
2 Hours to Data Innovation with Cloudera Data Warehouse

October 11th, 2023



Cloudera Data Warehouse Workshop

Self Service Hands On Lab

Index:

Business Use Case	3
Introduction to the Lab Data	3
Lab Setup	5
Get Started - Log into CDP	5
Lab 1 - Business Analyst: Explore Completed Dashboard	7
Lab 2 - Business Analyst: Self Service (in CDW)	9
Lab 3 - Administrator: “Productionalize” the Open Data Lakehouse (Optional Bonus Material)	16
Optional Lab 1 - CDW Tour	31
Optional Lab 2 - Data Security & Governance	32

Business Use Case

The following Labs will take you through a Self-Service Analytics example. We will use a fictitious company named, Special Marketing Group (SMG). A little company background on **SMG is that they are a marketing data aggregator company** that is currently working closely **with Duty-Free Shops in many airports throughout the U.S. and 20+ countries**. **SMG is working with the Duty-Free Shop owners on an initiative to drive additional revenue into the shops**. The **duty-free shop owners** want to partner with specific airlines to offer **discounts through the airlines** to their **passengers who spend more time in an international terminal** than warranted for their flight.

SMG believes that it can help drive more revenue by driving more traffic to the duty-free shops. SMG realizes that they have **some questions that need to be answered**, such as:

- Which airlines have the most passengers who have long layovers built into their tickets?
- Which airlines have the most passengers who have delayed international legs in their itinerary causing an extended layover in their flight itinerary?
- Which airlines have the most passengers who are displaced by a missed international connection, caused by a delay in their previous leg?

By answering these burning business questions SMG will be able to work with duty-free shop owners to develop campaigns that they can run to drive additional revenue. Since, SMG also does quite a bit of business with many of the major airlines and believes that the analytics used to drive more revenue for duty-free shops could also help the airlines improve customer satisfaction during extended wait times, and duty-free shops can increase sales.

Our Labs today will walk through the steps to learn how SMG uses the power of Cloudera's Data Warehouse Data Service to leverage the Self-Service capabilities to build out this new solution. If/when the Business Analyst proves that this new data will work to solve the need, they can turn it over to the admin team to "productionalize" it.

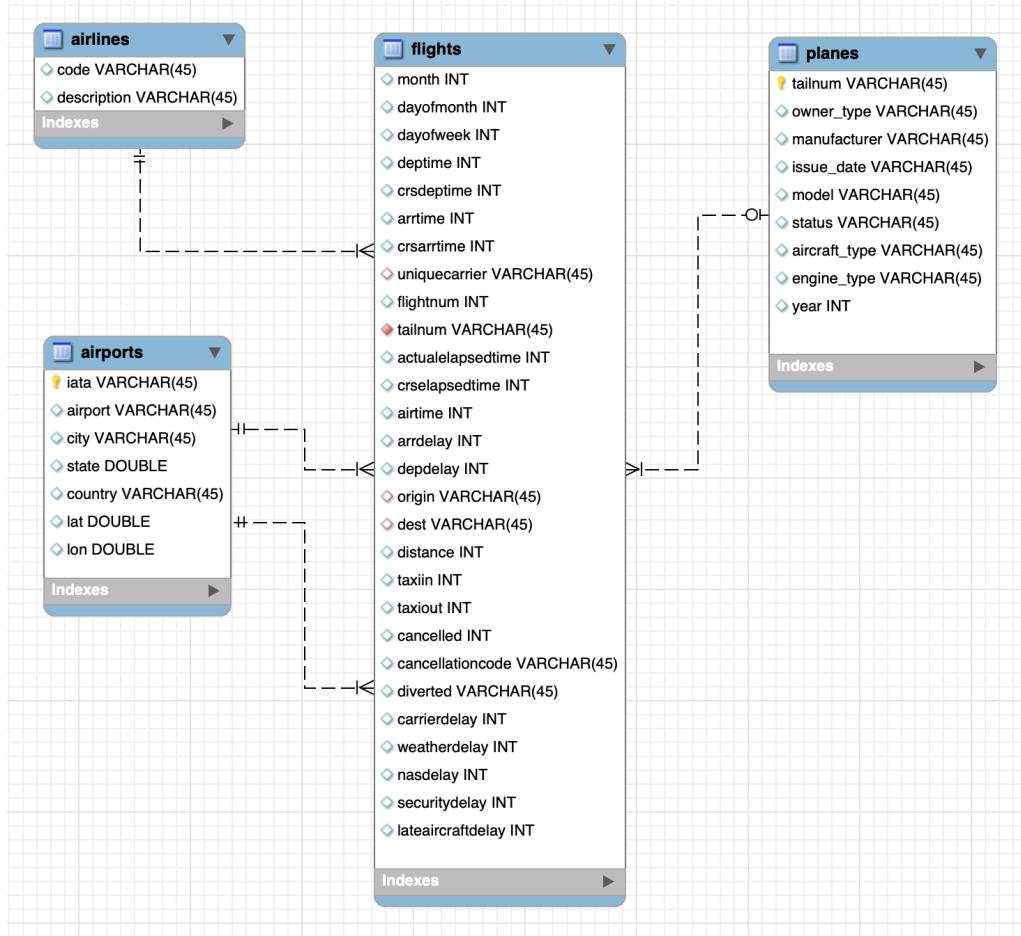
Introduction to the Lab Data

For the following Workshop Hands On Labs, we will dive into this Scenario to show Cloudera Data Warehouse (CDW) is used to enable SMG to gain a competitive advantage - and at the same time, it highlights the performance and automation capabilities that help ensure performance is maintained while controlling costs.

The Hands On Labs will take you through how to use the Cloudera Data Warehouse service to quickly explore raw data, create curated versions of the data for simple reporting and dashboarding, and then scale up usage of the curated data by exposing it to more users.

ER - Diagram of the demo Data Lakehouse:

- **Fact table:** flights (86M rows)
- **Dimension tables:** airlines (1.5k rows), airports (3.3k rows) and planes (5k rows)



Self Service data file - Passenger Ticket manifest:

- This is a CSV file with 1,000 records
- Each record represents a unique Passenger Ticket with 2 Flight Legs
- It contains the following schema

ticketnumber BIGINT
 leg1flightnum BIGINT
 leg1uniquecarrier STRING
 leg1origin STRING
 leg1dest STRING
 leg1month BIGINT
 leg1dayofmonth BIGINT
 leg1dayofweek BIGINT
 leg1deptime BIGINT
 leg1arrrtime BIGINT
 leg2flightnum BIGINT

CLOUDERA

leg2uniquecarrier STRING
leg2origin STRING
leg2dest STRING
leg2month BIGINT
leg2dayofmonth BIGINT
leg2dayofweek BIGINT
leg2deptime BIGINT
leg2arrtime BIGINT

Lab Setup

Workshop Attendees will be provided with a Login. Please enter your details below for reference throughout these lab exercises.

- URL: <https://rb.gy/wx75l>
 - Use Incognito Browser window
- Login:
 - **User:** _____
 - **Password:** _____

In the labs when you see:

- \${user_id} or <user_id> - this will indicate to use your User (Workload User) provided for you to login to CDP

Zoom Link:

<https://cloudera.zoom.us/j/94846291323>

Get Started - Log into CDP

GET STARTED

- Preparatory work for us to get ourselves oriented in CDW and ready to build out the use case
- Login to CDP
- Learn a little about Cloudera Data Warehouse (CDW) Data Service

In this Lab you will login to CDP and complete your user setup.

1. Log in to CDP - open a browser and open the URL from the Lab Setup section
 - When prompted use the Login details from the Lab Setup section for the User/PW
2. On the HOME page of CDP you will see all of the Services available to you

CLOUDERA



- Multi-function analytics - Data Services; today we'll just focus on the Cloudera Data Warehouse Data Service
- There are other Data Services that usually are part of an Analytic use case
 - Data Flow - for data ingestion needs
 - Data Engineering - for ELT/ETL, transformations, data wrangling, etc.
 - Machine Learning - for Data Science teams to collaboratively build and productionalize ML/AI applications and models for the Enterprise
- However, Cloudera provides other services such as Data Catalog, Replication Manager, Workload Manager, and Management Console provide continuity and functionality throughout the platform.



Note: The platform removes the necessity to spend unnecessarily on integration tax, where other solutions combine various pieces together in the ability to provide continuity and functionality throughout an end-to-end use case but this is difficult because it introduces many moving parts.

Lab 1 - Business Analyst: Explore Completed Dashboard

SEE THE END RESULT (WHAT IS CREATED) (LAB 1)

- Explore the created Dashboard after it has been “productionalized”
 - Analyst first validates that the combination of data answers questions & builds initial Dashboard to start gaining insights on the data
 - Administrator picks up and formalizes the data ingestion for the new data, incorporates it into the Data Lakehouse, adds security, & completes the Dashboard

In this Lab you will explore the Dashboard that will be created as the result of the Self-Service Labs you will complete shortly.

3. Open Cloudera Data Visualization, which is part of the Cloudera Data Warehouse Data Service
 - First let's open Cloudera Data Warehouse (CDW) -
 - Click on the Cloudera Data Warehouse (CDW) tile



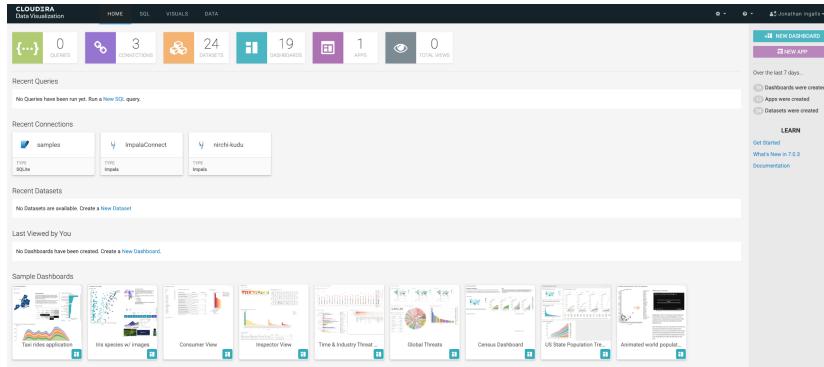
- We'll come back to this screen in Lab 2 for more details

A screenshot of the Cloudera Data Warehouse service overview. The left sidebar shows 'Data Warehouse' selected. The main area has a 'Welcome to Cloudera Data Warehouse Service' message and three main sections: 'Get Started With Data Warehouse' (with links to Start Guide, CDP Patterns, Explore Use Cases, Community Forum, Documentation), 'Create' (with a 'See More' button), and 'Query and Visualize Data' (with a 'See More' button). At the bottom, there's a 'Resources and Downloads' section with a 'See More' button.

- Now let's open Cloudera Data Visualization (CDV) - to explore the finished product
 - Open Cloudera Data Visualization (CDV or Data Viz)
 - On the left navigation panel click “Data Visualization” - this will open a new browser tab
 - On the row with **airlines-dataviz-#**, to the right, click the Data VIZ button

A screenshot of the Cloudera Data Visualization home page. The left sidebar shows 'Data Visualization' selected. The main area has a search bar and a table with one row. The table columns are NAME, DATA VISUALIZATION ID, Environment ID, VERSION, CPU, MEMORY, UPTIME, and CREATED BY. The single row shows 'airlines-dataviz' with 'viz-1696817984-vtvj' as the ID, 'env-4gb8mn' as the environment, '7.1.3-b36' as the version, '4' as the CPU, '16 GB' as memory, '3 minutes' as uptime, and 'graman' as the creator. There are 'Data VIZ' and 'More' buttons to the right of the row.

- If you see the “What’s New” page, you can read it, or click on the GOT IT button
- Cloudera Data Visualization (CDV) Home page



- There are 4 areas of CDV - HOME, SQL, VISUALS, DATA - these are the tabs at the top of the screen in the black bar to the right of the Cloudera Data Visualization banner
 - HOME - this is the starting point; it shows some statistics at the top, followed by some quick access details to recent content - Queries, Connections, Datasets, and Dashboards
 - SQL - allows you to manually build queries against data to perform quick discovery against the data. Below is an example of a query that was built and Run

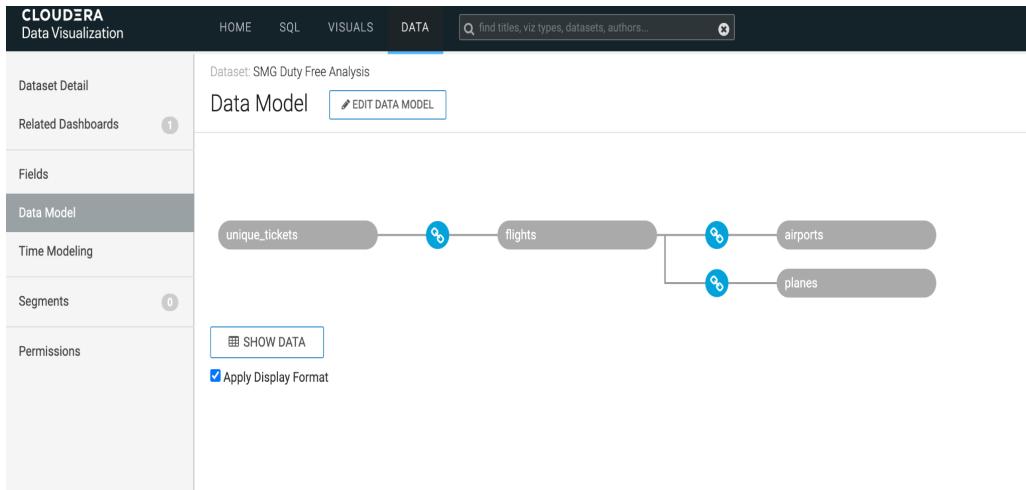
The screenshot shows the SQL editor in the Cloudera Data Visualization interface. The editor has a 'Data Connection' dropdown set to 'samples'. The 'SQL' tab is selected. A query is being run: 'select * from main.cereals limit 100'. The results show a table with 100 rows of cereal data. The columns include: cereal_name, manufacturer_code, old_or_new, calories, protein_grams, fat_grams, sodium_mg, dietary_fiber_grams, complex_carbohydrates_grams, sugars_grams, display_shelf, potassium_mg, and vits.

cereal_name	manufacturer_code	old_or_new	calories	protein_grams	fat_grams	sodium_mg	dietary_fiber_grams	complex_carbohydrates_grams	sugars_grams	display_shelf	potassium_mg	vits
100%, Bran	N	C	70	4	1	130	10	5	6	3	280	25
100%, Bran Bran	O	C	120	3	5	15	2	8	8	3	135	0
All Bran	K	C	70	4	1	260	9	7	5	3	320	25
All Bran with Extra Fiber	K	C	50	4	0	140	14	8	0	3	330	25
Almond Delight	R	C	110	2	2	250	1	14	8	3	-1	25
Apple, Cinnamon, Cereals	G	C	110	2	2	180			10	1	70	25
Apple Jacks	K	C	110	2	0	125	1	11	14	2	30	25
Banana	O	C	180	3	2	210	2	18	8	3	100	25
Bran Chex	R	C	90	2	1	220	4	15	6	1	125	25

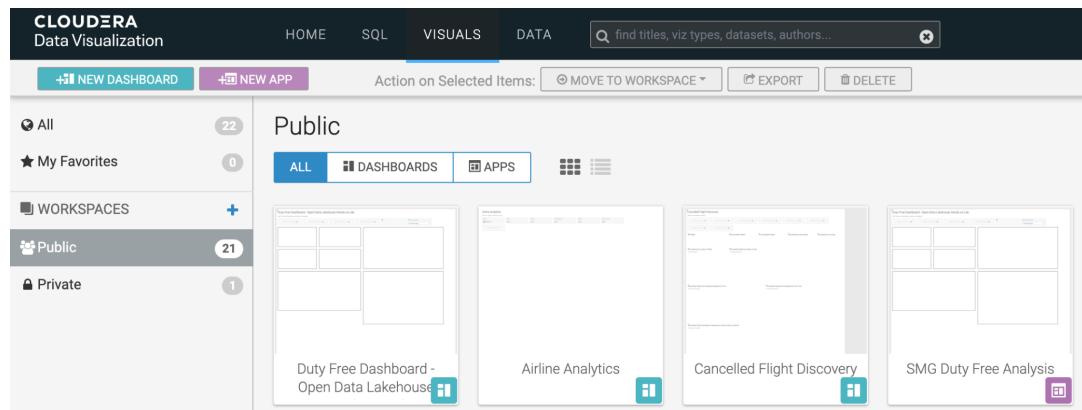
- VISUALS - an area for viewing/building/modifying visuals, dashboards, and applications

The screenshot shows the VISUALS section of the Cloudera Data Visualization interface. It displays a grid of visualizations including a histogram, a map of NYC, a store details card, and a cereal comparisons chart. Navigation tabs at the top include 'DASHBOARDS', 'VISUALS', 'APPS', and 'ALL'. On the left, there are sections for 'All' (17 items), 'My Favorites' (empty), 'WORKSPACES' (Public), and a search bar.

- DATA - interface for access to datasets (aka: metadata model) image below, connections, and the Connection Explorer. Explore the Data model for SMG Duty Free Analysis



- Click on the VISUALS tab at the top of the screen (in black banner)



Lab 2 - Business Analyst: Self Service (in CDW)

BUSINESS ANALYST - SELF SERVICE TO VALIDATE NEW DATA ANSWERS QUESTIONS (LAB 2)

- Upload new data - passenger ticket manifest(**already preloaded data**)
 - See if new data can help to answer questions prior to undertaking a complete project
 - See how to upload data
- Answer burning business questions to see if the new passenger data**
 - Combine uploaded data with existing data warehouse
 - Start digging into the data - running a few queries to see if the new dataset can help with answering the burning questions
 - Visualize the data - modify existing Dashboard to add new content with the data just added

In this Lab you will take advantage of the **Self Service capabilities within CDW** to upload a data file and combine it with an existing Data Lakehouse. You will then perform analytics (SQL & Visualizations) on the combined data to ensure that the burning business questions can first be answered before “productionalizing” the new data source.

4. Navigate back to **CDW Overview**

CLOUDERA

- Click on the browser tab where CDW is open -
- Click on Overview in the left navigation

The screenshot shows the Cloudera Data Warehouse (CDW) interface. On the left, there's a dark sidebar with the CDW logo and the text "Data Warehouse". Below it are four menu items: "Overview", "Database Catalogs", "Virtual Warehouses", and "Data Visualization", with "Data Visualization" being the active one. The main content area is titled "Data Visualization" and contains a search bar labeled "Search" and a list of "NAME"s. One item, "airlines-dataviz-1", is highlighted with a green checkmark.

5. The following is a quick explanation of the CDW User Experience so that you have a basic knowledge of the data service. The **CDW Data Service allows you to create independent, self-service data warehouses and data marts that autoscale up and down to meet your varying workload demands**. It provides isolated compute instances for each data warehouse/mart, has built-in automatic optimization, and ultimately enables you to meet SLAs - at the same time save costs.

The screenshot shows the Cloudera Data Warehouse (CDW) interface. On the left, there's a dark sidebar with the CDW logo and the text "Data Warehouse". Below it are four menu items: "Overview", "Database Catalogs", "Virtual Warehouses", and "Data Visualization", with "Overview" being the active one. The main content area is titled "Welcome to Cloudera Data Warehouse Service" and contains a "Get Started With Data Warehouse" section with a "Start Guide", "CDP Patterns", "Explore Use Cases", "Community Forum", and "Documentation". Below this are three main sections: "Create" (with a "See More" button), "Query and Visualize Data" (with a "See More" button), and "Resources and Downloads" (with a "See More" button). At the bottom, there are tabs for "Environments (1)", "Database Catalogs (1)", and "Virtual Warehouses (2)". The "Virtual Warehouses (2)" tab is selected, showing a table with two entries:

Status	Name	Type	Version	CPU	Nodes	Uptime	Actions
Good Health	airlines-impala-vw	Impala Unified Analytics	2023.0.15....	83	<div style="width: 83%;"></div>	4 days	Suspend
Good Health	airlines-hive-vw	Hive Compactor Unified Analytics	2023.0.15....	36	<div style="width: 36%;"></div>	4 days	Suspend

- There are 2 areas on this screen - **Database Catalogs (DBC)**, and **Virtual Warehouses (VW)**, for today's lab, we will just concentrate on Virtual Warehouses or VW for short
 - **Database Catalogs (DBC) - is a logical collection of table and view metadata, security permissions, and other information.** As you create databases, tables, views, etc. in the Data Warehouse, it collects the metadata. When you activate an Environment for CDW, the CDW service will automatically create a default DBC associated with this Environment.
 - DBCs are in the middle column

Status	Name	Virtual Warehouses	Version	Uptime	Actions
Good Health	evolve-sydney23-default warehouse-1696479234-69lw evolve-sydney	2	2023.0.15.1-2	4 days	Suspend

- **Virtual Warehouses (VW) - is a set of compute resources running in Kubernetes to execute the queries.** This is something that can allow for Self Service capabilities or can be controlled by Administrators. A VW binds compute and storage to execute secured queries that access tables and views of your data via the DBC. A VW can scale automatically to ensure performance even with high concurrency, and can auto-scale down in situations of low demand. Tools that access data via JDBC/ODBC can connect to VWs to run queries

Status	Name	Type	Version	CPU	Nodes	Apps	Uptime	Actions
Stopped	airlines-hive-vw compute-1696479743-fpmf evolve-sydney23-default evolve-sydney	Hive Compactor Unified Analytics	2023.0.15.1-2	36	<div style="width: 50%;"> </div>	HUE	4 days	Start

- See options available for a VW - click on the button on the bottom right corner of tile **airlines-hive-vw-#**

The list of options will vary slightly depending on whether this is a Hive or Impala VW. This is also where you would go to get the JDBC URL and Driver to use to connect your Business Intelligence software to this VW, and when a new version is released you could Upgrade this VW to the latest release without having to upgrade all VWs at the same time.

- **The *airlines-hive-vw-#*** - this will highlight the Environment and DBCs that are associated with this VW.
 - These tiles will display information about current workload on a VW. Below you will see items indicating when a VW is idle and has auto scaled so nothing is running for this VW; and where the VW needed to auto-scale up to handle incoming SQL requests handling concurrency

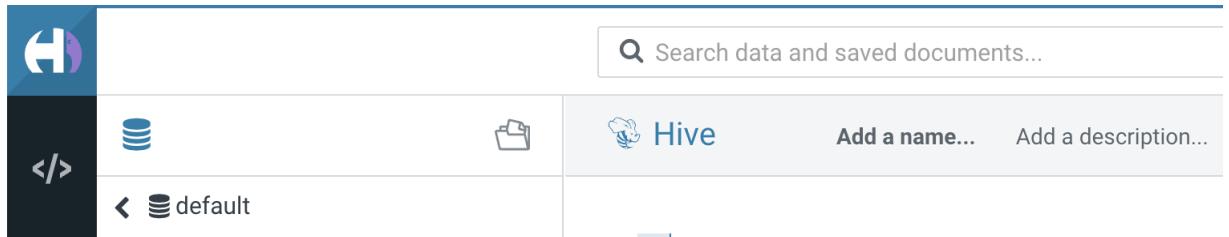
Status	Name	Type	Version	CPU	Nodes	Apps	Uptime	Actions
Good Health	airlines-hive-vw compute-1696479743-fpmf evolve-sydney23-default evolve-sydney	Hive Compactor Unified Analytics	2023.0.15.1-2	36	<div style="width: 50%;"> </div>	HUE	4 days	Suspend

- **On the *airlines-hive-vw-#* tile** - In the upper right corner click on the HUE button to enter into the SQL Editor



6. Explore the Data Lakehouse

- You will be using HUE for the [airlines-hive-vw-# Virtual Warehouse](#) until you get to Step 13



- Copy & paste the following SQL into the query Editor window

```
-- Run prior to importing Passenger Tickets; to ensure correct access and that everything is good to go
-- Query to find all international flights: flights where destination airport country is not the same as origin airport country
SELECT DISTINCT
    flightnum,
    uniquecarrier,
    origin,
    dest,
    month,
    dayofmonth,
    `dayofweek`
FROM
    airlines.flights f
JOIN airlines.airports oa ON f.origin = oa.iata
JOIN airlines.airports da ON oa.country <> da.country
WHERE f.dest = da.iata
ORDER BY
    month ASC, dayofmonth ASC
;
```

```

1 -- Run prior to importing Passenger Tickets; to
2 -- Query to find all international flights: fli
3 SELECT DISTINCT
4   flightnum,
5   uniquecarrier,
6   origin,
7   dest,
8   month,
9   dayofmonth,
10  "dayofweek"
11 FROM
12  airlines.flights f
13 JOIN airlines.airports oa
14  ON f.origin = oa.iata
15 JOIN airlines.airports da
16  ON oa.country <> da.country
17 WHERE
18  f.dest = da.iata
19 ORDER BY
20  month ASC,
21  dayofmonth ASC
22;

```

- Click on the ➤ to the bottom left of the SQL window to run this SQL command to test and ensure you have access to the Data Lakehouse
 - You should see something similar to the following in the Results tab

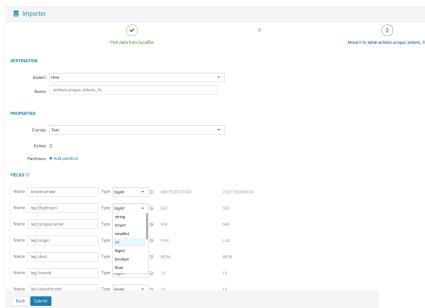
flightnum	uniquecarrier	origin	dest	month	dayofmonth	dayofweek
1	XE	BTR	IAH	1	1	2
2	DL	BTR	ATL	1	1	4
3	XE	IAH	BTR	1	1	4
4	MQ	DFW	BTR	1	1	2
5	DL	ATL	BTR	1	1	4
6	XE	IAH	BTR	1	1	4
7	N	ATL	BTR	1	1	7

- This validates that you have the correct permissions via the SDX security to access the Data Lakehouse

7. Upload Passenger Ticket data file - already completed for you and shown as part of the main presentation

- Below are some images that highlight the 3 steps to upload this file: 1) Open the Importer, 2) Pick a file & options for the file upload, 3) Name the table, specify table options, and provide the table metadata (column names, data types, etc.) (NOT FOR LAB EXECUTION)

FlightNumber	LegFlightNum	LegUniqueCarrier	LegFlight	LegIndex	LegLength	LegIsEnd	LegIsEndFlight	LegIsEndMonth	LegIsEndWeek	LegIsEndYear	LegJourneyTime
9G101000000000000000	900	NW	900	900	100	10	2	3	500	1407	1407
230115555414	550	NW	LAX	550	10	10	2	3	550	1517	1517
419130170956	202	NW	SAT	550	10	10	2	3	550	731	731
745545450494	1024	NW	M0	550	10	10	2	3	550	727	727



3)

- Since the data is already uploaded we will use the table **airlines.unique_tickets**

8. Explore the Data Lakehouse and Passenger Tickets (unique_tickets) data to see if it can answer the burning questions. The Data Analyst can use a SQL Editor to perform this task by executing queries against this data.

- Delete the current query from the SQL Window
- Copy & paste the following SQL into the SQL Editor window

```
-- Run after Uploading the Passenger Tickets data to see if we can answer the "burning
questions" to support Duty Free Stores
-- Number of passengers on the airline that have long, planned layovers for a flight (good
target to send promotion to)
SELECT
    a.leg1uniquecarrier as carrier,
    count(a.leg1uniquecarrier) as passengers
FROM
    airlines.unique_tickets a
where
    a.leg2deptime - a.leg1arrtime > 90
group by
    a.leg1uniquecarrier
;
```

- Execute the Query by clicking on the ▶ button

- Review the output to see the number of passenger with a long planned layover (> 90 minutes) by Airline Carrier - one of the questions we wanted to answer

Hive Add a name... Add a description...

```
1 -- Run after Uploading the Passenger Tickets data to see if we can an
2 -- Number of passengers on the airline that have long, planned layove
3 SELECT
4     a.leg1uniquecarrier as carrier,
5     count(a.leg1uniquecarrier) as passengers
6 FROM
7     airlines.unique_tickets_1k a
8 where
9     a.leg2deptime - a.leg1arrtime>90
10 group by
11     a.leg1uniquecarrier
12;
13
```

INFO : RAW_INPUT_SPLITS_Map_1: 1
INFO : savedToCache: true
INFO : Completed executing command(queryId=hive_20230615133755_bad51879-dad2
INFO : OK

Query History Saved Queries Results (3)

	carrier	passengers
1	EV	315
2	NW	617
3	MQ	1

- Delete the current query from the SQL Window
- Copy & paste the following SQL into the SQL Editor window

```
-- Number of passengers on airlines that have elongated layovers for a flight caused by
delayed connection (potential customer satisfaction issue)
SELECT
    a.leg1uniquecarrier as carrier,
    count(a.leg1uniquecarrier) as passengers
FROM
    airlines.unique_tickets a
JOIN airlines.flights o
    ON a.leg1flightnum = o.flightnum
    AND a.leg1uniquecarrier = o.uniquecarrier
    AND a.leg1origin = o.origin
    AND a.leg1dest = o.dest
    AND a.leg1month = o.month
    AND a.leg1dayofmonth = o.dayofmonth
    AND a.leg1dayofweek = o.`dayofweek`
JOIN airlines.flights d
    ON a.leg2flightnum = d.flightnum
    AND a.leg2uniquecarrier = d.uniquecarrier
    AND a.leg2origin = d.origin
    AND a.leg2dest = d.dest
    AND a.leg2month = d.month
    AND a.leg2dayofmonth = d.dayofmonth
    AND a.leg2dayofweek = d.`dayofweek`
WHERE o.depdelay > 60
group by
    a.leg1uniquecarrier
;
```

- Execute the Query by clicking on the ➤ button
- Review the output to see that we can answer another burning question

The screenshot shows the Cloudera Manager Hive interface. At the top, there's a search bar with 'Hive' selected and fields for 'Add a name...' and 'Add a description...'. Below the search bar is the query code:

```
1 -- Number of passengers on airlines that have elongated layovers for a flight caused by
2 SELECT
3     a.leg1uniquecarrier as carrier,
4     count(a.leg1uniquecarrier) as passengers
5 FROM
6     airlines.unique_tickets a
7 JOIN airlines.flights o
8     ON a.leg1flightnum = o.flightnum
9     AND a.leg1uniquecarrier = o.uniquecarrier
10    AND a.leg1origin = o.origin
11    AND a.leg1dest = o.dest
12    AND a.leg1month = o.month
13    AND a.leg1dayofmonth = o.dayofmonth
14    AND a.leg1dayofweek = o.`dayofweek`
15 JOIN airlines.flights d
16     ON a.leg2flightnum = d.flightnum
17     AND a.leg2uniquecarrier = d.uniquecarrier
18     AND a.leg2origin = d.origin
19     AND a.leg2dest = d.dest
20     AND a.leg2month = d.month
21     AND a.leg2dayofmonth = d.dayofmonth
22     AND a.leg2dayofweek = d.`dayofweek`
```

A large green arrow button labeled '▶' is positioned between lines 21 and 22. Below the query editor, the command history shows:

```
INFO : RAW_INPUT_ORIGINAL_MAP_1: 24
INFO : RAW_INPUT_SPLITS_Map_2: 1
INFO : RAW_INPUT_SPLITS_Map_3: 21
INFO : Completed executing command(queryId=hive_20230615134217_fab7bc80-bd30)
INFO : OK
```

The results section shows a table with two columns: 'carrier' and 'passengers'. The data is:

carrier	passengers
1 MQ	1
2 EV	31
3 NW	6

- Now that we've validated that we can answer the business questions, it's time to visualize the data to see the insights we can gain from combining this data

Lab 3 - Administrator: “Productionalize” the Open Data Lakehouse

ADMINISTRATOR - “PRODUCTIONALIZE” INTO OPEN DATA LAKEHOUSE (LAB 3)

- See “how the sausage was made” - how to take advantage of Apache Iceberg from end to end to deliver a modern Open Data Lakehouse solution
 - Migrate existing tables to Iceberg Table Format
 - Create a new Iceberg table
 - Load data
 - Some Key Features of Iceberg
 - Partition evolution
 - Time Travel
- Monitor the Virtual Warehouses (compute) and watch as it scales up and down, suspends, etc.
- Security & Governance before launching to the masses

In this Lab you will be an Administrator and will use CDW to create and build an Open Data Lakehouse and take advantage of some of the key features with this approach.

Execute the following SQL/DDL/DML statements.

9. Stay in HUE for the CDW **Hive** Virtual Warehouse - **airlines-hive-vw-#**

- There should be an open CDW Browser tab, open the  browser tab
- Create a user Database to store all of the tables you will create in the following lab steps
 - In the SQL Editor window create a database for the remaining lab exercises, execute the following.
 - Delete the current query from the SQL Window
 - Copy & paste the SQL below

```
CREATE DATABASE ${user_id}_airlines;
```

- The “\${...}” notation is a hint to HUE to create a parameter. As you copy/paste in this you will see that HUE determined that there is a parameter that needs to be entered.
- In the “user_id” parameter box, enter your user id (see Lab Setup section)

user_id	<input type="text"/>
---------	----------------------

- Execute the Query by clicking on the  button
- Check to see if the Database was created
 - Delete the current query from the SQL Window
 - Copy & paste the SQL below

```
SHOW DATABASES;
```

- Execute the Query by clicking on the  button to see that the database was created

Results

```
DATABASE_NAME  
...  
user001_airlines  
...  
<user_id>_airlines  
...  
user100_airlines  
...
```

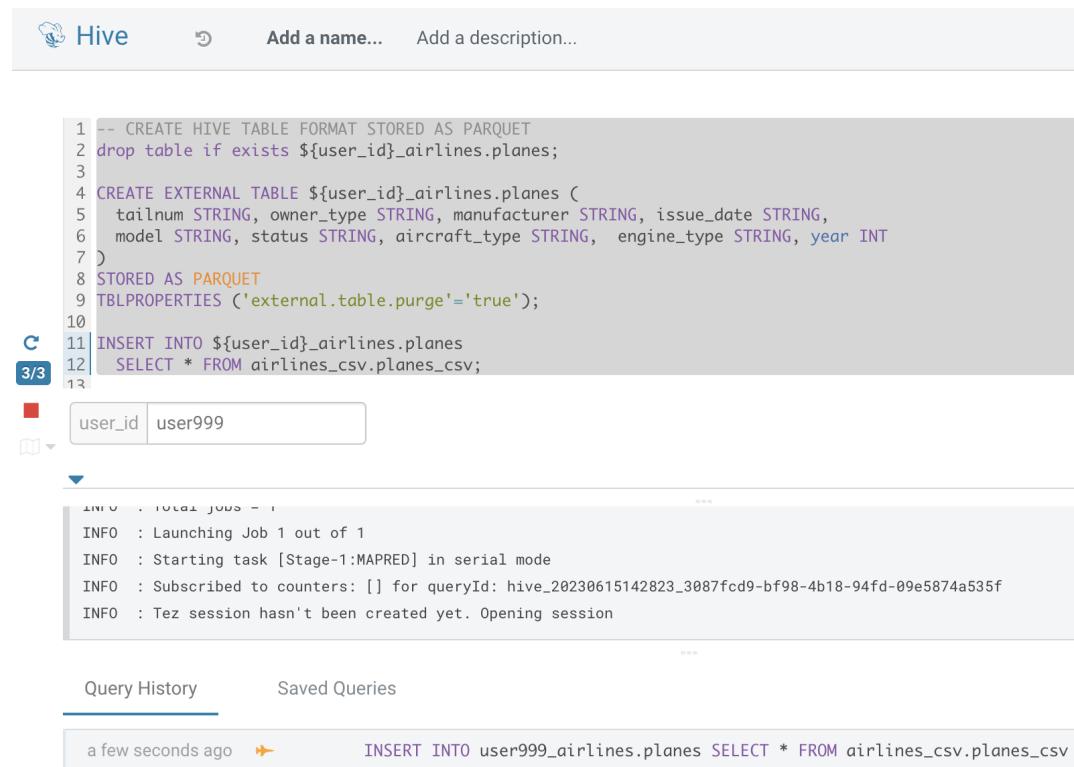
There may be many databases, scroll & look for the one that starts with your <user_id>

- **Let's simulate existing tables in our Data Lakehouse**

- Create planes table in *Hive Table Format*, stored in Parquet file format
 - Delete the current query from the SQL Window
 - Copy & paste the SQL below

```
-- CREATE HIVE TABLE FORMAT STORED AS PARQUET  
-- drop table if exists ${user_id}_airlines.planes;  
  
CREATE EXTERNAL TABLE ${user_id}_airlines.planes (  
    tailnum STRING, owner_type STRING, manufacturer STRING, issue_date STRING,  
    model STRING, status STRING, aircraft_type STRING, engine_type STRING, year INT  
)  
STORED AS PARQUET  
TBLPROPERTIES ('external.table.purge'='true');  
  
INSERT INTO ${user_id}_airlines.planes  
    SELECT * FROM airlines_csv.planes_csv;
```

- Select all of the content in the SQL Editor
- Execute the Query by clicking on the  button



The screenshot shows the Cloudera Manager Hive interface. At the top, there's a header with a Hive icon, the word "Hive", and input fields for "Add a name..." and "Add a description...". Below the header is a code editor containing a Hive script:

```

1 -- CREATE HIVE TABLE FORMAT STORED AS PARQUET
2 drop table if exists ${user_id}_airlines.planes;
3
4 CREATE EXTERNAL TABLE ${user_id}_airlines.planes (
5   tailnum STRING, owner_type STRING, manufacturer STRING,
6   model STRING, status STRING, aircraft_type STRING, engine_type STRING, year INT
7 )
8 STORED AS PARQUET
9 TBLPROPERTIES ('external.table.purge='true');
10
11 C INSERT INTO ${user_id}_airlines.planes
12   SELECT * FROM airlines_csv.planes_csv;
13

```

The line 11 is highlighted with a red background, and the value "user999" is selected in the dropdown below it. To the left of the code editor, there's a status indicator "3/3" and a small red square icon.

Below the code editor is a log window showing the execution of the query:

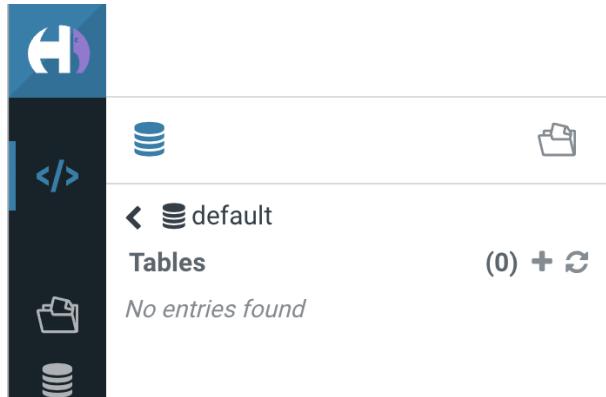
```

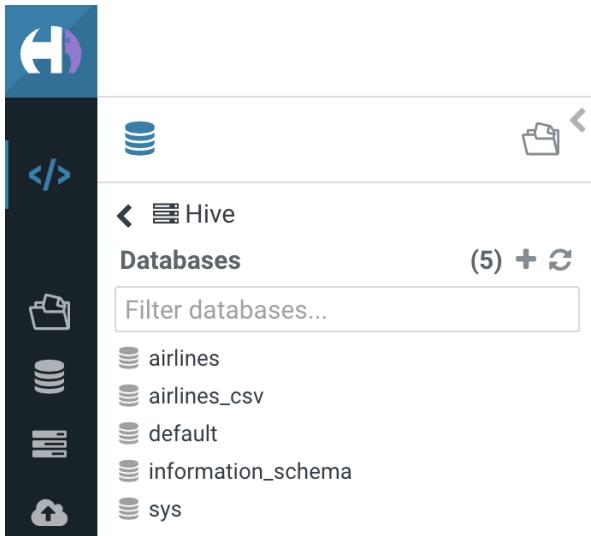
INFO : Total jobs = 1
INFO : Launching Job 1 out of 1
INFO : Starting task [Stage-1:MAPRED] in serial mode
INFO : Subscribed to counters: [] for queryId: hive_20230615142823_3087fc98-bf98-4b18-94fd-09e5874a535f
INFO : Tez session hasn't been created yet. Opening session

```

At the bottom of the interface, there are tabs for "Query History" and "Saved Queries", and a search bar with the query text "INSERT INTO user999_airlines.planes SELECT * FROM airlines_csv.planes_csv".

- Switch Database to the user Database
 - On the left side of the SQL Editor, click on the < arrow next to default





- To the right of Databases click on the button to refresh the list of Databases

Databases

- airlines
- airlines_csv
- default
- information_schema
- sys
- user999_airlines

- Click on your <user-id>_airlines Database - this will allow you to track what has been created in your database through the next steps

user999_airlines

Tables

(1)

- planes

- Delete the current query from the SQL Window
- Copy & paste the SQL below

```
DESCRIBE FORMATTED ${user_id}_airlines.planes;
```

- Execute the Query by clicking on the ➤ button
 - In the output look for the following fields - scroll down to look for the following properties: Location, Table Type, and SerDe Library (this value should reflect the ParquetHiveSerDe, indicating this is a Hive Table Format)

18	Location:	s3a://path-2-cloud-object-storage	NULL
19	Table Type:	EXTERNAL_TABLE	NULL
20	Table Parameters:	NULL	NULL
21	COLUMN_STATS_ACCURATE	0(BASIC_STATS)	NULL
22	EXTERNAL	TRUE	NULL
23	bucketing_version	2	NULL
24	external_table_purge	true	NULL
25	numFiles	1	NULL
26	numRows	5029	NULL
27	rowSize	45261	NULL
28	totalSize	111491	NULL
29	transient_lastDdlTime	1668538074	NULL
30	NULL	NULL	NULL
31	# Storage Information	NULL	NULL
32	Serde Library:	org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe	NULL
33	InputFormat:	org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat	NULL
34	OutputFormat:	org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat	NULL

10. Build the Data Lakehouse

- Stay in HUE for the CDW Hive Virtual Warehouse - [airlines-hive-vw](#).
- **Migrating Hive Table Format to Iceberg Table format** - If you already have created a Data Warehouse using the Hive Table Format, but would like to take advantage of the features offered in the Iceberg Table Format, **you have two (2) options: 1) Utilize the in-place table Migration feature; or 2) Use Create Table as Select (CTAS)**
 - **Option 1: Migrate the planes table in our Data Lakehouse from Hive Table Format to Iceberg Table Format using the Migration Utility.**
 - Delete the current query from the SQL Window
 - Copy & paste the SQL below

```
ALTER TABLE ${user_id}_airlines.planes
SET TBLPROPERTIES
('storage_handler'='org.apache.iceberg.mr.hive.HiveIcebergStorageHandler');

DESCRIBE FORMATTED ${user_id}_airlines.planes;
```

- Select all of the content in the SQL Editor
- Execute the Query by clicking on the ➤ button
- This migration to Iceberg happened in-place, there was no rewriting of data that occurred as part of this process. It retained the File Format of Parquet for the Iceberg table as well. There was a Metadata file that is created, which you can see when you run the DESCRIBE FORMATTED.
- In the output look for the following fields - scroll down to look for the following properties: look for the following (see image with highlighted fields) key values: Table Type, Location (location of where table data is stored), SerDe Library, and in Table Parameters look for properties MIGRATE_TO_ICEBERG, storage_handler, metadata_location, and table_type
 - Location - Data is stored in cloud storage in this case S3 in the same location as the Hive Table Format
 - Metadata_location - Since there is no need to regenerate data files

with in-place table migration, you save time generating Iceberg tables. Only metadata is regenerated, which points to source data files. Removes Hive Metastore as bottleneck.

- Table_type - indicates "ICEBERG" table format
- Storage_handler & SerDe Library - indicate what Serializer/Deserializer to use when reading/writing data in this case the "HiveIcebergSerDe"

18	Location:	s3a://path-2-cloud-object-storage	hive/user001_airlines.db/planes	NULL
19	Table Type:	EXTERNAL_TABLE		NULL
20	Table Parameters:	NULL		NULL
21		EXTERNAL		TRUE
22	MIGRATED_TO_ICEBERG	true		
23	bucketing_version	2		
24	engine.hive.enabled	true		
25	external.table.purge	true		
26	last_modified_by	jingails		
27	last_modified_time	1674156492		
28	metadata_location	s3a://path-2-cloud-object-storage		
29	numFiles	1		
30	numRows	5029		
31	previous_metadata_location	s3a://goes-se-sandbox01/warehouse/tablespace/extern		
32	rawDataSize	45261		
33	schema.name-mapping.default	[\n \`field-id\` : 1, \n \`names\` : [\n \`lalinum\` \n], \n \`f		
34	storage_handler	org.apache.iceberg.mr.hive.HiveIcebergStorageHandler		
35	table_type	ICEBERG		
36	totalSize	111491		
37	transient_LastDdlTime	1674156492		
38	uuid	2280a4cc-9950-4d80-8115-2999eed8d30f		
39	write.format.default	parquet		
40	NULL	NULL		
41	# Storage Information	NULL		
42	SerDe Library:	org.apache.iceberg.mr.hive.HiveIcebergSerDe		
43	InputFormat:	org.apache.iceberg.mr.hive.HiveIcebergInputFormat		
44	OutputFormat:	org.apache.iceberg.mr.hive.HiveIcebergOutputFormat		
45	Compressed:	No		
46	Sort Columns:	[]		NULL

- Option 2: Create airports table in **Iceberg Table Format**, using Create Table As Select (CTAS). Notice the syntax to create an Iceberg Table within Hive is "Stored by Iceberg"

- Delete the current query from the SQL Window

- Copy & paste the SQL below

```
-- drop table if exists ${user_id}_airlines.airports;
CREATE EXTERNAL TABLE ${user_id}_airlines.airports
STORED BY ICEBERG AS
SELECT * FROM airlines_csv.airports_csv;

DESCRIBE FORMATTED ${user_id}_airlines.airports;
```

- Select all of the content in the SQL Editor
- Execute the Query by clicking on the ▶ button
 - In the output - look for the following key values (just like previously): Table Type, Location (location of where table data is stored), SerDe Library, and in Table Parameters look for properties MIGRATE_TO_ICEBERG, storage_handler, metadata_location, and table_type
 - On the left you should see the airports table added to the list of Tables for your user Database

- Slowly changing dimension table airlines
 - **Iceberg is fully ACID compliant and can execute Insert, Update, Delete, and Merge Into statements**
 - Delete the current query from the SQL Window
 - Copy & paste the SQL below

```
-- SLOWLY CHANGING DIMENSION TABLE USE ACID CAPABILITIES OF ICEBERG
drop table if exists ${user_id}_airlines.airlines;

CREATE EXTERNAL TABLE ${user_id}_airlines.airlines (
  code string,
  description string
)
STORED BY ICEBERG
STORED AS PARQUET
tblproperties('format-version'='2');

-- LOAD DATA
INSERT INTO ${user_id}_airlines.airlines
SELECT * FROM airlines_csv.airlines_csv;

-- Check data to see a few records
SELECT *
FROM ${user_id}_airlines.airlines
WHERE code IN ("04Q", "05Q");
```

- Select all of the content in the SQL Editor
- Execute the Query by clicking on the ▶ button
 - This will create the Iceberg Table, load initial data, and display a few rows that will be modified in the next step
- Delete the current query from the SQL Window
- Copy & paste the SQL below

```
-- SLOWLY CHANGING DIMENSION TABLE USE ACID CAPABILITIES OF ICEBERG
MERGE INTO ${user_id}_airlines.airlines AS t
  USING airlines_csv.airlines_csv s ON t.code = s.code
  WHEN MATCHED AND t.code = "04Q" THEN DELETE
  WHEN MATCHED AND t.code = "05Q" THEN UPDATE SET description = "Comlux Aviation"
  WHEN NOT MATCHED THEN INSERT VALUES (s.code, s.description);

-- Check data to see records were deleted or updated
SELECT *
FROM ${user_id}_airlines.airlines
WHERE code IN ("04Q", "05Q");
```

- Select all of the content in the SQL Editor
- Execute the Query by clicking on the ▶ button
 - The Merge Into statement will check to see if a record needs to be Deleted (when the record key matches and the code = 04Q), when a new record needs to be Inserted (key doesn't match), and when an Update is needed (when the key matches and the code = 05Q)
 - Compare the output to see that only one record is returned with the description set to "Comlux Aviation"

- **Creating an Iceberg Table - for this step create a partitioned table, in Iceberg Table Format, stored in Parquet File Format.** Optionally, you could specify other File Formats, the supported formats for Iceberg are: Parquet, ORC, and Avro.

- Delete the current query from the SQL Window
- Copy & paste the SQL below

```

drop table if exists ${user_id}_airlines.flights;
CREATE EXTERNAL TABLE ${user_id}_airlines.flights (
    month int, dayofmonth int,
    dayofweek int, deptime int, crsdeptime int, arrtime int,
    crsarrrtime int, uniquecarrier string, flightnum int, tailnum string,
    actualelapsedtime int, crselapsedtime int, airtime int, arrdelay int,
    depdelay int, origin string, dest string, distance int, taxiin int,
    taxiout int, cancelled int, cancellationcode string, diverted string,
    carrierdelay int, weatherdelay int, nasdelay int, securitydelay int,
    lateaircraftdelay int
)
PARTITIONED BY (year int)
STORED BY ICEBERG
STORED AS PARQUET
tblproperties ('format-version'='2');

SHOW CREATE TABLE ${user_id}_airlines.flights;

```

- Select all of the content in the SQL Editor

- Execute the Query by clicking on the ▶ button

- Looking at the SHOW CREATE TABLE output, scroll down and notice the output is the unformatted version of the Describe Formatted as we would expect. The main item to pay attention to here is the PARTITIONED BY SPEC, currently we've partitioned by just the "year" column

```

29 'lateaircraftdelay' int,
30 'year' int)
31 PARTITIONED BY SPEC (
32 year
33 ROW FORMAT SERDE
34 'org.apache.iceberg.mr.hive.HiveIcebergSerDe'
35 STORED BY
36 'org.apache.iceberg.mr.hive.HiveIcebergStorageHandler'
37
38 LOCATION
39 's3a://path-2-cloud-object-storage' re/user001_airlines.db/flights'
40 TBLPROPERTIES (
41 'bucketing_version'=2,
42 'engine.hive.enabled'=true,
43 'metadatolocation' s3a://path-2-cloud-object-storage /user001_airlines.db/flights/metadata/00000-2ba52a08-8af7-4eeb-87f7-eb4ad4133be5.metadata.json',
44 'serialization.format'=1,
45 'table_type'=ICEBERG,
46 'transient_lastDdlTime'=1674157757,
47 'uuid'=837acSee-79e6-48d2-bf43-a82cd04b1d2ba,
48 'write.format.default'=parquet)

```

- When we insert data into this table it will write data together within the same partition (ie. all 2006 data is written to the same location, all 2005 data is written to the same location, etc.)

- Delete the current query from the SQL Window
- Copy & paste the SQL below

```

INSERT INTO ${user_id}_airlines.flights
SELECT * FROM airlines_csv.flights_csv
WHERE year <= 2006;

```

```
SELECT year, count(*)
FROM ${user_id}_airlines.flights
GROUP BY year
ORDER BY year desc;
```

- Select all of the content in the SQL Editor
- Execute the Query by clicking on the ▶ button
 - The Insert will insert data from years 1995 to 2006
 - Notice that each of the years have a range of data within a few million flights (each record in the flights table counts as a flight)

The screenshot shows the Cloudera Data Platform interface. At the top, there are tabs for 'Query History', 'Saved Queries', and 'Results (12)'. Below the tabs is a table with a header 'year' and 12 rows of data. The data is as follows:

year	_c1
1 2006	7141922
2 2005	7140596
3 2004	7129270
4 2003	6488540
5 2002	5271359
6 2001	5967780
7 2000	5683047
8 1999	5527884
9 1998	5384721
10 1997	5411843
11 1996	5351983
12 1995	5327435

Below the table, the Hive editor window shows the following code and logs:

```
1 INSERT INTO ${user_id}_airlines.flights
2 SELECT * FROM ${user_id}_airlines_raw.flights_csv
3 WHERE year <= 2006;
4
5 SELECT year, count(*)
6 FROM ${user_id}_airlines.flights
7 GROUP BY year
8 ORDER BY year desc;
9
10 user_id user001
```

```
INFO : ANALYZE TABLE ${user_id}_airlines.flights;
INFO : INPUT_FILES_Map_1: 1
INFO : RAM_INPUT_SPLITS_Map_1: 15
INFO : Completed executing command(queryId=hive_20230119204024_5863964c-0b1f-4090-a0e9-58585d4af257); Time taken: 2.54 seconds
INFO : OK
```

11. Performance improvements

- Check that you created the tables for the Data Lakehouse - on the list of tables to the left of the Editor window you should see the following tables

The screenshot shows the 'Tables' section of the interface. It includes a 'Tables' heading, a '(4) + ⚡' button, a 'Filter...' input field, and a list of four tables: 'airlines', 'airports', 'flights', and 'planes'.

- **Iceberg in-place Partition Evolution [Performance Optimization]**

- One of the key features for Iceberg tables is the ability to evolve the partition that is being used over time
 - Delete the current query from the SQL Window
 - Copy & paste the SQL below

```
ALTER TABLE ${user_id}_airlines.flights
SET PARTITION spec ('year, month');

SHOW CREATE TABLE ${user_id}_airlines.flights;
```

- Select all of the content in the SQL Editor
- Execute the Query by clicking on the ▶ button

- This was an in-place Partition Evolution, meaning that the existing data is not rewritten as part of the ALTER TABLE execution. What will happen is the next data that is loaded will use the new Partition definition.
- In the output scroll to see the PARTITIONED BY SPEC to see that it is now both year and month

```

30   `year` int)
31   PARTITIONED BY SPEC (
32   year,
33   month)

```

12. Performance Improvements - Open HUE for the CDW Impala Virtual Warehouse - [airlines-impala-vw](#).

- There should be an open CDW Browser tab, open the  browser tab
- Click on the [airlines-impala-vw](#)# tile
 - In the upper right corner click on the HUE button to enter into the SQL Editor



- Load new data into the flights table using the NEW partition definition - this will load new data to take advantage of the new partition specification. This also shows how Iceberg also supports multiple engines by allowing both Hive & Impala to create, load, query, and or modify Iceberg tables.
- Copy & Paste the following in the SQL Editor window

```

INSERT INTO ${user_id}_airlines.flights
SELECT * FROM airlines_csv.flights_csv
WHERE year = 2007;

```

- Execute the Query by clicking on the  button
- Run Explain Plans against some typical analytic queries we might run to see what happens with this new Partition definition.
 - In Impala we are in another engine which is another key feature of Iceberg - multi-function (or multiple engine) analytics. No need to copy the data or do more work to allow access to the same data.
 - Copy/paste the following in the Editor, but do not execute the query

```

SELECT year, month, count(*)
FROM ${user_id}_airlines.flights
WHERE year = 2006 AND month = 12
GROUP BY year, month
ORDER BY year desc, month asc;

```

- In the “user_id” parameter box, enter your user id (see Lab Setup section)

- Instead select (highlight) the statement click the  button, which is right below the Execute (▶) button and select Explain



- In the output notice the amount of data that needs to be scanned for this query (it would represent the volume of 1 years worth of data - about 139MB). Up to this point the data was loaded using the Partition of just a year.

The screenshot shows the 'Explain' tab of the Impala Query History. The tab has three tabs: 'Query History', 'Saved Queries', and 'Explain' (which is selected). The output area displays the execution plan and resource estimates:

```

Max Per-Host Resource Reservation: Memory=86.00MB Threads=3
Per-Host Resource Estimates: Memory=563MB
Dedicated Coordinator Resource Estimate: Memory=210MB
WARNING: The following tables are missing relevant table and/or column statistics.
user999_airlines.flights

PLAN-ROOT SINK
|
05:MERGING-EXCHANGE [UNPARTITIONED]
|   order by: `year` DESC, `month` ASC
|
02:SORT
|   order by: `year` DESC, `month` ASC
|   row-size=16B cardinality=7.14M
|
04:AGGREGATE [FINALIZE]
|   output: count:merge(*)
|   group by: `year`, `month`
|   row-size=16B cardinality=7.14M
|
03:EXCHANGE [HASH(`year`, `month`)]
|
01:AGGREGATE [STREAMING]
|   output: count()
|   group by: `year`, `month`
|   row-size=16B cardinality=7.14M
|
00:SCAN S3 [user999_airlines.flights]
|   S3 partitions=1/1 files=1 size=139.31MB
|   predicates: `month` = 12, `year` = 2006
|   row-size=8B cardinality=7.14M

```

The 'SCAN S3' step is highlighted with a red box, showing the following details:

- S3 partitions=1/1 files=1 size=139.31MB
- predicates: `month` = 12, `year` = 2006
- row-size=8B cardinality=7.14M

- Copy/paste the following in the Editor, but do not execute the query

```

SELECT year, month, count(*)
FROM ${user_id}_airlines.flights
WHERE year = 2007 AND month = 12
GROUP BY year, month
ORDER BY year desc, month asc;

```

- Instead select (highlight) the statement click the  button, which is right below the Execute (▶) button and select Explain
- In the output notice the amount of data that needs to be scanned for this query, about 10MB, is significantly less than that of the first, 138MB. This shows an important capability, Partition Pruning. Meaning that much less data is being scanned for this query and only the selected month of data is being scanned vs scanning the entire year of data. This should result in much faster query execution times.

Query History Saved Queries Explain

```

Max Per-Host Resource Reservation: Memory=86.00MB Threads=3
Per-Host Resource Estimates: Memory=308MB
Dedicated Coordinator Resource Estimate: Memory=119MB
WARNING: The following tables are missing relevant table and/or column statistics.
user999_airlines.flights

PLAN-ROOT SINK
|
05:MERGING-EXCHANGE [UNPARTITIONED]
| order by: `year` DESC, `month` ASC
|
02:SORT
| order by: `year` DESC, `month` ASC
| row-size=16B cardinality=614.14K
|
04:AGGREGATE [FINALIZE]
| output: count:merge(*)
| group by: `year`, `month`
| row-size=16B cardinality=614.14K
|
03:EXCHANGE [HASH(`year`, `month`)]
|
01:AGGREGATE [STREAMING]
| output: count(*)
| group by: `year`, `month`
| row-size=16B cardinality=614.14K
|
00:SCAN S3 [user999_airlines.flights]
| S3 partitions=1/1 files=1 size=10.27MB
| predicates: `month` = 12, `year` = 2007
| row-size=8B cardinality=614.14K

```

- Iceberg Snapshots Time Travel** - in the previous steps we have been loading data into the flights Iceberg table.
 - Each time data was changed in our Iceberg tables, a Snapshot is automatically captured. This is important for many reasons but the main point of the Snapshot is to ensure eventual consistency and allow for multiple reads/writes concurrently (from various engines or the same engine).
 - Show snapshots
 - Delete the current query from the SQL Window
 - Copy & paste the SQL below

```
DESCRIBE HISTORY ${user_id}_airlines.flights;
```

- Execute the Query by clicking on the ▶ button

In the output there should be 2 Snapshots, in the example below there are 3 snapshots as this example shows that data was also loaded via Impala and not just Hive. Also, keep in mind we have been reading/writing data from/to the Iceberg table from both Hive & Impala which is indicated by the () in the callouts below. This is important aspect of Iceberg Tables is that they support multi-function analytics - ie. many engines can work with Iceberg tables (Cloudera Data Warehouse [Hive & Impala], Cloudera Data Engineering [Spark], Cloudera Machine Learning [Spark], Cloudera DataFlow [NiFi], and DataHub Clusters)

creation_time	snapshot_id	parent_id	is_current_ancestor
2023-01-19 20:40:16.453000000	1517273939679890388	NULL	TRUE
2023-01-20 02:45:32.160000000	2995408773555868055	1517273939679890388	TRUE
2023-01-20 02:55:36.669000000	1112270214077101459	2995408773555868055	TRUE

■ Get Details for Snapshots - open a text editor/notepad

creation_time	snapshot_id	parent_id	is_current_ancestor
2023-01-19 20:40:16.453000000	1517273939679890388	NULL	TRUE
2023-01-20 02:45:32.160000000	2995408773555868055	1517273939679890388	TRUE
2023-01-20 02:55:36.669000000	1112270214077101459	2995408773555868055	TRUE

Text Editor - copy in a couple creation_time's and snapshot_id's

```
Creation_time
2023-01-19 20:40:16.453000000
2023-01-20 02:45:32.160000000

snapshot_id
1517273939679890388
2995408773555868055
```

■ Iceberg Time Travel [Table Maintenance] - copy/paste the following data into the Impala Editor, but do not execute.

- Delete the current query from the SQL Window
- Copy & paste the SQL below

```
-- SELECT DATA USING TIMESTAMP FOR SNAPSHOT
SELECT year, count(*)
FROM ${user_id}_airlines.flights
FOR SYSTEM_TIME AS OF '${create_ts}'
GROUP BY year
ORDER BY year desc;

-- SELECT DATA USING TIMESTAMP FOR SNAPSHOT
SELECT year, count(*)
FROM ${user_id}_airlines.flights
FOR SYSTEM_VERSION AS OF ${snapshot_id}
GROUP BY year
ORDER BY year desc;
```

- Once you copy this SQL into the Editor you will see 2 new parameters - `create_ts` and `snapshot_id`

The screenshot shows the Cloudera Impala Query Editor interface. At the top, there are three parameter input fields: `user_id` (set to `user001`), `create_ts` (empty), and `snapshot_id` (empty). Below the editor, a message indicates that the query is waiting for executors to start.

- The first SELECT statement will use the `create_ts`

- In the `create_ts` parameter box enter a date/time (this can be relative or specific timestamp). From your Text Editor copy the second entry under `creation_time`, paste it into the `create_ts` parameter. For this Time Travel you can use relative time periods and don't have to enter the specific timestamp for the Snapshot.
- Highlight the first SELECT statement, and execute it

The screenshot shows the results of the first SELECT statement. The results table has a single column `year` and a header `count(*)`. The data is as follows:

year	count(*)
1	7141922
2	7140596
3	7129270
4	6488540
5	5271359
6	5967780
7	5683047
8	5527884
9	5384721
10	5411843
11	5351983
12	5327435

- This returned the the data as it was in our initial load

- The second SELECT statement will use the `snapshot_id`

- From your Text Editor copy the second entry under `snapshot_id`, paste it into the `snapshot_id` parameter box
- Highlight the second SELECT statement, and execute it

The screenshot shows the results of the second SELECT statement. The results table has a single column `year` and a header `count(*)`. The data is as follows:

year	count(*)
1	7453215
2	7141922
3	7140596
4	7129270
5	6488540
6	5271359
7	5967780
8	5683047
9	5527884
10	5384721
11	5411843
12	5351983
13	5327435

- This returns the data for the specific Snapshot ID that was specified in the query which was the data for the initial insert for data up to year 2006 and the insert for data for year 2007

- **Security & Governance** - Now that performance has been optimized, we are almost ready to release this to the users. Before we do that we need to apply security - Security is always on and can be defined with fine grained access control. A security policy has already been created. This is how it is defined

Create Policy

Please ensure that users/groups listed in this policy have access to the column via an Access Policy. This policy does not implicitly grant access to the column.

Policy Details:

Policy Type	Masking	<input type="button" value="Add Validity Period"/>
Policy Name *	jingalls-iceberg-fgac	<input checked="" type="radio"/> Enabled <input type="radio"/> Normal
Policy Label	Policy Label	
Hive Database *	x jingalls_airlines	
Hive Table *	x planes	
Hive Column *	x tailnum	
Description		
Audit Logging	<input checked="" type="radio"/> Yes	

Mask Conditions:

Select Role	Select Group	Select User	Access Types	Select Masking Option
<input type="button" value="Select Roles"/>	<input type="button" value="Select Groups"/>	<input type="button" value="Select User"/>	<input checked="" type="radio"/> select <input type="radio"/> edit	<input checked="" type="radio"/> Hash <input type="radio"/> Mask

SQL Editor:

- Delete the current query from the SQL Window
- Copy & paste the SQL below Select all of the content in the SQL Editor
- Execute the Query by clicking on the ▶ button

```
1 SELECT * FROM airlines.planes;
```

▶

Locally submitted queries (even if running in exclusive) will start only one queries and queries submitted while in the background (unless UN_NODEN set to 1 or when small query optimization is triggered) can currently run.

Query 17416573231856ea:cd8c269600000000 100% Complete (1 out of 1)
Query 17416573231856ea:cd8c269600000000 100% Complete (1 out of 1)
Query 17416573231856ea:cd8c269600000000 100% Complete (1 out of 1)

17416573231856ea:cd8c269600000000

Query History **Saved Queries** **Results (5,029)**

tailnum	owner_type	manufacturer	issue_date	moc
82d6b5373e2ae6bd8e9a437f5a7680a21b11f1753721e9cd0968ca8e8d4dd31				
9c02c0930c86b5c841989aa3ddafe9e65a84c9e7780d461e4259dd53fffc7ab				
b16ae68d29c050a77412a89d54ae322ab7181cfd8c07f63ee983c6d927c8773				
a8c94ab3f0aea271b3993f8623ce46c3a5217dd5f9e7532cf05edf39fb2ebf7d				
4cd32f901791cb791650dadbbc701b269bea9fce4edca9e070efa53af5e7				
f07afae0cc8e13e9c2602d82e0e691e79cb5eefbfbee500020b23c950f02f95de				
a2cc953614cd5ee3fdd5bedd65b6e9897e8234a17c05ae26372ce83b522b1efa1				
0bd27cb31e035e19c7f04b82a53c6d835bd4c6f23569b3065104dbe4e2fc81b9				
74876a7424a5e2f67ebec433f6d3ebe058cfb2dfb75742aed61eb00d98047130				
b1231b80352bcbea01263b29582437e5a551cd8bbd845c9e0240b75fd4f58f0				
11 c5a63f3d36f5a51a6628d728d11n0lh34herl17rf44787q28n0ad4rh022r6327fe				

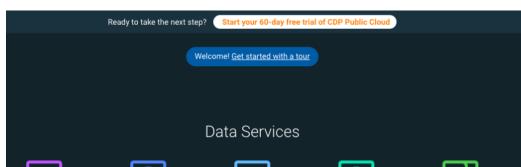
- You will see that the “tailnum” column is masked so you are unable to see the actual plane’s tail number, instead you see the Hashed value of the tailnum
-

(Optional Bonus Material)

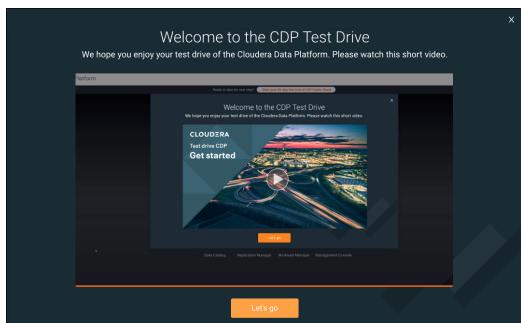
Optional Lab 1 - CDW Tour

In this Lab you will explore how to take advantage of CDW.

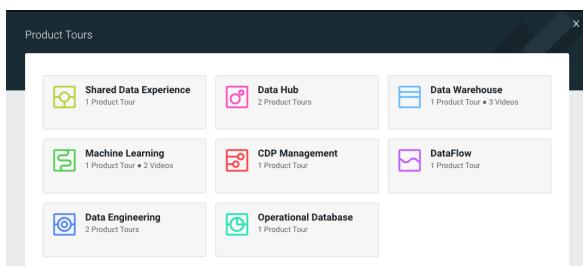
13. Click on the “Get Started with a Tour” at the top of the CDP Home screen



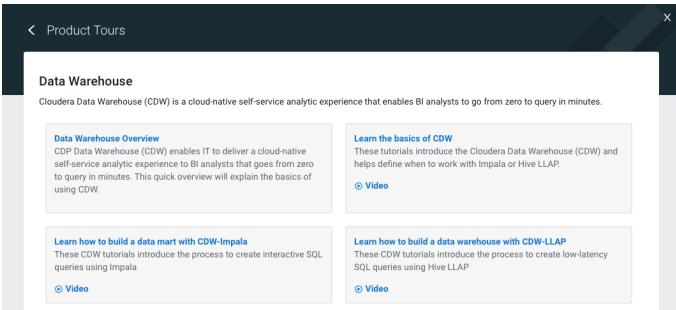
14. [Optional] Watch the “Test Drive” Video on the CDP Home screen (video is ~3 minutes)



15. Click on the “Let’s go” button at the bottom of the video, you can take tours of any of the Data Services
16. Click on the Data Warehouse tour tile



- Here you see the content you can take advantage of - there are 3 tiles where you can watch videos on an area of interest, you will see a “> Video” link on the tile if this is a video, and there is also a tour.



- Click on the “Data Warehouse Overview” tile, this is a click through tour of CDW
 - Perform all of the actions for this click through tour (~5 minutes)
 - Once completed click on the “X” in the top right corner to close the window

17. On the Product Tours screen, explore the various topics to learn more about Cloudera Data Warehouse
-

Optional Lab 2 - Data Security & Governance

In this lab you will experience the combination of what the Data Warehouse and the Shared Data Experience (SDX) offers. SDX enables you to provide Security and Governance tooling to ensure that you will be able to manage what is in the CDP Platform without having to stitch together multiple tools.

Robust Data Catalog (Governance) capability that:

- Enables you to understand, manage, secure, and govern data assets across
- Provides a 360 degree view of Assets - Lineage, Metadata, Security Policy applied to the asset
 - There are many assets - everything created in CDP is captured as an Asset
 - For this Workshop we have been working with Assets, like - databases, views, tables, etc.

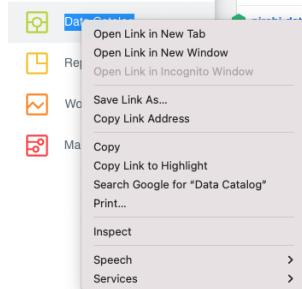
Powerful Security features like:

- Rule-based masking columns based on a user's role
- Group association or rule-based row filters
- Attribute-Based Access Control a.k.a. Tag-based security policies

18. Data Catalog - Governance (Lineage, Metadata, etc.)
 - On the left navigation menu click the next to Data Warehouse



- Right click on the Data Catalog and select “Open Link in New Tab” option



- Open browser tab that just opened, should have Data Catalog in the tab title

Type	Name	Qualified Name	Created On	Owner	Source
AWS S3 V2 Bucket	aws/se-sandbox01	a3a://pose-se-sandbox01@com	NA	NA	aws
AWS S3 V2 Bucket	proj-expedia0001	a3a://proj-expedia0001@com	NA	NA	aws
Impala Process Execution	QUERY exp_data_dc_consumption_data@cm:1674...	QUERY exp_data_dc_consumption_data@cm:1674...	NA	NA	impala
Impala Process Execution	QUERY exp_data_dc_consumption_data@cm:1674...	QUERY exp_data_dc_consumption_data@cm:1674...	NA	NA	impala
Impala Process Execution	QUERY exp_data_dc_consumption_data@cm:1674...	QUERY exp_data_dc_consumption_data@cm:1674...	NA	NA	impala
Impala Process Execution	QUERY exp_data_dc_consumption_data@cm:1674...	QUERY exp_data_dc_consumption_data@cm:1674...	NA	NA	impala
Impala Process Execution	QUERY exp_data_dc_consumption_data@cm:1674...	QUERY exp_data_dc_consumption_data@cm:1674...	NA	NA	impala
Impala Process Execution	QUERY exp_data_dc_consumption_data@cm:1674...	QUERY exp_data_dc_consumption_data@cm:1674...	NA	NA	impala
Impala Process Execution	QUERY exp_data_dc_consumption_data@cm:1674...	QUERY exp_data_dc_consumption_data@cm:1674...	NA	NA	impala
Impala Process Execution	QUERY exp_data_dc_consumption_data@cm:1674...	QUERY exp_data_dc_consumption_data@cm:1674...	NA	NA	impala
Impala Process Execution	QUERY exp_data_dc_consumption_data@cm:1674...	QUERY exp_data_dc_consumption_data@cm:1674...	NA	NA	impala
Impala Process Execution	QUERY exp_data_dc_consumption_data@cm:1674...	QUERY exp_data_dc_consumption_data@cm:1674...	NA	NA	impala
Impala Process Execution	QUERY exp_data_dc_consumption_data@cm:1674...	QUERY exp_data_dc_consumption_data@cm:1674...	NA	NA	impala
Impala Process Execution	QUERY exp_data_dc_consumption_data@cm:1674...	QUERY exp_data_dc_consumption_data@cm:1674...	NA	NA	impala
Impala Process Execution	QUERY exp_data_dc_consumption_data@cm:1674...	QUERY exp_data_dc_consumption_data@cm:1674...	NA	NA	impala

- To the left, under Data Lakes, ensure the Data Lake provided in Lab Setup radio button is selected

Data Lakes	
<input checked="" type="radio"/> aws se-aw-mdl	2110
<input type="radio"/> aws glue:se-aw-mdl	NA
<input type="radio"/> aws se-az-dl	635
<input type="radio"/> aws loki-datalake	41
<input type="radio"/> aws glue:loki-datalake	NA
<input type="radio"/> aws pp-ut-aw-dl	!
<input type="radio"/> aws glue:pp-ut-aw-dl	!

- Filter for Assets we created - below the Data Lakes on the left of the screen under Filters, select TYPE of Hive Table. The right side of the screen will update to reflect this selection

Filters

TYPE

Hive Table

HBase Table

+ Add New Value Clear

- Under DATABASE, click +Add new Value. In the box that appears start typing your <user_id> when you see the <user_id>_airlines database pop up select it

SEARCH RESULTS

SEARCH TERM	RESULTS
reddit_data	No results
prescribing_p_e	No results
hol_rj	No results
default	No results
information_schema	No results
user	user001_airlines_raw user001_airlines

- You should now see the tables and materialized views that have been created in the **<user_id>_airlines** database. Click on flights in the Name column to view more details on the flights table.

SETUP THE PROFILER FOR se-aw-mdl

Type	Name	Qualified Name	Created On	Owner	Source
Hive Table	airlines_src	user001_airlines.airlines_src@com	Thu Jan 19 2023	jingali	hive
Hive Table	airports	user001_airlines.airports@com	Thu Jan 19 2023	hive	hive
Hive Table	airlines	user001_airlines.airlines@com	Fri Jan 20 2023	jingali	hive
Hive Table	planes_csv	user001_airlines.raw_planes.csv@com	Wed Jan 18 2023	jingali	hive
Hive Table	traffic_cancel_airlines	user001_airlines.traffic_cancel_airlines@com	Fri Jan 20 2023	jingali	hive
Hive Table	planes	user001_airlines.planes@com	Thu Jan 19 2023	jingali	hive
Hive Table	airlines_dim_updates	user001_airlines.airlines_dim_updates@com	Thu Jan 19 2023	jingali	hive
Hive Table	flights	user001_airlines.flights@com	Thu Jan 19 2023	jingali	hive
Hive Table	airlines_csv	user001_airlines.raw_airlines_csv@com	Wed Jan 18 2023	jingali	hive
Hive Table	airports_csv	user001_airlines.raw_airports_csv@com	Wed Jan 18 2023	jingali	hive
Hive Table	flights_csv	user001_airlines.raw_flights_csv@com	Thu Jan 19 2023	jingali	hive

- This page shows information about the flights table such as the table owner, when the table was created, when it was last accessed, and other properties. Below the summary details is the Overview tab which shows the lineage - hoover over the flights click on the "i" icon that appears to see more detail on this table

ASSET DETAILS

flights

Properties

Type: HIVE TABLE
of Columns: 29
Data Lake: se-aw-mdl
Owner: jingali
Created On: Thu Jan 19 2023 13:49:17 GMT-0600 (Central Standard Time)
Last Access Time: Thu Jan 19 2023 13:49:17 GMT-0600 (Central Standard Time)
Table Type: EXTERNAL_TABLE
Database: user001_airlines
DB Catalog: em
Parent: user001_airlines

Classifications

Overview

Lineage

```

graph LR
    flights_src[flights_src] --> flights_csv[flights_csv]
    flights_csv --> flights[flights]
    flights_src --> flights[flights]
    flights_src --> traffic_cancel_airlines[traffic_cancel_airlines]
    flights_src --> flights[flights]
    flights_src --> flights[flights]
    flights_src --> flights[flights]
    
```

- The lineage shows
 - [blue box] flights data file residing in an s3 folder

- [purple box] is showing how the flights_csv Hive table is created, this table was created and points to the data location of flights' (blue box) s3 folder
- [orange box] is showing the flights Iceberg table and how it is created, it uses data from flights_csv Hive table (CTAS)
- Traffic_cancel_airlines is a Materialized View that uses data from the flights Iceberg table.



- Click on the Schema Tab to see Metadata on this table. The Metadata includes basic information from the table such as: column names and data types for the columns. You can also run Profilers that come as part of CDP:
 - Hive Table Profiler - allows you to gather additional information from the table including Min/Max values in the column, # records with Null Values in a column, etc.
 - Sensitivity Profiler - identifies columns that may contain sensitive information such as PII data, SSN, credit card numbers, ID numbers, etc. These items will be "Tagged" in the Classification column of the Schema page.

Schema									
Chart Type	Name	Type	Unique Values *	Null Values	Max	Min	Mean	Comment	Classifications
	actualElapsedTime								
	airtime								
	arrdelay								
	arrtime								
	cancellationcode								
	cancelled								
	carrierdelay								
	crsuntime								

- Click on the Policy tab to see what security policies have been applied on this table. There are 2 policies that have been defined to allow some users & groups access via policy “all - database, table, column” and another policy “all - database, table” access.
 - The Access Audits tab allows Administrators to be able to see who, when, where, why either accessed this table or was denied access to this table. Feel free to switch to this tab to take a look and switch back to the Policy tab.

Policy					
Resource Based Policies					
Policy ID	Policy Name	Status	Audit Logging	Group	Users
9	all - database, table, column	ENABLED	ENABLED	_c_ranger_admins_7925da73	hive, beacon, dpprofiler, hue, admin, impala, r...
10	all - database, table	ENABLED	ENABLED	_c_ranger_admins_7925da73	hive, beacon, dpprofiler, hue, admin, impala, r...

Tag Based Policies

- Click on the next to the “all - database, table” - to modify this policy or create a new policy

10	all - database, table
----	-----------------------

19. Security (Ranger) - modify and create security policies for the various CDP Data Services.

- View the Policy details and click the CANCEL button at the bottom.
- For this portion of the Lab let's restrict your access to a subset of data available in the "flights" table.

The screenshot shows the Ranger Service Manager interface with several service sections: HDFS, HBASE, HADOOP SQL, YARN, KNOX, and SOLR. Each section has a list of resources or tables with edit and delete icons. The HADOOP SQL section is currently selected.

- Click on the Hadoop SQL link in the upper left corner - to view the security policies in place for CDW. For this Workshop we will stick to the CDW related security features

The screenshot shows the Hadoop SQL Policies page with a table of existing policies. The columns include Policy ID, Policy Name, Policy Labels, Status, Audit Logging, Roles, Groups, Users, and Action. Most policies listed are for 'ranger_jdbc' and 'ranger_jdbc_public' roles.

- This screen shows the general Access related security policies - who has access to which Data Lakehouse databases, tables, views, etc. Click on the "Row Level Filter" tab to see the policies to restrict access to portions of data

The screenshot shows the Hadoop SQL Policies page with the Row Level Filter tab selected. The table now includes an additional column for Row Level Filter conditions. The 'ranger_jdbc' policy is shown with a filter condition of 'uniquecarrier="UA"'.

- There are currently no policies defined. Click on the Add New Policy button

The screenshot shows the Hadoop SQL Policies page with the 'Add New Policy' button highlighted in blue.

- Fill out the form as follows. For this policy let's say you are an Gate Agent for United Airlines and should not be able to see data for any other Airline. To setup the policy we need to apply a filter that is applied to any query that is run for your <user_id> so you only see rows for flights run by United Airlines.

- Policy Name: <user_id> Workshop Row Level Filter
- Hive Database: <user_id>_airlines (start typing, once you see this database in the list, select it)
- Hive Table: flights (start typing, once you see this table in the list, select it)
- Row Level Filtering Conditions
 - Select User - <user_id> (start typing, once you see this user in the list, select it)
 - Row Level Filter - **uniquecarrier="UA"**
 - For this you could have any number of filter conditions (for this Lab we will only create 1) with many different filter configurations
- Click **Add** button to accept this Policy

Please ensure that users/groups listed in this policy have access to the table via an Access Policy. This policy does not implicitly grant access to the table.

Policy Details:

- Policy Type: Row Level Filter
- Policy Name: workshop Row Level Filter
- Policy Label: Policy Label
- Hive Database*: user001.airlines
- Hive Table*: flights
- Description:
- Audit Logging: Yes

Row Filter Conditions:

Select Role	Select Group	Select User	Access Types	Row Level Filter
Select Roles	Select Groups	x jngalls x user001	select	uniquecarrier='UA'

- The new policy is added to the “Row Level Filter” policies (as below)

Policy ID	Policy Name	Policy Labels	Status	Audit Logging	Roles	Groups	Users	Action
395	workshop Row Level Filter	--	Enabled	Enabled	--	--	jngalls user001	

- Test the policy is working - Open HUE for the CDW **Impala** Virtual Warehouse - **airlines-impala-vw** and execute the following query

```
SELECT uniquecarrier, count(*)
FROM ${user_id}_airlines.flights
GROUP BY uniquecarrier;
```

You should now only see 1 row returned for this query - after the policy was applied you will only be able to access uniquecarrier = “UA” and no other carriers:

uniquecarrier	count(*)
UA	17642768