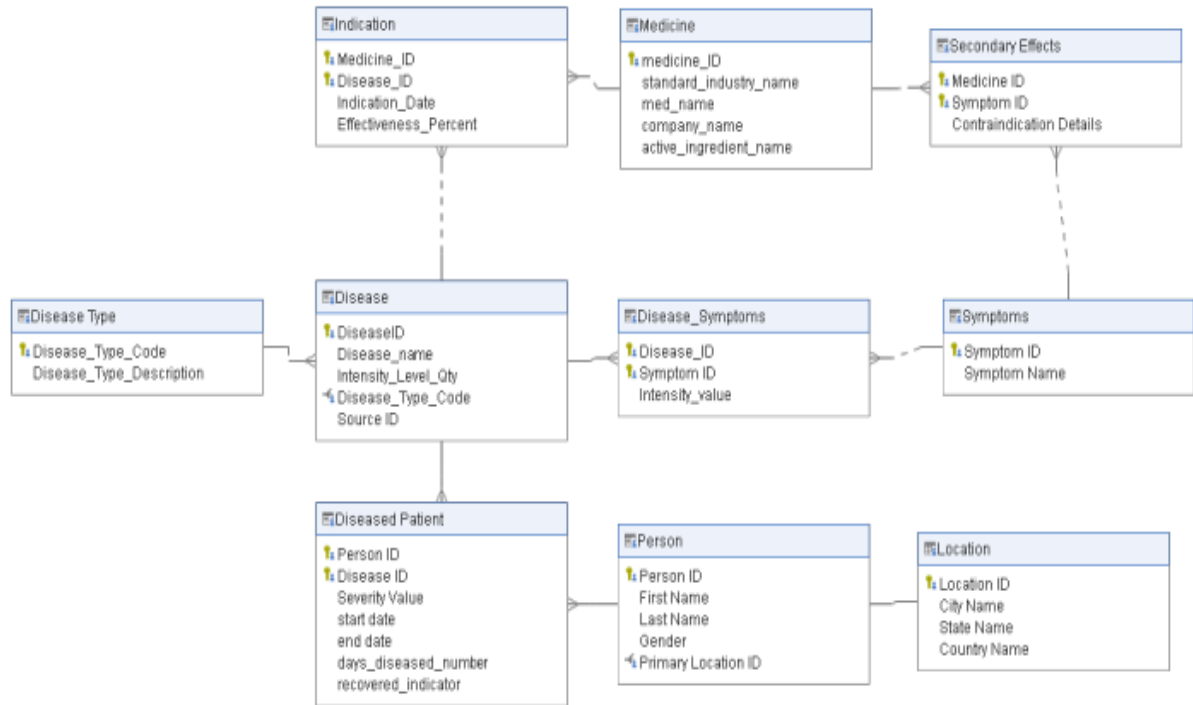


An ER Diagram



A Postgres loaded database

Creating Tables:





1. Disease Table:

	diseaseid [PK] integer	disease_name character varying (50)	intensity_level_qty integer	disease_type_code integer	sourceid integer
1	1	ear pain	5	2	4
2	2	urinary track infection	4	2	6
3	3	skin infection	2	2	7
4	4	bronchitis	5	2	3
5	5	Celiac Disease	2	1	8




2. Disease type Table:

	disease_type_code [PK] integer	disease_type_description character varying (100)
1	1	genetic
2	2	infectious
3	3	deficiency
4	4	psychological

3. disease_symptom Table:

	 diseaseid [PK] integer 	symptomid [PK] integer 	intensity_value integer 
1	1	9	2
2	1	1	5
3	1	2	3
4	2	2	3
5	2	6	1
6	3	13	3
7	3	12	2
8	4	3	5
9	4	8	2
10	4	2	4
11	5	12	3
12	5	10	2
13	5	4	3

4. symptoms Table:

	 symptomid [PK] integer 	symptom_name character varying (50) 
1	1	headache
2	2	fever
3	3	respiratory problem
4	4	diarrhea
5	5	body aches
6	6	severe pain
7	7	hair loss
8	8	sore throat
9	9	dizziness
10	10	vomit
11	11	Constipation
12	12	Skin rash
13	13	dermatitis

5. diseased_patient Table:

	personid [PK] integer	diseaseid [PK] integer	severity_value integer	start_date date	end_date date	days_disease_number integer	recovers_indicator boolean
1	1	2	5	2020-06-25	2020-06-28	3	true
2	2	1	5	2020-06-01	2020-06-05	4	true
3	3	4	5	2020-06-01	[null]	29	false
4	4	1	3	2020-06-12	2020-06-15	3	true
5	5	2	2	2020-06-08	2020-06-11	3	true
6	6	2	5	2020-06-13	2020-06-16	3	true
7	7	3	4	2020-06-21	2020-06-26	5	true
8	8	5	2	2020-06-04	[null]	26	false
9	9	4	5	2020-06-20	[null]	10	false

6. person Table:

	personid [PK] integer	first_name character varying (50)	last_name character varying (50)	gender character varying (2)	primary_locationid integer	age integer
1	1	Jane	Doe	F	2	8
2	2	Lei	Li	F	3	7
3	3	Wei	Li	M	4	8
4	4	Dong	Fang	M	5	6
5	5	Mary	Lu	F	10	5
6	6	Sara	Smith	M	1	5
7	7	Andrea	Johnson	F	6	8
8	8	Louis	Armstrong	M	7	9
9	9	Sujia	Tong	M	4	8

7. locations Table:

	locationid [PK] integer	city_name character varying (50)	state_name character varying (50)	country_name character varying (50)
1	1	New York	NY	United States
2	2	Los Angeles	CA	United States
3	3	Beijing	Beijing	China
4	4	Qingdao	Shandong	China
5	5	Wuhan	Hubei	China
6	6	San Francisco	CA	United States
7	7	Atlanta	GA	United States

8. indication table:

	medicineid [PK] integer	diseaseid [PK] integer	effectiveness_percent double precision
1	1	2	75.7
2	2	1	80.4
3	3	3	98.6
4	4	4	84.5
5	5	5	66.7

9. medicine table:

	medicineid [PK] integer	standard_industry_name character varying (70)	med_name character varying (50)	company_name character varying (50)	active_ingredient_name character varying (50)
1	1	Urinary Pain Relief	AZO UTI	i-Health	Phenazopyridine Hydrochlori...
2	2	Earache Relief Children	SoothEar	Mommys Bliss	Isomalt
3	3	Antibiotic	NA	dicloxacillin	Roche
4	4	Bronchial Asthma Relief Tab...	NA	Walgreens	Ephedrine HCl
5	5	Digestive Health	Probiotic	Lactobacillus rhamnosus GG	Mommys Bliss

10. secondary_effect table:

	medicineid [PK] integer	symptomid [PK] integer	contradiction_details character varying (100)
1	1	4	stomach upsetness may occur
2	3	4	diarrhea may occur
3	4	14	drowsiness may occur
4	2	12	rash may appear
5	5	11	constipation may occur

Creating View

1.

```
CREATE OR REPLACE VIEW cases_info4 AS
SELECT x.effectiveness_percent, n.intensity_value, n.start_date, n.end_date, n.recovers_indicator, n.severity_value,
n.person_id, n.disease_id, n.symptom_id, n.medicine_id, n.primary_location
FROM public.indication x
LEFT JOIN cases_info3 n
ON x.medicineID=n.medicineID
and x.diseaseID=n.diseaseID
```

	effectiveness_percent double precision	intensity_value integer	start_date date	end_date date	recovers_indic... boolean	severity_value... integer	person_id integer	diseaseid integer	symptomid integer	medicineid integer	primary_locatio... integer
1	75.7	3	2020-06-25	2020-06-28	true	5	1	2	2	1	2
2	75.7	1	2020-06-25	2020-06-28	true	5	1	2	6	1	2
3	80.4	5	2020-06-01	2020-06-05	true	5	2	1	1	2	3
4	80.4	3	2020-06-01	2020-06-05	true	5	2	1	2	2	3
5	80.4	2	2020-06-01	2020-06-05	true	5	2	1	9	2	3
6	84.5	4	2020-06-01	[null]	false	5	3	4	2	4	4
7	84.5	5	2020-06-01	[null]	false	5	3	4	3	4	4
8	84.5	2	2020-06-01	[null]	false	5	3	4	8	4	4

.....

22	84.5	5	2020-06-20	[null]	false	5	9	4	3	4	4
23	84.5	2	2020-06-20	[null]	false	5	9	4	8	4	4
24	66.7	3	2020-06-03	[null]	false	2	10	5	4	5	8
25	66.7	2	2020-06-03	[null]	false	2	10	5	10	5	8
26	66.7	3	2020-06-03	[null]	false	2	10	5	12	5	8

2.

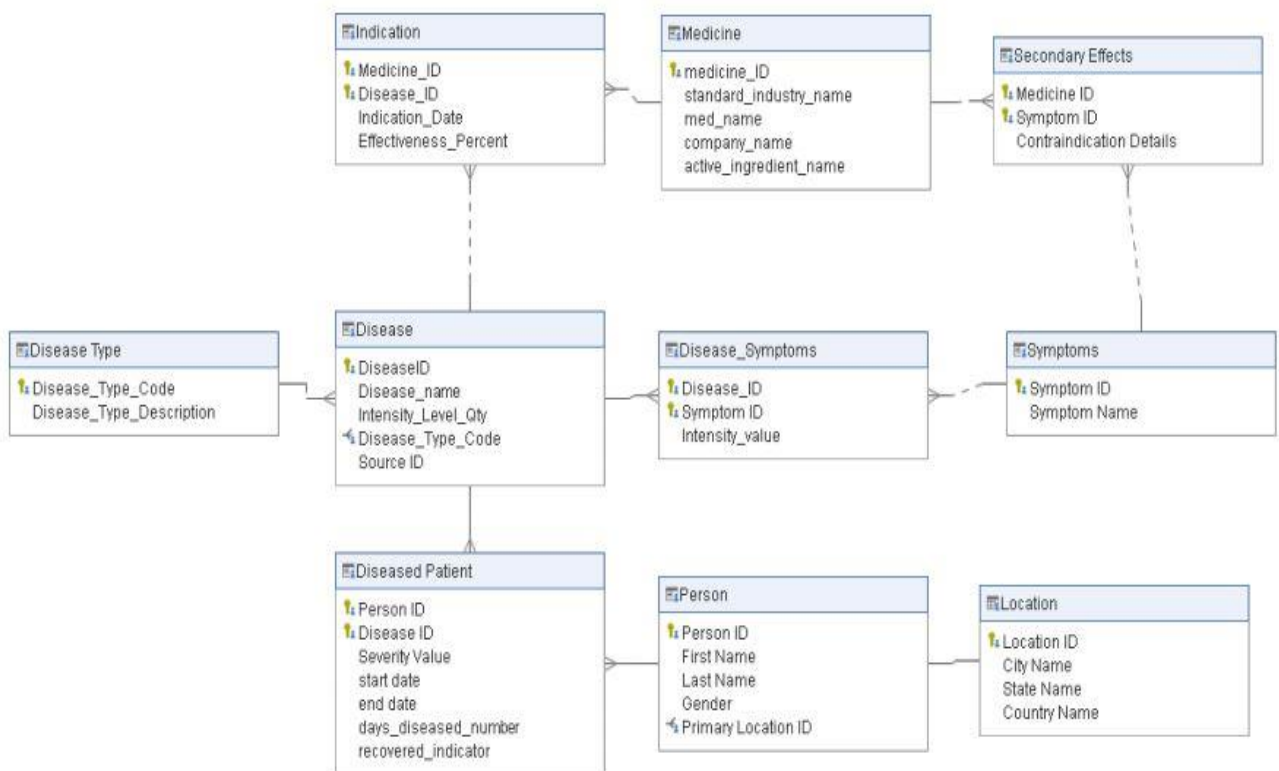
```

CREATE OR REPLACE VIEW cases_info AS
SELECT DISTINCT a.diseaseID, a.symptomID, b.disease_type_code, c.personID, d.medicineID, e.primary_locationID
FROM public.disease_symptom a, public.disease b, public.diseased_patient c, public.indication d, public.person e
where a.diseaseID=b.diseaseID
AND a.diseaseID=c.diseaseID
AND a.diseaseID=d.diseaseID
AND c.personID=e.personID
order by c.personID
SELECT * FROM cases_info

```

	diseaseid integer	symptomid integer	disease_type_code integer	personid integer	medicineid integer	primary_locationid integer
1	2	2	2	2	1	2
2	2	6	2	2	1	2
3	1	1	2	2	2	3
4	1	2	2	2	2	3
5	1	9	2	2	2	3
6	4	2	2	2	3	4
7	4	3	2	2	3	4
8	4	8	2	2	3	4
9	1	1	2	2	4	5
10	1	2	2	2	4	5
11	1	9	2	2	4	5
12	2	2	2	2	5	10
13	2	6	2	2	5	10

The Dimensional Model



Data Warehouse

Creating a data warehouse called: disease_dw

Creating disease table and insert value

```
CREATE TABLE disease_dw.disease(  
diseaseID serial primary key,  
disease_name varchar(50),  
intensity_level_qty int,  
disease_type_code int,  
sourceID int);  
  
INSERT INTO disease_dw.disease  
(SELECT * FROM public.disease);
```

	diseaseid [PK] integer	disease_name character varying (50)	intensity_level_qty integer	disease_type_code integer	sourceid integer
1	1	ear pain	5	2	4
2	2	urinary track infection	4	2	6
3	3	skin infection	2	2	7
4	4	bronchitis	5	2	3
5	5	Celiac Disease	2	1	8

Other tables will use same method to create and data will move from Public SCHEMA.

Creating fact table:

```
CREATE TABLE disease_dw.cases_dis_fact
```

```
(caseID SERIAL PRIMARY KEY,
```

```
start_date date,
```

```
end_date date,
```

```
recovered_indicator boolean,
```

```
symptoms_intensity_value int,
```

```
indication_effectiveness int,
```

```
severity int,
```

```
diseaseID int references disease_dw.disease (diseaseID),
```

```
disease_type_code int references disease_dw.disease_type (disease_type_code ),
medicineID int references disease_dw.medicine (medicineID),
symptomID int references disease_dw.symptoms (symptomID),
personID int references disease_dw.person (personID),
locationID int references disease_dw.locations (locationID)
ON delete CASCADE);
```

Insert for fact table

```
INSERT INTO disease_dw.cases_dis_fact (start_date,end_date, recovered_indicator,
symptoms_intensity_value,indication_effectiveness, severity, diseaseID , disease_type_code,
medicineID,symptomID, personID, locationID)
```

```
(SELECT DISTINCT f.start_date,f.end_date,
f.recovers_indicator,f.intensity_value,f.effectiveness_percent, f.severity_value,
a.diseaseID, d.disease_type_code,c.medicineID, b.symptomID, e.personID,
e.primary_locationID
```

```
FROM disease_dw.disease a, disease_dw.symptoms b, disease_dw.person e,
disease_dw.medicine c,disease_dw.disease_type d,public.cases_info4 f
```

```
WHERE a.diseaseID=f.diseaseID
```

```
AND b.symptomID=f.symptomID
```

```
AND c.medicineID=f.medicineID
```

```
AND d.disease_type_code=f.disease_type_code
```

```
AND e.personID=f.personID
```

```
AND e.primary_locationID=f.primary_locationID);
```

	Data Output	Explain	Messages	Notifications								
	caseid [PK] integer	start_date date	end_date date	recovered_indicator boolean	symptoms_intensity_value integer	indication_effectiveness integer	severity integer	disea integ				
1		1 2020-06-01	2020-06-05	true		2	80	5				
2		2 2020-06-01	2020-06-05	true		3	80	5				
3		3 2020-06-01	2020-06-05	true		5	80	5				
4		4 2020-06-01	[null]	false		2	84	5				
5		5 2020-06-01	[null]	false		4	84	5				
6		6 2020-06-01	[null]	false		5	84	5				
7		7 2020-06-03	[null]	false		2	67	2				
8		8 2020-06-03	[null]	false		3	67	2				
9		9 2020-06-03	[null]	false		3	67	2				

Testing the referential integrity:

1. On update CASCADE / delete cascade:

Deleting a disease from the database Delete where disease id=3 from disease and now on disease was deleted.

```
SELECT * FROM disease_dw.cases_dis_fact

delete from disease where diseaseid = '5';
```

	caseid [PK] integer	start_date date	end_date date	recovered_indicator boolean	symptoms_intensity_value integer	indication_effectiveness integer	severity integer	diseaseid integer	disease_type. integer
1	1	2020-06-01	2020-06-05	true	2	80	5	1	
2	2	2020-06-01	2020-06-05	true	3	80	5	1	
3	3	2020-06-01	2020-06-05	true	5	80	5	1	
4	4	2020-06-01	[null]	false	2	84	5	4	
5	5	2020-06-01	[null]	false	4	84	5	4	
6	6	2020-06-01	[null]	false	5	84	5	4	

Update a disease from the database Delete where disease id=5 from disease and now on disease id is 6.

```
update disease set diseaseid = '6' where diseaseid = '5'
```

	medicineid [PK] integer	diseaseid [PK] integer	effectiveness_percent double precision
1	1	2	75.7
2	2	1	80.4
3	3	3	98.6
4	4	4	84.5
5	5	6	66.7

Business analytics on the data:

1. effectiveness_percent vs. disease vs. intensity_value vs medicine

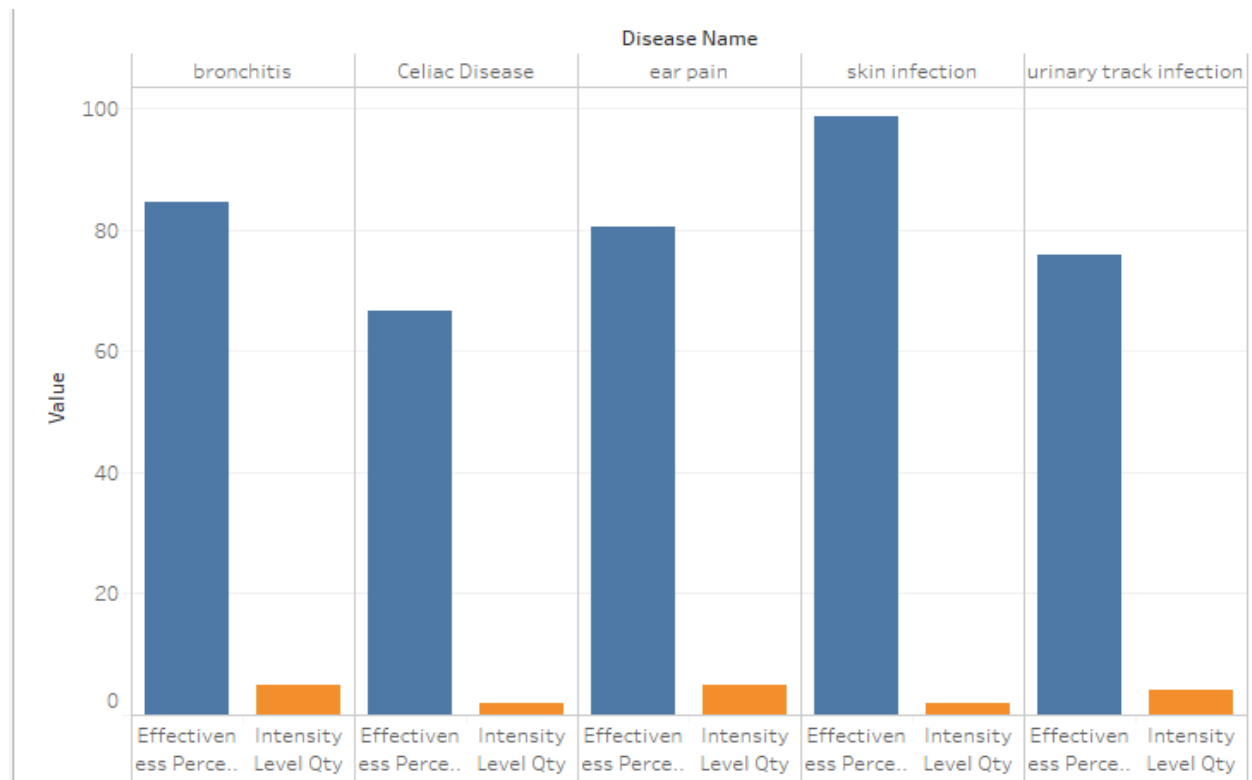
```
1  -----1. effectiveness_percent vs. disease vs. intensity_value vs medicine
2  CREATE OR REPLACE VIEW effectiveness AS
3  SELECT x.effectiveness_percent, n.intensity_value,n.severity_value, n.disease_type_code,n.diseaseID, n.medicineID
4  FROM public.indication x
5  LEFT JOIN cases_info3 n
6  ON x.medicineID=n.medicineID
7  and x.diseaseID=n.diseaseID
8
9  ---The result and analysis
10 SELECT effectiveness_percent, intensity_value, disease_type_code, diseaseID,medicineID
11 FROM effectiveness
12 ORDER BY effectiveness_percent desc;
13
14 SELECT AVG(effectiveness_percent) AS AVG_effectiveness_percent, AVG(intensity_value) as AVG_intensity_value
15 FROM effectiveness
16
```


	effectiveness_percent double precision	intensity_value integer	disease_type_code integer	diseaseid integer	medicineid integer
1	98.6	3	2	3	3
2	98.6	2	2	3	3
3	84.5	5	2	4	4
4	84.5	4	2	4	4
5	84.5	5	2	4	4
6	84.5	2	2	4	4
7	84.5	4	2	4	4
8	84.5	2	2	4	4
9	80.4	5	2	1	2

.....

22	66.7	2	1	5
23	66.7	3	1	5
24	66.7	3	1	5
25	66.7	2	1	5
26	66.7	3	1	5

Using Tableau:



The average:

Data Output	Explain	Messages	Notifications
avg_effectiveness_percent double precision	avg_intensity_value numeric		
1	78.50000000000001	2.8846153846153846	

From above analysis, the most effectiveness percent is 98.6%, and they are all belong to medicine 3, and they are all used for disease type 2. With its low intensity value and high effectiveness percent, we can make a conclusion that disease 3 will be the disease that is not very risky and dangerous for patients. On the other hand, the least effectiveness percent is 66.7%, and they are all belong to medicine 5, and they are all used for disease 1. With its low effectiveness percent, we can make a conclusion that disease 1 will be the disease that is hard to find the most effective treatment for patients.

Also, the average of effectiveness percent is 78.5%, which mean most of the medicine and treatment could help patients to recover and release their pain. And the average intensity value is 2.88, which means the disease in this dataset are not that risky for the most patients.

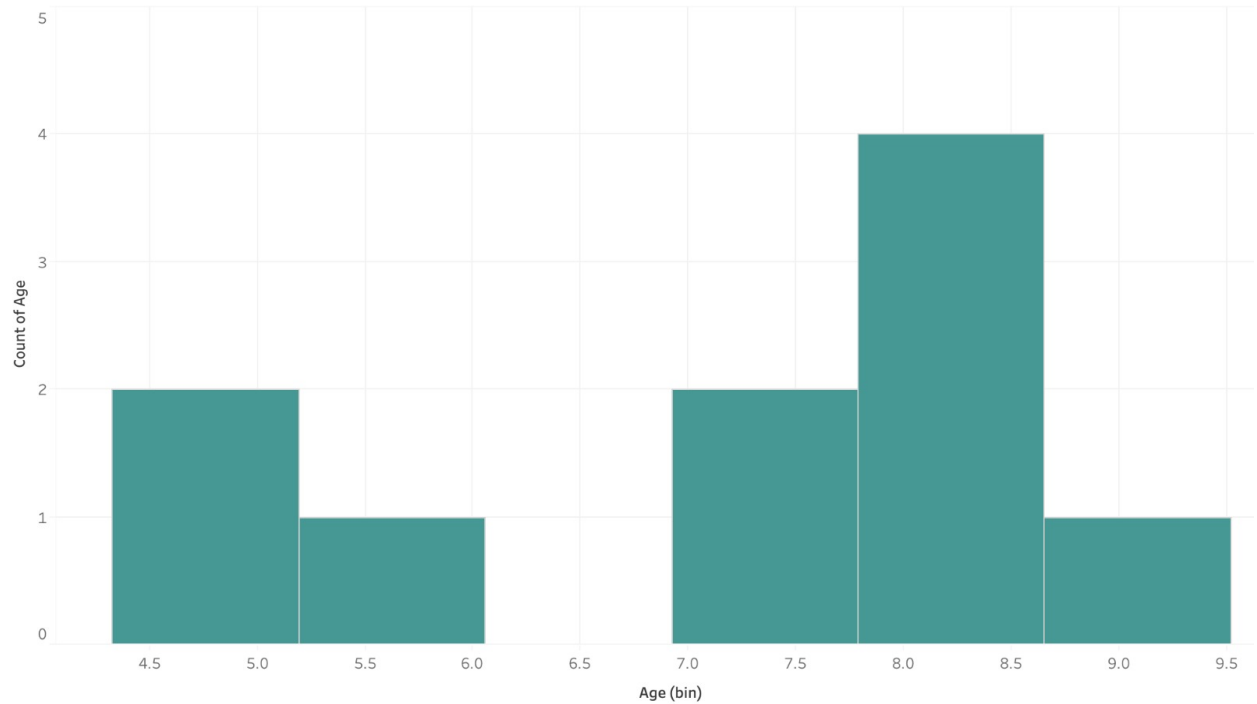
2. Analyzing the number of patient and their age for each disease:

```
19 SELECT Disease.Disease_name,COUNT(diseased_patient.personID) AS Number_Of_Patients,person.age FROM diseased_patient
20 LEFT JOIN Disease ON Disease.diseaseID = diseased_patient.diseaseID
21 LEFT JOIN person ON diseased_patient.personID = person.personID
22 GROUP BY Disease_name,person.age
```

The result:

	Data Output	Explain	Messages	Notifications
	disease_name character varying (50)	number_of_patients bigint	age integer	
1	Celiac Disease	1	9	
2	ear pain	1	6	
3	urinary track infection	2	5	
4	ear pain	1	7	
5	bronchitis	2	8	
6	urinary track infection	1	8	
7	skin infection	1	8	
8	Celiac Disease	1	7	

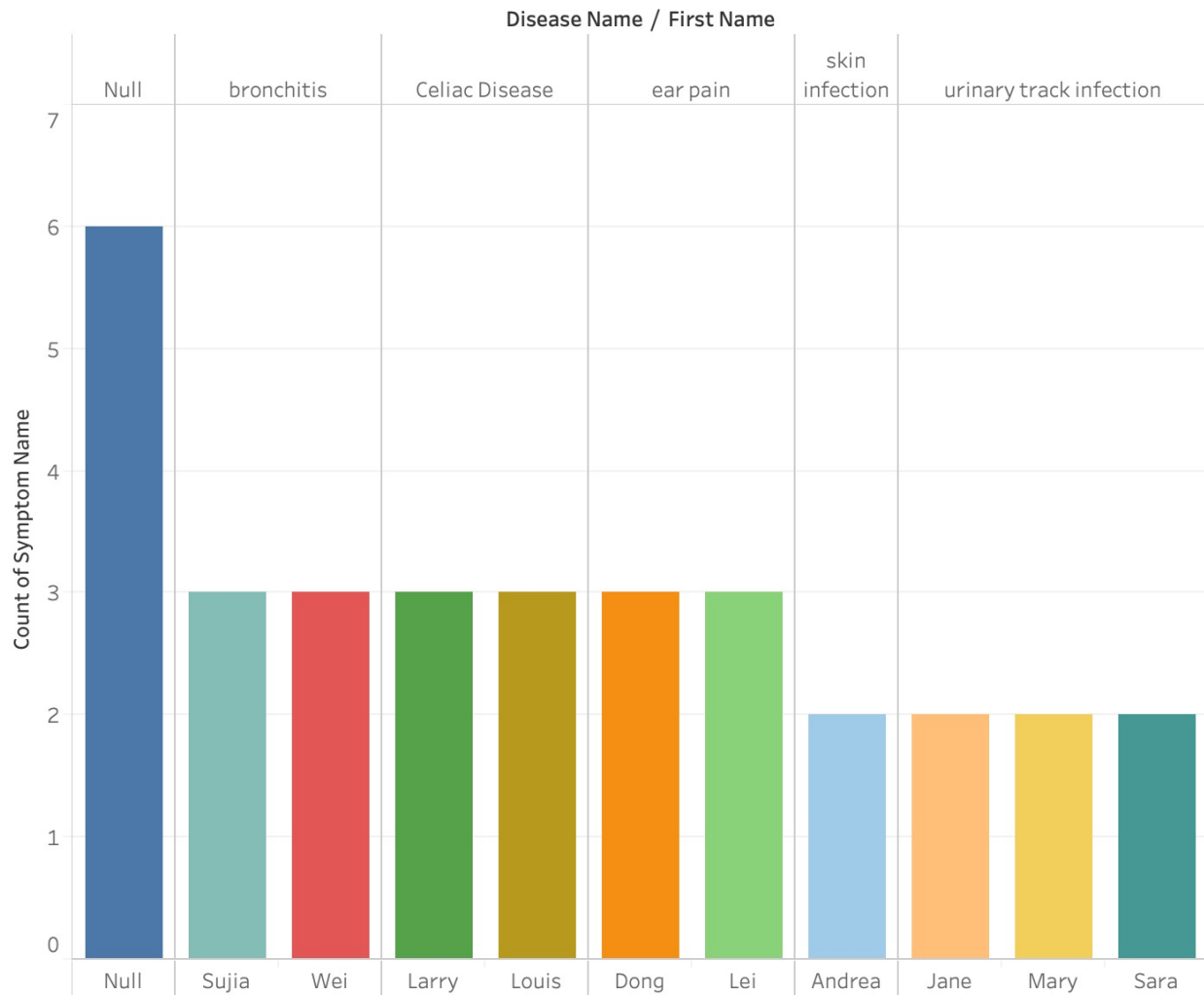
Using Tableau:



From above analysis, urinary track infection and bronchitis has two patients who have been diseased, and other diseases only have on patient, which means urinary track infection and bronchitis might has higher infection rate compare to others, and based on the patients' age, all these diseases might infect between age group from 5 to 9, which are children. As researchers, they could pay more attention at this age group to decrease the infection rate and improve medicine treatment for specific age group.

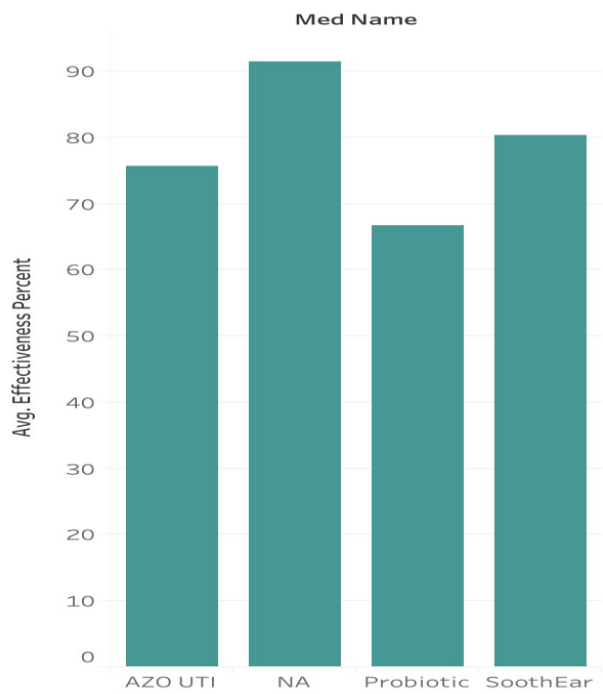
More business analytics by using Tableau:

1.



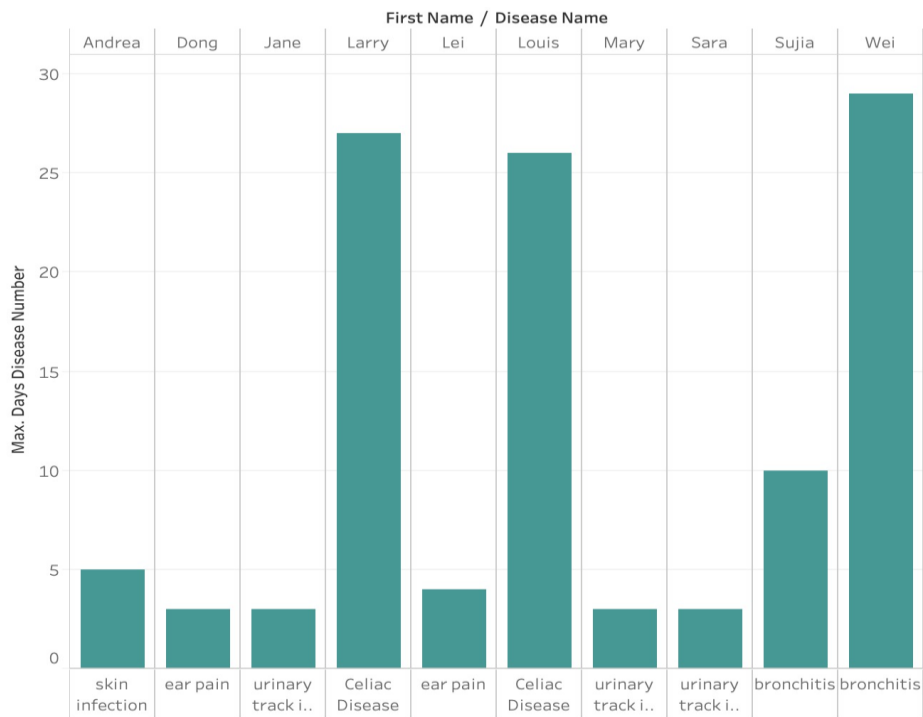
Based on the graph, we can see the distribution of the disease by patients, also the count of the symptom by diseases.

2.



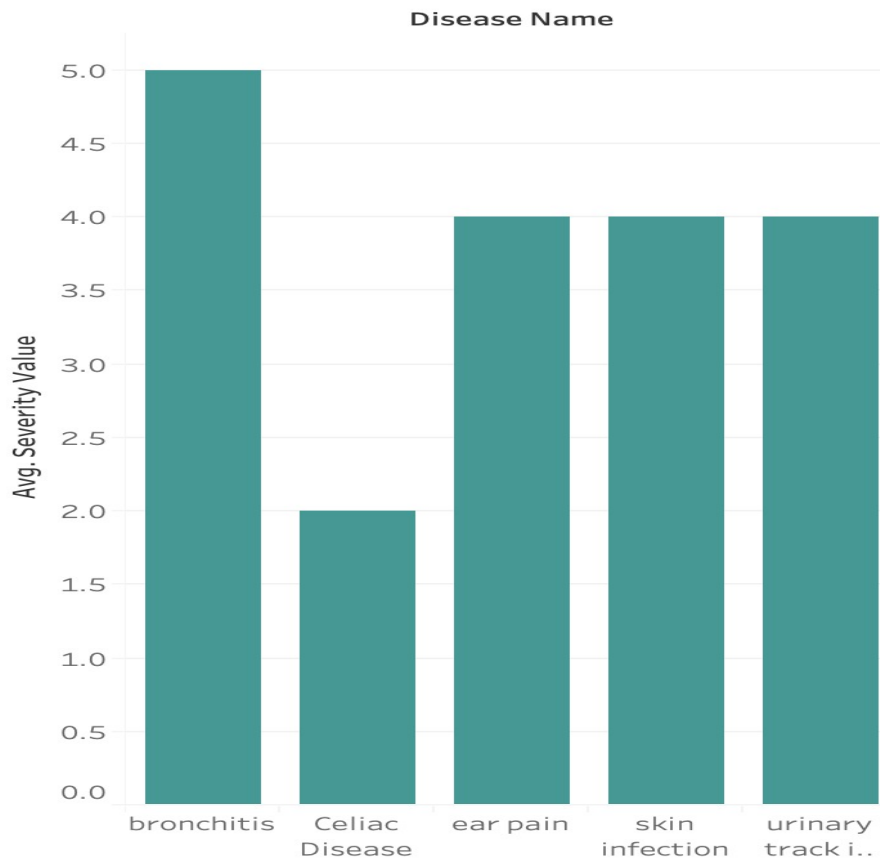
As the graph shows, it has the average effectiveness percent for each medicine.

3.



As the graph shows, it has the distribution of how many days the patient has been diseased. Bronchitis has the max number of days patient has been diseased.

4.



This graph shows the average severity value of each disease; as it shows, bronchitis has the highest average severity value.

Explain how the structure of your database would be different in a specific NO SQL, non-relational database (e.g. MongoDB, Neo4J).

Relational databases are table-based. NoSQL databases can be document based, graph databases, key-value pairs, or wide-column stores. The SQL database could show the relationships between each table, such as disease and patient, medicine, and indication. However, the NOSQL database does not include relationships. To make our model work in NOSQL, we would use key-value to have large amounts of data with simple lookup queries, such as our patients' information and medicines' information. Using wide-column to store large amounts of data with predictable query patterns, and using the graph to analyze relationships between connected data, such as disease and patient.

According to Neo4J, Neo4J is a graphing database, which shows the relationships between data as equally important to the data itself. It is intended to hold data without constricting it to a pre-defined model, so we might find some interesting relationship between each attribute that relational databases cannot find. For example, the patient's location with the disease.

Architecture and Process (including AWS Software and products) to house in AWS in a Resilient, Secure, Performant manner.

Since it is a disease model, I would give access to the database for doctors, researchers, and hospitals/clinics.

Doctors and hospitals could update a patient's information and data to their database systems, but researchers need to have the authorization to have the access to the information for security reasons. Also, the more doctors look up a patient's case, the higher chance of better treatment this patient could get to core his disease. Therefore, doctors have access to other doctors' patients if they get approved by the doctors and the patients.

For a public, people do not have the access to the database unless the doctor or hospital release a certain patient's data and it needs to be approved by the patient.

There are the infrastructures that satisfied the above requirement:

Using Amazon Relational Database Service to build our database, which provides cost-efficient and resizable capacity while automating time-consuming administration tasks such as hardware provisioning, database setup, patching, and backups could help doctors and hospitals build their database fast and easily with fast performance, high availability, and high security. For doctors, hospitals, and researchers, we will use a network address translation gateway to check if the user has access to the database, otherwise, they will only have the access to the public database.

Also, using Amazon ElastiCache to seamlessly set up, run, and scale popular open-source compatible in-memory data stores in the cloud, which doctors and hospitals could operate patients' data. For public and private servers, we can use Amazon's Elastic Compute Cloud for them to access the data.

Last but not least, we could use Amazon Elastic File System to have file-level security, which allows only certain doctors and certain people could have full access to the datasets and edit the datasets.

Reference:

1. <https://www.mongodb.com/nosql-explained/nosql-vs-sql>
2. <https://neo4j.com/product/neo4j-graph-database/>
3. <https://aws.amazon.com/efs/>
4. <https://aws.amazon.com/elasticache/>
5. <https://aws.amazon.com/rds/>
6. <https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway.html>
7. <https://www.fool.com/investing/general/2013/12/29/the-5-most-common-infectious-diseases.aspx>