*Case Study(Report)*

‘Accounts Payable Database’

***Student Details:***

* ***Rishabh Rajpurohit (9919103096)***
* ***Soumya Gupta (9919103120)***
* ***P. Paul Jonathan (9919103121)***



## Database Systems and Web Lab

## Jaypee Institute of Information Technology Noida

#### July 11, 2021

# ACKNOWLEDGEMENT

Presentation inspiration and motivation have always played a key role in the success of any venture.

We pay our deep sense of gratitude to, **Vice Chancellor Dr. S C Saxena**, **Jaypee institute of information technology, Noida** to encourage us to the highest peak and to provide us the opportunity to prepare the project. We are immensely obliged to for their elevating inspiration, encouraging guidance and kind supervision in the completion of our project.

We feel to acknowledge our indebtedness and deep sense of gratitude to our guide ***Mrs. Vartika Puri*** whose invaluable guidance and kind supervision given to us throughout the course which shaped the present work as its show.

Last, but not the least, **our parents** are also an important inspiration for time. So with due regards. We express our gratitude.

***TABLE OF CONTENT***

[ACKNOWLEDGEMENT 2](#_Toc3392)

[1.](#_Toc16108) **[PART:1](#_Toc16108)** [5](#_Toc16108)

*[1.1 TITLE / EER](#_Toc20042)* [5](#_Toc20042)

*[1.2 APPLICATIONS](#_Toc15546)* [6](#_Toc15546)

[2.](#_Toc29565) **[PART:2](#_Toc29565)** [7](#_Toc29565)

*[2.1 CREATING RELATIONS](#_Toc8416)* [7](#_Toc8416)

[2.2](#_Toc26786) *[INSERTING VALUES](#_Toc26786)* [8](#_Toc26786)

[3.](#_Toc4813) **[PART:3](#_Toc4813)** [12](#_Toc4813)

*[3.1 QUERY: TRIGGER 1](#_Toc31849)* [12](#_Toc31849)

*[3.2 QUERY: TRIGGER 2](#_Toc489)* [13](#_Toc489)

*[3.3 QUERY: TRIGGER 2](#_Toc31327)* [14](#_Toc31327)

*[3.4 QUERY: PROCEDURE 1](#_Toc17996)* [15](#_Toc17996)

*[3.5 QUERY: PROCEDURE 2](#_Toc4621)* [17](#_Toc4621)

*[3.6 QUERY: PROCEDURE 3](#_Toc29151)* [21](#_Toc29151)

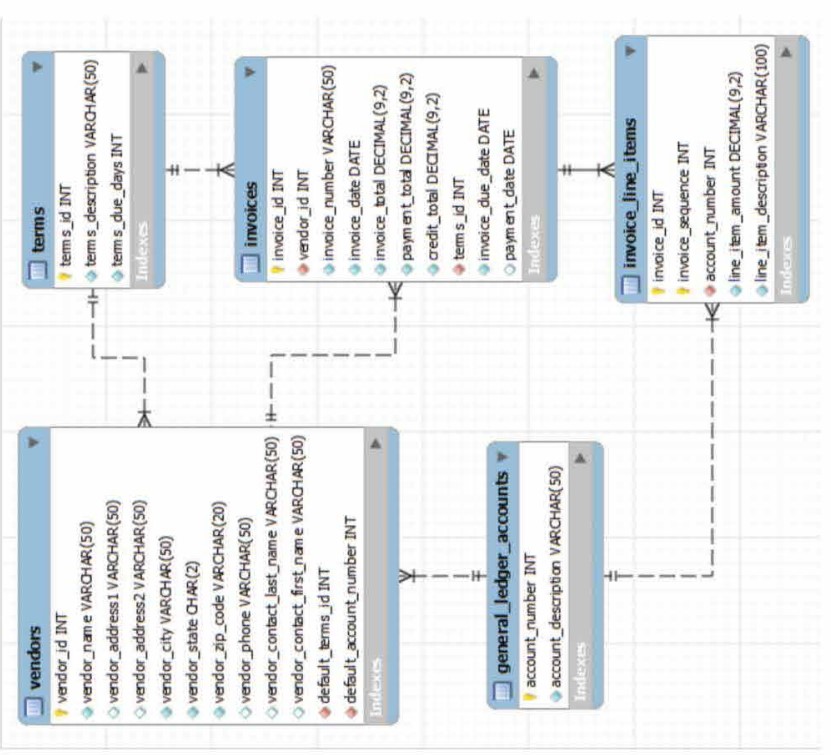
*[3.7 QUERY: SCHEDULED EVENT 1](#_Toc21528)* [24](#_Toc21528)

[4.](#_Toc1938) **[BIBLIOGRAPHY](#_Toc1938)** [25](#_Toc1938)

## PART:1

### *1.1 TITLE / EER*

*‘Accounts Payable’* Database

  
***EER*** *Diagram for the database*

### *1.2 APPLICATIONS*

By Implementing the above project in real time database:

* we can automate the manual stuff like keeping records of due payments
* sending alerts for upcoming payments
* charging penalties
* updating late fees daily automatically and
* offering discounts wherever needed

Thus this reduces the scope of error and greatly reduces human efforts.

## PART:2

### *CREATING RELATIONS*

use dswproject;

create table invoices(

  invoice\_id int primary key AUTO\_INCREMENT,

  vendor\_id int not null,

  invoice\_number VARCHAR(50) not null,

  invoice\_date date not null,

  invoice\_total decimal(9, 2) default 0,

  credit\_total decimal(9, 2) DEFAULT 0,

  terms\_id int not null,

  invoice\_due\_date date,

  payment\_date date

);

CREATE table terms(

  terms\_id INT PRIMARY key AUTO\_INCREMENT,

  terms\_description VARCHAR(50) NOT NULL,

  terms\_due\_days int

);

CREATE TABLE vendors(

  vendor\_id int PRIMARY key AUTO\_INCREMENT,

  vendor\_name VARCHAR (50) not null,

  vendor\_address1 VARCHAR (50) not null,

  vendor\_address2 VARCHAR (50),

  vendor\_city VARCHAR (50),

  vendor\_state char(20),

  vendor\_zip\_code VARCHAR (20),

  vendor\_phone VARCHAR (50),

  vendor\_contact\_last\_name VARCHAR (50),

  vendor\_contact\_first\_name VARCHAR (50),

  default\_terms\_id int not null,

  default\_account\_number int not null

);

CREATE table general\_ledger\_accounts(

  account\_number int PRIMARY KEY AUTO\_INCREMENT,

  account\_description VARCHAR (50)

);

CREATE TABLE invoice\_line\_items(

    invoice\_id int not null,

    invoice\_sequence int not null,

    account\_number int not null,

    line\_item\_amount decimal(9,2) not null,

    line\_item\_description VARCHAR (100),

    PRIMARY key(invoice\_id, invoice\_sequence),

    FOREIGN key(account\_number) REFERENCES general\_ledger\_accounts (account\_number)

);

alter table invoices

add constraint invoices\_fk\_vendors FOREIGN key (vendor\_id) REFERENCES vendors (vendor\_id),

add constraint invoices\_fk\_terms FOREIGN key (terms\_id) REFERENCES terms (terms\_id);

ALTER TABLE vendors

add constraint vendors\_fk\_terms FOREIGN key (default\_terms\_id) REFERENCES terms (terms\_id),

add constraint vendors\_fk\_accounts FOREIGN key (default\_account\_number) REFERENCES general\_ledger\_accounts (account\_number);

### *INSERTING VALUES*

INSERT INTO

`general\_ledger\_accounts`

VALUES

(1, 'Illo quia ipsum itaque sit iste.'),(2, 'Itaque eum et occaecati non repellendus.'),(3, 'Sunt labore quia iure optio.'),(4,'Iusto dolor aut mollitia eius impedit asperiores.'

),(5,'Voluptas dolorem enim ut omnis temporibus nulla.'

), -- 45 records more

INSERT INTO

`invoice\_line\_items`

VALUES

(103777,91708,1,986.41,'Distinctio et omnis impedit voluptatibus.'

),(109188,18779,24,6477.80,'Fuga aut consequatur ipsa sunt earum.'

),(114164,20835,34,1348369.18,'Sunt magnam veritatis architecto id dignissimos.'

),(138716,23728,35,0.00,'Architecto debitis autem libero aliquid sapiente.'

),(142781,76299,15,1165.00,'Non reiciendis molestias dolorum quaerat voluptatem consequatur sit.'

), --45 records more

INSERT INTO

`invoices` VALUES

(1,1,'43637','1991-04-24',1900.14,0.00,1,'1995-11-13','2006-02-04'),

(2,2,'450','1996-05-02',5899197.04,2047011.50,2,'1974-07-02','1971-06-08'),

(3,3,'4765','2003-08-05',78.71,9999999.99,3,'2006-12-23','2015-09-23'),

(4,4,'4627186','1998-10-06',3813.58,36568.31,4,'2007-03-13','2003-08-08'),

(5,5,'','1996-07-30',0.00,50261.92,5,'1992-10-23',

'2014-10-31'), --45 records more

INSERT INTO

`terms`

VALUES

(1, 'Quas qui aut occaecati.', 2019),(

2,

'Nihil modi omnis dolorum voluptate eligendi repell',

1972

),(

3,

'Sed qui rerum consequatur sit reprehenderit modi.',

2008

),(4, 'Sint maiores consequatur sed.', 1989),(

5,

'Cumque illum et qui iusto praesentium sit enim.',

2007

), --45 records more

INSERT INTO

`vendors`

VALUES

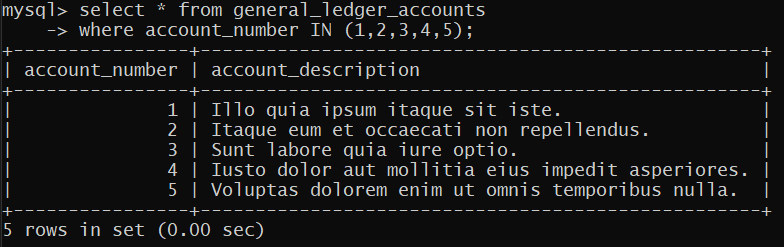
(1,'Carter Jerde MD','50300 Rosanna Point Suite 746','74703 Rohan Views','Cassandrehaven','Wisconsin','460878','07965412824','139.711.9651','01868411465',1,1),

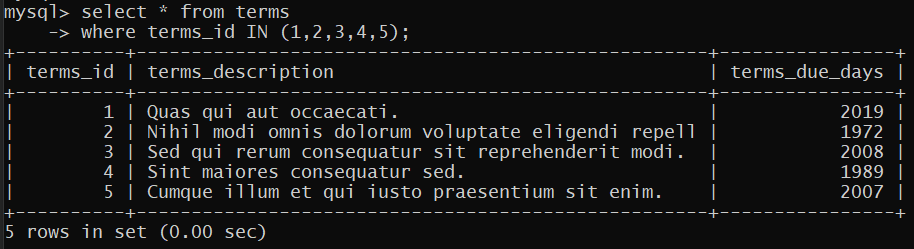
(2,'Prof. Judd Harris','0745 Streich Court Suite 482','08065 Graham Court Suite 773','Moriahville','Indiana','411798','1-279-686-9444','+41(3)2483486604','(190)302-9905x54471',2,2),

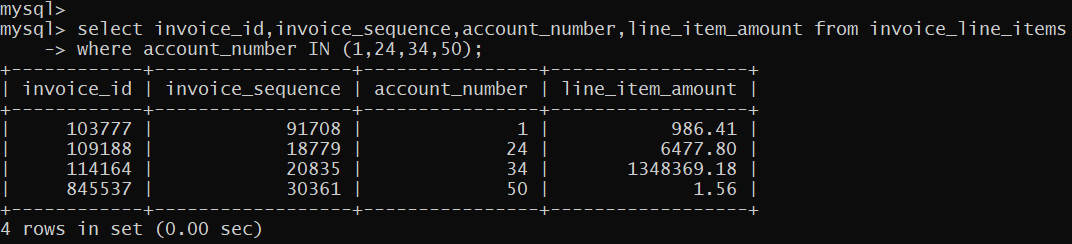
(3,'Dandre Koepp','9256 Lockman Corner Suite 353','535 Mortimer Haven Suite 537','Lake Estevan','Vermont','09986359834','832026','392-932-6974x475','336.615.1911x199',3,3),

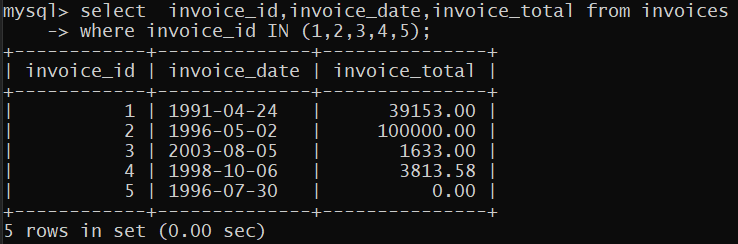
(4,'Dan Lesch','4378 Corkery Lock Suite 432','64436 Simonis Lights','California','West Anastasiaberg','133-511-4358','883743','852-915-4148x546','1-797-074-8152',4,4),

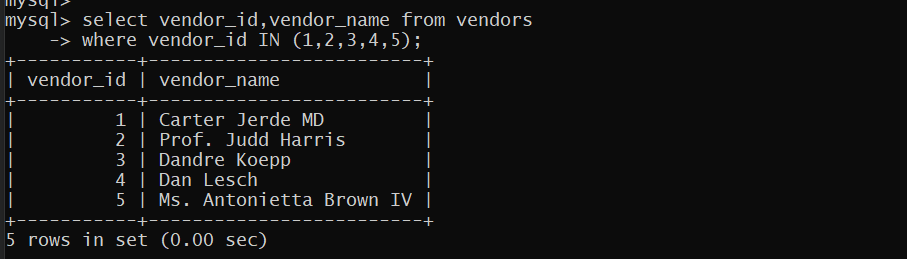
(5,'20214 Aidan Highway','Ms. Antonietta Brown IV','Lake Annamarievill','453 Sipes Mills Apt. 200','387820','Pennsylvania','864.316.0417','(138)284-0952x3786','1-497-025-9384',5,5), --45 records more











PTO

## PART:3

### *QUERY: TRIGGER 1*

/\*1 trigger: BEFORE delete on terms table as it is foreign key constrained\*/

DELIMITER //

CREATE TRIGGER deleteTermsNotAllowed

BEFORE DELETE ON terms FOR EACH ROW

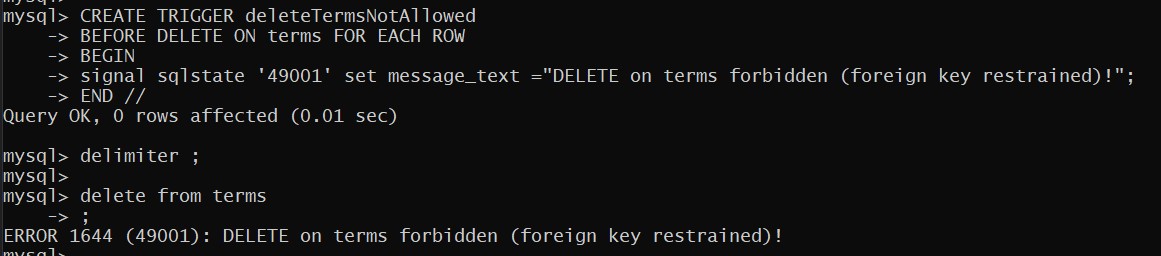
BEGIN

signal sqlstate '49001' SET message\_text = "DELETE on terms forbidden (foreign key restrained)!";

END //

DELIMITER ;

DELETE FROM terms;



### *3.2 QUERY: TRIGGER 2*

/\*2 trigger: BEFORE delete on general Ledger Accounts table as it is foreign key constrained\*/

DELIMITER //

CREATE TRIGGER deleteAccountsNotAllowed

BEFORE DELETE ON general\_ledger\_accounts FOR EACH ROW

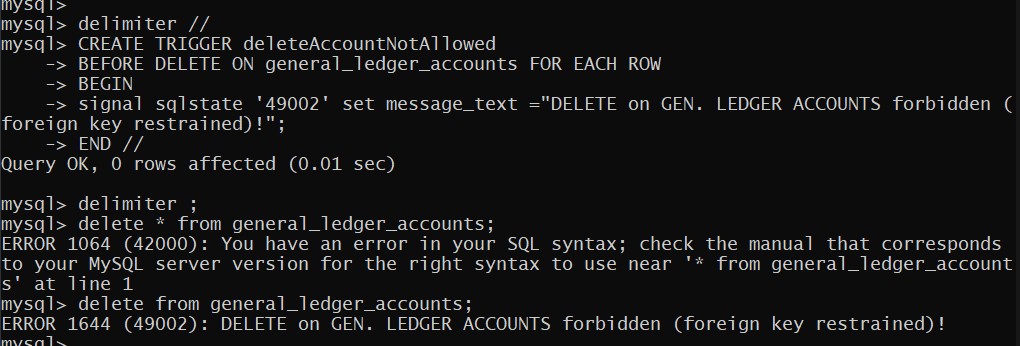
BEGIN

signal sqlstate '49002' SET message\_text = "DELETE on general\_ledger\_accounts forbidden (foreign key restrained)!";

END //

DELIMITER ;

DELETE FROM general\_ledger\_accounts;

**

### *3.3 QUERY: TRIGGER 2*

/\*3 trigger: BEFORE INSERT on invoices table discount 10% if invoice\_total > 100000 \*/

DELIMITER //

CREATE TRIGGER updateInvoiceDiscount

BEFORE INSERT ON invoices FOR EACH ROW

BEGIN

IF new.invoice\_total > 100000 THEN

SET new.invoice\_total = (new.invoice\_total)\*0.9;

END IF;

END //

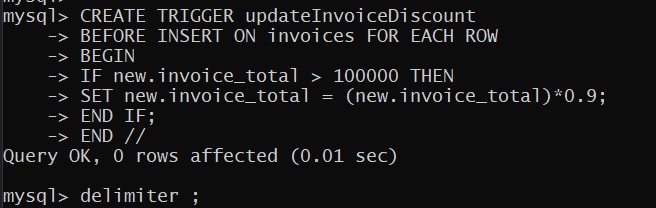
DELIMITER ;

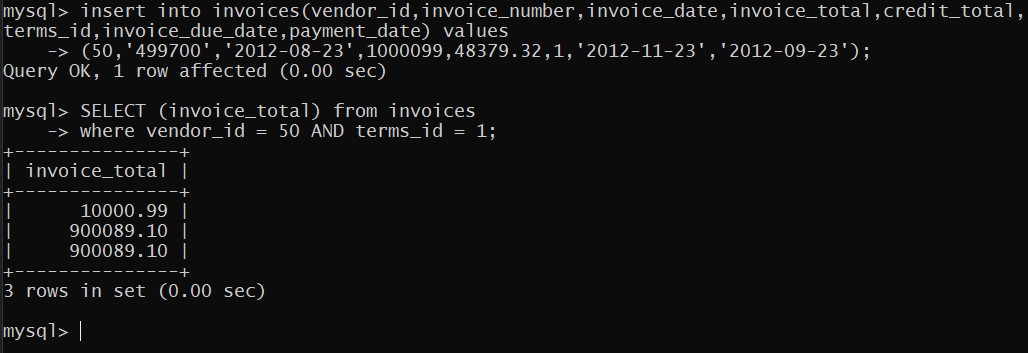
INSERT INTO invoices(vendor\_id,invoice\_number,invoice\_date,invoice\_total,credit\_total,terms\_id,invoice\_due\_date,payment\_date)

VALUES (50,'499700','2012-08-23',1000099,48379.32,1,'2012-11-23','2012-09-23');

SELECT (invoice\_total) FROM invoices

WHERE vendor\_id = 50 AND terms\_id = 1;





### *3.4 QUERY: PROCEDURE 1*

/\*4 Procedure: To get names of all vendors\*/

DELIMITER //

CREATE PROCEDURE getvendors(INOUT vendorslist VARCHAR (10000))

BEGIN

DECLARE isdone INT DEFAULT 0;

DECLARE vendorname varchar(50);

DECLARE vendorcursor CURSOR FOR

SELECT vendor\_name from vendors;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET isdone = 1;

OPEN vendorcursor;

vendorloop: LOOP

FETCH vendorcursor INTO vendorname;

IF isdone = 1 THEN

LEAVE vendorloop;

END IF;

SET vendorslist = CONCAT(vendorname, ", ", vendorslist);

END LOOP vendorloop;

CLOSE vendorcursor;

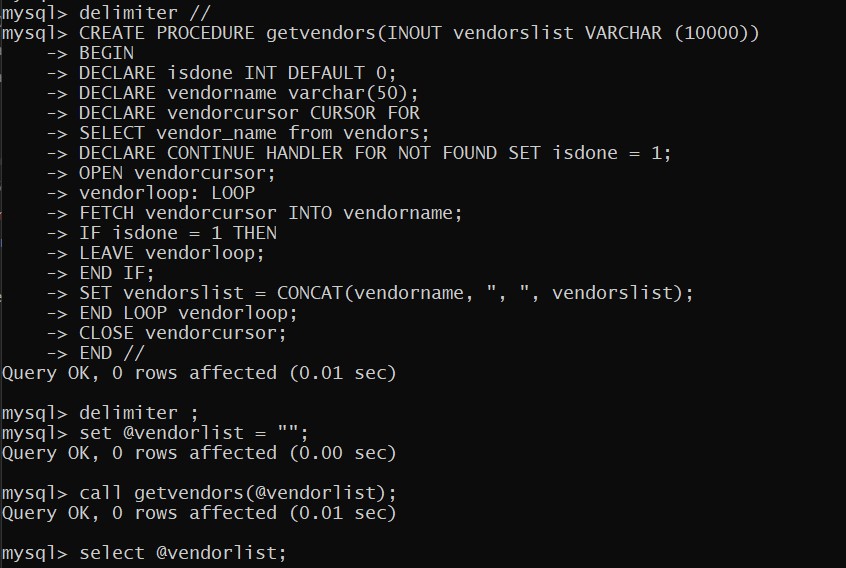
END //

DELIMITER ;

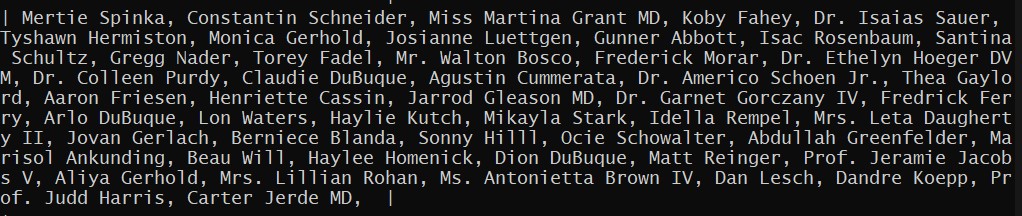
SET @vendorlist = "";

CALL getvendors(@vendorlist);

SELECT @vendorlist;



***OUTPUT:***



PTO

### *3.5 QUERY: PROCEDURE 2*

/\*5: procedure to add late fees\*/

-- there were no dates greater than current date so added a few

update invoices

SET payment\_date = '2021-10-21'

WHERE invoice\_id IN (1,3,5,7,9);

-- some invoice\_total values were offlimit, thus

update invoices

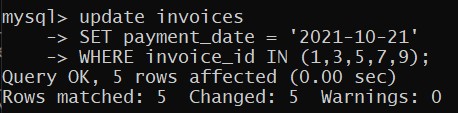
SET invoice\_total = 100000

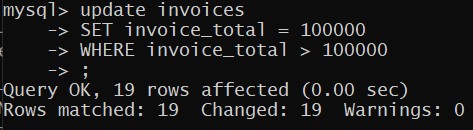
WHERE invoice\_total > 100000

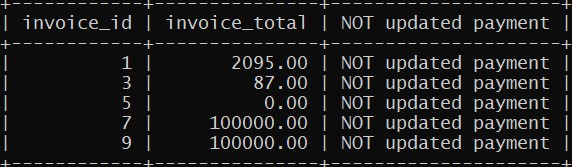
-- glimpse of values about to be updated

select invoice\_id,invoice\_total,'NOT updated payment' from invoices

WHERE invoice\_id IN (1,3,5,7,9);

**

**

**

-- procedure

DELIMITER //

CREATE procedure addlatepayment()

BEGIN

DECLARE isdone INT DEFAULT 0;

DECLARE invid INT DEFAULT 0;

DECLARE invtotal INT DEFAULT 0;

DECLARE oldpayment INT DEFAULT 0;

DECLARE invpdate date DEFAULT 0;

DECLARE invcursor CURSOR FOR

SELECT invoice\_id,payment\_date,invoice\_total FROM invoices;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET isdone = 1;

OPEN invcursor;

invloop: LOOP

FETCH invcursor INTO invid,invpdate,invtotal;

SET oldpayment = invtotal;

IF isdone = 1 THEN

LEAVE invloop;

END IF;

IF invpdate > CURDATE() THEN

SET invtotal = invtotal \* 1.05;

UPDATE invoices

SET invoice\_total = invtotal

WHERE invoice\_id = invid;

SELECT invoice\_id as ID,oldpayment as PREVIOUS\_TOTAL,invoice\_total as UPDATED\_TOTAL,'UPDATED' as STATUS from invoices

WHERE invoice\_id = invid;

END IF;

END LOOP invloop;

SELECT 'Late dues added successfully';

CLOSE invcursor;

END //

DELIMITER ;

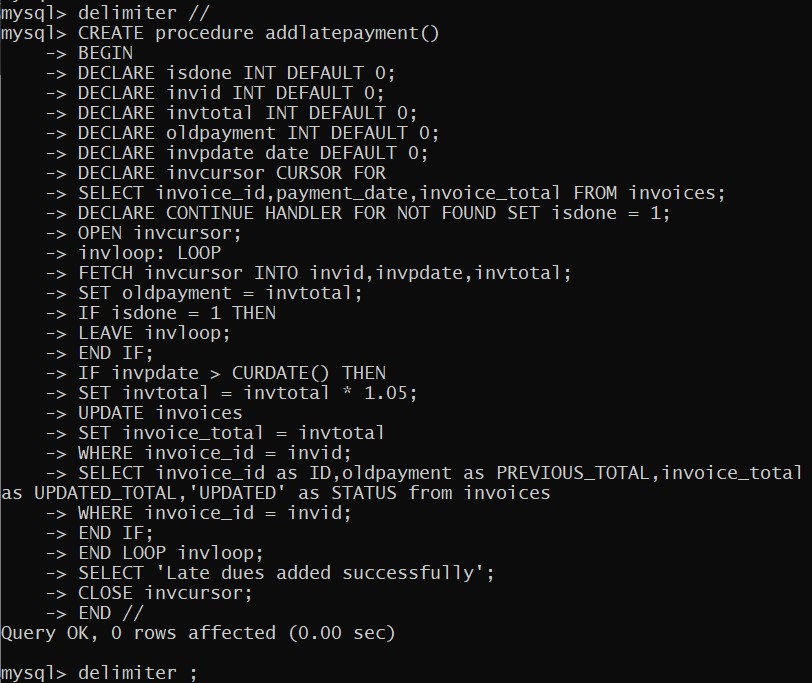
-- calling PROCEDURE

call addlatepayment();

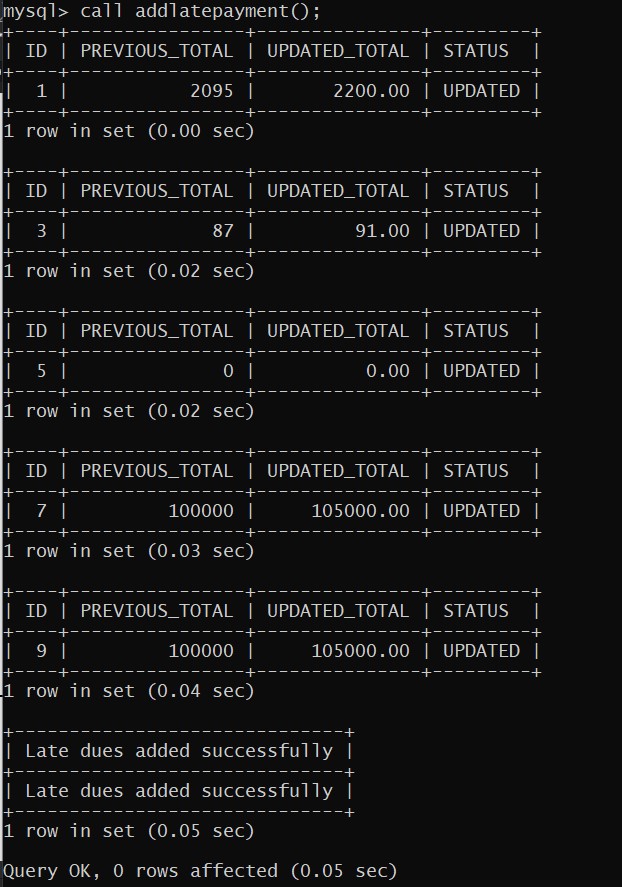
--actually whole table is traversed but

--here only those tuples shown which were actually updated

PTO



PTO



### *3.6 QUERY: PROCEDURE 3*

/\*6: procedure to show due payments\*/

-- procedure

DELIMITER //

CREATE procedure showduepayments()

BEGIN

DECLARE isdone INT DEFAULT 0;

DECLARE invid INT DEFAULT 0;

DECLARE invtotal INT DEFAULT 0;

DECLARE invpdate date DEFAULT 0;

DECLARE cursorinv CURSOR FOR

SELECT invoice\_id,payment\_date,invoice\_total FROM invoices;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET isdone = 1;

OPEN cursorinv;

invloop: LOOP

FETCH cursorinv INTO invid,invpdate,invtotal;

IF isdone = 1 THEN

LEAVE invloop;

END IF;

IF invpdate > CURDATE() THEN

SELECT invoice\_id as ID,payment\_date,invoice\_total as DUE\_INVOICE,'PENDING' as STATUS from invoices

WHERE invoice\_id = invid;

END IF;

END LOOP invloop;

CLOSE cursorinv;

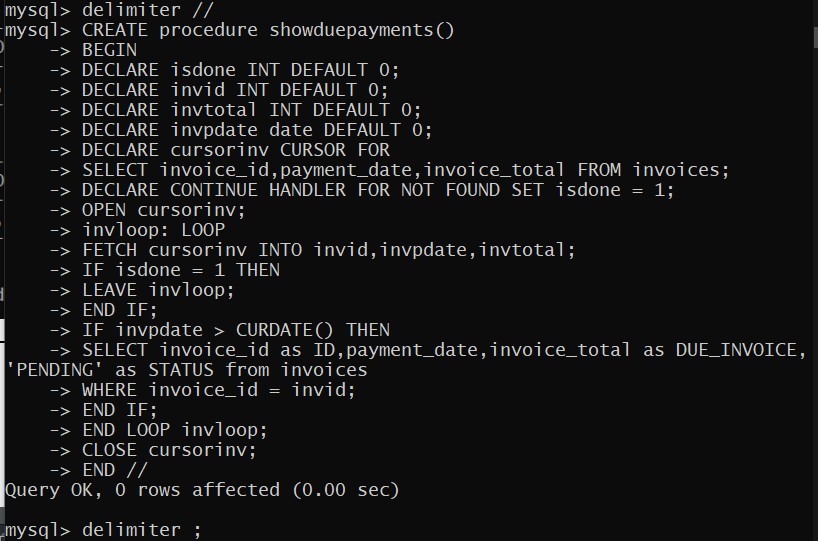
END //

DELIMITER ;

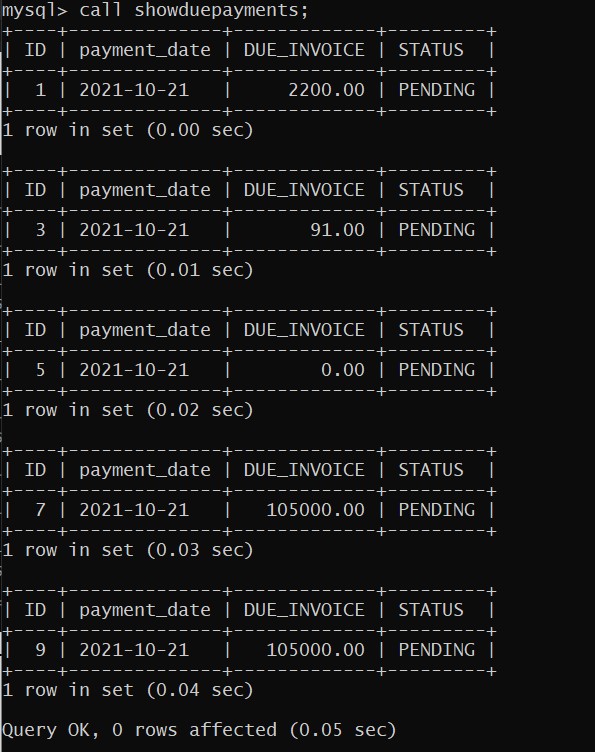
-- calling PROCEDURE

call showduepayments();

PTO

**

*PTO*

**

PTO

### *3.7 QUERY: SCHEDULED EVENT 1*

/\*7: Timestamp trigger to run procedure addlatepayment() daily once\*/

-- to run scheduled events event organiser should be ON

-- set it to 0 to turn it OFF

SET GLOBAL event\_scheduler = 1;

-- sample event to see results if working

CREATE EVENT updatepayment

ON SCHEDULE EVERY 5 SECOND

DO

CALL dswproject.addlatepayment();

select invoice\_id,invoice\_total from invoices

where invoice\_id IN (1,3,5,7,9);

-- repeat after 5 seconds

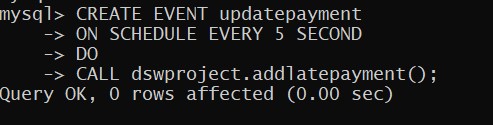
select invoice\_id,invoice\_total from invoices

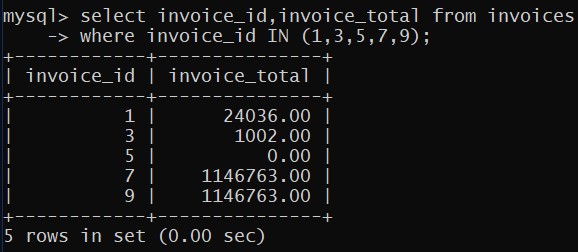
where invoice\_id IN (1,3,5,7,9);

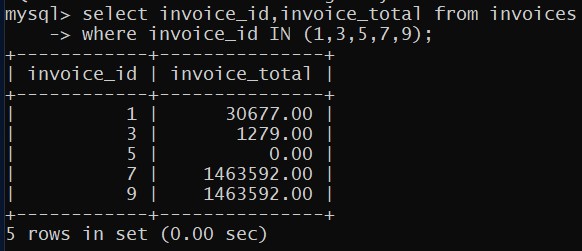
--dropping sample event

drop event if exists updatepayment;

*7eventenable*

**

**

**

## 4. BIBLIOGRAPHY

[1] MySQL scheduled events

<https://www.mysqltutorial.org/mysql-triggers/working-mysql-scheduled-event/>

1. MySQL Reference Manual  
   <https://dev.mysql.com/doc/>
2. Murach’s MySQL 3rd Edition (book)
3. StackOverflow (for finding solutions to errors)

END