CS411 Homework 6

SARAH MENTEL

This problem is on the ap database using MySQL. Create the ap database tables on your MySQL account on the CSIS machine. Note that you don't have a privilege to create a database on your MySQL account on the CSIS machine: you can only create tables.

1. (10) Create a view called num_invoices_view that displays the vendor name and the total number of invoices for each vendor.
   - Show the result of the view with
     `select * from num_invoices_view;`

```
CREATE OR REPLACE VIEW num_invoices_view AS
SELECT vendor_name,
COUNT(*) AS invoice_count
FROM vendors JOIN invoices
ON vendors.vendor_id = invoices.vendor_id
GROUP BY    vendor_name
ORDER BY COUNT(*) DESC;
SELECT *
FROM num_invoices_view;
```

vendor_name	invoice_count
Federal Express Corporation	47
United Parcel Service	9
Zylka Design	8
Pacific Bell	6
Malloy Lithographing Inc	5
Roadway Package System, Inc	4
Blue Cross	3
Ingram	2
Data Reproductions Corp	2

num_invoices_view 4

Output

Action Output

#	Time	Action	Message
1	17:35:42	select * from num_invoices_view LIMIT 0, 1000	34 row(s) returned

2. (10) Create a view called top_ten_unpaid_invoices_view that diplays the vendor name, the total number of invoices for each vendor, and the sum of invoices_total for each vendor. Show top 10 vendors which have the largest sum of invoices_total.
   - Show the result of the view with
     `select * from top_unpaid_invoices_view;`

```
CREATE OR REPLACE VIEW top_ten_unpaid_invoices_view AS

SELECT vendor_name, COUNT(*) AS invoice_count,
SUM(invoices.invoice_total) AS sum_invoice_total

FROM vendors JOIN invoices

ON vendors.vendor_id = invoices.vendor_id

GROUP BY   vendor_name

ORDER BY sum_invoice_total DESC

LIMIT 10;

SELECT * FROM top_ten_unpaid_invoices_view;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

vendor_name	invoice_count	sum_invoice_total
Malloy Lithographing Inc	5	119892.41
United Parcel Service	9	23177.96
Data Reproductions Corp	2	21927.31
Digital Dreamworks	1	7125.34
Zylka Design	8	6940.25
Bertelsmann Industry Svcs. Inc	1	6940.25
Yesmed, Inc	1	4901.26
Federal Express Corporation	47	4378.02
Computerworld	1	2433.00
Cahners Publishing Company	1	2184.50

top_ten_unpaid_invoices_view8

Output

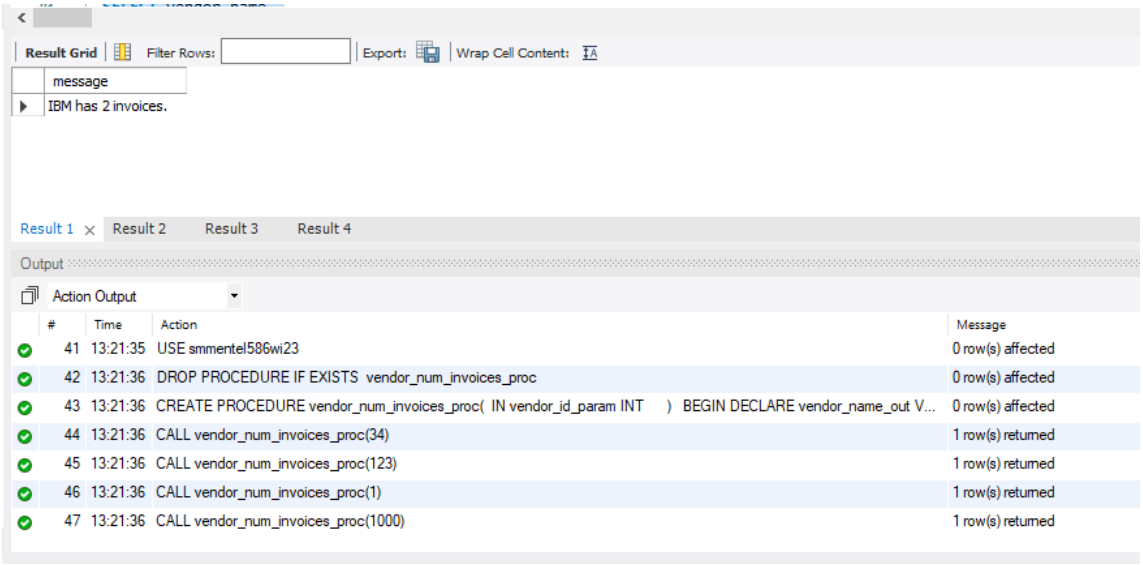
Action Output

#	Time	Action	Message
1	11:46:14	SELECT * FROM top_ten_unpaid_invoices_view LIMIT 0, 1000	10 row(s) returned

3. (10) Create a stored procedure called vendor_num_invoices_proc that shows the name of the vendor and the total number of invoices for each vendor. It should take the vendor id as an input parameter, and it has to print "xx has yy invoices", where xx is the vendor's name and yy is the total number of invoices created for the vendor.
   - Show the result of the procedure with
     - `CALL vendor_num_invoices_proc(34);`
     - `CALL vendor_num_invoices_proc(123);`
     - `CALL vendor_num_invoices_proc(1);`
     - `CALL vendor_num_invoices_proc(1000);`
     - Display a proper message when the vendor name does not exists.

```
-- select the database
USE smmentel586wi23;
DROP PROCEDURE IF EXISTS  vendor_num_invoices_proc;
DELIMITER //
-- Create vendor invoice procedure
CREATE PROCEDURE vendor_num_invoices_proc(
IN vendor_id_param INT
  )
-- Start procedure
BEGIN
DECLARE vendor_name_out VARCHAR(60);
DECLARE invoice_count INT;
-- Get invoice count based on vendor id
SELECT COUNT(vendor_id_param)
INTO invoice_count
FROM invoices
WHERE vendor_id = vendor_id_param;
```

```
-- Get vendor name based upon vendor id
SELECT vendor_name
INTO vendor_name_out
FROM vendors
WHERE vendor_id = vendor_id_param;
-- Ensure vendor exists
 IF  vendor_name_out IS NOT NULL THEN
   SEIECT CONCAT( vendor_name_out,' has ',invoice_count, ' invoices.') AS message;
 ELSE
   SELECT 'Vendor does not exist' AS message;
 END IF;
COMMIT;
END // delimiter ;
-- call procedure
CALL vendor_num_invoices_proc(34);
CALL vendor_num_invoices_proc(123);
CALL vendor_num_invoices_proc(1);
CALL vendor_num_invoices_proc(1000);
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	message
▶	Federal Express Corporation has 47 invoices.

Result 1 | Result 2 × | Result 3 | Result 4

Result Grid | Filter Rows: | Export: | Wrap Cell Cont

	message
▶	US Postal Service has 0 invoices.

Result 1 | Result 2 | Result 3 × | Result 4

Output

Action Output

Result Grid | Filter Rows: | Export: | W

	message
▶	Vendor does not exist

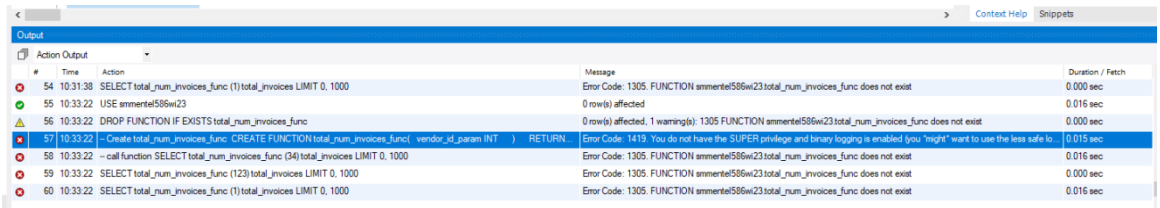
Result 1 | Result 2 | Result 3 | Result 4 ×

Output

4. (10) Create a user defined function called `total_num_invoices_func` that returns the total number of invoices for a vendor.
    - Show the result of the function with

```
SELECT total_num_invoices_func (34) total_invoices;
SELECT total_num_invoices_func (123) total_invoices;
SELECT total_num_invoices_func (1) total_invoices;
```

Unable to create function on mycsis machine



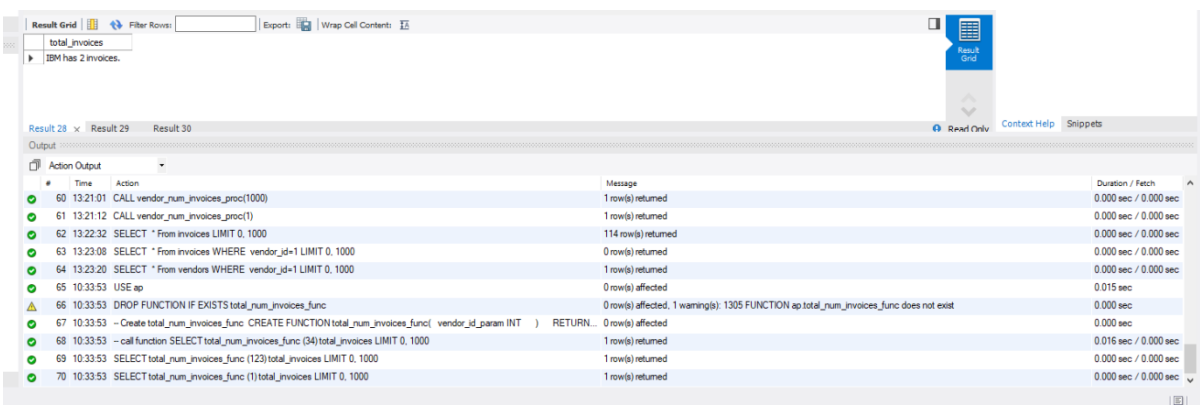
Output

Action Output

#	Time	Action	Message	Duration / Fetch
54	10:31:38	SELECT total_num_invoices_func (1) total_invoices LIMIT 0, 1000	Error Code: 1305. FUNCTION smmentel586wi23.total_num_invoices_func does not exist	0.000 sec
55	10:33:22	USE smmentel586wi23	0 row(s) affected	0.016 sec
56	10:33:22	DROP FUNCTION IF EXISTS total_num_invoices_func	0 row(s) affected, 1 warning(s): 1305 FUNCTION smmentel586wi23.total_num_invoices_func does not exist	0.000 sec
57	10:33:22	-- Create total_num_invoices_func CREATE FUNCTION total_num_invoices_func(vendor_id_param INT) RETURN...	Error Code: 1419. You do not have the SUPER privilege and binary logging is enabled (you "might" want to use the less safe lo...	0.015 sec
58	10:33:22	-- call function SELECT total_num_invoices_func (34) total_invoices LIMIT 0, 1000	Error Code: 1305. FUNCTION smmentel586wi23.total_num_invoices_func does not exist	0.016 sec
59	10:33:22	SELECT total_num_invoices_func (123) total_invoices LIMIT 0, 1000	Error Code: 1305. FUNCTION smmentel586wi23.total_num_invoices_func does not exist	0.000 sec
60	10:33:22	SELECT total_num_invoices_func (1) total_invoices LIMIT 0, 1000	Error Code: 1305. FUNCTION smmentel586wi23.total_num_invoices_func does not exist	0.016 sec

```
-- select the database
USE ap;
DROP FUNCTION IF EXISTS total_num_invoices_func;
DELIMITER //
-- Create total_num_invoices_func
CREATE FUNCTION total_num_invoices_func(
 vendor_id_param INT
    )
    RETURNS VARCHAR(60)
    NOT DETERMINISTIC READS SQL DATA

-- Start function
BEGIN
DECLARE vendor_name_out VARCHAR(60);
DECLARE invoice_count INT;
DECLARE result VARCHAR(60);
-- Get invoice count based on vendor id
SELECT COUNT(vendor_id_param)
INTO invoice_count
FROM invoices
WHERE vendor_id = vendor_id_param;
```

```
-- Get vendor name based upon vendor id
SELECT vendor_name
INTO vendor_name_out
FROM vendors
WHERE vendor_id = vendor_id_param;
-- Ensure vendor exists
  IF  vendor_name_out IS NOT NULL THEN
   SELECT CONCAT( vendor_name_out,' has ',invoice_count, '
invoices.') AS message
   INTO result;
  ELSE
    SELECT 'Vendor does not exist' AS message
     INTO result;
  END IF;

RETURN(result);
END // delimiter ;
-- call function
SELECT total_num_invoices_func (34) total_invoices;

SELECT total_num_invoices_func (123) total_invoices;

SELECT total_num_invoices_func (1) total_invoices;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

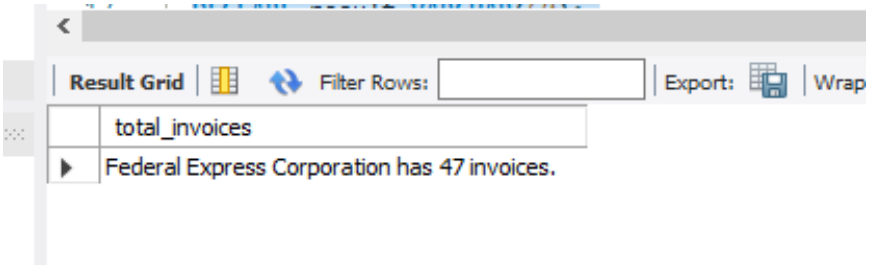
total_invoices
IBM has 2 invoices.

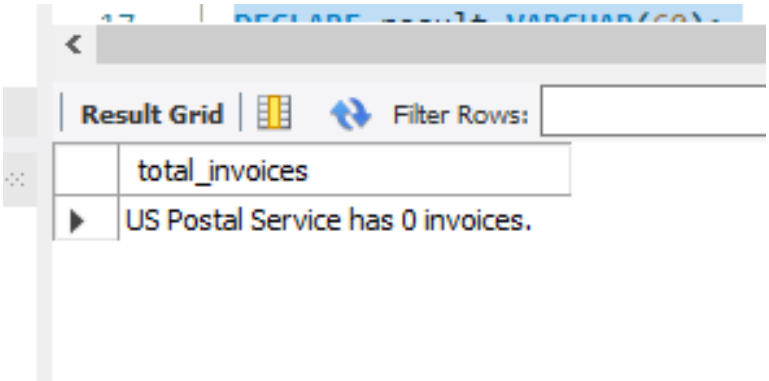
Result 28 × Result 29 Result 30

Output

Action Output

#	Time	Action	Message	Duration / Fetch
60	13:21:01	CALL vendor_num_invoices_proc(1000)	1 row(s) returned	0.000 sec / 0.000 sec
61	13:21:12	CALL vendor_num_invoices_proc(1)	1 row(s) returned	0.000 sec / 0.000 sec
62	13:22:32	SELECT * From invoices LIMIT 0, 1000	114 row(s) returned	0.000 sec / 0.000 sec
63	13:23:08	SELECT * From invoices WHERE vendor_id=1 LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
64	13:23:20	SELECT * From vendors WHERE vendor_id=1 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
65	10:33:53	USE ap	0 row(s) affected	0.015 sec
66	10:33:53	DROP FUNCTION IF EXISTS total_num_invoices_func	0 row(s) affected, 1 warning(s): 1305 FUNCTION ap.total_num_invoices_func does not exist	0.000 sec
67	10:33:53	-- Create total_num_invoices_func CREATE FUNCTION total_num_invoices_func(vendor_id_param INT) RETURN...	0 row(s) affected	0.000 sec
68	10:33:53	-- call function SELECT total_num_invoices_func (34) total_invoices LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
69	10:33:53	SELECT total_num_invoices_func (123) total_invoices LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
70	10:33:53	SELECT total_num_invoices_func (1) total_invoices LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec





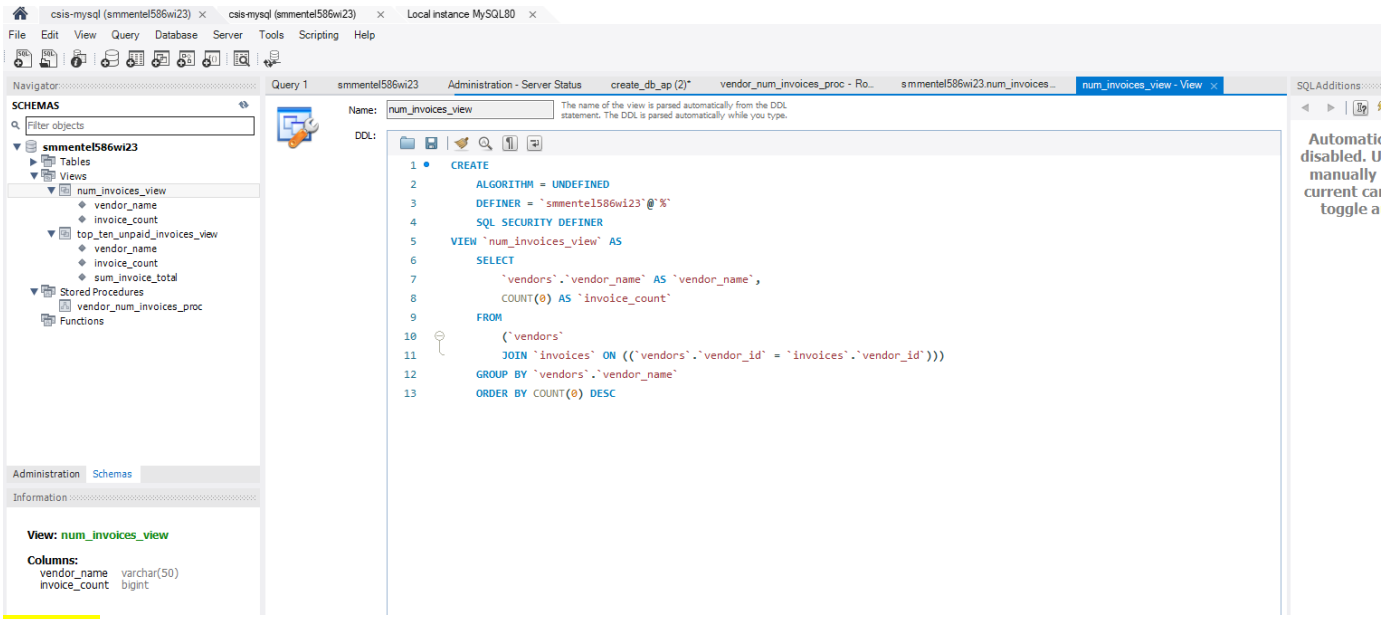
5. (10) Create a trigger named first_time_invoice_trigger that shows "An invoice is created for the first time from xx" if an invoice is created from a vendor named xx for the first time.
   - Show the result of the trigger with
     `INSERT INTO invoices VALUES (200, 1, ...);`
   - Note that 200 is for the invoice number and 1 is for the venor id. Make sure that no invoice is created from vendor_id 1 yet.
   - You need to fill out the … part
   - There is no print statement in MySQL. You can create a new table for the message to be displayed.

```
drop trigger if exists first_time_invoice_trigger;


delimiter //


CREATE TRIGGER first_time_invoice_trigger

AFTER INSERT ON invoices

FOR EACH ROW
```

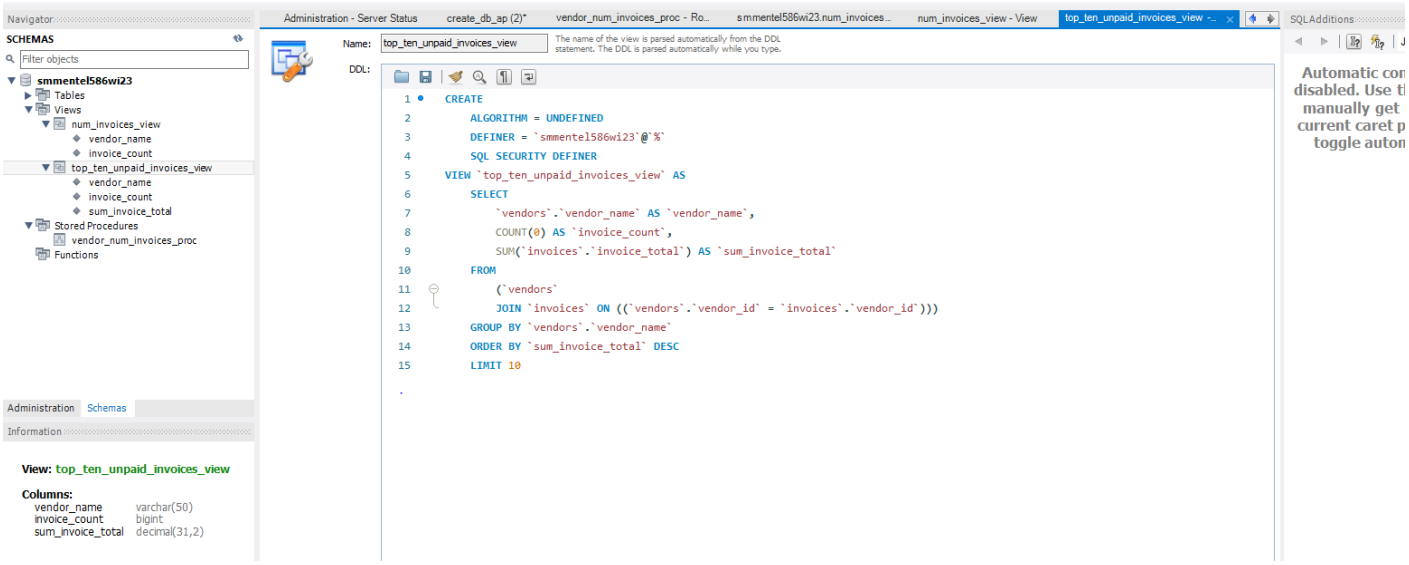
```
BEGIN

DECLARE vendor_name_out VARCHAR(50);

DECLARE vendor_invoice_count INT;


-- Get vendor name based upon vendor id

SELECT vendor_name

INTO vendor_name_out

FROM vendors

WHERE vendor_id = NEW.vendor_id;


-- Get count of invoices

SELECT COUNT(vendor_id) INTO vendor_invoice_count

FROM invoices;


   IF vendor_invoice_count = 1 THEN

insert into invoice_first_time values (vendor_name_out, 'has created an invoice for the first time',
now());

END IF;

END //


select * from invoices where invoice_id = 200;

INSERT INTO invoices   VALUES (200,1,'989319-457','2018-04-08','3813.33','3813.33','0.00',3,'2018-05-
08','2018-05-07');

select * from invoice_first_time;
```

6. (10) Show a screenshot that displays all the stored procedures, functions, and triggers, and their codes.

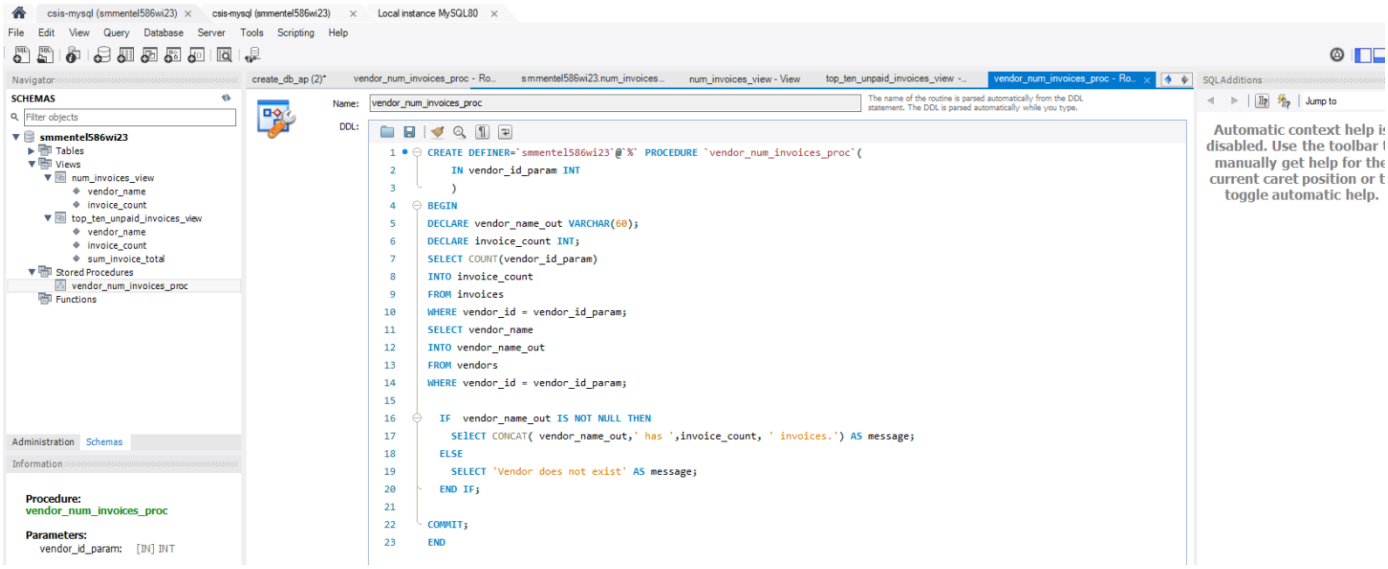
1st view



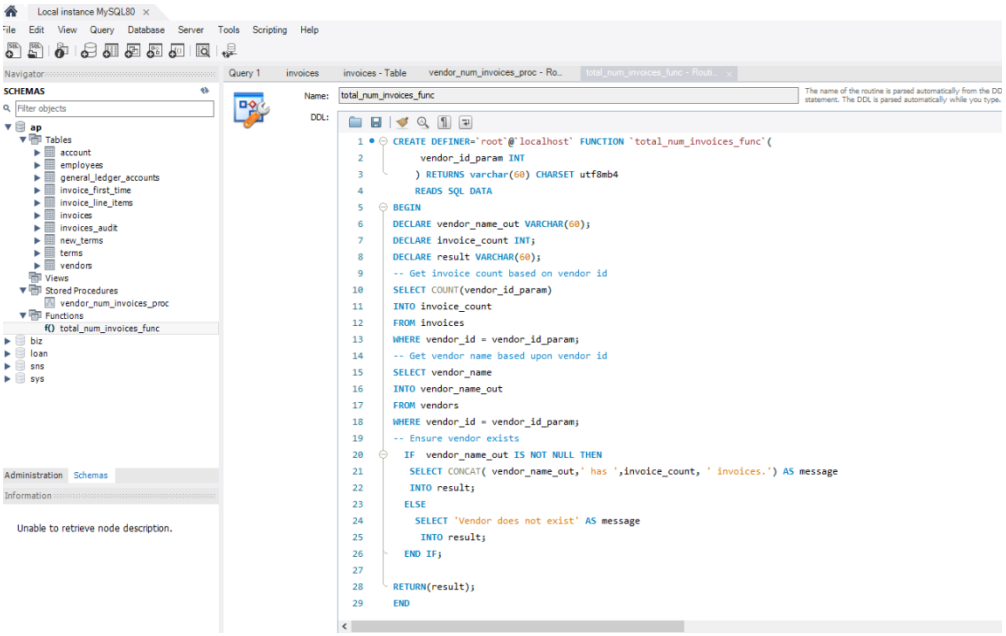
2nd view



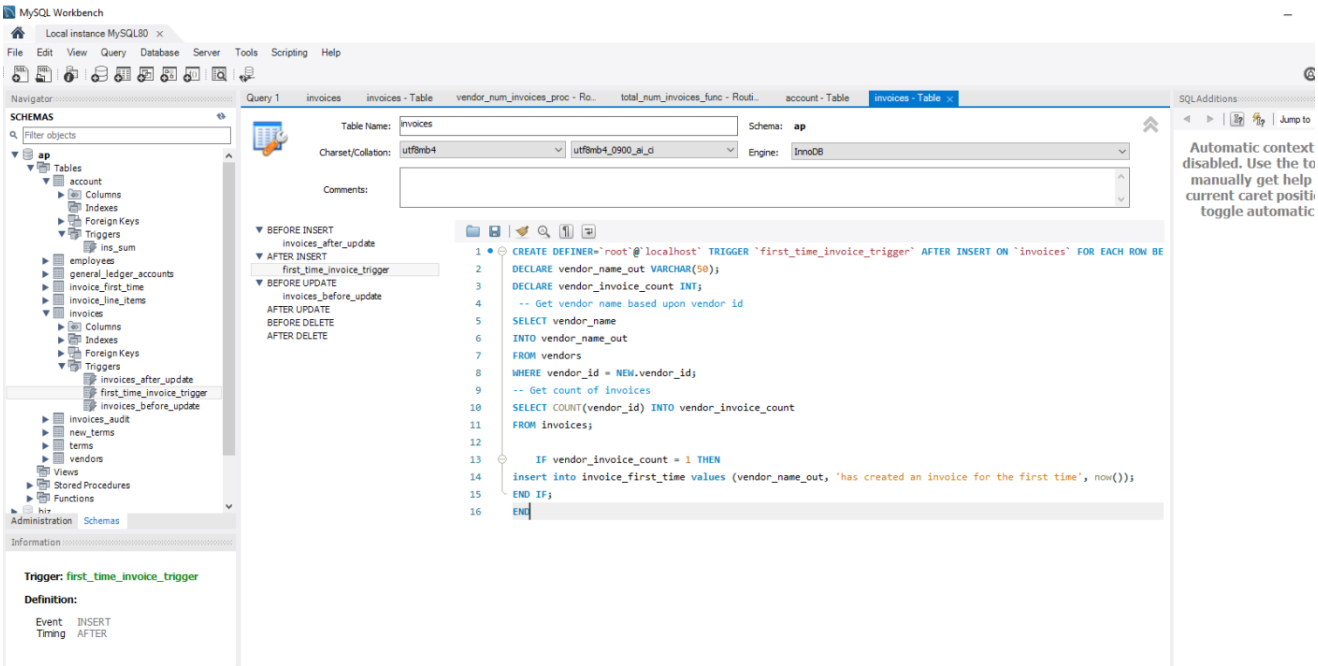
Stored Procedure



Function



Trigger


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
ap
Tables
account
Columns
Indexes
Foreign Keys
Triggers
ins_sum
employees
general_ledger_accounts
invoice_first_time
invoice_line_items
invoices
Columns
Indexes
Foreign Keys
Triggers
invoices_after_update
first_time_invoice_trigger
invoices_before_update
invoices_audit
new_terms
terms
vendors
Views
Stored Procedures
Functions
Administration Schemas
Information
Trigger: first_time_invoice_trigger
Definition:
Event INSERT
Timing AFTER
Query 1 invoices invoices - Table vendor_num_invoices_proc - Ro.. total_num_invoices_func - Routi.. account - Table invoices - Table
Table Name: invoices
Schema: ap
Charset/Collation: utf8mb4 utf8mb4_0900_ai_ci
Engine: InnoDB
Comments:
BEFORE INSERT
invoices_after_update
AFTER INSERT
first_time_invoice_trigger
BEFORE UPDATE
invoices_before_update
AFTER UPDATE
BEFORE DELETE
AFTER DELETE
CREATE DEFINER=`root`@`localhost` TRIGGER `first_time_invoice_trigger` AFTER INSERT ON `invoices` FOR EACH ROW BE
DECLARE vendor_name_out VARCHAR(50);
DECLARE vendor_invoice_count INT;
-- Get vendor name based upon vendor id
SELECT vendor_name
INTO vendor_name_out
FROM vendors
WHERE vendor_id = NEW.vendor_id;
-- Get count of invoices
SELECT COUNT(vendor_id) INTO vendor_invoice_count
FROM invoices;
IF vendor_invoice_count = 1 THEN
insert into invoice_first_time values (vendor_name_out, 'has created an invoice for the first time', now());
END IF;
END
SQLAdditions
Jump to
Automatic context disabled. Use the to manually get help current caret positi toggle automatic

7. (20) Create the following tables.

employee

id	name	superid	salary	bdate	dno
1	john	3	100000	1960-01-01	1
2	mary	3	50000	1964-12-01	3
3	bob	NULL	80000	1974-02-07	3
4	tom	1	50000	1978-01-17	2
5	bill	NULL	NULL	1985-01-20	1

department

dnumber	dname
1	Payroll
2	TechSupport
3	Research

deptsal

dnumber	totalsalary
1	0
2	0
3	0

- Make sure that id is the primary key of the employee table and dnumber is the primary key for the department and deptsal tables.
- Make sure that dno is the foreign key to department and deptsal.
- Make sure that when a department is deleted from the department table, the employees working for the deparment are also deleted from the employee table.

a. (5) Show the SQL scripts that creates the above tables with primary keys and foreign key constraints.

```
CREATE TABLE employee

(

id INT PRIMARY KEY NOT NULL UNIQUE,

ename VARCHAR(20),

superid INT,

salary INT,

bdate DATE NOT NULL,

dno INT NOT NULL

);


CREATE TABLE department

(

dnumber INT PRIMARY KEY NOT NULL UNIQUE,

dname VARCHAR(20)  NOT NULL

);
```

```
CREATE TABLE deptsal
(
dnumber INT PRIMARY KEY NOT NULL UNIQUE,
totalsalary INT NOT NULL
);

INSERT INTO employee
VALUES (1,'john',3,'100000','1960-01-01',1),
(2,'mary',3,'50000','1964-12-01',3),
(3,'bob',NULL,'80000','1974-02-07',3),
(4,'tom',1,'50000','1978-01-17',2),
(5,'bill',NULL,NULL, '1985-01-20',1);

INSERT INTO department
VALUES(1,'payroll'),
(2,'TechSupport'),
(3,'Research');

INSERT INTO deptsal
VALUES(1,0),
(2,0),
(3,0);
```

```
ALTER TABLE employee
ADD FOREIGN KEY(dno)
REFERENCES department(dnumber)
ON  DELETE CASCADE;

ALTER TABLE department
ADD FOREIGN KEY(dnumber)
REFERENCES employee(dno) ;

ALTER TABLE deptsal
ADD FOREIGN KEY(dnumber)
REFERENCES employee(dno) ;
```

b. (5) Add one more depatment called ‘HumanResource’ and add two more employees including your name and your CS116 instructor who work for HumanResource. Make john as a supervisor for both, and enter arbitary value for the salary.

```
SET FOREIGN_KEY_CHECKS=0;
INSERT INTO employee
VALUES(6,'sarah',1,60000,'2001-05-15',4),
(7,'aos',1,60000, '1980-03-03',4);
INSERT INTO department
VALUES(4,'HumanResource');
```

c. (5) Write a stored procedure/function or trigger that updates the deptsal table when a new employee is added, an employee is deleted, or when a department is deleted.

Unable to create function on mycsis machine

13	17:14:54	CALL find_managed ('bob')	Error Code: 1172. Result consisted of more than one row	0.000 sec
14	17:20:42	CREATE TRIGGER invoices_after_insert AFTER INSERT ON invoices FOR EACH ROW BEGIN INSERT INTO ...	Error Code: 1419. You do not have the SUPER privilege and binary logging is enabled (you "might" want to use the les...	0.000 sec
15	17:20:42	CREATE TRIGGER invoices_after_delete AFTER DELETE ON invoices FOR EACH ROW BEGIN INSERT INT...	Error Code: 1419. You do not have the SUPER privilege and binary logging is enabled (you "might" want to use the les...	0.016 sec

Output

Action Output

#	Time	Action	Message
1	17:27:26	USE biz	0 row(s) affected
2	17:27:26	-- trigger for insert CREATE TRIGGER employee_after_insert AFTER INSERT ON employee FOR EACH ROW BEGIN...	Error Code: 1359. Trigger already exists
3	17:27:26	-- trigger for delete CREATE TRIGGER employee_after_delete AFTER DELETE ON employee FOR EACH ROW BEG...	Error Code: 1359. Trigger already exists
4	17:27:26	-- trigger for delete department CREATE TRIGGER employee_after_depdel AFTER DELETE ON department FOR EA...	Error Code: 1359. Trigger already exists
5	17:27:26	DELETE FROM department WHERE dnumber = 3; select * FROM employee_audit;	0 row(s) affected
6	17:27:26	DELETE FROM department WHERE dnumber = 3; select * FROM employee_audit;	Error Code: 1146 Table 'biz.employee_audit' doesn't exist

```
-- select the database
USE biz;
DELIMITER //
-- trigger for insert
CREATE TRIGGER employee_after_insert
  AFTER INSERT ON employee
  FOR EACH ROW
BEGIN
    INSERT INTO employee_audit VALUES
    (NEW.ename, 'INSERTED', NOW());
END//
DELIMITER //
-- trigger for delete
CREATE TRIGGER employee_after_delete
  AFTER DELETE ON employee
  FOR EACH ROW
BEGIN
    INSERT INTO employee_audit VALUES
```

```
  (OLD.ename, 'DELETED', NOW());
END//
DELIMITER //
-- trigger for delete department
CREATE TRIGGER employee_after_depdel
 AFTER DELETE ON department
 FOR EACH ROW
BEGIN
  INSERT INTO employee_audit VALUES
  (ename, 'DELETED', NOW());
END//
DELETE FROM department WHERE dnumber = 3;
select  * FROM employee_audit;
```

d. (5) Execute the following query: DELETE FROM department WHERE dnumber = 3; Then, show the result of the following queries.
SELECT * FROM department;
SELECT * FROM employee;
SELECT * FROM deptsal;

Result Grid | Filter Rows:

dnumber	dname
1	payroll
2	TechSupport
4	HumanResource
NULL	NULL

Result Grid | Filter Rows: | Edit:

id	ename	superid	salary	bdate	dno
1	john	3	100000	1960-01-01	1
2	mary	3	50000	1964-12-01	3
3	bob	NULL	80000	1974-02-07	3
4	tom	1	50000	1978-01-17	2
5	bill	NULL	NULL	1985-01-20	1

department 17 | employee 18 | deptsal 19

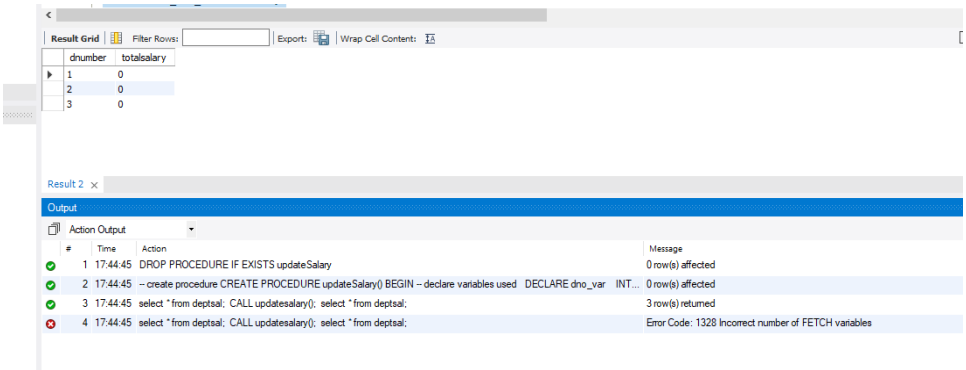
Result Grid | Filter Rows:

dnumber	totalsalary
1	0
2	0
3	0
NULL	NULL

8. (10) Write a stored procedure called update_salary_proc() that updates the deptsal table with total salary of the employees from each department. You want to use Cursor to achieve the task.
   - Show the result for the following statements:

```
select * from deptsal;
CALL update_salary();
select * from deptsal;

-- drop if exists
DROP PROCEDURE IF EXISTS updateSalary;
DELIMITER //
-- create procedure
CREATE PROCEDURE updateSalary()
BEGIN
-- declare variables used
  DECLARE dno_var    INT;
  DECLARE salary_var DECIMAL(9,2);
  DECLARE row_not_found     TINYINT DEFAULT FALSE;
  DECLARE update_count      INT DEFAULT 0;
  -- cursor
DECLARE employee_cursor CURSOR FOR
    SELECT salary FROM employee
    WHERE dno = dno_var;

DECLARE CONTINUE HANDLER FOR NOT FOUND
    SET row_not_found = TRUE;

OPEN employee_cursor;

-- loop through each employees salaray
  WHILE row_not_found = FALSE DO
    FETCH employee_cursor INTO dno_var, salary_var;

-- update deptsal
      UPDATE deptsal
      SET total_salary = total_salary + salary_var
      WHERE dno = dno_var;
      SET update_count = update_count + 1;
```

```
  END WHILE;

  CLOSE employee_cursor;

  SELECT CONCAT(update_count, ' row(s) updated.');
END//
select * from deptsal;
CALL updatesalary();
select * from deptsal;
```



9. (10) Decide upon using stored procedure, stored function, or view to find out the names of employees managed by a given employee. Name it find_managed.

```
DROP PROCEDURE IF EXISTS find_managed;


DELIMITER //

-- Create find_managed

CREATE PROCEDURE  find_managed(

 IN employee_name_param VARCHAR(60)

   )

-- Start

BEGIN

DECLARE employee_name_out VARCHAR(60);

DECLARE manager_id INT;

DECLARE result VARCHAR(60);

-- Get managers id

SELECT id

INTO manager_id
```

```
FROM employee

WHERE ename = employee_name_param;


-- Get employees based on manager id

SELECT ename

INTO result

FROM employee

WHERE superid = manager_id;

  SELECT result;

COMMIT ;

END // delimiter ;


-- call function

CALL find_managed ('john');

CALL find_managed ('bob');
```

- Find out employees managed by john. Show your answer and the result.

- Find out employees managed by bob. Show your answer and the result

62	17:02:10	-- Create find_managed CREATE PROCEDURE find_managed(IN employee_name_param VARCHAR(60) ...	0 row(s) affected
63	17:02:10	-- call function CALL find_managed ('john')	Error Code: 1172. Result consisted of more than one row
64	17:02:10	CALL find_managed ('bob')	Error Code: 1172. Result consisted of more than one row