ENHANCING DATASET WITH API-DRIVEN ADDRESS ENRICHMENT

TEAM -7

TEAM 7

Rishika Lall

Vaishali Patidar

Mirudula Muralidhar

Govardhan Gattu

Manish Bhatia

INTRODUCTION

Objective: Overview of data download and cleaning process

Tools: Precisely Data Integrity Suite, Postman, Python

Key Focus: Data separation (residential, commercial, mixed-use) and cleaning

3 WAYS DATA ENRICHMENT WORKS FOR TELCO

1. Building the optimal network :
   - Geospatial intelligence provides unique opportunities
   - Better understand where customers and potential customers are located
   - Improve network coverage to better serve those audiences.

2. Knowing your customers
   - Knowing your customer is a challenge
   - Multiple service options, professional roles, and additional family members to that equation, things can get complicated very quickly.

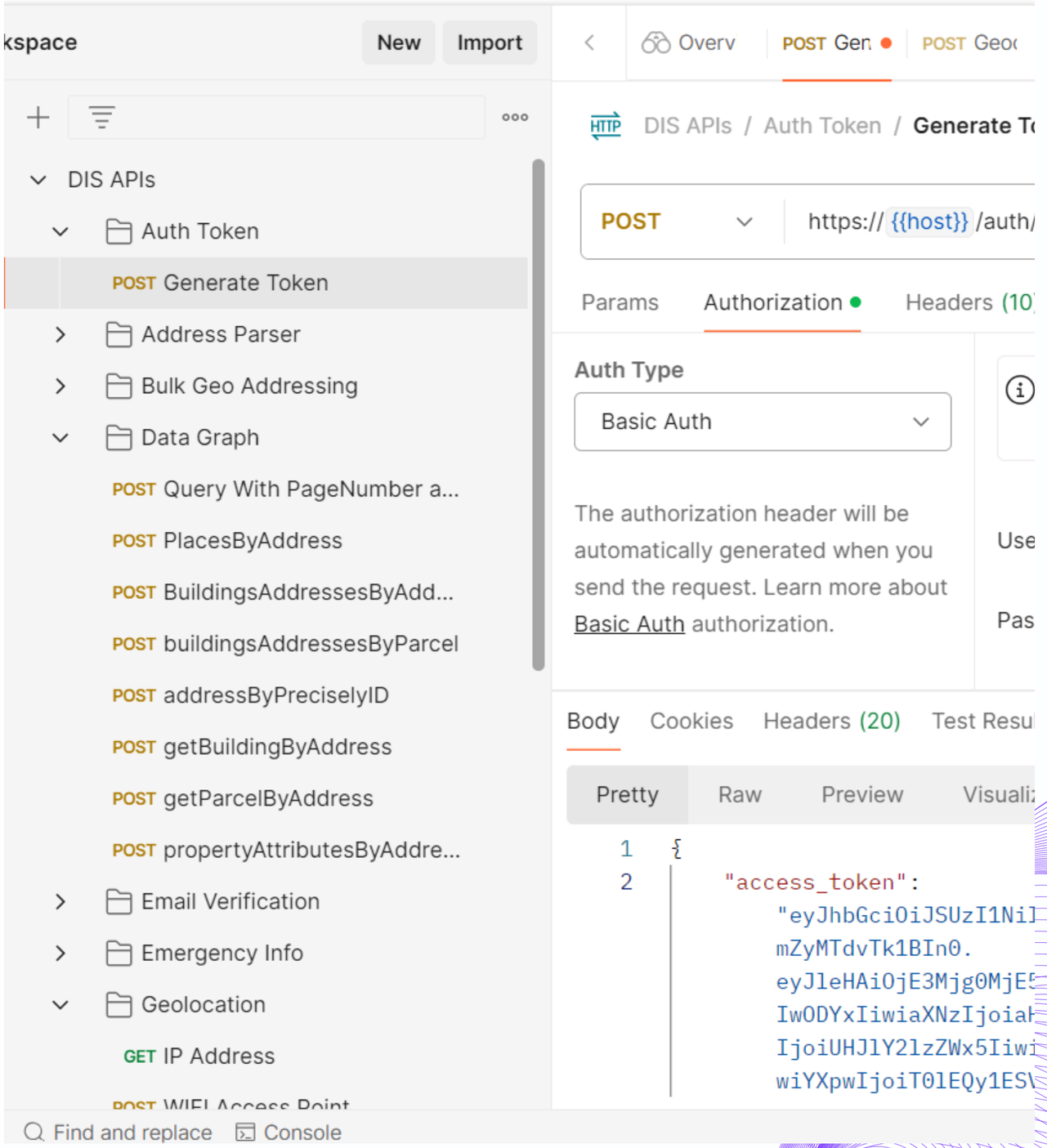
3. Optimizing efficiency
   - Efficiency and effectiveness of telecom resources.
   - Use cases may include predictive analytics to determine where problems are likely to occur

POSTMAN

- Widely used API testing tool
- Allows us to send API requests (GET, POST, PUT, and DELETE)
- Interact with APIs by sending data, retrieving information

FETCHING APIs

- Enter the API's URL.
- Select the HTTP method.
- Provide parameters or headers.
- Hit 'Send' to view the response.



PRECISELY'S API SUITE

PlacesbyAddress

PreciselyID, AddressNumber

AddressBypreciselyID

City, latitude, longitude, PostalCode

BuildingsAddressbyAddress

StreetNumber, Unit

getBuidingByAddress

BuildingID, Buildingtype

DATA DOWNLOADING

API:

propertyAttributesByAddress

"https://api.cloud.precisely.com/data-graph/graphql"

query type: getByAddress

- Precisely ID
- Living Square Footage
- Sale Amount

```
def fetch_property_data(row):
    address = f"{row['NUMBER']} {row['STREET']}, {row['CITY']} PA".strip()
    payload = f"""
    {{
      "query": "query propertyAttributesByAddress {{\\n  getByAddress(address: \\"{address}\\")
        {{\\n    propertyAttributes {{\\n      data {{\\n        preciselyID\\n        livingSquareFootage\\n        saleAmount\\n      }}\\n    }}\\n  }}\\n}}",
      "variables": {{}}
    }}
    """
```

API:

getBuildingByAddress

query type: getByAddress

Building Area

Building ID

Building Type

Elevation

```
def fetch_building_data(row):
    address = f"{row['NUMBER']} {row['STREET']}, {row['CITY']} PA".strip()
    payload = f"""
    {{
      "query": "query getBuildingByAddress {{\\n  getByAddress(address: \\"{address}\\")
        {{\\n    buildings {{\\n      data {{\\n        buildingID\\n        buildingType {{\\n          description\\n        }}
        \\n        elevation\\n        maximumElevation\\n        minimumElevation\\n        buildingArea\\n      }}\\n    }}\\n  }}\\n}}",
      "variables": {{}}
    }}
    """
```

DATA OVERVIEW

New Data

	LAT	NUMBER	STREET	UNIT	CITY	DISTRICT	REGION	POSTCODE	ID	HASH	preciselyID	livingSquareFootage	saleAmount	buildingID	buildingType	elevation	maximumElevation	minimumElevation
	39.969039	4102	OGDEN ST	AB	Philadelphia	University City	West Philadelphia	19104	886740095	6fe85ffddef02977	NaN	NaN	NaN	B000CTRV1G0R	Residential	113.0	114.0	112.0
	39.956518	4215	CHESTNUT ST	A	Philadelphia	University City	West Philadelphia	19104	272012837	a9767341c0cbe589	P0000JCD8L2K	NaN	NaN	B000CTQ2IAN0	Mixed	76.0	87.0	73.0
	39.956518	4215	CHESTNUT ST	B	Philadelphia	University City	West Philadelphia	19104	272012839	46e6a17201df7dd2	P0000JCD8L2K	NaN	NaN	B000CTQ2IAN0	Mixed	76.0	87.0	73.0
	39.956518	4215	CHESTNUT ST	C	Philadelphia	University City	West Philadelphia	19104	272012841	a166854dac25162f	P0000JCD8L2K	NaN	NaN	B000CTQ2IAN0	Mixed	76.0	87.0	73.0
	39.956518	4215	CHESTNUT ST	D	Philadelphia	University City	West Philadelphia	19104	272012843	d916601b5ac1a0d4	P0000JCD8L2K	NaN	NaN	B000CTQ2IAN0	Mixed	76.0	87.0	73.0

New Columns Acquired

```
df.columns
```

✓ 0.0s

```
Index(['LON', 'LAT', 'NUMBER', 'STREET', 'UNIT', 'CITY', 'DISTRICT', 'REGION',
       'POSTCODE', 'ID', 'HASH', 'preciselyID', 'livingSquareFootage',
       'saleAmount', 'buildingID', 'buildingType', 'elevation',
       'maximumElevation', 'minimumElevation', 'buildingArea'],
      dtype='object')
```

DATA OVERVIEW

Data Description

```
df.describe()
```

✓ 0.0s

	LON	LAT	POSTCODE	ID	livingSquareFootage	saleAmount	elevation	maximumElevation	minimumElevation	buildingArea
count	11947.000000	11947.000000	11947.0	1.194700e+04	8.680000e+03	7.970000e+03	9694.000000	9694.000000	9694.000000	9.694000e+03
mean	-75.202920	39.963826	19104.0	3.291068e+08	5.602510e+03	3.212985e+05	96.447081	97.746441	94.851661	5.251468e+03
std	0.006509	0.006959	0.0	2.990271e+08	6.441663e+04	1.626680e+06	16.698386	15.965979	17.459439	5.382492e+04
min	-75.220497	39.943492	19104.0	6.100400e+07	7.200000e+01	1.000000e+00	13.000000	13.000000	-18.000000	1.550000e+02
25%	-75.208272	39.959912	19104.0	6.225325e+07	1.194000e+03	1.510000e+04	87.000000	88.000000	85.000000	7.440000e+02
50%	-75.203641	39.964653	19104.0	2.422315e+08	1.766000e+03	7.100000e+04	99.000000	100.000000	98.000000	9.700000e+02
75%	-75.198729	39.968458	19104.0	2.721176e+08	2.596500e+03	2.448975e+05	109.000000	110.000000	108.000000	1.401000e+03
max	-75.181492	39.977401	19104.0	8.882702e+08	4.560000e+06	8.800000e+07	141.000000	141.000000	140.000000	1.423610e+06

Note: snapshot includes on-surface statistical overview of the data

DATA CLEANING

CHECKING ON MISSING VALUES

Columns like 'CITY', 'DISTRICT', and 'REGION' contain missing values.
Decide whether to fill these with appropriate default values or drop these rows if they're not essential for analysis.

CHECK FOR DUPLICATES

Identify and remove duplicate rows, especially where 'LAT', 'LON', 'NUMBER', 'STREET', and 'POSTCODE' values are repeated.

HANDLING MISSING DATA IN COLUMNS

Standardize the 'POSTCODE' field to ensure all entries follow the correct postal format.
Verify the consistency of 'NUMBER' and 'UNIT' values

FORMAT DATA

Ensure that 'NUMBER' is an integer and 'POSTCODE' is treated as a string.
Format column names uniformly to maintain consistency across the dataset.

THANK YOU