ASSIGNMENT COVER PAGE

Programme	Course Code and Title
Bachelor of Computer Science (Hons)	CDB3033N (Database Programming)
Student's name / student's id	**Lecturer's name**
0204677 Lim Zhe Yuan	Ts. Chng Chern Wei

Date issued	Submission Deadline	Indicative Weighting
Week 3 - 26/09/2023	Week 7 - 17/10/2023	30%
Assignment [1]	SQL Queries & Stored Procedures	

This assessment assesses the following course learning outcomes

# as in Course Guide	UOWM KDU Penang University College Learning Outcome
CLO1	Develop scripting for prototyping database applications with predefined functions. (C5, PLO3)
CLO2	Apply database integrity in a concurrent environment (C3, PLO3)
CLO3	
CLO4	
# as in Course Guide	**University of Lincoln Learning Outcome**
CLO1	Use appropriate tools and techniques to design a database
CLO2	Appraise the structure of a database design using standard evaluation mechanisms
CLO3	
CLO4	

Student's declaration

I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.

Student's signature:	Submission Date:
Zhe Yuan	27/10/2023

Table of Contents

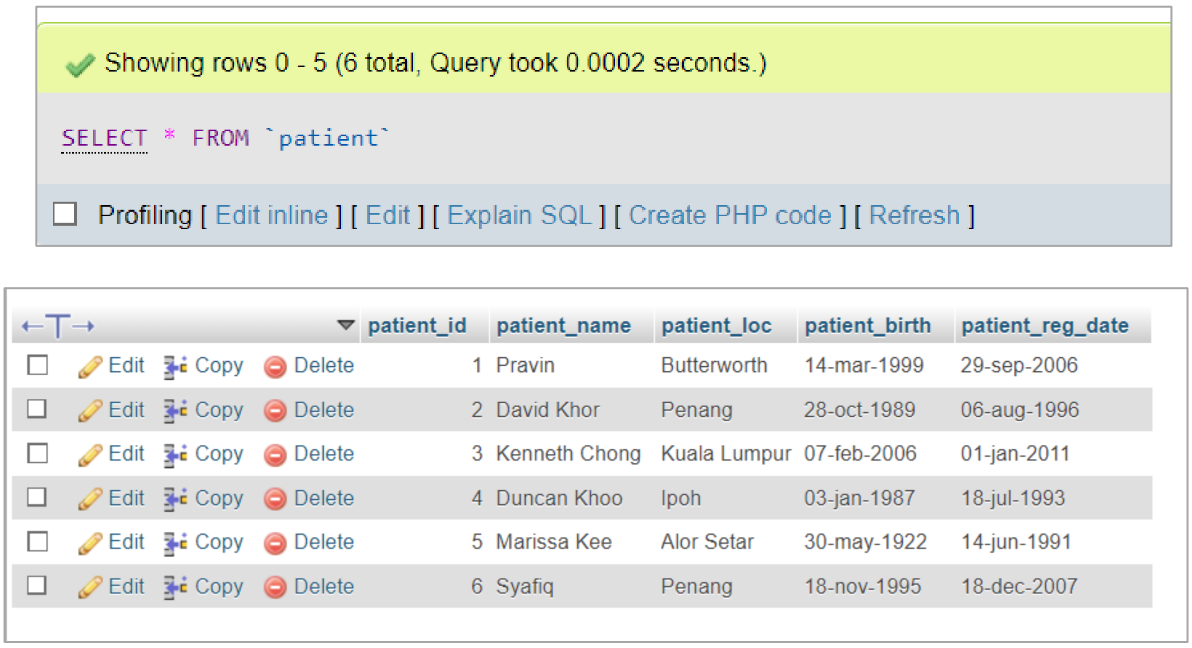
Main Report

Task 1

(a) Display id, name, date of birth, today's date, and age in years for those patients who are under the age of 30 and who have received a given vaccination in 2011 or later.

The name should be displayed in upper case. All dates should be displayed in full (e.g., 31-Jan-2011). The age in years should be a whole number, i.e., any fractions should be removed.

Firstly, a list of all recorded patients in the `patient` relation is queried as shown in **Figure 1**. From the query result, it can be expected that Pravin, Kenneth and Syafiq fall under the age of 30 as their birth dates are within the 30-year range from the current year, while others are above the age of 30 and will not be selected.



Showing rows 0 - 5 (6 total, Query took 0.0002 seconds.)

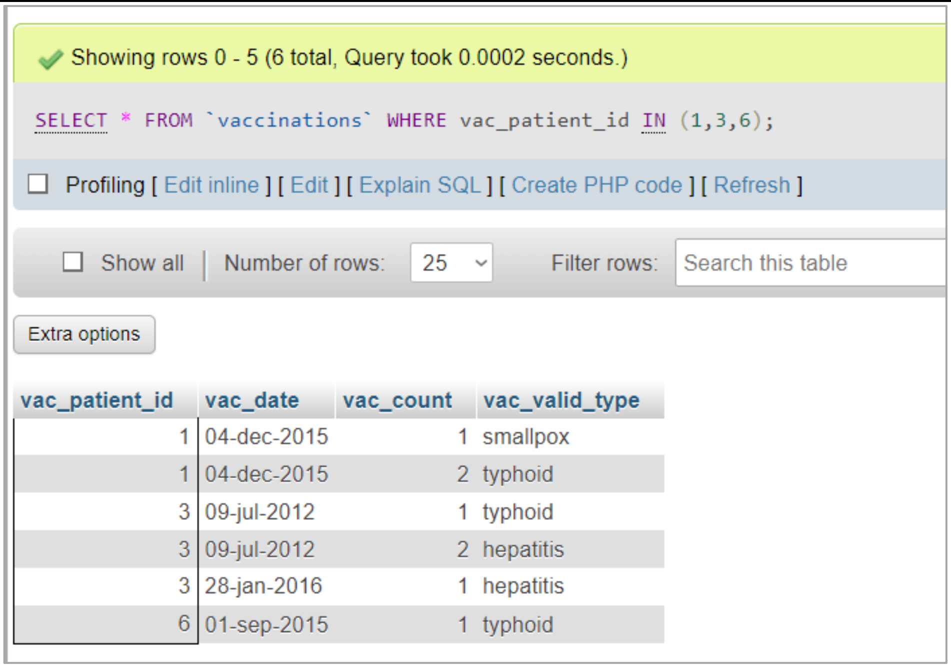
```sql
SELECT * FROM `patient`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Edit
Copy
Delete

patient_id	patient_name	patient_loc	patient_birth	patient_reg_date
1	Pravin	Butterworth	14-mar-1999	29-sep-2006
2	David Khor	Penang	28-oct-1989	06-aug-1996
3	Kenneth Chong	Kuala Lumpur	07-feb-2006	01-jan-2011
4	Duncan Khoo	Ipoh	03-jan-1987	18-jul-1993
5	Marissa Kee	Alor Setar	30-may-1922	14-jun-1991
6	Syafiq	Penang	18-nov-1995	18-dec-2007

***Figure 1**: Full `patient` relation data.*

Based on this inference, another query is executed to check for these patients' latest vaccination visit dates using their patient ID. The result shown in **Figure 2** confirms that all 3 patients have checked in for a vaccination visit since 2011. Therefore, Pravin, Kenneth, and Syafiq's patient record are the result expectations of the query designated to find patients under the age of 30 and who have received a given vaccination in 2011 or later.



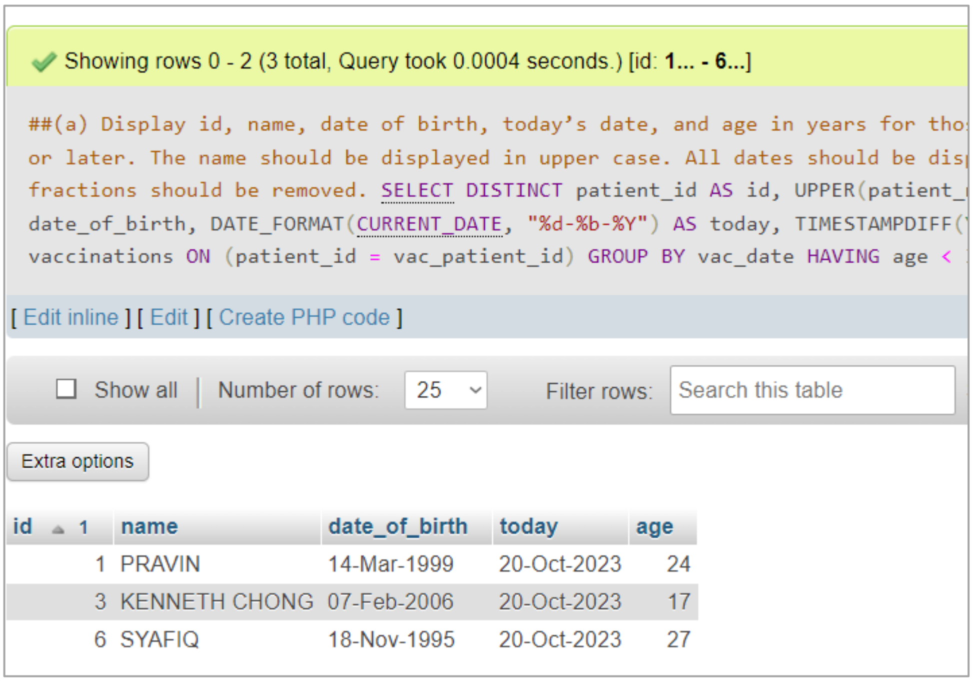
Showing rows 0 - 5 (6 total, Query took 0.0002 seconds.)

```sql
SELECT * FROM `vaccinations` WHERE vac_patient_id IN (1,3,6);
```

vac_patient_id	vac_date	vac_count	vac_valid_type
1	04-dec-2015	1	smallpox
1	04-dec-2015	2	typhoid
3	09-jul-2012	1	typhoid
3	09-jul-2012	2	hepatitis
3	28-jan-2016	1	hepatitis
6	01-sep-2015	1	typhoid

***Figure 2**: Patient records in `vaccinations` relation that are expected under age 30 and went for vaccination in 2011 or later.*

To obtain the expected result, the SELECT query written in **Appendix 1** is executed. As a result, the system outputs the expected patient records, and lists out their id, name, date of birth, current date, and the calculated age between the 2 dates. In conformance of the requirements, patient names are uppercased, dates are written clearly with the date-mon-year format, and age values are set as whole numbers.



Showing rows 0 - 2 (3 total, Query took 0.0004 seconds.) [id: 1... - 6...]

```sql
##(a) Display id, name, date of birth, today's date, and age in years for tho
or later. The name should be displayed in upper case. All dates should be dis
fractions should be removed. SELECT DISTINCT patient_id AS id, UPPER(patient_
date_of_birth, DATE_FORMAT(CURRENT_DATE, "%d-%b-%Y") AS today, TIMESTAMPDIFF(
vaccinations ON (patient_id = vac_patient_id) GROUP BY vac_date HAVING age <
```

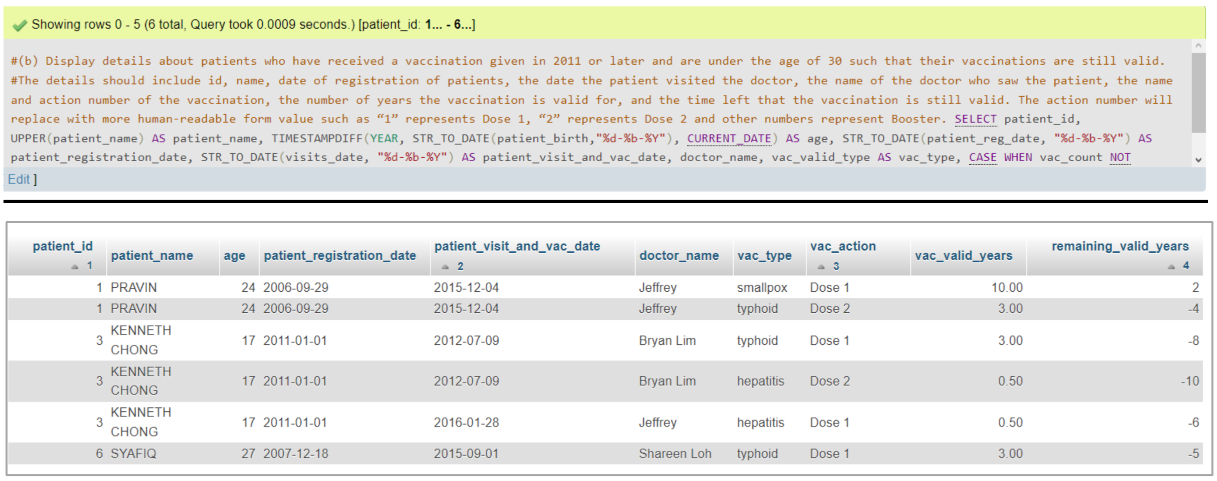
id	name	date_of_birth	today	age
1	PRAVIN	14-Mar-1999	20-Oct-2023	24
3	KENNETH CHONG	07-Feb-2006	20-Oct-2023	17
6	SYAFIQ	18-Nov-1995	20-Oct-2023	27

***Figure 3**: Query execution that selects patients that are under age 30 and went for vaccination in 2011 or later.*

(b) Display details about patients who have received a vaccination given in 2011 or later and are under the age of 30 such that their vaccinations are still valid.

The details should include id, name, date of registration of patients, the date the patient visited the doctor, the name of the doctor who saw the patient, the name and action number of the vaccination, the number of years the vaccination is valid for, and the time left that the vaccination is still valid. The action number will replace with more human-readable form value such as "1" represents *Dose 1*, "2" represents *Dose 2* and other numbers represent *Booster*.

Based on the previous query in (a), it is known that Pravin, Kenneth, and Syafiq are patients that are under the age of 30 and came for a vaccination visit after 2011. To further verify the correctness of the final query that additionally checks for currently valid vaccinations, the query written in **Appendix 2** is executed with the IDs of the known valid patients specified in the HAVING statement instead. The system outputs all vaccination records of Pravin, Kenneth, and Syafiq without checking for vaccination effect validity.



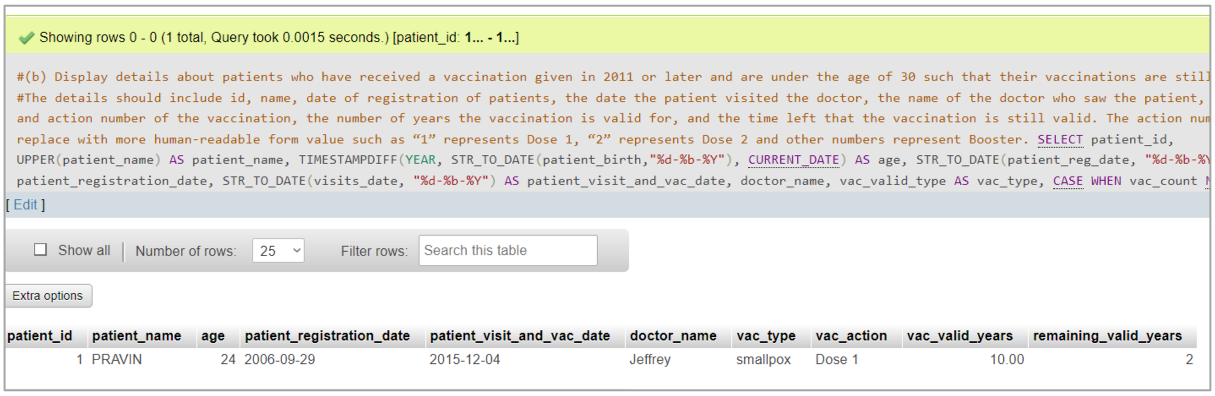
```
Showing rows 0 - 5 (6 total, Query took 0.0009 seconds.) [patient_id: 1... - 6...]

#(b) Display details about patients who have received a vaccination given in 2011 or later and are under the age of 30 such that their vaccinations are still valid.
#The details should include id, name, date of registration of patients, the date the patient visited the doctor, the name of the doctor who saw the patient, the name
and action number of the vaccination, the number of years the vaccination is valid for, and the time left that the vaccination is still valid. The action number will
replace with more human-readable form value such as “1” represents Dose 1, “2” represents Dose 2 and other numbers represent Booster. SELECT patient_id,
UPPER(patient_name) AS patient_name, TIMESTAMPDIFF(YEAR, STR_TO_DATE(patient_birth,"%d-%b-%Y"), CURRENT_DATE) AS age, STR_TO_DATE(patient_reg_date, "%d-%b-%Y") AS
patient_registration_date, STR_TO_DATE(visits_date, "%d-%b-%Y") AS patient_visit_and_vac_date, doctor_name, vac_valid_type AS vac_type, CASE WHEN vac_count NOT
Edit ]
```

patient_id 1	patient_name	age	patient_registration_date	patient_visit_and_vac_date 2	doctor_name	vac_type	vac_action 3	vac_valid_years	remaining_valid_years 4
1	PRAVIN	24	2006-09-29	2015-12-04	Jeffrey	smallpox	Dose 1	10.00	2
1	PRAVIN	24	2006-09-29	2015-12-04	Jeffrey	typhoid	Dose 2	3.00	-4
3	KENNETH CHONG	17	2011-01-01	2012-07-09	Bryan Lim	typhoid	Dose 1	3.00	-8
3	KENNETH CHONG	17	2011-01-01	2012-07-09	Bryan Lim	hepatitis	Dose 2	0.50	-10
3	KENNETH CHONG	17	2011-01-01	2016-01-28	Jeffrey	hepatitis	Dose 1	0.50	-6
6	SYAFIQ	27	2007-12-18	2015-09-01	Shareen Loh	typhoid	Dose 1	3.00	-5

***Figure 4**: Relation of all valid patients in terms of age and vaccination date regardless of vaccination effect validity.*

According to **Figure 4**, all except Pravin's first dose of smallpox vaccination have gone invalid. Therefore, it is expected that the final query selects Pravin's first dose of smallpox vaccination as the result. After fully executing the query written in **Appendix 2**, the system outputs the expected result.



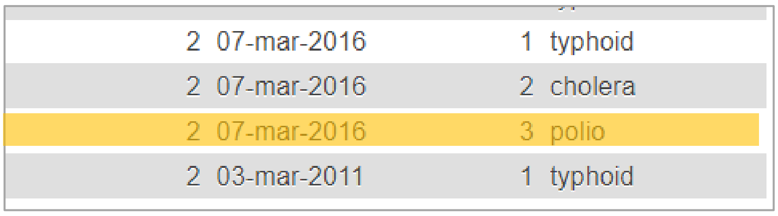
```
Showing rows 0 - 0 (1 total, Query took 0.0015 seconds.) [patient_id: 1... - 1...]

#(b) Display details about patients who have received a vaccination given in 2011 or later and are under the age of 30 such that their vaccinations are still
#The details should include id, name, date of registration of patients, the date the patient visited the doctor, the name of the doctor who saw the patient,
and action number of the vaccination, the number of years the vaccination is valid for, and the time left that the vaccination is still valid. The action num
replace with more human-readable form value such as “1” represents Dose 1, “2” represents Dose 2 and other numbers represent Booster. SELECT patient_id,
UPPER(patient_name) AS patient_name, TIMESTAMPDIFF(YEAR, STR_TO_DATE(patient_birth,"%d-%b-%Y"), CURRENT_DATE) AS age, STR_TO_DATE(patient_reg_date, "%d-%b-%
patient_registration_date, STR_TO_DATE(visits_date, "%d-%b-%Y") AS patient_visit_and_vac_date, doctor_name, vac_valid_type AS vac_type, CASE WHEN vac_count
[ Edit ]
```

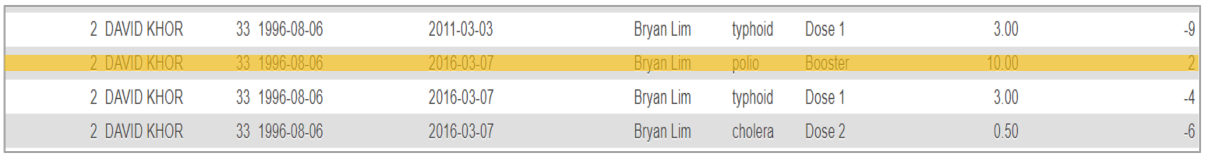
patient_id	patient_name	age	patient_registration_date	patient_visit_and_vac_date	doctor_name	vac_type	vac_action	vac_valid_years	remaining_valid_years
1	PRAVIN	24	2006-09-29	2015-12-04	Jeffrey	smallpox	Dose 1	10.00	2

***Figure 5**: Relation of patients valid in terms of age and vaccination date, and vaccination effect is still valid.*

The output includes all information specified in the requirements. Besides that, vaccination action numbers are also mapped to their corresponding output. Numbers 1 and 2 are mapped to “Dose 1” and “Dose 2” respectively, and other action numbers are mapped to “Booster”. **Figure 6** shows the mapping of other action numbers to “Booster” in the output, which is not shown in previous figures.



2	07-mar-2016	1	typhoid
2	07-mar-2016	2	cholera
2	07-mar-2016	3	polio
2	03-mar-2011	1	typhoid



2	DAVID KHOR	33	1996-08-06	2011-03-03	Bryan Lim	typhoid	Dose 1	3.00	-9
2	DAVID KHOR	33	1996-08-06	2016-03-07	Bryan Lim	polio	Booster	10.00	2
2	DAVID KHOR	33	1996-08-06	2016-03-07	Bryan Lim	typhoid	Dose 1	3.00	-4
2	DAVID KHOR	33	1996-08-06	2016-03-07	Bryan Lim	cholera	Dose 2	0.50	-6

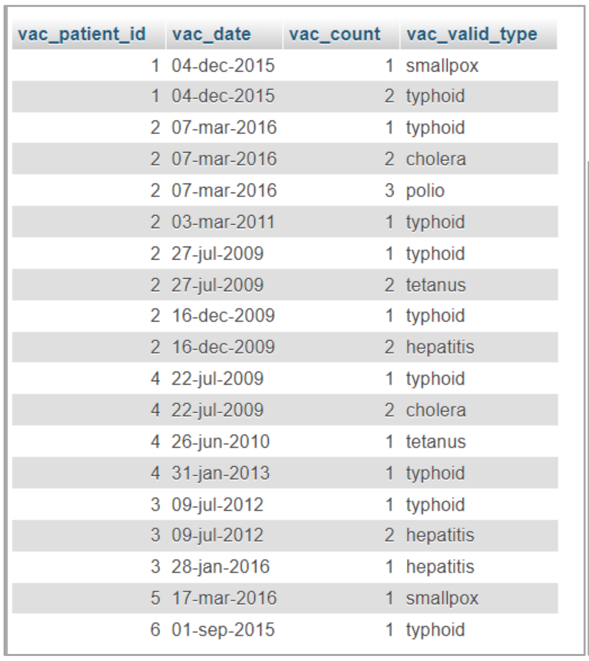
***Figure 6**: Mapping of other action number to “Booster”*

Task 2

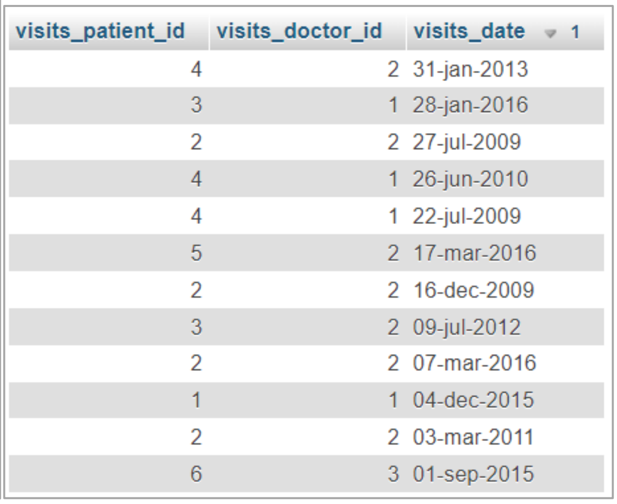
A stored procedure called add_vaccine with appropriate input parameters is required for Task 2. The procedure inserts relevant `vaccinations` and `visits` records to the database according to the given inputs. To prompt the users to enter relevant vaccination details, phpMyAdmin's DBMS user interface is used to easily prompt the user for inputs and execute the procedure. The following sections elaborates on the event handling and behaviors of the procedure, of which code is written in **Appendix 3**.

- **Add vaccine**

This is the basic usage of the add_vaccine procedure. To demonstrate the working functionality of the procedure, **Figure 7** is used to show the original `vaccinations` and `visits` relation data before the procedure is executed.



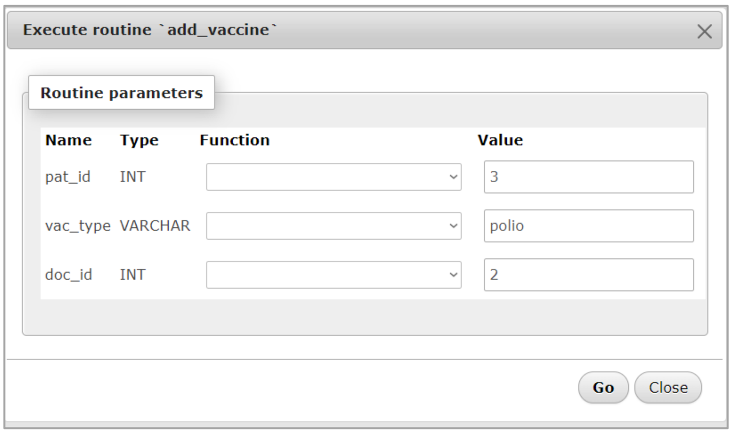
vac_patient_id	vac_date	vac_count	vac_valid_type
1	04-dec-2015	1	smallpox
1	04-dec-2015	2	typhoid
2	07-mar-2016	1	typhoid
2	07-mar-2016	2	cholera
2	07-mar-2016	3	polio
2	03-mar-2011	1	typhoid
2	27-jul-2009	1	typhoid
2	27-jul-2009	2	tetanus
2	16-dec-2009	1	typhoid
2	16-dec-2009	2	hepatitis
4	22-jul-2009	1	typhoid
4	22-jul-2009	2	cholera
4	26-jun-2010	1	tetanus
4	31-jan-2013	1	typhoid
3	09-jul-2012	1	typhoid
3	09-jul-2012	2	hepatitis
3	28-jan-2016	1	hepatitis
5	17-mar-2016	1	smallpox
6	01-sep-2015	1	typhoid



visits_patient_id	visits_doctor_id	visits_date ▾ 1
4	2	31-jan-2013
3	1	28-jan-2016
2	2	27-jul-2009
4	1	26-jun-2010
4	1	22-jul-2009
5	2	17-mar-2016
2	2	16-dec-2009
3	2	09-jul-2012
2	2	07-mar-2016
1	1	04-dec-2015
2	2	03-mar-2011
6	3	01-sep-2015

***Figure 7**: Full `vaccinations` and `visits` relation data before procedure execution*

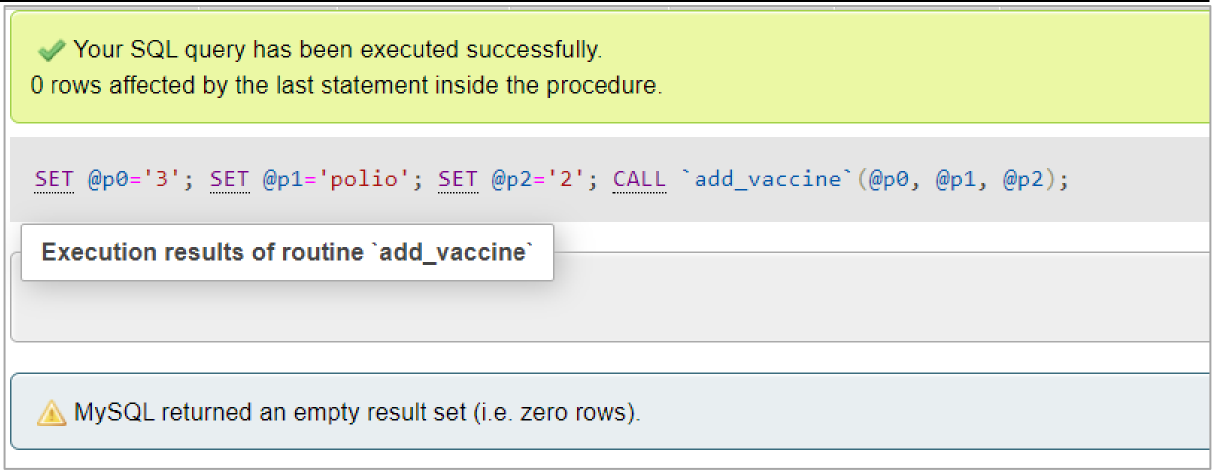
To add a vaccination record, the user is required to enter and specify values for the patient's ID, the responsible doctor's ID, and the vaccination type as shown in **Figure 8**. Once done, they can execute the procedure by clicking on the 'Go' button to call the procedure with the parameter values provided.



Execute routine `add_vaccine`

Routine parameters

Name	Type	Function	Value
pat_id	INT		3
vac_type	VARCHAR		polio
doc_id	INT		2



Your SQL query has been executed successfully.
0 rows affected by the last statement inside the procedure.

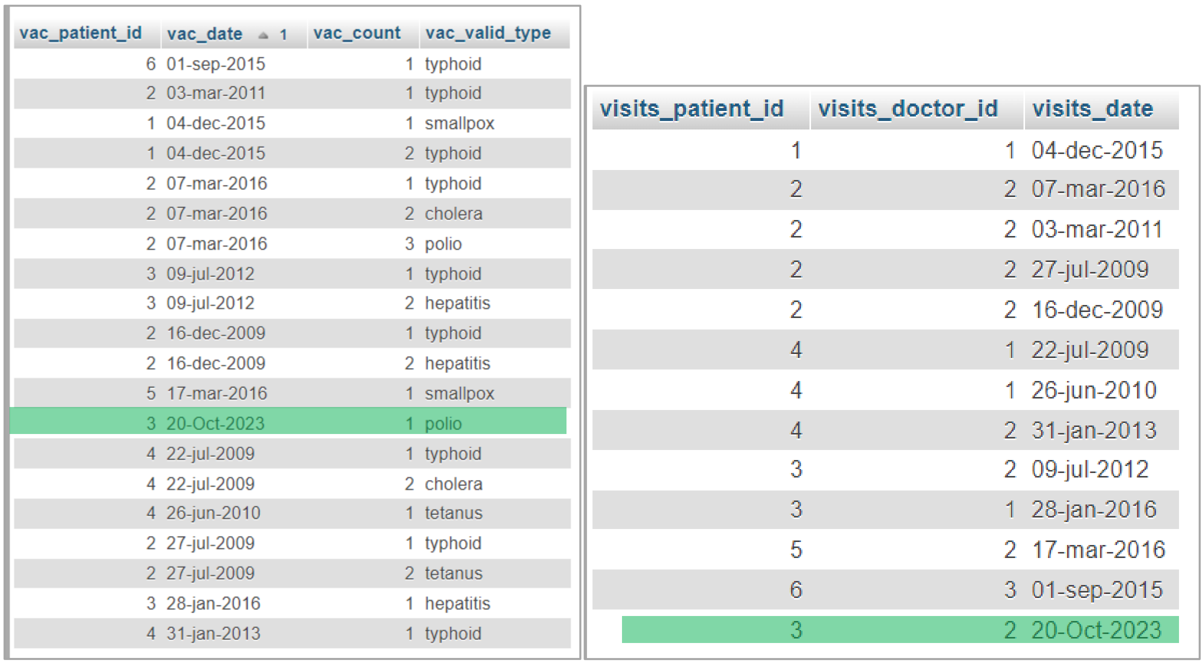
```
SET @p0='3'; SET @p1='polio'; SET @p2='2'; CALL `add_vaccine`(@p0, @p1, @p2);
```

Execution results of routine `add_vaccine`

MySQL returned an empty result set (i.e. zero rows).

***Figure 8**: Stored procedure execution*

As a result as shown in **Figure 9**, a new vaccination record that contains the user-specified values have been inserted into the `vaccinations` relation, and another new patient visit record is inserted into the `visits` relation if the patient-date pair still does not exist.



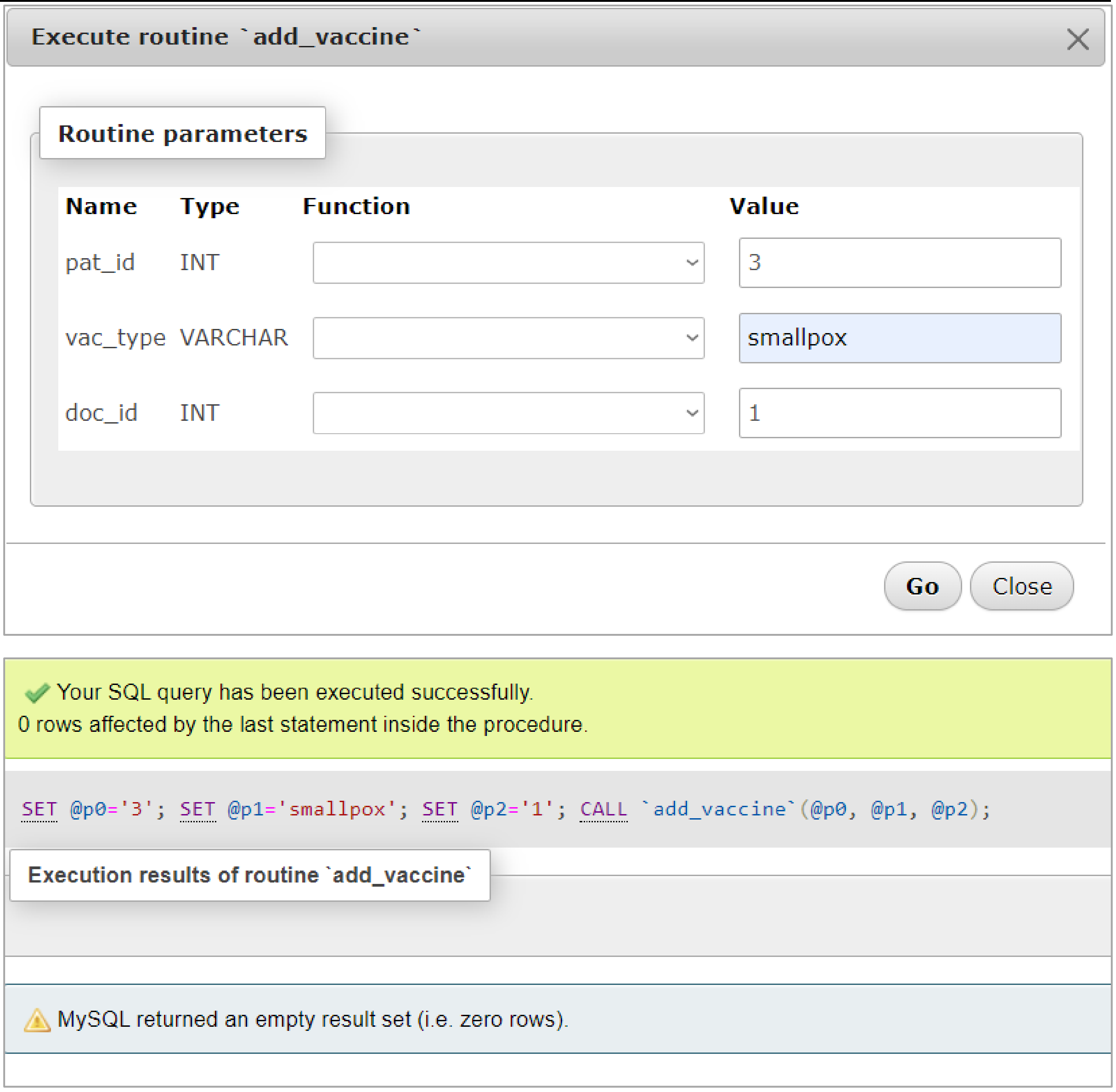
vac_patient_id	vac_date ▲ 1	vac_count	vac_valid_type
6	01-sep-2015	1	typhoid
2	03-mar-2011	1	typhoid
1	04-dec-2015	1	smallpox
1	04-dec-2015	2	typhoid
2	07-mar-2016	1	typhoid
2	07-mar-2016	2	cholera
2	07-mar-2016	3	polio
3	09-jul-2012	1	typhoid
3	09-jul-2012	2	hepatitis
2	16-dec-2009	1	typhoid
2	16-dec-2009	2	hepatitis
5	17-mar-2016	1	smallpox
3	20-Oct-2023	1	polio
4	22-jul-2009	1	typhoid
4	22-jul-2009	2	cholera
4	26-jun-2010	1	tetanus
2	27-jul-2009	1	typhoid
2	27-jul-2009	2	tetanus
3	28-jan-2016	1	hepatitis
4	31-jan-2013	1	typhoid

visits_patient_id	visits_doctor_id	visits_date
1	1	04-dec-2015
2	2	07-mar-2016
2	2	03-mar-2011
2	2	27-jul-2009
2	2	16-dec-2009
4	1	22-jul-2009
4	1	26-jun-2010
4	2	31-jan-2013
3	2	09-jul-2012
3	1	28-jan-2016
5	2	17-mar-2016
6	3	01-sep-2015
3	2	20-Oct-2023

***Figure 9**: Result of procedure execution*

- **Auto-incrementing vaccination count**

The procedure also keeps tracks of the number of vaccinations that a patient has received on the current date. To demonstrate using the previous example above, a new vaccination record is added for the same patient with ID 3 with another vaccination type in **Figure 10**.



Execute routine `add_vaccine`

Routine parameters

Name	Type	Function	Value
pat_id	INT		3
vac_type	VARCHAR		smallpox
doc_id	INT		1

Go Close

Your SQL query has been executed successfully.
0 rows affected by the last statement inside the procedure.

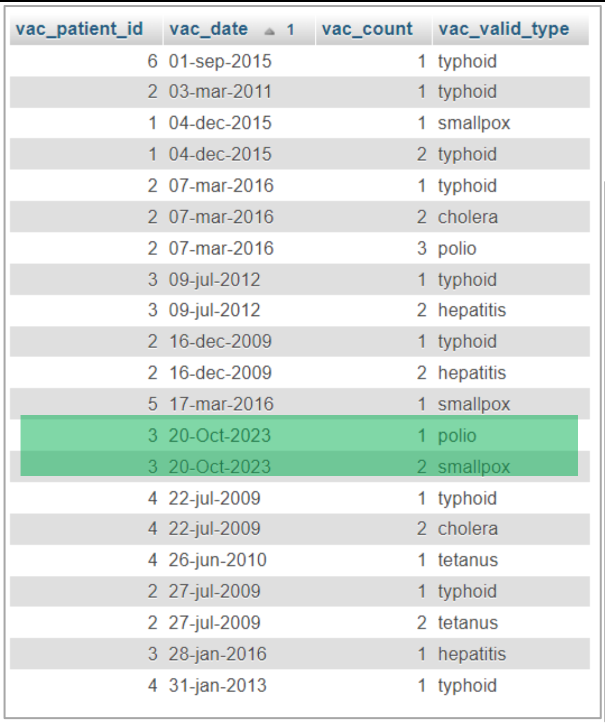
```
SET @p0='3'; SET @p1='smallpox'; SET @p2='1'; CALL `add_vaccine`(@p0, @p1, @p2);
```

Execution results of routine `add_vaccine`

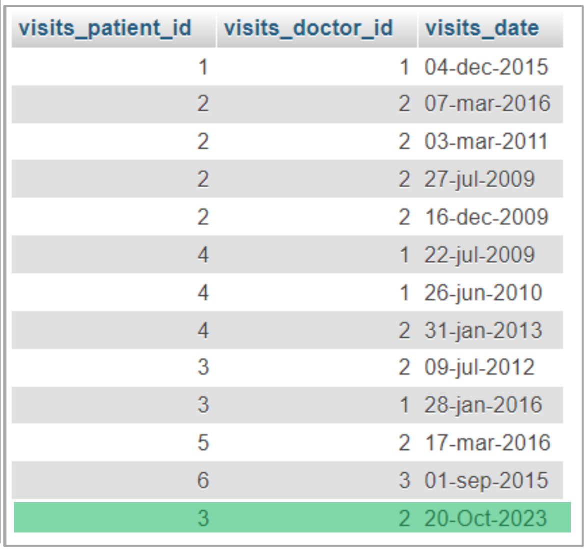
MySQL returned an empty result set (i.e. zero rows).

***Figure 10**: Procedure executed again with same `pat_id` but different values for other parameters.*

As shown in the result of **Figure 11**, the `vac_count` attribute of the new record in the `vaccination` relation is set to 2 instead of starting from 1. This is because the same patient had previously received a vaccination on the same date. As a side note, the new doctor ID specified from the prompt which still does not exist in the `visits` relation for the current date is not recorded due to an assumption of only a single doctor being on duty in each day, and therefore is regarded as a misinput and ignored.



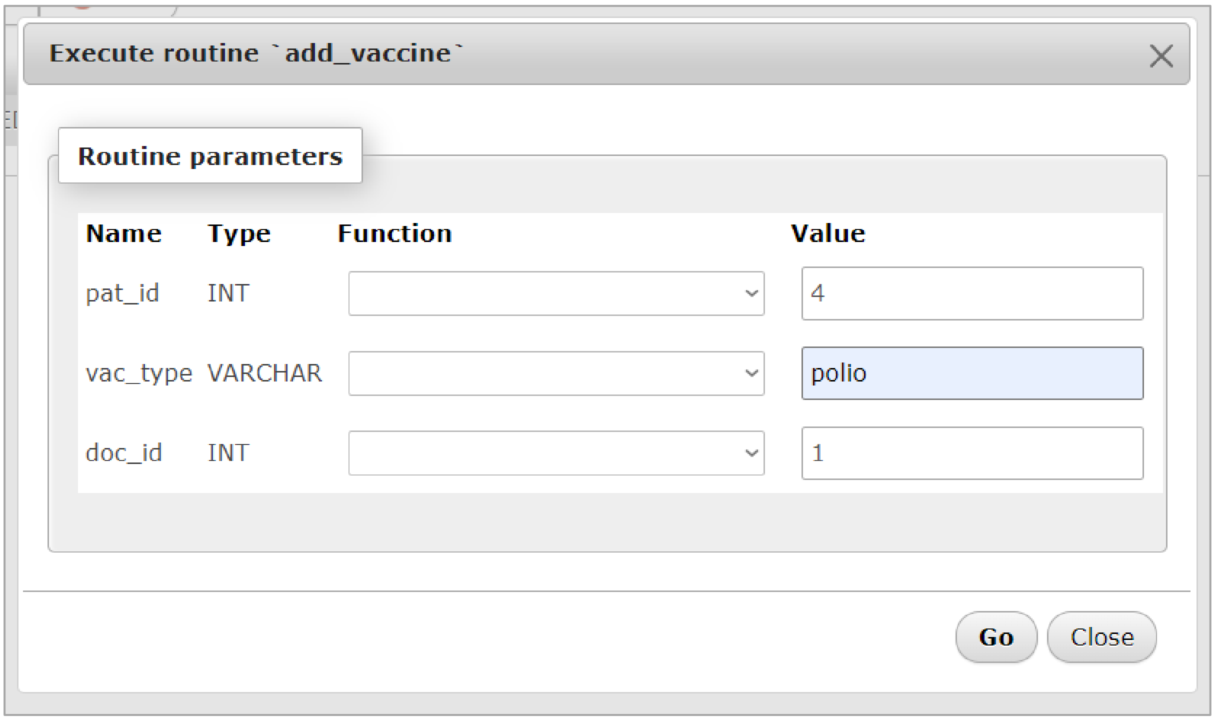
vac_patient_id	vac_date ▲ 1	vac_count	vac_valid_type
6	01-sep-2015	1	typhoid
2	03-mar-2011	1	typhoid
1	04-dec-2015	1	smallpox
1	04-dec-2015	2	typhoid
2	07-mar-2016	1	typhoid
2	07-mar-2016	2	cholera
2	07-mar-2016	3	polio
3	09-jul-2012	1	typhoid
3	09-jul-2012	2	hepatitis
2	16-dec-2009	1	typhoid
2	16-dec-2009	2	hepatitis
5	17-mar-2016	1	smallpox
3	20-Oct-2023	1	polio
3	20-Oct-2023	2	smallpox
4	22-jul-2009	1	typhoid
4	22-jul-2009	2	cholera
4	26-jun-2010	1	tetanus
2	27-jul-2009	1	typhoid
2	27-jul-2009	2	tetanus
3	28-jan-2016	1	hepatitis
4	31-jan-2013	1	typhoid



visits_patient_id	visits_doctor_id	visits_date
1	1	04-dec-2015
2	2	07-mar-2016
2	2	03-mar-2011
2	2	27-jul-2009
2	2	16-dec-2009
4	1	22-jul-2009
4	1	26-jun-2010
4	2	31-jan-2013
3	2	09-jul-2012
3	1	28-jan-2016
5	2	17-mar-2016
6	3	01-sep-2015
3	2	20-Oct-2023

Figure 11*: `vac_count` attribute of new record is auto-incremented from previous count (1), `visits` relation is unchanged.*

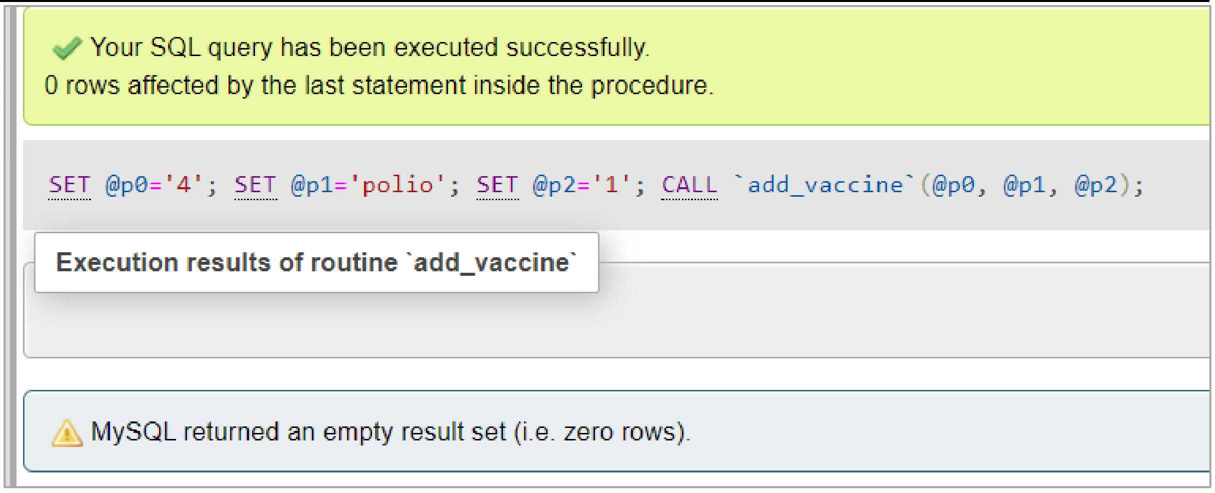
If the user immediately adds another vaccination record for a different patient, the system checks for the new patient's vaccination count separately from the previous patient and sets the value for them accordingly. This is shown in **Figure 12,** where the `vac_count` attribute of the new vaccination record for patient with ID 4 is set to "1" because they have not received any vaccinations at the current date, and do not continue from the previous vaccination count (count 2 for patient ID 3).



Execute routine `add_vaccine`

Routine parameters

Name	Type	Function	Value
pat_id	INT		4
vac_type	VARCHAR		polio
doc_id	INT		1

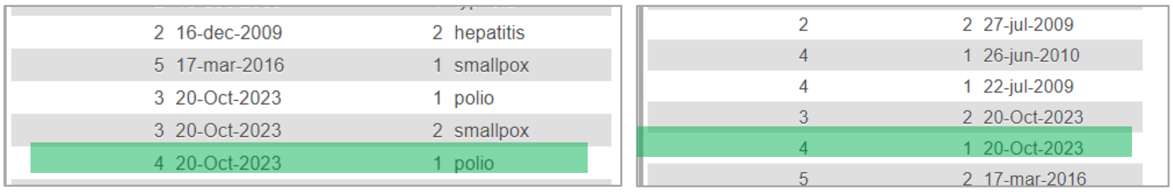


Your SQL query has been executed successfully.
0 rows affected by the last statement inside the procedure.

```
SET @p0='4'; SET @p1='polio'; SET @p2='1'; CALL `add_vaccine`(@p0, @p1, @p2);
```

Execution results of routine `add_vaccine`

MySQL returned an empty result set (i.e. zero rows).



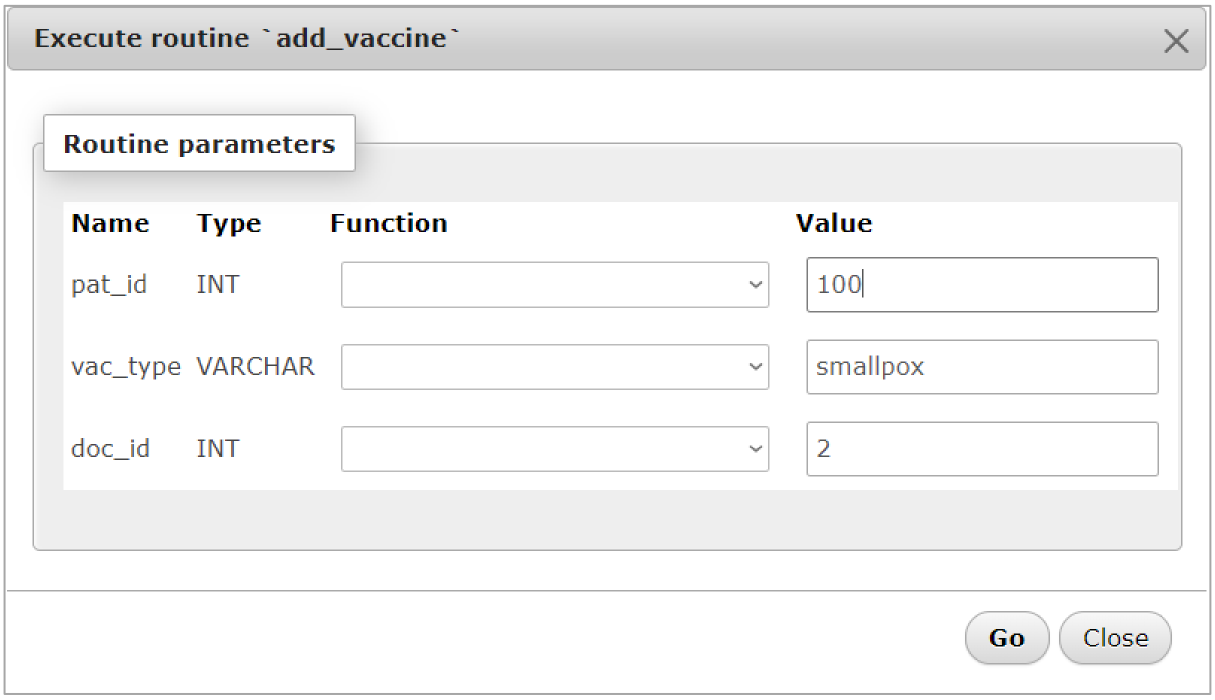
2	16-dec-2009	2	hepatitis
5	17-mar-2016	1	smallpox
3	20-Oct-2023	1	polio
3	20-Oct-2023	2	smallpox
4	20-Oct-2023	1	polio

2	2	27-jul-2009
4	1	26-jun-2010
4	1	22-jul-2009
3	2	20-Oct-2023
4	1	20-Oct-2023
5	2	17-mar-2016

***Figure 12**: New record with different `pat_id` tracks separate `vac_count` value, `visits` relation records new patient visit date.*

- **Patient ID validation**

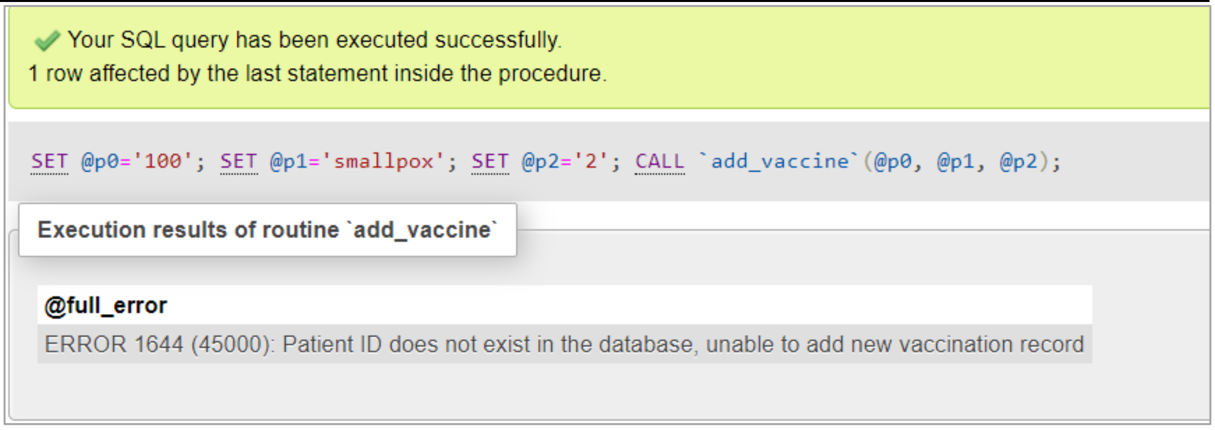
The procedure validates for the patient ID before inserting new records into the `vaccinations` and `visits` table. This is done to ensure that reference integrity to the `patient` relation is not violated. **Figure 13** shows that the procedure throws an error message when the user tries to set the `pat_id` parameter to 100, which does not exist in the `patient` relation.



Execute routine `add_vaccine`

Routine parameters

Name	Type	Function	Value
pat_id	INT		100
vac_type	VARCHAR		smallpox
doc_id	INT		2



Your SQL query has been executed successfully.
1 row affected by the last statement inside the procedure.

```
SET @p0='100'; SET @p1='smallpox'; SET @p2='2'; CALL `add_vaccine`(@p0, @p1, @p2);
```

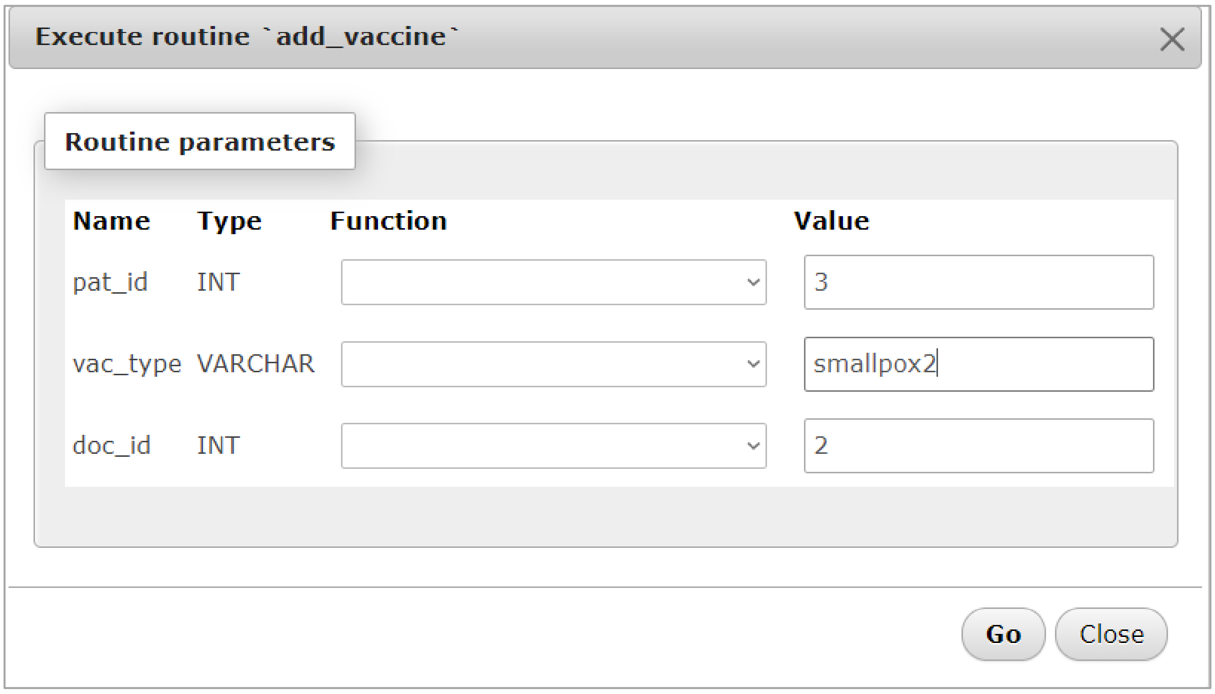
Execution results of routine `add_vaccine`

@full_error
ERROR 1644 (45000): Patient ID does not exist in the database, unable to add new vaccination record

***Figure 13**: Validation of patient ID in the `add_vaccine` procedure.*

- **Vaccination type validation**

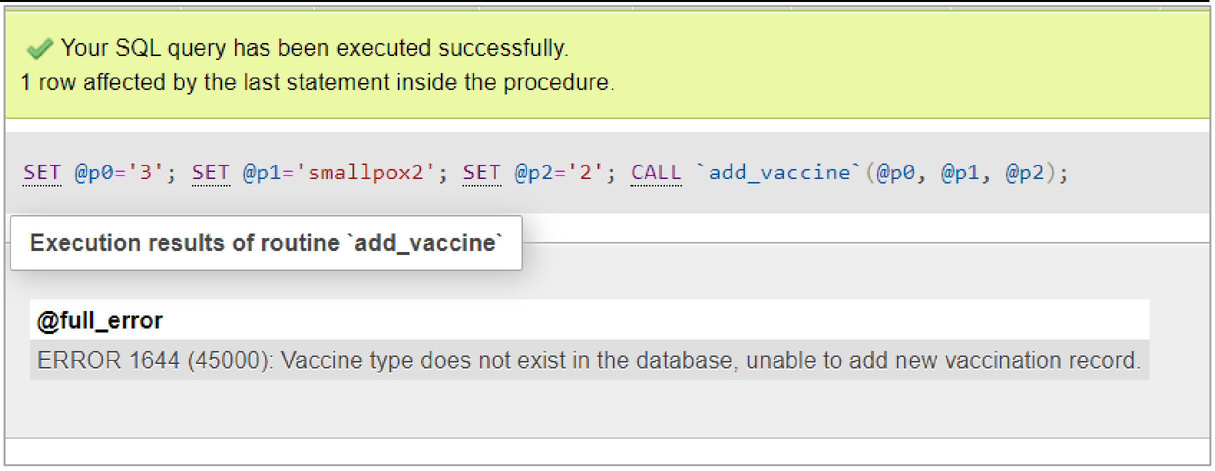
The procedure validates for the vaccination type before inserting new records into the `vaccinations` table. This is done to ensure that reference integrity to the `valid_for` relation is not violated. **Figure 14** shows that the procedure throws an error message when the user tries to set the `vac_type` parameter to “smallpox2”, which does not exist in the `valid_for` relation.



Execute routine `add_vaccine`

Routine parameters

Name	Type	Function	Value
pat_id	INT		3
vac_type	VARCHAR		smallpox2
doc_id	INT		2



Your SQL query has been executed successfully.
1 row affected by the last statement inside the procedure.

```
SET @p0='3'; SET @p1='smallpox2'; SET @p2='2'; CALL `add_vaccine`(@p0, @p1, @p2);
```

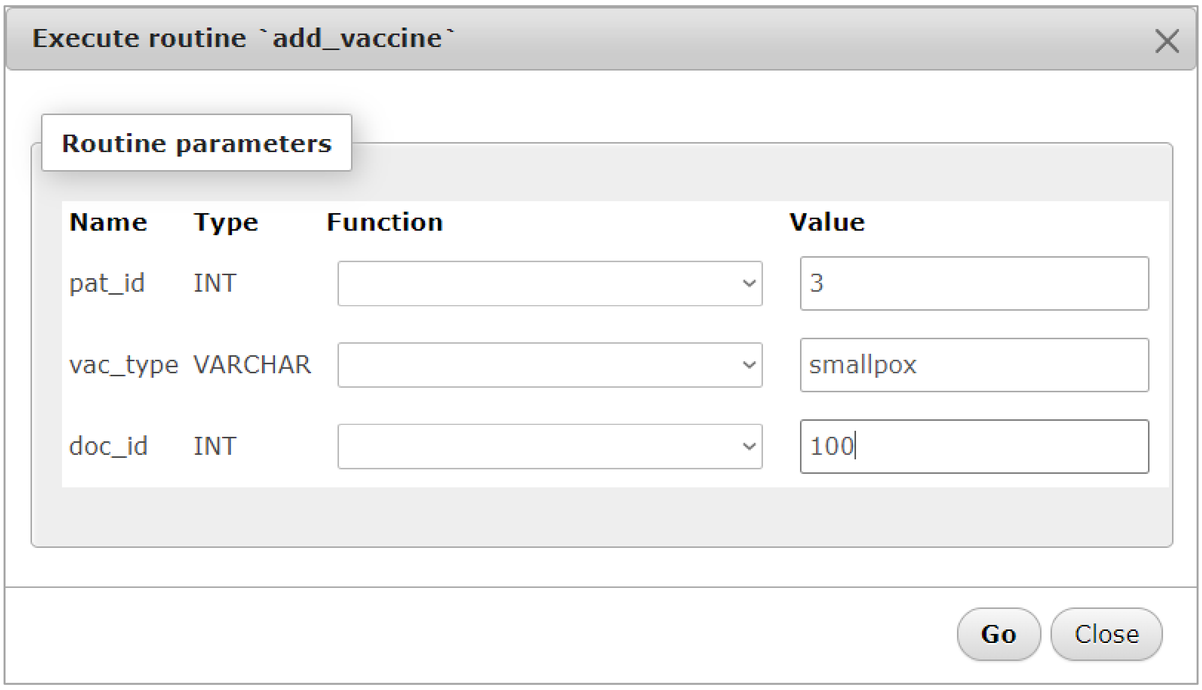
Execution results of routine `add_vaccine`

@full_error
ERROR 1644 (45000): Vaccine type does not exist in the database, unable to add new vaccination record.

***Figure 14**: Validation of vaccine type in the `add_vaccine` procedure.*

- **Doctor ID validation**

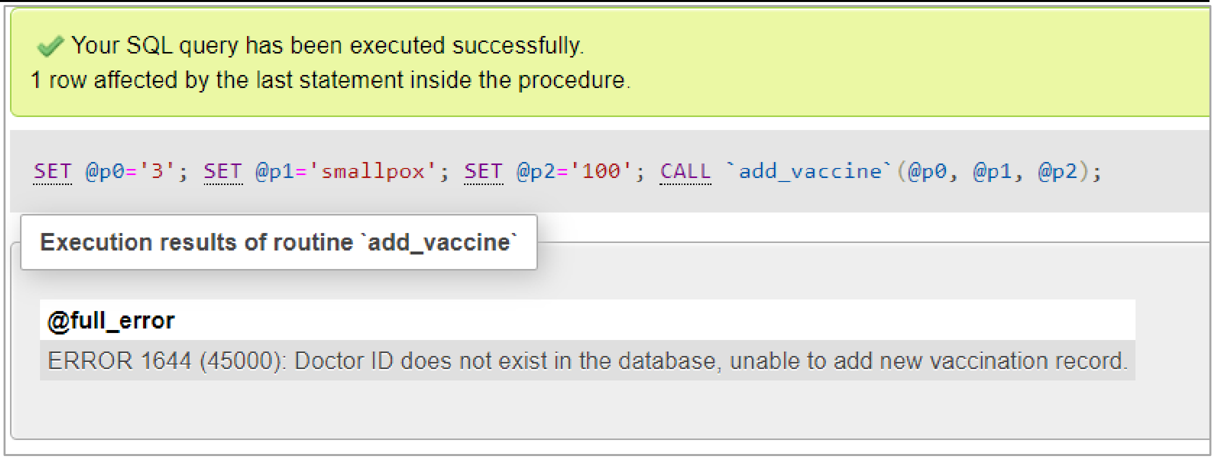
The procedure validates for the doctor ID before inserting new records into the `visits` table. This is done to ensure that reference integrity to the `doctor` relation is not violated. **Figure 15** shows that the procedure throws an error message when the user tries to set the `doc_id` parameter to 100, which does not exist in the `doctor` relation.



Execute routine `add_vaccine`

Routine parameters

Name	Type	Function	Value
pat_id	INT		3
vac_type	VARCHAR		smallpox
doc_id	INT		100



Your SQL query has been executed successfully.
1 row affected by the last statement inside the procedure.

```
SET @p0='3'; SET @p1='smallpox'; SET @p2='100'; CALL `add_vaccine`(@p0, @p1, @p2);
```

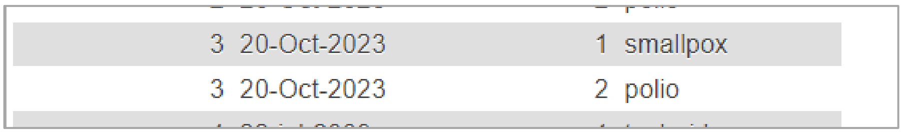
Execution results of routine `add_vaccine`

@full_error
ERROR 1644 (45000): Doctor ID does not exist in the database, unable to add new vaccination record.

***Figure 15**: Validation of doctor ID in the `add_vaccine` procedure.*

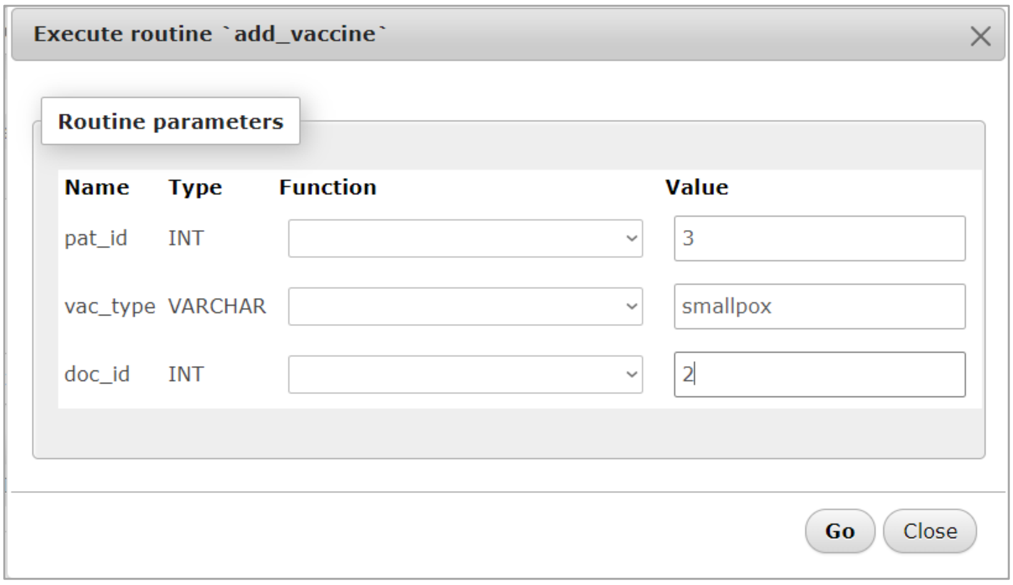
- **Daily vaccination quota**

In conformance to the procedure requirements, the procedure also checks for the daily vaccination quota of the target patient when adding new vaccination records for them. Patients can only receive a maximum of 2 vaccinations daily. To demonstrate, the procedure throws an error message in **Figure 17** when a user tries to add a new vaccination record for patient with ID 3 after 2 records have been added at the current date as shown in **Figure 16**, even though the entered inputs are valid and legal.



3	20-Oct-2023	1	smallpox
3	20-Oct-2023	2	polio

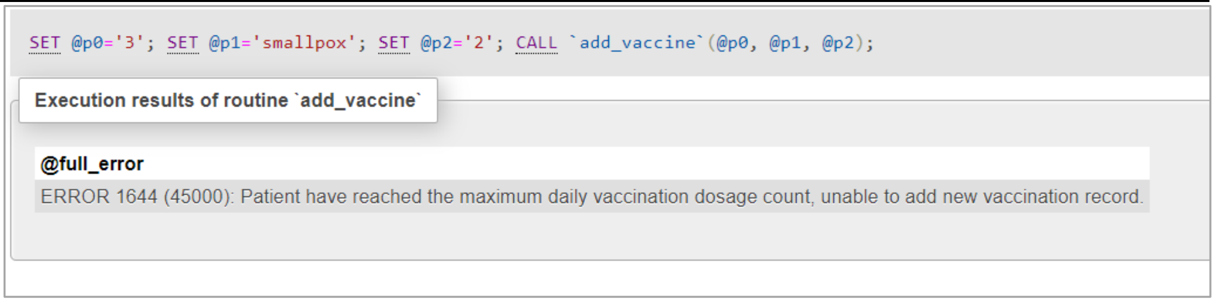
***Figure 16**: Existence of 2 vaccination records for patient ID '3'.*



Execute routine `add_vaccine`

Routine parameters

Name	Type	Function	Value
pat_id	INT		3
vac_type	VARCHAR		smallpox
doc_id	INT		2



```
SET @p0='3'; SET @p1='smallpox'; SET @p2='2'; CALL `add_vaccine`(@p0, @p1, @p2);
```

```
@full_error
ERROR 1644 (45000): Patient have reached the maximum daily vaccination dosage count, unable to add new vaccination record.
```

***Figure 17**: Validation of daily vaccination quota in the `add_vaccine` procedure.*

Appendix

Appendix 1: Task 1(a) query

```
##(a)   Display id, name, date of birth, today’s date, and age in years for
those patients who are under the age of 30 and who have received a given
vaccination in 2011 or later. The name should be displayed in upper case. All
dates should be displayed in full (e.g., 31-Jan-2011). The age in years should
be a whole number, i.e., any fractions should be removed.

SELECT DISTINCT patient_id AS id, UPPER(patient_name) AS name,
DATE_FORMAT(STR_TO_DATE(patient_birth,"%d-%b-%Y"), "%d-%b-%Y") AS
date_of_birth, DATE_FORMAT(CURRENT_DATE, "%d-%b-%Y") AS today,
TIMESTAMPDIFF(YEAR, STR_TO_DATE(patient_birth,"%d-%b-%Y"), CURRENT_DATE) AS
age
FROM patient INNER JOIN vaccinations ON (patient_id = vac_patient_id)
GROUP BY vac_date
HAVING age < 30 AND YEAR(STR_TO_DATE(vac_date,"%d-%b-%Y")) >= 2011
ORDER BY id;
```

Appendix 2: Task 1(b) query

```
#(b)    Display details about patients who have received a vaccination given
in 2011 or later and are under the age of 30 such that their vaccinations are
still valid.

#The details should include id, name, date of registration of patients, the
date the patient visited the doctor, the name of the doctor who saw the
patient, the name and action number of the vaccination, the number of years
the vaccination is valid for, and the time left that the vaccination is still
valid. The action number will replace with more human-readable form value such
as “1” represents Dose 1, “2” represents Dose 2 and other numbers represent
Booster.


SELECT patient_id, UPPER(patient_name) AS patient_name, TIMESTAMPDIFF(YEAR,
STR_TO_DATE(patient_birth,"%d-%b-%Y"), CURRENT_DATE) AS age,
STR_TO_DATE(patient_reg_date, "%d-%b-%Y") AS patient_registration_date,
STR_TO_DATE(visits_date, "%d-%b-%Y") AS patient_visit_and_vac_date,
doctor_name, vac_valid_type AS vac_type,
CASE
    WHEN vac_count NOT IN(1,2) THEN "Booster"
    WHEN vac_count = 1 THEN "Dose 1"
    ELSE "Dose 2"
END AS vac_action,
valid_volume AS vac_valid_years, TIMESTAMPDIFF(YEAR, CURRENT_DATE,
DATE_ADD(STR_TO_DATE(vac_date, "%d-%b-%Y"), INTERVAL valid_volume YEAR)) AS
remaining_valid_years
FROM patient INNER JOIN vaccinations ON (patient_id = vac_patient_id)
```

```
             INNER JOIN visits ON (patient_id = visits_patient_id AND vac_date
= visits_date)
             INNER JOIN doctor ON (visits_doctor_id = doctor_id)
             INNER JOIN valid_for ON (vac_valid_type = valid_type)
HAVING YEAR(patient_visit_and_vac_date) >= 2011 AND
       age < 30 AND
       remaining_valid_years > 0
ORDER BY patient_id, patient_visit_and_vac_date, vac_action,
remaining_valid_years;
```

Appendix 3: Task 2 stored procedure

```
#Write a stored procedure called add_vaccine with appropriate input
parameters. Execute this procedure will insert both vaccination and visit
records to the database. You are required to prompt the users to enter
relevant vaccination details with appropriate messages. You are also required
to add the following requirements in this procedure:

DELIMITER //
CREATE PROCEDURE add_vaccine(IN pat_id INT(10), IN vac_type VARCHAR(255), IN
doc_id INT(10))
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        GET DIAGNOSTICS CONDITION 1
            @sqlstate = RETURNED_SQLSTATE,
            @errno = MYSQL_ERRNO,
            @text = MESSAGE_TEXT;

        SET @full_error = CONCAT("ERROR ", @errno, " (", @sqlstate, "): ",
@text);
        SELECT @full_error;
    END;

    #   Verify reference integrity for 3 inputs
    SELECT
        COUNT(*) INTO @isPatientFound
    FROM patient
    WHERE patient_id = pat_id;

    IF (@isPatientFound = 0) THEN
        SIGNAL SQLSTATE
            '45000'
        SET
            MESSAGE_TEXT = "Patient ID does not exist in the database, unable
to add new vaccination record";
    END IF;
```

```sql
    SELECT
        COUNT(*) INTO @isDoctorFound
    FROM doctor
    WHERE doctor_id = doc_id;

    IF (@isDoctorFound = 0) THEN
        SIGNAL SQLSTATE
            '45000'
        SET
            MESSAGE_TEXT = "Doctor ID does not exist in the database, unable
to add new vaccination record.";
    END IF;

    SELECT
        COUNT(*) INTO @isVacTypeFound
    FROM valid_for
    WHERE valid_type = vac_type;

    IF (@isVacTypeFound = 0) THEN
        SIGNAL SQLSTATE
            '45000'
        SET
            MESSAGE_TEXT = "Vaccine type does not exist in the database,
unable to add new vaccination record.";
    END IF;

    #   A business rule that no more than two vaccinations are allowed per
patient per day.
    SELECT
        MAX(vac_count) INTO @sameDayVaccinationCount
    FROM vaccinations
    WHERE vac_patient_id = pat_id AND STR_TO_DATE(vac_date, "%d-%b-%Y") =
CURRENT_DATE;

    IF (@sameDayVaccinationCount > 1) THEN
        SIGNAL SQLSTATE
            '45000'
        SET
            MESSAGE_TEXT = "Patient have reached the maximum daily vaccination
dosage count, unable to add new vaccination record.";
    END IF;

    #Check if patient-date pair exist in visits
    SELECT
         COUNT(*) INTO @isVisitRecorded
    FROM visits
    WHERE visits_patient_id = pat_id AND STR_TO_DATE(visits_date, "%d-%b-%Y")
= CURRENT_DATE;
```

```sql

    #INSERT for visits relation only if current visit does not exist yet
    IF (@isVisitRecorded = 0) THEN
        INSERT INTO visits (visits_patient_id, visits_doctor_id, visits_date)
        VALUES (pat_id, doc_id, DATE_FORMAT(CURRENT_DATE, "%d-%b-%Y"));
    END IF;

    #-  A business rule that the first vaccination for a given patient on a
given visit date has an action number 1 and the second vaccination
have         action no 2 on the same date for the same patient.

    SELECT
        CASE
            WHEN MAX(vac_count) IS NULL THEN 1
            ELSE 2
        END INTO @newActionNumber
    FROM vaccinations
    WHERE vac_patient_id = pat_id AND STR_TO_DATE(vac_date, "%d-%b-%Y") =
CURRENT_DATE;


    #INSERT for vaccination relation
    INSERT INTO vaccinations (vac_patient_id, vac_date, vac_count,
vac_valid_type)
    VALUES (pat_id, DATE_FORMAT(CURRENT_DATE, "%d-%b-%Y"), @newActionNumber,
vac_type);
END//
DELIMITER ;
```

CDB3033N DATABSE PROGRAMMING

MARKING RUBRIC

ASSIGNMENT [1]

SCRIPTING & STORED PROCEDURE

Section (1)-12%

LEARNING OUTCOME	MARKING CRITERIA	SCALE							
		Fail (0-49)	3rd Class (50-59)	2nd Lower Class (60-69)	2nd Upper Class (70-79)	1st Class (80-100)	YOUR MARKS/COMMENTS		
							100 %	Weightage	Actual Marks
CLO1	**Script (a)** **(20%)**	The script implemented with major flaws	The script implemented with some flaws	The script implemented with minor flaws	Good implementation in the script but not in exceptional way	The script implemented with excellent result and fulfil all the assignment requirements		0.2	
	Script (b) **(20%)**	The script implemented with major flaws	The script implemented with some flaws	The script implemented with minor flaws	Good implementation in the script but not in exceptional way	The script implemented with excellent result and fulfil all the assignment requirements		0.2	
						Total (40%)			

CDB3033N DATABSE PROGRAMMING

MARKING RUBRIC

ASSIGNMENT [1]

SCRIPTING & STORED PROCEDURE

Section (2)-18%

LEARNING OUTCOME	MARKING CRITERIA	SCALE							
							YOUR MARKS/COMMENTS		
		Fail (0-49)	**3rd Class (50-59)**	**2nd Lower Class (60-69)**	**2nd Upper Class (70-79)**	**1st Class (80-100)**	**100 %**	**Weightage**	**Actual Marks**
CLO2	**Stored Procedure** **(50%)**	The procedure implemented with major flaws	The procedure implemented with some flaws	The procedure implemented with minor flaws	Good implementation in the procedure but not in exceptional way	The procedure implemented with excellent result and fulfil all the assignment requirements		0.5	
	Screen shots & Test Cases **(10%)**	Minor or no screen shots and test cases provided.	Some screen shots and test cases provided but with some flaws	Appropriate screen shots and test cases provided but with some flaws	Good screen shots and test cases provided but with minor flaws	Excellent screen shots and test cases provided with clear explanation		0.1	
						Total (60%)			
						Overall Score (100%)			