Clinical Trial Database Management System

Student Name: Nirmalkumar Thirupallikrishnan Kesavan

Executive Summary:

This project aims to design and implement a comprehensive Clinical Trial Database System to manage patient data, clinical trials, treatments, and outcomes. The system integrates both relational (MySQL) and NoSQL (MongoDB) databases to handle structured and unstructured data. Key tables and relationships have been modeled to represent patients, treatments, clinical trials, researchers, suppliers, products, and outcomes. The system supports advanced queries to extract meaningful insights such as treatment success rates, patient participation, and trial outcomes. Additionally, Python-based visualizations are used to provide reporting capabilities and track trial progress, while enabling the analysis of key performance metrics. This project demonstrates how integrated data management can enhance clinical trial processes and support real-time decision-making.

I. Introduction

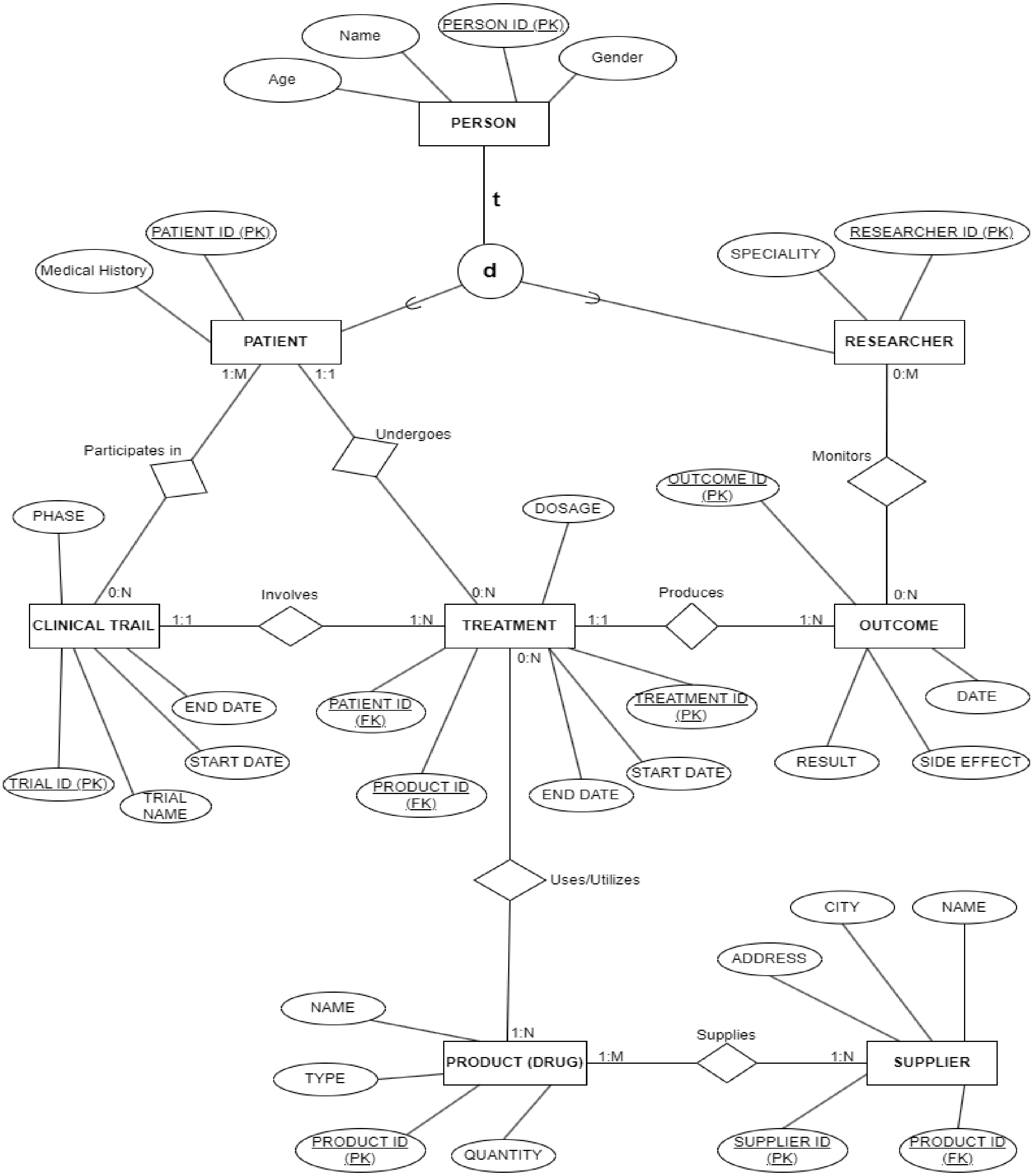
The primary objective of this project is to develop a comprehensive Clinical Trial Data Management System that centralizes trial information, integrates data, and provides real-time analytics and visualization capabilities. This system aims to empower healthcare professionals and researchers with accurate and easily accessible data for improved decision-making during clinical trials.

The benefits of this solution are multifold. By automating manual processes, the system minimizes errors and significantly enhances efficiency. The inclusion of interactive dashboards allows researchers to analyze treatment effectiveness and patient outcomes seamlessly, enabling evidence-based decisions. Furthermore, the streamlined management of trial data accelerates drug development timelines, reducing the time-to-market for new therapies and associated costs. Ultimately, this system facilitates better treatment outcomes, supporting the healthcare industry in delivering more effective solutions to patients..

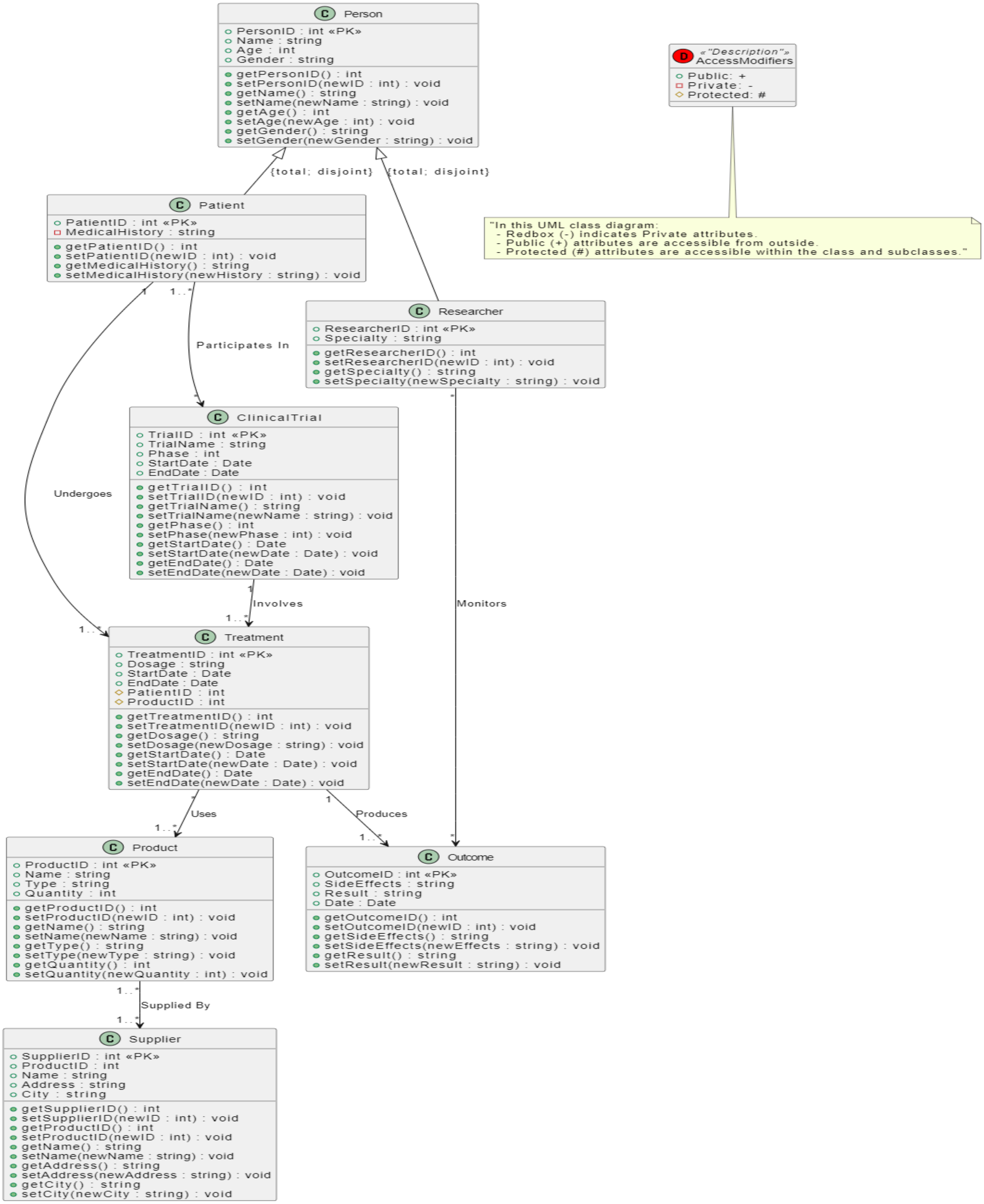
Inefficiencies in Data Management cause delays in medication development, higher expenses, and perhaps inaccurate data in clinical trials. Many healthcare organizations rely on disorganized systems that store data in several places, which makes it challenging to monitor patient outcomes, and assess the efficacy of treatments. The administration of clinical trials is additionally made more difficult by manual procedures and clumsy data entry techniques, which slows down the time it takes to introduce new therapies and products to the market. By creating a centralized clinical trial data management system we can overcome these problems. The technology will increase clinical trial efficiency overall, lower mistake rates, and fasten the time it takes to bring therapies to market by automating data entry procedures, enabling real-time access to clinical trial information, and supplying visualization capabilities.

II. Conceptual Data Modeling

1. EER Diagram



2. UML Diagram



III. Mapping Conceptual Model to Relational Model

Primary Key- Underlined *Foreign Key- Italicized*

PERSON (Person ID, Age, Name , Gender)
– Person ID is the Primary Key
– Person ID should be NOT NULL
– Age should be NOT NULL
– Name should be NOT NULL
– Gender should be NOT NULL

PATIENT (*Patient ID*, Medical History)
– Patient ID is the Primary Key and also a Foreign Key referring to Person ID from the relation PERSON
– Patient ID should be NOT NULL

TREATMENT (Treatment ID, Start date, End date, Dosage, *Patient ID*, *Trial ID*)
– Treatment ID is the Primary Key
– Treatment ID should be NOT NULL
– Patient ID is a Foreign Key referring to the Patient ID from the relation PATIENT
– Patient ID should be NOT NULL
– Trial ID is a Foreign Key referring to Trail ID from the relation CLINICAL TRIAL
– Trial ID should be NOT NULL

CLINICAL TRIAL (Trail ID, Trail Name, Start date, End date, Phase)
– Trial ID is the Primary Key
– Trial ID should be NOT NULL

PARTICIPATES_IN (*Trial ID, Patient ID*)
– Trial ID is a Foreign Key referring to Trail ID from the relation CLINICAL TRIAL
– Trial ID should be NOT NULL
– Patient ID is a Foreign Key referring to the Patient ID from the relation PATIENT
– Patient ID should be NOT NULL

PRODUCT (Product ID, Quantity, Name, Type)
– Product ID is the Primary Key
– Product ID should be NOT NULL

SUPPLIER (Supplier ID, Name, City, Address)
– Supplier ID is the Primary Key
– Supplier ID should be NOT NULL

SUPPLIES (*Supplier ID, Product ID*)
– Supplier ID is a Foreign Key referring to Supplier ID from the relation SUPPLIER
– Supplier ID should be NOT NULL
– Product ID is a Foreign Key referring to the Product ID from the relation PRODUCT
– Product ID should be NOT NULL

USES (*Product ID, Treatment ID*)
– Product ID is a Foreign Key referring to the Product ID from the relation PRODUCT
– Product ID should be NOT NULL
– Treatment ID is a Foreign Key referring to Treatment ID from the relation TREATMENT
– Treatment ID should be NOT NULL

OUTCOME (Outcome ID, *Treatment ID*, Result, Date, Side effect)
– Outcome ID is the Primary Key
– Outcome ID should be NOT NULL
– Treatment ID is a Foreign Key referring to Treatment ID from the relation TREATMENT
– Treatment ID should be NOT NULL

RESEARCHER (*Researcher ID*, Speciality)
– Researcher ID is the Primary Key and also a Foreign Key referring to Person ID from the relation PERSON
– Researcher ID should be NOT NULL

MONITORS (*Researcher ID, Outcome ID*)
– Researcher ID is a Foreign Key referring to the Researcher ID from the relation RESEARCHER
– Researcher ID should be NOT NULL
– Outcome ID is a Foreign Key referring to Treatment ID from the relation OUTCOME
– Outcome ID should be NOT NULL

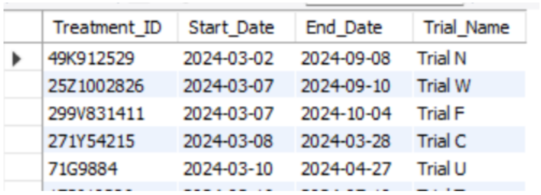
IV. Implementation of Relation Model via MySQL and NoSQL

MySQL Implementation:

The database was created in MySQL and the following queries were performed:

Simple Query: Retrieve Treatments with Start Dates after March 1, 2024, Including Trial Names

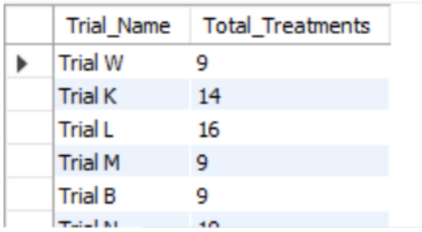
```
SELECT T.Treatment_ID, T.Start_Date, T.End_Date,
CT.Trial_Name
FROM TREATMENT T
JOIN CLINICAL_TRIAL CT ON T.Trial_ID = CT.Trial_ID
WHERE T.Start_Date > '2024-03-01'
ORDER BY T.Start_Date ASC;
```



Treatment_ID	Start_Date	End_Date	Trial_Name
49K912529	2024-03-02	2024-09-08	Trial N
25Z1002826	2024-03-07	2024-09-10	Trial W
299V831411	2024-03-07	2024-10-04	Trial F
271Y54215	2024-03-08	2024-03-28	Trial C
71G9884	2024-03-10	2024-04-27	Trial U

Aggregate Query: Count Total Treatments for Each Clinical Trial

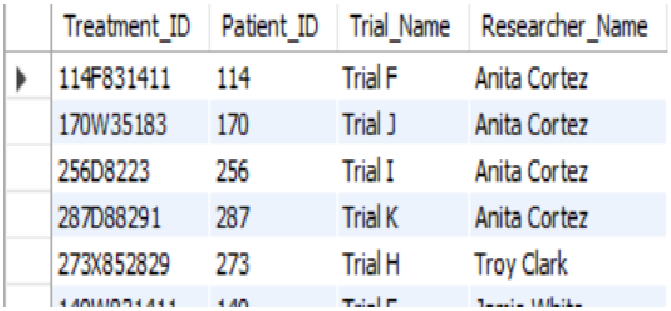
```
SELECT CT.Trial_Name, COUNT(T.Treatment_ID) AS
Total_Treatments
FROM CLINICAL_TRIAL CT
JOIN TREATMENT T ON CT.Trial_ID = T.Trial_ID
GROUP BY CT.Trial_Name;
```



Trial_Name	Total_Treatments
Trial W	9
Trial K	14
Trial L	16
Trial M	9
Trial B	9

Inner Join Query: Retrieve the details of treatments, including Treatment_ID, Patient_ID, Trial_Name, and the Researcher monitoring the outcomes.

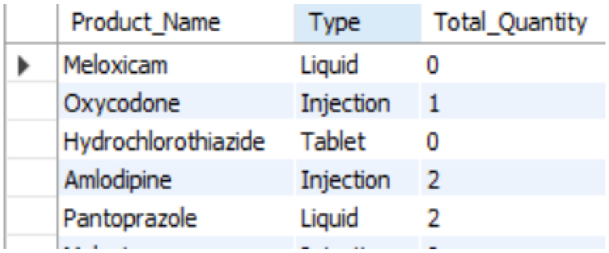
```
SELECT  T.Treatment_ID, T.Patient_ID, CT.Trial_Name,
P.Name AS Researcher_Name
FROM TREATMENT T
JOIN CLINICAL_TRIAL CT ON T.Trial_ID = CT.Trial_ID
JOIN OUTCOME O ON T.Treatment_ID = O.Treatment_ID
JOIN MONITORS M ON O.Outcome_ID = M.Outcome_ID
JOIN RESEARCHER R ON M.Researcher_ID =
R.Researcher_ID
JOIN PERSON P ON R.Researcher_ID = P.Person_ID;
```



Treatment_ID	Patient_ID	Trial_Name	Researcher_Name
114F831411	114	Trial F	Anita Cortez
170W35183	170	Trial J	Anita Cortez
256D8223	256	Trial I	Anita Cortez
287D88291	287	Trial K	Anita Cortez
273X852829	273	Trial H	Troy Clark

Outer Join Query: Retrieve Product_Name, Type, and Total_Quantity used in treatments, including products that have not been used in any treatments.

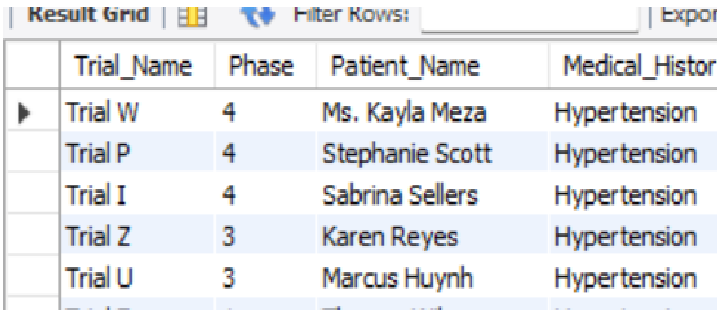
```
SELECT PR.Name AS Product_Name, PR.Type,
SUM(U.Product_ID IS NOT NULL) AS Total_Quantity
FROM PRODUCT PR
LEFT JOIN USES U ON PR.Product_ID = U.Product_ID
GROUP BY PR.Product_ID;
```



Product_Name	Type	Total_Quantity
Meloxicam	Liquid	0
Oxycodone	Injection	1
Hydrochlorothiazide	Tablet	0
Amlodipine	Injection	2
Pantoprazole	Liquid	2

Nested Query 1: Retrieve Trial_name, Phase, and Patients with Hypertension as Medical History

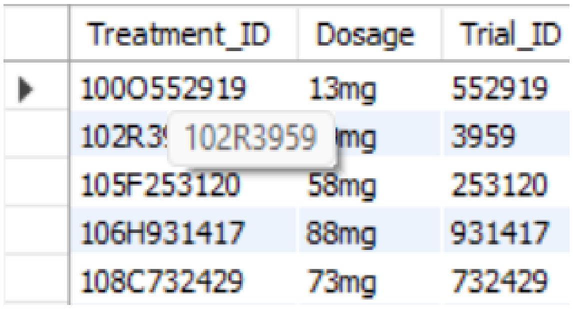
```
SELECT CT.Trial_Name,
       CT.Phase,
       Subquery.Name AS Patient_Name,
       Subquery.Medical_History
FROM CLINICAL_TRIAL CT
JOIN PARTICIPATES_IN PI ON
CT.Trial_ID = PI.Trial_ID
JOIN (
    SELECT PT.Patient_ID, PERSON.Name,
PT.Medical_History
    FROM PATIENT PT
    JOIN PERSON ON PT.Patient_ID =
PERSON.Person_ID
    WHERE PT.Medical_History = 'Hypertension'
) Subquery ON
PI.Patient_ID = Subquery.Patient_ID;
```



Trial_Name	Phase	Patient_Name	Medical_Histor
Trial W	4	Ms. Kayla Meza	Hypertension
Trial P	4	Stephanie Scott	Hypertension
Trial I	4	Sabrina Sellers	Hypertension
Trial Z	3	Karen Reyes	Hypertension
Trial U	3	Marcus Huynh	Hypertension

Correlated Query: Retrieve Treatment_ID with Dosages above the Average Dosage per trial.

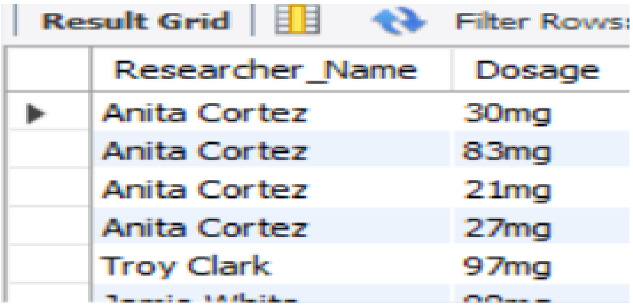
```
SELECT T.Treatment_ID, T.Dosage, T.Trial_ID
FROM TREATMENT T
WHERE CAST(SUBSTRING_INDEX(T.Dosage, 'mg', 1)
AS UNSIGNED) >
     (SELECT AVG(CAST(SUBSTRING_INDEX(Dosage,
'mg', 1) AS UNSIGNED))
       FROM TREATMENT WHERE Trial_ID = T.Trial_ID);
```



Treatment_ID	Dosage	Trial_ID
100O552919	13mg	552919
102R3959	mg	3959
105F253120	58mg	253120
106H931417	88mg	931417
108C732429	73mg	732429

Correlated Query: Retrieve Treatment_ID with Dosages above the Average Dosage per trial.

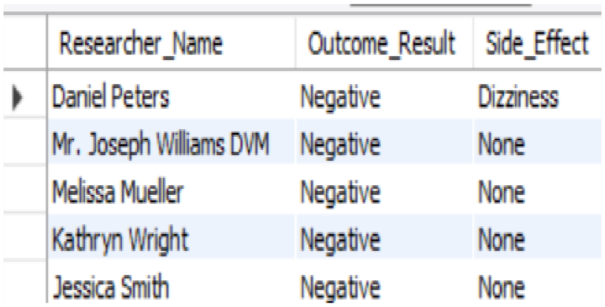
```
SELECT DISTINCT P.Name AS Researcher_Name, T.Dosage
FROM RESEARCHER R
JOIN PERSON P ON R.Researcher_ID = P.Person_ID JOIN
MONITORS M ON R.Researcher_ID = M.Researcher_ID
JOIN OUTCOME O ON M.Outcome_ID = O.Outcome_ID
JOIN TREATMENT T ON O.Treatment_ID = T.Treatment_ID
WHERE CAST(SUBSTRING_INDEX(T.Dosage, 'mg', 1) AS
UNSIGNED) > ANY (
   SELECT CAST(SUBSTRING_INDEX(T2.Dosage, 'mg', 1)
AS UNSIGNED)
   FROM TREATMENT T2
   JOIN CLINICAL_TRIAL CT ON T2.Trial_ID = CT.Trial_ID
   WHERE CT.Phase = 3
```



Researcher_Name	Dosage
Anita Cortez	30mg
Anita Cortez	83mg
Anita Cortez	21mg
Anita Cortez	27mg
Troy Clark	97mg

Exists Query: Retrieve all researchers who have monitored outcomes associated with treatments with "Negative" results.

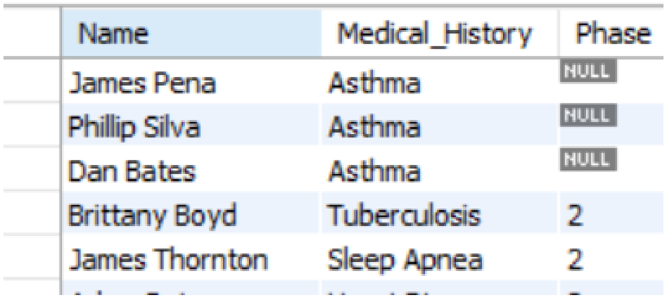
```
SELECT DISTINCT P.Name AS Researcher_Name,
            O.Result AS Outcome_Result,
            O.Side_Effect
FROM RESEARCHER R
JOIN PERSON P ON R.Researcher_ID = P.Person_ID
JOIN MONITORS M ON R.Researcher_ID = M.Researcher_ID
JOIN OUTCOME O ON M.Outcome_ID = O.Outcome_ID
WHERE EXISTS (
   SELECT 1
   FROM OUTCOME O_Sub
   WHERE O_Sub.Outcome_ID = O.Outcome_ID
     AND O_Sub.Result = 'Negative');
```



Researcher_Name	Outcome_Result	Side_Effect
Daniel Peters	Negative	Dizziness
Mr. Joseph Williams DVM	Negative	None
Melissa Mueller	Negative	None
Kathryn Wright	Negative	None
Jessica Smith	Negative	None

Set Operations (Union) Query: Retrieve the names of patients who have a medical history of "Asthma" or have participated in a "Phase 2" clinical trial.

```
SELECT P.Name, PT.Medical_History, NULL AS Phase
FROM PERSON P
JOIN PATIENT PT ON P.Person_ID = PT.Patient_ID  --
Adjusted join to use Patient_ID in PATIENT
WHERE PT.Medical_History = 'Asthma'
UNION
SELECT P.Name, PT.Medical_History, CT.Phase
FROM PERSON P
JOIN PATIENT PT ON P.Person_ID = PT.Patient_ID  --
Adjusted join to use Patient_ID in PATIENT
JOIN PARTICIPATES_IN PI ON PT.Patient_ID = PI.Patient_ID
JOIN CLINICAL_TRIAL CT ON PI.Trial_ID = CT.Trial_ID
WHERE CT.Phase = 2;
```

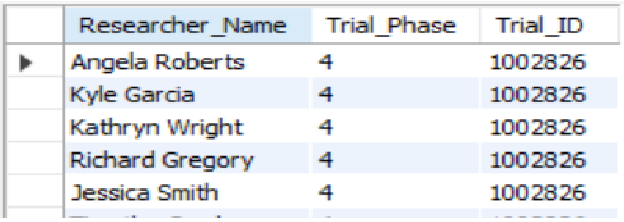


Name	Medical_History	Phase
James Pena	Asthma	NULL
Phillip Silva	Asthma	NULL
Dan Bates	Asthma	NULL
Brittany Boyd	Tuberculosis	2
James Thornton	Sleep Apnea	2

Subquery: Find all researchers who are monitoring outcomes from the trial with the maximum phase.

```
SELECT DISTINCT P.Name AS Researcher_Name,  CT.Phase AS Trial_Phase,  CT.Trial_ID
```

```
FROM RESEARCHER R
JOIN PERSON P ON R.Researcher_ID = P.Person_ID
JOIN MONITORS M ON R.Researcher_ID = M.Researcher_ID
JOIN OUTCOME O ON M.Outcome_ID = O.Outcome_ID
JOIN TREATMENT T ON O.Treatment_ID = T.Treatment_ID
JOIN CLINICAL_TRIAL CT ON T.Trial_ID IN (
    SELECT Trial_ID
    FROM CLINICAL_TRIAL
    WHERE Phase = (SELECT MAX(Phase) FROM
CLINICAL_TRIAL));
```



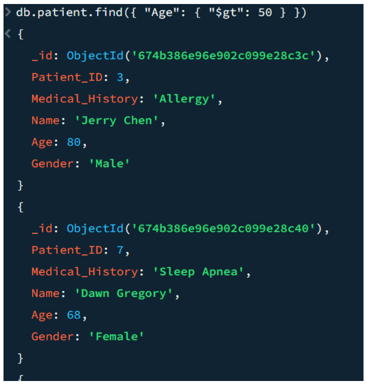
	Researcher_Name	Trial_Phase	Trial_ID
▶	Angela Roberts	4	1002826
	Kyle Garcia	4	1002826
	Kathryn Wright	4	1002826
	Richard Gregory	4	1002826
	Jessica Smith	4	1002826

NoSQL Implementation:

All the tables (clinical_trial, monitors, uses, participates_in, outcome, patient, person, researcher, product, supplier, supplies, uses) created in MySQL have been imported into MongoDB Atlas and the queries are executed using MongoDB Compass. The following NoSQL queries were done:

Query 1: Retrieving Patients with Age Greater Than 50

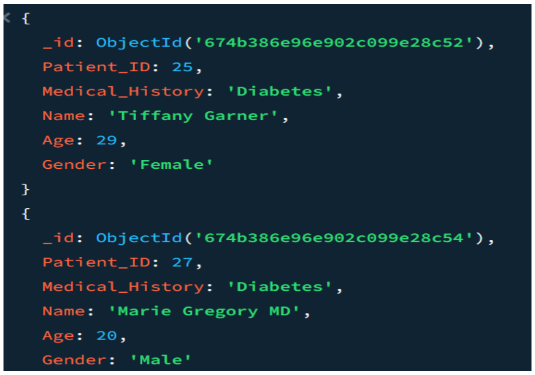
```
db.patient.find({ "Age": { "$gt": 50 } })
```



```
> db.patient.find({ "Age": { "$gt": 50 } })
< {
    _id: ObjectId('674b386e96e902c099e28c3c'),
    Patient_ID: 3,
    Medical_History: 'Allergy',
    Name: 'Jerry Chen',
    Age: 80,
    Gender: 'Male'
  }
  {
    _id: ObjectId('674b386e96e902c099e28c40'),
    Patient_ID: 7,
    Medical_History: 'Sleep Apnea',
    Name: 'Dawn Gregory',
    Age: 68,
    Gender: 'Female'
  }
```

Query 2: Retrieving Patients with Either Hypertension or Diabetes in Medical History

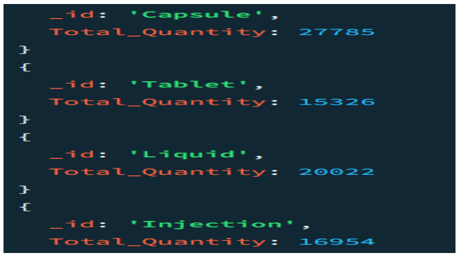
```
db.patient.find({
"$or": [
        { "Medical_History": "Hypertension" },
            { "Medical_History": "Diabetes" }
    ] } )
```



```
< {
    _id: ObjectId('674b386e96e902c099e28c52'),
    Patient_ID: 25,
    Medical_History: 'Diabetes',
    Name: 'Tiffany Garner',
    Age: 29,
    Gender: 'Female'
  }
  {
    _id: ObjectId('674b386e96e902c099e28c54'),
    Patient_ID: 27,
    Medical_History: 'Diabetes',
    Name: 'Marie Gregory MD',
    Age: 20,
    Gender: 'Male'
```

Query 3: Retrieving Total Quantity of Products Grouped by Product Type

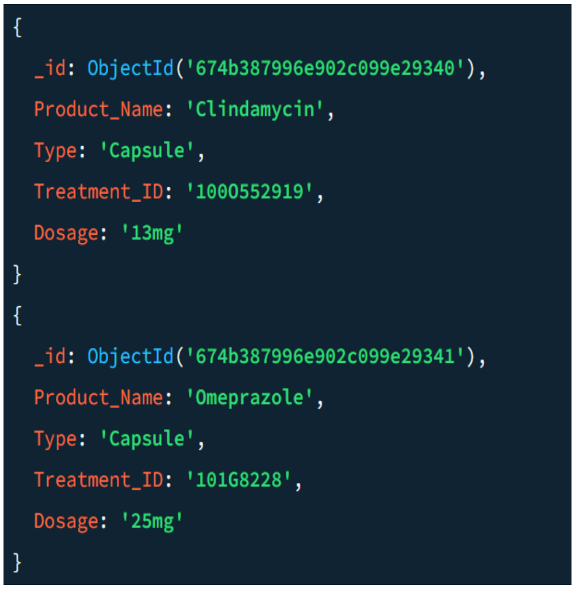
```
db.product.aggregate([
  {
    "$group": {
      "_id": "$Type",
      "Total_Quantity": { "$sum": "$Quantity" }
    }
  } ])
```



```
    _id: 'Capsule',
    Total_Quantity: 27785
  }
  {
    _id: 'Tablet',
    Total_Quantity: 15326
  }
  {
    _id: 'Liquid',
    Total_Quantity: 20022
  }
  {
    _id: 'Injection',
    Total_Quantity: 16954
```

Query 4: Retrieving Product Details with Treatment Information Including Dosage and Type

```
db.uses.aggregate([
  {
    "$lookup": {
      "from": "product",
      "localField": "Product_ID",
      "foreignField": "Product_ID",
      "as": "product_details"
    }
  },
  { "$unwind": "$product_details" },
  {
    "$lookup": {
      "from": "treatment",
      "localField": "Treatment_ID",
      "foreignField": "Treatment_ID",
      "as": "treatment_details"
    }
  },
  { "$unwind": "$treatment_details" },
  {
    "$project": {
      "Product_Name": "$product_details.Name",
      "Type": "$product_details.Type",
      "Treatment_ID": "$treatment_details.Treatment_ID",
      "Dosage": "$treatment_details.Dosage"  } } ] )
```



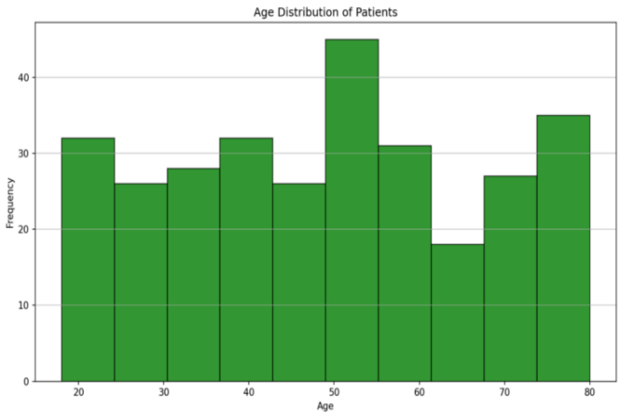
```
{
  _id: ObjectId('674b387996e902c099e29340'),
  Product_Name: 'Clindamycin',
  Type: 'Capsule',
  Treatment_ID: '1000552919',
  Dosage: '13mg'
}
{
  _id: ObjectId('674b387996e902c099e29341'),
  Product_Name: 'Omeprazole',
  Type: 'Capsule',
  Treatment_ID: '101G8228',
  Dosage: '25mg'
}
```

V. Database Access via Python

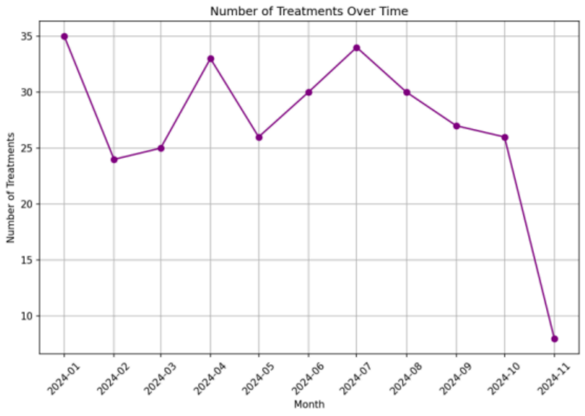
We used the mysql.connector library to establish a connection between Python and the MySQL database clinicaltrailnirmal by providing the database credentials (host, user, password, and database name). A cursor object was created to execute SQL queries, and the fetchall() method was used to retrieve the query results. We then used the tabulate library to format and display the results in a readable table format. Finally, we closed both the cursor and the database connection to ensure proper resource management.

We performed various visualizations to analyze the data, leveraging a combination of basic and advanced charts to uncover relationships and distributions across different variables. For this, we utilized Matplotlib, a powerful library for creating static, interactive, and animated visualizations, and Seaborn, which builds on Matplotlib to provide a high-level interface for creating aesthetically pleasing and informative statistical graphics. Additionally,we used mpl_toolkits.mplot3d from Matplotlib to generate 3D scatter plots for multidimensional data visualization.

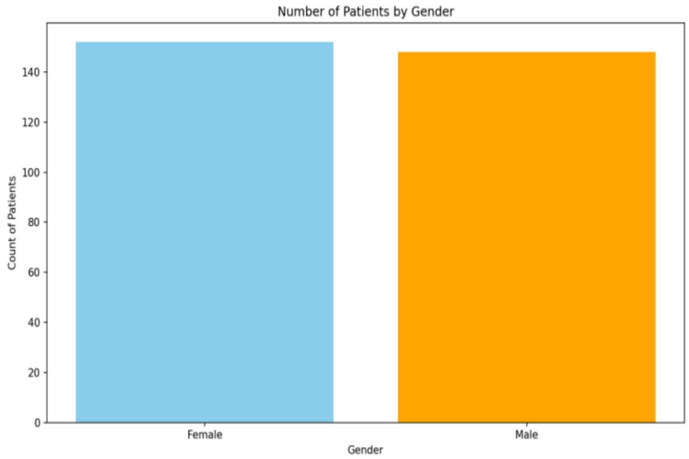
Age Distribution of Patients



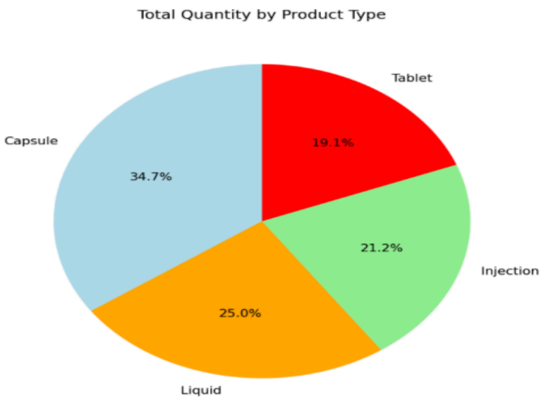
Number of Treatments Over Time



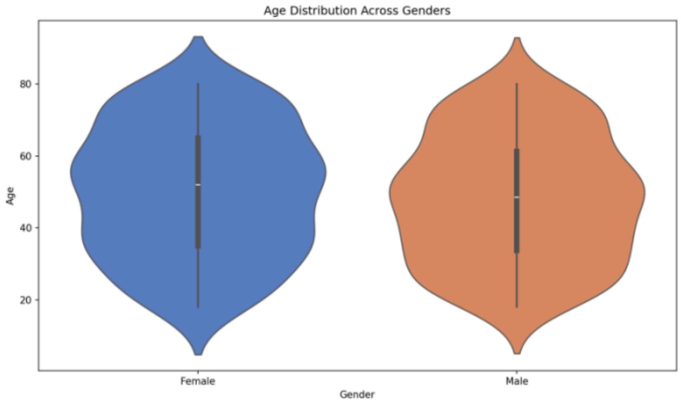
Number of Patients by Gender



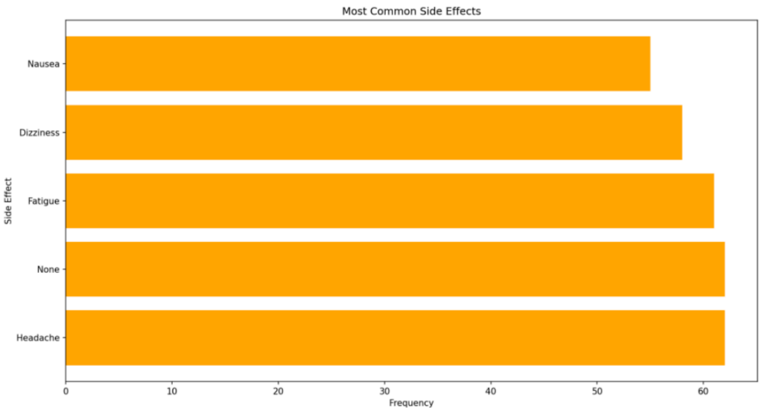
Total Quantity by Product Type



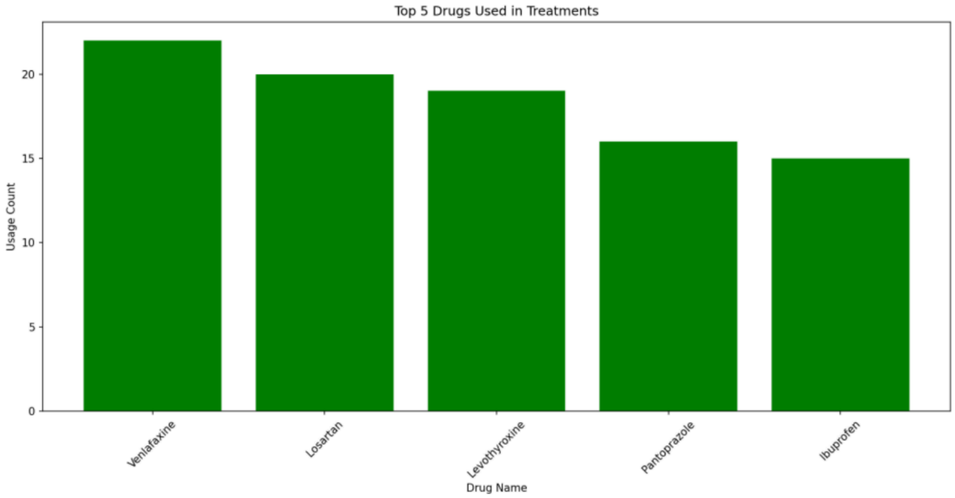
Age Distribution Across Genders



Most Common Side Effects



Top 5 drugs used in treatments



VII. Summary and Recommendation:

The Clinical Trial Database System successfully integrates relational and NoSQL databases to manage the complexity of clinical trial data. It offers a scalable and flexible platform for recording and analyzing critical trial-related information, including patient treatments, outcomes, and overall trial progress. By combining MySQL and MongoDB, the system efficiently handles both structured and unstructured data, enabling complex querying and real-time data analysis. Python-based visualizations are employed to generate insightful reports, showcasing patient demographics, treatment effectiveness, and trial status over time.

To further enhance its functionality, future updates could incorporate real-time data integration for continuous trial monitoring, along with the implementation of strong data governance mechanisms to ensure security and regulatory compliance. Additionally, the system could benefit from the integration of advanced analytics tools, such as predictive modeling, to improve decision-making and forecast trial outcomes. Expanding scalability to accommodate larger, multi-center trials and collaborating with external research tools could also optimize the system for broader use in clinical research. With these enhancements, the system would provide even more powerful support for clinical trial management, improving efficiency and decision-making in the healthcare industry.