Data Pipelines with Airflow

▾ airflow/plugins/operators/stage_redshift.py 3

```
1  from airflow.contrib.hooks.aws_hook import AwsHook
2  from airflow.hooks.postgres_hook import PostgresHook
3  from airflow.models import BaseOperator
4  from airflow.utils.decorators import apply_defaults
5
6  class StageToRedshiftOperator(BaseOperator):
7      ui_color = '#358140'
8
9      json_copy_sql = """
10         COPY {}
11         FROM '{}'
12         ACCESS_KEY_ID '{}'
13         SECRET_ACCESS_KEY '{}'
14         JSON '{}'
15         COMPUPDATE OFF
16     """
17
18     csv_copy_sql = """
19         COPY {}
20         FROM '{}'
21         ACCESS_KEY_ID '{}'
22         SECRET_ACCESS_KEY '{}'
23         IGNOREHEADER {}
24         DELIMITER '{}'
25     """
26
```

SUGGESTION

You should add a template field that allows it to load timestamped files from S3 based on the execution time and run backfills.

`template_fields = ("s3_key",)`

```
27     @apply_defaults
28     def __init__(self,
29                  redshift_conn_id="",
30                  aws_credentials_id="",
31                  table="",
32                  s3_bucket="",
33                  s3_key="",
34                  json_path="",
35                  file_type="",
36                  delimiter=",",
37                  ignore_headers=1,
38                  *args, **kwargs):
39
40         super(StageToRedshiftOperator, self).__init__(*args, **kwargs)
41         self.table = table
42         self.redshift_conn_id = redshift_conn_id
43         self.s3_bucket = s3_bucket
44         self.s3_key = s3_key
45         self.json_path = json_path
46         self.file_type = file_type
47         self.aws_credentials_id = aws_credentials_id
48         self.delimiter = delimiter
49         self.ignore_headers = ignore_headers
50
51     def execute(self, context):
52         aws_hook = AwsHook(self.aws_credentials_id)
```

AWESOME

Good job connecting to the database via an Airflow hook.

```
53         credentials = aws_hook.get_credentials()
54         redshift = PostgresHook(postgres_conn_id=self.redshift_conn_id)
55
56         self.log.info("Copying data from S3 to Redshift")
57         rendered_key = self.s3_key.format(**context)
58         s3_path = "s3://{}/{}".format(self.s3_bucket, rendered_key)
59
60         if self.file_type == "json":
```

AWESOME

Nice condition to test the file type 👏🏻

```
61             formatted_sql = StageToRedshiftOperator.json_copy_sql.format(
62                 self.table,
63                 s3_path,
64                 credentials.access_key,
```

```
                credentials.secret_key,
                self.json_path
            )
            redshift.run(formatted_sql)

        if self.file_type == "csv":
            formatted_sql = StageToRedshiftOperator.csv_copy_sql.format(
                self.table,
                s3_path,
                credentials.access_key,
                credentials.secret_key,
                self.ignore_headers,
                self.delimiter
            )
            redshift.run(formatted_sql)
```

▸ airflow/plugins/operators/load_dimension.py 1

▸ 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