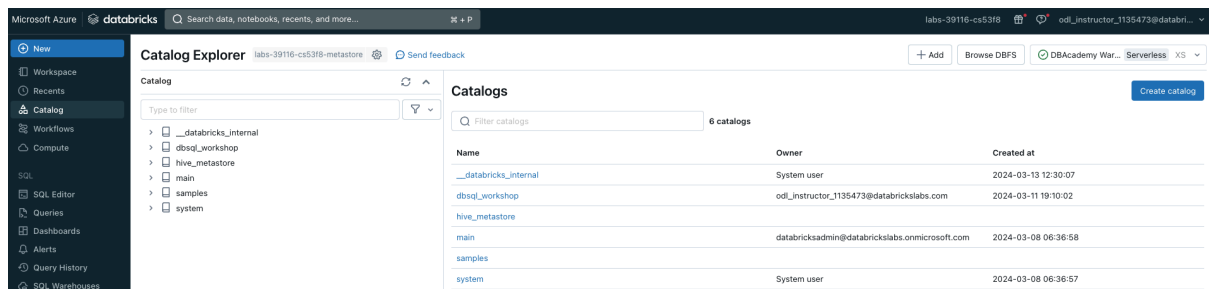


Set up - write down your odl user address (click on the O on the top right when logged in your workspace)

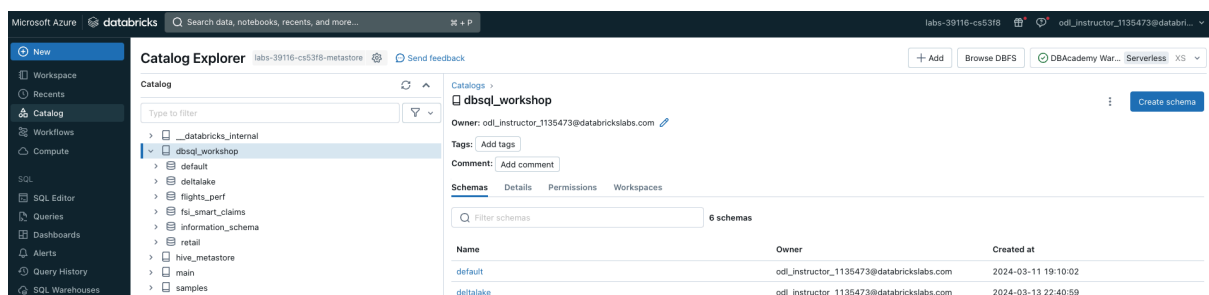
Lab 1 - Unity Catalog Ali Azzouz

Part 1 - Create a catalog and a schema

1. First, we need to create our catalog ! To do so:
2. Create the catalog from the UI
 - a. Click on Catalog
 - b. Click on the “Create catalog” button on the right
 - c. Enter the name of your catalog such as “yourfirstname_yourlastname” and click create



3. Go back to the Catalog page, click on the refresh button, your new catalog should be there
4. Now, let's create our first schema/database:
 - a. Click on Catalog
 - b. Click on the “Create schema button on the right
 - c. Enter the name of your schema such as “dbsql_workshop” and click create
5. Go back to the Catalog page, click on the refresh button, click on your catalog, your new schema should be there



Part 2 - Create a volume and a table

1. First, download the airport and airlines csv files from the github repo
2. As for the catalog and the schema, you can create a volume in the UI.
 - a. Go to your catalog and click on your new schema
 - b. Click on Volumes
 - c. To create a volume from the UI, you can click on the “Create” button and select “Create volume”

- d. Type the “data_files” as the name of your volume, select “Managed Volume” and click create
- e. Click on “Upload to this volume”
- f. Select and upload the csv files you have downloaded from the repo
- g. Click Upload
- h. Your new files should be there as shown below

Catalog Explorer > ali_azzouz > dbsql_workshop >

ali_azzouz.dbsql_workshop.data_files ☆ ⋮ [Upload to this volume](#)

Overview Details Permissions

/Volumes/ali_azzouz/dbsql_workshop/data_files

Name	Size	Last modified	
airlines.csv	40.09 KB	42 minutes ago	⋮
airports.csv	296.89 KB	42 minutes ago	⋮

1

About this volume

Owner: odl_instructor_1135473@databricksla...

Tags: [Add tags](#)

Comment: [Add comment](#)

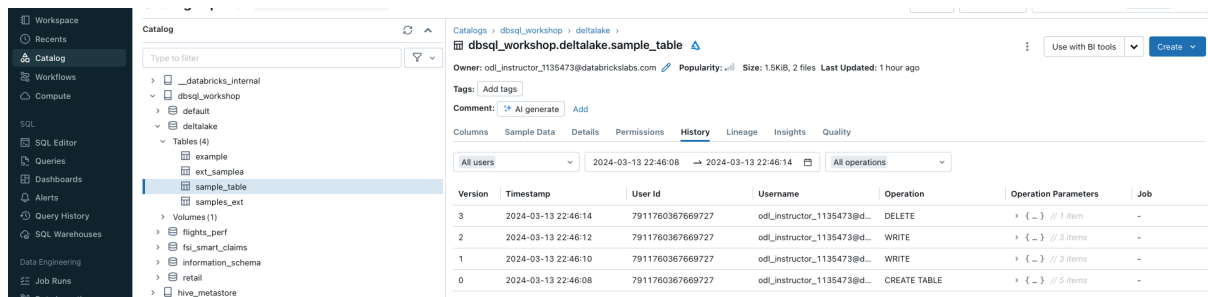
3. As for the catalog and the schema, you can create a table in the UI.
 - a. To create a table from the UI and from the files you have uploaded in your volume, you can click on the three dots on the right of airlines.csv and, click on “Create table”
 - b. Specify your catalog “yourfirstname_yourlastname”, your schema “dbsql_workshop” and “airlines” as the name of your table

Name	Size	Last modified	
airlines.csv	40.09 KB	an hour ago	⋮
airports.csv	296.89 KB	an hour ago	⋮

Comment: [Add comment](#)

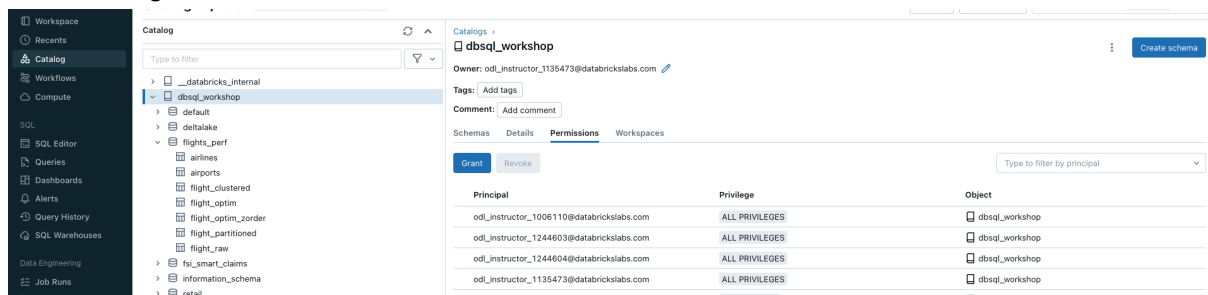
Copy path
Download file
Delete file

4. Once you have created your table, you can go back on the Catalog page, click on your catalog, click on your schema, you should see the name of your table
5. Click on the table and visit the different tabs (columns, sample data, details, permissions,...) You will notice that in the Details page, the type of the table is Managed



Part 3 - Grant access to your catalog/schema/tables

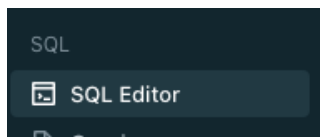
1. You can grant access to your data assets via the UI:
 - a. Go back to the catalog page
 - b. Click on your catalog name “yourfirstname_yourlastname”
 - c. Click on on the Permissions tab
 - d. Click on the Grant button
 - e. In the principal section, add the email address of your teammate (the email should start by odl_user)
 - f. Grant him/her some privileges (you can choose some privilege presets as data reader or data editor)
 - g. Click Grant



2. Once you have granted access to your catalog, you can go back on the Catalog page and check if you have access to your teammate’s catalog and check if he/she has access to your catalog.
3. Note you can grant access to Catalog, schemas, tables, views, functions, models, volumes

Lab 2 - Dataviz + Dashboard basics Eugénie Vinet


1. Go to SQL -> SQL Editor



2. Click on the + on the right - Create a new query

Time for some SQL !

3. Let's leverage our `flights_perf` database;

- a. Click on  to see your catalog
- b. Open the `flights_perf` database within the `dbsql_wokshop` catalog.
Here the goal is to know what are the worst airports in terms of % of cancelled flights over all years. Let's take a look at our two tables `flight_optim_zorder` and `airports`.

We will need to join our two tables on `flight_optim_zorder.Origin = airport.iata` to get the airport name and lat long.

4. To write your query, you don't have to be a SQL expert, let's leverage our Databricks Assistant to help us !

- a. Go to the Databricks Assistant:  and write this natural language query :

"Show me the top 30 airports with the highest percentage of canceled flights. For each airport, include its identifier (iata), name, location (latitude and longitude), total number of flights, number of canceled flights, and the percentage of cancellations. Use data from both the flight and airport tables, joining them on the appropriate columns."

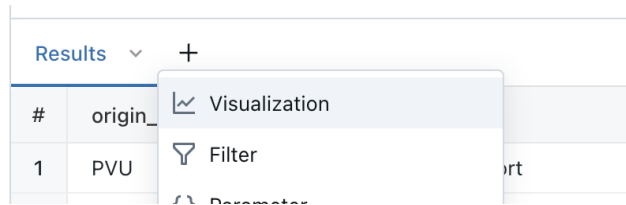
- b. The Databricks Assistant should output the following query or similar. You might have to correct some of the fields (latitude, longitude, etc)

```
SELECT
  a.iata AS origin_airport,
  a.name AS airport_name,
  a.lat AS airport_latitude,
  a.lon AS airport_longitude,
  SUM(f.cancelled) AS cancelled_flights,
  COUNT(*) AS total_flights,
  (SUM(f.cancelled) / COUNT(*)) * 100 AS cancellation_percentage
FROM
  dbsql_workshop.flights_perf.flight_optim_zorder f
JOIN
  dbsql_workshop.flights_perf.airports a
ON
  f.Origin = a.iata
GROUP BY
  f.Origin, a.name, a.lat, a.lon
```

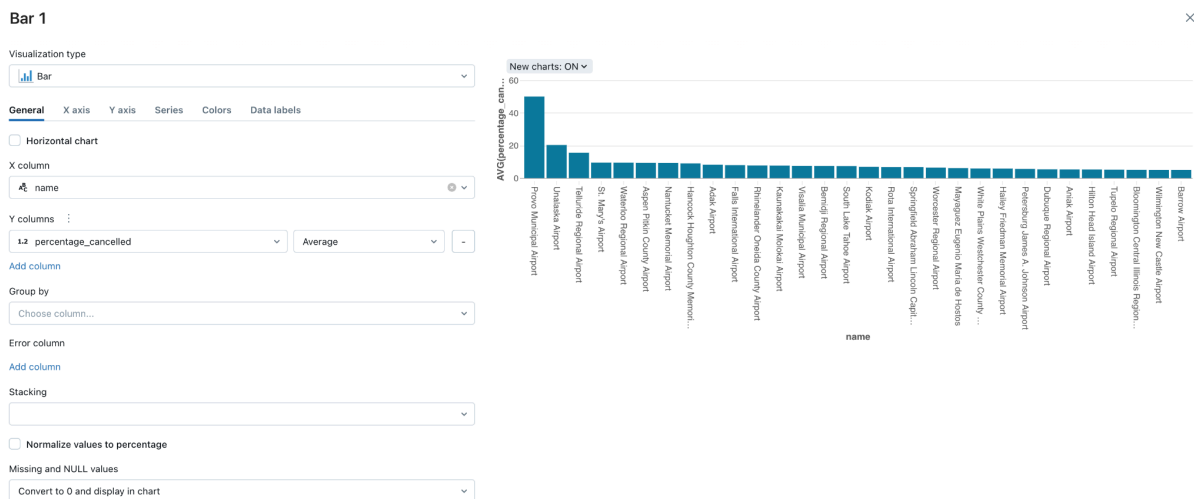
```
ORDER BY
    cancellation_percentage DESC
LIMIT
    30;
```

5. Visualization of the query result:

- a. Click on visualize on the + at the right

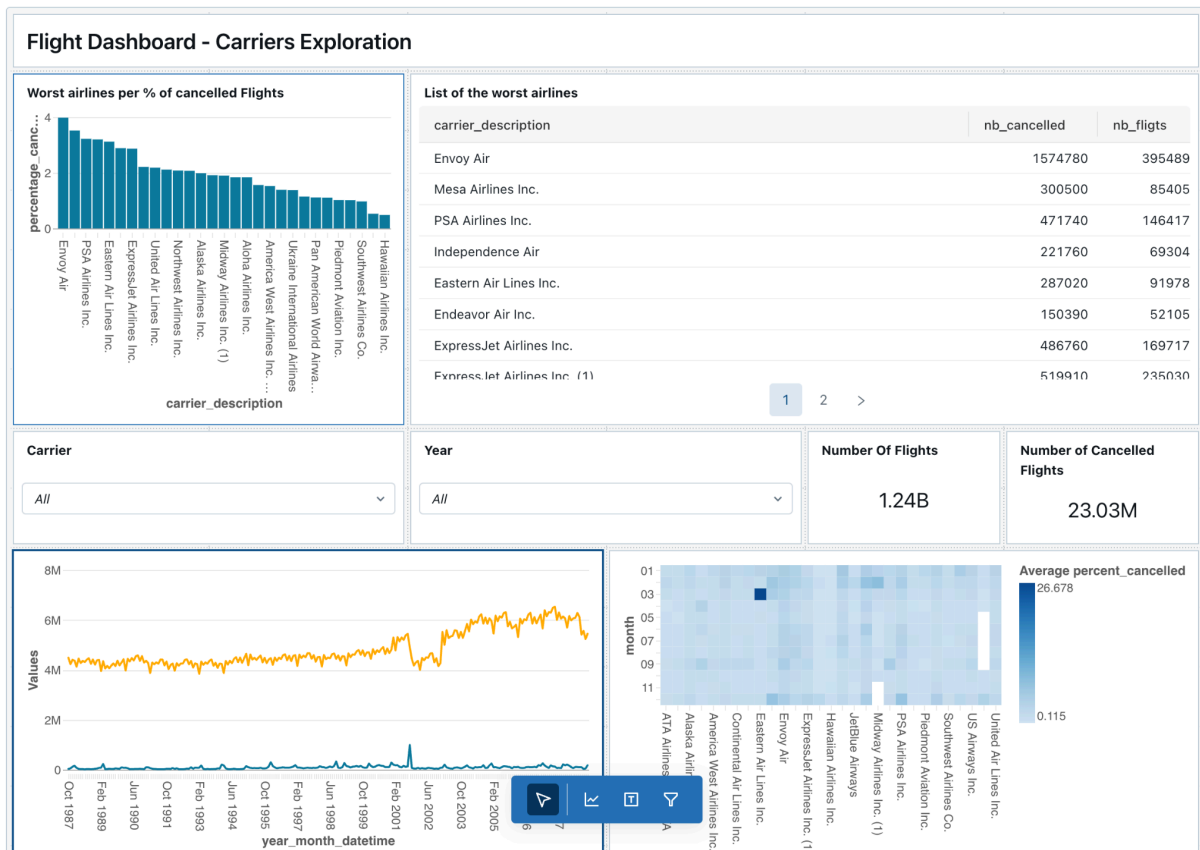


- Choose a Bar Visual
- Select airport_name as the X column
- Select cancellation percentage - AVG as Y
- Click on validate



Lab 3 - Lakeview Eugénie Vinet

Let's now create a Lakeview dashboard. The goal is to have this kind of dashboard at the end :



1. Create a Dashboard :

- Go to Dashboard: **Dashboards**, click on **Lakeview Dashboards** **New**
- Click on **Create Lakeview dashboard** on the top right
- Click on the data tab

2. Create the query we will be using:

- Click on **Create From SQL** and enter :

```
SELECT
    UniqueCarrier AS carrier_code,
    airlines.Description AS carrier_description,
    SUM(cancelled) AS nb_cancelled,
    SUM(1) AS nb_flights,
    (SUM(cancelled) / COUNT(*)) * 100 AS percentage_cancelled
FROM dbsql_workshop.flights_perf.flight_optim_zorder
JOIN dbsql_workshop.flights_perf.airlines
ON flight_optim_zorder.UniqueCarrier = airlines.UniqueCode
GROUP BY carrier_code, carrier_description
ORDER BY percentage_cancelled DESC
```

- Click on **Run**
- Name it **Cancelled Flights - By Carriers - Aggregated**

3. Second query :

- Click on **Create From SQL** and enter:

```
SELECT
```

```

year,
month,
date(CONCAT(year, '-', month, '-01')) AS year_month_datetime,
UniqueCarrier AS carrier_code,
airlines.Description AS carrier_description,
sum(cancelled) AS nb_cancelled,
sum(1) AS nb_flights,
(sum(cancelled) / sum(1)) * 100 AS percent_cancelled
FROM dbsql_workshop.flights_perf.flight_optim_zorder
JOIN dbsql_workshop.flights_perf.airlines
ON flight_optim_zorder.UniqueCarrier = airlines.UniqueCode
WHERE isnotnull(flight_optim_zorder.Cancelled)
AND isnotnull(flight_optim_zorder.year)
GROUP BY year, month, year_month_datetime, carrier_code, carrier_description

```

- b. Click on *Run*
 - c. Rename it *Cancelled flights By Carriers - By Months*
4. Click on Canvas
5. To create a text box, click on *Add a text box* on the bottom. You can use markdown for formatting.
6. In the first Visualization, we want to see the worst airlines, visualizing it by descending order of % cancellations of flights:
 - a. Click on Add a visualization
 - b. Select Cancelled Flights - By Carriers - Aggregated as Dataset on the right
 - c. X_axis : carrier_description
 - d. Y_axis: percentage canceled
 - e. Click on the 3 dots beside X_axis
 - f. Order By Descending Order by Y_axis :
7. Similarly, create a table listing the worst airlines :

By Y axis



Dataset

Cancelled Flights - By Carriers - ...

Visualization

Table

☒ Title
☐ Description

Columns
Grid

☐ Display row number

carrier_code	
carrier_description	
nb_cancelled	
nb_flights	
percentage_cancel...	

8. Create a filter for carrier :
 - a. Click on *Add a filter* on the bottom of your screen
 - b. Filter on single value on Cancelled flights by Carriers - By Months
 - c. Choose `carrier_description`
9. Similarly, create a filter for year:

Filter

☒ Title ☐ Description

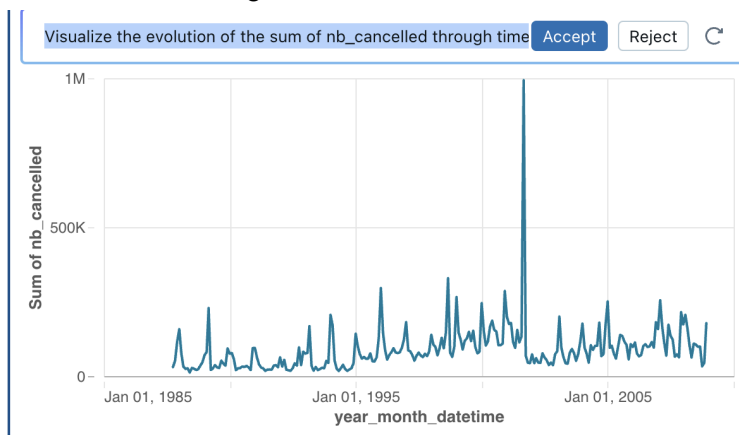
Filter on +

1² 3 Cancelled flights By Carri... —

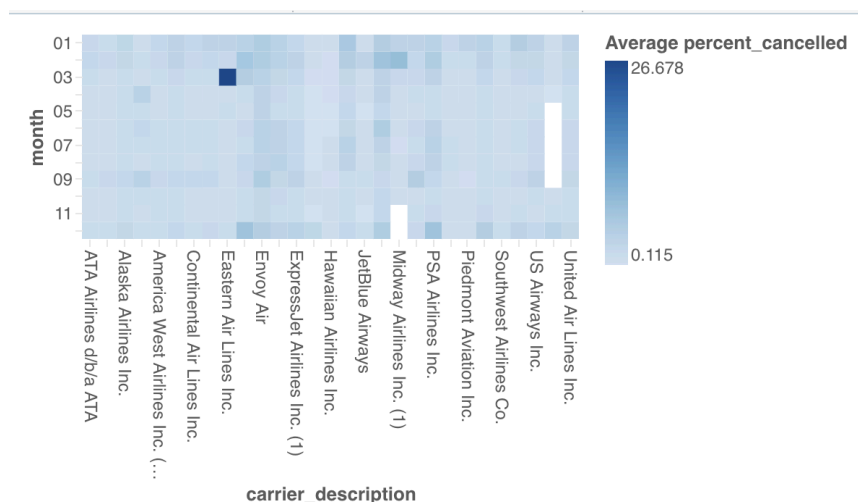
Filter Settings

☒ Allow All

10. You can then create some counters for the number of flights and cancelled flights:
 - a. `SUM(nb_flights)`
 - b. `SUM(nb_cancelled)`
11. Let's leverage AI capabilities to create a line chart. Write: *"Visualize the evolution of the sum of nb_cancelled through time (year_month_datetime) in a line chart"* You should see this diagram :



12. Last, create a heatmap visualization to visualize the percentage of canceled flights as a function of Month and carriers to identify if canceling flights is more frequent in some periods of the year. You should see something like this :



13. Click on publish to publish your Dashboard and you are done! Take a look at your artwork by clicking on the “published” version on the top.


Lab 4 - Genie Spaces - AI/BI Eugénie Vinet

Now, imagine your business users want to explore the data and extract insights and KPIs from it without being proficient with SQL.

They have many specific questions that your dashboard cannot answer, and you don't have the bandwidth to answer each one.

Good news, now with Databricks, you can leverage Genie for that and empower them to be more autonomous!


1. Create a Genie Space

- a. Go to  Genie under the SQL tools on the left menu
- b. Click on *New* on the top right
- c. Give it a meaningful title :
Flight Genie Space - YourFirstName YourLastName
- d. Select a warehouse
- e. Choose the right table to expose to your users. Here you want to expose only the gold data to your end users :
 - `dbsql_workshop.flights_perf.flight_optim_zorder`
 - `dbsql_workshop.flights_perf.airports`
 - `dbsql_workshop.flights_perf.airlines`
- f. Click on *Save*

2. Your Genie Space is ready to use and test. Let's test it with simple questions:

- a. Let's ask a first question :
“What is the average arrival delay for flights operated by a specific airline carrier in the year 2000?”
- b. Let's now ask: What is the month-over-month growth rate of delayed departures for Colorado Springs Airport?
- c. You can see generated sql request by clicking on “Show Generated Code”
- d. Click on “Auto Visualize”

3. Let's ask for more advanced questions.

- a. For that you need to set up instructions. Click on  Instructions on the left pane and copy as General Instructions :

- * `Origin` corresponds to the `iata` code of the airport of origin
- * `Dest` corresponds to the `iata` code of destination airport
- * `airport`, `origin` or `dest`, should always be displayed by name and not `iata` code

* UniqueCarrier in table flight_optim_zorder corresponds to UniqueCode in the airline table
* A delayed flight at arrival is a flight with more than 5 mins delay at the destination airport
* When asked about the worst airport, ask to clarify whether it is in terms of delay or in terms of proportion of cancelled flight
* a major airport is an airport with more than 100000 flights per year

b. Let's write some more advanced questions to the tool :

- Which are the 10 airports that have, on average, the highest number of delayed arrival flights (evaluate the number of delayed flights over the year)? What is the average duration of the delay?

We have some results, and it takes into consideration the instruction about a flight being delayed when the delay is > 5 minutes, but we see the majority of the biggest airports with the highest number of flights.

- What do we get if we analyze the proportion of delayed flights? Include the absolute number of departing flights.

It is better, but we see also the smallest airports, let's select only the major airports

- Give me only the major airports
(See instructions: Major airports have more than 100,000 flights a year)

Lastly, ask for the top 10 worst airports.

4. Feel free to explore with new questions and instructions!

Lab 4 - AI Integration Jérôme Ivain

NOTA: <your_catalog>.<your_schema> needs to be replaced with your own catalog and schema

From the Queries menu, we will create several queries to test the different capabilities that DBSQL offers in terms of AI integration.

1) Call a Built-in AI function (*Create a new Query from the menu*)

Test the following examples :

```
-- to mask specified entities in a given text using SQL.
SELECT ai_mask(
    'John Doe lives in New York. His email is john.doe@example.com. Contact
him at 555-1234',
    array('person', 'address', 'email', 'phone number')
);

-- to classify input text according to labels you provide using SQL
SELECT ai_classify(
    'My password is leaked.',
    array('urgent', 'not urgent')
);

-- to perform sentiment analysis on input text using SQL
SELECT ai_analyze_sentiment('The dinner was awesome but the waiter was
rude');

-- to answer the user-provided prompt using SQL
SELECT ai_gen('Generate a concise, cheerful pitch for Databricks');
```

The full list of AI Functions can be found here:

<https://docs.databricks.com/en/large-language-models/ai-functions.html>

2) Call a Foundation Model (LLM) hosted on Databricks (Create a new Query from the menu)

Create a Query using Mixtral-8x7b-Instruct model:

```
SELECT ai_query('databricks-mixtral-8x7b-instruct',  
               "What is Databricks?")
```

3) Create a SQL Function calling a Foundation Model (LLM) (Create a new Query from the menu)

- Create a function defining the following ai_query:

```
CREATE OR REPLACE FUNCTION <your_catalog>.<your_schema>.summarize(name  
STRING)  
RETURNS STRING  
RETURN ai_query(  
    'databricks-dbrx-instruct',  
    CONCAT('Résumé de l histoire de la compagnie aérienne ',  
           Name,  
           ' en 20 mots en français'));
```

Now the function is available in Unity Catalog at <your_catalog>.<your_schema>

- Create a table that will store the result of the call of the above function:

```
CREATE OR REPLACE TABLE <your_catalog>.<your_schema>.summarize AS  
SELECT UniqueCode, Description as name,  
<your_catalog>.<your_schema>.summarize(Name) AS summary FROM  
dbsql_workshop.flights_perf.airlines  
limit 10;
```

Now the table is available in Unity Catalog at <your_catalog>.<your_schema>

- Query the table you have just created to see the result:

```
SELECT * FROM <your_catalog>.<your_schema>.summarize;
```

4) Query a Custom Model serving endpoint *(Create a new Query from the menu)*

- Find out the ML model already registered in Unity Catalog under `dbsql_workshop.retail.dbdemos_customer_churn`
- Test the following examples:

```
-- Customer churning from Spain on Mobile, Male and following
characteristics:
SELECT ai_query('dbdemos_customer_churn',
    named_struct("user_id","4ce55564-5f57-4b92-a9a8-be20bf02f0b6",
"canal","MOBILE", "country","SPAIN", "gender",0, "age_group",6,
"order_count",3, "total_amount",215, "total_item",7,
"last_transaction","2024-03-05T06:24:49", "platform","other",
"event_count",3, "session_count",3, "days_since_creation",253,
"days_since_last_activity",6, "days_last_event",2),
    returnType => 'STRING') AS predicted_churn;
```

The result should show 1, meaning a customer with the corresponding characteristics will churn

```
-- Non churning customer from Spain on Web, Female and following
characteristics:
SELECT ai_query('dbdemos_customer_churn',
    named_struct("user_id","b662ce5f-6294-4831-ae8b-33aecfa7d6e9",
"canal","WEBAPP", "country","SPAIN", "gender",1, "age_group",5,
"order_count",2, "total_amount",124, "total_item",4,
"last_transaction","2024-03-04T21:28:14", "platform","ios", "event_count",2,
"session_count",2, "days_since_creation",89, "days_since_last_activity",11,
"days_last_event",10),
    returnType => 'STRING') AS predicted_churn;
```

The result should show 0, meaning a customer with the corresponding characteristics will not churn

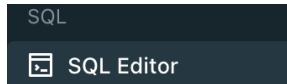
5) Query an External Model *(Follow Along)*

Here, it's all about calling an endpoint that is not hosted on the Databricks platform. We will be calling an Azure Open AI function.

Lab 5 - Observability Jérôme Ivain

Part 1 : Query the Billing table to get all spending above a certain threshold {{ budget }} for serverless compute

1. Go to the query editor
2. Create a new Query and copy :

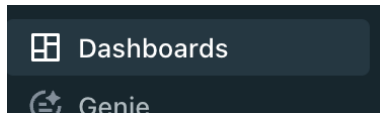


```
SELECT
  t1.workspace_id, t1.billing_origin_product,
  SUM(t1.usage_quantity * list_prices.pricing.default) as list_cost
FROM system.billing.usage t1
INNER JOIN system.billing.list_prices on
  t1.cloud = list_prices.cloud and
  t1.sku_name = list_prices.sku_name and
  t1.usage_start_time >= list_prices.price_start_time and
  (t1.usage_end_time <= list_prices.price_end_time or list_prices.price_end_time is
null)
WHERE
  t1.sku_name LIKE '%SERVERLESS%'
  AND t1.usage_date >= CURRENT_DATE() - INTERVAL 30 DAYS
GROUP BY
  t1.workspace_id, t1.billing_origin_product
HAVING
  list_cost > {{ budget }}
ORDER BY t1.workspace_id
```

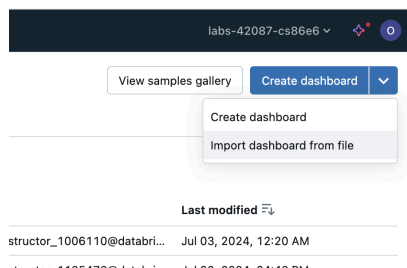
Part2 : Import the Account Usage Dashboard.

1. From the git repository, download the the dashboard definition file "account_usage_dashboard.lvdash.json" **to your computer**
2. Import the dashboard in lakeview AI/BI dashboards

Go to Dashboards



Import using 'Create dashboard' (Top right hand corner)



Result

