Workshop "Data Ingestion with dlt": Homework

Dataset & API

We'll use **NYC Taxi data** via the same custom API from the workshop:

🔷 **Base API URL:**

```
https://us-central1-dlthub-analytics.cloudfunctions.net/data_engineering_zoomcamp_api
```

🔷 **Data format:** Paginated JSON (1,000 records per page).

🔷 **API Pagination:** Stop when an empty page is returned.

Question 1: dlt Version

1. **Install dlt**:

```
!pip install dlt[duckdb]
```

```
Collecting dlt[duckdb]
  Downloading dlt-1.6.1-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: PyYAML>=5.4.1 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: click>=7.1 in /usr/local/lib/python3.11/dist-packages (fr
Requirement already satisfied: duckdb>=0.9 in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: fsspec>=2022.4.0 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: gitpython>=3.1.29 in /usr/local/lib/python3.11/dist-packa
Collecting giturlparse>=0.10.0 (from dlt[duckdb])
  Downloading giturlparse-0.12.0-py2.py3-none-any.whl.metadata (4.5 kB)
Collecting hexbytes>=0.2.2 (from dlt[duckdb])
  Downloading hexbytes-1.3.0-py3-none-any.whl.metadata (3.3 kB)
Requirement already satisfied: humanize>=4.4.0 in /usr/local/lib/python3.11/dist-package
Collecting jsonpath-ng>=1.5.3 (from dlt[duckdb])
  Downloading jsonpath_ng-1.7.0-py3-none-any.whl.metadata (18 kB)
Collecting makefun>=1.15.0 (from dlt[duckdb])
  Downloading makefun-1.15.6-py2.py3-none-any.whl.metadata (3.2 kB)
Requirement already satisfied: orjson!=3.10.1,!=3.9.11,!=3.9.12,!=3.9.13,!=3.9.14,<4,>=3
Requirement already satisfied: packaging>=21.1 in /usr/local/lib/python3.11/dist-package
Collecting pathvalidate>=2.5.2 (from dlt[duckdb])
  Downloading pathvalidate-3.2.3-py3-none-any.whl.metadata (12 kB)
Collecting pendulum>=2.1.2 (from dlt[duckdb])
  Downloading pendulum-3.0.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.
Requirement already satisfied: pluggy>=1.3.0 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: pytz>=2022.6 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: requests>=2.26.0 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: requirements-parser>=0.5.0 in /usr/local/lib/python3.11/d
Collecting rich-argparse<2.0.0,>=1.6.0 (from dlt[duckdb])
  Downloading rich_argparse-1.7.0-py3-none-any.whl.metadata (14 kB)
Collecting semver>=3.0.0 (from dlt[duckdb])
  Downloading semver-3.0.4-py3-none-any.whl.metadata (6.8 kB)
```

```
Requirement already satisfied: setuptools>=65.6.0 in /usr/local/lib/python3.11/dist-pack
Collecting simplejson>=3.17.5 (from dlt[duckdb])
  Downloading simplejson-3.20.1-cp311-cp311-manylinux_2_5_x86_64.manylinux1_x86_64.manyl
Requirement already satisfied: tenacity>=8.0.2 in /usr/local/lib/python3.11/dist-package
Collecting tomlkit>=0.11.3 (from dlt[duckdb])
  Downloading tomlkit-0.13.2-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.11/dis
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: ply in /usr/local/lib/python3.11/dist-packages (from json
Requirement already satisfied: python-dateutil>=2.6 in /usr/local/lib/python3.11/dist-pa
Collecting time-machine>=2.6.0 (from pendulum>=2.1.2->dlt[duckdb])
  Downloading time_machine-2.16.0-cp311-cp311-manylinux_2_5_x86_64.manylinux1_x86_64.man
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dis
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: types-setuptools>=69.1.0 in /usr/local/lib/python3.11/dis
Requirement already satisfied: rich>=11.0.0 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (fr
Downloading giturlparse-0.12.0-py2.py3-none-any.whl (15 kB)
Downloading hexbytes-1.3.0-py3-none-any.whl (4.9 kB)
Downloading jsonpath_ng-1.7.0-py3-none-any.whl (30 kB)
Downloading makefun-1.15.6-py2.py3-none-any.whl (22 kB)
```

> Or choose a different bracket—`bigquery`, `redshift`, etc.—if you prefer another primary destination. For this assignment, we'll still do a quick test with DuckDB.

2. **Check** the version:

```
!dlt --version
```

```
dlt 1.6.1
```

or:

```
import dlt
print("dlt version:", dlt.__version__)
```

```
dlt version: 1.6.1
```

Answer:

- Provide the **version** you see in the output.

Question 2: Define & Run the Pipeline (NYC Taxi API)

Use dlt to extract all pages of data from the API.

Steps:

1️⃣ Use the `@dlt.resource` decorator to define the API source.

2️⃣ Implement automatic pagination using dlt's built-in REST client.

3️⃣ Load the extracted data into DuckDB for querying.

```
import dlt
from dlt.sources.helpers.rest_client import RESTClient
from dlt.sources.helpers.rest_client.paginators import PageNumberPaginator


# Define the API resource for NYC taxi data
@dlt.resource(name="rides")   # <--- The name of the resource (will be used as the table nam
def ny_taxi():
    client = RESTClient(
        base_url="https://us-central1-dlthub-analytics.cloudfunctions.net/data_engineering_z
        paginator=PageNumberPaginator(
            base_page=1,
            total_path=None
        )
    )

    for page in client.paginate("data_engineering_zoomcamp_api"):    # <--- API endpoint for
        yield page   # <--- yield data to manage memory


pipeline = dlt.pipeline(
    pipeline_name="ny_taxi_pipeline",
    destination="duckdb",
    dataset_name="ny_taxi_data"
)
```

Load the data into DuckDB to test:

```
load_info = pipeline.run(ny_taxi)
print(load_info)
```

```
Pipeline ny_taxi_pipeline load step completed in 2.67 seconds
1 load package(s) were loaded to destination duckdb and into dataset ny_taxi_data
The duckdb destination used duckdb:////content/ny_taxi_pipeline.duckdb location to store
Load package 1739692836.2228692 is LOADED and contains no failed jobs
```

Start a connection to your database using native `duckdb` connection and look what tables were generated:

```
import duckdb
from google.colab import data_table
data_table.enable_dataframe_formatter()

# A database '<pipeline_name>.duckdb' was created in working directory so just connect to it
# Connect to the DuckDB database
conn = duckdb.connect(f"{pipeline.pipeline_name}.duckdb")

# Set search path to the dataset
conn.sql(f"SET search_path = '{pipeline.dataset_name}'")

# Describe the dataset
conn.sql("DESCRIBE").df()
```

1 to 4 of 4 entries Filter

index	database	schema	name	column_names	column_types	temporary
0	ny_taxi_pipeline	ny_taxi_data	_dlt_loads	['load_id' 'schema_name' 'status' 'inserted_at' 'schema_version_hash']	['VARCHAR' 'VARCHAR' 'BIGINT' 'TIMESTAMP WITH TIME ZONE' 'VARCHAR']	false
1	ny_taxi_pipeline	ny_taxi_data	_dlt_pipeline_state	['version' 'engine_version' 'pipeline_name' 'state' 'created_at' 'version_hash' '_dlt_load_id' '_dlt_id']	['BIGINT' 'BIGINT' 'VARCHAR' 'VARCHAR' 'TIMESTAMP WITH TIME ZONE' 'VARCHAR' 'VARCHAR' 'VARCHAR']	false
2	ny_taxi_pipeline	ny_taxi_data	_dlt_version	['version' 'engine_version' 'inserted_at' 'schema_name' 'version_hash' 'schema']	['BIGINT' 'BIGINT' 'TIMESTAMP WITH TIME ZONE' 'VARCHAR' 'VARCHAR' 'VARCHAR' 'VARCHAR']	false
				['end_lat' 'end_lon' 'fare_amt' 'passenger_count' 'payment_type' 'start_lat' 'start_lon' 'tip_amt'	['DOUBLE' 'DOUBLE' 'DOUBLE' 'BIGINT' 'VARCHAR' 'DOUBLE' 'DOUBLE' 'DOUBLE' 'DOUBLE'	

Answer:

- How many tables were created?

Question 3: Explore the loaded data

Inspect the table `ride`:

```
df = pipeline.dataset(dataset_type="default").rides.df()
df
```

1 to 25 of 10000 entries Filter

ff_date_time	trip_pickup_date_time	surcharge	vendor_name	_dlt_load_id	_dlt_id	store_anc
0:00	2009-06-14 23:23:00+00:00	0.0	VTS	1739692836.2228692	fo6GauDPxnmRog	
0:00	2009-06-18 17:35:00+00:00	1.0	VTS	1739692836.2228692	B9+/CmVokVTYGw	
0:00	2009-06-10 18:08:00+00:00	1.0	VTS	1739692836.2228692	My8l+1hJlCmU/Q	
0:00	2009-06-14 23:54:00+00:00	0.5	VTS	1739692836.2228692	C//Tq2KMl5mTmQ	
0:00	2009-06-13 13:01:00+00:00	0.0	VTS	1739692836.2228692	HqzPtI8HuRMjcQ	
0:00	2009-06-10 19:43:00+00:00	1.0	VTS	1739692836.2228692	L81XGpWJJjFxng	
0:00	2009-06-10 20:06:00+00:00	0.5	VTS	1739692836.2228692	HXASFe/rw/tJ3A	
0:00	2009-06-14 20:57:00+00:00	0.5	VTS	1739692836.2228692	Mt1I20doS/mLDA	
0:00	2009-06-14 12:49:00+00:00	0.0	VTS	1739692836.2228692	t/5+oE/Vmg1p/Q	
0:00	2009-06-10 18:03:00+00:00	1.0	VTS	1739692836.2228692	EMgkX2K7jXeIBg	
0:00	2009-06-14 11:24:00+00:00	0.0	VTS	1739692836.2228692	pBEkLiDEa4jJUQ	
0:00	2009-06-13 19:17:00+00:00	0.0	VTS	1739692836.2228692	Bq6BppvDsHumYw	
0:00	2009-06-10 19:38:00+00:00	1.0	VTS	1739692836.2228692	X6+JK6PgedHqlw	
0:00	2009-06-14 02:34:00+00:00	0.5	VTS	1739692836.2228692	mqFgQoIb4pML9Q	
0:00	2009-06-16 12:56:00+00:00	0.0	VTS	1739692836.2228692	oVn4VsX93Kfqsw	
0:00	2009-06-16 12:39:00+00:00	0.0	VTS	1739692836.2228692	zZoj5L7Z5blRew	
0:00	2009-06-15 20:05:00+00:00	0.5	VTS	1739692836.2228692	ZKNYZRj7Ailm8w	
0:00	2009-06-16 12:44:00+00:00	0.0	VTS	1739692836.2228692	4FBrl8XgKyWbcA	
0:00	2009-06-10 17:57:00+00:00	1.0	VTS	1739692836.2228692	cgYmj3c7aQe16Q	
0:00	2009-06-14 17:53:00+00:00	0.0	VTS	1739692836.2228692	ftziV67yIIF5hA	
0:00	2009-06-14 11:16:00+00:00	0.0	VTS	1739692836.2228692	KpE+uZ/GlVILXQ	
0:00	2009-06-18 17:02:00+00:00	1.0	VTS	1739692836.2228692	P9WzO8pjaNOBHw	
0:00	2009-06-14 19:04:00+00:00	0.0	VTS	1739692836.2228692	30sHf3Z9JseSeA	
0:00	2009-06-15 19:17:00+00:00	1.0	VTS	1739692836.2228692	OD8yfa1iOmsZbQ	
0:00	2009-06-10 19:13:00+00:00	1.0	VTS	1739692836.2228692	VNg9tDY+mSFcSA	

Show 25 per page

1 2 10 100 300 390 400

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.describe()
```

1 to 8 of 8 entries Filter

index	end_lat	end_lon	fare_amt	passenger_count	start_lat
count	10000.0	10000.0	10000.0	10000.0	10000.0
mean	40.381455149	-73.30134047729999	10.066001000000002	2.0853	40.356752234599995
std	3.8701086329545915	7.0249862646647205	8.245156052970446	2.580094881774512	3.994385294121354
min	0.0	-74.330058	2.5	1.0	0.0
25%	40.736812	-73.99124625	5.7	1.0	40.73734925
50%	40.754705	-73.97995900000001	7.7	1.0	40.7540945
75%	40.7691105	-73.96498925	11.3	3.0	40.768395999999996
max	41.310787	0.005538	194.0	208.0	41.156413

Show 25 per page

Answer:

- What is the total number of records extracted?

Question 4: Trip Duration Analysis

Run the SQL query below to:

- Calculate the average trip duration in minutes.

```
with pipeline.sql_client() as client:
    res = client.execute_sql(
            """
            SELECT
            AVG(date_diff('minute', trip_pickup_date_time, trip_dropoff_date_time))
            FROM rides;
            """
        )
    # Prints column values of the first row
    print(res)
```

```
[(12.3049,)]
```

Answer:

- What is the average trip duration?

Submitting the solutions

- Form for submitting: TBA

Solution

We will publish the solution here after deadline.