Airflow Summit

Advanced Apache Superset for Data Engineers
- A passion for building data tools!
- Started Apache **Airflow** at Airbnb in 2014
- Started Apache **Superset** at Airbnb in 2015
- Started **Preset** - The Apache Superset company in 2019
Agenda!

- Superset Overview / Demo
- SQL Lab for data engineers
- Scheduling Queries
- Building a visualization plugin
- Building charts and dashboards dynamically
Superset Overview / Demo!
Enhancing Jinja Context

```python
# in superset_config.py
JINJA_CONTEXT_ADDONS = {
    "say_hello": lambda: 'hello',
}
```
Scheduling Queries
experimental feature!
feat: Scheduling queries from SQL Lab #7416

betodealmeida merged 8 commits into apache:lyft-release-sp8 from lyft:VIZ-3a on May 3, 2019

betodealmeida commented on Apr 30, 2019 • edited by mistercrunch

SUMMARY

This PR introduces a lightweight way of scheduling queries in SQL Lab. If the feature flag SCHEDULED_QUERIES is enabled with proper configuration, a button called "Schedule Query" will show up in SQL Lab. The button allows queries to be saved with extra metadata that allows an external scheduler to run it periodically by polling the /savedqueryviewapi/api/read endpoint.

The sample configuration can be changed or expanded to support different metadata needed, depending on the scheduler. We tested it with Apache Airflow at Lyft successfully.
FEATURE_FLAGS = {
    # Configuration for scheduling queries from SQL Lab. This information is
    # collected when the user clicks "Schedule query", and saved into the 'extra'
    # field of saved queries.
    # See: https://github.com/mozilla-services/react-jsonschema-form
    'SCHEDULED QUERIES': {
        'JSONSCHEMA': {
            'title': 'Schedule',
            'description': 'In order to schedule a query, you need to specify when it
            should start running, when it should stop running, and how
            often it should run. You can also optionally specify
            dependencies that should be met before the query is
            executed. Please read the documentation for best practices
            and more information on how to specify dependencies.
        },
        'type': 'object',
        'properties': {
            'output_table': {
                'type': 'string',
                'title': 'Output table name',
            },
            'start_date': {
                'type': 'string',
                'title': 'Start date',
                # date-time is parsed using the chrono library, see
                # https://www.npmjs.com/package/chrono-node#usage
                'format': 'date-time',
                'default': 'tomorrow at 9am',
            },
            'end_date': {
                'type': 'string',
                'title': 'End date',
                # date-time is parsed using the chrono library, see
                # https://www.npmjs.com/package/chrono-node#usage
                'format': 'date-time',
                'default': 'tomorrow at 9am',
            },
            'schedule_interval': {
                'ui:placeholder': '@daily, @weekly, etc.',
                'ui:help': 'Check the documentation for the correct format when
                defining dependencies.'
            },
        },
    },
    'UI_SCHEMA': {
        'schedule_interval': {
            'ui:placeholder': '@daily, @weekly, etc.',
        },
        'dependencies': {
            'ui:help': 'Check the documentation for the correct format when
            defining dependencies.'
        },
    },
    'VALIDATION': [
        # ensure that start_date <= end_date
        {
            'name': 'less_equal',
            'arguments': ['start_date', 'end_date'],
            'message': 'End date cannot be before start date',
            'container': 'end_date',
        },
    ],
    # link to the scheduler; this example links to an Airflow pipeline
    # that uses the query id and the output table as its name
    'LINKBACK': {
        'https://airflow.example.com/admin/airflow/tree/
        'dag_id=query_${id}_${extra.json.schedule_info.output_table}'
    },
}
Schedule Query

Label
AIRFLOW SUMMIT !!!

Description
Write a description for your query

Schedule
In order to schedule a query, you need to specify when it should start running, when it should stop running, and how often it should run. You can also optionally specify dependencies that should be met before the query is executed. Please read the documentation for best practices and more information on how to specify dependencies.

Output table name

Start date
mm/dd/yyyy, --:-- --

End date
mm/dd/yyyy, --:-- --

Schedule interval
@daily, @weekly, etc.

Dependencies

Check the documentation for the correct format when defining dependencies.
null,

"extra": {

  "schedule_info": {
    "dependencies": [
      "hive://SOURCE_TABLE/{{ds}}",
    ],
    "output_table": "THIS_IS_THE_OUTPUT_TABLE",
    "schedule_interval": "@daily",
    "start_date": "2020-07-08T19:08:00.000Z"
  }
},

"extra_json": "{"schedule_info":{"output_table":"THIS_IS_THE_OUTPUT_TABLE",
"hive://SOURCE_TABLE/{{ds}}"}}",

"id": 2,

"label": "AIRFLOW SUMMIT !!!",

"schema": "superset",

"sql": "SELECT 'HELLO AIRFLOW SUMMIT' as label",

"sqlalchemy_uri": "mysql://root@localhost/examples?charset=utf8",

"user_email": "admin@fab.org"
Visualization Plugins
Hello World

How to Build a Viz Plugin for Apache Superset

Superset Developers

So, You Want to Build a Superset Viz Plugin...

Evan Rusackas  July 02, 2020

https://preset.io/blog/
Superset Plugins as a data product development platform

- Build data products without writing much backend code
- Tap into Superset’s Data Access Layer (auth, perm, cache, audit)
- Rich controls at your fingertips
- Focus on the visualization / frontend
- Bring into a dashboard (surround with context / add interactions)
Dynamic Chart/Dashboard Creation
Rest API!

Swagger API @ /swaggerview/v1
Using SQLAlchemy (improper)

```
defaults = {
    "compare_lag": "10",
    "compare_suffix": "olBY",
    "limit": "25",
    "granularity_sqa": "ds",
    "groupby": [],
    "row_limit": config["ROW_LIMIT"],
    "since": "100 years ago",
    "until": "now",
    "viz_type": "table",
    "markup_type": "markdown",
}

admin = security_manager.find_user("admin")

print("Creating some slices")
slices = [
    Slice(
        slice_name="Participants",
        viz_type="big_number",
        datasource_type="table",
        datasource_id=tbl.id,
        params=SliceJson(
            defaults,
            viz_type="big_number",
            granularity_sqa="ds",
            compare_lag="5",
            compare_suffix="over 5Y",
            metric=metric,
        ),
    ),
    Slice(
        slice_name="Genders",
        viz_type="pie",
        datasource_type="table",
```
We’re hiring!

Careers at Preset

Preset is actively hiring a team to build features and services in and around Apache Superset, the leading open source analytics and data visualization platform.