

Lake tables: Iceberg, Delta and Parquet

From the SourceLace manual: How to let people query Apache Iceberg, Delta Lake and Parquet tables in your Amazon S3 storage with read-only SQL, when you have no query engine (such as Athena, Trino, Snowflake or Databricks) for SourceLace to ask. SourceLace reads the table files itself, so an admin m

SourceLace version 0.1.0 · Made October 7, 2026

The latest version is at <https://sourcelace.onrender.com/docs>

Lake tables: Iceberg, Delta and Parquet

How to let people query Apache Iceberg, Delta Lake and Parquet tables in your Amazon S3 storage with read-only SQL, when you have no query engine (such as Athena, Trino, Snowflake or Databricks) for SourceLace to ask. SourceLace reads the table files itself, so an admin must turn this on, and only for tables that have no row or column rules.

Status: Preview. New, offered for pilots and provided as is. Built from the Iceberg and Delta specifications and AWS's documentation, and covered by automated tests with real table files in a simulated S3 bucket; not yet used with a customer's live account.

At a glance

	Connects through	Sign-in	Network	Status	First test question
Lake tables	Amazon S3 API, with AWS Glue Data Catalog or an Iceberg REST catalog	Shared account	Nothing to open	Preview	"Describe the orders table in lake:warehouse."

Add each source with the [usual steps](#). The network answers are explained in [What your network needs](#). If the first test question fails, the error message tells you what to fix: look it up under **When something goes wrong** below.

When to use it

If your tables already have a query engine, connect that instead (for example [Snowflake or Databricks](#)): the engine applies each person's own permissions, including row and column rules, and does the heavy work.

Use a lake source when there is no engine, for tables everyone allowed to use the source may see in full. It reads the files directly, which **skips** any row filters, column masks or per-person grants that AWS Lake Formation, Unity Catalog, Ranger or a query engine would apply. That is why it only works once an admin sets the `direct_reads` option to on.

Like [Amazon S3](#), a lake source has no per-person sign-in: everyone who may use it reads with your organization's own AWS access, set on the source. To narrow who in your organization may use it, use [access by group](#).

What people can do

- `search_schema` lists the tables the admin declared; `describe_object` lists a table's columns and types, from the table's own metadata.
- `query` (language `sql`) takes **one** `SELECT` or `WITH ... SELECT` statement in DuckDB's SQL dialect over the declared tables, by name, such as `SELECT region, sum(amount) FROM orders WHERE order_date >= DATE '2026-01-01' GROUP BY region`. Joins across the source's tables work, whatever their format.
- Anything else is refused **before** any file is read: changes, more than one statement, other tables, file paths or web addresses, and functions that read files or settings (`read_parquet`, `read_csv`, `glob`, `query` and the like).

- SourceLace reads only the columns a query uses and skips the parts of files, and whole files, that a filter rules out (for example a filter on a partition column), so name the columns you need and filter on partition columns.
- Results are capped like other queries (2,000 rows kept, 50 shown at a time), and held for at most 30 minutes and never written to a database (see [How long things are kept](#)).
- Lake sources never accept changes.

Supported tables:

Format	How to name it	Notes
Iceberg, by its folder or metadata file	orders = iceberg sales/orders/	The newest metadata file is found from version-hint.text, or the highest vN.metadata.json. Renamed and added columns are followed.
Iceberg, in AWS Glue Data Catalog	orders = iceberg glue:sales_db.orders	SourceLace asks Glue (GetTable) for the table's current metadata file.
Iceberg, in an Iceberg REST catalog	orders = iceberg rest:sales.orders	Needs the catalog_uri option, and usually catalog_token.
Delta Lake	customers = delta crm/customers/	Read from the latest checkpoint and the commits after it. Column mapping is supported.
Parquet	clicks = parquet raw/clicks/	Every .parquet file in the folder (or one file). key=value folders become extra text columns. The files must share one schema.

Not supported yet: Iceberg tables with row-level deletes (delete files or deletion vectors), Delta tables with deletion vectors or V2 checkpoints, ORC and Avro data files, time travel, and storage other than Amazon S3. Such a table is refused with a message saying so.

1. Give SourceLace read access in AWS (AWS admin)

SourceLace needs to list and read the objects under the tables' location, and, for tables in Glue, to read their definitions. Create an IAM policy like this one, with your bucket and folder:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::corvanta-lake",
      "Condition": { "StringLike": { "s3:prefix": ["warehouse/*"] } }
    },
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::corvanta-lake/warehouse/*"
    },
    {
      "Effect": "Allow",
      "Action": "glue:GetTable",
      "Resource": [
        "arn:aws:glue:us-east-1:123456789012:catalog",
        "arn:aws:glue:us-east-1:123456789012:database/sales_db",
        "arn:aws:glue:us-east-1:123456789012:table/sales_db/*"
      ]
    }
  ]
}
```

```

    ]
  }
}

```

Leave out the Glue statement if no table is in Glue. If the bucket uses a customer-managed KMS key, also allow `kms:Decrypt` on that key.

Then give SourceLace the policy in one of two ways, exactly as for an [Amazon S3 source](#):

- **A role (recommended).** Create an IAM role with the policy, trusting SourceLace's AWS account, with a trust policy condition that requires the **external ID** SourceLace shows for your organization on the source (once saved). SourceLace assumes the role for each query and never stores the short-lived credentials.
- **Access keys.** An IAM user with the policy and an access key. The secret is stored encrypted and never shown again.

2. Add the source (SourceLace admin)

On **Manage sources**, add a source of kind **Lake tables** (lake), such as `lake:warehouse`, with:

Option	What to enter
location	Where the tables are, such as <code>s3://corvanta-lake/warehouse/</code> . SourceLace never reads outside it, even if a table's metadata points elsewhere.
region	The bucket's AWS region, such as <code>us-east-1</code> .
tables	The tables, separated by semicolons (or new lines), each <code>name = format where</code> , as in the table above. Folders are relative to the location, or a full <code>s3://</code> address inside it. Table names are lowercase letters, digits and underscores.
role_arn, or access_key_id and secret_access_key	Your organization's AWS access from step 1.
direct_reads	on: you confirm these tables have no row or column rules. Without it, the source is refused.
catalog_uri	For <code>rest: tables</code> : the catalog's https address, such as <code>https://catalog.corvanta.com/api/catalog</code> .
catalog_warehouse	For <code>rest: tables</code> , if your catalog needs one.
catalog_token	For <code>rest: tables</code> : a bearer token with read access to the tables' metadata. Stored encrypted, never shown again, and dropped if <code>catalog_uri</code> changes.
max_bytes_read, max_files_read	Optional: lower this source's caps (below).

Then connect it once like any other source (there is no sign-in page) and try `describe_object` on a table.

Limits and safety

- **Bytes and files.** One query may read at most 2 GB and 2,000 files (data and metadata together) unless your limits say otherwise; a source's `max_bytes_read` and `max_files_read` options can only lower them. SourceLace counts every read as it happens and stops the query at the cap, so the cap is never exceeded.
- **Time.** Queries have the usual query time limit and are stopped when it runs out.
- **Kept apart.** Each query runs on its own and shares nothing with other queries, people or organizations. It reads only the files of the tables you declared, inside the location, and cannot reach the internet or other buckets.
- **Working space.** A very large query may use temporary working space while it runs. That space is not encrypted and is deleted when the query ends. Nothing else is written down.

When something goes wrong

- *"... skips any row or column rules ..."*: set `direct_reads` to `on`, but only if the tables have no such rules.
- *"Amazon S3 refused ... access"*: the role or keys cannot read that table's files. Check the IAM policy (and KMS key) covers the whole location.
- *"... was not found"*: check the table's entry in `tables` (folder, Glue database and table, or catalog namespace).
- *"... names a file outside the source's location"*: the table's metadata points outside `location`. Widen `location` if that is expected.
- *"... would read more than ..."*: select fewer columns, filter on partition columns, or ask an admin to raise the limit.
- *"... cannot read directly yet"*: the table uses a feature listed above as not supported. Query it through your query engine.
- *"The Iceberg catalog refused ..."*: enter a new `catalog_token`.