USING dbt CLOUD TO GENERATE ANALYTICS AND ML-READY PIPELINES

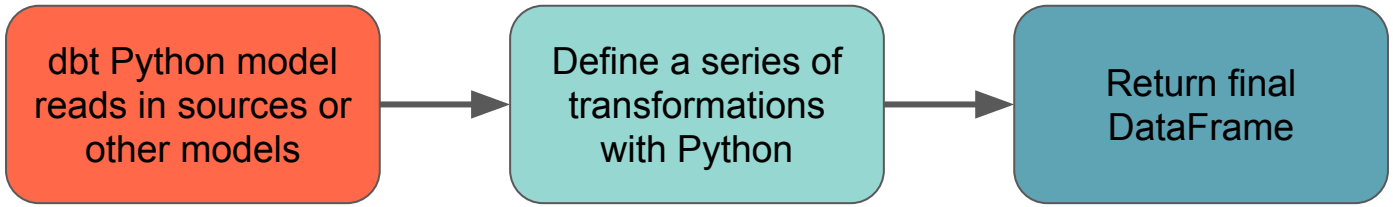
SNOWPARK MAKING THIS ALL POSSIBLE

PART OF THE PRESENTATION BY MIKKO SULONEN & dbt
@ACCELERATE DATA WITH DBT & SNOWFLAKE
MARCH 30TH, 2023

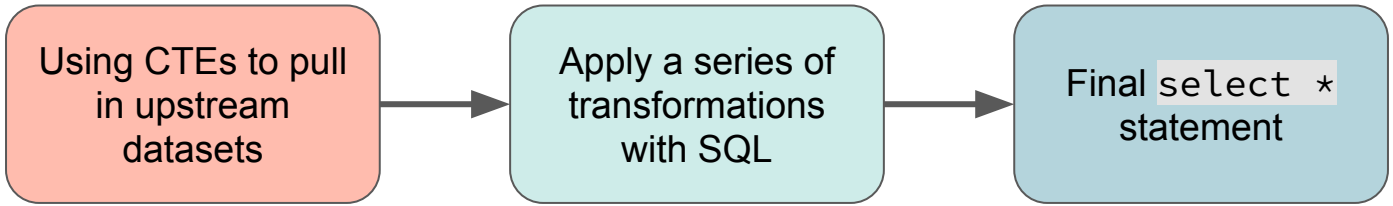
HOW IT WORKS: SNOWFLAKE

- GA release of Snowpark for python allows dbt users to seamlessly develop using python scripts that write to tables
- As always, dbt still does **not** move your data, the code is executed on Snowflake
- dbt python code is compiled and interpolated by Snowflake and run as a stored procedure
    - Snowflake → Snowpark and pandas dfs

HOW IT WORKS: WORKFLOW



👀 Looks similar to…



HOW IT WORKS: PYTHON MODEL FUNCTION

- Define a dbt function called model
  - `dbt`: A class compiled by dbt, which enables you to run your Python code in the context of your dbt project and DAG
  - `session`: A class representing your Snowflake's connection to the Python backend.
  - Model function must return a single DataFrame

<> models/my_python_model.py

```
import pandas as pd

def model(dbt, session):

    my_sql_model_df = dbt.ref('my_sql_model').to_pandas()

    final_df = my_sql_model_df['revenue'].groupby('month').agg('mean')

    return final_df
```

HOW IT WORKS: MODEL CONFIGURATIONS

- All the same great testing, documentation, and lineage capabilities as SQL
- Configs supported in:
  - `dbt_project.yml`
  - `.yml` files
  - `.py` using `dbt.config()`

<> models/my_python_model.py

```python
def model(dbt, session):

    # setting configuration
    dbt.config(materialized="table")
```

<> models/config.yml

```yaml
version: 2

models:
  - name: my_python_model

    # Document within the same codebase
    description: My transformation written in Python

    # Configure in ways that feel intuitive and familiar
    config:
      materialized: table
      tags: ['python']

    # Test the results of my Python transformation
    columns:
      - name: id
        # Standard validation for 'grain' of Python results
        tests:
          - unique
          - not_null
    tests:
      # Write your own validation logic (in SQL) for Python results
      - custom_generic_test
```

HOW IT WORKS: REFERENCES & MATERIALIZATIONS

- Python models only support `table` and `incremental` materializations
- Sources and references supported
  - Part of the dbt class
- Python models can be referenced downstream like any other model

models/my_python_model.py

```
def model(dbt, session):

    # DataFrame representing an upstream model
    upstream_model = dbt.ref("upstream_model_name")

    # DataFrame representing an upstream source
    upstream_source =
dbt.source("upstream_source_name", "table_name")

    ...
```

models/downstream_model.sql

```
with upstream_python_model as (

    select * from {{ ref('my_python_model') }}

),

...
```

HOW IT WORKS: PYTHON PACKAGE CONFIGURATIONS

- Able to use all packages that are supported on Snowflake
  - Query directly in snowflake to check which packages are supported
- Explicitly configure packages and versions as best practice
  - Track in dbt's metadata

<> models/my_python_model.py

```python
def model(dbt, session):
    dbt.config(
        packages = ["numpy==1.23.1", "scikit-learn"]
    )
```

<> models/config.yml

```yaml
version: 2

models:
  - name: my_python_model
    config:
      packages:
        - "numpy==1.23.1"
        - scikit-learn
```

DEVELOP, TEST, AND DEPLOY DATA PRODUCTS IN YOUR WAREHOUSE