Meets Specifications

Hello there,

Congratulations on finishing the project 🎉  
Keep doing the great work and all the best for future project.

**General**

**DAG can be browsed without issues in the Airflow UI**

Great start.  
The DAG loads without any error on the Airflow Server UI.

**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.**

This is correctly setup.

Each task have a dependency and there are no task without dependencies.  
Great work

**Dag configuration**

**DAG contains default\_args dict, with the following keys:**

* **Owner**
* **Depends\_on\_past**
* **Start\_date**
* **Retries**
* **Retry\_delay**
* **Catchup**

All the default argument have been correctly setup. This is important as we don't have to repeat for each DAG.  
You have also specified an end\_date, this is nice, as it limits the DAG to run only for a specific time frame.  
Also, you have correctly set up the Start\_date and end\_date.  
Learn more about them from here  
<https://airflow.apache.org/code.html>

**The DAG object has default args set**

Yes, for each dag, there is a binding to the default dag with dag=dag command  
Nice work

**The DAG should be scheduled to run once an hour**

This is correctly set up to hourly.

**Staging the data**

**There is a task that to stages data from S3 to Redshift. (Runs a Redshift copy statement)**

Great work, the code correctly loads the data from staging tables to redshift.

**Instead of running a static SQL statement to stage the data, the task uses params to generate the copy statement dynamically**

Nicely done  
I can see that you have used a dynamic copy statement that takes the parameters from the code and it is correctly copying the data from S3 buckets to staging tables on redshift

**The operator contains logging in different steps of the execution**

There is enough logging done in the code. This is helpful during error debugging. Great work

**The SQL statements are executed by using a Airflow hook**

Nice work, using Airflow hooks ensures that built in capability is used instead of re-writing the code.

You can also create the the tables from the DAG itself, This is not something many student will implemented. Great work

**Loading dimensions and facts**

**Dimensions are loaded with on the LoadDimension operator**

Great job.

1. There is separate functional operator for dimensions.
2. There is dynamic sql statements.
3. You have made use of optimal logging, this is important during error debugging.
4. There is switch for append only and truncate insert option for dimension operator.

**Facts are loaded with on the LoadFact operator**

Great job.

1. There is separate functional operator for dimensions.
2. There is dynamic sql statements.
3. You have made use of optimal logging, this is important during error debugging.

**Instead of running a static SQL statement to stage the data, the task uses params to generate the copy statement dynamically**

Nicely Done.

Parameters are passed on to the operators, which is then used to create dynamic sql statements  
Awesome work

**The DAG allows to switch between append-only and delete-load functionality**

Great,  
There is switch for append only and truncate insert option for dimension operator  
This is very commonly missed during this project

**Data Quality Checks**

**Data quality check is done with correct operator**

Perfect.

That's right we do data quality check on the dimension tables as they are our reference tables, we never data quality check on facts tables.

**The DAG either fails or retries n times**

**Operator uses params to get the tests and the results, tests are not hard coded to the operator**