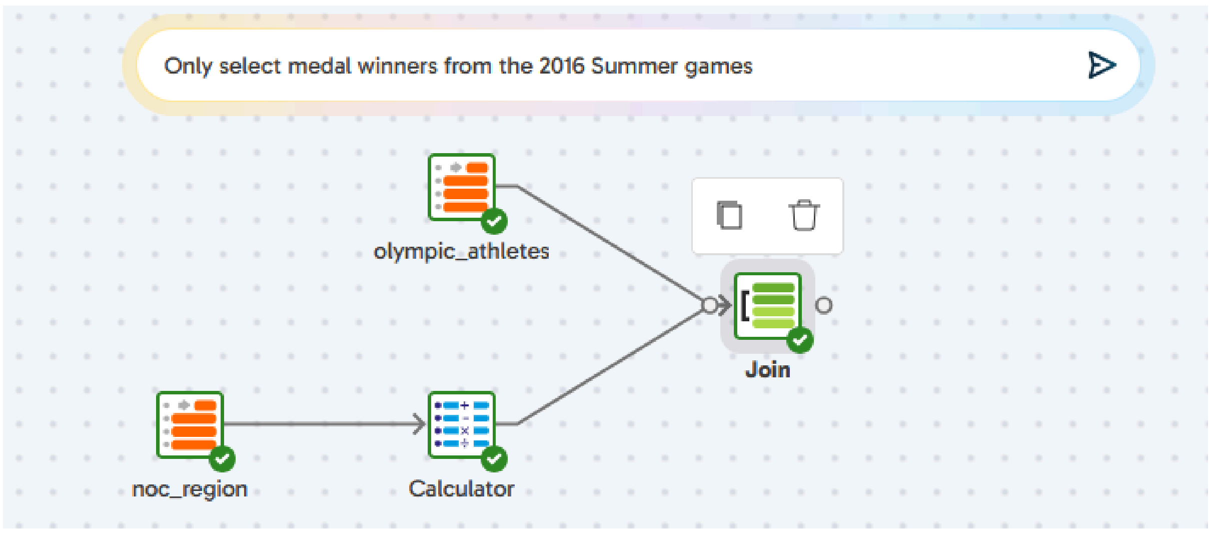
- Name: **Athletes by country**

- Main Table: **olympic_athletes**
- Main Table Alias: **ath**
- Joins:
  - Join Table: **Correct Naming**
  - Join Alias: **noc**
  - Join Type: **Left**
- Join Expressions: **"ath"."NOC" = "noc"."NOC"**
- Column Mappings: copy and paste from this link while in text mode

Check that the component validates cleanly, showing a green border. Verify that the join has been performed correctly, with no additional or lost rows. There should be 271116 records.

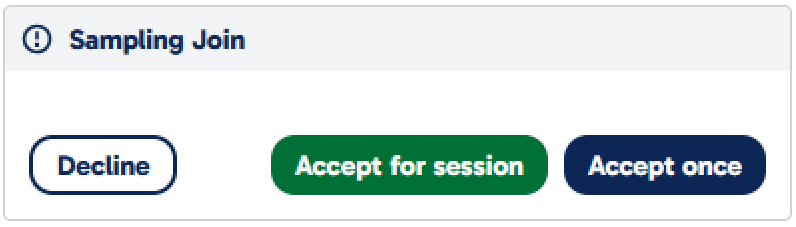
Instead of creating the next few components manually, it's time to use Maia some more.

- Select the Athletes by country component
- Add the following prompt to the Maia widget: **Only select medal winners from the 2016 Summer games**
- Press the Send icon (which looks like a paper airplane)



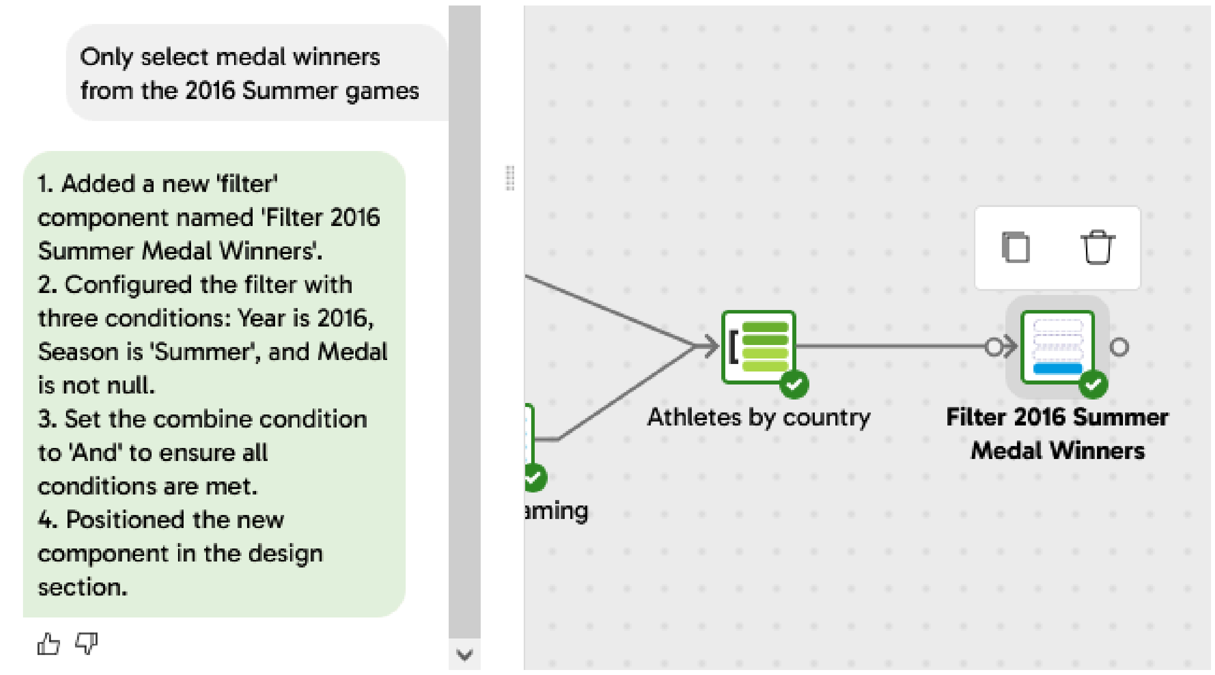
By default, **Maia does not have access to your data.** It can only read metadata (table names, column names etc). By permitting access to read data, you can make the generated solution more accurate, because it can check exactly what to put in the filter.

Accept sampling for the session like this:



Note that you do **not** have to accept data sampling. Maia will still be able to make a guess at the data, but you'll probably have to tweak the filters yourself afterwards.

Generative AI is never deterministic so your results may vary. But you should find a new Filter component prefilled with relevant conditions.


Only select medal winners from the 2016 Summer games

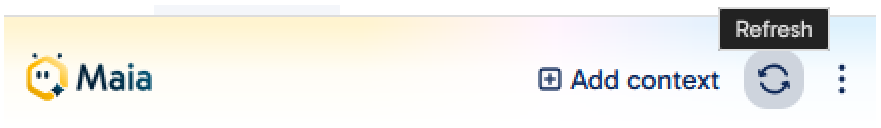
1. Added a new 'filter' component named 'Filter 2016 Summer Medal Winners'.
2. Configured the filter with three conditions: Year is 2016, Season is 'Summer', and Medal is not null.
3. Set the combine condition to 'And' to ensure all conditions are met.
4. Positioned the new component in the design section.

Double check the conditions are:

- Year is equal to 2016
- Season is equal to Summer
- Medal is not equal to NA

The row count - by strange coincidence - should be 2023.

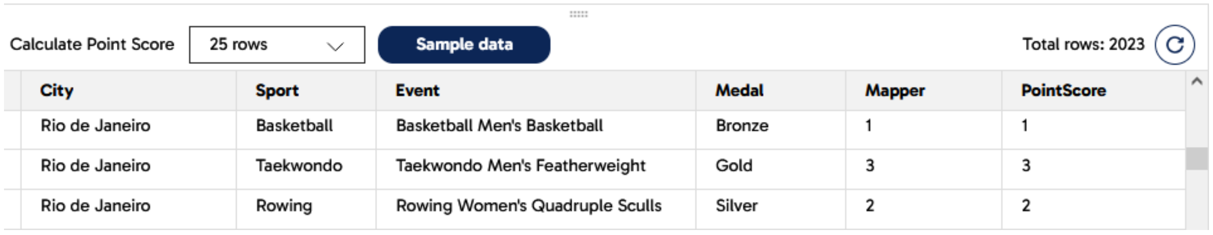
If Maia keeps adding more components, you can stop the AI generation at any time by pressing the conversation "refresh" link.



Next to allocate points for medals won. Highlight the new filter component and use Maia again with a new prompt: **Add a calculated point score to this pipeline, with 3 for a gold medal, 2 for a silver and 1 for a bronze**

There is more than one way to do this, so you might find a new Map Values component, and/or another Calculator.

Use a data sample to check that a point score column has been added, and that it has correctly allocated the points.



City	Sport	Event	Medal	Mapper	PointScore
Rio de Janeiro	Basketball	Basketball Men's Basketball	Bronze	1	1
Rio de Janeiro	Taekwondo	Taekwondo Men's Featherweight	Gold	3	3
Rio de Janeiro	Rowing	Rowing Women's Quadruple Sculls	Silver	2	2

To find the total points by athlete, use another new prompt: **Add an aggregate Point Score by Name to this pipeline**. This should add an Aggregate component, by Name, that sums the point score.

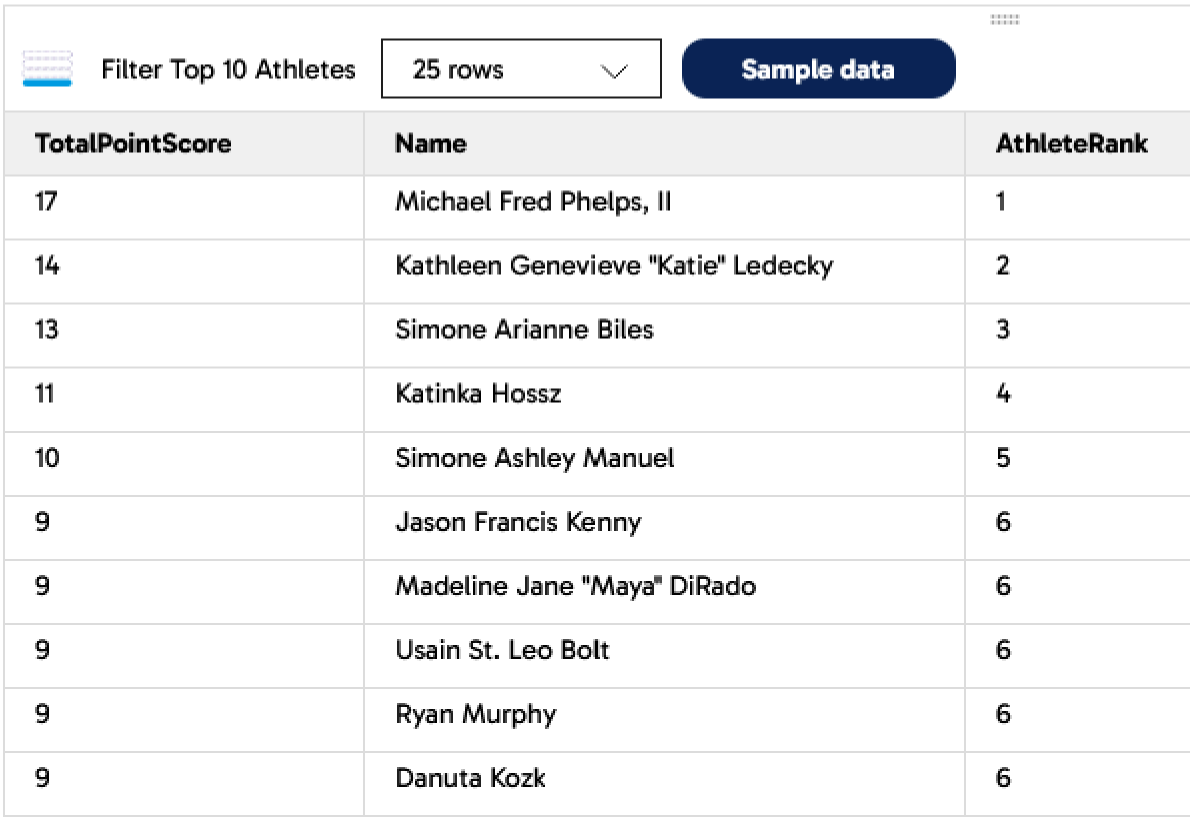
Check that the rowcount after aggregation is 1855. This is the number of different athletes who won at least one medal during the 2016 Summer Olympics.

As a last Maia exercise, highlight the new Aggregate and run another prompt: **Rank these athletes by the sum of their medal points descending, and select only the top 10.**

This time you should find:

- A new Rank component, either using ROW_NUMBER(), RANK() or DENSE_RANK() that sorts the records by the total medal points in descending order
- A new Filter component, that only selects the records ranked 10 or lower

Once again, there's more than one way to achieve this and your results may not be identical. But however you get there, you should end up with 10 records, looking something like this:



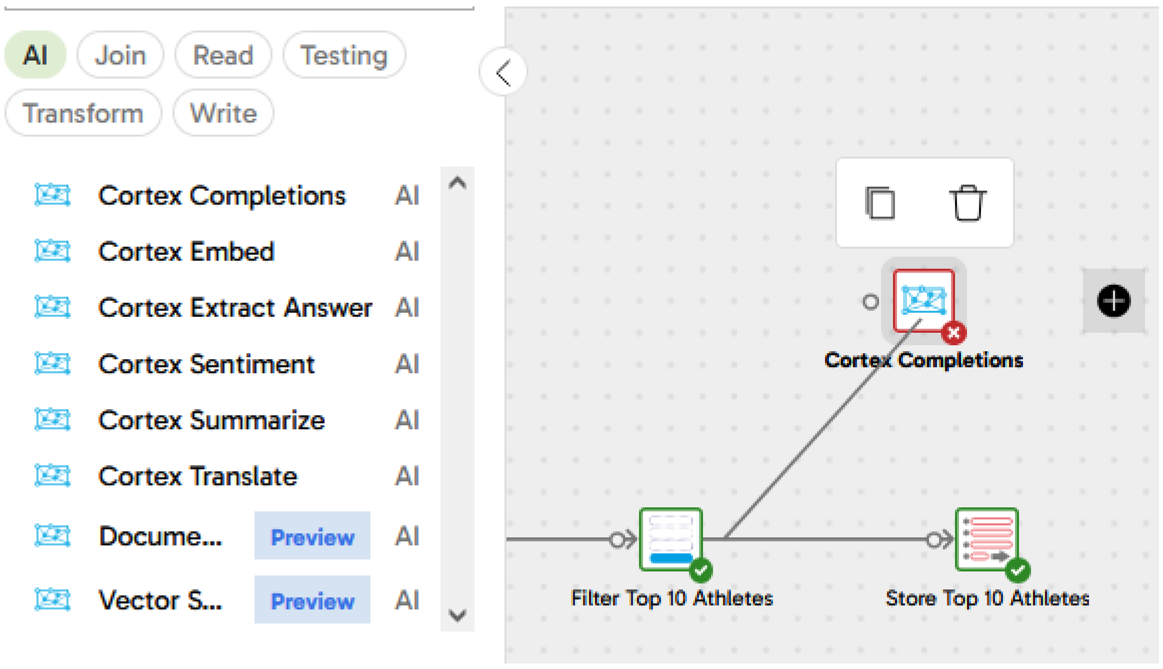
TotalPointScore	Name	AthleteRank
17	Michael Fred Phelps, II	1
14	Kathleen Genevieve "Katie" Ledecky	2
13	Simone Arianne Biles	3
11	Katinka Hossz	4
10	Simone Ashley Manuel	5
9	Jason Francis Kenny	6
9	Madeline Jane "Maya" DiRado	6
9	Usain St. Leo Bolt	6
9	Ryan Murphy	6
9	Danuta Kozk	6

Once you have the data looking like this, it's time to save it to the database. Add a Rewrite Table to the end of the pipeline. This will perform a Create Table As Select (CTAS) at runtime. If you prefer, use Maia again, instructing it to: **Store the output in a new table named top_10_athletes_summer_2016.**

Run the pipeline now. Maia may prompt you to do this anyway.

This new table will be used shortly with OpenAI, but meanwhile the lab will spend some time exploring how to use Snowflake Cortex to interact with a Large Language Model from **inside** the database.

Add a new Cortex Completions component to the canvas, connecting it downstream of the component immediately before the Rewrite Table.



Configure the new component as follows:

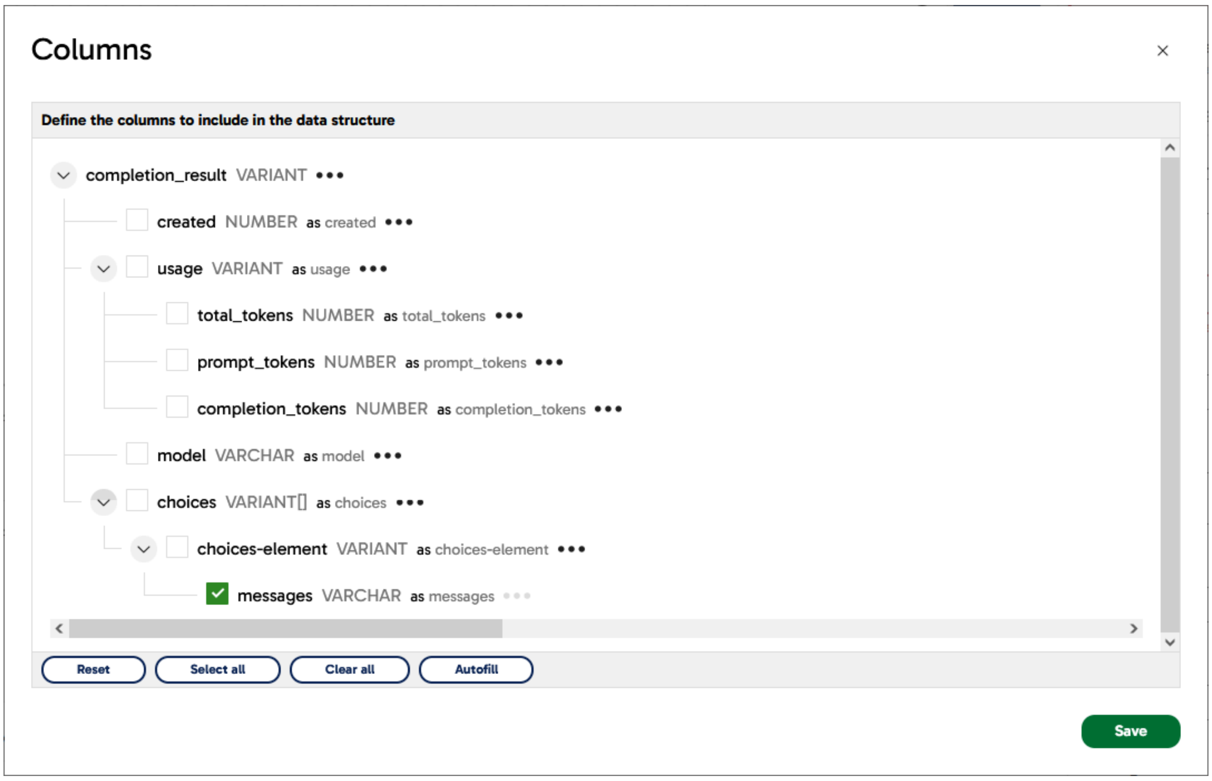
- Model: **mistral-7b**
- System Prompt: **You are a reporter on athletic events**
- User Prompt: **Write a short report on how this named athlete performed in the Summer 2016 Olympic Games**
- Inputs: **Name / Athlete Name**

Once the Cortex Completions component is valid (green border), use the Data Sample panel to preview the data. After a couple of seconds while the LLM calculates its response, you'll find the same 10 records again - this time with a new column named completion_result. The Data Sample is showing the output of live calls to Snowflake's CORTEX.COMPLETE SQL function. The data should look something like this:

```
{
 "choices": [
  {
   "messages": " Michael Fred Phelps ... swimmer of all time."
  }
 ],
 "created": 1729262047,
 "model": "mistral-7b",
 "usage": {
  "completion_tokens": 335,
  "prompt_tokens": 56,
  "total_tokens": 391
 }
}
```

You'll notice immediately that this is a semi-structured column, which needs some processing to extract the element of interest.

Matillion offers the Extract Nested Data component for this purpose, so attach one after the Cortex call, and configure the **Columns** property by first choosing **Autofill**, then deselecting all columns except for the innermost **messages** field.



Columns

Define the columns to include in the data structure

```
completion_result VARIANT •••
    [ ] created NUMBER as created •••
    [ ] usage VARIANT as usage •••
        [ ] total_tokens NUMBER as total_tokens •••
        [ ] prompt_tokens NUMBER as prompt_tokens •••
        [ ] completion_tokens NUMBER as completion_tokens •••
    [ ] model VARCHAR as model •••
    [ ] choices VARIANT[] as choices •••
        [ ] choices-element VARIANT as choices-element •••
            [✓] messages VARCHAR as messages •••
```

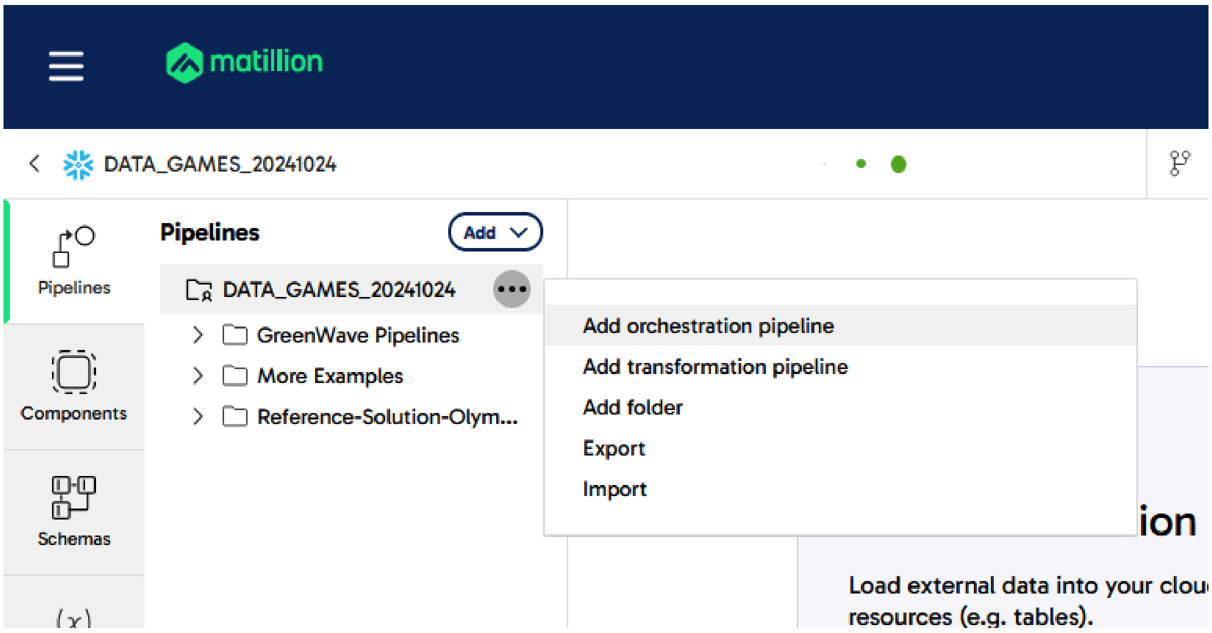
When you sample the Extract Nested Data component you'll find a new messages column at the end, containing just the text from the LLM. If you want, you can save this as another new table, using another Rewrite Table component.

Note that every time you run a data sample at this point, the entire data transformation runs again, and it may take a few seconds to complete while the LLM is invoked for every record.

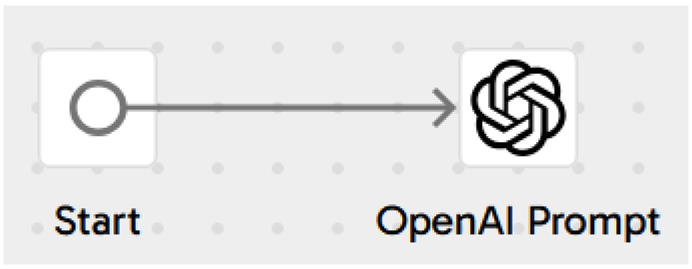
Olympic Data - Utilize OpenAI

The previous data pipeline ended up with sampling data from Snowflake Cortex, and by saving a list of 10 athletes into a new database table. In this last part of the guided lab, you'll run the contents of this table through another LLM: OpenAI this time.

This needs another new **orchestration** pipeline, so go ahead and add a new one by expanding the Pipelines panel, and choosing Add Orchestration Pipeline.



This pipeline only needs a single component: an OpenAI Prompt. Drag it into place and link it after the Start:



Configure the OpenAI Prompt component like this:

- Model: **gpt-3.5-turbo** (you can take the time to experiment with this choice)
- API Key: **OpenAI-key** (from the dropdown)
- Temperature: 1
- Top P: 1
- N: 1
- Source / Table: **top_10_athletes_summer_2016**
- Key Column: **Rank**
- User Context: **You are a reporter on athletic events. Write a short bio on how the named athlete performed in the Summer 2016 Olympic Games**
- Inputs: **Name**

- Output Format: **TEXT**
- Destination / Table: **top_10_athletes_summer_2016_OpenAI**
- Create Table Options: **Replace if Table Exists**

Run the pipeline, and wait for it to run all the rows through the OpenAI LLM.

Once it has finished, you'll need to add a new transformation pipeline to view the data in the new top_10_athletes_summer_2016_OpenAI table. It's common to have a temporary transformation pipeline on standby for tasks like this.

Why not experiment with different OpenAI models, saving them to different tables. This is a good way to gain insight into the cost/performance tradeoff between LLMs.

Hint: put the name of the LLM into a column, or into the table name, then join all the tables in a transformation pipeline on the common **Name** column.

Can Maia build the entire pipeline?

Spoiler alert.. the answer is yes 😀

It may take a few minutes to build as it works through the schema and the components. You can sample from components while Maia is building. You can interrupt the automation at any time by following the Refresh button in the Maia panel.

Close any open pipelines, and give Maia this prompt:

I need information on the top 10 performing athletes in the Summer 2016 games based on total medal points.

The olympic_athletes table contains one record per olympic athlete, per event, in every Olympic games. Note the table and column names are in mixed and lowercase.

Join it to the noc_region table which contains the NOC region codes. In noc_region, use the Notes field as the Region name if present, otherwise fall back to the Region column.

From this joined data, allocate 3 points for a gold medal, 2 for silver and 1 for bronze.

For the top 10 athletes by medal points, use the llama3-8b LLM to write a short summary of each athlete's performance at the Summer 2016 games.

I only need the text from the nested data returned by the LLM.

More data engineering scenarios

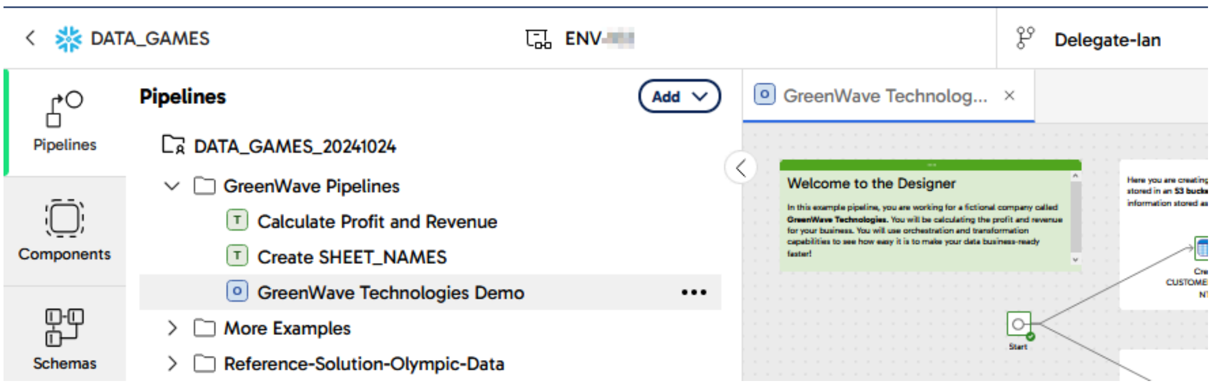
Once you have finished the Olympic data pipelines, you are free to use the lab time to experiment with more data samples. The Matillion team members will remain online to help and answer any questions.

Three more data sets are available:

- GreenWave Pipelines
- Bank Complaints
- Barista Reviews

GreenWave Pipelines

In the Pipelines panel, under GreenWave Pipelines, open and run GreenWave Technologies Demo orchestration pipeline.



Welcome to the Designer

In this example pipeline, you are working for a fictional company called **GreenWave Technologies.** You will be calculating the profit and revenue for your business. You will use orchestration and transformation capabilities to see how easy it is to make your data business-ready faster!

Note that this pipeline will not initially **validate** successfully, although it will **run** cleanly.

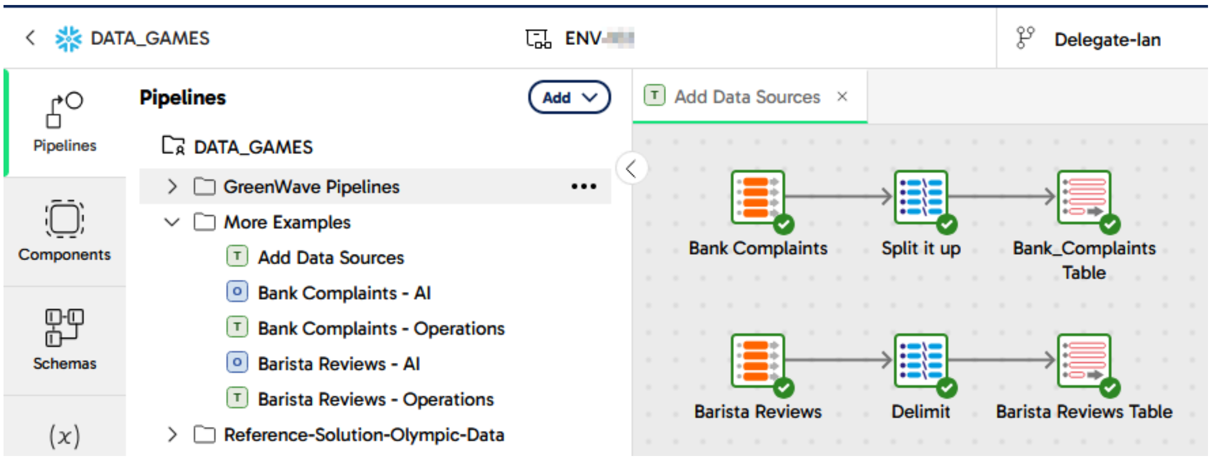
The pipeline creates new tables including:

- GW_CUSTOMER_ACCOUNTS, GW_ITEMS, GW_ORDERS and GW_ORDER_ITEMS (four source tables)
- GW_PROFIT_BY_ACCOUNT (a calculated table)
- GW_SHEET_NAMES (metadata)

Spend some time to look at how the pipeline works. Especially the iterator component, which creates three of the source tables inside a loop.

Bank Complaints and Barista Reviews

Under More Examples, open and run the Add Data Sources transformation pipeline:



The pipeline creates new tables including:

- Bank_Complaints
- Barista_Reviews

These tables contain artificially generated sample data containing unstructured text: bank complaints and barista reviews respectively.

To process the data, run:

- **Bank Complaints - AI** then **Bank Complaints - Operations**
- **Barista Reviews - AI** then **Barista Reviews - Operations**

These pipelines use generative AI to process the unstructured text, and arrive at operational insights and actions.

Spend some time to look at how the pipelines work. Experiment with the different forms of generative AI that are available.

After the event - Option to continue independently

After the event has finished, your Matillion account will remain fully available for several days. You are free to work and experiment independently, perhaps uploading your own data.

Note the setup for this lab does **not** give access to any private data. If you were using your own Matillion account during a POC, you would set it up with access to your own cloud storage and your own VPC. For this public facing lab, Matillion only has access to public resources.

The OpenAI key will remain active, and you will continue to have access to Snowflake Cortex.

Your access to this lab account will disappear after a few days, when the environment gets recycled for following events.

If you want to keep your work, take a moment to export your pipelines. This will save what you have built as a .zip file that you can download.

Later, if you'd like to run your own trial of the Matillion Data Productivity Cloud, and continue where you left off:

- Log into to the Matillion Hub again
- Create a new Account
- Set up your own Project within the Account
- Import your pipelines from the saved .zip file

Have fun!