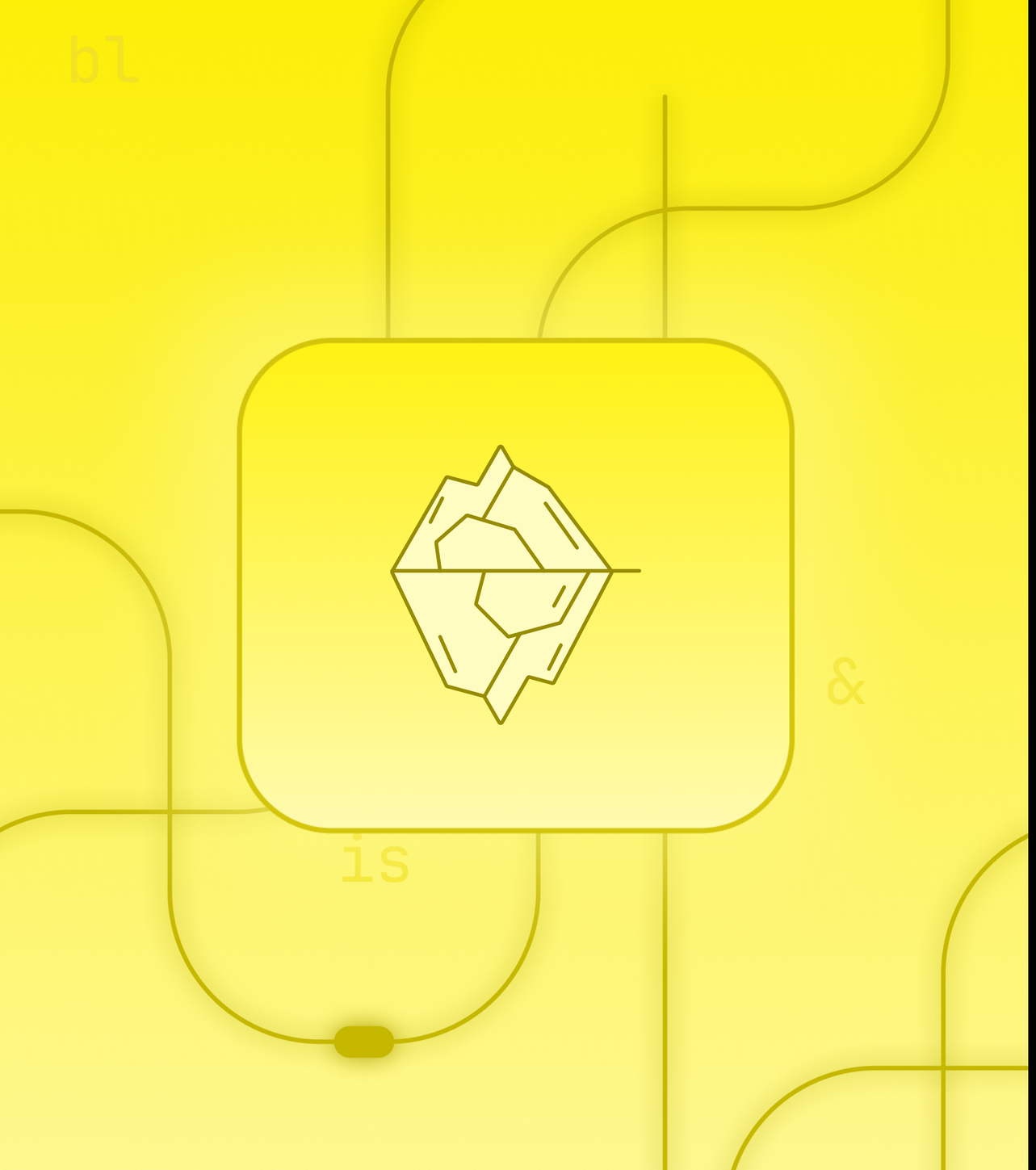
Welcome

ICEBERG SUMMIT 2026



ICEBERG SUMMIT 2026

Exploring the Iceberg Ecosystem with DuckDB-Iceberg

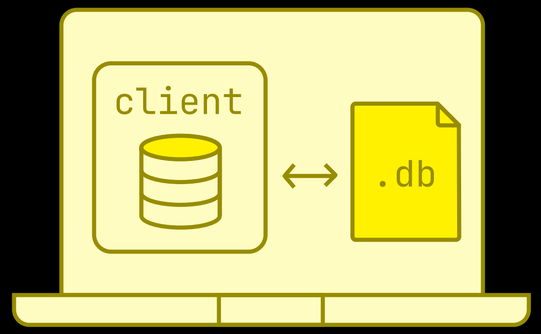
TOM EBERGEN

(DUCKDB LABS)

Agenda

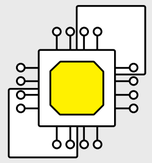
1. What is DuckDB?
2. The Iceberg Ecosystem
3. DuckDB-Iceberg: The Journey
    1. Iceberg engine compatibility
    2. REST Catalog differences
4. DuckDB-Iceberg's Testing Approach
    1. Workflows tested
    2. Catalogs tested

DuckDB = a SQL database for analytics



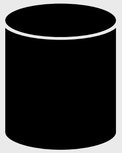

```
$ curl https://install.duckdb.org | sh
$ duckdb my.db

D CREATE SECRET (
    TYPE s3,
    PROVIDER credential_chain
  );

D ATTACH 's3_tables_arn' AS s3_tables (
     TYPE iceberg,
     ENDPOINT_TYPE s3_tables
  );

D CREATE TABLE s3_tables.my_ns.my_table
    AS SELECT * FROM 'my_file.csv';
```

The Iceberg Ecosystem



Engine Layer
Bring your own engine

Catalog layer
Unlocks ACID properties



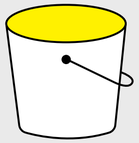
Lakekeeper

Polaris

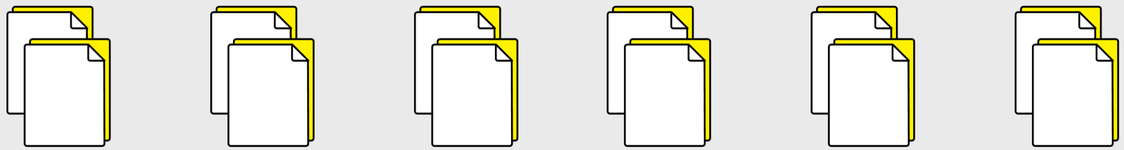
Nessie

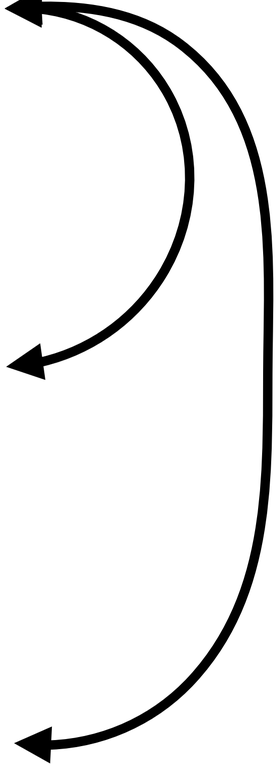
AWS Glue
Data Catalog

Amazon
S3 Tables

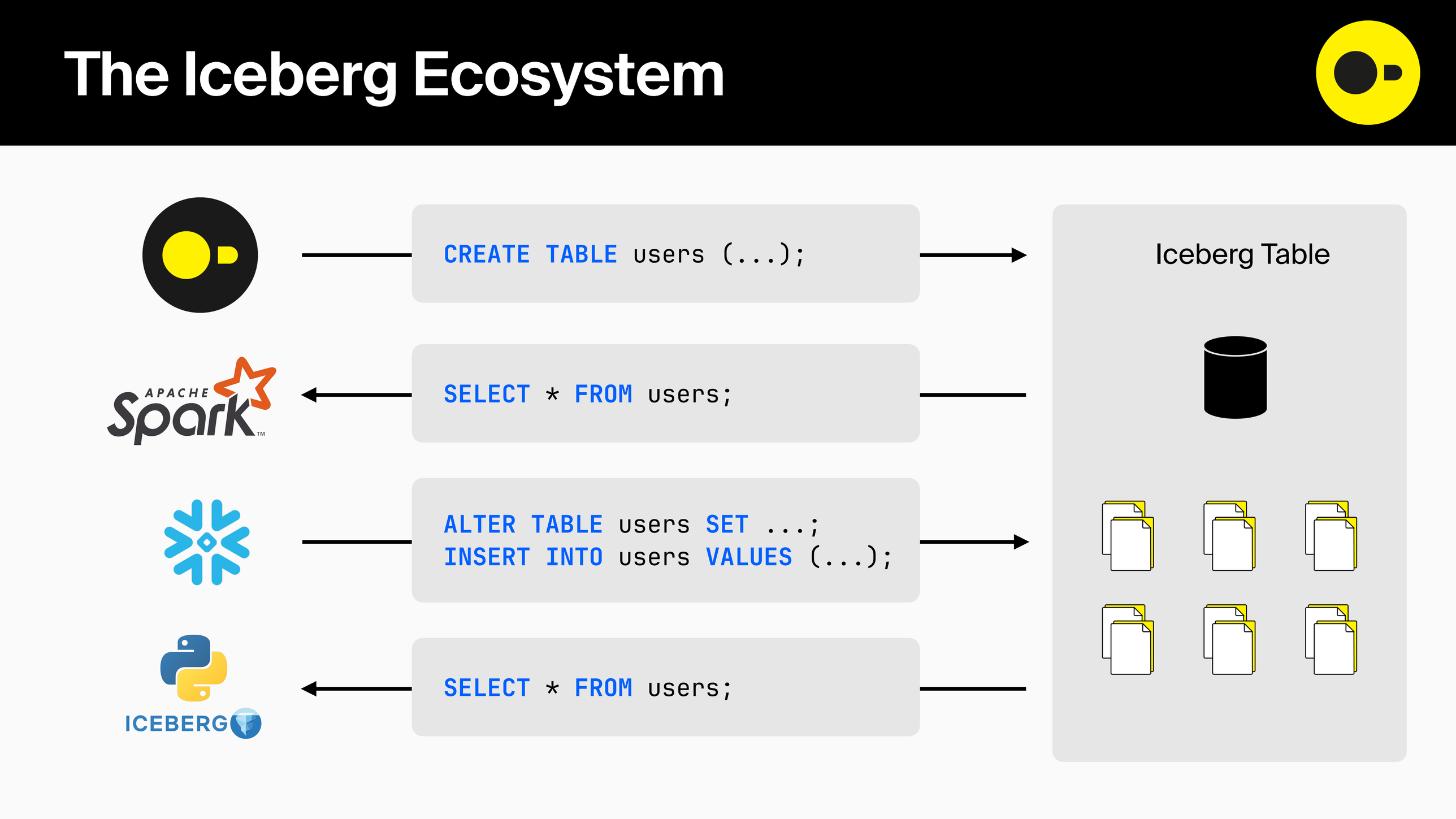


Storage layer
Scalable data layer





The Iceberg Ecosystem



Iceberg Engines

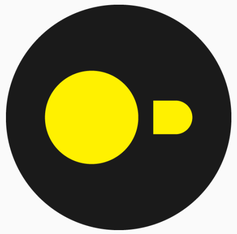
Engine Integration

Iceberg has been designed and developed to be an open community standard with a specification to ensure compatibility across languages and implementations.

```
CREATE TABLE joined (
    customer_id INT,
    joined_ts TIMESTAMPTZ,
    aux VARCHAR
  ) PARTITIONED BY (day(joined_ts)) AS
SELECT * FROM ...;
```

https://iceberg.apache.org/docs/nightly/#open-standard

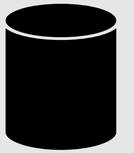
Engine Integration

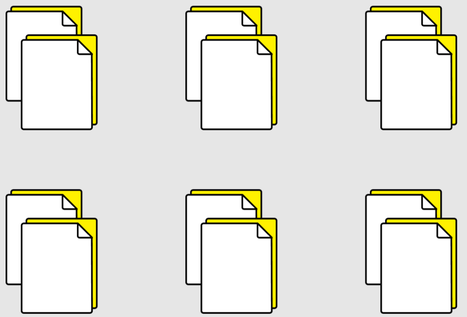


```
CREATE TABLE joined (
    customer_id INT,
    joined_ts TIMESTAMPTZ,
    aux VARCHAR
  ) PARTITIONED BY (day(joined_ts)) AS
SELECT * FROM ...;
```

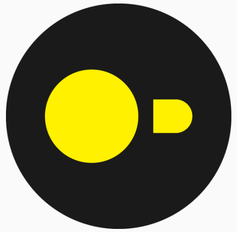
```
SELECT * FROM joined;
```

Iceberg Table





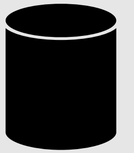
Engine Integration Challenges

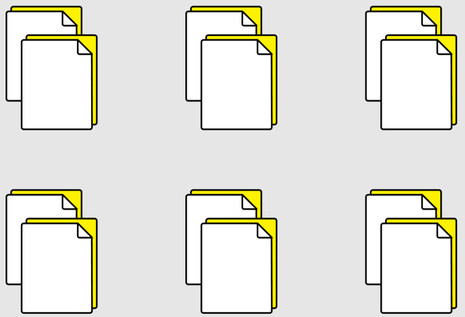


```
CREATE TABLE joined (...)
  PARTITIONED BY (day(joined_ts)) AS ...;
```

```
SELECT * FROM joined;
```

Iceberg Table





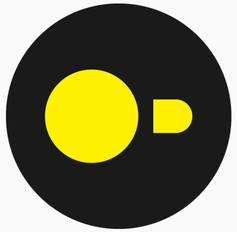
1. Avro schemas

2. Partitioning

3. Types

- Day partition types
- Lower and upper bounds
- Variant

Engine Integration Challenges



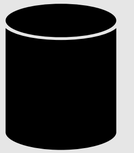
```sql
CREATE TABLE joined (...)
  PARTITIONED BY (day(joined_ts)) AS ...;
```

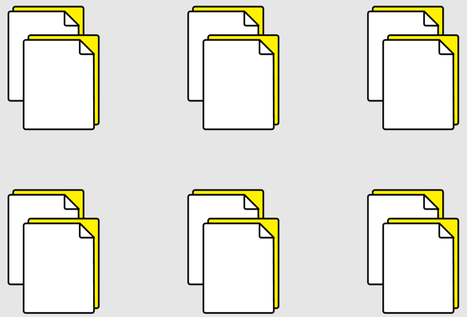
```sql
SELECT * FROM joined;
```

1. Avro schemas
2. Partitioning
3. Types

```
26/03/25 12:40:15 WARN ErrorHandlers:
Unable to parse error response

UncheckedIOException:...MismatchedInputException:
No content to map due to end-of-input
 at [Source: (String)""; line: 1, column: 0]
```

Iceberg Table





Engine Integration Challenges

1. Avro schemas

required	*required*	`2 data_file`	`data_file` `struct` (see below)	File path, partition tuple, metrics, ...

DuckDB

```
{
 "name" : "data_file",
 "type" : [ "null", {
   "type" : "record",
   "name" : "data_file",
   "fields" : [ {
```

Spark

```
{
 "name" : "data_file",
 "type" :  {
   "type" : "record",
   "name" : "data_file",
   "fields" : [ {
```

Engine Integration Challenges

1. Avro schemas

required	*required*	`2 data_file`	`data_file` `struct` (see below)	File path, partition tuple, metrics, ...

```
26/03/25 12:40:15 WARN ErrorHandlers:
Unable to parse error response

UncheckedIOException:...MismatchedInputException:
No content to map due to end-of-input
 at [Source: (String)""; line: 1, column: 0]
```

Engine Integration Challenges

1. Avro schemas

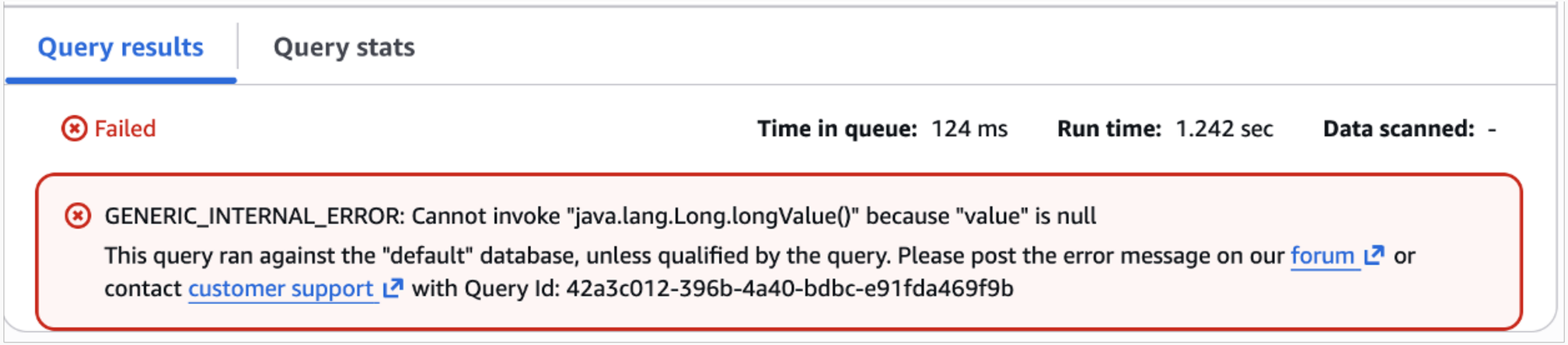
required	*required*	`2 data_file`	`data_file` `struct` (see below)	File path, partition tuple, metrics, ...

3.2	End of Life	0.13.0	1.4.3	iceberg-spark-runtime-3.2_2.12, iceberg-spark-runtime-3.2_2.13

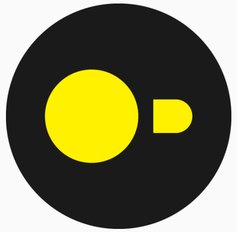
Engine Integration Challenges

1. Avro schemas

Amazon Athena



Engine Integration Challenges

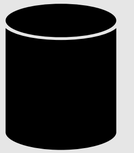


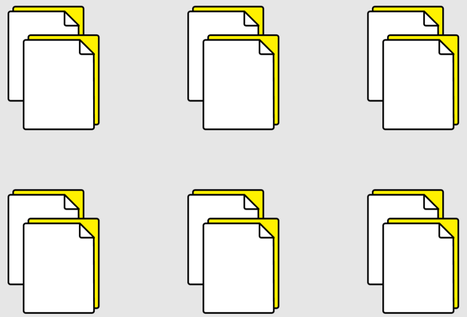
```
CREATE TABLE joined (...)
   PARTITIONED BY (day(joined_ts)) AS ...;
```


```
SELECT * FROM joined;
```

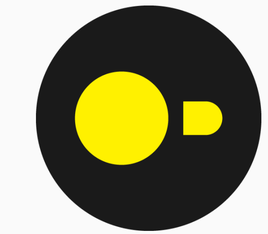
1. Avro schemas
2. Partitioning
3. Types

Iceberg Table





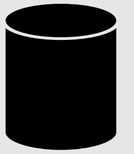
Engine Integration Challenges



```
CREATE TABLE joined (...)
  PARTITIONED BY (day(joined_ts)) AS ...;
```

```
SELECT * FROM joined;
```

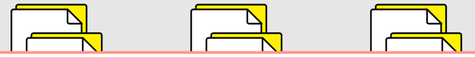
Iceberg Table



1. Avro schemas

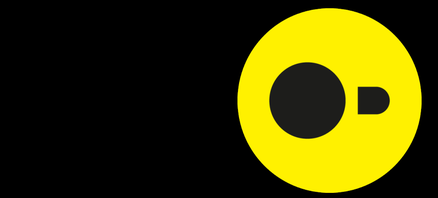
2. Partitioning

3. Types



```
*** py4j.protocol.Py4JJavaError: An error occurred while calling o71.showString.
: java.lang.ArrayStoreException:
org.apache.iceberg.shaded.org.apache.avro.generic.GenericData$Record
        at java.base/java.util.LinkedList.toArray(LinkedList.java:1114)
        at org.apache.iceberg.GenericManifestFile.set(GenericManifestFile.java:375)
        at org.apache.iceberg.GenericManifestFile.put(GenericManifestFile.java:387)
        at org.apache.iceberg.avro.ValueReaders$IndexedRecordReader.set(ValueReaders.java:738)
        at org.apache.iceberg.avro.ValueReaders$IndexedRecordReader.set(ValueReaders.java:706)
```

Engine Integration Challenges



2. Partitioning

Avro Record names in the Manifest List

DuckDB

```
{
 "name" : "partitions",
 "type" : [ "null", {
   "type" : "array",
   "items" : {
     "type" : "record",
     "name" : "partitions_record",
     "fields" : [ {
       "name" : "contains_null",
       "type" : "boolean",
       "field-id" : 509
     }, {
```

Spark

```
{
 "name" : "partitions",
 "type" : [ "null", {
   "type" : "array",
   "items" : {
     "type" : "record",
     "name" : "r508",
     "fields" : [ {
       "name" : "contains_null",
       "type" : "boolean",
       "field-id" : 509
     }, {
```

Engine Integration Challenges

2. Partitioning

Avro Record names in the Manifest List

```
*** py4j.protocol.Py4JJavaError: An error occurred while calling o71.showString.
: java.lang.ArrayStoreException:
org.apache.iceberg.shaded.org.apache.avro.generic.GenericData$Record
        at java.base/java.util.LinkedList.toArray(LinkedList.java:1114)
```

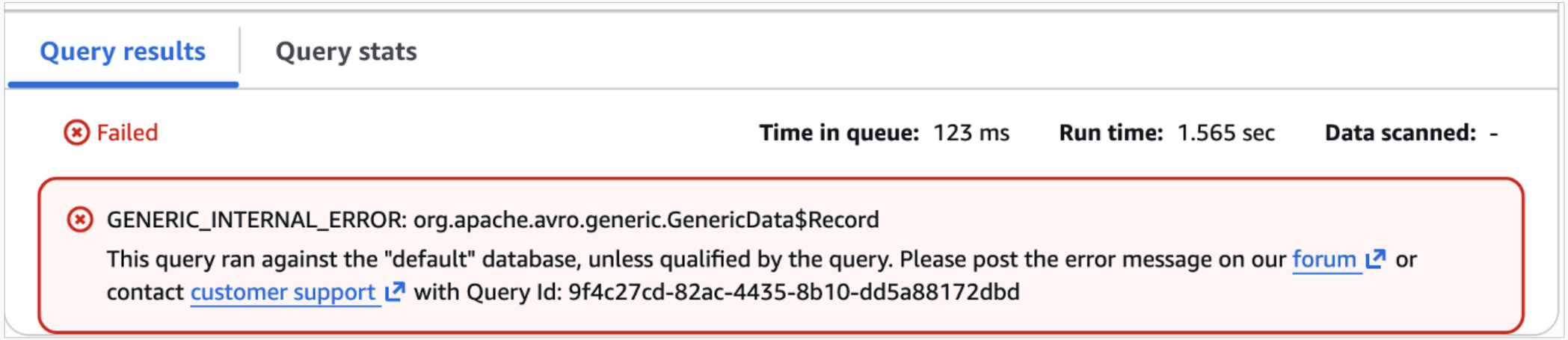
3.2	End of Life	0.13.0	1.4.3	iceberg-spark-runtime-3.2_2.12, iceberg-spark-runtime-3.2_2.13

Engine Integration Challenges

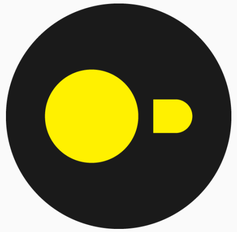
2. Partitioning

Avro Record names in the Manifest List

Amazon Athena



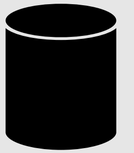
Engine Integration Challenges

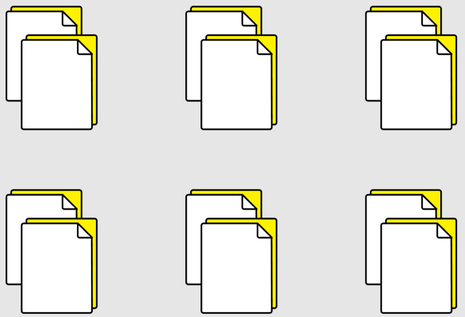


```
CREATE TABLE joined (...)
  PARTITIONED BY (day(joined_ts)) AS ...;
```

```
SELECT * FROM joined;
```

Iceberg Table





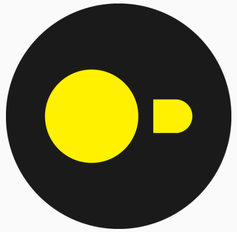
1. Avro schemas

2. Partitioning

3. Types

- Day partition types
- Lower and upper bounds
- Variant

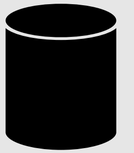
Engine Integration Challenges

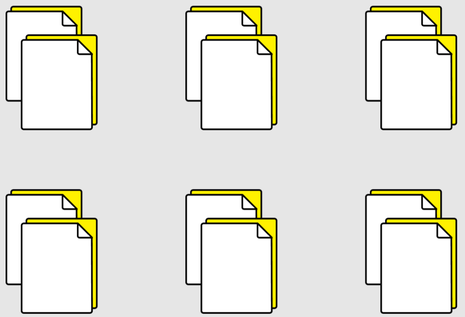


```
CREATE TABLE joined (...)
   PARTITIONED BY (day(joined_ts)) AS ...;
```


```
SELECT * FROM joined;
```

Iceberg Table





1. Avro schemas

2. Partitioning

3. Types

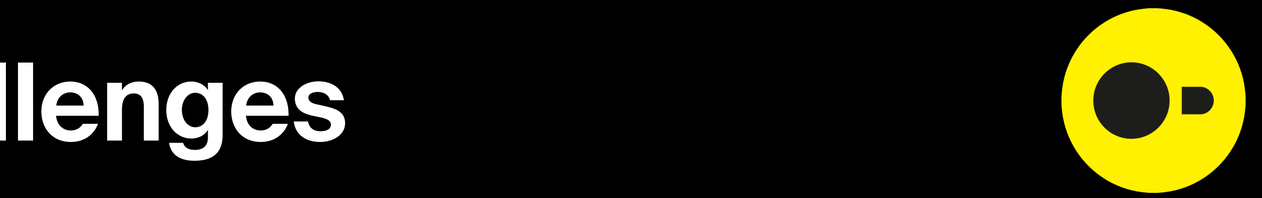
- Day partition types
- Lower and upper bounds
- Variant

Engine Integration Challenges

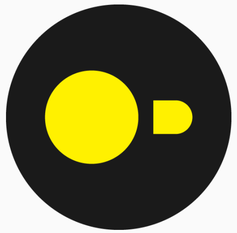
3. Types

- Day partition types
- Lower and upper bounds
    - Temporal types
    - Varchar
- Variant

Engine Integration Challenges



3. Types

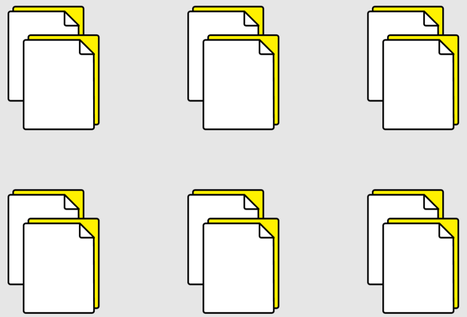


```
CREATE TABLE joined (...)
  PARTITIONED BY (day(joined_ts)) AS ...;
```

```
SELECT * FROM joined;
```

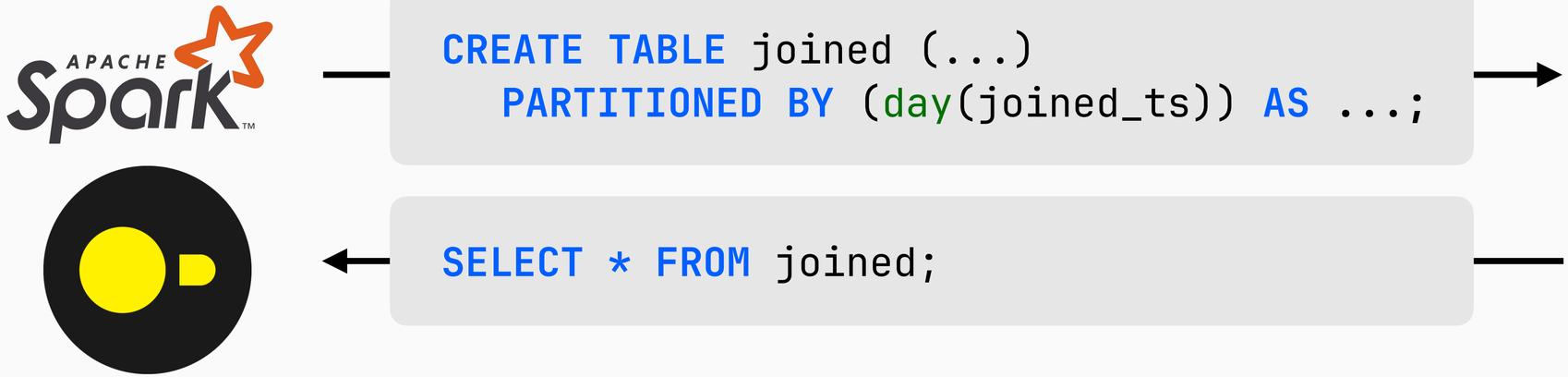
```
Conversion Error:
Unimplemented type for cast (DATE -> INTEGER)
```

Iceberg Table

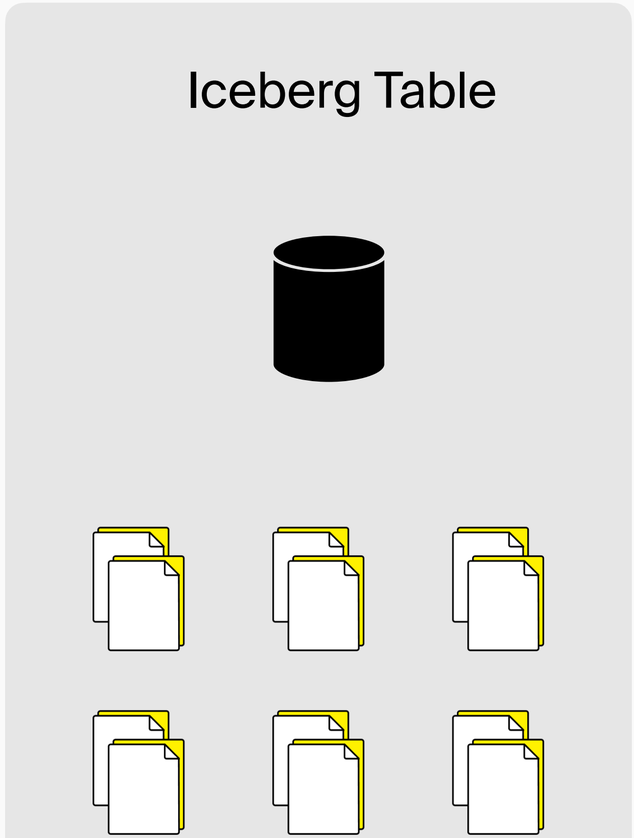


Engine Integration Challenges

3. Types



day	Extract a date or timestamp day, as days from 1970-01-01	`date`, `timestamp`, `timestamptz`, `timestamp_ns`, `timestamptz_ns`	`int`



Engine Integration Challenges

3. Types

`day`	Extract a date or timestamp day, as days from 1970-01-01	`date`, `timestamp`, `timestamptz`, `timestamp_ns`, `timestamptz_ns`	`int`

Partition transforma function day returns date but document describes int #14348

New issue

Open

jinchengchenghh opened on Oct 16, 2025

Apache Iceberg version

None

Assignees

No one assigned

Labels

bug

Engine Integration Challenges

3. Types

`day`	Extract a date or timestamp day, as days from 1970-01-01	`date`, `timestamp`, `timestamptz`, `timestamp_ns`, `timestamptz_ns`	`int`

```
$ duckdb

SELECT partition
FROM (SELECT unnest(data_file)
      FROM read_avro('snowflake/created/manifest.avro');
```

partition struct(event_date_day date)
{'event_date_day': 1970-01-01}
{'event_date_day': 2023-12-31}

Engine Integration Challenges

3. Types

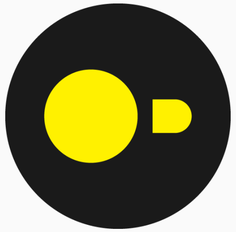
`day`	Extract a date or timestamp day, as days from 1970-01-01	`date`, `timestamp`, `timestamptz`, `timestamp_ns`, `timestamptz_ns`	`int`


```
$ duckdb

SELECT partition
FROM (SELECT unnest(data_file)
      FROM read_avro('pyiceberg/created/manifest.avro');
```

partition struct(event_date_day date)
{'event_date_day': 1970-01-01} {'event_date_day': 2023-12-31}

Engine Integration Challenges

3. Types



```
CREATE TABLE joined (customer_id INT, joined_ts TIMESTAMPTZ, aux VARCHAR)
PARTITIONED BY (day(joined_ts));

INSERT INTO joined VALUES (0, TIMESTAMPTZ 'infinity', 'test');
```

Failed **Time in queue:** 70 ms **Run time:** 1.477 sec **Data scanned:** -

ICEBERG_CURSOR_ERROR: Failed to read Parquet file: s3://dcf014ea-b53f-4157-sr1j3sp3fs4ze7tinwxbnq6wxgzcoeuc1b--table-s3/data/019d2aab-485f-72b7-8f92-b0d5705475bd.parquet

This query ran against the "default" database, unless qualified by the query. Please post the error message on our forum or contact customer support with Query Id: 3a407033-b315-4b7f-b4c9-0f11392c11db

Engine Integration Challenges

3. Types

```
CREATE TABLE joined (customer_id INT, joined_ts TIMESTAMPTZ, aux VARCHAR)
PARTITIONED BY (day(joined_ts));

INSERT INTO joined VALUES (0, TIMESTAMPTZ 'infinity', 'test');
```

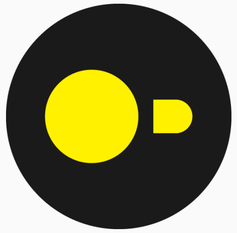
```
CREATE OR REPLACE ICEBERG TABLE joined
  EXTERNAL_VOLUME = 'iceberg_external_volume'
```

SQL execution internal error: Processing aborted due to error 300010:1258336960; incident 9368015.

Engine Integration Challenges

3. Types

```
INSERT INTO joined VALUES
  (0, '2026-01-01'::TIMESTAMPTZ, '江苏省太仓市经济开发区'),
  (1, '2026-01-02'::TIMESTAMPTZ, 'aux text 1'),
  (2, '2026-01-03'::TIMESTAMPTZ, 'aux text 2');
```



```
SELECT * FROM joined WHERE aux = 'aux text 1';
```

```
Invalid Input Error:
Invalid string encoding found in Parquet file: value
 "\xE6\xB1\x9F\xE8\x8B\x8F\xE7\x9C\x81\xE5\xA4\xAA\xE4\xBB\x93\xE5\xB8\x82\xE7
 \xBB\x8F\xE6\xB5\x8E\xE5\xBC\x80\xE5\x8F\x91\xE5\x8D" is not valid UTF8!
```

Engine Integration Challenges

3. Types

```
INSERT INTO variant_table VALUES
  (1,  {'name': 'alice', 'age': 30}::VARIANT);
```

```
SELECT * FROM variant_table;
```

```
Not implemented Error
Decimal with PhysicalType (INT16) not implemented for shredded Variant
```

Engine Integration Challenges

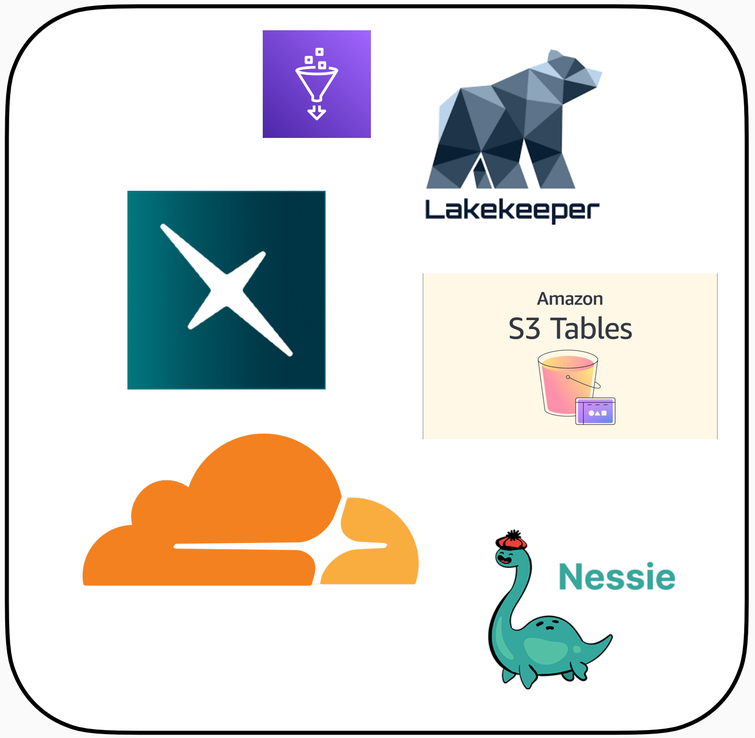
3. Types

```
INSERT INTO variant_table VALUES
  (1,  {'name': 'alice', 'age': 30}::VARIANT);
```


```
SELECT * FROM variant_table;
```

```
java.lang.IllegalArgumentException: Cannot find field name in metadata: age
```

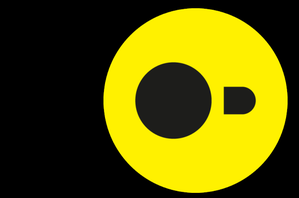
Iceberg Catalogs



Catalog Integration Challenges

1. /config responses
2. Type and format-version verification
3. Vended credential questions
    - Scope
    - Region

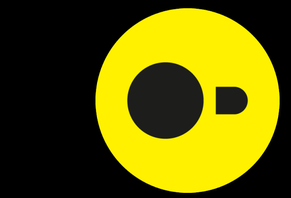
Catalog Integration Challenges: Config



Amazon S3 Tables

```
{
    "defaults": {
        "write.object-storage.partitioned-paths": "false",
        "s3.delete-enabled": "false",
        "io-impl": "org.apache.iceberg.aws.s3.S3FileIO",
        "write.object-storage.enabled": "true",
        "prefix": "arn%3Aaws%3As3tables%3Aeu-central-1...bucket%2Fathena-table-bucket-2",
        "rest-metrics-reporting-enabled": "false"
    },
    "overrides": {}
}
```

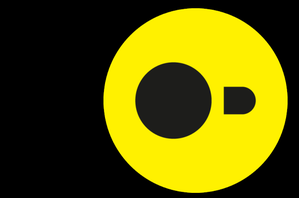
Catalog Integration Challenges: Config



AWS Glue

```
{
    "defaults": {
        "header.Content-Type": "application/x-amz-json-1.1",
        "rest.sigv4-enabled": "true",
        "rest-table-scan-enabled": "true",
        "prefix": "840140254803",
        "rest.signing-region": "us-east-1",
        "rest.signing-name": "glue",
        "token-refresh-enabled": "false",
        "rest-data-commit-enabled": "true"
    },
    "overrides": {
        "prefix": "catalogs/840140254803"
    }
}
```

Catalog Integration Challenges: Config



Snowflake

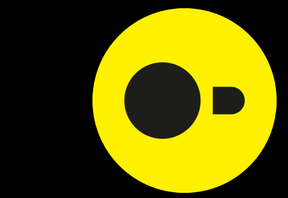
```
{
    "defaults": {
        "default-base-location": "gs://duckdblabs-iceberg-testing/…",
        "opencatalog.privatelink.enabled": "false"
    },
    "overrides": {
        "prefix": "GCS_catalog"
    },
    "endpoints": [
        "GET /v1/{prefix}/namespaces",
        "...",
        "POST /v1/{prefix}/views/rename"
    ]
}
```

Catalog Integration Challenges: Config

Cloudflare R2

```
{
    "overrides": {
        "prefix": "...",
        "s3.signer.uri": "https://catalog.cloudflarestorage.com/catalog/v1/..."
    },
    "defaults": {},
    "endpoints": [
        "GET /v1/config",
        "GET /v1/{prefix}/namespaces",
        "...",
        "POST /v1/{prefix}/views/rename"
    ]
}
```

Catalog Integration Challenges: Config



Google BigLake

```
{
    "overrides": {
        "catalog_credential_mode": "CREDENTIAL_MODE_VENDED_CREDENTIALS",
        "table-overrides.format-version": "2",
        "prefix": "projects/{account_id}/catalogs/biglake-public-nyc-taxi-iceberg",
        "biglake_service_account": "{email}"
    },
    "defaults": {
        "prefix": "projects/{account_id}/catalogs/biglake-public-nyc-taxi-iceberg",
        "table-defaults.format-version": "2"
    },
    "endpoints": [
        "GET /v1/{prefix}/namespaces",
         "...",
    ]
}
```

Catalog Integration Challenges: Config

Catalog	Prefix URL encoded	Prefix is separate components	Response includes endpoints
Amazon S3 Tables	✅	❌	❌
AWS Glue	❌	✅ / ❌	❌
Cloudflare R2	❌	❌	✅
Google BigLake	❌	✅	✅
Snowflake Open Catalog	❌	❌	✅

Catalog Integration Challenges: Config

Catalog	Prefix URL encoded	
Amazon S3 Tables	✅	`arn%3Aaws%3As3tables%3A{{region}}%3A{{account}}%3Abucket%2F{{Bucket}}`
AWS Glue	❌ / ✅	`catalogs/840140254803`
Cloudflare R2	❌	`a82557a0-2136-11f0-849b-9b56e804fbc5`
Google BigLake	❌	`projects/{account_id}/catalogs/biglake-public-nyc-taxi-iceberg`
Snowflake Open Catalog	❌	`GCS_catalog`

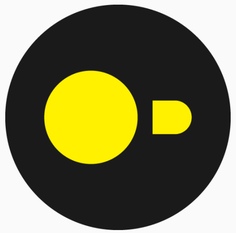
Catalog Integration Challenges: Config

Amazon S3 Tables & Glue do not provide a list of supported endpoints

1. `transactions/commit` is a default endpoint
2. Hitting this endpoint results in??

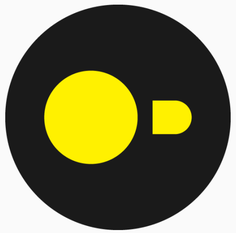
```
500: <UnknownOperationException/>
```

Catalog Integration Challenges: Format-Version & Type Support



```
CREATE TABLE default.timestampns_table (a TIMESTAMPNS)
WITH ('format-version' = 2);
```

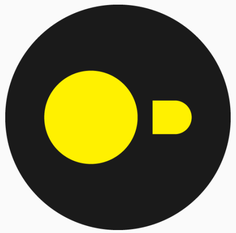
Catalog Integration Challenges: Format-Version & Type Support



```
CREATE TABLE default.timestampns_table (a TIMESTAMPNS)
WITH ('format-version' = 2);
```

Catalog	Throws error?
Amazon S3 Tables	✅
AWS Glue	✅
Cloudflare R2	❌
Nessie	✅
Polaris	✅

Catalog Integration Challenges: Format-Version & Type Support



```sql
CREATE TABLE default.v3_table (a INT)
WITH ('format-version' = 3);
```

Catalog	Throws error?
Amazon S3 Tables	❌
AWS Glue	❌
Cloudflare R2	❌
Nessie*	❌
Polaris	❌

*Nessie does not support `'format-version' = 3` yet

Nessie throws an error when creating a table with a V3 type in a V2 table, but not when creating a V3 table!

Catalog Integration Challenges:
Region & Prefix

1. Snowflake with S3 storage
   No "client.region" information returned!
2. User must fill in region themselves

```
...
  "storage-credentials" : [ {
     "prefix" : "s3://{bucket}/...",
     "config" : {
       "s3.access-key-id" : "...",
       "s3.secret-access-key" : "...",
       "s3.session-token" : "...",
       "expiration-time" : "1775144782000"
     }
...
```

```
D ATTACH '{
    TYPE ic
    ENDPOIN
    DEFAULT
  );
```

AWS Configurations

The following configurations should be respected when working with tables stored in AWS S3

- `client.region` : region to configure client for making requests to AWS
- `s3.access-key-id` : id for credentials that provide access to the data in S3
- `s3.secret-access-key` : secret for credentials that provide access to data in S3
- `s3.session-token` : if present, this value should be used for as the session token

Catalog Integration Challenges: Region & Prefix

1. Cloudflare R2 Catalog returns a prefix of "/" , which does not match data location, requiring specific logic

```
...
  "config" : {
   "s3.path-style-access" : "true",
   "region" : "auto",
   "s3.endpoint" : "https://..",
   "s3.remote-signing-enabled" : "false",
   "client.region" : "auto",
   "s3.region" : "auto",
   "s3.access-key-id" : "...",
   "s3.secret-access-key" : ".."
},
"storage-credentials" : [ {
   "prefix" : "/",
   "config" : {
     "s3.access-key-id" : "...",
     "s3.secret-access-key" : "..."
   }
} ]
...
```

Testing DuckDB-Iceberg

Testing DuckDB-Iceberg

1. Catalog testing
   - Nessie
   - Polaris
   - Lakekeeper
   - Rest-Fixture
   - Cloud Catalogs
2. Engine testing
   - Spark (multiple versions)
   - Pylceberg

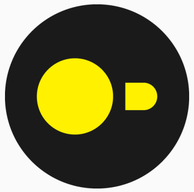
Testing DuckDB-Iceberg

Catalog

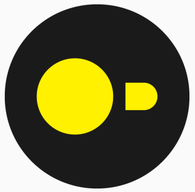
apache/iceberg-rest-fixture

Written by

Multiple versions



Update/Delete/Insert



Read by

Multiple versions

User Generated Tables
from GH Issues

Testing DuckDB-Iceberg

Cloud Catalogs

Written by

Update/Delete/Insert

Read by

DuckDB-Iceberg: Takeaways

Engines

1. The Spec is good 98% of the time
2. When implementing an Iceberg engine with writes, be careful with avro files
3. Verify upper and lower bound validity
4. Verify variant compatibility

Catalogs

1. Verify format-version with format-version supported types
2. Verify compatibility with vended credentials spec
3. Prefix encoding will get more complicated and will need spec definitions.
4. Published Rest-API versioning would be helpful

DuckDB-Iceberg: Moving Forward

DuckDB v1.5.2 - April 13, 2026

1. Geometry support
2. Update & Delete from Partitioned Tables
3. Truncate & Bucket Partition Support
4. Alter Table Support

ICEBERG

SUMMIT 2026