Using MotherDuck: a managed DuckDB-in-the-cloud service

1. Let's read data from md's shared sample db. Paste this sql command into UI & run

```sql
SELECT
passenger_count,
avg(total_amount)
    FROM sample_data.nyc.taxi
        GROUP BY passenger_count
        ORDER by passenger_count;
```

```sql
SELECT
  created_date,
  agency_name,
  complaint_type,
  descriptor,
  incident_address,
  resolution_description
FROM
  sample_data.nyc.service_requests
WHERE
  created_date >= '2022-03-27'
  AND created_date <= '2022-03-31';
```

```sql
SELECT
  passenger_count,
  avg(total_amount)
FROM
  's3://us-prd-motherduck-open-
datasets/nyc_taxi/parquet/yellow_cab_nyc_2022_11.parquet'
GROUP BY
  passenger_count
ORDER BY
  passenger_count;
```

2. Let's add our own data to analyse! Create a new database

    ```sql
    CREATE DATABASE tennis;
    ```

    Create table from files; Database & schema: 'tennis'; Create table from files

Upload a csv file to md. Switch db in a cell db selection field. Run sql.

```sql
select * from tennis.main.results limit 10;
```

3. Share your db (keep proposed name)

```
-- Run this snippet to attach database
ATTACH 'md:_share/tennis/79f9d605-5cdd-4fcf-bf93-23b9a51ea747';
```

4. Let's automate authentication to our Cloud-based md. Motherduck UI -> Settings -> Integrations -> Access Tokens -> Create token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImFudG9uaW1hcnJvLnluNjVwQHNsbWFpbC5tZSIsInNlc3Npb24iOiJhbnRvbmltYXJyby55bjY1cC5zbG1haWwubWUiLCJwYXQiOiJKNnRRYUlQZU5GWWVKOG9yRXliM3I0R1EyUGJzbUpzaUlIcjVpUnRFWHQ0IiwidXNlcklkIjoiNmNjN2MzNTktNTcxMC00Mjc2LWI3MmQtYjZlNDlkMGY3NzlkIiwiaXNzIjoibWRfcGF0IiwicmVhZE9ubHkiOmZhbHNlLCJ0b2tlblR5cGUiOiJyZWFkX3dyaXRlIiwiaWF0IjoxNzU2ODIzNDQ3LCJleHAiOjE3NTgwMzMwNDd9.K0hYl6-jD8SIQvnMkxqS1w0YF9v8iknfG3UAk-AFMSc

~~~
Then:
```shell
export
motherduck_token='eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImFudG9
uaW1hcnJvLnluNjVwQHNsbWFpbC5tZSIsInNlc3Npb24iOiJhbnRvbmltYXJyby55bjY1cC5zb
G1haWwubWUiLCJwYXQiOiJKNnRRYUlQZU5GWWVKOG9yRXliM3I0R1EyUGJzbUpzaUlIcjVpUnR
FWHQ0IiwidXNlcklkIjoiNmNjN2MzNTktNTcxMC00Mjc2LWI3MmQtYjZlNDlkMGY3NzlkIiwia
XNzIjoibWRfcGF0IiwicmVhZE9ubHkiOmZhbHNlLCJ0b2tlblR5cGUiOiJyZWFkX3dyaXRlIiw
iaWF0IjoxNzU2ODIzNDQ3LCJleHAiOjE3NTgwMzMwNDd9.K0hYl6-
jD8SIQvnMkxqS1w0YF9v8iknfG3UAk-AFMSc'
```
~~~

5. Test automatic authentication with:

```
duckdb "md:tennis"
.tables
```

The 'results' table we created earlier should be on the list.

6. Query shared Cloud-based db from Locally installed duckdb

```
duckdb
.open md:
```

```
show databases;
DETACH sample_data;
use tennis;
.tables
```

7. Additional ideas that are worth studying:

- uploading data from s3 massively
- Hybrid workflows: combining locally & Cloud stored data

Extra: some additional MD's features worth exploring!

8. Sharing snapshots of remote databases you created

```
CREATE SHARE shared_tennis FROM tennis;
```

Share the link received with your Team mates. They should **attach** the remote database you shared with them like this:

```
ATTACH 'md:_share/shared_tennis/58b1023f-f618-4169-ad65-8744340bcc3d' AS
shared_tennis;
```

In case you'd like to see the shares you previously created run:

```
LIST SHARES;
```

Let's say you apply some changes to your tennis database. For example:

```
update tennis.main.results set winner_ioc=lower(winner_ioc) where 1=1;
```

You now have to update the DB's snapshot you shared earlier so the changes become visible to your peers:

```
UPDATE SHARE shared_tennis;
```

OR

Database -> ⋮ -> Share -> 'Auto-updating': 'ON'.

Getting rid of the share that you no longer need is also easy:

```
USE tennis;
DETACH shared_tennis;
```

9. Using Generative AI to query data: harnessing the power of LLMs for DuckDB

Let's first attach a data base with a number of interesting datasets into our MotherDuck instance:

```
ATTACH 'md:_share/share_sample_data/23b0d623-1361-421d-ae77-62d701d471e6'
AS sample_data;
USE sample_data;
```

We'll be using World Health Organisation's database on air quality index:

```
select * from sample_data.who.ambient_air_quality limit 10;
```

Here comes AI!

```
pragma prompt_query('What is the second most popular city in
who.ambient_air_quality table?');
```

Let's verify that:

```
select city, count(*) as num
    from who.ambient_air_quality
        GROUP BY city order by num desc;
```

How did AI do that?

```
call prompt_sql('What is the second most popular city in
who.ambient_air_quality table?');
```

Let's feed it a more challenging task:

```
pragma prompt_query('Show me city with the highest pm25 concentration in
2018 and with population of more than 10 million people');
```

```
call prompt_sql('Show me city with the highest pm25 concentration in 2018
and with population of more than 10 million people');

select city from who.ambient_air_quality where year='2018' and
pm25_concentration=(select max(pm25_concentration) from
who.ambient_air_quality where year='2018') and population>10000000;
```

10. Have you noticed the little AI-powered 'auto-fix' tool in the upper right corner of code cells?

```
call prompt_sq('Show me city with the highest pm25 concentration in 2018
and with population of more than 10 million people');
```

11. Let's add some Vibe-coding.

When working on a SQL-query, press Crtl + K (Cmd + K on Mac) in the target area of your query & insert your desired prompt.

```
SELECT city
FROM who.ambient_air_quality /*click for a prompt insert HERE! prompt:
'only list cities ending with letter a'*/
WHERE "year" = 2018 AND pm25_concentration = (
    SELECT MAX(pm25_concentration)
    FROM who.ambient_air_quality
    WHERE "year" = 2018 AND population != 'NA' AND population::BIGINT >
10000000
) AND population::BIGINT > 10000000;
```