Contracts Database

Under the Data Act of 2014, the Department of the Treasury and Office of Management and Budget release all federal contract data as a PostgreSQL data dump. The data is available at USASpending. The full database is quite large (~1.5 Terabytes). This is a guide to show you how to connect to and use the database the we have set up with Spark.

Prerequisites

There are some required tools or accounts you need to connect and query the database

1. Access to the `ds-dod-contract-access` Github Org. If you do not already have access to this Org (this link doesn't work), then reach out to Ian Saucy, who will add you.
2. `cloudflared` - This is a Cloud Flare command line tool to gain access to the computer the database is hosted on.
3. [*Optional*] PGAdmin. If you are comfortable with command SQL and proficient with `pqsl`, then you can skip this. If this is the case, you likely won't need any of this guide.

Connect to the database

The first step to connect to the database, once you have all the tools downloaded, is to make a cloud flare tunnel with the following command

```
cloudflared access tcp --hostname ds-dod-contracts.buspark.io --url 127.0.0.1:5432
```

This first time you run this, you are likely to get a message with a link prompting you to sign up. Click the link and sign in with your GitHub account. You only need to do this once per computer you are using.

Once you are connected, you can connect to the Postgres instance however you normally do. If you are not familiar with Postgres, we will use PGAdmin. However you connect, you will use the following parameters:

Syntax	Description	Notes
Host	127.0.0.1	You have tunneled the host machine to localhost
Port	5432	Standard PostgreSQL port
Database Name	contracts	
username	fa23_team	*Replace with your term*
password	*N/A*	*Get your teams password from Ian Saucy*

From here on out, we will assume you are using PGAdmin. Once you have PGAdmin open, register a new server.

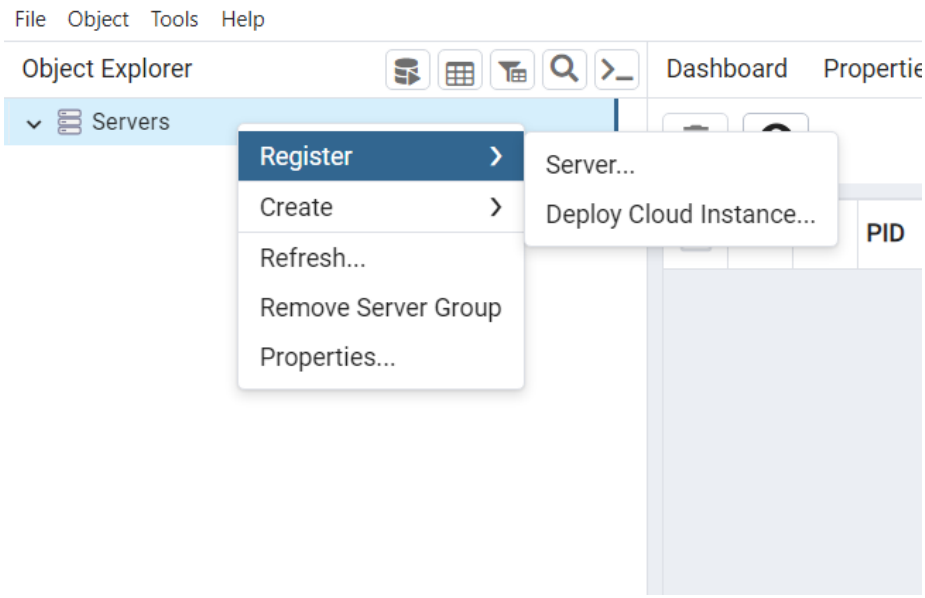

Figure 1: Image 1

Then you can fill in the General and Connection tabs with the following data

Once all the information is filled in, press Save. Now, whenever you click on the `Contracts` server, you will connnect to this server. You must have the cloudflare command running whenever you use the database, or the connection will fail. Once connected you should be table to see the following under "Servers->Contracts":

Database Structure

We have only restored two tables from the database, `rpt.award_search` and `rpt.recipient_lookup`. This contains all the general data about awards and recipients. There are around 50 other tables and three other schemas that are present in the database, but unpolulated. If there is data in one of these tables that you want to populate, see the section "Recreating the Database". Otherwise, these tables should contain most the information regarding Kaija's interest. We have also added a Materialized View `igf_mv` holds which awards have the string `IGF` in their description. This will be all the awards that were once "Inherently Govermental Functions", or activities that were once run by the govermnent, but have since been outsourced.

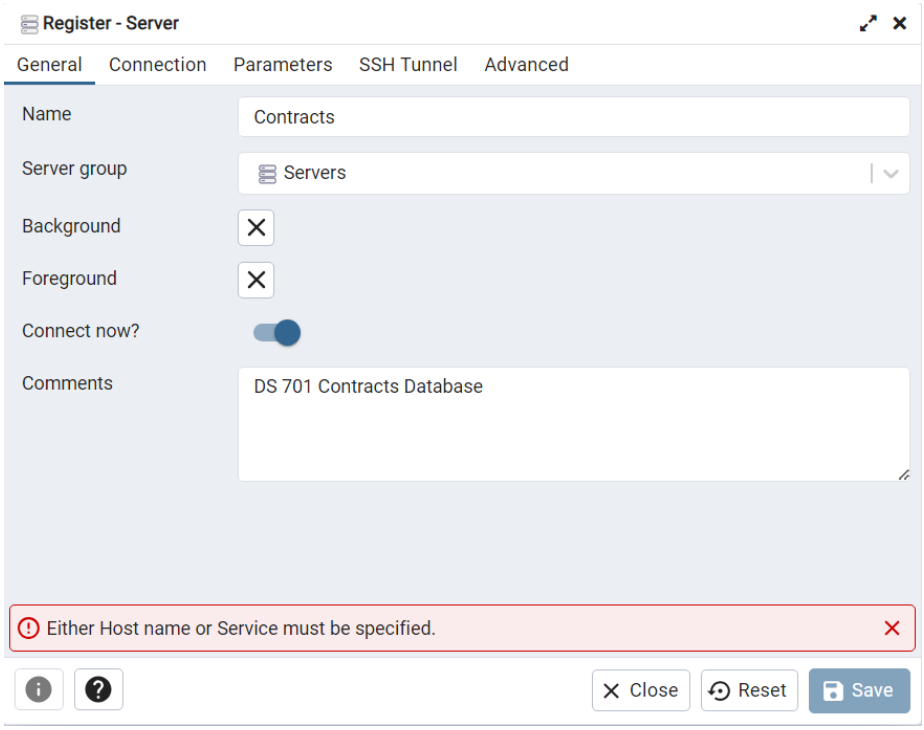

Figure 2: Image 2

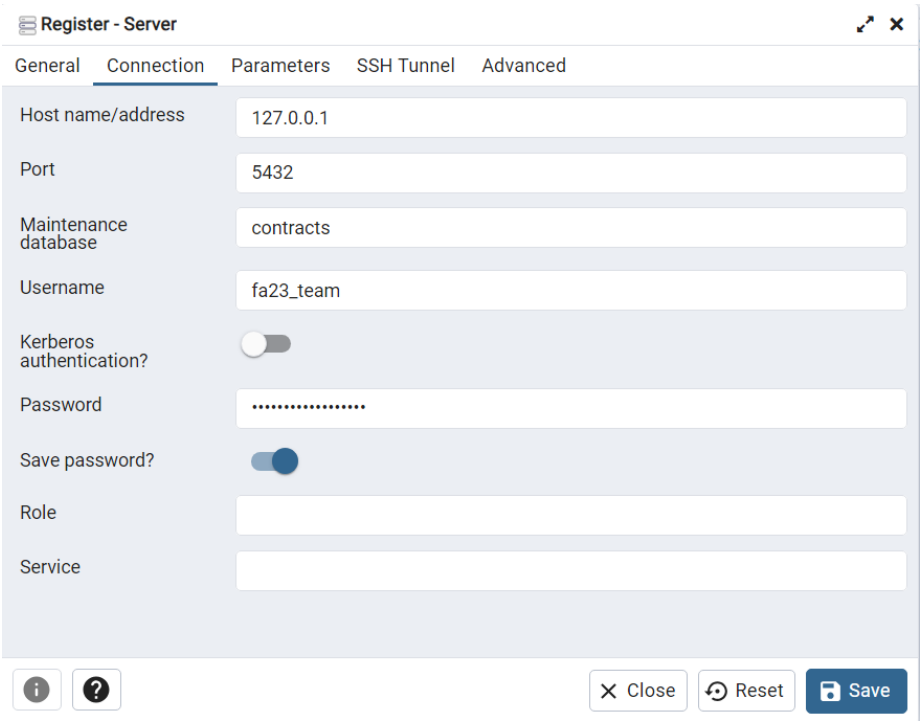

Figure 3: Image 3

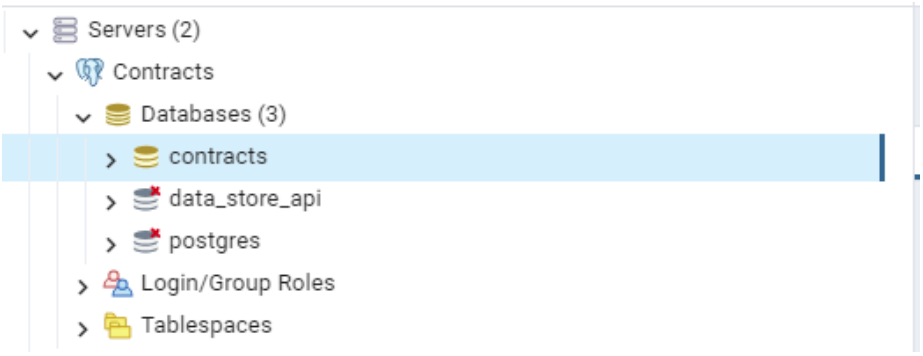

Figure 4: Image 4

Querying the database

You can query this database with any standard SQL that you wish. I have included a query that you could build off of, specifically one that uses our indexes and materialized views.

- This is a query that returns the top ten awards, by dollar value, for each NAICS code that are `IGF` but also have the `CI` subtype.

```
WITH RankedAwards AS (
    SELECT
        award_amount,
        naics_code,
        ROW_NUMBER() OVER (PARTITION BY naics_code ORDER BY award_amount DESC) as rank
    FROM
        igf_mv
    WHERE
        description LIKE '%IGF::CI%'
)
SELECT
    award_amount,
    naics_code
FROM
    RankedAwards
WHERE
    rank <= 10;
```

Helpful tables and columns

Note :: The USASpending website has a great glossary if you need descriptions of any of the terms mentioned in the database found here
Here is a list of some helpful columns in each database and some breif descriptions | Table | Column Name | Notes | | ———— | ———— | ———— | | rpt.award_search | award_amount | The amount that the federal government has promised to pay (obligated) a recipient, because it has signed a contract, awarded a grant, etc.| | rpt.award_search | total_outlay | An outlay occurs when federal money is actually paid out, not just promised to be paid (“obligated”). | | rpt.award_search | period_of_performance_start_date | The date that the award begins, as agreed upon by the parties involved. Note that the first transaction for the award (known as the Base Transaction Action Date) may be different than this date. | | rpt.award_search | period_of_performance_current_end_date |The date that the award ends, as agreed upon by the parties involved without exercising any pre-determined extension options. Note that the latest transaction for the award (known as the Latest Transaction Action Date) may be different than this date. | | rpt.award_search | naics_code | NAICS stands for the North American Industrial Classification System. This 6-digit code tells you what industry the work falls into. Each contract record has a NAICS code. That

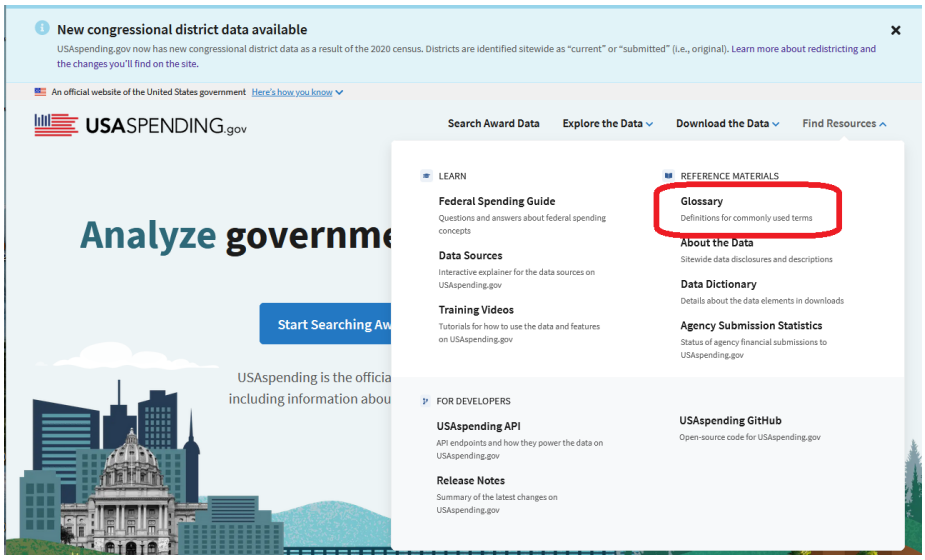

Figure 5: Image 5

means you can look up how much money the U.S. government spent in a specific industry. | | rpt.recipient_lookup | state | State where the recipient is based | | rpt.recipient_lookup | alternname_name | Other names that this company has gone by | | rpt.recipient_lookup | congressional_district | Which congressional district this company is based in |

Recreating the Database

The database is already created, but if you would want to recreate it with different tables or fields, you can do the following:

1. Hace docker installed and ready to use
2. Download, unzip and reorder one of the databases. The databses can be found here: Database Download. WARNING: These files are very large.

```
wget https://files.usaspending.gov/database_download/usaspending-db-subset_20231108.zip
mkdir d
unzip usaspending-db-subset_*.zip -d d
```

At this points you should have a directory that looks like this

3. To create the Docker container, you should have two files, called `Dockerfile` and `resotre.sh`. The contents of these files is listed below

`Dockerfile`:

```
FROM postgres:16-bullseye
WORKDIR /d
```

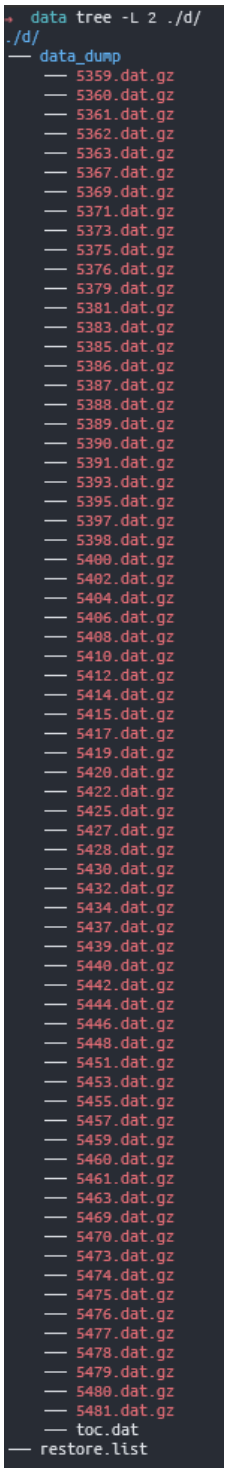

Figure 6: Image 6

```
WORKDIR /
COPY ./restore.sh .
```

restore.sh:

```
psql -U postgres -c "DROP DATABASE IF EXISTS contracts"
psql -U postgres -c "CREATE DATABASE contracts"

pg_restore --list /d/data_dump | sed '/MATERIALIZED VIEW DATA/d' > /d/restore.list
pg_restore -U postgres \
           --no-owner \
           --jobs 16 \
           --dbname contracts \
           --verbose \
           --exit-on-error \
           --schema-only \
           --use-list /d/restore.list \
           /d/data_dump

pg_restore -U postgres \
           --no-owner \
           --jobs 16 \
           --dbname contracts \
           --verbose \
           --exit-on-error \
           --data-only \
           --schema=rpt \
           --table=award_search \
           --table=recipient_lookup \
           --use-list /d/restore.list \
           /d/data_dump\
```

With these files you can build you docker container with

```
docker build -t p .
docker run --name contracts -e POSTGRES_PASSWORD=p -p 5432:5432 -v ./d:/d -d  p
docker exec -it $(docker ps -alq) bash
```

These commands will pull the standard Postgres 16 docker image, and then create the docker container with our data folder `d` mounted into the container. We are using this docker container for its `pg_restore` file. Finally the `exec` comman will bring you into the container to a terminal.

4. From this container the next step is to restore the database. To do this, simply give the `restore.sh` script persmissions to run, and then run it.

```
chmod 777 restore.sh
./restore.sh
```

This will then run and restore your database. If this command fails, consult Ian Saucey.

Modifying the resotre script

To change how the database is restored, you can modify the restore scipt. There is ony line that you will modify specifically.

```
pg_restore -U postgres \
           --no-owner \
           --jobs 16 \
           --dbname contracts \
           --verbose \
           --exit-on-error \
           --data-only \
           --schema=rpt \
           --table=award_search \
           --table=recipient_lookup \
           --use-list /d/restore.list \
           /d/data_dump\
```

In this command, you can modify the `--table` and `--schema` flags. Tables are represented by `<schema>.<table>`, so if you want to add new tables from a schema other than the `rpt` schema, then you have to add both `--schema <your_schema>` and `--table <your_table>`. You can have many table and schema flags in each command.

For more information on this, you can visit the pg_restore documentation