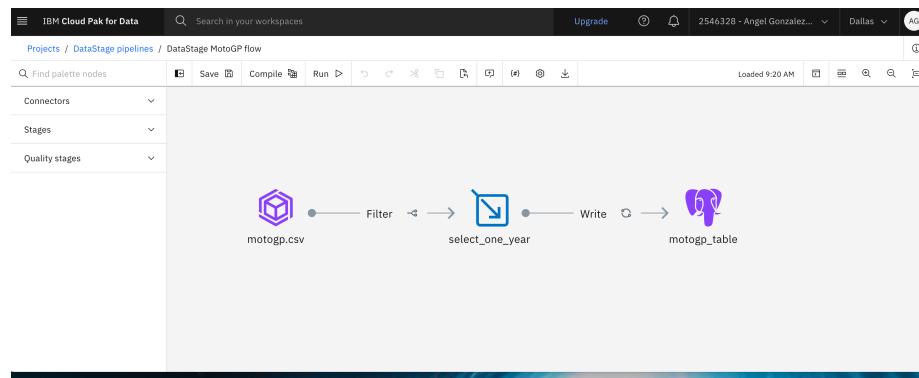


DataStage Pipelines

In this section we will build a DataStage job that performs the same operations as the two pipelines reviewed so far.

1. Build a DataStage Job

This is the DataStage job that we will create from scratch. As you see, it reads a csv file, performs some filtering and write the results on a Postgres :



1.1. Create a data asset for the input file motogp.csv

This git repository contains a small file motogp.csv (2500 records) that we will use as input file.

Go to the main project panel and upload the file into the project to create an data asset. Just follow the instructions on the following picture and leave all fields you see with the default values:

The screenshot shows the "Assets" tab of the IBM Cloud Pak for Data interface. A red arrow points to the "Assets" tab in the navigation bar. Another red arrow points to the "Import assets" button. A third red arrow points to the "New asset" button. A fourth red arrow points to the "Drop data files here or browse for files to upload" field, which has a dashed border. A fifth red arrow points to the "drag and drop the csv file here" instruction above the upload field.

Name	Last modified	⋮
AngelFlow DataStage flow	5 days ago Modified by you	⋮
DataStage MotoGP flow DataStage flow	5 days ago Modified by you	⋮
Postgres Connection	5 days ago Modified by you	⋮
project_cos_connection Connection	5 days ago Modified by you	⋮
motogp.csv CSV	5 days ago Modified by you	⋮

1.2 Create a connection asset for Postgres

On the same project main panel of the picture above, click on new asset. Select an asset connection:



Type the required values as shown on the next screenshot and test the connection. Do not forget to save the asset.

The screenshot shows the 'Edit connection: PostgreSQL' dialog. On the left, a sidebar lists 'Connection overview', 'Connection details', 'Credentials', 'Certificates', 'Other properties', 'Private connectivity', and 'Location and sovereignty'. The main area has tabs for 'Connection details' and 'Credentials'. Under 'Connection details', there are fields for 'Name' (Postgres), 'Database (required)' (postgres), 'Hostname or IP address (required)' (pg-nodeport-postgres.itzroks-1100005cc8-2lbzmg-6ccd7f378ae819553d37d5), and 'Port (required)' (30208). Under 'Credentials', there are fields for 'Username (required)' (postgres) and 'Password (required)' (redacted). A 'Test connection' button is located at the top right of the dialog. Red arrows point to each of these fields with descriptive labels: 'enter a name' for the Name field, 'default database name: postgres' for the Database field, 'the route to the postgres service (Postgres external hostname)' for the Hostname or IP address field, 'the external Postgres port' for the Port field, 'default user name: postgres' for the Username field, and 'default password: postgres' for the Password field.

1.3 Create a DataStage flow

Again, on the main project pannel, click on new asset and search for DataStage:



Type a name so that you can recognize it later in the Databand GUI and click on create.

The screenshot shows the 'Create a DataStage flow' dialog. On the left, there is a sidebar with '+ New' and 'Local file' options. The main area is titled 'Define details' and has a 'Name' field containing 'DataStage MotoGP Flow'. Below the name field is a 'Description (optional)' section with a text area. At the bottom, there are three buttons: 'Cancel', 'Back', and a blue 'Create' button.

1.4 Add the csv file to the flow

An empty workspace will be shown after creating the job. Now, just click on asset browser:



Add the `motogp.csv` file as instructed in the next screenshot:



The workspace will show an icon representing the input file.

1.5 Add the Postgres table to the flow

In this step, we will add the destination of the data. As we have created an asset for Postgres, we can add it from the access browser again:



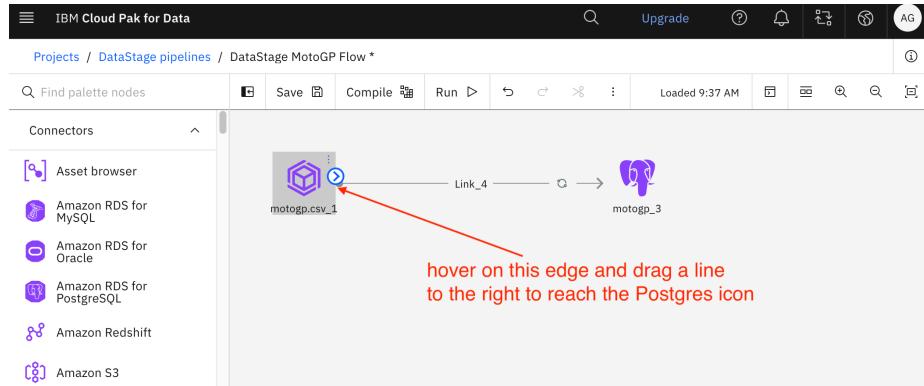
Follow the instructions on the screenshot to incorporate the `motogp` table to the

flow:



1.6 Connect the two assets

We will connect the input file with the table as follows:



1.7 Add a filter

We don't want to load the full data, but only one season of the championships. So, we add a filter between the assets as follows:



If you did it correctly, it will look like this:



1.8 Specify the filter condition

We will select a portion of the csv file by typing an SQL-like clause, although the input data is not a relational table. Click on the filter icon and go to edit:

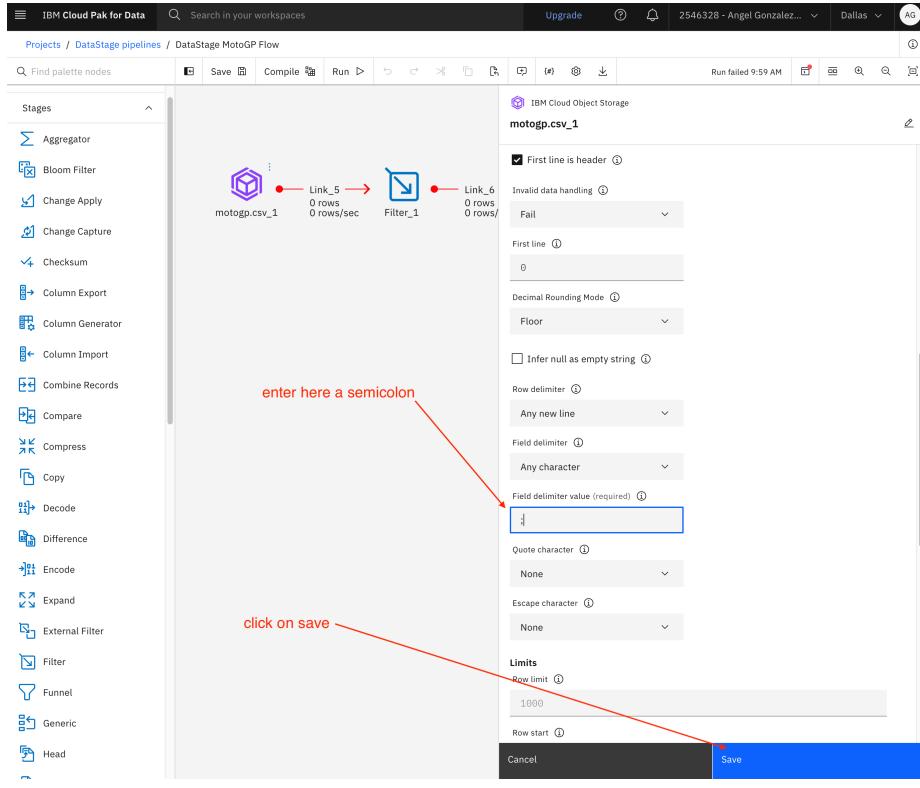
The screenshot shows the IBM Cloud Pak for Data interface with the title bar "IBM Cloud Pak for Data". Below it, the path "Projects / DataStage pipelines / DataStage MotoGP Flow *". On the left, a sidebar titled "Stages" lists various components: Aggregator, Bloom Filter (selected), Change Apply, Change Capture, Checksum, Column Export, and Column Generator. In the center, a flow diagram shows a node labeled "motogp.csv_1" connected to a "Filter_1" stage. The "Filter_1" stage has tabs for "Stage", "Input", and "Output". The "Output" tab is selected, showing a table with one row: "Link_6" under "Output link", "Primary" under "Target link type", and "Where clause" empty. A red arrow points to the "Edit" button in the top right corner of the output panel.

A new panel will open and that is the place to enter the filter condition:

The screenshot shows the "Filter_1" configuration dialog box. At the top, it says "Edit where clause. For example, city='Providence' OR city='San Jose'". Below is a search bar and a table with columns "Output link" (Link_6) and "Target link type" (Primary). A red arrow points to the "Where clause" input field, which contains "Season = 2020". Another red arrow points to the "Apply and return" button at the bottom right. A third red arrow points to the "Cancel" button at the bottom left. A fourth red arrow points to the "click here" label at the bottom left of the dialog.

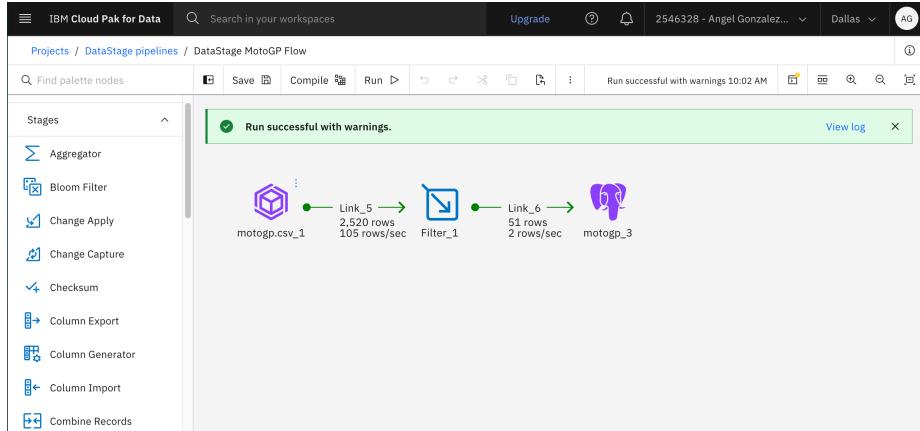
1.9. Correct the delimiter character option of the csv file

DataStage requires to enter the delimiter character of the CSV files. Just click on the motogp.csv icon of our flow, unfold the properties panel and find the place to specify a semicolon like follows:



10. Test the flow

To run the flow just click on the arrow at the top and wait about one minute. If everything went well, you will see something like this:



The warnings can be ignored. They simply indicate that the job will not be parallelized, which is OK for our data volume.

1.11. Schedule the flow

Go to the main project panel and proceed as indicated in the following screenshot:

The screenshot shows the 'IBM Cloud Pak for Data' interface with the 'Projects / DataStage pipelines' path selected. The 'Assets' tab is active. On the left, there's a sidebar with '13 assets' and categories like 'All assets', 'Asset types', 'Data access', 'Data', and 'Flows'. In the main area, a table lists assets: 'AngelFlow DataStage flow' (modified 5 days ago), 'DataStage MotoGP flow DataStage flow' (selected and highlighted with a red box, modified 5 days ago), 'Postgres Connection' (modified 5 days ago), 'project_cos_connection Connection' (modified 5 days ago), 'motogp.csv CSV' (modified 5 days ago), and 'CREDIT_SCORE_shaped CSV' (modified 1 week ago). To the right of the table is a context menu with options: 'New job' (highlighted with a blue box), 'Duplicate', 'Download', and 'Delete'. Red annotations include a curved arrow pointing from the text 'locate the flow' to the selected asset in the list, and another curved arrow pointing from the text 'click on new job' to the 'New job' button in the context menu.

The job schedule panel is very straightforward, just take care to specify the right schedule interval:

The screenshot shows the 'Create a job' wizard. The left sidebar has tabs: 'Define details' (selected), 'Settings', 'Schedule' (selected), 'Notify', and 'Review and create'. The 'Schedule' tab contains a 'Schedule' section with a 'Schedule' button (slid to the right), a 'Time zone: GMT+0100 (Central European Standard Time)' dropdown, and a 'Repeat' checkbox. Below it is a 'Repeat the job every' input field with '15' minutes specified. Other options include 'Start of schedule', 'Exclude days', and 'End of schedule'. At the bottom are 'Back' and 'Next' buttons.

And verify that the job will run as specified:

Job Details

Overview

1	Runs complet...
0	Runs failed
Repeat Every 15 minutes	

Runs (1)

Start time	Run name	Status	Duration	Asset type
Mar 22, 2023 10:10:22 AM Started by you	job run	Completed	00:01:28	DataStage

Items per page: 10 | 1-1 of 1 item | 1 of 1 page

2. Observe performance data

The DataStage performance data exhibit more variability due to the concurrent execution of other jobs dropping randomly the destination table in Postgres, which generated errors and interruptions.

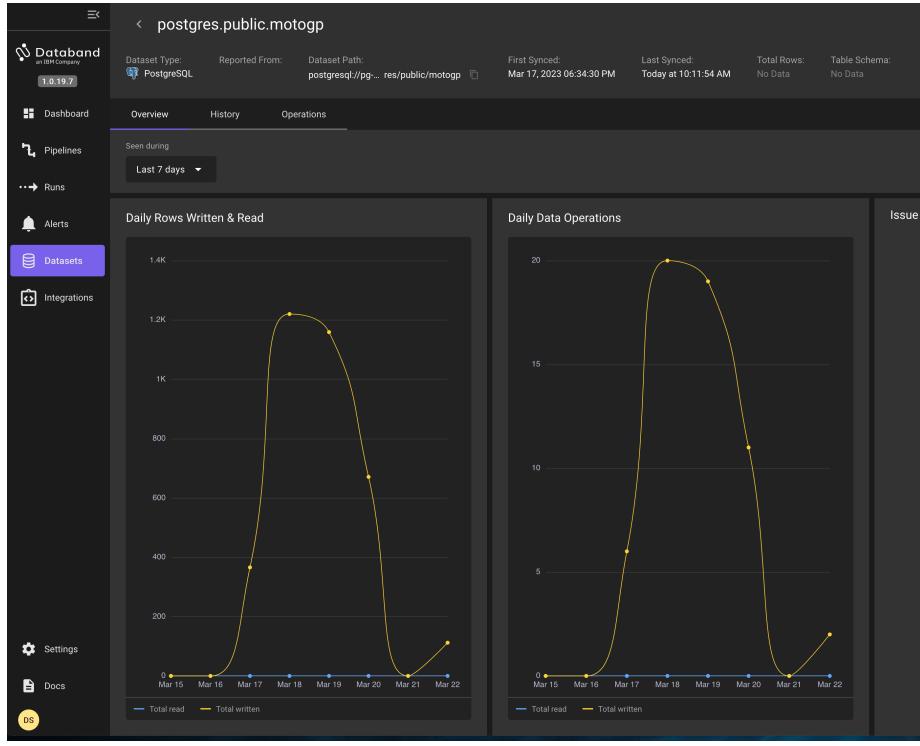
select the runs menu
choose a DataStage run with this name
inspect the read / write history

click here to see more historical data

Type	Datasets	Issue type	Operations	Schema	Records	History Trend
DataStage	DataStage_MotoGP_flow.cloudObjectStoragePX.motogp...	No issues	Write	6 columns	2,520 records	
DataStage	DataStage_MotoGP_flow.cloudObjectStoragePX.motogp...	No issues	Read	6 columns	2,520 records	
DataStage	DataStage_MotoGP_flow.cloudObjectStoragePX.motogp...	No issues	Write	6 columns	61 records	
DataStage	DataStage_MotoGP_flow.cloudObjectStoragePX.motogp...	No issues	Read	6 columns	2,520 records	
PostgreSQL	postgres.public.motogp	No issues	Write	6 columns	61 records	
DataStage	DataStage_MotoGP_flow.cloudObjectStoragePX.motogp...	No issues	Read	6 columns	61 records	

Records per page: 25 | 1-6 of 6

Notice how the destination table is explicitly labeled and how the historical data is shown:



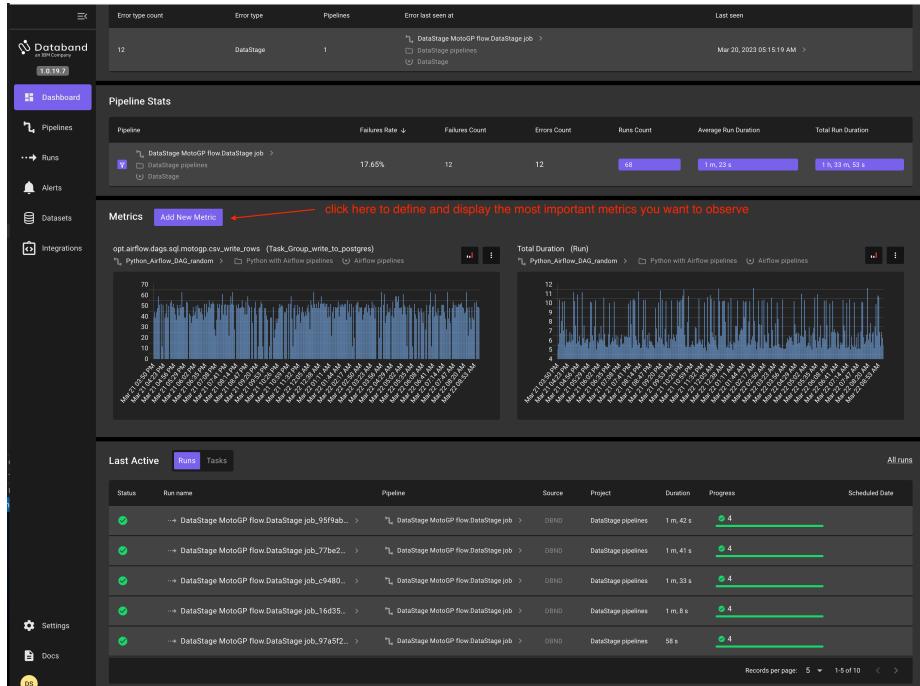
If you schedule at short intervals all the pipelines we built so far, a considerable number of runs and statistics will appear. Consider the use of filtering to find out what you are looking for:



See how we reduced the analysis from over 1000 pipelines to 68.



Remember that you can define the most important metrics to be shown in the dashboard:



We will explore the metrics and alerts in the next sections

Next Section: Alerts and Exceptions

Previous Section: Python on Airflow pipelines

[Return to main](#)