STRUCTURE DATA MANAGEMENT

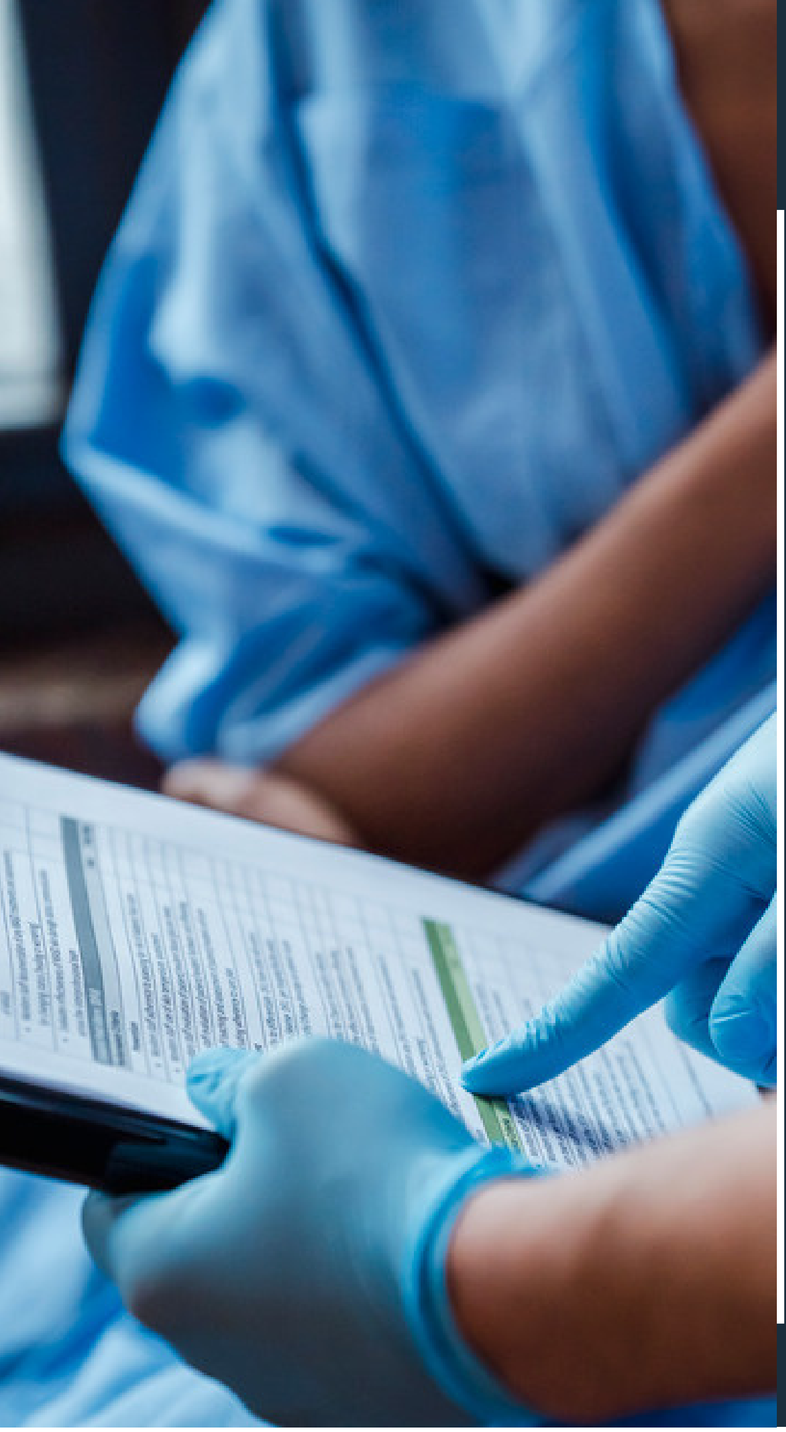
Final Project

DISEASE MODEL

Canva

Introduction

In today's healthcare landscape, the effective management and analysis of disease-related data are critical for making informed decisions, providing quality care, and advancing medical research.
This project aims to address the complexities of disease data management by implementing a robust and scalable database solution.



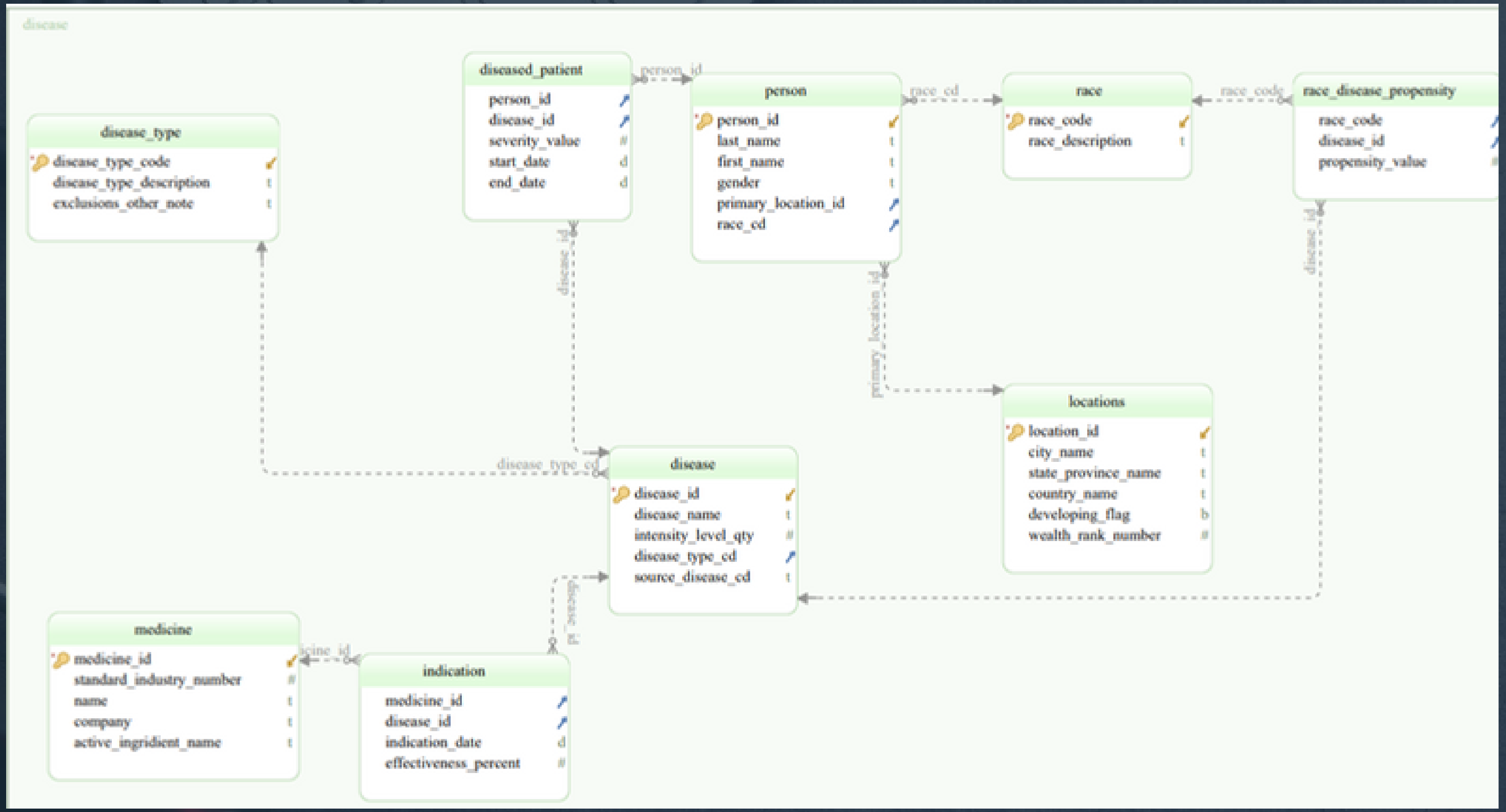
Business Problem

Healthcare institutions, research organizations, and medical professionals face challenges in handling and analyzing disease-related data efficiently. There is a need for a comprehensive database solution that not only stores extensive disease data but also enables seamless operations, analytics, and insightful reporting.

Database Design Approach

Our database design is rooted in a sophisticated Entity-Relationship model that captures the nuances of disease-related information. Drawing upon the foundational concepts of data modeling, our design includes entities such as Patients, Diseases, Treatments, Medicines, Symptoms, and more. The ER diagram reflects the relationships between these entities, their cardinality, and attributes that define their characteristics

E.R - DIAGRAM


disease
diseased_patient
person_id
disease_id
severity_value
start_date
end_date
person_id
person
person_id
last_name
first_name
gender
primary_location_id
race_cd
race_cd
race
race_code
race_description
race_code
race_disease_propensity
race_code
disease_id
propensity_value
disease_type
disease_type_code
disease_type_description
exclusions_other_note
disease_id
primary_location_id
locations
location_id
city_name
state_province_name
country_name
developing_flag
wealth_rank_number
disease_type_cd
disease
disease_id
disease_name
intensity_level_qty
disease_type_cd
source_disease_cd
disease_id
medicine
medicine_id
standard_industry_number
name
company
active_ingridient_name
medicine_id
indication
medicine_id
disease_id
indication_date
effectiveness_percent

Data Dictionaries

Main Layout

Table disease

* Pk	disease_id	varchar(50)
	disease_name	varchar(200)
	intensity_level_qty	double precision
	disease_type_cd	varchar(20)
	source_disease_cd	varchar(30)
Indexes		
Pk	disease_pkey	disease_id
Foreign Keys		
	disease_disease_type_cd_fkey (disease_type_cd) ref disease_type (disease_type_code)	

Table diseased_patient

	person_id	integer
	disease_id	varchar(50)
	severity_value	double precision
	start_date	date
	end_date	date
Foreign Keys		
	diseased_patient_disease_id_fkey (disease_id) ref disease (disease_id)	
	diseased_patient_person_id_fkey (person_id) ref person (person_id)	

Table person

* Pk	person_id	integer
	last_name	varchar(50)
	first_name	varchar(50)
	gender	varchar(12)
	primary_location_id	integer
	race_cd	varchar(20)
Indexes		
Pk	person_pkey	person_id
Foreign Keys		
	person_primary_location_id_fkey (primary_location_id) ref locations (location_id)	
	person_race_cd_fkey (race_cd) ref race (race_code)	

Table indication

	medicine_id	varchar(20)
	disease_id	varchar(50)
	indication_date	date
	effectiveness_percent	double precision

Foreign Keys

	indication_disease_id_fkey (disease_id) ref disease (disease_id)
	indication_medicine_id_fkey (medicine_id) ref medicine (medicine_id)

Table medicine

* Pk	medicine_id	varchar(20)
	standard_industry_number	integer
	name	varchar(100)
	company	varchar(100)
	active_ingridient_name	varchar(50)

Indexes

Pk	medicine_pkey	medicine_id

Table locations

* Pk	location_id	integer
	city_name	varchar(50)
	state_province_name	varchar(50)
	country_name	varchar(50)
	developing_flag	boolean
	wealth_rank_number	integer

Indexes

Pk	locations_pkey	location_id

Table race

* Pk	race_code	varchar(30)
	race_description	varchar(300)
Indexes		
Pk	race_pkey	race_code

Table disease_type

* Pk	disease_type_code	varchar(20)
	disease_type_description	varchar(200)
	exclusions_other_note	text
Indexes		
Pk	disease_type_pkey	disease_type_code

Table race_disease_propensity

race_code	varchar(20)
disease_id	varchar(50)
propensity_value	double precision
Foreign Keys	
race_disease_propensity_disease_id_fkey (disease_id) ref disease (disease_id)	
race_disease_propensity_race_code_fkey (race_code) ref race (race_code)	

Operational Queries and Reports

Let's take some examples of some queries of People Suffering from Cancer

```sql
--Q1. people who are suffering from cancer (disease_id=23)

SELECT
    P.Person_id,
    P.Last_Name,
    P.First_Name,
    DP.Severity_Value,
    DP.Start_date,
    DP.End_Date
FROM
    DISEASED_PATIENT DP
JOIN
    PERSON P ON DP.Person_ID = P.Person_id
WHERE
    DP.Disease_ID = '23';
```

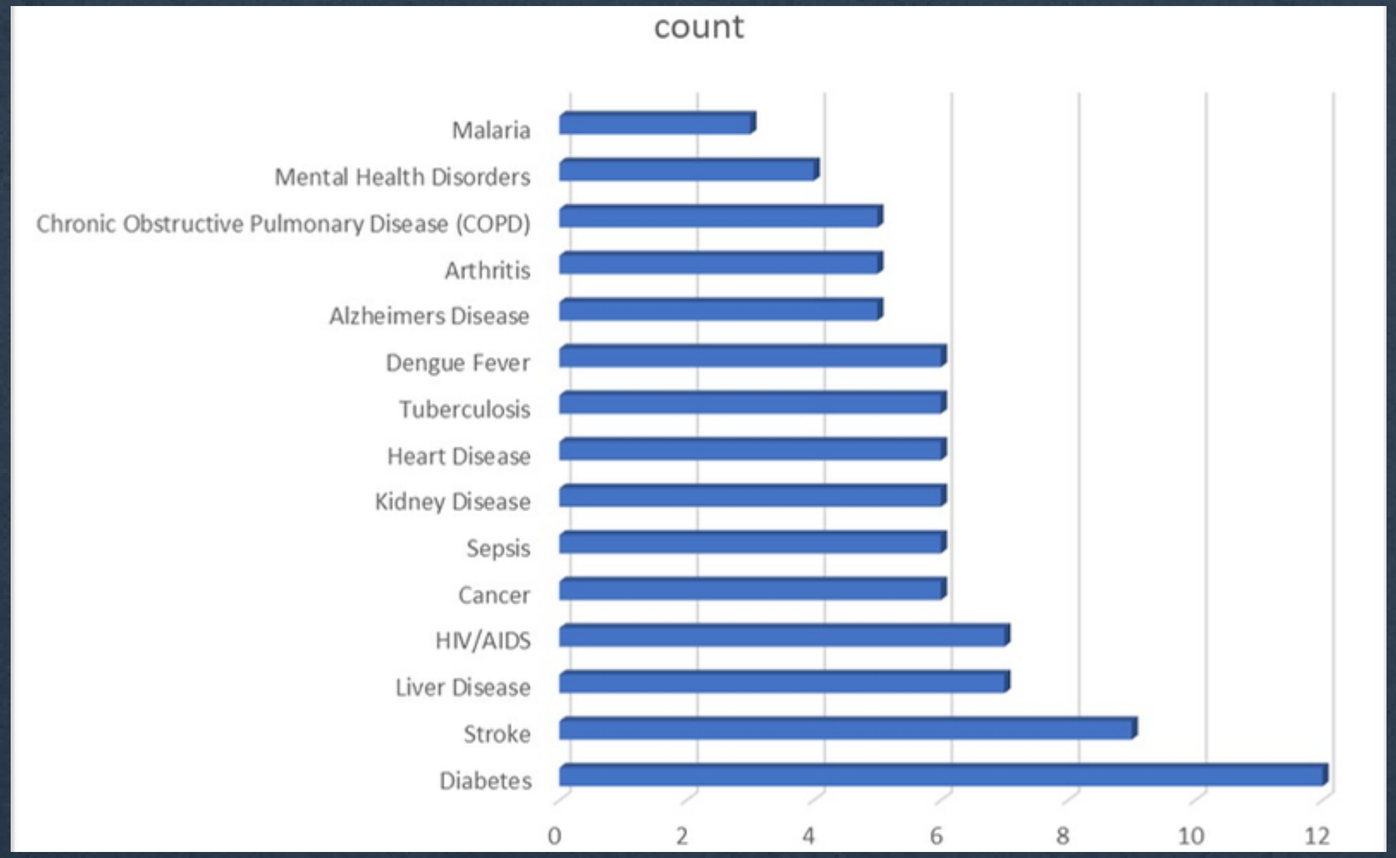
	person_id integer	last_name character varying	first_name character varying	severity_value double precision	start_date date	end_date date
1	58	Martinez	Sophia	0.8	2023-06-05	2023-12-28
2	64	Jackson	Ava	0.3	2023-08-14	2024-02-08
3	92	Cook	Grace	0.8	2023-06-05	2023-12-28
4	95	Reed	Elijah	0.3	2023-08-14	2024-02-08
5	113	Myers	Oliver	0.9	2023-12-03	2024-06-25
6	121	Chavez	Lucy	0.8	2023-06-05	2023-12-28

Count of people for each Disease

```sql
--Q2. Count of people for each disease
with disease_counts as (Select disease_id,count(*) from diseased_patient
group by disease_id)

Select d.disease_name , dc."count"
from disease_counts as dc
inner join
disease d
on d.disease_id = dc.disease_id
order by 2 desc;
```

	disease_name character varying	count bigint
1	Diabetes	12
2	Stroke	9
3	Liver Disease	7
4	HIV/AIDS	7
5	Cancer	6
6	Sepsis	6
7	Kidney Disease	6
8	Heart Disease	6
9	Tuberculosis	6
10	Dengue Fever	6
11	Alzheimers Disease	5
12	Arthritis	5
13	Chronic Obstructive Pulmonary Disease (COPD)	5
14	Mental Health Disorders	4
15	Malaria	3



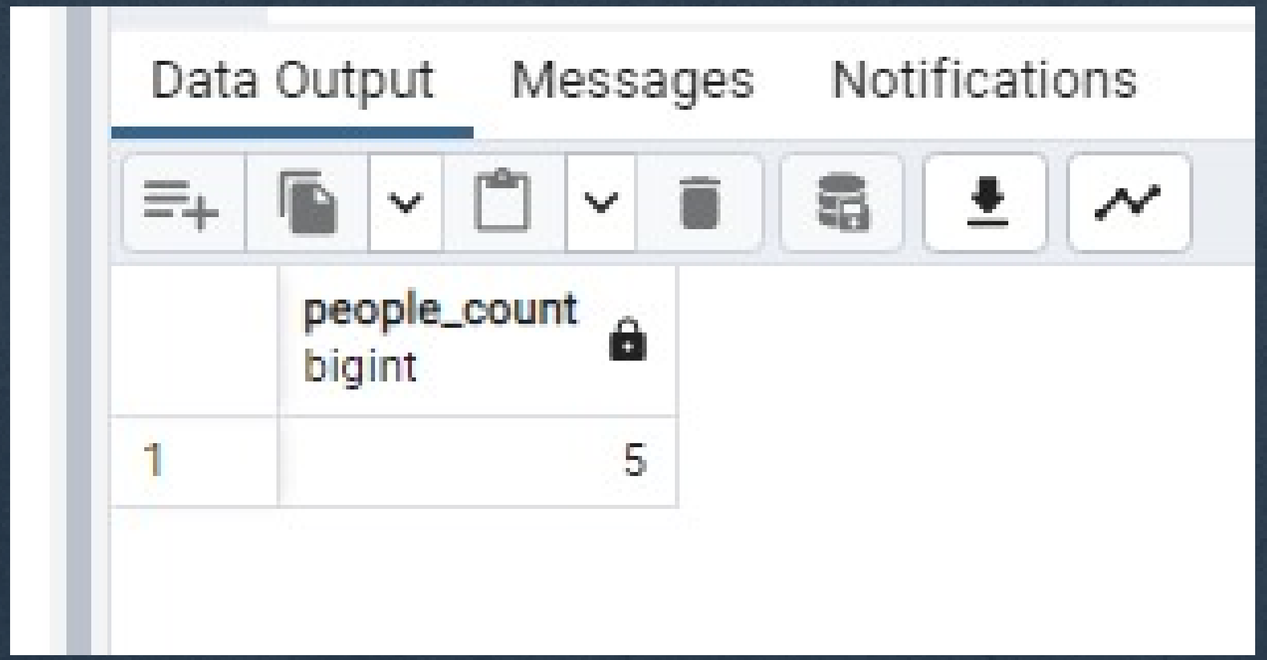
Common Disease for Each Country

```sql
--Q3 For each country which is common disease
WITH RankedDiseases AS (
    SELECT
        L.Country_Name,
        D.Disease_ID,
        D.Disease_Name,
        COUNT(*) AS Disease_Count,
        ROW_NUMBER() OVER (PARTITION BY L.Country_Name ORDER BY COUNT(*) DESC) AS rn
    FROM
        LOCATIONS L
    JOIN
        PERSON P ON L.Location_ID = P.Primary_location_ID
    JOIN
        DISEASED_PATIENT DP ON P.Person_id = DP.Person_ID
    JOIN
        DISEASE D ON DP.Disease_ID = D.Disease_ID
    GROUP BY
        L.Country_Name, D.Disease_ID, D.Disease_Name
)
SELECT
    Country_Name,
    Disease_ID,
    Disease_Name,
    Disease_Count
FROM
    RankedDiseases
WHERE
    rn = 1;
```

	country_name character varying	disease_id character varying	disease_name character varying	disease_count bigint
1	Argentina	30	HIV/AIDS	3
2	Australia	26	Stroke	2
3	Brazil	33	Tuberculosis	2
4	Canada	27	Kidney Disease	1
5	China	23	Cancer	2
6	Egypt	31	Arthritis	2
7	France	32	Sepsis	2
8	Germany	24	Alzheimers Disease	2
9	India	24	Alzheimers Disease	1
10	Japan	28	Liver Disease	1
11	Mexico	25	Chronic Obstructive Pulmonary Disease (COPD)	2
12	Russia	21	Diabetes	2
13	South Africa	21	Diabetes	2
14	United Kingdom	22	Heart Disease	2
15	United States	35	Dengue Fever	3

Number of people from United States

```
58  --Q4
59  SELECT
60      COUNT(*) AS People_Count
61  FROM
62      LOCATIONS L
63  JOIN
64      PERSON P ON L.Location_ID = P.Primary_location_ID
65  WHERE
66      L.Country_Name = 'United States';
67
```



Combination of Medicine and Corresponding Disease

```sql
--Q5--
SELECT
    M.Medicine_ID,
    M.Name AS Medicine_Name,
    M.Company,
    M.Active_Ingridient_Name,
    D.Disease_ID,
    D.Disease_Name
FROM
    MEDICINE M
JOIN
    INDICATION I ON M.Medicine_ID = I.Medicine_ID
JOIN
    DISEASE D ON I.Disease_ID = D.Disease_ID;
```

	medicine_id character varying	medicine_name character varying	company character varying	active_ingridient_name character varying	disease_id character varying	disease_name character varying
1	1	Metformin	Acme Pharmaceuticals	metformin hydrochloride	21	Diabetes
2	2	Insulin	Novo Nordisk	insulin glargine	22	Heart Disease
3	3	Lipitor	Pfizer	atorvastatin calcium	23	Cancer
4	4	Aricept	Eisai	donepezil hydrochloride	24	Alzheimers Disease
5	5	Sinemet	Merck & Co.	levodopa and carbidopa	25	Chronic Obstructive Pulmonary Dise
6	6	Copaxone	Teva Pharmaceutical Industries	glatiramer acetate	26	Stroke
7	7	Truvada	Gilead Sciences	emtricitabine and tenofovir disoproxil fumarate	27	Kidney Disease
8	8	Interferon	Merck Sharp & Dohme	interferon alfa-2b	28	Liver Disease
9	9	Isoniazid	Pfizer	isoniazid	29	Mental Health Disorders
10	10	Lariam	Roche	mefloquine hydrochloride	30	HIV/AIDS
11	11	Denguevax	Sanofi Pasteur	dengue virus type 2 envelope glycoprotein	31	Arthritis
12	12	ZMapp	Mapp Biopharmaceutical	mixture of monoclonal antibodies	32	Sepsis
13	13	Tamiflu	Roche	oseltamivir phosphate	33	Tuberculosis
14	14	Amoxicillin	Pfizer	amoxicillin trihydrate	34	Malaria
15	15	Ciprofloxacin	Bayer	ciprofloxacin hydrochloride	35	Dengue Fever
16	16	Soliqua 100-33	Soliqya	insulin glargine and lixisenatide	21	Diabetes
17	17	Plavix	Sanofi	clopidogrel	22	Heart Disease
18	18	Lisinopril	Generic Pharma	lisinopril	22	Heart Disease
19	19	Tamoxifen	AstraZeneca	tamoxifen	23	Cancer
20	20	Cisplatin	Pfizer	cisplatin	23	Cancer
21	21	Donepezil	Eisai	donepezil	24	Alzheimers Disease
22	22	Memantine	Novartis	memantine	24	Alzheimers Disease
23	23	Aspirin	Bayer	aspirin	26	Stroke
24	24	Clopidogrel	Sanofi	clopidogrel	26	Stroke

DML

Update DML Query and its Effect

```
1  select m.medicine_id, m.Active_Ingridient_Name,i.disease_id,i.effectiveness_percent
2  from medicine m
3  inner join
4  indication i
5  on m.medicine_id=i.medicine_id where m.medicine_id = '26';
6
```

ata Output | Messages | Notifications

	medicine_id character varying (20)	active_ingridient_name character varying (50)	disease_id character varying (50)	effectiveness_percent double precision
	26	Paracetamol	D1	0.9

```
46  -- Updating data in the MEDICINE table
47  UPDATE MEDICINE SET Medicine_ID = '26' WHERE Medicine_ID = 'M1';
48  UPDATE MEDICINE SET Active_Ingridient_Name = 'Paracetamol' WHERE Medicine_ID = '26';
49
50  -- Updating data in the INDICATION table
```

Data Output | Messages | Notifications

UPDATE 1

```
70  select m.medicine_id, m.Active_Ingridient_Name,i.disease_id,i.effectiveness_percent
71  from medicine m
72  inner join
73  indication i
74  on m.medicine_id=i.medicine_id where m.medicine_id = 'M1';
75
```

Data Output | Messages | Notifications

	medicine_id character varying (20)	active_ingridient_name character varying (50)	disease_id character varying (50)	effectiveness_percent double precision
	M1	Acetaminophen	D1	0.9

Delete DML Statement and its Cascading Effect

```
select * from person where person_id = 2;
```

Data Output | Messages | Notifications

person_id [PK] integer	last_name character varying (50)	first_name character varying (50)	gender character varying (12)	primary_location_id integer	race_cd character varying (20)
2	Smith	Jane	Female	2	R2

```
56  -- Deleting data in the PERSON table
57  DELETE FROM PERSON WHERE Person_id = 2;
58
```

Data Output | Messages | Notifications

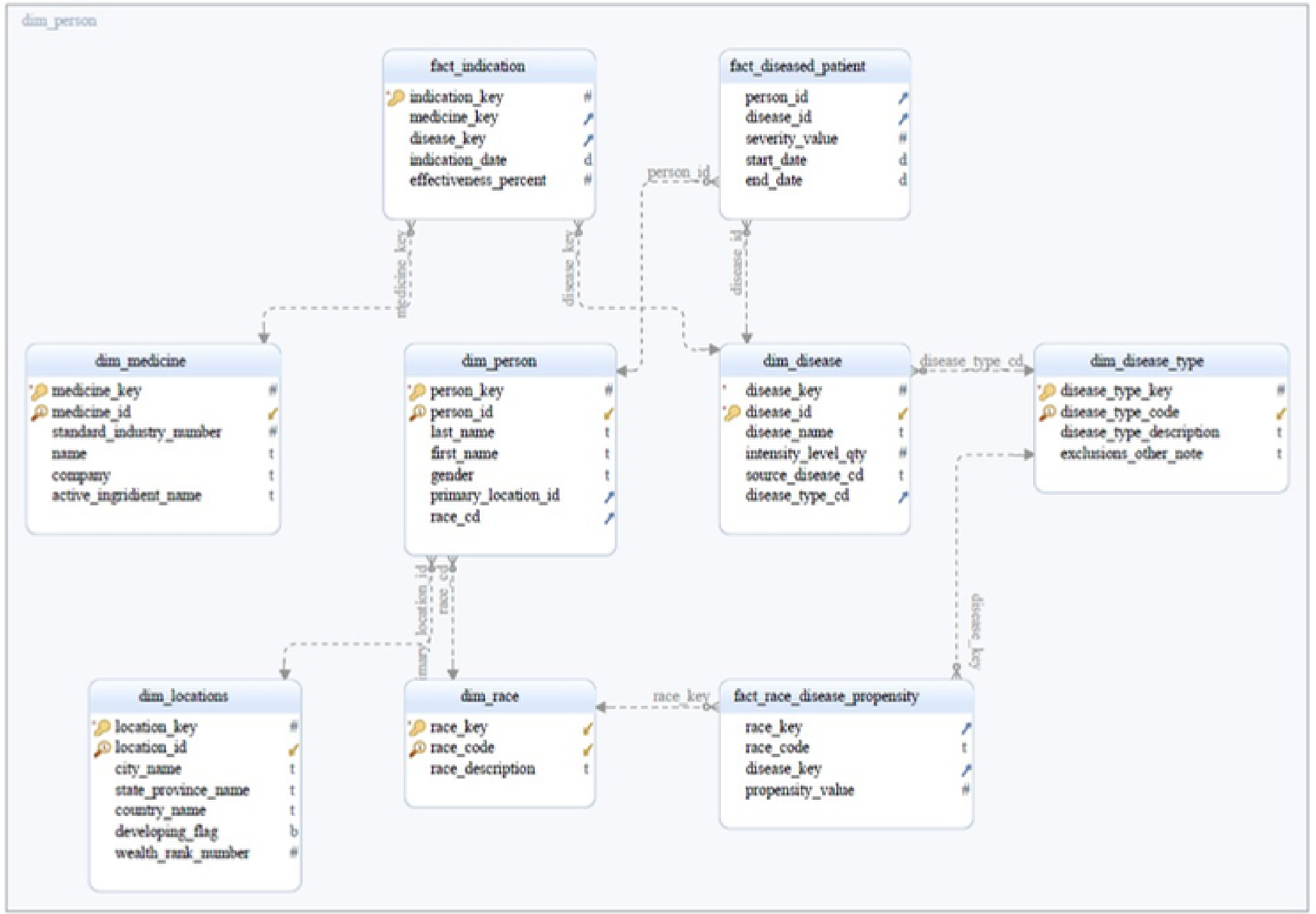
```
DELETE 1
```

```
72  select * from diseased_patient where person_id = 2;
73
```

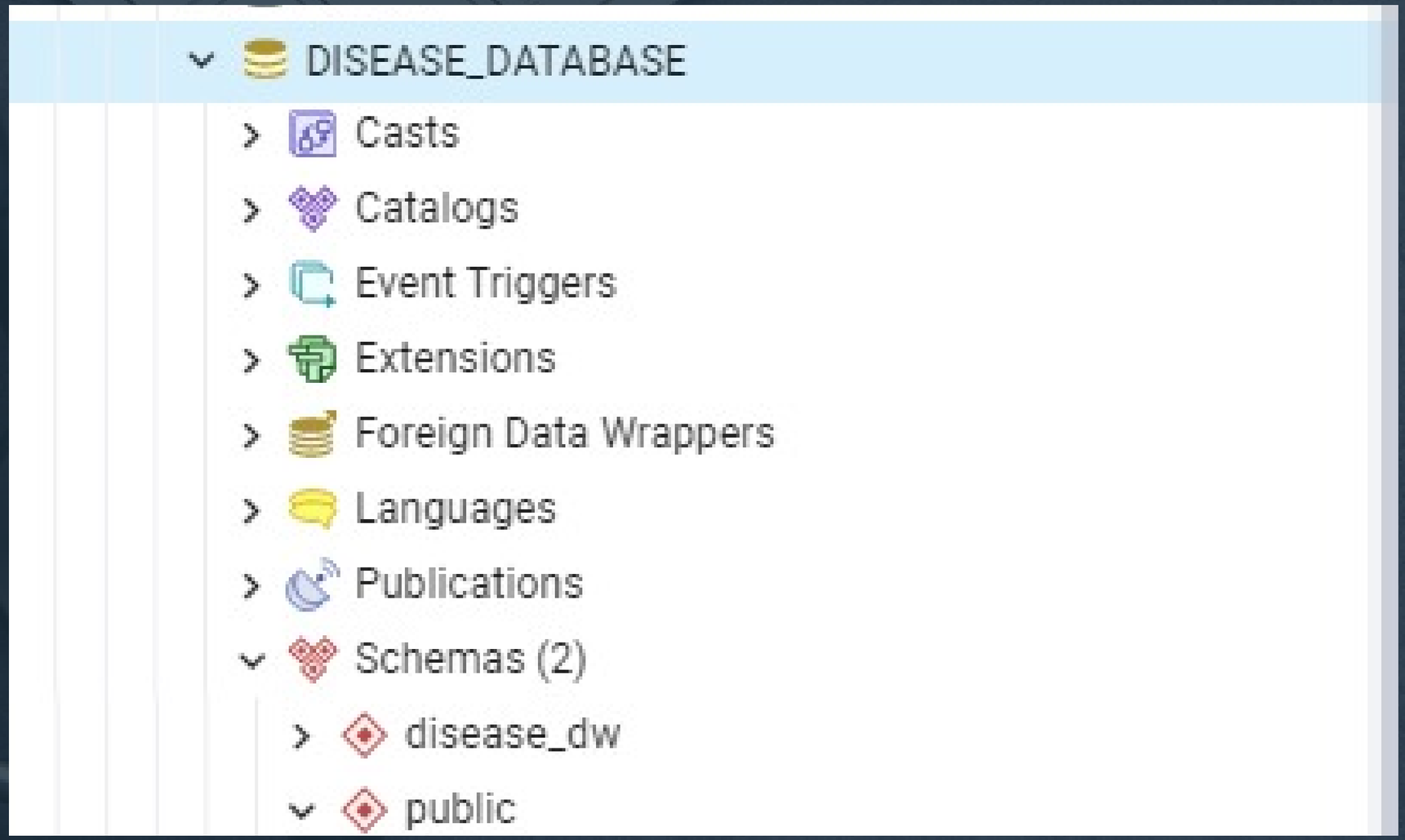
Data Output | Messages | Notifications

person_id integer	disease_id character varying (50)	severity_value double precision	start_date date	end_date date

DATA WAREHOUSE



Datawarehouse Database Schema



For Moving Data Between Databases

SERVER SIDE CODE

```
---Server Side Codes


-- Create a view to retrieve information about patients with diseases
CREATE VIEW disease_dw.ViewPatients AS
SELECT
    DP.Person_ID,
    P.Last_Name,
    P.First_Name,
    D.Disease_Name,
    DP.Severity_Value,
    DP.Start_date,
    DP.End_Date
FROM
    disease_dw.FACT_DISEASED_PATIENT DP
JOIN
    disease_dw.DIM_PERSON P ON DP.Person_ID = P.Person_id
JOIN
    disease_dw.DIM_DISEASE D ON DP.Disease_ID = D.Disease_ID;
```

	person_id integer	last_name character varying	first_name character varying	disease_name character varying	severity_value double precision	start_date date	end_date date
1	41	Patel	Michael	Diabetes	0.7	2023-03-12	2023-10-05
2	42	Garcia	Samantha	Chronic Obstructive Pulmonary Disease (COPD)	0.4	2023-08-24	2023-12-10
3	43	Smith	John	Mental Health Disorders	0.2	2023-01-05	2023-06-14
4	44	Lee	Jessica	Alzheimers Disease	0.8	2023-05-01	2023-11-22
5	45	Williams	Christopher	Diabetes	0.9	2022-12-21	2023-09-18
6	46	Brown	Ashley	Stroke	0.5	2023-07-10	2024-01-08
7	47	Jones	David	Liver Disease	0.3	2023-02-20	2023-08-01
8	48	Miller	Jennifer	Kidney Disease	0.6	2023-06-28	2023-12-17
9	49	Davis	Matthew	Stroke	0.1	2023-04-12	2023-10-26
10	50	Garcia	Elizabeth	Diabetes	0.5	2023-09-04	2024-02-15
11	51	Wilson	Thomas	HIV/AIDS	0.8	2023-01-25	2023-07-27
12	52	Lopez	Emily	Arthritis	0.9	2023-03-08	2023-09-25
13	53	Johnson	James	Tuberculosis	0.2	2023-05-25	2023-11-10
14	54	Smith	Anna	Diabetes	0.1	2023-02-15	2023-08-17
15	55	Moore	William	Dengue Fever	0.6	2023-07-22	2024-01-22
16	56	Taylor	Olivia	Diabetes	0.3	2023-09-01	2023-11-15
17	57	Anderson	Daniel	Heart Disease	0.6	2023-02-12	2023-08-30
18	58	Martinez	Sophia	Cancer	0.8	2023-06-05	2023-12-28
19	59	Hernandez	Ethan	Dengue Fever	0.5	2023-04-15	2023-10-10
20	60	Young	Grace	Arthritis	0.4	2023-07-18	2024-01-05

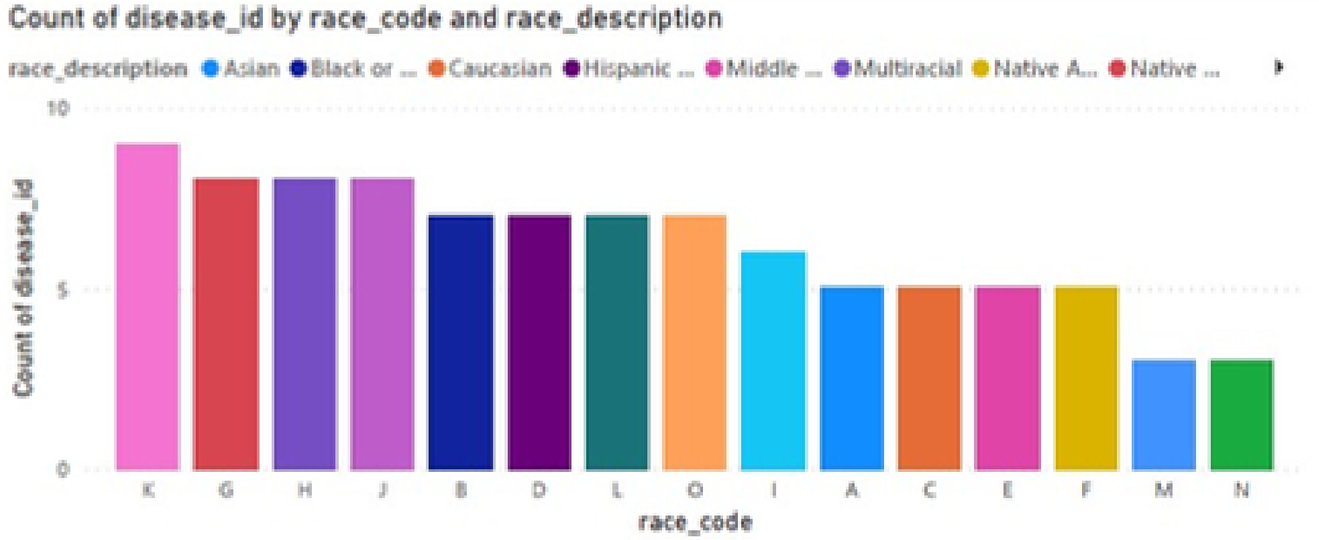
Stored Procedures

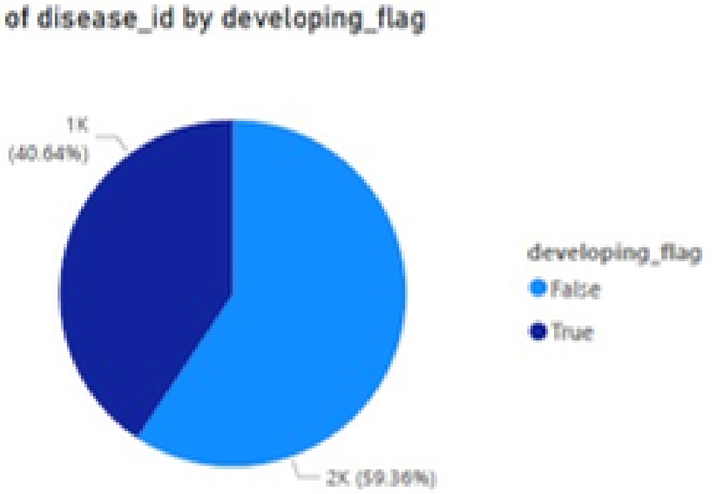
```
24  CREATE OR REPLACE FUNCTION CalculateAverageSeverity(p_DiseaseID VARCHAR(50)) RETURNS FLOAT AS $$
25  DECLARE
26      avgSeverity FLOAT;
27  BEGIN
28      SELECT AVG(Severity_Value) INTO avgSeverity
29      FROM disease_dw.FACT_DISEASED_PATIENT
30      WHERE Disease_ID = p_DiseaseID;
31
32      RETURN avgSeverity;
33  END;
34  $$ LANGUAGE plpgsql;
35
36
37  SELECT CalculateAverageSeverity('22');
38
39
40
```

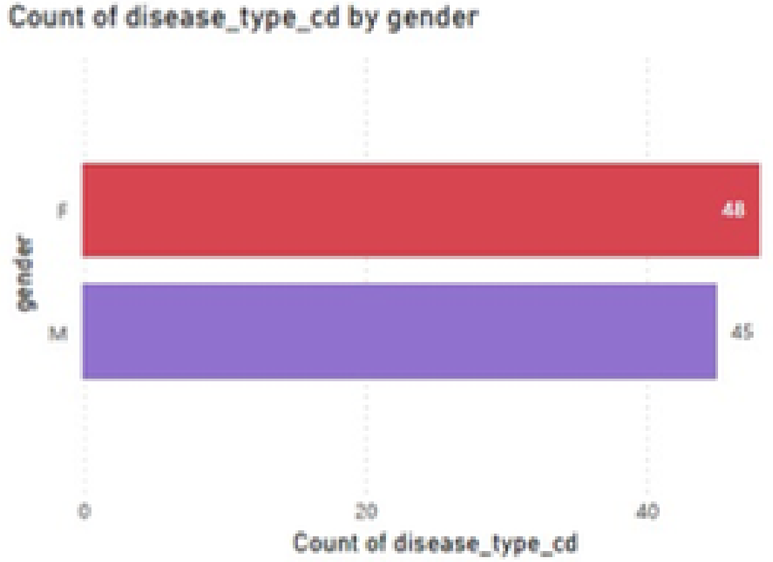
Data Output | Messages | Notifications

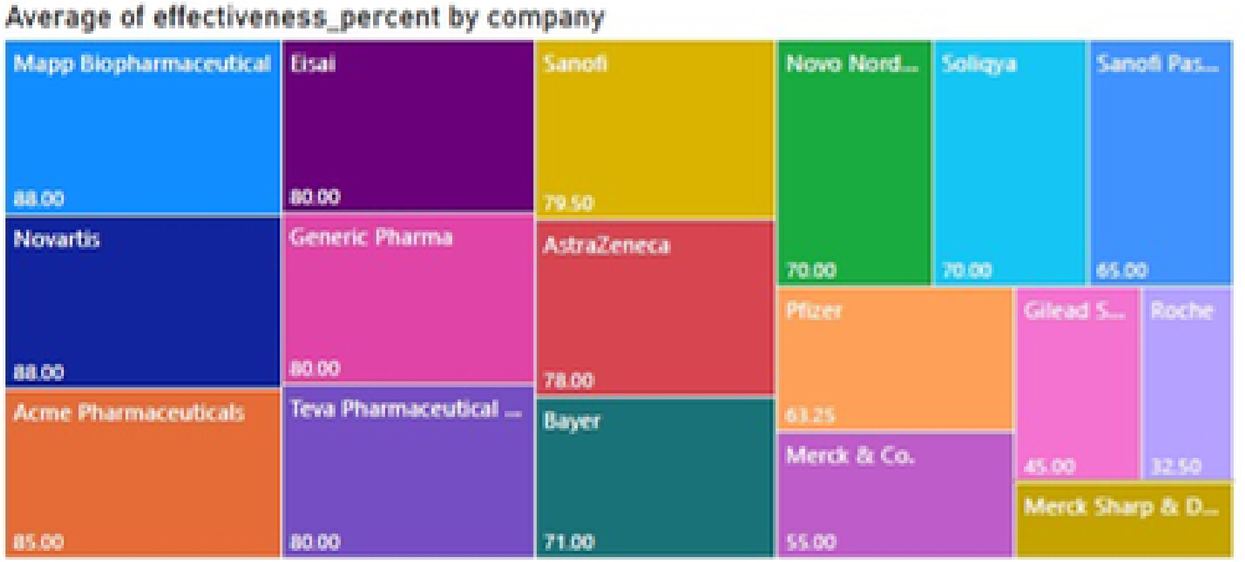
	calculateaverageseverity double precision
1	0.6333333333333333

Power-BI Dashboard - Disease Model









AWS

Architecture Components:

1. **Batch Processing**:

-*Data Source:*

- Batch data can be obtained from various sources like S3 buckets, relational databases, or streaming services.

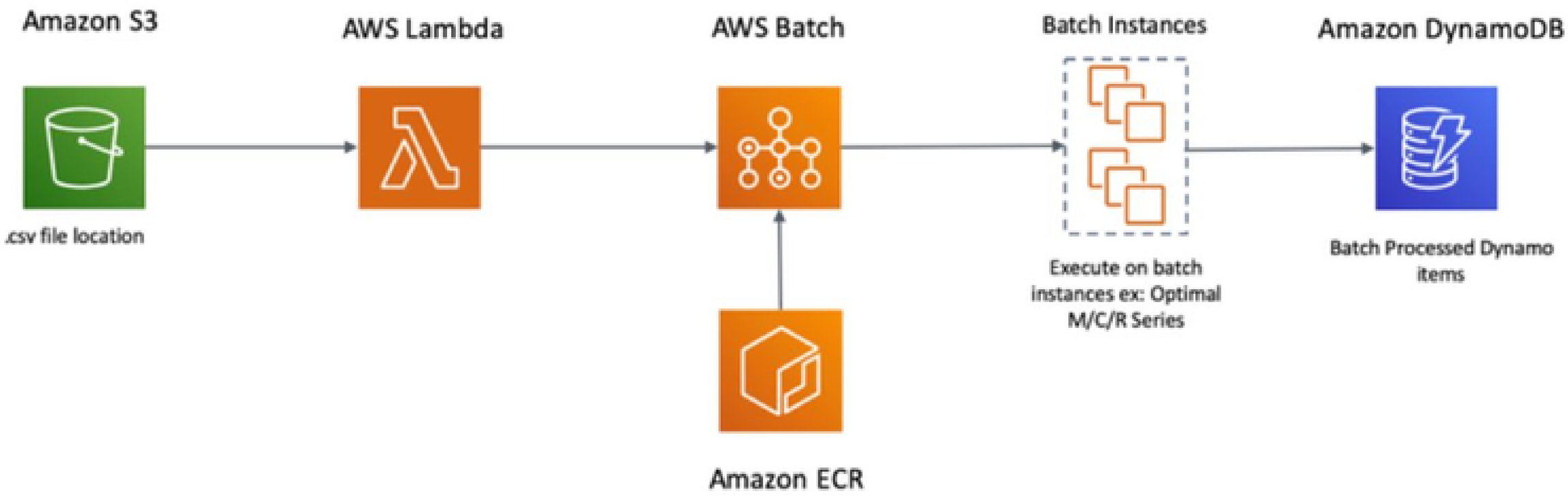
- ***Data Ingestion (AWS Glue)**:*

- Use AWS Glue for ETL (Extract, Transform, Load) processes to prepare and transform data for the data warehouse.

- Schedule Glue jobs to run at specified intervals to process batch data.

- ***Data Warehouse (Amazon Redshift or Snowflake)**:*

- Load the transformed data into a data warehouse, such as Amazon Redshift or Snowflake, for analytics.



Orchestrating an Application Process with AWS Batch

2. Real-time Processing:

- *Data Source:*
  - Real-time data may come from streaming services like Amazon Kinesis or Apache Kafka.

- *Streaming Ingestion (Kinesis Firehose/Kafka to S3):*
  - Use Kinesis Firehose or Kafka to capture and stream real-time data.
  - Store raw data in S3 for further processing.

- *Real-time Processing (Kinesis Analytics):*
  - Utilize Kinesis Analytics to perform real-time analytics on streaming data.
  - Process and aggregate data in near real-time.

- *Data Warehouse:*
  - Load the processed real-time data into the data warehouse

3. **Resilience, High Performance, and Security**:

High Availability (Multi-AZ Deployment):*

- Deploy services across multiple Availability Zones (AZs) for high availability.
- Use services like Amazon RDS, Redshift, or Snowflake that inherently provide high availability.

Auto Scaling:*

- Configure auto-scaling for compute resources to handle varying workloads efficiently.

Security (IAM, VPC, Encryption):*

Implement Identity and Access Management (IAM) for access control.

- Use Virtual Private Cloud (VPC) for network isolation.
- Enable encryption at rest and in transit for sensitive data.

Monitoring and Logging (CloudWatch, CloudTrail):*

- Set up CloudWatch for monitoring and CloudTrail for auditing.

Advantages in Snowflake:

Separation of Storage and Compute:*

Snowflake's architecture separates storage and computing, allowing for independent scaling.

- Offers better elasticity and cost-effectiveness.

Zero Maintenance:*

- Snowflake manages the underlying infrastructure, reducing operational overhead.

NO SQL

I

1. **Document-Oriented Model:**
   - MongoDB is a document-oriented database, which means it stores data in flexible, JSON-like documents. Each record, or document, can have a different structure.
   - Each record in the RACE collection would be a JSON document representing a race.
   - Each record in the LOCATIONS collection would be a JSON document representing a location.
   - Similarly, other tables like MEDICINE, DISEASE_TYPE, PERSON, DISEASE, INDICATION, RACE_DISEASE_PROPENSITY, and DISEASED_PATIENT would be represented as collections with JSON documents.

2. **Embedding and Referencing:**
   - NoSQL databases like MongoDB support both embedding and referencing.
   - we can embed documents within others for improved query performance when the data is frequently read together.
   - For example, information about diseases, indications, and medicines could be embedded within the PERSON document if they are not too large and are frequently accessed together.
   - Alternatively, we could use references to link related documents. For instance, the PERSON document might reference the LOCATIONS and RACE documents by their respective identifiers.

3. Schema Flexibility:
   - MongoDB allows for schema-less or schema-flexible design. You can easily add new fields to documents without affecting existing ones.
   - This flexibility contrasts with the rigid structure of tables in a relational database, allowing for easier adaptation to changing requirements.

4. Indexing:
   - NoSQL databases often use automatic indexing, and MongoDB is no exception. You can create indexes on fields for efficient querying, which can improve performance for specific types of queries.

5. Scalability:
   - NoSQL databases are designed for horizontal scalability, making it easier to distribute data across multiple servers or clusters.
    - This scalability is valuable for handling large amounts of data and high transaction volumes.

6. Query Language:
   - MongoDB uses a rich query language, allowing for complex queries, including geospatial queries. The queries are expressed in JSON-like syntax

Conclusion

Our endeavor to revolutionize disease management through advanced database modeling has been a testament to the power of theory-meeting practice. By meticulously crafting a database solution that not only organizes information but empowers healthcare stakeholders with actionable insights, we've laid the foundation for a more informed and efficient healthcare landscape.


THANK YOU