Data Pipelines with Airflow

REVIEW | CODE REVIEW 6 | HISTORY

▸ airflow/plugins/operators/stage_redshift.py 3

▾ airflow/plugins/operators/load_dimension.py 1

```
1  from airflow.hooks.postgres_hook import PostgresHook
2  from airflow.models import BaseOperator
3  from airflow.utils.decorators import apply_defaults
4
5  class LoadDimensionOperator(BaseOperator):
6
7      ui_color = '#80BD9E'
8
9      insert_sql = """
10         TRUNCATE TABLE {};
11         INSERT INTO {}
12         {};
13         COMMIT;
14     """
15
16     @apply_defaults
17     def __init__(self,
18                  redshift_conn_id="",
19                  table="",
20                  load_sql_stmt="",
21                  *args, **kwargs):
22
23         super(LoadDimensionOperator, self).__init__(*args, **kwargs)
24         self.redshift_conn_id = redshift_conn_id
25         self.table = table
26         self.load_sql_stmt = load_sql_stmt
27
28     def execute(self, context):
29         redshift = PostgresHook(postgres_conn_id=self.redshift_conn_id)
30         self.log.info(f"Loading dimension table {self.table} in Redshift")
31         formatted_sql = LoadDimensionOperator.insert_sql.format(
32             self.table,
33             self.table,
34             self.load_sql_stmt
35         )
```

SUGGESTION

Here is another way to switch between append-only and delete-load functionality.
for example:

```
class LoadDimensionOperator(BaseOperator):

    ui_color = '#80BD9E'
    load_dimension_table_insert = """
        INSERT INTO {} {}
    """
    load_dimension_table_truncate = """
        TRUNCATE TABLE {}
    """

    @apply_defaults
    def __init__(self,
                 query="",
                 redshift_conn_id="",
                 t_name="",
                 operation="",
                 *args, **kwargs):

        super(LoadDimensionOperator, self).__init__(*args, **kwargs)
        self.query=query
        self.redshift_conn_id=redshift_conn_id
        self.t_name=t_name
        self.operation=operation

    def execute(self, context):
        self.log.info(f"Started LoadDimensionOperator {self.t_name} started with mode {self.operation}
")
        redshift_hook = PostgresHook(postgres_conn_id=self.redshift_conn_id)

        if(self.operation == "append"):
            redshift_hook.run(LoadDimensionOperator.load_dimension_table_insert.format(self.t_name, sel
f.query))
        if(self.operation == "truncate"):
            redshift_hook.run(LoadDimensionOperator.load_dimension_table_truncate.format(self.t_name))
            redshift_hook.run(LoadDimensionOperator.load_dimension_table_insert.format(self.t_name, sel
f.query))
        self.log.info(f"Ending LoadDimensionOperator {self.t_name} with a Success on Operation  {self.op
eration}")
```

```
36         redshift.run(formatted_sql)
```

- airflow/plugins/operators/data_quality.py 1
- airflow/dags/udac_example_dag.py 1
- airflow/plugins/operators/load_fact.py
- airflow/plugins/operators/__init__.py
- airflow/plugins/helpers/sql_queries.py
- airflow/plugins/helpers/__init__.py
- airflow/plugins/__init__.py
- airflow/create_tables.sql

RETURN TO PATH

Rate this review