Cloudera Runtime 7.2.17

Iceberg support for Atlas

Date published: 2023-06-26
Date modified:

https://docs.cloudera.com/

Legal Notice

Contents

Iceberg support for Atlas (Technical Preview)

Atlas integration with Iceberg helps you identify the Iceberg tables to scan data and provide lineage support. Learn how Atlas works with Iceberg and what schema evolution, partition specification, partition evolution are with examples.

How Atlas works with Iceberg

You can use Atlas to find, organize, and manage different aspects of data about your Iceberg tables and how they relate to each other. This enables a range of data stewardship and regulatory compliance use cases.

The Atlas connectors distinguish between Hive and Iceberg tables. The Iceberg table is available in a “typedef” format which implies that the underlying data can be retrieved by querying the Iceberg table. All attributes of the Hive table are available in the Iceberg table and this equivalence is achieved by creating the Iceberg table as a sub-type of the underlying Hive table. Optionally, the Iceberg table can also be queried by Hive or Impala engine. For more information about Iceberg and related concepts, see Apache Iceberg features and Apache Iceberg in CDP.

Both Iceberg and Hive tables have equality in Atlas in terms of data tagging. Data evolution and transformation are features unique to Iceberg tables. Iceberg adds tables to compute engines including Spark, Hive and Impala using a high-performance table format that works just like a SQL table. Also, the lineage support for Iceberg table is available. For example, when a Hive table is converted to Iceberg format, the lineage is displayed for the conversion process in Atlas UI.

- Migration of Hive tables to Iceberg is achieved with the following:
  - Using in-place migration by running a Hive query with the ALTER TABLE statement and setting the table properties
  - Executing CTAS command from Hive table to the Iceberg table.
- Schema evolution allows you to easily change a table's current schema to accommodate data that changes over time. Schema evolution enables you to update the schema that is used to write new data while maintaining backward compatibility with the schemas of your old data. Later the data can be read together assuming all of the data has one schema.
  - Iceberg tables supports the following schema evolution changes:

    Add – add a new column to the table or to a nested struct

    Drop– remove an existing column from the table or a nested struct

    Rename– rename an existing column or field in a nested struct

    Update– widen the type of a column, struct field, map key, map value, or list element

    Reorder – change the order of columns or fields in a nested struct
  - Partition specification allows you to initiate queries faster by grouping similar rows together when writing.

  As an example, queries for log entries from a logs table usually include a time range, like the following query for logs between 10 A.M. and 12 A.M.

  SELECT level, message FROM logs

  WHERE event_time BETWEEN '2018-12-01 10:00:00' AND '2018-12-0112:00:00'

Configuring the logs table to partition by the date of event_time groups log events into files with the same event date. Iceberg keeps track of that date and uses it to skip files for other dates that do not have useful data.

- Partition evolution across Iceberg table partitioning can be updated in an existing table because queries do not reference partition values directly.

When you evolve a partition specification, the old data written with an earlier specification remains unchanged. New data is written using the new specification in a new layout. The metadata for each of the partition versions is stored separately.

Due to this nature of partition evolution, when you start writing queries, you get split planning. This is where each partition layout plans files separately using the filter it derives for that specific partition layout.

Using the Spark shell

Using Spark, you can create an Iceberg table followed by schema evolution, partition specification, and partition evolution.

Before you begin

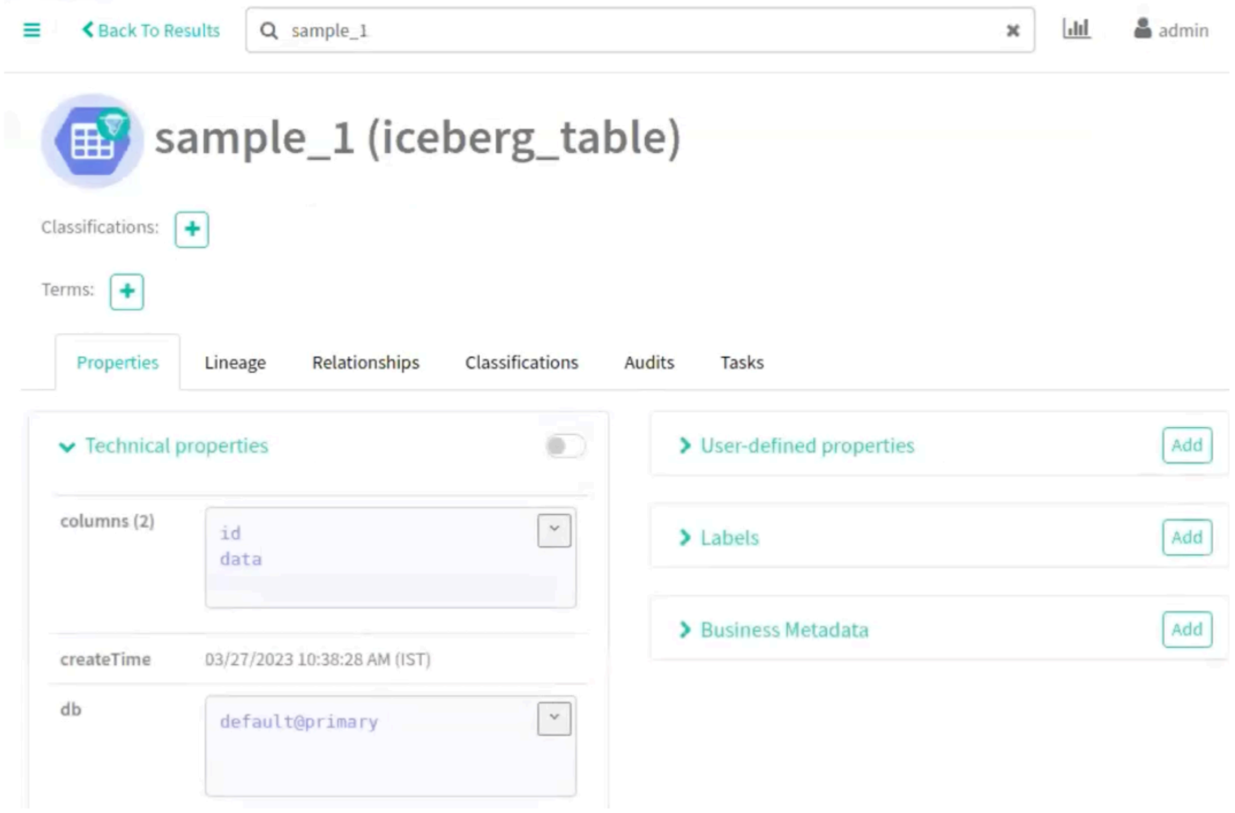
You must configure the Spark shell as such you have included the valid Spark runtime version.

Run the following command in your Spark shell to create a new Iceberg table

Procedure

1.
```
spark.sql("CREATE TABLE spark_catalog.default.sample_1 ( id bigint COMMENT
'unique id', data string) USING iceberg");
```

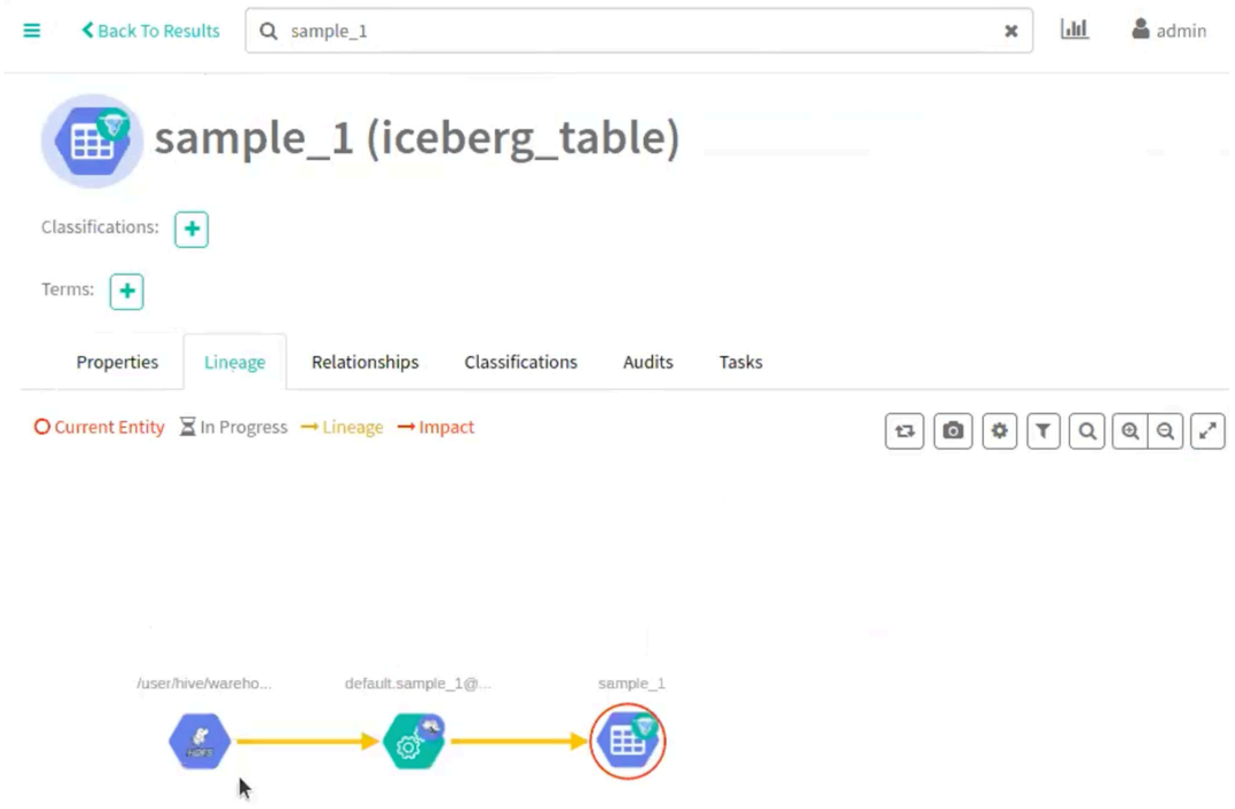
2. Navigate accordingly in the Atlas UI to view the changes.
   The following images provide information about Iceberg table creation process.

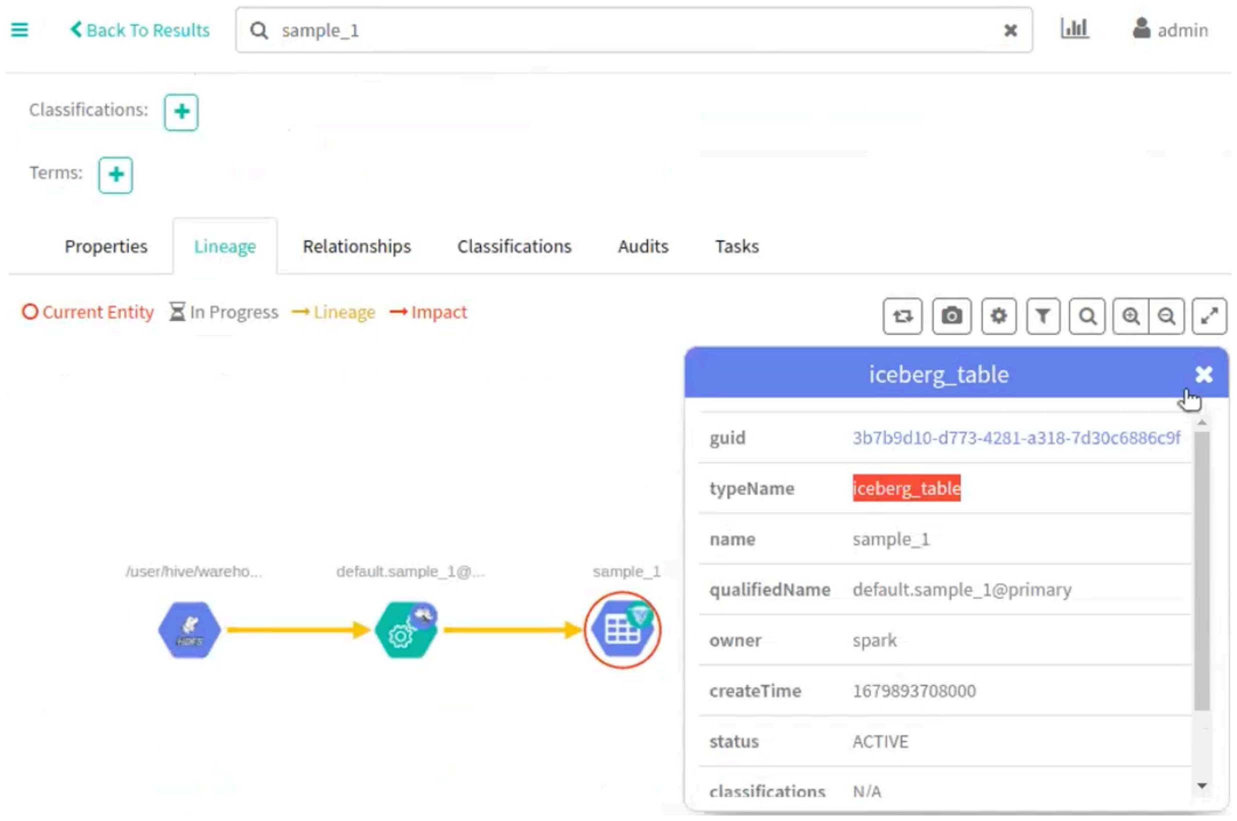


columns (2): id, data

createTime: 03/27/2023 10:38:28 AM (IST)

db: default@primary





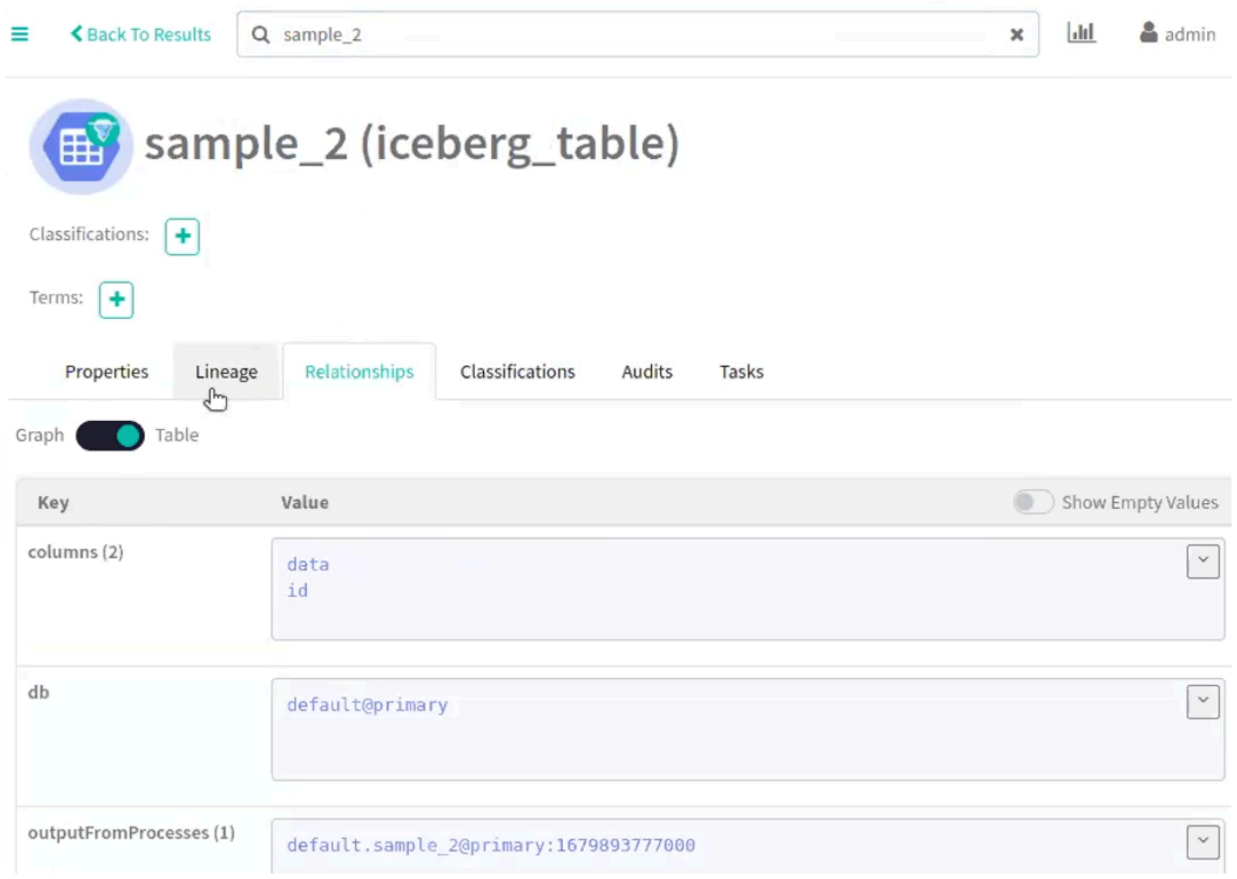
guid	3b7b9d10-d773-4281-a318-7d30c6886c9f
typeName	iceberg_table
name	sample_1
qualifiedName	default.sample_1@primary
owner	spark
createTime	1679893708000
status	ACTIVE
classifications	N/A

Run the following command in your Spark shell to create a Schema Evolution in a new table. For example - sample_2.

3. spark.sql("CREATE TABLE spark_catalog.default.sample_2 (id bigint COMMENT 'unique id', data string) USING iceberg");

4. Navigate accordingly in the Atlas UI to view the changes.

   The following image provide information about Iceberg schema evolution process.



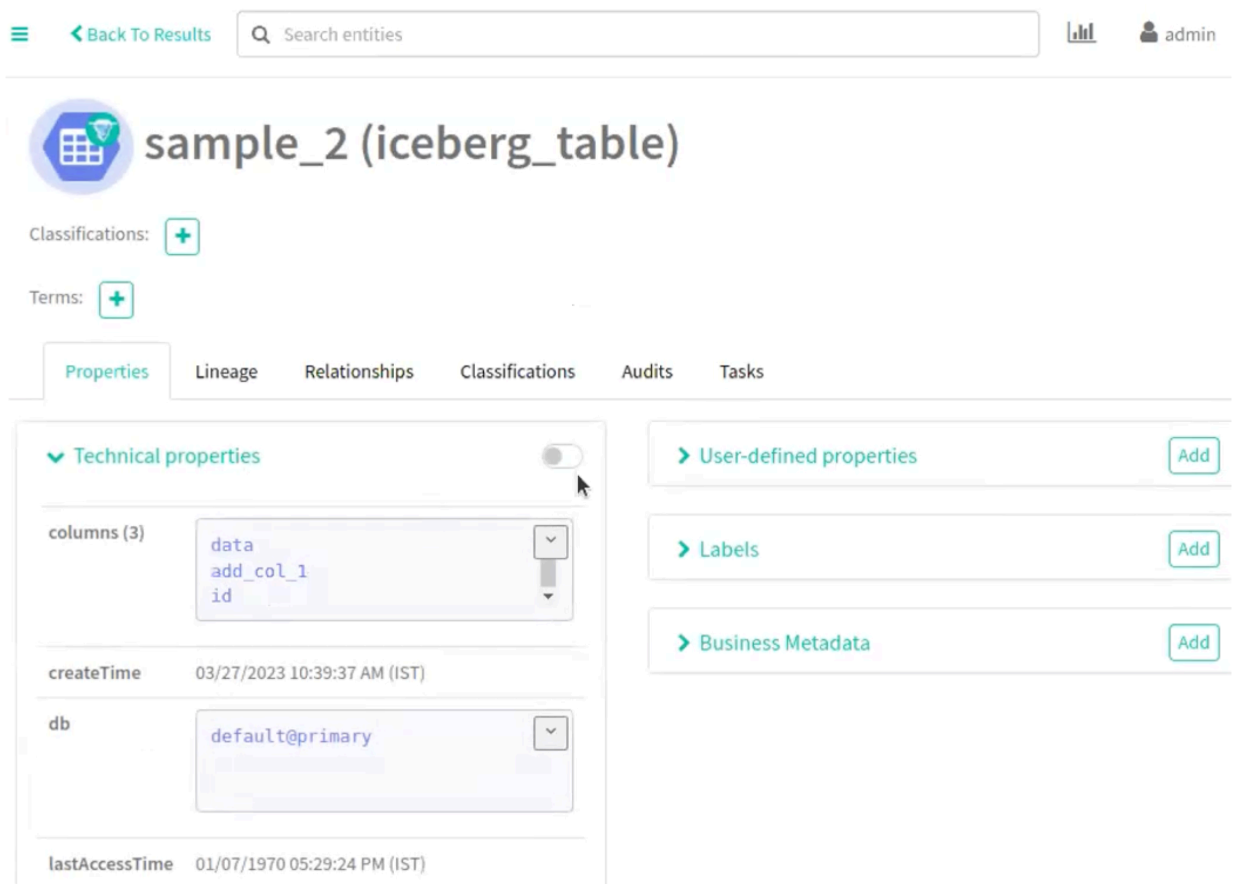
Key	Value
columns (2)	data id
db	default@primary
outputFromProcesses (1)	default.sample_2@primary:1679893777000

   Run the following command in your Spark shell to include a column:

5. spark.sql("ALTER TABLE spark_catalog.default.sample_2 ADD COLUMN (add_col_1 string)");

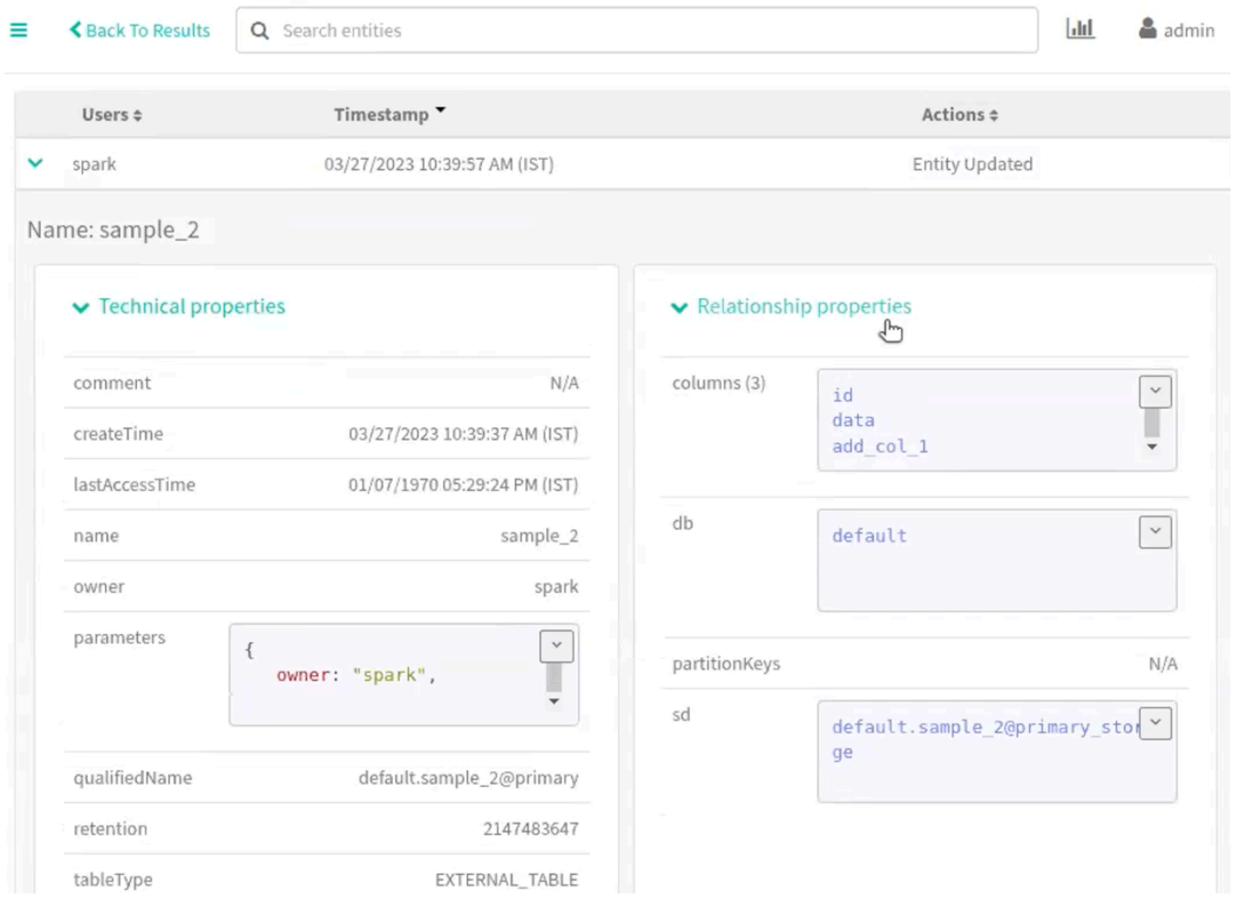
6. Navigate accordingly in the Atlas UI to view the changes.

   The following images provide information about Iceberg schema creation process.



sample_2 (iceberg_table)

columns (3)	data add_col_1 id
createTime	03/27/2023 10:39:37 AM (IST)
db	default@primary
lastAccessTime	01/07/1970 05:29:24 PM (IST)



Users	Timestamp	Actions
spark	03/27/2023 10:39:57 AM (IST)	Entity Updated

Name: sample_2

Technical properties

comment	N/A
createTime	03/27/2023 10:39:37 AM (IST)
lastAccessTime	01/07/1970 05:29:24 PM (IST)
name	sample_2
owner	spark
parameters	{ owner: "spark",
qualifiedName	default.sample_2@primary
retention	2147483647
tableType	EXTERNAL_TABLE

Relationship properties

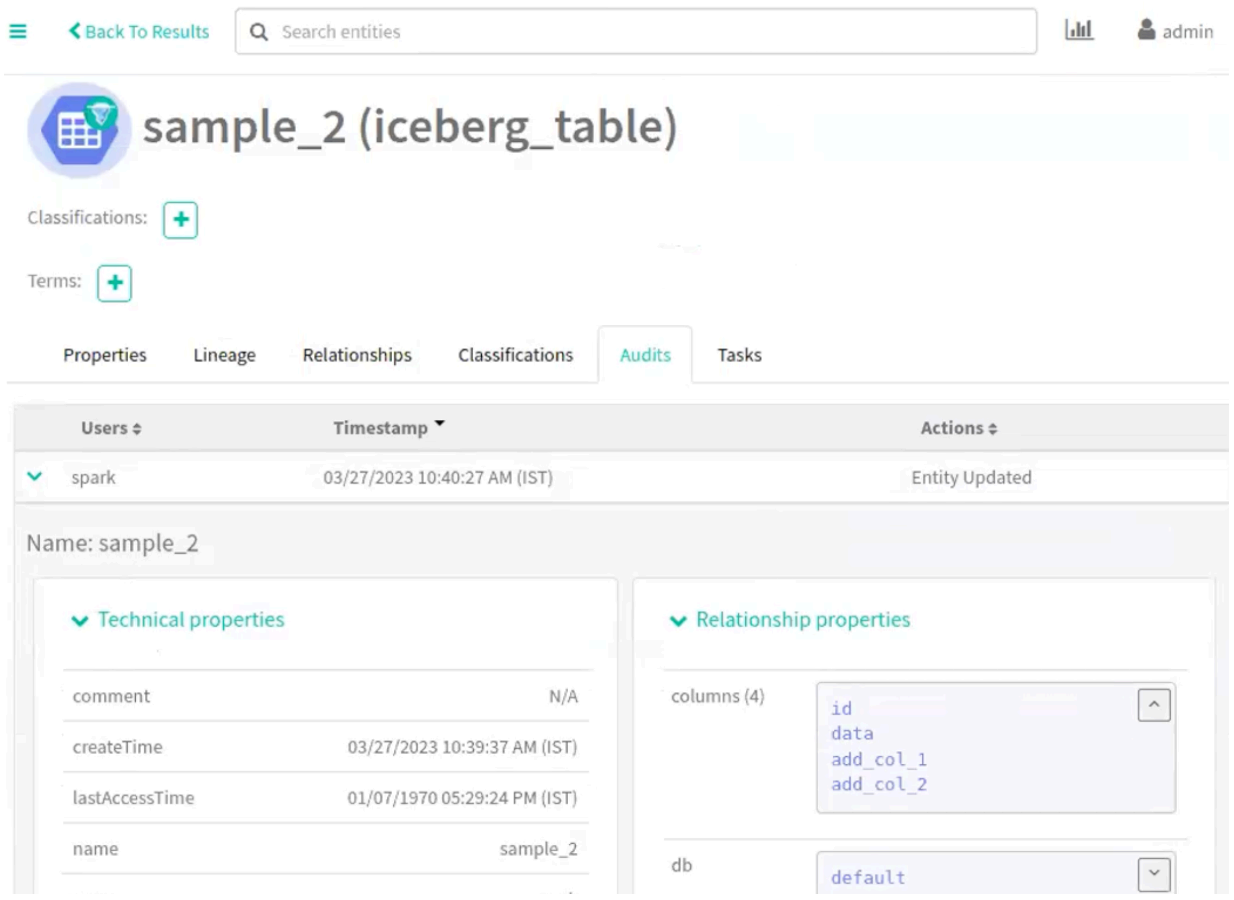
columns (3)	id data add_col_1
db	default
partitionKeys	N/A
sd	default.sample_2@primary_storage

Run the following command in your Spark shell to include the second column:

7. spark.sql("ALTER TABLE spark_catalog.default.sample_2 ADD COLUMN (add_col_2 string)");

8. Navigate accordingly in the Atlas UI to view the changes.

The following image provide information about Iceberg schema creation process.



Users	Timestamp	Actions
spark	03/27/2023 10:40:27 AM (IST)	Entity Updated

Name: sample_2

Technical properties

comment	N/A
createTime	03/27/2023 10:39:37 AM (IST)
lastAccessTime	01/07/1970 05:29:24 PM (IST)
name	sample_2

Relationship properties

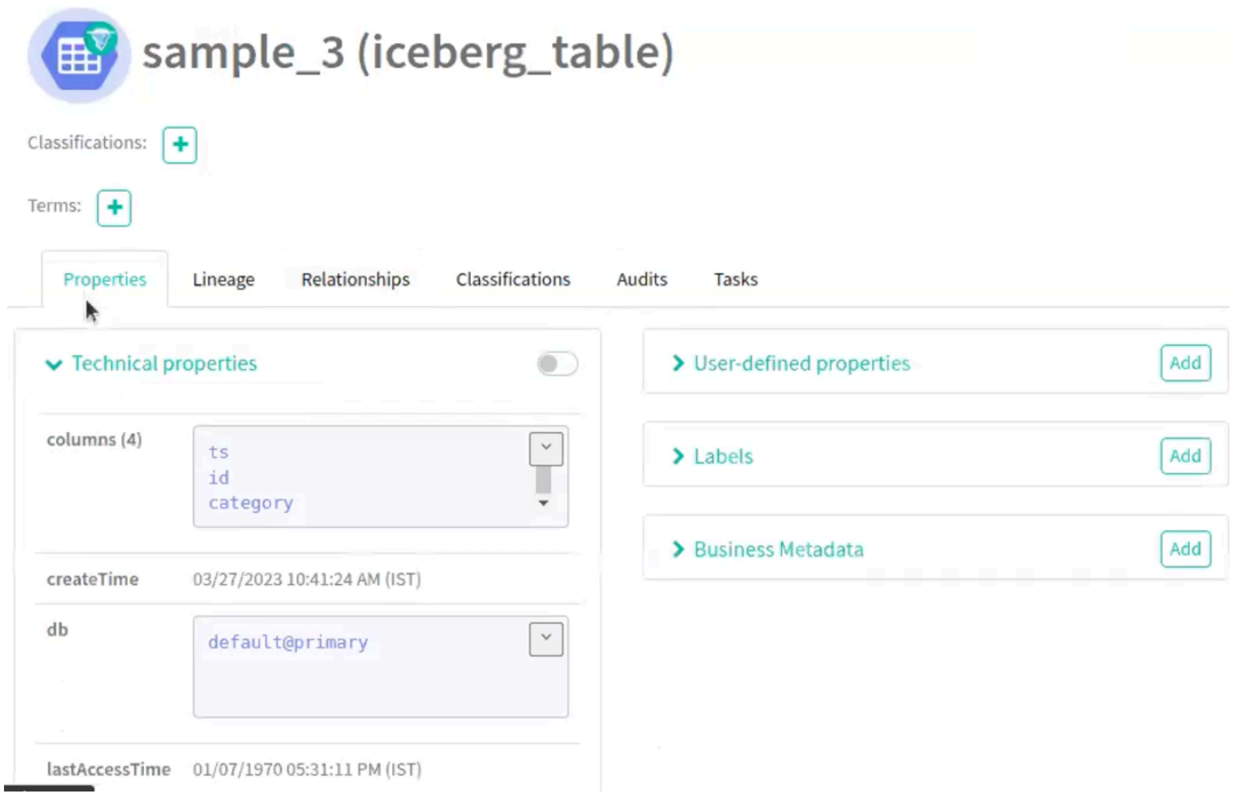
columns (4)	id data add_col_1 add_col_2
db	default

Run the following command in your Spark shell to create a Partition Specification in a new table (sample_3):

9. spark.sql("CREATE TABLE spark_catalog.default.sample_3 (id bigint,data string,category string,ts timestamp) USING iceberg PARTITIONED BY (bucket(16, id), days(ts), category)");

10. Navigate accordingly in the Atlas UI to view the changes.

The following images provide information about Iceberg partition specification process.


sample_3 (iceberg_table)

Classifications: +

Terms: +

Properties | Lineage | Relationships | Classifications | Audits | Tasks

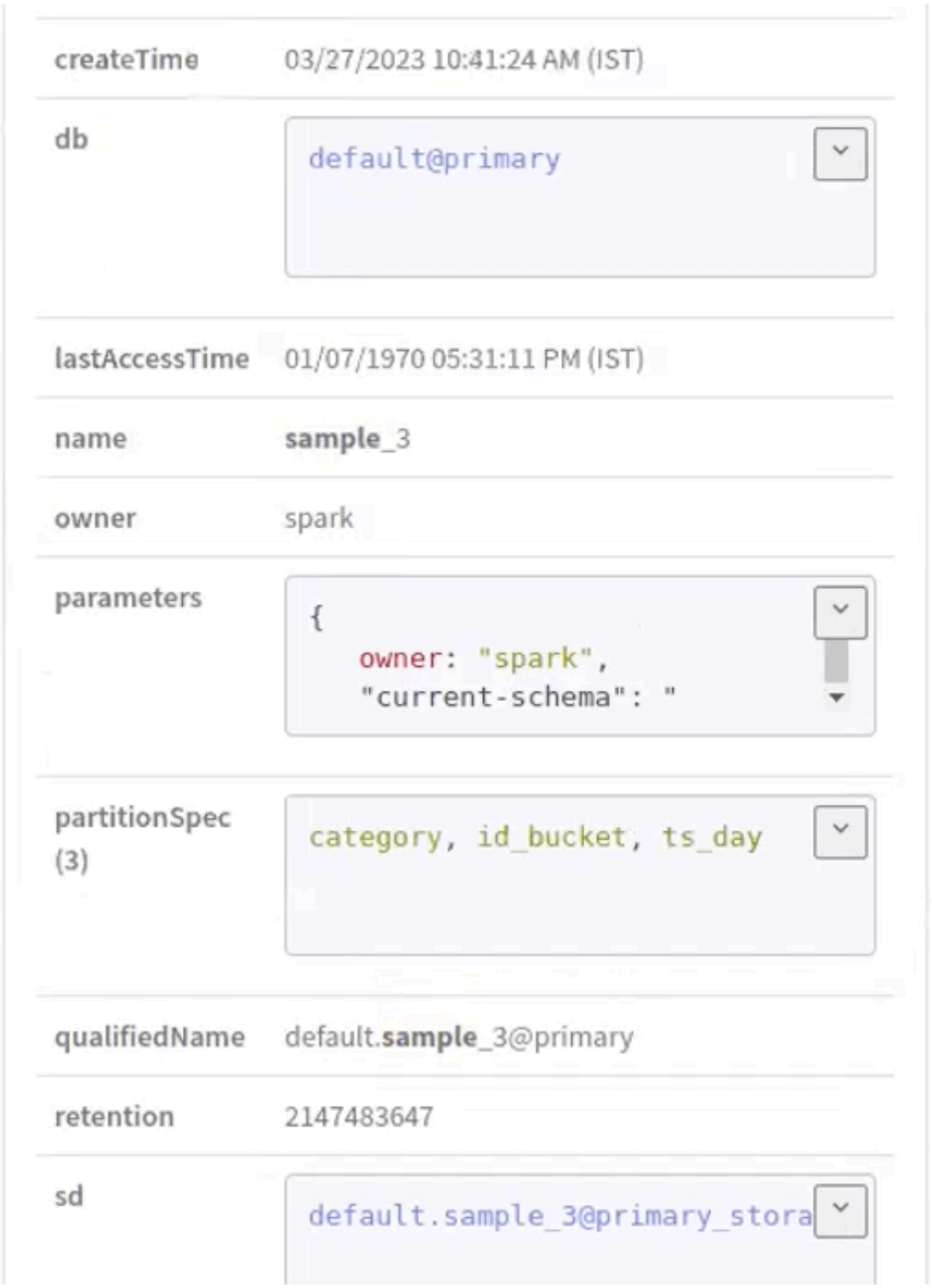
Technical properties

columns (4)	ts id category
createTime	03/27/2023 10:41:24 AM (IST)
db	default@primary
lastAccessTime	01/07/1970 05:31:11 PM (IST)

User-defined properties Add

Labels Add

Business Metadata Add



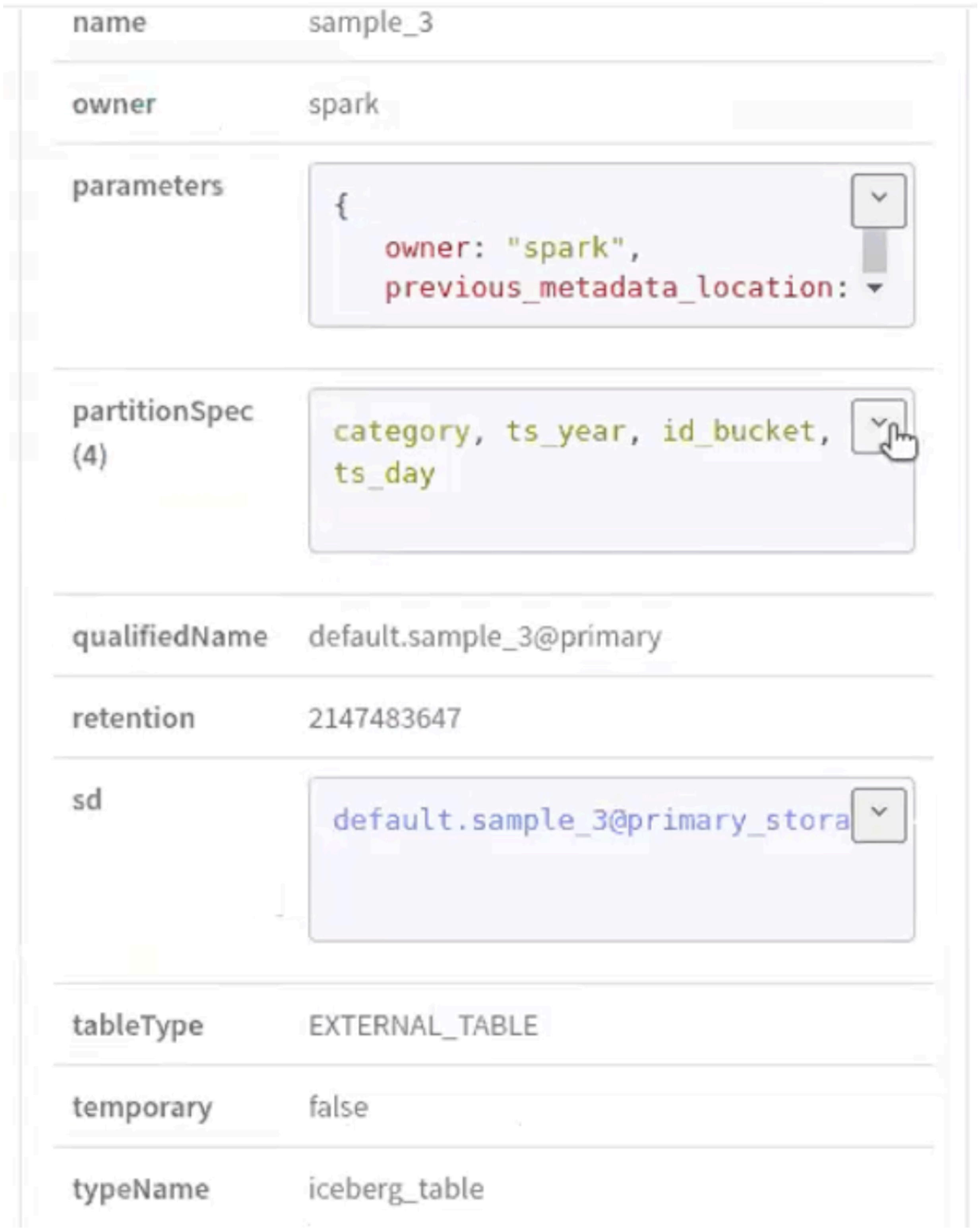
createTime	03/27/2023 10:41:24 AM (IST)
db	default@primary
lastAccessTime	01/07/1970 05:31:11 PM (IST)
name	**sample**_3
owner	spark
parameters	{ owner: "spark", "current-schema": "
partitionSpec (3)	category, id_bucket, ts_day
qualifiedName	default.**sample**_3@primary
retention	2147483647
sd	default.sample_3@primary_stora

Run the following command in your Spark shell to create a Partition Evolution in a new table (sample_3):

11. spark.sql("ALTER TABLE spark_catalog.default.sample_3 ADD PARTITION FIELD years(ts)");

12. Navigate accordingly in the Atlas UI to view the changes.

The following images provide information about Iceberg partition evolution process.



name	sample_3
owner	spark
parameters	{ owner: "spark", previous_metadata_location:
partitionSpec (4)	category, ts_year, id_bucket, ts_day
qualifiedName	default.sample_3@primary
retention	2147483647
sd	default.sample_3@primary_stora
tableType	EXTERNAL_TABLE
temporary	false
typeName	iceberg_table

Using the Hive shell

Using Hive, you can create an Iceberg table followed by using the CTAS command to alter or copy the existing Hive table and its properties into the Iceberg table.

Before you begin

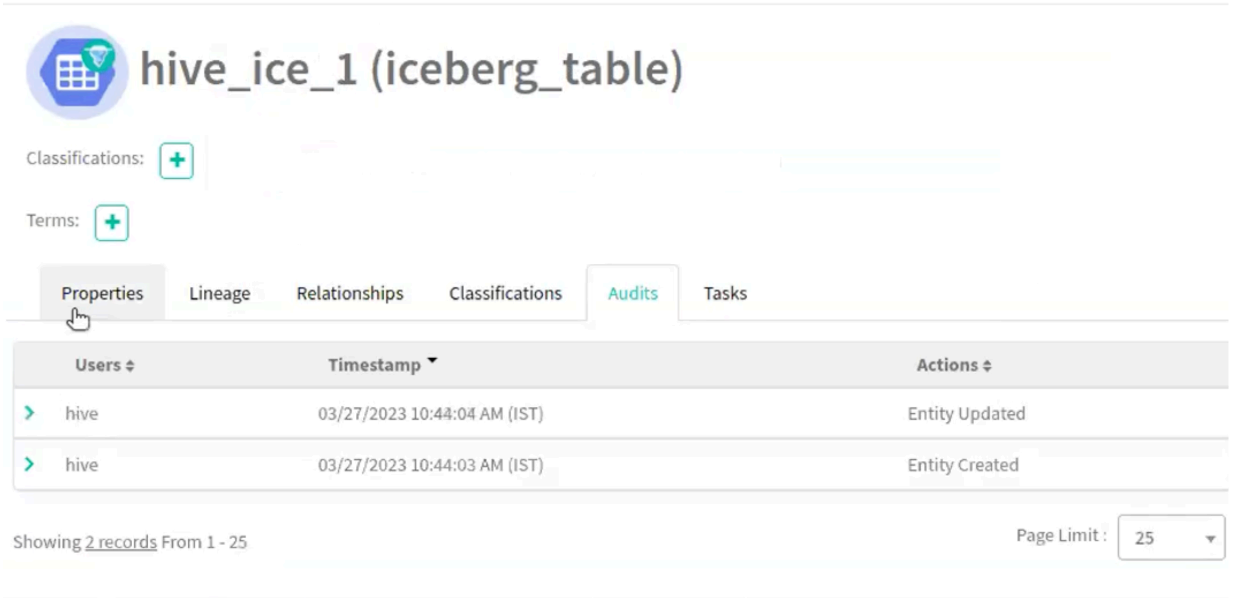
In this case, you create an external table and alter an existing Hive table to Iceberg table using the Hive engine.

Run the following command in your Hive shell to create an Iceberg table.

Procedure

1. create external table if not exists hive_ice_1 (CountryID int, CountryName string, Capital string, Population string) STORED BY ICEBERG STORED AS PARQUET;

2. Navigate accordingly in the Atlas UI to view the changes.
   The following images provide information about Iceberg table creation process.



hive_ice_1 (iceberg_table)

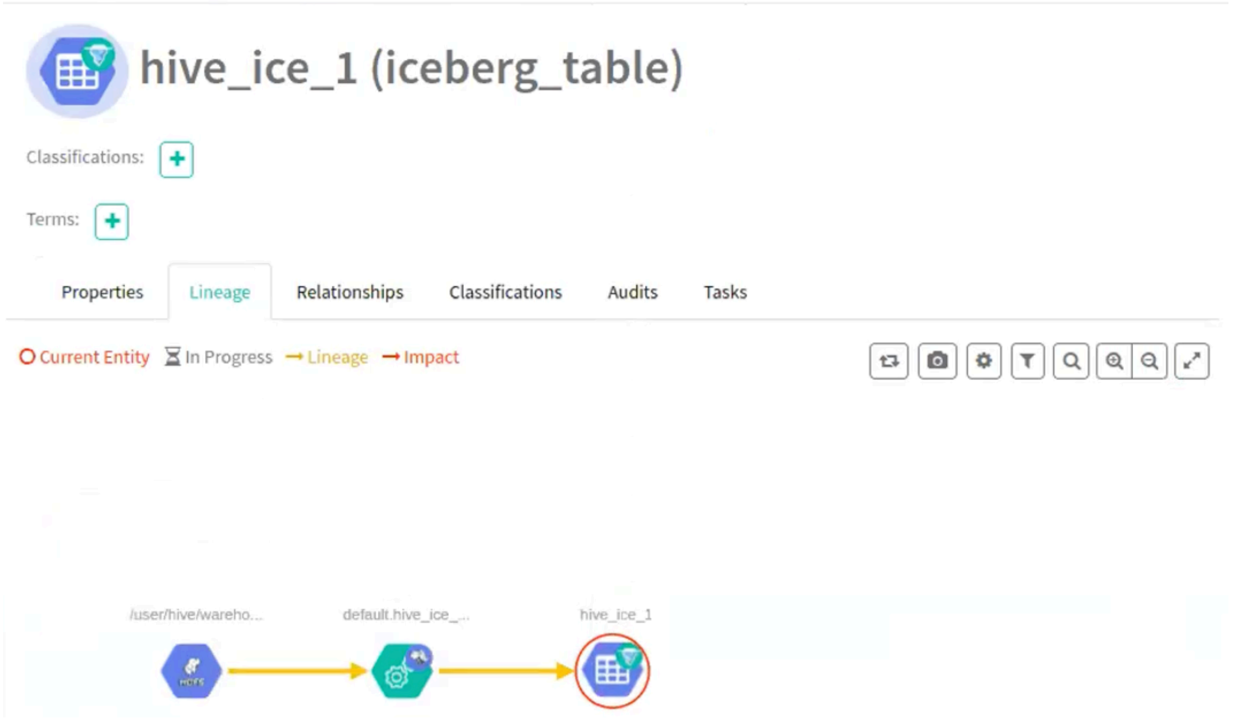
Classifications:

Terms:

Users	Timestamp	Actions
hive	03/27/2023 10:44:04 AM (IST)	Entity Updated
hive	03/27/2023 10:44:03 AM (IST)	Entity Created

Showing 2 records From 1 - 25

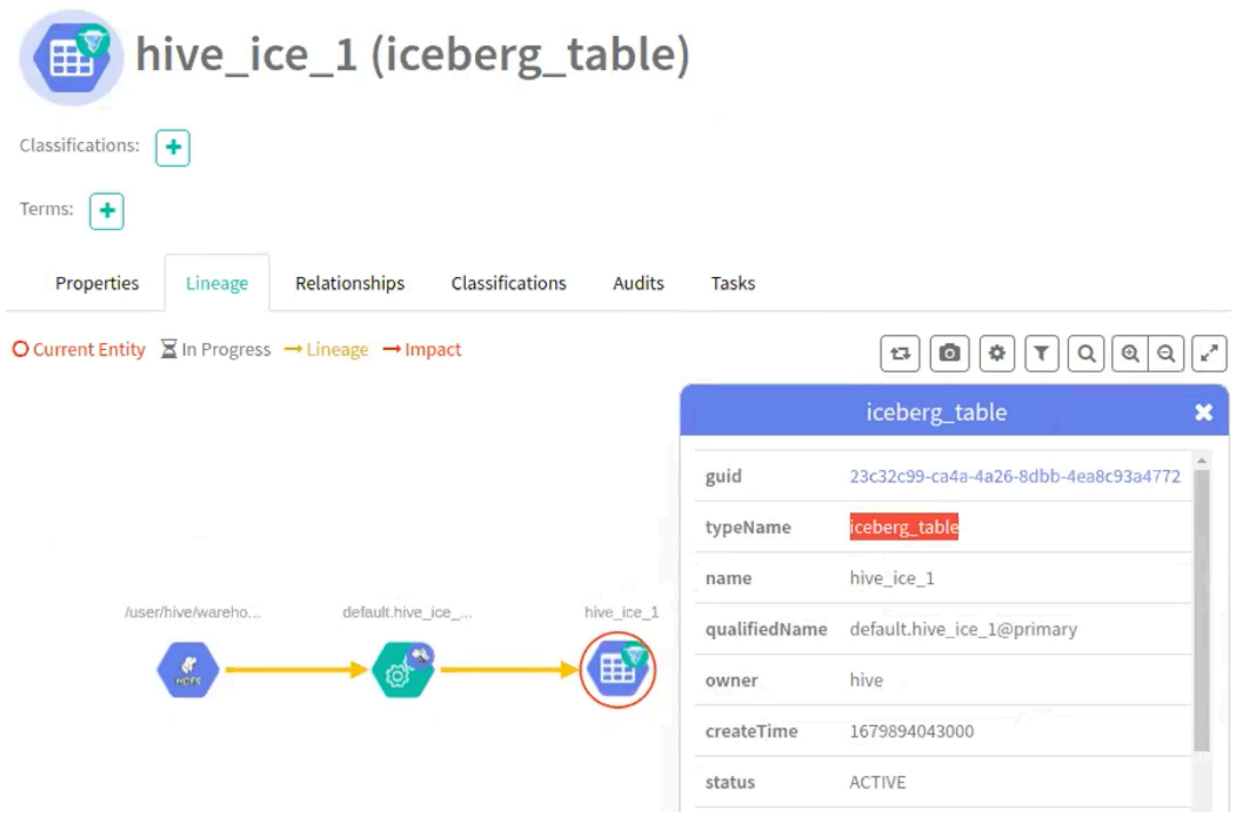
Page Limit : 25



hive_ice_1 (iceberg_table)

Classifications:

Terms:



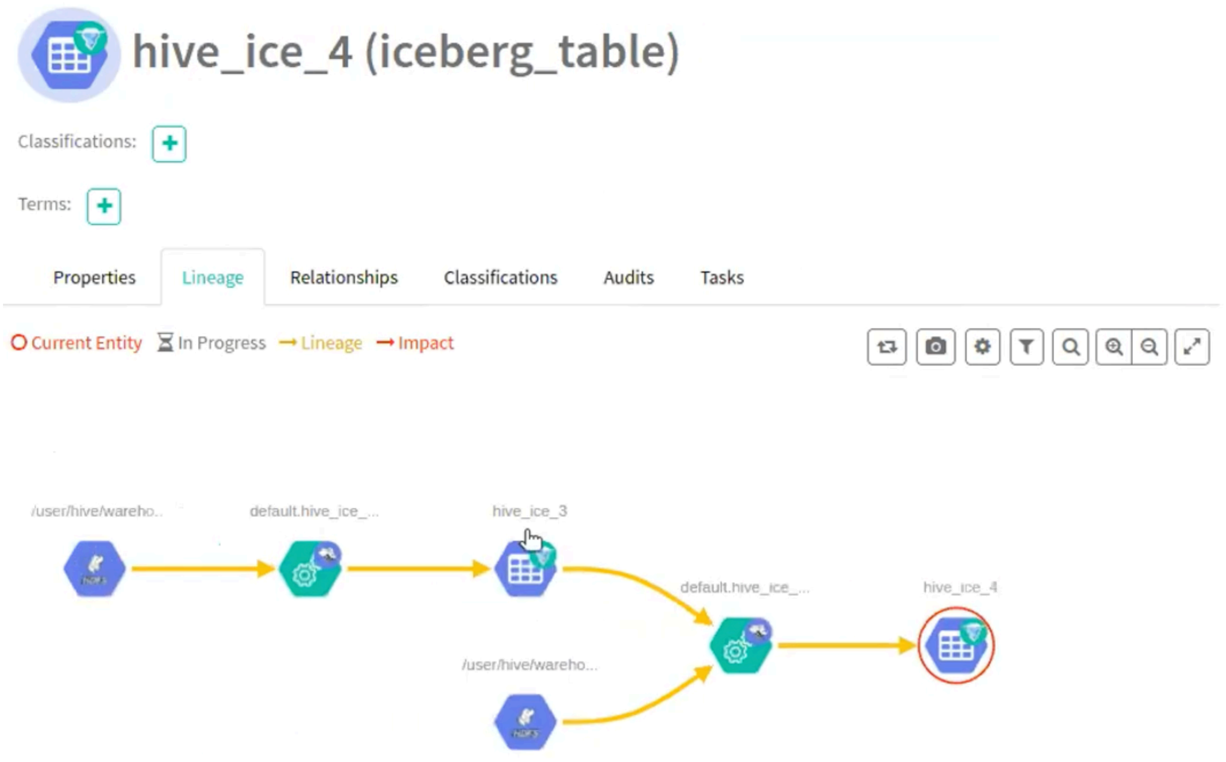
guid	23c32c99-ca4a-4a26-8dbb-4ea8c93a4772
typeName	iceberg_table
name	hive_ice_1
qualifiedName	default.hive_ice_1@primary
owner	hive
createTime	1679894043000
status	ACTIVE

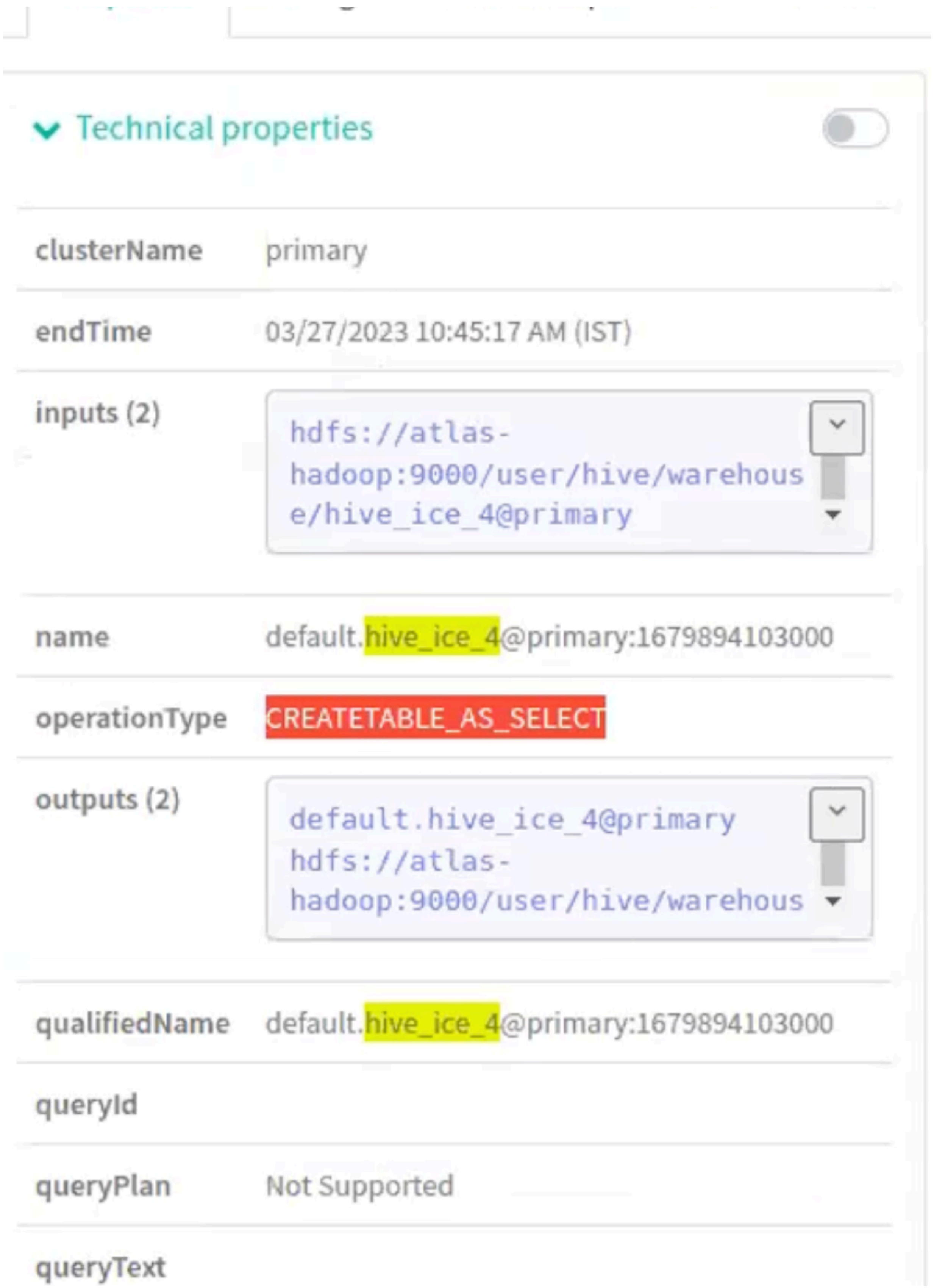
Run the following commands in your Hive shell to copy the contents of one table (hive_ice_3) to another newly created table (hive_ice_4).

3. create external table if not exists hive_ice_3 (CountryID int, CountryName string, Capital string, Population string) STORED BY ICEBERG STORED AS PARQUET;

4. create external table if not exists hive_ice_4 STORED BY ICEBERG STORED AS PARQUET as select * from hive_ice_3;

   The following images provide information about copying contents from one table to another.





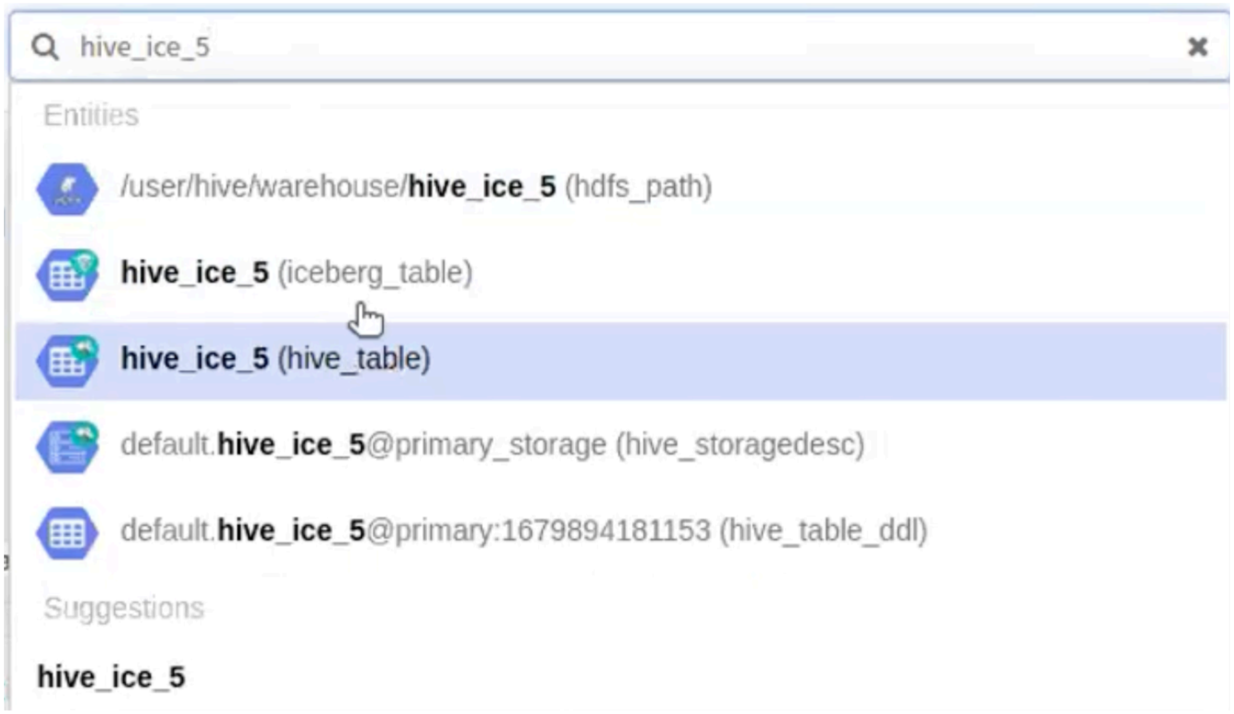
clusterName	primary
endTime	03/27/2023 10:45:17 AM (IST)
inputs (2)	hdfs://atlas-hadoop:9000/user/hive/warehouse/hive_ice_4@primary
name	default.hive_ice_4@primary:1679894103000
operationType	CREATETABLE_AS_SELECT
outputs (2)	default.hive_ice_4@primary hdfs://atlas-hadoop:9000/user/hive/warehous
qualifiedName	default.hive_ice_4@primary:1679894103000
queryId	
queryPlan	Not Supported
queryText	

You can alter an existing Hive table to Iceberg table.

5. create external table if not exists hive_ice_5 (CountryID int,   CountryName string, Capital string, Population st
   ring)STORED AS   PARQUET;

6. ALTER TABLE hive_ice_5 SET TBLPROPERTIES ('storage_handler'='org.apache.iceberg.mr.hive.HiveIcebergStorageHandler');

   The following images provide information about alter tables operations.



hive_ice_5

Entities

/user/hive/warehouse/**hive_ice_5** (hdfs_path)

hive_ice_5 (iceberg_table)

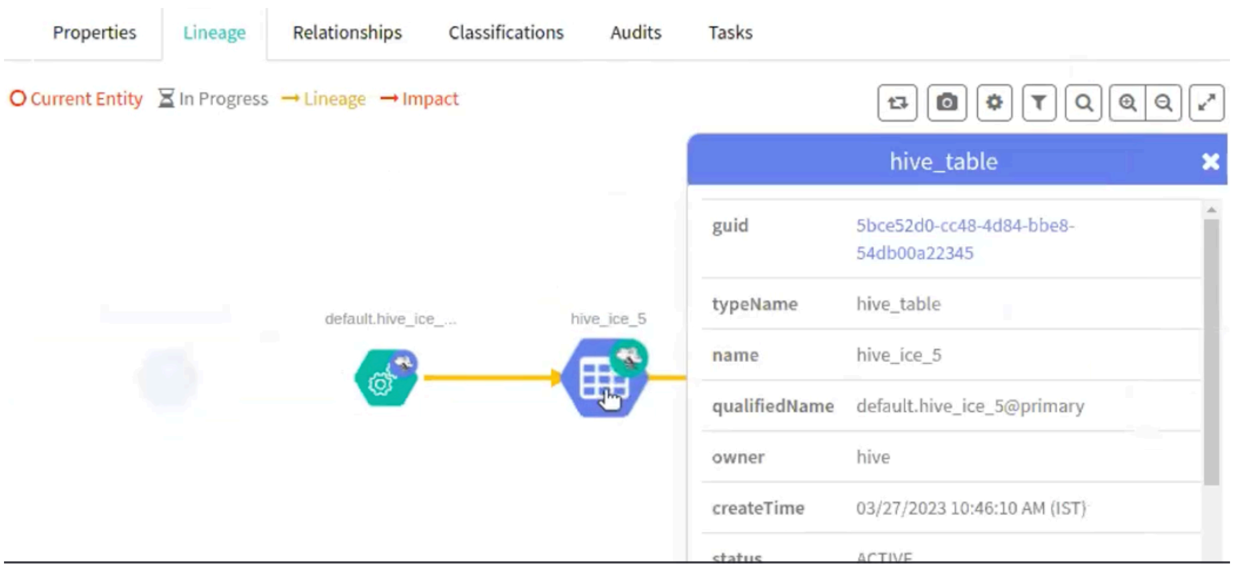
hive_ice_5 (hive_table)

default.**hive_ice_5**@primary_storage (hive_storagedesc)

default.**hive_ice_5**@primary:1679894181153 (hive_table_ddl)

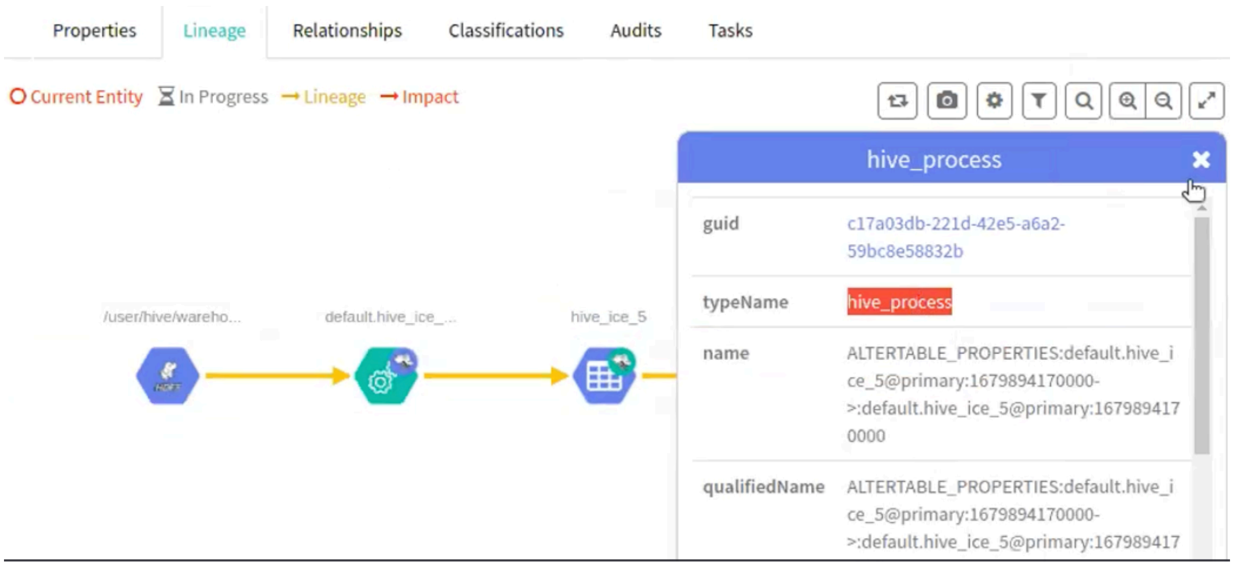
Suggestions

hive_ice_5



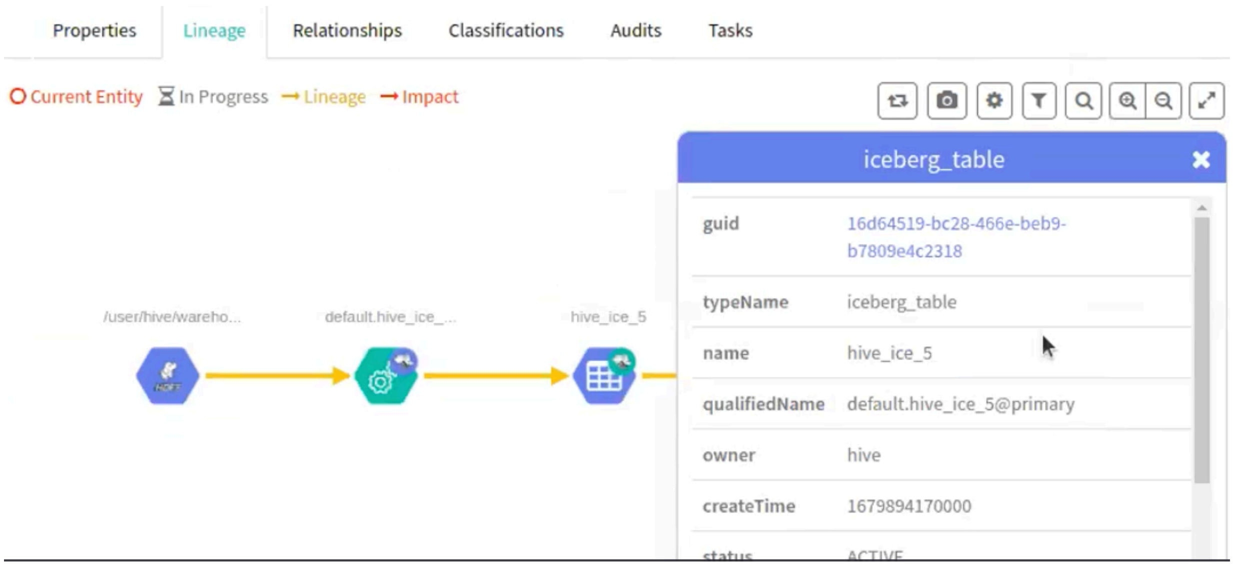
hive_table

guid	5bce52d0-cc48-4d84-bbe8-54db00a22345
typeName	hive_table
name	hive_ice_5
qualifiedName	default.hive_ice_5@primary
owner	hive
createTime	03/27/2023 10:46:10 AM (IST)
status	ACTIVE



hive_process

guid	c17a03db-221d-42e5-a6a2-59bc8e58832b
typeName	hive_process
name	ALTERTABLE_PROPERTIES:default.hive_ice_5@primary:1679894170000->:default.hive_ice_5@primary:1679894170000
qualifiedName	ALTERTABLE_PROPERTIES:default.hive_ice_5@primary:1679894170000->:default.hive_ice_5@primary:167989417



iceberg_table

guid	16d64519-bc28-466e-beb9-b7809e4c2318
typeName	iceberg_table
name	hive_ice_5
qualifiedName	default.hive_ice_5@primary
owner	hive
createTime	1679894170000
status	ACTIVE

Using the Impala shell

Using Impala, you can create an Iceberg table followed by Schema evolution, partition specification, partition evolution and CTAS operation.

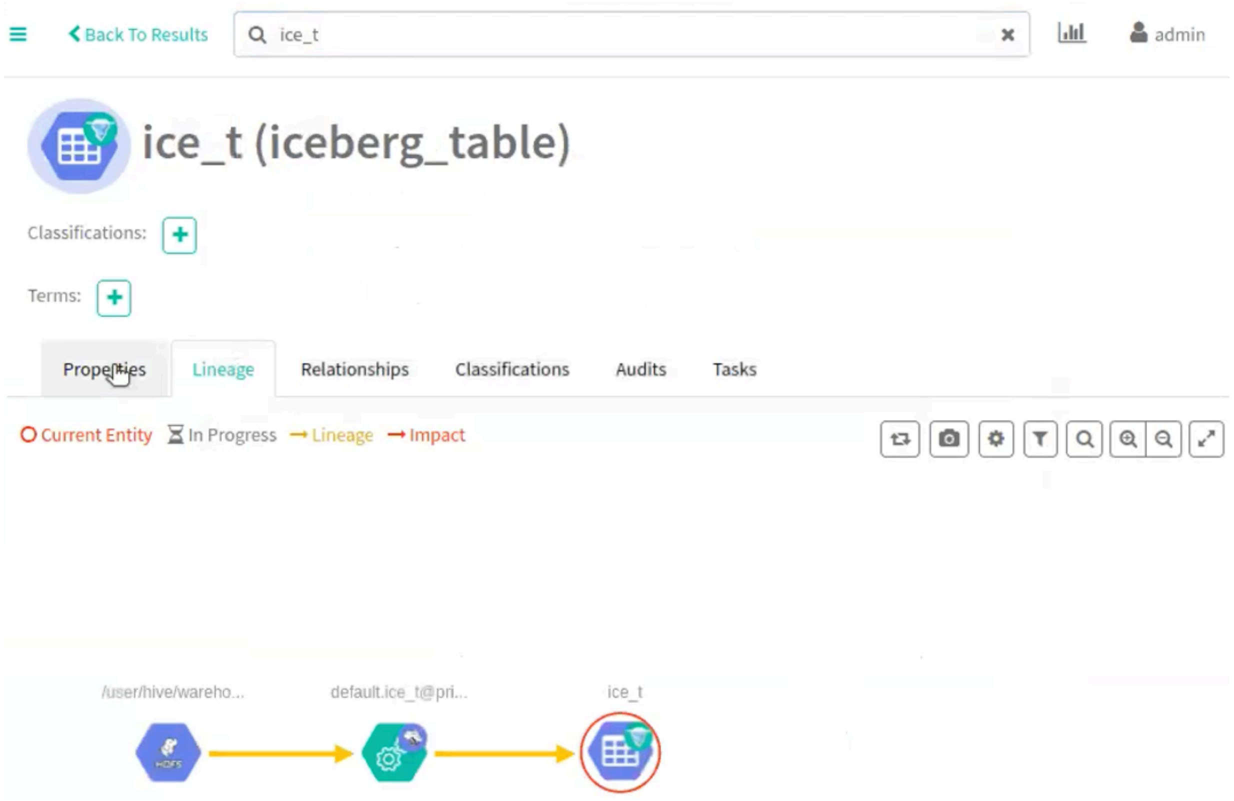
Before you begin

Run the following command in your Impala shell to create a new Iceberg table

Procedure

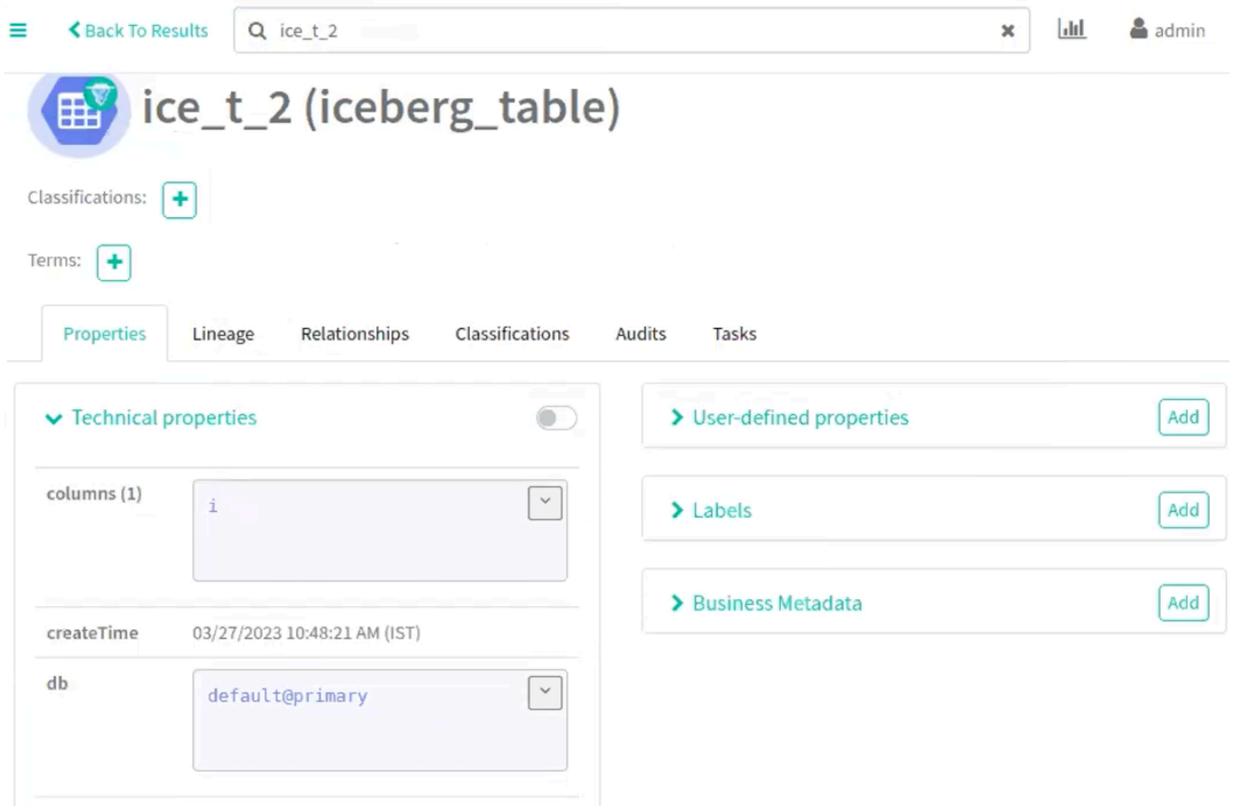
1. CREATE TABLE ice_t (i INT) STORED AS ICEBERG;

2. Navigate accordingly in the Atlas UI to view the changes.
   The following images provide information about Iceberg table creation process.



   Run the following command in your Impala shell to create a scheme evolution:

3. CREATE TABLE ice_t_2 (i INT) STORED AS ICEBERG;

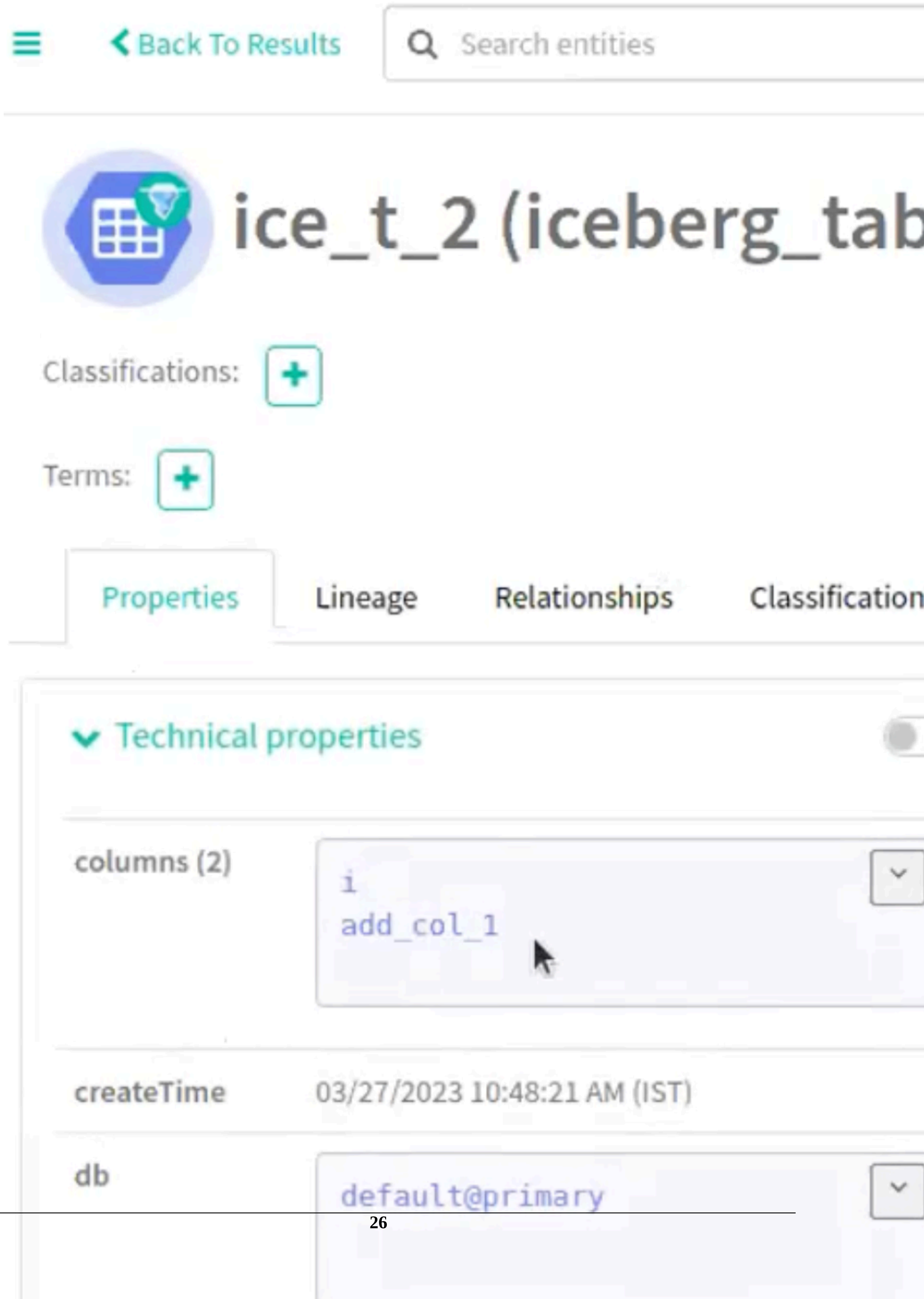


ice_t_2 (iceberg_table)

columns (1)	i
createTime	03/27/2023 10:48:21 AM (IST)
db	default@primary

Run the following command in your Impala shell to add a column to the existing table (ice_t_2):

4. alter table ice_t_2 ADD COLUMNS (add_col_1 string);



ice_t_2 (iceberg_tab

Classifications:

Terms:

Technical properties

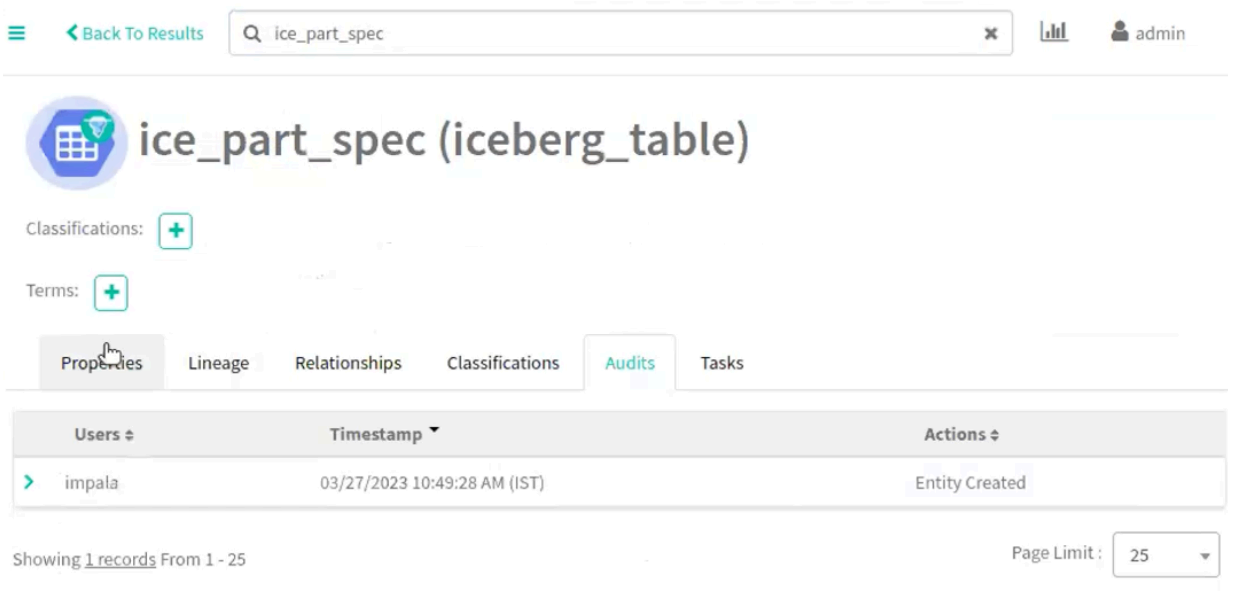
columns (2)
i
add_col_1

createTime 03/27/2023 10:48:21 AM (IST)

db
default@primary

Run the following command in your Impala shell to create a partition specification.

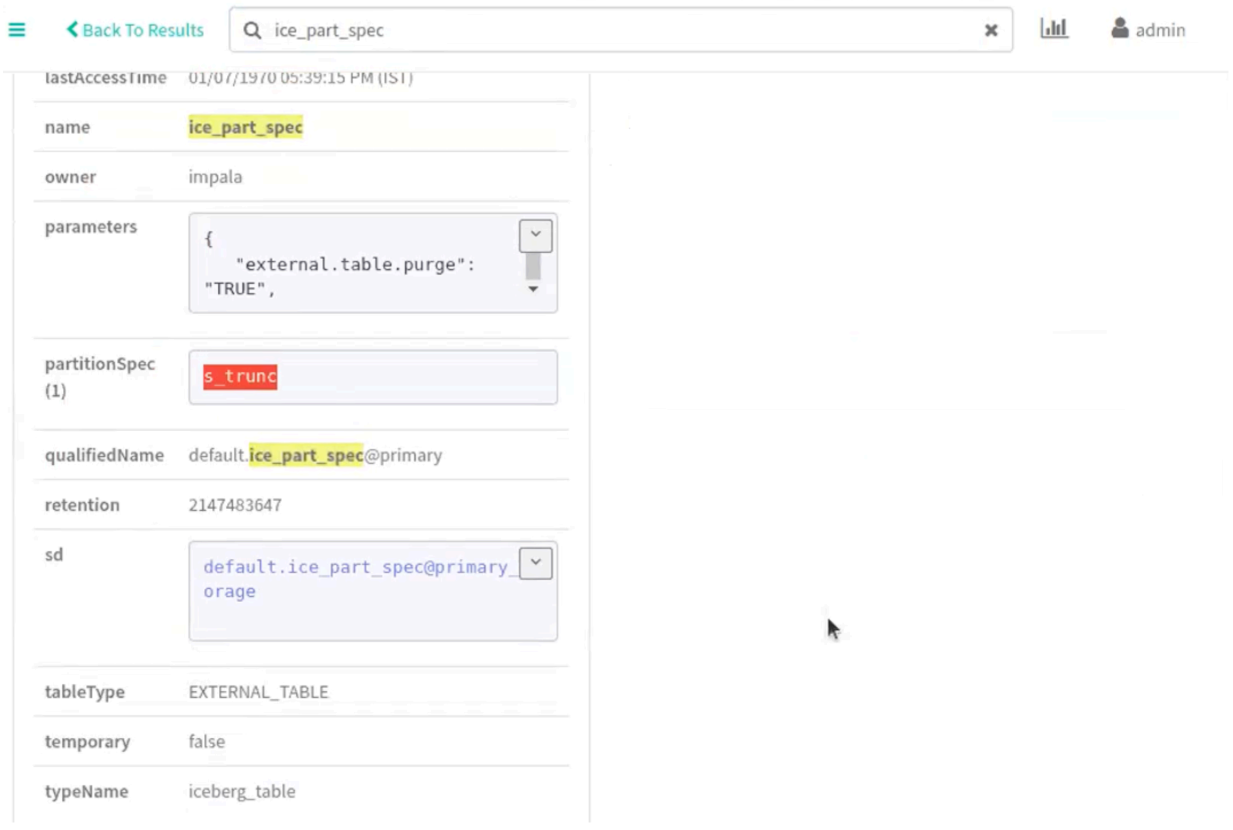
5. CREATE TABLE ice_part_spec (s string , b string) PARTITIONED BY SPEC (truncate(3, s)) STORED AS ICEBERG ;



Users	Timestamp	Actions
impala	03/27/2023 10:49:28 AM (IST)	Entity Created

Showing 1 records From 1 - 25

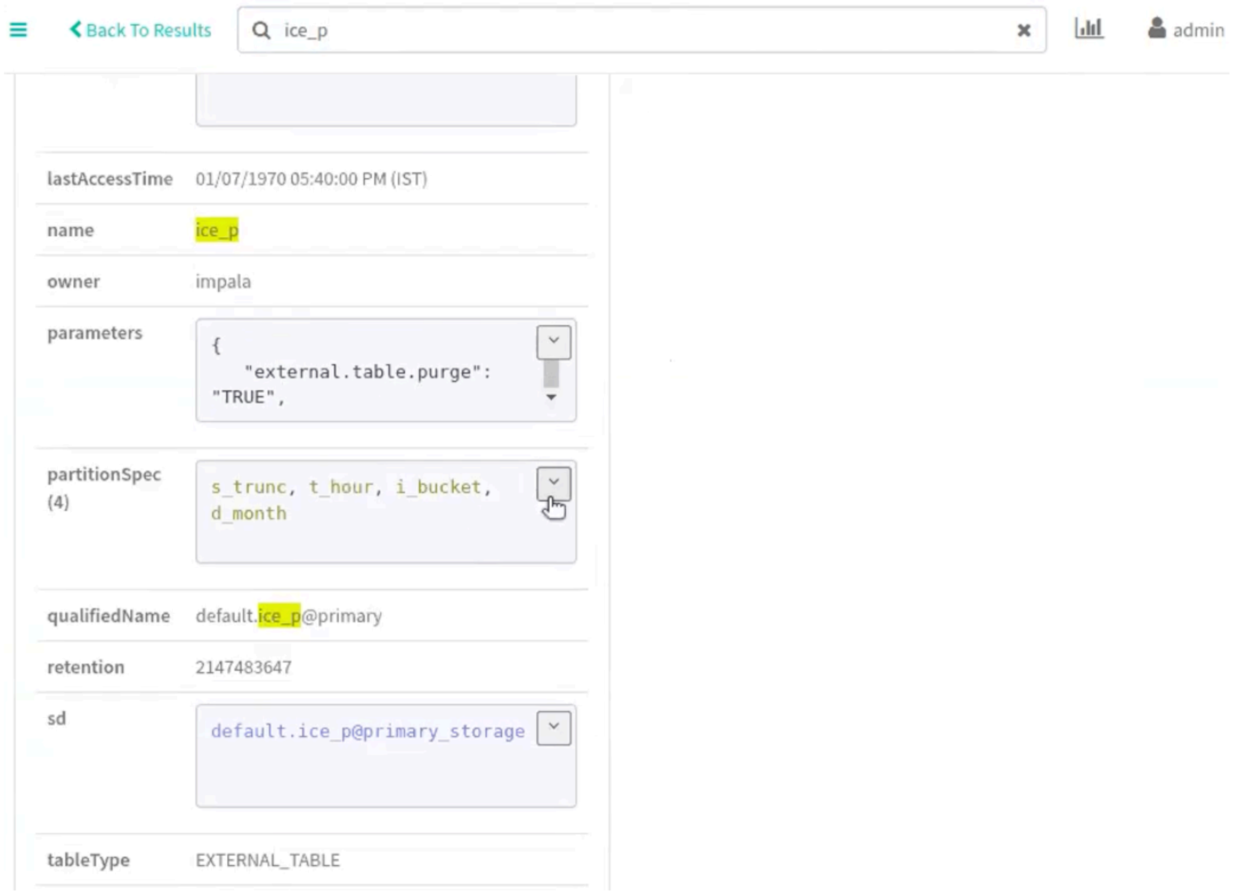
Page Limit : 25



lastAccessTime	01/07/1970 05:39:15 PM (IST)
name	ice_part_spec
owner	impala
parameters	{ "external.table.purge": "TRUE",
partitionSpec (1)	s_trunc
qualifiedName	default.ice_part_spec@primary
retention	2147483647
sd	default.ice_part_spec@primary_orage
tableType	EXTERNAL_TABLE
temporary	false
typeName	iceberg_table

Run the following command in your Impala shell to create a partition evolution.

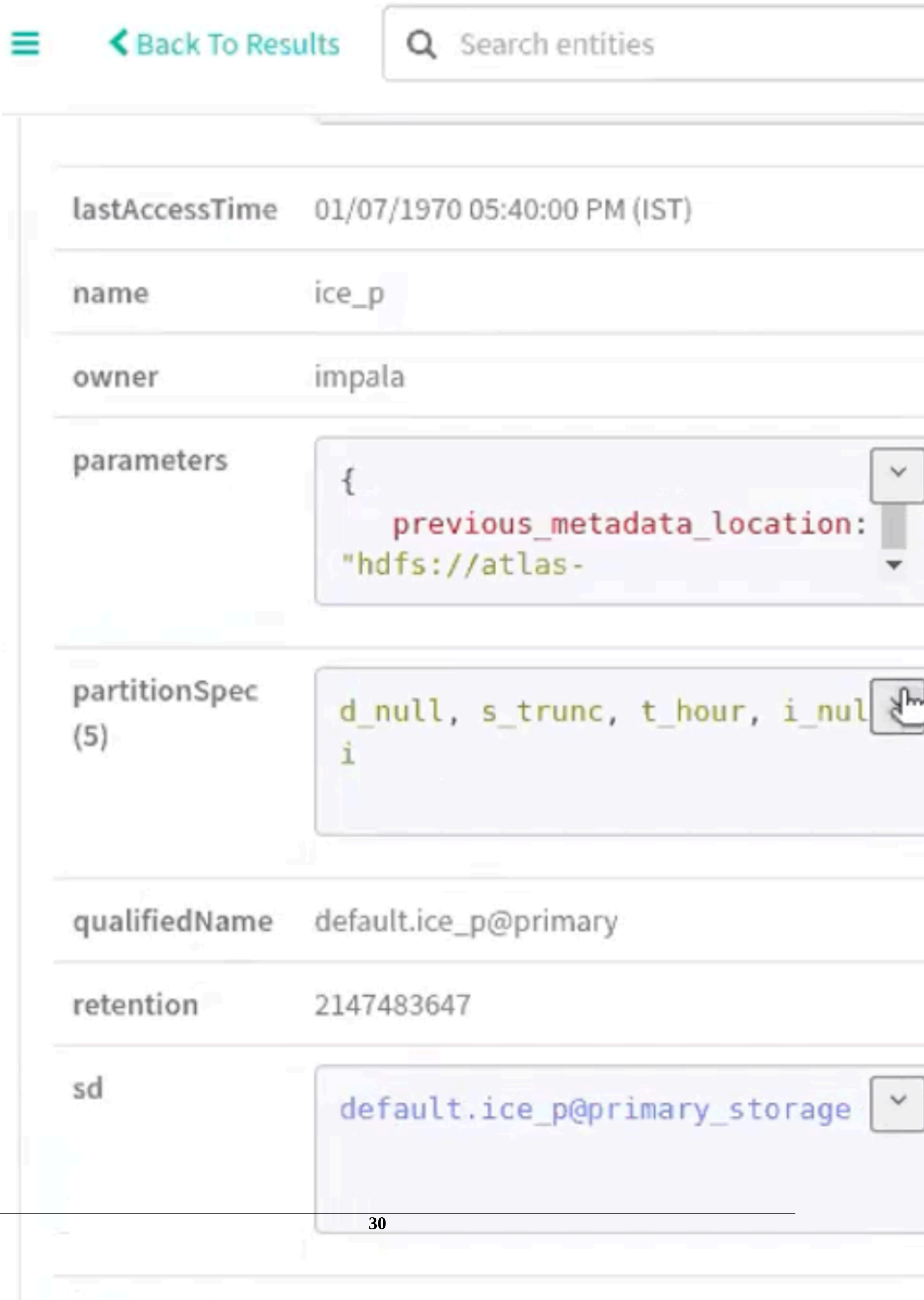
6. CREATE TABLE ice_p (i INT, d DATE, s STRING, t TIMESTAMP) PARTITIONED BY SPEC (BUCKET(5, i), MONTH(d), TRUNCATE(3, s), HOUR(t))STORED AS ICEBERG;



lastAccessTime	01/07/1970 05:40:00 PM (IST)
name	ice_p
owner	impala
parameters	{ "external.table.purge": "TRUE",
partitionSpec (4)	s_trunc, t_hour, i_bucket, d_month
qualifiedName	default.ice_p@primary
retention	2147483647
sd	default.ice_p@primary_storage
tableType	EXTERNAL_TABLE

Run the following command in your Impala shell to modify the partition specification

7. ALTER TABLE ice_p SET PARTITION SPEC (VOID(i), VOID(d), TRUNCATE(3, s), HOUR(t), i);

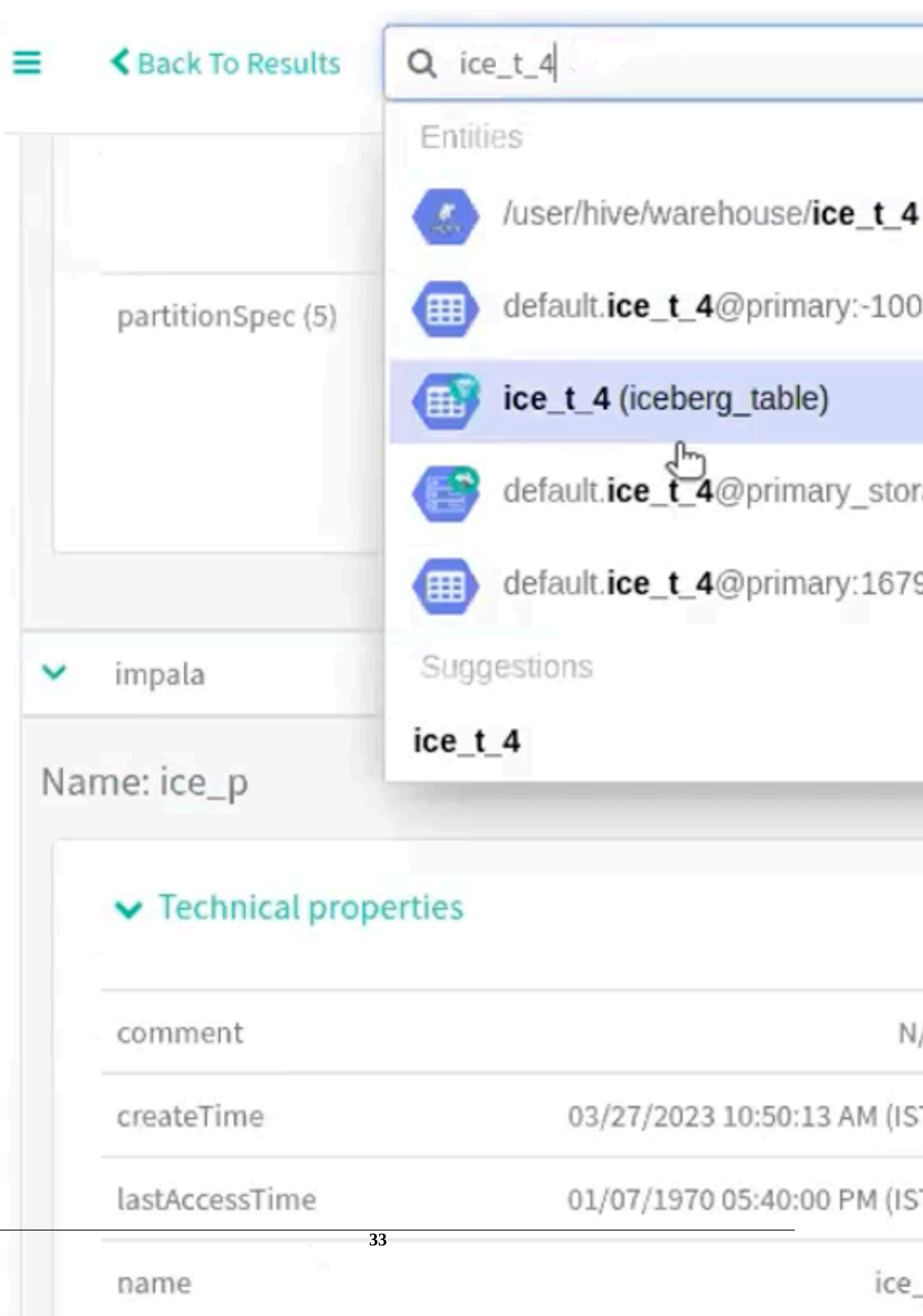


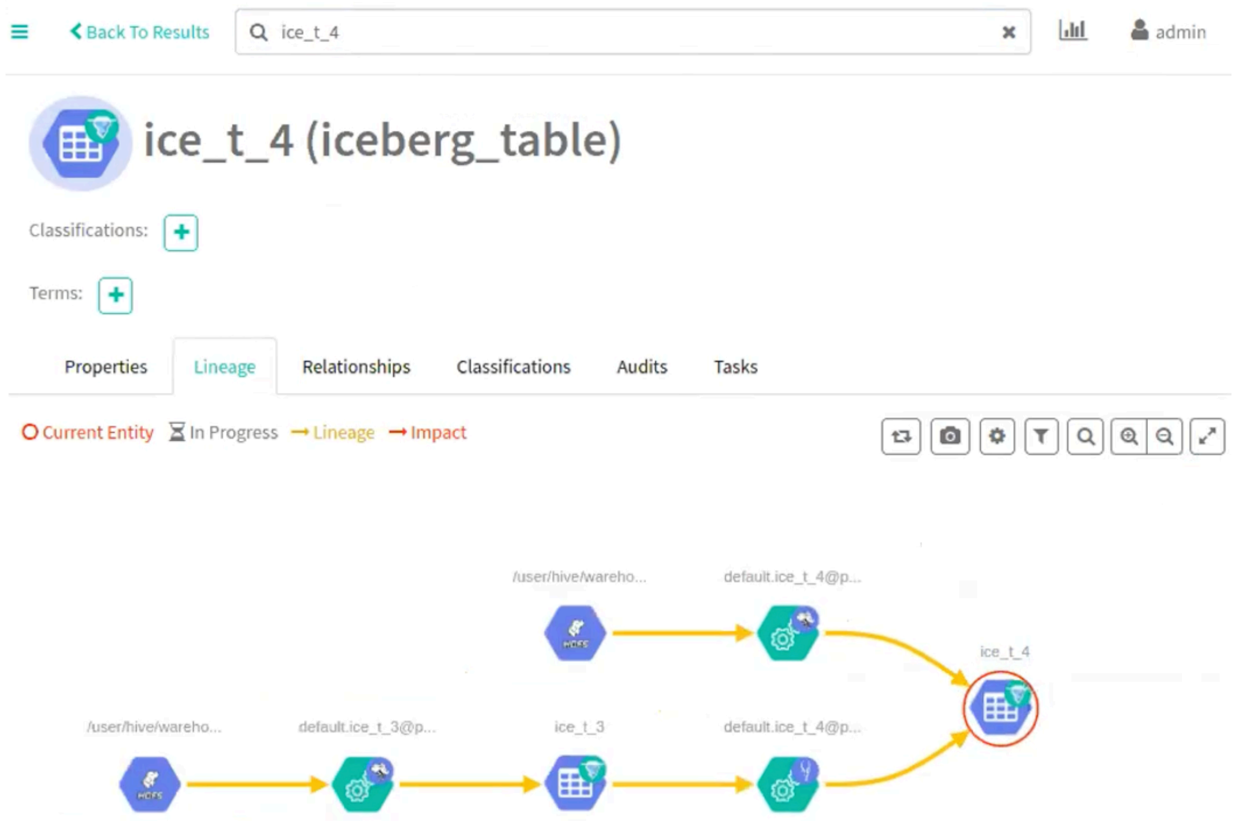
lastAccessTime	01/07/1970 05:40:00 PM (IST)
name	ice_p
owner	impala
parameters	{ previous_metadata_location: "hdfs://atlas-
partitionSpec (5)	d_null, s_trunc, t_hour, i_nul i
qualifiedName	default.ice_p@primary
retention	2147483647
sd	default.ice_p@primary_storage

Run the following commands in your Impala shell to create the contents of one table (ice_t_3) to another table (ice_t_4).

8. CREATE TABLE ice_t_3 (i INT) STORED AS ICEBERG;

9. CREATE TABLE ice_t_4 STORED AS ICEBERG as select * from ice_t_3;


Back To Results
ice_t_4
Entities
/user/hive/warehouse/ice_t_4
default.ice_t_4@primary:-100
ice_t_4 (iceberg_table)
default.ice_t_4@primary_stor
default.ice_t_4@primary:1679
Suggestions
ice_t_4
partitionSpec (5)
impala
Name: ice_p
Technical properties
comment
N/
createTime
03/27/2023 10:50:13 AM (IS
lastAccessTime
01/07/1970 05:40:00 PM (IS
name
ice_



The process is completed.