How to OPTIMIZE your data lake

Dremio training series

1 2 3 4

dremio

Dremio training series

1 Dremio for users & developers

- System and concept overview
- Using the Dremio UI
- Dremio ANSI SQL
- Jobs, queues and reflections
- ODBC, JDBC, and REST clients

2 Dremio for data engineers

- Semantic layer concepts
- Permissioning
- Overview of reflections
- Reflection best practices

3 Query analysis & performance

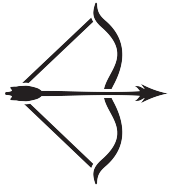
- Using query profiles
- Troubleshooting performance
- Query profile examples
- Query acceleration techniques

4 Dremio for administrators

- Deployment architecture
- Managing capacity and jobs
- System and query monitoring
- Metadata and reflection freshness
- Using the Dremio admin CLI

System and concept overview

Including

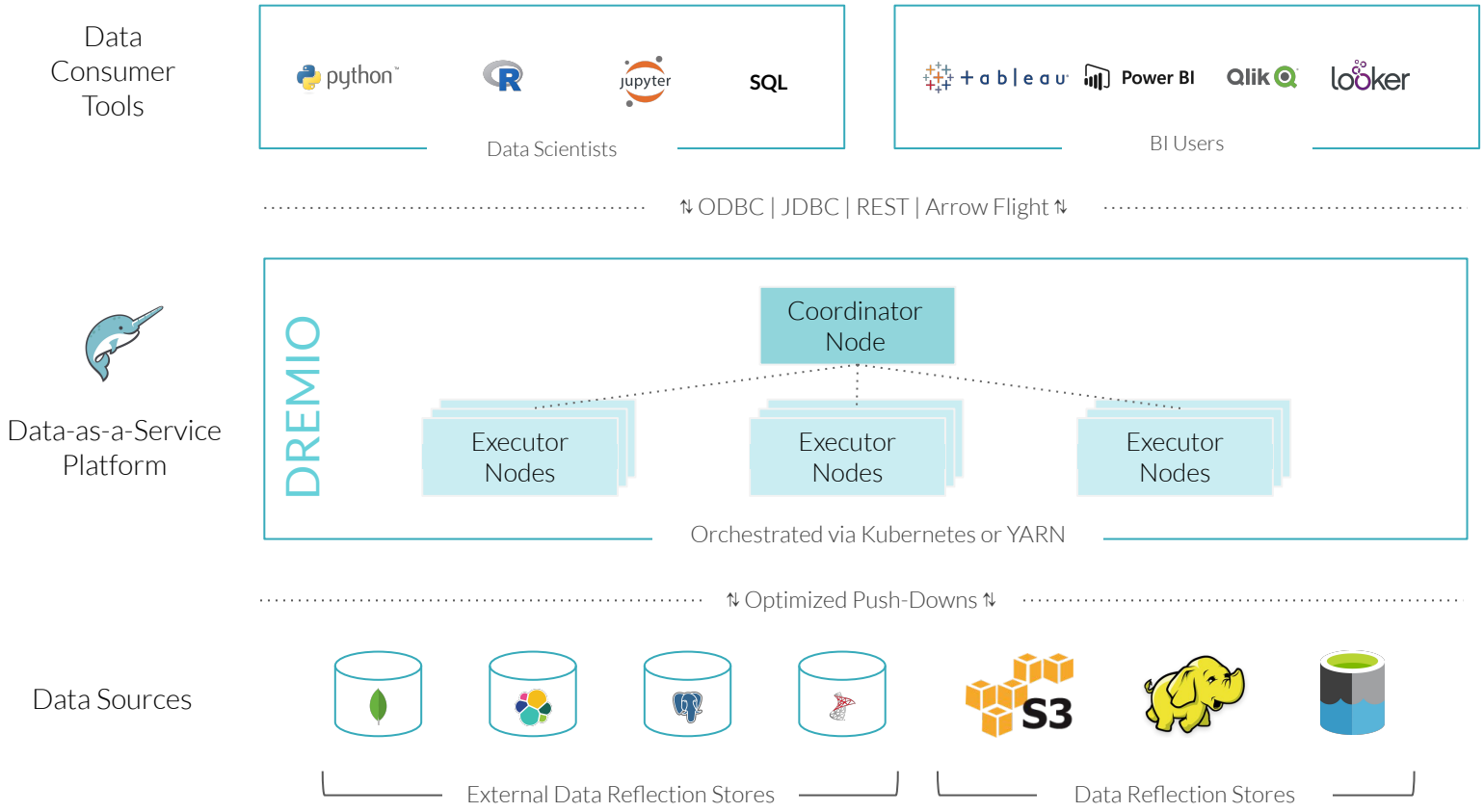


Arrow

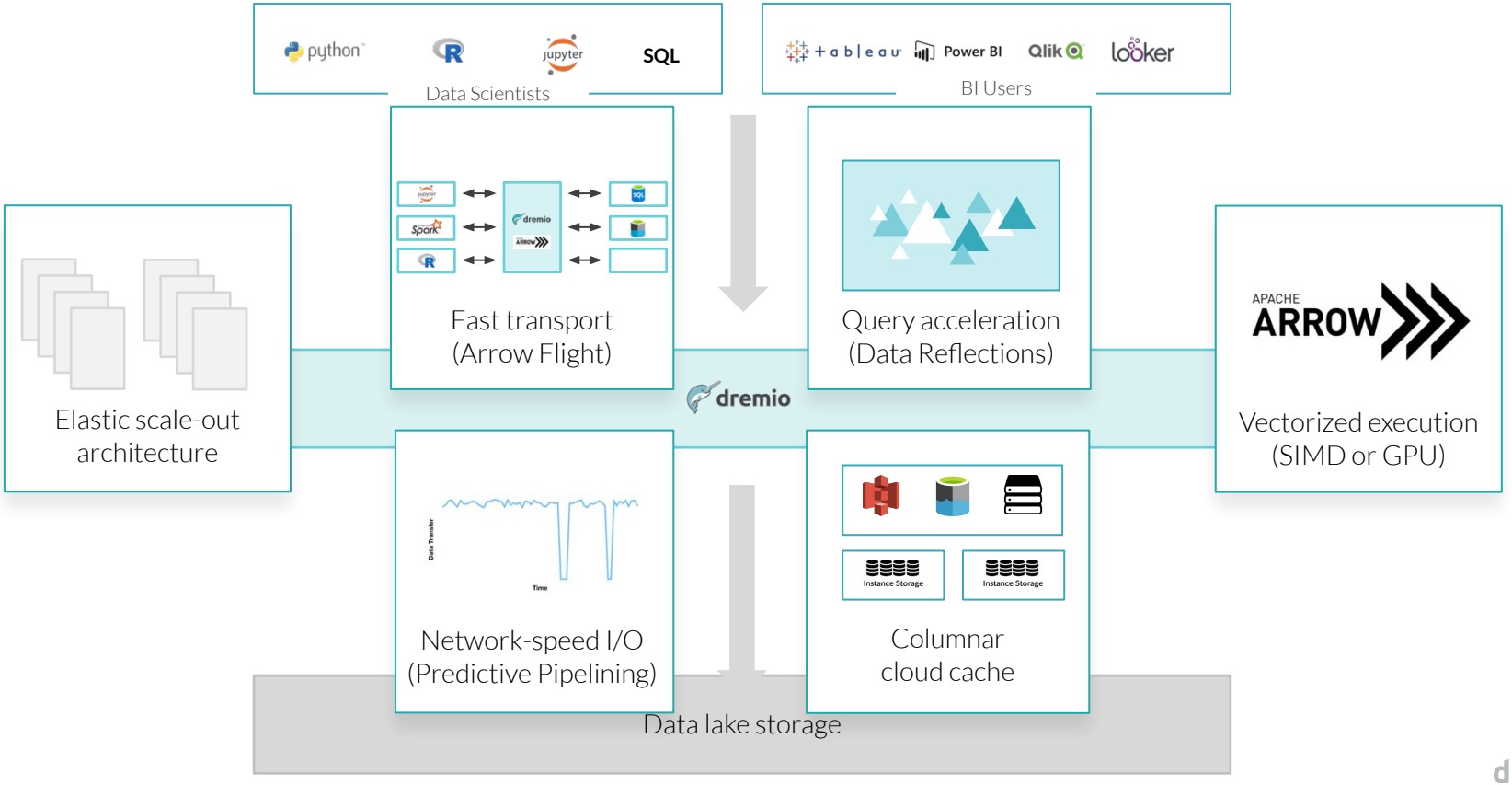
Parquet

Gandiva

Dremio deployment architecture



How Dremio delivers data lake performance



Key components of Dremio architecture

Coordinator node

Responsible for:

- Hosting ODBC/JDBC & HTTP endpoints
- Query planning
- Managing metadata

Metadata includes information about data sources, virtual datasets, access controls, job profiles and more

- Persisted to file system
- No dependence on external database
- Uses RocksDB & Lucene

Executor nodes

Responsible for:

- Query execution and data access
- Storing query results, user uploads, user downloads & data reflections (caches) in distributed file system

The number of executors in the cluster

- Scaled by workload size
- System can tolerate N-1 executor failures, but may be degraded before then

Key objects

Sources and datasets

Source - a data store connected to Dremio

Physical DataSet (PDS) - a source dataset - e.g. RDBMS table, HDFS directory

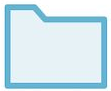
Virtual DataSet (VDS) - Similar to an RDBMS view, but with more features. VDSs have lineage

Reflection - Similar to a materialized view. Accelerates query performance. Transparent to users.

Spaces

Space - a location that organizes shared VDSs

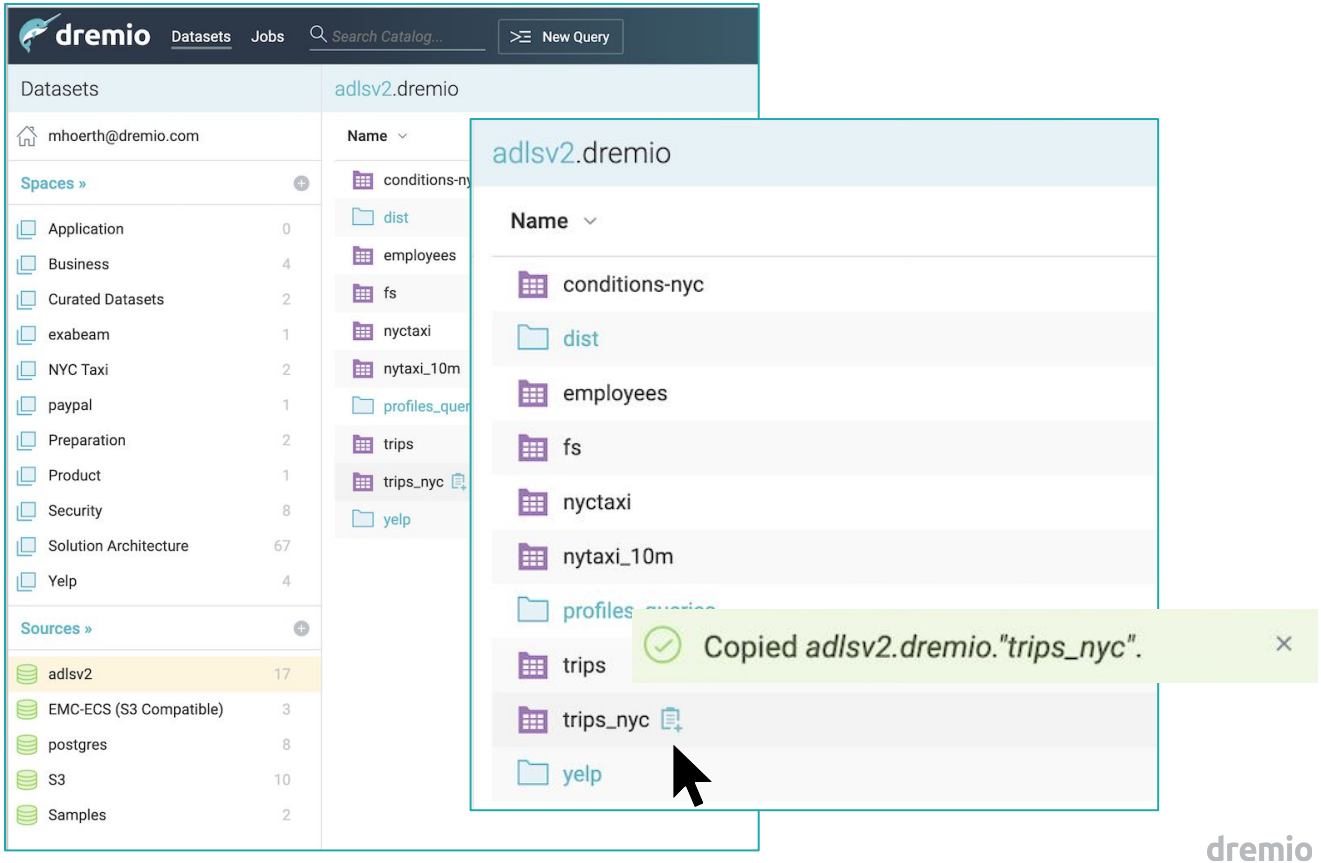
Home space - a private location for your own PDS and VDS



Folder - a subdirectory to organize items in a space

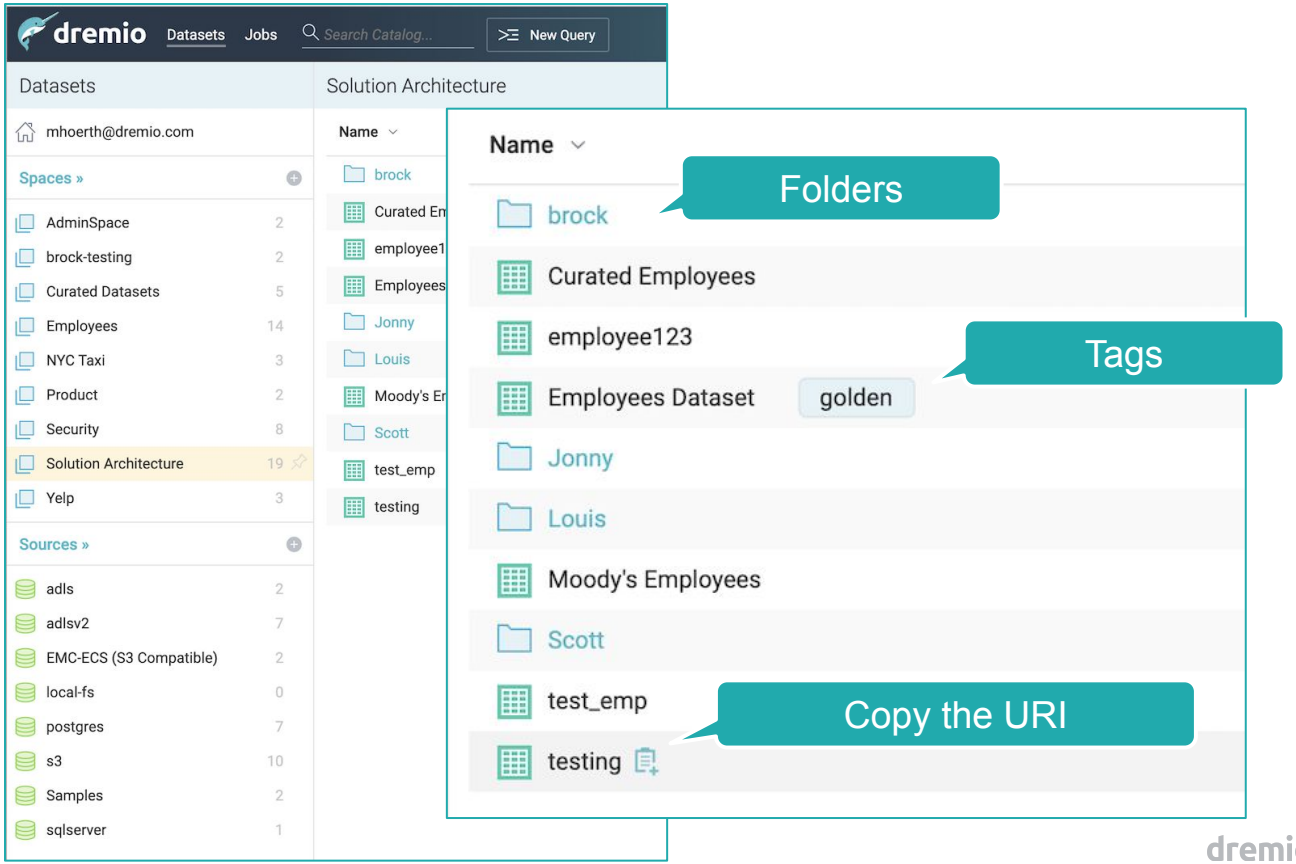
Physical Dataset (PDS)

- Data from a configured source, or uploaded file with data format
- Immutable, data curation contained in upper-level VDS definitions
- Identified with Universal Resource Identifier which includes data lake prefix/directory structure
- Access controlled by user & group, normally restricted in Dremio to administrator



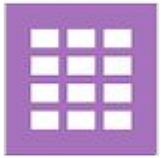
Virtual Datasets (VDS)

- View of the curated data from one or more PDS
- Data curation can include joins, filters, type casting, parsing, derived columns.
- Defined by SQL statements, which ideally include business logic for repeatable rapid execution
- Saved in Home space or shared space



PDS, VDS, and reflections

PDS



Data from a source or uploaded from a data file

VDS

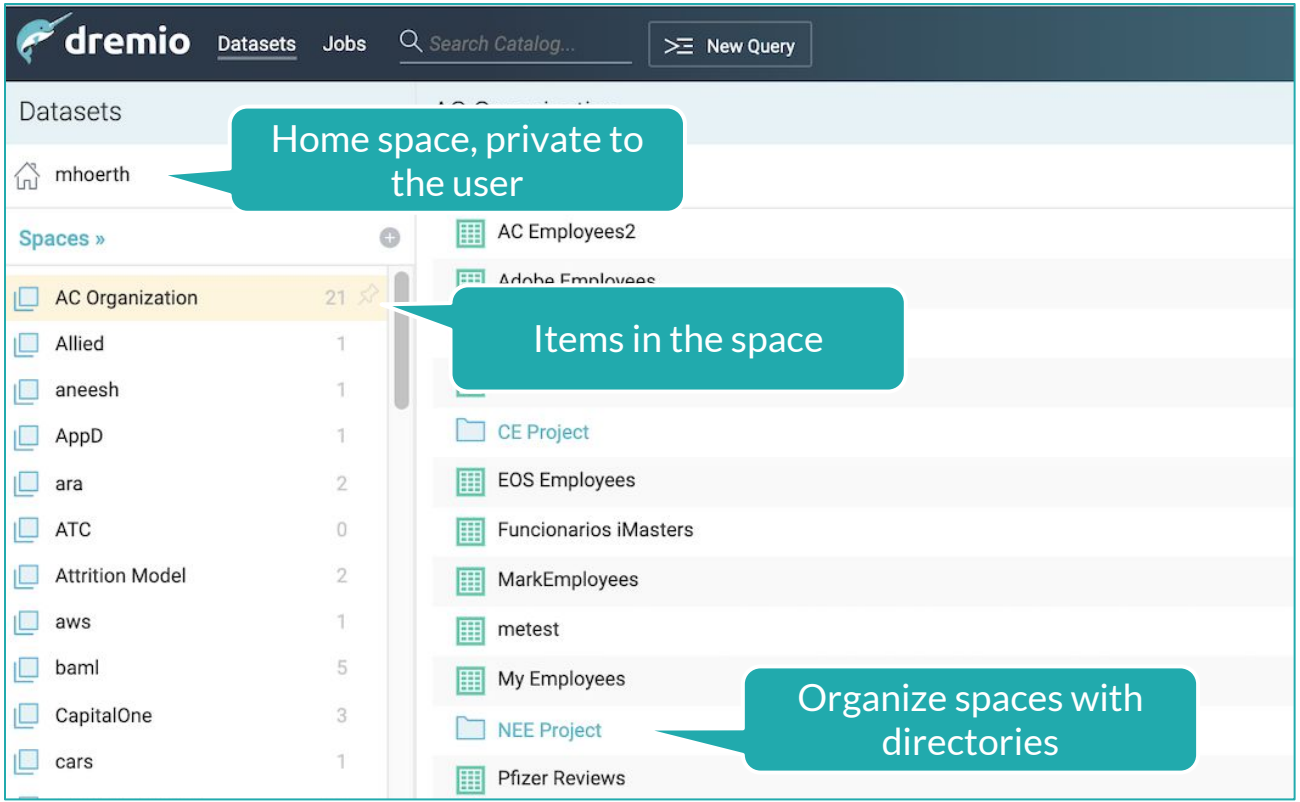
Like a view -- Curated from one or more PDS

Reflection

Like a materialized view -- accelerates queries on both PDS or VDS

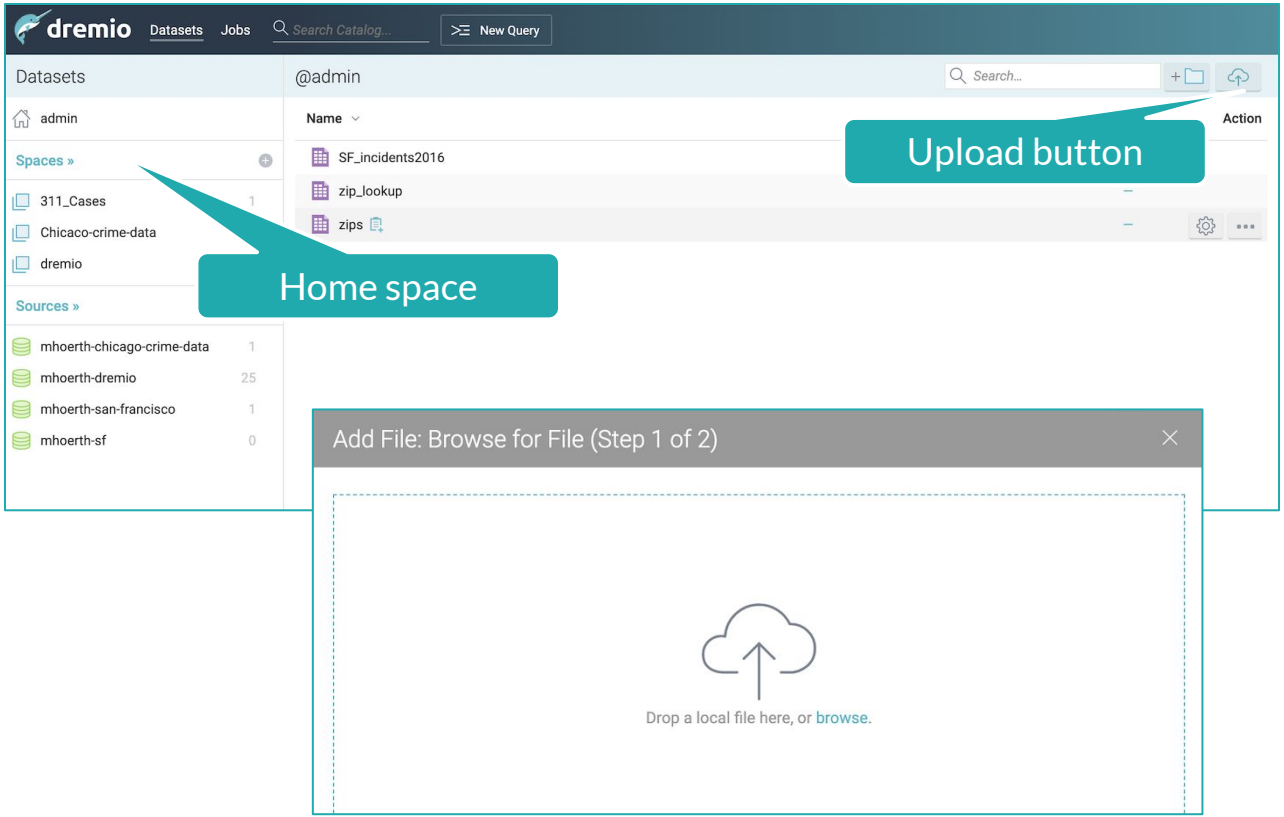
Spaces enable data sharing and organization

- Shared locations for saving virtual datasets
- Sharing can be configured for each space
- Group datasets by a common theme such as a project or geographic region
- Enables building a semantic layer
- Home Space is private user work area
- Users will not see spaces for which they have no authorization



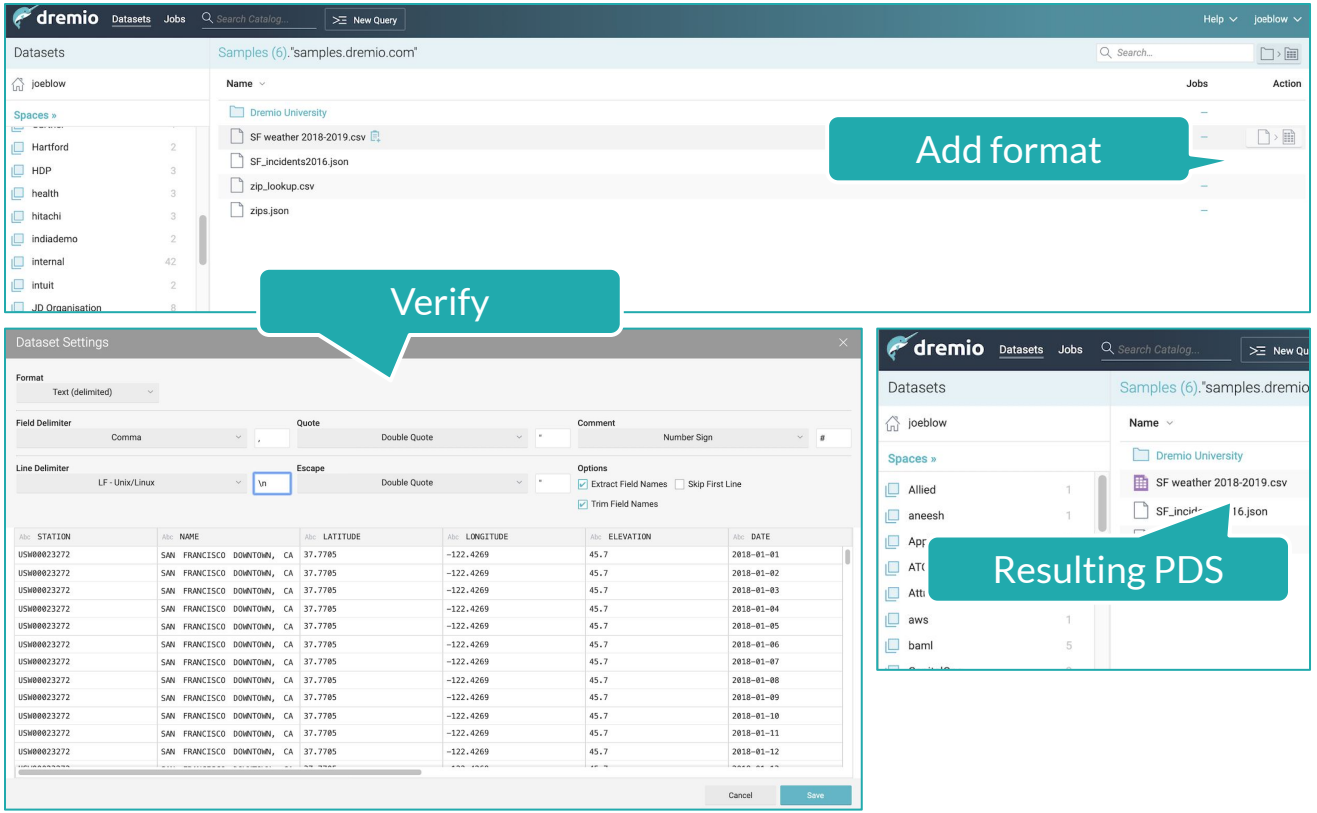
User uploaded data files

- Analyze auxiliary data on desktop / shared location by uploading it to Dremio
- Supported File Formats: Text (Delimited), JSON, Parquet, Excel, XLS
- Data is uploaded to user's personal space
- Can be shared with others by saving a Virtual Dataset in a shared Space



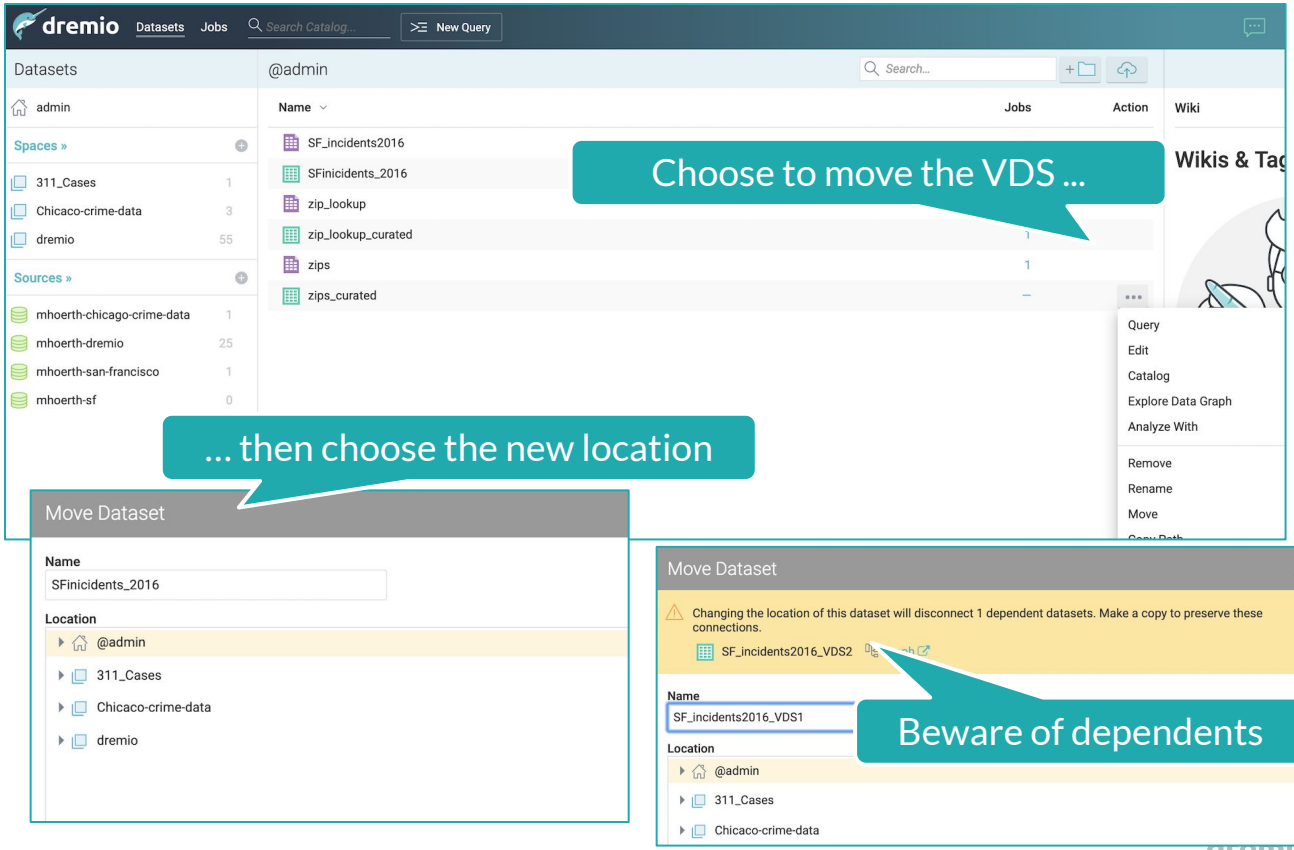
Promoting a file system source to a PDS

- Promote Individual File or Directory of similarly structured files into a Physical Dataset
- Upon promotion of Directory into PDS, sub-directory hierarchy is available as virtual columns (dir0, dir1 ..) in the PDS



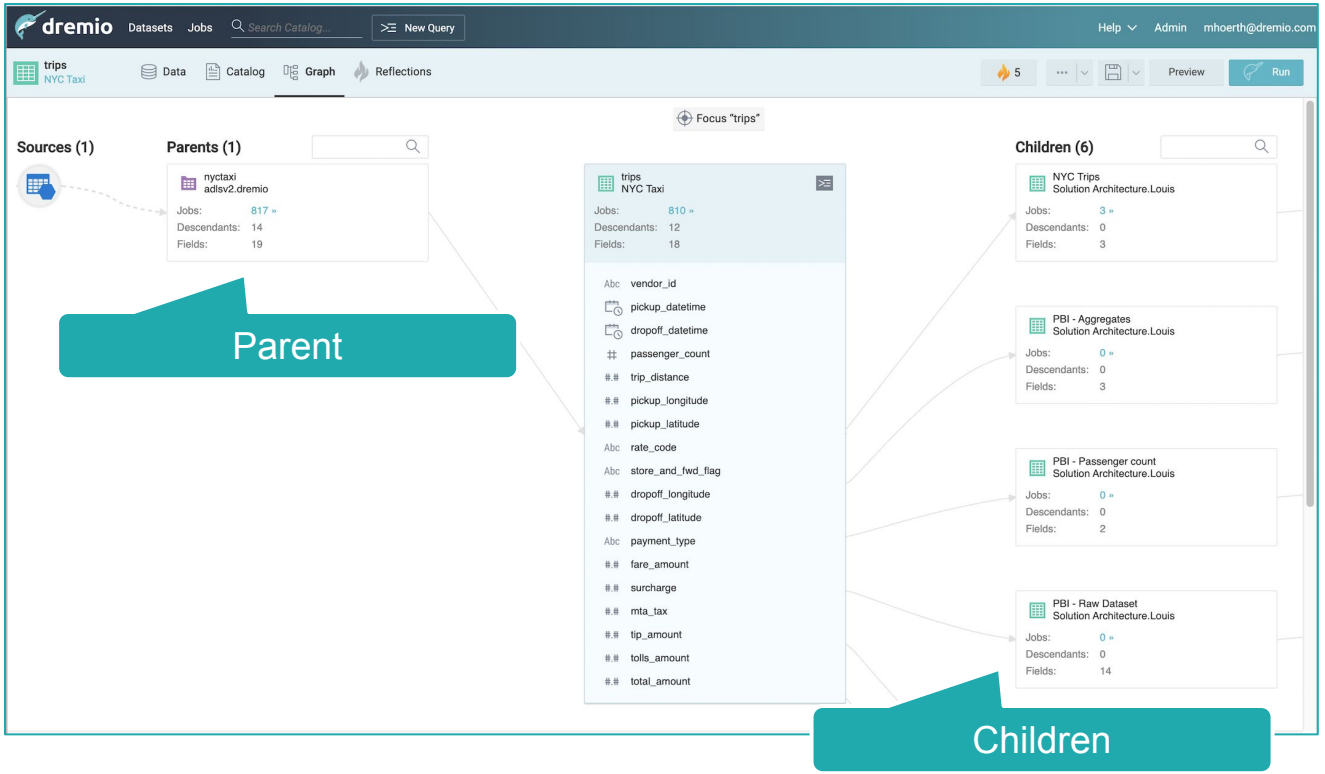
Moving to a new space

- Option to save a new VDS in your home space
- Iterate on the logic
- Move the VDS to a shared space when it's appropriate to share with others



Data lineage and searchable catalog

- See all downstream and upstream virtual and physical datasets
- Trace back to the original sources
- Fully navigate the lineage by clicking on a downstream or upstream dataset
- Add tags for additional richness in search



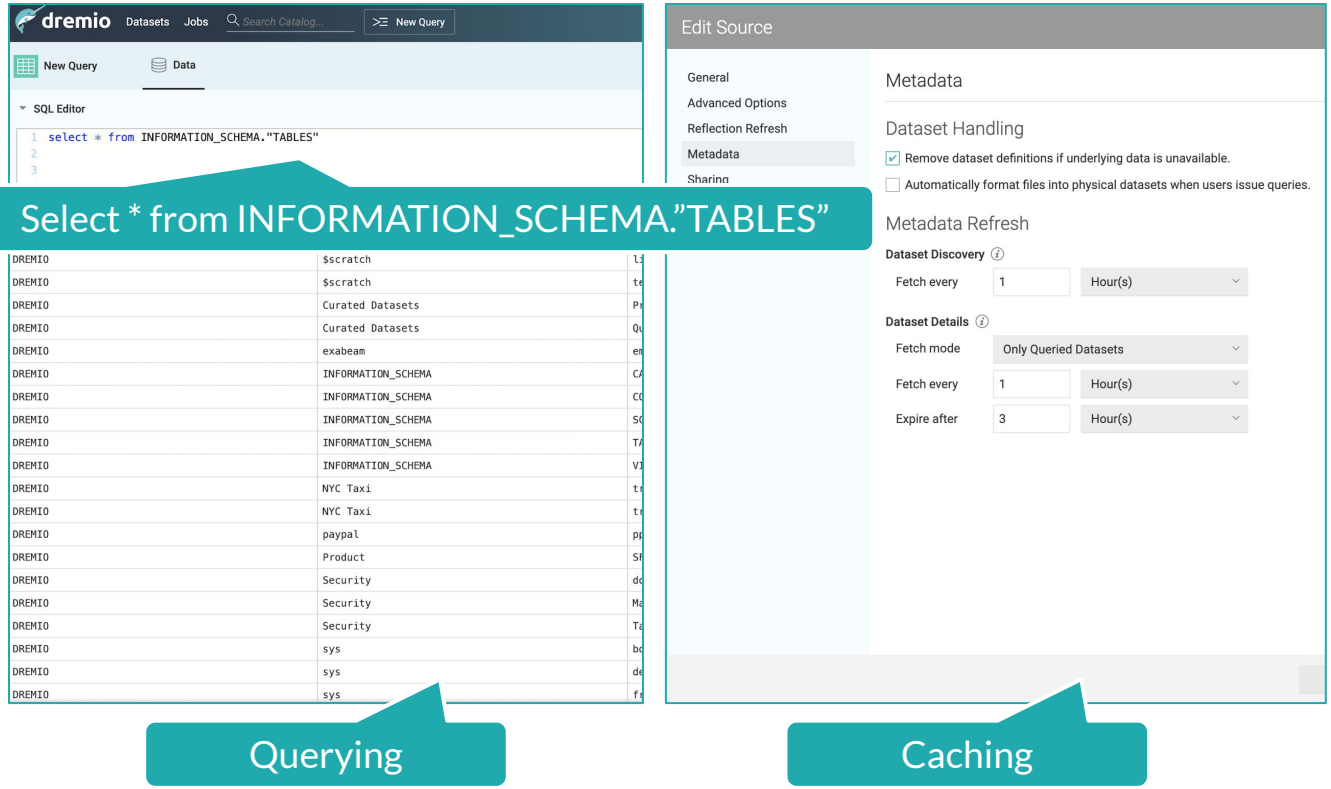
Caching and querying source metadata

Caching

- **Dataset Discovery:** interval for source DBs and tables. Lightweight.
- **Dataset Details:** Refreshes fields, types, files, etc. used for query planning.

Querying

- Dremio catalogs, schemas, physical datasets, virtual datasets and columns c using **INFORMATION_SCHEMA** queries.



Using the Dremio UI

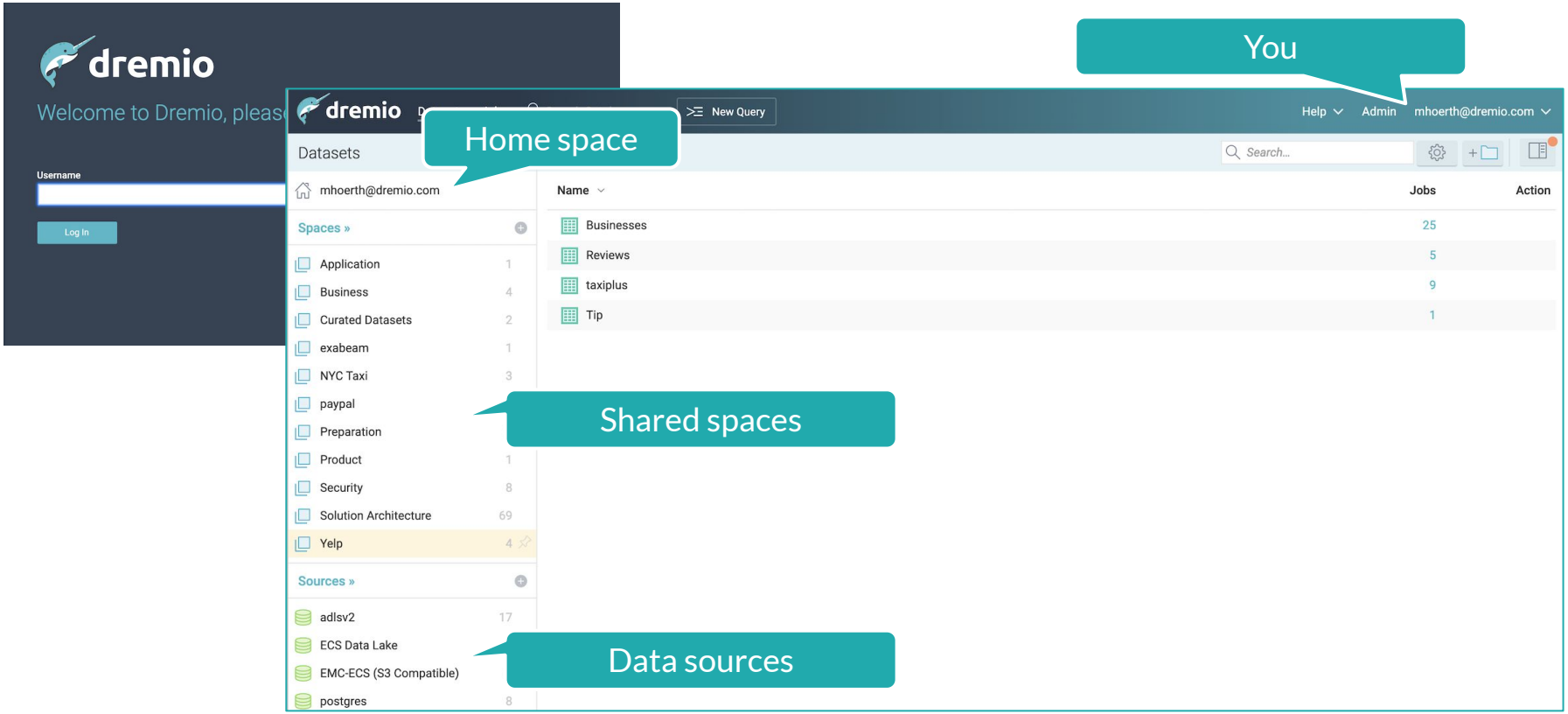
A few of the topics

Adding a column

Group By

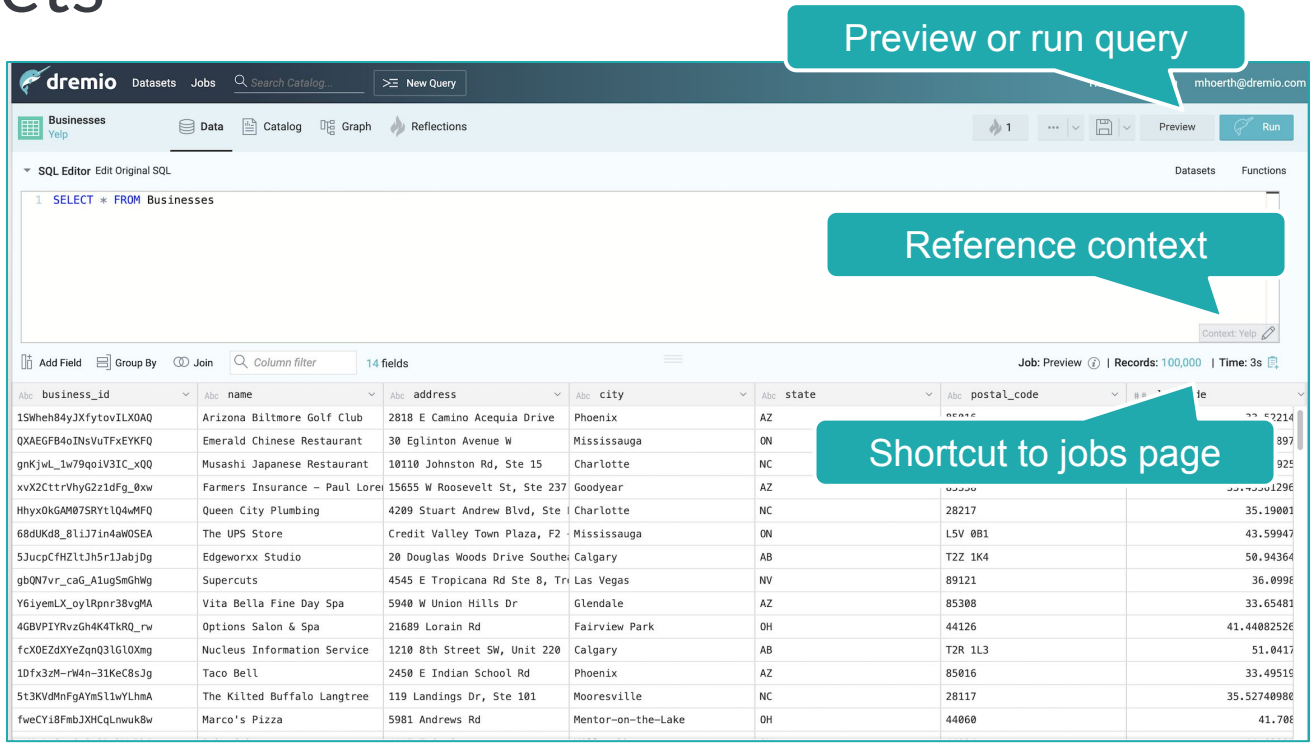
Join

The Dremio home screen



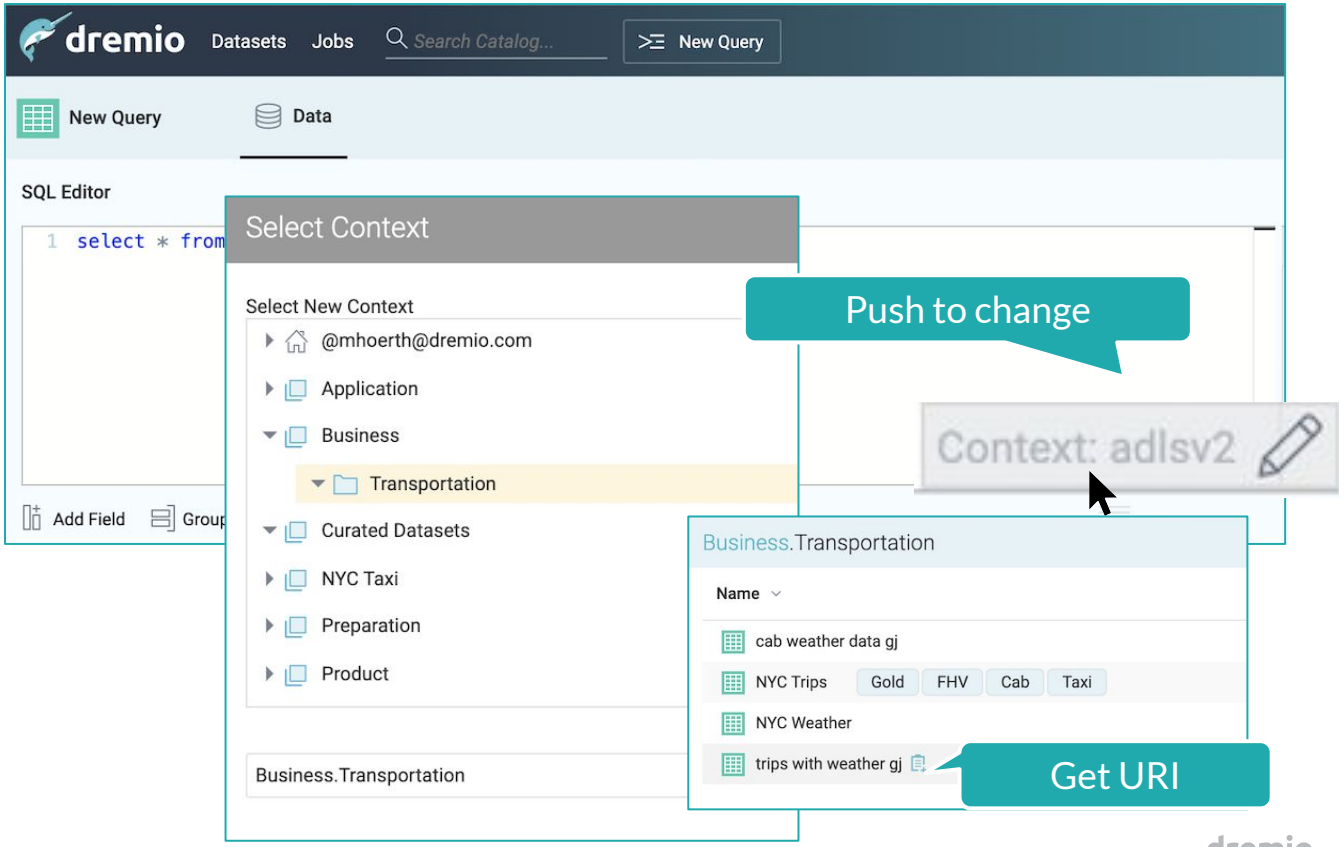
Viewing datasets

- View and edit the SQL that defines the dataset
- Run the query or “Preview” 10k rows
- Create new VDS built on this and possibly additional VDS
- Click the record count for a shortcut to the jobs page
- Column count of the dataset on the page
- Tabs to view the catalog and define reflections



Context vs. fully-qualified names

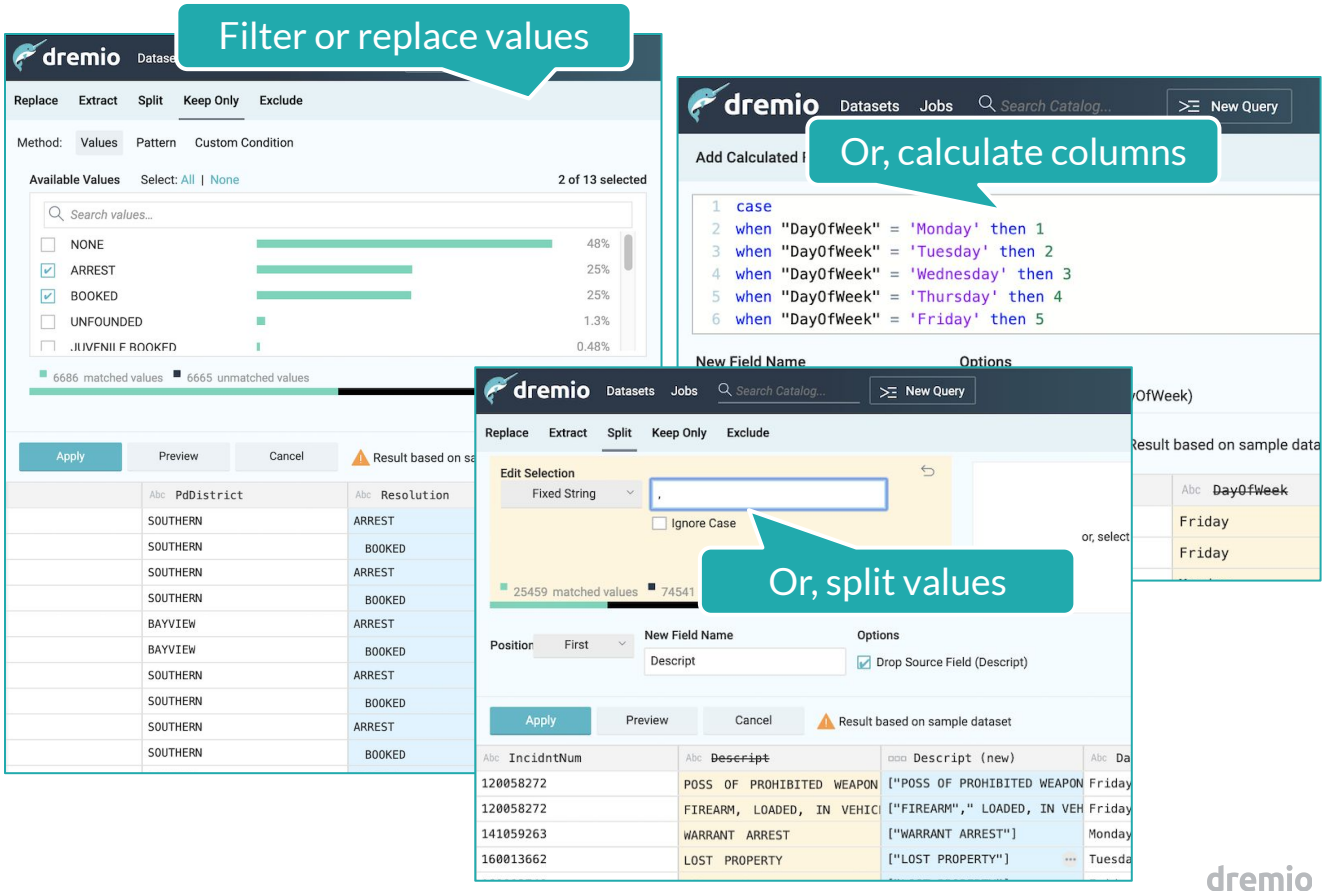
- PDS and VDS referenced by fully-qualified names, **source\space.folder.dataset_name**
- Optionally set a context on each reference
- Include the full URI in **FROM <tablename>** references for maximum portability to another space
- Get the full URI from the dataset clipboard icon



Editing VDS

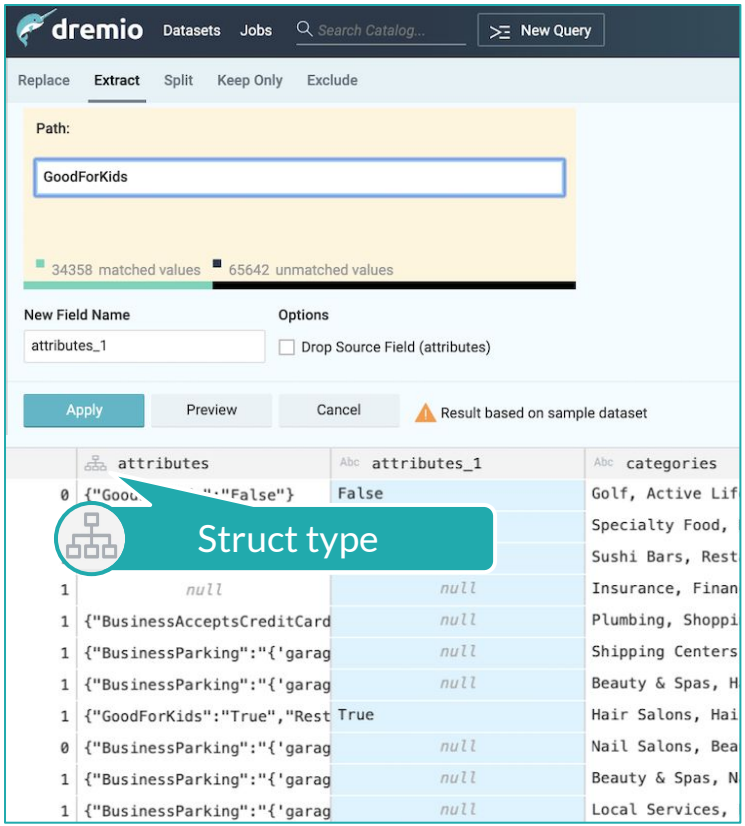
Curate data by applying data transformation functions

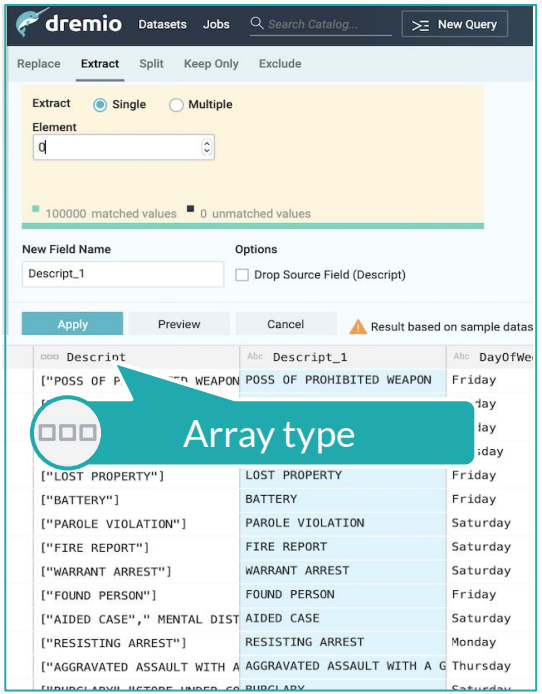
- Change type of a column
- Drop and rename columns
- Adding derived columns
- Split columns
- Extract strings from columns
- Handle complex fields (arrays/nested)
- Group by and join datasets



Working with complex types

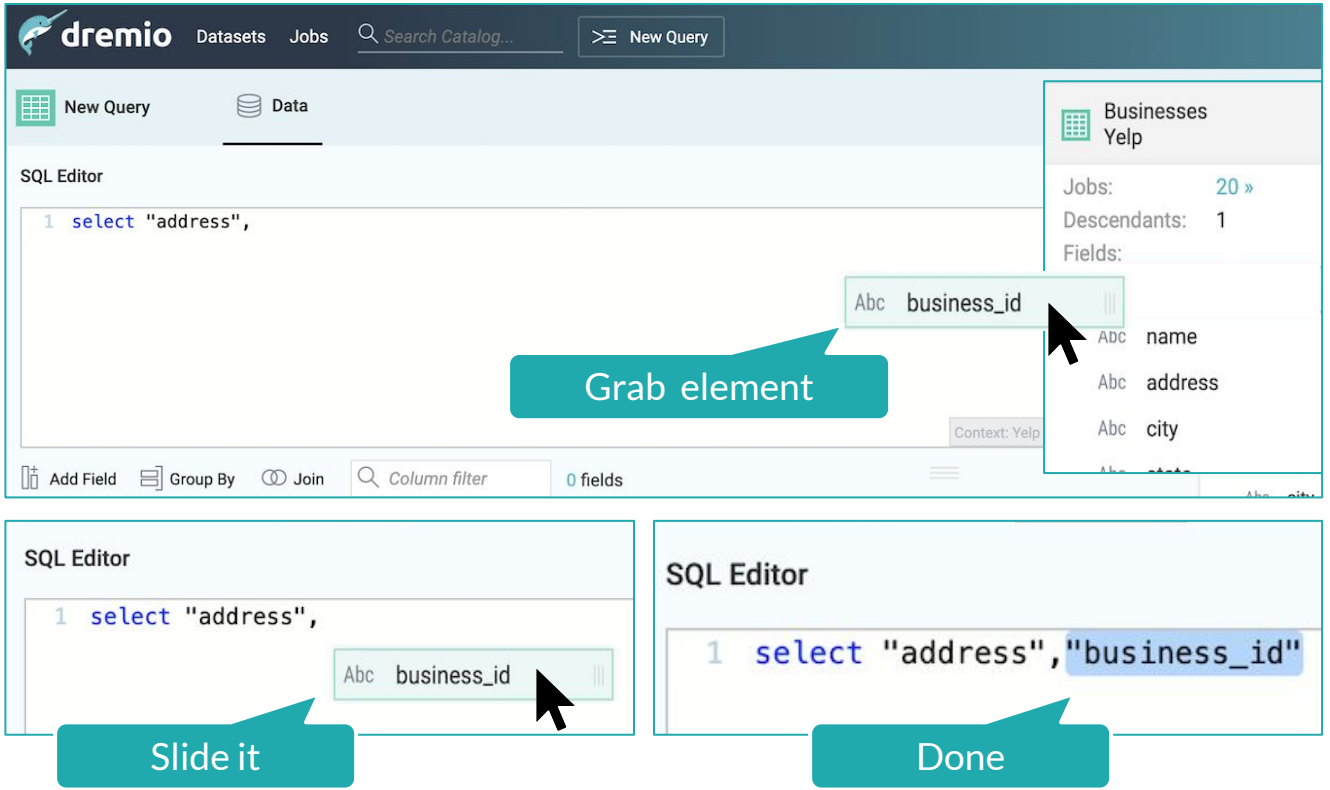
- Reference individual elements in complex type, or flatten
- Extract and unnest complex array types
- Extract from complex struct types
- Keep original column or replace





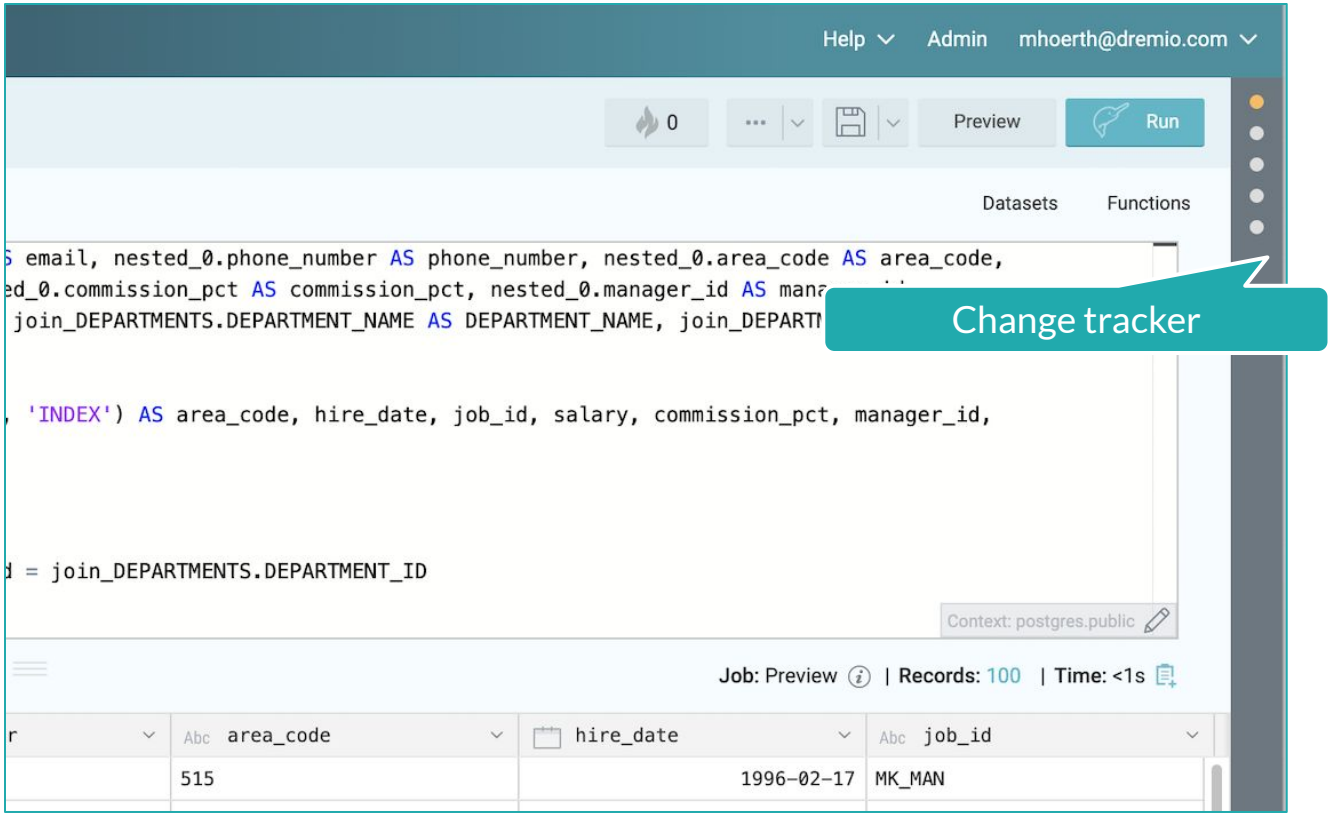
Drag and drop

- Easily create selects, joins and “group by” queries with drag and drop
- Recommended joins based on query history and schema
- Group by dimensions and measures identified for easy query building



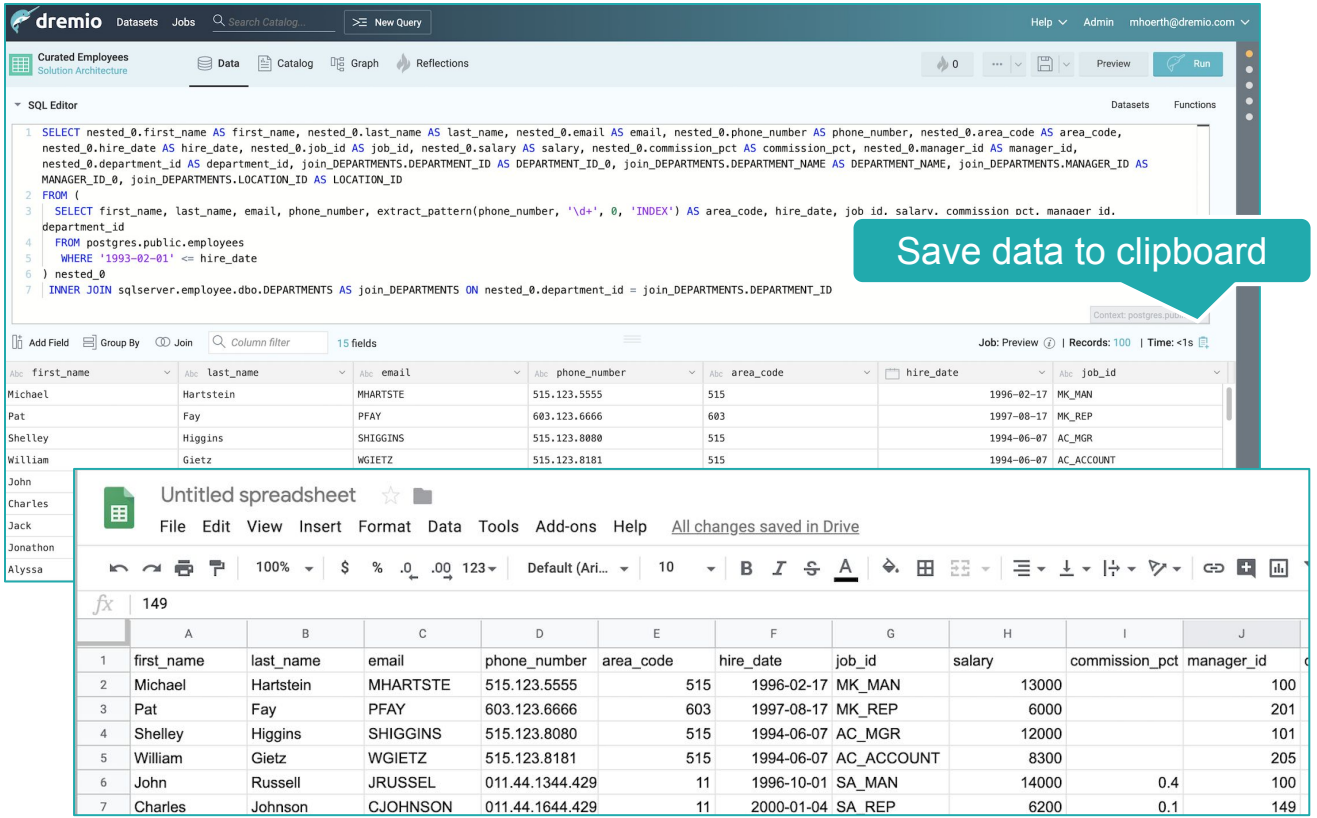
VDS version tracking

- Every change to a virtual dataset is tracked
- You can see what the change was and who made it
- Go back to a particular version



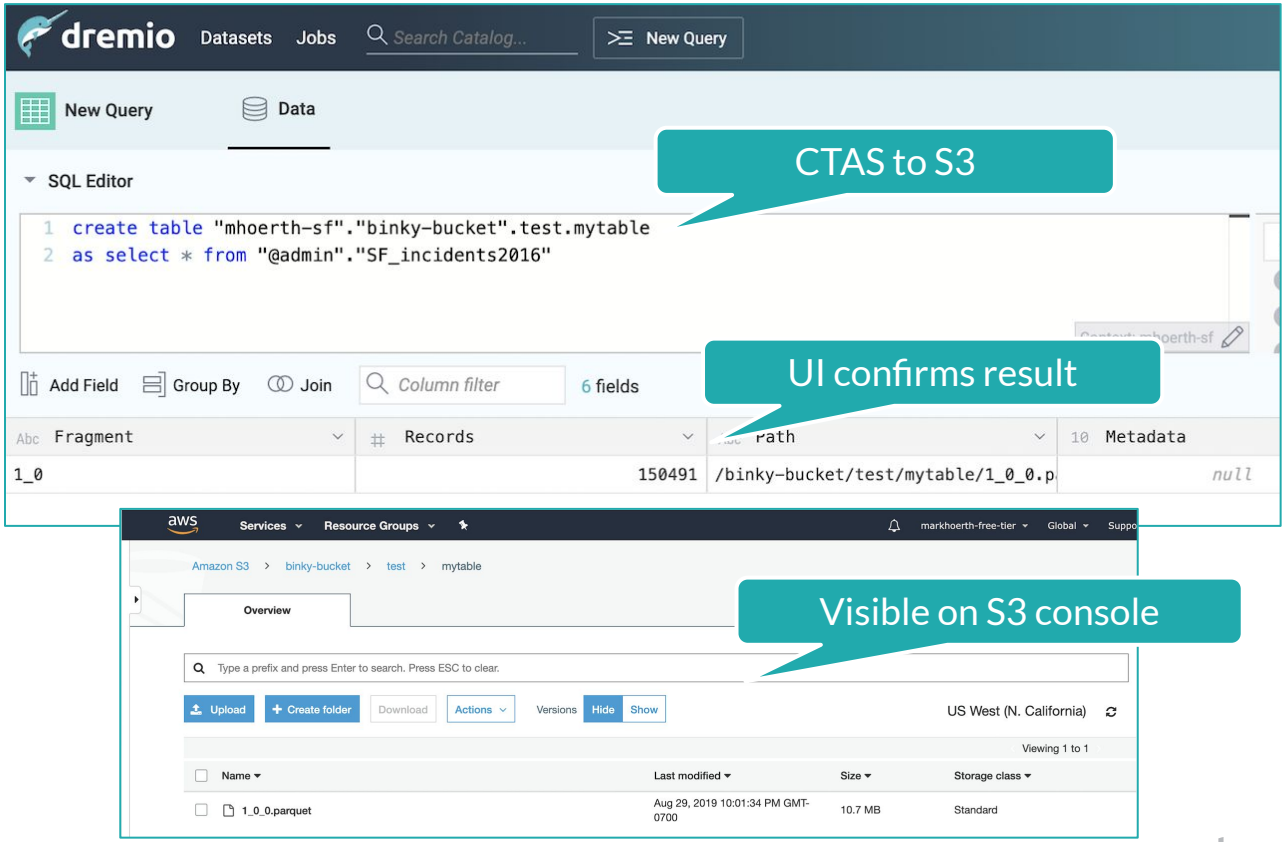
Export data using the clipboard

- Preview the dataset or run the query
- Easily export to other tools such as spreadsheets
- Start BI tools such as Tableau from the Dremio UI



Creating tables as select (CTAS)

- Tables created in file system source types or in $scratch
- Parquet format
- $scratch is readable and writable by all users.
- Must first enable exports for the source. If enabled, all users/groups who have access to the source can leverage CTAS and DROP TABLE on sources.



Dremio ANSI SQL

Including

0101
1101
001

Grammar	Types	Functions

Example differences in grammar

	Hive	Dremio
String constants quoted with non ANSI SQL double quotes	• `SELECT CONCAT (col1, "xyz")` • `SELECT REGEXP_REPLACE (col1, "x", "y")` • `CASE WHEN acquiring_portfolio = "chase" ...`	• `SELECT CONCAT (col1, 'xyz')` • `SELECT REGEXP_REPLACE(col1, 'x', 'y')` • `CASE WHEN acquiring_portfolio = 'chase' ...`
Otherwise invalid column names are escaped (`) in Hive but double-quoted in Dremio (")	• `SELECT `DATE`, `HOUR`, cast(`VALUE` as INT)` • `SELECT sum(sales) as `2018_sales``	• `SELECT "DATE", "HOUR", cast("VALUE" as INT)` • `SELECT sum(sales) as "2018_sales"`

Implicit casting example of type difference

- Hive supports implicit casting in SELECT/ WHERE, JOIN, UNION etc clauses.
- Dremio requires explicit casting if the columns being compared, joined, unioned are of different type family (string<->numeric, string<->date...).
- Hive supports untyped NULLs while Dremio requires explicit casting

Hive	Dremio

`-- Mixed types`
`SELECT Coalesce (string_col, 1)`
	`SELECT coalesce`
	`(cast(string_col as INT) ,1)`
	`--or --`
	`SELECT Coalesce(string_col,'1')`

`-- Mixed types` `SELECT sum(int_col * string_col)`	`SELECT sum(int_col * cast (string_col as int))`

`-- JOIN keys mixed types`
`FROM table1 A`
`INNER JOIN table2 B`
`ON a.str_col=b.int_col`
	`FROM table1 A`
	`INNER JOIN table2 B`
	`ON a.str_col=`
	`cast(b.int_col as varchar)`

`-- Implicit NULL cast` `SELECT col1, NULL as col2, col3.....`	`SELECT col1, CAST(NULL as varchar) as col2, col3.....`

Example function differences

- Some function unique to other platforms may not exist on Dremio or differ in their input arguments

Hive	Dremio
`TO_DATE(string_col)`	`TO_DATE(string_col, 'yyyy-mm-dd')` `CAST(string_col as DATE)`
`SPLIT (col1, '_')[0]`	`SPLIT_PART (col1, '_' , 1)`
`Date_format (string, format)`	`TO_CHAR(CAST(string as DATE), format) -- no Date_format`
`CAST (col1 as String)`	`CAST (col1 as Varchar)`
`DAY (col1) function`	`EXTRACT(DAY from col1) -- no Day()`
`PMOD (col1)`	`ABS(MOD(col1)) -- no PMOD ()`
`SELECT current_time stamp()`	`SELECT current_timestamp -- w/no ()`
`MAP, LATERAL VIEW EXPLODE functions`	`Not supported`
`LEFT SEMI JOIN`	`Not supported`

Jobs in Dremio

Including

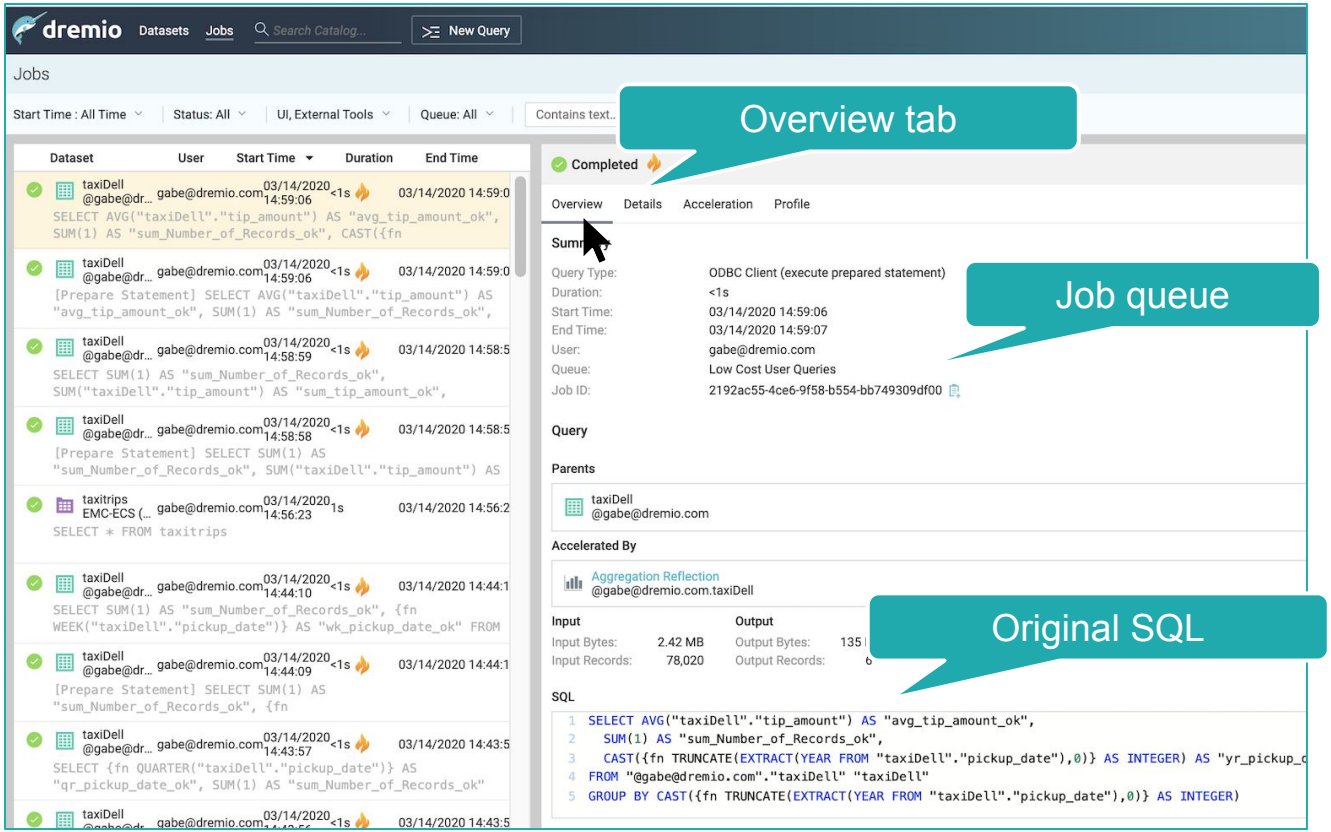
Job status

Accelerated queries

Profiles

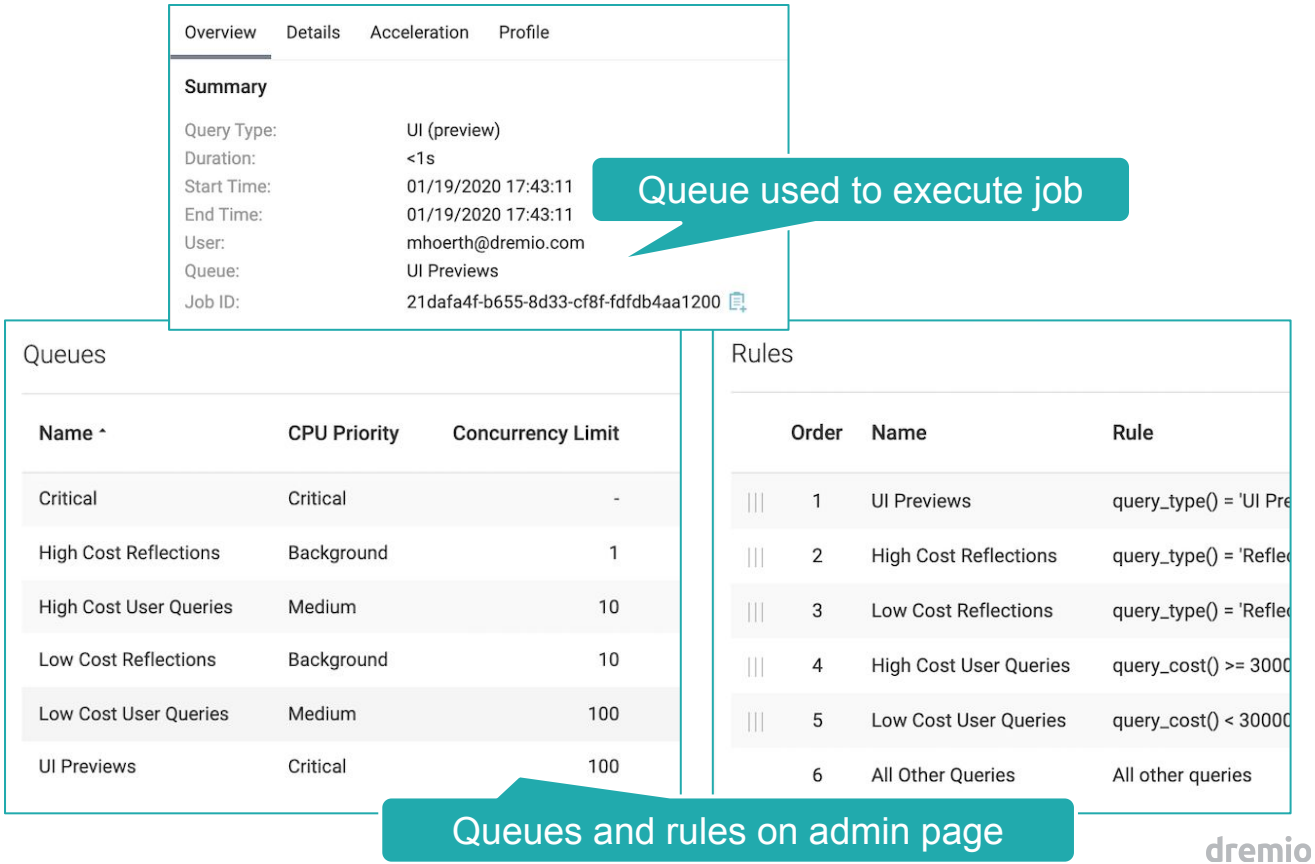
Jobs page

- All jobs are tracked
- Jobs can be filtered by
  - Status
  - Time of execution
  - Type
  - User
- Each user can see their own jobs, admin sees all jobs



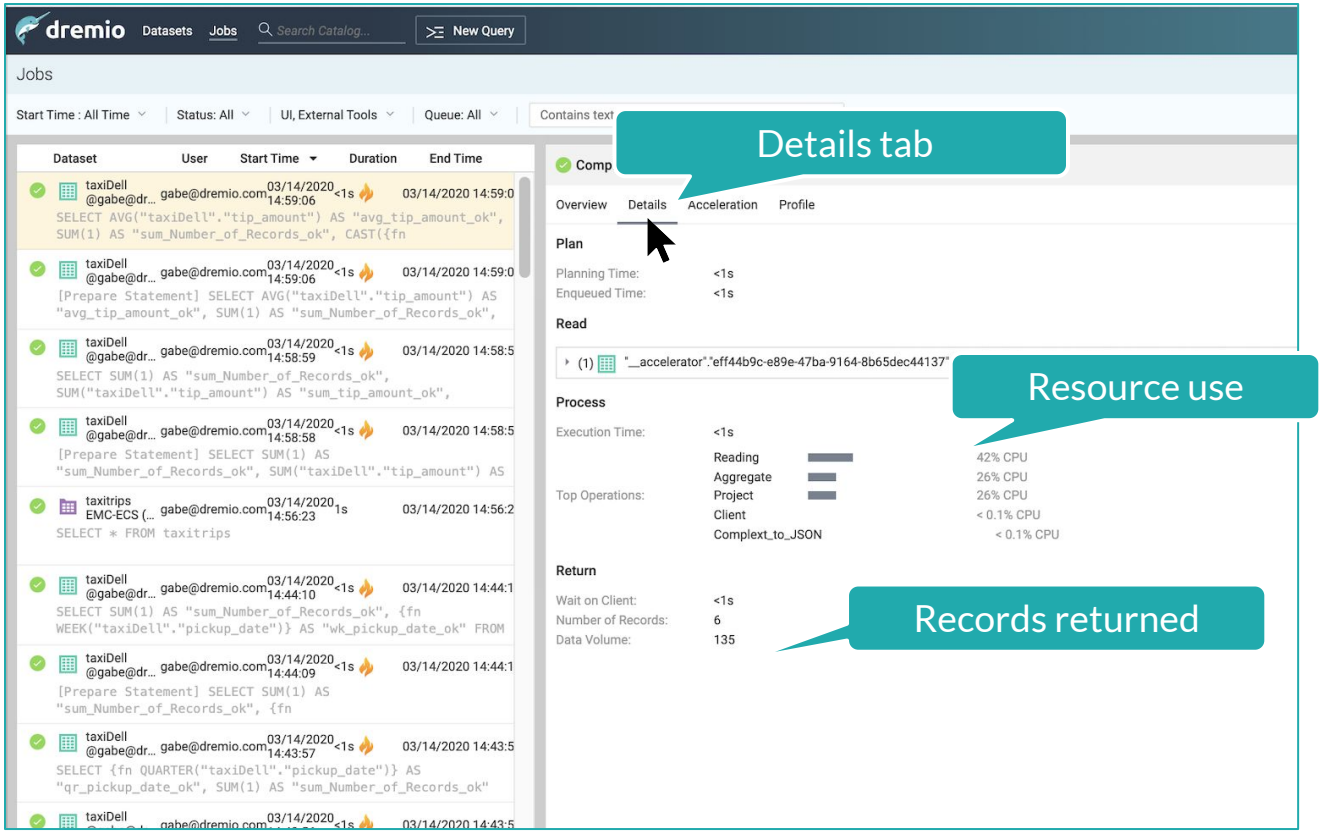
Introduction to job queues

- Runs each job in a queue with specific characteristics (such as memory limits, CPU priority, and queueing and runtime timeouts)
- Administrator can define rules that specify which query is assigned to which queue.
- Provides workload isolation and predictability for users and groups.



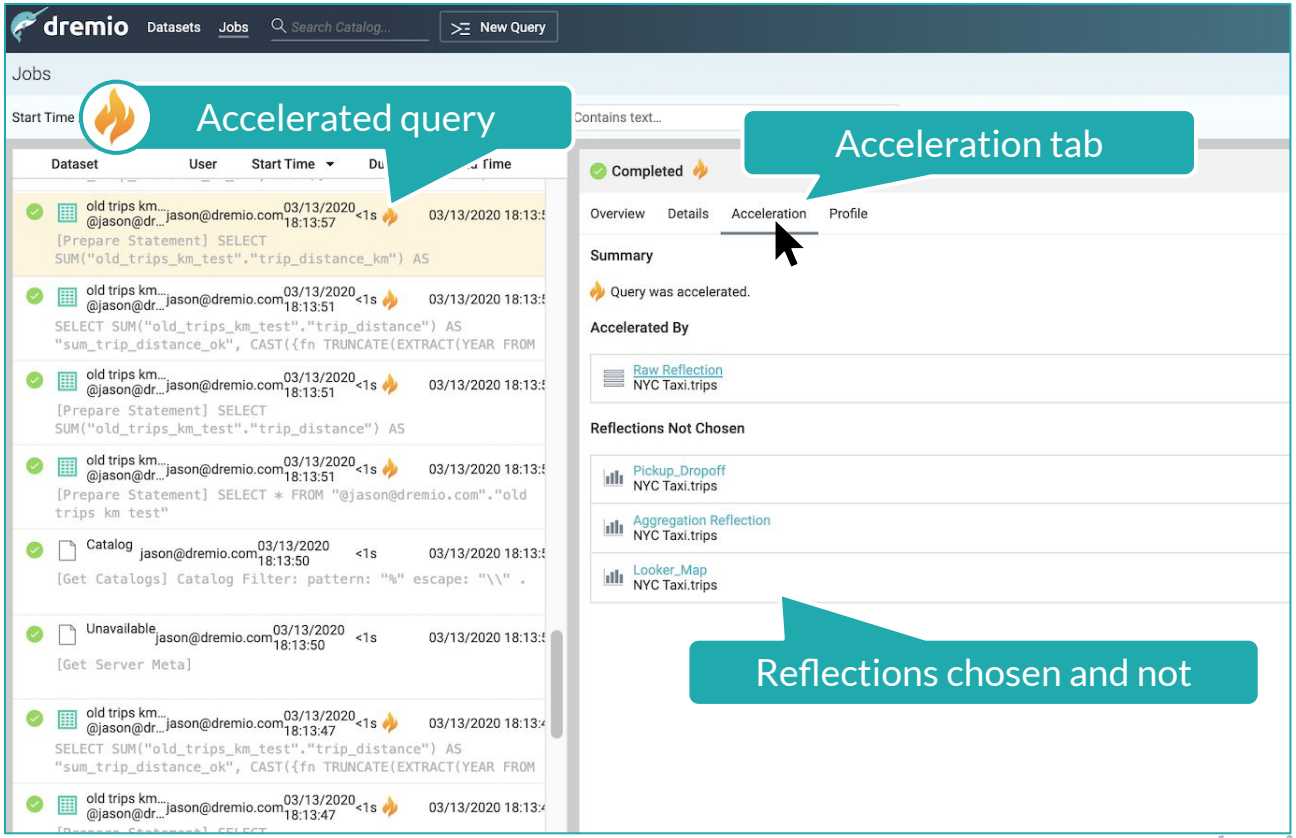
Job details

- Provides the query planning and execution time, and the amount of CPU used during data reading and writing phases.
- Lists the number of records returned and the data volume.
- Includes information on acceleration



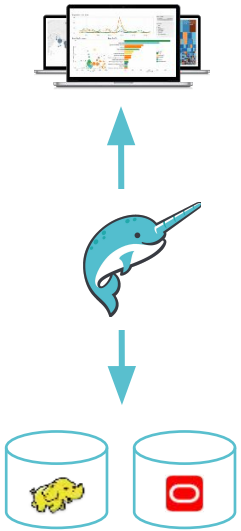
Job acceleration

- The jobs page graphically shows queries which have been accelerated by a reflection.
- For each job, the page shows the reflections chosen, not chosen.
- Query profile provides additional information about the query. The profile can be viewed in the UI or downloaded and sent to Dremio.



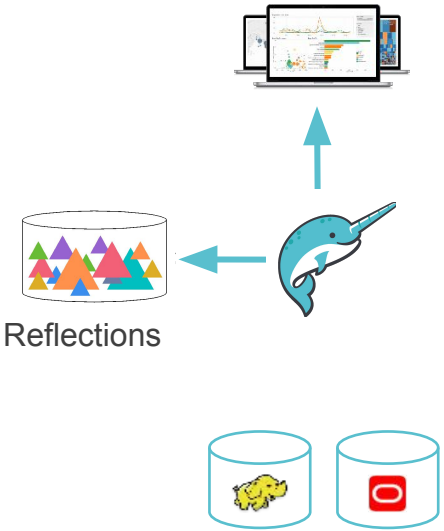
How reflections work

No reflections



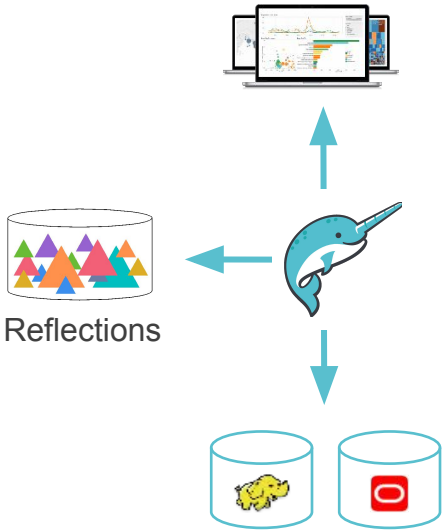
Query push-down

Acceleration



Query satisfied by reflection

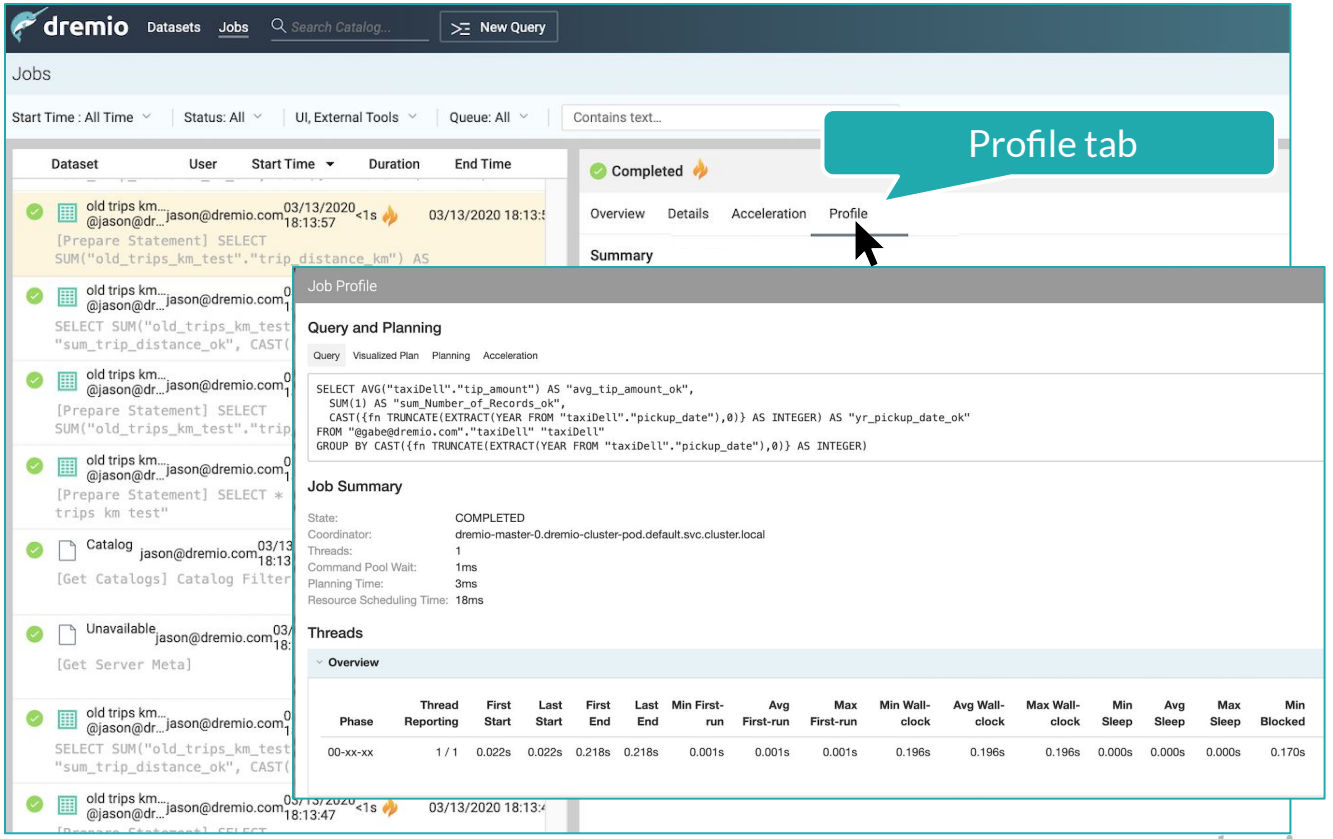
Partial acceleration



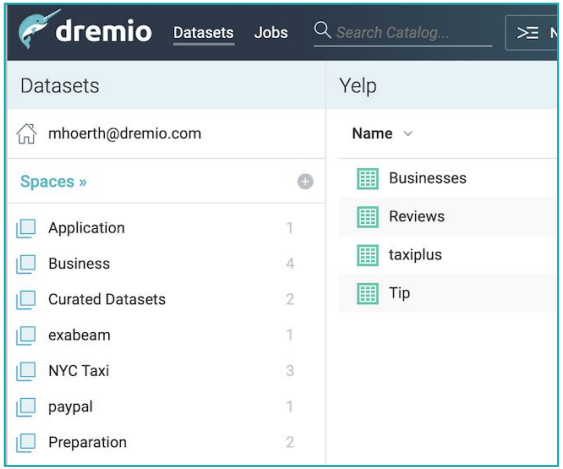
Part of the query accelerated, remainder pushed down

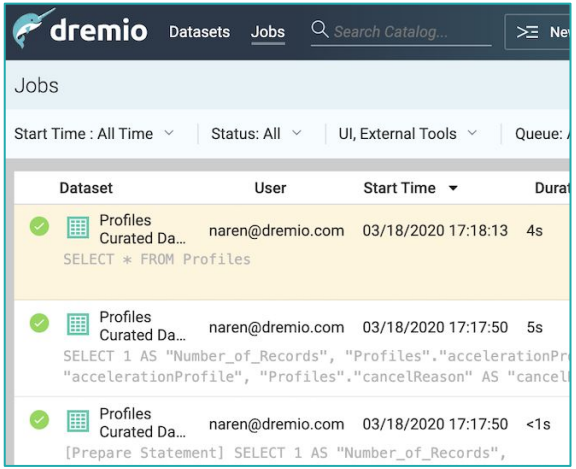
Job profile

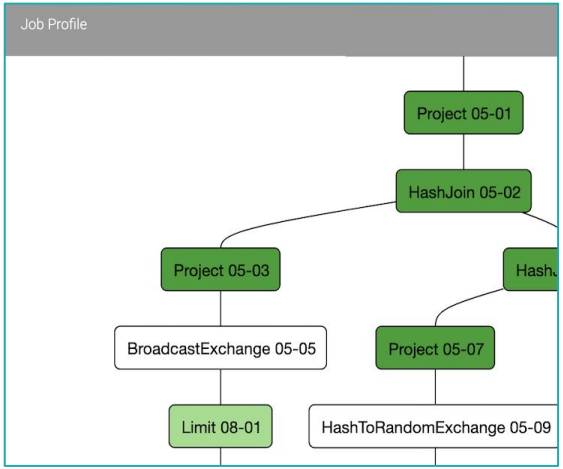
- Shows when a thread is running, blocked and waiting for data, or sleeping. A thread is usually sleeping if the thread is ready to run but another thread is currently using the CPU.
- Shows details of query planning (logical and physical)
- Shows how reflections were evaluated and resulting matching
- Topic of separate module



Demo of Dremio







ODBC, JDBC, and REST clients

Including

ODBC

JDBC

REST API

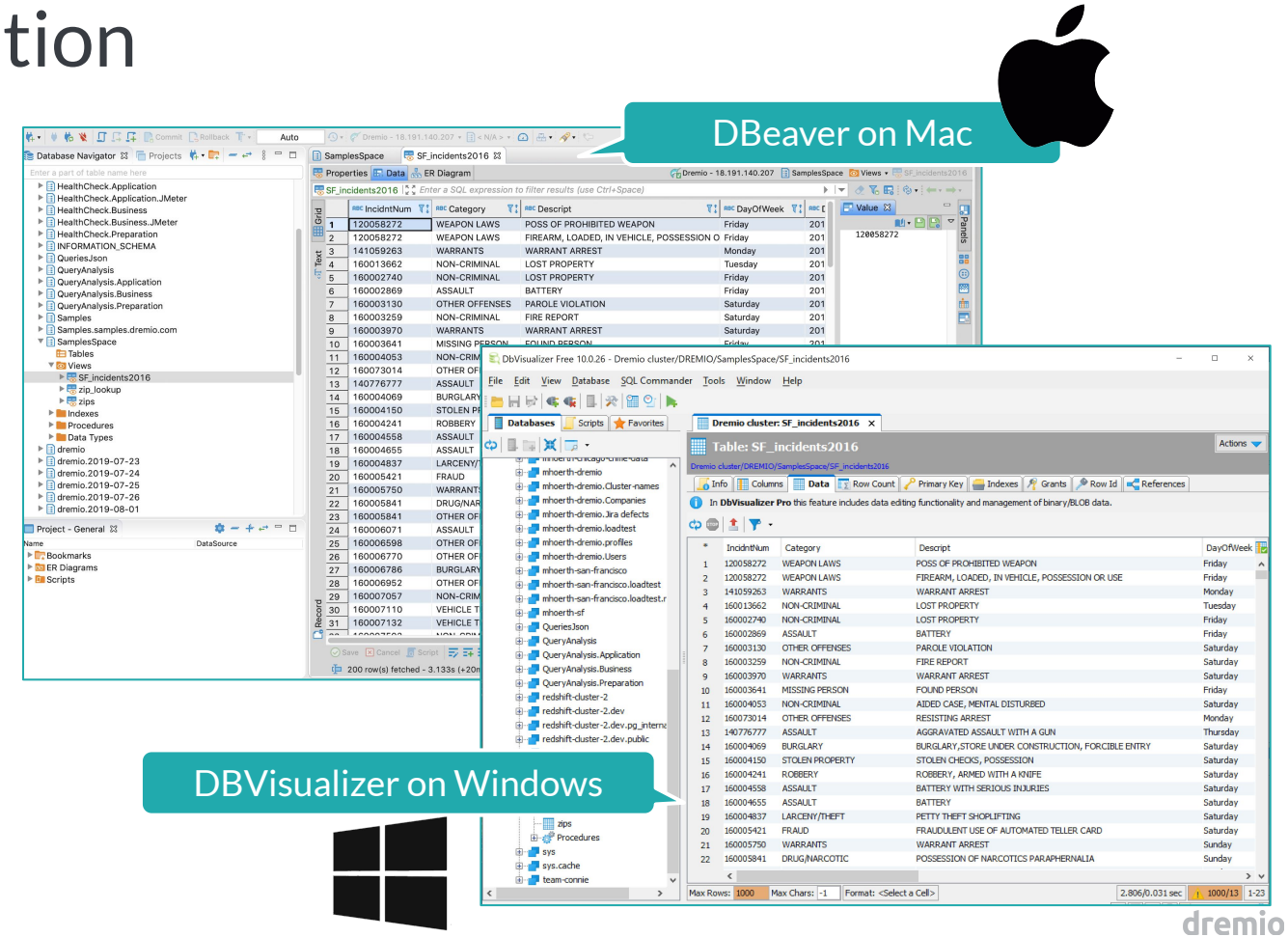
Client integration

BI tools

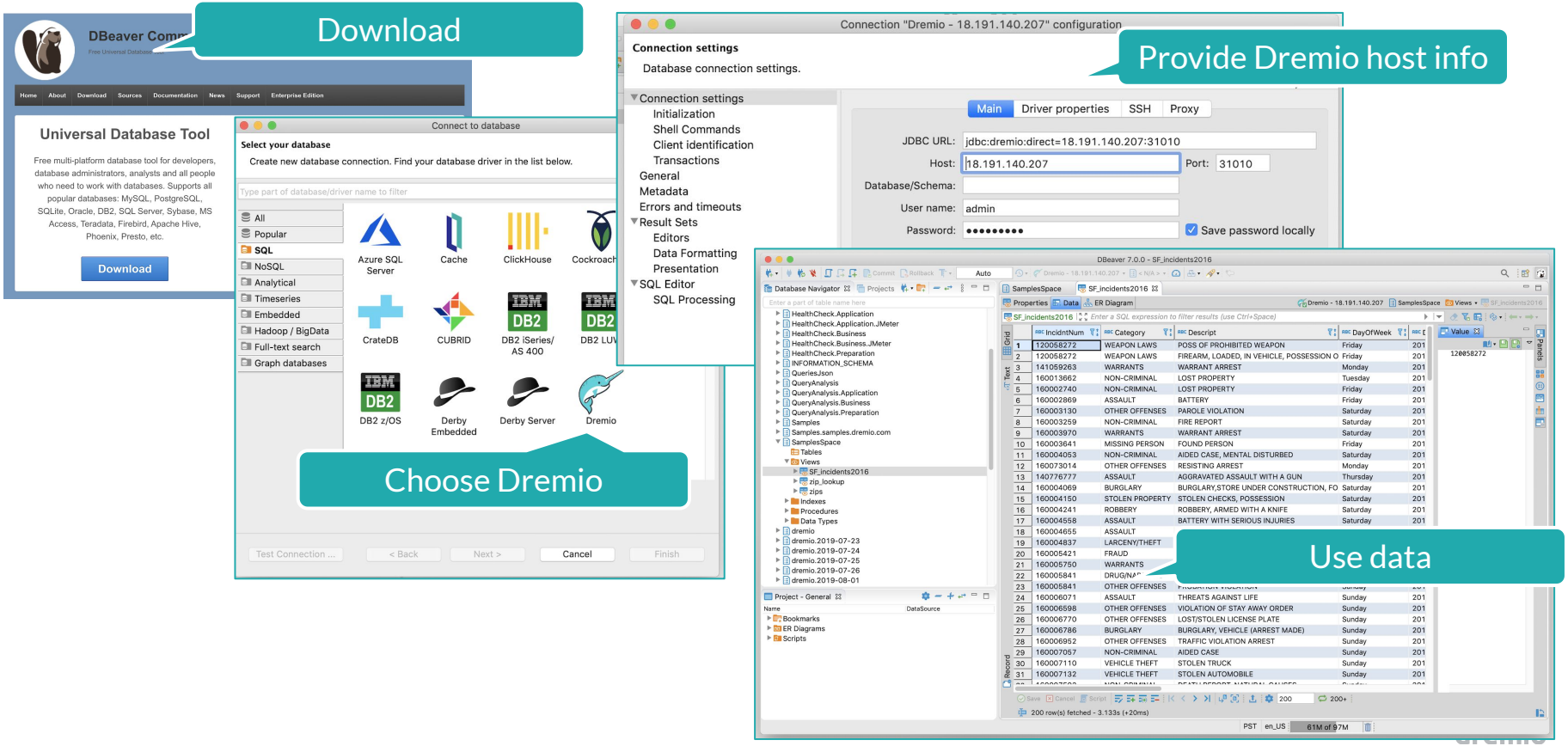
- Tableau
- Looker
- PowerBI
- Spotfire

SQL clients

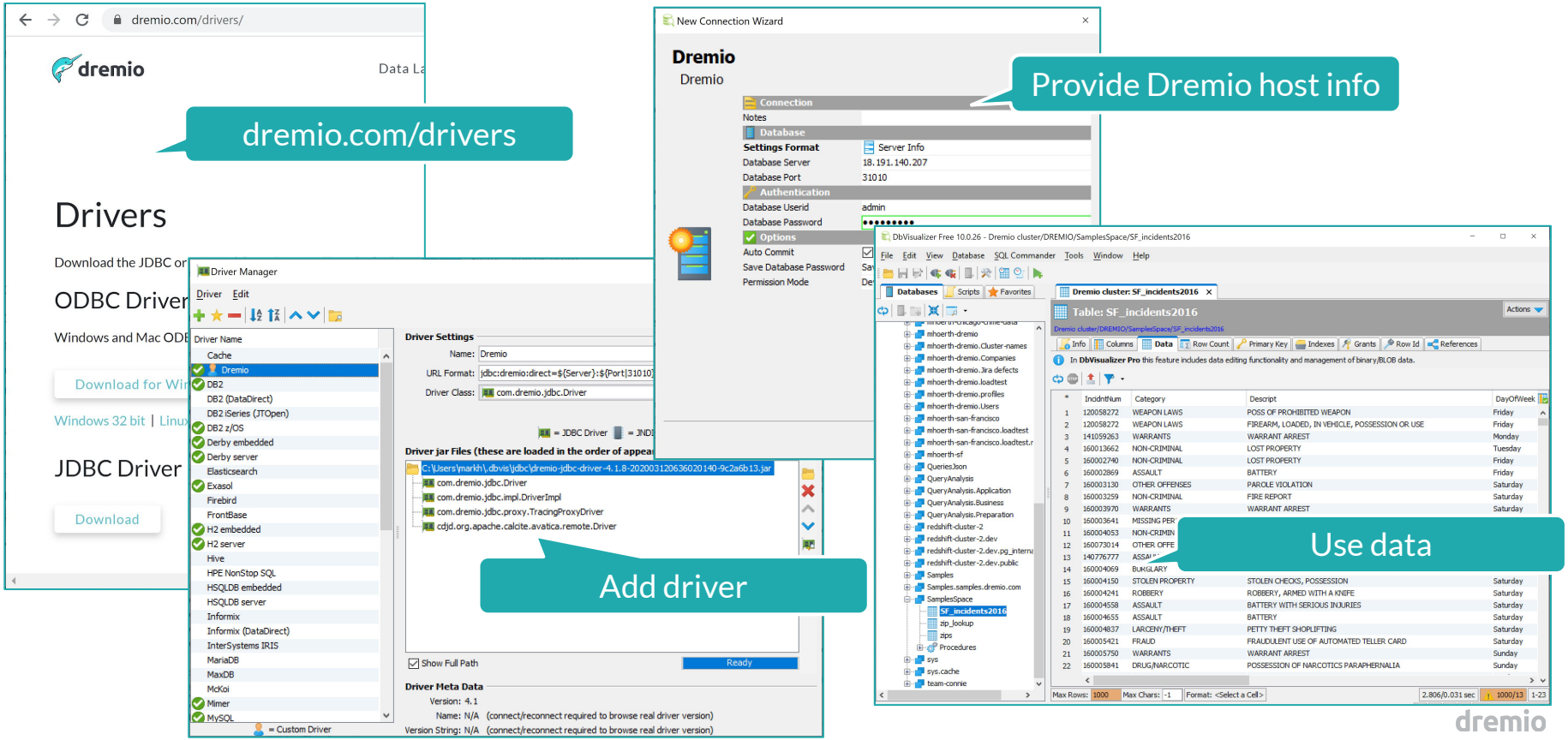
- DBeaver
- DBVisualizer
- Jupyter
- R Studio
- Alteryx Designer



Some tools integrate the driver

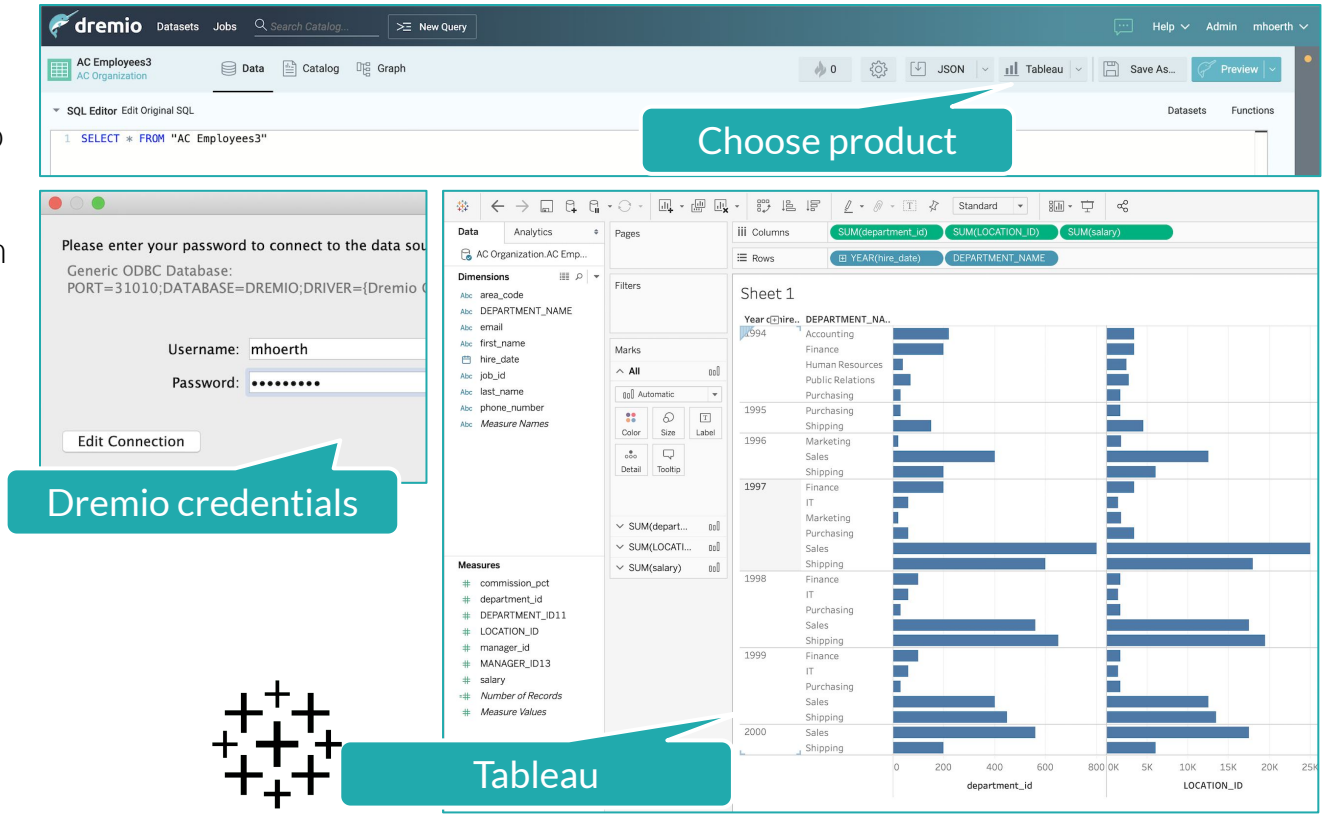


For others, download driver from dremio.com



Launch BI tools directly or from the UI

- Open Physical & Virtual Datasets in BI tools straight from the Dremio UI
- Select BI tool installed on client machine
- Clicking will download connection file



Dremio REST API

Many potential uses

- Integrate ETL processes
- Interact with sources, spaces and datasets.
- Trigger reflection refresh
- Submit queries
- Manage cluster resources and queues

POST

```
{ "userName": "mhoerth",
  "password":"binky" }
```

```
{"token":
"tokenstring",
...}
```

GET/POST/PUT/DELETE

```
Authorization: _dremio tokenstring
...
```

REST API overview

All API URLs have the following base URL:

`{DREMIO_ORIGIN}/api/v3`

Versions prior to `v3` are considered internal and subject to change without version bumps.

Curly braces (`{ }`) are used to indicate sections of URLs where caller provides a value

Legend of REST API calls:

GET	*Retrieve information about the resource*
POST	*Add a resource*
PUT	*Update a resource*
DELETE	*Delete a resource*

	Account	
GET	**/user/{id}**	*Get a user by id*
GET	**/group/{id}**	*Get a group by id*
	User	
GET	**/user/{id}/token**	*Get all user access tokens*
POST	**/usr/{id}/token**	*Create access token*
DELETE	**/user/{id}/token**	*Delete all access token*
DELETE	**/user/{id}/token/{tid}**	*Delete this access token*
	Tokens	
DELETE	**/token**	*Delete all tokens*
DELETE	**/token/{tid}**	*Delete this token*

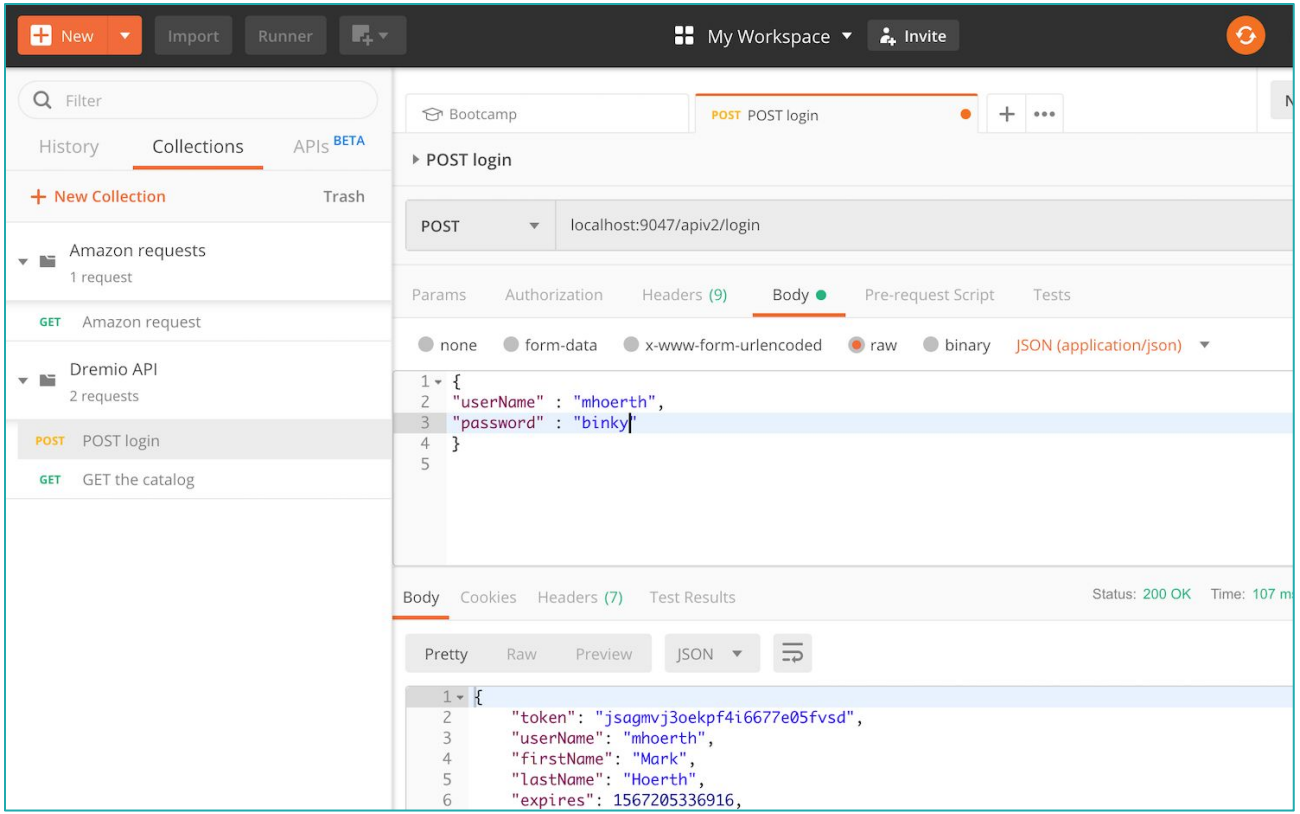
REST API overview

	Catalog	
GET	/catalog	*List all top-level containers*
GET	/catalog/{id}	*A specific source, space, folder, file or dataset*
GET	/catalog/by-path/{path}	*Get using the path*
GET	/catalog/{id}/collaboration	*Get tags and wiki content*
POST	/catalog	*Create a new catalog entry*
POST	/catalog/{id}	*Promote a file to a PDS*
POST	/catalog/{id}/refresh	*Refresh all reflections on PDS*
POST	/catalog/{id}/collaboration	*Create tags and wiki content*
PUT	/catalog	*Update existing dataset metadata*
DELETE	/catalog	*Delete catalog entry*
	Jobs	
GET	/job	*Get job information*
POST	/job	*Post job information*
	SQL	
POST	/sql	*Submit an SQL query*

	Reflections	
GET	/reflection	*Retrieve all reflections*
GET	/reflection/{id}	*Retrieve specific reflection*
GET	/reflection/summary	*Retrieve summary of all reflections*
POST	/reflection	*Create reflection*
PUT	/reflection	*Update specific reflection*
DELETE	/reflection	*Delete specific reflection*
	Workloads	
GET	/wlm/queue	*Get a list of queues*
POST	/wlm/queue	*Create a new queue*
PUT	/wlm/queue/{id}	*Update queue attributes*
DELETE	/wlm/queue/{id}	*Delete a queue*
GET	/wlm/rule	*Get a list of rules*
PUT	/wlm/rule	*Create, update, delete rules*
	Vote	
GET	/vote	*List all votes summary*

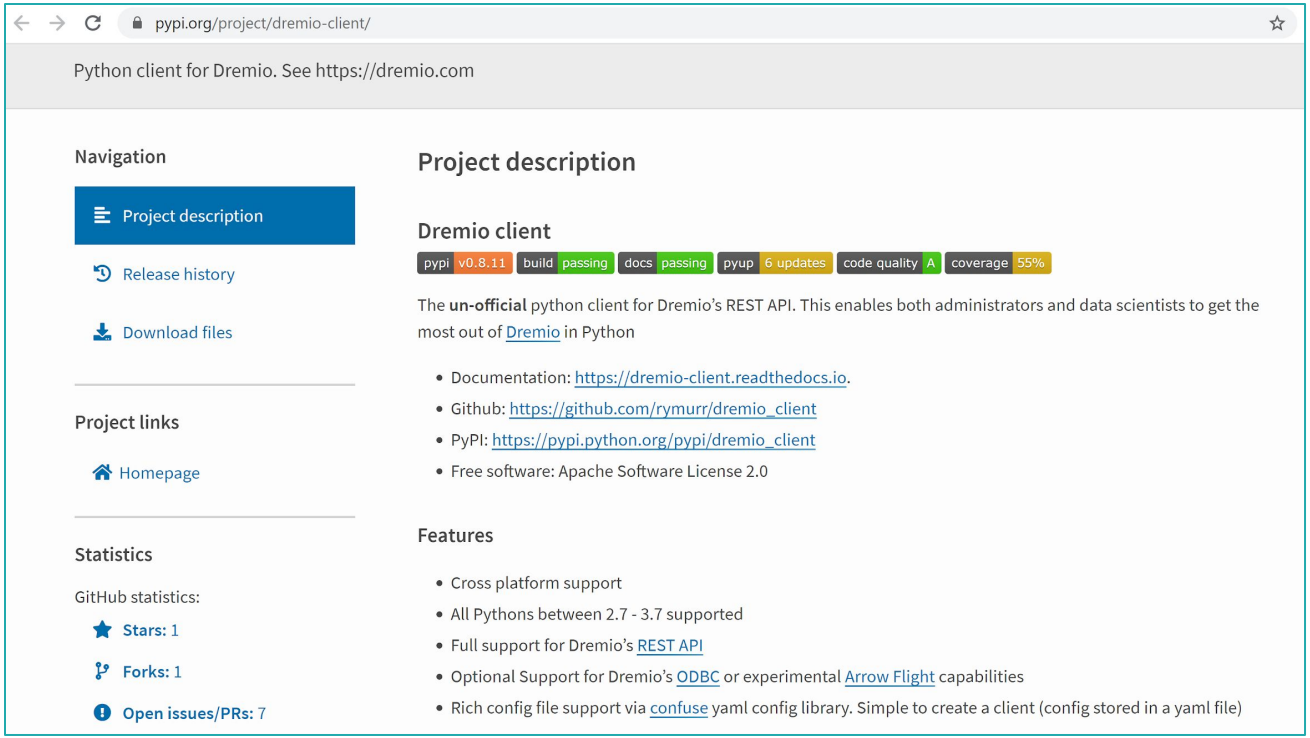
Using the Dremio Rest API

- Install curl or an API development environment such as Postman, pictured here.
- Issue a login request to Dremio and receive an authorization token
- Using the token, get a list of the top level catalog entries from Dremio
- What other information can you get from the system?



Dremio Python client

- All Pythons between 2.7 - 3.7 supported
- Full support for Dremio's REST API
- Optional Support for Dremio's ODBC or experimental Arrow Flight capabilities
- Rich config file support via confuse yaml config library. Simple to create a client (config stored in a yaml file)
- CLI interface for integration with scripts



https://pypi.org/project/dremio-client/

Dremio Java client: Creating a VDS

In class VDSCreator { main() method:

```
for (int i = 0; i < sqlFiles.length; i++) {
   File sqlFile = sqlFiles[i].getCanonicalFile();
   StringBuffer sqlText = new StringBuffer();

   if (sqlFile.getName().endsWith(".sql")) {
      System.out.println("Processing SQL file: " +
                         sqlFile.getName());

      // Obtain the SQL from the file
      List<String> lines = Files.readAllLines(sqlFile.
          getCanonicalFile().toPath(),
          Charset.defaultCharset());

      for (String line:lines) {
         sqlText.append(line);
      }

      System.out.println("VDS Definition: " + sqlText);

      payload = scrubText(sqlText.toString());
      response = dremio.postSQL(payload, null);
      System.out.println("Response: " + response);
   }
}
```

In class DremioAPI :

```
public String postSQL(String sql, String context) {
   /* Submits a SQL query.
    * https://docs.dremio.com/rest-api/
    sources/post-source.html
    */

   String url = this.host + "/api/v3/sql";
   String body = null;
   if (context != null && context.length() > 0) {
      body = "{\"sql\": \"" + sql + "\", \"context\":
          \"" + context + "\"}";
   } else {
       body = "{\"sql\": \"" + sql + "\"}";
   }

   return doPost(url, body, true);
}
```

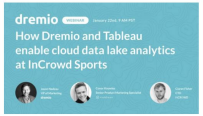
Java doPost() part of Java client

```
private String doPost(String dremioURL, String bodyPayload, boolean withToken) {
   StringBuffer jsonString = new StringBuffer();
   try {

      // connection setup code

      if (withToken) {
         connection.setRequestProperty("Authorization", this.token);
      }
      OutputStreamWriter writer = new OutputStreamWriter(connection.getOutputStream());
      if (bodyPayload != null) {
         writer.write(bodyPayload);
      }
      writer.close();
      BufferedReader br = new BufferedReader(new InputStreamReader(connection.getInputStream()));

      String line;
      while ((line = br.readLine()) != null) {
         jsonString.append(line);
      }
      br.close();
      connection.disconnect()
   } catch (Exception e) {
      throw new RuntimeException(e.getMessage());
   }

   return jsonString.toString();
}
```

Additional resources

Including

Dremio University



How Dremio and Tablea
cloud data lake analytics
InCrowd Sports

Mar 3, 2020

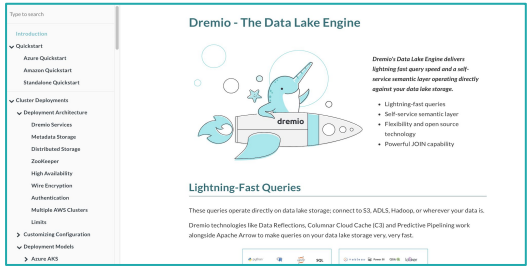
In this webinar we explore how to accele
performance for BI and Data Science wo

Blog

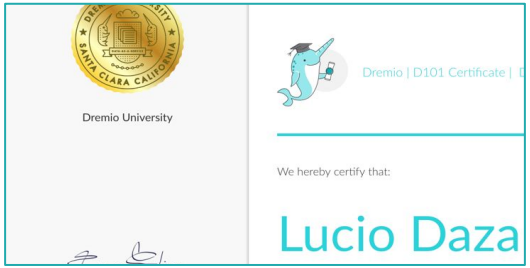
Documentation

Dremio digital learning resources

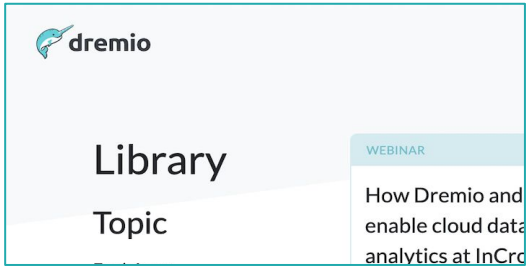
Documentation
https://docs.dremio.com



Dremio University
https://university.dremio.com

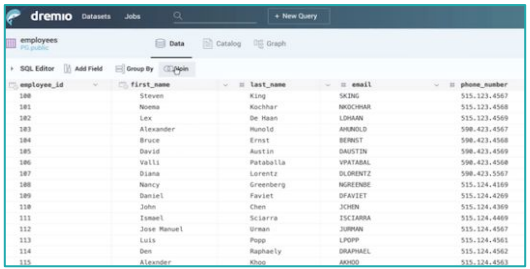


Library
https://www.dremio.com/library/

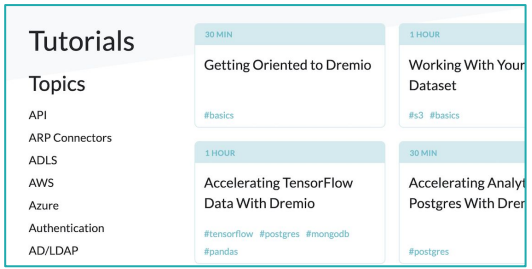


Blog
https://www.dremio.com/blog/

Demos
https://www.dremio.com/demos/



Tutorials
https://www.dremio.com/tutorials/



Dremio University

- Guided, self-paced courses with hands-on labs in cloud-hosted Dremio instance
- Free to register - visit https://university.dremio.com
- Courses available: “Dremio Fundamentals”, “Data Reflections”, “Dremio for Data Consumers”
- Post questions in the Dremio University topic on community.dremio.com

Key digital training resources

			Dremio University	White Papers	Tutorials	Webinars
Dremio Administrators	Data Engineers	Data Consumers	• Dremio Fundamentals • Dremio for Data Consumers	• What Is a Data Lake Engine? • What Is Data-as-a-Service?	• Getting Oriented to Dremio • Working With Your First Dataset • Combining Data From Multiple Datasets	
			• Data Reflections	• Dremio Security Architecture Guide • Dremio Semantic Layer Best Practices • BI on Big Data	• How To Share A Query Profile • Data Curation With Dremio • Using Dremio to Fix Data Inconsistency	• Data Science Across Data Sources With Apache Arrow • Data Reflections: Accelerate your Queries Without Copies
				• Dremio Data Reflections Overview & Best Practices • Dremio Deployment Considerations Guide • Dremio Architecture Guide	• Adding Users to Dremio • Getting Started With Data Reflections	• Dremio 4.0 – Technical Deep Dive • Running SQL-Based Workloads in the Cloud Using Apache Arrow

Click on the titles above to retrieve the resource.