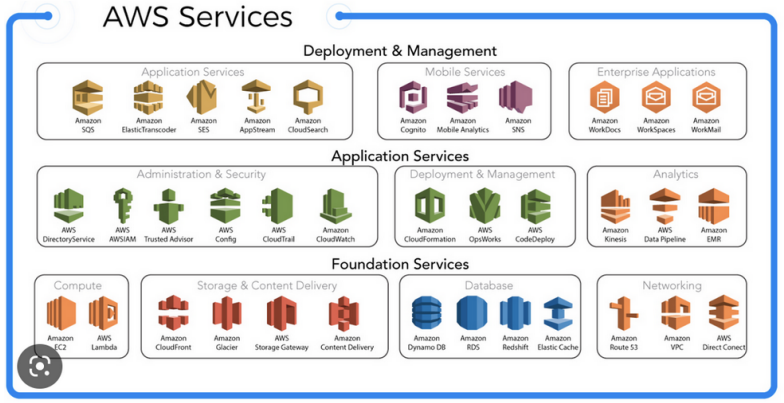
DuckDB - What's all the hype!

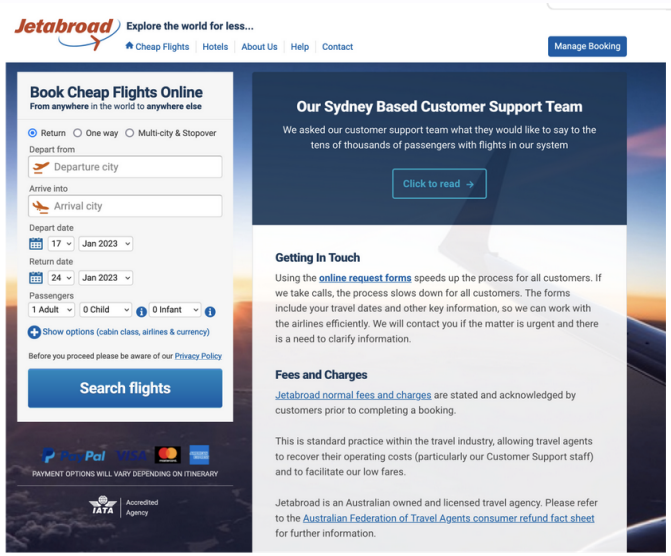
An intro to using DuckDB with Python

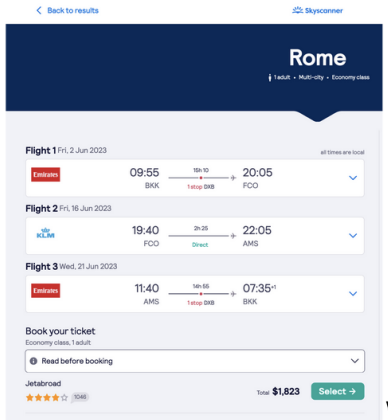
https://ap76ap.github.io/thaipy-duckdb/slides/

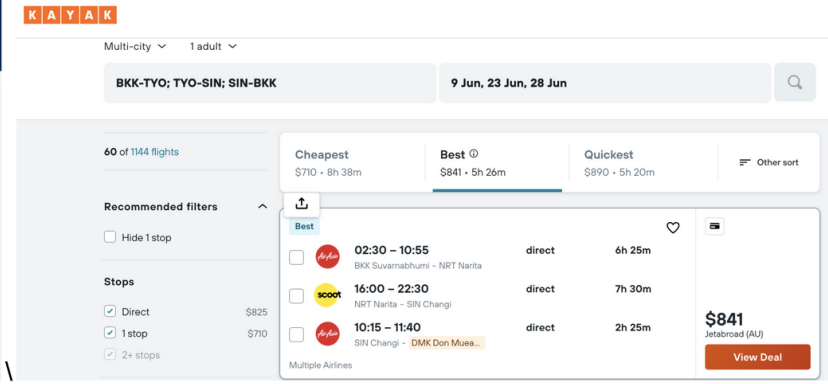
- About Me
- DuckDB - What is it?
- Install DuckDB
- Level One : CSV
- Data Lake Storage - Super Quick Primer
- Level Two : Parquet
- Level Three : Parquet - Against Cloud Storage
- Other things to explore

About Me - Andrew Purser

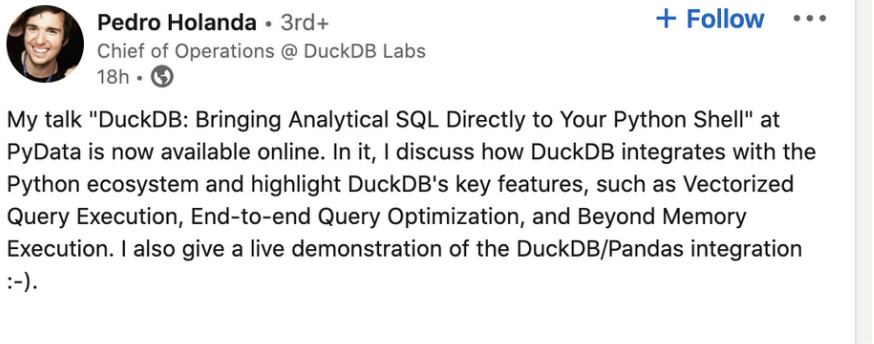


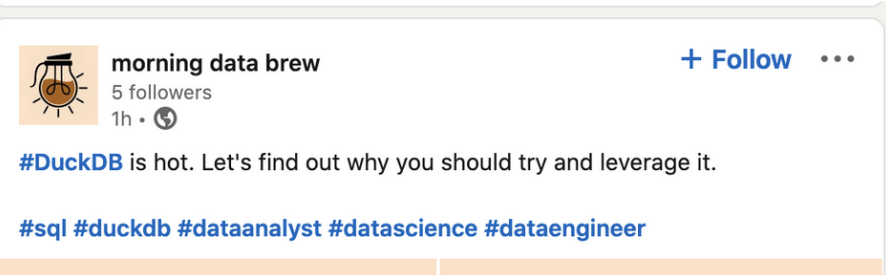


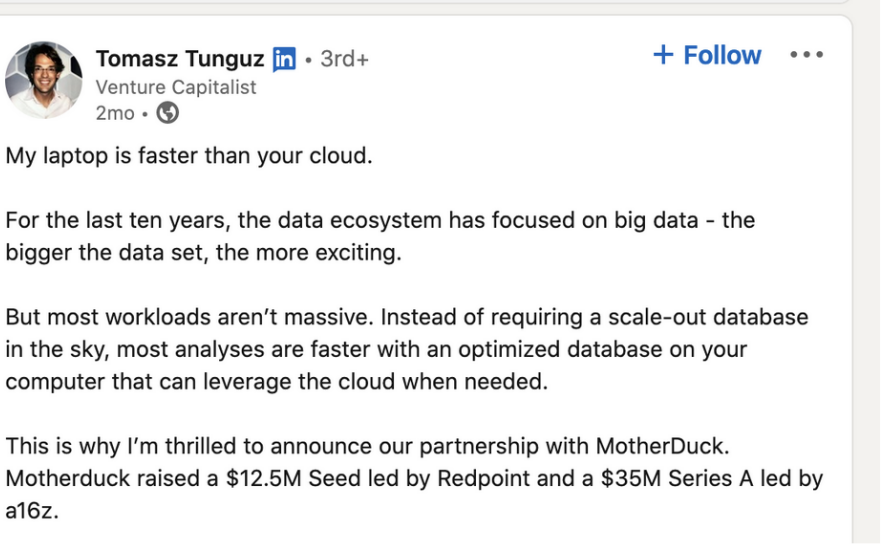


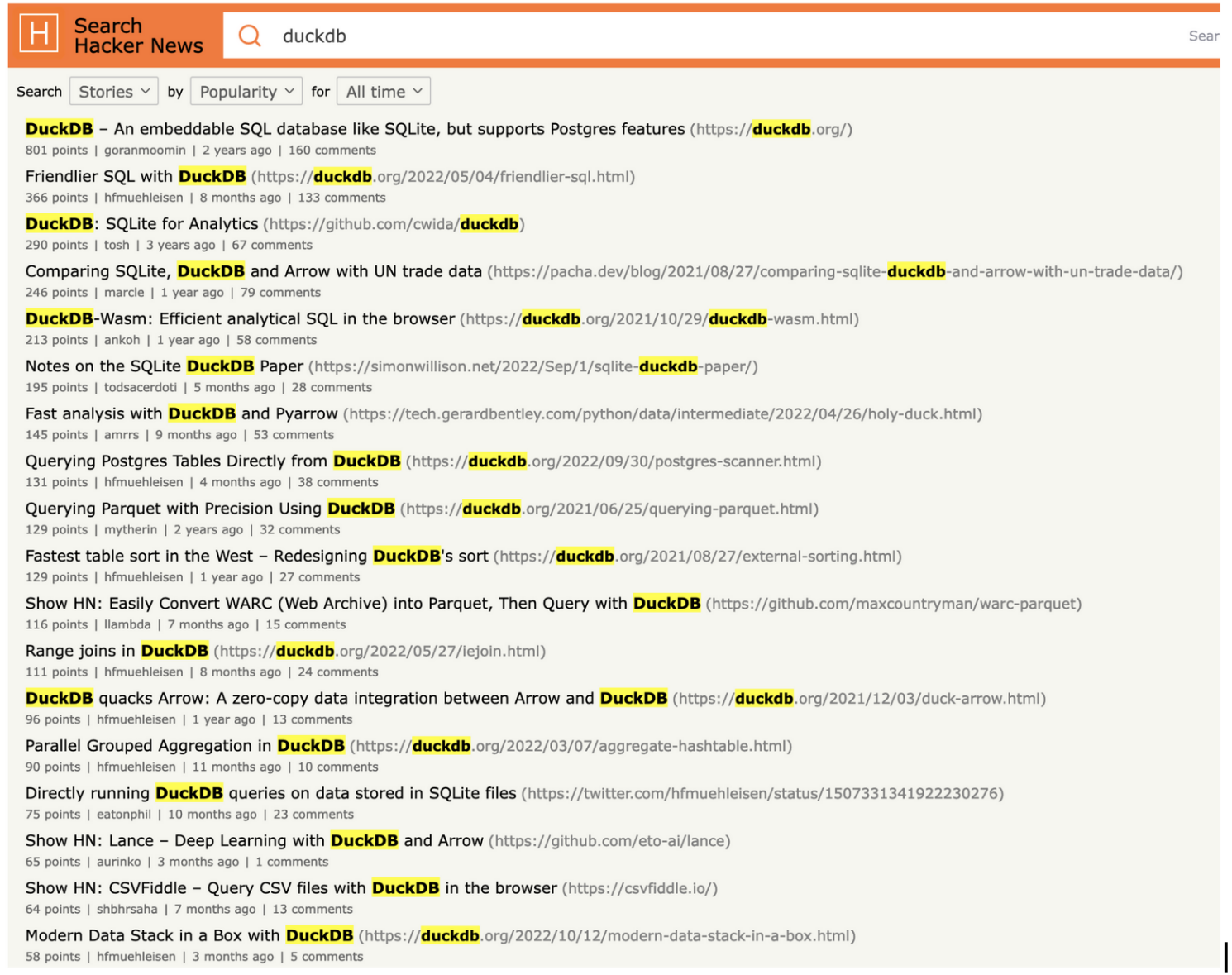


DuckDB - What is it?

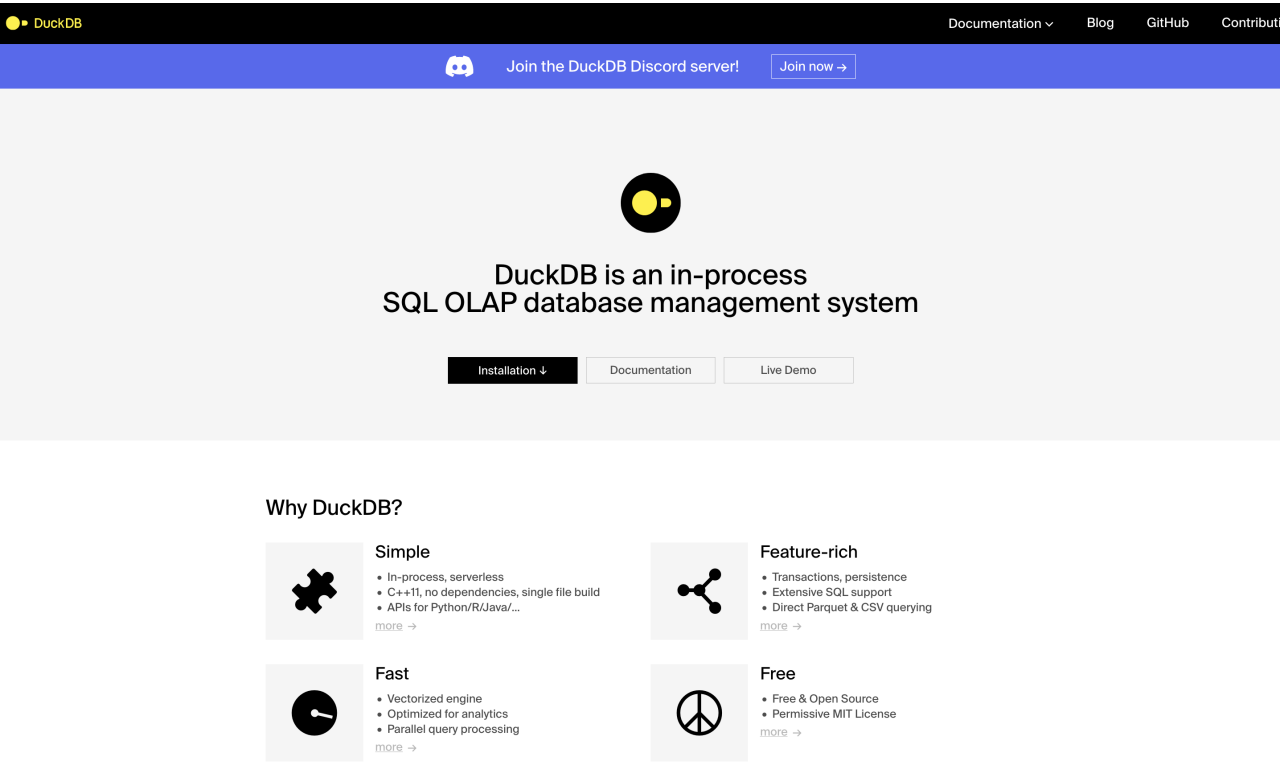








DuckDB



Doxyfile	Rework build directory somewhat, now build to build/debug and	4 years ago
DuckDBConfig.cmake.in	Split export set for CMake versions < 3.13	3 years ago
DuckDBConfigVersion.cmake.in	Add cmake install targets	3 years ago
LICENSE	add foundation	last year
Makefile	limiting std::move check to r package	last week
README.md	Update README.md	3 months ago

Languages

C++ 89.9% C 5.2% Python 2.0% Julia 0.7% Java 0.6% R 0.6% Other 1.0%

README.md

DuckDB

Main passing codefactor A codecov 83% chat 196 online

DuckDB

DuckDB is a high-performance analytical database system. It is designed to be fast, reliable and easy to use. DuckDB provides a rich SQL dialect, with support far beyond basic SQL. DuckDB supports arbitrary and nested correlated subqueries, window functions, collations, complex types (arrays, structs), and more. For more information on the goals of DuckDB, please refer to the Why DuckDB page on our website.

let's get it installed and kick the tyres!

```
> pip install duckdb

Collecting duckdb
  Downloading
  duckdb-0.6.1-cp310-cp310-macosx_10_9_x86_64.whl (13.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.6/13.6 MB 2.5 MB/s eta 0:00:00

Requirement already satisfied: numpy>=1.14 in
site-packages (from duckdb) (1.24.1)

Installing collected packages: duckdb
Successfully installed duckdb-0.6.1
```

The simplest Python Example.

```
import duckdb
cursor = duckdb.connect()

print(cursor.execute('SELECT 42').fetchall())
[(42,)]
```

Time for Level one - CSV

Titanic

kaggle

Search

Create

Home

Competitions

Datasets

Code

Discussions

Learn

More

Your Work

RECENTLY VIEWED

Titanic - Machine Lear...

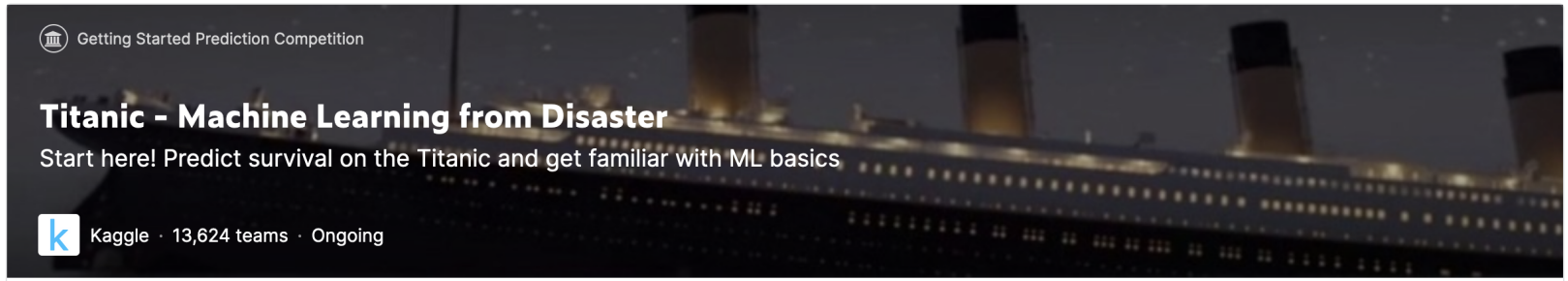
Titanic csv

weather.csv

hey - nice dataset. I thi...

IMDb data csv

View Active Events



Overview Data Code Discussion Leaderboard Rules

Submit Predictions

Overview

Description

Evaluation

Frequently Asked Questions

👋🚢 Ahoy, welcome to Kaggle! You're in the right place.

This is the legendary Titanic ML competition – the best, first challenge for you to dive into ML competitions and familiarize yourself with how the Kaggle platform works.

The competition is simple: use machine learning to create a model that predicts which passengers survived the Titanic shipwreck.

Read on or watch the video below to explore more details. Once you're ready to start competing, click on the "Join Competition button to create an account and gain access to the competition data. Then check out Alexis Cook's Titanic Tutorial that walks you through step by step how to make your first submission!

From: https://www.kaggle.com/c/titanic

train.csv

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7.25,,S
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thayer)",female,38,1,0,PC 17599,71.2833,C85,C
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3101282,7.925,,S
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",female,35,1,0,113803,53.1,C123,S
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.05,,S
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.8625,E46,S
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,349909,21.075,,S
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)",female,27,0,2,347742,11.1333,,S
10,1,2,"Nasser, Mrs. Nicholas (Adele Achem)",female,14,1,0,237736,30.0708,,C
11,1,3,"Sandstrom, Miss. Marguerite Rut",female,4,1,1,PP 9549,16.7,G6,S
12,1,1,"Bonnell, Miss. Elizabeth",female,58,0,0,113783,26.55,C103,S
13,0,3,"Saundercock, Mr. William Henry",male,20,0,0,A/5. 2151,8.05,,S
14,0,3,"Andersson, Mr. Anders Johan",male,39,1,5,347082,31.275,,S
```

```
rowset = cursor.execute("SELECT COUNT(*) FROM './data/train.csv';")
rowset.fetchall()
[(891,)]

rowset = cursor.execute("SELECT * FROM './data/train.csv' LIMIT 1;")
rowset.fetchall()
[(1,
  0,
  3,
  'Braund, Mr. Owen Harris',
  'male',
  22.0,
  1,
  0,
  'A/5 21171',
  7.25,
  None,
  'S')]
```

```
rowset.description
[('PassengerId', 'NUMBER', None, None, None, None, None),
 ('Survived', 'NUMBER', None, None, None, None, None),
 ('Pclass', 'NUMBER', None, None, None, None, None),
 ('Name', 'STRING', None, None, None, None, None),
 ('Sex', 'STRING', None, None, None, None, None),
 ('Age', 'NUMBER', None, None, None, None, None),
 ('SibSp', 'NUMBER', None, None, None, None, None),
 ('Parch', 'NUMBER', None, None, None, None, None),
 ('Ticket', 'STRING', None, None, None, None, None),
 ('Fare', 'NUMBER', None, None, None, None, None),
 ('Cabin', 'STRING', None, None, None, None, None),
 ('Embarked', 'STRING', None, None, None, None, None)]
```

```
rowset.fetch_df()
   PassengerId  Survived  Pclass                     Name   Sex   Age  SibSp  Parch     Ticket  Fare Cabin Embarked
0            1         0       3  Braund, Mr. Owen Harris  male  22.0      1      0  A/5 21171  7.25   NaN        S
```

```
sql = """
SELECT
  Name,
  Age,
  Sex,
  Fare,
  Survived
FROM
  './data/train.csv'
ORDER BY
  Fare DESC
LIMIT 10
"""

rowset = cursor.execute(sql)
rowset.fetch_df()
```

	Name	Age	Sex	Fare	Survived
0	Ward, Miss. Anna	35.0	female	512.3292	1
1	Cardeza, Mr. Thomas Drake Martinez	36.0	male	512.3292	1
2	Lesurer, Mr. Gustave J	35.0	male	512.3292	1
3	Fortune, Mr. Charles Alexander	19.0	male	263.0000	0
4	Fortune, Miss. Mabel Helen	23.0	female	263.0000	1
5	Fortune, Miss. Alice Elizabeth	24.0	female	263.0000	1
6	Fortune, Mr. Mark	64.0	male	263.0000	0
7	Ryerson, Miss. Emily Borie	18.0	female	262.3750	1
8	Ryerson, Miss. Susan Parker "Suzette"	21.0	female	262.3750	1
9	Baxter, Mr. Quigg Edmond	24.0	male	247.5208	0

Titanic Passenger Summary

Name: Miss Annie Moore Ward
Titanic Survivor
Born: Saturday 1st August 1874 in Calderhead, Lanarkshire, Scotland
Age: 37 years 8 months and 14 days (Female)
Nationality: Scottish American
Marital Status: Single
Occupation: Personal Maid to Mrs Charlotte Wardle Cardeza
1st Class Passengers
Embarked: Cherbourg on Wednesday 10th April 1912
Ticket No. 17755, £512 6s 7d
Rescued (boat 3)
Disembarked Carpathia: New York City on Thursday 18th April 1912
Died: Sunday 25th December 1955 aged 81 years
Buried: West Laurel Hill Cemetery, Bala Cynwyd, Pennsylvania, United States

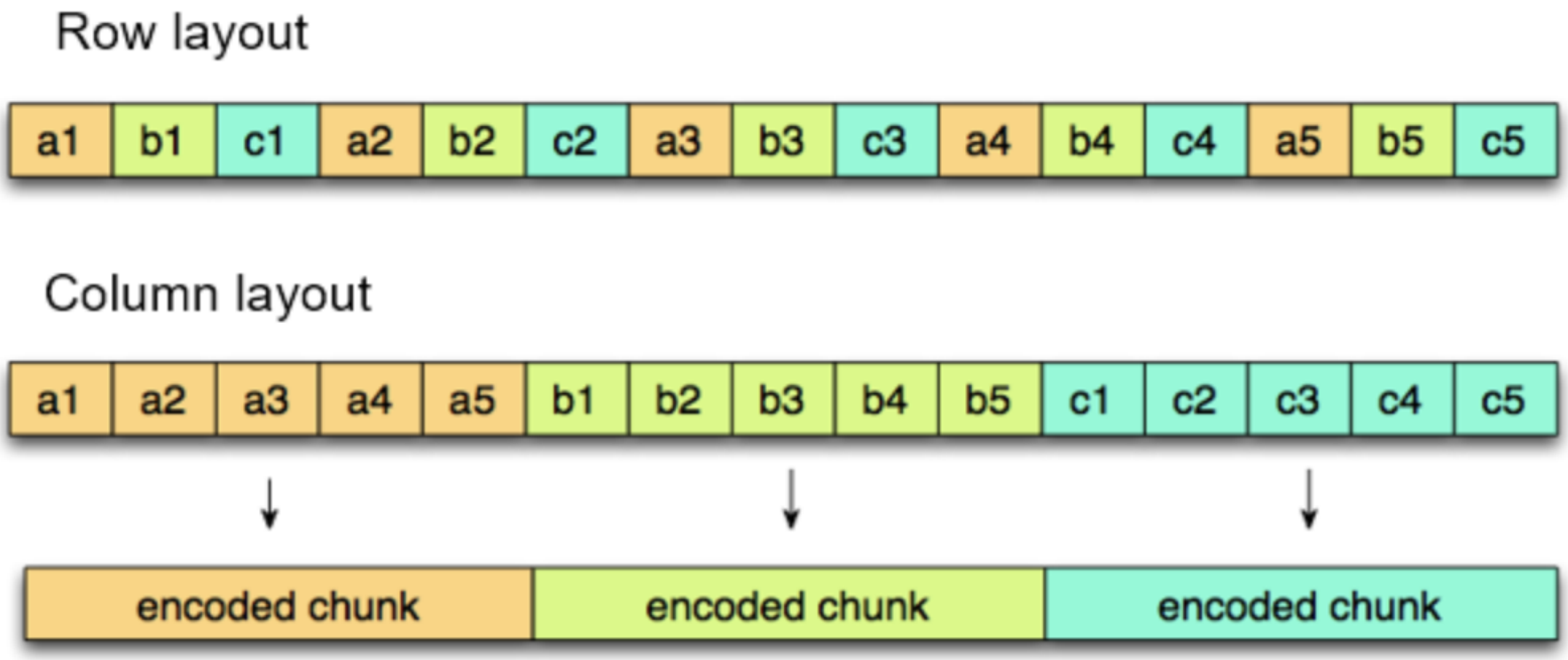
From: https://www.encyclopedia-titanica.org/titanic-survivor/annie-moore-ward.html

Super Quick Data Lake Storage Primer

Logical Table Representation

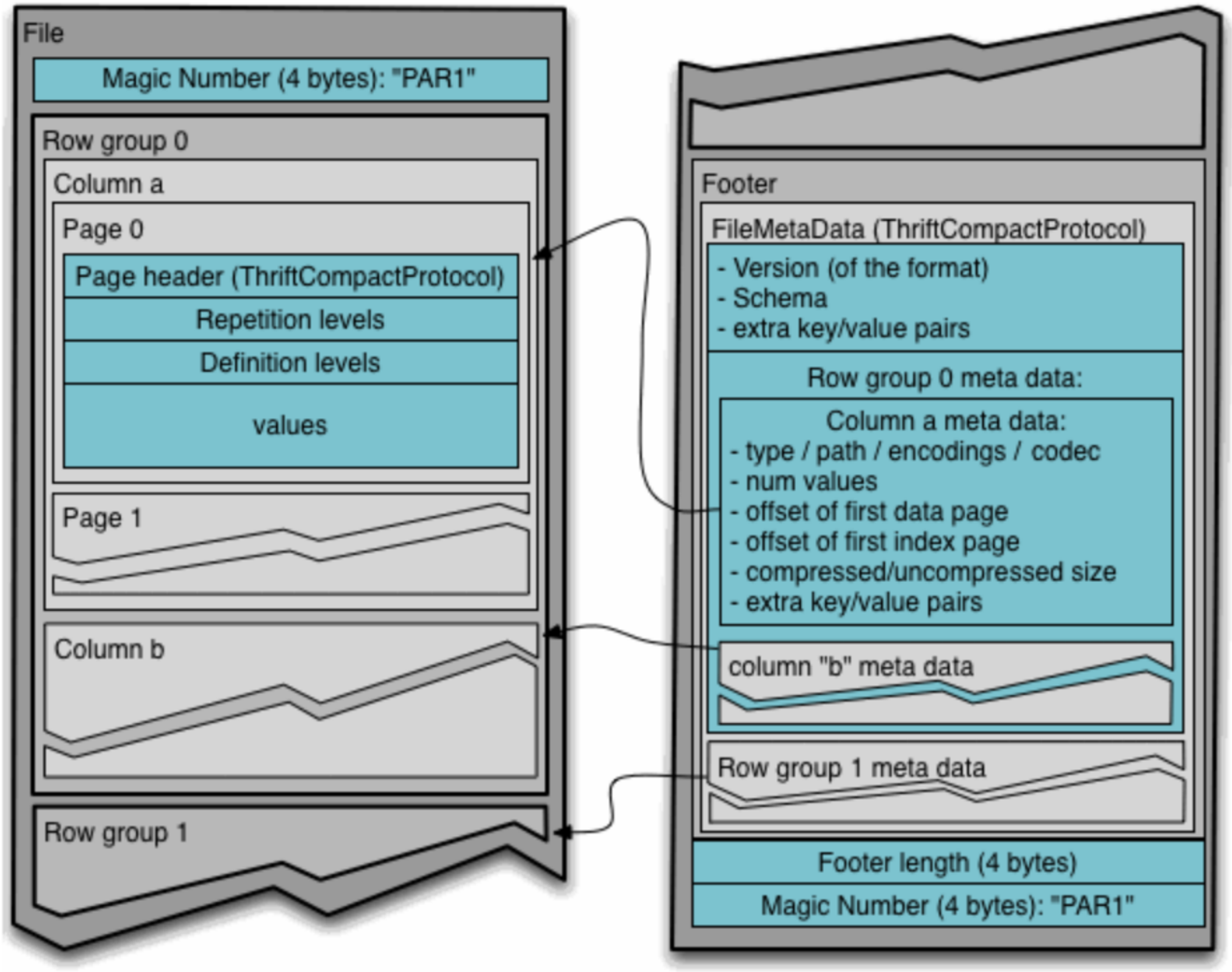
a	b	c
a1	b1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c4
a5	b5	c5

Physical Table Representation



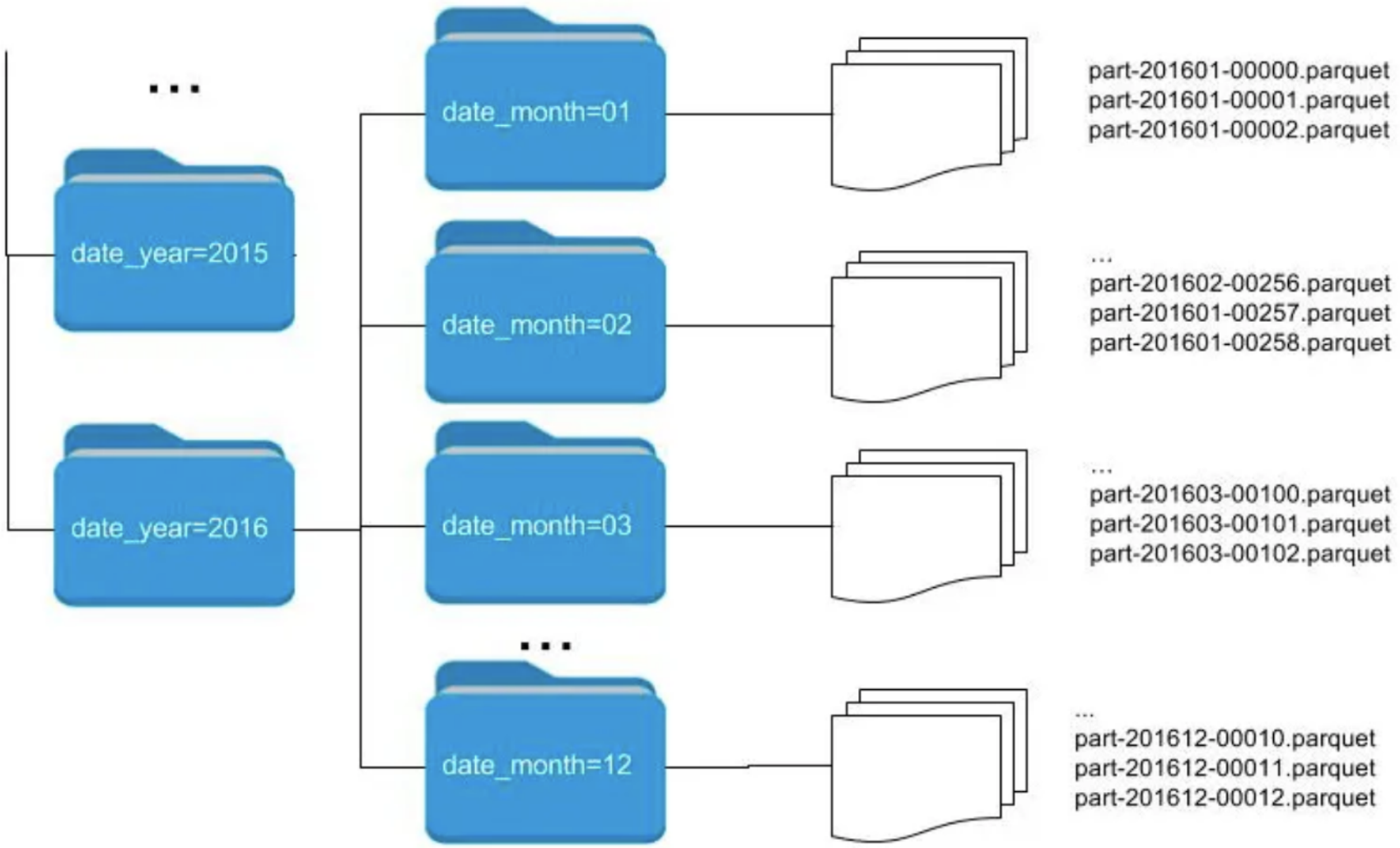
From: https://datacadamia.com/data/type/relation/structure/column_store

Parquet Format



From: https://parquet.apache.org/docs/file-format/

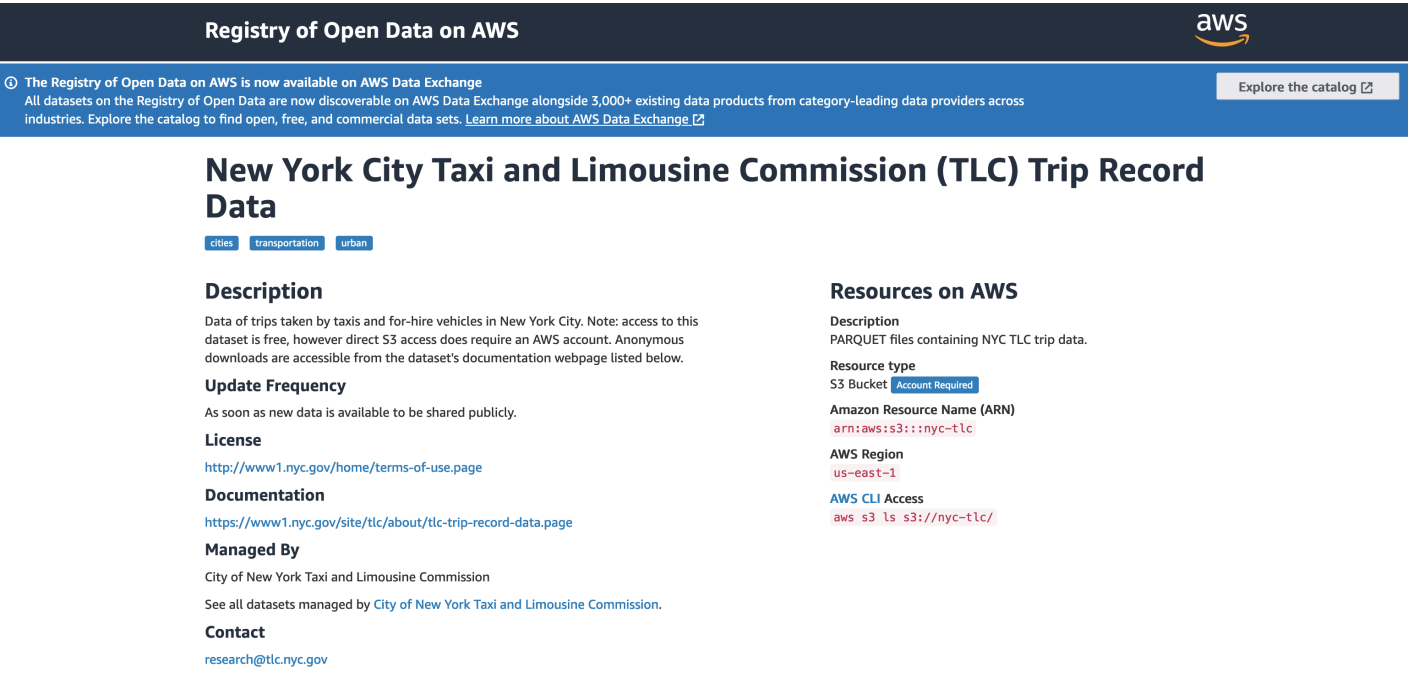
Partitioning



From: https://www.datio.com/iaas/understanding-the-data-partitioning-technique/

On to Level Two - Parquet

NYC Taxi Data




```
aws s3 ls "s3://nyc-tlc/"

2022-05-12 21:55:45   21381938 trip data/yellow_tripdata_2020-09.parquet
2022-05-12 21:55:45   26306876 trip data/yellow_tripdata_2020-10.parquet
2022-05-12 21:55:45   23583368 trip data/yellow_tripdata_2020-11.parquet
2022-05-12 21:55:45   23020036 trip data/yellow_tripdata_2020-12.parquet
2022-05-12 21:55:45   21686067 trip data/yellow_tripdata_2021-01.parquet
2022-05-12 21:55:45   21777258 trip data/yellow_tripdata_2021-02.parquet
2022-05-12 21:55:45   30007852 trip data/yellow_tripdata_2021-03.parquet
```

From: https://registry.opendata.aws/nyc-tlc-trip-records-pds/

```
sql = """
SELECT
  COUNT(*)
FROM
  './data/yellow_tripdata_2022-10.parquet';
"""
rowset = cursor.execute(sql)
rowset.fetchall()
[(3675411,)]
```

```
sql = """
SELECT
  *
FROM
  './data/yellow_tripdata_2022-10.parquet'
LIMIT 1;
"""
rowset = cursor.execute(sql)
rowset.description
[('VendorID', 'NUMBER', None, None, None, None, None),
 ('tpep_pickup_datetime', 'DATETIME', None, None, None, None, None),
 ('tpep_dropoff_datetime', 'DATETIME', None, None, None, None, None),
 ('passenger_count', 'NUMBER', None, None, None, None, None),
 ('trip_distance', 'NUMBER', None, None, None, None, None),
 ('RatecodeID', 'NUMBER', None, None, None, None, None),
 ('store_and_fwd_flag', 'STRING', None, None, None, None, None),
 ('PULocationID', 'NUMBER', None, None, None, None, None),
 ('DOLocationID', 'NUMBER', None, None, None, None, None),
 ('payment_type', 'NUMBER', None, None, None, None, None),
 ('fare_amount', 'NUMBER', None, None, None, None, None),
 ('extra', 'NUMBER', None, None, None, None, None),
 ('mta_tax', 'NUMBER', None, None, None, None, None),
 ('tip_amount', 'NUMBER', None, None, None, None, None),
 ('tolls_amount', 'NUMBER', None, None, None, None, None),
 ('improvement_surcharge', 'NUMBER', None, None, None, None, None),
 ('total_amount', 'NUMBER', None, None, None, None, None),
 ('congestion_surcharge', 'NUMBER', None, None, None, None, None),
 ('airport_fee', 'NUMBER', None, None, None, None, None)]
```

```
sql = """
SELECT
  *
FROM
  './data/yellow_tripdata_2022-10.parquet'
LIMIT 1;
"""
rowset = cursor.execute(sql)

print(rowset.fetch_df().T.to_markdown())
```

	0
VendorID	1
tpep_pickup_datetime	2022-10-01 00:03:41
tpep_dropoff_datetime	2022-10-01 00:18:39
passenger_count	1.0
trip_distance	1.7
RatecodeID	1.0
store_and_fwd_flag	N
PULocationID	249
DOLocationID	107
payment_type	1
fare_amount	9.5
extra	3.0
mta_tax	0.5
tip_amount	2.65
tolls_amount	0.0
improvement_surcharge	0.3
total_amount	15.95
congestion_surcharge	2.5
airport_fee	0.0

```
sql = """
    SELECT
    DATE_PART('hour', tpep_pickup_datetime) as dropoff_hour,
    COUNT(*) as trip_count,
    ROUND(SUM(fare_amount) / COUNT(*), 2) as avg_fare_amount,
    ROUND(100 * (SUM(tip_amount) / SUM(fare_amount)), 2) as avg_tip_pct
    FROM
    './data/yellow_tripdata_2022-10.parquet'
    GROUP BY
    1
    ORDER BY
    dropoff_hour
"""
rowset = cursor.execute(sql)
print(rowset.fetch_df().to_markdown())
```

	dropoff_hour	trip_count	avg_fare_amount	avg_tip_pct
0	0	107210	15.62	18.61
1	1	71720	13.84	19.15
2	2	50726	13.34	18.94
3	3	33938	14.14	18.26
4	4	22468	18.73	17.17
5	5	23985	22.22	16.67
6	6	55190	18.45	17.17
7	7	105110	15.6	18.29
8	8	139141	15.27	18.51
9	9	156717	14.5	18.51
10	10	173381	14.42	18.28
11	11	187832	14.68	18.3
12	12	198638	15.15	18.08
13	13	200519	15.99	17.95
14	14	213367	16.36	17.96
15	15	221032	16.54	18.06
16	16	223512	16.53	18.66
17	17	248394	15.36	19.17
18	18	254694	14.49	19.59
19	19	239130	14.07	19.62
20	20	213405	14.24	19.5
21	21	202483	14.58	19.55
22	22	183706	15.41	19.38
23	23	149113	16.07	18.84

Finallly, Level 3 - Parquet on S3 (/Cloud Storage)

```
import duckdb
cursor = duckdb.connect()
cursor.execute("INSTALL httpfs;")
cursor.execute("LOAD httpfs;")
cursor.execute("SET s3_region='us-east-1';")
cursor.execute("SET s3_access_key_id='****************';")
cursor.execute("SET s3_secret_access_key='************************';")
```

```
sql = """
SELECT
  COUNT(*)
FROM
  read_parquet('s3://is-thaipy/nyctaxi/year=2022/month=10/yellow_tripdata_2022-10.parquet')
  ;
"""
rowset = cursor.execute(sql)
rowset.fetchall()
[(3675411,)]
```

```
sql = """
SELECT
  COUNT(*)
FROM
  parquet_scan('s3://is-sd-z001/nyctaxi/year=*/month=*/*.parquet')
  ;
"""
rowset = cursor.execute(sql)
rowset.fetchall()
[(33003832,)]
```

```
sql = """
    SELECT
      DATE_PART('hour', tpep_pickup_datetime) as dropoff_hour,
      COUNT(*) as trip_count,
      ROUND(SUM(fare_amount) / COUNT(*), 2) as avg_fare_amount,
      ROUND(100 * (SUM(tip_amount) / SUM(fare_amount)), 2) as avg_tip_pct
    FROM
      parquet_scan('s3://is-sd-z001/nyctaxi/year=*/month=*/*.parquet')
    GROUP BY
      1
    ORDER BY
      dropoff_hour
"""
rowset = cursor.execute(sql)
print(rowset.fetch_df().to_markdown())
```

	dropoff_hour	trip_count	avg_fare_amount	avg_tip_pct
0	0	916905	15.41	18.49
1	1	602773	14.19	18.56
2	2	405248	13.45	18.5
3	3	267859	14.28	17.63
4	4	183428	18.84	16.7
5	5	210743	21.55	16.39
6	6	517720	17.24	16.98
7	7	958538	14.97	18.09
8	8	1.2736e+06	14.46	18.45
9	9	1.41934e+06	14.17	18
10	10	1.56194e+06	13.8	17.97
11	11	1.69816e+06	14.25	17.7
12	12	1.83514e+06	14.36	17.9
13	13	1.87048e+06	14.8	17.82
14	14	2.00555e+06	15.29	17.8
15	15	2.05901e+06	15.54	17.85
16	16	2.05026e+06	15.5	18.56
17	17	2.23669e+06	14.69	19.02
18	18	2.35226e+06	13.61	19.66
19	19	2.1173e+06	13.35	19.77
20	20	1.81635e+06	13.7	19.57
21	21	1.73885e+06	14.02	19.68
22	22	1.60565e+06	14.66	19.48
23	23	1.30004e+06	15.33	18.85

- Writing to cloud storage with DuckDB
- Running DuckDB serverless (in AWS Lambda?)
- Poor Man's Data Lake (https://dagster.io/blog/duckdb-data-lake)

Would Donald Duck use DuckDB for analytics?

It's hard to say for sure whether Donald Duck would use DuckDB for analytics, but it certainly seems like a good fit for his needs.

DuckDB's column-store architecture and ability to handle big data make it well suited for data warehousing and analytics workloads. This would be especially useful for Donald Duck as he is running a business, like his famous "Donald's Boat" and he might want to analyze data about his customers, sales and revenue.

Additionally, DuckDB's ability to run on embedded systems would make it easy for Donald Duck to access and analyze data on the go, whether he's on his boat or at his home.

Overall, it seems like DuckDB would be a great choice for Donald Duck's analytics needs. Quack quack!