# Data Engineering — Building an ETL Pipeline for NYC Taxi Trip Data

```
Sahil Sharma · Follow
4 min read · Apr 11, 2023
```

○ 3

152

The ETL Process Explained Extract Transform Load Processes and organizes Moves transformed data Retrieves and verifies data to a data repository extracted data so it is usable from various sources

dropoff times and locations, fare amount, and payment type are included in the dataset. The TLC makes the data available in parquet format, with one file per month.

Every month, the New York City Taxi and Limousine Commission (TLC)

publishes a dataset of taxi trips in New York City. Details such as pickup and

You are tasked as a data engineer with creating an ETL pipeline to extract data from monthly parquet files, transform it to a consistent format, and load it into Snowflake for analysis and reporting.

Solution: To solve this issue, we can construct our ETL pipeline using Python and a number of third-party libraries. **Importing libraries** 

Installing the snowflake library.

## pip install snowflake-connector-python

```
# Importing libraries
    import pandas as pd
    import snowflake.connector
    import pyarrow.parquet as pq
Extracting the Data
The data can be downloaded from — <u>TLC Trip Record Data — TLC (nyc.gov)</u>
We will take 3 months of data from Jan 2023 till Nov 2022 in parquet format
and combine it into a single pandas dataframe.
```

data\_frames.append(df)

# Parquet files parquet\_files = ['yellow\_tripdata\_2023-01.parquet', 'yellow\_tripdata\_2022-12.parquet',

'yellow\_tripdata\_2022-11.parquet']

# Extracting data into dataframes data\_frames = [] for file in parquet\_files: df = pq.read\_table(file).to\_pandas()

all\_data = pd.concat(data\_frames, ignore\_index=True)

```
Transforming the Data
Now let's transform the data,
1. we will use dropna() to remove the rows with missing data.
2. Update the source columns with proper names.
3. Update the datetime fields with proper formats.
4. Create the required derived fields like, trip_duration and
   trip_duration_minutes.
```

all\_data = all\_data.dropna() # Remove rows with missing data

all\_data = all\_data.rename(columns={'tpep\_pickup\_datetime': 'pickup\_datetime',

all\_data['trip\_duration'] = all\_data['dropoff\_datetime'] - all\_data['pickup\_date all\_data['trip\_duration\_minutes'] = all\_data['trip\_duration'].dt.total\_seconds()

all\_data['pickup\_datetime'] = pd.to\_datetime(all\_data['pickup\_datetime'])

all\_data['dropoff\_datetime'] = pd.to\_datetime(all\_data['dropoff\_datetime'])

Now let's load the data in Snowflake, first we will create the connection with

Snowflake after that we will create the ddl structure for the table and then

load the data into the table taxi\_trips in this case.

Account names varies based on different region and cloud,

use the link above to choose the right one.

**Loading the Data** 

# Transform the data

Link for Snowflake account name criteria. # Snowflake connection parameters

snowflake\_database = 'database\_name' # Specify the desired database name

# Specify the desired Schema name

# Create a Snowflake connection conn = snowflake.connector.connect( user=snowflake\_user,

snowflake\_account = 'account\_name'

snowflake\_warehouse = 'warehouse\_name'

snowflake\_user = 'user\_name'

snowflake\_password = 'password'

snowflake\_schema = 'schema\_name'

```
password=snowflake_password,
       account=snowflake_account,
        warehouse=snowflake_warehouse
    # Create a cursor
    cur = conn.cursor()
    # Create the database if it doesn't exist
    cur.execute(f"CREATE DATABASE IF NOT EXISTS {snowflake_database}")
    # Switch to the specified database
    cur.execute(f"USE DATABASE {snowflake_database}")
    # Create schema if it doesn't exist
    cur.execute(f"CREATE SCHEMA IF NOT EXISTS {snowflake_schema}")
    # Use the specified schema
    cur.execute(f"USE SCHEMA {snowflake_schema}")
    # Create the table if it doesn't exist
    cur.execute("""
        CREATE OR REPLACE TABLE IF NOT EXISTS taxi_trips (
            pickup_datetime TIMESTAMP,
            dropoff_datetime TIMESTAMP,
            passenger_count INTEGER,
            trip_distance FLOAT,
            rate_code_id INTEGER,
            store_and_fwd_flag STRING,
            payment_type INTEGER,
            fare_amount FLOAT,
            extra FLOAT,
            mta_tax FLOAT,
            tip_amount FLOAT,
            tolls_amount FLOAT,
            improvement_surcharge FLOAT,
            total_amount FLOAT,
            trip_duration INTERVAL MINUTE
    """)
    # Load data from 'all_data' DataFrame into the Snowflake table 'taxi_trips'
    cur.execute("TRUNCATE TABLE taxi_trips") # Clear existing data if needed
    cur.execute("BEGIN") # Begin a transaction
    # Use the 'copy_into' method from the DataFrame to load data into Snowflake
    all_data.to_sql('taxi_trips', conn, if_exists='replace', index=False, schema=snd
    # Commit the transaction
    cur.execute("COMMIT")
    # Close the cursor and connection
    cur.close()
    conn.close()
The pyarrow.parquet module is used in this implementation to read data
from Parquet files into a Pandas DataFrame. In addition, we've changed the
CREATE TABLE statement to match the schema of the data we're loading.
Finally, we use the to_sql() function in Snowflake to write the transformed
data to a table called taxi_trips.
Note that to use the PyArrow library, you will need to install the pyarrow
package. You can install it via pip:
```

# Importing libraries import pandas as pd import os import snowflake.connector import pyarrow.parquet as pq # Extract data from Parquet files parquet\_files = ['yellow\_tripdata\_2023-01.parquet', 'yellow\_tripdata\_2022-12.parquet', 'yellow\_tripdata\_2022-11.parquet']

### all\_data['pickup\_datetime'] = pd.to\_datetime(all\_data['pickup\_datetime']) all\_data['dropoff\_datetime'] = pd.to\_datetime(all\_data['dropoff\_datetime']) all\_data['trip\_duration'] = all\_data['dropoff\_datetime'] - all\_data['pickup\_date all\_data['trip\_duration\_minutes'] = all\_data['trip\_duration'].dt.total\_seconds()

# Snowflake connection parameters

use the link above to choose the right one.

# Transform the data

data\_frames = []

for file in parquet\_files:

data\_frames.append(df)

df = pq.read\_table(file).to\_pandas()

all\_data = pd.concat(data\_frames, ignore\_index=True)

all\_data = all\_data.dropna() # Remove rows with missing data

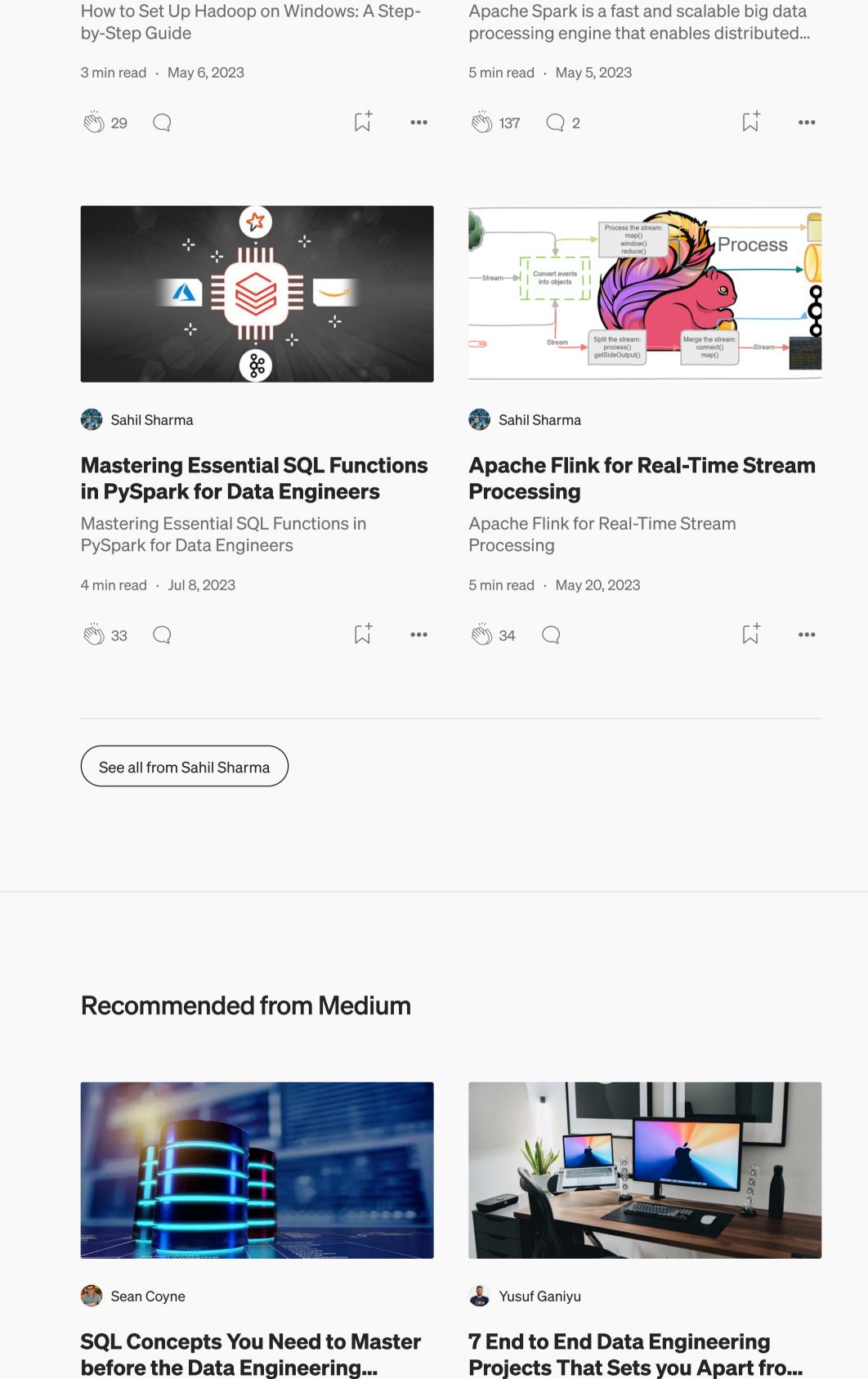
Account names varies based on different region and cloud,

all\_data = all\_data.rename(columns={'tpep\_pickup\_datetime': 'pickup\_datetime',

pip install pyarrow

Find the complete code below,

```
snowflake_account = 'account_name'
    snowflake_user = 'user_name'
    snowflake_password = 'password'
    snowflake_warehouse = 'warehouse_name'
    snowflake_database = 'database_name' # Specify the desired database name
    snowflake_schema = 'schema_name'
                                         # Specify the desired Schema name
    # Create a Snowflake connection
    conn = snowflake.connector.connect(
       user=snowflake_user,
       password=snowflake_password,
       account=snowflake_account,
       warehouse=snowflake_warehouse
    # Create a cursor
    cur = conn.cursor()
    # Create the database if it doesn't exist
    cur.execute(f"CREATE DATABASE IF NOT EXISTS {snowflake_database}")
    # Switch to the specified database
    cur.execute(f"USE DATABASE {snowflake_database}")
    # Create schema if it doesn't exist
    cur.execute(f"CREATE SCHEMA IF NOT EXISTS {snowflake_schema}")
    # Use the specified schema
    cur.execute(f"USE SCHEMA {snowflake_schema}")
    # Create the table if it doesn't exist
    cur.execute("""
        CREATE OR REPLACE TABLE IF NOT EXISTS taxi_trips (
            pickup_datetime TIMESTAMP,
           dropoff_datetime TIMESTAMP,
            passenger_count INTEGER,
            trip_distance FLOAT,
            rate_code_id INTEGER,
           store_and_fwd_flag STRING,
            payment_type INTEGER,
            fare_amount FLOAT,
            extra FLOAT,
           mta_tax FLOAT,
            tip_amount FLOAT,
            tolls_amount FLOAT,
            improvement_surcharge FLOAT,
            total_amount FLOAT,
            trip_duration INTERVAL MINUTE
    """)
    # Load data from 'all_data' DataFrame into the Snowflake table 'taxi_trips'
    cur.execute("TRUNCATE TABLE taxi_trips") # Clear existing data if needed
    cur.execute("BEGIN") # Begin a transaction
    # Use the 'copy_into' method from the DataFrame to load data into Snowflake
    all_data.to_sql('taxi_trips', conn, if_exists='replace', index=False, schema=sno
    # Commit the transaction
    cur.execute("COMMIT")
    # Close the cursor and connection
    cur.close()
    conn.close()
Wrapping Up
Please feel free to post in comments if you have some specific suggestions to
be covered in next series or if you feel some of the information is inaccurate.
If you found this post useful, Follow me as I go on my content journey!
  Data Engineering
                    Data Engineering 101
                                                Case Study
  Data Engineering Project
152
        О 3
Written by Sahil Sharma
576 Followers
```



|| Data Engineer || - || Big Data || Technology || AI & ML || CDE ||

h=doop

Spark

**Introduction to Apache Spark for** 

Sahil Sharma

Installation

on

Windows

More from Sahil Sharma

Sahil Sharma

How to Set Up Hadoop on

**Windows: A Step-by-Step Guide** 

### In order to ace the data engineering interview you will need to master essential SQL... 8 min read · Sep 22, 2023

213 Q 2

Hanif Agung Prayoga

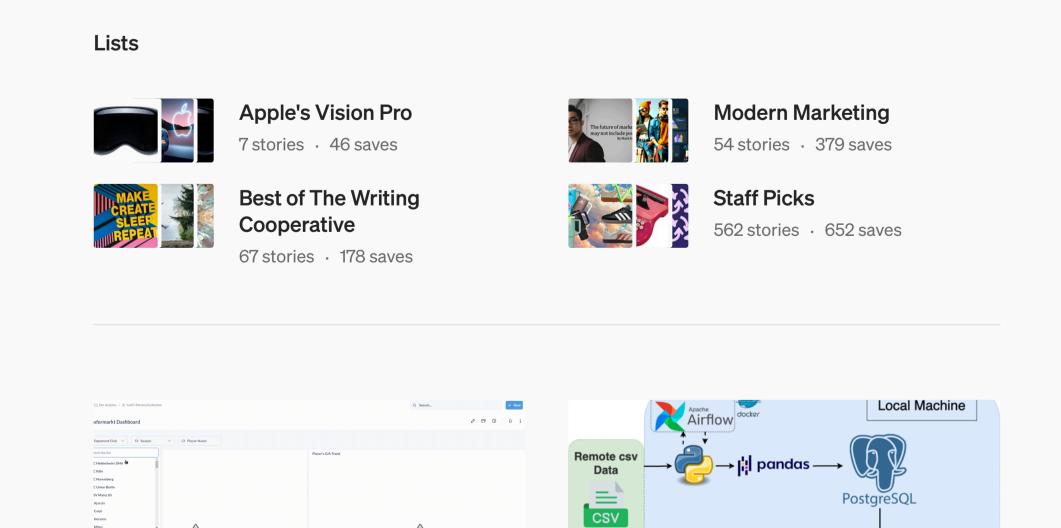
**How I Built My First Data** 

See more recommendations

These past weeks, I've been trying to learn

data engineering, which is quite a new...

**Engineering Project** 



In the ever-evolving field of data engineering,

standing out from the crowd is about...

4 min read · Dec 1, 2023

278 🔾 3

Dogukan Ulu

Repository

**Data Engineering End-to-End** 

Project—PostgreSQL, Airflow,...

11 min read · Dec 13, 2023		7 min read · Sep 19, 2023	
410	<u> </u>	581 Q 10	<u> </u>
The second results of	Model design was also control of an an an annual of an an an annual of an	APPLICATIONS  DATA INGESTION  OPERATIONAL PLANE  TOT SENSORS	DATA PRODUCTS  AI MODEL  REPORT  DASHBOARD  ANALYTICAL  PLANE  API

As we dive into 2024, let's take a sneak peek Over the course of two articles, I will into the exciting world of data engineering. ... thoroughly explore data ingestion, a... 3 min read · Dec 3, 2023 11 min read · Nov 27, 2023 222  $\bigcirc$  2 1.7K Q 20

Help Status About Careers Blog Privacy Terms Text to speech Teams

Deepanshu tyagi in DataEngineering.py janmeskens in The Modern Scientist **Popular Data Engineering Data Ingestion — Part 1: StackYou Should Know in 2024 Architectural Patterns**