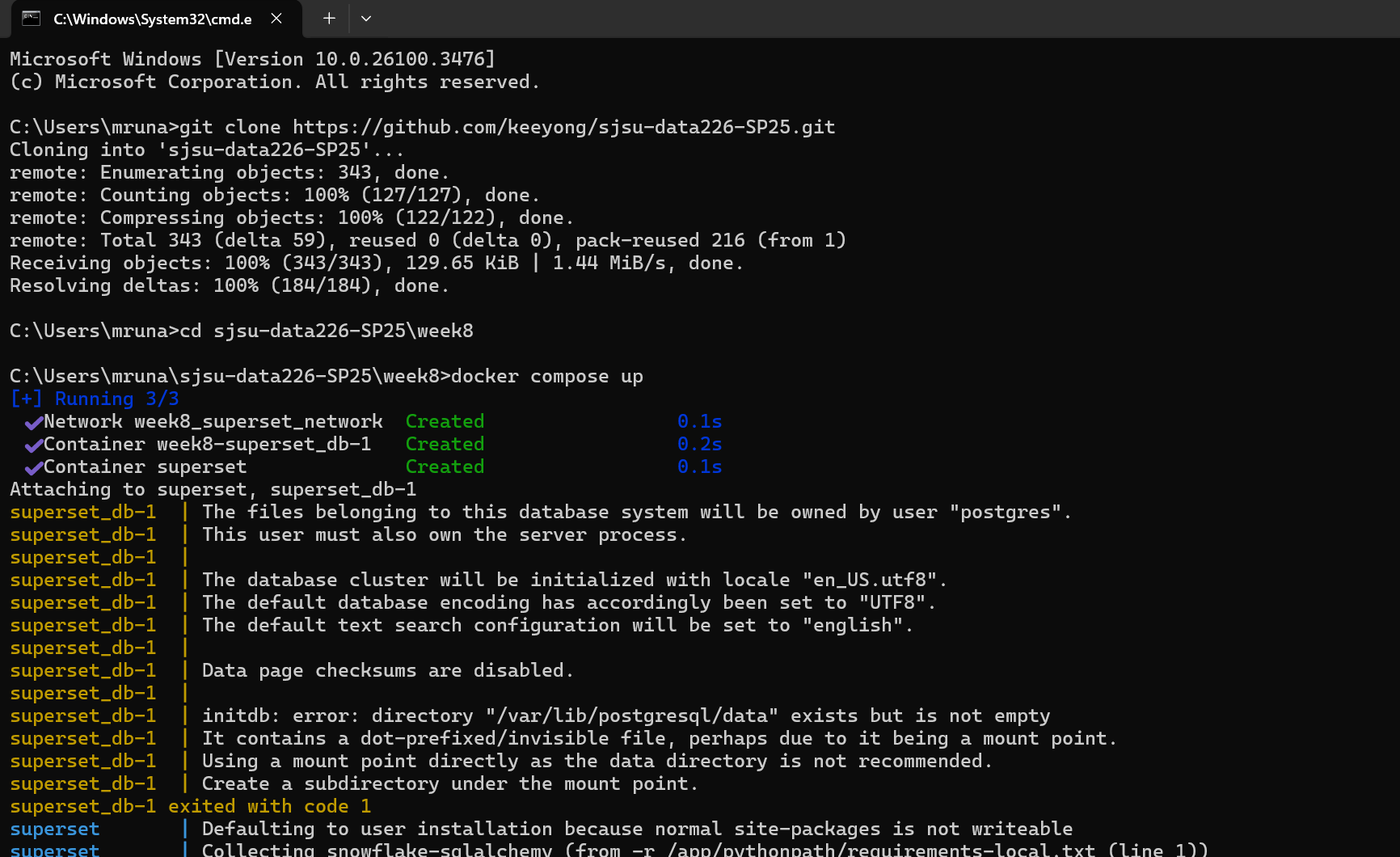
Name: Mrunali Katta

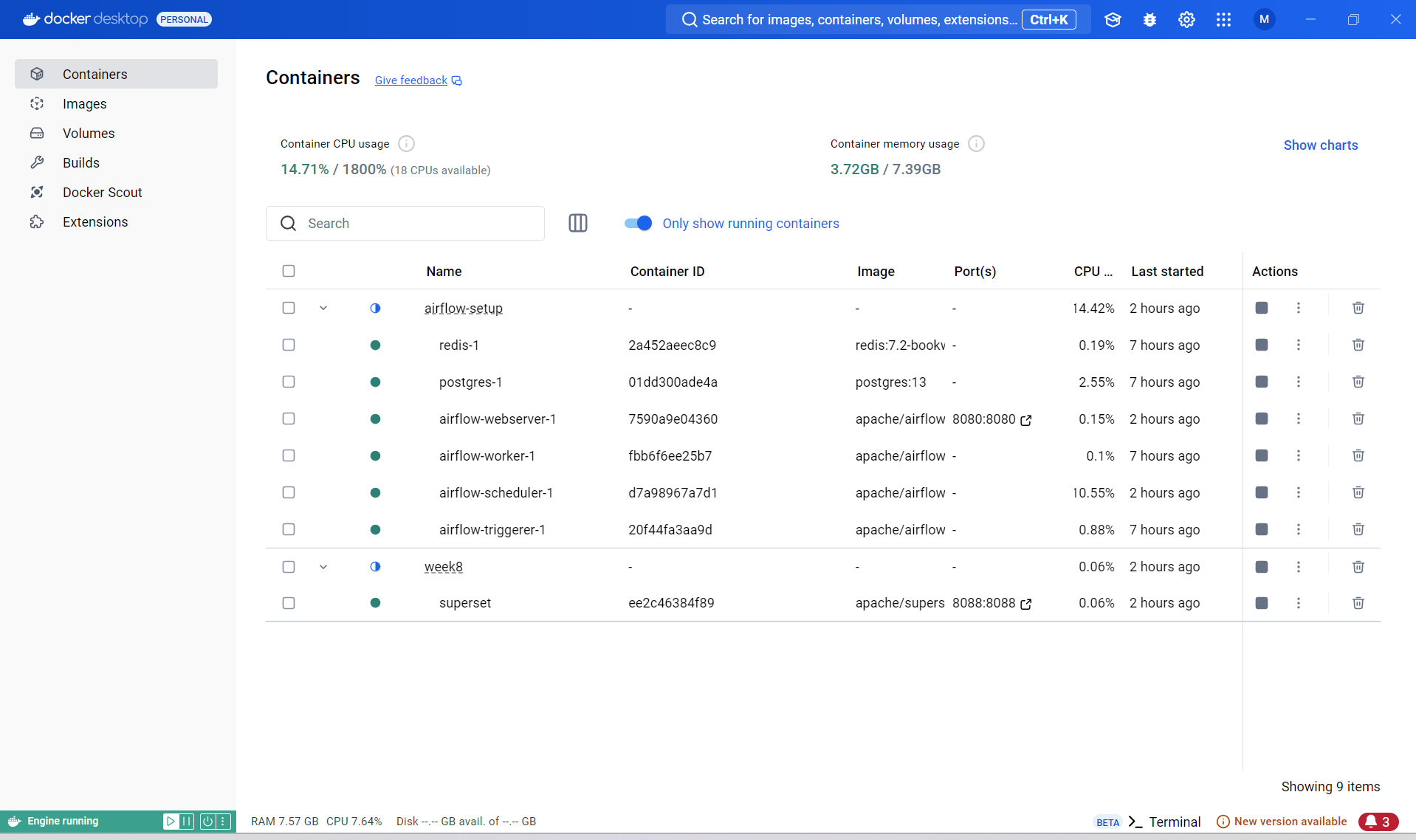
ID: 017516785

Github files link:

**Homework 06**

1. Import [two tables in your SnowflakeLinks to an external site.](https://github.com/keeyong/sjsu-data226-SP25/blob/main/week8/How-to-setup-ETL-tables-for-ELT.md) as an ETL DAG in your Airflow (+3pt)
   1. user\_session\_channel and session\_timestamp (under raw schema or equivalent)
   2. Capture the screenshot of this DAG’s detailed page from the Web UI (#1)

**🡪** Docker



Below is the code for ETL pipeline to create 2 tables in my snowflake:

**etl\_s3.py**

from airflow import DAG

from airflow.models import Variable

from airflow.decorators import task

from airflow.operators.python import get\_current\_context

from airflow.providers.snowflake.operators.snowflake import SnowflakeOperator

from airflow.providers.snowflake.hooks.snowflake import SnowflakeHook

from datetime import datetime

from datetime import timedelta

import logging

import snowflake.connector

# Snowflake connection function

def return\_snowflake\_conn():

    hook = SnowflakeHook(snowflake\_conn\_id='snowflake\_conn')  # 'snowflake\_conn' should be defined in Airflow connections

    conn = hook.get\_conn()

    return conn.cursor()

# Task to create tables

@task

def create\_tables(con, table1, table2):

    try:

        con.execute("BEGIN")

        con.execute(f"""

            CREATE TABLE IF NOT EXISTS {table1} (

                userId INT NOT NULL,

                sessionId VARCHAR(32) PRIMARY KEY,

                channel VARCHAR(32) DEFAULT 'direct'

            );

        """)

        con.execute(f"""

            CREATE TABLE IF NOT EXISTS {table2} (

                sessionId VARCHAR(32) PRIMARY KEY,

                ts TIMESTAMP

            );

        """)

        con.execute("COMMIT")

        return table1, table2

    except Exception as e:

        con.execute("ROLLBACK")

        print(e)

        raise(e)

# Task to load data from S3 to Snowflake

@task

def load\_data(tables, url):

    try:

        con.execute("BEGIN")

        con.execute(f"""

            CREATE OR REPLACE STAGE dev.raw.blob\_stage

            URL = '{url}'

            FILE\_FORMAT = (TYPE = CSV, SKIP\_HEADER = 1, FIELD\_OPTIONALLY\_ENCLOSED\_BY = '"');

        """)

        con.execute(f"""

            COPY INTO {tables[0]} FROM @dev.raw.blob\_stage/user\_session\_channel.csv;

        """)

        con.execute(f"""

            COPY INTO {tables[1]} FROM @dev.raw.blob\_stage/session\_timestamp.csv;

        """)

        con.execute("COMMIT")

    except Exception as e:

        con.execute("ROLLBACK")

        print(e)

        raise(e)

# Define the DAG

with DAG(

    dag\_id='etl\_s3',  # The ID for the DAG

    description='ETL Pipeline from Amazon S3 to Snowflake',  # Description of the DAG

    start\_date=datetime(2024, 10, 6),  # Start date of the DAG

    catchup=False,  # Don't backfill the DAG

    tags=['ETL'],  # Tags for categorizing the DAG

    schedule\_interval='30 15 \* \* \*'  # Schedule the DAG to run at 3:30 PM UTC daily

) as dag:

    # Initialize Snowflake connection

    con = return\_snowflake\_conn()

    # Get the URL from Airflow's Variable

    url = Variable.get("url")  # Ensure the "url" variable is set in Airflow

    # Define table names

    table1 = "dev.raw.user\_session\_channel"

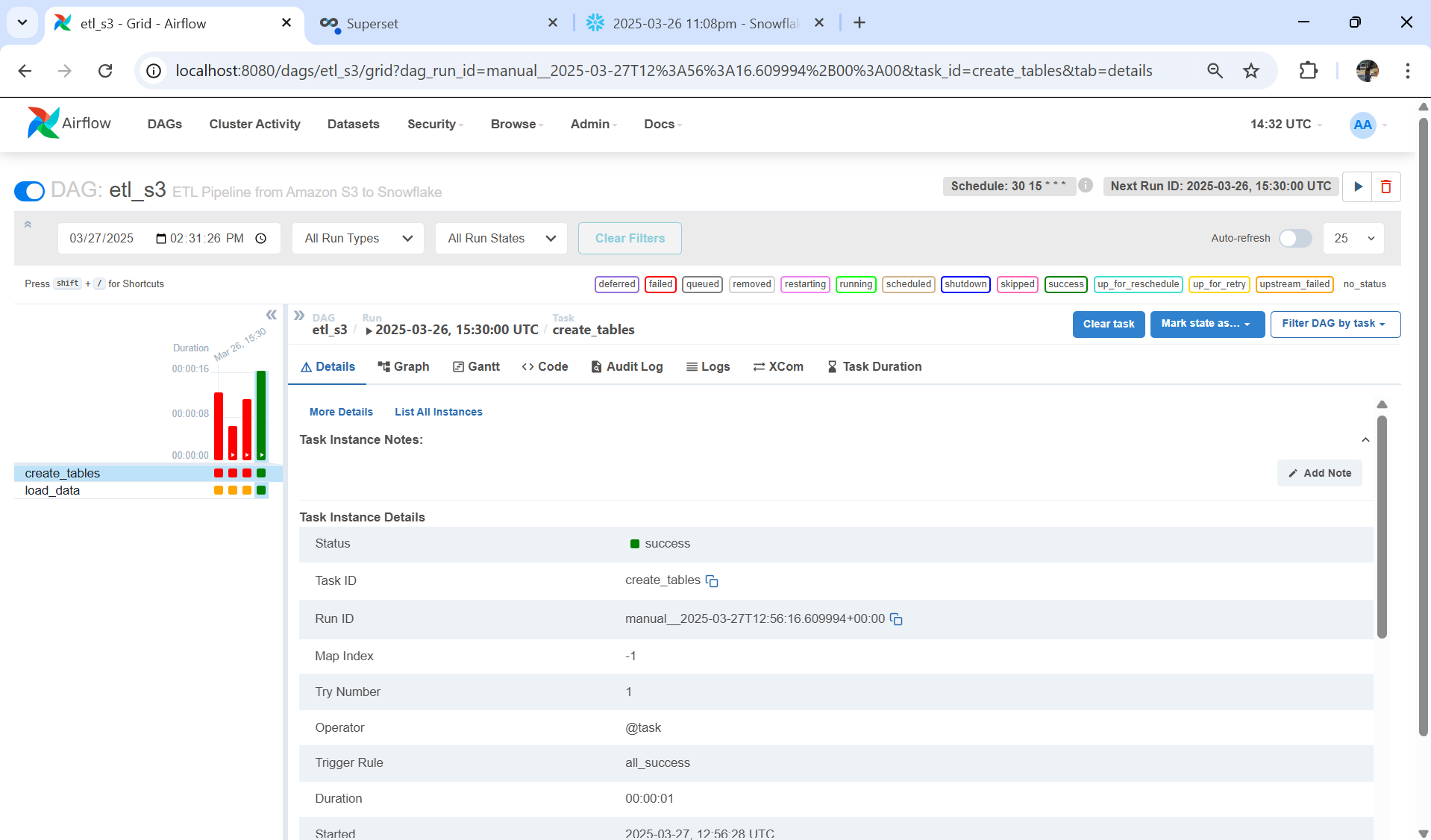
    table2 = "dev.raw.session\_timestamp"

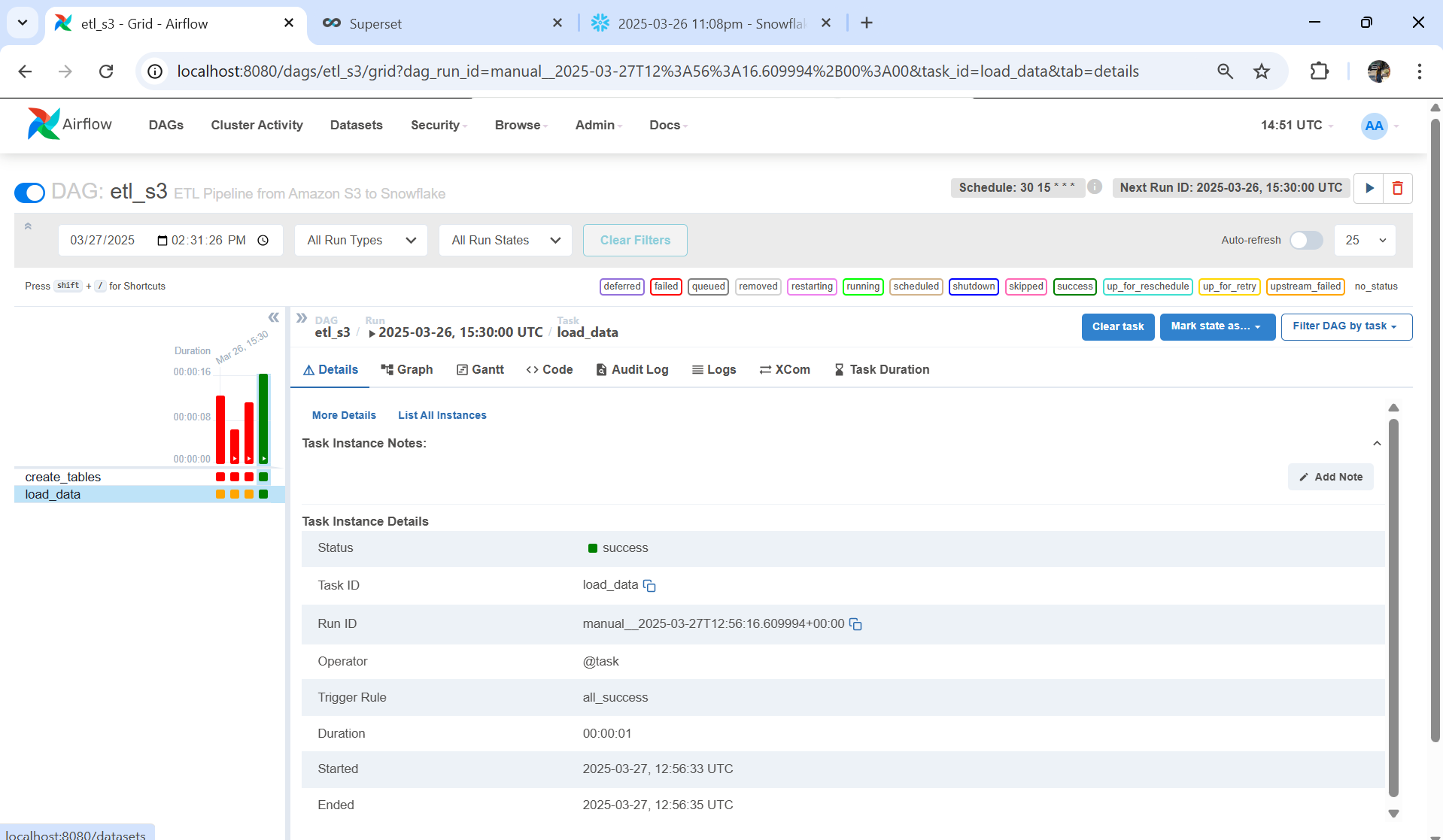
    # Create the tables

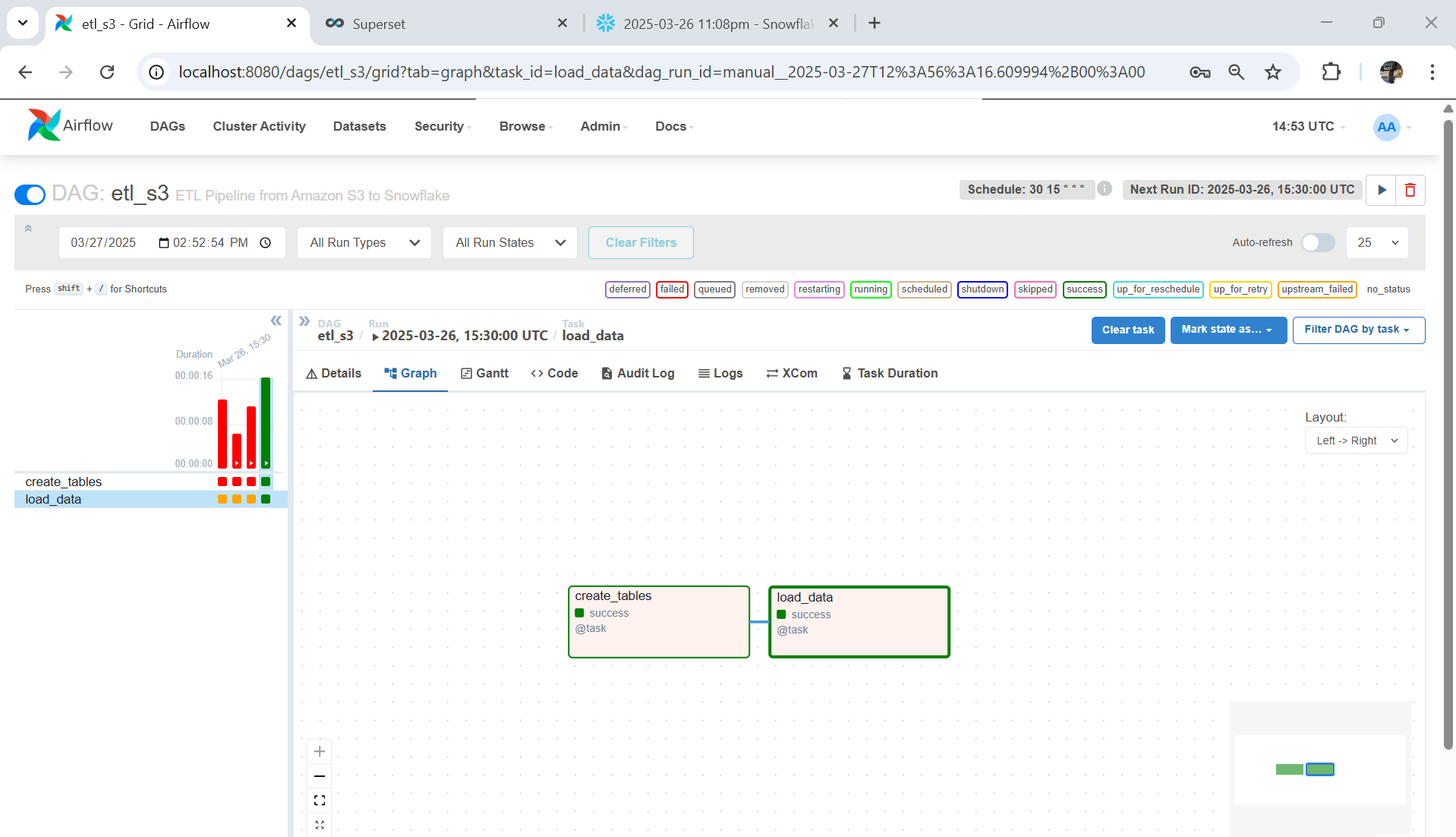
    tables = create\_tables(con, table1, table2)

    # Load the data from S3 into Snowflake

    load = load\_data(tables, url)







1. Create a ELT DAG in your Airflow to create a JOINED table of the two as described during the class (+3pt)
   1. session\_summary (under analytics)
   2. **Extra point**: add one more condition to check duplicate records (+1pt)
   3. Capture the screenshot of this DAG’s detailed page from the Web UI (#2)

* Below is the code to create join of tables in analytics schema and condition to check for duplicate records:

**BuildELT\_CTAS.py**

from airflow import DAG

from airflow.models import Variable

from airflow.decorators import task

from airflow.operators.python import get\_current\_context

from airflow.providers.snowflake.operators.snowflake import SnowflakeOperator

from airflow.providers.snowflake.hooks.snowflake import SnowflakeHook

from datetime import datetime

from datetime import timedelta

import logging

import snowflake.connector

# Function to return a Snowflake connection

def return\_snowflake\_conn():

    hook = SnowflakeHook(snowflake\_conn\_id='snowflake\_conn')

    conn = hook.get\_conn()

    return conn.cursor()

# Task to create or replace the table in Snowflake

@task

def run\_ctas(table, select\_sql, primary\_key=None):

    logging.info(table)

    logging.info(select\_sql)

    cur = return\_snowflake\_conn()

    try:

        cur.execute("BEGIN;")

        sql = f"CREATE OR REPLACE TABLE {table} AS {select\_sql}"

        logging.info(sql)

        cur.execute(sql)

        # Check for duplicates based on the primary key

        if primary\_key is not None:

            sql = f"SELECT {primary\_key}, COUNT(1) AS cnt FROM {table} GROUP BY 1 ORDER BY 2 DESC LIMIT 1"

            logging.info(sql)

            cur.execute(sql)

            result = cur.fetchone()

            logging.info(f"Primary key check: {result}")

            if int(result[1]) > 1:

                logging.error("Primary key uniqueness failed!")

                raise Exception(f"Primary key uniqueness failed: {result}")

        cur.execute("COMMIT;")

    except Exception as e:

        cur.execute("ROLLBACK")

        logging.error(f'Failed to execute SQL. Completed ROLLBACK! Error: {e}')

        raise

# DAG definition for the ELT process

with DAG(

    dag\_id='BuildELT\_CTAS',

    start\_date=datetime(2024, 10, 2),

    catchup=False,

    tags=['ELT'],

    schedule\_interval='45 2 \* \* \*'  # Customize this schedule as needed

) as dag:

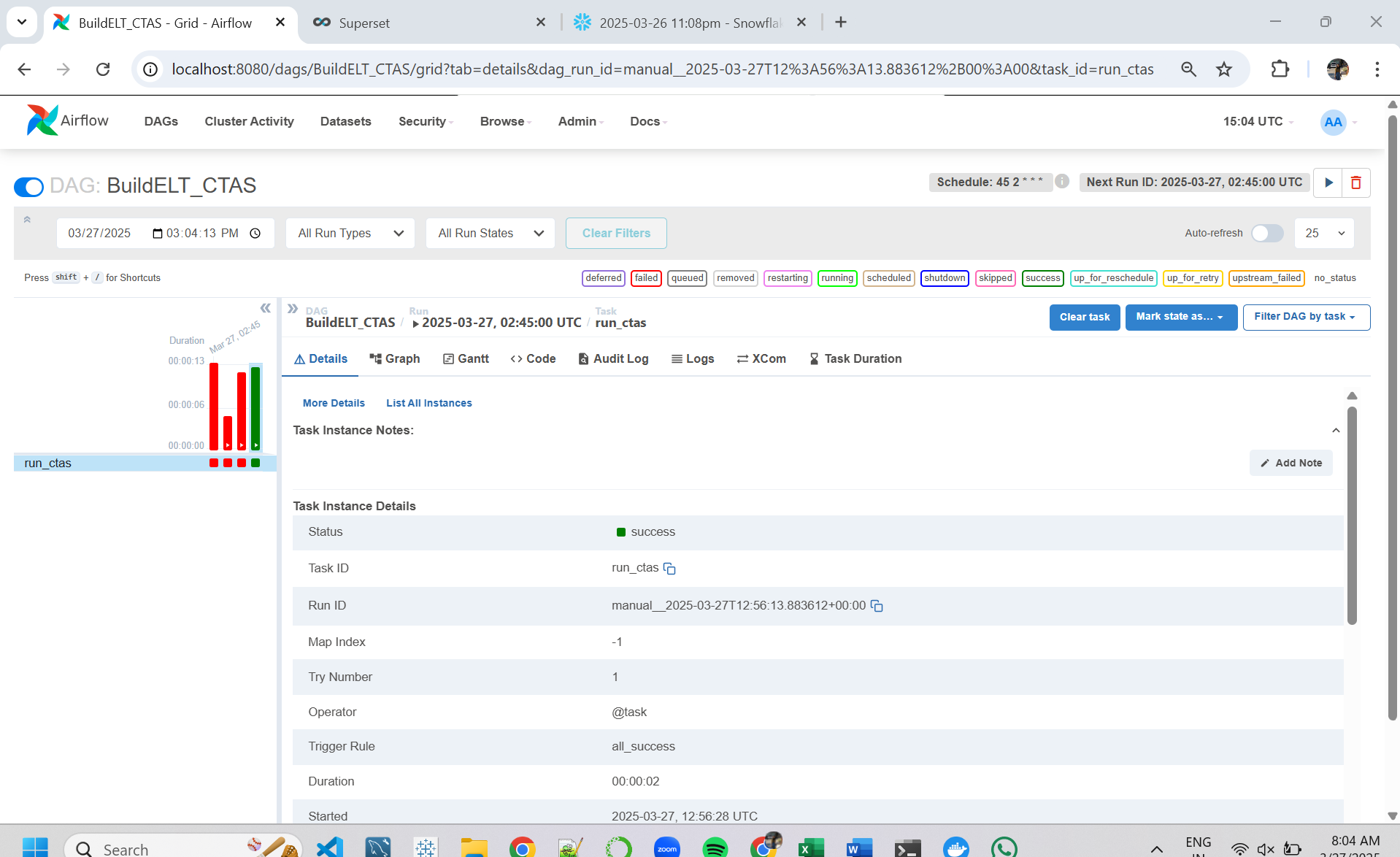
    table = "dev.analytics.session\_summary"

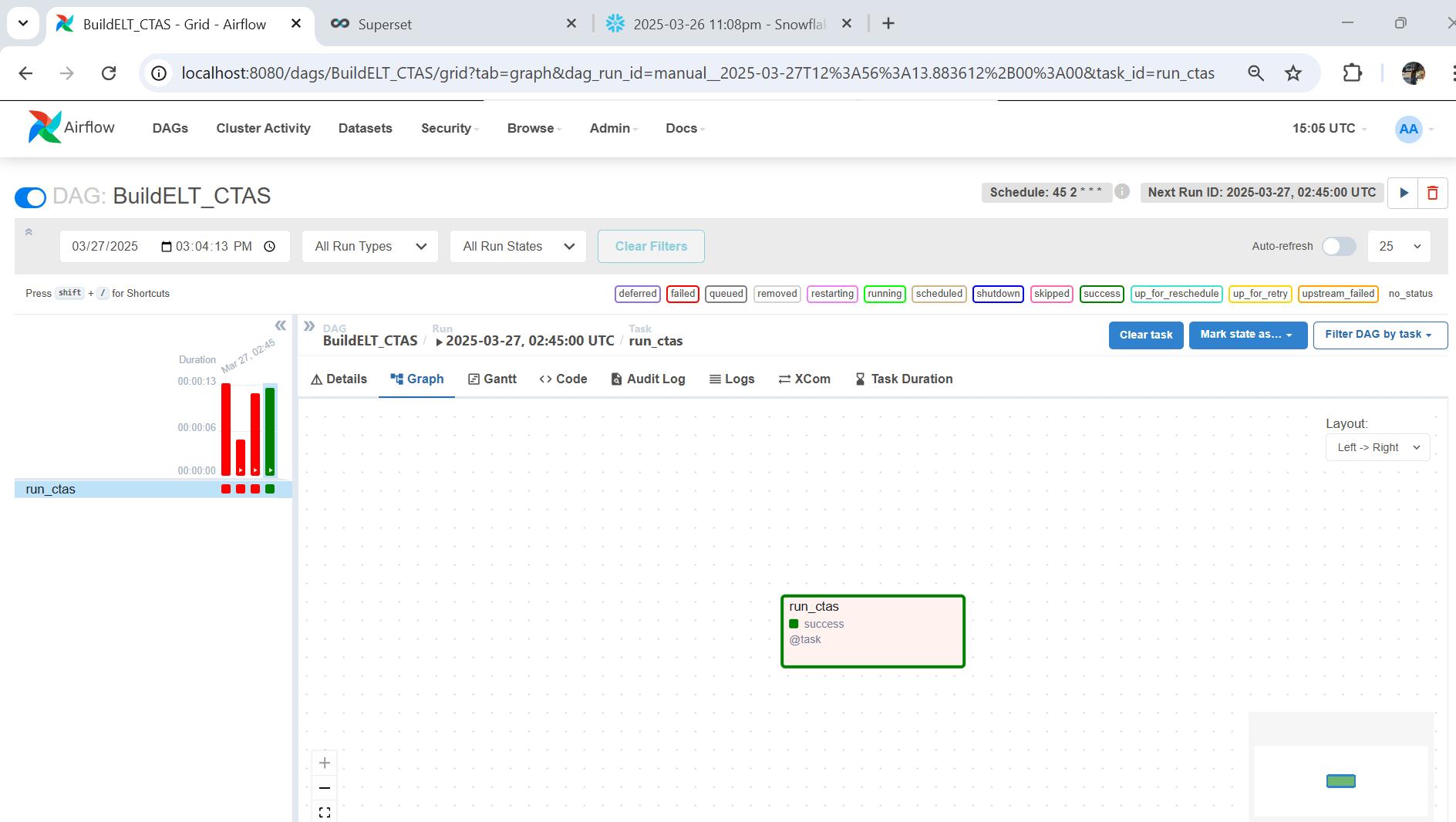
    select\_sql = """SELECT u.\*, s.ts FROM dev.raw.user\_session\_channel u

                    JOIN dev.raw.session\_timestamp s ON u.sessionId = s.sessionId"""

    # Run the task to create the joined table

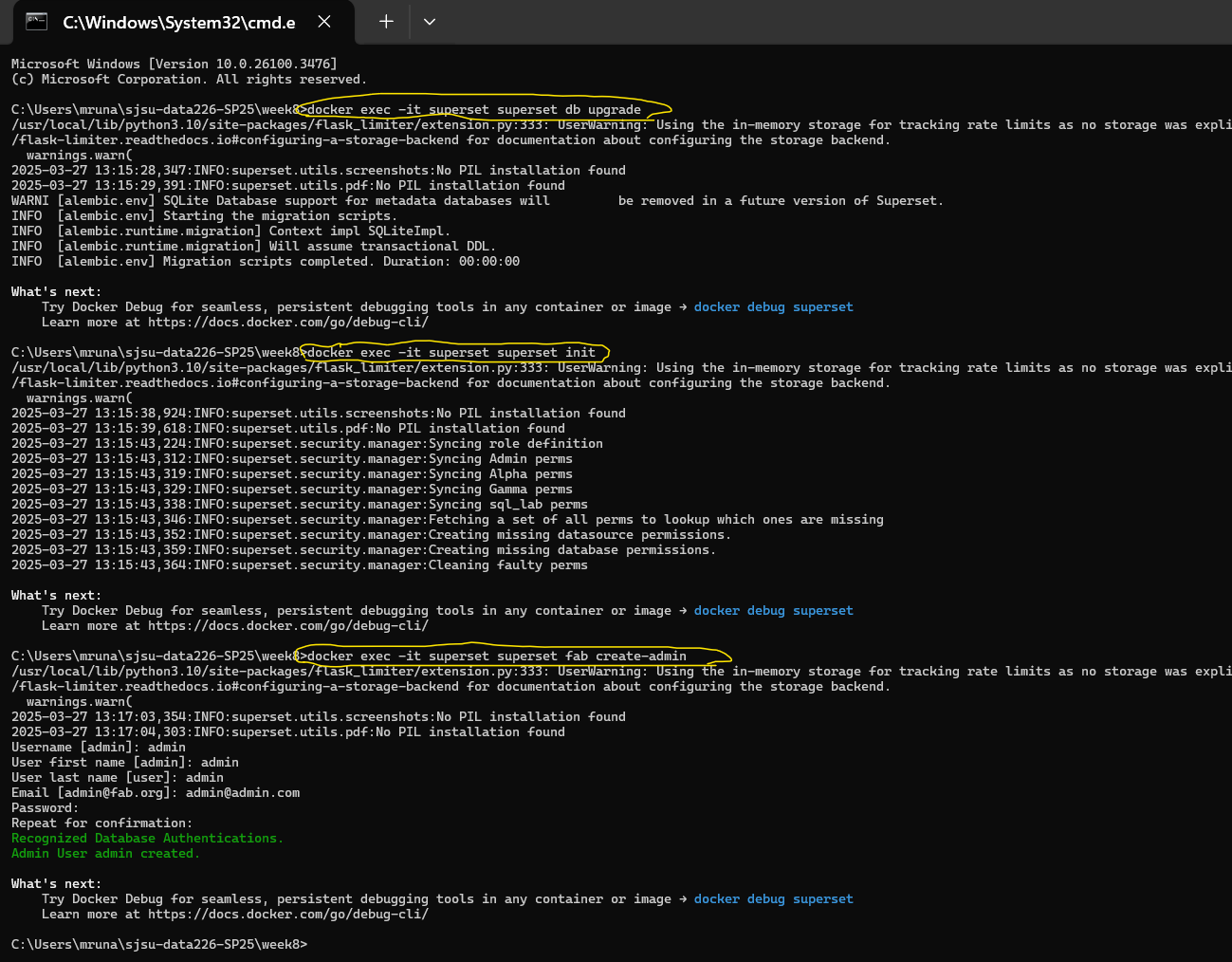
    run\_ctas(table, select\_sql, primary\_key='sessionId')



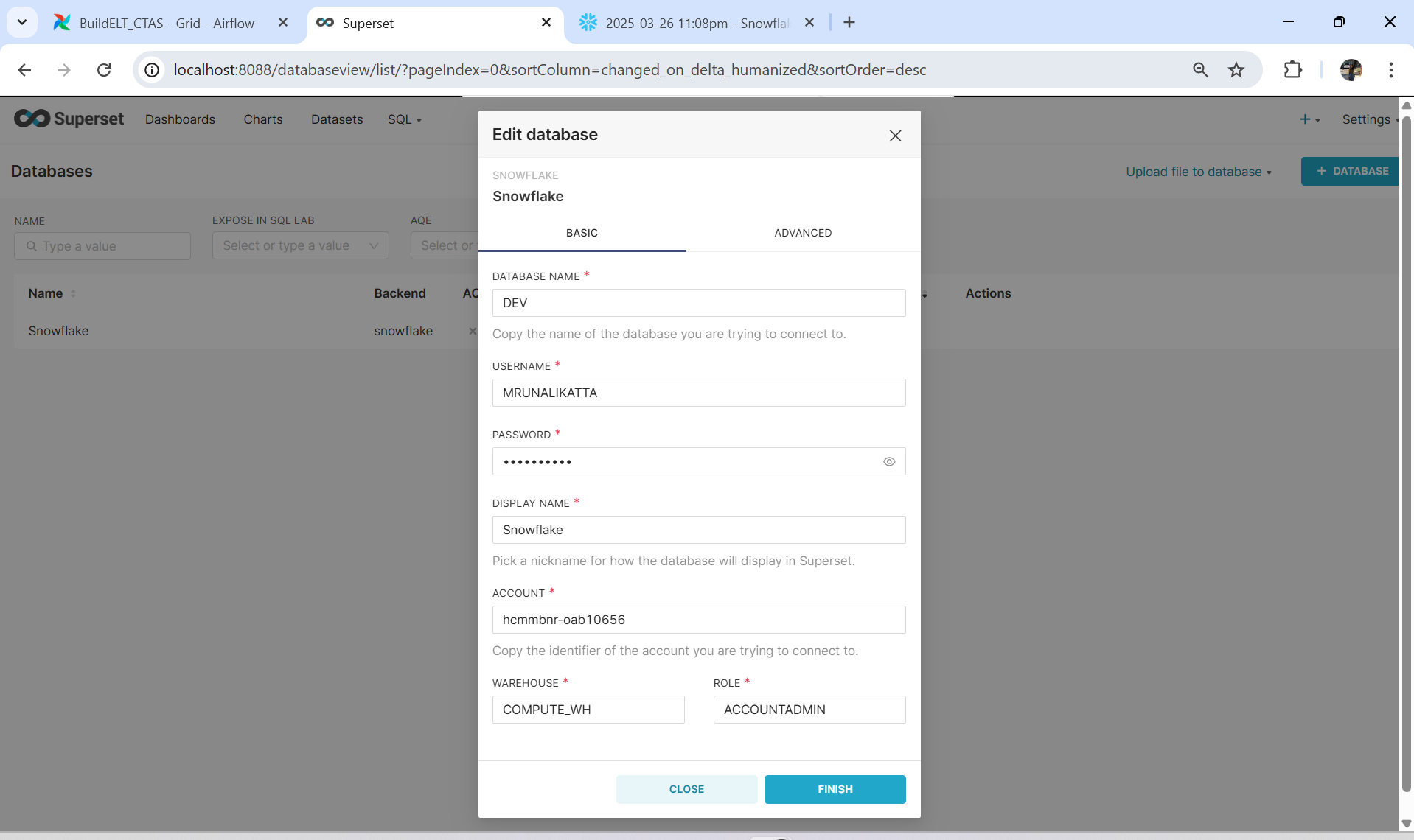


1. Set up your Preset account or [Docker Superset environmentLinks to an external site.](https://github.com/keeyong/sjsu-data226-SP25/blob/main/week8/How-to-Run-Superset-On-Docker.md) or Tableau account (+2pt)
   1. This includes setting up Snowflake connection and import session\_summary from step 2
   2. Capture the screenshot of your Datasets (#3)

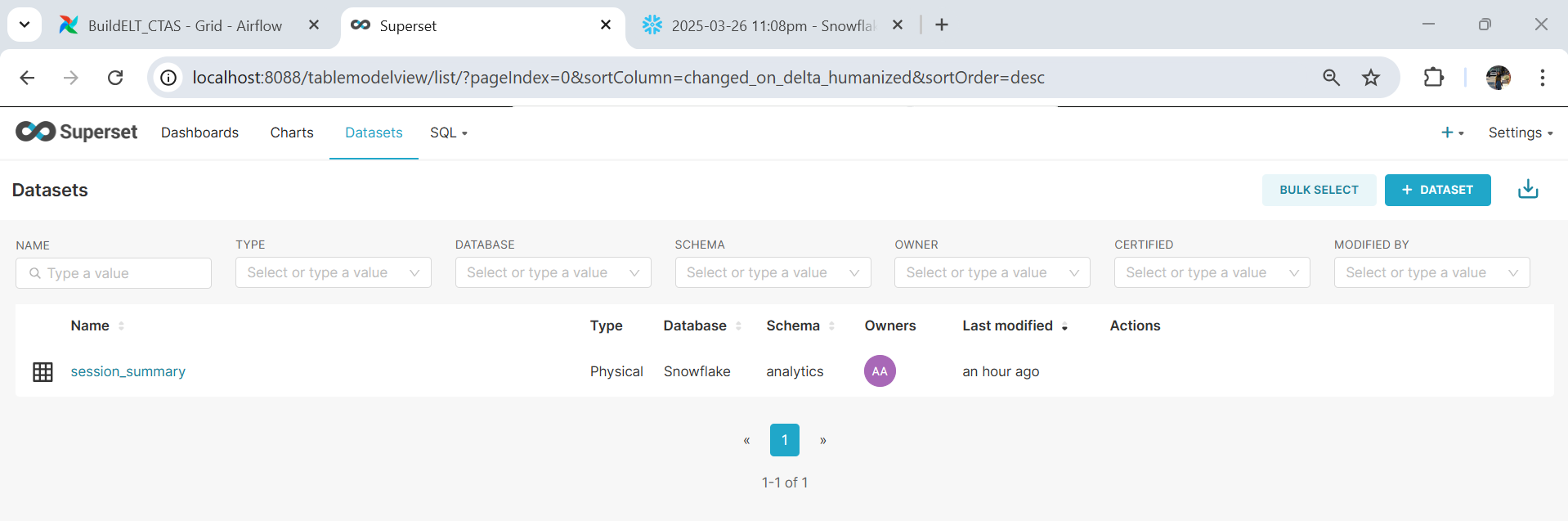
* Below are the commands to setup Docker Superset Environment:



Below is the screenshot of SuperSet opened via Docker in which i am starting to connect my Snowflake account:

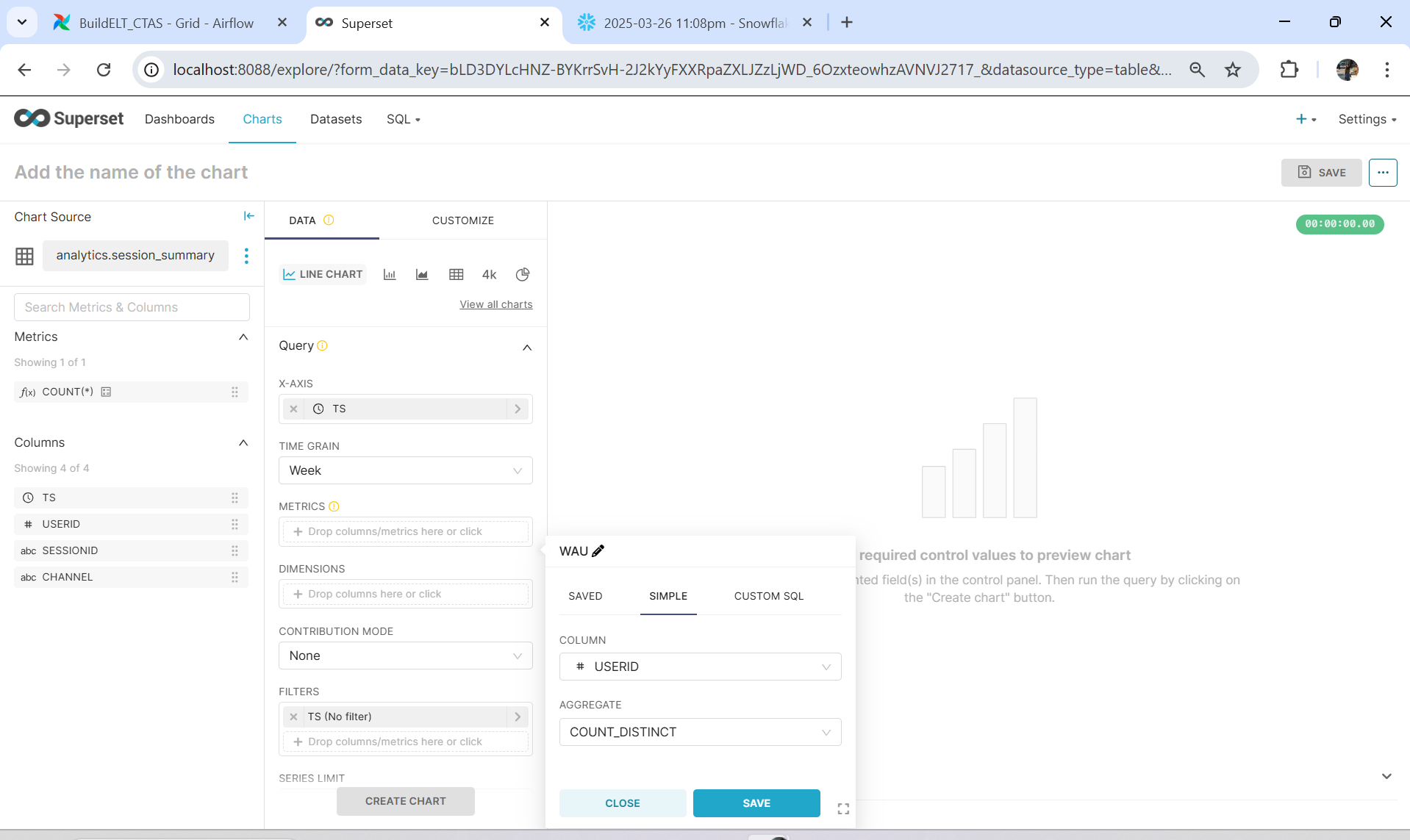


Below are the screenshots of my dataset:



1. Create your WAU chart (+2pt)
   1. Make sure you rename the metrics field to **WAU**
   2. Capture the screenshot of the chart (#4)

* Below is the screenshot where I rename the metric field :



**Line Chart**

