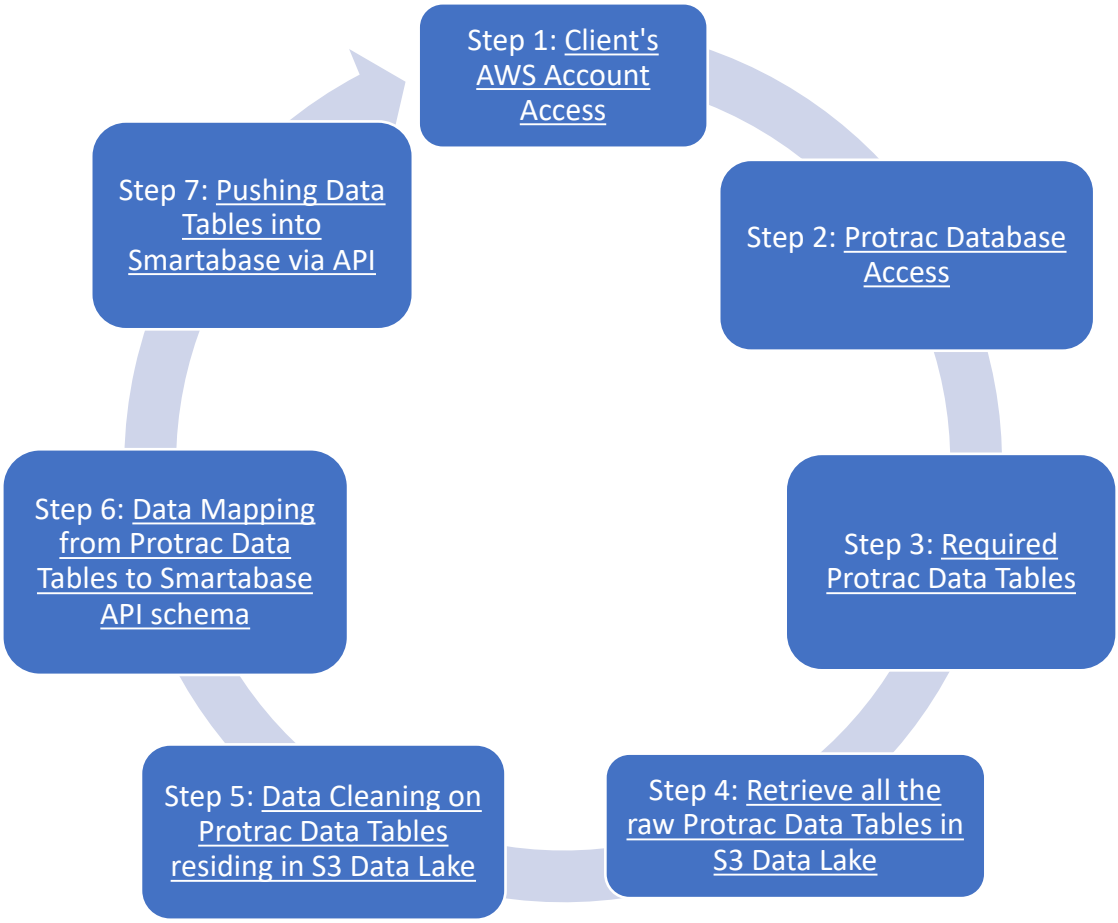
Data Migration Project
Content

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 **Protrac_swans (Production)** data server in **DBeaver** through **SSH Tunnel**
4. Extract required Protrac Data Tables
   - Proving a list of **Table name, Column name, and Data type** to **Thomas** using **DBeaver SQL scripts**
   - Meeting with int/ext s**takeholders** to discuss the required **data tables** from **Protrac**
5. Ingest raw Protrac 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 Protrac
     - 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 Protrac Data Tables to SmartaBase Schema
8. Pushing finalised Data Tables into SmartaBase API via Postman or Python API connection

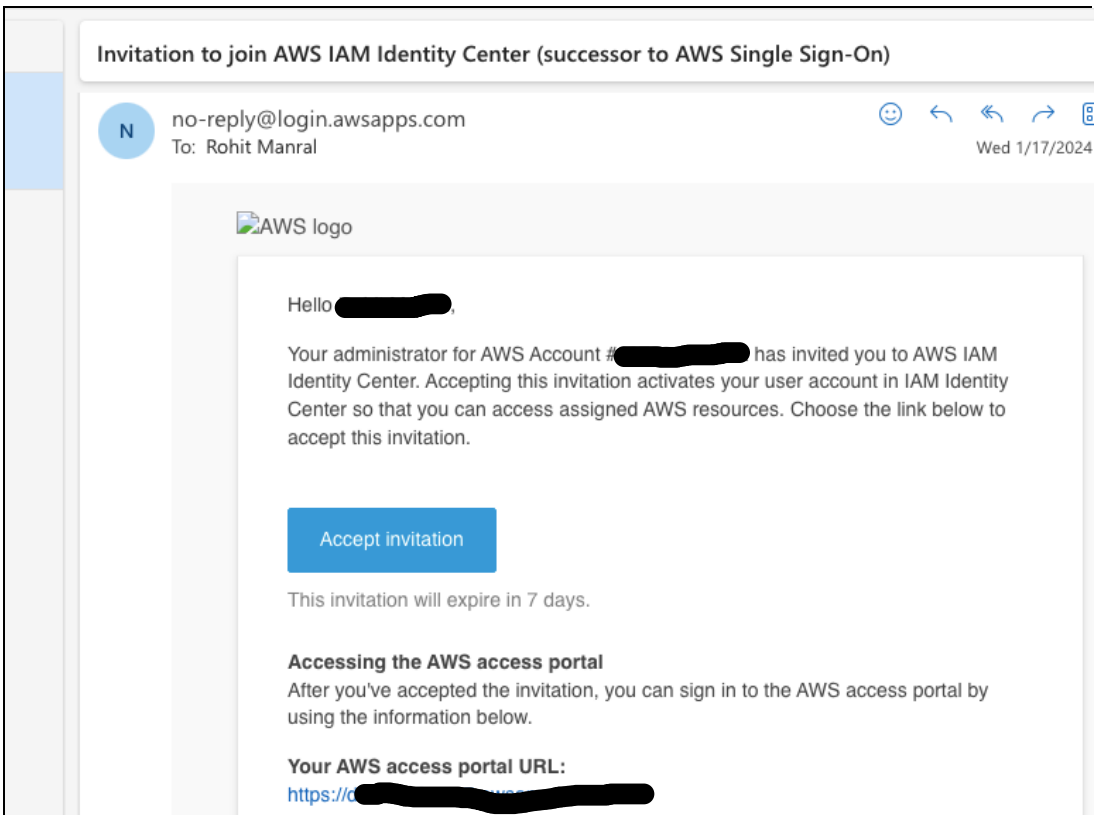
- convert the CSV file to JSON
- Finalise data tables to Push into the SmartaBase
- POST using POSTMAN

Data Migration Process Flow

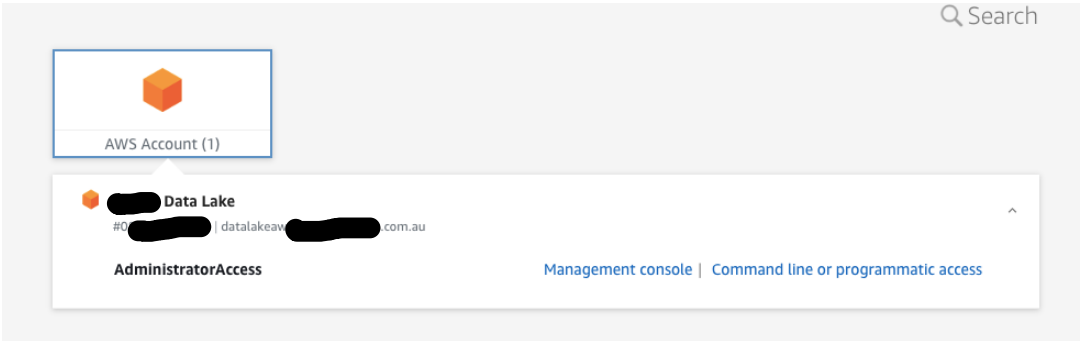


Step 1: Client's AWS Account Access

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

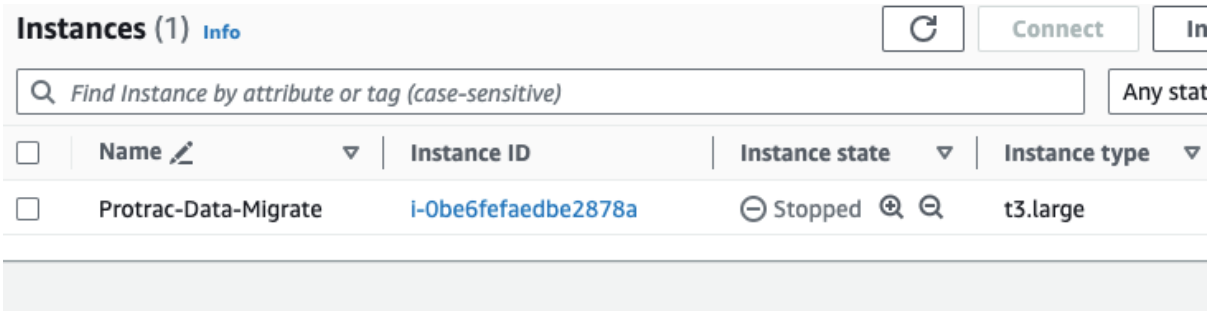


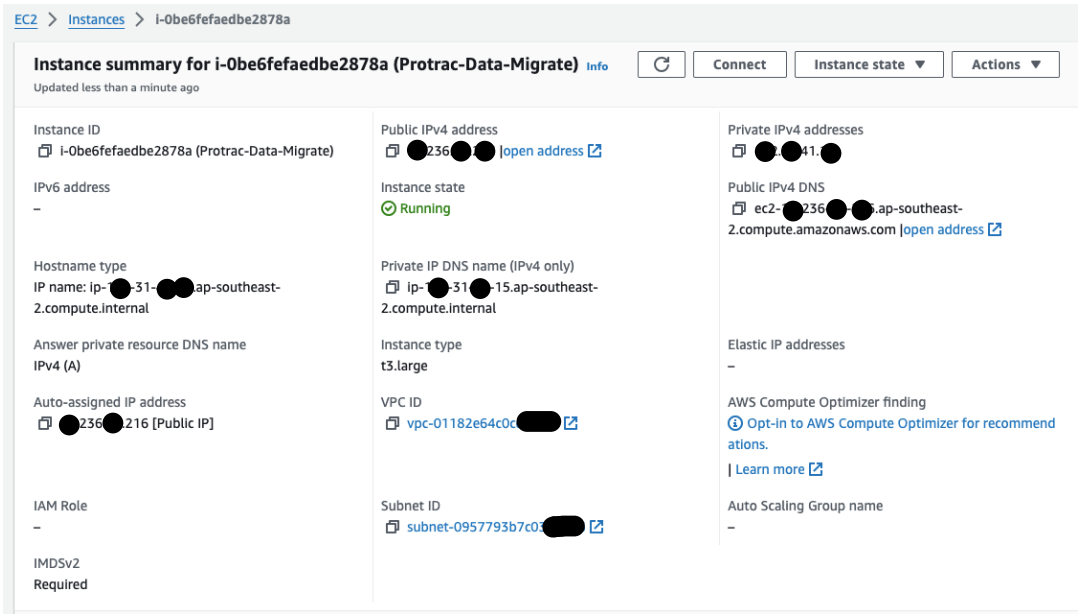
    - Go to **Management Console**

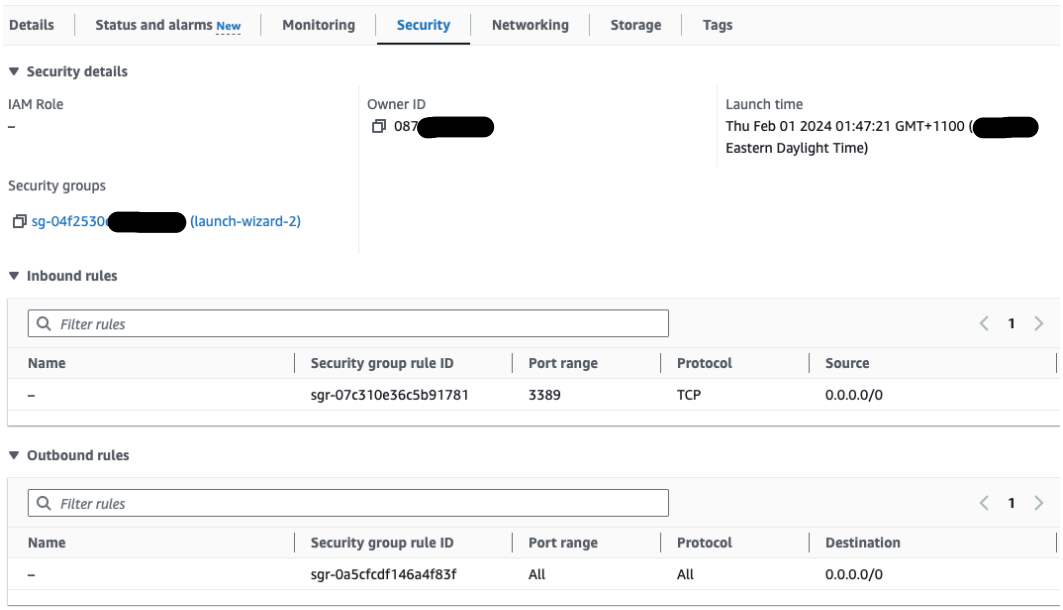


Step 2: Protrac Database Access

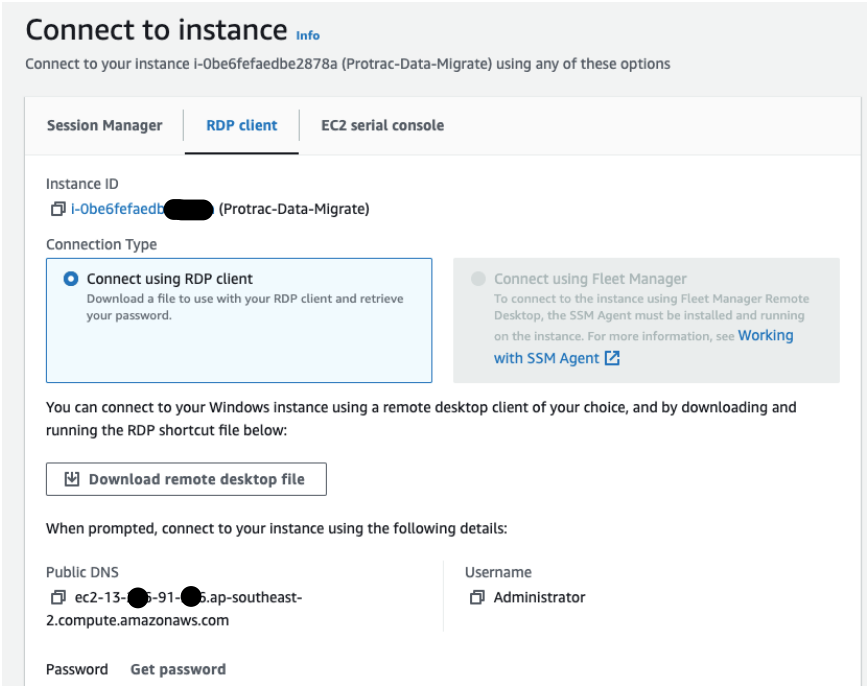
- Create a new **EC2 windows instance (t3.large)**



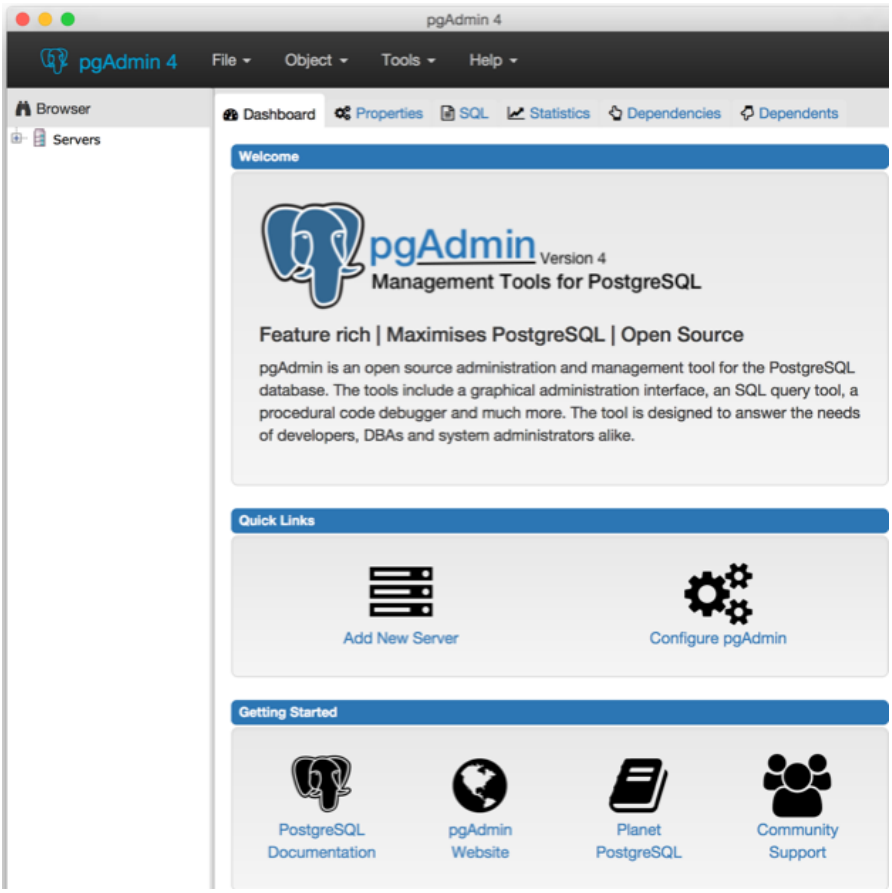




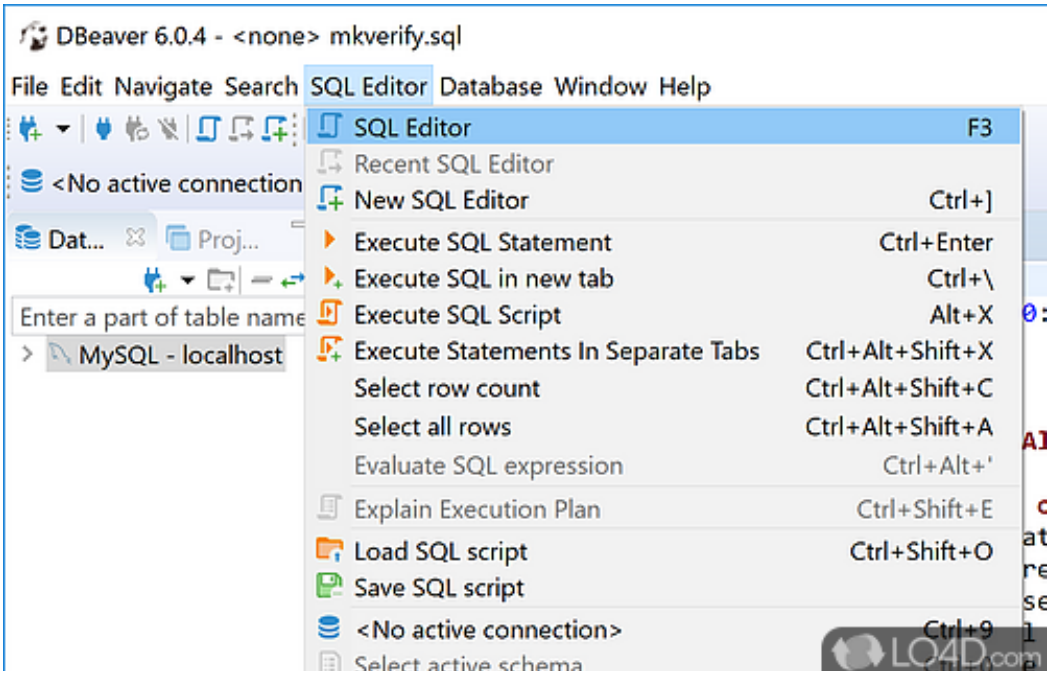
- Now, connect to the **EC2 windows instance (t3.large)** by downloading **Remote Desktop File**



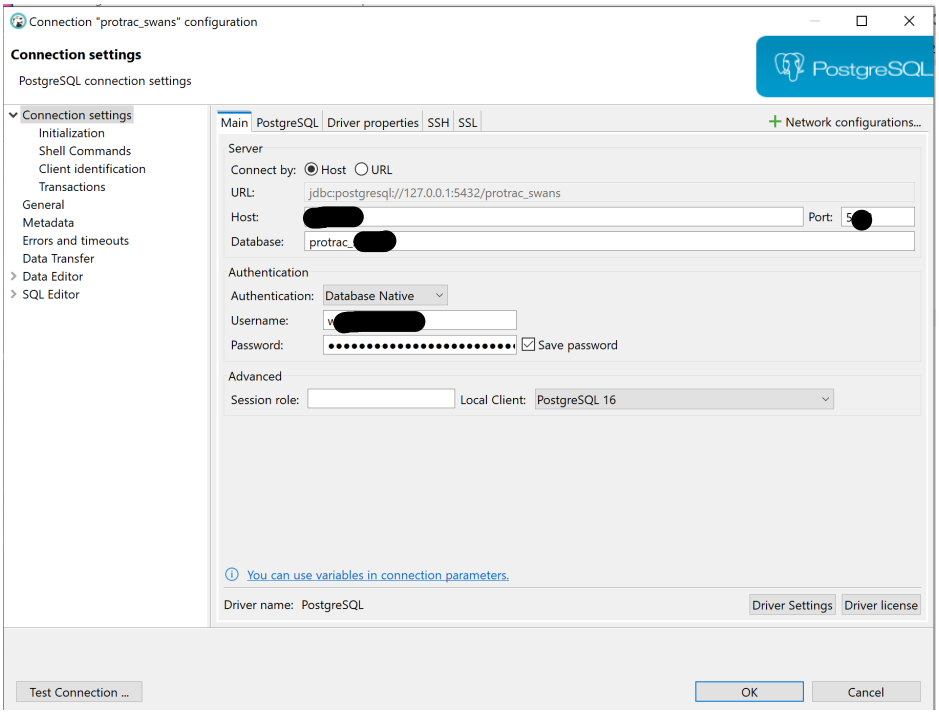
  - Install **PgAdmin (PostgreSQL 16)**

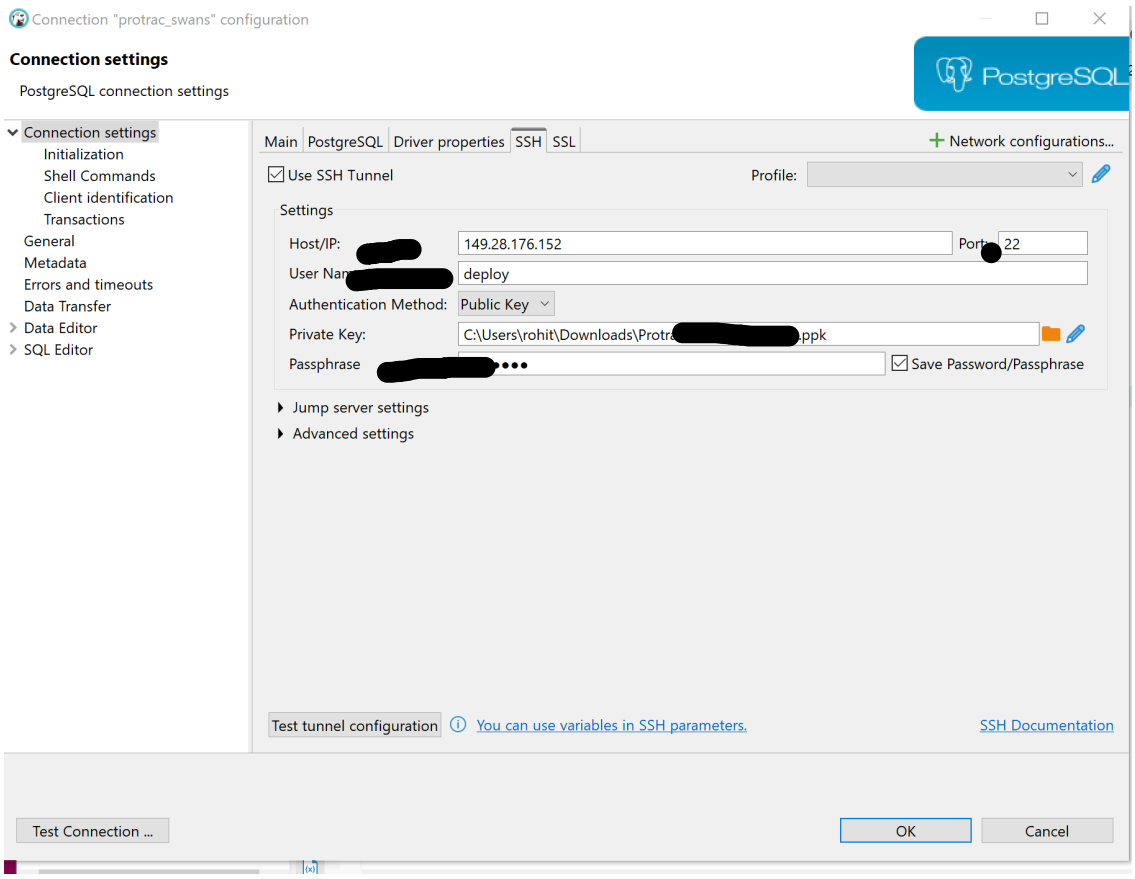


- Install **DBeaver**



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





Step 3: Required Protrac Data Tables

- Proving a list of **Table name, [redacted] 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 **Client** to discuss the required **data tables** from **PgAdmin data server**

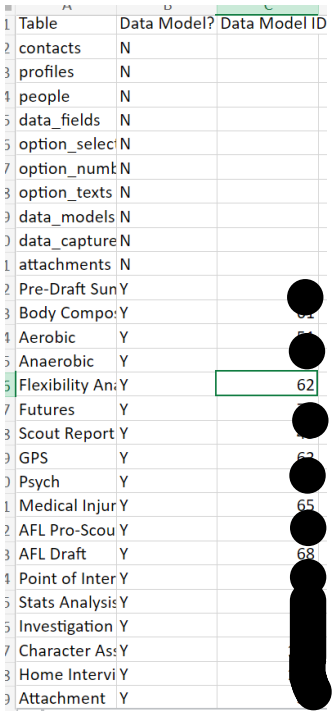
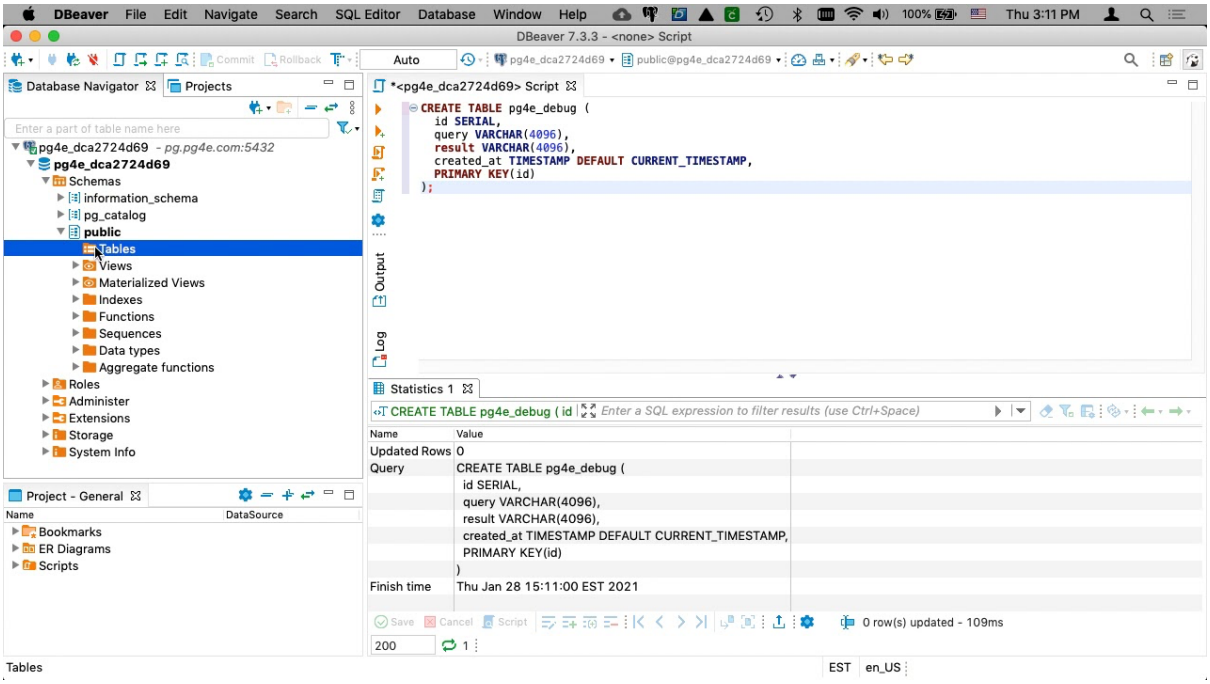


	Table	Data Model?	Data Model ID
2	contacts	N	
3	profiles	N	
4	people	N	
5	data_fields	N	
6	option_selec	N	
7	option_numb	N	
8	option_texts	N	
9	data_models	N	
10	data_capture	N	
11	attachments	N	
12	Pre-Draft Sun	Y	
13	Body Compo:	Y	
14	Aerobic	Y	
15	Anaerobic	Y	
16	Flexibility An	Y	62
17	Futures	Y	
18	Scout Report	Y	
19	GPS	Y	
20	Psych	Y	
21	Medical Injur	Y	65
22	AFL Pro-Scou	Y	
23	AFL Draft	Y	68
24	Point of Inter	Y	
25	Stats Analysis	Y	
26	Investigation	Y	
27	Character As:	Y	
28	Home Intervi	Y	
29	Attachment	Y	

Step 4: Retrieve all the required raw 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:

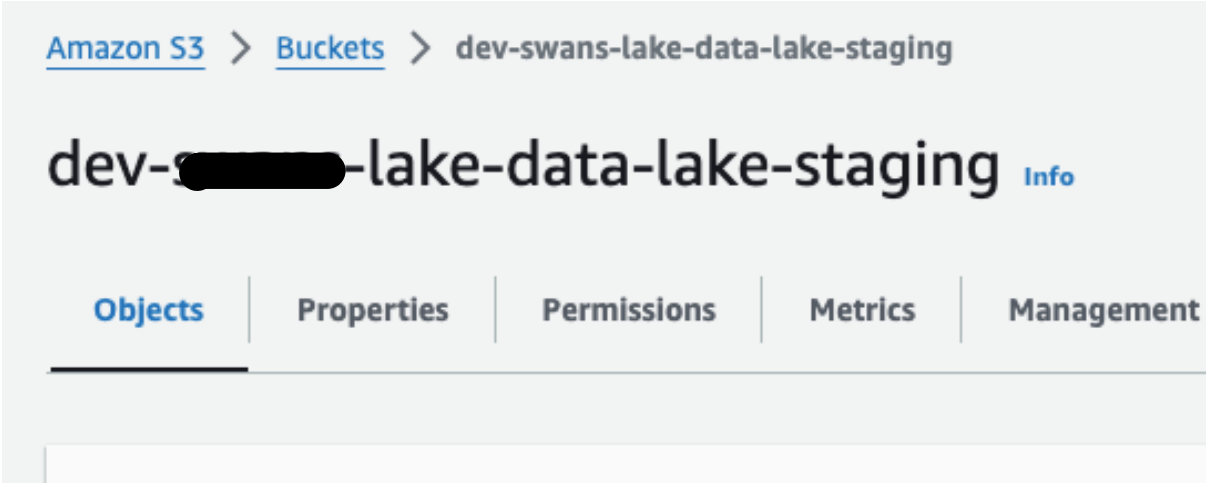
```
SELECT table_name
FROM information_schema.tables
WHERE table_schema = 'public'
ORDER BY table_name;
```

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

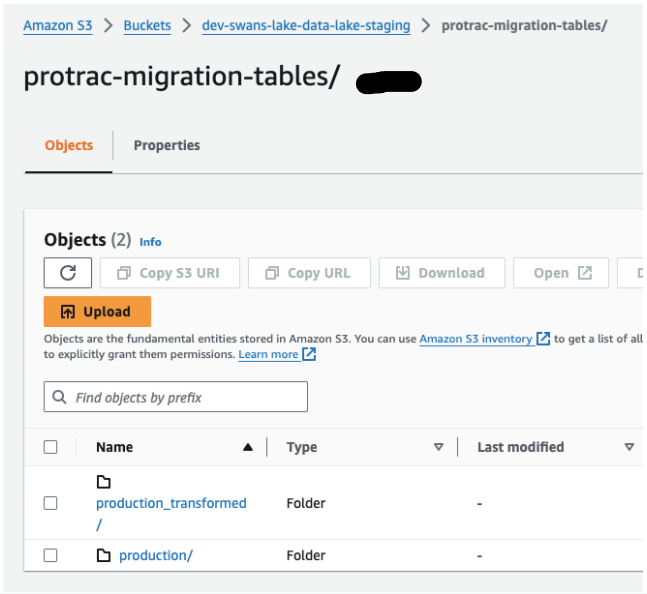
```
SELECT
table_name,
column_name,
data_type
FROM information_schema.columns
WHERE table_schema = 'public'
AND table_name = 'calendar_groups';
SELECT *
FROM public.data_models;
```

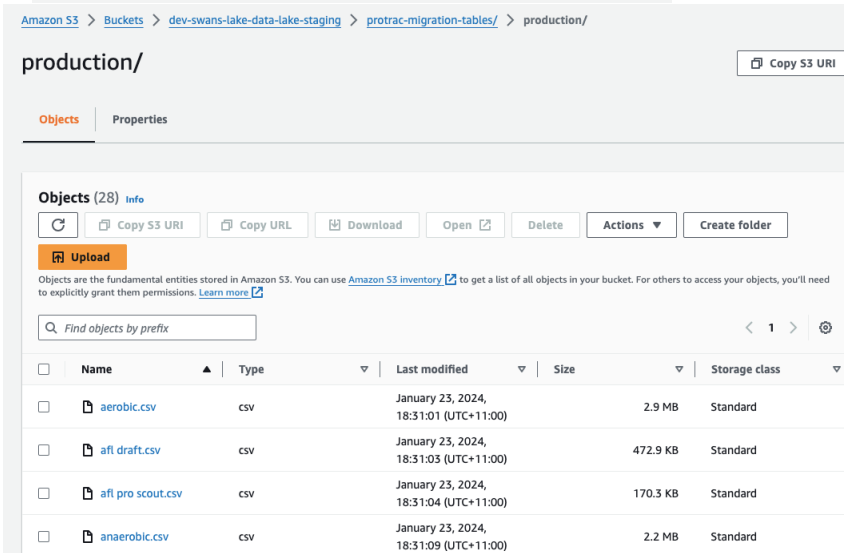
```sql
SELECT *
FROM public.data_models
ORDER BY created_at DESC;



SELECT *
FROM public.profiles;


SELECT COUNT(*)
FROM public.attachments;


SELECT *
FROM public.data_captures
WHERE data_model_id = 4;


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


SELECT *
FROM public.data_captures
WHERE data_model_id in(71,2,3,4);


SELECT COUNT(*)
FROM public.attachments;


select *
FROM public.attachments;


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

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



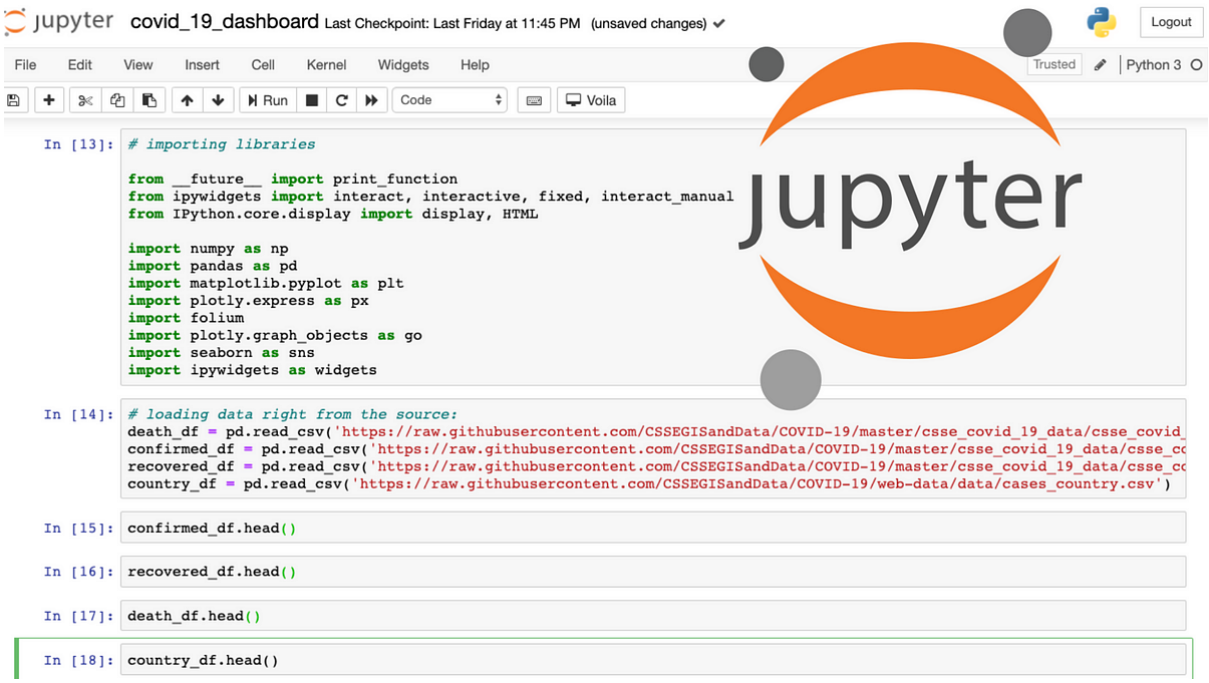
  - Create a **production** folder & store all the raw production data tables (csv) into it





Step 5: Data Cleaning/ Transformation on Protrac 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:

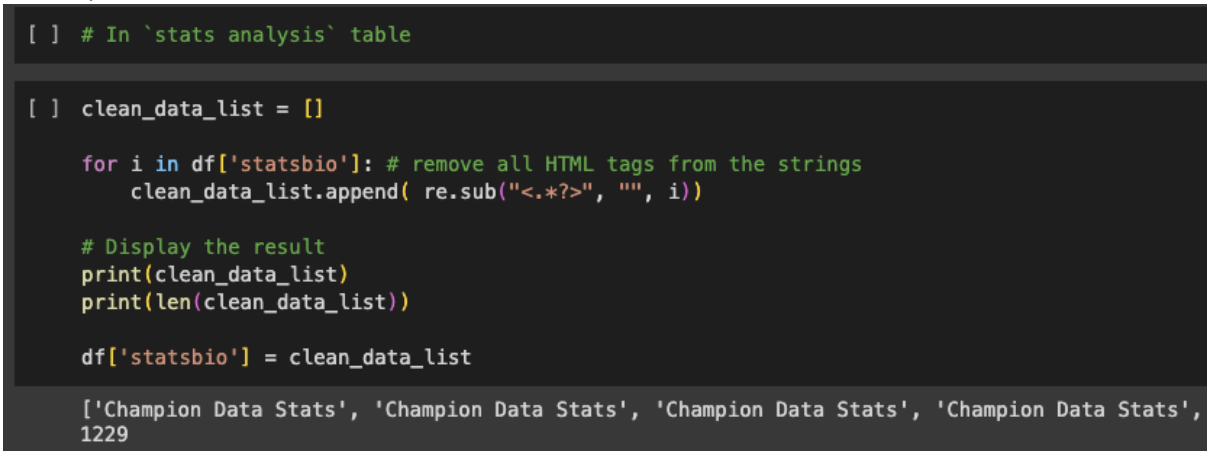
  A. **Step A:** Confirm all data is present (NULL values issue) after splitting **properties** column data

⇒ **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. **Step B:** 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. **Step C:** Need to replace Dropdown Categories from Protrac

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

D. **Step D:** 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. **Step E:** 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. **Step F:** Renaming columns

```
[ ] df_profiles.columns
```

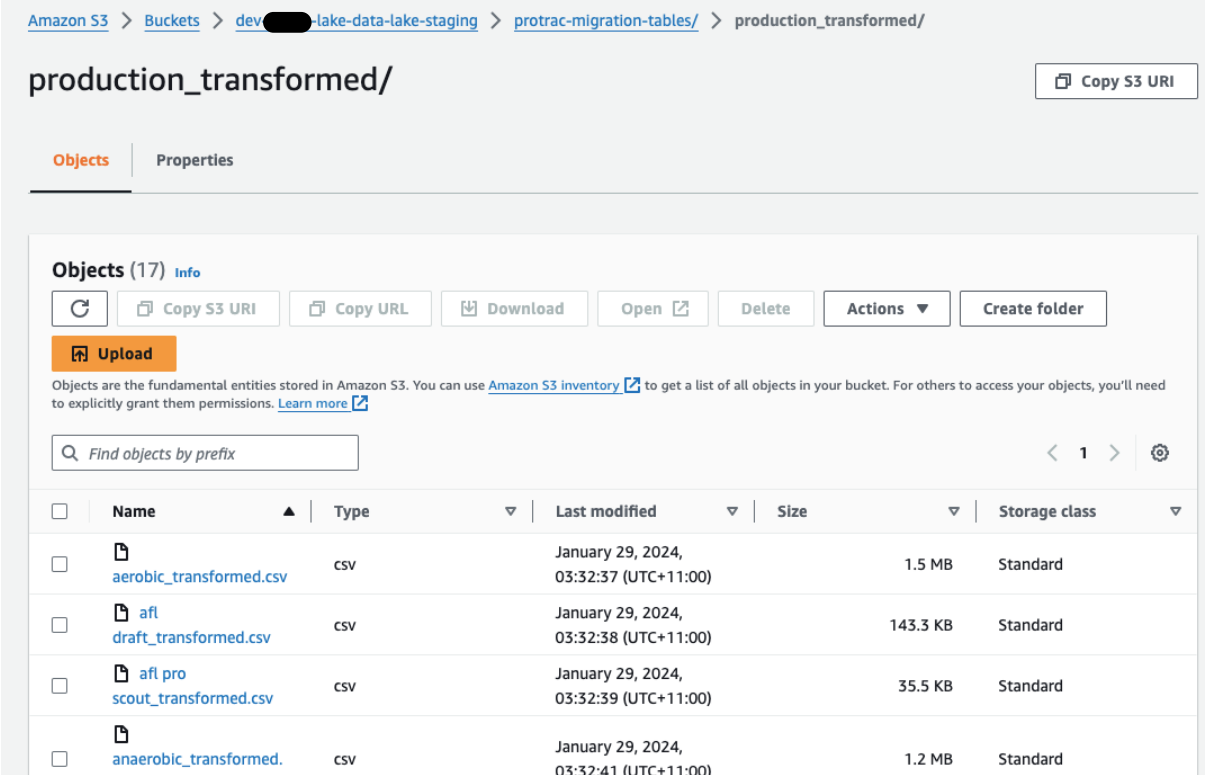
```
Index(['id', 'person_id', 'data_model_id', 'properties', 'created_at',
       'updated_at', 'age', 'club', 'honors', 'category', 'comments',
       'schooling', 'afl_history', 'talent_list', 'family_sport',
       'brother_sister', 'preferred_foot', 'professional_goals',
       'likely_afl_position', '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



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. **Protrac 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

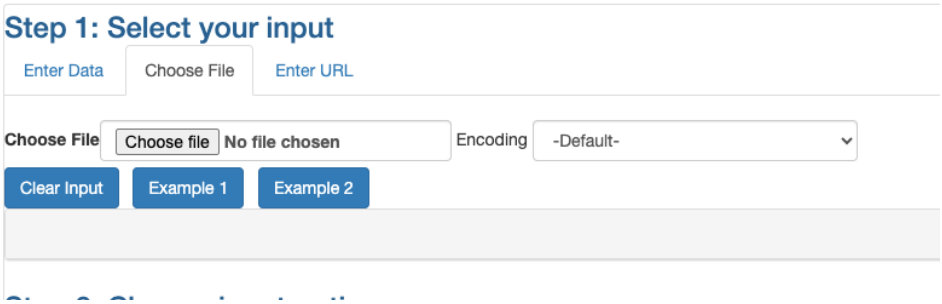
Smartabase Object	Smartabase Field	Transformation/Mapping/Notes	SB Notes
Recruiting - Pre Draft Summary			
Recruiting - Pre Draft Summary	UserID		SB UserID could be different to Protrac UserID
Recruiting - Pre Draft Summary	On Date		Confirm field name
Recruiting - Pre Draft Summary	First Name		
Recruiting - 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 into JSON format



- 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]s a very intelligent young man.	NULL	NULL	NULL	NULL	NULL	NULL
1	2008-04-11	J[redacted]	[redacted]	3	NULL	5	4	5	3	NULL	NULL
2	2008-05-09	[redacted]	L[redacted]	NULL	[redacted] described himself as being concerned wit...	NULL	NULL	NULL	NULL	NULL	NULL
3	2008-07-03	J[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	Ch[redacted] ...	NULL

- Then, POST using POSTMAN: