**Assumptions**

1. **Snowflake Connection**: The script assumes that Airflow has access to Snowflake using appropriate credentials stored in a connection.
2. **Datetime Column**: The script assumes that the specified datetime column is indexed for efficient querying.
3. **Anomaly Detection**: The script will use a basic statistical approach for anomaly detection (e.g., z-scores). More advanced techniques can be substituted based on requirements.
4. **Notifications**: Slack notifications will be used for simplicity. Configuring Slack tokens and channels are assumed to be handled.
5. **Python Environment**: The Airflow environment has all necessary packages installed (pandas, snowflake-connector-python, scipy, slack\_sdk).

**1. DAG Initialization**

* **Task**: Initialize the Airflow DAG with necessary configurations and default arguments.
* **Details**: Sets up the owner, start date, retry policy, and schedule interval.

**2. Data Fetching from Snowflake**

* **Task**: Fetch data from the specified Snowflake table.
* **Details**: Connects to Snowflake using Airflow's SnowflakeHook and retrieves data from the last 30 days based on the datetime column. The fetched data is stored in XCom for downstream tasks.

**3. Anomaly Detection**

* **Task**: Detect anomalies in the fetched data.
* **Details**: Retrieves the fetched data from XCom, identifies numeric columns, and calculates z-scores for anomaly detection. Anomalies are identified if the z-score is greater than 3. Detected anomalies are stored in XCom and a Slack alert is sent if anomalies are found.

**4. Notification**

* **Task**: Send Slack notifications for detected anomalies.
* **Details**: Uses the Slack API to send messages to a specified Slack channel with details of detected anomalies.