Medical laboratory system database

Name	ID
Jomana El Gammal	231000050
Nour Menshawy	231002265
Batoul Tamer	231000822

Introduction

This project consists P NNof a structured Mysql relational schema and a comprehensive medical laboratory database system. A database for a medical laboratory system was created and put into use by us. The system makes it easier to store laboratory, patient, medical test, test result, and test component (materials) data.

Conceptual design

Entities and Attributes

The medical laboratory system consists of the following main entities along with their attributes:

- **Laboratorian**: Represents the staff who perform medical tests.
  Attributes:
  - ID (Primary Key)
  - Name
  - Phone number
  - Address
- **Patient**: Represents individuals who undergo medical tests.
  Attributes:
  - PatientID (Primary Key)
  - Name
  - Phone number
  - Address
  - Birth Date
  - Job

- **Component**: Materials or supplies used in conducting medical tests.
  Attributes:
  - ComponentID (Primary Key)
  - Name
  - Available Quantity
  - Minimum Quantity
- **Medical Test**: The tests available in the laboratory.
  Attributes:
  - TestID (Primary Key)
  - Name
  - Price
- **Test Result**: Stores the outcome of tests performed on patients.
  Attributes:
  - TestID (Foreign Key referencing Medical Test)
  - Date (Date test was performed)
  - PatientID (Foreign Key referencing Patient)
  - LaboratorianID (Foreign Key referencing Laboratorian)
  - Result (Test outcome)

Cardinality Ratios and Participation Constraints

In the medical laboratory system, the relationships between entities are defined with specific cardinality ratios and participation constraints to accurately model the real-world scenario.

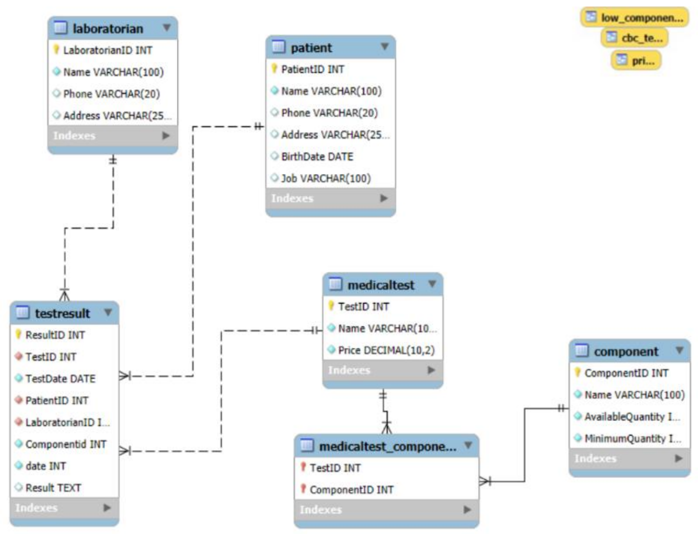
1. **Patient to Test Result**
   - **Cardinality:** One-to-many (1:N)
     Each patient can have zero or more test results because a patient may undergo multiple medical tests over time.
   - **Participation:** Partial participation on Patient side, total participation on Test Result side
     Not every patient necessarily has a test result recorded at all times, but every test result must be associated with exactly one patient.
2. **Laboratorian to Test Result**
   - **Cardinality:** One-to-many (1:N)
     Each laboratorian can perform many tests, resulting in multiple test results associated with them.
   - **Participation:** Partial participation on Laboratorian side, total participation on Test Result side
     Some laboratorians might not have performed tests yet, but every test result must be linked to one laboratorian.
3. **Medical Test to Test Result**
   - **Cardinality:** One-to-many (1:N)
     Each medical test can be conducted multiple times, producing multiple test results.
   - **Participation:** Partial participation on Medical Test side, total participation on Test Result side

Not all medical tests may have been performed yet, but every test result must refer to exactly one medical test.

4. **Medical Test to Component**
   - **Cardinality:** Many-to-many (M:N), simplified here as One-to-many (1:N) for design clarity
     Each medical test requires one or more components, and components can be used in multiple tests. For simplicity, in this design, each test is associated with one primary component.

     Participation: Partial participation on both sides
     Some components may not be currently used in any tests, and some tests may use components not explicitly modeled here

Relational schema

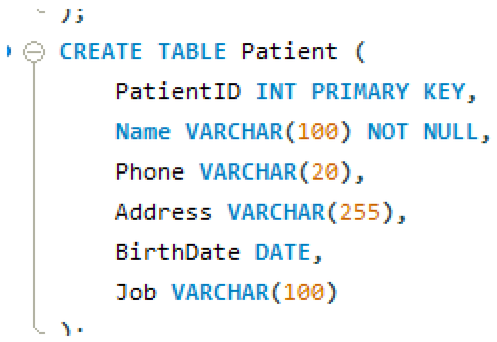


-Each entity is converted to a table with attributes and primary keys. Foreign keys ensure referential integrity between related tables. For example, `Medical_Test` references `Component`, and `Test_Result` references `Patient`, `Laboratorian`, and `Medical_Test`.

-

Queries with Screenshots :

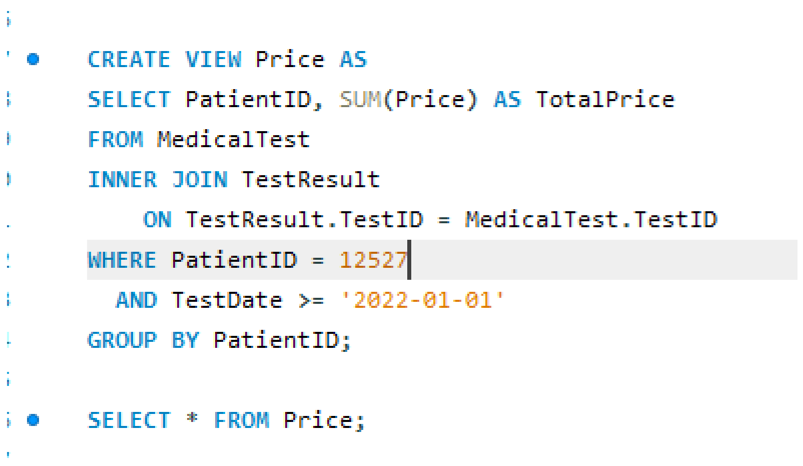
- About schema



```
);
CREATE TABLE Patient (
    PatientID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Phone VARCHAR(20),
    Address VARCHAR(255),
    BirthDate DATE,
    Job VARCHAR(100)
);
```

Each entity is converted to a table with attributes and primary keys. Foreign keys ensure referential integrity between related tables. For example, `Medical_Test` references `Component`, and `Test_Result` references `Patient`, `Laboratorian`, and `Medical_Test`.

-. List the total money paid by a patient with ID 12527 in the last three years

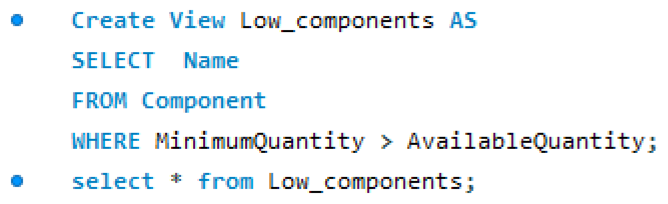


```
CREATE VIEW Price AS
SELECT PatientID, SUM(Price) AS TotalPrice
FROM MedicalTest
INNER JOIN TestResult
    ON TestResult.TestID = MedicalTest.TestID
WHERE PatientID = 12527
  AND TestDate >= '2022-01-01'
GROUP BY PatientID;

SELECT * FROM Price;
```

This view calculates the total amount of money paid by the patient with `PatientID = 12527` for all tests taken since the beginning of 2022 by summing up the `Price` from the `Medical_Test` table.

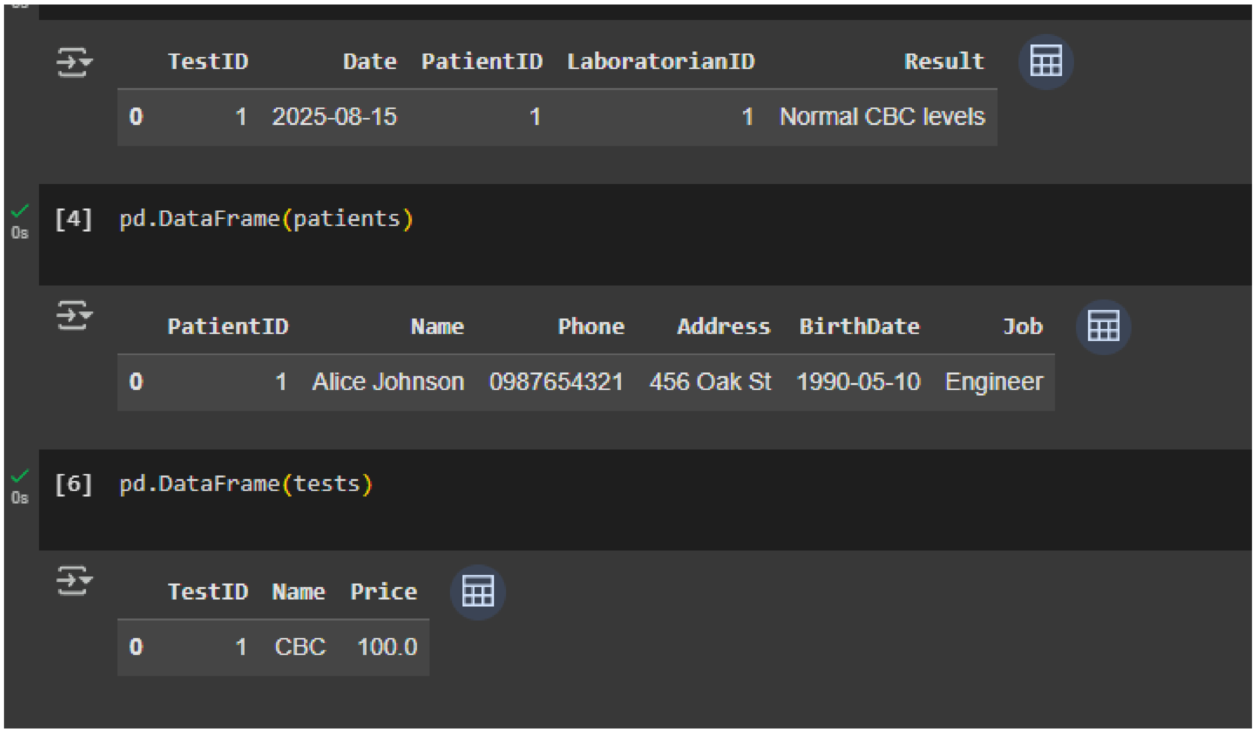
. List the names of components that are below the minimum quantity



```
Create View Low_components AS
SELECT  Name
FROM Component
WHERE MinimumQuantity > AvailableQuantity;
select * from Low_components;
```

This query checks which test components have an `AvailableQuantity` that is less than the `MinimumQuantity`, indicating that those components need to be restocked.

UI in database:



```
   TestID        Date  PatientID  LaboratorianID             Result
0       1  2025-08-15          1               1  Normal CBC levels
```

```
[4] pd.DataFrame(patients)
```

```
   PatientID           Name       Phone     Address   BirthDate       Job
0          1  Alice Johnson  0987654321  456 Oak St  1990-05-10  Engineer
```

```
[6] pd.DataFrame(tests)
```

```
   TestID  Name  Price
0       1   CBC  100.0
```

https://colab.research.google.com/drive/1NL-lyV1zWq-pBKeO3HozFb3X7TGkzNfm?usp=sharing

An interactive interface (e.g., in Google Colab or a web app) can be built to add/view data dynamically, making the system user-friendly .