Test Case 1

Project: New Haven Urgent Care Team #8

Test Date: 12/11/2021

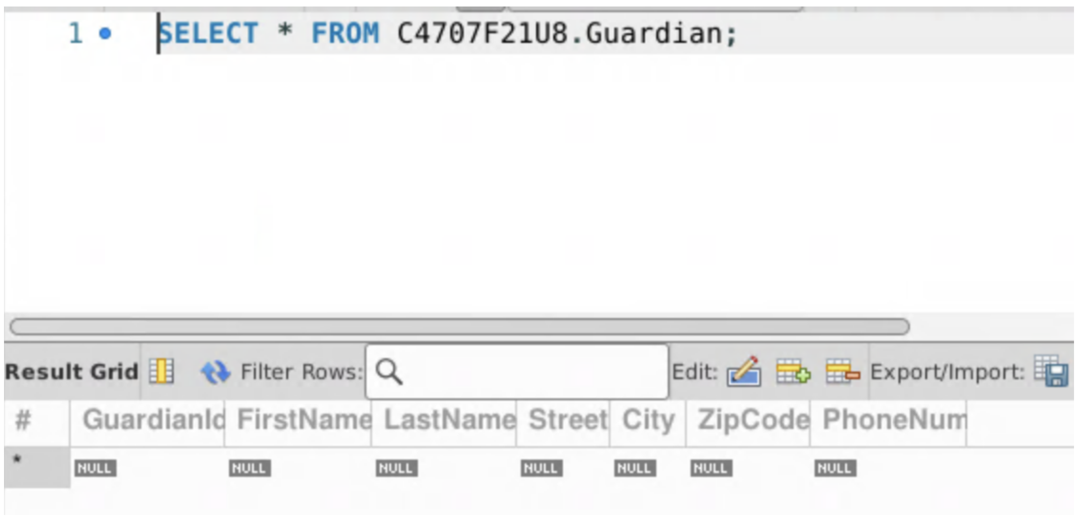
Test Case ID#: 1
Name(s) of Tester(s): Anushka Angamuthu

Test Description (What are you testing? – you must be specific):
I am testing the requirement that all patients under the age of 18 must have a parent or guardian in the system. I will test this by checking if it is possible to insert a patient into the system who is under the age of 18 without first having a parent or guardian who is in the system.

SQL Query(s) used for testing:

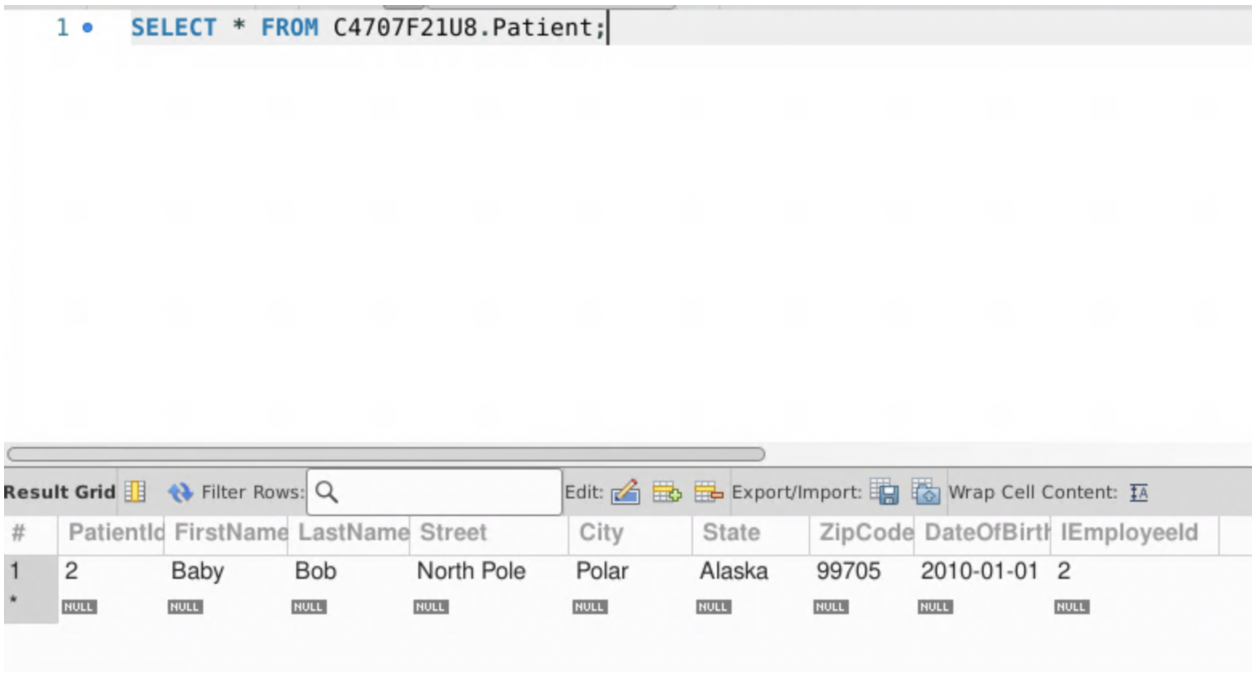
```
-- Insert a record into the intake clerk table with matching employee id to satisfy FK constraint on
the patient table
INSERT INTO C4707F21U8.IntakeClerk(EmployeeID)
VALUES(2);

-- Insert a patient under 18 with the following test data
INSERT INTO C4707F21U8.Patient(PatientId, FirstName, LastName, Street, City, State,
ZipCode, DateofBirth, IEmployeeId)
VALUES (2, 'Baby', 'Bob', 'North Pole', 'Polar', 'Alaska', 99705, '2010-01-01', 2);

-- indicate that the patient is a child by adding the patient record into the child table
INSERT INTO C4707F21U8.Child(PatientId, GaurdianId)
VALUES(2, 1);
```

Results:
-- Guardian table is empty



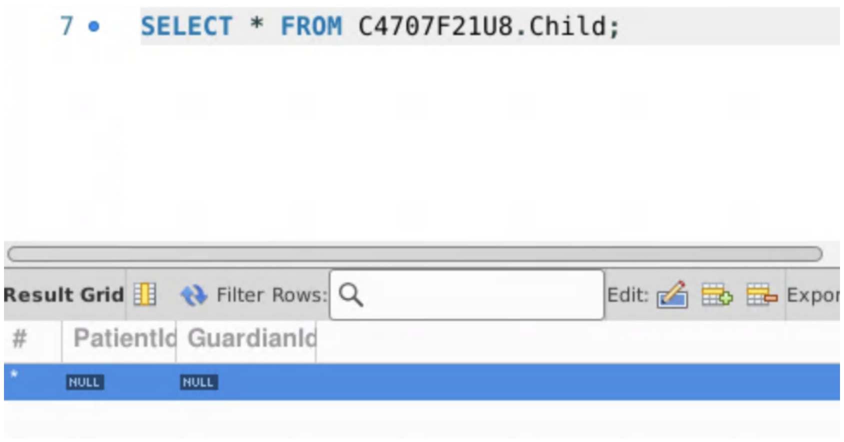
```sql
SELECT * FROM C4707F21U8.Guardian;
```

#	GuardianId	FirstName	LastName	Street	City	ZipCode	PhoneNum
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL



```sql
SELECT * FROM C4707F21U8.Patient;
```

#	PatientId	FirstName	LastName	Street	City	State	ZipCode	DateOfBirth	lEmployeeId
1	2	Baby	Bob	North Pole	Polar	Alaska	99705	2010-01-01	2
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



```sql
SELECT * FROM C4707F21U8.Child;
```

#	PatientId	GuardianId
*	NULL	NULL

Discussion/Explanation:

This test case passed succesfully. This is because the Child table is empty therefore not allowing a child to be created without first having a guardian. This is because the query is not successful without a valid guardian id due to the FK constraint being enforced by the database. One thing to note is that it is still possible to insert the child into the patients table without a guardian. This is something that may potentially lead to unwanted behaviors. In order to circumvent this, a possible design change consideration would be to separate out child and adult patients into their own tables and enforce FK constraints directly on those tables.

Test Case 2

Project: New Haven Urgent Care   Team #8

Test Date: 12/12/2021

Test Case ID#: 2
Name(s) of Tester(s): Anushka Angamuthu

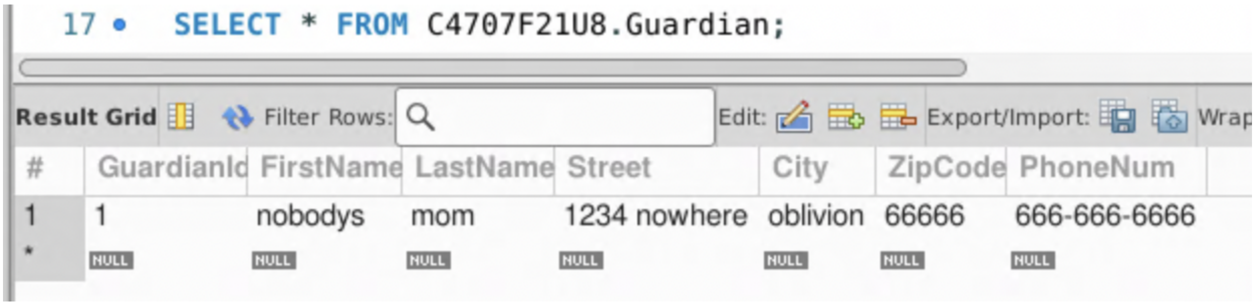
Test Description (What are you testing? – you must be specific):
I am testing that a patient over the age of 18 does not have a parent or guardian. I will test this by attempting to insert a record for a patient over the age of 18 who has a parent or guardian in the system

SQL Query(s) used for testing:

```
-- create a new guardian
INSERT INTO C4707F21U8.Guardian(GaudianId, FirstName, LastName, Street, City, ZipCode,
PhoneNum)
VALUES(1, ‘nobodys’, ‘mom’, ‘1234 nowhere’, ‘oblivion’, ‘66666’, ‘666-666-6666’);

-- Insert a record into the intake clerk table with matching employee id to satisfy FK constraint on
the patient table
INSERT INTO C4707F21U8.IntakeClerk(EmployeeID)
VALUES(3);

-- Insert a patient over 18 with the following test data
INSERT INTO C4707F21U8.Patient(PatientId, FirstName, LastName, Street, City, State,
ZipCode, DateofBirth, IEmployeeId)
```

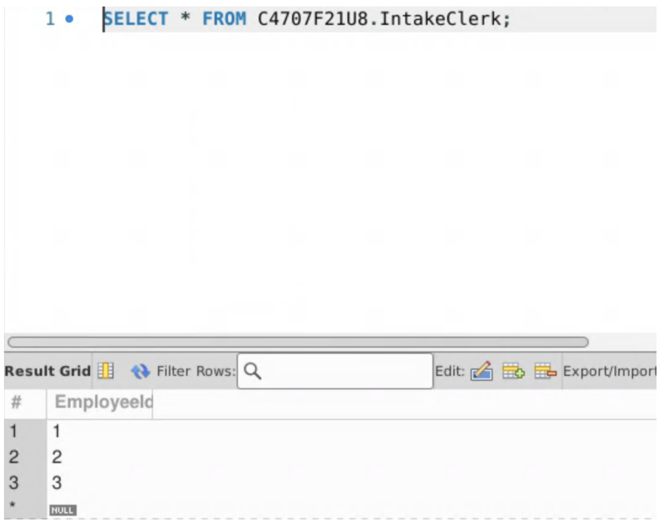
```
VALUES (3, ‘Adult’, 'Bob', 'North Pole', 'Polar', 'Alaska', 99705, '1999-01-01', 3);

-- indicate that the patient is an adult by adding the patient record into the adult table
INSERT INTO C4707F21U8.Adult(PatientId, PreferredMethodOfContact)
VALUES(3, ‘email’);
```

Results:



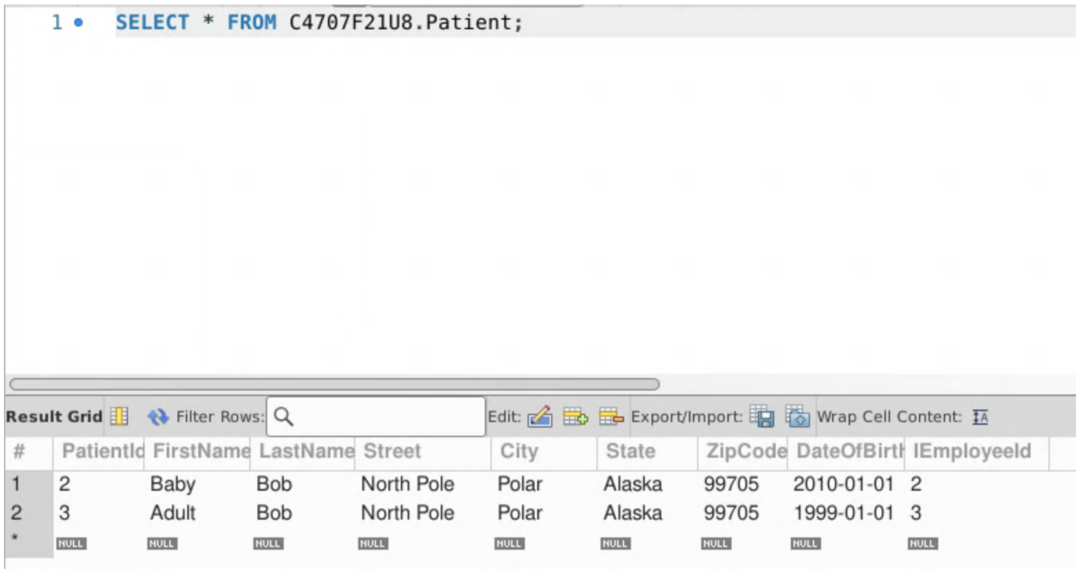
```
SELECT * FROM C4707F21U8.Guardian;
```

#	GuardianId	FirstName	LastName	Street	City	ZipCode	PhoneNum
1	1	nobodys	mom	1234 nowhere	oblivion	66666	666-666-6666
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL



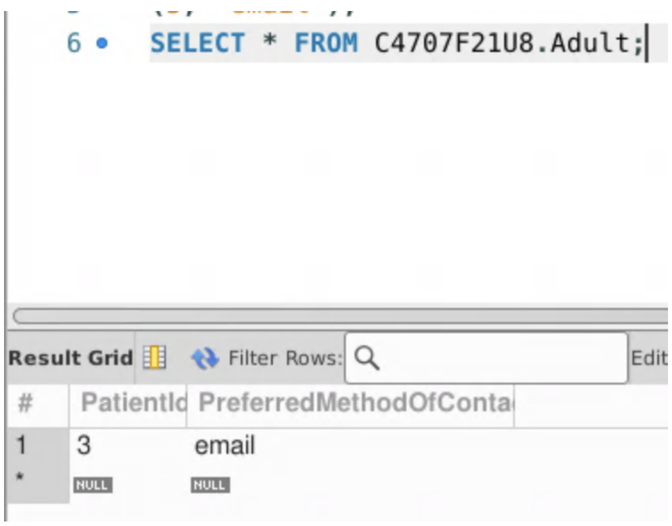
```
SELECT * FROM C4707F21U8.IntakeClerk;
```

#	EmployeeId
1	1
2	2
3	3
*	NULL



```
SELECT * FROM C4707F21U8.Patient;
```

#	PatientId	FirstName	LastName	Street	City	State	ZipCode	DateOfBirt	IEmployeeId
1	2	Baby	Bob	North Pole	Polar	Alaska	99705	2010-01-01	2
2	3	Adult	Bob	North Pole	Polar	Alaska	99705	1999-01-01	3
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



```
SELECT * FROM C4707F21U8.Adult;
```

```
#  PatientId  PreferredMethodOfConta
1  3          email
*  NULL       NULL
```

Discussion/Explanation:

This test case passed successfully. This is because there is nothing in the database tying an adult patient to a guardian. The guardian was able to be created successfully and the adult patient was able to be created successfully separately. However, the two do not have any dependency on each other which means an adult patient will never be permitted to have an associated guardian. In fact, there is nothing tying any patient record to a guardian which was mentioned in the test case above as a possible point of vulnerability in the design of the system.

Test Case 3

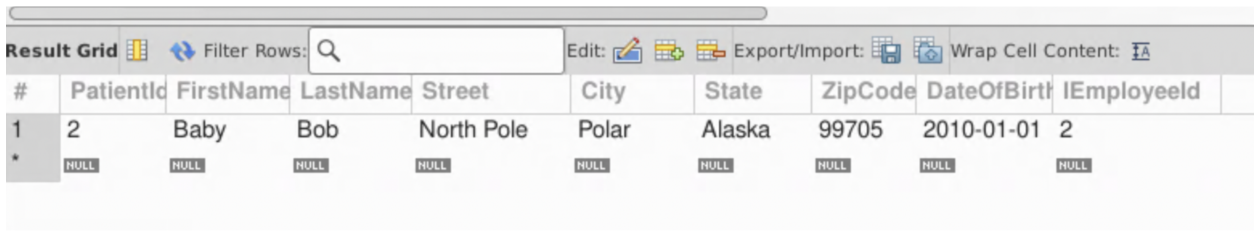
Project: New Haven Urgent Care Team #8

Test Date: 12/12/2021

Test Case ID#: 3
Name(s) of Tester(s): Anushka Angamuthu

Test Description (What are you testing? – you must be specific):
I am testing if an underaged patient can have different parents or guardians assigned to them for different visits. I will test this by attempting to insert a patient into the system who is under the age of 18 with 2 different parents or guardians assigned to them.

Test Data: We will use baby bob's data from test case 1

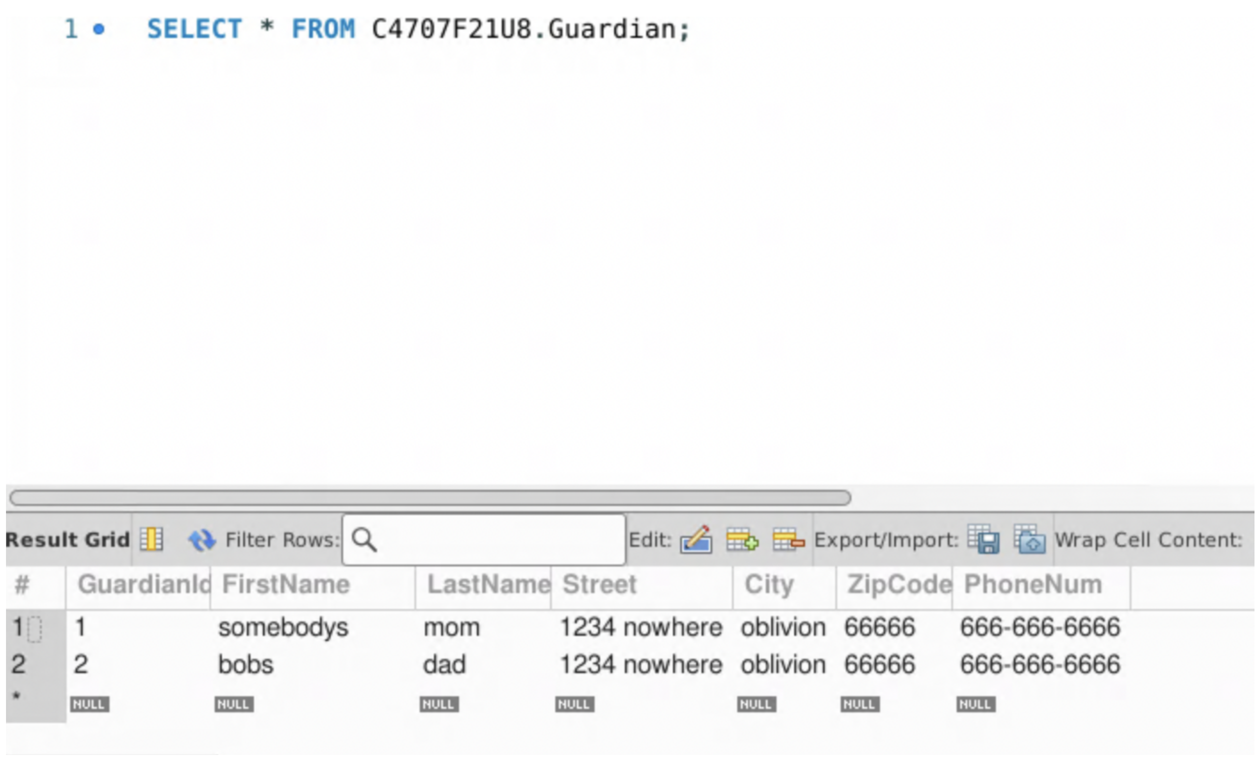


#	PatientId	FirstName	LastName	Street	City	State	ZipCode	DateOfBirth	IEmployeeId
1	2	Baby	Bob	North Pole	Polar	Alaska	99705	2010-01-01	2
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

SQL Query(s) used for testing:

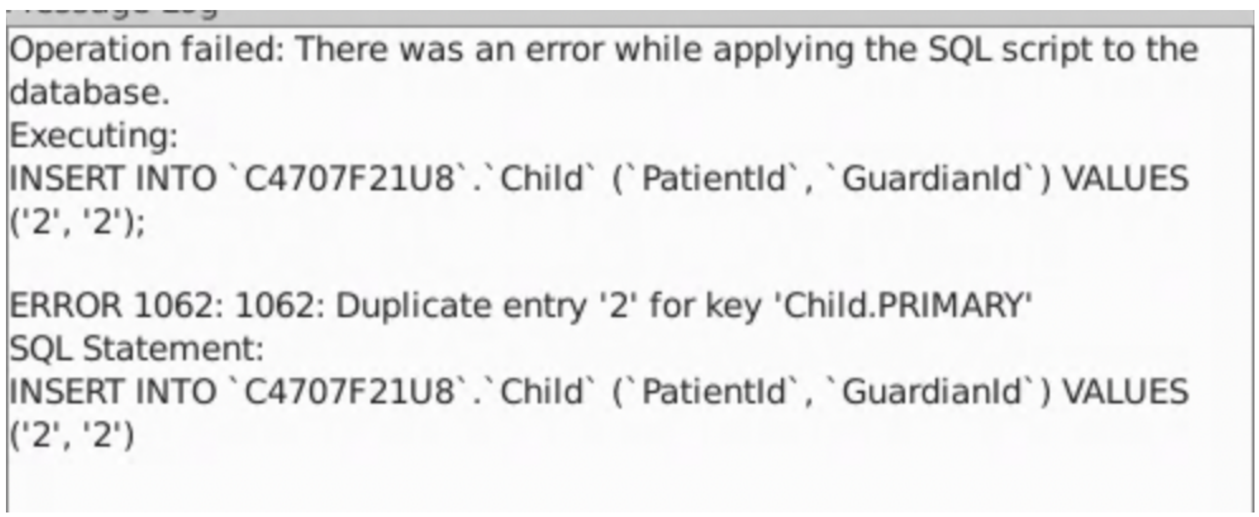
```
-- Insert baby bob as a patient
INSERT INTO C4707F21U8.Patient(PatientId, FirstName, LastName, Street, City, State,
ZipCode, DateofBirth, IEmployeeId)
VALUES (2, 'Baby', 'Bob', 'North Pole', 'Polar', 'Alaska', 99705, '2010-01-01', 2);

-- create 2 guardians for Baby Bob
INSERT INTO C4707F21U8.Guardian(GaurdianId, FirstName, LastName, Street, City, ZipCode,
PhoneNum)
VALUES(1, ‘somebodys’, ‘mom’, ‘1234 nowhere’, ‘oblivion’, ‘66666’, ‘666-666-6666’);

INSERT INTO C4707F21U8.Guardian(GaurdianId, FirstName, LastName, Street, City, ZipCode,
PhoneNum)
VALUES(2, ‘bobs’, ‘dad’, ‘1234 nowhere’, ‘oblivion’, ‘66666’, ‘666-666-6666’);

-- add baby bob into the child table with both guardians
INSERT INTO C4707F21U8.Child(PatientId, GaurdianId)
VALUES(2, 1);

INSERT INTO C4707F21U8.Child(PatientId, GaurdianId)
VALUES(2, 2);
```

Results:



```
SELECT * FROM C4707F21U8.Guardian;
```

#	GuardianId	FirstName	LastName	Street	City	ZipCode	PhoneNum
1	1	somebodys	mom	1234 nowhere	oblivion	66666	666-666-6666
2	2	bobs	dad	1234 nowhere	oblivion	66666	666-666-6666
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL



```
Operation failed: There was an error while applying the SQL script to the
database.
Executing:
INSERT INTO `C4707F21U8`.`Child` (`PatientId`, `GuardianId`) VALUES
('2', '2');

ERROR 1062: 1062: Duplicate entry '2' for key 'Child.PRIMARY'
SQL Statement:
INSERT INTO `C4707F21U8`.`Child` (`PatientId`, `GuardianId`) VALUES
('2', '2')
```

Discussion/Explanation:

This test case failed. This is because we are not able to insert multiple guardians for one child patient. The way that the database has been designed prevents us from doing so because the child table's PK is patient id and therefore a single patient can only exist in the table once. This is why the system allows us to insert the record (2,1) which assigns our first guardian to the child, but not the record (2,2). This is because the primary key constraint is violated. One possibility to solve this problem would be to create an intervening table in order to capture the many to many relationship between guardians and children. Another option could be to potentially make guardian id a multivalued attribute.

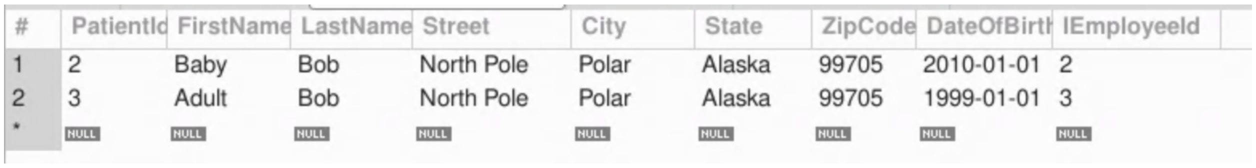
Test Case 4

Project: New Haven Urgent Care Team #8

Test Date: 12/12/2021
Test Case ID#: 4
Name(s) of Tester(s): Anushka Angamuthu

Test Description (What are you testing? – you must be specific):
I am testing if a patient can be uninsured and have their credit card information collected to be charged for the visit and any supplies used during the visit. I will test this by inserting a new uninsured patient into the database along with credit card information that is tied to them.

Test Data: We will use adult bob's data from test case 1

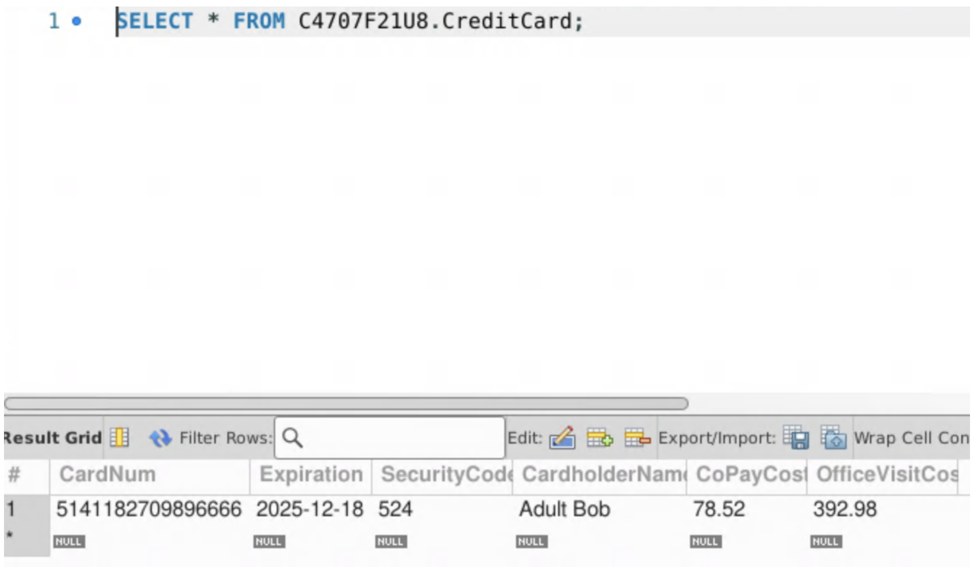


#	PatientId	FirstName	LastName	Street	City	State	ZipCode	DateOfBirth	IEmployeeId
1	2	Baby	Bob	North Pole	Polar	Alaska	99705	2010-01-01	2
2	3	Adult	Bob	North Pole	Polar	Alaska	99705	1999-01-01	3
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

SQL Query(s) used for testing:

```
-- Insert adult bob as a patient
INSERT INTO C4707F21U8.Patient(PatientId, FirstName, LastName, Street, City, State,
ZipCode, DateofBirth, IEmployeeId)
VALUES (3, ‘Adult’, 'Bob', 'North Pole', 'Polar', 'Alaska', 99705, '1999-01-01', 3);

-- Add a credit card for adult bob
INSERT INTO C4707F21U8.CreditCard(CardNum, Expiration, SecurityCode, CardholderName,
CoPayCost, OfficeVisitCost)
VALUES (‘5141182709896666’, ‘2025-12-18’, ‘524’, ‘Adult Bob’, 78.52, 392.98);

-- Add adult bob as an uninsured patient
INSERT INTO C4707F21U8.Uninsured(PatientId, CCardNum)
VALUES(3, ‘5141182709896666’);
```

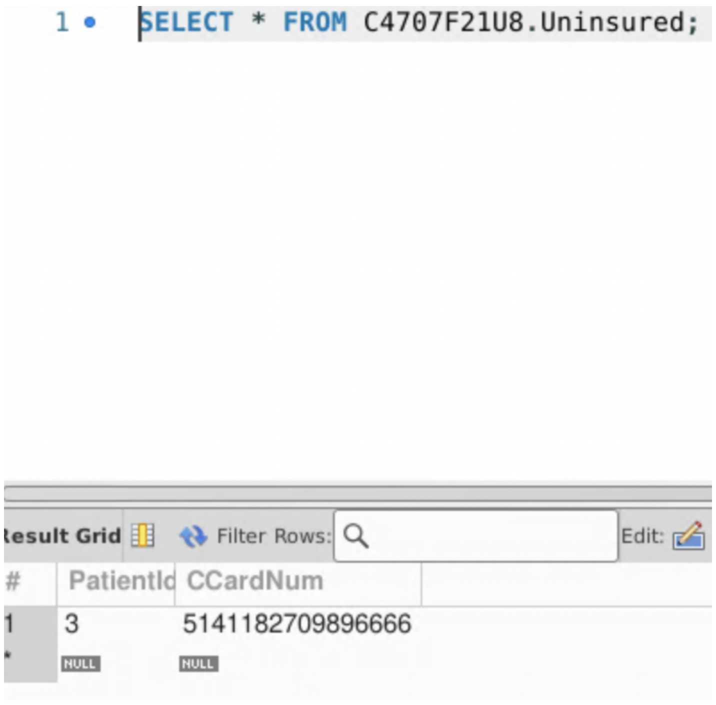
Results:



```
SELECT * FROM C4707F21U8.CreditCard;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Con

#	CardNum	Expiration	SecurityCod	CardholderNam	CoPayCost	OfficeVisitCos
1	5141182709896666	2025-12-18	524	Adult Bob	78.52	392.98
*	NULL	NULL	NULL	NULL	NULL	NULL



```
SELECT * FROM C4707F21U8.Uninsured;
```

Result Grid | Filter Rows: | Edit:

#	PatientId	CCardNum
1	3	5141182709896666
*	NULL	NULL

Discussion/Explanation:

This test case passed successfully. This is because we were able to insert an adult patient into the database, insert a new credit card into the credit card table, and tie that new adult patient to the newly inserted credit card record through a FK constraint which allowed us to indicate that the patient is uninsured. Since their credit card information has been collected before they can be registered as an uninsured patient, they may successfully be charged for the visit and any supplies used during the visit.

Test Case 5

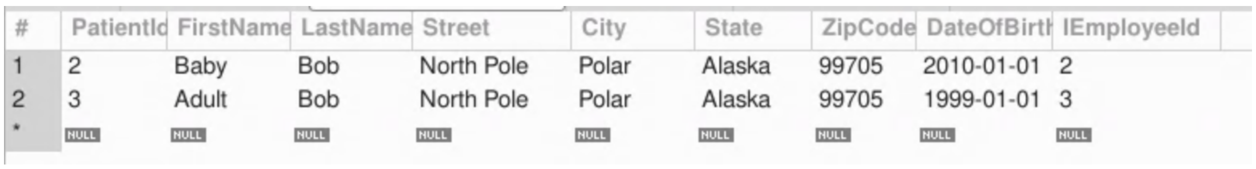
Project: New Haven Urgent Care Team #8

Test Date: 12/12/2021

Test Case ID#: 5
Name(s) of Tester(s): Anushka Angamuthu

Test Description (What are you testing? – you must be specific):
I am testing if an uninsured patient can be in the system without their credit card information being collected first. I will test this by attempting to insert a new patient into the system who is uninsured and does not have any credit card information tied to them in the system.

Test Data: We will use adult bob's data from test case 1

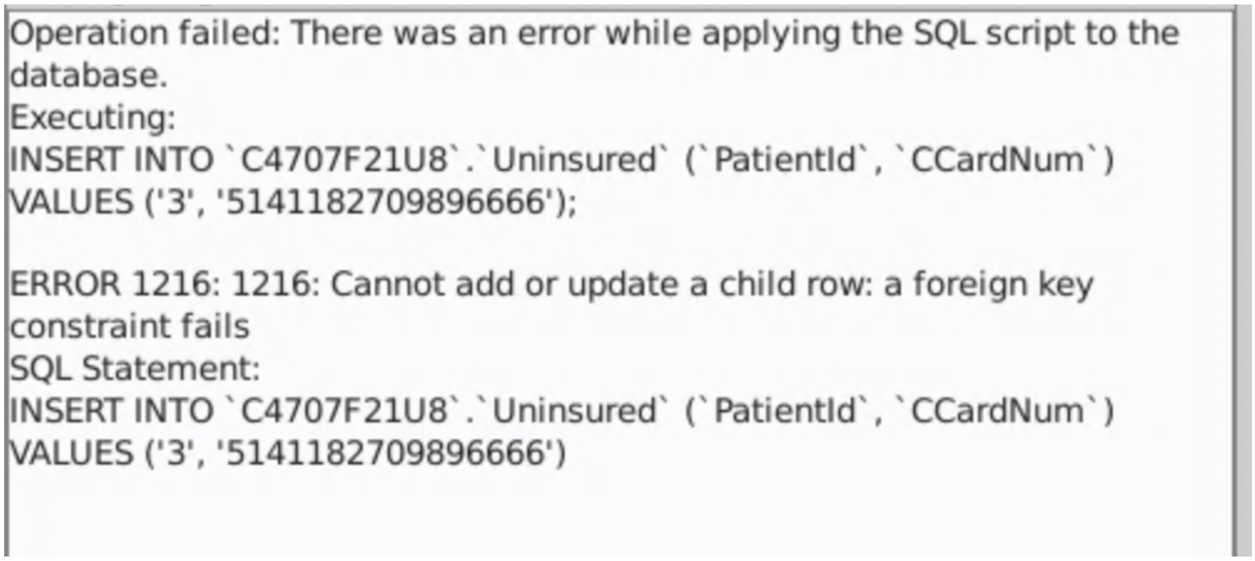


#	PatientId	FirstName	LastName	Street	City	State	ZipCode	DateOfBirth	IEmployeeId
1	2	Baby	Bob	North Pole	Polar	Alaska	99705	2010-01-01	2
2	3	Adult	Bob	North Pole	Polar	Alaska	99705	1999-01-01	3
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

SQL Query(s) used for testing:

```
-- Insert adult bob as a patient
INSERT INTO C4707F21U8.Patient(PatientId, FirstName, LastName, Street, City, State,
ZipCode, DateofBirth, IEmployeeId)
VALUES (3, ‘Adult’, 'Bob', 'North Pole', 'Polar', 'Alaska', 99705, '1999-01-01', 3);

-- Add adult bob as an uninsured patient
INSERT INTO C4707F21U8.Uninsured(PatientId, CCardNum)
VALUES(3, ‘5141182709896666’);
```

Results:



```
Operation failed: There was an error while applying the SQL script to the
database.
Executing:
INSERT INTO `C4707F21U8`.`Uninsured` (`PatientId`, `CCardNum`)
VALUES ('3', '5141182709896666');

ERROR 1216: 1216: Cannot add or update a child row: a foreign key
constraint fails
SQL Statement:
INSERT INTO `C4707F21U8`.`Uninsured` (`PatientId`, `CCardNum`)
VALUES ('3', '5141182709896666')
```

Discussion/Explanation:

This test case passed successfully. This is because we were not able to put an uninsured patient into the system without first collecting their credit card information. We know this because there is a FK constraint on the uninsured table on the credit card number field. This constraint states that we cannot register an uninsured patient in the system if we do not have a credit card to tie them to so they can be charged for any visits. The same point which was noted in test cases 1 and 2 that a patient record may still be inserted with no issue from the database also applies to this test case.
