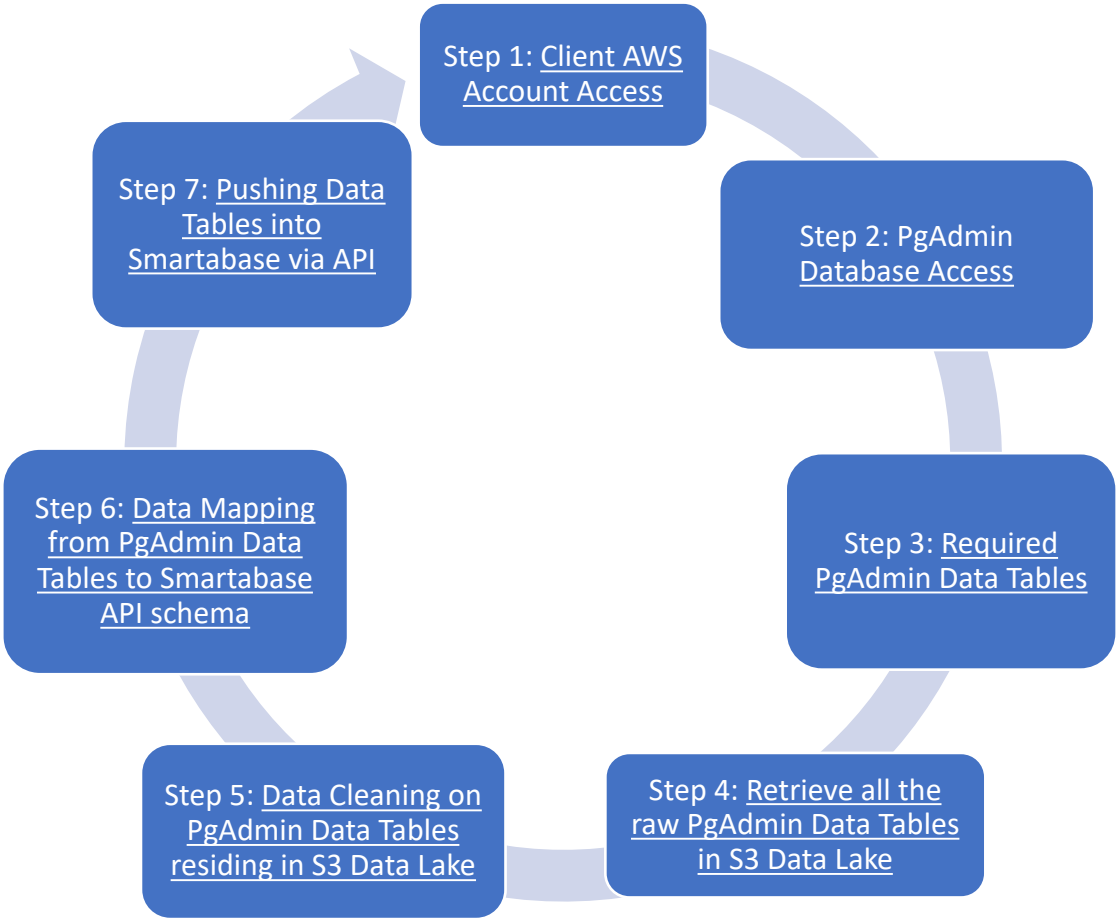
Data Migration Process
Content

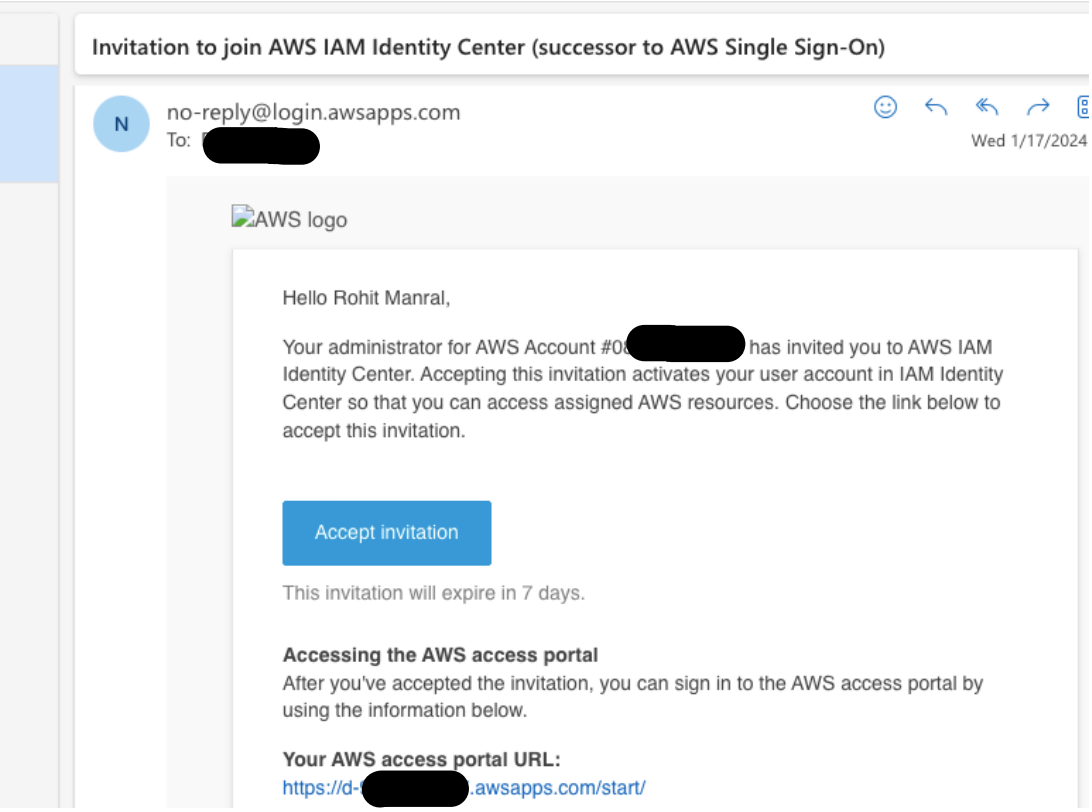
1. Process Flow Diagram
2. Client's AWS Account Access
3. PgAdmin (PostgreSQL) Database Access
   - Create a new **EC2 windows instance (t3.large)**
   - connect to the **EC2 windows instance (t3.large)** by downloading **Remote Desktop File**
     - Install **PgAdmin (PostgreSQL 16)**
     - Install **DBeaver**
   - Connect to client's **PgAdmin (Production)** data server in **DBeaver** through **SSH Tunnel**
4. Extract required PgAdmin Data Tables
   - Proving a list of **Table name, Column name, and Data type** to **int/ext stakeholders** using **DBeaver SQL scripts**
   - Meeting with **int/ext stakeholders** to discuss the required **data tables** from **PgAdmin**
5. Ingest raw PgAdmin Data Tables into S3 Data Lake
   - Using SQL scripts in DBeaver, extract all the required data tables (.csv)
   - Create/ Locate an S3 Bucket in Client's S3 Data Lake
     - Create a **production** folder & store all the raw production data tables (csv) into it
6. Data Cleaning using Jupyter Notebook
   - Install Jupyter Notebook/ Python IDE in the EC2 windows instance
   - Run python scripts to read all raw production data files from S3 Bucket
   - Run python scripts to transform all raw production data tables (csv) in S3 Bucket
     - Confirm all data is present (NULL values issue) after splitting **properties** column data
     - Removing HTML Tags
     - Need to replace Dropdown Categories from PgAdmin tables
     - Numerical conversions (mm -> cm or g -> kg) of Height & Weight columns
     - Calculation Age from Date of Birth
     - Renaming columns
   - Finally, store all transformed production data files into S3 Bucket in a new folder
7. Data Mapping from PgAdmin Data Tables to SmartaBase Schema
8. Pushing finalised Data Tables into SmartaBase API via Postman or Python API connection
   - Convert the CSV file to JSON
   - Finalise data tables to Push into the SmartaBase
   - POST using POSTMAN

Data Migration Process Flow

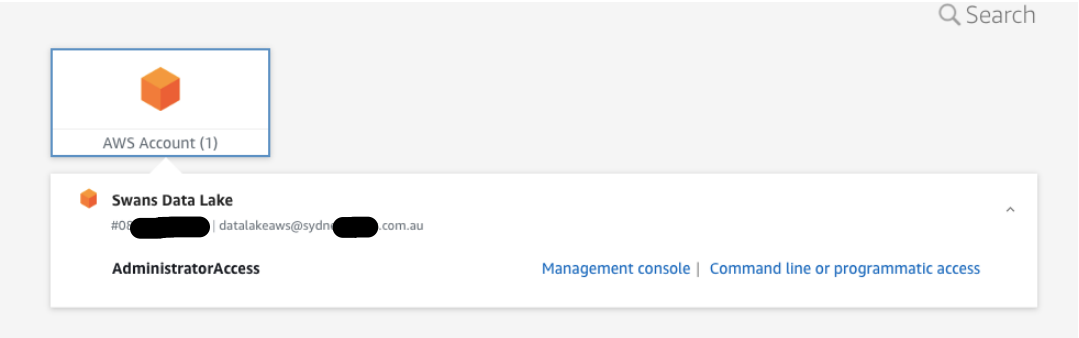


Step 1: Client AWS Account Access

- **Client AWS account** access
    - Go to **Outlook**, search for **Identity Center**, and open the matching email from Client. Then, select **AWS access portal URL**

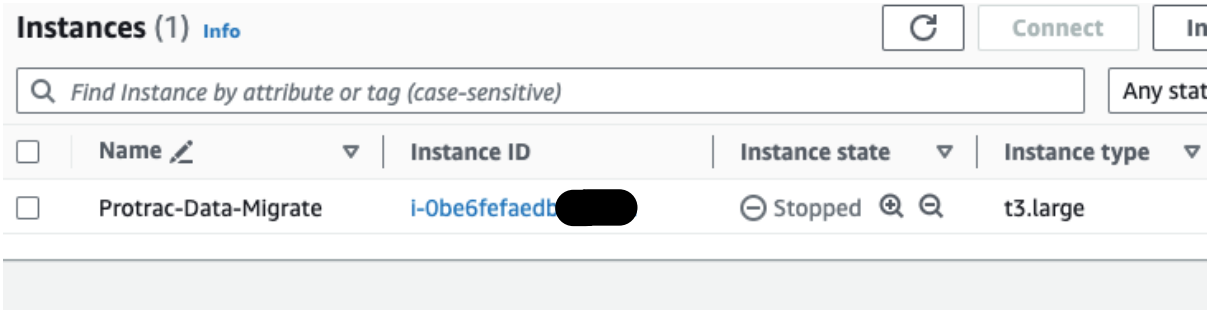


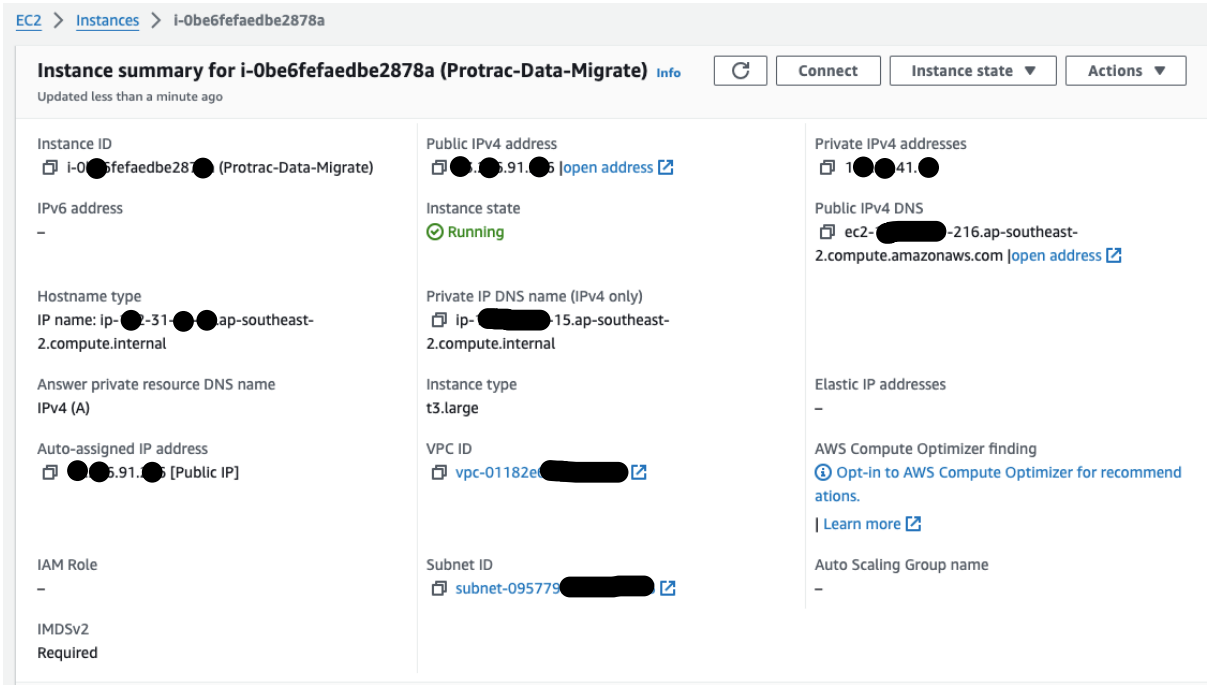
    - Go to **Management Console**



Step 2: PgAdmin Database Access

- Create a new **EC2 windows instance (t3.large)**





Details | Status and alarms New | Monitoring | Security | Networking | Storage | Tags

▼ Security details

IAM Role
–

Owner ID
0[redacted]

Launch time
Thu Feb 01 2024 01:47:21 GMT+1100 ([redacted] Eastern Daylight Time)

Security groups
sg-04f[redacted]d4db (launch-wizard-2)

▼ Inbound rules

Filter rules

Name	Security group rule ID	Port range	Protocol	Source
–	sgr-07c310e36c[redacted]	3389	TCP	0.0.0.0/0

▼ Outbound rules

Filter rules

Name	Security group rule ID	Port range	Protocol	Destination
–	sgr-0a5cfcdf1[redacted]	All	All	0.0.0.0/0

- Now, connect to the **EC2 windows instance (t3.large)** by downloading **Remote Desktop File**

Connect to instance Info

Connect to your instance i-0be6fefaedbe2878a (Protrac-Data-Migrate) using any of these options

Session Manager | RDP client | EC2 serial console

Instance ID
i-0be6fefaedbe2878a (Protrac-Data-Migrate)

Connection Type

Connect using RDP client
Download a file to use with your RDP client and retrieve your password.

Connect using Fleet Manager
To connect to the instance using Fleet Manager Remote Desktop, the SSM Agent must be installed and running on the instance. For more information, see Working with SSM Agent

You can connect to your Windows instance using a remote desktop client of your choice, and by downloading and running the RDP shortcut file below:

Download remote desktop file

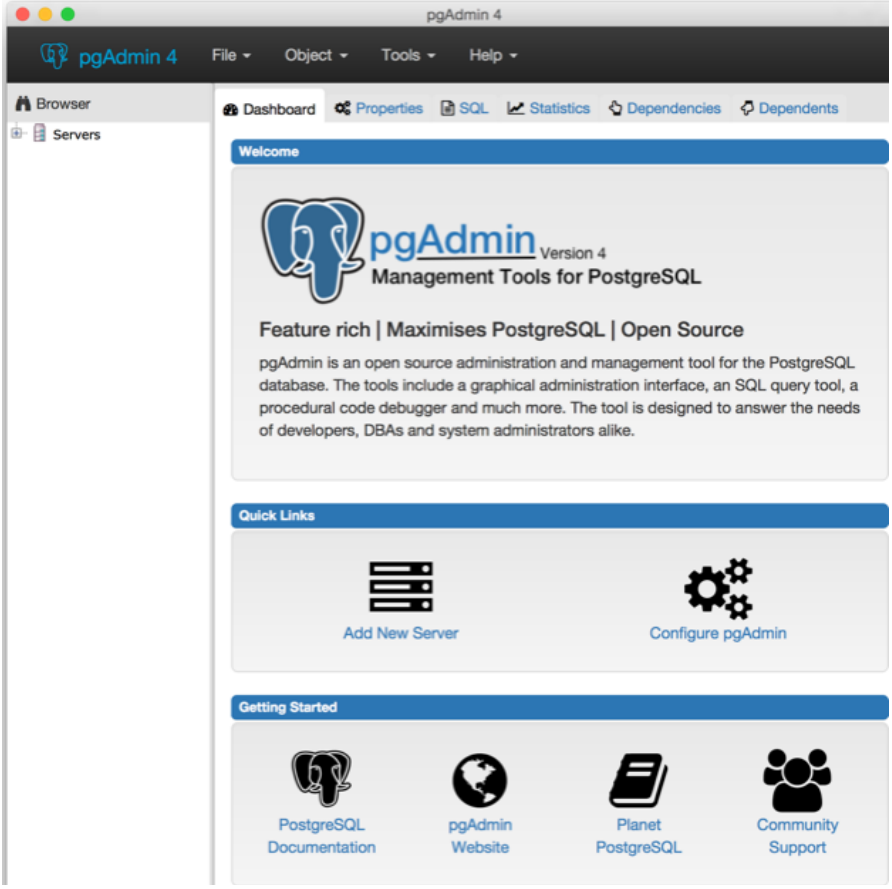
When prompted, connect to your instance using the following details:

Public DNS
ec2-[redacted]-91[redacted].ap-southeast-2.compute.amazonaws.com

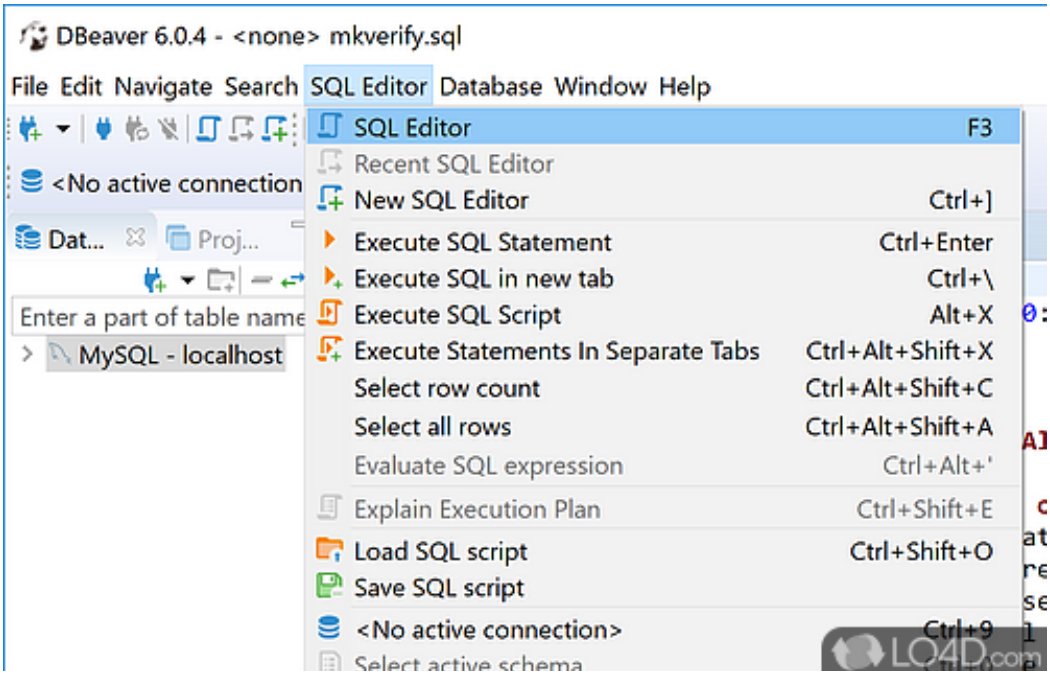
Username
Administrator

Password Get password

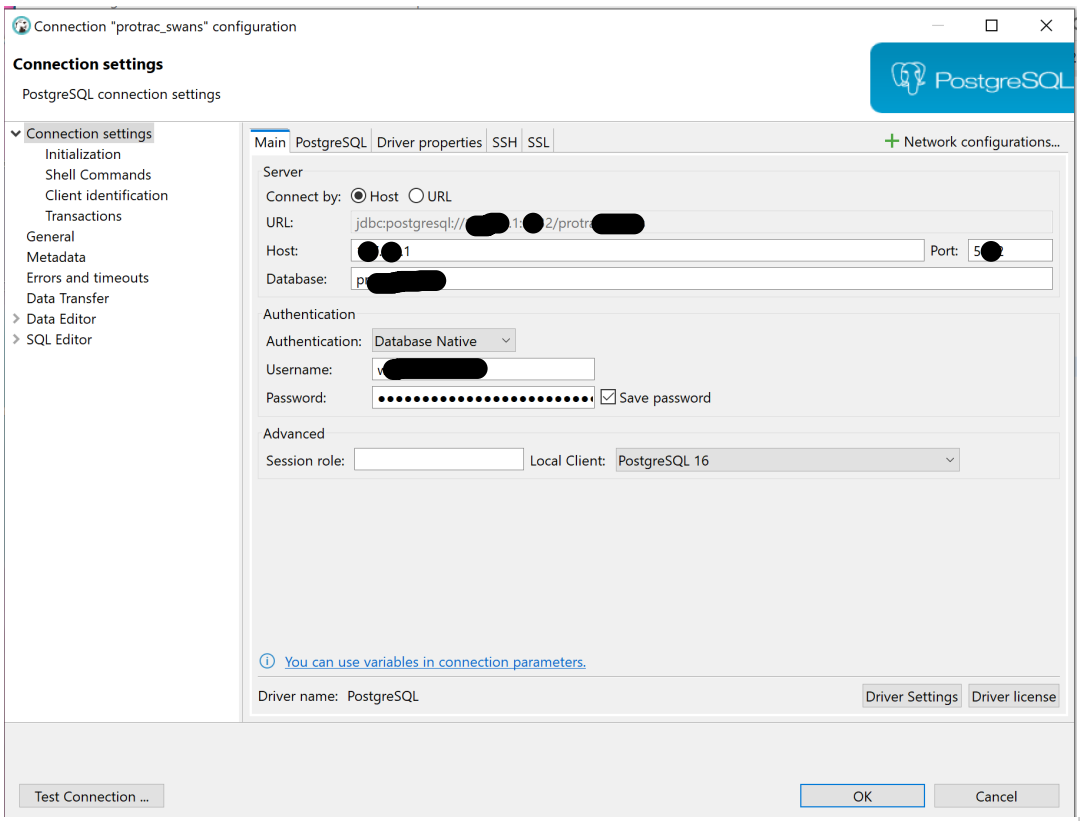
- Install **PgAdmin (PostgreSQL 16)**

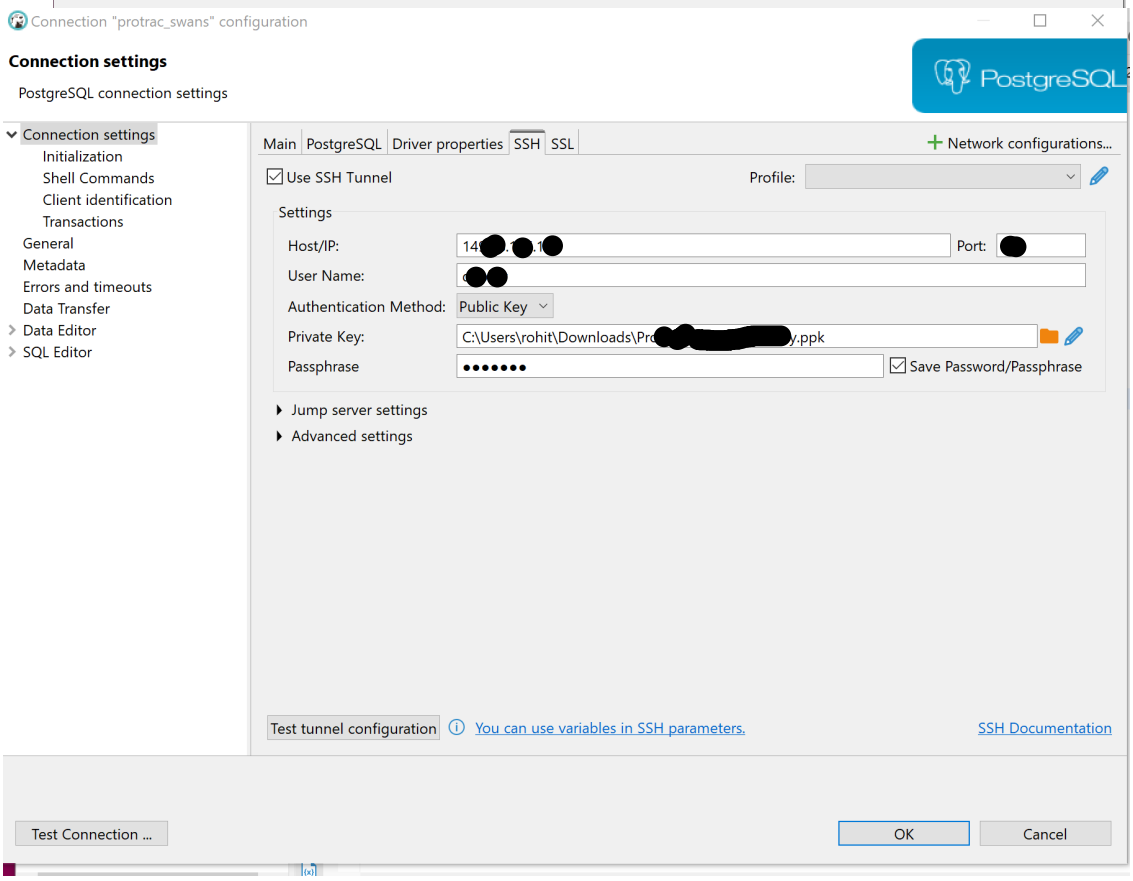


- Install **DBeaver**



- Connect to **Client_prod (Production)** data server in **DBeaver** through **SSH Tunnel**





Step 3: Required PgAdmin Data Tables

- Proving a list of **Table name, Column name, and Data type** to **Thomas** using **DBeaver SQL scripts**

SQL Script:

```
SELECT
table_name,
column_name,
data_type
FROM information_schema.columns
WHERE table_schema = 'public'
ORDER BY table_name;
```

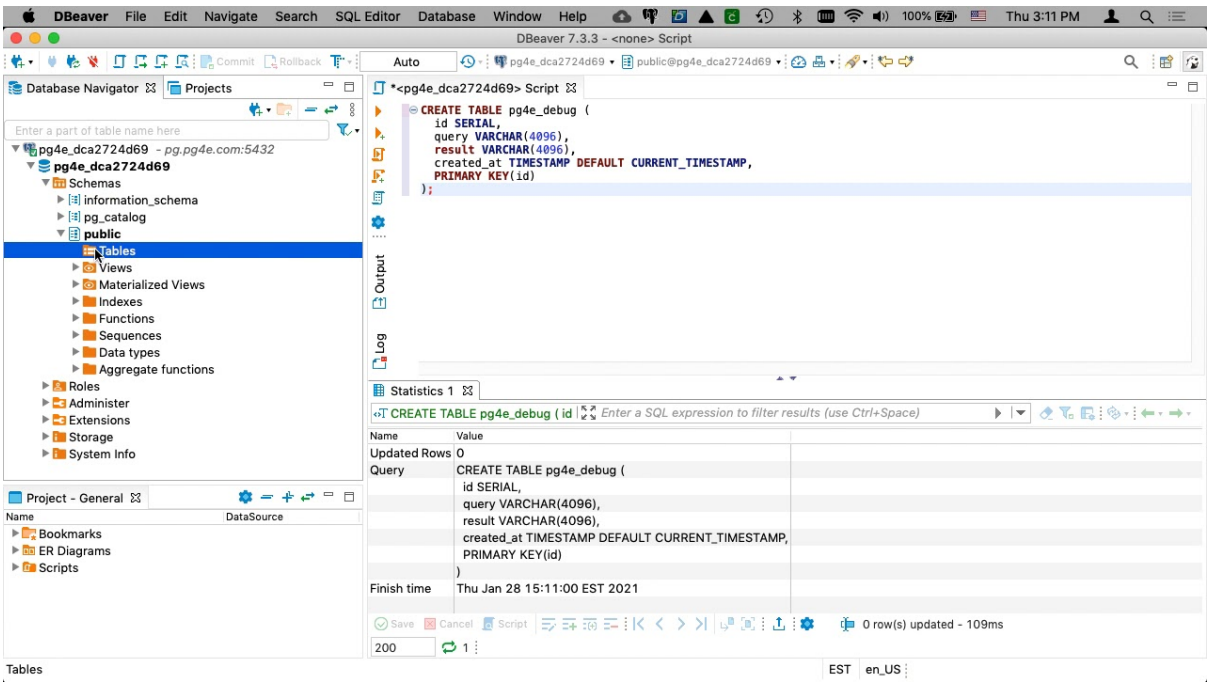
table_name	column_name	data_type
active_storage_attachments	created_at	timestamp without time zone
active_storage_attachments	name	character varying
active_storage_attachments	record_type	character varying
active_storage_attachments	id	bigint
active_storage_attachments	record_id	bigint
active_storage_attachments	blob_id	bigint
active_storage_blobs	metadata	text
active_storage_blobs	service_name	character varying
active_storage_blobs	checksum	character varying
active_storage_blobs	id	bigint
active_storage_blobs	byte_size	bigint
active_storage_blobs	created_at	timestamp without time zone
active_storage_blobs	key	character varying
active_storage_blobs	filename	character varying
active_storage_blobs	content_type	character varying
active_storage_variant_records	id	bigint
active_storage_variant_records	variation_digest	character varying
active_storage_variant_records	blob_id	bigint
ar_internal_metadata	value	character varying

- Meeting with **stakeholders** to discuss the required **data tables** from **PgAdmin**

Table	Data Model?	Data Model ID
contacts	N	
profiles	N	
people	N	
data_fields	N	
option_select	N	
option_numb	N	
option_texts	N	
data_models	N	
data_capture	N	
attachments	N	
Pre-Draft Sun	Y	[redacted]
Body Compo	Y	[redacted]
Aerobic	Y	[redacted]
Anaerobic	Y	[redacted]
Flexibility An	Y	[redacted]
Futures	Y	[redacted]
Scout Report	Y	[redacted]
GPS	Y	[redacted]
Psych	Y	[redacted]
Medical Injur	Y	65
AFL Pro-Scou	Y	[redacted]
AFL Draft	Y	[redacted]
Point of Inter	Y	[redacted]
Stats Analysis	Y	[redacted]
Investigation	Y	[redacted]
Character Ass	Y	[redacted]
Home Intervi	Y	[redacted]
Attachment	Y	[redacted]

Step 4: Retrieve all the raw PgAdmin Data Tables in S3 Data Lake

- Using SQL scripts in DBeaver, extract all the required data tables (.csv)



Some SQL scripts for Table extraction are mentioned below:

```
SELECT table_name
FROM information_schema.tables
WHERE table_schema = 'public'
ORDER BY table_name;
```

```
SELECT
table_name,
column_name,
data_type
FROM information_schema.columns
WHERE table_schema = 'public'
ORDER BY table_name;
```

```
SELECT
table_name,
column_name,
data_type
FROM information_schema.columns
WHERE table_schema = 'public'
AND table_name = 'calendar_groups';
SELECT *
FROM public.data_models;
```

```sql
SELECT *
FROM public.data_models
ORDER BY created_at DESC;
```

```sql
SELECT *
FROM public.profiles;
```

```sql
SELECT COUNT(*)
FROM public.attachments;
```

```sql
SELECT *
FROM public.data_captures
WHERE data_model_id = 4;
```

```sql
select subquery.* from (
SELECT public.data_captures.*,
RANK() OVER (
PARTITION BY data_model_id order by created_at desc
)
FROM public.data_captures
WHERE data_model_id IN(1,2,3,4,5,6,7,8)
) subquery where rank<=2
```

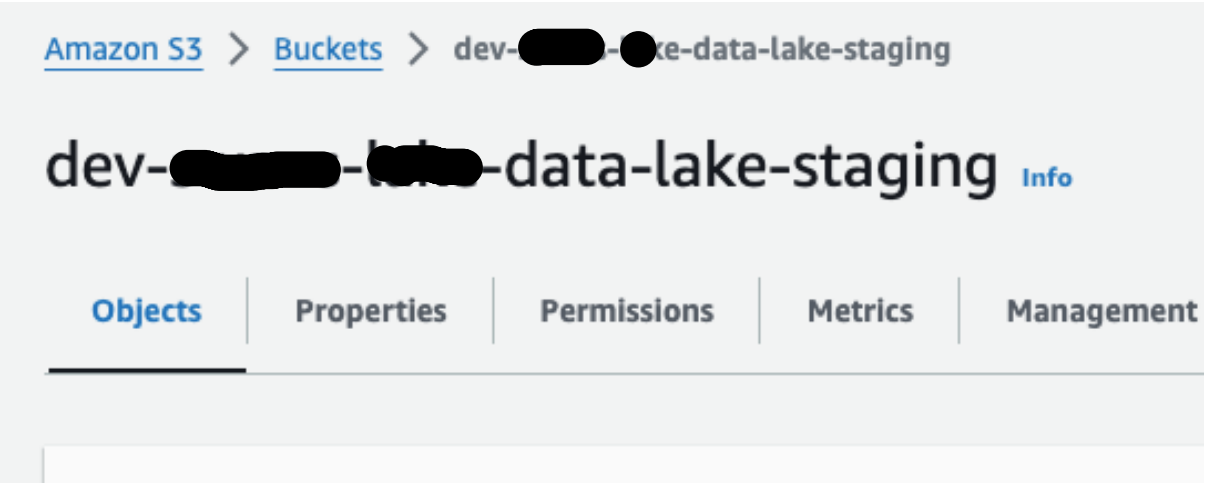
```sql
SELECT *
FROM public.data_captures
WHERE data_model_id in(1,2,3);
```

```sql
SELECT COUNT(*)
FROM public.attachments;
```

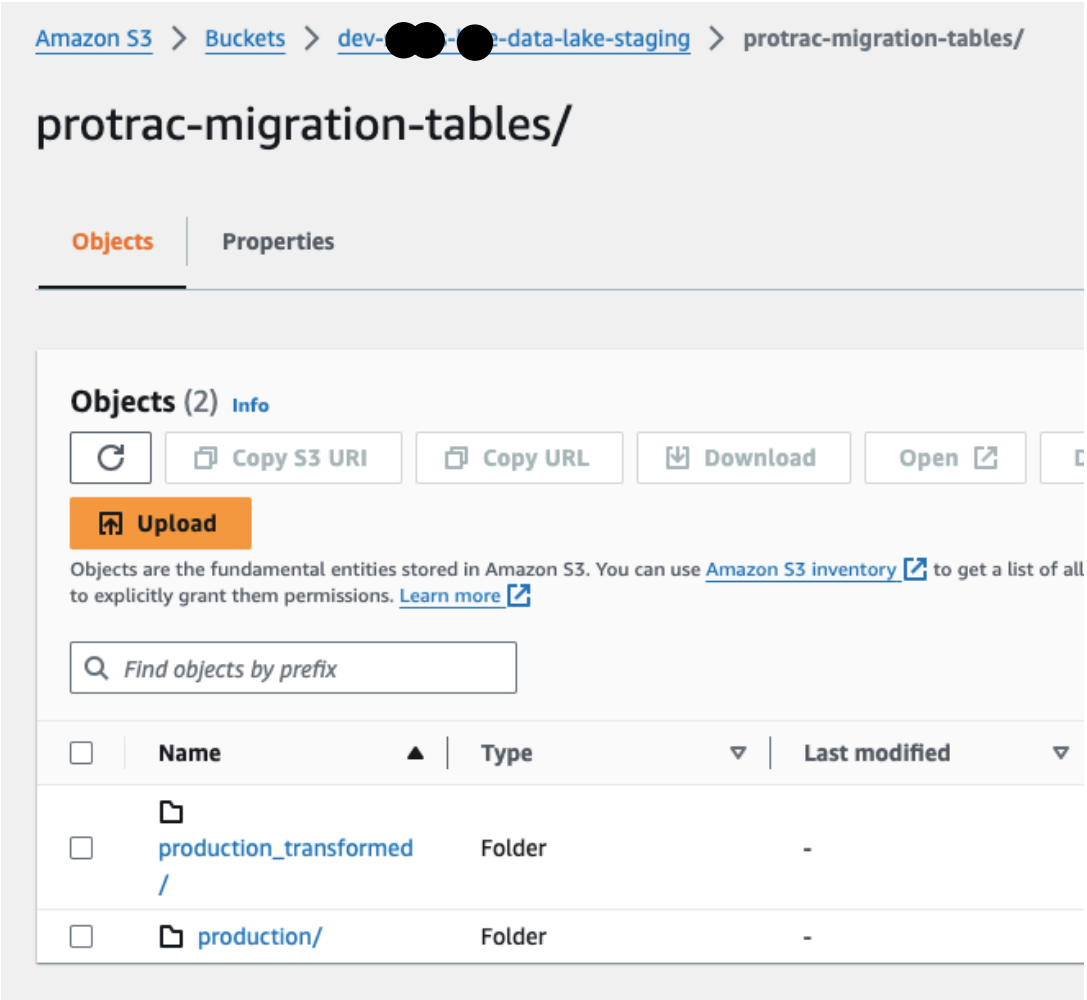
```sql
select *
FROM public.attachments;
```

```sql
SELECT *
FROM public.profiles
ORDER BY created_at DESC;
```

- Create/ Locate an S3 Bucket in Client S3 Data Lake



- Create a **production** folder & store all the raw production data tables (csv) into it



Amazon S3 > Buckets > dev-[redacted]-[redacted]e-data-lake-staging > protrac-migration-tables/ > production/

production/

Copy S3 URI

Objects | Properties

Objects (28) Info

Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder

Upload

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

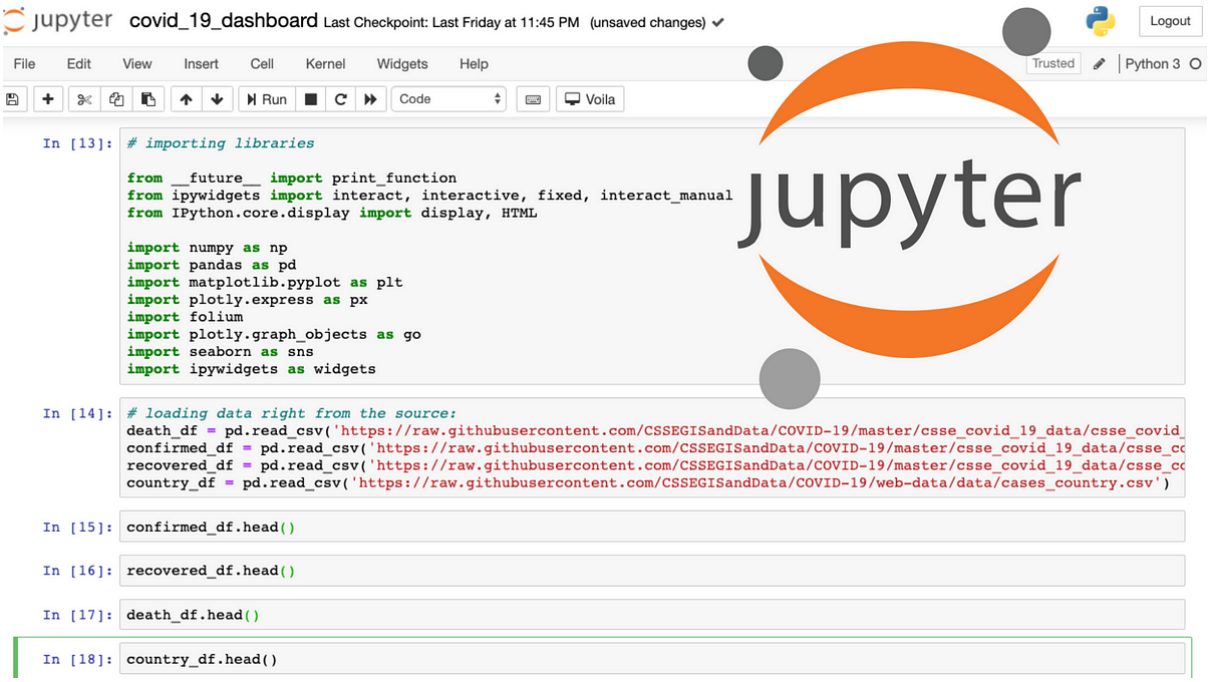
Find objects by prefix

1

Name	Type	Last modified	Size	Storage class
aerobic.csv	csv	January 23, 2024, 18:31:01 (UTC+11:00)	2.9 MB	Standard
afl draft.csv	csv	January 23, 2024, 18:31:03 (UTC+11:00)	472.9 KB	Standard
afl pro scout.csv	csv	January 23, 2024, 18:31:04 (UTC+11:00)	170.3 KB	Standard
anaerobic.csv	csv	January 23, 2024, 18:31:09 (UTC+11:00)	2.2 MB	Standard

Step 5: Data Cleaning/ Transformation on PgAdmin Data Tables residing in S3 Data Lake

- Install Jupyter Notebook/ Python IDE in the EC2 windows instance



- Run python scripts to read all raw production data files from S3 Bucket

To access a CSV file residing in an S3 bucket from a Jupyter notebook using Python, you can use the `boto3` library, which is the official AWS SDK for Python. First, you need to install the `boto3` library if you haven't already:

```
pip install boto3
```

Then, you can use the following code in your Jupyter notebook to access the CSV file:

```
import pandas as pd
import boto3
from io import StringIO
# Define AWS credentials and S3 bucket name
aws_access_key_id = 'YOUR_ACCESS_KEY_ID'
aws_secret_access_key = 'YOUR_SECRET_ACCESS_KEY'
bucket_name = 'YOUR_BUCKET_NAME'
file_key = 'path/to/your/file.csv'  # Path to the CSV file in the bucket
# Create an S3 client
s3 = boto3.client('s3', aws_access_key_id=aws_access_key_id, aws_secret_access_key=aws_secret_access_key)
```

```
# Read the CSV file from S3
obj = s3.get_object(Bucket=bucket_name, Key=file_key)
df = pd.read_csv(obj['Body'])
# Display the DataFrame
print(df)
```

- Run python scripts to transform all raw production data tables (csv) in S3 Bucket

  So, we are going to follow a set of Data Cleaning/ Transformation steps as mentioned below:

  A. **<u>Step A:</u>** Confirm all data is present (NULL values issue) after splitting **properties** column data

⇒ **Now, we have to find the best way to do split:**

- Step 1: Replace NULL with " "
- Step 2: Split with **", "**

Hence, the NULL values issue is fixed, and we are getting all the data.

B. **<u>Step B:</u>** Removing HTML Tags

Almost all data tables got some columns with HTML Tags inside them, we can get through that using the script below:

```
[ ] # In `stats analysis` table

[ ] clean_data_list = []

    for i in df['statsbio']: # remove all HTML tags from the strings
        clean_data_list.append( re.sub("<.*?>", "", i))

    # Display the result
    print(clean_data_list)
    print(len(clean_data_list))

    df['statsbio'] = clean_data_list

    ['Champion Data Stats', 'Champion Data Stats', 'Champion Data Stats', 'Champion Data Stats',
    1229
```

Miscellaneous: Remove other special characters as well.
Example: \r \n , &, *, etc.

C. **<u>Step C:</u>** Need to replace Dropdown Categories from PgAdmin

⇒ Column values before replacing dropdown categories in **preferred_foot** column:

preferred_foot
NaN
7636
7635
7636
NaN
7636
NaN

Python script applied on **preferred_foot** column below:

```
for i in range(0, len(df)):
    for j in range(0, len(dropdown_options)):
        if df['preferred_foot'][i][:-2] == str(dropdown_options['id'][j]):
            df['preferred_foot'][i] = str(dropdown_options['description'][j])
print(df['preferred_foot'])
```

```
0
1       Right
2        Left
3       Right
4
        ...
7688    Right
7689
7690    Right
7691     Left
7692    Right
Name: preferred_foot, Length: 7693, dtype: object
```

D. **<u>Step D:</u>** Numerical conversions (mm -> cm or g -> kg) of Height & Weight columns

```
df['weight'] = pd.to_numeric(df['weight'], errors='coerce')
df['weight'] = df['weight'] / 1000
df['weight']
```

```
0        81.0
1        79.6
2        70.5
3        83.8
4        81.3
         ...
10852     0.0
10853    84.8
10854    71.8
10855    90.2
10856    76.3
Name: weight, Length: 10857, dtype: float64
```

E. **<u>Step E:</u>** Calculation Age from Date of Birth

```
import pandas as pd
from datetime import datetime


# Convert Date_of_Birth column to datetime
df_profiles['age'] = pd.to_datetime(df_profiles['age'], errors = 'coerce')


# Function to calculate age
def calculate_age(dob):
    current_date = datetime.now()
    age = current_date.year - dob.year - ((current_date.month, current_date.day) < (dob.month, dob.day))
    return age

# Apply calculate_age function to Date_of_Birth column
df_profiles['Age'] = df_profiles['age'].apply(calculate_age)

df_profiles
```

talent_list	family_sport	brother_sister	preferred_foot	professional_goals	likely_afl_position	other_sports_played	Age
NaN	NaN	NaN	NaN	NaN	Outside Midfielder		27.0
NaN	NaN	NaN	7636.0	NaN	Ruck		36.0

emotely or in another tab. Show diff

Output in **Age** column:

Date of Birth	Age
1996-03-23	27.0
1988-01-08	36.0
2004-08-13	19.0
2004-09-13	19.0
1992-07-17	31.0

F. **<u>Step F:</u>** Renaming columns

```
[ ] df_profiles.columns
```

```
Index(['id', 'person_id', 'data_model_id', 'properties', 'created_at',
       'updated_at', 'age', 'club', 'honors', 'category', 'comments',
       'schooling', 'afl_history', 'talent_list', 'family_sport',
       'brother_sister', 'preferred_foot', 'professional_goals',
       'likely_afl_position', 'other_sports_played', 'Age'],
      dtype='object')
```

```
# Dictionary mapping old column names to new column names

new_names = {  'person_id':'UserID', 'age':'Date of Birth',
       'club':'Club', 'honors':'Honors',
       'category':'Category', 'comments':'Comments',
               'schooling':'School', 'afl_history':'AFL History',
               'family_sport':'Family Sport', 'brother_sister':'Siblings',
               'professional_goals':'Professional Goals'
               , 'likely_afl_position':'AFL Role',
               'other_sports_played':'Other Sports Played',
               'talent_list':'Talent List'
}
```

```
[ ] # Rename columns
    df_profiles = df_profiles.rename(columns=new_names)
    df_profiles
```

Club	Honors	Category	...	School	AFL History	Talent List	Family Sport	Siblings	preferred_foot	Professional Goals	AFL Role	Other Sports Played	Age

- Finally, store all transformed production data files into S3 Bucket in a new folder

Amazon S3 > Buckets > dev[redacted]e-data-lake-staging > [redacted]-tables/ > production_transformed/

production_transformed/

Copy S3 URI

Objects | Properties

Objects (17) Info

Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder

Upload

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Find objects by prefix

1

Name	Type	Last modified	Size	Storage class
aerobic_transformed.csv	csv	January 29, 2024, 03:32:37 (UTC+11:00)	1.5 MB	Standard
afl draft_transformed.csv	csv	January 29, 2024, 03:32:38 (UTC+11:00)	143.3 KB	Standard
afl pro scout_transformed.csv	csv	January 29, 2024, 03:32:39 (UTC+11:00)	35.5 KB	Standard
anaerobic_transformed.	csv	January 29, 2024, 03:32:41 (UTC+11:00)	1.2 MB	Standard

Step 6: Data Mapping from PgAdmin Data Tables to Smartabase API schema

Below is an example of Data Mapping from PgAdmin to Smartabase table:

i. **PgAdmin Table**: Pre-Draft Summary

Protrac Table Name	Data Model	Data Model ID	Column Name	Properties	Data Type
Pre-Draft Summary	Y	81	data_model_id	N	Int
Pre-Draft Summary	Y	81	person_id	N	Int
Pre-Draft Summary	Y	81	data_capture_date	N	Int
Pre-Draft Summary	Y	81	firstname	N	String
Pre-Draft Summary	Y	81	lastname	N	String
Pre-Draft Summary	Y	81	club	Y	String
Pre-Draft Summary	Y	81	draft_rd	Y	Drop Down
Pre-Draft Summary	Y	81	draft_yr	Y	Drop Down
Pre-Draft Summary	Y	81	position	Y	Drop Down
Pre-Draft Summary	Y	81	strengths	Y	String
Pre-Draft Summary	Y	81	weaknesses	Y	String

ii. **Smartabase Table:** Recruiting Pre Draft Summary

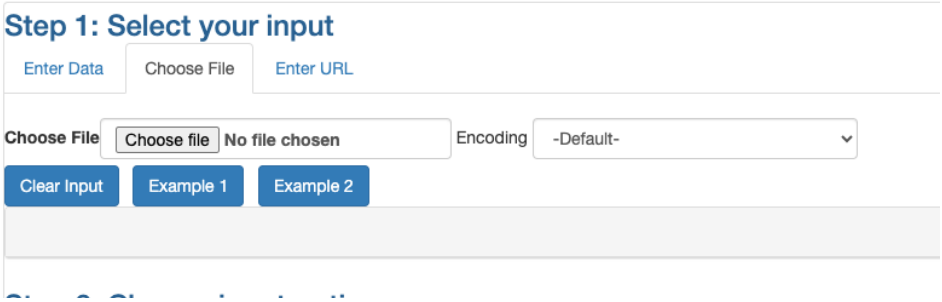
Smartabase Object	Smartabase Field	Transformation/Mapping/Notes	SB Notes
Recruiting - Pre Draft Summary			
Recruiting - Pre Draft Summary	UserID		[redacted] Us[redacted] could be different to [redacted] UserID
Recruiting - Pre Draft Summary	On Date		Confirm field name
Recruiting - Pre Draft Summary	First Name		
Recruiting - Pre Draft Summary	Last Name		
Recruiting - Pre Draft Summary	Club		
Recruiting - Pre Draft Summary	Likely Draft		Check drop down
Recruiting - Pre Draft Summary	Draft Year		Check drop down
Recruiting - Pre Draft Summary	Position		Check drop down
Recruiting - Pre Draft Summary	Strengths		
Recruiting - Pre Draft Summary	Waknesses		

Make sure the data is in proper structure and form before pushing into Smartabase.

Step 7: Pushing Data Tables into Smartabase

Using POSTMAN to push data into the SmartaBase:

- First, convert the CSV file to JSON



- That's the schema to Push data into the SmartaBase

	On Date	First Name	Last Name	Poise	Comments	Work Ethic	Decision Making	Professionalism	Vision Awareness	Personality Profile	Introverted / Extroverted
0	2008-04-09	[redacted]	[redacted]	NULL	[redacted] is a very intelligent young man.	NULL	NULL	NULL	NULL	NULL	NULL
1	2008-04-11	[redacted]	[redacted]	3	NULL	5	4	5	3	NULL	NULL
2	2008-05-09	[redacted]	Lynch	NULL	[redacted]ch described himself as being concerned wit...	NULL	NULL	NULL	NULL	NULL	NULL
3	2008-07-03	[redacted]	[redacted]	NULL	Sounds a very articulate & focused young man o...	NULL	NULL	NULL	NULL	NULL	NULL
4	2008-11-05	[redacted]	[redacted]	4	Impressive young man.	3	4	4	4	Chame[redacted] - F[redacted] 30, Thinker 27.5, Enforcer ...	NULL

- Then, POST using POSTMAN: