

---

# **Intake-Postgres Documentation**

***Release 0.3.0***

**Anaconda, Inc.**

**Nov 22, 2018**



---

## Contents:

---

|          |                           |          |
|----------|---------------------------|----------|
| <b>1</b> | <b>Quickstart</b>         | <b>3</b> |
| <b>2</b> | <b>API Reference</b>      | <b>5</b> |
| <b>3</b> | <b>Indices and tables</b> | <b>7</b> |



Read data from PostgreSQL into Pandas dataframes using Intake.



# CHAPTER 1

---

## Quickstart

---

This guide illustrates how to get started using *Intake-Postgres*, an *Intake* plugin that adds support for ingesting data from the [PostgreSQL](#) RDBMS. Before continuing, please complete the *Intake* quickstart.

## 1.1 Installation

If you have a [conda-based installation](#), install *Intake* and the *Intake-Postgres* plugin with the following command:

```
conda install -c intake intake-postgres
```

## 1.2 Usage (via *catalog.yml*)

Usage of *Intake-Postgres* is easiest to illustrate with an example.

In the *catalog.yml* file:

```
sources:
  all_users:
    driver: postgres
    args:
      uri: 'postgresql://postgres@localhost:5432/postgres'
      sql_expr: 'select * from users'
```

There are two things to note in the above example:

1. *intake\_postgres* is included under “plugins”. This only needs to be done once for each *catalog.yml* file.
2. Any “sources” entry which includes the field *driver: postgres* includes some additional fields that are specific to the *Intake-Postgres* plugin. Specifically, we need to provide a *uri* to the database, and a *sql\_expr* (SQL query expression).

Intake can then be accessed as normal, and provided that *Intake-Postgres* is installed:

```
>>> import intake
>>> catalog = intake.Catalog('catalog.yml')
>>> ds = catalog.all_users
>>> ds.discover()
>>> df = ds.read()
>>> df.tail()
```

The code above reads the *catalog.yml* file as normal, calls *discover()* on the *Intake-Postgres* data source, and then reads it into a dataframe for further analysis.

## 1.3 Usage (via Python library)

*Intake-Postgres* can also be accessed directly as a library. This usage pattern is for users who desire to call *Intake-Postgres* from inside another application, or just want more control over how data is ingested.

Here is the same example as above, except accessing *Intake-Postgres* as a library instead of through the *catalog.yml*:

```
>>> import intake_postgres
>>> ds = intake_postgres.PostgresSource('postgresql://postgres@localhost:5432/postgres
↳ ', 'select * from users')
>>> source.discover()
>>> df = source.read()
>>> df.tail()
```



## CHAPTER 2

---

### API Reference

---

**class** `intake_postgres.intake_postgres.PostgresSource` (*uri*, *sql\_expr*, *pg\_kwargs*={},  
*metadata*=None)  
Read data from PostgreSQL to dataframes  
**uri: str** Connection to PostgreSQL server  
**sql\_expr: str** The full text of the SQL query to execute  
**pg\_kwargs: dict** Further args passed to `postgresadapter.PostgresAdapter`, see <https://github.com/ContinuumIO/PostgresAdapter/blob/master/postgresadapter/core/PostgresAdapter.pyx#L281>



## CHAPTER 3

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



## P

PostgresSource (class in *in-*  
*take\_postgres.intake\_postgres*), [5](#)