

# Traffic Data pipeline with Airflow

Erickson Figueroa

## Executive Summary

The provided Airflow DAG (Directed Acyclic Graph) coordinates an Extract, Transform, Load (ETL) process for traffic data, automating the extraction and transformation from diverse sources, including CSV, TSV, and fixed-width text files. The primary functionalities encompass:

- Establishing a directory structure.
- Retrieving and extracting raw data from an AWS URL.
- Processing it into a cohesive and transformed dataset.

Key objectives of the ETL pipeline include:

- Extracting specific columns from CSV and TSV files.
- Managing fixed-width text files.
- Merging the extracted datasets.
- Transforming the vehicle type column to uppercase.

The DAG directs these tasks, utilizing Airflow's task dependencies for orderly execution.

The project improves data quality and uniformity by standardizing column names and formatting across various file types. The Airflow DAG streamlines scheduling and automation, facilitating periodic updates of the ETL process with new data. This scalable and modular approach ensures effective data management, preparing the dataset for subsequent analytics and reporting.



Do not use SQLite as metadata DB in production – it should only be used for dev/testing. We recommend using Postgres or MySQL. [Click here](#) for more information.

Do not use the SequentialExecutor in production. [Click here](#) for more information.

### DAGs

Filter DAGs by tag Search DAGs Auto-refresh

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
dataset_consumes_1	airflow	0	Dataset		On s3://dag1/output_1.txt			
dataset_consumes_1_and_2	airflow	0	Dataset		0 of 2 datasets updated			
dataset_consumes_1_never_scheduled	airflow	0	Dataset		0 of 2 datasets updated			
dataset_consumes_unknown_never_scheduled	airflow	0	Dataset		0 of 2 datasets updated			
dataset_produces_1	airflow	0	daily		2024-02-01, 00:00:00			
dataset_produces_2	airflow	0	None					
example_branch_datetime_operator	airflow	0	daily		2024-02-01, 00:00:00			
example_branch_datetime_operator_2	airflow	0	daily		2024-02-01, 00:00:00			
example_branch_datetime_operator_3	airflow	0	daily		2024-02-01, 00:00:00			

## DAG project: traffic\_data\_etl

Do not use SQLite as metadata DB in production – it should only be used for dev/testing. We recommend using Postgres or MySQL. [Click here](#) for more information.

Do not use the SequentialExecutor in production. [Click here](#) for more information.

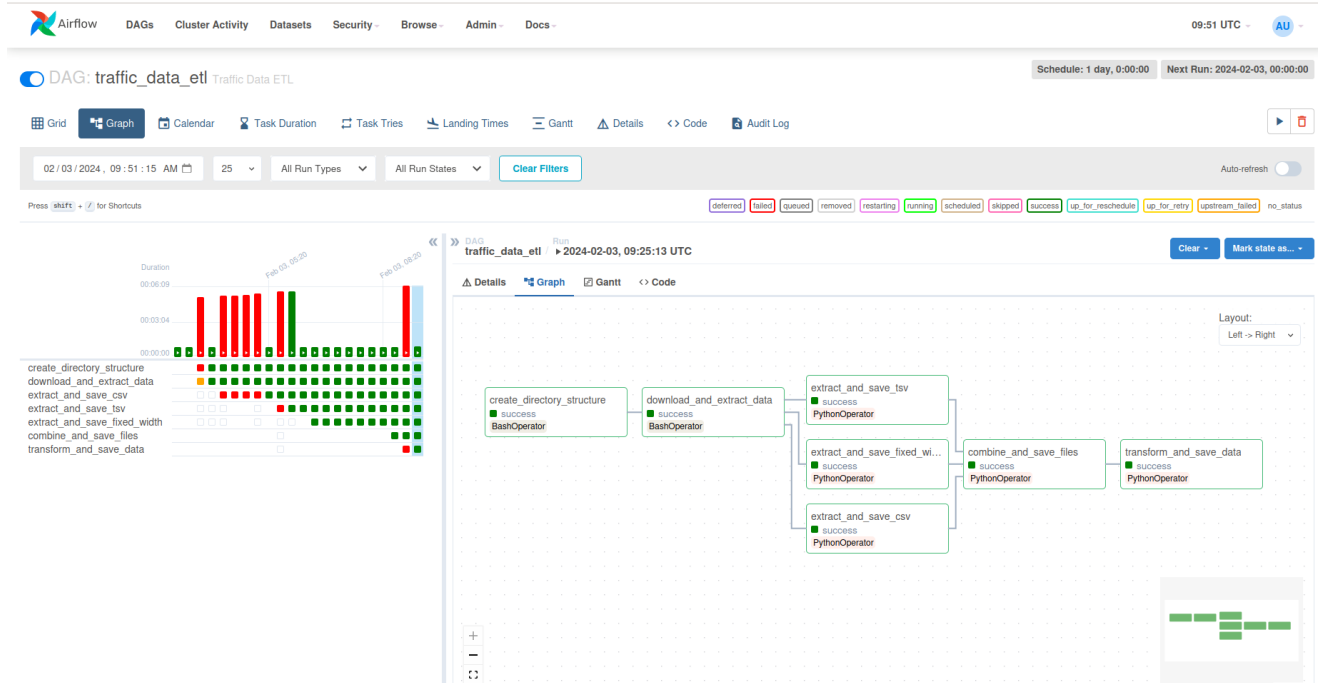
### DAGs

Filter DAGs by tag Search DAGs Auto-refresh

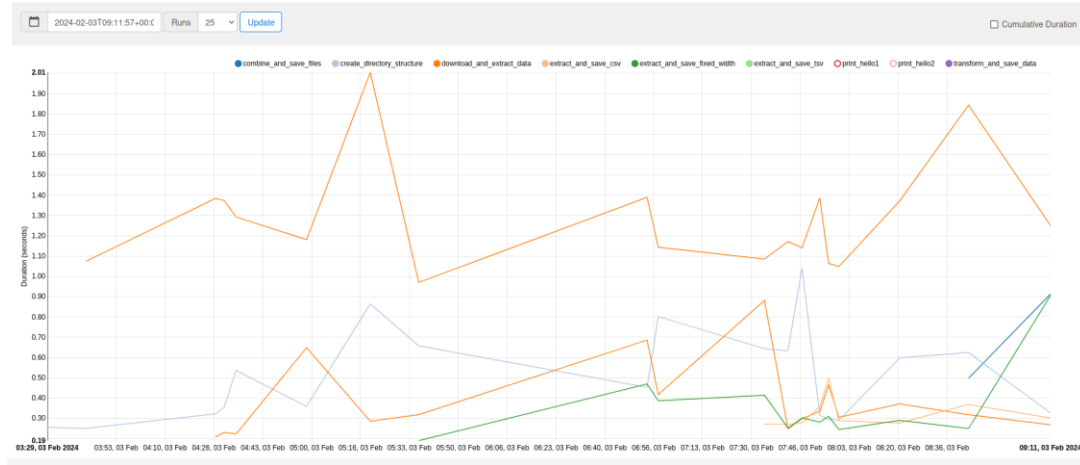
DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
traffic_data_etl	erickson	0	1 day, 0:00:00	2024-02-03, 09:25:13	2024-02-03, 00:00:00			

Showing 1-1 of 1 DAGs

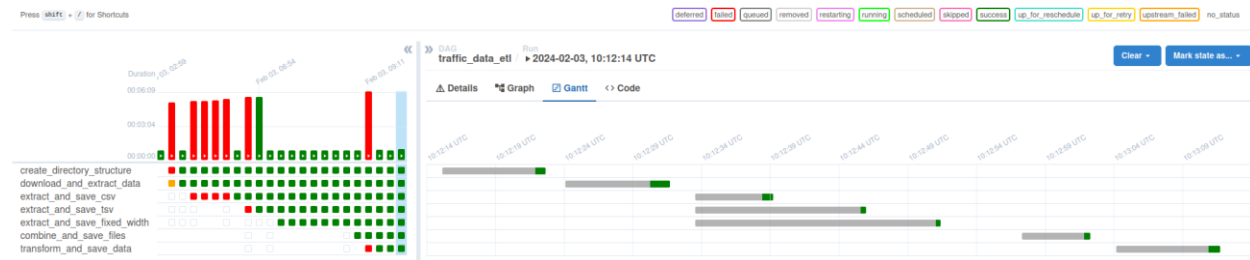
## Graph view and schedule of all tasks



## Task duration report



## Gantt view (tasks)



### Sample log file (download\_and\_extract\_data task)

DAG

Run Task

traffic\_data\_etl | ▶ 2024-02-03, 10:12:14 UTC / download\_and\_extract\_data

Clear task

Mark state as...

Filter Tasks >

△ Details

📊 Graph

📅 Gantt

<> Code

**≡ Logs**

⚙️ XCom

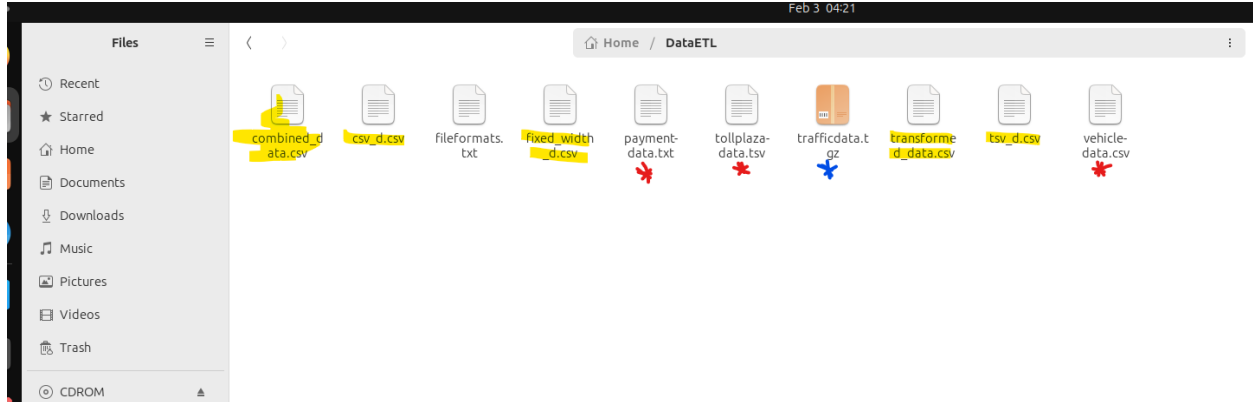
(by attempts)

1

All Levels ▾ | All File Sources ▾ | ☐ Wrap Download See More

```
***
* /home/adminuser/airflow/logs/dag_id=traffic_data_etl/run_id>manual_2024-02-03T10:12:14.151907+00:00/task_id=download_and_extract_data/attempt=1.log
[2024-02-03, 10:12:30 UTC] [taskinstance.py:1956] INFO - Dependencies all met for dep_context=non-requeueable deps ti=<TaskInstance: traffic_data_etl.download_and_extract_data manual_2024-02-03T10:12:14.151907+00:00>
[2024-02-03, 10:12:30 UTC] [taskinstance.py:1956] INFO - Dependencies all met for dep_context=requeueable deps ti=<TaskInstance: traffic_data_etl.download_and_extract_data manual_2024-02-03T10:12:14.151907+00:00>
[2024-02-03, 10:12:30 UTC] [taskinstance.py:2170] INFO - Starting attempt 1 of 2
[2024-02-03, 10:12:30 UTC] [taskinstance.py:2191] INFO - Executing <Task(BashOperator): download_and_extract_data> on 2024-02-03 10:12:14.151907+00:00
[2024-02-03, 10:12:30 UTC] [standard_task_runner.py:60] INFO - Started process 91801 to run task
[2024-02-03, 10:12:30 UTC] [standard_task_runner.py:87] INFO - Running: ['airflow', 'tasks', 'run', '[traffic_data_etl]', 'download_and_extract_data', 'manual_2024-02-03T10:12:14.151907+00:00']
[2024-02-03, 10:12:30 UTC] [standard_task_runner.py:88] INFO - Job 110: Subtask download_and_extract_data
[2024-02-03, 10:12:30 UTC] [task_command.py:423] INFO - Running <TaskInstance: traffic_data_etl.download_and_extract_data manual_2024-02-03T10:12:14.151907+00:00 [running]> on
[2024-02-03, 10:12:30 UTC] [taskinstance.py:2480] INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='erickson' AIRFLOW_CTX_DAG_ID='traffic_data_etl' AIRFLOW_CTX_TASK_ID='download_and_extract_data'
[2024-02-03, 10:12:30 UTC] [subprocess.py:63] INFO - Temp dir root location: /tmp
[2024-02-03, 10:12:30 UTC] [subprocess.py:75] INFO - Running command: ['/usr/bin/bash', '-c', 'curl -o /home/adminuser/DataETL/trafficdata.tgz https://elasticbeanstalk-us-east-1-901111111111.s3.amazonaws.com/trafficdata.tgz']
[2024-02-03, 10:12:31 UTC] [subprocess.py:93] INFO - % Total    % Received % Xferd Average Speed   Time    Time     Time Current
                                 Dload  Upload   Total             Spent    Left      Speed
[2024-02-03, 10:12:31 UTC] [subprocess.py:93] INFO - 
0      0  0      0  0      0  0      0 --:--:-- --:--:-- --:--:--  0
100  516k 100  516k  0      0  942k --:--:-- --:--:-- --:--:--  944k
[2024-02-03, 10:12:31 UTC] [subprocess.py:93] INFO - fileformats.txt
[2024-02-03, 10:12:31 UTC] [subprocess.py:93] INFO - payment-data.txt
[2024-02-03, 10:12:31 UTC] [subprocess.py:93] INFO - _tollplaza-data.tsv
[2024-02-03, 10:12:31 UTC] [subprocess.py:93] INFO - tollplaza-data.tsv
[2024-02-03, 10:12:31 UTC] [subprocess.py:93] INFO - _vehicle-data.csv
[2024-02-03, 10:12:31 UTC] [subprocess.py:93] INFO - vehicle-data.csv
[2024-02-03, 10:12:31 UTC] [subprocess.py:97] INFO - Command exited with return code 0
[2024-02-03, 10:12:31 UTC] [taskinstance.py:1138] INFO - Marking task as SUCCESS. dag_id=traffic_data_etl, task_id=download_and_extract_data, execution_time=20240203T101214, status=SUCCESS
[2024-02-03, 10:12:32 UTC] [local_task_job_runner.py:234] INFO - Task exited with return code 0
[2024-02-03, 10:12:32 UTC] [taskinstance.py:3280] INFO - 3 downstream tasks scheduled from follow-on schedule check
```

## File directory



**csv\_d.csv**

Open ▾				csv_d.csv ~/DataETL	
Rowid	Timestamp	Anonymized_Vehicle_number	Vehicle_type		
1	Thu Aug 19 21:54:38 2021	125094	car		
2	Sat Jul 31 04:09:44 2021	174434	car		
3	Sat Aug 14 17:19:04 2021	8538286	car		
4	Mon Aug 2 18:23:45 2021	5521221	car		
5	Thu Jul 29 22:44:20 2021	3267767	car		
6	Sat Aug 14 03:57:47 2021	8411850	car		
7	Thu Aug 12 03:41:22 2021	6064250	car		
8	Sun Aug 22 10:29:58 2021	6871937	van		
9	Fri Aug 6 14:23:08 2021	2055930	car		
10	Sun Aug 15 13:43:51 2021	8775910	car		
11	Wed Aug 4 15:52:13 2021	4525747	car		
12	Wed Aug 18 02:22:15 2021	5276143	van		
13	Thu Aug 19 16:01:02 2021	5405144	van		
14	Tue Aug 3 14:55:00 2021	9309466	car		
15	Thu Aug 5 04:00:30 2021	6494070	van		
16	Sun Aug 8 13:08:17 2021	4880588	car		
17	Wed Aug 4 04:56:15 2021	6964507	car		
18	Sun Aug 15 11:22:44 2021	5740782	truck		
19	Fri Jul 30 11:20:47 2021	1060261	van		
20	Sun Aug 8 21:22:56 2021	6599783	car		
21	Wed Aug 11 10:33:15 2021	4573378	car		

**Tsv\_d.csv**

			csv_d.csv
Number_of_axles	Tollplaza_id	Tollplaza_code	
2	4856	PC7C042B7	
2	4154	PC2C2EF9E	
2	4070	PCEECA8B2	
2	4095	PC3E1512A	
2	4135	PCC943ECD	
2	4680	PCC422F4D	
2	4702	PCDBC3AC9	
2	4592	PC627AA14	
2	4100	PCC6DD8D5	
2	4143	PC5F47DCF	
2	4466	PC6E93A05	
2	4352	PC99B9343	
2	4574	PCC1F6F13	
2	4902	PC425C6EC	
2	4012	PCEF5C190	
2	4458	PCECEBBFA	
2	4886	PC4D892D7	
2	4680	PCC422F4D	

## fixed\_width\_d.csv

Open ▾ 🔍 fixed\_width\_d.csv  
~/Data/ETL

Type_Of_Payment_Code,Vehicle_Code
PTE,VC965
PTP,VC965
PTE,VC965
PTP,VC965
PTE,VC965
PTE,VC965
PTE,VC965
PTE,VCD2F
PTP,VC965
PTC,VC965
PTE,VC965
PTC,VCD2F
PTP,VCD2F
PTE,VC965
PTC,VCD2F
PTE,VC965
PTC,VC965
PTP,VCB43
PTP,VCD2F
PTP,VC965
PTC,VC965

## combined\_data.csv

Open ▾ 🔍 combined\_data.csv  
~/Data/ETL

RowId,Timestamp,Anonymized_Vehicle_number,Vehicle_type,Number_of_axles,Tollplaza_id,Tollplaza_code,Type_Of_Payment_Code,Vehicle_Code
1.0,Thu Aug 19 21:54:38 2021,125094.0,car,,,,,
2.0,Sat Jul 31 04:09:44 2021,174434.0,car,,,,,
3.0,Sat Aug 14 17:19:04 2021,8538286.0,car,,,,,
4.0,Mon Aug 2 18:23:45 2021,5521221.0,car,,,,,
5.0,Thu Jul 29 22:44:20 2021,3267767.0,car,,,,,
6.0,Sat Aug 14 03:57:47 2021,8411850.0,car,,,,,
7.0,Thu Aug 12 03:41:22 2021,6064250.0,car,,,,,
8.0,Sun Aug 22 10:29:58 2021,6871937.0,car,,,,,
9.0,Fri Aug 6 14:23:08 2021,2055930.0,car,,,,,
10.0,Sun Aug 15 13:43:51 2021,8775910.0,car,,,,,
11.0,Wed Aug 4 15:52:13 2021,4525747.0,car,,,,,
12.0,Wed Aug 18 02:22:15 2021,5276143.0,car,,,,,
13.0,Thu Aug 19 16:01:02 2021,5405144.0,car,,,,,
14.0,Tue Aug 3 14:55:00 2021,9309466.0,car,,,,,
15.0,Thu Aug 5 04:00:30 2021,6494070.0,car,,,,,
16.0,Sun Aug 8 13:08:17 2021,4880588.0,car,,,,,
17.0,Wed Aug 4 04:56:15 2021,6964507.0,car,,,,,
18.0,Sun Aug 15 11:22:44 2021,5740782.0,car,,,,,
19.0,Fri Jul 30 11:20:47 2021,1060261.0,car,,,,,
20.0,Sun Aug 8 21:22:56 2021,6599783.0,car,,,,,
21.0,Wed Aug 11 10:22:15 2021,4525747.0,car,,,,,

## transformed\_data.csv

Open ▾ 🔍 transformed\_data.csv  
~/Data/ETL

RowId,Timestamp,Anonymized_Vehicle_number,Vehicle_type,Number_of_axles,Tollplaza_id,Tollplaza_code,Type_Of_Payment_Code,Vehicle_Code
1.0,Thu Aug 19 21:54:38 2021,125094.0,CAR,,,,,
2.0,Sat Jul 31 04:09:44 2021,174434.0,CAR,,,,,
3.0,Sat Aug 14 17:19:04 2021,8538286.0,CAR,,,,,
4.0,Mon Aug 2 18:23:45 2021,5521221.0,CAR,,,,,
5.0,Thu Jul 29 22:44:20 2021,3267767.0,CAR,,,,,
6.0,Sat Aug 14 03:57:47 2021,8411850.0,CAR,,,,,
7.0,Thu Aug 12 03:41:22 2021,6064250.0,CAR,,,,,
8.0,Sun Aug 22 10:29:58 2021,6871937.0,VAN,,,,,
9.0,Fri Aug 6 14:23:08 2021,2055930.0,CAR,,,,,
10.0,Sun Aug 15 13:43:51 2021,8775910.0,CAR,,,,,
11.0,Wed Aug 4 15:52:13 2021,4525747.0,CAR,,,,,
12.0,Wed Aug 18 02:22:15 2021,5276143.0,VAN,,,,,
13.0,Thu Aug 19 16:01:02 2021,5405144.0,VAN,,,,,
14.0,Tue Aug 3 14:55:00 2021,9309466.0,CAR,,,,,
15.0,Thu Aug 5 04:00:30 2021,6494070.0,VAN,,,,,
16.0,Sun Aug 8 13:08:17 2021,4880588.0,CAR,,,,,
17.0,Wed Aug 4 04:56:15 2021,6964507.0,CAR,,,,,
18.0,Sun Aug 15 11:22:44 2021,5740782.0,TRUCK,,,,,
19.0,Fri Jul 30 11:20:47 2021,1060261.0,VAN,,,,,

UPPER CASE

## Source Code

```
# Importing libraries
from airflow.models.dag import DAG
from airflow.operators.python import PythonOperator
from airflow.utils.dates import days_ago
from datetime import datetime, timedelta
from airflow.operators.bash import BashOperator
import pandas as pd

# Folder path
folder_path = "/home/adminuser/DataETL/"

# Data source from the following url
url = 'https://elasticbeanstalk-us-east-2-340729127361.s3.us-east-2.amazonaws.com/trafficdata.tgz'

# Function to extract the csv file
def extract_and_save_csv():
    input_file = f'{folder_path}/vehicle-data.csv'
    output_file = f'{folder_path}/csv_d.csv'

    # Read the CSV file, select columns 1, 2, 3, 4
    df = pd.read_csv(input_file, header=None, usecols=[0, 1, 2, 3])

    # Rename columns
    df.columns = ['Rowid', 'Timestamp', 'Anonymized_Vehicle_number',
'Vehicle_type']

    # Save the result to a new CSV file
    df.to_csv(output_file, index=False)

# Function to extract the tsv file
def extract_and_save_tsv():
    tsv_input_file = f'{folder_path}/tollplaza-data.tsv'
    tsv_output_file = f'{folder_path}/tsv_d.csv'

    # Read the TSV file, select columns 5, 6, 7
    tsv_df = pd.read_csv(tsv_input_file, sep='\t', header=None, usecols=[4, 5,
6])

    # Rename columns
    tsv_df.columns = ['Number_of_axles', 'Tollplaza_id', 'Tollplaza_code']

    # Save the result to a new CSV file
    tsv_df.to_csv(tsv_output_file, index=False)
```



```

# Function to extract the txt fixed-width file
def extract_and_save_fixed_width():
    fixed_width_input_file = f'{folder_path}/payment-data.txt'
    fixed_width_output_file = f'{folder_path}/fixed_width_d.csv'

    # Fixed-column positions for Type of Payment code and Vehicle Code
    column_positions = [(58,61), (61, 68)]

    # Read the fixed-width file
    fixed_width_df = pd.read_fwf(fixed_width_input_file,
    colspecs=column_positions, header=None)

    # Rename columns
    fixed_width_df.columns = ['Type_Of_Payment_Code', 'Vehicle_Code']

    # Save the result to a new CSV file
    fixed_width_df.to_csv(fixed_width_output_file, index=False)

# Function to combine the extracted files
def combine_and_save_files():
    # Load the three CSV files
    csv_df = pd.read_csv(f'{folder_path}/csv_d.csv')
    tsv_df = pd.read_csv(f'{folder_path}/tsv_d.csv')
    fixed_width_df = pd.read_csv(f'{folder_path}/fixed_width_d.csv')

    # Concatenate the DataFrames along the rows (axis=0)
    combined_df = pd.concat([csv_df, tsv_df, fixed_width_df], axis=0,
    ignore_index=True)

    # Save the combined DataFrame to a new CSV file
    combined_output_file = f'{folder_path}/combined_data.csv'
    combined_df.to_csv(combined_output_file, index=False)

# Function to tranform the data
# vehicle type as upercase
def transform_and_save_data():
    # Load the combined data file
    combined_df = pd.read_csv(f'{folder_path}/combined_data.csv')

    # Convert "Vehicle type" column to upercase
    combined_df['Vehicle_type'] = combined_df['Vehicle_type'].str.upper()

    # Save the transformed DataFrame to a new CSV file
    transformed_output_file = f'{folder_path}/transformed_data.csv'
    combined_df.to_csv(transformed_output_file, index=False)

```

```

# Dictionary to default args
default_args = {
    'owner': 'erickson',
    'depends_on_past': False,
    'start_date': datetime(2024, 2, 3),
    'schedule': '@daily',
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
    'catchup': False
}

# Instantiate the DAG
dag = DAG(
    dag_id='traffic_data_etl',
    default_args=default_args,
    description='Traffic Data ETL',
    schedule_interval=timedelta(days=1),
)

# Task to create the directory using BashOperator
create_directory_structure_task = BashOperator(
    task_id='create_directory_structure',
    bash_command=f'mkdir -p {folder_path}',
    dag=dag,
)

# Task to download and extract the files using BashOperator
download_and_extract_data_task = BashOperator(
    task_id='download_and_extract_data',
    bash_command=f'curl -o {folder_path}/trafficdata.tgz {url} && tar -xvzf {folder_path}/trafficdata.tgz -C {folder_path}',
    dag=dag,
)

# Task to save csv file using PythonOperator
extract_and_save_csv_task = PythonOperator(
    task_id='extract_and_save_csv',
    python_callable=extract_and_save_csv,
    dag=dag,
)

```

```

# Task to save tsv file using PythonOperator
extract_and_save_tsv_task = PythonOperator(
    task_id='extract_and_save_tsv',
    python_callable=extract_and_save_tsv,
    dag=dag,
)

# Task to save fixed-width file using PythonOperator
extract_and_save_fixed_width_task = PythonOperator(
    task_id='extract_and_save_fixed_width',
    python_callable=extract_and_save_fixed_width,
    dag=dag,
)

# Task to combine and save files using PythonOperator
combine_and_save_files_task = PythonOperator(
    task_id='combine_and_save_files',
    python_callable=combine_and_save_files,
    dag=dag,
)

# Task to transform and save data using PythonOperator
transform_and_save_data_task = PythonOperator(
    task_id='transform_and_save_data',
    python_callable=transform_and_save_data,
    dag=dag,
)

#-----
#-- Set up task dependencies execution --#
#-----#

# Runs first to create the necessary directory structure.
create_directory_structure_task >> download_and_extract_data_task

# The three tasks can run in parallel as they depend on the
# completion of download_and_extract_data_task.
download_and_extract_data_task >> [extract_and_save_csv_task,
                                   extract_and_save_tsv_task,
                                   extract_and_save_fixed_width_task]

# Combine task, depends on the completion of the three extraction tasks.
[extract_and_save_csv_task,
 extract_and_save_tsv_task,
 extract_and_save_fixed_width_task] >> combine_and_save_files_task

# Transform task, depends on the completion of combine task
combine_and_save_files_task >> transform_and_save_data_task

```