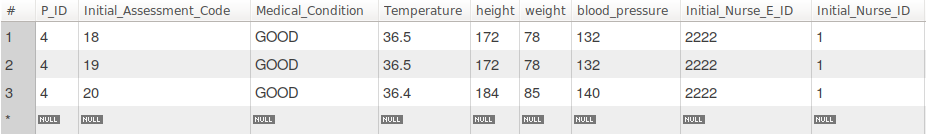
|  |  |
| --- | --- |
| **Project: New Haven Urgent Care** | **Team# 1** |
|  | **Test Date: 12/8/2019** |
| **Test Case ID#: 19** | **Name(s) of Tester(s): Kun Ki Lee, Eric Hwang, Kyeongtak Han, Dongha Kang** |
| **Test Description (What are you testing? – you must be specific): We are testing the maximum numbers for the relations as same as the cardinality we designed. Form used one and only one initial assessment.** |  |

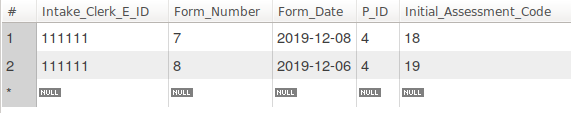
**NOTE: The following information must be provided to be given credit for any test**

**Test Data (Provide the file name of the script used to insert data, provide a screen capture to reflect data, or provide script here):**

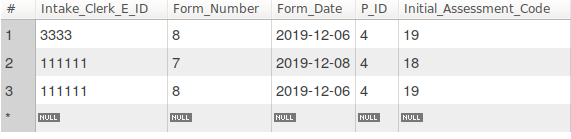
**Initial Assessment with P\_ID = 4**



Form with P\_ID = 4



Result of putting more Form with using same initial assessment



**SQL Query(s) used for testing:**

INSERT INTO Form VALUES (111111, 8, '2019-12-06', 4, 18);

INSERT INTO Form VALUES (3333, 8, '2019-12-06', 4, 19);

INSERT INTO Form VALUES (111111, 8, '2019-12-07', 4, 18);

Explanation: Even if the cardinality sets up to one and only one between Initial Assessment and Form, our data is unable to capture the over-insertion the data in one to one relation. Though Form table can only have the one initial assessment, inserting the different intake\_clerk\_E\_ID with the same assessment and P\_ID enables the user to put the same assessment and patient\_ID in the data. The duplication of P\_ID in the Form is okay, but duplication of Initial\_assessment\_code wouldn’t be allowed by the cardinality.

SO, the possible solution is that we should modify the code to have the primary key that identifies the uniqueness of the Initial\_assessment\_code and P\_ID. Thus, we only track the duplication of the combination of the P\_ID and Initial\_Assessment\_Code.