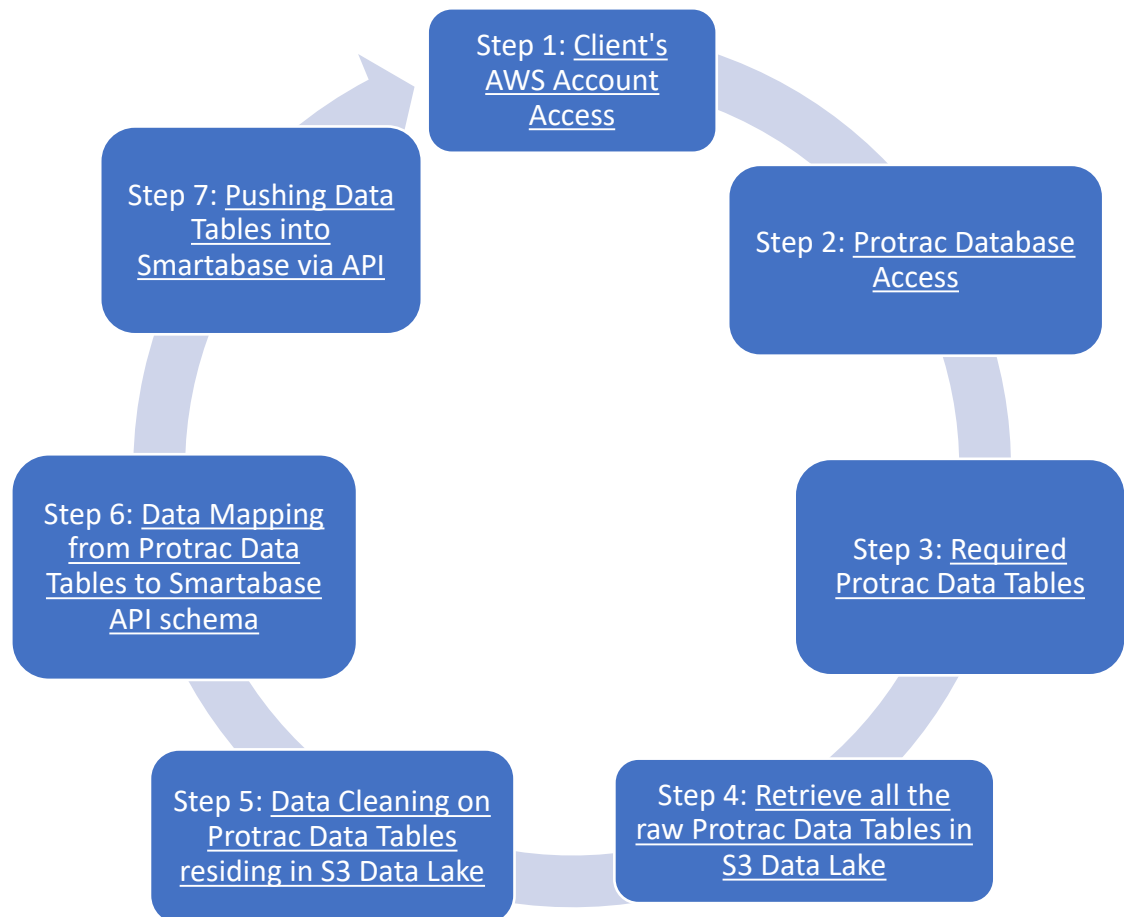


# Data Migration Project

## Content

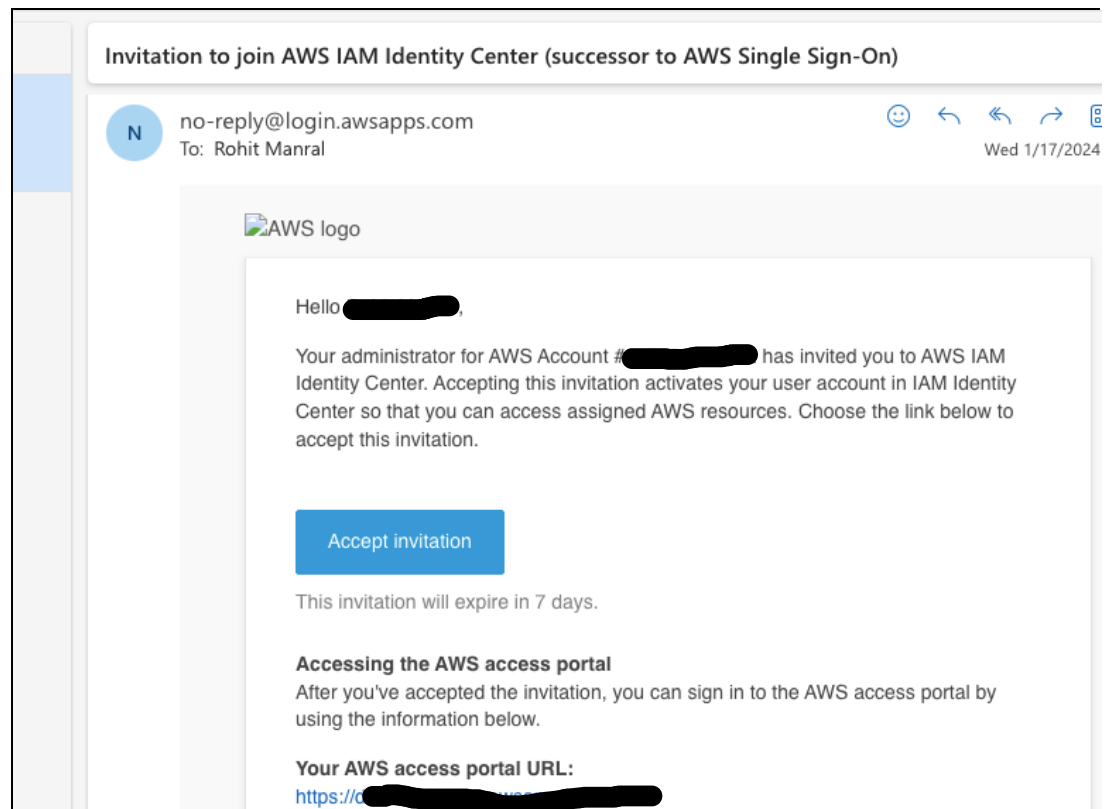
1. Process Flow Diagram
2. Client's AWS Account Access
3. PgAdmin (PostgreSQL) Database Access
  - Create a new **EC2 windows instance (t3.large)**
  - connect to the **EC2 windows instance (t3.large)** by downloading **Remote Desktop File**
    - Install **PgAdmin (PostgreSQL 16)**
    - Install **DBeaver**
  - Connect to **Protrac\_swans (Production)** data server in **DBeaver** through **SSH Tunnel**
4. Extract required Protrac Data Tables
  - Providing a list of **Table name, Column name, and Data type** to **Thomas** using **DBeaver SQL scripts**
  - Meeting with int/ext **stakeholders** 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

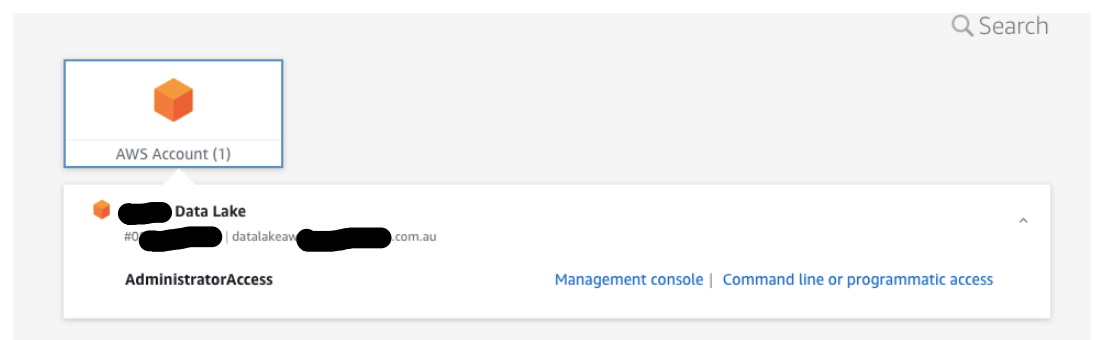


## Step 1: Client's AWS Account Access

- **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)**

Instances (1) <a href="#">Info</a>		<a href="#">Refresh</a>	<a href="#">Connect</a>	<a href="#">In</a>
<input type="text" value="Find Instance by attribute or tag (case-sensitive)"/>		<a href="#">Any stat</a>		
<input type="checkbox"/>	Name <a href="#">✎</a> ▼	Instance ID	Instance state ▼	Instance type ▼
<input type="checkbox"/>	Protrac-Data-Migrate	<a href="#">i-Obe66fefaedbe2878a</a>	<a href="#">⊖</a> Stopped <a href="#">⊕</a> <a href="#">🔍</a> <a href="#">🔍</a>	t3.large

EC2

>

Instances

>

i-0be6fefaedbe2878a

Instance summary for i-0be6fefaedbe2878a (Protrac-Data-Migrate) Info

Updated less than a minute ago

Instance ID

i-0be6fefaedbe2878a (Protrac-Data-Migrate)

IPv6 address

—

Hostname type

IP name: ip-10-31-100-15.ap-southeast-2.compute.internal

Answer private resource DNS name

IPv4 (A)

Auto-assigned IP address

10.236.100.216 [Public IP]

IAM Role

—

IMDSv2

Required

Public IPv4 address

10.236.100.100 [open address](#)

Instance state

Running

Private IP DNS name (IPv4 only)

ip-10-31-100-15.ap-southeast-2.compute.internal

Instance type

t3.large

VPC ID

vpc-01182e64c0c100000 [open](#)

Subnet ID

subnet-0957793b7c0100000 [open](#)

Private IPv4 addresses

10.236.100.41

Public IPv4 DNS

ec2-10-236-100-15.ap-southeast-2.compute.amazonaws.com [open address](#)

Elastic IP addresses

—

AWS Compute Optimizer finding

Opt-in to AWS Compute Optimizer for recommendations.
 [Learn more](#)

Auto Scaling Group name

—

Details

Status and alarms [New](#)

Monitoring

**Security**


Networking

Storage


Tags

▼ Security details

IAM Role  
-

Owner ID  
 087-XXXXXXXXXX

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

Security groups  
 [sg-04f2530](#) XXXXXXXXXX (launch-wizard-2)

▼ Inbound rules

< 1 >

Name	Security group rule ID	Port range	Protocol	Source
-	sgr-07c310e36c5b91781	3389	TCP	0.0.0.0/0

▼ Outbound rules

< 1 >

Name	Security group rule ID	Port range	Protocol	Destination
-	sgr-0a5cfcdf146a4f83f	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-0be6feaedbe2878a (Protrac-Data-Migrate) using any of these options

Session Manager

**RDP client**

EC2 serial console

Instance ID

i-0be6feaedb... (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**

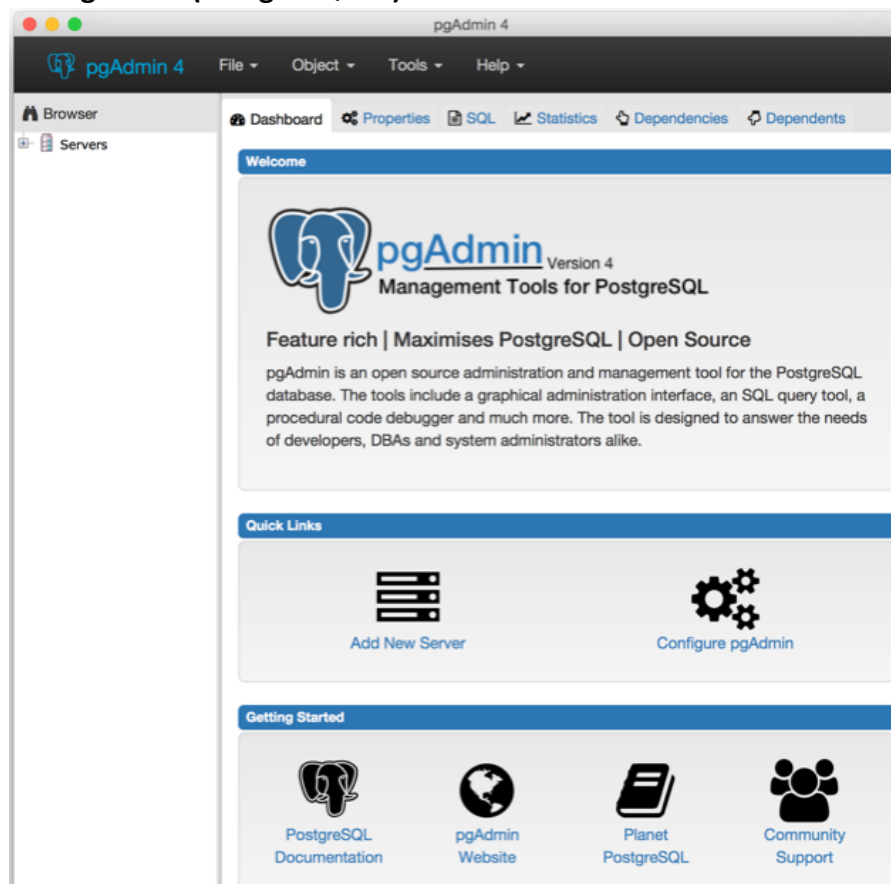
When prompted, connect to your instance using the following details:

Public DNS  
 ec2-13-...-91-...s.ap-southeast-2.compute.amazonaws.com

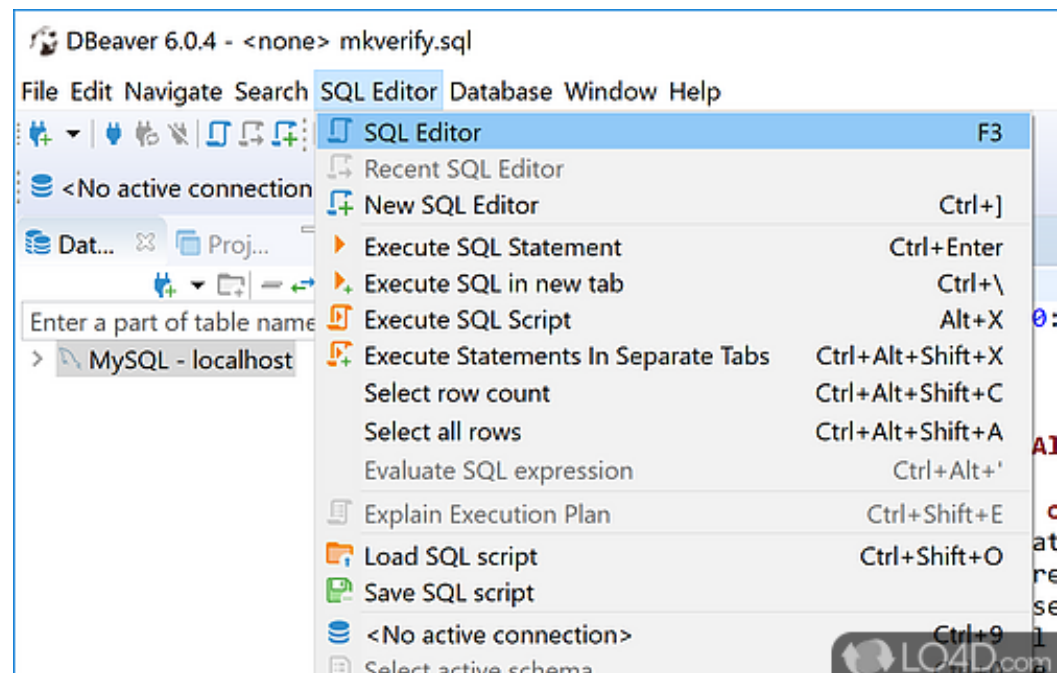
Username  
 Administrator

Password    [Get password](#)

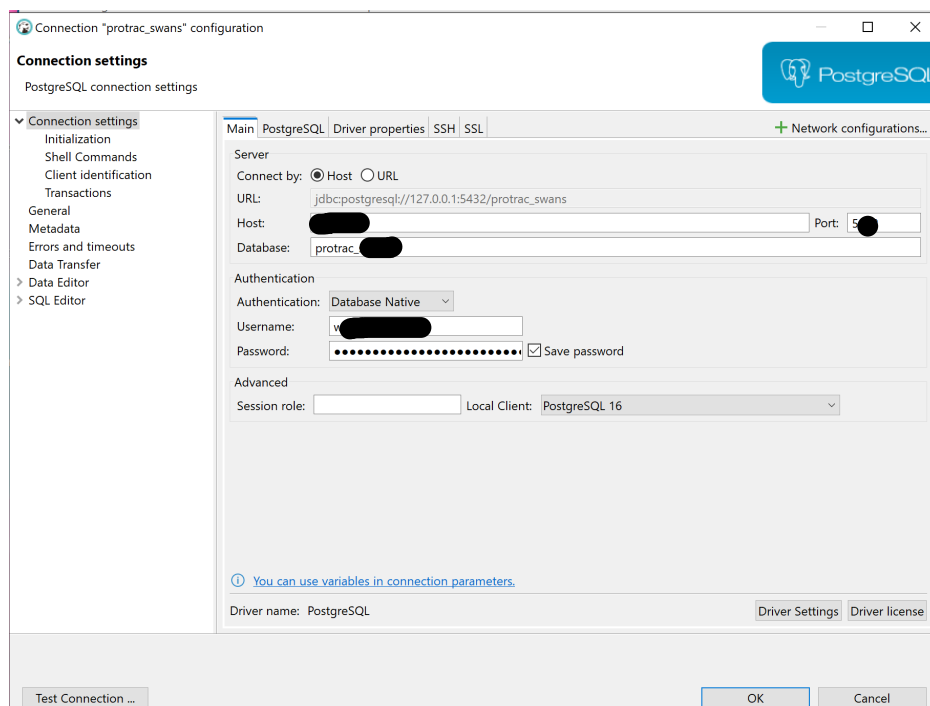
- Install **PgAdmin (PostgreSQL 16)**

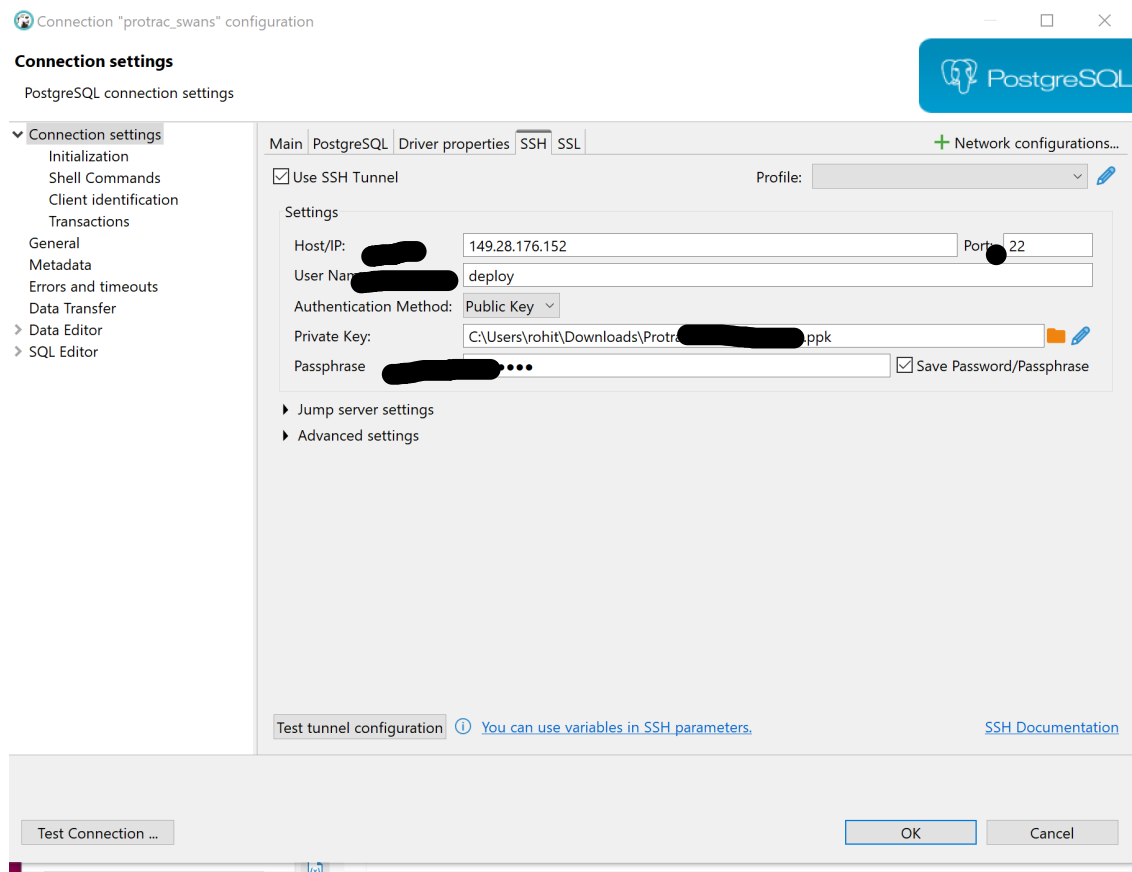


- Install **DBeaver**



- Connect to **Protrac\_swans (Production)** data server in **DBeaver** through **SSH Tunnel**





### Step 3: Required Protrac Data Tables

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

• [REDACTED]

#### **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

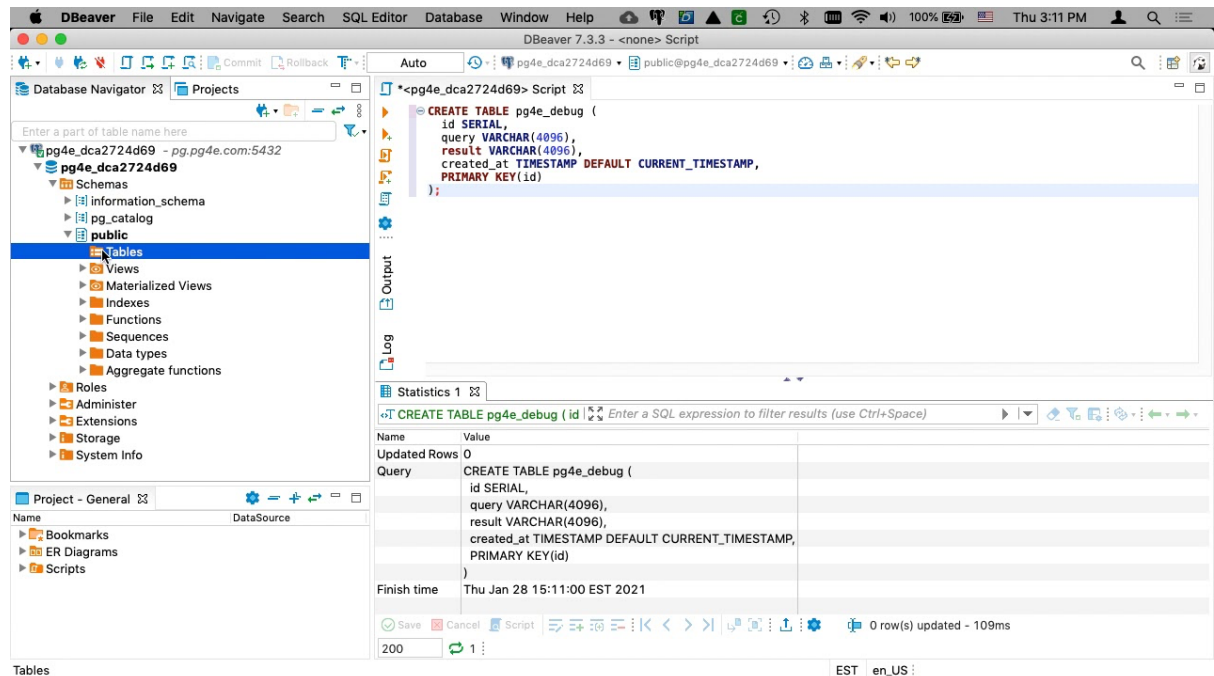
- Meeting with **Client** to discuss the required **data tables** from **PgAdmin data server**

Table	Data Model?	Data Model ID
contacts	N	
profiles	N	
people	N	
data_fields	N	
option_select	N	
option_number	N	
option_texts	N	
data_models	N	
data_capture	N	
attachments	N	
Pre-Draft Sun	Y	
Body Compos	Y	61
Aerobic	Y	62
Anaerobic	Y	63
Flexibility An	Y	62
Futures	Y	
Scout Report	Y	64
GPS	Y	65
Psych	Y	66
Medical Injur	Y	65
AFL Pro-Scou	Y	67
AFL Draft	Y	68
Point of Inter	Y	69
Stats Analysis	Y	70
Investigation	Y	71
Character As	Y	72
Home Intervi	Y	73
Attachment	Y	74



## 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;
```

```
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
```

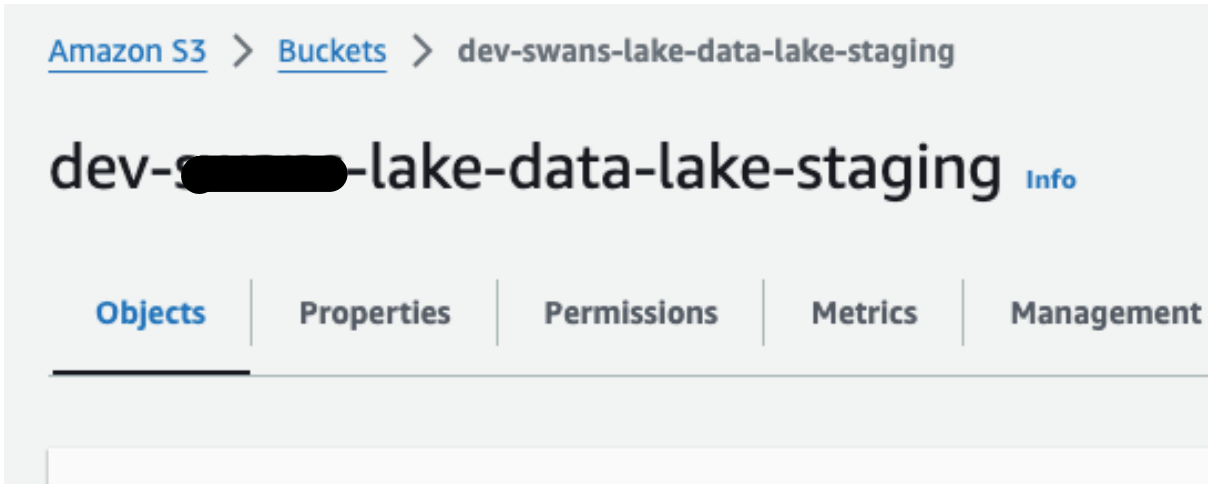
```
SELECT *  
FROM public.data_captures  
WHERE data_model_id in(71,2,3,4);
```

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

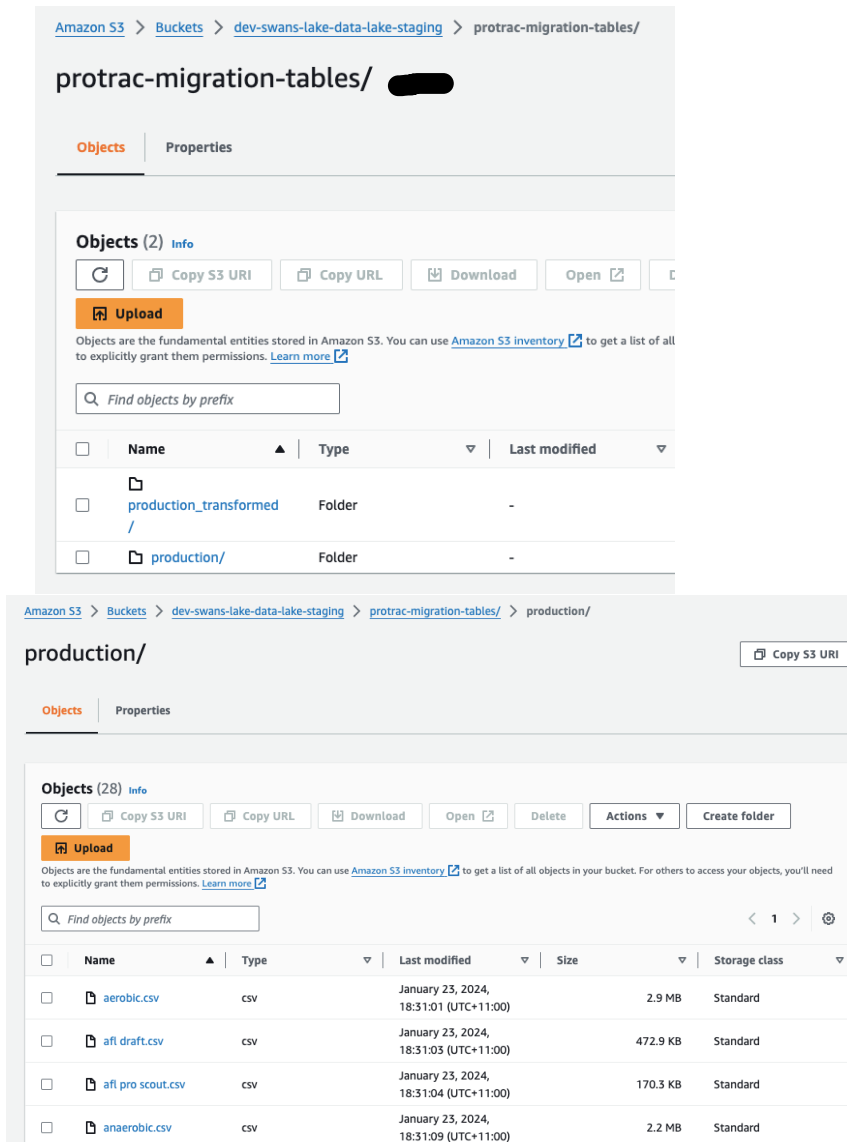
```
select *  
FROM public.attachments;
```

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

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

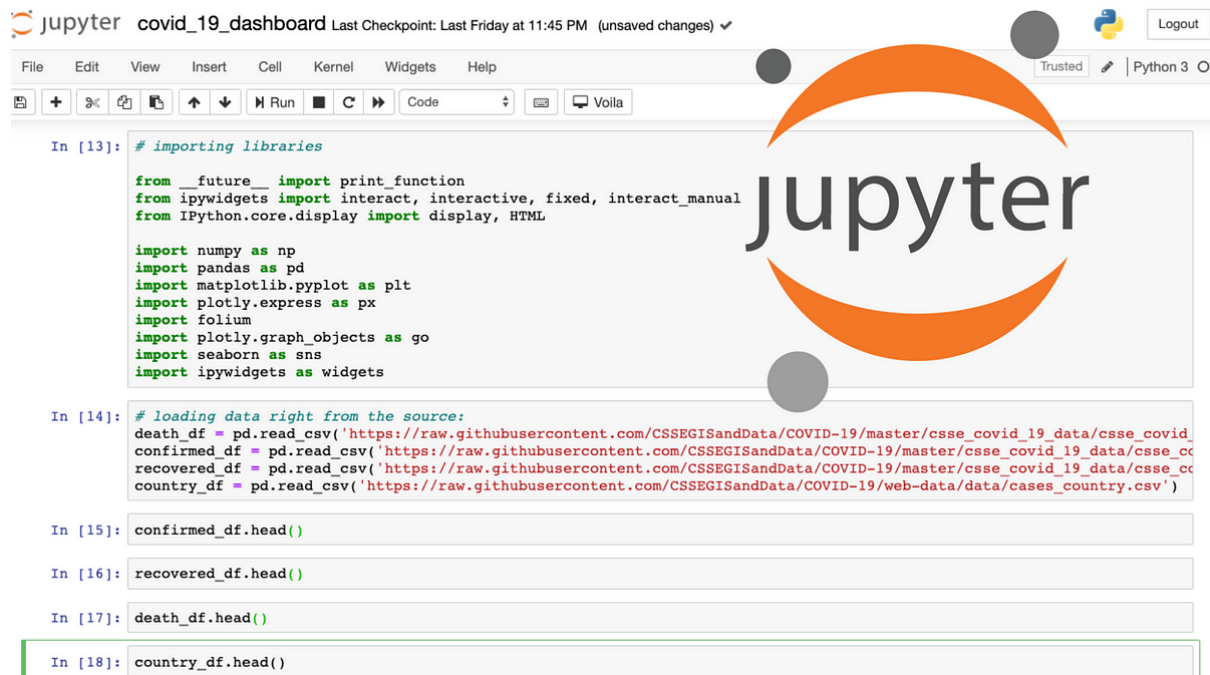


- Create a **production** folder & store all the raw production data tables (csv) into it



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

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



The screenshot shows a Jupyter Notebook window titled 'covid\_19\_dashboard'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running code, and viewing output. The notebook content consists of several code cells:

```
In [13]: # importing libraries

from __future__ import print_function
from ipywidgets import interact, interactive, fixed, interact_manual
from IPython.core.display import display, HTML

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import folium
import plotly.graph_objects as go
import seaborn as sns
import ipywidgets as widgets

In [14]: # loading data right from the source:
death_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_data/death_df.csv')
confirmed_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_data/confirmed_df.csv')
recovered_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_data/recovered_df.csv')
country_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/web-data/data/cases_country.csv')

In [15]: confirmed_df.head()

In [16]: recovered_df.head()

In [17]: death_df.head()

In [18]: country_df.head()
```

- 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:

- 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.

- 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

remotely 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

Amazon S3 > Buckets > dev-

-lake-data-lake-staging

 > [protrac-migration-tables/](#) > production\_transformed/

production\_transformed/

Copy S3 URI

Objects

Properties

Objects (17) Info

Refresh

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 > ⚙

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	<div><div></div><div><a href="#">aerobic_transformed.csv</a></div></div>	csv	January 29, 2024, 03:32:37 (UTC+11:00)	1.5 MB	Standard
<input type="checkbox"/>	<div><div></div><div><a href="#">afl draft_transformed.csv</a></div></div>	csv	January 29, 2024, 03:32:38 (UTC+11:00)	143.3 KB	Standard
<input type="checkbox"/>	<div><div></div><div><a href="#">afl pro scout_transformed.csv</a></div></div>	csv	January 29, 2024, 03:32:39 (UTC+11:00)	35.5 KB	Standard
<input type="checkbox"/>	<div><div></div><div><a href="#">anaerobic_transformed.</a></div></div>	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. 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	Weaknesses		

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

Step 1: Select your input

Enter Data

Choose File

Enter URL

Choose File

Choose file

No file chosen

Encoding

-Default-

Clear Input

Example 1

Example 2

- 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			NULL	is a very intelligent young man.	NULL	NULL	NULL	NULL	NULL	NULL
1	2008-04-11	J		3		5	4	5	3	NULL	NULL
2	2008-05-09			NULL	described himself as being concerned wit...	NULL	NULL	NULL	NULL	NULL	NULL
3	2008-07-03			NULL	Sounds a very articulate & focused young man o...	NULL	NULL	NULL	NULL	NULL	NULL
4	2008-11-05			4	Impressive young man.	3	4	4	4	CH TH ...	NULL

- Then, POST using POSTMAN:

Overview

POST sydney.smartabase.cc

New Collection

Runner

No Environment

sydney.smartabase.com/swans/api/v2/user/loginUser

Save

POST

sydney.smartabase.com/swans/api/v2/user/loginUser

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

9093

9094

9095

9096

9097

9098

9099

9100

9101

....."Draft Year": 2023,

....."Position": "Small/Med Def",

....."Strengths": " Good pace as a small rebounding defender Likes to run & carry the ball at pace Has a good step at

....."Weaknesses": " Lot's of unforced errors especially over head (drops marks & takes the ball to his chest far too

....."FIELD1":

....."First Name":

....."Last Name":

Body

Cookies (2)

Headers (18)

Test Results

Status: 200 OK

Time: 2.00 s

Size: 2.32 KB

Save as example

Name	Value	Domain	Path	Expires	HttpOnly	Secure
AWSALB	coSQZuU5Vv...	sy.smart...	/	Thu, 08 Feb 1...	false	false
AWSALB	coSQZuU5Vv...	y.smart...	/	Thu, 08 Feb 1...	false	false