**Test Case 1:** Can a patient have different visits with the same diagnosis?

**Member:** Rebecca Chu

**Data:** Inserted data for a PATIENT and created 2 visits for them that had different diagnosis from 2 different service providers. Had to add bits of info to lots of different tables due to foreign key constraints (PATIENT, DEPARTMENT, EMPLOYEE, ADDITIONAL\_STAFF, NURSE, DIAGNOSIS, VISIT, LOGS).

**SQL Query:**

#### SELECT v.PID AS PatientID, d.DIAGNOSISID AS DiagnosisID, COUNT(DISTINCT v.VISITID) AS VisitCount

#### FROM VISIT v, LOGS l, DIAGNOSIS d

#### WHERE v.SERVICEPROVIDERID = l.EMPLOYEEID

#### AND l.DIAGNOSISID = d.DIAGNOSISID

#### GROUP BY v.PID, d.DIAGNOSISID

#### HAVING COUNT(v.VISITID) > 1;

**Expected Result:**

* ID of the patient from the visit.
* The diagnosis (or diagnoses) for the visit.
* The count of visits where diagnosisID and PID is repeated

**Outcome:**

This returns the query successfully.

**Test Case 2:** Can a patient have different visits with the same treatment?

**Member:** Angel Vang

**Data:** Inserted data into PATIENT, TREATMENT id and name, VISIT id and pid, TREATMENTID had to be unique so sorted by treatment name instead.

**SQL Query:**

#### 

#### SELECT v.PID, t.TREATMENTNAME, COUNT(DISTINCT v.VISITID) AS VisitCount

#### FROM VISIT v

#### JOIN TREATMENT t ON v.VISITID = t.VISITID

#### WHERE v.VISITID = t.VISITID

#### GROUP BY v.PID, t.TREATMENTNAME

#### HAVING COUNT(DISTINCT v.VISITID) > 1;

**Expected Result:**

* Patient ID from Visit
* The name of the treatment received in visit where the visitIDs match Visit and Treatment
* The count of visits sorted by the PID and Treatment name

**Outcome:**

This query returned and counted successfully.

**Test Case 3:** Can you tell what doctor give what diagnosis(s) for a visit and the prescribed treatments for that visit (if any)?

**Member:** Baanee Singh

**Description:** Checks whether the diagnosis and treatments prescribed for a specific visit can be retrieved by joining the relevant tables.

**Data:**

**SQL Query:**

-- Query to check diagnosis and treatment for the visit

#### SELECT

#### v.VISITID,

#### v.PID,

#### d.DIAGNOSISNAME,

#### t.TREATMENTNAME,

#### doc.EMPLOYEEID AS DOCTORID,

#### doc.FNAME AS DOCTOR\_FNAME,

#### doc.LNAME AS DOCTOR\_LNAME

#### FROM VISIT v

#### JOIN DIAGNOSIS d ON v.VISITID = d.DIAGNOSISID

#### LEFT JOIN TREATMENT t ON v.VISITID = t.VISITID

#### JOIN EMPLOYEE doc ON v.SERVICEPROVIDERID = doc.EMPLOYEEID

#### WHERE doc.EMPLOYEEID IS NOT NULL;

**Expected Result:**

* The query should return the name of the doctor who provided the diagnosis for the visit.
* The diagnosis (or diagnoses) for the visit should be listed.
* If applicable, the treatments prescribed for that visit should also be listed.

**Outcome:**

This returns the query successfully.

**Test Case 4:** Can the same clerk do the intake for the same patient on different visits?

**Member:** Baanee Singh

**Description:** Verifies that an intake clerk can perform intake for the same patient on multiple visits by checking the visit records.

**Data:**

**SQL Query:**

#### SELECT

#### v.PID,

#### v.INTAKECLERKEMPLOYEEID,

#### COUNT(v.VISITID) AS NUM\_VISITS

#### FROM VISIT v

#### GROUP BY v.PID, v.INTAKECLERKEMPLOYEEID

#### HAVING COUNT(v.VISITID) > 1;

**Expected:**

* The query should return multiple rows, each corresponding to a different visit for the same patient (PID), where the same intake clerk (identified by INTAKECLERKEMPLOYEEID) was responsible for the intake.
* This confirms that the same intake clerk can perform intake for the same patient on different visits.

**Outcome:**

This returns the query successfully.

**Test Case 5:** Can you see who authorized permission for treatment of a minor?

**Member:** Daniela Ortiz

**Description:** This query retrieves information about treatments that have been approved by corresponding guardians for minors. Details regarding the patient, their guardian, and the approved treatments linked to the specific visit will be returned.

**Data:**

**SQL Query:**

#### SELECT

#### AM.PID AS PatiendID,

#### AM.GuardianID AS GuardianID,

#### PG.FName AS GuardianFName,

#### PG.LName AS GuardianLName,

#### AM.TreatmentID AS TreatmentID,

#### T.TreatmentName AS TreatmentName,

#### AM.VisitID AS VisitID

#### FROM

#### APPROVE\_MINOR AM

#### JOIN PARENT\_GUARDIAN PG ON AM.GuardianID = PG.GuardianID

#### JOIN TREATMENT T ON AM.TreatmentID = T.TreatmentID;

**Expected Results:**

* Patient ID and Guardian ID
* Guardian’s name (First and Last)
* Treatment details (ID and Name)
* Visit ID where treatment occurred.

**Outcome:**

This returns the query successfully.

**Project Testing Report**

The purpose of creating a database is to evaluate whether the database design based on the ER/EER Diagrams and Relational Schema is fit to manage patients, visits, diagnoses, treatments, and related information which fulfills the project’s functional requirements. We utilized five test cases to verify specific scenarios and ensure the database’s design effectively supports the intended use. The results of each of the test cases are detailed below, along with an analysis of whether the database captured the specified requirements. Adjustments made to the database during the testing process and explanation of the database structure are also included.

The database includes several interconnected tables, such as PATIENT, VISIT, EMPLOYEE, DIAGNOSIS, TREATMENT, and APPROVE\_MINOR. Each table has clearly defined relationships, with foreign keys linking related records. For instance, the VISIT table links patients (via patient ID) to diagnoses, treatments, and service providers (via employee ID). To ensure referential integrity, foreign key constraints were implemented, requiring valid references between related entities. Additionally, lookup tables like PARENT\_GUARDIAN were created to manage details of guardians and their roles in authorizing treatments for minors. This relational structure ensures that the database can capture and retrieve detailed information for each visit and its associated records.

In Test Case 1, we assessed whether a patient could have multiple visits associated with the same diagnosis. To test this, we created data for a single patient and recorded two separate visits, each with the same diagnosis provided by different service providers. Data entry involved populating various related tables due to foreign key constraints, such as PATIENT, DEPARTMENT, EMPLOYEE, DIAGNOSIS, VISIT, and LOGS. The SQL query grouped the visits by patient ID and diagnosis ID, counting occurrences where a patient had more than one visit with the same diagnosis. An issue here is that with inserting data, the same doctor (employee ID) cannot diagnose with the same diagnosis ID. By using different doctor employee IDs, the query was able to return expected results, confirming the patient ID, diagnosis, and the count of visits. This outcome demonstrated that the database supports multiple diagnoses across different visits for the same patient.

For Test Case 2, we evaluated whether the same patient could receive identical treatments across different visits. This scenario was tested by inserting data into tables such as PATIENT, TREATMENT, and VISIT. We sorted by treatment name because if the data was sorted by treatment ID, it would be harder to recognize. Since treatment ID had to be unique and no duplicate data was allowed, by finding the same treatment by name made it more clear to see what was received by the patient. By sorting by treatment name, and not treatment ID could be an issue with our database structure. If all treatment IDs have to be unique, the treatment data itself will have to be different, making it harder/impossible to check for “same” treatment by ID. By comparing the visit ID in VISIT and TREATMENT, we could check what treatment the patient got at their visit. Using a query that grouped visits by patient ID and treatment name, we confirmed that the database correctly identified multiple visits where the same treatment was applied. The query produced the expected results, showing the patient ID, treatment name, and count of visits.

Test Case 3 focused on determining whether we could identify the doctor who provided a diagnosis and prescribed treatments during a specific visit. For this test, data was entered into the VISIT, DIAGNOSIS, TREATMENT, and EMPLOYEE tables to link visits to both diagnoses and treatments. The SQL query returned successfully retrieving the visit ID, patient ID, diagnosis name, treatment name (if applicable), and details of the attending doctor, including their first name, last name and employee ID. This outcome confirmed that the database design supports the tracking of the relationship between a doctor, their diagnoses, and prescribed treatments.

In Test Case 4, we verified whether the same intake clerk could perform intake for the same patient across multiple visits. Data entry included associating an intake clerk (identified by employee ID) with multiple visits for a single patient. The SQL query grouped the results by patient ID and intake clerk ID, counting for the number of visits. The results confirmed that the same clerk could be linked to multiple visits for the same patient, demonstrating that the database design supports this scenario. This feature ensures continuity in patient care by allowing consistent staff involvement across visits.

Lastly, Test Case 5 addresses whether the database could track who authorized treatment for a minor. For this scenario, we inserted data into the APPROVE\_MINOR, PARENT\_GUARDIAN, TREATMENT, and VISIT tables. The SQL query joined these tables to retrieve the patient ID, guardian ID, guardian’s name, treatment details, and visit ID. The query executed successfully and returned the expected results, confirming that the database effectively tracks guardianship and authorization for minors. This feature is critical for ensuring compliance with legal requirements and maintaining accurate records of treatment authorization.

Overall, the testing process revealed that the database design effectively captures all the specified requirements. The queries for each test case returned the expected results, confirming the database’s ability to handle the tested scenarios. While entering data, adjustments were made, necessary to address foreign key constraints, but these changes did not impact the overall functionality of the database in managing the established relationships.

In conclusion, the database design successfully met all functional requirements. The results of our testing confirm that the system can accurately track patient visits, manage diagnoses and treatments, identify staff roles, and record guardianship details. All queries were returned correctly so no significant design flaws were identified during our limited testing with the provided queries. Thus, we concluded that the system is equipped to handle the project’s requirements, but we acknowledge that further testing could reveal some flaws in the system that weren’t covered by the test cases.