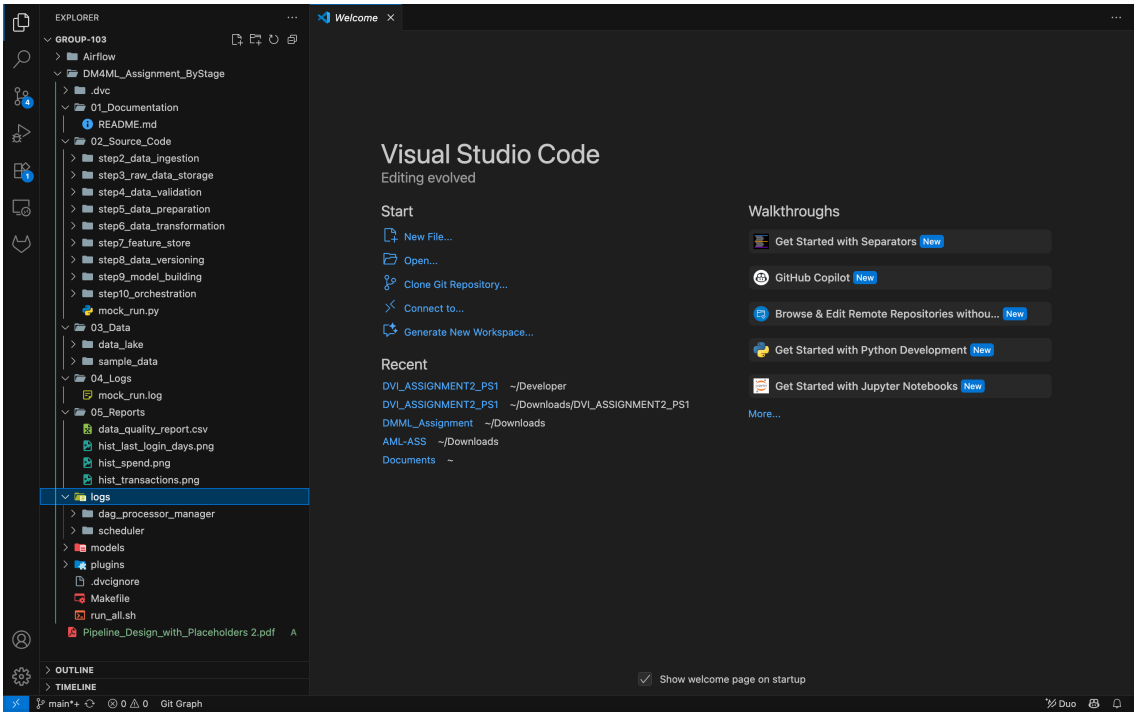
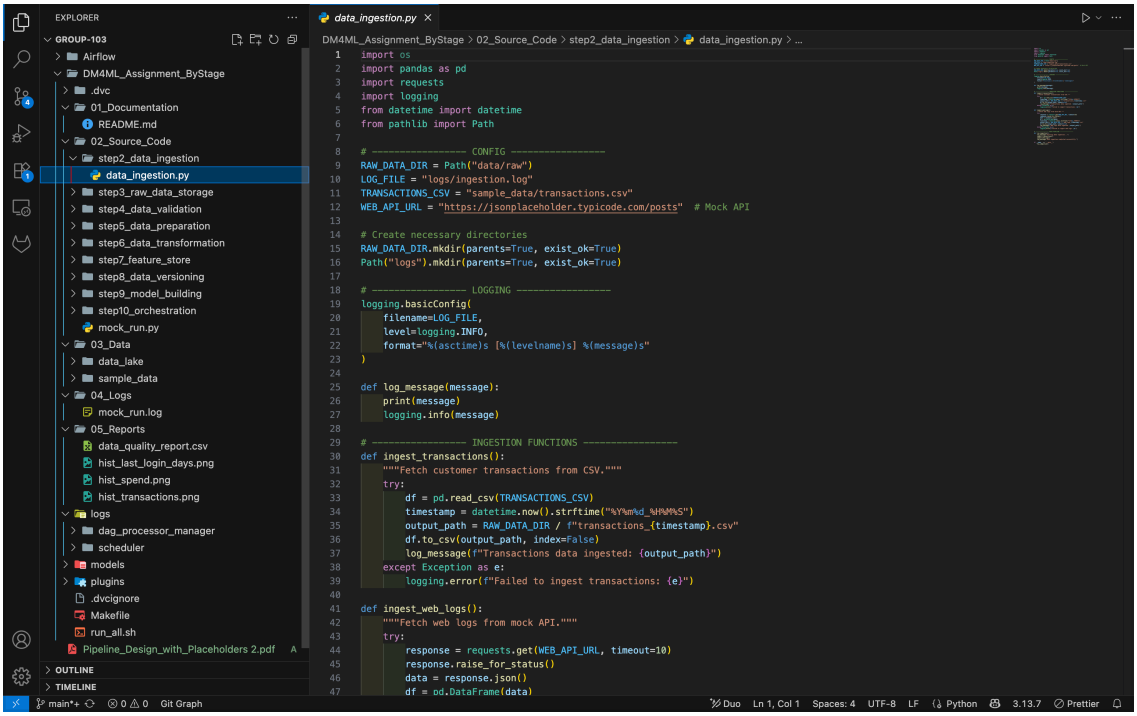
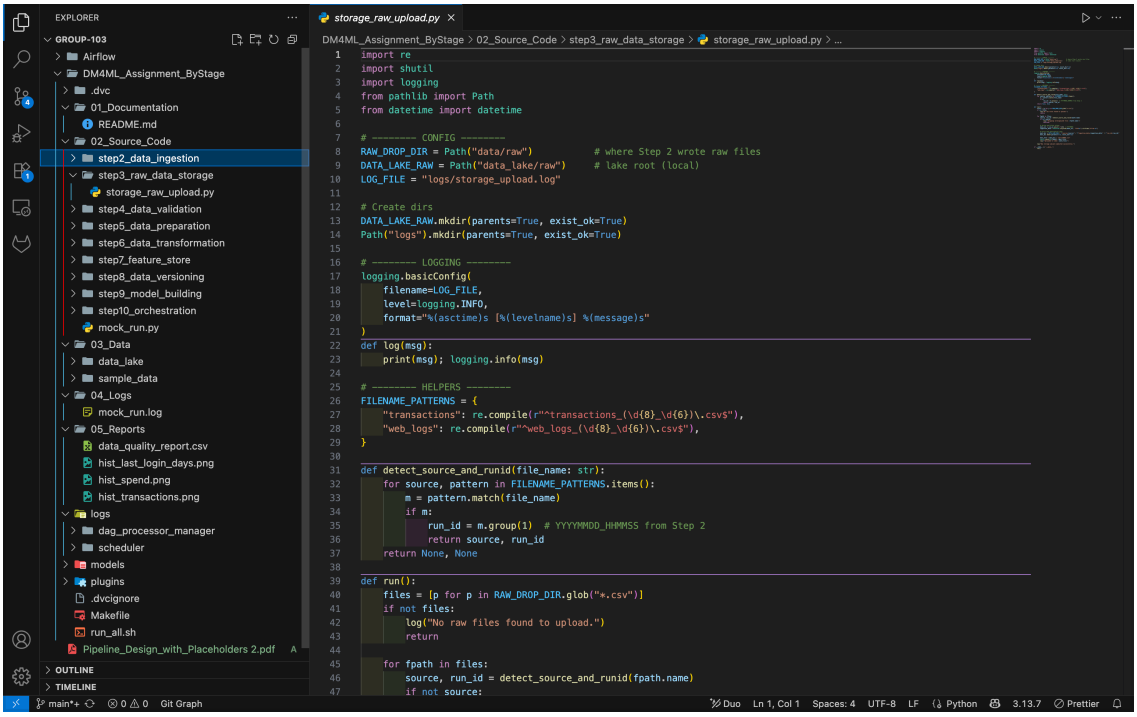
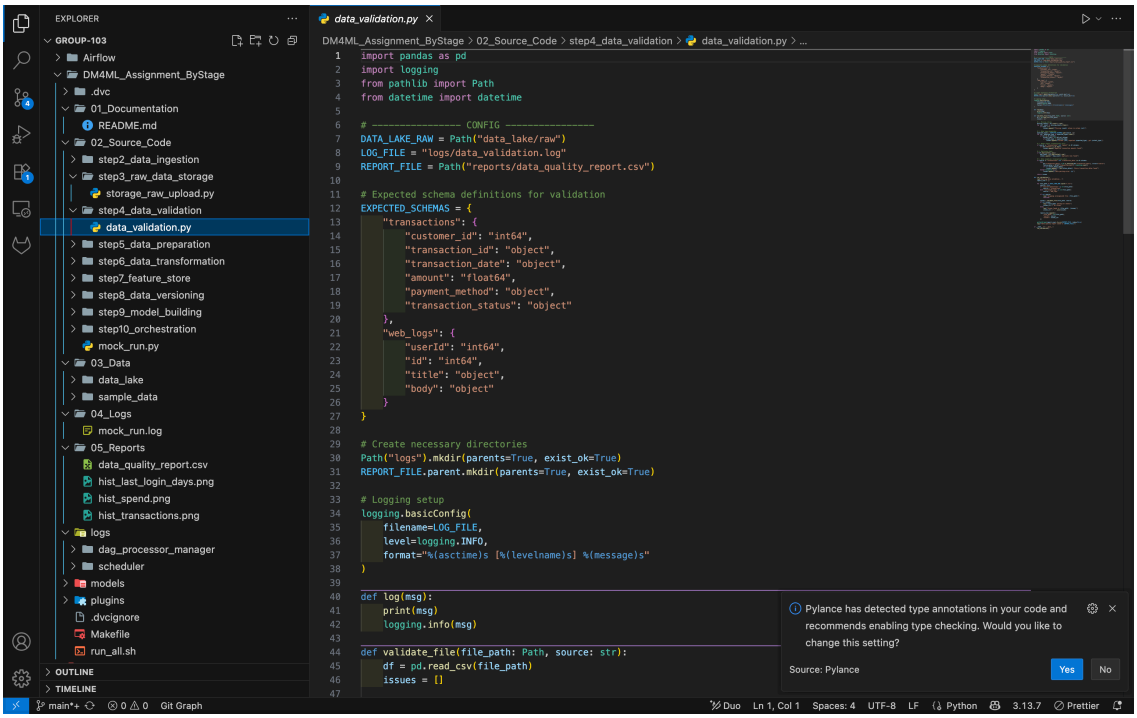
DMML Video Walkthrough

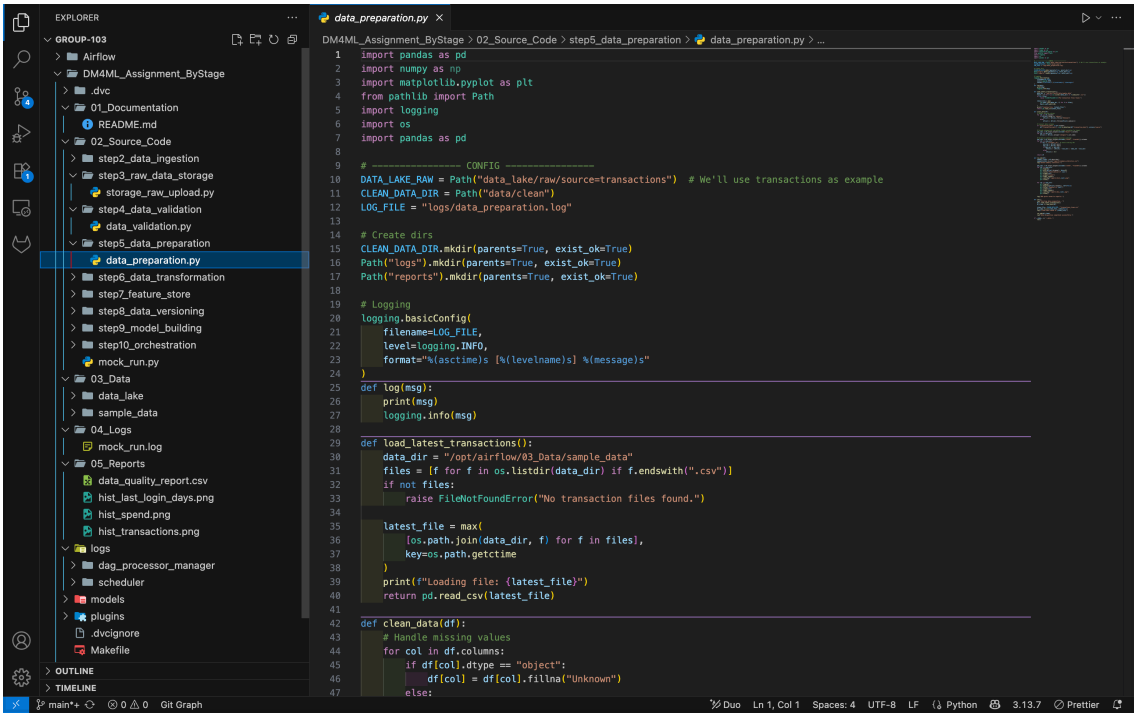
Hello Sir,
We were unable to upload the video along with the assignment as the portal has a file size limit, and the video exceeds that limit. Therefore, I am sharing the video link here for your reference:
https://youtu.be/fW-mXFeO420
Additionally, I have attached the screenshot of the code and pipeline.

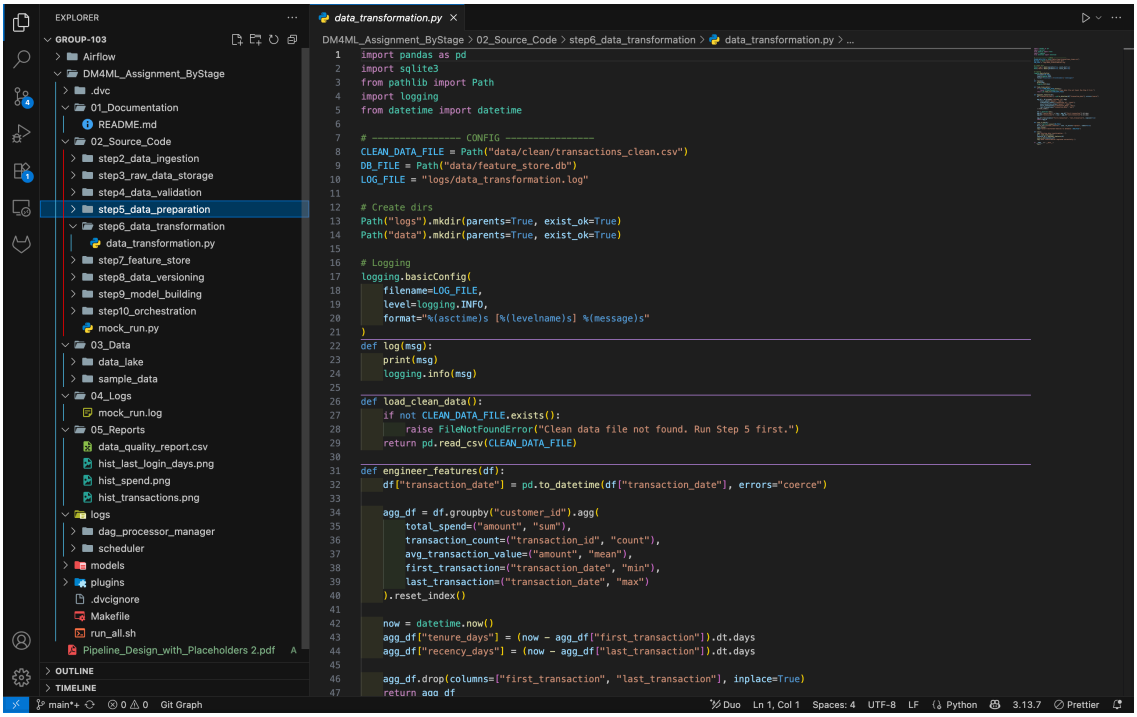


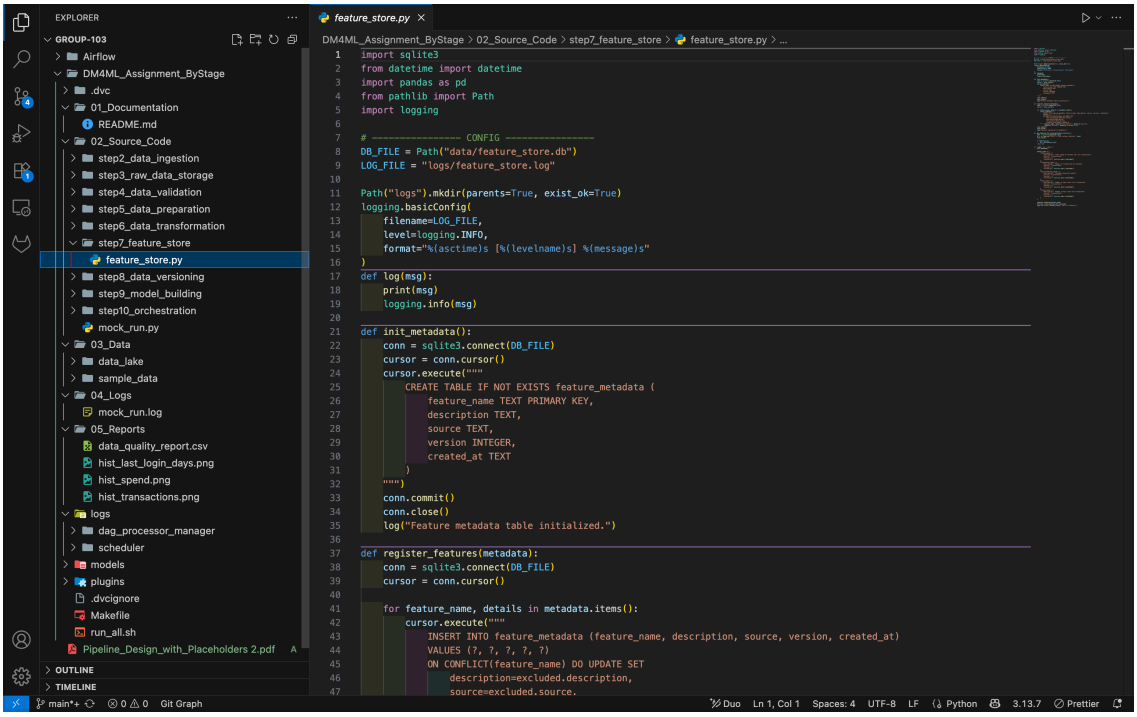

EXPLORER
GROUP-103
Airflow
DM4ML_Assignment_ByStage
.dvc
01_Documentation
README.md
02_Source_Code
step2_data_ingestion
data_ingestion.py
step3_raw_data_storage
step4_data_validation
step5_data_preparation
step6_data_transformation
step7_feature_store
step8_data_versioning
step9_model_building
step10_orchestration
mock_run.py
03_Data
data_lake
sample_data
04_Logs
mock_run.log
05_Reports
data_quality_report.csv
hist_last_login_days.png
hist_spend.png
hist_transactions.png
logs
dag_processor_manager
scheduler
models
plugins
.dvcignore
Makefile
run_all.sh
Pipeline_Design_with_Placeholders 2.pdf
OUTLINE
TIMELINE
data_ingestion.py
DM4ML_Assignment_ByStage > 02_Source_Code > step2_data_ingestion > data_ingestion.py > ...
import os
import pandas as pd
import requests
import logging
from datetime import datetime
from pathlib import Path
# ---------- CONFIG ----------
RAW_DATA_DIR = Path("data/raw")
LOG_FILE = "logs/ingestion.log"
TRANSACTIONS_CSV = "sample_data/transactions.csv"
WEB_API_URL = "https://jsonplaceholder.typicode.com/posts" # Mock API
# Create necessary directories
RAW_DATA_DIR.mkdir(parents=True, exist_ok=True)
Path("logs").mkdir(parents=True, exist_ok=True)
# ---------- LOGGING ----------
logging.basicConfig(
filename=LOG_FILE,
level=logging.INFO,
format="%(asctime)s [%(levelname)s] %(message)s"
)
def log_message(message):
print(message)
logging.info(message)
# ---------- INGESTION FUNCTIONS ----------
def ingest_transactions():
"""Fetch customer transactions from CSV."""
try:
df = pd.read_csv(TRANSACTIONS_CSV)
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
output_path = RAW_DATA_DIR / f"transactions_{timestamp}.csv"
df.to_csv(output_path, index=False)
log_message(f"Transactions data ingested: {output_path}")
except Exception as e:
logging.error(f"Failed to ingest transactions: {e}")
def ingest_web_logs():
"""Fetch web logs from mock API."""
try:
response = requests.get(WEB_API_URL, timeout=10)
response.raise_for_status()
data = response.json()
df = pd.DataFrame(data)
main*+ 0 0 Git Graph
Duo Ln 1, Col 1 Spaces: 4 UTF-8 LF Python 3.13.7 Prettier

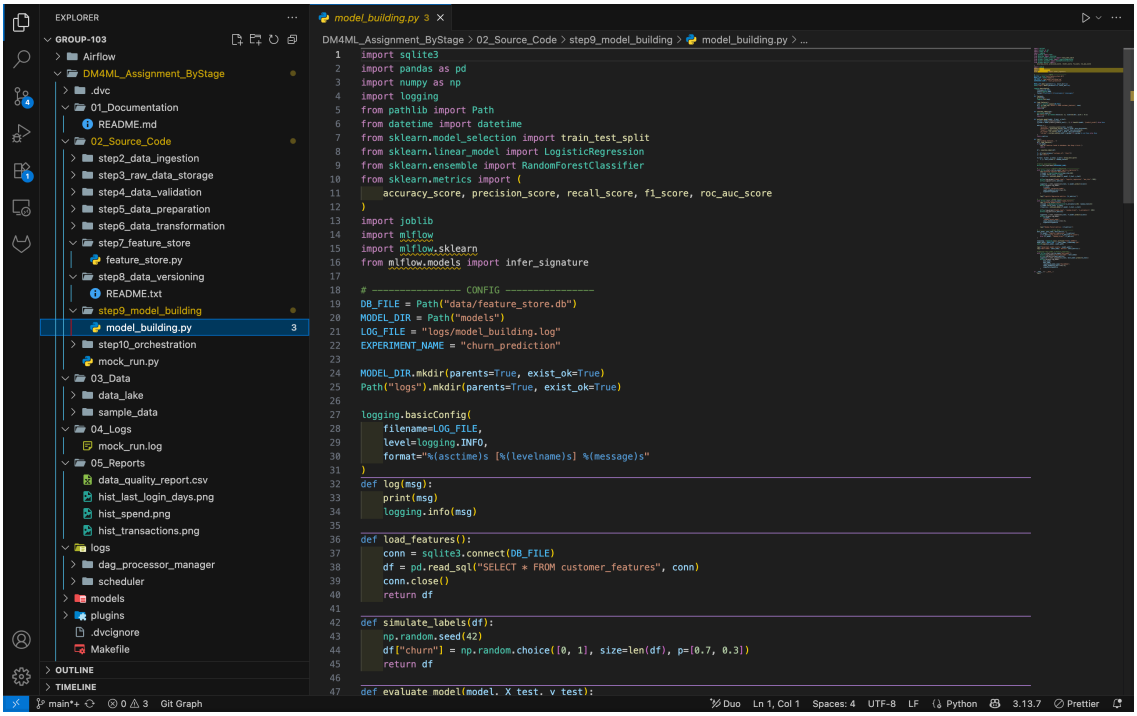

EXPLORER
GROUP-103
Airflow
DM4ML_Assignment_ByStage
.dvc
01_Documentation
README.md
02_Source_Code
step2_data_ingestion
step3_raw_data_storage
storage_raw_upload.py
step4_data_validation
step5_data_preparation
step6_data_transformation
step7_feature_store
step8_data_versioning
step9_model_building
step10_orchestration
mock_run.py
03_Data
data_lake
sample_data
04_Logs
mock_run.log
05_Reports
data_quality_report.csv
hist_last_login_days.png
hist_spend.png
hist_transactions.png
logs
dag_processor_manager
scheduler
models
plugins
.dvcignore
Makefile
run_all.sh
Pipeline_Design_with_Placeholders 2.pdf
OUTLINE
TIMELINE
storage_raw_upload.py
DM4ML_Assignment_ByStage > 02_Source_Code > step3_raw_data_storage > storage_raw_upload.py > ...
import re
import shutil
import logging
from pathlib import Path
from datetime import datetime
# -------- CONFIG --------
RAW_DROP_DIR = Path("data/raw") # where Step 2 wrote raw files
DATA_LAKE_RAW = Path("data_lake/raw") # lake root (local)
LOG_FILE = "logs/storage_upload.log"
# Create dirs
DATA_LAKE_RAW.mkdir(parents=True, exist_ok=True)
Path("logs").mkdir(parents=True, exist_ok=True)
# ------- LOGGING -------
logging.basicConfig(
filename=LOG_FILE,
level=logging.INFO,
format="%(asctime)s [%(levelname)s] %(message)s"
)
def log(msg):
print(msg); logging.info(msg)
# -------- HELPERS --------
FILENAME_PATTERNS = {
"transactions": re.compile(r"^transactions_(\d{8}_\d{6})\.csv$"),
"web_logs": re.compile(r"^web_logs_(\d{8}_\d{6})\.csv$"),
}
def detect_source_and_runid(file_name: str):
for source, pattern in FILENAME_PATTERNS.items():
m = pattern.match(file_name)
if m:
run_id = m.group(1) # YYYYMMDD_HHMMSS from Step 2
return source, run_id
return None, None
def run():
files = [p for p in RAW_DROP_DIR.glob("*.csv")]
if not files:
log("No raw files found to upload.")
return
for fpath in files:
source, run_id = detect_source_and_runid(fpath.name)
if not source:
main*+ 0 0 Git Graph
Duo Ln 1, Col 1 Spaces: 4 UTF-8 LF Python 3.13.7 Prettier

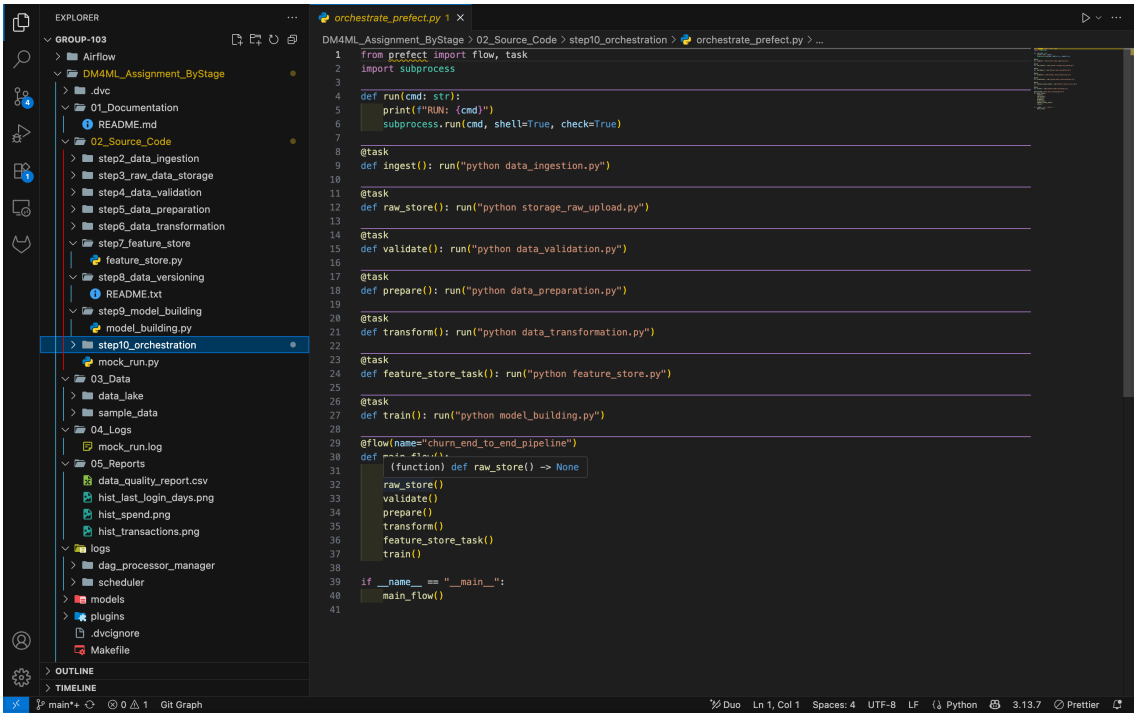

EXPLORER
GROUP-103
Airflow
DM4ML_Assignment_ByStage
.dvc
01_Documentation
README.md
02_Source_Code
step2_data_ingestion
step3_raw_data_storage
storage_raw_upload.py
step4_data_validation
data_validation.py
step5_data_preparation
step6_data_transformation
step7_feature_store
step8_data_versioning
step9_model_building
step10_orchestration
mock_run.py
03_Data
data_lake
sample_data
04_Logs
mock_run.log
05_Reports
data_quality_report.csv
hist_last_login_days.png
hist_spend.png
hist_transactions.png
logs
dag_processor_manager
scheduler
models
plugins
.dvcignore
Makefile
run_all.sh
OUTLINE
TIMELINE
data_validation.py
DM4ML_Assignment_ByStage > 02_Source_Code > step4_data_validation > data_validation.py > ...
import pandas as pd
import logging
from pathlib import Path
from datetime import datetime
# ---------- CONFIG ----------
DATA_LAKE_RAW = Path("data_lake/raw")
LOG_FILE = "logs/data_validation.log"
REPORT_FILE = Path("reports/data_quality_report.csv")
# Expected schema definitions for validation
EXPECTED_SCHEMAS = {
"transactions": {
"customer_id": "int64",
"transaction_id": "object",
"transaction_date": "object",
"amount": "float64",
"payment_method": "object",
"transaction_status": "object"
},
"web_logs": {
"userId": "int64",
"id": "int64",
"title": "object",
"body": "object"
}
}
# Create necessary directories
Path("logs").mkdir(parents=True, exist_ok=True)
REPORT_FILE.parent.mkdir(parents=True, exist_ok=True)
# Logging setup
logging.basicConfig(
filename=LOG_FILE,
level=logging.INFO,
format="%(asctime)s [%(levelname)s] %(message)s"
)
def log(msg):
print(msg)
logging.info(msg)
def validate_file(file_path: Path, source: str):
df = pd.read_csv(file_path)
issues = []
Pylance has detected type annotations in your code and recommends enabling type checking. Would you like to change this setting?
Source: Pylance
Yes
No
main*+
0 0
Git Graph
Duo
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Python
3.13.7
Prettier

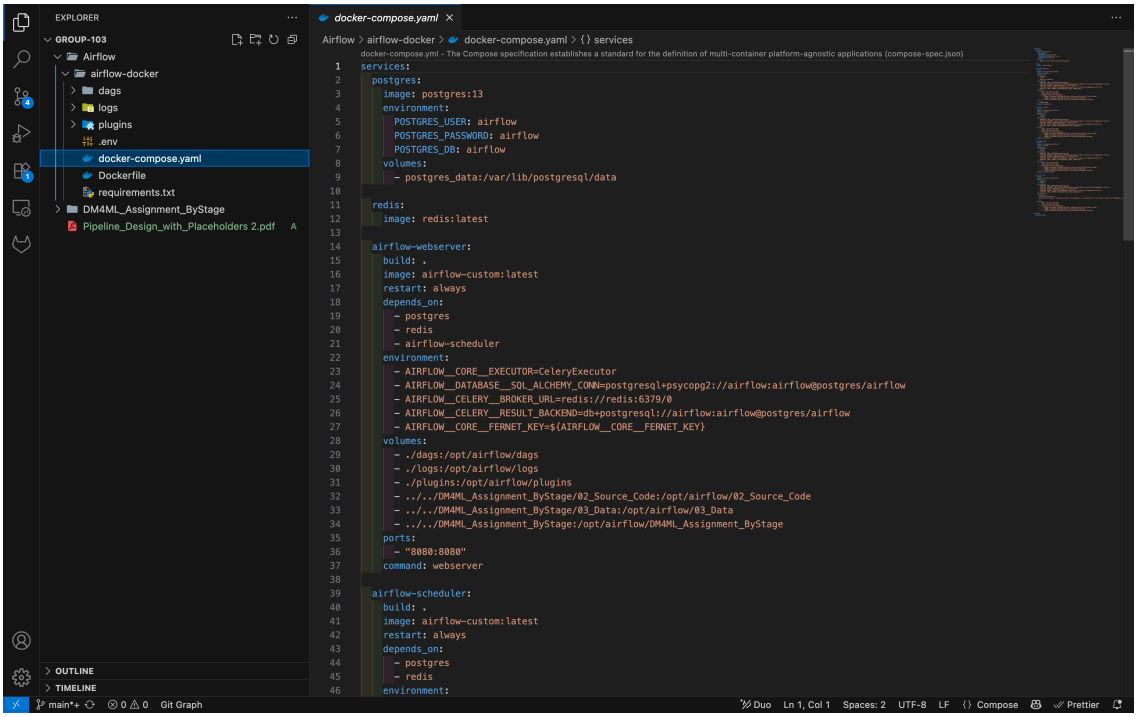

EXPLORER
GROUP-103
Airflow
DM4ML_Assignment_ByStage
.dvc
01_Documentation
README.md
02_Source_Code
step2_data_ingestion
step3_raw_data_storage
storage_raw_upload.py
step4_data_validation
data_validation.py
step5_data_preparation
data_preparation.py
step6_data_transformation
step7_feature_store
step8_data_versioning
step9_model_building
step10_orchestration
mock_run.py
03_Data
data_lake
sample_data
04_Logs
mock_run.log
05_Reports
data_quality_report.csv
hist_last_login_days.png
hist_spend.png
hist_transactions.png
logs
dag_processor_manager
scheduler
models
plugins
.dvcignore
Makefile
OUTLINE
TIMELINE
data_preparation.py
DM4ML_Assignment_ByStage > 02_Source_Code > step5_data_preparation > data_preparation.py > ...
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
import logging
import os
import pandas as pd
# ---------- CONFIG ----------
DATA_LAKE_RAW = Path("data_lake/raw/source=transactions") # We'll use transactions as example
CLEAN_DATA_DIR = Path("data/clean")
LOG_FILE = "logs/data_preparation.log"
# Create dirs
CLEAN_DATA_DIR.mkdir(parents=True, exist_ok=True)
Path("logs").mkdir(parents=True, exist_ok=True)
Path("reports").mkdir(parents=True, exist_ok=True)
# Logging
logging.basicConfig(
filename=LOG_FILE,
level=logging.INFO,
format="%(asctime)s [%(levelname)s] %(message)s"
)
def log(msg):
print(msg)
logging.info(msg)
def load_latest_transactions():
data_dir = "/opt/airflow/03_Data/sample_data"
files = [f for f in os.listdir(data_dir) if f.endswith(".csv")]
if not files:
raise FileNotFoundError("No transaction files found.")
latest_file = max(
[os.path.join(data_dir, f) for f in files],
key=os.path.getctime
)
print(f"Loading file: {latest_file}")
return pd.read_csv(latest_file)
def clean_data(df):
# Handle missing values
for col in df.columns:
if df[col].dtype == "object":
df[col] = df[col].fillna("Unknown")
else:
main*+
0 0
Git Graph
Duo
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Python
3.13.7
Prettier

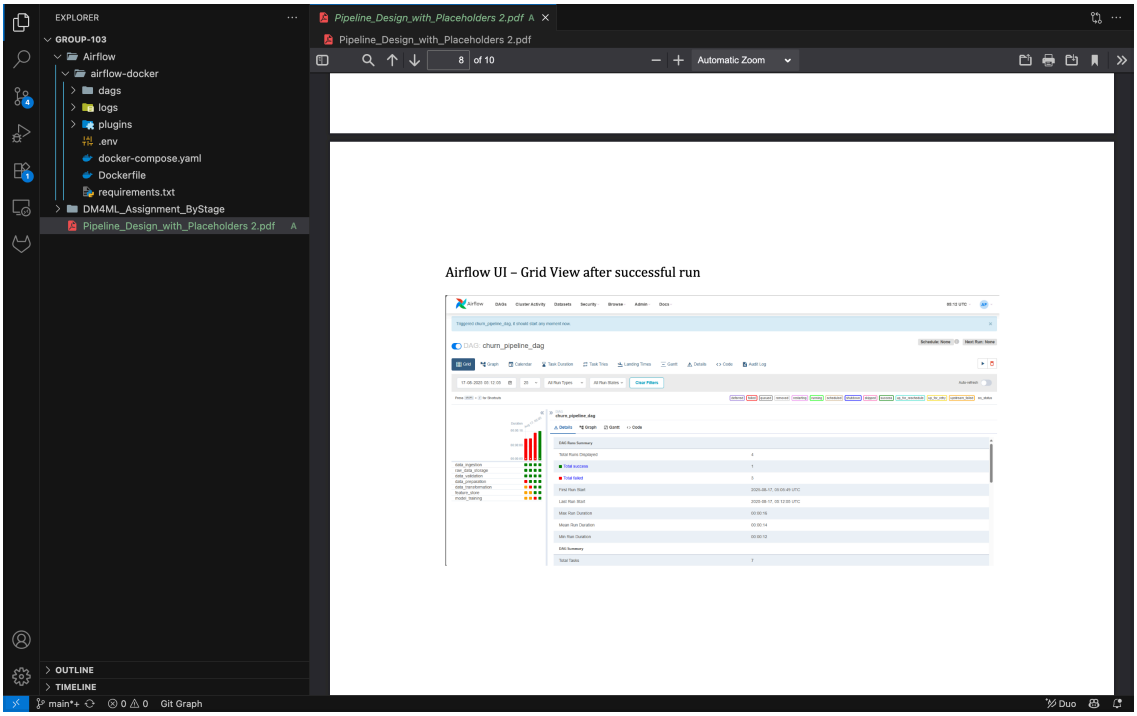

EXPLORER
GROUP-103
Airflow
DM4ML_Assignment_ByStage
.dvc
01_Documentation
README.md
02_Source_Code
step2_data_ingestion
step3_raw_data_storage
step4_data_validation
step5_data_preparation
step6_data_transformation
data_transformation.py
step7_feature_store
step8_data_versioning
step9_model_building
step10_orchestration
mock_run.py
03_Data
data_lake
sample_data
04_Logs
mock_run.log
05_Reports
data_quality_report.csv
hist_last_login_days.png
hist_spend.png
hist_transactions.png
logs
dag_processor_manager
scheduler
models
plugins
.dvcignore
Makefile
run_all.sh
Pipeline_Design_with_Placeholders 2.pdf
OUTLINE
TIMELINE
data_transformation.py
DM4ML_Assignment_ByStage > 02_Source_Code > step6_data_transformation > data_transformation.py > ...
import pandas as pd
import sqlite3
from pathlib import Path
import logging
from datetime import datetime
# ---------- CONFIG ----------
CLEAN_DATA_FILE = Path("data/clean/transactions_clean.csv")
DB_FILE = Path("data/feature_store.db")
LOG_FILE = "logs/data_transformation.log"
# Create dirs
Path("logs").mkdir(parents=True, exist_ok=True)
Path("data").mkdir(parents=True, exist_ok=True)
# Logging
logging.basicConfig(
filename=LOG_FILE,
level=logging.INFO,
format="%(asctime)s [%(levelname)s] %(message)s"
)
def log(msg):
print(msg)
logging.info(msg)
def load_clean_data():
if not CLEAN_DATA_FILE.exists():
raise FileNotFoundError("Clean data file not found. Run Step 5 first.")
return pd.read_csv(CLEAN_DATA_FILE)
def engineer_features(df):
df["transaction_date"] = pd.to_datetime(df["transaction_date"], errors="coerce")
agg_df = df.groupby("customer_id").agg(
total_spend=("amount", "sum"),
transaction_count=("transaction_id", "count"),
avg_transaction_value=("amount", "mean"),
first_transaction=("transaction_date", "min"),
last_transaction=("transaction_date", "max")
).reset_index()
now = datetime.now()
agg_df["tenure_days"] = (now - agg_df["first_transaction"]).dt.days
agg_df["recency_days"] = (now - agg_df["last_transaction"]).dt.days
agg_df.drop(columns=["first_transaction", "last_transaction"], inplace=True)
return agg_df
main*+
0 0
Git Graph
Duo
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Python
3.13.7
Prettier

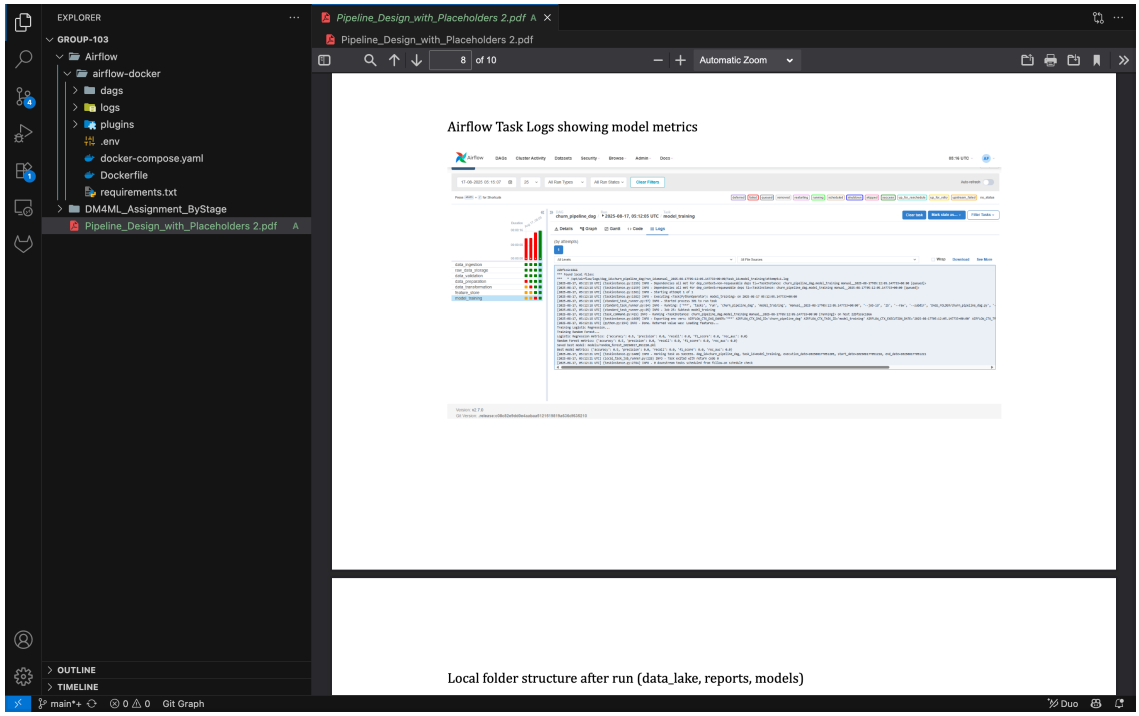

EXPLORER
GROUP-103
Airflow
DM4ML_Assignment_ByStage
.dvc
01_Documentation
README.md
02_Source_Code
step2_data_ingestion
step3_raw_data_storage
step4_data_validation
step5_data_preparation
step6_data_transformation
step7_feature_store
feature_store.py
step8_data_versioning
step9_model_building
step10_orchestration
mock_run.py
03_Data
data_lake
sample_data
04_Logs
mock_run.log
05_Reports
data_quality_report.csv
hist_last_login_days.png
hist_spend.png
hist_transactions.png
logs
dag_processor_manager
scheduler
models
plugins
.dvcignore
Makefile
run_all.sh
Pipeline_Design_with_Placeholders 2.pdf
OUTLINE
TIMELINE
feature_store.py
DM4ML_Assignment_ByStage > 02_Source_Code > step7_feature_store > feature_store.py > ...
import sqlite3
from datetime import datetime
import pandas as pd
from pathlib import Path
import logging
# ---------- CONFIG ----------
DB_FILE = Path("data/feature_store.db")
LOG_FILE = "logs/feature_store.log"
Path("logs").mkdir(parents=True, exist_ok=True)
logging.basicConfig(
filename=LOG_FILE,
level=logging.INFO,
format="%(asctime)s [%(levelname)s] %(message)s"
)
def log(msg):
print(msg)
logging.info(msg)
def init_metadata():
conn = sqlite3.connect(DB_FILE)
cursor = conn.cursor()
cursor.execute("""
CREATE TABLE IF NOT EXISTS feature_metadata (
feature_name TEXT PRIMARY KEY,
description TEXT,
source TEXT,
version INTEGER,
created_at TEXT
)
""")
conn.commit()
conn.close()
log("Feature metadata table initialized.")
def register_features(metadata):
conn = sqlite3.connect(DB_FILE)
cursor = conn.cursor()
for feature_name, details in metadata.items():
cursor.execute("""
INSERT INTO feature_metadata (feature_name, description, source, version, created_at)
VALUES (?, ?, ?, ?, ?)
ON CONFLICT(feature_name) DO UPDATE SET
description=excluded.description,
source=excluded.source,
main*+
0 0
Git Graph
Duo
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Python
3.13.7
Prettier


EXPLORER
GROUP-103
Airflow
DM4ML_Assignment_ByStage
.dvc
01_Documentation
README.md
02_Source_Code
step2_data_ingestion
step3_raw_data_storage
step4_data_validation
step5_data_preparation
step6_data_transformation
step7_feature_store
feature_store.py
step8_data_versioning
README.txt
step9_model_building
model_building.py 3
step10_orchestration
mock_run.py
03_Data
data_lake
sample_data
04_Logs
mock_run.log
05_Reports
data_quality_report.csv
hist_last_login_days.png
hist_spend.png
hist_transactions.png
logs
dag_processor_manager
scheduler
models
plugins
.dvcignore
Makefile
OUTLINE
TIMELINE
model_building.py 3
DM4ML_Assignment_ByStage > 02_Source_Code > step9_model_building > model_building.py > ...
import sqlite3
import pandas as pd
import numpy as np
import logging
from pathlib import Path
from datetime import datetime
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
)
import joblib
import mlflow
import mlflow.sklearn
from mlflow.models import infer_signature
# ---------------- CONFIG ----------------
DB_FILE = Path("data/feature_store.db")
MODEL_DIR = Path("models")
LOG_FILE = "logs/model_building.log"
EXPERIMENT_NAME = "churn_prediction"
MODEL_DIR.mkdir(parents=True, exist_ok=True)
Path("logs").mkdir(parents=True, exist_ok=True)
logging.basicConfig(
filename=LOG_FILE,
level=logging.INFO,
format="%(asctime)s [%(levelname)s] %(message)s"
)
def log(msg):
print(msg)
logging.info(msg)
def load_features():
conn = sqlite3.connect(DB_FILE)
df = pd.read_sql("SELECT * FROM customer_features", conn)
conn.close()
return df
def simulate_labels(df):
np.random.seed(42)
df["churn"] = np.random.choice([0, 1], size=len(df), p=[0.7, 0.3])
return df
def evaluate model(model. X test. v test):
main+* 0 3 Git Graph
Duo Ln 1, Col 1 Spaces: 4 UTF-8 LF Python 3.13.7 Prettier


EXPLORER
GROUP-103
Airflow
DM4ML_Assignment_ByStage
.dvc
01_Documentation
README.md
02_Source_Code
step2_data_ingestion
step3_raw_data_storage
step4_data_validation
step5_data_preparation
step6_data_transformation
step7_feature_store
feature_store.py
step8_data_versioning
README.txt
step9_model_building
model_building.py
step10_orchestration
mock_run.py
03_Data
data_lake
sample_data
04_Logs
mock_run.log
05_Reports
data_quality_report.csv
hist_last_login_days.png
hist_spend.png
hist_transactions.png
logs
dag_processor_manager
scheduler
models
plugins
.dvcignore
Makefile
OUTLINE
TIMELINE
orchestrate_prefect.py 1
DM4ML_Assignment_ByStage > 02_Source_Code > step10_orchestration > orchestrate_prefect.py > ...
from prefect import flow, task
import subprocess
def run(cmd: str):
print(f"RUN: {cmd}")
subprocess.run(cmd, shell=True, check=True)
@task
def ingest(): run("python data_ingestion.py")
@task
def raw_store(): run("python storage_raw_upload.py")
@task
def validate(): run("python data_validation.py")
@task
def prepare(): run("python data_preparation.py")
@task
def transform(): run("python data_transformation.py")
@task
def feature_store_task(): run("python feature_store.py")
@task
def train(): run("python model_building.py")
@flow(name="churn_end_to_end_pipeline")
def main_flow():
(function) def raw_store() -> None
raw_store()
validate()
prepare()
transform()
feature_store_task()
train()
if __name__ == "__main__":
main_flow()
main+* 0 1 Git Graph
Duo Ln 1, Col 1 Spaces: 4 UTF-8 LF Python 3.13.7 Prettier


EXPLORER
GROUP-103
Airflow
airflow-docker
dags
logs
plugins
.env
docker-compose.yaml
Dockerfile
requirements.txt
DM4ML_Assignment_ByStage
Pipeline_Design_with_Placeholders 2.pdf
docker-compose.yaml
Airflow > airflow-docker > docker-compose.yaml > { } services
docker-compose.yml - The Compose specification establishes a standard for the definition of multi-container platform-agnostic applications (compose-spec.json)
services:
postgres:
image: postgres:13
environment:
POSTGRES_USER: airflow
POSTGRES_PASSWORD: airflow
POSTGRES_DB: airflow
volumes:
- postgres_data:/var/lib/postgresql/data
redis:
image: redis:latest
airflow-webserver:
build: .
image: airflow-custom:latest
restart: always
depends_on:
- postgres
- redis
- airflow-scheduler
environment:
- AIRFLOW__CORE__EXECUTOR=CeleryExecutor
- AIRFLOW__DATABASE__SQL_ALCHEMY_CONN=postgresql+psycopg2://airflow:airflow@postgres/airflow
- AIRFLOW__CELERY__BROKER_URL=redis://redis:6379/0
- AIRFLOW__CELERY__RESULT_BACKEND=db+postgresql://airflow:airflow@postgres/airflow
- AIRFLOW__CORE__FERNET_KEY=${AIRFLOW__CORE__FERNET_KEY}
volumes:
- ./dags:/opt/airflow/dags
- ./logs:/opt/airflow/logs
- ./plugins:/opt/airflow/plugins
- ../../DM4ML_Assignment_ByStage/02_Source_Code:/opt/airflow/02_Source_Code
- ../../DM4ML_Assignment_ByStage/03_Data:/opt/airflow/03_Data
- ../../DM4ML_Assignment_ByStage:/opt/airflow/DM4ML_Assignment_ByStage
ports:
- "8080:8080"
command: webserver
airflow-scheduler:
build: .
image: airflow-custom:latest
restart: always
depends_on:
- postgres
- redis
environment:
OUTLINE
TIMELINE
main*+
Git Graph
Duo
Ln 1, Col 1
Spaces: 2
UTF-8
LF
Compose
Prettier


EXPLORER
GROUP-103
Airflow
airflow-docker
dags
logs
plugins
.env
docker-compose.yaml
Dockerfile
requirements.txt
DM4ML_Assignment_ByStage
Pipeline_Design_with_Placeholders 2.pdf
Pipeline_Design_with_Placeholders 2.pdf
Pipeline_Design_with_Placeholders 2.pdf
8 of 10
Automatic Zoom
Airflow UI – Grid View after successful run
churn_pipeline_dag
OUTLINE
TIMELINE
main*+
Git Graph
Duo


EXPLORER
GROUP-103
Airflow
airflow-docker
dags
logs
plugins
.env
docker-compose.yaml
Dockerfile
requirements.txt
DM4ML_Assignment_ByStage
Pipeline_Design_with_Placeholders 2.pdf
Pipeline_Design_with_Placeholders 2.pdf
Pipeline_Design_with_Placeholders 2.pdf
8 of 10
Automatic Zoom
Airflow Task Logs showing model metrics
Local folder structure after run (data_lake, reports, models)
OUTLINE
TIMELINE
main*+
0 0
Git Graph
Duo