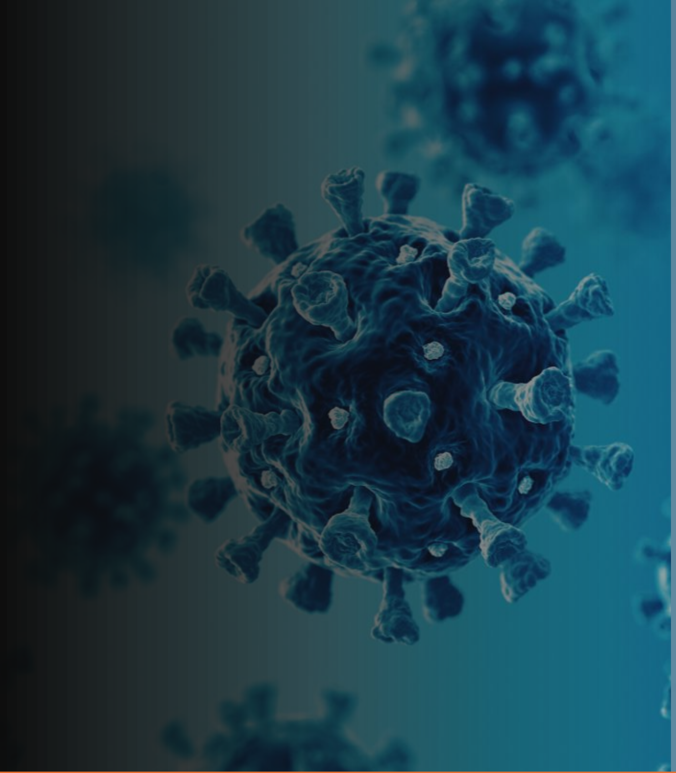
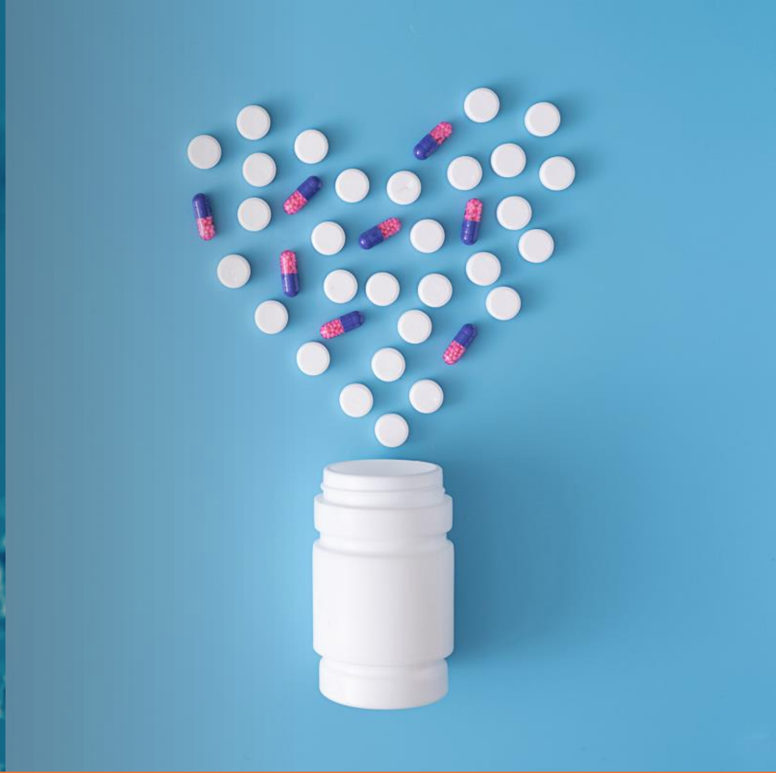
World Health Organization

Disease Tracker for Decision Making





Team

- Nandini Reddy Basupally
- Ashish Yakub Beary
- Aswin Karthik Panneer Selvam

Project Introduction

Objective

- Aid our client, the World Health Organization (WHO) to leverage data-driven insights to effectively monitor, analyze, and respond to global disease outbreaks.
- Make the solution as an accelerator to deploy it for other government-based and not for profit health management boards across the globe.

Use-Cases and Decisions

Use-Cases	Potential Decisions
Disease Outbreak Identification and Tracking	Deploy rapid response teams to affected areas, implement containment measures, and allocate resources effectively
Disease Severity Assessment and Prioritization	Prioritize resource allocation to diseases with increasing severity, develop targeted interventions, and allocate funds for research and development
Identifying Vulnerable Populations	Develop targeted public health campaigns, allocate resources to vulnerable communities, and implement preventive measures
Evaluating Medicine Effectiveness and Manufacturer Performance	Make informed decisions about drug procurement, recommend effective treatments, and establish partnerships with reliable manufacturers.
Optimizing Resource Allocation	Deploy medical personnel and equipment to areas with high disease burden, prioritize research efforts, and establish surveillance systems

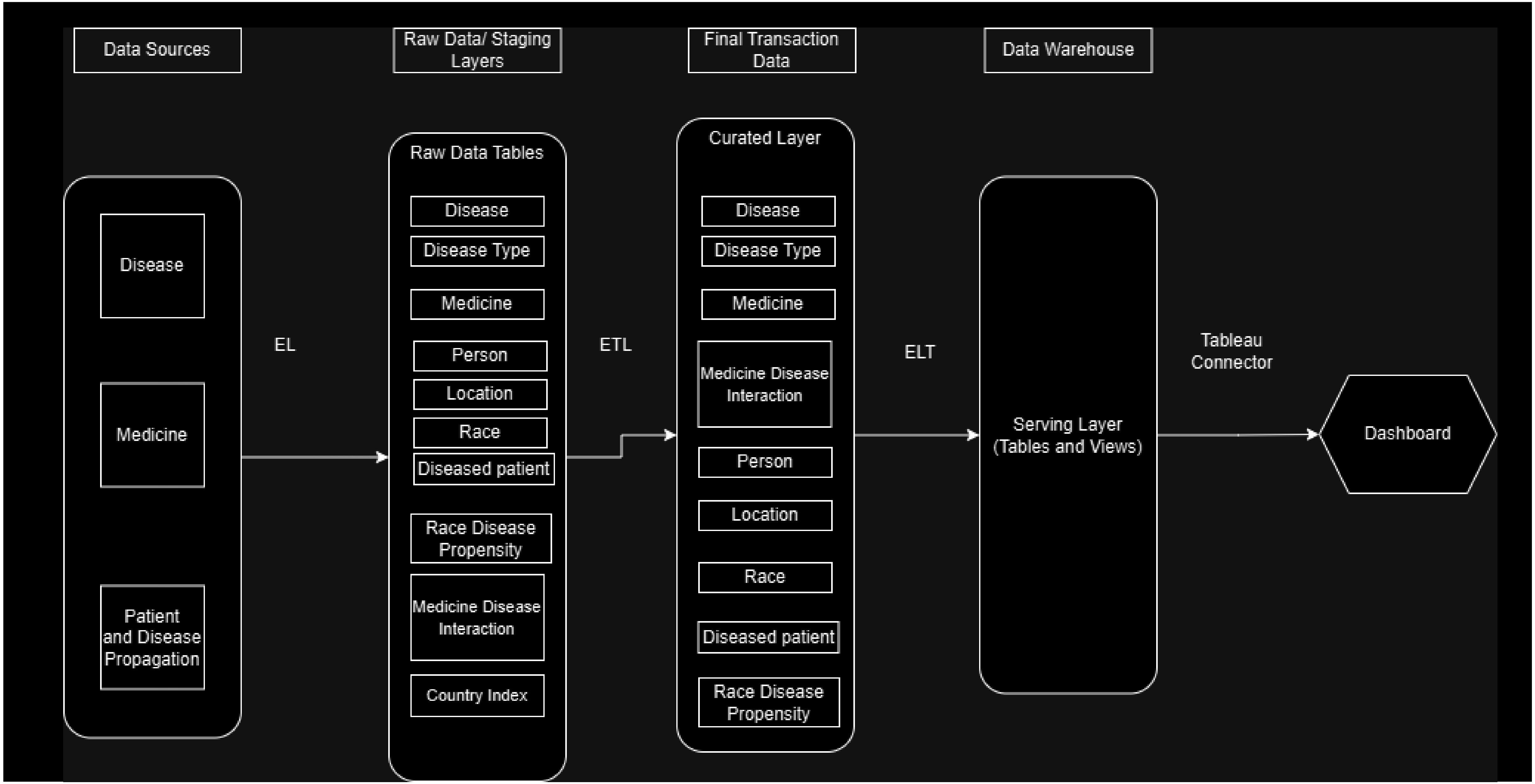
Business Problem

The World Health Organization (WHO) faces the persistent challenge of effectively monitoring and responding to global disease outbreaks. Traditional methods of disease surveillance often rely on manual data collection and analysis, leading to delays in identifying emerging threats and implementing timely interventions. Moreover, the complex interplay of factors such as disease severity, geographical distribution, and demographic characteristics necessitates a comprehensive and data-driven approach to understand and address global health challenges.

To overcome these limitations, WHO requires a robust and scalable data analytics solution that can efficiently process and analyze large volumes of diverse health data. This solution should enable the organization to identify disease outbreaks early, track their spread, assess their impact on vulnerable populations, and evaluate the effectiveness of interventions. By harnessing the power of data, WHO can make informed decisions, allocate resources strategically, and ultimately improve global health outcomes.

This solution can be extended to any government-based health management board or NGOs across the globe. Our motivation is the case of Palantir and how its platform was used to provide real-time COVID-19 insights to various European countries during the pandemic.

Flow Architecture



Data Sources and Staging Layer – Overview (1/2)

Schema	raw_data_staging_layer

Source	Tables	Description
Healthcare providers	Disease_Type	Metadata about different disease types, including their codes, names, and detailed descriptions.
Healthcare providers	Disease	Information about individual diseases, including their identifiers, names, intensity levels, and references to their associated disease type or source diseases.
Pharmaceutical Companies	Medicine	Details of medicines, including their names, active ingredients, industry standard numbers, and the companies that manufacture them.
Pharmaceutical Companies	Medicine_Disease_Interaction	Records the interaction between medicines and diseases, with details on effectiveness and the availability of medicines for treating specific diseases.
Healthcare providers	Location	Geographical information about cities, states, and countries, providing context for patient or disease data.
Global UN Reports	Country_index	Details about countries, including their identifiers, wealth rankings, and developmental status.
Global UN Reports	Race	Includes information about races, with unique codes and descriptions to categorize individuals demographically.
Healthcare providers	Person	Represents individuals in the system, storing personal details like name, gender, race, and location information.
Healthcare providers	Diseased_Patient	Tracks patients diagnosed with diseases, including details about the disease, severity, and the duration of the illness.
WHO internal Reports	Race_Disease_Propensity	Captures data on the propensity of specific races to be affected by particular disease, based on internal reports or studies.

Data Sources and Staging Layer – Overview (2/2)

Refer to **raw_data_staging_layer_ER_Diagram** PDF file for more information

Data Sources and Staging Layer – ETL Process

- Python based automated ETL Pipeline – Reusable and Extendable
- Can be used to connect to varied data sources and streaming services easily
- Can be written as Spark pipeline to handle large data
- Not a lot of restriction on the schema and connectedness of the data
- We will take a closer look into the Python file – **raw_data_load.py**

Snippet of python function to load patients' data from the source system

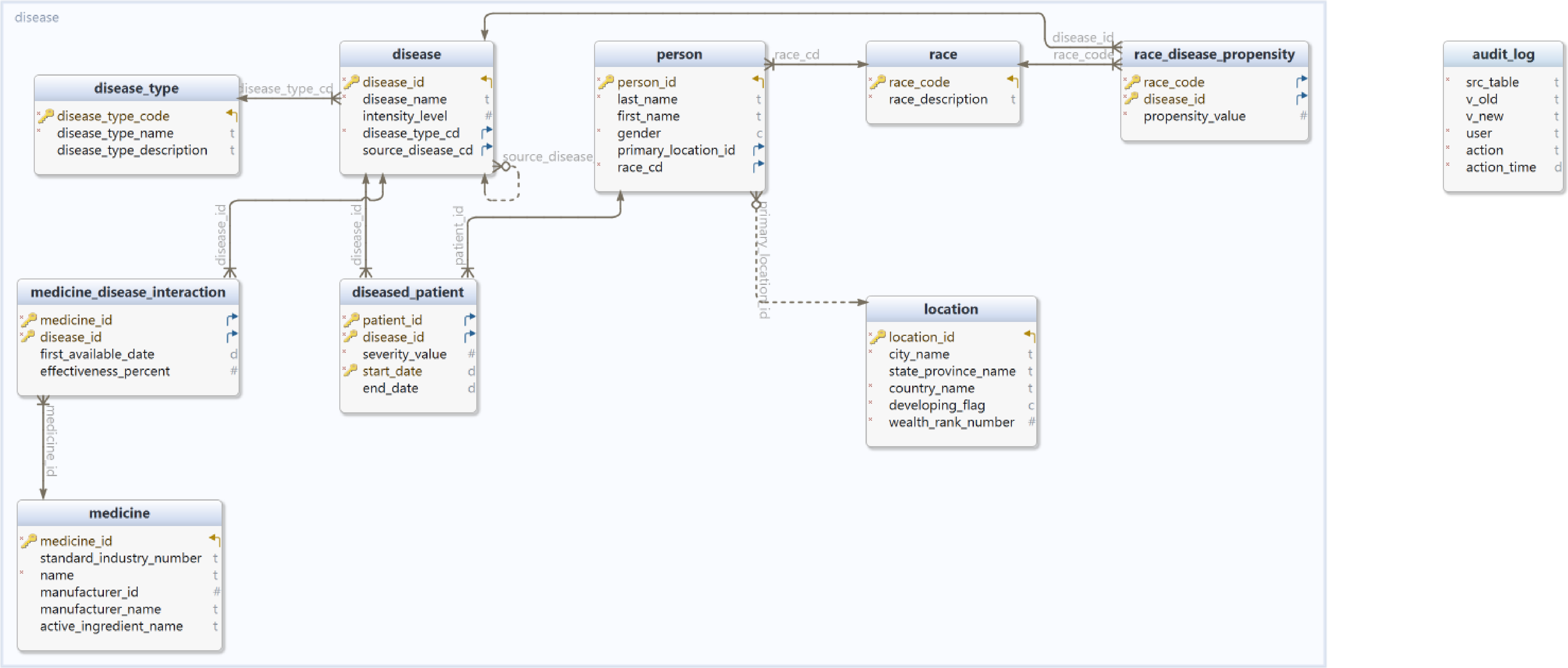
```python
def insert_patients_to_postgres(patient_df):
    try:
        patient_df['race_cd'] = patient_df['race_cd'].astype(str).str.zfill(2)

        conn = psycopg2.connect(**DB_CONFIG)
        cur = conn.cursor()

        patient_data = patient_df[['person_id', 'last_name', 'first_name', 'gender', 'race_cd', 'primary_location_id']]
        patient_data.columns = ['person_id', 'last_name', 'first_name', 'gender', 'race_code', 'location_id']
        print("Inserting Values into Person Table")
        for _, row in patient_data.iterrows():
            cur.execute(
                """
                INSERT INTO raw_data_staging_layer.person (person_id, last_name, first_name, gender, race_code, location_id)
                VALUES (%s, %s, %s, %s, %s, %s)
                ON CONFLICT (person_id) DO NOTHING;
                """,
                (row["person_id"], row["last_name"], row["first_name"], row["gender"], row["race_code"], row["location_id"]),
            )
            # print(f'inserted {_+1} record(s)')

        conn.commit()
        print("Data inserted successfully!")
    except Exception as e:
        print("Error:", e)
    finally:
        cur.close()
        conn.close()
```

Curated Transaction Layer – Overview (1/2)

Schema	curated_layer

Tables	Description
Disease_Type	Metadata about different disease types, including their codes, names, and detailed descriptions.
Disease	Information about individual diseases, including their identifiers, names, intensity levels, and references to their associated disease type or source diseases.
Medicine	Details of medicines, including their names, active ingredients, industry standard numbers, and the companies that manufacture them with company identifier.
Medicine_Disease_Interaction	Records the interaction between medicines and diseases, with details on effectiveness and the availability of medicines for treating specific diseases. Deleting a medicine also removes associated interactions.
Location	Geographical information about cities, states, and countries. The Developing_Flag and Wealth_Rank_Number provide economic context.
Race	Includes information about races, with unique codes and descriptions to categorize individuals demographically.
Person	Represents individuals in the system, storing personal details like name, gender, race, and location information.
Diseased_Patient	Tracks patients diagnosed with diseases, including details about the disease, severity, and the duration of the illness. Removing a person also deletes their disease data.
Race_Disease_Propensity	Captures data on the propensity of specific races to be affected by particular disease, based on internal reports or studies. Updates in Race_Code reflect here
Audit_Log	Logs changes, providing traceability for data updates.

Curated Transaction Layer – Overview (2/2)



Refer to **Curated_Layer_ER_Diagram** PDF file for more information

Curated Transaction Layer – ETL Process

- Python based automated ETL Pipeline – Reusable and Extendable
- Classes are used to update or extend the logic without affecting the existing code. It also makes sure that the minimal pipeline template is followed.
- Transformation layer and verify and clean layer are the key components. They make sure all the schema constraints are followed.
- Advanced concepts like Named Entity Recognition to maintain sanctity of data while curating from different sources can also be implemented
- Can be written as Spark pipeline to handle large data
- We will take a closer look into the Python file – **curated_data_load.py**

Key Pipeline Steps

Curated Layer Insertion Pipeline

Extract from Raw Data → **Transform Raw Data** → **Verify and Clean** → **Load into Curated Transaction Layer**

Curated Transaction Layer DML Illustrations

Case 1: The intensity level of a specific disease has been re-evaluated and needs to be updated

```
--The intensity level of a specific disease(i.e Anxiety Disorders) has been re-evaluated and needs to be updated.

--Data Before Update
select * from curated_layer.Disease
WHERE Disease_ID = 7;
```

Data Output | Messages | Notifications

	disease_id [PK] integer	disease_name character varying (100)	intensity_level integer	disease_type_cd character (10)	source_disease_cd integer
1	7	Anxiety Disorders	6	OTHER	[null]

```
--Data After Update
UPDATE curated_layer.Disease
SET Intensity_Level = 8  -- New intensity level
WHERE Disease_ID = 7;
select * from curated_layer.Disease WHERE Disease_ID = 7;
```

	disease_id [PK] integer	disease_name character varying (100)	intensity_level integer	disease_type_cd character (10)	source_disease_cd integer
1	7	Anxiety Disorders	8	OTHER	[null]

Case 2: A person has moved to a new city, so their Primary_ID needs to be updated.

```
--2. A person has moved to a new city, so their Primary_Location_ID needs to be updated.

-- Data Before Update
SELECT * FROM curated_layer.Person WHERE Person_ID = 145;
```

	person_id [PK] integer	last_name character varying (50)	first_name character varying (50)	gender character (1)	primary_location_id integer	race_cd character varying (5)
1	145	Berry	Victoria	M	32014	BLK

```
-- Data After Update
UPDATE curated_layer.Person
SET Primary_Location_ID = 30938  -- New location ID
WHERE Person_ID = 145;
SELECT * FROM curated_layer.Person WHERE Person_ID = 145;
```

	person_id [PK] integer	last_name character varying (50)	first_name character varying (50)	gender character (1)	primary_location_id integer	race_cd character varying (5)
1	145	Berry	Victoria	M	30938	BLK

Curated Transaction Layer DML Illustrations

Case 3: The manufacturer of a medicine changed, and the manufacturer_name needs to be updated

```
-- 3. The manufacturer of a medicine changed, and the Manufacturer_Name needs to be updated accordingly.

-- Data Before Update
SELECT * FROM curated_layer.Medicine WHERE Medicine_ID = 101;

--Data After Update
UPDATE curated_layer.Medicine
SET Manufacturer_Name = 'Novartis' -- New Manufacturer_Name
WHERE Medicine_ID = 101;
SELECT * FROM curated_layer.Medicine WHERE Medicine_ID = 101;
```

	medicine_id [PK] integer	standard_industry_number character varying (25)	name character varying (250)	manufacturer_id integer	manufacturer_name character varying (150)	active_ingredient_name character varying (150)
1	101	0004-0112-50	Rifadin	30	Sanofi	Rifampin

	medicine_id [PK] integer	standard_industry_number character varying (25)	name character varying (250)	manufacturer_id integer	manufacturer_name character varying (150)	active_ingredient_name character varying (150)
1	101	0004-0112-50	Rifadin	30	Novartis	Rifampin

Case 4: A person is no longer part of the system and needs to be removed.

```
-- Data Before Delete
SELECT * FROM curated_layer.Person Where Person_ID = 888;
-- Data After Update
DELETE FROM curated_layer.Person
WHERE Person_ID = 888;
SELECT * FROM curated_layer.Person Where Person_ID = 888;

-- Verify the deletion in the Diseased_Patient table
SELECT * FROM curated_layer.Diseased_Patient
WHERE Patient_ID = 888;
```

	person_id [PK] integer	last_name character varying (50)	first_name character varying (50)	gender character (1)	primary_location_id integer	race_cd character varying (5)
1	888	Torres	Timothy	F	35095	WHT

	person_id [PK] integer	last_name character varying (50)	first_name character varying (50)	gender character (1)	primary_location_id integer	race_cd character varying (5)

	patient_id [PK] integer	disease_id [PK] integer	severity_value integer	start_date [PK] date	end_date date

Curated Transaction Layer DML Illustrations

Case 5 : A medicine is discontinued and must be deleted from the system.

```
-- Data Before Delete
SELECT * FROM curated_layer.Medicine  WHERE Medicine_ID = 134;
-- Data After Update
DELETE FROM curated_layer.Medicine
WHERE Medicine_ID = 101;
SELECT * FROM curated_layer.Medicine  WHERE Medicine_ID = 101;
```

```
-- Verify the deletion in the Medicine_Disease_Interaction table
SELECT Medicine_ID, Disease_ID, First_Available_Date
FROM curated_layer.Medicine_Disease_Interaction
WHERE Medicine_ID = 101;
```

	medicine_id [PK] integer	standard_industry_number character varying (25)	name character varying (250)	manufacturer_id integer	manufacturer_name character varying (150)	active_ingredient_name character varying (150)
1	134	0002-0805-20	Creon	1	AbbVie	Pancrelipase

	medicine_id [PK] integer	standard_industry_number character varying (25)	name character varying (250)	manufacturer_id integer	manufacturer_name character varying (150)	active_ingredient_name character varying (150)

Data Output

	medicine_id [PK] integer	disease_id [PK] integer	first_available_date date

Server-side aspects (1/2)

```sql
CREATE OR REPLACE TRIGGER curated_layer.audit_tf() RETURNS TRIGGER LANGUAGE plpgsql as
$$
DECLARE
  k text;
  v text;
  col_list text;
  j_new jsonb := to_jsonb(new);
  j_old jsonb := to_jsonb(old);
BEGIN
    IF TG_OP = 'INSERT' THEN -- only shows new values
        INSERT INTO curated_layer.audit_table (src_table, v_new, action)
            VALUES (TG_TABLE_NAME, j_new, TG_OP);
    ELSIF TG_OP = 'UPDATE' THEN -- shows new and old values

        INSERT INTO curated_layer.audit_table (src_table, v_new, v_old, action)
            VALUES (TG_TABLE_NAME, j_new, j_old, TG_OP);

    ELSIF TG_OP = 'DELETE' THEN -- only shows old values
        INSERT INTO curated_layer.audit_table (src_table, v_old, action)
            VALUES (TG_TABLE_NAME, j_old, TG_OP);
    END IF;
    RETURN NULL;
END;
$$;

CREATE OR REPLACE TRIGGER location_audit_trigger
AFTER INSERT OR UPDATE or DELETE ON curated_layer.location
FOR EACH ROW EXECUTE PROCEDURE curated_layer.audit_tf();
```

Server-side aspects (2/2)

	src_table text	v_old text	v_new text	user text	action text	action_time timestamp with time zone
1	person	{"gender": "M", "race_cd": "ASN", "last_name"...	{"gender": "M", "race_cd": "ASI", "last_nam...	postgres	UPDATE	2024-12-16 20:04:07.22532-05
2	person	{"gender": "F", "race_cd": "ASN", "last_name": ...	{"gender": "F", "race_cd": "ASI", "last_nam...	postgres	UPDATE	2024-12-16 20:04:07.22532-05
3	race_disease_propensity	{"race_code": "ASN", "disease_id": 76, "prope...	{"race_code": "ASI", "disease_id": 76, "pro...	postgres	UPDATE	2024-12-16 20:04:07.22532-05
4	person	{"gender": "U", "race_cd": "ASN", "last_name":...	{"gender": "U", "race_cd": "ASI", "last_nam...	postgres	UPDATE	2024-12-16 20:04:07.22532-05
5	person	{"gender": "F", "race_cd": "ASN", "last_name": ...	{"gender": "F", "race_cd": "ASI", "last_nam...	postgres	UPDATE	2024-12-16 20:04:07.22532-05
6	person	{"gender": "U", "race_cd": "ASN", "last_name":...	{"gender": "U", "race_cd": "ASI", "last_nam...	postgres	UPDATE	2024-12-16 20:04:07.22532-05
7	race_disease_propensity	{"race_code": "ASN", "disease_id": 35, "prope...	{"race_code": "ASI", "disease_id": 35, "pro...	postgres	UPDATE	2024-12-16 20:04:07.22532-05
8	race_disease_propensity	{"race_code": "ASN", "disease_id": 52, "prope...	{"race_code": "ASI", "disease_id": 52, "pro...	postgres	UPDATE	2024-12-16 20:04:07.22532-05
9	race_disease_propensity	{"race_code": "ASN", "disease_id": 91, "prope...	{"race_code": "ASI", "disease_id": 91, "pro...	postgres	UPDATE	2024-12-16 20:04:07.22532-05
10	race	{"race_code": "ASN", "race_description": "Asi...	{"race_code": "ASI", "race_description": "A...	postgres	UPDATE	2024-12-16 20:04:07.22532-05

Audit Log Snippets

	src_table text	v_old text	v_new text	user text	action text	action_time timestamp with time zone
1	disease_type	[null]	{"disease_type_code": "GENET", "dis...	postgres	INSERT	2024-12-16 20:01:00.201489-05
2	disease_type	[null]	{"disease_type_code": "OTHER", "di...	postgres	INSERT	2024-12-16 20:01:00.201489-05
3	disease_type	[null]	{"disease_type_code": "BACTE", "dis...	postgres	INSERT	2024-12-16 20:01:00.201489-05
4	disease_type	[null]	{"disease_type_code": "VIRAL", "dise...	postgres	INSERT	2024-12-16 20:01:00.201489-05
5	disease_type	[null]	{"disease_type_code": "AUIMM", "di...	postgres	INSERT	2024-12-16 20:01:00.201489-05
6	race	[null]	{"race_code": "WHT", "race_descript...	postgres	INSERT	2024-12-16 20:01:00.248891-05
7	race	[null]	{"race_code": "BLK", "race_descripti...	postgres	INSERT	2024-12-16 20:01:00.248891-05
8	race	[null]	{"race_code": "HIS", "race_descripti...	postgres	INSERT	2024-12-16 20:01:00.248891-05
9	race	[null]	{"race_code": "MID", "race_descripti...	postgres	INSERT	2024-12-16 20:01:00.248891-05
10	race	[null]	{"race_code": "IND", "race_descripti...	postgres	INSERT	2024-12-16 20:01:00.248891-05

Data Warehouse – Overview (1/2)

A quick recap of our objective and use-cases

Project Introduction

Objective

- Aid our client, the World Health Organization (WHO) to leverage data-driven insights to effectively monitor, analyze, and respond to global disease outbreaks.
- Make the solution as an accelerator to deploy it for other government-based and not for profit health management boards across the globe.

Use-Cases and Decisions

Use-Cases	Potential Decisions
Disease Outbreak Identification and Tracking	Deploy rapid response teams to affected areas, implement containment measures, and allocate resources effectively
Disease Severity Assessment and Prioritization	Prioritize resource allocation to diseases with increasing severity, develop targeted interventions, and allocate funds for research and development
Identifying Vulnerable Populations	Develop targeted public health campaigns, allocate resources to vulnerable communities, and implement preventive measures
Evaluating Medicine Effectiveness and Manufacturer Performance	Make informed decisions about drug procurement, recommend effective treatments, and establish partnerships with reliable manufacturers.
Optimizing Resource Allocation	Deploy medical personnel and equipment to areas with high disease burden, prioritize research efforts, and establish surveillance systems

Data Warehouse – Overview (2/2)

Fact and Dimension Tables

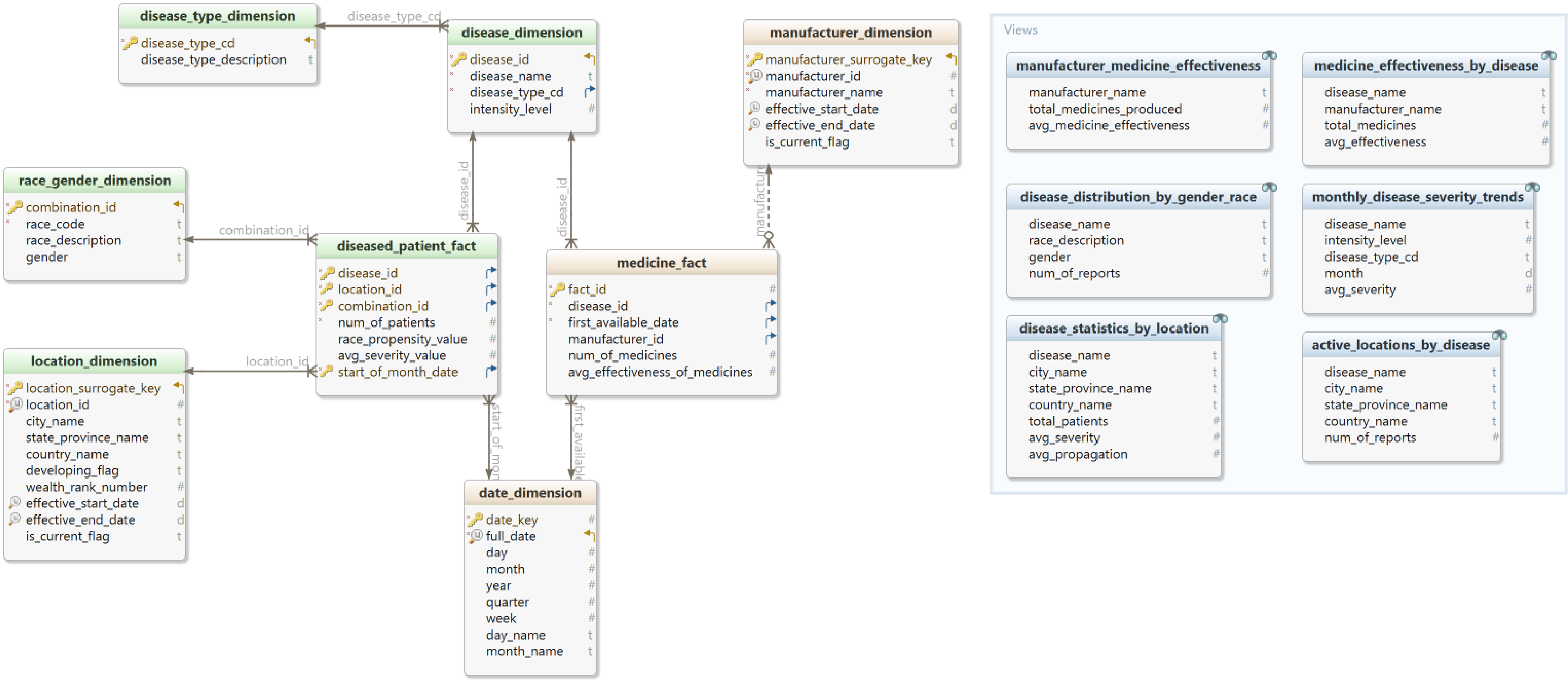
Dimension Tables
Disease_Type_Dimension
Disease_Dimension
Location_Dimension
Date_Dimension
Race_Gender_Dimension
Manufacturer_Dimension

Fact Tables	Grain	Metrics
Diseased_Patient_Fact	The number of patients affected by a disease in a specific location, for a specific race-gender combination, during a specific time period	• Number of Patients • Race Propensity Value • Average Severity Value
Medicine_Fact	The aggregate data of medicines associated with a disease, manufactured by a specific company	• Number of Medicines • Average Effectiveness • First Available Date

Other Salient Features

- **Snowflake Schema** – Disease_type and Disease dimension
- **SCD Type 2** on Location Dimension and Manufacturer Dimension
- **Junk Dimension** on Race Gender Dimension
- **Degenerate Dimension** on Medicine Fact table due to fact_id

Data Warehouse - Schema



Refer to **Data Warehouse ER Diagram** PDF file for more information

Data Warehouse – ELT Process

- Postgres SQL based automated ELT Pipeline – Fast and SQL Native
- Python is just used to automate the process
- Advanced ML/ Analytical transformation are not required at this layer.
- We will take a closer look into the Python file – **ELT_curated_to_warehouse.py**

Key Pipeline Steps

Reporting Layer (Data Warehouse) Insertion Pipeline

Temporary Stage Tables from Curated layer → **Transformation to DW dimensions and Facts and Insert** → **Drop Temporary Tables**

ELT Snippet

```
# Load Manufacturer_Dimension
execute_query(conn, """
    INSERT INTO reporting_layer_DW.Manufacturer_Dimension
        (manufacturer_id, manufacturer_name, is_current_flag)
    SELECT Distinct manufacturer_id, manufacturer_name, 'Y'
    FROM staging_medicine WHERE manufacturer_id IS NOT NULL;
""")
```

Live Tableau Report Snippets from a few Views created

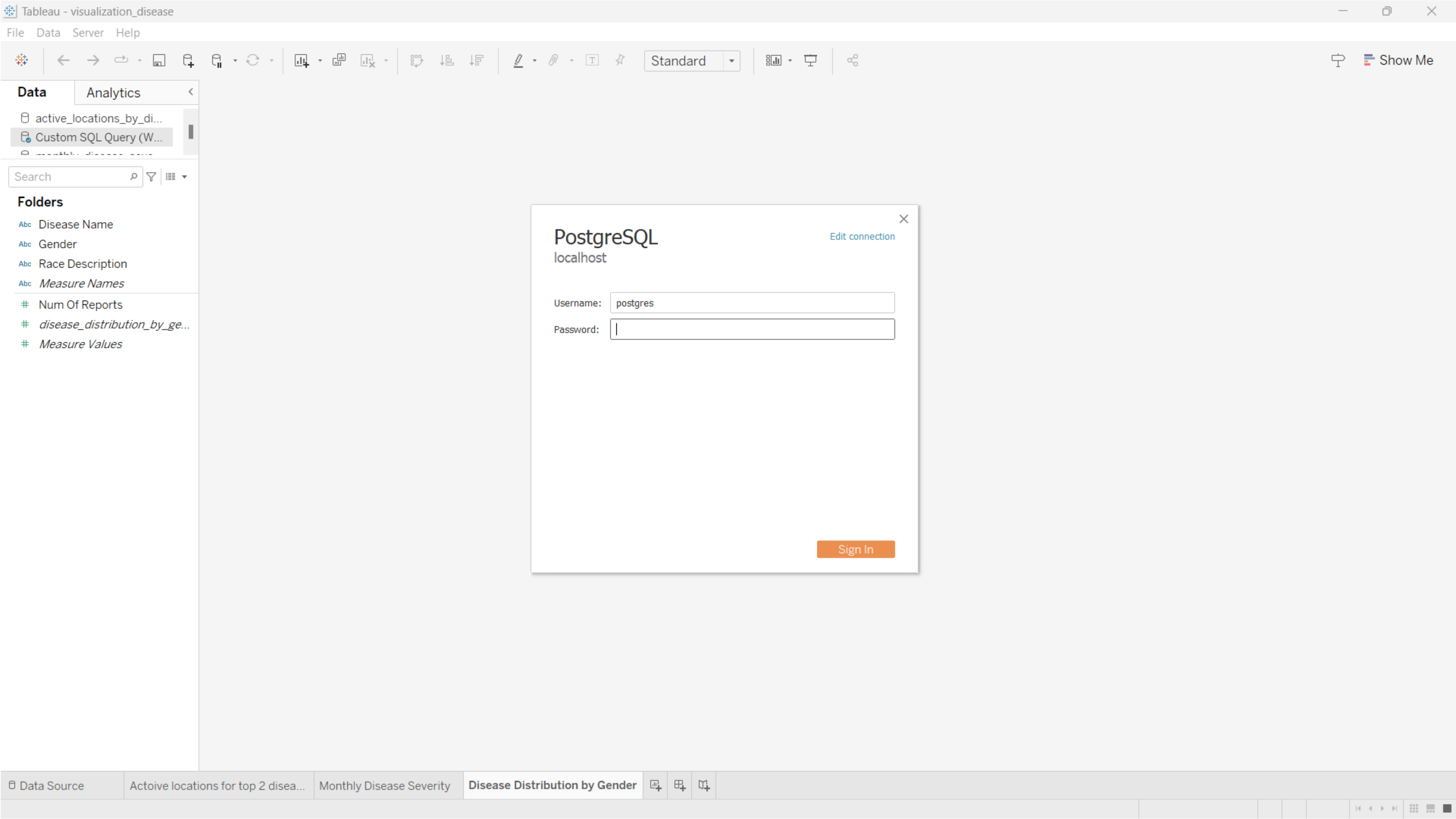
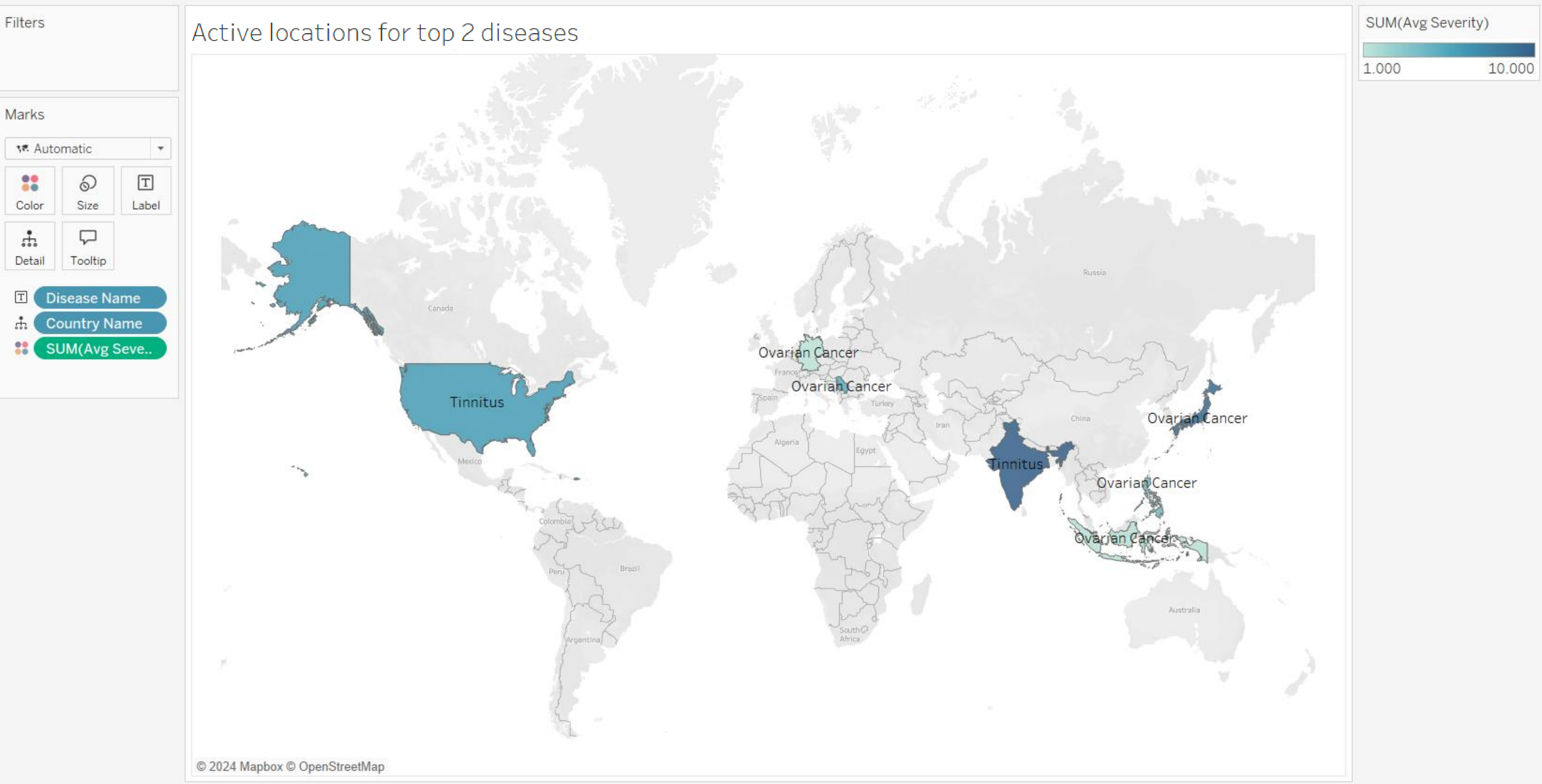
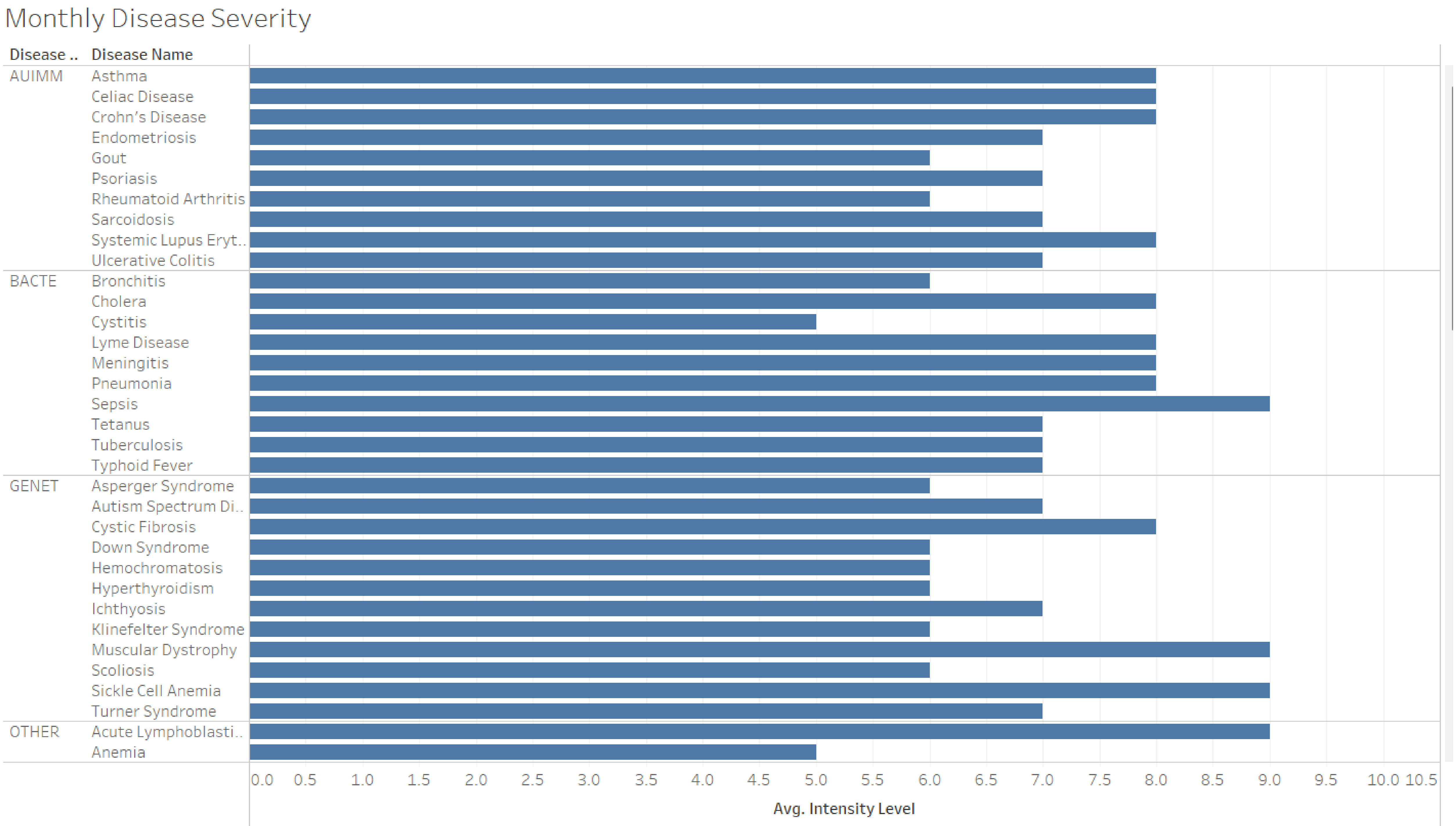


Tableau is directly connected to the views of data warehouse.

With different username for predefined roles, we can restrict the information in terms of rows, columns and even tables which the user can see. For e.g. the WHO health liaison to Africa needs to see the same information for Africa region alone.


Filters
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Disease Name
Country Name
SUM(Avg Seve..
Active locations for top 2 diseases
Ovarian Cancer
Ovarian Cancer
Tinnitus
Ovarian Cancer
Tinnitus
Ovarian Cancer
Ovarian Cancer
Canada
Russia
Mexico
China
Brazil
Australia
© 2024 Mapbox © OpenStreetMap
SUM(Avg Severity)
1.000
10.000

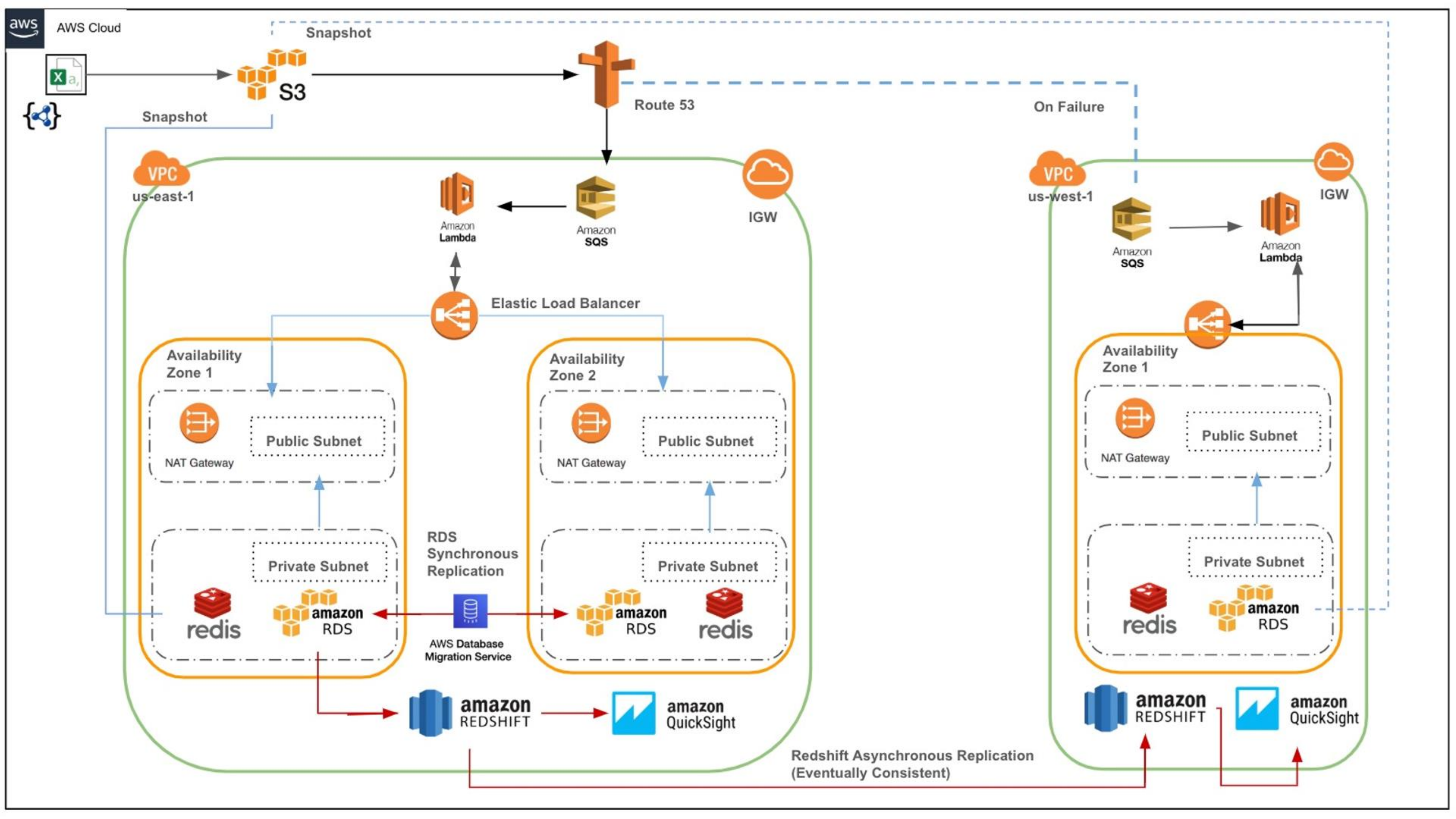


Monthly Disease Severity
Disease ..
Disease Name
AUIMM
Asthma
Celiac Disease
Crohn's Disease
Endometriosis
Gout
Psoriasis
Rheumatoid Arthritis
Sarcoidosis
Systemic Lupus Eryt..
Ulcerative Colitis
BACTE
Bronchitis
Cholera
Cystitis
Lyme Disease
Meningitis
Pneumonia
Sepsis
Tetanus
Tuberculosis
Typhoid Fever
GENET
Asperger Syndrome
Autism Spectrum Di..
Cystic Fibrosis
Down Syndrome
Hemochromatosis
Hyperthyroidism
Ichthyosis
Klinefelter Syndrome
Muscular Dystrophy
Scoliosis
Sickle Cell Anemia
Turner Syndrome
OTHER
Acute Lymphoblasti..
Anemia
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5 10.0 10.5
Avg. Intensity Level

Disease Distribution by Gender

Gender	Disease Name	
F	Glaucoma	2
	Ovarian Cancer	4
	Preeclampsia	2
	Rheumatoid Arthritis	2
M	Crohn’s Disease	2
	Cystitis	2
	Hypertensive Heart ..	2
	Sickle Cell Anemia	3
	Tinnitus	4
	Ulcerative Colitis	2
U	Hemochromatosis	2
	Ovarian Cancer	8
	Pneumonia	2
	Sarcoidosis	2
	Tuberculosis	2
	Ulcerative Colitis	2

AWS Architecture for the Project

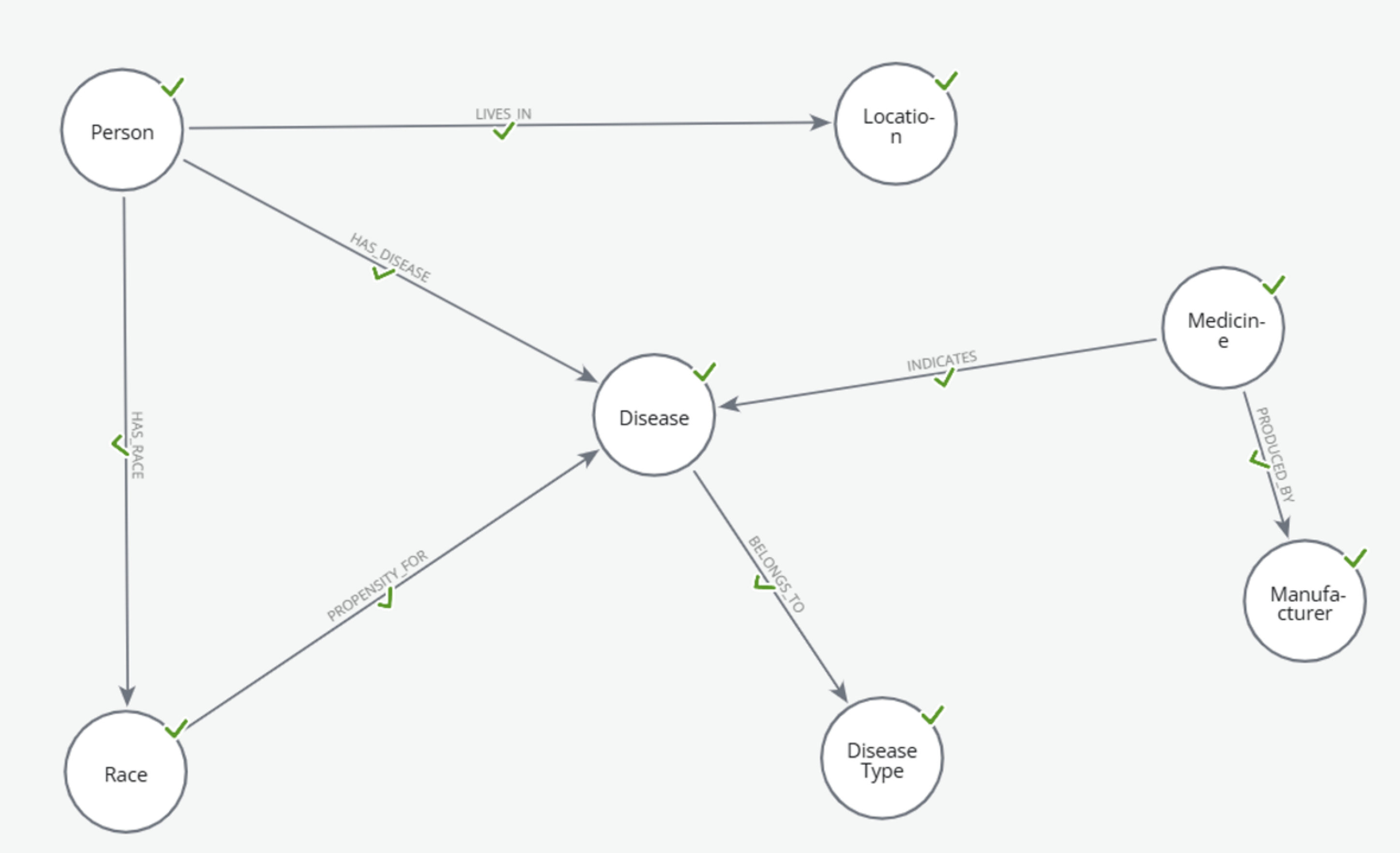


Comment on Usage of No-SQL tools – Neo4j

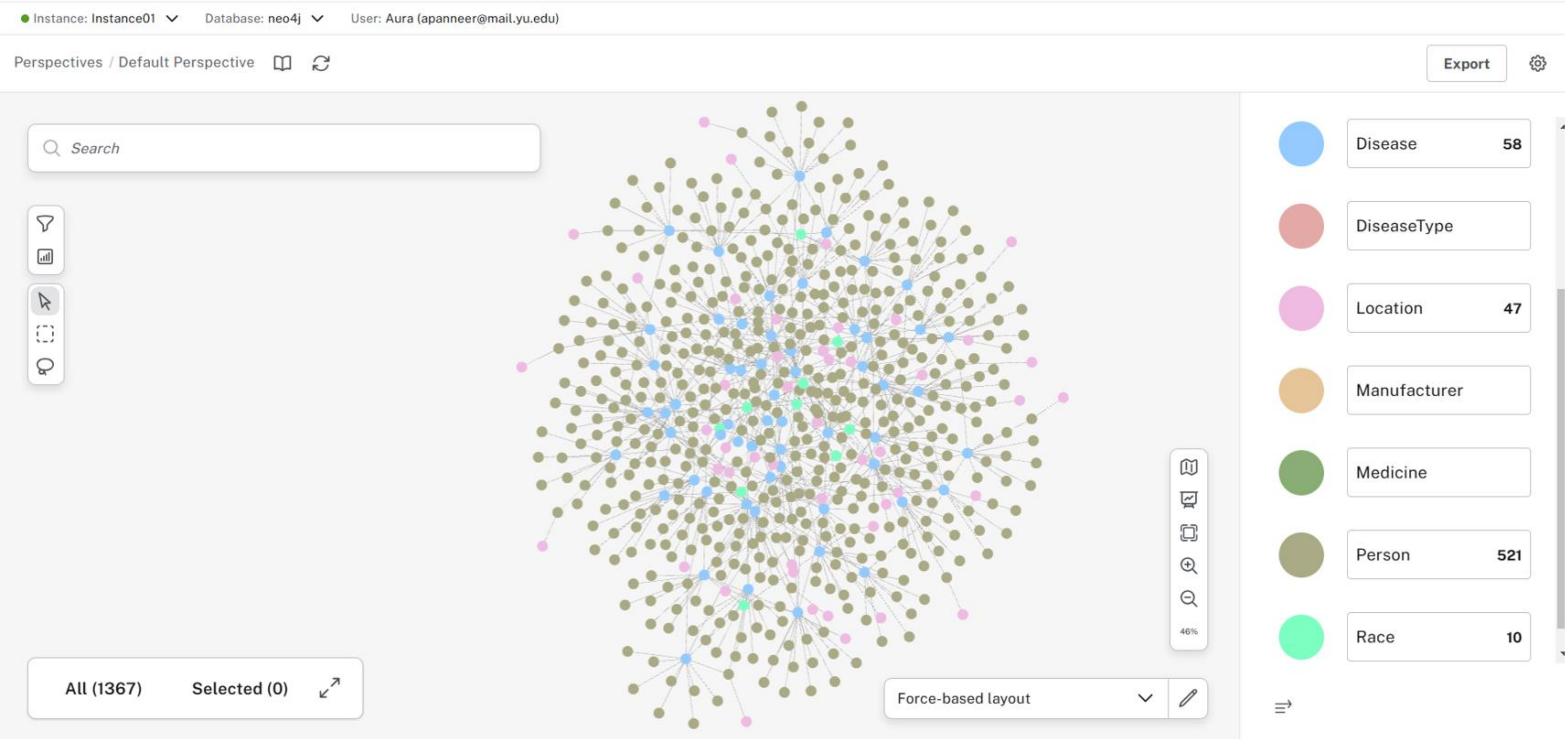
Advantages of Neo4j over Postgres for our current model

- Relationship-Centric Data Modeling
- Schema Flexibility
- Built-In Graph Algorithms (especially Centrality, Community Detection)
- Polyglot resolutions and Performance on Connected Data
- Potential to use Pre-defined Ontologies to tap into vast Knowledge base

Graph Model



Comment on Usage of No-SQL tools – Neo4j



Comment on Usage of Snowflake (1/2)

- Inherent Data Integration with Stages, Streams, Tasks, Pipes
  - Effective to handle polyglot persistence which is the primary challenge for our model

- Effective Partitioning for Automated query tuning
  - Helpful to handle huge quantity of data of world-wide patients and disease propagation

- Intuitive Profiler tool to optimize reporting layer queries
  - Ideal to reduce cost of compute usage

- Robust Role and Encryption based Security
  - Roles for various users inside WHO and other stakeholder like countries government across the world

Comment on Usage of Snowflake (2/2)

- Attractive pricing model using credits.
  - Separate cost for storage and compute. Ideal for NGOs and humanitarian organization like WHO.

- Dynamic scaling of Warehouse size, depending on the requirements
  - In normal scenario, our reports are updated every month. But in pandemic situations like COVID-19, real-time updates and daily updates are required

- Vast Partner Network and Community through Marketplace
  - Since we rely on 3rd party data, tapping into data providers network on snowflake will reduce unwanted setup and configuration tasks

- Time-Travel
  - Our reports are used for making sensitive decision for welfare of people. Hence, any mishaps to data due to negligence or cyber-attacks can be mitigated using time travel feature

Thank You