CODE	:	Airflow_Practice_T01_v1.0
TYPE	:	N/A
LOC	:	N/A
DURATION	:	180 MINUTES

Practice Final Exam

1. **Project Requirement:**

The startup **TuneStream** needs an automated system to orchestrate and monitor their data warehouse ETL workflows.

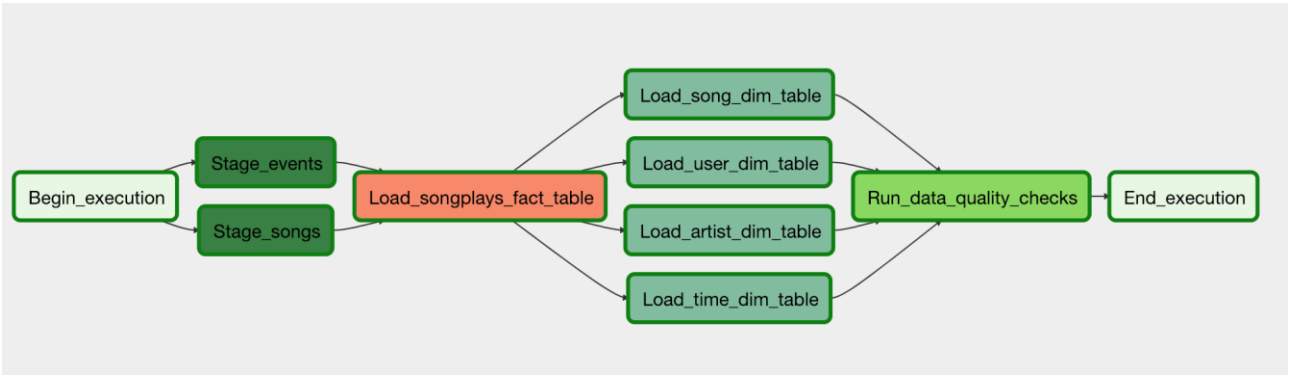
Note: The pipeline supports two configurations:

- Loading data from local storage into a local PostgreSQL database, or
- Loading data from Amazon S3 into Amazon Redshift.

Using **Apache Airflow**, the ETL pipeline extracts JSON data files, processes, and transforms them, and loads the results into a star-schema relational database according to the chosen setup. After the ETL process is finished, data quality checks are conducted to detect any inconsistencies in the datasets.

Your tasks:

- Choose one of the two configurations. If you select S3 as the raw data storage, upload the data files to Amazon S3.
- Write Airflow DAGs to build a pipeline with the following tasks:



- **Task begin_execution**: Use a DummyOperator to signal the start of the workflow.
- **Task Stage_songs**: Load song data into the “staging_songs” table.
- **Task Stage_events**: Load log data into the “staging_events” table.
- **Task Load_songplays_fact_table**: Insert data into the “songplays” fact table.
- **Task Load_song_dim_table**: Insert records into the “songs” dimension table.
- **Task Load_user_dim_table**: Populate the “users” dimension table.
- **Task Load_artist_dim_table**: Insert data into the “artists” dimension table.
- **Task Load_time_dim_table**: Load entries into the “time” dimension table.
- **Task data_quality_checks**: Perform data quality checks after the ETL process.

- **Task End_execution**: Use a DummyOperator to mark the end of the workflow.

Configuring the DAG
In the DAG, add default parameters according to these guidelines

- The DAG does not have dependencies on past runs
- On failure, the task are retried 3 times
- Retries happen every 5 minutes
- Catchup is turned off
- Do not email on retry

2. Dataset:

Song data:

- Each file is in JSON format
- Contains metadata about a song and the artist of that song.
- Files are partitioned by the first three letters of each song's track ID.

example of file paths to two files in song dataset.

```
song_data/A/B/C/TRABCEI128F424C983.json
song_data/A/A/B/TRAABJL12903CDCF1A.json
```

example of single song file, TRAABJL12903CDCF1A.json:

```
{"num_songs": 1,
"artist_id": "ARJIE2Y1187B994AB7",
"artist_latitude": null,
"artist_longitude": null,
"artist_location": "",
"artist_name": "Line Renaud",
"song_id": "SOUPIRU12A6D4FA1E1",
"title": "Der Kleine Dompfaff",
"duration": 152.92036,
"year": 0}
```

Log data:

- Log files in JSON format generated by event simulator based on the songs in the song dataset.
- These simulate activity logs from a music streaming app based on specified configurations.
- Log files are partitioned by year and month.

example, here are filepaths to two files in log dataset.

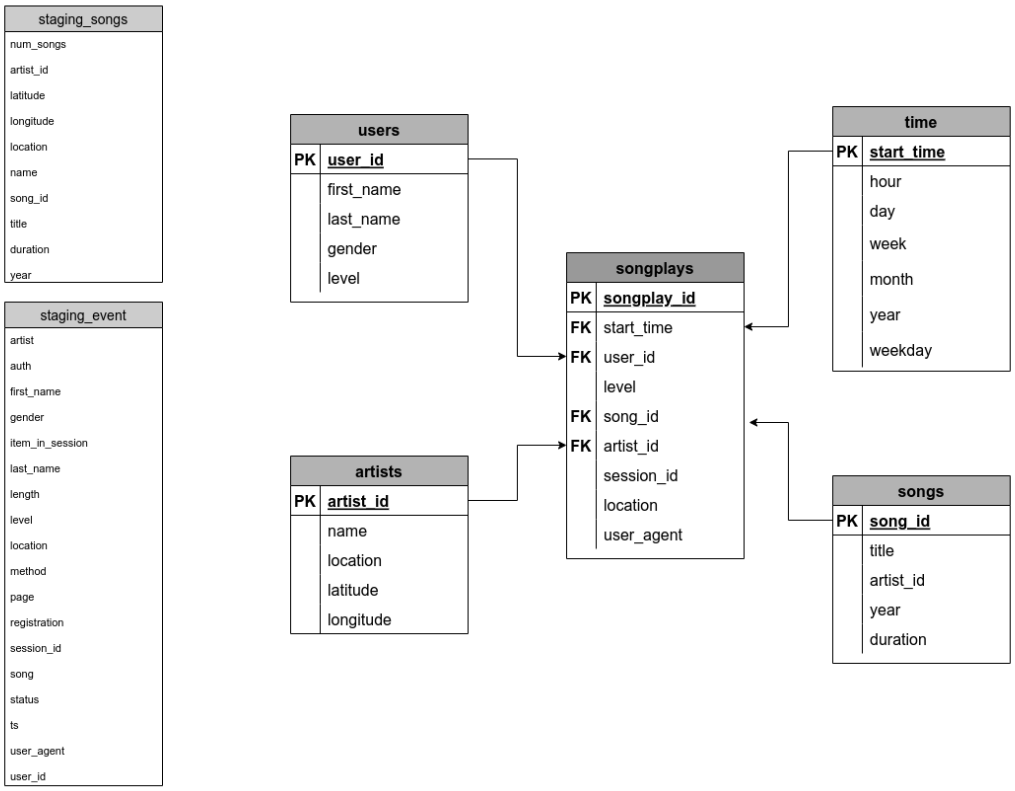
```
log_data/2018/11/2018-11-12-events.json
log_data/2018/11/2018-11-13-events.json
```

example of single log file, 2018-11-12-events.json:

```
{"artist":null,
"auth":"Logged In",
"firstName":"Celeste",
```

"gender":"F",
"itemInSession":0,
"lastName":"Williams",
"length":null,
"level":"free",
"location":"Klamath Falls, OR",
"method":"GET",
"page":"Home",
"registration":1541077528796.0,
"sessionId":438,
"song":null,
"status":200,
"ts":1541990217796,
"userAgent":"\"Mozilla\/5.0 (Windows NT 6.1; WOW64) AppleWebKit\/537.36 (KHTML, like Gecko) Chrome\/37.0.2062.103 Safari\/537.36\"",
"userId":"53"}

3. Data warehouse



Create_table.sql: Queries to create table

Sql_queries.py: Queries to insert data

4. Project Rubic:

- **General**

Criteria	Submission Requirements
The dag does not give an error when imported to Airflow	DAG can be browsed without issues in the Airflow UI
All tasks have correct dependencies	The dag follows the data flow provided in the instructions, all the tasks have a dependency and DAG begins with a start_execution task and ends with a end_execution task.

- **Dag configuration**

Criteria	Submission Requirements
Default_args object is used in the DAG	DAG contains default_args dict, with the following keys: • Owner • Depends_on_past • Start_date • Retries • Retry_delay • Catchup
Defaults_args are bind to the DAG	The DAG object has default args set
The DAG has a correct schedule	The DAG should be scheduled to run once an hour

- **Staging the data**

Criteria	Submission Requirements
Data Staging Implementation	Data is correctly loaded into the staging tables prior to transformation or loading steps.
Configuration Accuracy	The staging process operates correctly for the selected setup (S3/Redshift or Local/PostgreSQL).

- **Loading dimensions and facts**

Criteria	Submission Requirements
Dimension Table Loading	Data is accurately inserted into all relevant dimension tables according to the star schema design.
Fact Table Loading	Fact table receives the correct records with appropriate foreign keys and measures.
Data Consistency & Integrity	Relationships between facts and dimensions are consistent; referential integrity is maintained.

- **Data Quality Checks**

Criteria	Submission Requirements
Quality Check Implementation	Data quality checks are implemented and executed after ETL steps (e.g., no null checks, uniqueness, etc.).

5. Submit Project

- **Source Code:**

- Submit the complete source code of your project, including DAG files, ETL scripts, and any relevant configuration files.
- Organize the code clearly in directories, and include a README file or brief instructions on how to run your solution.

- **Screenshots:**

- Provide screenshots to illustrate your workflow, including:
  - The Airflow DAGs as shown in the web UI (DAG structure, task status).
  - Results of staging, loading dimension/fact tables, and data quality checks (e.g., logs, database table previews).
  - If there are any errors or warnings, include screenshots of the relevant log sections.
- Place all screenshots in a separate folder (e.g., /screenshots/) or embed them directly in your report/README.

- **Submission Format:**

- Compress (zip) your entire project, including the source code, screenshots, and README/report.
- Alternatively, submit via a repository link (GitHub, GitLab, etc.) with clear instructions and screenshots included.

Estimated Time to complete: 180 minutes

-- THE END --