Test Results

Test Case 01

- **Result:** Pass
- **Explanation:** Our test case data inserts two users: one who is under the age of 18 and has no Guardian ID., and one who is under the age of 18 and has a guardian_ID. After inserting this patient into our data set, we test it by selecting this user by his patient ID. Our test query produces a table with patient 2, the patient with a guardian_ID. The first user is never inserted into the table from the start.

Test Case 02

- **Result:** Fail
- **Explanation:** We can use the same entries from test case 1 to test this case as well. In this case, we entered a person over the age of 18 along with a guardian_ID attribute that was not NULL. Based on the requirements, the system should not allow this , but we did not include that constraint when we created our tables, so we failed the test. Additionally, inserting a patient over the age of 18 without a guardian_ID fails to be inserted into the table because of the not NULL constraint. This is also incorrect.

Test Case 03

- **Result:** Fail
- **Explanation:** To test this constraint, we entered records into the Patient table, and attempted to update the Ins_Num to NULL. Since we set Ins_Num to have a NOT NULL requirement in the table, we fail the test, and are not able to represent patients without insurance in the system

Test Case 04

- **Result:** Fail
- **Explanation:** To test this constraint, we entered multiple records into the Insures table with the same Patient ID, to test that the table is capable of holding full insurance history for each patient. We then selected those records from the Insures table and were able to see the start and end dates of their entire insurance history. However, when we return to the patient table to try and check the Ins_ID, our system fails to update it to the active insurance, so we failed the test overall.

Test Case 05

- **Result:** Fail
- **Explanation:** To test this constraint, we used the log table to try to add two records with the same In_Time, Out_Time, and PID, with different service provider SSNs, which would signify a patient on the same visit being logged by two service providers. Since both records are entered successfully, our test fails.

Test Case 06

- **Result:** Pass
- **Explanation:** To test this requirement, we entered multiple records with the same Patient ID and Service Provider in the Diagnosis table, but different diagnoses, which represents a Service provider logging more than one Diagnosis for the same patient. This test passes as the records are able to be loaded, and Visit information is not needed.

<u>Test Case 07:</u>

- **Result:** Pass
- **Explanation:** We included this constraint in our Diagnosis table, as in order for a diagnosis to be made, a service provider's SSN is required to be attached and a different class of employee would not be compatible, so we pass the test. When attempting to create a diagnosis using a nurse's SSN, this query cannot execute due to the foreign key constraint on the SSN.

<u>Test Case 08:</u>

- **Result:** Fail
- **Explanation:** In order to test this requirement, we attempted to enter multiple records into the treatment table, with some having the correct ICD-10-PCS coding format, and some having codes in the incorrect format. In the results, we see that the test fails because both diagnoses have different ICD-10-PCS length.

<u>Test Case 09:</u>

- **Result:** Fail
- **Explanation:** To test this requirement, we created records in the initial evaluation table and the Insures table, which account for the Copay and Insurance Info information of a patient respectively. While there exists a constraint in the Initial Evaluation table to log the clerk along with the copay, the Insurance info is entered without having a clerk associated, so the test fails.

<u>Test Case 10:</u>

- **Result:** Pass
- **Explanation:** To test the constraint, we created multiple initial evaluations for a single patient but with unique clerk_ids for each. The test passes because both initial evaluations are logged for the patient.

<u>Test Case 11:</u>

- **Result:** Pass
- **Explanation:** We updated our previous LOGS table (from Test Case 05) so that there are two visits by the same user on different dates, representing the success of this test.

<u>Test Case 12:</u>

- **Result:** Pass

- **Explanation:** We created the data records in the VItal_Records table, which ties the NSSN to the Nurse table. Using the foreign key and primary key relations, we could track the nurse information from the Employee table. Hence, the test passes.

Test Case 13:
- **Result:** Pass
- **Explanation:** We created the records in the Vital_Records table with multiple patients' records in the Patient table. As a result, we could retrieve multiple records from Vital_Records with the same nurse SSN. So the test passes.

Test Case 14:
- **Result:** Pass
- **Explanation:** We created a query for adding the nurse data into the Vital_Records table. The query could initiate the assessment with only one nurse but not the same query during the second time. Since the table only constraint that the SSN and PID attribute must be not null, the test passes.

Test Case 15:
- **Result:** Fail
- **Explanation:** We inserted a query for inputting multiple patients records into the Vital_Records table. While there exist different nurses doing assessments, it could not track a single visit. Thus, we have two or more records of the the patient since we do not have the visit as an attribute. So, the test fails.

Test Case 16:
- **Result:** Fail
- **Explanation:** We created the query for the multiple employee records in the Employee table. Although the SSN in the Employee table is not null and unique, the Pay_Type attribute is not properly defined because we can input other entries besides "salaried" or "hourly". Thus, the test fails.

Test Case 17:
- **Result:** Pass
- **Explanation:** We assigned service providers as the doctors who do the treatments. So we make multiple queries from different service providers and different tests. Since the doctors only care about treatment, they can access multiple tests as long as the ICD_10_PCS is not null and unique to guide them doing the treatment procedure. Hence, the test passes.

Test Case 18:
- **Result:** Pass

- **Explanation:** Since the PID in the Treatment table is foreign key, it is not necessary for a patient to have the test/procedure. So the test passes as long as the initial evaluation has the records of PID, Med_Info, Copay, Visit_Date, Clerk_ID, Patient_ID.

Test Case 19:

- **Result:** Pass
- **Explanation:** We created one data record in EMERGENCY_CONTACT table, and two data records in PATIENT table. While there exist multiple patients with their unique SSN, we could retrieve different patients' information from the PATIENT table with Emergency_ID tied back to the same Emergency Contact person. Thus, the first test passes. Similarly, TREATMENT table holds the relationship between service provider and patient. So we created data records in the TREATMENT table. While there exists data in both Patient and Service_Provider, one or many service providers can treat multiple patients. Hence, the second test passes.

Test Case 20:

- **Result:** Pass
- **Explanation:** We create the query for the INSURANCE_PROVIDERS table. Then we run the same query again. The second query would not run because of the primary key constraint where Insurer_ID must be not null and unique. So the first, test passes. Similarly, we created the data records for the DEPARTMENT table. While there exists Dep_ID as primary key, the second query where it has the same Dep_ID fails to execute. Thus, the second test passes.

Flaws like making all attributes not null is a big mistake and is evident in multiple tests. As a refinement, we need to consider somecase like Guard_ID in the PATIENT table is not always NOT NULL. We could refine the PATIENT table by creating a trigger to sort whether the patient is under 18 or over 18.