#### Step 1: Connect to the Snowflake SQL

- Download SNOWSQL: https://developers.snowflake.com/snowsql/
- Open cmd
  - > snowsql -a xsymtcm-em56590 -u amritaneogi
  - > Password: \*\*\*\*\*\*\*\*\*\*

Here, account\_name = xsymtcm-em56590

login name = amritaneogi

# Step 2: Create a virtual Schema or Database

### Step 3: Create a virtual Data Warehouse; COMPUTE WH is the default warehouse provided.

Having a data warehouse in Snowflake is crucial, unlike other databases. The warehouse serves as the essential resource allocator for various computations, including running queries and performing SQL operations. This necessity arises from Snowflake's nature as a cloud-based data warehouse, where every operation consumes resources. The resources are used from the warehouse select, and based on this the cost is determined.

- > CREATE WAREHOUSE PRICE WH;
- amritaneogi#PRICE\_WH@HOUSE\_PRICE.PUBLIC >

# \*\* To reconnect-> Use Warehouse PRICE\_WH

**Use Database HOUSE PRICE;** 

### Step 4: Create a 'Stage'.

In other databases we load the file directly into the table. However, in Snowflake, we have to load it into a stage. A stage is nothing but a location which stores data files.

While creating stage we also need to mention the mention the format of the data files that it will store.

CREATE OR REPLACE FILE FORMAT CSV\_FORMAT TYPE = csv FIELD\_DELIMITER = ',' FIELD\_OPTIONALLY\_ENCLOSED\_BY=''''; Creating a new stage since we are uploading the data directly into stage. The data is in .csv file but the format is JSON.

- -- creating stage
- CREATE OR REPLACE STAGE OTODOM\_STAGE FILE FORMAT = CSV FORMAT;

## Step 5: Import file in the Stage.

We can directly export the file (either in .csv or JASON format) from the BrightData; but we need to purchase the file for that.

Instead, we will directly upload the file in the Snowflake 'Stage' that we created earlier.

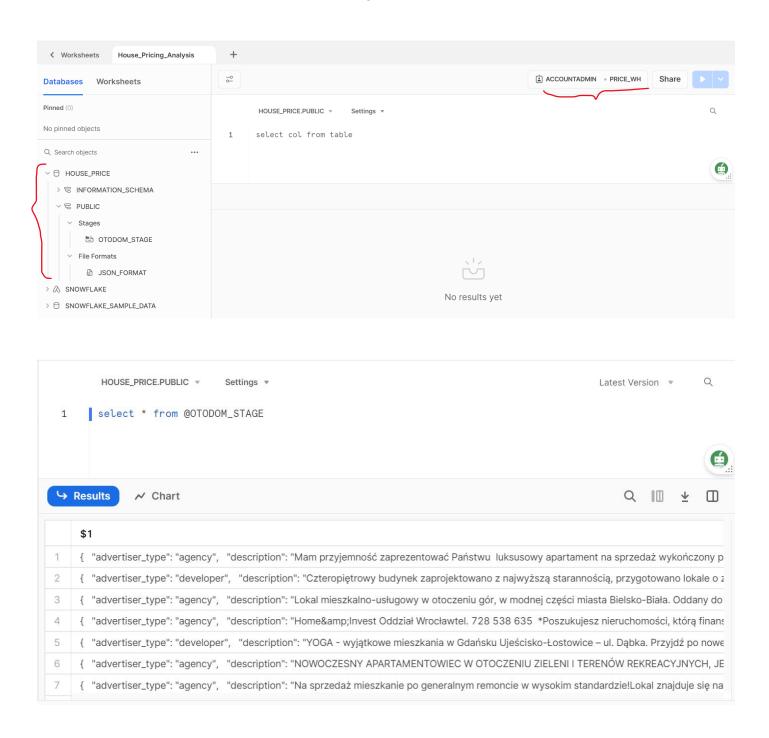
PUT file:///C:\Users\amrit\OneDrive\Documents\GitHub\Data\_Analytics\_Project-Hou sing Price Profiler\Data Set Backup\Otodom Poland.json @OTODOM STAGE;

While using the CSV\_FORMAT we will do it the following way:

- PUT file:///C:\Users\amrit\OneDrive\Documents\GitHub\Data\_Analytics\_Project-Hou sing\_Price\_Profiler\Data\_Set\_Backup\Otodom\_Apartment\_major\_cities\_dataset\_ORG\_JSON\_Format\_P art1.csv @OTODOM\_STAGE;
- PUT file:///C:\Users\amrit\OneDrive\Documents\GitHub\Data\_Analytics\_Project-Hou sing\_Price\_Profiler\Data\_Set\_Backup\Otodom\_Apartment\_major\_cities\_dataset\_ORG\_JSON\_Format\_P art2.csv @OTODOM\_STAGE;
- ➤ PUT file:///C:\Users\amrit\OneDrive\Documents\GitHub\Data\_Analytics\_Project-Hou sing\_Price\_Profiler\Data\_Set\_Backup\Otodom\_Apartment\_major\_cities\_dataset\_ORG\_JSON\_Format\_P art3.csv @OTODOM\_STAGE;

SON_Format_Part1.csv @OTO	Set_Backup\Otodom_Apartment_major_cities_dataset_ORG_J					
   Source   source_size   target_size   source_compression	target					
· 	Otodom_Apartment_major_cities_dataset_ORG_JSON_Format_Part1.csv.gz		13439728	NONE	GZIP	U
	Set_Backup\Otodom_Apartment_major_cities_dataset_ORG_J					
	target			•		+
· 	Otodom_Apartment_major_cities_dataset_ORG_JSON_Format_Part2.csv.gz	62710502	12115216	NONE	GZIP	u
	t\OneDrive\Documents\GitHub\Data_Analytics_Project-Hou set_Backup\Otodom_Apartment_major_cities_dataset_ORG_J	•		•		
	target	<b>:</b>				
 	Otodom_Apartment_major_cities_dataset_ORG_JSON_Format_Part3.csv.gz				GZIP	u
				,		

This is now the Snowflake server look once all the changes are made:



\*\* for the data we are using, it's too large to display

## Step 6: Once the data is loaded into the stage, create a table.

We are creating a table with only one column here 'json data'.

# writing codes on the snowflake console directly and not using cmd.

```
Create table OTODOM_DATA
(
json_data VARIANT -- creating table with one column
);
```

'VARIANT' → this is a special kind of data type. It is used to store semi-structured data from JSON file.

## **Step 7: Load data from stage to table.**

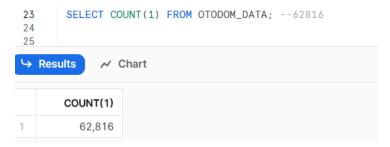
--copying data from stage to the table

copy into OTODOM\_DATA

from @OTODOM STAGE

on error='skip file'; -- skip any kind of error that could abandon the entire file.

**NOTE**: If the data is exported directly from BrightData, then it will have 2 extra files of one record each. This are the files that BrightData sends for the purpose of testing the connection.



### Step 8: Flattening of file.

The data we loaded is in JSON format, it is semi structured and not apt for any kind of analysis. So, we need to transform this into appropriate table columns. This process is called flattening of files.

Actual record (semi-structured):

{ "advertiser type": "agency", "description": "Chcesz mieszkać w ścisłym centrum Głogowa? A może poszukujesz nieruchomości, pod katem inwestycyjnym? Świetnie! Mam dla Ciebie idealne rozwiązanie. Mieszkanie zlokalizowane jest na Placu Jana z Głogowa, nieopodal skweru z fontanna oraz Parku Słowiańskiego i Parku Sapera, zatem w otoczeniu zieleni. W pobliżu odnajdziesz wszystko co jest niezbędne do funkcjonowania – sklepy spożywcze, piekarnię, cukiernię, punkty usługowe oraz gastronomiczne. Blok posiada dodatkową bramę wejściową zabezpieczona domofonem. Mieszkanie charakteryzuje się 36,5 m2 powierzchni użytkowej. Nieruchomość posadowiona jest na parterze pięcio-kondygnacyjnego budynku i składa się z: • Salonu z bardzo słonecznym, dużym oknem oraz balkonem, • Aneksu kuchennego, który jest oddzielony ścianką lecz bez problemu można go otworzyć na salon uzyskując większą przestrzeń, Sypialni • Łazienki, wyposażonej w wannę, umywalkę oraz WC, • Przedpokoju. Mieszkanie jest utrzymane w bardzo dobrej kondycji, stolarka okienna pcv biała. Ściany zostały odświeżone, a nieruchomość nadaje się do zamieszkania. Jest to również nieruchomość gotowa pod inwestycję, bowiem całe wyposażenie pozostaje do dyspozycji nowego nabywcy. Ogrzewanie miejskie, mieszkanie z zasobów Spółdzielni Mieszkaniowej. Do nieruchomości przynależy również piwnica. Chcesz dowiedzieć się więcej? Zadzwoń i umów się na bezpłatną prezentację nieruchomości. Istnieje możliwość negocjacji ceny. Biuro Nieruchomości CUPRUM HOUSE Anna Wernik – Kaniewska Specjalista ds. nieruchomości tel. 723 411 712 \r\n\r\nOferta wysłana z programu dla biur nieruchomości ASARI CRM ()\r\n", " ": "pełna własność", "is for sale": true, "lighting": null, "location": "Głogów, głogowski, dolnośląskie", "market": [ "secondary" ], "no\_of\_rooms": 2, "posting\_id": "4mSyJ", "price": "210000", "remote\_support": null, "surface": "36.5", "timestamp": "2023-09-20", "title": "Dwa pokoje w ścisłym centrum do zamieszkania", "url": "https://otodom.pl/pl/oferta/dwa-pokoje-w-scislym-centrum-do-zamieszkania-ID4mSyJ" }

## For example:

```
TO
        select JSON_DATA: price
17
        from OTODOM_DATA;
18
19
→ Results

✓ Chart

    JSON_DATA: PRICE
    "731000"
1
2
    "506600"
3
    "452000"
    "660000"
4
    "629000"
5
6
    "434175"
7
    "440000"
```

# SELECT JSON\_DATA:price

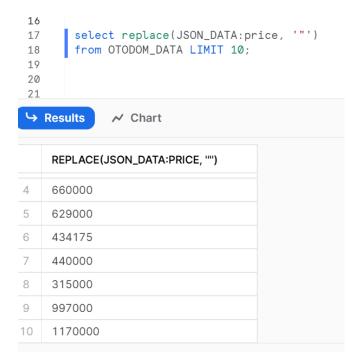
## FROM OTODOM DATAt limit 5;

This might throw error since the field is text but has JSON format data hence need to use PARSE\_JSON function to parse the JSON data in this field as shown in above query:

# SELECT PARSE JSON(json data):price

# FROM OTODOM DATA limit 5;

Remove the double quotes:



We will do the same for all other columns:

```
CREATE OR REPLACE table OTODOM DATA FLATTEN
select row number() over(order by title) as rn
, x.*
from (
select replace(parse json(json data):advertiser type,"")::string as advertiser type
, replace(parse json(json data):balcony garden terrace,"")::string as balcony garden terrace
, regexp replace(replace(parse json(json data):description,'"'), '<[^>]+>')::string as description
, replace(parse json(json data):heating,"")::string as heating
, replace(parse json(json data):is for sale,"")::string as is for sale
, replace(parse json(json data):lighting,"")::string as lighting
, replace(parse json(json data):location,"")::string as location
, replace(parse json(json data):price,"")::string as price
, replace(parse json(json data):remote support,"")::string as remote support
, replace(parse json(json data):rent sale,"")::string as rent sale
, replace(parse json(json data):surface,"")::string as surface
, replace(parse json(json data):timestamp,"")::date as timestamp
, replace(parse json(json data):title,"")::string as title
, replace(parse json(json data):url,"")::string as url
, replace(parse json(json data):form of property,"")::string as form of property
, replace(parse json(json data):no of rooms,"")::string as no of rooms
, replace(parse json(json data):parking space,"")::string as parking space
from OTODOM DATA
) x;
```

#### Note:

The code uses regexp\_replace to remove HTML tags from the "description" column. This is done by replacing anything that matches the regular expression  $<[^>]+>'$  (which matches HTML tags) with an empty string.

## Step 9: Transformation of data for further analysis

- 1. Locations contains longitudinal and latitudinal values
- 2. Change the language from Polish to English

We will use Python and Google sheet for the same.

## **Step 9.1: Create Python Virtual Environment**

Open Terminal from Jupyter Notebook

conda create --name data\_analytics\_project python=3.11
 All the required packages are installed with it at the same time

Changes done specific to my code (ran code on Jupyter Notebook):

- > conda active data analytics project
- > conda install -n data analytics project jupyter
- > jupyter notebook Open a new Jupyter notebook

select \* from OTODOM\_DATA\_FLATTEN\_ADDRESS;

52 202721725827894 21 118257425902224

- > conda install -n data analytics project -c conda-forge snowflake-sqlalchemy -
- !pip install snowflake-sqlalchemy

64

65

- ➤ # import all the packages and libraries (refer the .ipynb file)
- Execute the Python code → Transform into proper Longitude and Latitude coordinates → Convert between addresses and geographic coordinates using GOOGLE GEOCODE API → Add new ADDRESS column to the original dataset with proper address → Create a new Table "OTODOM\_DATA\_FLATTEN\_AADRESS" and insert proper address to their corresponding geographic coordinates.

From the newly created Jupyter

```
66
67
→ Results

✓ Chart

                LOCATION
                                                               ADDRESS
            1
                52.23614,21.00817
                                                               Marszałkowska 138, 00-004 Warszawa, Poland
            2
                52.336575,21.029306
                                                               DW633 94, 03-044 Warszawa, Poland
            3
                51.10710682881388,16.94346882507325
                                                               Graniczna 2aa, 54-516 Wrocław, Poland
            4
                50.10361,20.00665
                                                               Osiedle Bohaterów Września 82P, 31-620 Kraków, Poland
                                                               DW633 94, 03-044 Warszawa, Poland
            5
                52.336575.21.029306
            6
                52.336575,21.029306
                                                               DW633 94, 03-044 Warszawa, Poland
            7
                52.2044154,20.8805653
                                                               Posag 7 Panien 16, 02-495 Warszawa, Poland
            8
                50.29034,19.00576
                                                               Bytkowska 1, 40-147 Katowice, Poland
            9
                50.29034.19.00576
                                                               Bytkowska 1, 40-147 Katowice, Poland
                50.29034,19.00576
           10
                                                               Bytkowska 1, 40-147 Katowice, Poland
           11
                52.19622,21.17823
                                                               ul. Nenufarów 8, 04-958, 04-701 Warszawa, Poland
```

Wał Miedzeszyński 472M 03-994 Warszawa Poland

We have more than 300K records, so it was not possible to compute the code for that many rows. So ran codes in different chunks and uploaded the result in the form of csv to a new table OTODOM\_DATA\_FLATTEN\_ADDRESS\_FULL with the same columns as in OTODOM\_DATA\_FLATTEN\_ADDRESS.

Create table

```
CREATE TABLE OTODOM_DATA_FLATTEN_ADDRESS_FULL

(
ID int
LOCATION text
ADDRESS text )
)
```

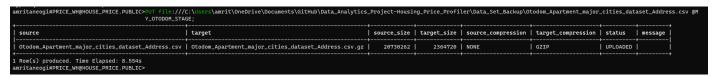
> Create the File Format and stage

```
CREATE OR REPLACE FILE FORMAT CSV_FORMAT TYPE = csv
FIELD_DELIMITER = ','
FIELD_OPTIONALLY_ENCLOSED_BY='''';

CREATE OR REPLACE STAGE MY_OTODOM_STAGE FILE_FORMAT = CSV_FORMAT;
```

Upload data (.csv file) from path to stage

PUT file:///C:\Users\amrit\OneDrive\Documents\GitHub\Data\_Analytics\_Project-Housing\_Price\_Profiler\Data\_Set\_Backup\Otodom\_Apartment\_major\_cities\_dataset\_Address.csv @M Y OTODOM STAGE;



> Copy data from stage to table

COPY INTO OTODOM\_DATA\_FLATTEN\_ADDRESS\_FULL FROM @MY\_OTODOM\_STAGE;

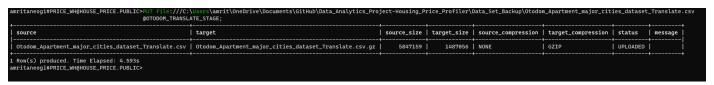
### **Step 9.2: Translate the address to English**

- -- Translatet the Address title from Polish to English using Google Sheet
- -- Refer Python scripts for the entire translation

```
select * from otodom_data_flatten_translate;
select count(1) from otodom_data_flatten_translate; -- sample 300 records
```

To translate the entire dataset we will directly upload .csv into the snowflake table

→ Follow the same process of creating stage and copy data from stage to table



### > Summary

```
select * from OTODOM_DATA_FLATTEN; --All data
select * from OTODOM_DATA_FLATTEN_ADDRESS_FULL; -- All geocode converted to
address/locations
select * from OTODOM_DATA_FLATTEN_TRANSLATE_FULL; --All translated Address data
```

## ANALYSIS AND REPORT BUILDING

## Step 10: Join all three tables

Do basic transformation to:

- extract 'suburbs', 'city' and 'country' from the address
- remove 'PLN' and '€' from the Price
- convert Euros (€) to Polish Złoty by multiplying with 4.43

CREATE OR REPLACE TABLE OTODOM DATA TRANSFORMED

- remove m<sup>2</sup> and M<sup>2</sup> from the Surface column
- OTODOM site has a number of for flats, garage, etc., however, we are only interested in flats. So, create a new flag for 'apartment' and 'non apartment'

# with cte as (select ot.\* , case when price like 'PLN%' then try to number(replace(price, 'PLN', '), '999, 999, 999) when price like '€%' then try to number(replace(price,'€',"),'999,999,999,999) \* 4.43 end as price new , try to double(replace(replace(replace(surface, 'm2', "), 'M2', "), ', ", "), ', ', ', '), '9999.99') as surface new , replace(parse json(addr.address):suburb,"", ") as suburb , replace(parse json(addr.address):city,"", ") as city , replace(parse json(addr.address):country,"", ") as country , trans.title eng as title eng from otodom data flatten ot left join otodom data flatten address full addr on ot.ID=addr.ID left join otodom data flatten translate full trans on ot.ID=trans.ID) select \* , case when lower(title eng) like '%commercial%' or lower(title eng) like '%office%' or lower(title eng) like '%shop%' then 'non apartment'

when is for sale = 'false' and surface new <=330 and price new <=55000 then 'apartment'

when is for sale = 'true' and surface new <=600 and price new <=20000000 then 'apartment'

when is for sale = 'false' then 'non apartment'

```
when is_for_sale = 'true' then 'non apartment'
end as apartment_flag
from cte;
```

Refer the 'Analysis.pdf' file for the problems and solutions.